UNIVERSIDAD NACIONAL DEL ALTIPLANO

ESCUELA DE POSGRADO

DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN

TESIS

PREDICCIÓN DE CÁNCER EN EXPRESIONES GENÉTICAS DE MICROARRAYS MEDIANTE UN ENSAMBLE DE MODELOS HETEROGÉNEOS DE MACHINE LEARNING

PRESENTADA POR:

LUIS BELTRAN PALMA TTITO

PARA OPTAR EL GRADO ACADÉMICO DE:

DOCTOR EN CIENCIAS DE LA COMPUTACIÓN

PUNO, PERÚ

2023

Reporte de similitud

NOMBRE DEL TRABAJO

PREDICCIÓN DE CÁNCER EN EXPRESION ES GENÉTICAS DE MICROARRAYS MEDI ANTE UN ENSAMBLE DE MODELOS HET EROG

AUTOR

LUIS BELTRAN PALMA TTITO

UNA PUNO

Firmado digitalmente por CANQUI FLORES Bernabe FAU 20145496170 hard
Motivo: Soy el autor del documento
Fecha: 24.01.2024 22:08:40 -05:00

RECUENTO DE PALABRAS

22265 Words

RECUENTO DE CARACTERES

117711 Characters

RECUENTO DE PÁGINAS

132 Pages

TAMAÑO DEL ARCHIVO

6.7MB

FECHA DE ENTREGA

Jan 24, 2024 10:05 PM GMT-5

FECHA DEL INFORME

Jan 24, 2024 10:06 PM GMT-5

8% de similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para cada base c

- 8% Base de datos de Internet
- 1% Base de datos de publicaciones
- Base de datos de Crossref
- Base de datos de contenido publicado de Crossr
- 5% Base de datos de trabajos entregados

Excluir del Reporte de Similitud

- Material bibliográfico
- Material citado
- Material citado
- Coincidencia baja (menos de 12 palabras)

VB CIEPG

UNA PUNO

Firmado digitalmente por LUQUE COYLA Ruben Jared FAU 20145496170 hard
Motivo: Doy V° B°
Fecha: 25.01.2024 08:50:03 -05:00

Resumen

UNIVERSIDAD NACIONAL DEL ALTIPLANO

ESCUELA DE POSGRADO

DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN

TESIS

PREDICCIÓN DE CÁNCER EN EXPRESIONES GENÉTICAS DE MICROARRAYS MEDIANTE UN ENSAMBLE DE MODELOS HETEROGÉNEOS DE MACHINE LEARNING

PRESENTADA POR:

LUIS BELTRAN PALMA TTITO

PARA OPTAR EL GRADO ACADÉMICO DE:

DOCTOR EN CIENCIAS DE LA COMPUTACIÓN

APROBADA POR EL JURADO SIGUIENTE:

PRESIDENTE

.....
Dr. LEONEL COYLA IDME

PRIMER MIEMBRO

.....
D.Sc. JUAN CARLOS JUAREZ VARGAS

SEGUNDO MIEMBRO

.....
D.Sc. ADOLFO CARLOS JIMÉNEZ CHURA

ASESOR DE TESIS

.....
Dr. BERNABÉ CANQUI FLORES

Puno, 22 de mayo de 2023

ÁREA : Ciencias de la Computación
TEMA: Predicción de cáncer en expresiones genéticas de microarray mediante un ensamble de modelos heterogéneos de machine learning
LÍNEA: Desarrollo de aplicaciones

DEDICATORIA

A mi esposa Roxana, a mis hijas, Mayra y Laura, quienes me motivaron para la conclusión de la presente investigación, y dieron su apoyo incondicional.

AGRADECIMIENTOS

A mi Asesor, el Dr. Bernabé Canqui Flores, por su apoyo en la orientación y elaboración de la investigación.

A mis profesores del Doctorado de Ciencias de la Computación, de la Universidad del Altiplano - Puno, quienes impartieron sus conocimientos, y motivaron, en la elaboración de la presente.

Al National Center for Biotechnology Information del gobierno de Estados Unidos, y al European Bioninformatics Institute, instituciones que fueron claves para la obtención de datos de cáncer de mama, utilizados en los experimentos de la presente investigación.

ÍNDICE GENERAL

Pág.

DEDICATORIA i
AGRADECIMIENTOS ii
ÍNDICE GENERAL iii
ÍNDICE DE TABLAS vi
ÍNDICE DE FIGURAS vii
ÍNDICE DE ANEXOS x
RESUMEN xi
ABSTRACT xii
INTRODUCCIÓN 1

CAPÍTULO I

REVISIÓN DE LITERATURA

1.1. Marco teórico 4
1.1.1. ADN 4
1.1.2. ARN 6
1.1.3. Proteína 6
1.1.4. Cromosoma 7
1.1.5. Gen 7
1.1.6. Dogma central de biología molecular 8
1.1.7. Código genético 9
1.1.8. ADN complementario 10
1.1.9. Hibridación 11
1.1.10. Mutación 12
1.1.11. Cáncer 14
1.1.12. Cáncer de seno 14
1.1.13. Microarray de oligonucleótido 15
1.1.14. Aprendizaje Automático 20
1.1.15. Algoritmos de clasificación 21
1.1.16. K vecinos más cercanos (KNN) 23
1.1.17. Máquina de vector de soporte para clasificación (SVC) 23
1.1.18. Árbol de decisión (AD) 25
1.1.19. Clasificador Naïve Bayes (NB) 26
1.1.20. Red neuronal perceptrón multicapa (MLP) 28

1.1.21. Ensambles 30
1.1.22. Test de ANOVA 33
1.1.23. Test de Tukey 34
1.1.24. Test de Welch ANOVA y test Games Showell 35
1.2. Antecedentes 35

CAPÍTULO II

PLANTEAMIENTO DEL PROBLEMA

2.1. Identificación del problema 40
2.2. Enunciados del problema 40
2.3. Justificación 43
2.4. Objetivos 43
2.4.1. Objetivo general 43
2.4.2. Objetivos específicos 44
2.5. Hipótesis 44
2.5.1. Hipótesis general 44
2.5.2. Hipótesis específicas 44

CAPÍTULO III

MATERIALES Y MÉTODOS

3.1. Lugar de estudio 45
3.2. Población 45
3.3. Muestra 45
3.4. Método de investigación 45
3.5. Descripción detallada de métodos por objetivos específicos 46

CAPÍTULO IV

RESULTADOS Y DISCUSIÓN

4.1. Objetivo específico 1 49
Primer experimento (*E-GEOD73002*) 49
Segundo Experimentos (*E-GEOD-42568*) 50
4.2. Objetivo específico 2 52
Primer Experimentos (E-GEOD-73002) 52
Segundo Experimentos (*E-GEOD-42568*) 56
4.3. Objetivo específico 3 59
Primer experimento (E-GEOD-73002) 59
Segundo experimento (E-GEOD-42568) 64

CONCLUSIONES 68
RECOMENDACIONES 69
BIBLIOGRAFÍA 70
ANEXOS 74

ÍNDICE DE TABLAS

Pág.

1. Código genético, que transcribe un codón (tres nucleótidos) en aminoácidos 10
2. Valor de hiper parámetros que optimizan los algoritmos de aprendizaje automático 52
3. Resumen del test de Games Showell, donde se identifica las diferencias significativas 55
4. Valor de hiper parámetros que optimizan los algoritmos de aprendizaje automático 56
5. Resumen del test de Games Showell, donde se identifica las diferencias significativas 58
6. Valor de hiper parámetros que optimizan los algoritmos utilizando en los ensambles 61
7. Exactitud de predicción de cáncer de mama, con el 25% de datos de test 61
8. Resumen del test Games Showel 64
9. Valores de configuración de hiper parámetros de los algoritmos utilizados en modelos ensambles 65
10. Exactitud obtenida en la fase de test de modelos ensamble, con el 75% de datos 65
11. Hiper parámetros experimentados en la búsqueda de la mejor configuración 85
12. Valor de hiper parámetros óptimos encontrados mediante búsqueda aleatoria en cuadricula 86
13. Resumen de la tabla Games Showell 91

ÍNDICE DE FIGURAS

Pág.

1. Subunidades del ADN, que proporciona un mecanismo para la herencia 4
2. ADN de doble cadena y ADN de doble hélice 5
3. Bloque de construcción de ADN: azúcar-fosfato + base = nucleótido 5
4. Cadena de ADN 5
5. Cromosomas humano 7
6. Dogma central de la biología molecular (El ADN se transcribe para producir ARNm, el que se traduce para generar proteínas) 8
7. Dogma completo de la biología molecular 9
8. Desnaturalización y Renaturalización de ADN 11
9. Hibridación de ADN-ARN 11
10. La anemia de la célula falciforme es causada por una proteína alterada 12
11. Glóbulo rojo normal y falciforme 12
12. Operaciones básica de mutaciones cromosómicas. 13
13. Estadios del cáncer de mama, de izquierda a derecha, 1er, 2do, 3er y 4to estadio respectivamente. 15
14. Microarray de oligonucleótidos 16
15. Chip de sondas 17
16. Hibridación 17
17. Análisis de imagen 18
18. Matriz de expresión génica. 19
19. Resolución de matrices de expresión génica 19
20. Análisis de microarray de oligonucleótidos y de ADNc 20
21. Tareas de Aprendizaje Automático 21
22. Hiperplano identificado por SVM 24
23. Kernel de SVM 24
24. Árbol de decisión 25
25. Red del clasificador Naïve Bayes 27
26. Neurona bilógica 28
27. Neurona artificial 28
28. Perceptrón multicapa 29
29. Ensamble Voting 30
30. Ensamble Bagging 31

31. Ensamble Boosting 32
32. Ensamble Stacking 33
33. Expresiones de ARN (E-GEOD-73002) 49
34. Distribución de clases, a la izquierda datos originales, a la derecha datos sin las clases minoritarias 50
35. Expresiones génicas de ADN (E-GEOD-24568) 50
36. Distribución de clases de datos de expresiones génicas (E-GEOD-24568) 51
37. Gráfico en barras, que representa la puntuación asignada a cada sonda como variable predictora, habiendo utilizado el método Ganancia de Información y ANOVA 51
38. Valores de exactitud de predicción de cada algoritmo aplicando validación cruzada, k=10 52
39. Gráficos cuartil-cuartil, que muestras si los datos de predicción de la figura 38 siguen una distribución normal 53
40. Estadísticos del test de Welch ANOVA 53
41. Resultados de aplicar el test de Games Showell, para determinar ¿Cuál es el menor clasificador? 54
42. Valores de exactitud de predicción de cada algoritmo aplicando validación cruzada, k=10 56
43. Gráficos cuartil-cuartil, que muestras si los datos de predicción de la figura 37 siguen una distribución normal 57
44. Estadísticos del test de Welch ANOVA 57
45. Resultados de aplicar el test de Games Showell, para determinar ¿Cuál es el menor clasificador? 58
46. Voting heterogéneo planteado 59
47. Bagging heterogéneo planteado 59
48. Boosting heterogéneo planteado 60
49. Stacking heterogéneo planteado 60
50. Valores de exactitud de diagnósticos de cáncer de mama, aplicando validación cruzada, con k=10 62
51. Grafico cuartil-cuartil, que permite detectar conjunto valores de exactitud de predicción, siguen una distribución normal 62
52. Estadístico obtenidos con el test de Welch ANOVA 62
53. Estadísticos obtenidos con el test de Games Showel 63

54. Valores de exactitud de predicción de cáncer de mama, encontrados aplicando validación cruzada k=10, valores a utilizar en la elección del mejor predictor. 66
55. Gráfico cuartil-cuartil, que muestra la normalidad de datos de predicción con validación cruzada 66
56. Estadísticos obtenidos por el test Welch ANOVA. 66
57. Explorador de dataset de microarray 76
58. Detalles del dataset de cáncer de seno metastásico y sepsis. 77
59. Distribución de clases del dataset GSE45827 80
60. Expresiones génicas de muestra de cáncer de seno, la imagen visualiza únicamente 5 primeras muestras de un total de 151, y 11 genes de un total de 54675. 80
61. Célula eucariota, que posee el humano 81
62. Cromosomas de homo sapiens 81
63. Cromosoma 6 a detalle, las zonas oscuras, corresponden a los genes. 82
64. Región 1007_s_at 82
65. sub región 1007_s_at_1 82
66. Secuencia completa de la región 1007_s_at 82
67. Selección de genes importantes en la predicción 83
68. Distribución de clases después de la reducción de la dimensionalidad 83
69. Matrices de confusión de los cinco algoritmos de clasificación 87
70. Diagrama de bigotes de exactitud de predicción obtenida con validación cruzada (k=10). 88
71. Gráfica cuartil-cuartil de los resultados de exactitud obtenidas con validación cruzada (k=10) 88
72. Exactitud de algoritmos de clasificación obtenido por validación cruzada con k=10 89
73. Test de Welch ANOVA 89
74. Resultado del test de Game Showell 90

ÍNDICE DE ANEXOS

Pág.

1. Consideraciones en el análisis y exploración de microarray de cáncer de mama 74
2. Proceso integral de obtención de microarray y aplicación de modelos de aprendizaje automático 76
3. Código fuente de la aplicación para determinación del mejor algoritmo de predicción de cáncer de mama 92
4. Código fuente de la aplicación para determinacióndel mejor ensamble de predicción de cáncer de mama 105

RESUMEN

El cáncer de seno, es una de las enfermedades, que aproximadamente genera 2.26 millones de muertes a nivel mundial anualmente, según la Organización Mundial de la Salud. El diagnóstico de la enfermedad, en etapas iniciales es importante, para permitir un tratamiento que elimine y/o alivie las consecuencias del mismo. Proveer de diversas técnicas para la detección del cáncer de seno, dará mayores opciones a los pacientes para el diagnóstico, y permitirá la disminución de costos. Por ello, es necesario conocer, ¿qué ensambles heterogéneos de aprendizaje automático, tiene mejor predicción de cáncer de seno, a partir de datos de expresiones géneticas de microarray?. En la presente investigación, se diseñó e implemento, cuatro ensambles de algoritmos heterogéneos: voting, bagging, boosting y stacking, los cuales fueron entrenados con un dataset de 4113 muestras miARN, cada uno con 2542 atributos, luego se aplicó los test Welch ANOVA y test de Games Showel, con diez resultados de exactitud, obtenidos por validación cruzada, y se detectó, que los ensambles no presentan diferencias significativas, logrando alcanzar una exactitud de predicción promedio de 98.23%. También se aplicó la misma metodología a, 121 muestras ADN extraídas por biopsia de células de mama, que constan de 54676 atributos, obteniendo una exactitud de predicción promedio de 99.99%.

Palabras clave: Aprendizaje automático, cáncer de seno, ensambles, expresión génica, microarray, Welch ANOVA.

ABSTRACT

Breast cancer is one of the diseases that generates approximately 2.26 million deaths worldwide annually, according to the World Health Organization. The diagnosis of the disease, in its initial stages, is important, to allow a treatment that eliminates and/or alleviates its consequences. Providing various techniques for the detection of breast cancer will give patients greater options for diagnosis, and will allow cost reduction. Therefore, it is necessary to know which heterogeneous machine learning ensembles have the best prediction of breast cancer, based on microarray genetic expression data? In the present investigation, four sets of heterogeneous algorithms were designed and implemented: voting, bagging, boosting and stacking, which were trained with a dataset of 4113 miRNA samples, each with 2542 attributes, then the Welch ANOVA and Games Showel test, with ten accuracy results, obtained by cross validation, and it was detected that the assemblies do not present significant differences, achieving an average prediction accuracy of 98.23%. The same methodology was also applied to 121 DNA samples extracted by breast cell biopsy, consisting of 54,676 attributes, obtaining an average prediction accuracy of 99.99%.

Keywords: Machine learning, breast cancer, ensembles, gene expression, microarray, Welch ANOVA.

INTRODUCCIÓN

El cáncer es la segunda causa de muerte en el mundo, cada año aproximadamente mueren cerca de diez millones de habitantes, se estima que, de cada seis muertes, una se debe al cáncer. También manifiestan, que, aproximadamente el 70% de muertes por cáncer ocurre en países de bajos o medianos ingresos. Y afirman, que, el cáncer más común a nivel mundial, es el cáncer de mama, cerca de 2.26 millones de casos por año. Estas afirmaciones, evidencian la importancia, de crear nuevos mecanismos de diagnóstico de la enfermedad para que los personas tengan mayor accesibilidad y a menores costos (Organización Mundial de la Salud, 2021)

El Instituto Nacional de Enfermedades Neoplásica de Perú, manifiesta que muchas personas pueden tener una vida plena, si se detecta la enfermedad en las fases iniciales, y si han recibido un tratamiento contra el cáncer, por ello, la importancia de la detección temprana de cáncer, para disminuir las muertes por esta enfermedad (Instituto Nacional de Enfermedades Neoplásicas, 2023)

En Knife (2023), manifiestan, que, en enfermedades de alto riego, como el cáncer, los pacientes generalmente solicitan una segunda opinión, con el propósito, de tener la certeza del diagnostico primigenio, este hecho es corroborado por especialistas, quienes, además agregan, que está práctica tiene un gran potencial. Este mismo hecho es manifestado por (American Cancer Society 2023). Según Mira et al. (2006), consideran que la segunda opinión es un derecho de toda persona, según la investigación, los expertos mencionan que una segunda opinión aumenta la calidad de asistencia sanitaria, la confianza y credibilidad. Nacional Instituto del Cáncer (2023) afirma que, en el campo de la medicina, una segunda opinión ofrece al paciente una opinión que confirma o cuestiona el diagnostico original, y el plan de tratamiento. Según Sanchez *et al*. (2021), la segunda opinión en las encuestas realizada con diferentes estratos sociales de pacientes en Francia, son solicitadas en caso de enfermedades complejas y en casos de riesgo, La investigación, no encontró que esta decisión dependiera de sexo, edad, región de procedencia o profesión, y una segunda opinión mejora la calidad de atención.

En Weigl *et al*. (2021), manifiestan que, en Alemania, los pacientes tienen derecho a una segunda opinión, en caso de cirugías, y que el seguro social debe hacer un reembolso. Diferentes investigaciones e instituciones médicas del cáncer, manifiestan que una segunda opinión mejora la calidad de atención y confirma o niega el primer diagnóstico,

motivo por el cual una problemática a ser considera, es dar a los pacientes la posibilidad de una segunda opinión.

Mezzasoma *et al*. (2012), Los microarrays de ADN, son una tecnología de la biología molecular que tiene diversas aplicaciones; permite cuantificar los niveles de expresiones génicas, de los genes en estudio, una de las posibles aplicaciones con los niveles de expresión génica, es la detección de mutaciones en genes específicos para un posterior diagnóstico de la enfermedad. Manifestar que el cáncer es provocado por mutaciones genéticas, y por ello los microarray, son una tecnología muy adecuada, para el diagnóstico de esta enfermedad.

Mezzasoma *et al*. (2012), El microarray de ADN, es una colección de manchas adheridas a la superficie sólida, cada mancha de ADN, contiene miles de copias de una secuencia de ADN específico (secuencias que se desean estudiar). Estas secuencias, suelen ser una sección corta de un gen, la que se utilizan para, hibridar con muestras de ADNc (ADN complementario). La hibridación permite cuantificar la abundancia de transcripciones de las muestras. La intensidad de las manchas determina la cantidad de secuencias que son complementarias una de la otra, la tonalidad manifiesta la diferencia entre la muestra control (adheridas a la superficie). Los microarray de ADN, poseen manchas, y la intensidad de la mancha puede ser evaluada con mayor exactitud por método informáticos.

De Guia *et al*. (2020), Existen diferentes modelos de machine learning, en la detección de cáncer a partir de microarray, algunos utilizan modelos predictivos y otros descriptivos, los diferentes autores presentan modelos lógicos, y las propuestas siempre pretenden mejorar el proceso de identificación y clasificación. De lo manifestado podemos inferir, que otros modelos lógicos de machine learning, permiten diagnosticar de manera más certera el cáncer.

Hengpraprohm & Jungjit (1988), Una de las problemáticas, de los microarray, es la alta dimensionalidad, y enfatiza que es necesario la selección de genes predictores de la enfermedad. Siendo muy necesario propuestas de selección de genes.

Por lo manifestado en párrafos anteriores, se pretende utilizar, varios modelos de machine learning para la detección de cáncer de mama, de forma que cada modelo arroje un diagnóstico, el cual corresponderá a una segunda opinión. Y la decisión final, será una

estrategia consensuada, emulando el diagnóstico de expertos humanos.

Los expertos médicos, manifiestan que la detección temprana de diagnóstico de cáncer, permite que el tratamiento, que se aplica a los pacientes, permita un mejor control de la enfermedad y en la mayoría de casos evita las muertes. También manifiestan, que los pacientes que hayan recibido un tratamiento al cáncer en una fase inicial, en muchos casos tengan una vida bastante normal. Lo manifestado por especialistas, da ha conocer la gran importancia de la detección de cáncer en etapas iniciales.

Una segunda opinión, en caso de enfermedades de riesgo de vida, como el cáncer, proveen al paciente una mejor calidad en el diagnóstico de la enfermedad, corrigiendo falsos positivos o falsos negativos en caso de existir, y también, pueden alterar los tratamientos, por otros más apropiados a la enfermedad.

Proveer al paciente, de mayor cantidad, de métodos de diagnóstico de cáncer, incita, que los costos de diagnóstico tiendan a disminuir, y sean más accesibles.

El uso de tecnologías más recientes, como los microarray, permite, a los especialistas, tener mayor certeza del diagnóstico realizado.

Por los manifiestos anteriores, el presente trabajo tiene como propósito determinar la presencia o ausencia de cáncer de mama, en base al nivel de expresión génica de microarray a través de un ensamble de modelos heterogéneos de machine learning.

El presente informe posee cuatro capítulos. El Capítulo I evidencia la revisión de literatura, seguidamente en el Capitulo II, se identifica la problemática, en el Capitulo III se da a conocer los materiales y métodos utilizados. Y el Capitulo IV se muestran los resultados obtenidos y la discusión con otras investigaciones, finalmente se da a conocer las conclusiones y recomendaciones.

CAPÍTULO I

REVISIÓN DE LITERATURA

1.1. Marco teórico

1.1.1. ADN

El ADN consta de nucleótidos, conformados por cuatro tipos de subunidades adenina (A), citosina (C), guanina (G) y timina (T), figura 1. El ADN está formado por dos cadenas de nucleótidos, conocidos como cadena de ADN, o hebra, la secuencia de cadenas van de forma antiparalela, figura 2, los bloques de ADN, se forman por fosfato, azúcar, y una base o nucleótido, figura 3. El azúcar puede ir unido a una de las siguientes bases: adenina, citosina, guanina, o timina, figura 4.

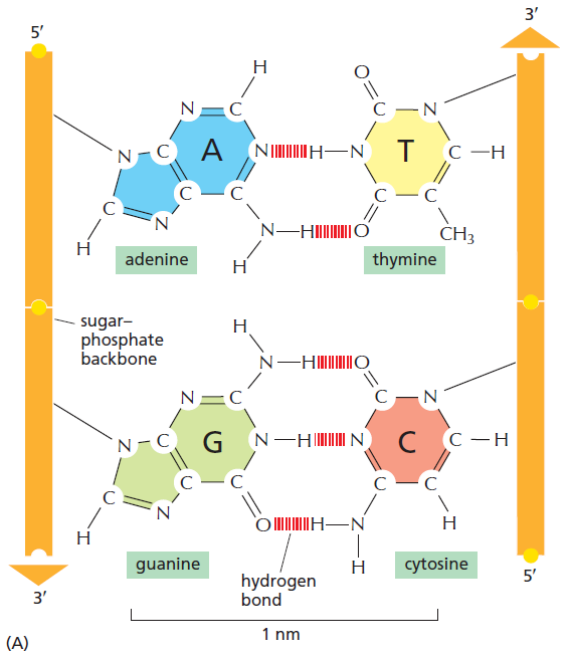


Figura 1. Subunidades del ADN, que proporciona un mecanismo para la herencia

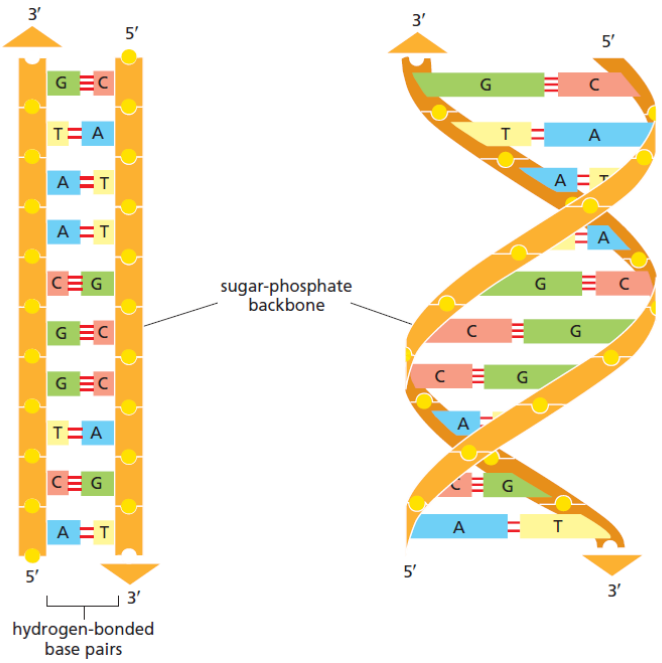


Figura 2. ADN de doble cadena y ADN de doble hélice

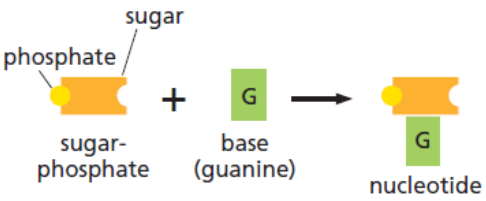


Figura 3. Bloque de construcción de ADN: azúcar-fosfato + base = nucleótido

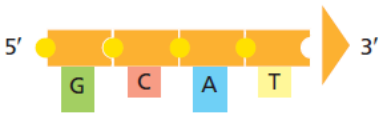


Figura 4. Cadena de ADN

La forma en que se unen los nucleótidos, le da a la hebra de ADN, una polaridad química. Por un extremo un azúcar, que corresponde al fosfato, y por el otro extremo el agujero (hidroxilo). Esta polaridad en una cadena de ADN se referencia por un extremo 5’ (azúcar) y 3’ (agujero), figura 2.

El descubrimiento del ADN respondió a dos preguntas fundamentales: ¿cómo era posible transportar información en forma química?, y ¿cómo la información puede ser copiada de una generación a otra? La respuesta a la primera pregunta se respondió con la comprensión de que el ADN está formado por cuatro tipos

diferentes de nucleótidos, presentados en secuencia. La respuesta a la segunda pregunta provino de la naturaleza de doble cadena de la estructura: porque cada hebra de ADN contiene una secuencia de nucleótidos que es exactamente complementaria a la secuencia de nucleótidos de su hebra asociada, cada hebra puede actuar como plantilla, o molde, para la síntesis de un nuevo complemento hebra. En otras palabras, si designamos las dos cadenas de ADN como S y S′, la cadena S puede servir como plantilla para hacer una nueva hebra S′, mientras que la hebra S′ puede servir como plantilla para hacer una nueva hebra S (Slack, 2013)

1.1.2. ARN

El ARN consta de nucleótidos, conformados por cuatro tipos de subunidades: adenina (A), citosina (C), guanina (G) y Uracilo (U). La cadena de ARN en comparación a la cadena de ADN, es una cadena monocatenaria. En un proceso de transcripción, se genera secuencias de ARN, a partir de una secuencia de ADN, transcribiendo las bases: A, G, C, T a su correspondientes A, G, C, U (Slack, 2013)

1.1.3. Proteína

Las proteínas son las macromoléculas que llevan a cabo diferentes funciones de una célula: como enzimas, las proteínas aceleran enormemente la velocidad de las reacciones metabólicas; como estructuras, las proteínas proporcionan mecanismos soporte tanto dentro de las células como fuera de sus perímetros; como hormonas, permiten el crecimiento y activación de genes; como membrana receptores y transportadores, las proteínas determinan qué tipos de sustancias entran o salen de la célula; como contráctil filamentos y motores moleculares, las proteínas constituyen la maquinaria para los movimientos biológicos; las proteínas actúan como anticuerpos, sirven como toxinas, forman la sangre, coágulos, absorben o refractan la luz y transportan sustancias de una parte del cuerpo a otra. ¿Cómo puede un tipo de molécula tener tantas funciones variadas? La explicación reside en las estructuras moleculares virtualmente ilimitadas que pueden asumir. Cada proteína, sin embargo, tiene una estructura única y definida que le permite llevar a cabo una determinada función. Lo más importante, las proteínas tienen formas y superficies. que les permiten interactuar selectivamente con otras moléculas. Las proteínas, en otras palabras, exhiben un alto grado de especificidad. Es posible, que una proteína reconozca un segmento de ADN que contiene una secuencia específica de ocho

nucleótidos, ignorando todas las otras 65,535 secuencias posibles compuesto por este número de nucleótidos (Lyall, 2016)

1.1.4. Cromosoma

Los ADN nuclear de los eucariotas, se divide en cromosomas, los cromosomas portan genes, las unidades funcionales de la herencia, el ser humano posee, 22 pares de cromosomas, como se aprecia en la figura 5, y, dos cromosomas sexuales X e Y. A toda la secuencia de ADN de los 46 cromosomas en caso del ser humano, se le conoce como genoma (Slack, 2013)

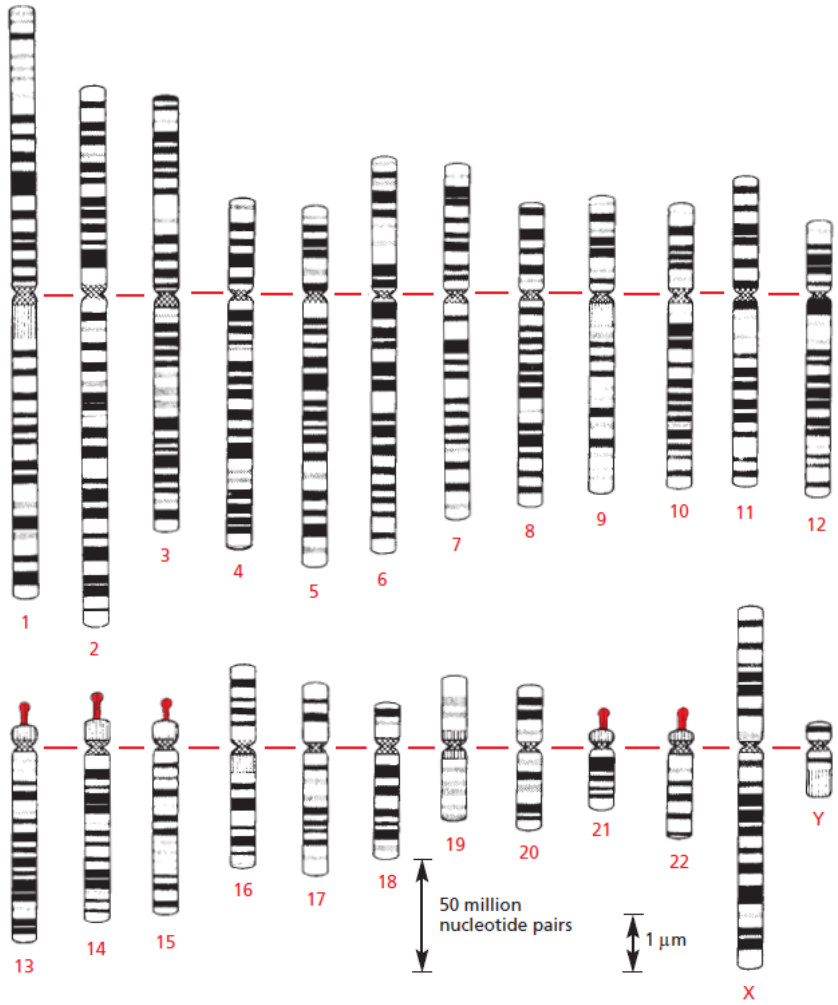


Figura 5. Cromosomas humano

1.1.5. Gen

Es un segmento de ADN, que contiene instrucciones para fabricar una proteína (o un conjunto de proteínas). Un gen es responsable de dar una funcionalidad o

característica al organismo, en la figura 5 se aprecia los cromosomas, en ella las regiones en color oscuro, corresponden a los genes (Slack, 2013)

1.1.6. Dogma central de biología molecular

El dogma de la biología molecular fue descrito por primera vez por Francis Crick, en la que detalla, como a partir de la información de ADN, se genera una secuencia de ARN y luego se codifican aminoácidos, para más adelante ensamblar secuencias de aminoácidos y así generar proteínas. Llamamos ***transcripción*** al proceso de generar secuencias de ARN a partir de ADN, y ***traducción*** al proceso de generar aminoácidos a partir de ARN. El proceso descrito, también es conocido como fabricación de proteínas o síntesis de proteínas, la figura 6 ejemplifica la síntesis de proteínas.

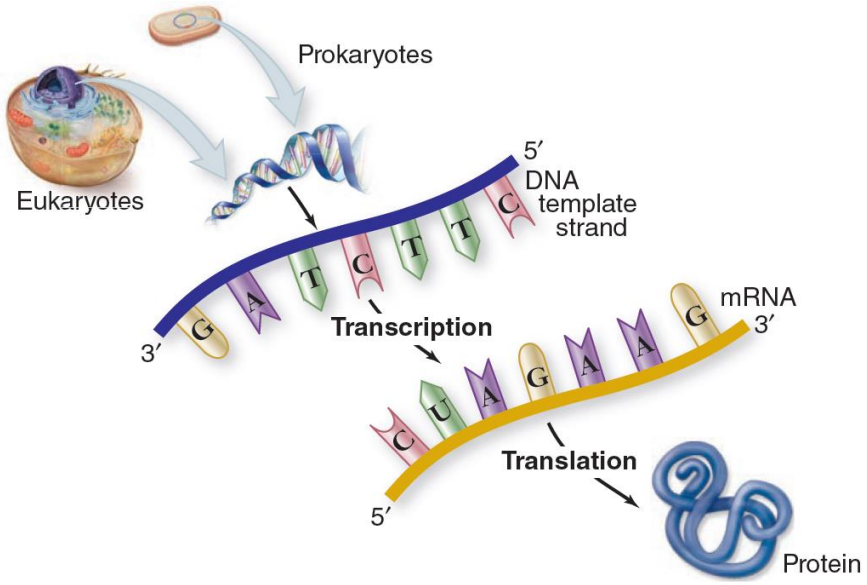


Figura 6. Dogma central de la biología molecular (El ADN se transcribe para producir ARNm, el que se traduce para generar proteínas)

Desde la formulación original del dogma central de la biología molecular, se descubrió una clase de virus llamados ***retrovirus***, que tiene capacidad de convertir su genoma de ARN en una copia de ADN, usando la enzima viral ***transcriptasa inversa***. Esta conversión provoco el replanteo del dogma de la biología molecular, en la cual se incorporó, el flujo inverso de información de ARN a ADN, El cual se muestra en la figura 7.

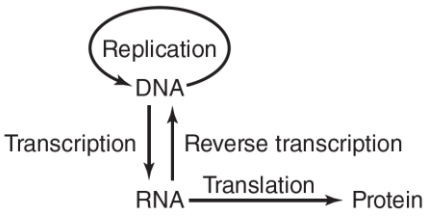


Figura 7. Dogma completo de la biología molecular

La figura 7, muestra las operaciones conocidas hasta la fecha en la biología molecular. En la fase de mitosis (duplicación de células) ocurre la replicación de ADN, en la que se crea una copia de la secuencia de ADN, de forma que cada célula posea la misma copia de ADN. En la fabricación de proteínas, se realiza el proceso de transcripción, generando una secuencia de ARN a partir de ADN y luego el proceso de traducción generando aminoácidos a partir de ARN. Los retrovirus tienen capacidad de generar secuencia de ADN a partir de secuencia de ARN, operación conocida como transcripción reversa (Mason *et al*, 2008)

1.1.7. Código genético

El código genético, es la codificación que se utiliza para generar aminoácidos a partir de codones (tres nucleótidos consecutivos), cuyos detalles se aprecian en la tabla 1. Algunos puntos relevantes son: ya que existen 4 nucleótidos distintos y se combina en tríos, se tiene un total de 64 codones, de los cuales se utilizan 61, para generar aminoácidos. Tres codones, UAA, UGA y UAG, están reservados para la función de parada (parada en la generación de aminoácidos). Se utiliza el codón AUG, para señalar el inicio, (inicio de codificación de aminoácidos), también el triplete AUG cumple la función de generar el aminoácido metionina (Met). En la tabla 1, se aprecia que solo se codifican 20 aminoácidos distintos, ya que existen diferentes codones que pueden generar un mismo aminoácido.

Tabla 1

Código genético, que transcribe un codón (tres nucleótidos) en aminoácidos

PRIMERA BASE	SEGUNDA BASE U		SEGUNDA BASE C		SEGUNDA BASE A		SEGUNDA BASE G		TERCERA BASE
U	UUU	fenilalanina	UCU	serina	UAU	tirosina	UGU	cisteina	U
	UUC		UCC		UAC		UGC		C
	UUA	leucina	UCA		UAA	stop	UGA	stop	A
	UUG		UCG		UAG		UGG	triptófano	G
C	CUU	leucina	CCU	prolina	CAU	histidina	CGU	arginina	U
	CUC		CCC		CAC		CGC		C
	CUA		CCA		CAA	glutamina	CGA		A
	CUG		CCG		CAG		CGG		G
A	AUU	isoleucina	ACU	treonina	AAU	aspargina	AGU	serina	U
	AUC		ACC		AAC		AGC		C
	AUA	metionina	ACA		AAA	lisima	AGA	arginina	A
	AUG		ACG		AAG		AGG		G
G	GUU	valina	GCU	alanina	GAU	aácido aspartico	GGU	glicina	U
	GUC		GCC		GAC		GGC		C
	GUA		GCA		GAA	ácido glutámico	GGA		A
	GUG		GCG		GAG		GGG		G

Un codón consta de tres nucleótidos, leídos en la secuencia de la tabla 1 (tabla de 3 entradas), (primera, segunda y tercera letra respectivamente). A través del cual se generan aminoácidos. Un aminoácido, puede ser definido por varios codones, por ejemplo, la treonina se especifica mediante cuatro codones diferentes: ACU, ACC, ACA y ACG (Mason *et al*, 2008)

1.1.8. ADN complementario

Desnaturalización del ADN

Cuando el ADN se disuelve en solución salina y la solución se calienta lentamente, ocurre la separación de las hebras, al finalizar el proceso la solución contiene moléculas monocatenarias completamente separadas de sus

hebras complementarias, a esta cadena monocatenarias de ADN se conoce como ADN complementario (ADNc).

Renaturalización del ADN

Si se enfrían lentamente una solución de ADN, que había sido desnaturalizado térmicamente, el ADN recupera las propiedades de la doble hélice; y se recobra sus propiedades, este proceso se conoce como: renaturalización o recocido.

De lo anterior, afirmamos que, él ADN complementario, es un cadena monocatenaria de ADN, que puede obtenerse a través de la desnaturalización de ADN, como muestra la figura 8 (Lyall, 2016)

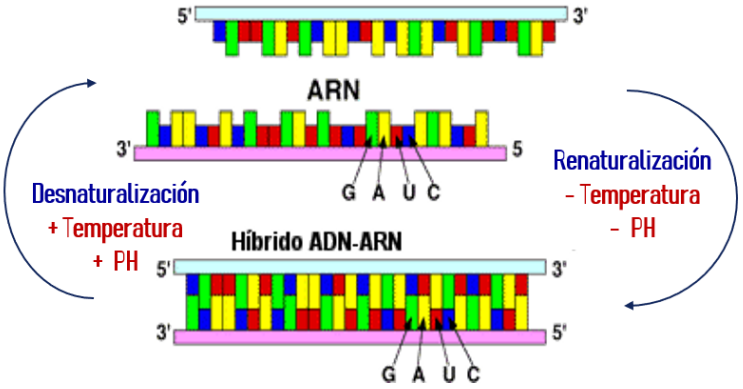


Figura 8. Desnaturalización y Renaturalización de ADN

1.1.9. Hibridación

El recocido o desnaturalización de ADN, ha permitido desarrollar la metodología denominada **hibridación de ácidos nucleicos**, en la que, cadenas complementarias de ácidos nucleicos de diferentes fuentes pueden mezclarse para formar moléculas de doble cadena (híbridas), como se muestra en la figura 9 (Lyall, 2016)

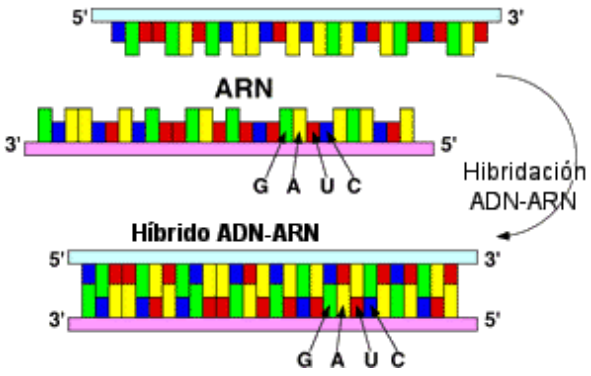


Figura 9. Hibridación de ADN-ARN

1.1.10. Mutación

Una mutación es un cambio hereditario en el material genético. Un cambio tan simple como alterar una sola base, puede resultar en una sustitución de aminoácidos que puede conducir a un fenotipo (rasgos observables de un organismo) clínico debilitante. En la figura 10, se aprecie un ejemplo, en el que, se cambia la base **A** por una **T**, provocando el reemplazo de aminoácido glutamina por valina, lo que provoca que se altere el glóbulo rojo y la célula normal se convierta en una célula falciforme (figura 11), lo que lleva al estado de enfermedad.

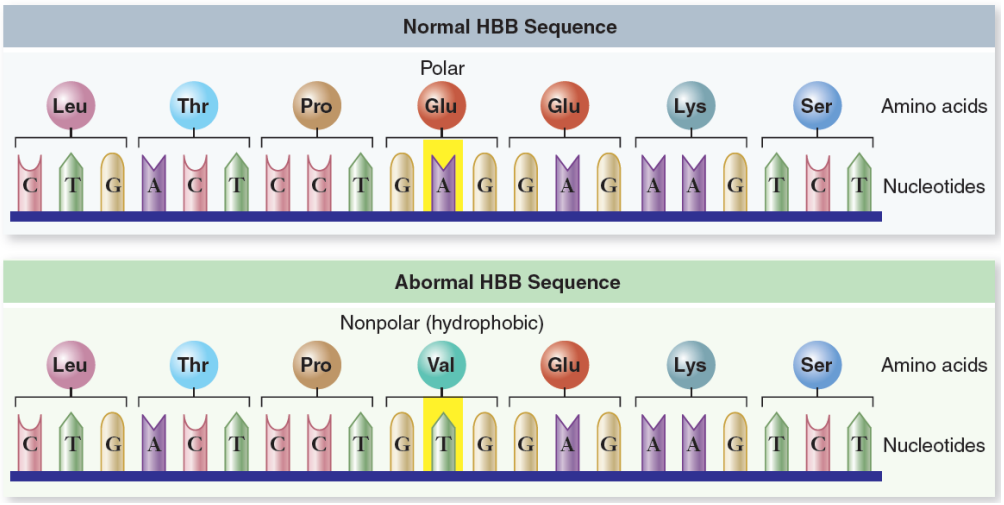


Figura 10. La anemia de la célula falciforme es causada por una proteína alterada.

La hemoglobina está compuesta por un tetrámero de dos α-globinas y dos cadenas de β-globina, motivo que provoca la sustitución de aminoácido glutamina por valina, lo que provoca regiones hidrofóbicas en la superficie de la proteína "pegajosa" lo que lleva a distorsionar la forma de los glóbulos rojos

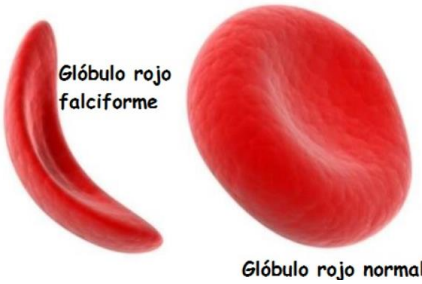


Figura 11. Glóbulo rojo normal y falciforme

Las mutaciones pueden ser provocada por cuatro operaciones básica:

Eliminación: eliminación de una o varias bases consecutivas, en una secuencia de ADN.

Duplicación: duplicación de uno o más bases consecutivas, en la secuencia de ADN.

Inversión: Inversión de una dos o más bases consecutivas, en la secuencia de ADN.

Translocación: Cuando una parte de un cromosoma se rompe y se une con otro cromosoma.

La figura 12, muestra ejemplo de las mutaciones cromosómicas.

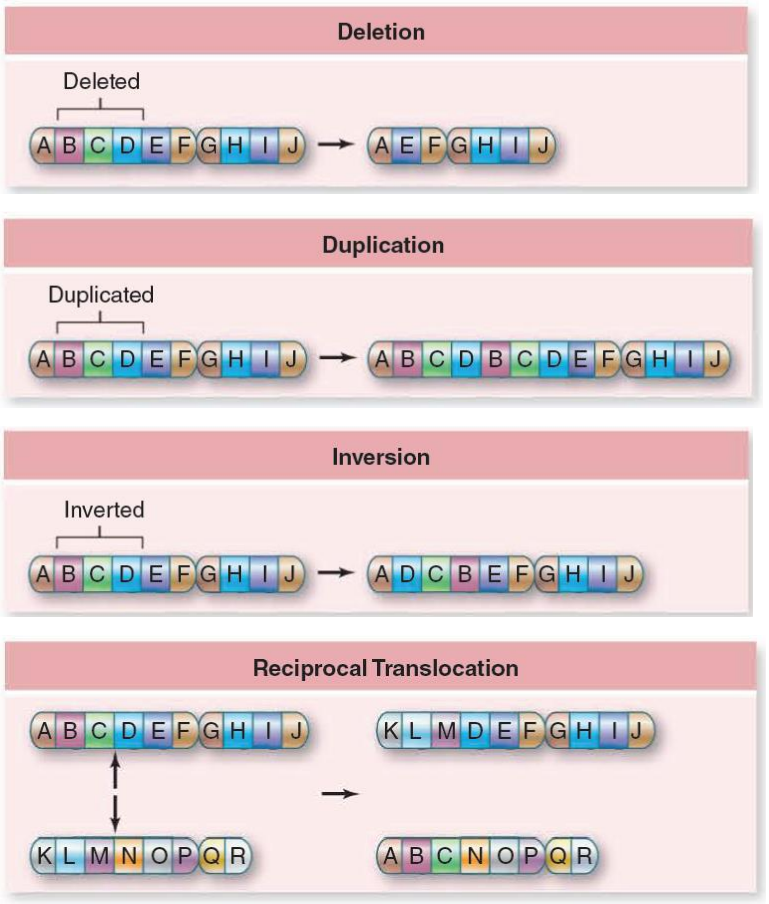


Figura 12. Operaciones básica de mutaciones cromosómicas

Las mutaciones, pueden provocar cambios desfavorables para el organismo como el ejemplo de la figura 11; pero también puede provocar cambios favorables para el organismo, provocando evolución de las especies. También es posible que una mutación no genere cambios, si la mutación ocurre en una región no codificante del ADN o la mutación genera la misma secuencia de aminoácidos (Lyall, 2016)

1.1.11. Cáncer

Los estudios han revelado que el cáncer es una mutación, que convierte una célula normal, en una célula cancerosa, las células cancerosas tienen la particularidad de duplicarse incontroladamente. Se ha detectado que existen dos tipos de genes implicados: **oncogenes** que pueden causar cáncer con ganancia de función mutación o activación inapropiada, y genes supresores de tumores que conducen al cáncer con mutaciones de pérdida de función. Estos también pueden considerarse como mutaciones positivas de tipo "acelerador", y mutaciones negativas de tipo “freno”.

Algunos genes pueden mutar en múltiples tipos de tumores, y diferentes tumores pueden tener desde unas pocas, hasta cientos de mutaciones. Esta avalancha de variedad de mutaciones, ha generado la necesidad de análisis de problemas. No existe un modelo simple, en la que, las mutaciones de un conjunto de genes conduzcan siempre a un tipo específico de cáncer; pero suele existir cierta regularidad. También se ha percibido que un mismo tipo de tumores, tienen heterogeneidad. A la larga, caracterizar cada tipo de tumor ayudará tanto en el diagnóstico como en el diseño tratamientos basados en el genotipo (conjunto de genes de un organismo) del paciente y las mutaciones en el tumor de cada paciente (Slack, 2013)

1.1.12. Cáncer de seno

El cáncer de mama o cáncer de seno, se origina cuando las células mamarias crecen sin control. Las células cancerosas del seno forman un tumor, que puede ser detectado por imágenes radiográficas o puede palpar como una masa o bulto. El cáncer de seno es frecuente que se presente en las mujeres, pero también suele presentarse en varones. Existen tumores no cancerosos (benignos), que son crecimientos anormales, pero, no se propagan fuera de los senos. Los cánceres de seno, pueden originarse en diferentes partes, la mayoría comienza en los conductos que llevan la leche hacia el pezón (cánceres ductales), otros se originan en las glándulas que producen leche (cánceres lobulillares), también existen, los menos frecuentes como el tumor filodes y el angiosarcoma, y un número más reducido comienza en otros tejidos del seno.

El cáncer de mama, puede ser categorizado en 5 estadios, los estadios indican el grado de extensión.

Estadio 0: Las células cancerosas no presentan carácter invasivo y están localizadas únicamente en el interior de los conductos mamarios: el tumor es un cáncer in situ.

Estadio 1: El cáncer se concreta en un nódulo (bulto) inferior a los 2 cm de diámetro y habitualmente no se extiende aún a otros tejidos fuera de la mama.

Estadio 2: El tumor sigue siendo inferior a 2 cm, pero se extiende a los ganglios de la axila, también puede ocurrir que: el nódulo haya crecido, sin sobrepasar los 5 cm, con un 50% de probabilidades de haberse extendido a los ganglios axilares.

Estadio 3: El nódulo no ha alcanzado aún 5 cm, pero se ha extendido ya a las axilas. O el tumor, se extiende por los tejidos cercanos a la glándula mamaria.

Estadio 4: Las células cancerosas se han extendido a otros tejidos y órganos del cuerpo, proliferando también en alguno de ellos, condición conocida como metástasis, la figura 13 (American Cancer Society, 2023)

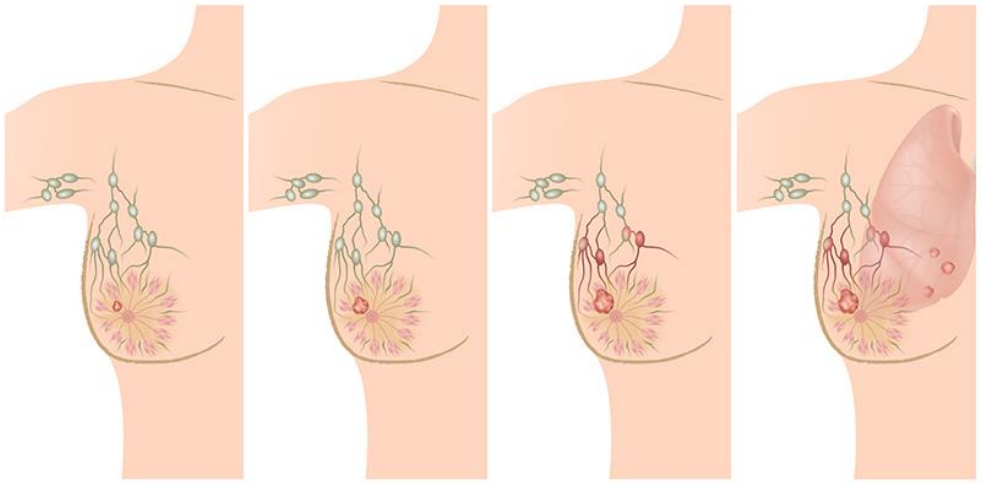


Figura 13. Estadios del cáncer de mama, de izquierda a derecha, 1er, 2do, 3er y 4to estadio respectivamente

1.1.13. Microarray de oligonucleótido

El microarray de oligonucleótido, son chips de ADN o ARN. Que consisten en una superficie sólida, en la cual se deposita una colección de fragmentos de ADN

(sondas). Estos fragmentos pueden ser ADN complementario o ARN, y contienen secuencias que corresponden a fragmentos de ADN del organismo con el que se quiere trabajar, figura 14. Estas secuencias de ADN o ARN, son fijadas a la superficie sólida que contendrá el microarray en forma de puntos ordenados en forma de matriz. La superficie sobre la que se depositan suele ser de materiales semiconductores o de cristal. Posteriormente se aísla el ARN o ADN de la muestra (paciente) con la que se quiere trabajar. Estas secuencias se marcan posteriormente mediante fluorescencia. Una vez construido el microarray con las secuencias deseadas, y teniendo la muestra del paciente, procesada y marcada, se hibridan ambos. Las secuencias del microarray (sondas) y la de muestra se unen (hibridan), y como las muestras están marcada con fluorescencia, es posible detectar donde se encuentran la mayor proporción de marcadores fluorescentes. El microarray, una vez hibridado, se escanea utilizando técnicas de procesamiento de imagen, mediante láseres y microscopios confocales, para detectar la presencia de estos puntos fluorescentes, y así poder realizar una medida indirecta de la abundancia de las secuencias en la muestra.

Entre los usos de esta técnica, está el análisis de expresión génica, para determinar la intensidad de la fluorescencia y así estimar el nivel de expresión de genes (Tabas Madrid, 2018)

La figura 17, muestra, de forma gráfica el proceso que se realiza en el análisis de microarray ARN y ADN.

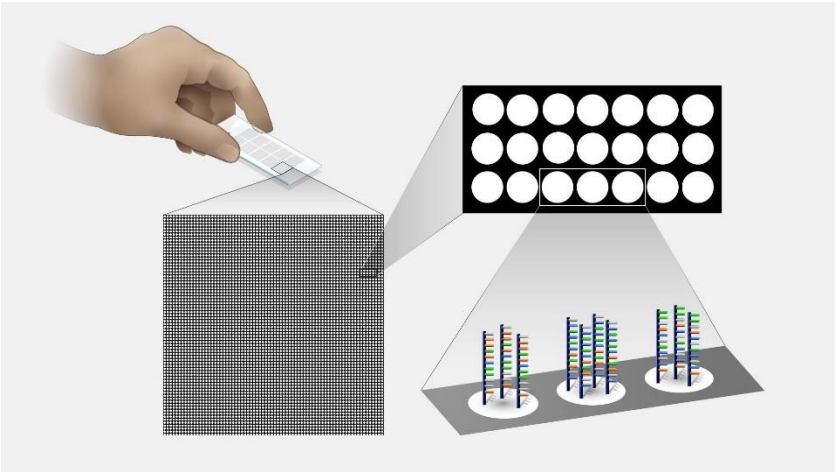


Figura 14. Microarray de oligonucleótidos

Microarray de oligonucleótidos de ARN

En cada celda de un chip se pegan miles de copias de ARNm (sonda). Celdas distintas contienen ARNm también distintos. Todas las celdas contienen el mismo número de segmentos, figura 15 (Tabas Madrid, 2018)

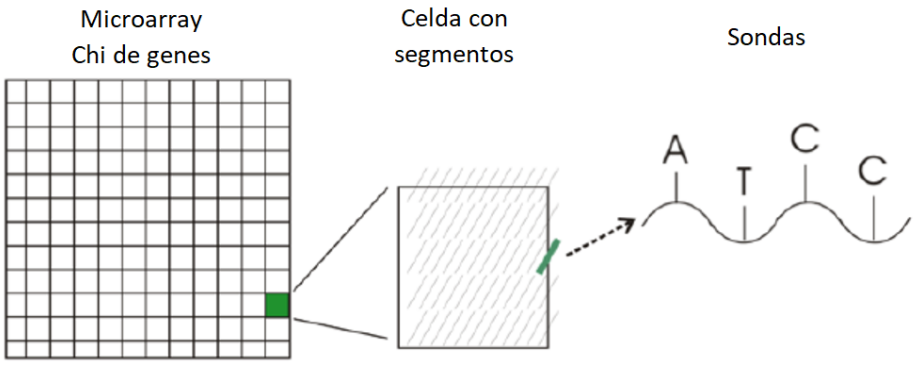


Figura 15. Chip de sondas

Por otra parte, se prepara una muestra de ARNm, de nuestro caso de estudio (paciente), luego se marcan con una sustancia fluorescente, posteriormente se mezclan las sondas del microarray, provocando la hibridación por atracción bioquímica, de este modo algunas sondas tendrán adosadas ARNm que la complementa, pero, ya que están marcadas con un fluorescente se podrá percibir la cantidad de sondas que tienen fluorescencia, figura 17 (Tabas Madrid, 2018)

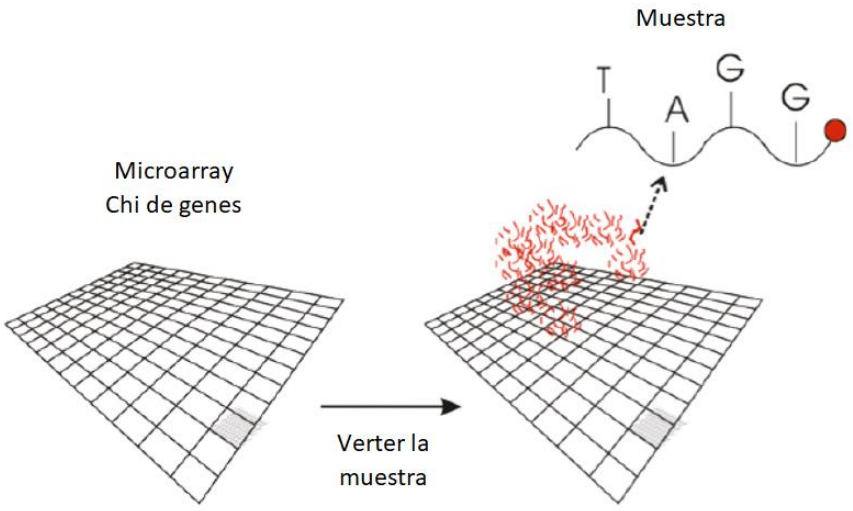


Figura 16. Hibridación

Finalmente se realiza el análisis de imagen, convirtiendo las cantidades de secuencias hibridadas, en una intensidad de luz (un número), proceso que se realiza, mediante un escáner que lee la intensidad de luz y convierte a correspondiente valor numérico, como muestra la figura 17 (Tabas Madrid, 2018)

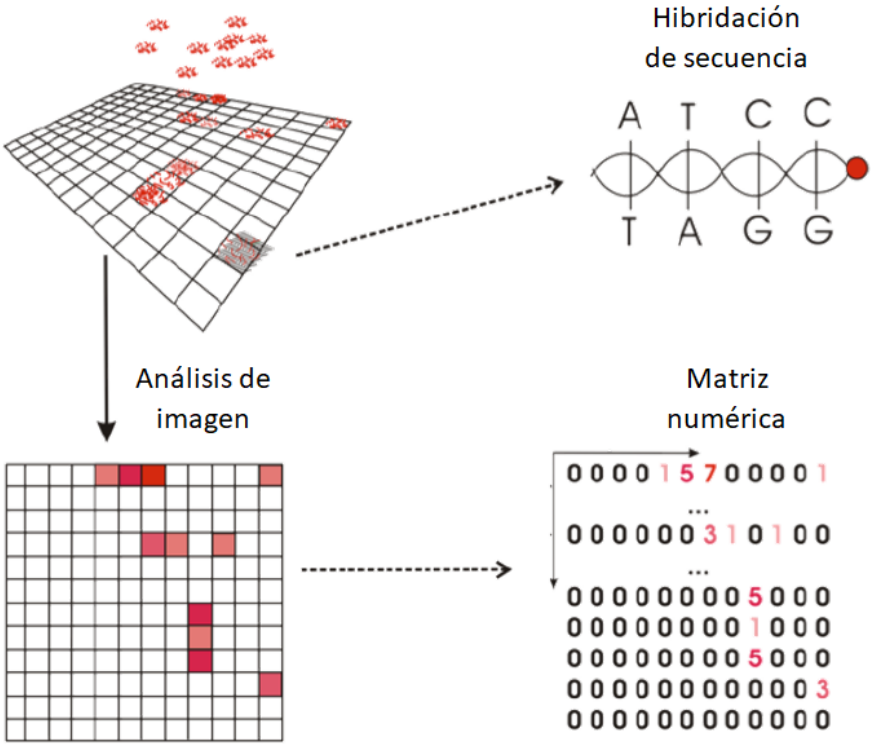


Figura 17. Análisis de imagen

Las matrices de expresión génica, contienen la información de muchos microarray. Para lograr este hecho, se recolecta información de un conjunto de microarray, la información de cada microarray se estira a una columna de la matriz de expresión, como se aprecia en la figura 18. Donde las columnas contienes diferentes condiciones (array, ensayo, casos, muestras, factores experimentales) y las filas contiene sondas o conjunto de sondas, los que se consideraran como genes de un individuo o paciente (Tabas Madrid, 2018)

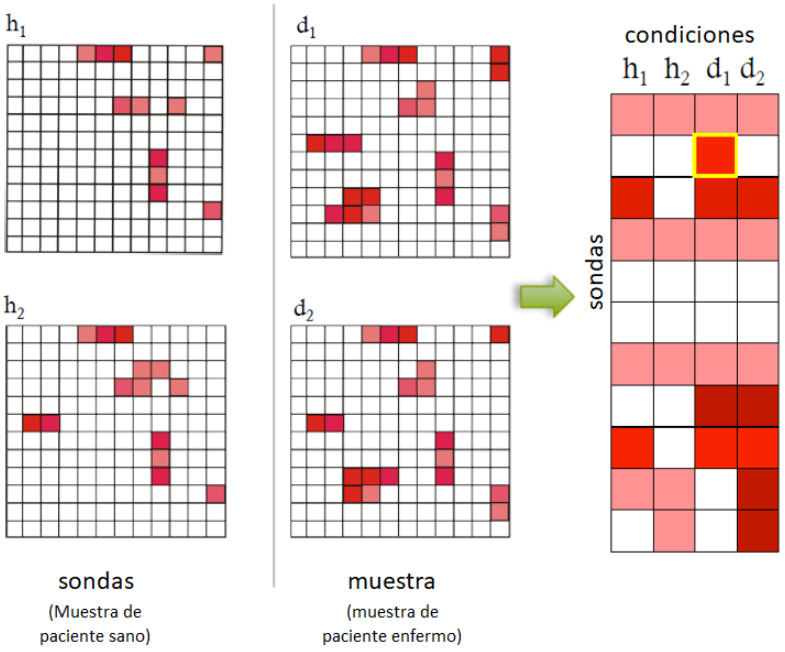


Figura 18. Matriz de expresión génica

Las matrices de expresiones génica, suelen registrar la información con una resolución considerable, como se muestra en la figura 19, y por este motivo es necesario que para un análisis de imagen se utilicen métodos informáticos (Tabas Madrid, 2018)

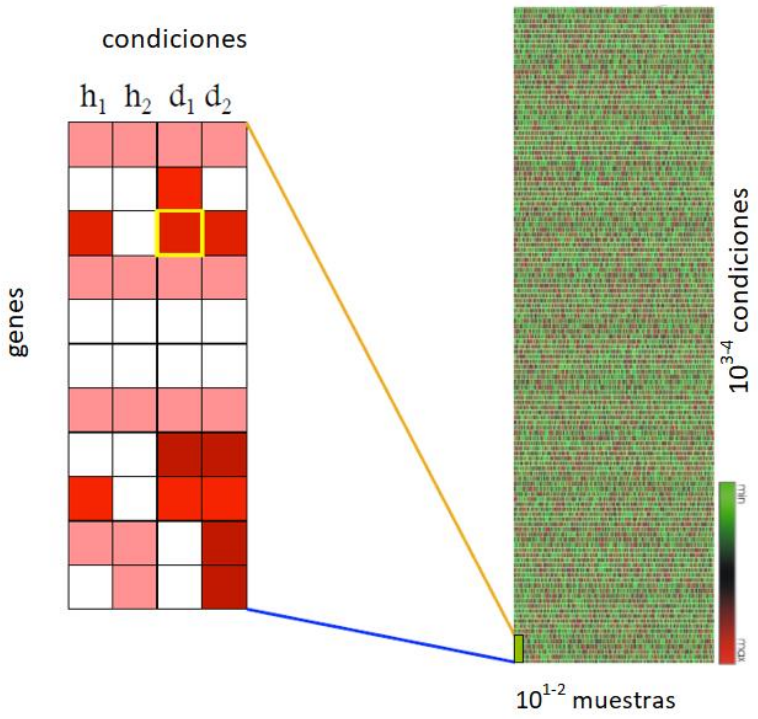


Figura 19. Resolución de matrices de expresión génica

Microarray de ADN complementario

En los microarray de ADN, se extrae dos muestras, una muestra de un individuo sano (control), y una muestra de un paciente sospechoso de enfermedad (mutante), las muestras sanas se le suministra una sustancia fluorescente verde, y las muestras mutantes una sustancia fluorescente roja, se mezclan las muestras, y se vierten en un chip de microarray en la que ocurre la hibridación. Como se aprecia en la figura 20 (Tabas Madrid, 2018)

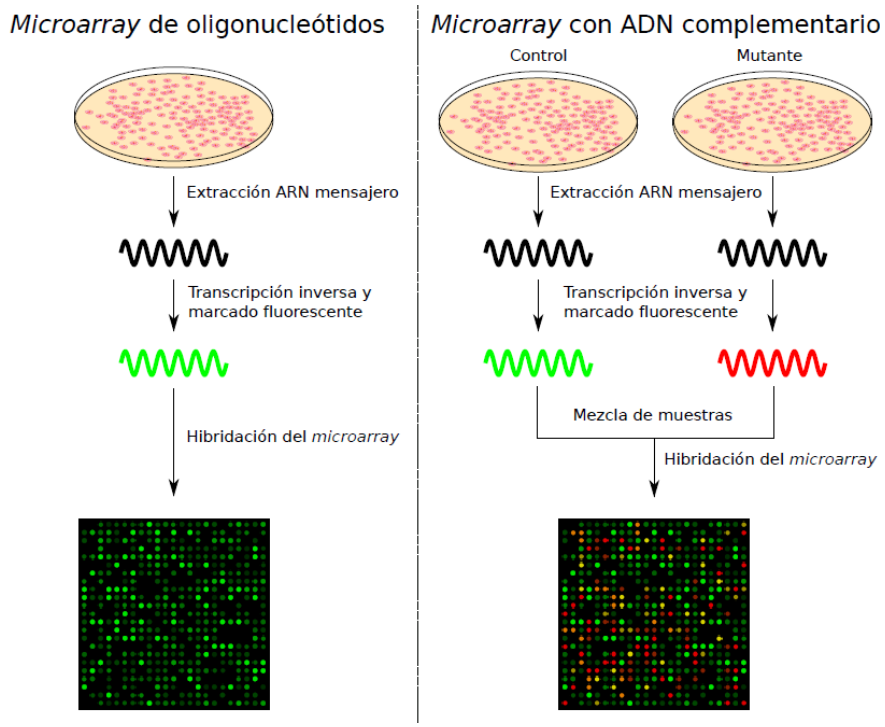


Figura 20. Análisis de microarray de oligonucleótidos y de ADNc

1.1.14. Aprendizaje Automático

El objetivo de un algoritmo de aprendizaje automático, es permitir que el algoritmo aprenda a partir de un conjunto de datos, denominado conjunto de datos de entrenamiento, es decir crear un modelo o una hipótesis, capaz de relacionar los atributos de entrada de un objeto con los atributos de salida del objeto. La hipótesis es la tentativa de explicación de algún fenómeno o problema que puede ser corroborado mediante observación o experimentación.

Cuando un algoritmo de aprendizaje automático está aprendiendo a partir de un conjunto de datos, está planteándose una hipótesis. El espacio de posibles hipótesis, capaz de describir como se relacionan los objetos, hacen uso de representaciones, cada algoritmo de aprendizaje automático tiene su propia forma de representar la información para describir las hipótesis (matrices de pesos en las redes neuronales, árboles de decisión, reglas de decisión, tablas de probabilidades en Naïve Bayes, etc.)

Las tareas de aprendizaje automático pueden ser: predictivas y descriptivas. En las tareas predictivas, la meta es encontrar una función que utilice los datos de entrenamiento, y determine un valor o etiqueta a los nuevos valores, la predicción utiliza el paradigma de aprendizaje supervisado.

En las tareas descriptivas, la meta es explorar o describir un conjunto de datos, la descripción de datos utiliza el paradigma de aprendizaje no supervisado, la figura 21, muestra la clasificación de tareas de aprendizaje automático (Faceli & Lorena, 2021)

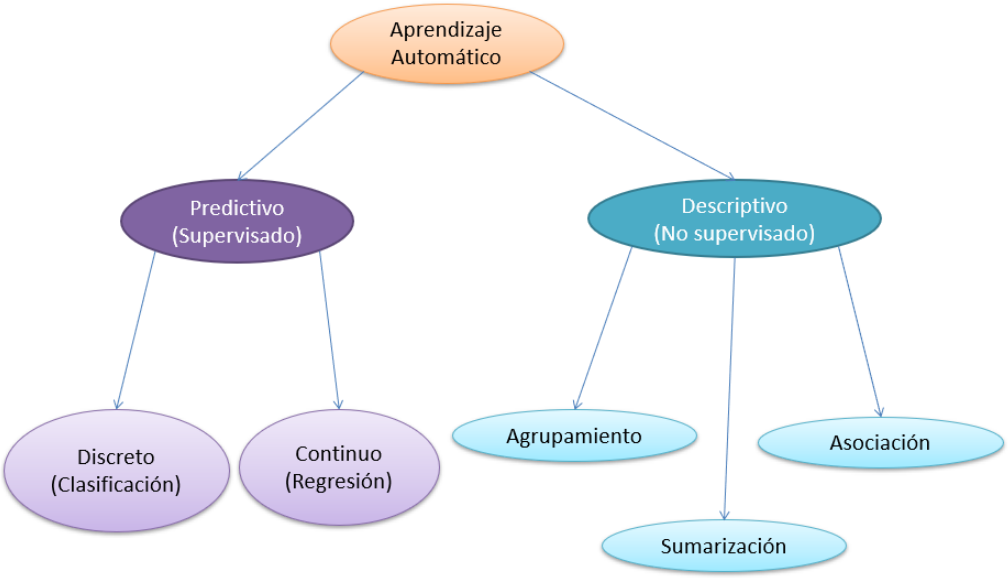


Figura 21. Tareas de Aprendizaje Automático

1.1.15. Algoritmos de clasificación

Los algoritmos de clasificación, pertenecen a los algoritmos predictivos del tipo discreto, que aplican un paradigma de aprendizaje supervisado.

Existe una gran variedad de algoritmos de clasificación, a continuación, listamos por categorías los algoritmos de mayor uso:

Métodos basados en distancia

- Un vecino más cercano
- K vecinos más cercanos
- Raciocino basado en casos

Métodos probabilísticos

- Clasificador Naïve Bayes
- Redes bayesianas para clasificación

Métodos basados en búsqueda

- Árboles de decisión para clasificación
- Reglas de división para clasificación
- Reglas de decisión

Métodos basados en optimización

- Redes neuronales artificiales
    - Perceptrón y Adaline
    - Perceptrón multicapa
- Máquina de vector de soporte
    - Lineales
    - No lineales
- Redes neuronales convolucionales
- Redes recurrentes
- Transformers

Modelos múltiple predictivos

- Combinación de predictores
    - Voting
    - Seriales
- Combinación de algoritmos homogéneos

- Bagging
- Boosting

- Combinación de algoritmos heterogéneos
  - Stacking

1.1.16. K vecinos más cercanos (KNN)

El algoritmo KNN, representa cada objeto como un punto en *n* dimensiones, e incorpora cada objeto, a una clase, usando el método de votación por mayoría simple de sus *k* vecinos más cercanos. La identificación de vecinos, es aplicando una medida de similitud, por distancias.

Distancia euclidiana:

Sea dos objetos *P* y *Q*, con n atributos, los atributos de *P* será *P(p1, p2, p3, ... pn)* y *Q(q1, q2, q3, ..., qn)*, el espacio *n* euclidiano, entonces es la distancia absoluta de la línea segmento dibujado de *P* a *Q*, como se aprecia en la *ecuación (1)*.

$$dE_{(P,Q)} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \qquad \textit{Ecuación (1)}$$

También es, es posible utilizar como medida de similitud, distancia de Manhattan, Minkowski, Mahalanobis, coseno, Hamming, entre otros (Ray & Ray, 2021)

1.1.17. Máquina de vector de soporte para clasificación (SVC)

En las máquinas de vectores de soporte, cada objeto se representa como un punto, en ***n*** dimensiones, y se busca el hiperplano óptimo de separación que pueda distinguir dos clases, un hiperplano será optimo, cuando está ubicado en la mejor posición de la frontera, como se aprecia en la figura 22.

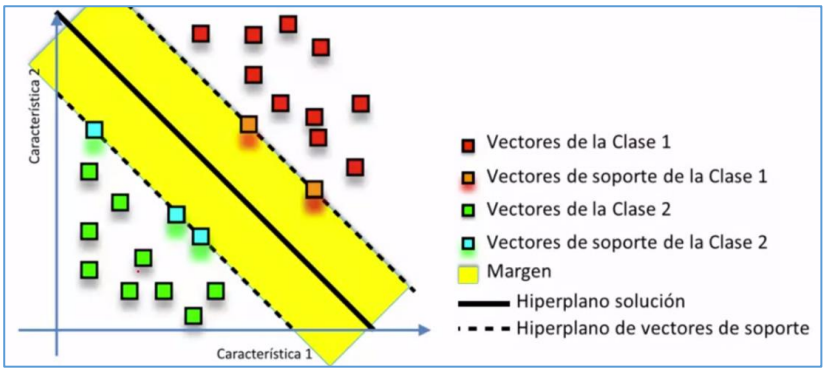


Figura 22. Hiperplano identificado por SVM

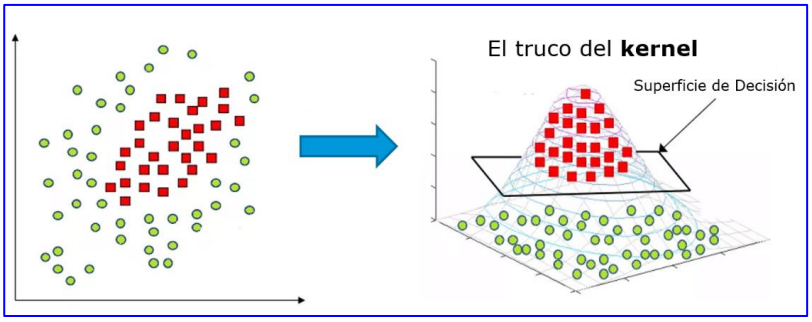


Figura 23. Kernel de SVM

En caso de resolver problemas lineales, las SVC utilizan un kernel, como muestra la figura 23, El kernel, se construye a partir de un conjunto de funciones matemáticas, las que reciben datos de entrada y las convierten en la salida deseada, *ecuación (2)*.

$$K(p,q) =< f(p), f(q) > \qquad \textit{Ecuación (2)}$$

Donde *K* es la función kernel (núcleo), *p*, *q* son objetos de *d* dimensiones, *f* es una función, que va de una dimensión *n*, a una dimensión *m*, normalmente *m* es mayor que *n*, <*p*, *q*> denota el producto escalar, de los diferentes kernels, los kernel utilizados en el presente trabajo son: lineal, polinomial, sigmoide y gaussiana.

Kernel polinomial

Se trata de un kernel no estacionario, dado por *ecuación (3)*, donde *p*, *q*, son los datos de entrada, y es posible reajustar la pendiente con el valor de α, y *m* es una constante que permite variar el sesgo, siendo *x*, el grado del polinomio.

$$k(p,q) = (\alpha p^T q + m)^x \qquad \textit{Ecuación (3)}$$

También se posee los kernel: gaussiano, radial, Laplace, tangente, sigmoide, bessel, anova, spline, lineal, entre otros (Ray & Ray, 2021)

1.1.18. Árbol de decisión (AD)

Los árboles de decisión, son modelos que utilizan un árbol (figura 24), en la que se distinguen: Los nodos de decisión (color celeste), en los que, se evalúa un atributo, y dependiendo de su valor, se realiza bifurcación a una, u otra rama. Los nodos hojas (color amarillo, naranja y rojo), los nodos hojas, por ser nodos terminales, contiene la clase a la que pertenece un dato en particular, en la figura 24, los nodos amarrillos corresponden a objetos de la clase 1, los naranjas a la clase 2, y los rojos a la clase 3.

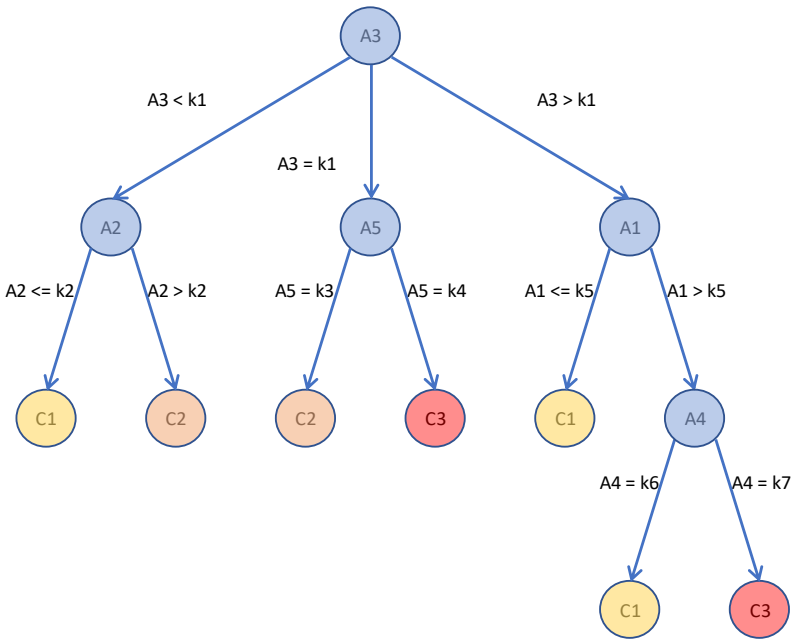


Figura 24. Árbol de decisión

El mayor problema de construcción de árboles de decisión, está, en la elección del atributo que va en el nodo raíz del árbol, y el punto de corte, es decir los valores del atributo, que se utilizan, para realizar bifurcaciones a una rama, u otra, este problema se resuelve por diferentes estrategias, entre ellas tenemos índice de impureza Gini, Ganancia de información con entropía o incertidumbre.

Impureza Gini

$$I_G(p) = 1 - \sum_{i=1}^{n} {p_i}^2 \qquad \textit{Ecuación (4)}$$

El índice de impureza Gini, determina el grado de impureza que genera un atributo, al ser utilizado en un nodo de decisión, el atributo, además, está, sujeto a una o varias condiciones, para generar bifurcaciones. En la *ecuación (4)*, *pi*, representa la probabilidad de la i-ésima clase, siendo la cantidad de clase igual a *n*.

Entropía

La entropía, mide el grado de incertidumbre, en base, al atributo evaluado, cuya ecuación matemática se aprecia en la *ecuación (5)*.

$$Entropia = -\sum_{i=1}^{k} p_i log_2(p_i) \qquad \textit{Ecuación (5)}$$

En la ecuación, *pi* corresponde a la probabilidad de la i-ésima clase, siendo la cantidad de clases igual que *k* (Faceli & Lorena, 2021)

1.1.19. Clasificador Naïve Bayes (NB)

El clasificador Naïve Bayes, considerado también un clasificador ingenuo, utiliza el teorema de Bayes, para, determinar probabilidades a posteriori, en base a probabilidades a priori. En la figura 25, se aprecia, la red de clasificación de Naïve Bayes, en ella, el nodo raíz determina la probabilidad a posteriori, de la ocurrencia de una clase. Los nodos hojas, contienen las probabilidades a priori, probabilidad de ocurrencia de una clase condicionado al valor de un atributo (probabilidad condicionada).

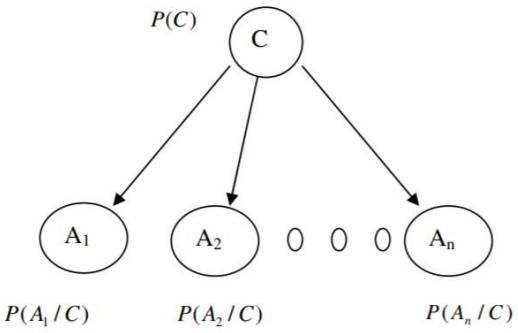


Figura 25. Red del clasificador Naïve Bayes

La probabilidad de ocurrencia de una clase (nodo raíz), está, en función a, las probabilidades condicionadas (nodos hojas), el modelo matemático utilizado se aprecia en la *ecuación (6)*.

$$P(C=c/a_1,a_2,...,a_n)=\frac{P(C=c)\prod_{i=1}^{n}P(a_i/C=c)}{P(a_1,a_2,...,a_n)}$$ *Ecuación (6)*

Donde:

***P(C)**: Probabilidad de ocurrencia de una clase.*

***P(a_i/C=c)**: Probabilidad condicionada, probabilidad que la clase C pertenezca a la clase c, sujeto al atributo a_i.*

***Π**: Multiplicación*

***P(a_1, a_2, …, a_n)**:Sumatoria de probabilidades condicionadas de cada atributo*

$$P(A_i/c)=\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(X-\mu)^2}{2\sigma^2}\right)$$ *Ecuación (7)*

En caso, que los atributos posean valores continuos, no es posible determinar la frecuencia, para ellos se suele utilizar la función de distribución normal, asumiendo que los datos siguen una distribución normal, cuyo modelo matemático se aprecia en la *ecuación (7)* (Faceli & Lorena, 2021)

1.1.20. Red neuronal perceptrón multicapa (MLP)

La red neuronal artificial, basa su funcionamiento en las redes neuronales biológicas, cuya unidad de procesamiento son las neuronas, la figura 26 muestra un dibujo de la neurona biológica. El principio de funcionamiento de la neurona, es como sigue: la neurona, recibe señales de entrada a través de sus dendritas, estas señales son ponderadas por el cuerpo de la neurona, luego la neurona envía una señal a otras neuronas a través del axón. El envío de señal, por el axón, depende de la intensidad de señal recibida por las dendritas, si las señales recibidas son de alta intensidad, la neurona se activa, y envía una señal por el axón. Esté principio, emulan las neuronas artificiales, que son la unidad básica de las redes neuronales artificiales.

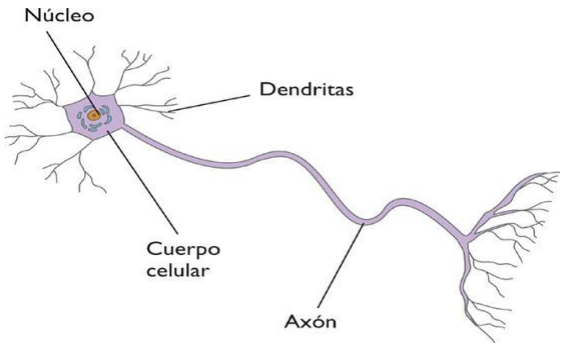


Figura 26. Neurona bilógica

En la figura 28, muestra el modelo de una neurona artificial, en ella se tiene un conjunto de entradas: $X = \{x_1, x_2, x_3, \ldots, x_n\}$, cada señal de entrada tiene asociado pesos: $W = \{w_1, w_2, w_3, \ldots, w_n\}$, el cuerpo de la neurona aplica la operación $\Sigma X.W$, operación conocida como producto punto, luego una función de activación, que decide si la neurona se activa o no, en caso de activarse envía una señal. la función de activación es variada, la *ecuación (8)*, muestra la función de activación sigmoide.

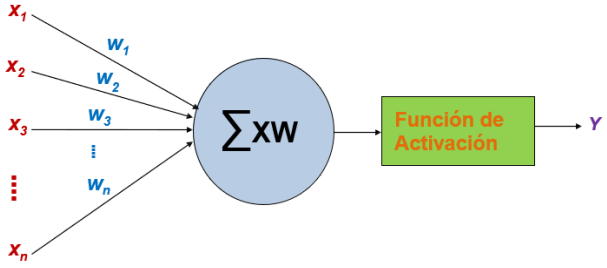


Figura 27. Neurona artificial

$$f(x) = \frac{1}{1 + e^{-x}}$$ *Ecuación (8)*

Las redes neuronales artificiales del tipo perceptrón multicapa (MLP ≅ Multi Layer Perceptron), son redes que constan de muchas unidades de neuronas y están organizadas en capas, emulando una red neuronal biológica, la figura 28, muestra un ejemplo particular de perceptrón multicapa.

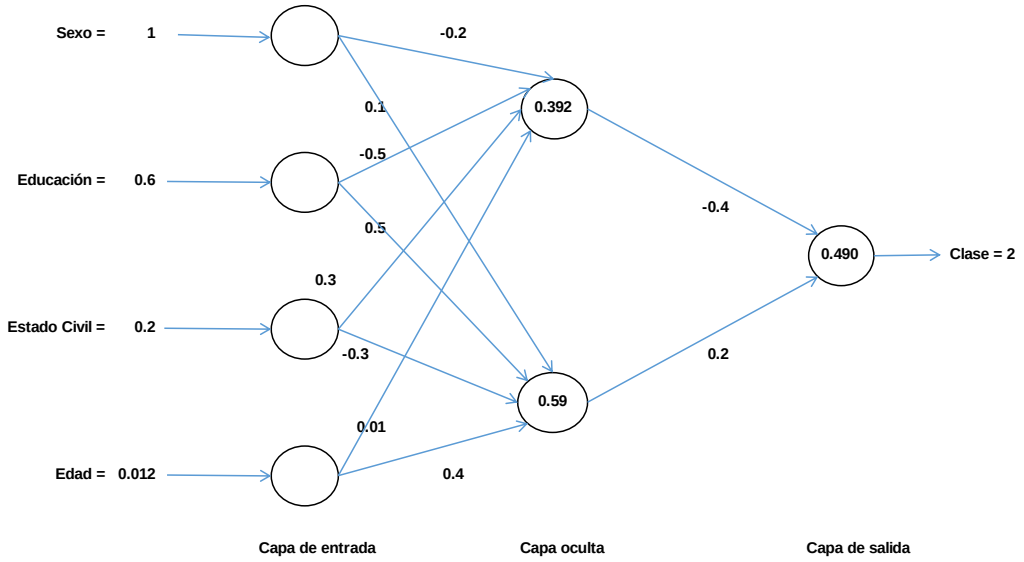


Figura 28. Perceptrón multicapa

El MLP, realiza el procesamiento en un solo sentido, en la figura 28, el MLP, procesa los datos de izquierda a derecha, y posee 3 capas: capa de entrada, que tiene 4 neuronas (una por cada atributo), están neuronas, solo son neuronas receptoras no realizan proceso; capa oculta, en el ejemplo posee 2 neuronas; y capa de salida 1 neurona (la cantidad de neuronas de la capa de salida debe coincidir con la cantidad de clases del problema. (Para una clasificación binaria es suficiente con una neurona en la salida), las dendritas tienen asociado pesos.

Existen diversos algoritmos de entrenamiento de MLP, entre ellos el algoritmo de retro propagación que utiliza descenso de gradiente.

Algoritmo de Retro-Propagación

A continuación, se muestra la lógica del algoritmo retro propagación:

1. *Inicializar pesos de las dendritas en valores pequeños.*

2. *Seleccionar el siguiente par de entrenamiento: entradas y salida deseada.*
3. *Calcular la salida del MLP.*
4. *Calcular el error cometido.*
5. *En base al error cometido, calcular el valor de descenso de gradiente.*
6. *Reajustar los pesos del MLP, de manera que se minimice el error cometido.*
7. *Repetir los pasos 2 al 6, para cada vector en el conjunto de entrenamiento, hasta que el error sea suficiente bajo.*

De forma general, los algoritmos de entrenamiento, de redes neuronales artificiales, siempre tienen como propósito, realizar reajustes de los pesos asociados a las dendritas, de forma que se minimice el error cometido (Faceli & Lorena, 2021)

1.1.21. Ensambles

Los ensambles, son modelos de aprendizaje automático, que combinan diferentes algoritmos de clasificación, emulando el comportamiento de expertos humanos en la tarea de predicción, o especialistas de dominio (Faceli & Lorena, 2021)

1.1.21.1. Voting

El método de votación, consiste en entrenar diferentes algoritmos de clasificación con el mismo conjunto de datos, y en la fase de predicción, se proporciona el nuevo dato a cada uno de los modelos entrenados, obteniendo una salida de cada algoritmo, la salida del ensamble voting, se decide, por votación de mayoría simple, como se aprecia en la figura 29 (Faceli & Lorena, 2021)

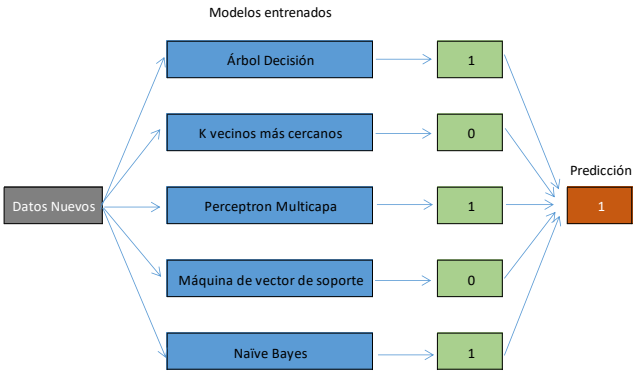


Figura 29. Ensamble Voting

1.1.21.2. Bagging

El ensamble bagging, separa los datos de entrenamiento en *n* muestras con reemplazo, donde *n* es la cantidad de clasificadores a utilizar. Cada clasificador es entrenado únicamente con una porción de los datos. En la fase de predicción, se proporciona el dato nuevo a cada clasificador entrenado, y se obtiene una salida por cada uno de ellos, la salida del ensamble bagging, se consigue por votación de mayoría simple, como muestra la figura 30 (Faceli & Lorena, 2021)

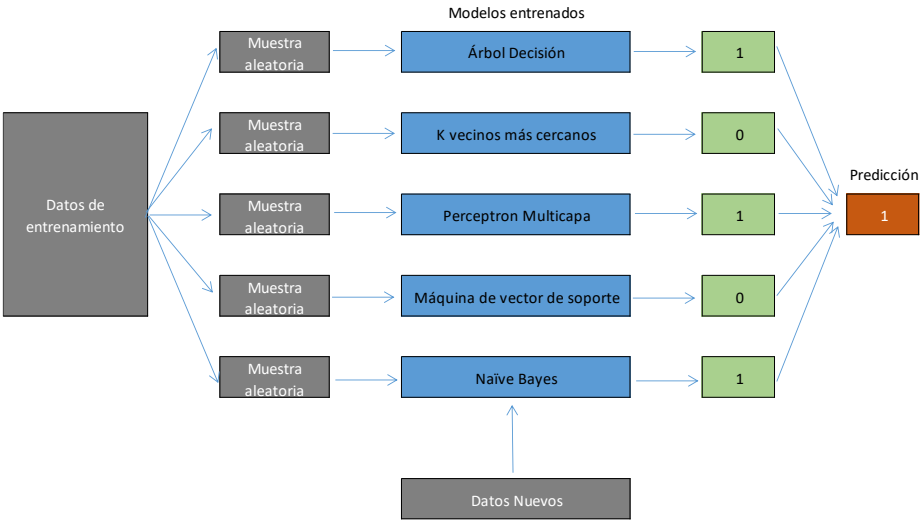


Figura 30. Ensamble Bagging

1.1.21.3. Boosting

El ensamble boosting, utiliza una secuencia de algoritmos de clasificación, y un conjunto de pesos asociados a cada uno de los datos de entrenamiento, como se aprecia en la figura 31 (Faceli & Lorena, 2021)

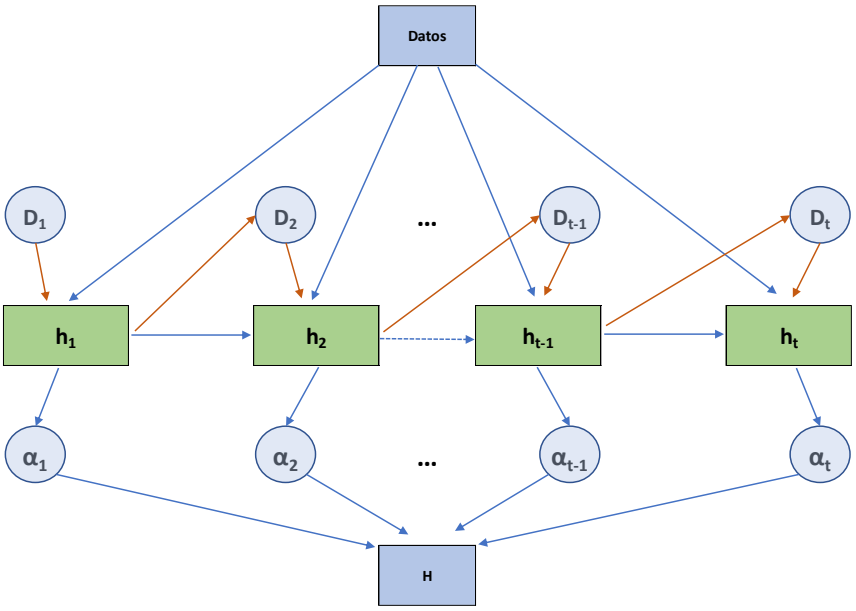


Figura 31. Ensamble Boosting

La figura 31, muestra la arquitectura del ensamble boosting, en ella *h1, h2, …, ht-1, ht*, corresponden a los clasificadores del tiempo *1, 2, …, t-1, t* respectivamente. Cada clasificador tiene asociado un conjunto de pesos, uno por cada dato de entrada. De cada clasificador se obtiene alfa. Alfa es obtenido a partir del error cometido por el clasificador y de aplicar una función exponencial. Cada uno de los clasificadores recibe el mismo conjunto de datos en entrenamiento.

Algoritmo de entrenamiento boosting

Sea el conjunto de *n* datos de entrenamiento $(x_1, y_1), \ldots, (x_n, y_n)$ en los que el atributo $x_i, \in X$, y la variable destino es $y_i \{-1, +1\}$, para $i = 1 \ldots n$.

Inicializamos los pesos $D_1(i) = \frac{1}{n}$, para $i = 1 \ldots n$.

Desde $t = 1$ hasta T repetir

- Hallar un clasificador $h^t(x)$, que minimice $P_{Dt}(h^t(x) \neq y) = Et$. Donde, Et, es el error cometido en las clasificaciones.

- Calcular alfa, con: $\alpha_t = \frac{1}{2} \ln \left(\frac{1-Et}{Et} \right)$

- Actualizar los pesos: $D_{t+1}(i) = D_t\, e^{(\alpha_t\, h_t(i) y_i)}$, donde D_{t+1} es el peso del tiempo $t+1$, D_t, es el peso del tiempo t, $h_t(i)$, es el clasificador del tiempo t, y_i, es la salida deseada.

- Normalizar D_t, es decir realizar los ajustes para que la sumatoria de probabilidades sea igual que uno.

Para realizar la predicción de un nuevo valor x, debe aplicar la siguiente función: $H^v(x) = signo(\sum_{t=1}^{T} \alpha_t\ \ h^t(x))$ (Faceli & Lorena, 2021)

1.1.21.4. Stacking

El ensamble stacking, posee un apilado de clasificadores y un clasificador final. La generalización apilada consiste, en apilar la salida de un clasificador individual y usar otro clasificador para calcular la predicción final. El apilamiento permite usar la fuerza de cada clasificador individual usando su salida como entrada de un clasificador final, la figura 32, muestra la arquitectura de un ensamble Stacking (Faceli & Lorena, 2021)

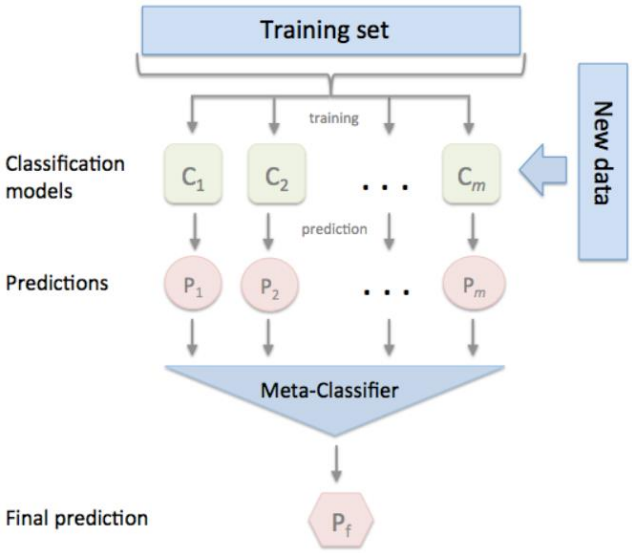


Figura 32. Ensamble Stacking

1.1.22. Test de ANOVA

El acrónimo ANOVA, proviene de **An**alysis **o**f **Va**riance, análisis de varianza, este test, basa su funcionamiento en el uso de varianza para determinar si los valores

medios de *k* variables son diferentes. El procedimiento compara la varianza de los valores medios de grupos. Para utilizar el test ANOVA, debe plantearse dos hipótesis, la hipótesis nula *Ho* y la hipótesis alterna *H1*.

Hipótesis nula y alternativa

Hipótesis nula (H0): las medias de *k* grupos son iguales.

$$H_0: \mu_1 = \mu_2 = \cdots = \mu_k$$

Hipótesis alternativa (H1): al menos dos valores medios de k grupos son distintas.

$$H_1: \mu_1 \neq \mu_2 = \cdots = \mu_k$$

Nivel de significancia

El nivel de significancia (alfa) debe ser seleccionado por la persona que aplique el test ANOVA, algunos valores típicos son:

α= 0.05 nivel de significancia (1-α=1-0.05=0.95 = 95% de nivel de confianza)

α= 0.10 nivel de significancia (1-α=1-0.10=0.90 = 90% de nivel de confianza)

α= 0.01 nivel de significancia (1-α=1-0.01=0.99 = 99% de nivel de confianza)

Regla de inferencia

El test de ANOVA, calcula el estadístico ***p-valor (p)***, luego se realiza un comparativo del p-valor con el nivel alfa de significancia (α), Si $\boldsymbol{p < \alpha}$, entonces rechazamos la hipótesis nula y nos quedamos con la alternativa. Quiere decir, que al menos hay dos valores de medias grupales que son diferentes. En caso contrario, no podemos rechazar la hipótesis nula y concluimos que no existen diferencias significativas entre los grupos evaluados.

Si $\boldsymbol{p < \alpha}$*, entones se rechaza H0 y se acepta H1.*(Huata Panca 2019)

1.1.23. Test de Tukey

El test de Tukey, es un test estadístico utilizado generalmente con el test ANOVA, El test Tukey se utiliza en experimentos que se tiene un número elevado de comparaciones. Tukey determina la diferencia por pares. Si estos intervalos incluyen al 0, entonces no se rechaza la hipótesis nula.

El test de Tukey permite determinar las diferencias entre las medias de las muestras, la comparación es "**Diferencia honestamente significativa**" (HSD), que se calcula mediante la siguiente expresión: $HSD = q_{\alpha,k,v} * \sqrt{\frac{MSe}{n}}$, donde, el $q_{\alpha,k,v}$, es un valor obtenido de tabla Tukey (α: significancia, k: cantidad grupos, v: grados de libertad de error obtenido por test de ANOVA) MSe: error cuadrado medio, n: tamaño de la muestra.

Regla de inferencia

Si se cumple que: $|\bar{y}_i - \bar{y}_J| > q_{\alpha,k,v}\sqrt{\frac{MSe}{n}}$, afirmamos que existe diferencia significativa entre las medias de los grupos y_i e y_j. caso contrario no existe diferencia significativa.

1.1.24. Test de Welch ANOVA y test Games Showell

El test ANOVA, es un test robusto para medir las diferencias significativas, cuando los datos tiene una distribución normal y por ende homogeneidad en la varianza; pero puede ser inapropiado cuando no existe homogeneidad en la varianza. Welch ANOVA y Games Showell, puede ser aplicable en este caso, en reemplazo de test ANOVA y el test de Tukey respectivamente (Hangcheng, 2015)

1.2. Antecedentes

Thottathyl *et al*. (2021) El problema de la investigación es, la predicción de cáncer de seno en muestras de microarray. Para ellos, los autores proponen, utiliza el algoritmo k-mens en versión paralela. Y realizan un comparativo de desempeño de velocidad, de los algoritmos: k vecinos más cercanos versión serial, máquina de vector de soporte versión serial y k-mens en versión serial. Los resultados obtenidos, evidencian, que, para diferentes tamaños de datos, el algoritmo k-mens versión paralela, tiene mejor desempeño en la velocidad de entrenamiento.

Khoirunnisa & Adiwijaya (2021) Los investigadores, manifiestan que los microarray es la tecnología popular, para la detección de cáncer. Y utilizan el algoritmo: Relevancia Máxima de Redundancia Mínima (mRMR) para reducción de dimensionalidad, y regresión logística para la clasificación. Logrando obtener una precisión promedio de 93.3% para muestras de cáncer de colon, pulmón, leucemia y ovario.

Fauzi *et al*. (2021) Los investigadores, afirman, que el cáncer, es la segunda causa de muerte a nivel mundial. Y con la intención de realizar la predicción de presencia de cáncer de forma temprana en muestras de microarray, y proponen utilizar análisis de componentes principales (PCA), para la reducción de dimensionalidad de los microarray; y una máquina de vector de soporte difuso (FSVM) para la predicción de la enfermedad. Los resultados obtenidos, muestran que utilizar FSVM con PCA obtiene una precisión de 96.92% para muestras leucemia.

Gumaei *et al*, (2021) manifiestan, que, el tema de las principales investigaciones, en las ciencias médicas informáticas, es el uso de aprendizaje automático en la detección de cáncer. A raíz de este problema, proponen utilizar, el método de Selección de Características de Correlación (CSF) para la reducción de dimensionalidad combinado con el modelo de Aprendizaje de Comité Aleatorio para la predicción de cáncer. Logrando obtener resultados de 95.098% de precisión para muestras de microarray con cáncer de próstata.

Ali *et al*. (2020) El artículo de investigación informa, que el cáncer de seno, es la enfermedad que mayor cantidad de muertes de mujeres ocasiona en el mundo. Y proponen utilizar los métodos, Boruta, y LASSO para reducción de dimensionalidad; y una máquina de vector de soporte (SVM) y regresión lineal (LR) como método de clasificación de muestras de cáncer en microarray. Los resultados obtenidos evidencian, que el algoritmo LASSO combinado con LR generan una precisión de 98.61%.

Palma-Ttito *et al*. (2020), Los investigadores proponen utilizar una máquina de vector de soporte de multiclase (SVMM), para la detección de catorce tipos cáncer: sangre, colón, seno, pulmón, piel, útero, sistema nervioso central, próstata, esófago, gástrico, hígado, páncreas, vejiga y hokings, cuyas muestras se encuentras en microarray. Y utilizan el filtro Inter cuartil, y filtro ANOVA para reducir dimensionalidad, obteniendo una precisión promedio de 87% en la detección de los cánceres mencionados.

Loey *et al*. (2020) La problemática planteada por los investigadores, es la detección de cáncer en muestras de microarray, y proponen utilizar el algoritmo de Ganancia de Información, para seleccionar las características importantes o genes predictores, luego plantean el uso del algoritmo optimizador lobo gris como un segundo método de reducción de dimensionalidad. Finalmente se utiliza una máquina de vector de soporte, como clasificador de cáncer de seno y de colon, obteniendo una precisión de 94.87% en

la detección de cáncer de seno, y 95.935% para el cáncer de colon.

Shah *et al*. (2020) Los investigadores propone utilizar Laplacian Score (LS) para reducción de dimensionalidad y una red neuronal convolucional modifica de 1D (1D-CNN) en la detección de diez tipos de cáncer expresados en microarray, entre ellos: cáncer de próstata, sistema nervioso central, colón, leucemia, tumor cerebral, Linfoma difuso de células B grandes, leucemia mieloide aguda, tumores de células azules y redondas. Obteniendo resultados de predicción entre 92% y 100% de precisión.

Hajieskandar *et al*. (2020) El artículo plantea la detección de cáncer de muestras de microarray, mediante el algoritmo de lobo gris para extraer características relevantes y una red densa (DNN) como aprendizaje profundo, para predicción de cáncer de estómago, pulmón, seno. Comparando con: SVM-lineal, KNN, Regresión lineal, Naïve Bayes, árbol de decisión, Xgboost, CNN, obteniendo valor de precisión alta: 99.37% cáncer de estómago, 99.91% cáncer de pulmón, y 99.89% cáncer de seno, frente al modelo comparado Xgboost con una exactitud de 98.33%, 95.29%, y 90.35% respectivamente, los algoritmos: : SVM-lineal, KNN, Regresión lineal, Naïve Bayes, árbol de decisión, CNN, realizan predicciones con precisión más baja.

Nurlaily *et al*. (2019) Los investigadores plantean, detectar cáncer de seno en muestras de microarray, y se proponen, usar el algoritmo colonia de hormigas (ACO) y algoritmo genético (AG) en la generación de datos, y para resolver el problema de datos desequilibrados usan la técnica de sobre muestreo de minoría sintética (SMOTE), este método, se encargara de generar datos sintéticos al azar de las clases minoritarias, luego mediante una máquina de vector de soporte (SVM), se realiza la predicción de la enfermedad, los resultados obtenidos muestra que el trío SVM-GA-SMOTE, tiene un accuracy de 92.5%, en microarray con cáncer de seno.

Fauzi *et al*. (2021) Los investigadores resuelven el problema de detección leucemia mieloide aguda (LMA) en microarray, que consistió en crear todas las combinaciones de 9 genes (genes provocan la enfermedad). Los resultados obtenidos muestran un área bajo la curva ROC (AUC) de 95.9%. para datos de LMA.

Astuti & Widi (2020) El autor manifiesta la importancia del diagnóstico de cáncer, y proponen utilizar el método de análisis de componentes principales (PCA), para reducción de dimensionalidad; y una red neuronal (ANN) para la predicción del cáncer

en microarray. Utilizando también un algoritmo genético (AG) para encontrar la mejor configuración de la red neuronal. Los resultados arrojan una precisión promedio de precisión de 83.33% 76.47%, 89.93% para cáncer de colon, próstata y pulmón respectivamente.

Sathya & Priya (2020) Los autores plantean resolver la predicción de cáncer de seno en microarray, mediante una optimización del Algoritmo Cardumen de Ballenas (MWO) para la selección de genes, en comparación al metaheurístico Optimización por Enjambre de Partículas (PSO). y en la fase de predicción usan una máquina de vector de soporte (SVM), k vecinos más cercanos y el clasificador Naïve Bayes. Los resultados obtenidos muestran que la SVM con MWO tiene una precisión más alta de 78.25%.

Loey *et al*. (2020) Resuelven el problema de detección de cáncer de seno en microarray, u utilizan un enfoque híbrido. Para reducción de dimensionalidad, el operador de Selección y Contracción Mínima Absoluta (LASSO), y algoritmo genético (AG); y un conjunto de clasificadores para detección de cáncer: regresión logística, máquina de vector de soporte, k vecinos más cercanos, bosque aleatorio, incrementando la exactitud de 89% a 96% al reducir la dimensionalidad.

Tavasoli *et al*. (2021) Los investigadores manifiestan, que los algoritmos híbridos son efectivos para resolver problemas de optimización y rara vez se han utilizado en selección de genes, la investigación propone utilizar el método de envoltura con cinco criterios (ROC, T-test, entropía, Wilcoxon, distancia de Bhattacharyya), en la selección de genes. Luego se plantea utilizar máquina de vector de soporte (SVM) con base gaussiana, polinomial y radial, comparado con rede neuronal artificial (ANN), k vecinos más cercanos (K-NN), en muestras de microarray con cáncer de linfoma, próstata, leucemia, colon y seno obteniendo una precisión de: 86.44%, 86.36%, 92.7%, 86.03% y 90.62% respectivamente.

Fogliatto *et al*. (2019) Los autores resuelven el problema de detección de cáncer de seno en datos de microarray, utilizando, análisis de componentes principales (PCA), y la distancia de Bhattacharyya para reducir dimensionalidad, y luego aplican, tres técnicas de predicción: k-vecinos más cercanos (K-NN), análisis discriminal lineal (LDA), red neuronal probabilística (PNN). K-NN obtuvo mejores resultados de 98.3% de exactitud en comparación a otros modelos.

Ali *et al*. (2020) Los autores manifiestan que el cáncer de mama, es el diagnóstico médico, más frecuente en mujeres, en el artículo proponen utilizar como método de selección de genes Boruta y LASSO, y mientras, que, en la clasificación se utiliza máquina de vector de soporte y regresión logística. Los resultados muestran que el mejor accuracy es 98.61%, utilizando regresión logística combinando con LASSO.

Shafi *et al*. (2020) Los autores resuelven la detección de cáncer de colon en datos de microarray, utilizando random forest para reducción de dimensionalidad, y utilizan un ensamble voting homogéneo, de árboles de decisión en la predicción. Y obtienen una precisión de 95%.

Rajasekaran & Sathyabama (2019) Los investigadores resuelven la detección de cáncer de seno en datos de microarray, y proponen usar Incrustación de Vecinos Estocásticos Kernelizados Gaussianos (GKNE-LGBC), para la selección de genes, luego se utiliza similitud de distancias como un segundo método de selección de genes. Para la fase de clasificación, se utiliza un ensamble de clasificadores débiles, y luego clasificadores fuertes. Obteniendo una exactitud de 96% de GKNE-LGBC.

Yawei & Yuan (2020) Los investigadores, proponen un nuevo método para clasificación de 6249 muestras de 14 tipos de cáncer, la propuesta plantea un método de ensamble voting con regresión logística, SVM, bosque aleatorio, XGBoost, y redes neuronales, mediante una estrategia de votación ponderada, obteniendo un accuracy de 71.46%.

CAPÍTULO II

PLANTEAMIENTO DEL PROBLEMA

2.1. Identificación del problema

¿Cuáles son los algoritmos de aprendizaje automático, que pueden detectar cáncer de mama en muestras de expresiones génicas de microarray?

¿Qué modelo de ensamble heterogéneo de aprendizaje automático, tendrá mejor precisión en la detección de cáncer de mama, en muestras de expresiones génicas de microarray?

2.2. Enunciados del problema

El cáncer es la segunda causa de muerte en el mundo, cada año aproximadamente mueren cerca de diez millones de habitantes, se estima que, de cada seis muertes, una se debe al cáncer. También manifiestan, que, aproximadamente el 70% de muertes por cáncer ocurre en países de bajos o medianos ingresos. Y afirman, que, el cáncer más común a nivel mundial, es el cáncer de mama, cerca de 2.26 millones de casos por año. Estas afirmaciones, evidencian la importancia, de crear nuevos mecanismos de diagnóstico de la enfermedad para que los personas tengan mayor accesibilidad y a menores costos (Organización Mundial de la Salud, 2021)

El Instituto Nacional de Enfermedades Neoplásica de Perú, manifiesta que muchas personas pueden tener una vida plena, si se detecta la enfermedad en las fases iniciales, y si han recibido un tratamiento contra el cáncer, por ello, la importancia de la detección temprana de cáncer, para disminuir las muertes por esta enfermedad (Instituto Nacional de Enfermedades Neoplásicas, 2023)

En Knife (2023), manifiestan, que, en enfermedades de alto riego, como el cáncer, los pacientes generalmente solicitan una segunda opinión, con el propósito, de tener la

certeza del diagnostico primigenio, este hecho es corroborado por especialistas, quienes, además agregan, que está práctica tiene un gran potencial. Este mismo hecho es manifestado por American Cancer Society (2023). Según Mira *et al*. (2006), consideran que la segunda opinión es un derecho de toda persona, según la investigación, los expertos mencionan que una segunda opinión aumenta la calidad de asistencia sanitaria, la confianza y credibilidad. Nacional Instituto del Cáncer (2023) afirma que, en el campo de la medicina, una segunda opinión ofrece al paciente una opinión que confirma o cuestiona el diagnostico original, y el plan de tratamiento. Según Sanchez *et al*. (2021), la segunda opinión en las encuestas realizada con diferentes estratos sociales de pacientes en Francia, son solicitadas en caso de enfermedades complejas y en casos de riesgo, La investigación, no encontró que esta decisión dependiera de sexo, edad, región de procedencia o profesión, y una segunda opinión mejora la calidad de atención.

Weigl *et al*. (2021), manifiestan que, en Alemania, los pacientes tienen derecho a una segunda opinión, en caso de cirugías, y que el seguro social debe hacer un reembolso. Diferentes investigaciones e instituciones médicas del cáncer, manifiestan que una segunda opinión mejora la calidad de atención y confirma o niega el primer diagnóstico, motivo por el cual una problemática a ser considerada, es dar a los pacientes la posibilidad de una segunda opinión

Mezzasoma *et al*. (2012), Los microarrays de ADN, son una tecnología de la biología molecular que tiene diversas aplicaciones; permite cuantificar los niveles de expresiones génicas, de los genes en estudio, una de las posibles aplicaciones con los niveles de expresión génica, es la detección de mutaciones en genes específicos para un posterior diagnóstico de la enfermedad. Manifestar que el cáncer es provocado por mutaciones genéticas, y por ello, son una tecnología muy adecuada, para el diagnóstico de esta enfermedad.

Mezzasoma *et al*. (2012), El microarray de ADN, es una colección de manchas adheridas a la superficie sólida, cada mancha de ADN, contiene miles de copias de una secuencia de ADN específico (secuencias que se desean estudiar). Estas secuencias, suelen ser una sección corta de un gen, la que se utilizan para, hibridar con muestras de ADNc (ADN complementario). La hibridación permite cuantificar la abundancia de transcripciones de las muestras. La intensidad de las manchas determina la cantidad de secuencias que son complementarias una de la otra, la tonalidad manifiesta la diferencia entre la muestra

control (adheridas a la superficie). Los microarray de ADN, poseen manchas, y la intensidad de la mancha puede ser evaluada con mayor exactitud por método informáticos.

De Guia *et al*. (2020), Existen diferentes modelos de machine learning, en la detección de cáncer a partir de microarray, algunos utilizan modelos predictivos y otros descriptivos, los diferentes autores presentan modelos lógicos, y las propuestas siempre pretenden mejorar el proceso de identificación y clasificación. De lo manifestado podemos inferir, que otros modelos lógicos de machine learning, permiten diagnosticar de manera más certera el cáncer.

Hengpraprohm & Jungjit (1988), Una de las problemáticas, de los microarray, es la alta dimensionalidad, y enfatiza que es necesario la selección de genes predictores de la enfermedad. Siendo muy necesario propuestas de selección de genes.

Por lo manifestado en párrafos anteriores, se pretende utilizar, varios modelos de machine learning para la detección de cáncer de mama, de forma que cada modelo arroje un diagnóstico, el cual corresponderá a una segunda opinión. Y la decisión final, será una estrategia consensuada, emulando el diagnóstico de expertos humanos.

Los expertos médicos, manifiestan que la detección temprana de diagnóstico de cáncer, permite que el tratamiento, que se aplica a los pacientes, permita un mejor control de la enfermedad y en la mayoría de casos evitar las muertes. También manifiestan, que los pacientes que hayan recibido un tratamiento al cáncer en una fase inicial, en muchos casos tienen una vida bastante normal. Lo manifestado por especialistas, da ha, conocer la gran importancia de la detección de cáncer en etapas iniciales.

Una segunda opinión, en caso de enfermedades de riesgo de vida, como el cáncer, proveen al paciente una mejor calidad en el diagnóstico de la enfermedad, corrigiendo falsos positivos o falsos negativos en caso de existir, y también, pueden alterar los tratamientos, por otros más apropiados a la enfermedad.

Proveer al paciente, de mayor cantidad, de métodos de diagnóstico de cáncer, incita, que los costos de diagnóstico tiendan a disminuir, y sean más accesibles.

El uso de tecnologías más recientes, como los microarray, permite, a los especialistas, tener mayor certeza del diagnóstico realizado.

Por los manifiestos anteriores el presente trabajo tiene como propósito determinar la presencia o ausencia de cáncer de mama, en base al nivel de expresión génica de microarray a través de un ensamble de modelos heterogéneos de machine learning.

2.3. Justificación

Los expertos médicos, manifiestan que la detección temprana de diagnóstico de cáncer, permite que el tratamiento, que se aplica a los pacientes, permita un mejor control de la enfermedad y en la mayoría de casos evita las muertes. También manifiestan, que los pacientes que hayan recibido un tratamiento al cáncer en una fase inicial, en muchos casos tienen una vida bastante normal. Lo manifestado por especialistas, da ha, conocer la gran importancia de la detección de cáncer en etapas iniciales.

Una segunda opinión, en caso de enfermedades de riesgo de vida, como el cáncer, proveen al paciente una mejor calidad en el diagnóstico de la enfermedad, corrigiendo falsos positivos o falsos negativos en caso de existir, y también, pueden alterar los tratamientos, por otros más apropiados a la enfermedad. Siendo necesario en este tipo de enfermedades una segunda opinión.

Proveer al paciente, de mayor cantidad, de métodos de diagnóstico de cáncer, incita, que los costos de diagnóstico tiendan a disminuir, y sean más accesibles.

El uso de tecnologías más recientes, como los microarray, permite, a los especialistas, tener mayor certeza del diagnóstico realizado.

2.4. Objetivos

2.4.1. Objetivo general

Diseñar e implementar un ensamble heterogéneo de aprendizaje automático para la predicción de presencia o ausencia de cáncer de mama, en base al nivel de expresión génica de microarray.

2.4.2. Objetivos específicos

- Identificar y seleccionar muestras de microarray de oligonucleótidos con expresión génica de presencia de cáncer de mama y muestras de control, de repositorios públicos de información biotecnológica.
- Reconocer los algoritmos de aprendizaje automático que mejor predicción de cáncer de mama obtienen en muestras de expresiones génicas de microarray.
- Identificar el modelo de ensamble heterogéneo que mejor predicción de cáncer de mama obtiene en muestras expresiones génicas de microarray.

2.5. Hipótesis

2.5.1. Hipótesis general

Un ensamble de modelos heterogéneos de machine learning, puede mejorar la predicción de presencia o ausencia de cáncer de mama, mediante comparaciones de niveles de expresiones génicas en microarray.

2.5.2. Hipótesis específicas

Hipótesis H1

Los repositorios públicos de información biotecnológica, poseen muestras de microarray de expresión génica con presencia de cáncer de mama y muestras de control.

Hipótesis H2

Varios algoritmos de aprendizaje automático tienen mejor predicción de la presencia o ausencia de cáncer de mama, en muestras expresiones génicas de microarray.

Hipótesis H3

Un modelo de ensamble heterogéneo de aprendizaje automático tiene una mejor predicción de presencia o ausencia de cáncer de mama, en muestras de expresiones génicas de microarray.

CAPÍTULO III

MATERIALES Y MÉTODOS

3.1. Lugar de estudio

En el presente proyecto, no se tiene interacción directa con personas u otros sujetos, ya que el objetivo es diseñar e implementar un prototipo de ensamble de modelos heterogéneo de aprendizaje automático. Y los datos a utilizar, son un conjunto de microarrays con información de expresiones génicas, obtenidos de repositorios biológicos: Centro Nacional de Información de Biotecnología de Estados Unidos de Norte América y del Instituto de Bioinformática de Europa. El prototipo generalizará los datos y podrá realizar predicciones de muestras que provengan de cualquier persona, independiente de la ubicación geográfica.

3.2. Población

La población, consta de las muestras de expresiones génicas de microarray, existente en los repositorios públicos: del Centro Nacional de Información Biotecnología de Estados Unidos de Norte América y del Instituto de Bioinformática de Europa.

3.3. Muestra

Para el muestreo de datos se aplica técnicas de análisis y exploración de datos, para así seleccionar muestras de expresiones génicas de microarray con presencia de cáncer de mama, y muestras de control, pertenecientes a homo sapiens, los microarray encontrados deberán tener mismas características: protocolo (marca y modelo de microarray), diseño (que posean mismas sondas), formato de grabación. Entre otros.

3.4. Método de investigación

La presente investigación es de naturaleza cuantitativa, y del tipo aplicada, ya que tiene por

objeto resolver un problema específico, enfocado en la búsqueda del conocimiento para su aplicación.

La metodología empleada, en el diseño e implementación del modelo de ensamble heterogéneo, que es el objetivo de la presente investigación, utiliza la metodología de aprendizaje automático KDD (descubrimiento de conocimiento en bases de datos).

3.5. Descripción detallada de métodos por objetivos específicos

El método utilizado para la recolección de datos consta de:

a) Diseño de muestreo

Objetivo Específico 1: Para la obtención de muestras de microarray de oligonucleótidos, se realiza un análisis y exploración de los repositorios de Centro Nacional de Información Biotecnológica de EE.UU (NCBI) y del Instituto Europeo de Bioinformática (EMBI-EBI), con el fin de encontrar microarray, que contenga expresiones génicas de células cancerosas de seno y expresiones génicas de células de mama sanas.

Objetivo Específico 2: Para identificar los algoritmos de aprendizaje automático que mejores predicciones tengan en el diagnóstico de cáncer de mama en expresiones génicas, es necesario identificar los diferentes algoritmos de aprendizaje automático, y comprender como representan la información los microarray, se necesitan de textos bibliográficos y artículos de investigación.

Objetivo Específico 3: Para identificar el modelo de ensamble heterogéneo, que realice el mejor diagnóstico de cáncer de mama en expresiones génicas. En necesario conocer sobre el principio de funcionamiento de los ensambles, para así diseñar e implementar, Para ello es necesario contar con textos bibliográficos y artículos de investigación, relacionados al tema.

b) Revisión detallada del uso de materiales, equipos, insumos, entre otros.

Objetivo Específico 1: Para ingresar a los repositorios de NCBI y EMBI-EBI, se accede mediante conexión a internet. Luego se utilizan aplicaciones que permitan realizar tareas de análisis y exploración de datos, para identifica microarray con expresiones génicas de cáncer de mama.

Objetivo Específico 2: Los algoritmos de aprendizaje automático, deben ser ***identificados, ejemplificados*** *y* ***usados*** en la detección de cáncer de mama, para cumplir dicho propósito se realiza lecturas, se ejemplifica mediante una dataset pequeño, y luego son utilizados a través de lenguaje de programación Python en el Cloud de Google, para finalmente comprobar ¿cuál o cuáles de los modelos tiene mejor predicción de cáncer de mama de expresiones génicas?

Objetivo Específico 3: Los modelos de ensamble heterogéneos de aprendizaje automático, deben ser **identificados**, **diseñados**, y **usados** en la detección de cáncer de mama. Para cumplir dicho propósito se realiza lecturas, luego se diseñan arquitecturas de ensamble heterogéneas (combinando distintos algoritmos de aprendizaje automático), para utilizar, se hace uso, de lenguaje de programación Python en el Cloud de Google, para finalmente comprobar ¿qué modelo de ensamble heterogéneo realiza mejor predicción de cáncer de mama de expresiones génicas?

c) Descripción de variables a ser analizados en el objetivo específico.

Objetivo Específico 1: Los microarray, que utilizamos en la investigación, son microarray de oligonucleótidos (ADNc o ARNm) con expresiones génicas. Cuyas variables a ser analizadas son:

Variable independiente: Nivel de expresiones génicas, es decir: intensidad de fluorescencia provocado por la hibridación (combinación de cadenas monocatenarias de ADNc o ARNm por atracción bioquímica) de genes de ADN.

Variable dependiente: Presencia o ausencia de cáncer.

Objetivo Específico 2: Los algoritmos de aprendizaje automático a ser utilizadas en la investigación son: k vecinos más cercanos (KNN), clasificador Naïve Bayes (NB), árbol de decisión (AD), red neuronal artificial del tipo perceptrón multicapa (MLP), máquina de vector de soporte para clasificación (SVC). Cuyas variables a ser analizadas son:

Variable independiente: Configuración específica de hiper parámetros de algoritmos de aprendizaje automático.

Variable dependiente: Tasa de aciertos en la predicción de cáncer de mama.

Objetivo Específico 3: Los modelos de ensamble heterogéneo de aprendizaje automático a ser utilizadas en la investigación son: voting, bagging, boosting y stacking, los cuales se diseñarán, implementar y utilizan en la predicción de cáncer de mama. Las variables a ser analizadas son:

Variable independiente: Arquitectura de los modelos de ensamble heterogéneo, y valores de hiper parámetros de los algoritmos de clasificación utilizados en los ensambles.

Variable dependiente: Tasa de aciertos en la predicción de cáncer de mama.

d) Aplicación de estadística inferencial.

Objetivo Específico 1: Se realiza una contrastación por observación.

Objetivo Específico 2: Se hará uso del test de ANOVA y Welch ANOVA.

Objetivo Específico 3: Se hará uso del test de ANOVA y Welch ANOVA.

CAPÍTULO IV

RESULTADOS Y DISCUSIÓN

En la investigación se hizo uso, de dos dataset: El primero, matrices de expresiones génicas de ARN (***E-GEOD-73002***), y el segundo, matrices de expresiones génicas de ADN (***E-GEOD-42568***), el motivo de realizar dos experimentos, es para verificar el desempeño de los modelos propuestos.

4.1. Objetivo específico 1

a) Interpretación de resultados

Primer experimento (*E-GEOD73002*)

	MIMAT0000062	MIMAT0000063	MIMAT0000064	MIMAT0000065	ESTADO
0	0.227433	0.227433	0.227433	0.227433	normal
1	0.355896	0.355896	0.355896	0.874377	normal
2	2.189441	2.189441	2.189441	2.189441	normal
3	0.666966	0.666966	0.666966	0.666966	normal
4	1.069203	1.069203	1.069203	1.441521	normal

Figura 33. Expresiones de ARN (E-GEOD-73002)

La figura 33 muestra, una porción muy pequeña de datos de expresión génicas de ARN, que corresponde a microARN (miARN). Los microARN, son moléculas de ARN, de entre 21 a 25 nucleótidos, los cuales son útiles para la detectar cáncer en fase iniciales. El dataset consta de: 1280 muestras de pacientes con cáncer de mama, 2686 muestras de personas sanas, 54 muestras de tumores benignos, y 93 muestras con cáncer de próstata, haciendo un total de 4113 microarray, dispuestos cada uno de ellos en un archivo y con un total de 2542 sondas o atributos.

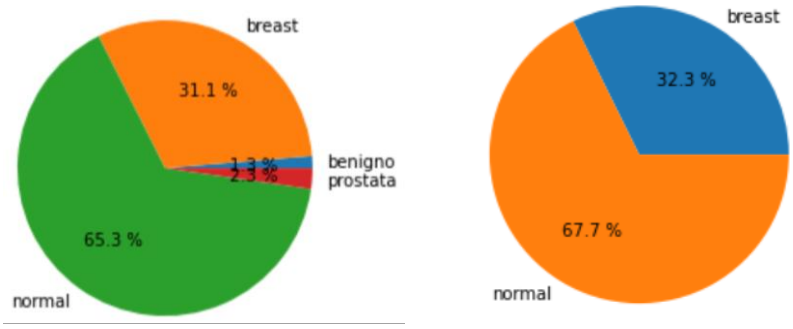


Figura 34. Distribución de clases, a la izquierda datos originales, a la derecha datos sin las clases minoritarias

La figura 34, muestra la proporción de clases del dataset original y del dataset final a utilizar en el presente experimento.

Es necesario, aclarar que, para conseguir el dataset, se realizó un conjunto de procesos laboriosos de análisis, exploración y preparación de datos. Tareas que implicaron hacer revisión de textos de medicina, oncología, biología molecular, entre otros.

Segundo Experimentos (*E-GEOD-42568*)

	1007_s_at	1053_at	117_at	121_at	1255_g_at	1294_at	1316_at	1320_at	1405_i_at	1431_at	ESTADO
0	7.018523	5.421016	4.489153	4.009733	2.312812	6.301683	5.685626	2.312812	9.296664	2.405673	Cancer
1	3.854983	5.464200	3.562947	3.574706	2.312812	7.468689	5.522856	2.312812	9.273398	2.312812	Cancer
2	7.113492	4.299268	3.939654	3.574706	2.312812	5.937422	7.849103	2.312812	8.639617	2.312812	Cancer
3	6.394748	4.900223	4.131843	3.574706	2.312812	4.823558	7.214731	2.312812	5.915115	2.312812	Cancer
4	7.460920	7.203254	6.046027	3.574706	2.312812	6.300663	5.050737	2.312812	9.360997	2.312812	Cancer

Figura 35. Expresiones génicas de ADN (E-GEOD-24568)

La figura 35 muestra, una porción muy pequeña de datos de expresión génicas de ADN, extraída por biopsias de células de cáncer de mama (extraídas antes del tratamiento de la enfermedad), el dataset consta de: 104 muestras con cáncer de mama, y 17 biopsias de mama normales, cuya proporción se aprecia en la figura 36, haciendo un total de 121 muestras.

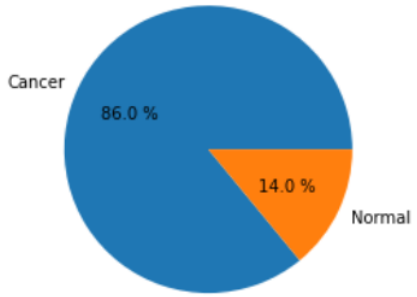


Figura 36. Distribución de clases de datos de expresiones génicas (E-GEOD-24568)

Figura 37. Gráfico en barras, que representa la puntuación asignada a cada sonda como variable predictora, habiendo utilizado el método Ganancia de Información y ANOVA

Cada muestra del dataset, consta de 54676 sondas, siendo bastante alta la dimensionalidad, motivo por el cual se aplicó reducción de dimensionalidad, utilizando el método de Ganancia de Información mediante entropía, método que permitió detectar las sondas de mayor importancia en la predicción de cáncer. Para la puntuación de ganancia de información, se utilizó el test de ANOVA, para detectar sondas que presentan una baja variabilidad, motivo que permite detectar sondas "redundantes", y así poder eliminarlas. La figura 37 muestra la importancia de cada una de las 54676 sondas, esta importancia se expresa por la altura de la barra, estos resultados permiten seleccionar las sondas que mayor importancia tiene en la predicción de cáncer de mama.

Se elimino las sondas cuya puntuación es menor al 10% de la máxima puntuación, quedándonos con un matriz de expresiones génicas de 121 muestras y 1515 atributos (sondas).

4.2. Objetivo específico 2

Primer Experimentos (E-GEOD-73002)

a) Interpretar los resultados

Tabla 2

Valor de hiper parámetros que optimizan los algoritmos de aprendizaje automático

Algoritmo	Hiper parametros	Valores	Exactitud
KNN	N vecinos	4	98.49%
	Pesos	uniform	
	Algoritmo	kd_tree	
AD	Criterio	gini	98.49%
	Divisor	best	
	Max prof árbol	20	
NB	No posee hiper parametros		90.62%
SVC	Kernel	linear	99.66%
	Costo	0.1	
MLP	Capas ocultas	(100, 50, 10)	98.49%
	Función activación	relu	
	Coef aprendizaje	invscaling	
	Momentum	0.5	

La tabla 2, muestra para cada algoritmo de aprendizaje automático utilizado en la presente investigación, el valor de hiper parámetros, encontrados mediante búsqueda aleatoria en matrices. Valores que optimizan el clasificador en el diagnóstico de cáncer de mama en expresiones génicas. Se aprecia en la última columna la exactitud conseguida con la configuración de hiper parámetros.

	KNN	SVC	MLP	AD	NB
CV = 1	0.994444	1.000000	0.961111	0.988889	0.961111
CV = 2	0.988827	0.994413	0.374302	0.983240	0.949721
CV = 3	0.983240	1.000000	0.949721	0.977654	0.927374
CV = 4	0.977654	0.983240	0.927374	0.983240	0.921788
CV = 5	0.994413	1.000000	0.994413	0.983240	0.916201
CV = 6	0.983240	1.000000	0.983240	0.983240	0.910615
CV = 7	0.994413	1.000000	0.921788	0.994413	0.910615
CV = 8	0.977654	1.000000	0.994413	0.994413	0.921788
CV = 9	0.977654	0.988827	0.983240	0.977654	0.910615
CV = 10	0.977654	1.000000	0.972067	0.988827	0.905028

Figura 38. Valores de exactitud de predicción de cada algoritmo aplicando validación cruzada, k=10

La figura 38, muestra para cada algoritmo de aprendizaje automático, el valor de exactitud, obtenido, aplicando validación cruzada (*k=10*), valores que son utilizados para calificar el desempeño de cada uno de los clasificadores.

Una vez obtenido la exactitud de cada algoritmo, debemos determinar el desempeño de cada uno de ellos y así identificar los mejores clasificadores para predicción de cáncer de mama. En la identificación de mejores clasificadores, se utiliza el test ANOVA, y para ello es necesario garantizar que los datos siguen una distribución normal, en caso que los datos no posean una distribución normal, se debe utilizar el test alternativo Welch ANOVA.

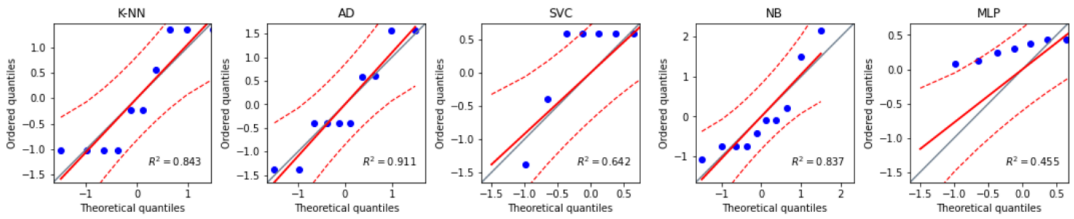


Figura 39: Gráficos cuartil-cuartil, que muestras si los datos de predicción de la figura 38 siguen una distribución normal

La figura 39, muestra la gráfica cuartil cuartil, que permite determinar si los datos de la figura 38 tiene o no distribución normal. Revisando las gráficas obtenidas, se aprecia que únicamente los datos de AD siguen una distribución normal, lo cual es un indicador que el test a utilizar para identificar los mejores clasificadores es, el test Welch ANOVA y el test Games Showell.

	Source	ddof1	ddof2	F	p-unc	np2
0	Algoritmo	4	21.686072	34.208966	4.445863e-09	0.174161

Figura 40. Estadísticos del test de Welch ANOVA

La figura 40, muestra los resultados de aplicar el test Welch ANOVA, y continuando con el procedimiento procedemos a plantear la hipótesis nula y la hipótesis alterna:

Ho: No existe diferencias significativas entre las medias de exactitud de predicción, de los algoritmos de clasificación, en la detección de cáncer de seno.

H1: Al menos existe una diferencia significativa, con las medias de exactitud de predicción, de dos algoritmos de clasificación, en la detección de cáncer de seno.

Según la figura 40 el **p-valor = 0.000000004**, expresado en el parámetro ***p-unc***. Y para un valor de significancia de alfa = 0.01 o un grado de confianza del 99% afirmamos inferimos que:

*Se **rechaza** la hipótesis **Ho** y se **acepta** la hipótesis alterna **H_1**, ya que **p-valor = 0.000000004 < alfa = 0.01**. esto quiere decir que al menos existe un par de medias de exactitud de predicción de cáncer de seno, con diferencia significativa. Y es necesario aplicar el test de Games Showell, para determinar el mejor clasificador.*

	A	B	mean(A)	mean(B)	diff	se	T	df	pval	hedges
0	AD	KNN	0.985481	0.984919	0.000562	0.003033	0.185199	17.204442	0.999708	0.079324
1	AD	MLP	0.985481	0.906167	0.079314	0.059686	1.328865	9.018265	0.682060	0.569176
2	AD	NB	0.985481	0.923485	0.061996	0.006092	10.175852	10.917668	0.000005	4.358493
3	AD	SVC	0.985481	0.996648	-0.011167	0.002687	-4.156597	17.999992	0.004711	-1.780342
4	KNN	MLP	0.984919	0.906167	0.078752	0.059702	1.319086	9.028272	0.687455	0.564987
5	KNN	NB	0.984919	0.923485	0.061434	0.006253	9.825182	11.921520	0.000004	4.208295
6	KNN	SVC	0.984919	0.996648	-0.011729	0.003033	-3.867655	17.200080	0.009286	-1.656583
7	MLP	NB	0.906167	0.923485	-0.017318	0.059936	-0.288951	9.169459	0.998180	-0.123763
8	MLP	SVC	0.906167	0.996648	-0.090481	0.059686	-1.515962	9.018241	0.577854	-0.649313
9	NB	SVC	0.923485	0.996648	-0.073163	0.006092	-12.009533	10.915249	0.000001	-5.143890

Figura 41. Resultados de aplicar el test de Games Showell, para determinar ¿Cuál es el menor clasificador?

La figura 41, muestra los resultados del test Games Showell, en ella se aprecia las diferencias realmente significativas en la columna ***diff***, entre cada par de algoritmos de clasificación, a partir de estos resultados obtenemos la tabla 3, resumen de test Games Showell.

Tabla 3

Resumen del test de Games Showell, donde se identifica las diferencias significativas.

Clasificador A	Clasificador B	$\lvert\bar{A} - \bar{B}\rvert$	Game Howell (GH)	Diferencia Realmente Significativa	Motivo
AD	KNN	0.00056	0.999710	No	| media(A) - media(B) | = 0.00056 < GH = 0.99971
AD	MLP	0.07931	0.682060	No	| media(A) - media(B) | = 0.07931 < GH = 0.68206
AD	NB	0.06199	0.000005	Si	| media(A) - media(B) | = 0.06199 > GH = 0.000005
AD	SVC	-0.01116	0.004710	No	| media(A) - media(B) | = -0.01116 < GH = 0.00471
KNN	MLP	0.07875	0.687450	No	| media(A) - media(B) | = 0.07875 < GH = 0.68745
KNN	NB	0.06143	0.000004	Si	| media(A) - media(B) | = 0.06143 > GH = 0.000004
KNN	SVC	-0.01172	0.009280	No	| media(A) - media(B) | = -0.01172 < GH = 0.00928
MLP	NB	-0.01731	0.998180	No	| media(A) - media(B) | = -0.01731 < GH = 0.99818
MLP	SVC	-0.09048	0.577850	No	| media(A) - media(B) | = -0.09048 < GH = 0.57785
NB	SVC	-0.07316	0.577850	No	| media(A) - media(B) | = -0.07316 < GH = 0.57785

Según la tabla 3, existen diferencias realmente significativas entre AD y NB, de igual modo entre KNN y NB, y apreciando también, la tabla 2, podemos concluir en: Los algoritmos SVC, KNN, MLP, son los mejores clasificadores, por no existir diferencias realmente significativas entre ellos. El siguiente mejor clasificador es AD. Y el peor clasificador es NB.

b) Discusión con otros autores

La investigación presentada por Tamura *et al*. (2016), muestran, un conjunto de pruebas realizadas, entre ellas están la sensibilidad, exhaustividad y precisión de algún modelo de clasificación, el cual no se detalla, pero si, se da a conocer la exactitud de predicción, el que llega a 97.1%, en el presente trabajo se determinó que los mejores modelos son SVC, KNN y MLP, con una exactitud promedio de 98.86%, superando en un pequeño margen.

Segundo Experimentos (*E-GEOD-42568*)

a) Interpretar los resultados

Tabla 4

Valor de hiper parámetros que optimizan los algoritmos de aprendizaje automático

Algoritmo	Hiper parametro	Valores	Exactitud
KNN	N vecinos	3	100.00%
	Pesos	distance	
	Algoritmo	ball_tree	
AD	Criterio	entropy	94.59%
	Divisor	random	
	Max prof árbol	50	
NB	No posee hiper parametros		86.49%
SVC	Kernel	poly	100.00%
	Costo	0.5	
MLP	Capas ocultas	(100, 50, 10)	97.30%
	Función activación	tanh	
	Coef aprendizaje	invscaling	
	Momentum	0.9	

La tabla 4, muestra para cada algoritmo de aprendizaje automático, el valor de hiper parámetros, encontrados mediante búsqueda aleatoria en matrices. Valores que optimizan el clasificador en el diagnóstico de cáncer de mama en expresiones génicas. Se aprecia en la última columna la exactitud conseguida con la configuración de hiper parámetros.

	SVC	MLP	KNN	NB	AD
CV = 1	1.000000	1.000000	1.000000	0.888889	1.000000
CV = 2	1.000000	0.888889	1.000000	0.888889	1.000000
CV = 3	1.000000	0.888889	1.000000	0.777778	0.777778
CV = 4	0.888889	0.888889	0.888889	0.777778	0.888889
CV = 5	1.000000	0.875000	1.000000	0.750000	1.000000
CV = 6	1.000000	1.000000	1.000000	0.875000	0.875000
CV = 7	1.000000	0.875000	1.000000	0.875000	0.750000
CV = 8	1.000000	1.000000	1.000000	0.875000	1.000000
CV = 9	1.000000	1.000000	1.000000	0.875000	0.875000
CV = 10	1.000000	1.000000	0.875000	0.875000	0.875000

Figura 42. Valores de exactitud de predicción de cada algoritmo aplicando validación cruzada, k=10

La figura 42, muestra para cada algoritmo de aprendizaje automático, el valor de exactitud, habiendo aplicado validación cruzada (*k=10*), valores que son, utilizados para

determinar, que clasificadores tienen mejor desempeño en la predicción de cáncer de mama.

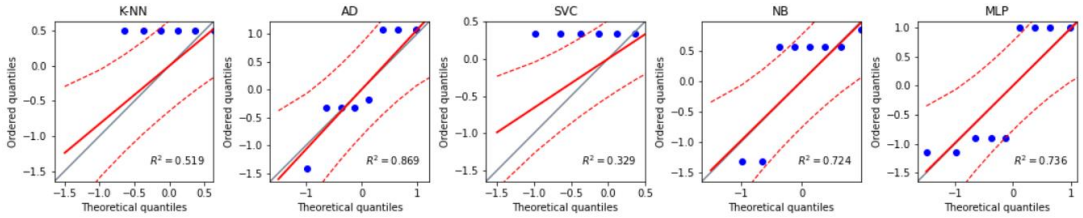


Figura 43. Gráficos cuartil-cuartil, que muestras si los datos de predicción de la figura 37 siguen una distribución normal

La figura 43, muestra para cada algoritmo de aprendizaje automático, si los datos de exactitud de la figura 42, siguen una distribución normal. Y podemos apreciar que, AD es el único que tiene una distribución normal, por lo que debemos aplicar el test Welch ANOVA y el test Games Showell.

	Source	ddof1	ddof2	F	p-unc	np2
0	Algoritmo	4	21.998362	12.510693	0.000019	0.437648

Figura 44. Estadísticos del test de Welch ANOVA

La figura 44, muestra los resultados de aplicar el test Welch ANOVA, y para continuar el procedimiento, planteamos la hipótesis nula y la hipótesis alterna:

Ho: No existe diferencias significativas entre las medias de exactitud de predicción, de los algoritmos de clasificación, en la detección de cáncer de seno.

H1: Al menos existe una diferencia significativa, con las medias de exactitud de predicción, de dos algoritmos de clasificación, en la detección de cáncer de seno.

Según la figura 44 el **p-valor = 0.000019**, expresado en el parámetro ***p-unc***. Y para un valor de significancia de alfa = 0.01 o un grado de confianza del 99% inferimos que:

*Se **rechaza** la hipótesis **Ho** y se **acepta** la hipótesis alterna **H_1**, ya que **p-valor = 0.000019 < alfa = 0.01**. esto quiere decir que al menos existe un par de medias de exactitud de predicción de cáncer de seno, con diferencia significativa. Y es necesario aplicar el test de Games Showell, para determinar el mejor clasificador.*

	A	B	mean(A)	mean(B)	diff	se	T	df	pval	hedges
0	AD	KNN	0.904167	0.976389	-0.072222	0.033609	-2.148915	13.709895	0.255181	-0.920417
1	AD	MLP	0.904167	0.941667	-0.037500	0.035516	-1.055873	15.555352	0.825651	-0.452249
2	AD	NB	0.904167	0.845833	0.058333	0.034266	1.702392	14.397897	0.462494	0.729164
3	AD	SVC	0.904167	0.988889	-0.084722	0.031688	-2.673599	11.474614	0.120104	-1.145149
4	KNN	MLP	0.976389	0.941667	0.034722	0.025090	1.383915	17.243902	0.645302	0.592754
5	KNN	NB	0.976389	0.845833	0.130556	0.023287	5.606462	17.879154	0.000225	2.401344
6	KNN	SVC	0.976389	0.988889	-0.012500	0.019295	-0.647834	16.166321	0.964601	-0.277479
7	MLP	NB	0.941667	0.845833	0.095833	0.025963	3.691136	17.703512	0.012933	1.580977
8	MLP	SVC	0.941667	0.988889	-0.047222	0.022453	-2.103205	14.282221	0.270930	-0.900839
9	NB	SVC	0.845833	0.988889	-0.143056	0.020418	-7.006460	15.434360	0.000031	-3.000988

Figura 45. Resultados de aplicar el test de Games Showell, para determinar ¿Cuál es el menor clasificador?

La figura 45, muestra los resultados del test de Games Showell, en ella se aprecia las diferencias realmente significativas en la columna ***diff***, entre cada par de algoritmos de clasificación, a partir de estos resultados obtenemos la tabla 5, resumen de test Games Showell.

Tabla 5

Resumen del test de Games Showell, donde se identifica las diferencias significativas.

Clasificador A	Clasificador B	$\lvert\bar{A} - \bar{B}\rvert$	Game Howell (GH)	Diferencia Realmente Significativa	Motivo
AD	KNN	-0.07222	0.255180	No	| media(A) - media(B) | = -0.07222 < GH = 0.25518
AD	MLP	-0.03750	0.825650	No	| media(A) - media(B) | = -0.0375 < GH = 0.82565
AD	NB	0.05833	0.462490	No	| media(A) - media(B) | = 0.05833 < GH = 0.46249
AD	SVC	-0.08472	0.120100	No	| media(A) - media(B) | = -0.08472 < GH = 0.1201
KNN	MLP	0.03472	0.645300	No	| media(A) - media(B) | = 0.03472 < GH = 0.6453
KNN	NB	0.13056	0.000230	Si	| media(A) - media(B) | = 0.13056 > GH = 0.00023
KNN	SVC	-0.01250	0.964600	No	| media(A) - media(B) | = -0.0125 < GH = 0.9646
MLP	NB	0.09583	0.012930	Si	| media(A) - media(B) | = 0.09583 > GH = 0.01293
MLP	SVC	-0.04722	0.270930	No	| media(A) - media(B) | = -0.04722 < GH = 0.27093
NB	SVC	-0.14306	0.000030	No	| media(A) - media(B) | = -0.14306 < GH = 0.00003

Según la tabla 5, existen diferencias realmente significativas entre KNN y NB, de igual modo entre MLP y NB, y apreciando también, la tabla 4, podemos concluir en:

Los algoritmos KNN, SVC, son los mejores clasificadores, por no existir diferencias realmente significativas entre ellos, los siguientes mejores clasificadores son AD y MLP, y el peor clasificador es NB.

b) Discusión con otros autores

Para el dataset E-GEOD-42568, no se encontraron trabajos relacionados, los autores del dataset, obtuvieron los mencionados datos, con el fin de determinar correlación de las redes transcripcionales con la supervivencia del cáncer de mama. Es decir, extraer grupos de genes coexpresados de grandes conjuntos de datos de expresión de ARN mensajero heterogéneo, habiendo detectado 11 grupos de genes (Clarke *et al*, 2013)

4.3. Objetivo específico 3

Primer experimento (E-GEOD-73002)

a) Interpretar los resultados

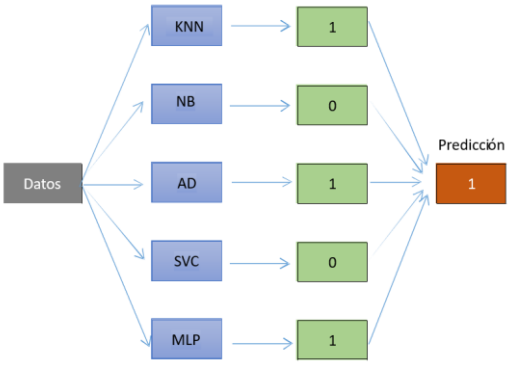


Figura 46. Voting heterogéneo planteado

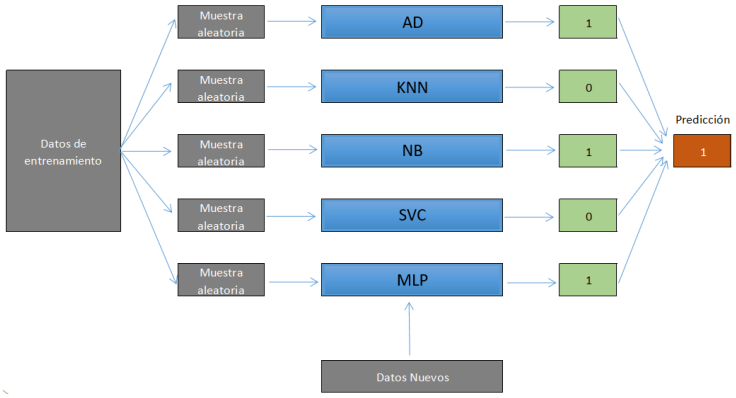


Figura 47. Bagging heterogéneo planteado

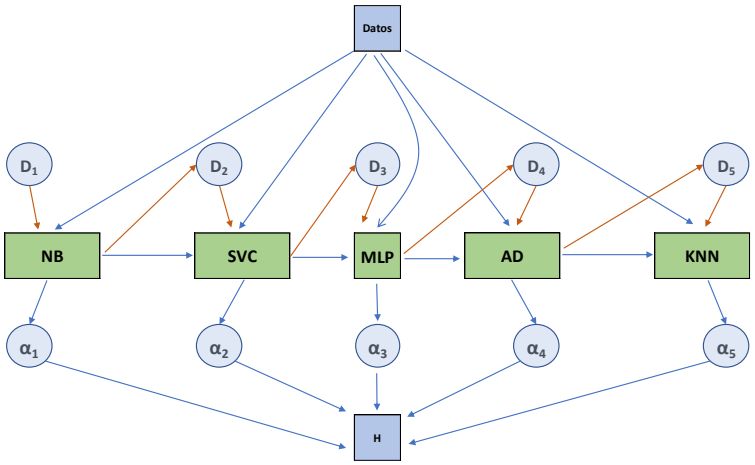


Figura 48. Boosting heterogéneo planteado

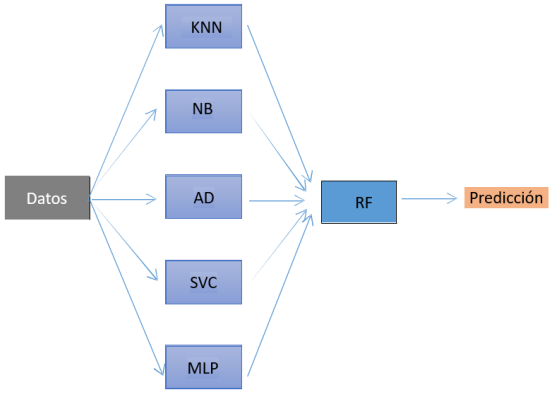


Figura 49. Stacking heterogéneo planteado

Las figuras 46, 47, 48, y 49 muestran, los diseños propuestos de los modelos de ensamble heterogéneo voting, bagging, boosting y stacking respectivamente. En dichos diseños se utilizan algoritmos de clasificación: MLP, SVC, KNN, NB, AD. Diseños que fueron implementados. El modelo voting y stacking por naturaleza son modelos heterogéneos, por lo que podemos encontrar implementaciones en diversas librerías; pero no existe implementaciones de bagging y boosting con algoritmos heterogéneos, más es necesario aclarar, que, tanto bagging y boosting utilizan algoritmos homogéneos.

Tabla 6

Valor de hiper parámetros que optimizan los algoritmos utilizando en los ensambles.

Algoritmo	Hiper parametro	Valores
KNN	N vecinos	4
	Pesos	uniform
	Algoritmo	ball_tree
AD	Criterio	gini
	Divisor	best
	Max prof árbol	20
NB	No posee hiper parametros	
SVC	Kernel	linear
	Costo	0.1
MLP	Capas ocultas	(100, 50, 10)
	Función activación	relu
	Coef aprendizaje	invscaling
	Momentum	0.5

Tabla 7

Exactitud de predicción de cáncer de mama, con el 25% de datos de test.

Ensamble	Exactitud
Voting	97.91%
Bagging	97.49%
Boosting	99.54%
Stacking	98.32%

La tabla 6, muestra para cada algoritmo, que se utiliza en el ensamblaje de meta modelos, los valores de hiper parámetros utilizados, valores encontrados con búsqueda aleatoria de matrices. La tabla 7, muestra la exactitud a la que se llega en el test de modelos de ensamble.

	Voting	Bagging	Boosting	Stacking
CV = 1	1.000000	0.966667	0.997906	1.000000
CV = 2	1.000000	0.950000	0.987506	0.988827
CV = 3	1.000000	0.956667	0.982736	1.000000
CV = 4	1.000000	0.943333	0.991265	0.983240
CV = 5	0.995816	0.973333	0.991296	1.000000
CV = 6	0.995816	0.923333	0.995692	0.994413
CV = 7	0.987448	0.980000	0.993754	1.000000
CV = 8	0.995816	0.979661	0.991879	1.000000
CV = 9	0.991597	0.993220	0.989275	0.988827
CV = 10	0.798319	0.949153	0.992474	0.988827
PROMEDIO	0.995816	0.961667	0.991588	0.997207

Figura 50. Valores de exactitud de diagnósticos de cáncer de mama, aplicando validación cruzada, con k=10

Para determinar el mejor ensamble, se obtiene 10 valores de exactitud, aplicando validación cruzada (*k=10*), La figura 50, muestra por cada ensamble 10 valores de exactitud obtenidos con validación cruzada. Valores que son utilizados para determinar el mejor algoritmo.

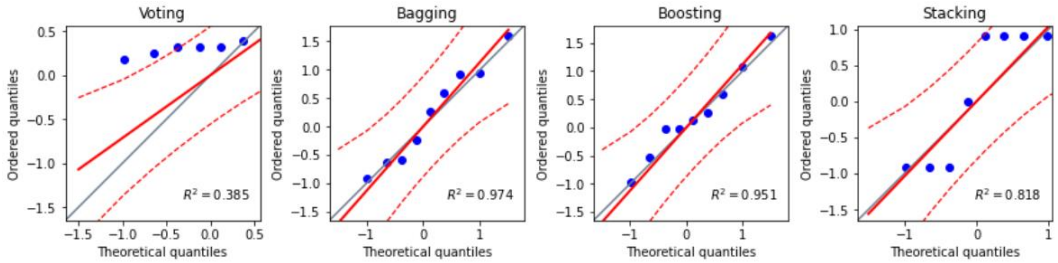


Figura 51. Grafico cuartil-cuartil, que permite detectar conjunto valores de exactitud de predicción, siguen una distribución normal

La figura 51, muestra el grafico cuartil-cuartil para cada modelo de ensamble, con los valores de exactitud de predicción de la figura 50. Y podemos apreciar que los datos que tiene distribución normal son el modelo: Bagging y Boosting. Motivo por el cual para determinar el mejor modelo utilizamos el test Welch ANOVA y test Games Showell.

	Source	ddof1	ddof2	F	p-unc	np2
0	Algoritmo	3	17.691898	7.166715	0.00237	0.14686

Figura 52. Estadístico obtenidos con el test de Welch ANOVA

La figura 52, muestra el los estadísticos obtenidos con el test Welch ANOVA, y para continuar el proceso se plantea las hipótesis correspondientes:

***Ho**: No existe diferencias significativas entre las medias de exactitud de predicción, de los ensambles heterogéneos de clasificación, en la detección de cáncer de seno.*

***H1**: Al menos existe una diferencia significativa, con las medias de exactitud de predicción, de dos ensambles heterogéneos de clasificación, en la detección de cáncer de seno.*

De acuerdo a la figura 52, se obtiene un valor de **p-valor = 0.00237**, expresado en el parámetro ***p-unc***, para un valor de significancia de alfa = 0.01 o un grado de confianza del 99% afirmamos que:

*Se **rechaza** la hipótesis **Ho** y se **acepta** la hipótesis alterna **H_1**, ya que **p-valor = 0.00237 < alfa = 0.01**. esto quiere decir que al menos existe un par de medias de exactitud de predicción, con diferencia significativa. Y concluimos que existe un mejor y peor modelo ensamble de clasificación.*

	A	B	mean(A)	mean(B)	diff	se	T	df	pval	hedges
0	Bagging	Boosting	0.961537	0.991378	-0.029842	0.006751	-4.420137	9.740917	0.006281	-1.893221
1	Bagging	Stacking	0.961537	0.994413	-0.032877	0.006924	-4.748488	10.695788	0.003074	-2.033860
2	Bagging	Voting	0.961537	0.976481	-0.014944	0.020914	-0.714578	10.977425	0.889255	-0.306066
3	Boosting	Stacking	0.991378	0.994413	-0.003035	0.002443	-1.242547	15.571087	0.610497	-0.532204
4	Boosting	Voting	0.991378	0.976481	0.014897	0.019885	0.749173	9.082540	0.874824	0.320884
5	Stacking	Voting	0.994413	0.976481	0.017932	0.019944	0.899128	9.190281	0.805698	0.385112

Figura 53. Estadísticos obtenidos con el test de Games Showel

Para determinar ¿Cuál es el mejor clasificador?, aplicaremos la comparación de diferencias realmente significativas (HSP) con el **test Game Showell**. La figura 53.

Tabla 8

Resumen del test Games Showel

Clasificador A	Clasificador B	$\lvert\bar{A} - \bar{B}\rvert$	Game Howell (GH)	Diferencia Realmente Significativa	Motivo
Bagging	Boosting	-0.02984	0.006280	No	| media(A) - media(B) | = -0.02984 < GH = 0.00628
Bagging	Stacking	-0.03287	0.003070	No	| media(A) - media(B) | = -0.03287 < GH = 0.00307
Bagging	Voting	-0.01494	0.889250	No	| media(A) - media(B) | = -0.01494 < GH = 0.88925
Boosting	Stacking	-0.00303	0.610490	No	| media(A) - media(B) | = -0.00303 < GH = 0.61049
Boosting	Voting	0.01489	0.874820	No	| media(A) - media(B) | = 0.01489 < GH = 0.87482
Stacking	Voting	0.01994	0.805690	No	| media(A) - media(B) | = 0.01994 < GH = 0.80569

La tabla 7: resumen de test Games Showell, informa, que no ha detectado **diferencias realmente significativas,** en pares de modelos analizados, a pesar que el test Welch ANOVA dice que, si existe diferencias significativas. Con lo anterior concluimos que los cinco modelos de ensamble, tienen misma precisión de exactitud en la predicción en cáncer de seno, en muestras de microarray, y si tomamos en cuenta el promedio, los ensambles tienen una exactitud de 98.32%.

b) discusiones con otros autores

La investigación presentada por Tamura *et al*. (2016), muestran, un conjunto de pruebas realizadas, entre ellas están la sensibilidad, exhaustividad y precisión de algún modelo de clasificación, el cual no se detalla, pero si se da a conocer la exactitud de predicción el que llega a 97.1%, en el presente trabajo se determinó que los mejores modelos de ensamble son voting, bagging, boosting y stacking, con una exactitud promedio de 98.32%, superando en un pequeño margen.

Segundo experimento (E-GEOD-42568)

a) Interpretar los resultados

El diseño de la arquitectura de los modelos de ensambles, en este experimento, son similares al primer experimento. Con la diferencia que la mejor configuración de hiper parámetros de los algoritmos de clasificación, son distintos, datos que se aprecian en la tabla 9, obtenidos mediante búsqueda aleatoria en matriz y con validación cruzada.

Tabla 9

Valores de configuración de hiper parámetros de los algoritmos utilizados en modelos ensambles.

Algoritmo	Hiper parametro	Valores	Exactitud
KNN	N vecinos	3	100.00%
	Pesos	distance	
	Algoritmo	ball_tree	
AD	Criterio	entropy	94.59%
	Divisor	random	
	Max prof árbol	50	
NB	No posee hiper parametros		86.49%
SVC	Kernel	poly	100.00%
	Costo	0.5	
MLP	Capas ocultas	(100, 50, 10)	97.30%
	Función activación	tanh	
	Coef aprendizaje	invscaling	
	Momentum	0.9	

Tabla 10

Exactitud obtenida en la fase de test de modelos ensamble, con el 75% de datos.

Ensamble	Exactitud
Voting	100.00%
Bagging	100.00%
Boosting	100.00%
Stacking	100.00%

En la tabla 10, se aprecia que, los cuatro modelos de ensamble heterogéneo llegan a realizar una predicción perfecta, del 100%. Como se sabe, no existe un sistema perfecto, asumimos que dichos valores de exactitud de predicción, pueden deberse a la pequeña cantidad de datos.

	Voting	Bagging	Boosting	Stacking
CV = 1	1.000000	0.900000	0.997906	1.000000
CV = 2	1.000000	1.000000	1.000000	1.000000
CV = 3	1.000000	1.000000	1.000000	0.888889
CV = 4	1.000000	1.000000	0.996827	0.888889
CV = 5	1.000000	1.000000	1.000000	1.000000
CV = 6	1.000000	1.000000	0.998753	1.000000
CV = 7	1.000000	1.000000	1.000000	1.000000
CV = 8	1.000000	0.933333	0.826425	1.000000
CV = 9	0.916667	0.933333	1.000000	1.000000
CV = 10	0.916667	0.800000	1.000000	0.888889
PROMEDIO	1.000000	1.000000	1.000000	1.000000

Figura 54. Valores de exactitud de predicción de cáncer de mama, encontrados aplicando validación cruzada k=10, valores a utilizar en la elección del mejor predictor

Para determinar el mejor modelo de ensamble en la predicción de cáncer de mama, no es suficiente con utilizar la exactitud de la tabla 10. Se realiza una validación cruzada (*k=10*) mostrados en la figura 54, a estos valores debe aplicar el test ANOVA o test Welch ANOVA dependiendo de la normalidad de datos.

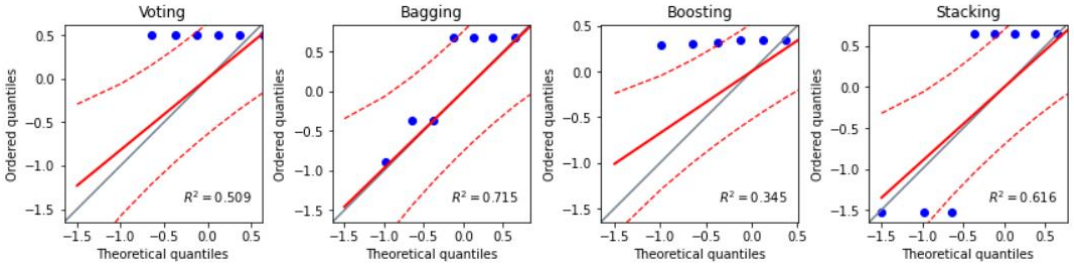


Figura 55. Gráfico cuartil-cuartil, que muestra la normalidad de datos de predicción con validación cruzada.

De acuerdo a la figura 55, gráfico cuartil-cuartil se aprecia que los valores de exactitud de predicción en cáncer de mama, no siguen una distribución normal, motivo por el cual, se utiliza los test Welch ANOVA y Games Showell.

	Source	ddof1	ddof2	F	p-unc	np2
0	Algoritmo	3	19.406472	0.52898	0.667669	0.045125

Figura 56. Estadísticos obtenidos por el test Welch ANOVA

La figura 56, muestra el ***p-valor***, valor obtenido, aplicando el test Welch ANOVA. A continuación, planteamos las hipótesis nula y alterna.

Ho: *No existe diferencias significativas entre las medias de exactitud de predicción, de los modelos de ensambles de clasificación, en la detección de cáncer de seno.*

H1: *Al menos existe una diferencia significativa, con las medias de exactitud de predicción, de dos modelos ensambles de clasificación, en la detección de cáncer de seno.*

Aplicando el test Welch ANOVA, los resultados obtenidos se aprecian en la figura 56, se obtiene un valor de **p-valor = 0.667669**, expresado en el parámetro ***p-unc***.

Para un valor de significancia de alfa = 0.01 o un grado de confianza del 99% afirmamos que:

*Se **acepta** la hipótesis **Ho** y se **rechaza** la hipótesis alterna **H_1**, ya que **p-valor = 0.667669 > alfa = 0.01**. esto quiere decir que el método Welch ANOVA, no detecta diferencias significativas entre los modelos de ensamble en la predicción de cáncer de seno.*

Por lo anterior se acepta la hipótesis Ho, y podemos afirmar que los cuatro modelos de ensamble, tienen misma exactitud de precisión, y tomando en cuenta la exactitud obtenida en la validación cruzada podemos afirmar que la exactitud de los diferentes modelos es de 99.99%, en el diagnóstico de cáncer de seno para el dataset E-GEOD-42568.

b) Discusiones con otros autores

Para el dataset E-GEOD-42568, no se encontraron trabajos relacionados, los autores del dataset, obtuvieron los mencionados datos, con el fin de determinar correlación de las redes transcripcionales con la supervivencia del cáncer de mama. Es decir, extraer grupos de genes coexpresados de grandes conjuntos de datos de expresión de ARN mensajero heterogéneo, habiendo detectado 11 grupos de genes (Clarke *et al*, 2013)

CONCLUSIONES

Se logró identificar y seleccionar muestras de microrray de oligonucleótidos. La matriz de microarrays codificado con E-GEOD-73002, posee expresiones génicas de ARN de cáncer de mama. Y por otra parte la matriz de microarrays codificado con E-GEOD-42568, posee expresiones génicas de ADN de cáncer de mama. La labor de obtención de datos implicó tareas de comprensión de dominio como: cáncer de mama, oncología, medicina, biología molecular, genética. Y tareas de análisis, exploración y preparación de datos como: integración, balanceo y reducción de dimensionalidad.

Se logro identificar que, los mejores algoritmos, en la identificación de cáncer de mama a partir de expresiones génicas de microarray, son SVC y KNN. Siendo la exactitud alcanzada con el dataset E-GEOD-73002, es de SVC, KNN, MLP 98.88%, AD 98.49% y NB 90.62%. Y con el dataset E-GEOD-42568, es de KNN y SVC 100.00%, AD y MLP 95.94%, y NB 86.49%. valores de exactitud conseguidas en un conjunto grande de experimentos, la estrategia utilizada en la determinación de la mejor configuración de hiper parámetros fue, la búsqueda aleatoria en cuadricula.

Se logro identificar que, los ensambles heterogéneos voting, bagging, boosting y stacking, tiene similar exactitud en la identificación de cáncer de mama a partir de expresiones génicas de microarray, afirmación que se realiza por los resultados obtenidos con los tests Welch ANOVA y Games Showell, que manifiestan que la exactitud de predicción de cáncer no posee diferencias significativas. Los lores exactitud obtenidos son: 98.2% para el dataset E-GEOD-73002, y 99.99% para dataset E-GEOD-42568.

Consideramos que aplicar el test ANOVA y test Tukey para datos con distribución normal, o el test Welch ANOVA y test Games Showell para datos que no siguen distribución normal, permite realizar afirmaciones más certeras, ya que no juzga el desempeño de un modelo de aprendizaje automático en función a un valor, sino a un conjunto de experimentos, tomando en cuenta las diferencias significativas.

RECOMENDACIONES

El dataset E-GOED-42568, utilizado en la presente investigación, posee un desbalance de clases 14% a 86%, ello puede provocar que el aprendizaje de los diferentes modelos no sea homogéneo, para mejorar este inconveniente, se recomienda ampliar los datos de la clase minoritaria, por recolección de datos o generando artificialmente.

El presente trabajo realiza una clasificación binaria de expresiones génicas, para trabajos futuros se recomienda realizar clasificaciones multiclase, ya que las expresiones génicas de microarray existentes, en los repositorios públicos tienen una gran variedad de propósitos. En el diseño e implementación de ensambles heterogéneos de aprendizaje automático de la presente investigación. Se dieron los siguientes casos: Para los ensambles voting y stacking, encontramos librerías que implementan tales casos. También encontramos implementaciones de ensambles bagging y boosting para modelos homogéneos. Pero era necesario modelos heterogéneos, motivo que genero la necesidad de diseñar e implementar, a raíz de este hecho recomendamos se plantee nuevos diseños de meta clasificadores, tomando en cuenta que no existen reglas para definir una determinada arquitectura.

BIBLIOGRAFÍA

Ali, Nursabillilah Mohd; Aziz, Nor Azlina Ab & Besar, Rosli. 2020. «Comparison of microarray breast cancer classification using support vector machine and logistic regression with LASSO and boruta feature selection». *Indonesian Journal of Electrical Engineering and Computer Science* 20 (2) (azaroak 1): 712–719. doi:10.11591/ijeecs.v20.i2.pp712-719.

American Cancer Society. 2023. «Cada cáncer. cada vida». *Cada Cáncer Cada vida*. Recuperado de: https://www.cancer.org/es.html.

Astuti, Kusumastuti Cahyaningrum; Adiwijaya & Widi. 2020. «06 - 2021 - Microarray Gene Expression Classification for Cancer Detection using Artificial Neural Networks and Genetic Algorithm Hybrid Intelligence _ IEEE Conference Publication _ IEEE Xplore». *International Conference on Data Science and Its Applications (ICoDSA)*.

Clarke, Colin et al. 2013. «Correlating transcriptional networks to breast cancer survival: A large-scale coexpression analysis». *Carcinogenesis*. doi:10.1093/carcin/bgt208. Recuperado de: https://academic.oup.com/carcin/article/34/10/2300/2463903?login=false.

Faceli, Katti & Lorena, Ana Carolina. 2021. «Inteligencia Artificial Uma Abordagem de Aprendizado de Máquina».

Fauzi, I. R.; Rustam, Z. & Wibowo, A. 2021. «Multiclass classification of leukemia cancer data using Fuzzy Support Vector Machine (FSVM) with feature selection using Principal Component Analysis (PCA)» in . *Journal of Physics: Conference Series*. Libk. 1725. IOP Publishing Ltd. doi:10.1088/1742-6596/1725/1/012012.

Fogliatto, Flavio S. et al. 2019. «Decision Support for Breast Cancer Detection: Classification Improvement Through Feature Selection». *Cancer Control* 26 (1) (urtarrilak 1). doi:10.1177/1073274819876598.

De Guia, Joseph; Devaraj, Madhavi & Vea, Larry. 2020. «CANCER CLASSIFICATION OF GENE EXPRESSION DATA USING MACHINE LEARNING» 6 (1): 70–77.

Hajieskandar, Ali Reza et al. 2020. «Molecular cancer classification method on microarrays gene expression data using hybrid deep neural network and grey wolf algorithm». *Journal of Ambient Intelligence and Humanized Computing*. doi:10.1007/s12652-020-02478-x.

Hangcheng, Liu. 2015. «Comparing Welch ANOVA, a Kruskal-Wallis test, and traditional ANOVA in case of heterogeneity of variance».

Hengpraprohm, Supoj & Jungjit, Suwimol. 1988. «Ensemble Feature Selection for Breast Cancer

Classification using Microarray Data» 23 (65): 100–114. doi:10.4114/intartif.vol23iss65pp100-114.

Huata Panca, Percy. 2019. «Estadistica Aplicada a la Investigación en Computación».

Instituto Nacional de Enfermedades Neoplásicas. 2023. «¿Qué es el Cáncer?» *¿Qué es el cáncer?* Recueprado de: http://portal.inen.sld.pe/infocancer-inen/.

Khoirunnisa, Azka & Adiwijaya. 2021. «Microarray data classification using minimum redundancy maximum relevance and modified logistic regression for high accuracy cancer detection». *ICIC Express Letters* 15 (5) (maiatzak 1): 429–437. doi:10.24507/icicel.15.05.429.

Knife, Gamma de Pacíico. 2023. «¿En qué situaciones debes buscar una segunda opinión médica?» *¿En qué situaciones debes buscar una segunda opinión médica?* recuperado de: https://blog.gammaknifedelpacifico.com/en-que-situaciones-debes-buscar-una-segunda-opinion-medica.

Loey, Mohammed et al. 2020. «Breast and colon cancer classification from gene expression profiles using data mining techniques». *Symmetry* 12 (3) (martxoak 1). doi:10.3390/sym12030408.

Lyall, Fiona. 2016. «Cell and molecular biology». *MRCOG Part One*: 53–74. doi:10.1017/cbo9781107587519.005.

Mason, Kenneth A; Losos, Jonathan B & Singer, Susan R. 2008. *Biology*.

Mezzasoma, Letizia et al. 2012. «Antigen microarrays for serodiagnosis of infectious diseases - PubMed». Recuperado de: https://pubmed.ncbi.nlm.nih.gov/11751547/.

Mira, José Joaquín et al. 2006. «El derecho a una segunda opinión. Ventajas, barreras y recomendaciones para su ejercicio responsable». *Revista de Calidad Asistencial* 21 (3): 120–128. doi:10.1016/S1134-282X(06)70767-9.

Nacional Instituto del Cáncer. 2023. «Segunda Opinión». Recuperado de: https://www.cancer.gov/espanol/publicaciones/diccionarios/diccionario-cancer/def/segunda-opinion.

Nurlaily, Diana et al. 2019. «Support vector machine for imbalanced microarray dataset classification using ant colony optimization and genetic algorithm» in . *AIP Conference Proceedings*. Libk. 2194. American Institute of Physics Inc. doi:10.1063/1.5139808.

Organización Mundial de la Salud. 2021. «Cáncer». Recuperado de: https://www.who.int/es/news-room/fact-sheets/detail/cancer.

Palma-Ttito, Luis B. et al. 2020. «09 - 2021 - Detection of oligonucleotide microarray mutations by multiclass support vector machine _ Signed in». *Iberian Journal of Information Systems and Technologies* E39: 643–657.

Rajasekaran, S. & Sathyabama, S. 2019. «Neighbor embedding feature selected light gradient boosting classification for breast cancer detection with gene expression data». *International Journal of Innovative Technology and Exploring Engineering* 8 (11 Special Issue) (irailak 1): 645–654. doi:10.35940/ijitee.K1108.09811S19.

Ray, Aritra & Ray, Hena. 2021. «Performance Analysis of Machine Learning Classifiers on Different Healthcare Datasets» (May): 99–111. doi:10.1007/978-981-33-4367-2_11.

Sanchez, Stéphane et al. 2021. «Predictive factors of diagnostic and therapeutic divergence in a nationwide cohort of patients seeking second medical opinion»: 1–9.

Sathya, M & Priya, S Manju. «Modified Whale Optimization Algorithm For Feature Selection In Micro Array Cancer Dataset». www.ijstr.org.

Shafi, A. S.M. et al. 2020. «Detection of colon cancer based on microarray dataset using machine learning as a feature selection and classification techniques». *SN Applied Sciences* 2 (7) (uztailak 1). doi:10.1007/s42452-020-3051-2.

Shah, Shamveel Hussain et al. 2020. «Optimized gene selection and classification of cancer from microarray gene expression data using deep learning». *Neural Computing and Applications*. doi:10.1007/s00521-020-05367-8.

Slack, J. M.W. 2013. *Molecular Biology of the Cell*. *Principles of Tissue Engineering: Fourth Edition*. doi:10.1016/B978-0-12-398358-9.00007-0.

Tabas Madrid, Daniel. 2018. «Herramientas eficientes para el análisis masivo de datos ómicos»: 167. Recuperado de: https://eprints.ucm.es/id/eprint/49798/1/T40488.pdf.

Tamura, Akihiko Shimomura et al. 2016. «Novel combination of serum microRNA for detecting breast cancer in the early stage - Shimomura - 2016 - Cancer Science - Wiley Online Library». Cáncer Sciense. doi:https://doi.org/10.1111/cas.12880. Recuperado de: https://onlinelibrary.wiley.com/doi/10.1111/cas.12880#.

Tavasoli, Niloofar et al. 2021. «An ensemble soft weighted gene selection-based approach and cancer classification using modified metaheuristic learning». *Journal of Computational*

Design and Engineering 8 (4) (abuztuak 1): 1172–1189. doi:10.1093/jcde/qwab039.

Thottathyl, Hymavathi; Pavan, Kanadam Karteeka & Panchadula, Rajeev Priyatam. 2021. «Microarray breast cancer data clustering using map reduce based k-means algorithm». *Revue d'Intelligence Artificielle* 34 (6) (abenduak 31): 763–769. doi:10.18280/RIA.340610.

Weigl, Martin et al. 2021. «Effects of a medical second opinion programme on patients ' decision for or against knee arthroplasty and their satisfaction with the programme» 5: 1–12.

Yawei, Li & Yuan, Luo. 2020. «Performance-weighted-voting model: an ensemble machine learning method for cancer type classification using whole-exome sequencing mutation». *Quantitative Biology* 8 (4) (abenduak 1): 347–358. doi:10.1007/s40484-020-0226-1.

ANEXOS

1. CONSIDERACIONES EN EL ANÁLISIS Y EXPLORACIÓN DE MICROARRAY DE CÁNCER DE MAMA.

Los datos registrados en microarray, son tan variados, que es necesario hacer un análisis y exploración de datos, para garantizar que los datos corresponden a muestras de cáncer de mama. A continuación, se listan algunos de los datos que se registran en los microarray:

- Microarray de ADN, ARN, aminoácidos, proteínas, células, tejidos.
- Microarray que registran cánceres: cáncer de próstata, pulmón, riñón, seno, tiroides, vejiga, páncreas, leucemia, hígado, colon, etc.
- Microarray de un tipo particular de célula de cáncer de seno: células del conducto de leche, células de las glándulas que produce leche, tejido conectivo que da soporte del tejido glandular (filoides), células que revisten los vasos sanguíneos o los vasos linfáticos (angiosarcoma), células de soporte de tejido mamario (sarcomas), células del sistema linfático, sistema que posee células que combaten infecciones (linfomas), células de otros órganos, contagiados por metástasis, otras células, menos frecuentes.
- Microarray que registra información de la evolución de células cancerosas, en base al tratamiento con un fármaco.
- Microarray de cáncer de seno que registrar información de diferentes estadios de la enfermedad:
    i. Etapa temprana: Tumor propagado en un área pequeña.
    ii. Localizado: Tumor que mide entre 20 a 50 mm.
    iii. Propagación regional: Tumor que mide más de 50 mm.
    iv. Propagación lejana: El cáncer se propago más allá del seno.
- Microarray que registra información de líneas celulares, de células cancerosas de seno (Las líneas celulares, son cultivos de tejidos cancerosos en laboratorio).
- Microarray que registrar una variedad de líneas celulares, cuyo propósito es la comparación de líneas celulares.
- Microarray que registra información de microARN. (Los microARN son secuencias de ARN, que se encuentran en la sangre, que son utilizadas en el diagnóstico y tratamiento de cáncer).

- Microarray que registran diferentes sub tipos de cáncer de seno, es decir muestras que, pertenecen a diferentes células cancerosas relacionadas al seno.
- Microarray con sub tipos de cáncer de seno, provenientes de líneas celulares.
- Microarray de cáncer de seno en diferentes estadios, provenientes de líneas celulares.
- Microarray que registra evolución de cáncer de seno, que recibe tratamiento de radioterapia o quimioterapia.
- Microarray de cáncer de seno, obtenido de diferentes plataformas, es decir que la fuente de datos es variada, por la marca, modelo del chip de microarray.
- Microarray de células cancerosas de seno y células sanas.
- Microarray de células cancerosas de seno, que pertenece a pacientes que fallecieron y de pacientes que siguen vivos.
- Muestras de microarray de un color y de dos colores:
    a. Microarray de 1 color: microarray que hibrida sondas de una muestra sospechosa de cáncer, con sondas adosadas en el chip (las sondas del chip, puede ser variado, ya que el diseñador decide que sondas se adosaran al chip)
    b. Microarray de 2 colores: microarray que hibrida sondas de dos muestras, la primera que pertenece a una persona sospechosa de cáncer, y la otra que pertenece a una persona sana.
- Microarray de datos de cáncer de seno, con diferentes diseños (se denomina diseño, a la cantidad y tipo de sondas, que posee el chip de microarray.
- Microarray que contiene información de predicción de la sensibilidad de un fármaco, aplicado a personas con cáncer de seno.
- Microarray de cáncer de seno de humanos y cáncer de seno de otras especies.
- Microarray de cáncer de seno humano por género.
- Microarray de muestras de cáncer a partir de sondas ADNc, ARNm.
- Microarray que registra la evolución de crecimiento de tumores de seno.
- Microarray de muestras de cáncer de seno, para clasificación binaria y multi clasificación.
- Microarray de muestras, cuyo propósito es identificar genes que provocan la aparición del cáncer de seno.

2. PROCESO INTEGRAL DE OBTENCIÓN DE MICROARRAY Y APLICACIÓN DEL MODELO DE APRENIZAJE AUTOMÁTICO

El repositorio NCBI, contiene información biotecnológica de: genoma, genes, secuencias de ADN, secuencias de ARN, secuencias de aminoácidos, proteínas, estructuras y taxonomía de diferentes especies. Una de las sub categorías del repositorio, corresponde a muestras de expresiones génicas de secuencias de ADN o ARN, almacenados en diferentes formatos, la búsqueda del dataset para la investigación es necesario tomar en cuenta las consideraciones detalladas en el ANEXO A de la presente investigación.

La imagen 57, muestra el explorador de dataset de microarray, en las que, se ha resaltado, tres dataset de cáncer de seno, uno perteneciente a ratón y dos pertenecientes a homo sapiens.

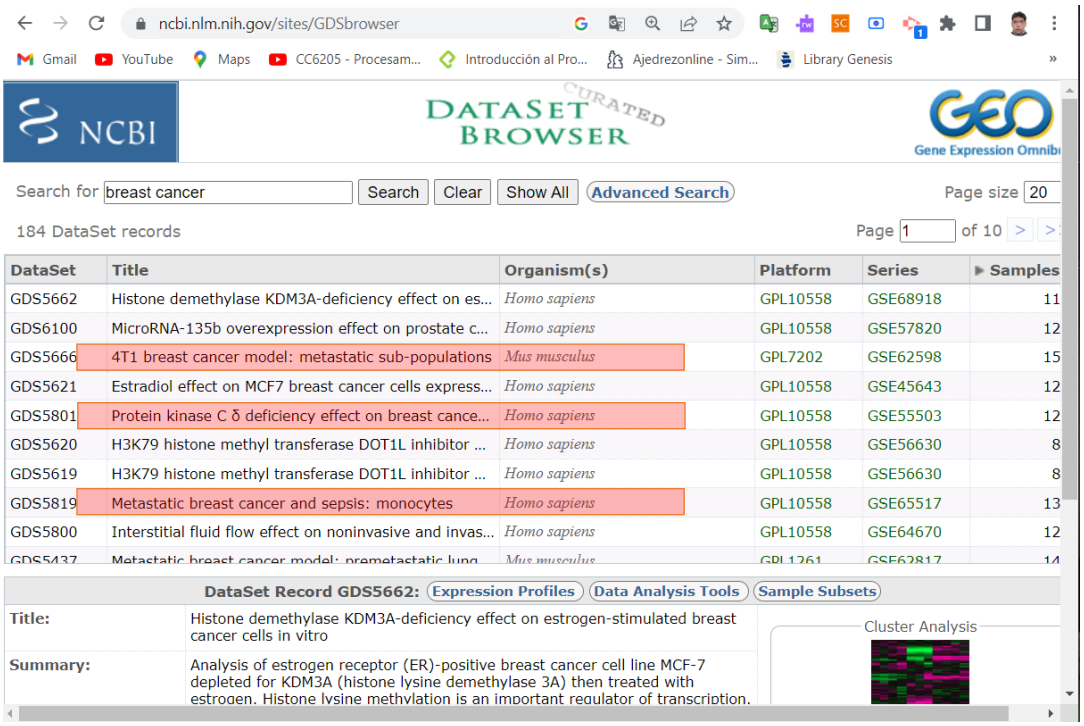


Figura 57. Explorador de dataset de microarray

La figura 58, muestra detalles de la información del dataset seleccionado, entre los datos de importancia que encontramos, están:

- Título: Cáncer de mama metastásico y sepsis.
- Resumen: El dataset muestra similitudes moleculares entre los monocitos (tipo de glóbulo blanco), y los monocitos inmunosupresores (permite que el sistema

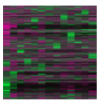
inmunitario sea menos capaz de detectar y destruir las células cancerosas o de combatir las infecciones) reprogramados de pacientes con sepsis (respuesta inmunitaria desbalanceada, anómala, que presenta el organismo, frente a una infección).

- Organismo: homo sapiens.
- Muestras: 13 (perteneciente a 13 individuos diferentes).
- Fecha: 03/02/2015.
- Clases: Cáncer de seno metastásico (4 muestras), Muestra de control saludable (3 muestras), Sepsis por gramnegativos (3 muestras), Tuberculosis (3 muestras).

DataSet Record GDS5819: Expression Profiles | Data Analysis Tools | Sample Subsets

Title:	Metastatic breast cancer and sepsis: monocytes
Summary:	Analysis of the total population of monocytes from patients with metastatic breast cancer (MBC), sepsis, or tuberculosis. Results provide insight into molecular similarities between monocytes from MBC patients and reprogrammed immunosuppressive monocytes from sepsis patients.
Organism:	*Homo sapiens*
Platform:	GPL10558: Illumina HumanHT-12 V4.0 expression beadchip
Citation:	Bergenfelz C, Larsson AM, von Stedingk K, Gruvberger-Saal S et al. Systemic Monocytic-MDSCs Are Generated from Monocytes and Correlate with Disease Progression in Breast Cancer Patients. *PLoS One* 2015;10(5):e0127028. PMID: 25992611
Reference Series:	GSE65517
Sample count:	13
Value type:	count
Series published:	2015/02/03

Cluster Analysis



Download

- DataSet full SOFT file
- DataSet SOFT file
- Series family SOFT file
- Series family MINiML file
- Annotation SOFT file

Sample Subsets

Samples	Factors: disease state	Title
GSM1599177	metastatic breast cancer	Primary human monocytes from patient with metastatic breast cancer #1
GSM1599178	metastatic breast cancer	Primary human monocytes from patient with metastatic breast cancer #2
GSM1599179	metastatic breast cancer	Primary human monocytes from patient with metastatic breast cancer #3
GSM1599180	metastatic breast cancer	Primary human monocytes from patient with metastatic breast cancer #4
GSM1599181	healthy control	Primary human monocytes from healthy control #1
GSM1599182	healthy control	Primary human monocytes from healthy control #2
GSM1599183	healthy control	Primary human monocytes from healthy control #3
GSM1599184	gram-negative sepsis	Primary human monocytes from patient with gram-negative sepsis #1
GSM1599185	gram-negative sepsis	Primary human monocytes from patient with gram-negative sepsis #2
GSM1599186	gram-negative sepsis	Primary human monocytes from patient with gram-negative sepsis #3
GSM1599187	tuberculosis	Primary human monocytes from patient with tuberculosis #1
GSM1599188	tuberculosis	Primary human monocytes from patient with tuberculosis #2
GSM1599189	tuberculosis	Primary human monocytes from patient with tuberculosis #3

Figura 58. Detalles del dataset de cáncer de seno metastásico y sepsis.

En el presente Anexo se muestra el proceso de análisis, exploración, preparación de datos y posterior aplicación de algoritmos de aprendizaje automático, para el dataset GSE45827:

- **Fecha de publicación**: 24 de marzo de 2016.
- **Fecha de actualización**: 5 de abril de 2019.

- **Autores**: Fatima Mechta-Grigoriou, Tina Gruosso, Yann Kieffer, Thierry Dubois.
- **URL referencia**:

  GEO-NCBI: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE45827

  EMBL-EBI: https://www.ebi.ac.uk/biostudies/arrayexpress/studies/E-GEOD-45827
- **Código:** E-GEOD-45827 en NCBI, GSE45827 en Instituto Europeo de Bioinformática.
- **Título**: datos de expresión génica, de subtipos de cáncer de mama.
- **Organismo**: homo sapiens
- **Descripción**: el dataset contiene datos de expresión de subtipos de cáncer de mama En un estudio de cohorte de cáncer de mama invasivo primario (41 TN, 30 HER2, 29 Luminal A y 30 Luminal B), así como 11 muestras de tejidos normales y 14 líneas celulares, obtuvimos una muestra de tumor en la cirugía antes cualquier tratamiento del paciente. Se extrajo el ARN total de todas las muestras y se cuantificó el transcriptoma completo con chips Affymetrix U133 Plus 2.0.
- **Preprocesado**: Los autores detallan que la tarea de pre procesado consistió en: normalización usando GC-RMA (R versión 2.14.1). Luego, se aplicó un modelo lineal mixto que utiliza la máxima verosimilitud restringida. El modelo incluyó los diferentes lotes experimentales y el tipo de muestra como efectos fijos, y trató la variación técnica como efectos aleatorios. No se incluyeron términos de interacción. Este modelo se usó para derivar la importancia de la expresión diferencial entre diferentes tipos de tumores y para la corrección de los efectos de lote e hibridación. Para un análisis posterior, se promediaron las réplicas. Se elegio descartar los conjuntos de sondas con señal de nivel de ruido del análisis: específicamente, consideramos como no expresados y, por lo tanto, se descartó, todos los conjuntos de sondas con una intensidad log2 de menos de 4 en el 95% o más de las muestras de tejido. ID_REF = VALOR = Intensidad log2 de GC-RMA.
- Plataforma: Matriz Affymetrix Human Genoma U133 Plus 2.0
- Cantidad de muestras: 155

Propósito del dataset

El dataset posee 6 subtipos de muestras de cáncer de mama, subtipos que detallamos a continuación:

- **HER2**: Es un tumor de seno, que se expresa por niveles altos de la proteína conocida como HER2. La HER2 es una proteína en el exterior de todas las células mamarias que promueve el crecimiento. Las células del cáncer de seno con niveles más altos de lo normal de HER2 se llaman HER2-positivas. Estos cánceres tienden a crecer y propagarse más rápido que otros tipos de cáncer de seno, pero responden al tratamiento con medicamentos que tienen como blanco a la proteína HER2.

- **BASAL**: Este tipo de cáncer, también denominado cáncer de mama triple negativo, incluye tumores que son negativos para el receptor de estrógeno, el receptor de progesterona y el receptor del factor de crecimiento epidérmico humano 2. Los pacientes con cáncer de mama de tipo basal tienen probabilidades de beneficiarse de la quimioterapia.

- **LINEA CELULAR**: Las líneas celulares, es un tipo de célula, que ha sido adaptado, para crecer en el laboratorio (como un cultivo), y que se utiliza en investigación, existen varios tipos de células de este tipo, entre ellas están: MCF-7, MDA, MB-231, ViBo, HeLa.

- **LUMINAL A**: El cáncer de mama luminal A es aquel con tumores RH positivos (RH ≅ receptor hormonal. Si el tumor es RH positivo (RH+), se trata de un tumor que tiene receptores de hormonas (estrógenos o progesterona) que favorecen el crecimiento tumoral), HER2 negativos y un Ki67 inferior al 20 %. Supone entre el 25 y el 50 % de todos los tipos de cáncer de mama y es el de mejor pronóstico.

- **LUMINAL B**: Luminal B es aquel con tumores RH+ y HER2 positivo o negativo.

- **NORMAL**: Células que no presentan cáncer.

La distribución de datos, se aprecia en la figura 59.

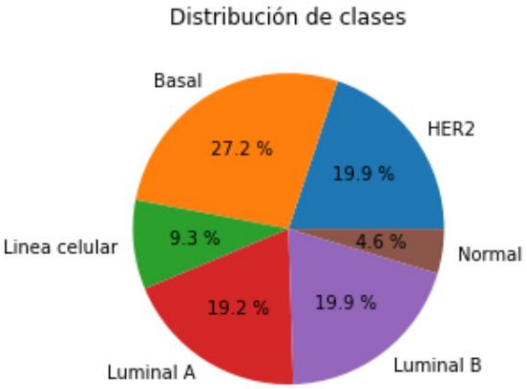


Figura 59. Distribución de clases del dataset GSE45827

La figura 60, muestra los datos de la matriz de expresiones génicas, cuyas características detallamos:

- Cada fila, corresponde a una muestra de un paciente, información extraída en un chip de microarray.
- La primera columna "*type*" corresponde el atributo objetivo o clase.
- De la segunda columna, a la última, registran expresiones génicas, de pequeñas secciones de ADN (sondas) del genoma humano, en la figura de referencia están rotulada como: *1007_s_at, 1053_at, 117_a7, ..., etc.*
- Las expresiones génicas están normalizadas mediante log2.

	type	1007_s_at	1053_at	117_at	121_at	1255_g_at	1294_at	1316_at	1320_at	1405_i_at	...	AFFX-r2-Ec-bioD-3_at	AFFX-r2-Ec-bioD-5_at
0	basal	9.850040	8.097927	6.424728	7.353027	3.029122	6.880079	4.963740	4.408328	8.870780	...	12.229711	11.852955
1	basal	9.861357	8.212222	7.062593	7.685578	3.149468	7.542283	5.129607	4.584418	7.767646	...	12.178531	11.809408
2	basal	10.103478	8.936137	5.735970	7.687822	3.125931	6.562369	4.813449	4.425195	9.417956	...	12.125108	11.725766
3	basal	9.756875	7.357148	6.479183	6.986624	3.181638	7.802344	5.490982	4.567956	9.022345	...	12.111235	11.719215
4	basal	9.408330	7.746404	6.693980	7.333426	3.169923	7.610457	5.372469	4.424426	9.400056	...	12.173642	11.861296

5 rows × 54676 columns

Figura 60. Expresiones génicas de muestra de cáncer de seno, la imagen visualiza únicamente 5 primeras muestras de un total de 151, y 11 genes de un total de 54675

- Utilizando las herramientas de bioinformática del NCBI, mostramos los detalles de una sonda en particular: **1007_s_at**, que está, ubicado en alguna región de uno de los 46 cromosomas del ser humano.
  - **Genoma humano**: Se considera genoma humano, es la secuencia completa de ADN, el que, está formado por 23 pares de cromosomas, identificando 24

cromosomas diferentes, cromosomas que se ubican en el núcleo de la célula, como se muestra en la figura 61.

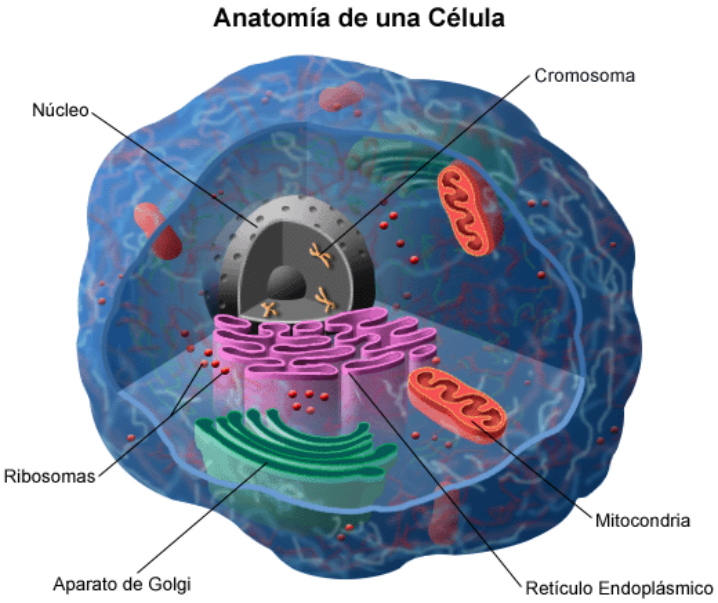


Figura 61. Célula eucariota, que posee el humano

- **Cromosomas**: La figura 62, muestra 22 cromosomas, más los cromosomas X e Y. La sonda **1007_s_at** está ubicada en una región del cromosoma 6.

Figura 62. Cromosomas de homo sapiens

- **Cromosoma 6**: La figura 63, muestra los detalles del cromosoma 6, en ella se aprecia una línea vertical roja, está, es la zona en la que se ubica, la sonda 1007_s_at.

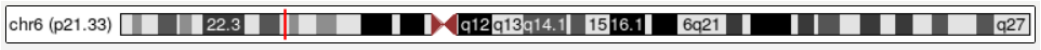


Figura 63. Cromosoma 6 a detalle, las zonas oscuras, corresponden a los genes

- **Región 1007_s_at**: La figura 78 muestra, los detalles de la región *1007_s_at*, La sonda en análisis, consta de 10 sub regiones (*1007_s_at_1, 1007_s_at_2, ...*), resaltados por las barras horizontales de color negro en la figura 64.

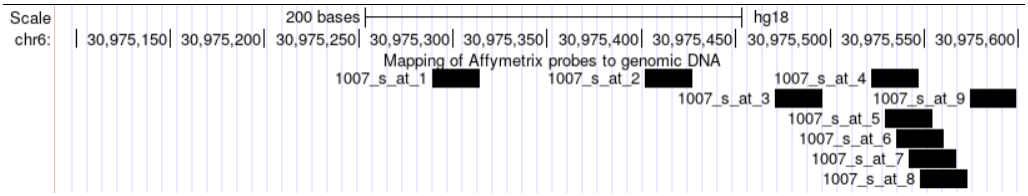


Figura 64. Región 1007_s_at

- **Sub región 1007_s_at_1**: la figura 65, muestra la sub región *1007_s_at_1*, a escala muy pequeña, en la que, se aprecia las bases (Adenina, Citosina, Guanina, Timina), se puede observar que la sub región consta de 25 bases, es decir la secuencia: CACCCAGCTGGTCCTGTGGCTGGGA.

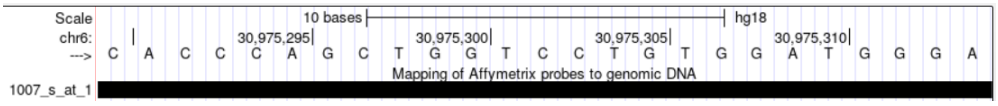


Figura 65. sub región 1007_s_at_1

- **Secuencia completa de región 1007_s_at**: La figura 66, muestra la secuencia completa de bases de ADN, de la región ***1007_s_at***, el que, consta de 451 bases. Estás 451 bases son representadas por un único valor, en la matriz de expresiones génicas de la figura 60. Este único valor representa la cantidad sondas 1007_s_at que se expresan en el organismo (cantidad de sondas 1007_s_at que hibridaron en el chip de microarray), es decir, la expresión génica equivale a la cantidad de sondas ***1007_s_at*** presentes en el paciente del individuo en estudio.

```
>hg18_ct_ProbeMapping_6499 range=chr6:30975150-30975600 5'pad=0 3'pad=0 strand=+ repeatMasking=none
CGACAGCCCATCACCTCTAATAGAGGCAGTGAGACTGCAGGTGGGCTGGG
CCCACCCAGGGAGCTGATGCCCCTTCTCCCCTTCCTGGACACACTCTCAT
GTCCCCTTCCTGTTCTTCCTTCCTAGAAGCCCCTGTCGCCCACCCAGCTG
GTCCTGTGGATGGGATCCTCTCCACCCTCCTCTAGCCATCCCTTGGGGAA
GGGTGGGGAGAAATATAGGATAGACACTGGACATGGCCCATTGGAGCACC
TGGGCCCCACTGGACAACACTGATTCCTGGAGAGGTGGCTGCGCCCCCAG
CTTCTCTCTCCCTGTCACACACTGGACCCCACTGGCTGAGAATCTGGGGG
TGAGGAGGACAAGAAGGAGAGGAAAATGTTTCCTTGTGCCTGCTCCTGTA
CTTGTCCTCAGCTTGGGCTTCTTCCTCCTCCATCACCTGAAACACTGGAC
C
```

Figura 66. Secuencia completa de la región 1007_s_at

- **Selección de genes:** El primer proceso, fue la reducción de dimensionalidad. La cantidad de atributos del dataset, es de 54675, se utilizó el método de ***Información Mutua***, que selecciona los atributos con la puntuación más alta en la tarea de

predicción, método basado en la comparación de entropía (grado de desorden) generado por dos o más atributos, combinado con el método ANOVA, responsable de asignar la puntuación de importancia a cada atributo. Los atributos (genes) de los microarray contienen información relevante, mezcladas con ruido e información redundante, ANOVA permite detectar atributos que tenga poca variabilidad, que son considerados como atributos "redundantes", y así poder eliminarlos. La figura 67, muestra, una gráfica de barras (54675 barras, una por cada atributo), la altura de las barras expresa la importancia del atributo para la predicción.

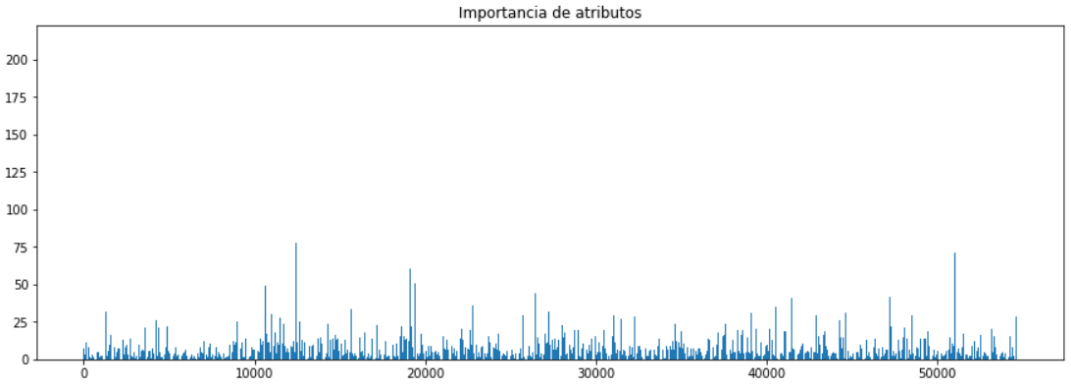


Figura 67. Selección de genes importantes en la predicción

De las 54675 sondas (atributos), se seleccionó las sondas con mayor puntuación. Sondas con puntuación mayor al 10% de la máxima puntuación, consiguiendo extraer con esta regla un total de 3631 sondas (genes).

También, fueron eliminadas las clases minoritarias, es decir las muestras normales que tiene 7 unidades, y las muestras de línea celular que poseen 14 unidades, reduciéndose a un total de 130 muestras, cuya distribución se aprecia en la figura 68.

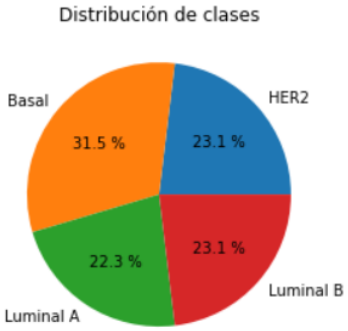


Figura 68. Distribución de clases después de la reducción de la dimensionalidad

- **Búsqueda de la mejor configuración de hiper parámetros, de algoritmos de aprendizaje automático**: Para determinar la mejor configuración de los algoritmos de aprendizaje automático, se utilizó el método de búsqueda aleatoria en cuadricula. En cada caso se estableció un conjunto de valores, para cada uno, de los hiper parámetros de los algoritmos de aprendizaje automático, de los cuales se seleccionó de forma aleatoria 50 configuraciones. Luego se utilizó validación cruzada con k=10, y se entrenó con cada una de las configuraciones, es decir se entrenaron un total de 50x10 modelos para cada algoritmo, buscando la mejor configuración, la tabla 11 muestra los valores con los que se experimentó.

Tabla 11

Hiper parámetros experimentados en la búsqueda de la mejor configuración

Algoritmo	Hiper parametro	Valores	Descripción
KNN	N vecinos	[2,3,4,5,6,7,8,9,10,11,12,13,14,15]	Número de vecinos a usar
	Pesos	['uniform', 'distance']	uniforme' : pesos uniformes. Todos los puntos en cada vecindario se ponderan por igual. 'distancia' : puntos de peso por el inverso de su distancia. en este caso, los vecinos más cercanos de un punto de consulta tendrán una mayor influencia que los vecinos más alejados.
	Algoritmo	['auto', 'ball_tree', 'kd_tree', 'brute']	Algoritmo utilizado para calcular los vecinos más cercanos: 'ball_tree' utilizará BallTree 'kd_tree' utilizará KDTree 'brute' utilizará una búsqueda de fuerza bruta. 'auto' intentará decidir el algoritmo más apropiado
AD	Criterio	['gini', 'entropy', 'log_loss']	La función para medir la calidad de una división. Los criterios admitidos son "gini" para la impureza de Gini y "log_loss" y "entropía", ambos para la ganancia de información de Shannon
	Divisor	['best', 'random']	La estrategia utilizada para elegir la división en cada nodo. Las estrategias admitidas son "mejor" para elegir la mejor división y "aleatoria" para elegir la mejor división aleatoria.
	Max prof árbol	[10, 20, 30, 40, 50, 100]	La profundidad máxima del árbol.
NB	No posee hiper parametros		
SVC	Kernel	['linear', 'poly', 'rbf', 'sigmoid']	Especifica el tipo de kernel que se utilizará en el algoritmo.
	Costo	[0.1, 0.2, 0.5, 0.8, 1, 2, 4, 8, 15]	Costo es el valor de penalización admitida en la deterción del hiperplano.
MLP	Capas ocultas	[(100, 50, 10), (1500, 500, 100), (1000, 100, 10)]	Número de neuronas en la capa oculta
	Función activación	['identity', 'logistic', 'tanh', 'relu']	Función de activación de la capa oculta. 'identidad', activación sin operación, útil para implementar un cuello de botella lineal, devuelve f(x) = x 'logística', la función logística sigmoidea, devuelve f(x) = 1 / (1 + exp(-x)). 'tanh', la función tan hiperbólica, devuelve f(x) = tanh(x). 'relu', la función de unidad lineal rectificada, devuelve f(x) = max(0, x)
	Coef aprendizaje	['invscaling', 'adaptive']	Algoritrmo que busca el valor de coef. de aprendizaje para acutalización de pesos. 'invscaling' disminuye gradualmente la tasa de aprendizaje en cada paso de tiempo 't' usando un exponente de escala inversa. 'adaptable' mantiene la tasa de aprendizaje constante en 'learning_rate_init' siempre que la pérdida de entrenamiento siga disminuyendo.
	Momentum	[0.5, 0.7, 0.9]	Momentum para la actualización del descenso de gradiente.

Habiendo aplicado validación cruzada, se obtuvo el valor de hiper parámetros, que optimizan la predicción, la tabla 12, muestra la mejor configuración encontrada para cada algoritmo, mediante el método búsqueda aleatoria en cuadricula.

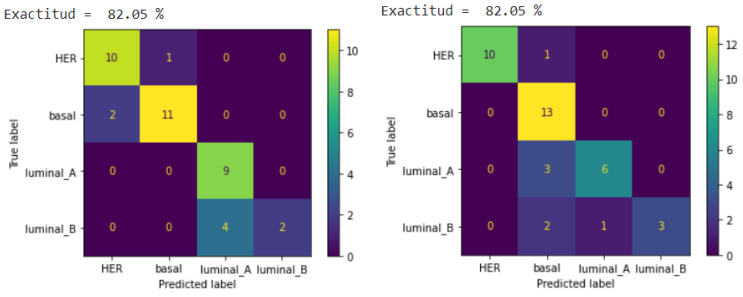
Tabla 12

Valor de hiper parámetros óptimos encontrados mediante búsqueda aleatoria en cuadricula

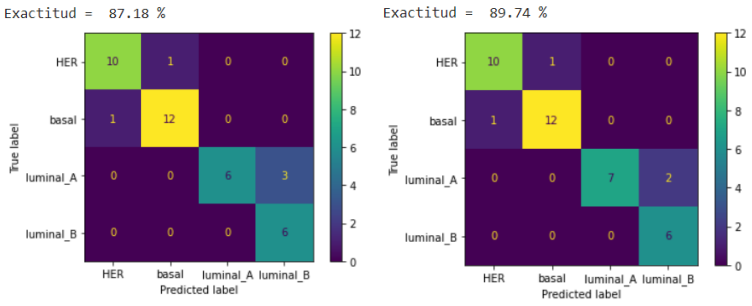
Algoritmo	Hiper parametro	Valores	Exactitud
KNN	N vecinos	4	82.05%
	Pesos	uniform	
	Algoritmo	kd_tree	
AD	Criterio	gini	82.05%
	Divisor	best	
	Max prof árbol	30	
NB	No posee hiper parametros		87.18%
SVC	Kernel	rbf	89.74%
	Costo	8	
MLP	Capas ocultas	(100, 50, 10)	69.23%
	Función activación	relu	
	Coef aprendizaje	adaptive	
	Momentum	0.9	

La figura 69, muestra la matriz de confusión, generada por los algoritmos KNN, AD, NB, SVC, MLP, utilizando los la mejor configuración de hiper parámetros de la tabla 12.

KNN ***AD***



NB ***SVC***



MLP

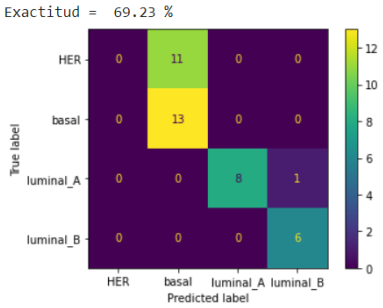


Figura 69. Matrices de confusión de los cinco algoritmos de clasificación

- **Valores medios encontrado con validación cruzada**: Para garantizar que los resultados de las métricas de predicción sean confiables, se aplicó nuevamente validación cruzada con *k=10*, obteniendo un total de 10 valores de predicción, cuya gráfica en diagrama de cajas se aprecia la figura 70. Estos valores más adelante se utilizan para identificar el mejor clasificador.

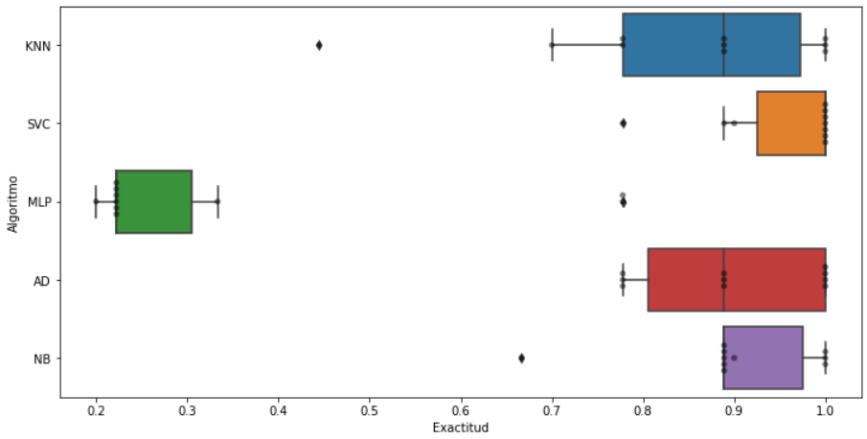


Figura 70. Diagrama de bigotes de exactitud de predicción obtenida con validación cruzada (k=10)

- **Cumplimiento de requisitos, para aplicar ANOVA y Tukey**: Para poder aplicar el test de ANOVA y luego el test de Tukey, es necesario que los datos posean distribución normal, la figura 71 muestra, una gráfica qqplot (cuartil cuartil), grafico obtenidos de valores de exactitud de predicción en cada uno de los test de validación cruzada de la figura 72. En la figura 71, se aprecia que, únicamente las métricas de KNN tiene distribución normal, por estar distribuidos en la diagonal secundaria (línea roja contina), evidenciando que el resto de clasificadores no tiene distribución normal.

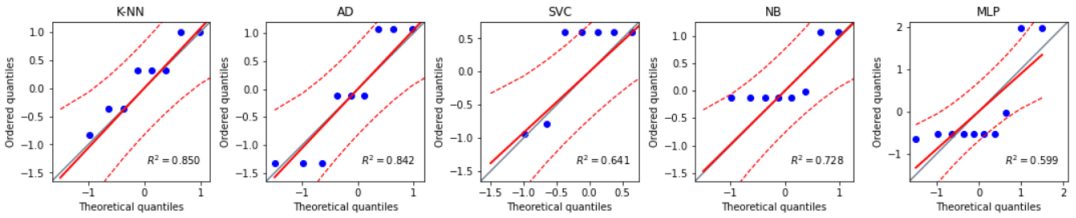


Figura 71: Gráfica cuartil-cuartil de los resultados de exactitud obtenidas con validación cruzada (k=10)

	KNN	SVC	MLP	AD	NB
CV = 1	0.700000	0.900000	0.200000	1.000000	0.900000
CV = 2	0.888889	1.000000	0.222222	0.888889	0.888889
CV = 3	0.888889	1.000000	0.222222	1.000000	0.888889
CV = 4	0.777778	1.000000	0.222222	0.777778	1.000000
CV = 5	1.000000	1.000000	0.777778	1.000000	1.000000
CV = 6	1.000000	1.000000	0.333333	1.000000	0.888889
CV = 7	0.888889	0.888889	0.777778	0.777778	0.888889
CV = 8	0.777778	1.000000	0.222222	0.888889	1.000000
CV = 9	0.444444	0.777778	0.222222	0.777778	0.666667
CV = 10	1.000000	1.000000	0.222222	0.888889	0.888889
PROMEDIO	0.888889	1.000000	0.222222	0.888889	0.888889

Figura 72. Exactitud de algoritmos de clasificación obtenido por validación cruzada con k=10

De los anterior, podemos afirmar que aplicar el método ANOVA y Tukey para identificar el mejor clasificador, no será posible, por tanto, optamos, por utilizar el test Welch ANOVA (que es una adaptación del test t Student y es más confiable cuando las muestras tienen varianzas desiguales o tamaños de muestra desiguales) y Game Showell (es un test similar Tukey, que hace uso de HSD ≅ diferencias realmente significativas, pero más robusta para varianza heterogénea).

- **Test de Welch ANOVA y Game Howell**: Las hipótesis que debemos probar son:

  Ho: *No existe diferencias significativas entre las medias de exactitud de predicción, de los algoritmos de clasificación.*

  H1: *Al menos existe una diferencia significativa, con las medias de exactitud de predicción, de dos algoritmos de clasificación.*

Aplicando la función de ***pairwise_gameshowell***, de la librería ***pingouin***, obtenemos la métrica del test de Welch ANOVA, mostrado en la figura 73. Donde ***p-valor*** es igual a 0.000006, expresado en el parámetro ***p-unc***.

	Source	ddof1	ddof2	F	p-unc	np2
0	Algoritmo	4	21.948169	14.718396	0.000006	0.722897

Figura 73. Test de Welch ANOVA

Para un valor de significancia de alfa = 0.01 o un grado de confianza del 99% afirmamos que:

*Se **rechaza** la hipótesis **Ho** y se **acepta** la hipótesis alterna **H_1**, ya que **p-valor = 0.000006 < alfa = 0.01**. esto quiere decir que al menos existe un par de medias de exactitud de predicción, con diferencia significativa. Y concluimos que existe un mejor y peor clasificador.*

Para determinar ¿Cuál es el mejor clasificador?, aplicaremos la comparación de diferencias realmente significativas (HSP) con el test Game Howell. Haciendo uso de la función ***pairwise_gameshowell*** de la librería ***pingouin,*** cuyos resultados se aprecian en la figura 74, en ella se aprecia la columna ***diff*** (diferencia entre cada par de clasificadores), ***pval*** (valores Games Showell, equivalente a p-tukey, valores serán analizados para determinar el mejor clasificador.

	A	B	mean(A)	mean(B)	diff	se	T	df	pval	hedges
0	AD	KNN	0.900000	0.836667	0.063333	0.062615	1.011480	14.201673	0.845989	0.433234
1	AD	MLP	0.900000	0.342222	0.557778	0.079674	7.000790	12.060235	0.000112	2.998559
2	AD	NB	0.900000	0.901111	-0.001111	0.043491	-0.025548	17.999989	1.000000	-0.010943
3	AD	SVC	0.900000	0.956667	-0.056667	0.039177	-1.446425	17.070443	0.608069	-0.619529
4	KNN	MLP	0.836667	0.342222	0.494444	0.091517	5.402733	16.605311	0.000433	2.314084
5	KNN	NB	0.836667	0.901111	-0.064444	0.062602	-1.029424	14.194903	0.837798	-0.440920
6	KNN	SVC	0.836667	0.956667	-0.120000	0.059686	-2.010523	12.426713	0.315921	-0.861142
7	MLP	NB	0.342222	0.901111	-0.558889	0.079664	-7.015570	12.055647	0.000110	-3.004890
8	MLP	SVC	0.342222	0.956667	-0.614444	0.077393	-7.939253	10.937647	0.000057	-3.400519
9	NB	SVC	0.901111	0.956667	-0.055556	0.039158	-1.418761	17.076136	0.624601	-0.607680

Figura 74. Resultado del test de Game Showell

De la figura 74, obtenemos la tabla 16 (resumen de Games Showell), en la que se aprecia:

Columna 1: Clasificador A

Columna 2: Clasificador B

Columna 3: Diferencia de medias del clasificador A y B

Columna 4: Estadístico Games Showell.

Columna 5: Indica si existe o no diferencia significativa entre clasificador A y B.

Columna 6: Motivo o justificación por él se considera que hay diferencia significativa.

En la tabla 13 podemos detectar los siguiente.

- Existe diferencias significativas entre los clasificadores AD y MLP, y de otra parte entre KNN y MLP.
- No existe diferencias significativas en los otros casos.
- Por otra parte, la exactitud de KNN = 82.05%, AD = 82.05% y MLP = 33.33%, según la figura 69.

Por los puntos anteriores, podemos concluir que el clasificador KNN y AD son los mejores clasificadores en comparación al clasificador MLP, pero el test de Games Howell, indica que no hay diferencia significativa entre KNN y SVC, KNN y NB, AD y SVC, ND y NB, de lo cual afirmamos que los mejores clasificadores son: KNN, AD, NB, SVC, y entre estos no existen diferencias significativas por tanto podemos afirmar que la exactitud de predicción es la misma, y de forma pesimista asumimos dicha exactitud es de 82.05%.

Tabla 13

Resumen de la tabla Games Showell

Clasificador A	Clasificador B	$\lvert\bar{A} - \bar{B}\rvert$	Game Howell (GH)	Diferencia Realmente Significativa	Motivo
AD	KNN	0.063	0.846	No	| media(A) - media(B) | = 0.063 < GH = 0.846
AD	MLP	0.558	0.0001	Si	| media(A) - media(B) | = 0.558 > GH = 0.0001
AD	NB	-0.001	1.000	No	| media(A) - media(B) | = -0.001 < GH = 1
AD	SVC	-0.057	0.608	No	| media(A) - media(B) | = -0.057 < GH = 0.608
KNN	MLP	0.494	0.0004	Si	| media(A) - media(B) | = 0.494 > GH = 0.0004
KNN	NB	-0.064	0.838	No	| media(A) - media(B) | = -0.064 < GH = 0.838
KNN	SVC	-0.120	0.316	No	| media(A) - media(B) | = -0.12 < GH = 0.316
MLP	NB	-0.559	0.0001	No	| media(A) - media(B) | = -0.559 < GH = 0.0001
MLP	SVC	-0.614	0.00006	No	| media(A) - media(B) | = -0.614 < GH = 0.00006
NB	SVC	-0.056	0.625	No	| media(A) - media(B) | = -0.056 < GH = 0.625

3. CÓDIGO FUENTE DE LA APLICACIÓN PARA DETERMINACIÓN DEL MEJOR ALGORITMO DE PREDICCIÓN DE CÁNCER DE MAMA

Detección de cáncer de seno en muestras de microarray mediante algoritmos: AD, MLP, SVC, NB, KNN

```
Propósito     : Cumplimiento del segundo objetivo de tesis de doctorado
Autor         : Luis Beltran Palma Ttito
Fecha         : 01-marzo-2023
Dataset       : Microarray de 2302 muestras, y 20546 genes
Descripción   : Predicción de cáncer de seno en datos de microarray, utilizando clasificadores:
                Naïve Bayes,K-vecinos más cercanos, Máquina de vector de soporte, Árbol de decisión, y red neuronal Perc
eptron Multicapa.
```

Librerías

In [ ]:
```
!pip install pingouin
```

In [ ]:
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# Búsqueda aleatoria de hiperparametros con cross validation
from sklearn.model_selection import RandomizedSearchCV

# Separación de datos para training y testing
from sklearn.model_selection import train_test_split

# Validación cruzada
from sklearn.model_selection import cross_val_score

# Algoritmos de ML: K-NN, AD, MLP, NB, SVM
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

# Métricas
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import ConfusionMatrixDisplay

# Acceso de Google Drive
from google.colab import drive

# Eliminación de genes poco importantes
from pandas import read_csv
# SelectKBest: Determina la importancia de variables para la predicción,
# por el método ganancia de información, la gancia de información se
# calcula por el método ANOVA, implementado en la f_classif
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif

# seaborn para graficar test
# pingouin implementa anoa, ganacia de información, tukey
import seaborn as sns
import pingouin as pg

%matplotlib inline
```

Importación de datos desde el drive

In [ ]:
```
drive.mount('/content/drive')
dataset = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/Tesis_Doctor_LBPT/Breast_microRNA.csv", header=0, sep=',')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=Tru
e).
```

Exploración de datos

In [ ]:
```
dataset.head()
```

Out[4]:

	MIMAT0010195	MIMAT0004481	MIMAT0000062	MIMAT0004482	MIMAT0000063	MIMAT0026472	MIMAT0000064	MIMAT0004484	MIMAT0000065	MIMAT0004485
0	1.441195	-0.204594	4.076867	2.095872	4.071476	-0.143009	4.364880	2.239251	4.165128	0.320514
1	0.847575	0.847575	4.444133	2.186096	6.067381	0.847575	4.098129	2.132050	3.840386	1.328136
2	0.938209	-0.190177	4.910433	2.735472	5.263752	-0.190177	4.685387	2.370218	4.773217	-0.190177
3	-0.158351	-0.158351	5.528345	2.611936	5.612961	2.100598	5.322585	2.198394	5.200062	0.831684
4	1.275486	-0.394415	4.052406	2.944940	4.617405	-0.394415	4.162038	2.138660	4.328434	0.336160

5 rows × 2542 columns

In [ ]:
```
dataset = dataset.drop(['CODIGO'], axis=1)
dataset.head()
```

Out[5]:

	MIMAT0010195	MIMAT0004481	MIMAT0000062	MIMAT0004482	MIMAT0000063	MIMAT0026472	MIMAT0000064	MIMAT0004484	MIMAT0000065	MIMAT0004485
0	1.441195	-0.204594	4.076867	2.095872	4.071476	-0.143009	4.364880	2.239251	4.165128	0.320514
1	0.847575	0.847575	4.444133	2.186096	6.067381	0.847575	4.098129	2.132050	3.840386	1.328136
2	0.938209	-0.190177	4.910433	2.735472	5.263752	-0.190177	4.685387	2.370218	4.773217	-0.190177
3	-0.158351	-0.158351	5.528345	2.611936	5.612961	2.100598	5.322585	2.198394	5.200062	0.831684
4	1.275486	-0.394415	4.052406	2.944940	4.617405	-0.394415	4.162038	2.138660	4.328434	0.336160

5 rows × 2541 columns

Dimensionalidad del dataset

In [ ]:
```
print(dataset.shape, ' muestras y genes respectivamente')
```

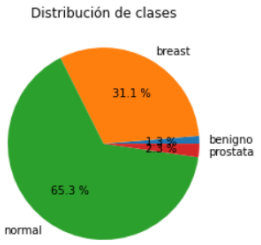
```
(4113, 2541)  muestras y genes respectivamente
```

Distribución de clases

In [ ]:
```
distribucion = dataset.groupby('ESTADO').size()
distribucion
```

Out[7]:
```
ESTADO
benigno       54
breast      1280
normal      2686
prostata      93
dtype: int64
```

In [ ]:
```
distribucion = dataset.groupby('ESTADO').size()
plt.pie(distribucion, labels = ['benigno','breast','normal','prostata'], autopct="%0.1f %%")
plt.title('Distribución de clases')
plt.show()
```



Extracción de datos de cancer de mama y muestras normales (sanas)

In [ ]:
```
dataset = dataset[(dataset['ESTADO'] == 'breast') | (dataset['ESTADO'] == 'normal')]
```

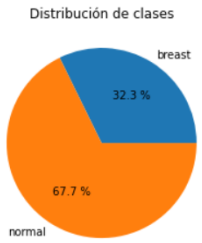
In [ ]:
```
dataset.shape
```

Out[10]:
```
(3966, 2541)
```

In [ ]:
```
distribucion = dataset.groupby('ESTADO').size()
distribucion
```

Out[11]:
```
ESTADO
breast    1280
normal    2686
dtype: int64
```

In [ ]:
```
# distribución de clases
distribucion = dataset.groupby('ESTADO').size()
plt.pie(distribucion, labels = ['breast','normal'], autopct="%0.1f %%")
plt.title('Distribución de clases')
plt.show()
```



Eliminación de datos faltantes

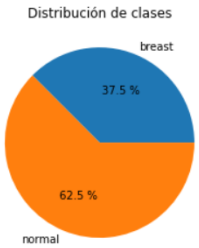
In [ ]:
```
df2 = dataset.dropna(axis=0)
dataset.shape, df2.shape
```

Out[13]:
```
((3966, 2541), (2388, 2541))
```

In [ ]:
```
distribucion = df2.groupby('ESTADO').size()
distribucion
```

Out[14]:
```
ESTADO
breast     896
normal    1492
dtype: int64
```

In [ ]:
```
distribucion = df2.groupby('ESTADO').size()
plt.pie(distribucion, labels = ['breast','normal'], autopct="%0.1f %%")
plt.title('Distribución de clases')
plt.show()
```



Grabación de datos en un archivo CSV

In [ ]:
```
df2.to_csv("/content/drive/MyDrive/Colab Notebooks/Tesis_Doctor_LBPT/Breast_microRNA_2388x2541.csv", sep=',', header=True, index=
```

In [ ]:
```
drive.mount('/content/drive')
datos = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/Tesis_Doctor_LBPT/Breast_microRNA_2388x2541.csv", sep=',',header=0)
X = datos.drop(['ESTADO'], axis=1)
y = datos['ESTADO']
```

```
Mounted at /content/drive
```

Selección de Genes importantes: Mediante Ganancia de Información y ANOVA

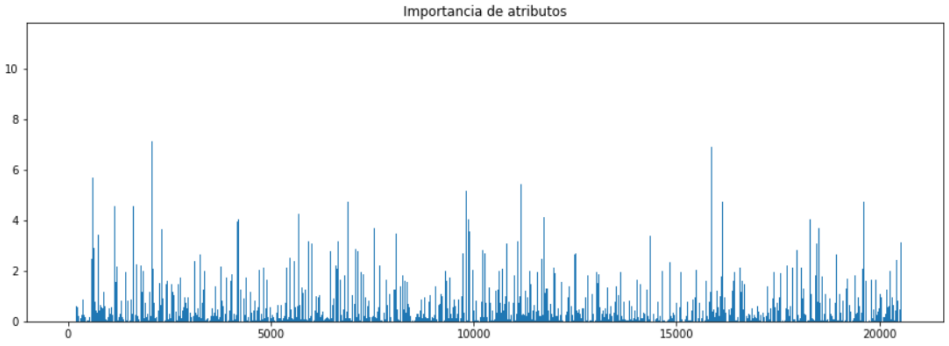
In [ ]:
```
# carga de archivo y separa en entradas X y salida y
def load_dataset(filename):
    data = read_csv(filename, header=0, sep=',')
    dataset = data.values
    X = dataset[:, :-1]
    y = dataset[:,-1]
    return X, y
```

In [ ]:
```
# Selecciona atributos importantes
def select_features(X_train, y_train, X_test, K = 100):
    # Selecciona K atributos importantes utilizando ganancia de información con ANOVA
    fs = SelectKBest(score_func=f_classif, k=K)
    fs.fit(X_train, y_train)
    # Transforma X_train, X_test, considerando solo k atributos importantes
    X_train_fs = fs.transform(X_train)
    X_test_fs = fs.transform(X_test)
    return X_train_fs, X_test_fs, fs
```

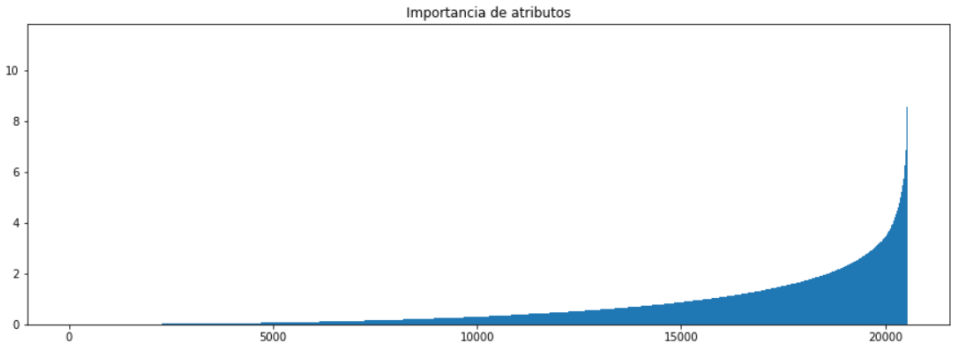
In [ ]:
```
X = dataset.drop(['ESTADO'], axis=1)
y = pd.DataFrame(dataset['ESTADO'])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=1)
X_train_fs, X_test_fs, fs = select_features(X_train, y_train, X_test, 100)
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed wh
en a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [ ]:
```
# Muestra un gráfico de barras, SIN ORDENAR
plt.figure(figsize=(15,5))
plt.bar([i for i in range(len(fs.scores_))], fs.scores_)
plt.title('Importancia de atributos')
plt.show()
```



In [ ]:
```
fs.scores_.sort()
plt.figure(figsize=(15,5))
plt.bar([i for i in range(len(fs.scores_))], fs.scores_)
plt.title('Importancia de atributos')
plt.show()
```



Identificación y eliminación de atributos poco importantes

In [ ]:
```
scores = []
for i in range(len(fs.scores_)):
    scores.append([fs.scores_[i],i])
scores.sort()
```

Genes sleccionados = genes con puntuación superior al 20% de la puntuación más alta

In [ ]:
```
scor_aux = pd.DataFrame(scores, columns=['score','fila'])
genes_selecionar = len(scor_aux[scor_aux['score'] >= max(scor_aux['score'])*0.20])
total_genes = len(scor_aux)
print('total genes = ', total_genes, ' genes a seleccionar = ', genes_selecionar)
```

```
total genes =  20545  genes a seleccionar =  1559
```

In [ ]:
```
Noimportantes = []
for i in range(total_genes - genes_selecionar-1):
    Noimportantes.append(dataset.columns[scores[i][1]])
Noimportantes = np.array(Noimportantes)
Noimportantes
```

Out[23]:
```
array(['RFC2', 'HSPA6', 'PAX8', ..., 'ZFP92', 'LINC01007', 'PABPC5'],
      dtype='<U22')
```

In [ ]:
```
#no eliminar el atributo type, ya que es la salida
Noimportantes = Noimportantes[Noimportantes != 'ESTADO']
```

In [ ]:
```
# Eliminar atributos poco importantes
dataset_ai = dataset.drop(Noimportantes, axis = 1)
dataset_ai.shape
```

Out[25]:
```
(465, 1561)
```

Grabación de datos únicamente con atributos importantes

In [ ]:
```
dataset_ai.head(1)
```

Out[27]:

	LINC01224	C20orf173	RPS11P6	GALNTL5	LOC101930081	CCDC114	KRT82	STARD13.AS	DEFB129	LOC101926892	...	CCDC63	C16orf52	LOC101929
0	4.593408	4.644979	4.433929	3.133147	3.678522	4.451129	4.769339	4.320356	2.965195	3.337486	...	4.134858	3.476486	4.5463

1 rows × 1561 columns

In [ ]:
```
dataset_ai.to_csv("/content/drive/MyDrive/Colab Notebooks/Tesis_Doctor_LBPT/Breast_465x1561_atrib.csv", sep=',', header=True, inc
```

Recuperación de dataset con atributos importantes

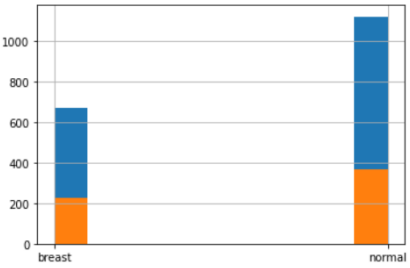
In [ ]:
```
datos = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/Tesis_Doctor_LBPT/Breast_465x1561_atrib.csv", sep=',',header=0)
X = datos.drop(['ESTADO'], axis=1)
y = datos['ESTADO']
```

Separación de datos en entrenamiento (75%) y test (25%)

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.75, random_state = 0)
```

In [ ]:
```
y_train.hist(), y_test.hist()
```

Out[10]: (<AxesSubplot:>, <AxesSubplot:>)



Predicción con K-vecinos más cercanos

Búsqueda de cuadricula para diferentes valores de hiper parametros

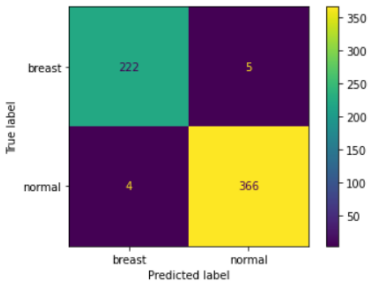
In [ ]:
```
model_knn = KNeighborsClassifier()
distribuciones = dict(n_neighbors=[2,3,4,5,6,7,8,9,10,11,12,13,14,15], weights=['uniform', 'distance'], algorithm=['auto', 'ball_
rsCV_knn = RandomizedSearchCV(estimator = model_knn, param_distributions = distribuciones, n_iter = 50, cv = 10, random_state=42)

busqueda_knn = rsCV_knn.fit(X_train,y_train)
busqueda_knn.best_params_
```

Out[20]: {'weights': 'uniform', 'n_neighbors': 2, 'algorithm': 'auto'}

Entrenamiento con hiper parametros que optimizan la solución

In [ ]:
```
model_knn = KNeighborsClassifier(weights = 'uniform', n_neighbors = 4, algorithm = 'auto')
model_knn.fit(X_train, y_train)
y_pred = model_knn.predict(X_test)

cm = confusion_matrix(y_test,y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=model_knn.classes_)
disp.plot()

accu_knn = accuracy_score(y_test, y_pred)
print('Exactitud = ', round(accu_knn * 100, 2), '%')
```

```
Exactitud =  98.49 %
```



Obtención de 10 valores de exactitud, aplicando validación cruzada, k=10

In [ ]:
```
scores_knn = cross_val_score(model_knn, X_train, y_train, cv=10)
scores_knn
```

Out[14]:
```
array([0.99444444, 0.98882682, 0.98324022, 0.97765363, 0.99441341,
       0.98324022, 0.99441341, 0.97765363, 0.97765363, 0.97765363])
```

Predicción con árbol de decisión

Búsqueda de cuadricula para diferentes valores de hiper parametros

In [ ]:
```
model_ad = DecisionTreeClassifier()
distribuciones = dict(criterion=['gini', 'entropy', 'log_loss'], splitter=['best', 'random'], max_depth=[10, 20, 30, 40, 50, 100]
rsCV_ad = RandomizedSearchCV(estimator = model_ad, param_distributions = distribuciones, n_iter = 50, cv = 10, random_state=42)
busqueda_ad = rsCV_ad.fit(X_train,y_train)
busqueda_ad.best_params_
```

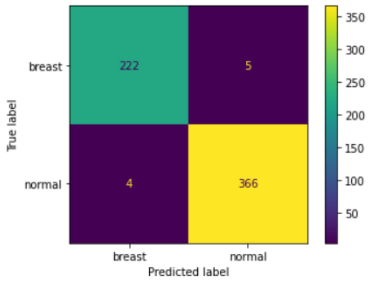
```
/usr/local/lib/python3.9/dist-packages/sklearn/model_selection/_search.py:305: UserWarning: The total space of parameters 36 is
smaller than n_iter=50. Running 36 iterations. For exhaustive searches, use GridSearchCV.
  warnings.warn(
```

Out[23]:
```
{'splitter': 'best', 'max_depth': 20, 'criterion': 'gini'}
```

Entrenamiento de árbol de decisión

In [ ]:
```
model_ad = DecisionTreeClassifier(splitter = 'best', max_depth = 20,  criterion = 'gini')
model_ad.fit(X_train, y_train)
y_pred = model_ad.predict(X_test)

cm = confusion_matrix(y_test,y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=model_ad.classes_)
disp.plot()


accu_ad = accuracy_score(y_test, y_pred)
print('Exactitud = ', round(accu_ad * 100, 2), '%')
```

```
Exactitud =  98.49 %
```



Obtención de 10 valores de exactitud, aplicando validación cruzada, k=10

In [ ]:
```
scores_ad = cross_val_score(model_ad, X_train, y_train, cv=10)
scores_ad
```

Out[16]:
```
array([0.98888889, 0.98324022, 0.97765363, 0.98324022, 0.98324022,
       0.98324022, 0.99441341, 0.99441341, 0.97765363, 0.98882682])
```

Predicción con clasificador Naïve Bayes

Entrenamiento del clasificador Naïve Bayes

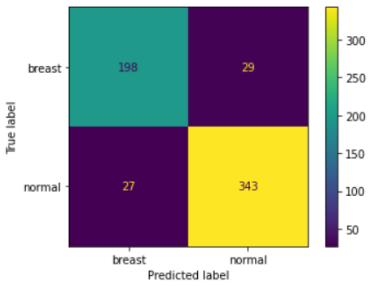
In [ ]:
```
# Naïve Bayes no posee hiperparametros
model_nb = GaussianNB()
rsCV_nb = RandomizedSearchCV(estimator = model_nb, cv=10, param_distributions={})

busqueda_nb = rsCV_nb.fit(X_train,y_train)
busqueda_nb.best_params_
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/model_selection/_search.py:305: UserWarning: The total space of parameters 1 is
smaller than n_iter=10. Running 1 iterations. For exhaustive searches, use GridSearchCV.
  warnings.warn(
```

Out[27]:
```
{}
```

In [ ]:
```
model_nb = GaussianNB()
model_nb.fit(X_train, y_train)
y_pred = model_nb.predict(X_test)

cm = confusion_matrix(y_test,y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=model_nb.classes_)
disp.plot()

accu_nb = accuracy_score(y_test, y_pred)
print('Exactitud = ', round(accu_nb * 100, 2), '%')
```

```
Exactitud =  90.62 %
```



Obtención de 10 valores de exactitud, aplicando validación cruzada, k=10

In [ ]:
```
scores_nb = cross_val_score(model_nb, X_train, y_train, cv=10)
scores_nb
```

Out[18]:
```
array([0.96111111, 0.94972067, 0.9273743 , 0.92178771, 0.91620112,
       0.91061453, 0.91061453, 0.92178771, 0.91061453, 0.90502793])
```

Predicción con máquina vector de soporte para clasificación

Búsqueda de cuadricula para diferentes valores de hiper parametros

In [ ]:
```
model_svc = SVC()
distribuciones = dict(kernel=['linear', 'poly', 'rbf', 'sigmoid'], C=[0.1, 0.2, 0.5, 0.8, 1, 2, 4, 8, 15])
rsCV_svc = RandomizedSearchCV(estimator = model_svc, param_distributions = distribuciones, n_iter = 50, cv = 10, random_state=42)
busqueda_svc = rsCV_svc.fit(X_train,y_train)
busqueda_svc.best_params_
```

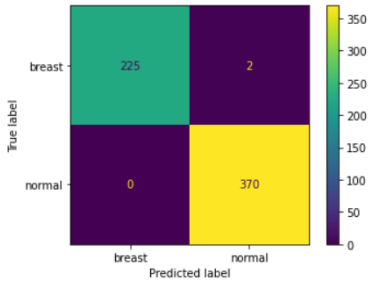
```
/usr/local/lib/python3.9/dist-packages/sklearn/model_selection/_search.py:305: UserWarning: The total space of parameters 36 is
smaller than n_iter=50. Running 36 iterations. For exhaustive searches, use GridSearchCV.
  warnings.warn(
```

Out[30]:
```
{'kernel': 'linear', 'C': 0.1}
```

Entrenamiento de máquina de vector de soporte

In [ ]:
```
model_svc = SVC(kernel = 'linear', C = 0.1)
model_svc.fit(X_train, y_train)
y_pred = model_svc.predict(X_test)

cm = confusion_matrix(y_test,y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=model_svc.classes_)
disp.plot()

accu_svc = accuracy_score(y_test, y_pred)
print('Exactitud = ', round(accu_svc * 100, 2), '%')
```

```
Exactitud =  99.66 %
```



Obtención de 10 valores de exactitud, aplicando validación cruzada, k=10

In [ ]:
```
scores_svc = cross_val_score(model_svc, X_train, y_train, cv=10)
scores_svc
```

Out[20]:
```
array([1.        , 0.99441341, 1.        , 0.98324022, 1.        ,
       1.        , 1.        , 1.        , 0.98882682, 1.        ])
```

Predicción con red neuronal artificial: perceptron multicapa

Búsqueda de cuadricula para diferentes valores de hiper parametros

In [ ]:
```
model_mlp = MLPClassifier()
distribuciones = dict(hidden_layer_sizes=[(100, 50, 10), (300, 100, 30)],
                      activation=['identity', 'logistic', 'tanh', 'relu'],
                      learning_rate=['invscaling', 'adaptive'], momentum=[0.5, 0.7, 0.9])
rsCV_mlp = RandomizedSearchCV(estimator = model_mlp, param_distributions = distribuciones, n_iter = 50, cv = 10, random_state=42)
busqueda_mlp = rsCV_mlp.fit(X_train,y_train)
busqueda_mlp.best_params_
```

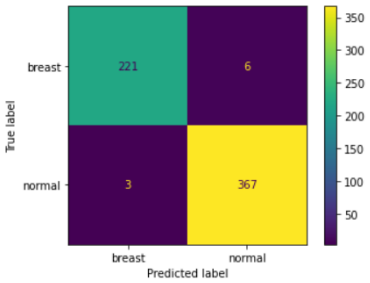
```
/usr/local/lib/python3.9/dist-packages/sklearn/model_selection/_search.py:305: UserWarning: The total space of parameters 48 is
smaller than n_iter=50. Running 48 iterations. For exhaustive searches, use GridSearchCV.
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:684: ConvergenceWarning: Stochastic Opt
imizer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
```

Out[33]:
```
{'momentum': 0.5,
 'learning_rate': 'invscaling',
 'hidden_layer_sizes': (100, 50, 10),
 'activation': 'relu'}
```

Entrenamiento de perceptrón multicapa

In [ ]:
```
model_mlp = MLPClassifier(momentum=0.5, learning_rate='invscaling', hidden_layer_sizes=(100,50, 10), activation='relu')
model_mlp.fit(X_train, y_train)
y_pred = model_mlp.predict(X_test)

cm = confusion_matrix(y_test,y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=model_mlp.classes_)
disp.plot()

accu_mlp = accuracy_score(y_test, y_pred)
print('Exactitud = ', round(accu_mlp * 100, 2), '%')
```

```
Exactitud =  98.49 %
```



Obtención de 10 valores de exactitud, aplicando validación cruzada, k=10

In [ ]:
```
scores_mlp = cross_val_score(model_mlp, X_train, y_train, cv=10)
scores_mlp
```

Out[12]:
```
array([0.96111111, 0.37430168, 0.94972067, 0.9273743 , 0.99441341,
       0.98324022, 0.92178771, 0.99441341, 0.98324022, 0.97206704])
```

Tabla comparativa de clasificadores, con exactitud de predicción de datos de test

In [ ]:
```
Resultado = pd.DataFrame()
Resultado['Clasificador'] = ['KNN', 'AD', 'SVM', 'NB', 'MLP']

#Resultado['Hiperparametros'] = [busqueda_knn.best_params_, busqueda_ad.best_params_, busqueda_svc.best_params_, busqueda_nb.best
Resultado['Exactitud'] = [accu_knn, accu_ad, accu_svc, accu_nb, accu_mlp]
print(Resultado)
```

```
  Clasificador  Exactitud
0          KNN   0.984925
1           AD   0.984925
2          SVM   0.996650
3           NB   0.906198
4          MLP   0.984925
```

In [ ]:
```
Resultado.to_csv("/content/drive/MyDrive/Colab Notebooks/Tesis_Doctor_LBPT/Breast_GS45827_config_130x28.csv", sep=',', header=Tru
```

Comparación de clasificadores, mediante ANOVA

Hipótesis

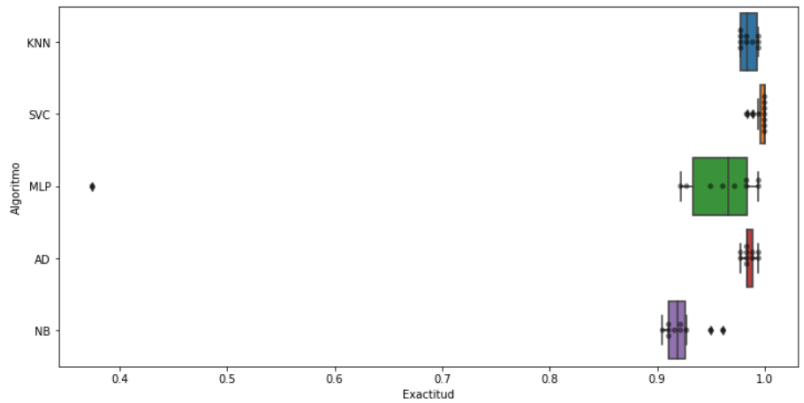
Hipótesis nula (Ho): No existe, diferencia significativa de predicción, entre los clasificadores: KNN, SVC, MLP, AD, NB
Hipótesis alterna (H1): Al menos una par de clasificadores (KNN, SVC, MLP, AD, NB) tienen diferencias significativas de predicción

Acopio de datos obtenidos con Cross Validation k=10

In [ ]:
```
datos = pd.DataFrame({'Algoritmo':np.repeat(['KNN'],10),'Exactitud':scores_knn})
dat_svc = pd.DataFrame({'Algoritmo':np.repeat(['SVC'],10),'Exactitud':scores_svc})
dat_mlp = pd.DataFrame({'Algoritmo':np.repeat(['MLP'],10),'Exactitud':scores_mlp})
dat_ad = pd.DataFrame({'Algoritmo':np.repeat(['AD'],10),'Exactitud':scores_ad})
dat_nb = pd.DataFrame({'Algoritmo':np.repeat(['NB'],10),'Exactitud':scores_nb})

datos = datos.append(dat_svc)
datos = datos.append(dat_mlp)
datos = datos.append(dat_ad)
datos = datos.append(dat_nb)
```

Análisis gráfico: por cada clasificador, tenemos un diagrama de caja de la exactitud obtenida mediante validación cruzada ¶

In [ ]:
```
fig, ax = plt.subplots(1, 1, figsize=(12, 6))
sns.boxplot(y="Algoritmo", x="Exactitud", data=datos, ax=ax)
sns.swarmplot(y="Algoritmo", x="Exactitud", data=datos, color='black', alpha = 0.5, ax=ax);
```



Acopio de exactitud de cros validation de los algoritmos KNN, NB, AD, SVC, MLP

In [ ]:
```
from numpy.lib.function_base import median
df = pd.DataFrame()
df['KNN'] = scores_knn
df['SVC'] = scores_svc
df['MLP'] = scores_mlp
df['AD'] = scores_ad
df['NB'] = scores_nb

dfp = pd.DataFrame(np.array([[median(scores_knn), median(scores_svc), median(scores_mlp), median(scores_ad), median(scores_nb)]]),
                   columns=['KNN','SVC','MLP','AD','NB'], index=['PROM'])
dfp

df = df.append(dfp)
df

df.index = ['CV = 1','CV = 2','CV = 3','CV = 4','CV = 5','CV = 6','CV = 7','CV = 8','CV = 9','CV = 10','PROMEDIO']
df
```

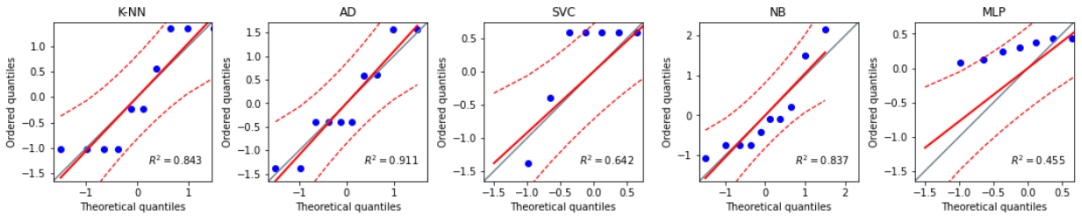
Out[32]:

	KNN	SVC	MLP	AD	NB
CV = 1	0.994444	1.000000	0.961111	0.988889	0.961111
CV = 2	0.988827	0.994413	0.374302	0.983240	0.949721
CV = 3	0.983240	1.000000	0.949721	0.977654	0.927374
CV = 4	0.977654	0.983240	0.927374	0.983240	0.921788
CV = 5	0.994413	1.000000	0.994413	0.983240	0.916201
CV = 6	0.983240	1.000000	0.983240	0.983240	0.910615
CV = 7	0.994413	1.000000	0.921788	0.994413	0.910615
CV = 8	0.977654	1.000000	0.994413	0.994413	0.921788
CV = 9	0.977654	0.988827	0.983240	0.977654	0.910615
CV = 10	0.977654	1.000000	0.972067	0.988827	0.905028
PROMEDIO	0.983240	1.000000	0.966589	0.983240	0.918994

Determinación, si los datos tienen distribución normal

Para poder utilizar ANOVA, los datos deben tener una distribución normal, se usa correlación verificar si tiene uns dist ribución normal.

In [ ]:
```
fig, axs = plt.subplots(1,5, figsize=(15, 7))
pg.qqplot(datos.loc[datos.Algoritmo=='KNN', 'Exactitud'], dist='norm', ax=axs[0])
axs[0].set_title('K-NN')
pg.qqplot(datos.loc[datos.Algoritmo=='AD', 'Exactitud'], dist='norm', ax=axs[1])
axs[1].set_title('AD')
pg.qqplot(datos.loc[datos.Algoritmo=='SVC', 'Exactitud'], dist='norm', ax=axs[2])
axs[2].set_title('SVC')
pg.qqplot(datos.loc[datos.Algoritmo=='NB', 'Exactitud'], dist='norm', ax=axs[3])
axs[3].set_title('NB')
pg.qqplot(datos.loc[datos.Algoritmo=='MLP', 'Exactitud'], dist='norm', ax=axs[4])
axs[4].set_title('MLP')
plt.tight_layout()
```



Normalidad de datos mediante la función normality

In [ ]:
```
pg.normality(data=datos, dv='Exactitud', group='Algoritmo')
```

Out[25]:

Algoritmo	W	pval	normal
KNN	0.803966	0.016200	False
SVC	0.646721	0.000200	False
MLP	0.490075	0.000003	False
AD	0.891897	0.178109	True
NB	0.836632	0.040206	False

Determinación de

In [ ]:
```
pg.homoscedasticity(data=datos, dv='Exactitud', group='Algoritmo', method='levene')
```

Out[26]:

	W	pval	equal_var
levene	1.62254	0.185085	True

Test de ANOVA

Se realiza un analisis de ANOVA, para determinar si existen diferencias significativas en la predicción que realizan los clasificadores en la fase de entrenamiento

In [ ]:
```
pg.anova(data=datos, dv='Exactitud', between='Algoritmo', detailed=True, )
```

Out[27]:

	Source	SS	DF	MS	F	p-unc	np2
0	Algoritmo	0.068425	4	0.017106	2.372513	0.066373	0.174161
1	Within	0.324457	45	0.007210	NaN	NaN	NaN

Comparación multiple post-hoc

Ya que existe diferencias significativa en la predicción de los clasificadores, realizamos el comparativo pareado median te Tukey.

In [ ]:
```
pg.pairwise_tukey(data=datos, dv='Exactitud', between='Algoritmo').round(3)
```

Out[28]:

	A	B	mean(A)	mean(B)	diff	se	T	p-tukey	hedges
0	AD	KNN	0.985	0.985	0.001	0.038	0.015	1.000	0.079
1	AD	MLP	0.985	0.906	0.079	0.038	2.089	0.243	0.569
2	AD	NB	0.985	0.923	0.062	0.038	1.633	0.485	4.358
3	AD	SVC	0.985	0.997	-0.011	0.038	-0.294	0.998	-1.780
4	KNN	MLP	0.985	0.906	0.079	0.038	2.074	0.249	0.565
5	KNN	NB	0.985	0.923	0.061	0.038	1.618	0.494	4.208
6	KNN	SVC	0.985	0.997	-0.012	0.038	-0.309	0.998	-1.657
7	MLP	NB	0.906	0.923	-0.017	0.038	-0.456	0.991	-0.124
8	MLP	SVC	0.906	0.997	-0.090	0.038	-2.383	0.139	-0.649
9	NB	SVC	0.923	0.997	-0.073	0.038	-1.927	0.318	-5.144

In [ ]:
```
pg.pairwise_ttests(dv='Exactitud', between='Algoritmo', padjust = 'holm', data = datos)
```

```
/usr/local/lib/python3.9/dist-packages/pingouin/pairwise.py:28: UserWarning: pairwise_ttests is deprecated, use pairwise_tests instead.
  warnings.warn("pairwise_ttests is deprecated, use pairwise_tests instead.", UserWarning)
```

Out[29]:

	Contrast	A	B	Paired	Parametric	T	dof	alternative	p-unc	p-corr	p-adjust	BF10	hedges
0	Algoritmo	AD	KNN	False	True	0.185199	18.0	two-sided	8.551435e-01	1.000000e+00	holm	0.402	0.079324
1	Algoritmo	AD	MLP	False	True	1.328865	18.0	two-sided	2.004932e-01	8.019729e-01	holm	0.73	0.569176
2	Algoritmo	AD	NB	False	True	10.175852	18.0	two-sided	6.824919e-09	6.142427e-08	holm	9.834e+05	4.358493
3	Algoritmo	AD	SVC	False	True	-4.156597	18.0	two-sided	5.927170e-04	4.149019e-03	holm	45.589	-1.780342
4	Algoritmo	KNN	MLP	False	True	1.319086	18.0	two-sided	2.036757e-01	8.019729e-01	holm	0.724	0.564987
5	Algoritmo	KNN	NB	False	True	9.825182	18.0	two-sided	1.170765e-08	9.366117e-08	holm	6.0e+05	4.208295
6	Algoritmo	KNN	SVC	False	True	-3.867655	18.0	two-sided	1.127852e-03	6.767112e-03	holm	27.275	-1.656583
7	Algoritmo	MLP	NB	False	True	-0.288951	18.0	two-sided	7.759203e-01	1.000000e+00	holm	0.409	-0.123763
8	Algoritmo	MLP	SVC	False	True	-1.515962	18.0	two-sided	1.468938e-01	7.344689e-01	holm	0.871	-0.649313
9	Algoritmo	NB	SVC	False	True	-12.009533	18.0	two-sided	4.982402e-10	4.982402e-09	holm	1.095e+07	-5.143890

test Welch ANOVA

Welch ANOVA es un test que se aplica a datos que no tienn distribución normal, para determinar si existen diferencias si gnificativas

In [ ]:
```
pg.welch_anova(data=datos, dv='Exactitud', between='Algoritmo')
```

Out[30]:

	Source	ddof1	ddof2	F	p-unc	np2
0	Algoritmo	4	21.686072	34.208966	4.445863e-09	0.174161

test Games Showell

Ya que existe diferencias significativa según el test Welch ANOVA, realizamos el comparativo pareado mediante Games Show ell.

In [ ]:
```
pg.pairwise_gameshowell(dv='Exactitud', between='Algoritmo', data = datos)
```

Out[31]:

	A	B	mean(A)	mean(B)	diff	se	T	df	pval	hedges
0	AD	KNN	0.985481	0.984919	0.000562	0.003033	0.185199	17.204442	0.999708	0.079324
1	AD	MLP	0.985481	0.906167	0.079314	0.059686	1.328865	9.018265	0.682060	0.569176
2	AD	NB	0.985481	0.923485	0.061996	0.006092	10.175852	10.917668	0.000005	4.358493
3	AD	SVC	0.985481	0.996648	-0.011167	0.002687	-4.156597	17.999992	0.004711	-1.780342
4	KNN	MLP	0.984919	0.906167	0.078752	0.059702	1.319086	9.028272	0.687455	0.564987
5	KNN	NB	0.984919	0.923485	0.061434	0.006253	9.825182	11.921520	0.000004	4.208295
6	KNN	SVC	0.984919	0.996648	-0.011729	0.003033	-3.867655	17.200080	0.009286	-1.656583
7	MLP	NB	0.906167	0.923485	-0.017318	0.059936	-0.288951	9.169459	0.998180	-0.123763
8	MLP	SVC	0.906167	0.996648	-0.090481	0.059686	-1.515962	9.018241	0.577854	-0.649313
9	NB	SVC	0.923485	0.996648	-0.073163	0.006092	-12.009533	10.915249	0.000001	-5.143890

4. CÓDIGO FUENTE DE LA APLICACIÓN PARA DETERMINACIÓN DEL MEJOR ENSAMBLE DE PREDICCIÓN DE CÁNCER DE MAMA

Detección de cáncer de seno en muestras de microarray mediante ensambles: voting, bagging, boosting, stacking

```
Propósito   : Cumplimiento del tercer objetivo de tesis de doctorado
Autor       : Luis Beltran Palma Ttito
Fecha       : 01-marzo-2023
Dataset     : Microarray de 2302 muestras, y 20546 genes
Descripción : Predicción de cáncer de seno en datos de microarray, utilizando clasificadores: Vooting, Bagging, Boostin
g, Stacking.
```

Librerías

In [ ]:
```
!pip install pingouin
```

In [ ]:
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import math
import random

# Búsqueda aleatoria de hiperparametros con cross validation
from sklearn.model_selection import RandomizedSearchCV

# Separación de datos para training y testing
from sklearn.model_selection import train_test_split

# Validación cruzada
from sklearn.model_selection import cross_val_score

# Algoritmos de ML: K-NN, AD, MLP, NB, SVM
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

# Métricas
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import ConfusionMatrixDisplay

# Acceso de Google Drive
from google.colab import drive

# Eliminación de genes poco importantes
from pandas import read_csv
# SelectKBest: Determina la importancia de variables para la predicción,
# por el método ganancia de información, la gancia de información se
# calcula por el método ANOVA, implementado en la f_classif
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif

# seaborn para graficar test
# pingouin implementa anoa, ganacia de información, tukey
import seaborn as sns
import pingouin as pg

%matplotlib inline
```

Importación de datos

In [ ]:
```
drive.mount('/content/drive')
datos = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/Tesis_Doctor_LBPT/Breast_microRNA_2388x2541.csv", sep=',',header=0)
X = datos.drop(['ESTADO'], axis=1)
y = pd.DataFrame(datos['ESTADO'])
datos.head()
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=Tru
e).
```

Out[84]:

	MIMAT0010195	MIMAT0004481	MIMAT0000062	MIMAT0004482	MIMAT0000063	MIMAT0026472	MIMAT0000064	MIMAT0004484	MIMAT0000065	MIMAT0004485
0	2.090734	0.227433	0.227433	3.679498	0.227433	1.263479	0.227433	2.398070	0.227433	0.227433
1	1.713455	0.355896	0.355896	3.352019	0.355896	1.818181	0.355896	2.319775	0.874377	0.355896
2	2.189441	2.189441	2.189441	2.560180	2.189441	2.189441	2.189441	2.550470	2.189441	2.189441
3	1.900954	0.666966	0.666966	2.843728	0.666966	1.426287	0.666966	2.213672	0.666966	0.666966
4	1.931519	1.069203	1.069203	2.464675	1.069203	1.597745	1.069203	2.992560	1.441521	1.069203

5 rows × 2541 columns

Dimensionalidad del dataset

In [ ]:
```
datos.shape
```

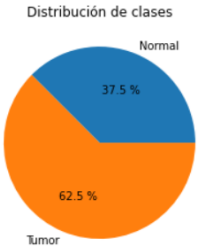
Out[4]:
```
(2388, 2541)
```

Distribución de datos

In [ ]:
```
distribucion = datos.groupby('ESTADO').size()
distribucion
```

Out[5]:
```
ESTADO
breast     896
normal    1492
dtype: int64
```

In [ ]:
```
distribucion = datos.groupby('ESTADO').size()
plt.pie(distribucion, labels = ['Normal', 'Tumor'], autopct="%0.1f %%")
plt.title('Distribución de clases')
plt.show()
```



Separación de datos para train y test

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.8, random_state = 0)
```

#Ensamble voting heterogéneo

In [ ]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.ensemble import VotingClassifier
from sklearn.metrics import classification_report
```

Definición de algoritmos de clasificación

In [ ]:
```
model_vknn = KNeighborsClassifier(weights = 'uniform', n_neighbors = 4, algorithm = 'auto')
model_vad = DecisionTreeClassifier(splitter = 'best', max_depth = 20, criterion = 'gini')
model_vnb = GaussianNB()
model_vsvc = SVC(kernel = 'linear', C = 0.1)
model_vmlp = MLPClassifier(momentum=0.5, learning_rate='invscaling', hidden_layer_sizes=(100, 50, 10), activation='relu')
```

Definición del ensamble voting

In [ ]:
```
ensamble_voting = VotingClassifier(estimators=[("K-NN", model_vknn), ("AD", model_vad), ("MLP", model_vmlp), ("NB", model_vnb), (
                                   voting="hard", weights=[1, 1, 1, 1, 1], n_jobs=-1)
ensamble_voting.fit(X_train, y_train)
```

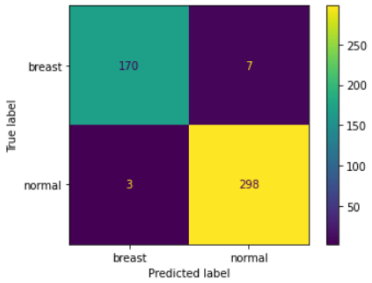
```
/usr/local/lib/python3.9/dist-packages/sklearn/preprocessing/_label.py:99: DataConversionWarning: A column-vector y was passed
when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.9/dist-packages/sklearn/preprocessing/_label.py:134: DataConversionWarning: A column-vector y was passed
when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, dtype=self.classes_.dtype, warn=True)
```

Out[10]:
```
VotingClassifier(estimators=[('K-NN', KNeighborsClassifier(n_neighbors=4)),
                             ('AD', DecisionTreeClassifier(max_depth=20)),
                             ('MLP',
                              MLPClassifier(hidden_layer_sizes=(100, 50, 10),
                                            learning_rate='invscaling',
                                            momentum=0.5)),
                             ('NB', GaussianNB()),
                             ('SVC', SVC(C=0.1, kernel='linear'))],
                 n_jobs=-1, weights=[1, 1, 1, 1, 1])
```

Entrenamiento del ensamble voting

In [ ]:
```
y_pred_voting = ensamble_voting.predict(X_test)

cm = confusion_matrix(y_test, y_pred_voting)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=ensamble_voting.classes_)
disp.plot()

accu_voting = accuracy_score(y_test, y_pred_voting)
print('Exactitud = ', round(accu_voting * 100, 2), '%')
```

```
Exactitud =  97.91 %
```



Validación cruzada k=10

In [ ]:
```
scores_voting = cross_val_score(ensamble_voting, X, y, cv=10)
scores_voting
```

Out[12]:
```
array([1.        , 1.        , 1.        , 1.        , 0.9958159 ,
       0.9958159 , 0.9874477 , 0.9958159 , 0.99159664, 0.79831933])
```

In [ ]:
```
scores_test_vooting = cross_val_score(ensamble_voting, X_test, y_test, cv=10)
scores_test_vooting
```

Out[13]:
```
array([0.97916667, 0.97916667, 1.        , 0.97916667, 0.9375    ,
       1.        , 1.        , 0.97916667, 1.        , 1.        ])
```

Ensamble Bagging heterogéneo

Separación de datos con reemplazo, en 5 dataset de igual tamaño, para crear un bagging heterogéneo

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.75, random_state = 0)
```

In [ ]:
```
# Obtiene la cantidad de datos totales, test y train de bagging
tam_bag = int(len(X_train)/5)
tam_bag
```

Out[15]:
```
358
```

In [ ]:
```
def separa(XX, yy, n):
    # reindexar para poder extrar las muestras de traing y testing
    XX.index = list(range(len(XX)))
    yy.index = list(range(len(yy)))

    # Genera datos para knn train
    print(len(XX), len(yy))
    new_X = XX[0:n]
    new_y = yy[0:n]

    # eliminar datos seleccionados
    XX = XX.drop(list(range(n)), axis=0)
    yy = yy.drop(list(range(n)), axis=0)
    print(len(XX), len(yy))

    return XX, yy, new_X, new_y
```

Datos para KNN, AD, NB, SVC, MLP

In [ ]:
```
X_train, y_train, X_train_baknn, y_train_baknn = separa(X_train, y_train, tam_bag)

X_train, y_train, X_train_baad, y_train_baad = separa(X_train, y_train, tam_bag)

X_train, y_train, X_train_banb, y_train_banb = separa(X_train, y_train, tam_bag)

X_train, y_train, X_train_basvc, y_train_basvc = separa(X_train, y_train, tam_bag)

X_train, y_train, X_train_bamlp, y_train_bamlp = separa(X_train, y_train, tam_bag)
```

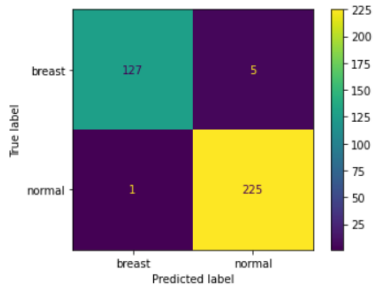
```
1791 1791
1433 1433
1433 1433
1075 1075
1075 1075
717 717
717 717
359 359
359 359
1 1
```

Entrenamos cada uno de los modelos con los dataset creados

k-vecinos más cercanos

In [ ]:
```
model_baknn = KNeighborsClassifier(weights = 'uniform', n_neighbors = 4, algorithm = 'auto')
model_baknn.fit(X_train_baknn, y_train_baknn)
y_pred_baknn = model_baknn.predict(X_train_baknn)

cm = confusion_matrix(y_train_baknn,y_pred_baknn)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=model_baknn.classes_)
disp.plot()

accu_baknn = accuracy_score(y_train_baknn, y_pred_baknn)
print('Exactitud = ', round(accu_baknn * 100, 2), '%')
```

```
Exactitud =  98.32 %
```



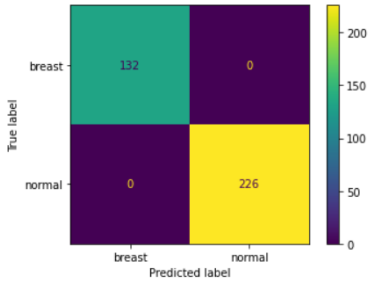
In [ ]:
```
scores_train_bagging_knn = cross_val_score(model_baknn, X_test, y_test, cv=10)
scores_train_bagging_knn
```

Out[19]:
```
array([0.96666667, 0.98333333, 0.98333333, 0.95      , 1.        ,
       0.91666667, 1.        , 0.98305085, 1.        , 0.96610169])
```

árbol de decisión

In [ ]:
```
model_baad = DecisionTreeClassifier(splitter = 'best', max_depth = 20,  criterion = 'gini')
model_baad.fit(X_train_baad, y_train_baad)
y_pred_baad = model_baad.predict(X_train_baad)

cm = confusion_matrix(y_train_baad,y_pred_baad)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=model_baad.classes_)
disp.plot()

accu_baad = accuracy_score(y_train_baad, y_pred_baad)
print('Exactitud = ', round(accu_baad * 100, 2), '%')
```

```
Exactitud =  100.0 %
```



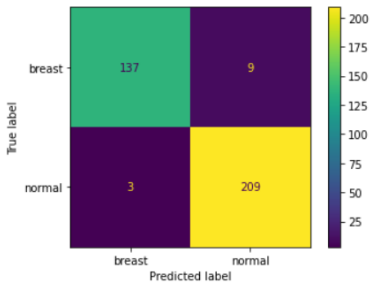
In [ ]:
```
scores_train_bagging_ad = cross_val_score(model_baad, X_test, y_test, cv=10)
scores_train_bagging_ad
```

Out[21]:
```
array([0.98333333, 0.98333333, 0.93333333, 0.96666667, 0.95      ,
       0.95      , 0.95      , 0.96610169, 0.98305085, 0.94915254])
```

red neuronal perceptron multicapa

In [ ]:
```
model_bamlp =  MLPClassifier( momentum=0.5, learning_rate='invscaling', hidden_layer_sizes=(100,50,10), activation='relu')
model_bamlp.fit(X_train_bamlp, y_train_bamlp)
y_pred_bamlp = model_bamlp.predict(X_train_bamlp)

cm = confusion_matrix(y_train_bamlp,y_pred_bamlp)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=model_bamlp.classes_)
disp.plot()

accu_bamlp = accuracy_score(y_train_bamlp, y_pred_bamlp)
print('Exactitud = ', round(accu_bamlp * 100, 2), '%')
```

```
Exactitud =  96.65 %
```



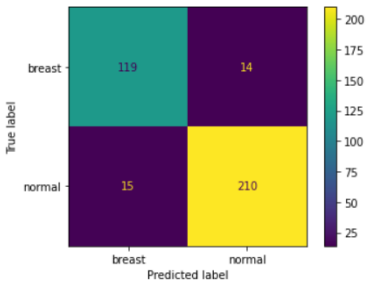
In [ ]:
```
scores_train_bagging_mlp = cross_val_score(model_bamlp, X_test, y_test, cv=10)
scores_train_bagging_mlp
```

Out[23]:
```
array([0.98333333, 0.98333333, 0.98333333, 0.95      , 1.        ,
       0.88333333, 0.96666667, 1.        , 1.        , 0.96610169])
```

clasificador Naïve Bayes

In [ ]:
```
model_banb = GaussianNB()
model_banb.fit(X_train_banb, y_train_banb)
y_pred_banb = model_banb.predict(X_train_banb)

cm = confusion_matrix(y_train_banb,y_pred_banb)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=model_banb.classes_)
disp.plot()

accu_bnb = accuracy_score(y_train_banb, y_pred_banb)
print('Exactitud = ', round(accu_bnb * 100, 2), '%')
```

```
Exactitud =  91.9 %
```



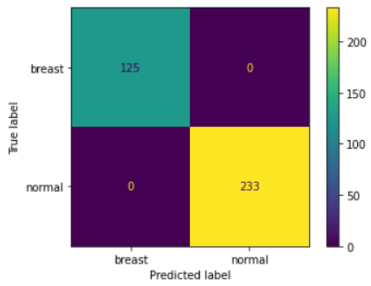
In [ ]:
```
scores_train_bagging_nb = cross_val_score(model_banb, X_test, y_test, cv=10)
scores_train_bagging_nb
```

Out[25]:
```
array([0.9       , 0.8       , 0.88333333, 0.86666667, 0.91666667,
       0.88333333, 0.98333333, 0.94915254, 0.98305085, 0.88135593])
```

máquina de vector de soporte ¶

In [ ]:
```
model_basvc = SVC(kernel = 'linear', C = 0.1)
model_basvc.fit(X_train_basvc, y_train_basvc)
y_pred_basvc = model_basvc.predict(X_train_basvc)

cm = confusion_matrix(y_train_basvc,y_pred_basvc)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=model_basvc.classes_)
disp.plot()

accu_bsvc = accuracy_score(y_train_basvc, y_pred_basvc)
print('Exactitud = ', round(accu_bsvc * 100, 2), '%')
```

```
Exactitud =  100.0 %
```



In [ ]:
```
scores_train_bagging_svc = cross_val_score(model_basvc, X_test, y_test, cv=10)
scores_train_bagging_svc
```

Out[27]:
```
array([1.        , 1.        , 1.        , 0.98333333, 1.        ,
       0.98333333, 1.        , 1.        , 1.        , 0.98305085])
```

Modelo de votación de bagging

In [ ]:
```
# hacer predicciones con todos los modelos
y_pred_baknn = model_baknn.predict(X_test)
y_pred_baad = model_baad.predict(X_test)
y_pred_bamlp = model_bamlp.predict(X_test)
y_pred_banb = model_banb.predict(X_test)
y_pred_basvc = model_basvc.predict(X_test)
```

In [ ]:
```
# Colocando los resultados de predicción en un data frame
y_pred_bagging = [y_pred_baknn, y_pred_baad, y_pred_bamlp, y_pred_banb, y_pred_basvc]
y_pred_bagging = pd.DataFrame(np.transpose(y_pred_bagging), columns=['KNN','AD','MLP','NB','SVC'])
y_pred_bagging.head()
```

Out[29]:

	KNN	AD	MLP	NB	SVC
0	breast	breast	breast	breast	breast
1	normal	normal	normal	normal	normal
2	breast	breast	breast	breast	breast
3	breast	breast	breast	breast	breast
4	normal	normal	normal	normal	normal

In [ ]:
```
# agregamos la columna Tumor, Normal y VB (votación bagging), donde colocamos,
# las predicciones de los 5 clasificadores y la predicción del método bagging heterogéneo
y_pred_bagging['Breast'] = 0
y_pred_bagging['Normal'] = 0
y_pred_bagging['VotacionBagging'] = ''
y_pred_bagging.head()
```

ut[30]:

	KNN	AD	MLP	NB	SVC	Breast	Normal	VotacionBagging
0	breast	breast	breast	breast	breast	0	0	
1	normal	normal	normal	normal	normal	0	0	
2	breast	breast	breast	breast	breast	0	0	
3	breast	breast	breast	breast	breast	0	0	
4	normal	normal	normal	normal	normal	0	0	

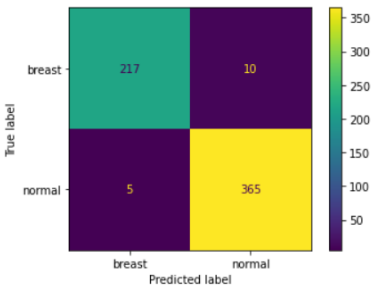
In [ ]:
```
# determinamos la frecuencia de Tumor y Normal de los 5 clasificadores
# y la salida del bagging heterogéneo
for i in y_pred_bagging.index:
    y_pred_bagging.loc[i, 'Breast'] = (y_pred_bagging.iloc[i] == 'breast').sum()
    y_pred_bagging.loc[i, 'Normal'] = (y_pred_bagging.iloc[i] == 'normal').sum()
    y_pred_bagging.loc[i, 'VotacionBagging'] = 'breast' if y_pred_bagging.iloc[i, 5] >= 3 else 'normal'
y_pred_bagging.head()
```

Out[31]:

	KNN	AD	MLP	NB	SVC	Breast	Normal	VotacionBagging
0	breast	breast	breast	breast	breast	5	0	breast
1	normal	normal	normal	normal	normal	0	5	normal
2	breast	breast	breast	breast	breast	5	0	breast
3	breast	breast	breast	breast	breast	5	0	breast
4	normal	normal	normal	normal	normal	0	5	normal

In [ ]:
```
# Ahora obtener las métricas del MODELO BAGGING heterogéneo propuesto por LBPT
cm = confusion_matrix(y_test, y_pred_bagging['VotacionBagging'])
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=model_basvc.classes_)
disp.plot()

accu_bagging = accuracy_score(y_test, y_pred_bagging['VotacionBagging'])
print('Exactitud = ', round(accu_bagging * 100, 2), '%')
```

```
Exactitud =  97.49 %
```



In [ ]:
```
df = pd.DataFrame()
df['knn'] = scores_train_bagging_knn
df['ad'] = scores_train_bagging_ad
df['mlp'] = scores_train_bagging_mlp
df['svc'] = scores_train_bagging_svc
df['bn'] = scores_train_bagging_nb
df['PROM'] = (df['knn'] + df['ad'] + df['mlp'] + df['svc'] + df['bn'])/5.0
df
```

Out[33]:

	knn	ad	mlp	svc	bn	PROM
0	0.966667	0.983333	0.983333	1.000000	0.900000	0.966667
1	0.983333	0.983333	0.983333	1.000000	0.800000	0.950000
2	0.983333	0.933333	0.983333	1.000000	0.883333	0.956667
3	0.950000	0.966667	0.950000	0.983333	0.866667	0.943333
4	1.000000	0.950000	1.000000	1.000000	0.916667	0.973333
5	0.916667	0.950000	0.883333	0.983333	0.883333	0.923333
6	1.000000	0.950000	0.966667	1.000000	0.983333	0.980000
7	0.983051	0.966102	1.000000	1.000000	0.949153	0.979661
8	1.000000	0.983051	1.000000	1.000000	0.983051	0.993220
9	0.966102	0.949153	0.966102	0.983051	0.881356	0.949153

In [ ]:
```
scores_bagging = np.array((df['knn'] + df['ad'] + df['mlp'] + df['svc'] + df['bn'])/5.0)
scores_bagging
```

Out[34]:
```
array([0.96666667, 0.95      , 0.95666667, 0.94333333, 0.97333333,
       0.92333333, 0.98      , 0.97966102, 0.99322034, 0.94915254])
```

Ensamble Boosting

Carga de datos

In [ ]:
```
drive.mount('/content/drive')
datos = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/Tesis_Doctor_LBPT/Breast_microRNA_2388x2541.csv", sep=',',header=0)
X = datos.drop(['ESTADO'], axis=1)
y = pd.DataFrame(datos['ESTADO'])
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=Tru
e).
```

Conversión de atributos de salida de categóricos a numéricos

In [ ]:
```
# Se convierte para aplicar el algoritmo AdaBoost
y[y['ESTADO'] == 'normal'] = 1
y[y['ESTADO'] == 'breast'] = -1
```

In [ ]:
```
y = y['ESTADO'].astype('int32')
```

In [ ]:
```
y = pd.DataFrame(y,columns=['ESTADO'])
```

###A. Genera pesos iniciales, todos reciben 1/n, donde n es la cantidad de datos

In [ ]:
```
def peso_ini(n):
    DD = pd.DataFrame()
    DD['N'] = range(n)
    DD['P'] = np.full(n,1.0/n)
    DD['D'] = np.zeros(n)
    DD.iloc[0,2] = DD.iloc[0,1]
    for i in range(n-1):
        DD.iloc[i+1,2] = DD.iloc[i+1,1] + DD.iloc[i,2]
    return DD
```

###B. Recupera datos para entrenar con boosting, datos recuperados con transformada inversa en base a una districución de pesos

In [ ]:
```
# retornar cm muestras aleatorias de XX con una distribución DD
def muestras_train_boosting(XX, yy, PP, cm):
    elegidos = np.array([])
    for i in range(cm):
        azar = random.random()
        k = 0
        while(PP.iloc[k,2] < azar):
            k = k + 1
        elegidos = np.append(elegidos,k)
        elegidos = elegidos.astype('int32')
    return XX.iloc[elegidos], yy.iloc[elegidos]
```

Modulo que entrena un algoritmo

In [ ]:
```
def modelo_boosting(XX_train, yy_train, algo='ad', OBJETIVO='ESTADO'):
    modelo = DecisionTreeClassifier(splitter = 'best', max_depth = 20,  criterion = 'gini')
    if (algo == 'knn'):
        modelo = KNeighborsClassifier(weights='uniform', n_neighbors = 4, algorithm = 'auto')
    elif(algo == 'mlp'):
        modelo = MLPClassifier(momentum=0.5, learning_rate='invscaling', hidden_layer_sizes=(100,50,10), activation='relu')
    elif(algo == 'svc'):
        modelo = SVC(kernel = 'linear', C = 0.1)
    elif(algo == 'nb'):
        modelo = GaussianNB()

    modelo.fit(XX_train, yy_train)
    pred = modelo.predict(XX_train)
    mc = confusion_matrix(yy_train, pred)
    accu = modelo.score(XX_train, yy_train)
    pred = pd.DataFrame(pred, index = XX_train.index, columns=[OBJETIVO])

    return mc, accu, pred, modelo
```

Módulo que actualiza pesos para la siguiente iteración

In [ ]:
```
def actualiza_pesos(mcc, PP, yy_pred, yy_train, OBJETIVO='ESTADO'):

    # Error=confusiones/total_casos = (FP+FN)/(FP+VP+FN+VN)
    error = (np.sum(mcc) - np.sum(np.diag(mcc)))/np.sum(mcc)

    # alfa
    alfa = 0.0
    if (error == 0):
        alfa = 20.0
    elif (error == 1):
        alfa = -20.0
    elif (error > 0 and error < 1):
        alfa = (1/2) * math.log((1-error)/error, math.exp(1))

    # ACTUALIZAR PESOS
    # 0. Copia pesos
    new_PP = PP.copy()
    # 1. Actualizar Probabilidad
    for indice in new_PP.index:
        if(indice in yy_train.index):
            ## predicción binaria correcta

            cant = len(yy_train.index[yy_train.index == indice])
            predi = yy_pred[OBJETIVO][indice]
            yy = yy_train[OBJETIVO][indice]

            if (cant > 1):
                predi = (yy_pred[OBJETIVO][indice]).iloc[0]
                yy = (yy_train[OBJETIVO][indice]).iloc[0]

            new_PP.iloc[lambda x: x.index == indice, 1] = float(new_PP['P'][indice] * math.exp((-alfa) * predi * yy))

        else:
            # no fue seleccionado para entrenar, se asigna como si fuese incorrecto, para que tenga probabilidad de ser elegido
            new_PP.iloc[lambda x: x.index == indice, 1] = new_PP['P'][indice] * math.exp(alfa)

    # 2. Normalizar pesos
    suma = new_PP['P'].sum()
    for k in range(len(new_PP)):
        new_PP.iloc[k, 1] = new_PP.iloc[k, 1]/suma

    # 3. Actualizar Distribución
    new_PP.iloc[0, 2] = new_PP.iloc[0,1]
    for k in range(len(new_PP)-1):
        new_PP.iloc[k+1, 2] = new_PP.iloc[k, 2] + new_PP.iloc[k+1, 1]

    return new_PP, error, alfa
```

Modulo que implementa boorting heterogéneo

In [ ]:
```
def boosting_lpt(XX, yy, lista_algo):
    n = len(XX)
    peso = peso_ini(n)

    rpta = np.ones(n)

    for algo in lista_algo:
        X_tra, y_tra = muestras_train_boosting(XX,yy,peso,n)
        mc, acu, y_pre, model = modelo_boosting(X_tra, y_tra, algo)
        p1, err, alfa = actualiza_pesos(mc, peso, y_pre, y_tra)

        pre = model.predict(XX)
        rpta = rpta + alfa*pre

    rpta[rpta >= 0] = 1
    rpta[rpta < 0] = -1
    rpta = list(rpta.astype('int32'))

    mc = confusion_matrix(yy, rpta)
    accu = np.sum(np.diag(mc))/np.sum(mc)

    return mc, accu
```

Definición de la secuencia de algoritmos en el modelo boosting y ejecución del algoritmo boosting

In [ ]:
```
algos = ['nb', 'svc', 'mlp', 'ad', 'knn']
MC, ACU = boosting_lpt(X,y, algos)
print(MC)
print(ACU)
```

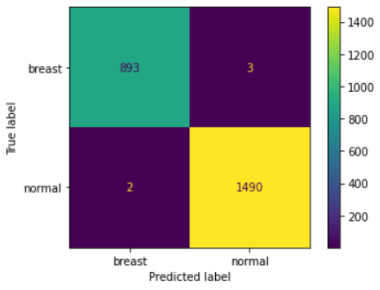
```
/usr/local/lib/python3.9/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed wh
en a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.9/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed wh
en a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.9/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:1096: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.9/dist-packages/sklearn/neighbors/_classification.py:215: DataConversionWarning: A column-vector y was p
assed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  return self._fit(X, y)
```

```
[[ 893    3]
 [   2 1490]]
0.9979061976549414
```

Métricas de boosting

In [ ]:
```
disp = ConfusionMatrixDisplay(confusion_matrix=MC, display_labels=model_basvc.classes_)
disp.plot()
print('Exactitud = ', round(ACU * 100, 2), '%')
```

```
Exactitud =  99.79 %
```



Predicción del meta boosting

In [ ]:
```
def meta_boosting(XX, yy, alf1, alf2, alf3, alf4, alf5, model1, model2, model3, model4, model5):
    pre1 = model1.predict(XX)
    pre2 = model2.predict(XX)
    pre3 = model3.predict(XX)
    pre4 = model4.predict(XX)
    pre5 = model5.predict(XX)
    rpta = alf1*pre1 + alf2*pre2 + alf3*pre3 + alf4*pre4 + alf5*pre5

    rpta[rpta >= 0] = 1
    rpta[rpta < 0] = -1
    rpta = list(rpta.astype('int32'))

    mc = confusion_matrix(yy, rpta)
    accu = np.sum(np.diag(mc))/np.sum(mc)

    return mc, accu
```

In [ ]:
```
mc, acu = meta_boosting(X, y, alf1, alf2, alf3, alf4, alf5, model1, model2, model3, model4, model5)
```

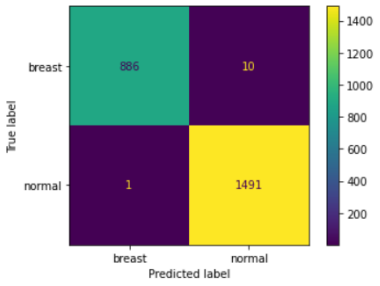
In [ ]:
```
mc, acu
```

Out[61]:
```
(array([[ 886,   10],
        [   1, 1491]]), 0.9953936348408711)
```

Exactitud conseguida con el meta bosting en tarea de predicción

In [ ]:
```
disp = ConfusionMatrixDisplay(confusion_matrix=mc, display_labels=model_basvc.classes_)
disp.plot()
print('Exactitud = ', round(acu * 100, 2), '%')
```

```
Exactitud =  99.54 %
```



Ensamble Stacking

Separación de datos

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.75, random_state = 0)
```

Configuración del modelo stacking

In [ ]:
```
from sklearn.ensemble import RandomForestClassifier

estimadores = [('KNN', KNeighborsClassifier(weights = 'uniform', n_neighbors = 4, algorithm = 'auto')),
               ('AD', DecisionTreeClassifier(splitter = 'best',  max_depth = 20,  criterion = 'gini')),
               ('SVC', SVC(kernel = 'linear', C = 0.1)),
               ('MLP', MLPClassifier(learning_rate='invscaling', momentum=0.5, hidden_layer_sizes=(100,50,10), activation='relu')),
               ('NB', GaussianNB())]

estimador_final = RandomForestClassifier(n_estimators = 500,  max_depth = 50,  criterion = 'gini')
```

Entrenamiento del modelo stacking

In [ ]:
```
from sklearn.ensemble import StackingClassifier

EnsambleStacking = StackingClassifier(
    estimators=estimadores,
    final_estimator=estimador_final, cv=10, passthrough=True)

EnsambleStacking.fit(X_train, y_train)

print('Score de training: ', EnsambleStacking.score(X_train, y_train))
print('Score de testing : ', EnsambleStacking.score(X_test, y_test))
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/preprocessing/_label.py:99: DataConversionWarning: A column-vector y was passed
when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.9/dist-packages/sklearn/preprocessing/_label.py:134: DataConversionWarning: A column-vector y was passed
when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, dtype=self.classes_.dtype, warn=True)
```

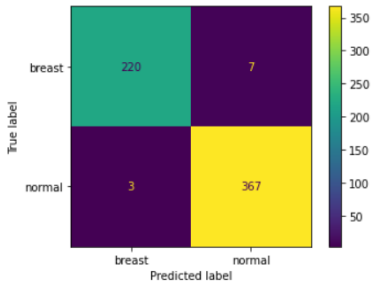
```
Score de training:  1.0
Score de testing :  0.983249581239531
```

In [ ]:
```
y_pred_stacking = EnsambleStacking.predict(X_test)
```

Exactitud conseguida con stacking

In [ ]:
```
cm = confusion_matrix(y_test, y_pred_stacking)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=model_basvc.classes_)
disp.plot()

accu_stacking = accuracy_score(y_test, y_pred_stacking)
print('Exactitud = ', round(accu_stacking * 100, 2), '%')
```

```
Exactitud =  98.32 %
```



In [ ]:
```
scores_stacking = cross_val_score(EnsambleStacking, X_train, y_train, cv=10)
scores_stacking
```

Out[70]:
```
array([1.        , 0.98882682, 1.        , 0.98324022, 1.        ,
       0.99441341, 1.        , 1.        , 0.98882682, 0.98882682])
```

Acopio de Datos

In [ ]:
```
datos = pd.DataFrame({'Algoritmo':np.repeat(['Voting'],10),'Exactitud':sc_voti})
dat_bagging = pd.DataFrame({'Algoritmo':np.repeat(['Bagging'],10),'Exactitud':sc_bagg})
dat_boosting = pd.DataFrame({'Algoritmo':np.repeat(['Boosting'],10),'Exactitud':sc_boot})
dat_stacking = pd.DataFrame({'Algoritmo':np.repeat(['Stacking'],10),'Exactitud':sc_stac})

datos = datos.append(dat_bagging)
datos = datos.append(dat_boosting)
datos = datos.append(dat_stacking)
datos
```

Out[72]:

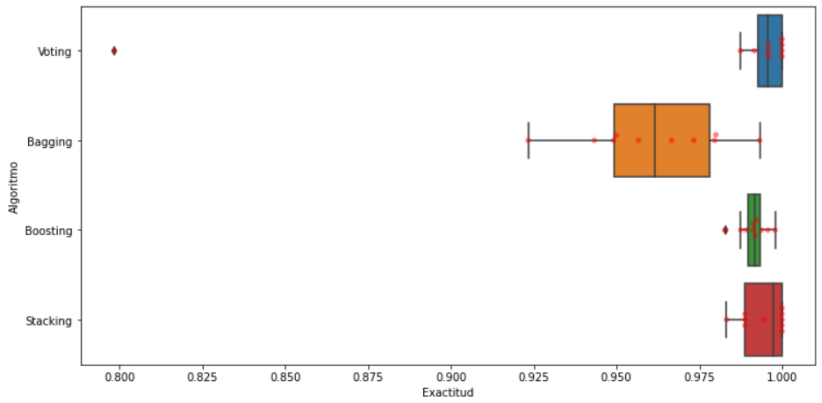
	Algoritmo	Exactitud
0	Voting	1.000000
1	Voting	1.000000
2	Voting	1.000000
3	Voting	1.000000

In [ ]:
```
from numpy.lib.function_base import median
df = pd.DataFrame()
df['Voting'] = sc_voti
df['Bagging'] = sc_bagg
df['Boosting'] = sc_boot
df['Stacking'] = sc_stac


dfp = pd.DataFrame(np.array([[median(sc_voti), median(sc_bagg), median(sc_boot), median(sc_stac)]]),
                   columns=['Voting','Bagging','Boosting','Stacking'], index=['PROM'])
dfp

df = df.append(dfp)
df

df.index = ['CV = 1','CV = 2','CV = 3','CV = 4','CV = 5','CV = 6','CV = 7','CV = 8','CV = 9','CV = 10','PROMEDIO']
df
```

Out[73]:

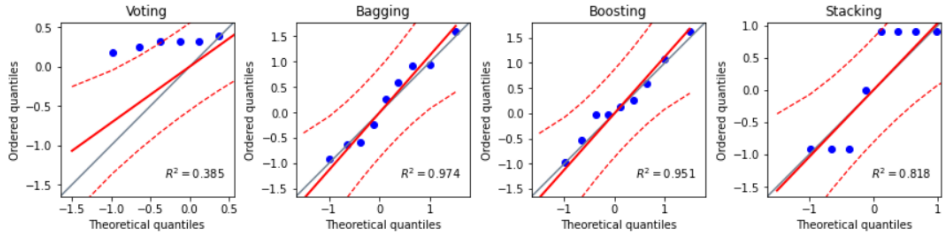
	Voting	Bagging	Boosting	Stacking
CV = 1	1.000000	0.966667	0.997906	1.000000
CV = 2	1.000000	0.950000	0.987506	0.988827
CV = 3	1.000000	0.956667	0.982736	1.000000
CV = 4	1.000000	0.943333	0.991265	0.983240
CV = 5	0.995816	0.973333	0.991296	1.000000
CV = 6	0.995816	0.923333	0.995692	0.994413
CV = 7	0.987448	0.980000	0.993754	1.000000
CV = 8	0.995816	0.979661	0.991879	1.000000
CV = 9	0.991597	0.993220	0.989275	0.988827
CV = 10	0.798319	0.949153	0.992474	0.988827
PROMEDIO	0.995816	0.961667	0.991588	0.997207

In [ ]:
```
fig, ax = plt.subplots(1, 1, figsize=(12, 6))
sns.boxplot(y="Algoritmo", x="Exactitud", data=datos, ax=ax)
sns.swarmplot(y="Algoritmo", x="Exactitud", data=datos, color='red', alpha = 0.5, ax=ax);
```



Distribución normal

In [ ]:
```
fig, axs = plt.subplots(1,4, figsize=(12, 7))
pg.qqplot(datos.loc[datos.Algoritmo=='Voting', 'Exactitud'], dist='norm', ax=axs[0])
axs[0].set_title('Voting')
pg.qqplot(datos.loc[datos.Algoritmo=='Bagging', 'Exactitud'], dist='norm', ax=axs[1])
axs[1].set_title('Bagging')
pg.qqplot(datos.loc[datos.Algoritmo=='Boosting', 'Exactitud'], dist='norm', ax=axs[2])
axs[2].set_title('Boosting')
pg.qqplot(datos.loc[datos.Algoritmo=='Stacking', 'Exactitud'], dist='norm', ax=axs[3])
axs[3].set_title('Stacking')
plt.tight_layout()
```



In [ ]:
```
pg.normality(data=datos, dv='Exactitud', group='Algoritmo')
# W: Estadistica de prueba
# pval: valor p
# normal: Verdadero si el dato se distribuye normalmente
```

Out[76]:

Algoritmo	W	pval	normal
Voting	0.421206	4.388888e-07	False
Bagging	0.974528	9.293644e-01	True
Boosting	0.966226	8.538195e-01	True
Stacking	0.797454	1.350131e-02	False

test ANOVA

In [ ]:
```
pg.anova(data=datos, dv='Exactitud', between='Algoritmo', detailed=True)
# source: nombre de factores
# SS: suma de cuadrado
# DF: Grados de libertad
# MS: cuadrado medios
# F: valores F
# p-unc: valores p no corregidos
# np2: tamaños parciales del eta-cuadrado
```

Out[77]:

	Source	SS	DF	MS	F	p-unc	np2
0	Algoritmo	0.006869	3	0.002290	2.065686	0.122	0.14686
1	Within	0.039901	36	0.001108	NaN	NaN	NaN

test Tukey

In [ ]:
```
pg.pairwise_tukey(data=datos, dv='Exactitud', between='Algoritmo').round(3)
# A: nombre del primer grupo
# B: nombre del segundo grupo
# mean(A): media del grupo A
# mean(B): media del grupo B
# diff: Diferencia de medias (= media(A) - media(B))
# se: Error estándar
# T: valores T
# p-tukey: Tukey-HSD corregido p-valores
# hedges: Tamaño del efecto de las coberturas (o cualquier tamaño del efecto definido en effsize)

# En nuestro ejemplo existe diferencias significativas entre:
# AD - MLP  : 0.496 > 0.000
# KNN - MLP : 0.443 > 0.000
```

Out[78]:

	A	B	mean(A)	mean(B)	diff	se	T	p-tukey	hedges
0	Bagging	Boosting	0.962	0.991	-0.030	0.015	-2.004	0.205	-1.893
1	Bagging	Stacking	0.962	0.994	-0.033	0.015	-2.208	0.140	-2.034
2	Bagging	Voting	0.962	0.976	-0.015	0.015	-1.004	0.748	-0.306
3	Boosting	Stacking	0.991	0.994	-0.003	0.015	-0.204	0.997	-0.532
4	Boosting	Voting	0.991	0.976	0.015	0.015	1.001	0.750	0.321
5	Stacking	Voting	0.994	0.976	0.018	0.015	1.204	0.628	0.385

test Welch ANOVA

In [ ]:
```
pg.welch_anova(data=datos, dv='Exactitud', between='Algoritmo')
```

Out[79]:

	Source	ddof1	ddof2	F	p-unc	np2
0	Algoritmo	3	17.691898	7.166715	0.00237	0.14686

test Games Showell

In [ ]:
```
pg.pairwise_gameshowell(dv='Exactitud', between='Algoritmo', data = datos)
```

Out[80]:

	A	B	mean(A)	mean(B)	diff	se	T	df	pval	hedges
0	Bagging	Boosting	0.961537	0.991378	-0.029842	0.006751	-4.420137	9.740917	0.006281	-1.893221
1	Bagging	Stacking	0.961537	0.994413	-0.032877	0.006924	-4.748488	10.695788	0.003074	-2.033860
2	Bagging	Voting	0.961537	0.976481	-0.014944	0.020914	-0.714578	10.977425	0.889255	-0.306066
3	Boosting	Stacking	0.991378	0.994413	-0.003035	0.002443	-1.242547	15.571087	0.610497	-0.532204
4	Boosting	Voting	0.991378	0.976481	0.014897	0.019885	0.749173	9.082540	0.874824	0.320884
5	Stacking	Voting	0.994413	0.976481	0.017932	0.019944	0.899128	9.190281	0.805698	0.385112

Universidad Nacional del Altiplano Puno | Vicerrectorado de Investigación | Repositorio Institucional

DECLARACIÓN JURADA DE AUTENTICIDAD DE TESIS

Por el presente documento, Yo Luis Beltran Palma Ttito,
identificado con DNI 23949672 en mi condición de egresado de:

☐ **Escuela Profesional,** ☐ **Programa de Segunda Especialidad,** ☒ **Programa de Maestría o Doctorado**

CIENCIAS DE LA COMPUTACIÓN,

informo que he elaborado el/la ☒ **Tesis** o ☐ **Trabajo de Investigación** denominada:

"Predicción de Cáncer en expresiones Genéticas de Microarrays
Mediante un Ensamble de Modelos Heterogeneos de Machine Learning"

Es un tema original.

Declaro que el presente trabajo de tesis es elaborado por mi persona y **no existe plagio/copia** de ninguna naturaleza, en especial de otro documento de investigación (tesis, revista, texto, congreso, o similar) presentado por persona natural o jurídica alguna ante instituciones académicas, profesionales, de investigación o similares, en el país o en el extranjero.

Dejo constancia que las citas de otros autores han sido debidamente identificadas en el trabajo de investigación, por lo que no asumiré como suyas las opiniones vertidas por terceros, ya sea de fuentes encontradas en medios escritos, digitales o Internet.

Asimismo, ratifico que soy plenamente consciente de todo el contenido de la tesis y asumo la responsabilidad de cualquier error u omisión en el documento, así como de las connotaciones éticas y legales involucradas.

En caso de incumplimiento de esta declaración, me someto a las disposiciones legales vigentes y a las sanciones correspondientes de igual forma me someto a las sanciones establecidas en las Directivas y otras normas internas, así como las que me alcancen del Código Civil y Normas Legales conexas por el incumplimiento del presente compromiso

Puno 25 de Enero del 2024

FIRMA (obligatoria) Huella

Universidad Nacional del Altiplano Puno | Vicerrectorado de Investigación | Repositorio Institucional

AUTORIZACIÓN PARA EL DEPÓSITO DE TESIS O TRABAJO DE INVESTIGACIÓN EN EL REPOSITORIO INSTITUCIONAL

Por el presente documento, Yo Luis Beltran Palma Ttito,
identificado con DNI 23949672 en mi condición de egresado de:

☐ **Escuela Profesional,** ☐ **Programa de Segunda Especialidad,** ☒ **Programa de Maestría o Doctorado**

CIENCIAS DE LA COMPUTACIÓN,

informo que he elaborado el/la ☒ **Tesis** o ☐ **Trabajo de Investigación** denominada:

" Predicción de Cáncer en Expresiones Genéticas de Microarrays
Mediante un Ensamble de Modelos Heterogéneos de Machine Learning
"

para la obtención de ☒**Grado,** ☐ **Título Profesional** o ☐ **Segunda Especialidad.**

Por medio del presente documento, afirmo y garantizo ser el legítimo, único y exclusivo titular de todos los derechos de propiedad intelectual sobre los documentos arriba mencionados, las obras, los contenidos, los productos y/o las creaciones en general (en adelante, los "Contenidos") que serán incluidos en el repositorio institucional de la Universidad Nacional del Altiplano de Puno.

También, doy seguridad de que los contenidos entregados se encuentran libres de toda contraseña, restricción o medida tecnológica de protección, con la finalidad de permitir que se puedan leer, descargar, reproducir, distribuir, imprimir, buscar y enlazar los textos completos, sin limitación alguna.

Autorizo a la Universidad Nacional del Altiplano de Puno a publicar los Contenidos en el Repositorio Institucional y, en consecuencia, en el Repositorio Nacional Digital de Ciencia, Tecnología e Innovación de Acceso Abierto, sobre la base de lo establecido en la Ley N° 30035, sus normas reglamentarias, modificatorias, sustitutorias y conexas, y de acuerdo con las políticas de acceso abierto que la Universidad aplique en relación con sus Repositorios Institucionales. Autorizo expresamente toda consulta y uso de los Contenidos, por parte de cualquier persona, por el tiempo de duración de los derechos patrimoniales de autor y derechos conexos, a título gratuito y a nivel mundial.

En consecuencia, la Universidad tendrá la posibilidad de divulgar y difundir los Contenidos, de manera total o parcial, sin limitación alguna y sin derecho a pago de contraprestación, remuneración ni regalía alguna a favor mío; en los medios, canales y plataformas que la Universidad y/o el Estado de la República del Perú determinen, a nivel mundial, sin restricción geográfica alguna y de manera indefinida, pudiendo crear y/o extraer los metadatos sobre los Contenidos, e incluir los Contenidos en los índices y buscadores que estimen necesarios para promover su difusión.

Autorizo que los Contenidos sean puestos a disposición del público a través de la siguiente licencia:

Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional. Para ver una copia de esta licencia, visita: https://creativecommons.org/licenses/by-nc-sa/4.0/

En señal de conformidad, suscribo el presente documento.

Puno 25 de Enero del 2024

FIRMA (obligatoria) Huella