



UNIVERSIDAD DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA QUÍMICA, BIOTECNOLOGÍA Y
MATERIALES

***VHL-Hunter, Servicio web de clasificación de
mutaciones en VHL***

DAVID ALFREDO MEDINA ORTIZ

Profesor Tutor: ÁLVARO OLIVERA NAPPA, PH.D.

Resumen Desarrollo de Proyecto,
Seminario 02

Santiago – Chile
Julio, 2018

TABLA DE CONTENIDOS

	página
Tabla de Contenidos	
Índice de Figuras	III
Índice de Tablas	v
Resumen	vi
1. INTRODUCCIÓN	7
1.1. Von Hippel-Lindau	7
1.2. Marco Teórico	10
1.2.1. Minería de Datos	10
1.2.2. Aprendizaje de Máquinas	12
1.2.3. Sistemas de Información y Modelo Vista Controlador	52
2. Hipótesis	54
3. Objetivos	56
4. Metodología y Herramientas	57
4.1. Preparación del Set de Datos	57
4.2. Análisis estadísticos del set de datos	58
4.3. Entrenamiento de modelos de clasificación	59
4.4. Adición de Información Topológica	60
4.5. Evaluación de Modelos	61
4.6. Selección de Modelos	61
4.7. Validación de las divisiones generadas	62
4.8. Unión de divisiones generadas	63
4.9. Implementación de Meta-Learning	63
4.10. Aplicación de Clúster para la búsqueda de sub grupos en el set de datos . .	64
4.11. División inducida por clustering recursivo	65
4.12. Generación de Grafos	65
4.12.1. Cálculo de energías de Enlaces Covalente	66
4.12.2. Cálculo de energías de Enlaces de Hidrógeno	67
4.12.3. Generación de Matriz de Adyacencia	67

4.12.4. Generación y Visualización del grafo	67
4.13. Desarrollo de estructura de proyecto, Diseño de Módulos y Ejecuciones Remotas	67
4.14. Herramientas Utilizadas	69
5. Resultados Parciales	71
5.1. Set de datos sin divisiones	71
5.1.1. Análisis Estadístico	71
5.1.2. Entrenamiento de Clasificadores	75
5.1.3. Selección de Modelos	75
5.1.4. Análisis de Características	81
5.2. Divisiones inducidas por sectores de interacción Proteína-Proteína	83
5.3. Divisiones inducidas por clúster	85
5.3.1. Generación de 2 particiones	85
5.3.2. Generación de 3 particiones	85
5.3.3. Generación de 4 particiones	86
5.3.4. Generación de 5 particiones	87
5.3.5. Selección de Particiones	88
5.4. Divisiones inducidas por comunidades	91
5.5. Divisiones inducidas por recursividad de clúster	92
5.6. Comentarios finales sobre los resultados obtenidos	94
6. Discusiones	95
6.1. Modelos en Set de Datos Completo.	95
6.2. Modelos en base a Set de Datos con particiones inducidas por Interacción Proteína-Proteína	99
6.3. Modelos en base a Set de Datos con particiones inducidas por Clustering .	102
6.4. Modelos en base a Set de Datos con particiones inducidas por Recursividad de Clustering	103
6.5. Evaluación de comunidades.	105
6.6. Selección de particiones	106
6.7. Unión de Sectores en Particiones inducidas por Interacción Proteína-Proteína	108
7. Conclusiones sobre resultados parciales	110
7.1. Modelos en Set de Datos Completo.	110
7.2. Modelos en base a Set de Datos con particiones inducidas por Interacción Proteína-Proteína	111

7.3. Modelos en base a Set de Datos con particiones inducidas por Clustering	111
7.4. Modelos en base a Set de Datos con particiones inducidas por Recursividad de Clustering	112
7.5. Evaluación de comunidades.	112
7.6. Conclusiones finales	112
8. Estado del Trabajo y Desarrollos a Futuro	114
8.1. Manejo de Set de Datos Completo	114
8.2. Particiones inducidas por Sectores de Interacción Proteína-Proteína	114
8.3. Particiones inducidas por clúster	115
8.4. Particiones inducidas por recursividad de clúster	115
8.5. Particiones inducidas por comunidades	115
9. Exposiciones y Postulaciones	116
Bibliografía	117
10. Anexos	122
10.1. Tabla Modelos de clasificación generados según partición por sectores de interacción	122
10.2. Tabla resumen mejores resultados obtenidos para divisiones obtenidas mediante técnica de clustering	149
10.3. Resumen mejores modelos para cada división inducida mediante técnicas de clustering	154
10.4. Resumen mejores modelos obtenidos a partir de particiones inducidas mediante clustering recursivo	163

ÍNDICE DE FIGURAS

	página
1.1. Resumen de incidencias de afecciones relacionadas a VHL.	8
1.2. Flujo de Trabajo de VHL-Hunter.	9
1.3. Componentes en la minería de datos	11
1.4. Muestra de desbalance de clases en SVM.	22
1.5. Esquema de hiperplanos en SVM.	23
1.6. Esquema representativo de validación cruzada.	26
1.7. Esquema representativo de Leave One.	26
1.8. Posibles inconvenientes con los datos, donde k-medias no funciona correctamente	32
1.9. Representación de resultados al aplicar la clusterización por DBSCAN . .	37
1.10. Esquema representativo de cambios durante las iteraciones en GMM . .	41
1.11. Representación esquemática de una Red Neuronal	50
4.1. Esquema resumen metodología de preparación del set de datos.	58
4.2. Esquema resumen metodología de desarrollo estadísticas para set de datos.	59
4.3. Esquema representativo metodología aplicación de entrenamiento de modelos para set de datos.	60
4.4. Esquema representativo metodología de evaluación de adición de información topológica.	61
4.5. Esquema representativo metodología de validación de divisiones generadas	62
4.6. Esquema representativo metodología de aplicación de algoritmos de clustering	64
4.7. Esquema representativo metodología de búsqueda de sectores con alta interacción electrostática.	66
4.8. Representación de Módulos e Interacciones según POO.	68
5.1. Distribución del Atributo SStructural	72
5.2. Histograma para el atributo yDDG.	72
5.3. Box Plot para el set de datos.	73
5.4. Matriz de Correlación para el set de datos.	74
5.5. Histograma para distribución de Tasa de Verdaderos Positivos en Fase Exploratoria.	75
5.6. Matriz de Confusión para modelo de Clasificación con GradientBoostingClassifier.	78
5.7. Curva ROC con evaluación mediante Validación Cruzada.	79

5.8. Curva de Aprendizaje para modelo de Clasificación con GradientBoostingClassifier.	79
5.9. Curva de Aprendizaje para modelo de Clasificación con GradientBoostingClassifier.	80
5.10. Curva de Precisión v/s Recall para modelo de Clasificación con GradientBoostingClassifier.	81
5.11. Importancia de características en entrenamiento Random Forest.	82
5.12. Aporte de varianza en base al número de componentes.	83
5.13. Distribución de ejemplos según los sectores de interacción PP en VHL.	84
5.14. Box Plot para el sector de interacción A.	84
5.15. Medidas de Desempeño para dos divisiones.	86
5.16. Medidas de Desempeño para tres divisiones.	86
5.17. Medidas de Desempeño para cuatro divisiones.	87
5.18. Medidas de Desempeño para cinco divisiones.	87
5.19. Histograma para Distribuciones de Precision en entrenamiento del grupo 5 en divisiones inducidas mediante clustering.	91
5.20. Grafo obtenido por matriz de adyacencia mediante energías de interacción.	92
5.21. Esquema resumen del proceso de generación de particiones mediante recursividad de clúster.	93
6.1. Diagrama de interacción entre modelos.	96
6.2. Sistema de clasificación y ponderación de elementos.	98
6.3. Comparación de Medidas de Desempeño en Full Data set v/s Divisiones inducidas por sectores de interacción	99
6.4. Validación cruzada para modelos P con respecto al resto de divisiones.	100
6.5. Resumen medias de medidas de desempeño para las particiones seleccionadas en inducción vía clustering.	102
6.6. Resumen medias de medidas de desempeño para las particiones seleccionadas en inducción vía clustering recursivo.	104
6.7. Comparación medidas de desempeño obtenidas para cada forma de partición.	106
6.8. Matriz de confusión asociada a unión de grupos B-N-U en métodos asociados a Ensamble como algoritmo de entrenamiento	108
6.9. Curva ROC asociada a unión de grupos B-N-U en métodos asociados a Ensamble como algoritmo de entrenamiento	109

ÍNDICE DE TABLAS

	página
1.1. Cuadro resumen de algoritmos de aprendizaje supervizado	45
5.1. Modelos seleccionados según Tasa de verdaderos positivos.	76
5.2. Modelos seleccionados según Valor de Accuracy.	77
5.3. Resumen de medidas de desempeño promedios para las particiones generadas.	89
5.4. Cantidad de ejemplos en partición generada.	90
10.1. Tabla resumen de modelos de clasificación para divisiones inducidas por sectores de interacción	149
10.2. Resumen medidas de desempeño obtenidas en divisiones inducidas por clustering	153
10.3. Resumen medidas de desempeño y modelos para cada división inducida por clustering.	163
10.4. Resumen mejores modelos obtenidos para divisiones inducidas por clustering recursivo.	177

RESUMEN

La enfermedad de von Hippel-Lindau (VHL) es un trastorno genético con afectación multisistémica. Se asocia con tumores múltiples, incluidos retina y el sistema nervioso central, hemangioblastomas, carcinoma renal de células claras y el feocromocitoma. Siendo asociado al resultado de mutaciones en el gen VHL. Se han reportado miles de mutaciones asociadas con VHL. Sin embargo, para un gran porcentaje de ellas, su relevancia clínica aún no está clara, esta es la razón por la que VHL-Hunter se propone como un sistema de clasificación para mutaciones en VHL.

La herramienta procesa la mutación puntual y entrega como resultado si es clínicamente relevante o no. VHL-Hunter calcula la información filogenética y las energías de interacción y estabilidad, a través del uso de las herramientas MOSST y SDM. Además, se considera la información topológica de la proteína, por lo que la mutación se clasifica previamente según su posición en la proteína en algunos de los sectores de interacción proteína-proteína que han sido notificados, lo que permite utilizar el modelo de clasificación que corresponde según su sitio de interacción.

VHL-Hunter se compone de 13 modelos de clasificación, correspondientes a cada sector de interacción proteína-proteína. Todos los modelos fueron entrenados aplicando una estrategia exploratoria de algoritmos de clasificación y variación de parámetros, obteniendo distribuciones de medidas de desempeño, lo que permitió obtener un conjunto de modelos para cada sector de interacción, los cuales fueron filtrados por técnicas estadísticas. Los modelos se obtuvieron con tasas de verdaderos positivos y negativos, altos, agregando un alto valor de precisión. En los casos en que los modelos no permitieron obtener las características deseadas, se aplicaron técnicas de meta-learning para aumentar el rendimiento.

1. INTRODUCCIÓN

1.1. Von Hippel-Lindau

La enfermedad de von Hippel-Lindau (VHL) es un síndrome de neoplasia dominante autosómica que resulta de una mutación en la línea germinal en el gen VHL [37, 38, 39, 40, 41, 42, 43]. VHL se caracteriza por ser un síndrome de cáncer hereditario caracterizado por el desarrollo de tumores vasculares del sistema nervioso central y retina, carcinomas renales de células claras, feocromocitomas, tumores de células de islotes pancreáticos, tumores del saco endolinfático y quistes benignos que afectan una variedad de órganos [44].

El producto canónico de la proteína VHL, *pVHL* en su isoforma 1 (*pVHL*₃₀), tiene dos dominios estructuralmente diferentes: un dominio desordenado de 53 aminoácidos N-terminal no necesario para la supresión tumoral y un dominio ordenado C-terminal que consiste en un dominio α -helicoidal (residuos 155-192) y un dominio principalmente de hoja β (residuos 63-154 y 193-204). *pVHL* forma un complejo ternario con las proteínas elongina C y elongina B (Complejo VCB) [47, 48] el cual es crítico para la estabilidad y la función en *pVHL* [49].

Las mutaciones que afectan a los residuos de unión a *pVHL* en elongina C se han descrito en ccRCC, lo que respalda la hipótesis de que los efectos tumorogénicos de las mutaciones de VHL se relacionan con la disfunción del complejo VCB. Por lo tanto, todo el complejo de VCB debe considerarse como una sola entidad cuando se evalúan los efectos estructurales y funcionales de las mutaciones de VHL [45].

VHL presenta una incidencia de 1 en 36.000 pacientes, los cuales se predisponen al desarrollo de hemangioblastoma retiniano, cerebeloso y espinal, quistes y carcinoma de páncreas y RCC [46], adicional a ello, han sido documentadas, sobre 1000 mutaciones en VHL, de las cuales cerca del 52% se desconoce su relevancia clínica o no se tiene una mayor información de ésta [45].

Las características viscerales del trastorno incluyen quistes y carcinomas renales, los feocromocitomas, los quistes pancreáticos y los tumores neuroendocrinos, así como los

cistadenomas del epidídimo y del ligamento ancho [43], cuyos índices de aparición se exponen en la Figura 1.1.

	Mean (range) age of onset (years)	Frequency in patients (%)
CNS		
Retinal haemangioblastomas	25 (1–67)	25–60%
Endolymphatic sac tumours	22 (12–50)	10%
Craniospinal haemangioblastomas		
Cerebellum	33 (9–78)	44–72%
Brainstem	32 (12–46)	10–25%
Spinal cord	33 (12–66)	13–50%
Lumbosacral nerve roots	Unknown (...)	<1%
Supratentorial	Unknown (...)	<1%
Visceral		
Renal cell carcinoma or cysts	39 (16–67)	25–60%
Phaeochromocytomas	30 (5–58)	10–20%
Pancreatic tumour or cyst	36 (5–70)	35–70%
Epididymal cystadenoma	Unknown (...)	25–60%
Broad ligament cystadenoma	Unknown (16–46)	Unknown

Figura 1.1: Resumen de incidencias de afecciones relacionadas a VHL.

El diagnóstico de la enfermedad de von Hippel-Lindau a menudo se basa en criterios clínicos. Los pacientes con antecedentes familiares y un hemangioblastoma del SNC (incluidos los hemangioblastomas retinianos), un feocromocitoma o un carcinoma renal de células claras son diagnosticados con la enfermedad. Los que no tienen antecedentes familiares relevantes deben tener dos o más hemangioblastomas del SNC o un hemangioblastoma del SNC y un tumor visceral (a excepción de quistes epididimarios y renales, que son frecuentes en la población general) para cumplir con los criterios diagnósticos [50, 51, 52].

Las correlaciones específicas de genotipo y fenotipo han surgido en las familias afectadas. Ahora se reconocen varios fenotipos familiares de la enfermedad de von Hippel-Lindau, que proporcionan información útil para examinar y aconsejar a las personas afectadas. Las familias tipo 1 tienen un riesgo muy reducido de feocromocitomas, pero pueden desarrollar todos los otros tipos de tumores generalmente asociados con la enfermedad. Las familias de tipo 2 tienen feocromocitomas, pero tienen un bajo riesgo (tipo 2A) o alto riesgo (tipo 2B) para los carcinomas de células renales. Las familias de tipo 2C solo tienen feocromocitomas, sin otros hallazgos neoplásicos de VHL [43].

Los avances en las pruebas genéticas para la detección de la enfermedad incluyen *Southern blotting cualitativo y cuantitativo*, que se ha agregado al análisis de la secuencia de ADN. Esto mejoró para uso personal. No obstante, su reproducción es sólo con permiso de

The Lancet Publishing Group. La prueba ha aumentado la tasa de detección de mutaciones de ADN en leucocitos de sangre periférica del 75 % a casi el 100 % [43].

Se han desarrollado diversos software que permiten explorar el efecto de las mutaciones en las proteínas que han sido detectadas y como afectan a su estabilidad térmica y qué efectos provocan en la proteína, siendo una de éstas SDM [53], el cual predice los cambios en la estabilidad esperados sobre la mutación en base a la consideración de información estructural y su secuencia, dado el uso de Tablas Específicas de entornos de sustitución (ESSTs). Por otro lado, se han desarrollado softwares que facilitan el estudio de mutaciones, mediante la evaluación de información filogenética y los cambios en la secuencia que se han observado en una familia de proteínas y una proteína de interés. Este es el caso de MOSST [54], el cual estima mediante cálculos matemáticos la propensión de mutaciones en la proteína a estudiar, con respecto a la familia de contraste. Otro de los software de interés que han sido desarrollados para la evaluación de mutaciones en VHL, según el riesgo de ésta es el expuesto en [45], en el cual, por medio de algoritmo Random Forest, se propone un sistema de clasificación de mutaciones según el riesgo que éstas presentan.

Dado a lo anterior, se ha denotado la falta de sistemas de clasificación de riesgo de mutaciones detectadas en VHL, las cuales han sido reportadas pero se desconoce si representarán un riesgo o no para el paciente. En base a esto y a los sistemas que se han desarrollado para trabajar con dichas mutaciones, se propone VHL-Hunter.

VHL-Hunter, es un servicio web de clasificación, el cual recibe una mutación puntual o una lista de ellas y evalúa si dicha mutación presenta una relevancia clínica. Un esquema representativo del flujo de trabajo de VHL-Hunter es como se expone en la Figura 1.2.

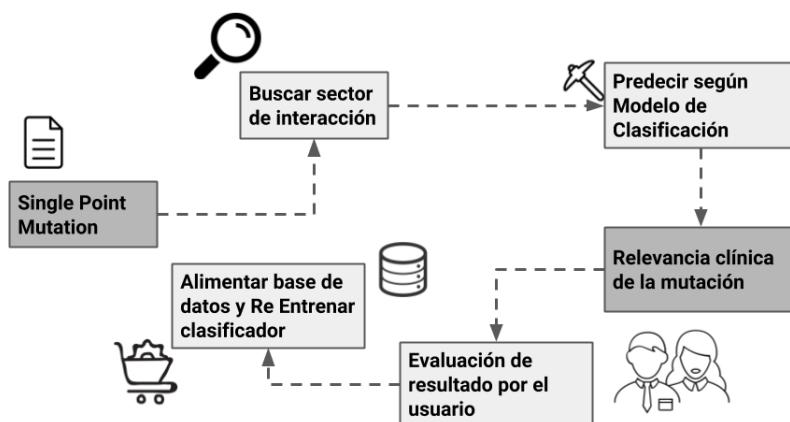


Figura 1.2: Flujo de Trabajo de VHL-Hunter.

Tal como se expone en la Figura 1.2, VHL-Hunter trabaja con mutuaciones puntuales como dato de entrada, el cual se compone por: Residuo original, Residuo Mutuado y la posición, en base a esto, la herramienta evalúa a qué sector de interacción pertenece el residuo y selecciona el modelo de clasificación a utilizar, se predice la relevancia clínica y se reporta el resultado al usuario.

El usuario tiene la opción de reportar si la clasificación generada es efectivamente correcta o no, según el conocimiento que éste disponga, dado esto, la mutuación puntual pasa a ser parte del sistema de almacenamiento persistente, se entrena nuevamente el modelo y se actualizan los datos asociados.

La selección del sector de interacción, hace referencia a información topológica que asocia la herramienta, en la cual se evalúan los sectores que interactúan con otras proteínas formando interacciones Proteína-Proteína. Se han reportado 13 sectores de interacción, razón por la cual, se utilizan como sectores independientes formando set de datos únicos, lo cual implica que son modelos de clasificación con algoritmos, parámetros y medidas de desempeño diferentes. Detalles los cuales serán explicados en la sección de Metodología.

Un punto importante a destacar es que, VHL-Hunter trabaja con información estructural y de estabilidad, en base a los resultados de predicción que entrega SDM [53] y adición de información filogenética, según las propensiones calculadas por MOSST [54], adicional a la información topológica que implica el conocimiento de los sectores de interacción Proteína-Proteína que presenta VHL.

Finalmente, el hecho de permitir al usuario reportar si la mutación fue clasificada correctamente, implica que los modelos de clasificación se alimentarán con nueva información constantemente, adicionando nuevos elementos a la base de datos, y permitiendo las actualizaciones de los modelos, en base a la nueva data. Generando impactos en las medidas de desempeño que estos posean y en las técnicas de validación que se usen para dicha instancia.

1.2. Marco Teórico

1.2.1. Minería de Datos

Minería de datos es el proceso de descubrimiento de patrones en set de datos, involucrando métodos asociados a Machine Learning, Estadísticas y sistemas de bases de datos. [33]. La minería de datos es un subcampo interdisciplinario de la informática, el cual tiene por objetivo general extraer información (a través de métodos inteligentes) de un conjunto de datos y transformar la información en una estructura comprensible para su uso posterior. [34, 35]. La minería de datos es el paso de análisis del proceso de *descubrimiento*

de conocimiento en bases de datos, o KDD. [36]. Además del análisis en bruto de los datos, también incluye aspectos de manipulación de bases de datos y pre procesamiento de datos, evaluaciones de modelo e inferencia, métricas de interés, consideraciones de complejidad, post procesamiento de estructuras descubiertas, visualización y actualización de la información.

En la Figura 1.3, se exponen las principales ramas que componen la minería de datos y los diferentes procesos que se asocian a dichas ramas.

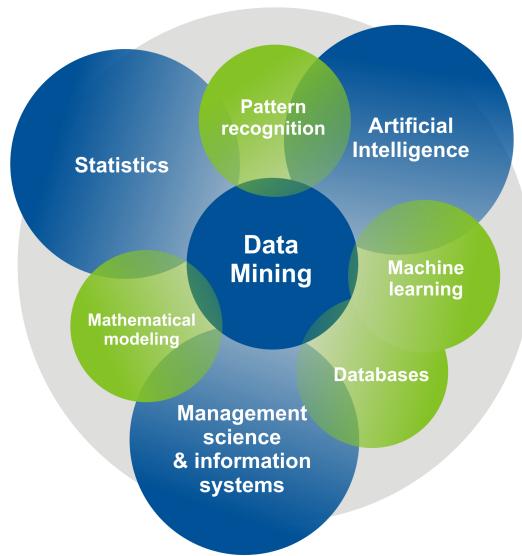


Figura 1.3: Componentes en la minería de datos

Son tres las principales áreas que abarca la minería de datos: Estadística, Inteligencia Artificial y Manipulación de sistemas de información, mientras que son distintos procesos los que interactúan entre estas ramas, tales como: Modelamiento Matemático, reconocimiento de patrones, Sistemas de almacenamiento persistente y machine learning.

Cada área en particular tiene un objetivo general y diversos objetivos específicos. Sin embargo, estas áreas interactúan entre sí, con el fin de poder extraer patrones de información que generen conocimientos a partir de la data de procesada.

La minería de datos se utiliza en diferentes campos, tales como: Genética, Evaluaciones proteómicas, Comercio, Sistemas de tránsito, Optimizaciones en procesos industriales, reconocimiento de patrones y rasgos cuantificables en enfermedades y más recientemente en áreas de dinámicas moleculares y parámetros para la generación de pipe lines automatizados de simulaciones cuánticas en sistemas químicos.

1.2.2. Aprendizaje de Máquinas

Aprendizaje de Máquina, es una rama de la inteligencia artificial que tiene por objetivo el desarrollo de técnicas que permitan a los computadores aprender, es decir, generalizar comportamientos a partir de una información no estructurada suministrada en forma de ejemplos.

La técnica es aplicada en: motores de búsqueda, diagnósticos médicos, detección de fraude en el uso de tarjetas de crédito, análisis del mercado de valores, clasificación de secuencias de ADN, reconocimiento del habla y del lenguaje escrito, etc.

Tiene como resultado un modelo para resolver una tarea dada. Entre los que se distinguen:

- Modelos geométricos: basados en espacio de instancias, siendo de múltiples dimensiones.
- Modelos probabilísticos: basados en la detección de la distribución de probabilidades asociada a una función de los valores.
- Modelos lógicos: expresan las probabilidades en reglas organizadas en forma de árboles de decisión.

A continuación se abordarán temáticas relacionadas con el procesamiento de los datos, seguido a su vez de explicación de algoritmos de aprendizaje de máquinas, tanto supervisados como no supervisados, debido a que juegan un rol fundamental a la hora de dar cumplimiento a los objetivos de la memoria de título.

Pre Procesamiento de Datos

PCA

Análisis de Componentes Principales (PCA por sus iniciales en inglés), es una técnica estadística que permite la conversión de un conjunto de variables posiblemente correlacionadas a un conjunto de variables no correlacionadas linealmente, los cuales se denominan componentes principales, siendo la cantidad menor o igual que las variables originales, donde la principal característica de los componentes principales es que son ordenados en base a la varianza que entregan a los datos, así el primer componente principal aporta una mayor varianza que el segundo y así sucesivamente, basándose principalmente en el uso de vectores propios.

PCA se utiliza principalmente como una herramienta en el análisis de la exploración de datos los cuales tienen como objetivo generar modelos de predicción y sus resultados normalmente se exponen en puntuaciones que tienen estrecha relación con el aporte de varianza que estos entregan [3].

Intuitivamente es posible pensar el PCA como un elipsoide n-dimensional de datos, donde cada eje del elipsoide representa un componente principal, esto implica que si algún eje del elipsoide es pequeño, la varianza correspondiente a lo largo de éste también lo es, por lo que omitir dicho eje no implica una pérdida importante de información, esto último es denotado como la reducción de la dimensionalidad en base a los aportes a las varianzas que denotan cada componente.

Definición

Matemáticamente, es posible definir PCA como una transformación lineal ortogonal que transforma los datos a un nuevo sistema de coordenadas tal que la mayor varianza por alguna proyección de los datos pasa a situarse en la primera coordenada (llamado el primer componente principal), la segunda mayor varianza en la segunda coordenada, y así sucesivamente [4].

Se considera un conjunto de datos, \mathbf{X} , con una media empírica 0, donde cada una de las filas (\mathbf{n}) representan ejemplos y las columnas características o atributos (\mathbf{p}).

La transformación está definida por un set de vectores de dimensión \mathbf{p} que poseen pesos denotados por $w_{(k)} = (w_1, \dots, w_p)_{(k)}$, los cuales para cada vector x_i en X se operan para dar un vector con los componentes principales $t_{(i)} = (t_1, \dots, t_k)_{(i)}$ el cual viene dado por $t_{k(i)} = x_i w_k$

De tal manera que las variables individuales de \mathbf{t} considerado sobre el conjunto de datos sucesivamente heredan la varianza máxima posible de \mathbf{x} , con cada carga del vector \mathbf{w} .

El primer componente w_1 tiene que satisfacer las siguientes características:

- $w_{(1)} = \arg \max_{\|w\|=1} \left\{ \sum_i (t_{(1)})_i^2 \right\} = \arg \max_{\|w\|=1} \left\{ \sum_i (x_{(i)} * w)^2 \right\}$
- $w_{(1)} = \arg \max_{\|w\|=1} \left\{ \|Xw\|^2 \right\} = \arg \max_{\|w\|=1} \left\{ w^T X^T X w \right\}$
- $w_{(1)} = \arg \max \left\{ \frac{w^T X^T X w}{w^T w} \right\}$

Los k restantes componentes son encontrados efectuando la extracción de los primeros $k-1$ componentes principales desde \mathbf{x} :

$$\hat{x}_k = x - \sum_{\delta=1}^{k-1} Xw_{(\delta)}w_{(\delta)}^T$$

A su vez, para encontrar el vector de carga, es necesario extraer la varianza máxima del nuevo set de datos, tal que:

$$w_{(k)} = \arg \max_{\|w\|=1} \{||\hat{x}_k w||^2\} = \arg \max \left\{ \frac{w^T X^T \hat{X}_k^T \hat{x}_k w}{w^T w} \right\}$$

La matriz de covarianza juega un rol fundamental en este análisis, cuyo valor entre dos componentes principales viene dado por:

$$\begin{aligned} Q(PC_j, PC_k) &\propto (Xw_{(j)})^T * (Xw_{(k)}) \\ Q(PC_j, PC_k) &= w_{(j)}^T X^T X w_{(k)} \\ Q(PC_j, PC_k) &= w_{(j)}^T \lambda_{(k)} w_{(k)} \\ Q(PC_j, PC_k) &= \lambda_{(k)} w_{(j)}^T w_{(k)} \end{aligned}$$

La principal característica que define al PCA es que es una técnica comúnmente utilizada para la reducción de la dimensionalidad, esto viene dado por la transformación que se genera, $T = xw$ donde cada vector $x_{(i)}$ existente en un espacio de coordenadas de variables p, es representado por un nuevo espacio en el cual las variables no se encuentran correlacionadas, sin embargo, si se utilizan L componentes principales para así utilizar los primeros L vectores de carga se obtiene una transformación truncada $T_L = XW_L$, de tal manera que la matriz T_L posee los n ejemplos originales. Sin embargo sólo posee L características que definen el set de datos, de tal manera que dicha transformación es posible expresarla como:

$t = W^T x$, donde $x \in R^p, t \in R^L$, para las cuales las columnas $p \times L$ de la matriz W forman una base ortogonal de las L características, de esta manera, al basarse en la construcción con sólo L columnas se maximiza la varianza original de los datos y se minimiza el error cuadrático tal que:

$$||TW^T - T_L W_L^T||_2^2 = ||X - X_L||_2^2$$

Normalmente esta reducción es usada para el manejo de set de datos de alta dimensionalidad [21].

Propiedades

Existen tres propiedades claras que facilitan la comprensión y la utilidad del PCA como medio de manejo de dimensionalidades, y transformación de datos a espacios con coordenadas similares no correlacionados linealmente, éstas son:

- Para cualquier entero $q, 1 \leq q \leq p$, se considera la transformación lineal $y = B'x$, donde y es un elemento del vector q y B es una matriz de $(q \times p)$ y la matriz de covarianza para y se puede denotar por $\sum_y = B' \sum B$, entonces la traza de \sum_y es maximizada tomando $B = A_q$ donde A_q es la primera columna q de A .

- Considerando la transformación ortogonal $y = B'x$, la traza de \sum_y es minimizada tomando $B = A_q^*$ donde A_q^* es la última columna q de A .
- Sea la descomposición espectral de $\sum = \lambda_1\alpha_1\alpha'_1 + \dots + \lambda_p\alpha_p\alpha'_p$, se tiene que $Var(x_j) = \sum_{k=1}^p \lambda_k \alpha_{kj}^2$.

En resumen, es posible denotar que todas las operaciones de los datos son centradas en cero, tal que $\frac{1}{N} \sum_{i=1}^N x_i = 0$, para lo cual se realizan tratamientos sobre la matriz de covarianza, la cual se puede exponer como: $C = \frac{1}{N} \sum_{i=1}^N x_i x_i^T$, con el fin de obtener los vectores propios, denotados como **eigen decomposition**: $\lambda v = Cv$, los cuales pueden ser reescritos como $\lambda x_i^T = x_i^T Cv \forall i \in [1, N]$, lo cual genera la transformación lineal en vectores con menor dimensionalidad expuestos por sus aportes de varianzas en los datos, existentes en espacios con coordenadas similares, sin correlación lineal entre ellos.

Algoritmos de Aprendizaje en Minería de Datos

Los algoritmos de Aprendizaje pueden ser definidos como una serie de pasos que permitan cumplir una tarea dada, el uso del cual dependerá de las características que posea dicha tarea [6].

Los algoritmos de aprendizaje pueden clasificarse en dos grandes grupos:

- **Supervisados**: se cumple un rol de predicción, clasificación, asignación, etc. a un conjunto de elementos con características similares, por lo que los datos de entrada son conocidos.
- **No Supervisados**: su objetivo es agrupar en conjuntos con características similares los elementos de entrada dado los valores de estos atributos, en base a la asociación de patrones característicos que representen sus comportamientos.

A continuación se hace referencia a los diferentes tipos de algoritmos existentes, considerando la clasificación expuesta anteriormente.

Algoritmos de Aprendizaje Supervisado

Es una técnica para deducir una función a partir de datos de entrenamiento. Los datos de entrenamiento consisten en pares de objetos (normalmente vectores): una componente del par son los datos de entrada y el otro, los resultados deseados. La salida de la función puede ser un valor numérico (como en los problemas de regresión) o una etiqueta de clase (como en los de clasificación).

El objetivo del aprendizaje supervisado es crear una función capaz de predecir el valor correspondiente a cualquier objeto de entrada válida después de haber visto una serie de ejemplos, los datos de entrenamiento. Para ello, tiene que generalizar a partir de los datos presentados a las situaciones no vistas previamente.

Existe una gran variedad de algoritmos de aprendizaje supervisado, dentro de los cuales destacan:

k Nearest Neighbors

Algoritmo de aprendizaje supervisado, el cual tiene por objetivo asociar un elemento a una clase en particular, dada la información de ejemplos de entrada que tengan asociadas características particulares, que puedan declararse como *vecinos* del nuevo ejemplo a clasificar, siendo **k** el número de vecinos que se está dispuesto a utilizar para aplicar la clasificación.

Con el fin de evaluar la cercanía de los ejemplos existentes contra el nuevo ejemplo a clasificar es necesario asociar ciertas medidas de distancia que permitan cuantificar esta característica, para así poder comparar esta distancia y evaluar la cercanía para asociarle una clase a este nuevo ejemplo.

Presenta algunos problemas, tales como: posibles errores al existir más de un elemento de distinta clase cercano al nuevo ejemplo a clasificar, sin embargo, dicho error estimado es reducido.

La distancia a emplear para evaluar la cercanía puede ser: Euclíadiana, Manhattan, coseno, Mahalanobis¹, entre las principales.

La mejor elección de **k** depende fundamentalmente de los datos; generalmente, valores grandes de **k** reducen el efecto de ruido en la clasificación, pero crean límites entre clases parecidas.

Existen dos variaciones para la aplicación de KNN: aplicación basada en las distancias y aplicación basada en radios con respecto a puntos, la primera es mayormente usada, no obstante, en el caso de que los puntos no se encuentren uniformemente distribuidos es una mejor opción usar la segunda alternativa, siendo muy eficaz en problemas conocidos como *la maldición de la dimensionalidad*.

KNN utiliza el componente de peso, es decir, valores asignados a puntos específicos para determinar si un elemento a clasificar es de una clase o no, normalmente se utilizan pesos uniformes. Sin embargo, es posible asignar valores de tal manera que al momento de realizar la votación puntos más cercanos en base a distancias presenten más peso que otros.

¹Se explican en detalle en la sección 1.2.2

Se han implementando diversos algoritmos a la hora de aplicar la técnica de KNN, los cuales tienen relación con el coste computacional que presentan, dentro de estos se encuentran: Brute Force, K-D Tree y Ball Tree [7].

Naive Bayes

Naive Bayes es un conjunto de algoritmos de aprendizaje supervisados basados en la aplicación del teorema de Bayes con la suposición *ingenua* de independencia entre cada par de características [8]. Dada una variable de clase y y un vector de característica dependientes de la forma x_1, \dots, x_n , el teorema de Bayes establece la siguiente relación:

$$P(y | x_1, \dots, x_n) = \frac{P(y)P(x_1, \dots, x_n | y)}{P(x_1, \dots, x_n)}$$

Utilizando la suposición ingenua de independencia de características, se tiene que:

$$P(x_i | y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i | y)$$

Para todo i , esta relación se simplifica a:

$$P(y | x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^n P(x_i | y)}{P(x_1, \dots, x_n)}$$

Dado que $P(x_1, \dots, x_n)$ es constante dada la entrada, se puede utilizar la siguiente regla de clasificación:

$$P(y | x_1, \dots, x_n) \propto P(y) \prod_{i=1}^n P(x_i | y)$$

⇓

$$\hat{y} = \arg \max_y P(y) \prod_{i=1}^n P(x_i | y),$$

A pesar de sus supuestos aparentemente simplificados, los clasificadores de Naive Bayes han funcionado bastante bien en muchas situaciones del mundo real, la famosa clasificación de documentos y el filtrado de spam son ejemplos de ello. Requieren una pequeña cantidad de datos de entrenamiento para estimar los parámetros necesarios. Pueden ser extremadamente rápido en comparación con métodos más sofisticados. El desacoplamiento de las distribuciones de las características condicionales de clase significa que cada distribución se puede estimar de forma independiente como una distribución unidimensional. Esto a su vez ayuda a aliviar los problemas derivados de la dimensionalidad.

Existen distintos tipos de clasificadores de Naive Bayes, diferenciándose entre sí en la función de distribución de probabilidad que utilizan, dentro de los que se encuentran:

- Gaussian Naive Bayes.

$$P(x_i | y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

- Multinomial Naive Bayes.

La distribución se parametriza por el vector $\theta_y = (\theta_{y1}, \dots, \theta_{yn})$ para cada clase y , donde n es el número de características y θ_{y1} es la probabilidad $P(x_i | y)$ de que la característica i aparezca en una muestra que pertenece a la clase y .

Cada θ_y es estimado por:

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

Donde $N_{yi} = \sum_{x \in T} x_i$ es el número de veces que aparece la característica i en la muestra de clase y en el set de entrenamiento T y $N_y = \sum_{i=1}^{|T|} N_{yi}$ representa el total de todas las características para la clase.

- Bernoulli Naive Bayes.

$$P(x_i | y) = P(i | y)x_i + (1 - P(i | y))(1 - x_i)$$

Árboles de Decisión

Se define árbol de decisión como un modelo de predicción utilizado en el ámbito de la inteligencia artificial, en el cual dado un conjunto de datos se fabrican diagramas de construcciones lógicas, muy similares a los sistemas de predicción basados en reglas, que sirven para representar y categorizar una serie de condiciones que ocurren de forma sucesiva, para la resolución de un problema.

El aprendizaje basado en árboles de decisión utiliza un árbol como un modelo predictivo que mapea las observaciones de las características que presenta un elemento [9].

En estas estructuras de árbol, las hojas representan etiquetas de conjuntos ya clasificados, los nodos, a su vez, nombres o identificadores de los atributos y las ramas representan posibles valores para dichos atributos.

Aprendizaje basado en árboles de decisión es un método comúnmente utilizado en la minería de datos, cuyo objetivo consiste en desarrollar un modelo de predicción para el valor de una variable de destino en función de diversas variables de entrada.

Un árbol de decisión es una representación simple para clasificar ejemplos, el aprendizaje basado en esta metodología es una de las técnicas más eficientes para la

clasificación supervisada. Donde cada ejemplo consta de atributos con valores discretos dentro de un dominio de conjunto finito, y existe un sólo término final denominado clasificación.

En un árbol de decisión, cada elemento del dominio de la clasificación se llama clase, cada nodo interno (no hoja) está etiquetado con una función de entrada, las ramas procedentes de un nodo etiquetado con una característica están asociados con cada uno de los posibles valores de la característica. Cada hoja del árbol se marca con una clase o una distribución de probabilidad sobre las clases.

Un árbol puede ser entrenado mediante el fraccionamiento del conjunto inicial en subconjuntos basados en una prueba de valor de atributo. Este proceso se repite en cada subconjunto derivado de una manera recursiva llamada particionamiento recursivo. La recursividad termina cuando el subconjunto en un nodo tienen todos el mismo valor de la variable objetivo, o cuando la partición ya no agrega valor a las predicciones.

Para cada división es necesario el uso de una función que entregue una medida de impureza en cada división, esto con el objetivo de seleccionar la mejor partición para un atributo dado, la elección de dicho atributo se basa en el objetivo de separar de mejor manera los ejemplos.

La selección de los atributos se basa en qué atributo al momento de clasificar genera nodos más puros, para ello se utiliza una función de ganancia de información, la cual representa la ganancia obtenida a partir de una división de los ejemplos de entrenamiento. Dicha función puede ser expuesta como sigue:

$$\Phi(D, t) = I(t) - \sum_{i=1}^l I(t_i)P_i \quad (1.1)$$

Donde:

- $I(t)$ representa la Medida de Impureza asociada al nodo **t**, desde el cual se comenzará a realizar la partición o nodo padre.
- $\sum_{i=1}^l I(t_i)$ representa la suma ponderada de las impurezas de los nodos hijos t_i generados a partir de una división **D**.
- P_i representa la proporción de ejemplos que siguen la rama **i** asociada a la división **D**.

Dentro de las medidas de impureza, existen:

- Gini Index = $\sum p_i x (1 - p_i)$
- Entropía = $-\sum p_i x \log_2(p_i)$

Siendo la más utilizada la medida de Entropía.

En ambas, p_i corresponde a la proporción de ejemplos asociados a cada una de las clases, presentes en el nodo evaluado.

El algoritmo para el cual se entrena y se clasifica un nuevo ejemplo es el que sigue:

- Partir desde un nodo inicial o padre.
- Seleccionar el mejor atributo que divide de una manera óptima los ejemplos, lo cual se observa por medio de la función de ganancia de información.
- Se clasifica los ejemplos del conjunto de entrenamiento de un nodo entre sus descendientes.
- El proceso finaliza si los ejemplos del conjunto de entrenamiento quedan perfectamente clasificados, esto ocurre en dos casos: todos los ejemplos pertenecen a una misma clase o se llega a una hoja.
- En el caso de no cumplirse lo del punto anterior, se itera para cada rama de manera recursiva, utilizando sólo los ejemplos que llegan a esa rama.

SVM

Support Vector Machine (SVM), también conocido como redes de soporte vectorial, son modelos de aprendizaje supervisados asociados al análisis de los datos utilizados para la clasificación. Dado un conjunto de ejemplos de entrenamiento, cada uno marcado como perteneciente a una u otra de las dos categorías, un algoritmo de entrenamiento de SVM construye un modelo que asigna nuevos ejemplos a una categoría u otra, convirtiéndolo en un clasificador binario lineal no probabilístico. Un modelo SVM es una representación de los ejemplos como puntos en el espacio, mapeados de modo que los ejemplos de las categorías separadas se dividan por un espacio claro que es tan amplio como sea posible. Nuevos ejemplos son entonces mapeados en ese mismo espacio y predicen si pertenecen a una categoría en base a qué lado del espacio son asignados [10].

SVM puede realizar eficientemente una clasificación no lineal utilizando funciones kernel, con el fin de generar transformaciones de espacio dimensional de los datos, para mapear implícitamente sus entradas en espacios característicos de alta dimensión.

Las ventajas de máquinas de soporte vectorial son:

- Efectivo en espacios de dimensiones altas.
- Efectivo aún en casos donde el número de dimensiones es mayor que el número de muestras.
- Utiliza un subconjunto de puntos de entrenamiento en la función de decisión (llamada vectores de soporte), por lo que también es memoria eficiente.
- Versátil: diferentes funciones del núcleo pueden ser especificadas para la función de decisión. Se proporcionan núcleos comunes, pero también es posible especificar núcleos personalizados.

Las desventajas de las máquinas de soporte vectorial incluyen:

- Si el número de características es mucho mayor que el número de muestras, es probable que el método tenga un mal desempeño.
- SVMs no proporciona directamente estimaciones de probabilidad, estos se calculan utilizando cinco veces una costosa validación cruzada.

Existen diversas variaciones de SVM, tales como: SVC, NuSVC y LinearSVC, los cuales son capaces de realizar una clasificación multiclas² en un conjunto de datos, es decir, ya no depender de un clasificador único para dos clases.

SVC presenta una aplicación basada en libsvm³. La complejidad del tiempo de ajuste se hace cuadrática con el número de muestras, lo que dificulta escalar a conjunto de datos con tamaño mayor a 10000 [11]. El apoyo multiclas es manejado según un esquema de uno vs uno.

Por otro lado NuSVC presenta características similares a SVC, pero, utiliza un parámetro para controlar el número de vectores de soporte. La aplicación se basa en libsvm.

LinearSVC es similar a SVC pero, se utiliza una función de kernel lineal, además es implementado en términos de liblinear en lugar de libsvm, por lo que tiene más flexibilidad en la elección de las penalizaciones y las funciones de pérdida y debería escalar mejor a un gran número de muestras. Esta clase soporta entradas densas y escasas y el soporte de multiclas se maneja de acuerdo con un esquema de uno contra el resto.

Cada uno de los clasificadores expuestos en los puntos anteriores toman como entrada el set de entrenamiento y las etiquetas asociadas a las clases, con el fin de generar tanto el testeо

²Implica la existencia de un número de clases mayor a dos

³Librería implementada para el desarrollo de máquina soporte de vectores, <https://www.csie.ntu.edu.tw/~cjlin/libsvm/>

como la validación del modelo, previo etapa de entrenamiento, la principal característica es que se utilizan vectores de apoyo para el set de entrenamiento, los que son denominados vectores de soporte, normalmente se utilizan funciones kernel para la obtención de estos vectores de soporte.

SVC y NuSVC implementan el enfoque *uno contra uno* para la clasificación multiclas. Si existen n clases, se construyen $\frac{n*(n-1)}{2}$ clasificadores, de los cuales cada uno forma un set datos de dos clases; por otro lado, LinearSVC implementa una estrategia multi-clase *uno contra el resto*, formando así modelos de n clases, los cuales son entrenados n veces. Si sólo hay dos clases, sólo se entrena un modelo.

Los algoritmos SVM están asociados a diversos problemas, sin embargo el principal radica en el desbalance de clases, ya sea por el número que presentan o por el peso asociado a éstas, tal como se expone en la Figura 1.4:

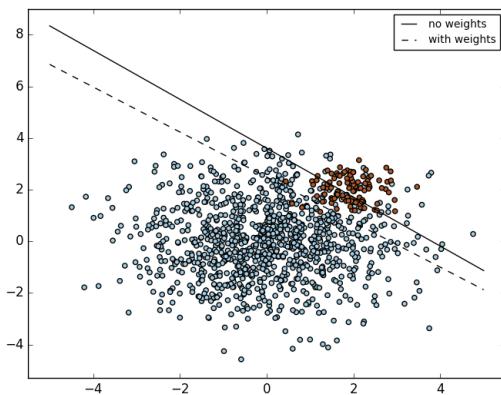


Figura 1.4: Muestra de desbalance de clases en SVM.

Complejidad

Las máquinas de soporte vectorial son herramientas poderosas, pero sus requerimientos de computación y almacenamiento aumentan rápidamente con el número de vectores de entrenamiento. El kernel de un SVM es un problema de programación cuadrática (QP), separando los vectores de soporte del resto de los datos de entrenamiento. Es posible esclarar esta solución entre $O(n_{features} \times n_{samples}^2)$ y $O(n_{features} \times n_{samples}^3)$ [9].

Formulación Matemática

SVM construye un hiperplano o conjunto de hiperplanos en un espacio dimensional

alto o infinito, que puede usarse para clasificación, regresión u otras tareas. Intuitivamente, se logra una buena separación por el hiperplano que tiene la mayor distancia a los puntos de datos de entrenamiento más próximos de cualquier clase (llamado margen funcional), ya que en general, cuanto mayor es el margen, menor es el error de generalización del clasificador, tal como se expone en la Figura 1.5:

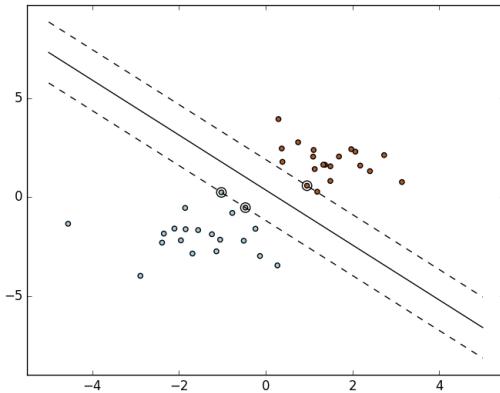


Figura 1.5: Esquema de hiperplanos en SVM.

Se expone la formulación matemática para cada uno de los clasificadores expuestos anteriormente:

SVC

Dado los vectores de entrenamiento $x_i \in R^p$, $i=1, \dots, n$, en dos clases, y un vector, $y \in \{1, -1\}^n$, SVC resuelve el siguiente problema primario:

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^n \zeta_i$$

$$\text{para la clase } y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i, \quad \zeta_i \geq 0, i = 1, \dots, n$$

Su doble es

$$\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - e^T \alpha$$

$$\text{para la clase } y^T \alpha = 0 \quad 0 \leq \alpha_i \leq C, i = 1, \dots, n$$

Donde e es el vector de todos los unos, $C > 0$ es el límite superior, Q es una matriz de $n \times n$ definida semipositiva, $Q_{ij} \equiv y_i y_j K(x_i, x_j)$, donde $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ es el kernel. Los vectores de entrenamiento son implícitamente mapeados en un espacio dimensional mayor (tal vez infinito) por la función ϕ .

La función de decisión es:

$$sgn(\sum_{i=1}^n y_i \alpha_i K(x_i, x) + \rho)$$

NuSVC

Se introduce el parámetro ν el cual controla el número de vectores de soporte y errores de entrenamiento. El parámetro $\nu \in (0, 1]$ es un límite superior en la fracción de errores de entrenamiento y un límite inferior de la fracción de vectores de soporte.

SVR

Dados los vectores de entrenamiento $x_i \in R^n$ ε -SVR resuelve el siguiente problema primario:

$$\min_{w,b,\zeta,\zeta^*} \frac{1}{2} w^T w + C \sum_{i=1}^n (\zeta_i + \zeta_i^*)$$

$$\text{para la clase } y_i - w^T \phi(x_i) - b \leq \varepsilon + \zeta_i,$$

$$w^T \phi(x_i) + b - y_i \leq \varepsilon + \zeta_i^*, \quad \zeta_i, \zeta_i^* \geq 0, i = 1, \dots, n$$

Donde e es el vector para todos, $C > 0$ es el límite superior, Q es una matriz de $n \times n$ definida semipositiva, $Q_{ij} \equiv K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ es el kernel. Aquí los vectores de entrenamiento son implícitamente mapeados en un espacio dimensional mayor (tal vez infinito) por la función ϕ .

La función de decisión es:

$$\sum_{i=1}^n (\alpha_i - \alpha_i^*) K(x_i, x) + \rho$$

Medidas de Evaluación de Modelos

Medir el desempeño del modelo predictivo es importante a la hora de evaluar qué tan efectivo es el entrenamiento y la clasificación que se genera, existen medidas que sólo se basan en la cantidad de aciertos o errores que comete el clasificador, otras que implican la eficiencia del modelo y otras que se basan en la precisión, se define brevemente algunas de las performance utilizadas a la hora de evaluar modelos de aprendizajes supervisados:

- **Tasa de Verdaderos Positivos:** corresponde a la medida asociada a las correctas clasificaciones versus el total de clasificaciones realizadas, es decir, cuántos predicciones efectivas se obtuvieron con respecto a una clase.

- **Tasa de Falsos Positivos:** corresponde a la medida asociada a las clasificaciones mal efectuadas, es decir, cuántas predicciones erradas existen con respecto a una clase.
- **Eficiencia Global:** corresponde al total de predicciones correctamente efectuadas con respecto al total de ejemplos existentes en la muestra.
- **Especificidad:** corresponde a la probabilidad asociada a clasificar de manera negativa un ejemplo positivo.
- **Precisión:** corresponde a la probabilidad asociada a predecir un ejemplo de manera correcta, con respecto a la predicción.
- **Alcance:** corresponde a la probabilidad asociada a predecir un ejemplo de manera correcta, con respecto a la realidad.
- **Medida F (F-measure):** es una medida de combinación entre la precisión y el alcance, utilizada como media entre los dos evaluadores.

Métodos de Validación de Aprendizajes Supervisados

Validación Cruzada

La validación cruzada, a veces llamada estimación de la rotación, es una técnica de validación del modelo para evaluar cómo los resultados de un análisis estadístico se generalizarán a un conjunto de datos independiente. Se utiliza principalmente en entornos donde la meta es la predicción, y se quiere estimar la precisión con la que un modelo predictivo se llevará a cabo en la práctica. En un problema de predicción, a un modelo se le suele asignar un conjunto de datos, de los datos conocidos sobre los que se ejecuta el entrenamiento (conjunto de datos de formación) y un conjunto de datos desconocidos (o primeros datos) contra los que se prueba el modelo. El objetivo de la validación cruzada es definir un conjunto de datos para *probar* el modelo en la fase de entrenamiento (es decir, el conjunto de datos de validación), con el fin de limitar problemas como sobre ajuste.

La idea es dividir el set de datos totales abarcando un set de entrenamiento y un set de validación, lo cual se puede explicar en la Figura 1.6:

Una ronda de validación cruzada implica dividir una muestra de datos en subconjuntos complementarios, realizar el análisis en un subconjunto (denominado conjunto de entrenamiento) y validar el análisis en el otro subconjunto (denominado conjunto de validación o conjunto de pruebas). Para reducir la variabilidad, varias rondas de validación

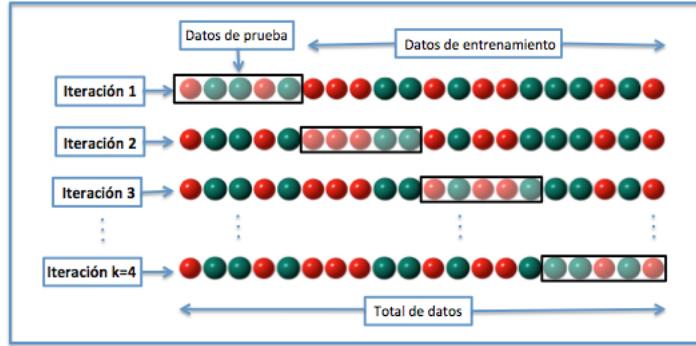


Figura 1.6: Esquema representativo de validación cruzada.

cruzada se realizan utilizando diferentes particiones, y los resultados de validación se promedian durante las rondas, siendo las más utilizadas *10-fold cross validation*.

Leave one out (Dejar uno)

Es un tipo especial de validación cruzada, en donde se tiene una muestra con n ejemplos en la etapa de entrenamiento se subdivide dicho set de datos considerando $n - 1$ elementos, de tal manera que 1 no se considera, la idea en particular radica en entrenar con los $n - 1$ ejemplos y validar o testear con el ejemplo restante, esto se itera n veces, tal como se expone en 1.7, implicando una mayor cantidad de iteraciones que validación cruzada, provocando además un mayor coste computacional.

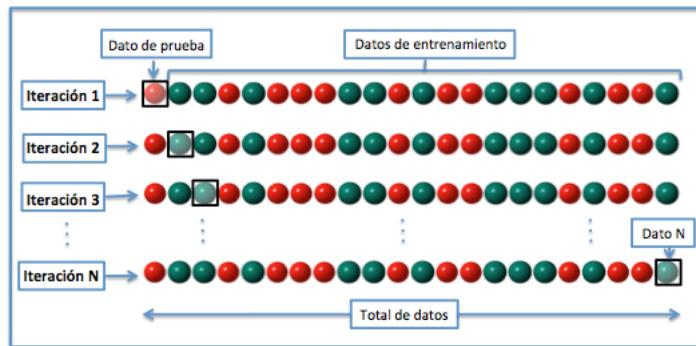


Figura 1.7: Esquema representativo de Leave One.

Principales problemas en Aprendizaje Supervisado

Dentro de los principales problemas que pueden presentar los modelos de aprendizaje supervisado se encuentran las situaciones en las que la cantidad de atributos que puede presentar un set de datos es muy mayor con respecto a la cantidad de ejemplos que se

posee, es decir si existen n ejemplos y la cantidad de atributos es $n \times n$ es posible que ocurra dicha problemática, para solucionar este problema, existen técnicas como PCA que permiten la reducción de atributos en base al aporte que entrega a la varianza total de la muestra. Otro posible problema que se puede denotar es el sobreajuste, esto quiere decir, que el modelo es extremadamente complejo, por lo que éste se ajusta muy bien al set de entrenamiento, no obstante a la hora de probar con nuevos set de datos no representa la performance obtenida.

Algoritmos de Aprendizaje No Supervisado

Es un método de Aprendizaje Automático donde un modelo es ajustado a las observaciones. Se caracteriza por el hecho de que no hay un conocimiento a priori. En el aprendizaje no supervisado, un conjunto de datos de objetos de entrada es tratado como un conjunto de variables aleatorias, siendo construido un modelo de densidad para el conjunto de datos.

Existen diversas formas de ver los algoritmos supervisados, ya sea en forma de reglas de asociación o como algoritmos propiamente tal, siendo explicadas ambas en los siguientes puntos.

Reglas de Asociación

Las reglas de asociación son un tipo de modelo asociado a aprendizaje no supervisado, el cual tiene por objetivo encontrar patrones de comportamiento de ejemplos, a partir de relaciones o asociaciones entre los atributos que caracterizan a estos, lo cual depende o está en función de la aparición de ciertos valores de uno, dos o más atributos.

Las reglas son del tipo *Si ocurre este evento y este atributo posee este valor, ocurre este acontecimiento*. Tiene una similitud a las reglas de decisión, siendo las reglas asociadas a atributos del tipo nominal, normalmente las reglas expresan combinaciones de valores de los atributos que suceden más frecuentemente.

Dentro de las aplicaciones se exponen:

- Asociar qué productos la gente compra habitualmente.
- Búsqueda de patrones en Internet.
- Bioinformática.
- En general, tareas asociadas a grandes volúmenes de datos.

Clusterización

La clusterización (clustering) se define como un método de aprendizaje no supervisado, en el cual se cuenta con un conjunto de datos que representan a una muestra y en base a dicha muestra de datos, se trata de obtener grupos de objetos, denominados clusters.

Los clusters deben cumplir con dos características fundamentales:

- Los objetos que pertenezcan a un mismo clúster deben ser bastante homogéneos entre ellos.
- Entre los clústers debe existir un alto grado de heterogeneidad.

En la clusterización se trata, fundamentalmente, de resolver el siguiente problema: Dado un conjunto de N individuos, caracterizados por la información de n variables X_j con j entre $1,..,n$, se plantea el reto de ser capaces de agruparlos de manera que los individuos pertenecientes a un grupo (cluster) (y siempre con respecto a la información disponible) sean tan similares entre sí como sea posible, siendo los distintos grupos entre ellos tan disimilares como sea posible.

Básicamente, el análisis constará de un algoritmo de clusterización que permitirá la obtención de una o varias particiones, de acuerdo con los criterios establecidos.

Criterios de Similitud

Tal como se ha mencionado anteriormente, el hecho de tener elementos pertenecientes a un mismo grupo bastante similares entre ellos y divergentes entre distintos clúster, es la característica primordial a tratar, esto último radica en la importancia de las variables que componen a un elemento en particular y en los valores que tomen éstas.

Por lo tanto se debe determinar que tan similares o diferentes son los valores que tomen las variables con respecto al elemento al cual pertenece.

Para medir lo similar o disimilar que son los individuos existe una enorme cantidad de índices de similaridad y de disimilaridad o divergencia. Todos ellos tienen propiedades y utilidades distintas y habrá que ser consciente de ellas para su correcta aplicación al momento de hacer uso de una de ellas.

Los índices expuestos normalmente pueden ser clasificados tal como sigue:

- Indicadores basados en la distancia, para lo cual se considera a los individuos como vectores en el espacio de las variables, en este sentido un elevado valor de la distancia entre dos individuos indicará un alto grado de disimilaridad entre ellos.

- Indicadores basados en coeficientes de correlación; la correlación permite indicar cuál es la fuerza y la dirección de una relación lineal entre dos variables. Se considera que dos variables cuantitativas están correlacionadas cuando los valores de una de ellas varían sistemáticamente con respecto a los valores de la otra, esto es: si se tiene dos variables (A y B) existe correlación si al aumentar los valores de A lo hacen también los de B y viceversa.
- Indicadores basados en tablas de datos de posesión o no de una serie de atributos, es decir teniendo dos vectores A y B que representan a dos individuos de una población se hace una comparación de los elementos existentes en A y no en B, los elementos existentes en B y no en A además de la intersección entre ellos, es decir, los elementos existentes en A y en B.

Criterios basados en distancias

La distancia o disimilaridad entre dos individuos i y j corresponde a una medida, indicada por $D_{(i,j)}$, que mide el grado de semejanza, entre ambos individuos, en relación a un cierto número de características cuantitativas y/o cualitativas. El valor de $D_{(i,j)}$ es siempre un valor positivo, y cuanto mayor sea este valor, mayor será la diferencia entre los individuos i y j .

Las propiedades que debe cumplir un indicador de distancia son:

- No negatividad, es decir, $D_{(i,j)} > 0$.
- $D_{(i,j)} = 0 \Leftrightarrow i = j$.
- Simetría, es decir, $D_{(i,j)} = D_{(j,i)}$.

Algunos de los tipos de distancias que pueden ser calculados son expuestos a continuación.

Distancia Euclíadiana

La distancia euclíadiana es la más conocida y más sencilla de comprender, pues su definición coincide con el concepto más común de distancia, es una recta que une dos puntos.

Su expresión es:

$$D_{(X,Y)} = \sqrt{\sum_{i=1}^l (X_i - Y_j)^2}$$

La distancia euclíadiana tiene dos graves inconvenientes:

- Es una distancia sensible a las unidades de medida de las variables: las diferencias entre los valores de variables medidas con valores altos contribuirán en mucha mayor medida que las diferencias entre los valores de las variables con valores bajos. Como consecuencia de ello, los cambios de escala determinarán, también, cambios en la distancia entre los individuos.
- Si las variables utilizadas están correlacionadas, estas variables darán una información, en gran medida redundante. Parte de las diferencias entre los valores individuales de algunas variables podrían explicarse por las diferencias en otras variables. Como consecuencia de ello la distancia euclíadiana inflará la disimilaridad o divergencia entre los individuos.

La distancia euclíadiana será, en consecuencia, recomendable cuando las variables sean homogéneas y estén medidas en unidades similares y/o cuando se desconozca la matriz de varianzas.

Distancia Manhattan

Es muy similar a la distancia euclíadiana, la diferencia radica en que se aplican distancia en zig-zag de los datos, se representa por: $D_{(X,Y)} = \sum_{i=1}^n |X_i - Y_i|$.

Distancia del Coseno

Considera cada ejemplo como un vector de n dimensiones, para los cuales estima el coseno del ángulo que forman, se representa por: $D_{(X,Y)} = \arccos\left(\frac{X^T Y}{\|X\| \|Y\|}\right)$

Distancia Mahalanobis

Esta distancia presenta propiedades que solucionan los inconvenientes de la aplicación de la distancia euclídeana: no varía a cambios de escala, por lo que no depende de las unidades de medida, además se consideran las correlaciones entre las variables, debido al uso de una matriz de covarianza entre los datos y se corrige el efecto de la redundancia. Se destaca además que no asume independencia entre los datos.

La distancia Mahalanobis se representa como: $D_{(X,Y)} = \sqrt{(X - Y)^T S^{-1} (X - Y)}$.

Criterios basados en similaridades

Existen indicadores que permiten medir el grado de homogeneidad entre los individuos, diferentes a las distancias expuestas en el punto anterior, conocidos como indicadores de

similitud.

Los indicadores de similitud indican que a medida que aumente su valor, la similitud entre los individuos será mayor.

La gran mayoría de los indicadores de similitud son basados en coeficientes de correlación o de asociación.

El Coeficiente de Correlación de Pearson utiliza preferentemente datos cuantitativos además del algoritmo de distancias mínimas, por otro lado, los Coeficientes de Correlación por rangos de Kendall y Spearman utilizan variables ordinales [12].

Existen otros criterios, tales como el índice binario y el índice de Tanimoto.

Algoritmos de Clustering

Existen diversos algoritmos de clustering, cada uno con características que los diferencian, los cuales, pueden ser aplicados a diversos casos, dependiendo de las características de los datos de entrada, es decir, de la geometría de estos datos, sin embargo, esta representación se basa principalmente en el uso de matrices, donde cada fila representa un ejemplo y cada columna el valor de un atributo o rasgo cualitativo para dicho ejemplo.

k-Medias (k-Means)

El algoritmo k-medias, trata la separación de muestras en n grupos de igual varianza, minimizando el criterio conocido como inercia, lo que se traduce en la suma de los cuadrados dentro de los clúster [13].

La principal característica y deficiencia a la vez, es que se requiere que el número de grupos sea entregado, es decir, se debe entregar el valor de k , así, si se selecciona un valor de $k = 3$, serán tres grupos los que se encontrarán.

Este algoritmo divide un set de N ejemplos X en K particiones distintas denominadas clúster C , cada uno de ellos descrito por la media μ_j de las muestras en el clúster. Esta media es llamada centroide, por lo que en general, k-medias elige sus centroides de tal manera que el principio de inercia sea reducido al mínimo, es decir, que la suma de los cuadrados de los integrantes de un mismo grupo sea mínima, a través de:

$$\sum_{i=0}^n \min_{\mu_j \in C} (\|x_j - \mu_i\|^2)$$

Sin embargo, el principio de inercia, o la suma de los cuadrados mínimos entre los integrantes de los clustering, puede sufrir varios inconvenientes:

- Se hace la suposición de que las agrupaciones son convexas e isotrópicas, lo cual no se da siempre, razón por la que responde mal ante a clusters que posean forma alargadas o con formas irregulares.
- No es una métrica normalizada, es decir, se sabe que los valores más bajos son mejores y el cero es óptimo, sin embargo, en espacios de muy alta dimensionalidad, las distancias euclidianas tienden a ser infladas, lo que se conoce como “la maldición de la dimensionalidad”, razón por la cual, son utilizados algoritmos de reducción de la dimensionalidad, tal como PCA.

Ambos puntos, son posibles observarlos en la Figura 1.8, en la cual se exponen, problemas con varianzas distintas, diferencias asociadas al tamaño de los clúster, anisotropía⁴ de los datos, etc.

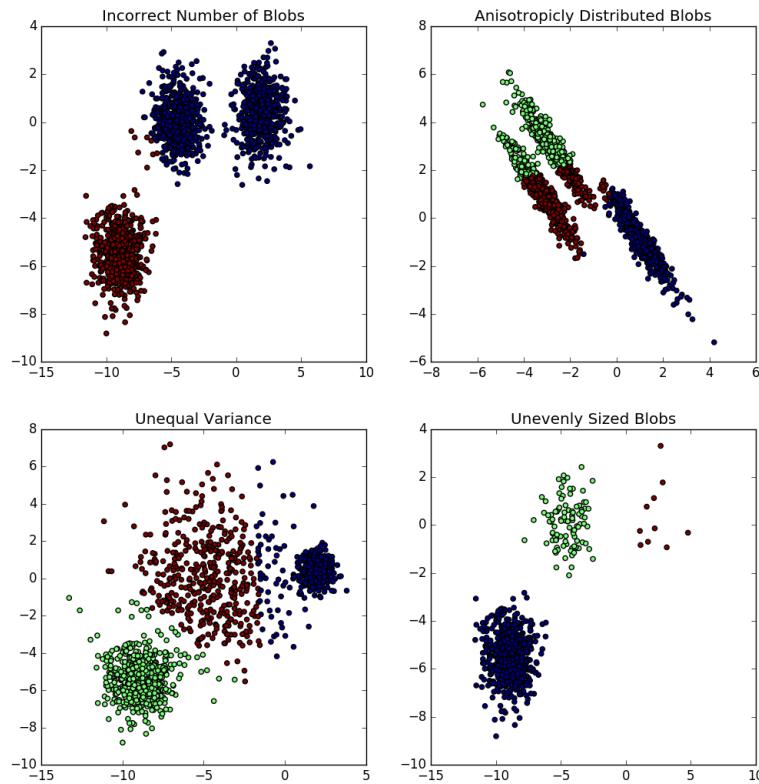


Figura 1.8: Posibles inconvenientes con los datos, donde k-medias no funciona correctamente

k-medias es referido a menudo como el algoritmo de Lloyd. En términos básicos, el algoritmo tiene tres pasos. El primer paso consiste en elegir los centroides iniciales, con

⁴Las variables varían en base a las direcciones en las que se examinan

el método más básico para elegir k muestras del conjunto de datos X . Después de la inicialización, k-medias consta de un bucle entre los otros dos pasos.

El primer paso asigna cada muestra a su centroide más cercano. En el segundo se crean nuevos centroides tomando el valor medio de todas las muestras asignadas a cada centroide anterior. La diferencia entre la media anterior y la actual (diferencia entre centroides) se calcula y se itera estas acciones hasta que este valor sea inferior a un umbral. En otras palabras, se repite hasta que los centroides no se mueven de manera significativa.

Una alternativa a k-medias , es conocida como el algoritmo **Mini lotes k-medias (Mini Batch K-means)** , cuya diferencia principal es que utiliza *mini segmentos* con el fin de optimizar el tiempo de cómputo, sin embargo, ambos cumplen con el mismo objetivo.

Estos “mini segmentos” son subconjuntos de los datos de entrada, seleccionados al azar en cada iteración de entrenamiento. Reducen drásticamente la cantidad de cálculo requerido para converger a una solución local.

El algoritmo itera entre dos pasos principales, similar a k-medias . En la primera etapa, las muestras b son elegidas aleatoriamente del conjunto de datos, para formar los mini segmentos. Se continúa con la asignación de estos al centroide más cercano. En el segundo paso, los centroides se actualizan. La principal diferencia, radica en cómo se efectúa la actualización de los centroides, debido a que esto se hace en función de cada muestra. Estos pasos se llevan a cabo hasta la convergencia o se alcance un número predeterminado de iteraciones.

Affinity Propagation (Propagación por Afinidad)

AffinityPropagation crea grupos mediante el envío de mensajes entre pares de muestras hasta la convergencia. Un conjunto de datos es descrito por el uso de un pequeño número de ejemplares, que se identifican como las más representativas de otras muestras. Los mensajes enviados entre pares representan la idoneidad para una muestra a ser el ejemplo de la otra, la cual se actualiza en respuesta a los valores de otros pares. Esta actualización ocurre de forma iterativa hasta la convergencia, momento en el que se eligen los ejemplares finales, y por lo tanto se da el agrupamiento final [14].

Se elige el número de grupos en base a los datos proporcionados. Para este propósito, los dos parámetros importantes son la preferencia, que controla el número de ejemplares que se utilizan, y el factor de amortiguamiento.

El principal inconveniente que presenta este algoritmo viene dado por la complejidad que posee, el cual se representa por $O(N^2T)$, donde N es el número de muestras y T

es el número de operaciones necesarias para convergir, razón por la cual, el uso de este algoritmo es para set de datos con pequeña cantidad de ejemplos.

Con respecto a la descripción del algoritmo, es posible mencionar que los mensajes enviados a los grupos, pertenecen a dos categorías, la primera es la responsabilidad $r(i, k)$ la cual consiste en la evidencia acumulada que denota que la muestra k podría ser un ejemplar para la muestra i . La segunda es la disponibilidad $a(i, k)$, la cual se define como la evidencia existente para que la muestra i pueda escoger a la muestra k para ser su ejemplar, además considera todos los valores de las otras muestras de k que podrían ser ejemplares. De esta manera, los ejemplares son escogidos por las muestras si:

- Existe una similaridad bastante alta con respecto a las muestras.
- Si es elegido por muchas muestras y resulta ser representativo de sí mismos.

En forma matemática es posible definir la responsabilidad como:

$$r(i, k) \leftarrow s(i, k) - \max[a(i, \hat{k}) + s(i, \hat{k})] \forall \hat{k} \neq k$$

Donde $s(i, k)$ es la similaridad entre las muestras i y k .

A su vez, la disponibilidad, es posible definirla como:

$$a(i, k) \leftarrow \min[0, r(k, k) + \sum_{\substack{i \in \text{samples} \\ i \neq \{i, k\}}} r(i, k)]$$

Variación Media (Mean Shift)

El algoritmo de Mean Shift tiene como objetivo descubrir manchas (blobs) en una densidad uniforme de las muestras. Es un algoritmo basado en centroides, que funciona mediante la actualización de los candidatos para centroides para ser la media de los puntos dentro de una región determinada. Estos candidatos se filtran en una etapa de post-procesamiento para eliminar duplicados y así formar el conjunto final de centroides.

Dado un candidato x_i para la iteración t , el candidato a centroide es actualizado en base a la ecuación:

$$x_i^{t+1} = x_i^t + m(x_i^t)$$

Donde $N(x_i)$ es la vecindad de las muestras dentro de una distancia dada alrededor x_i y m es el vector de desplazamiento medio, que se calcula para cada centroide que apunta hacia una región del aumento máximo en la densidad de puntos. Ésta se calcula utilizando la siguiente ecuación, en la que la actualización efectiva denota a un centroide ser la media de las muestras dentro de su vecindad:

$$m(x_i) = \frac{\sum_{x_j \in N(x_i)} K(x_j - x_i) x_j}{\sum_{x_j \in N(x_i)} K(x_j - x_i)}$$

El algoritmo ajusta automáticamente el número de grupos, dependiendo de un parámetro bandwidth, el cual se asocia al tamaño de la región en la que se debe buscar los centroides. El algoritmo no es altamente escalable, ya que requiere múltiples búsquedas de vecinos más cercanos durante la ejecución del algoritmo. El algoritmo se garantiza a converger, sin embargo, se detendrá la iteración cuando el cambio en centroides es pequeño.

Clustering Jerarquizado

La agrupación jerárquica o HCA (por sus siglas en inglés) es un método de análisis de conglomerados, que busca construir una jerarquía de agrupaciones. Las estrategias para la agrupación son posible dividirlas en dos:

- **Aglomerativa:** consiste en un enfoque de “abajo hacia arriba”: cada observación se inicia en su propio clúster, y pares de grupos se fusionan a medida que se asciende en la jerarquía.
- **Divisiva:** consiste en un enfoque “de arriba hacia abajo”: todas las observaciones se inician en un único clúster, y las divisiones se realizan de forma recursiva conforme se desciende en la jerarquía.

Siendo generalmente expuestos los resultados en forma de dendrograma, por otro lado, la complejidad del algoritmo, en el caso general es $O(n^2 \log(n))$, lo cual presenta problemas para set de datos extensos [15].

Con el fin de decidir qué grupos se deben combinar (por aglomeración), o cuando un grupo se debe dividir (por división), se requiere una medida de disimilitud entre los conjuntos de observaciones. En la mayoría de los métodos de agrupación jerárquica, esto se logra mediante el uso de una métrica apropiada (una medida de la distancia entre pares de observaciones), y un criterio de vinculación que especifica la disimilitud de conjuntos como una función de las distancias por pares de observaciones en los conjuntos.

Las métricas asociadas pueden ser las siguientes [8]:

- **Distancia Euclíadiana:** $\|a - b\|_2 = \sqrt{\sum_i (a_i - b_i)^2}$.
- **Distancia Euclíadiana cuadrática:** $\|a - b\|_2^2 = \sum_i (a_i - b_i)^2$.
- **Distancia Manhattan:** $\|a - b\|_1 = \sum_i |a_i - b_i|$.
- **Distancia Máxima:** $\|a - b\|_\infty = \max_i |a_i - b_i|$

- **Distancia de Mahalonobis:** $\sqrt{(a - b)^\top S^{-1}(a - b)}$, donde S es la matriz de covarianza.

El criterio de linkage determina la distancia entre conjuntos de observaciones como una función de las distancias por pares entre observaciones.

Algunos criterios de linkage de uso común entre los dos conjuntos de observaciones A y B son:

- **Acoplamiento máximo (complete linkage clustering):** $\max \{ d(a, b) : a \in A, b \in B \}$.
- **Acoplamiento mínimo (single-linkage clustering):** $\min \{ d(a, b) : a \in A, b \in B \}$.
- **Acoplamiento por la Media (average linkage clustering, UPGMA):**

$$\frac{1}{\|A\|\|B\|} \sum_{a \in A} \sum_{b \in B} d(a, b).$$
- **Acoplamiento por los centroides (Centroid linkage clustering, UPGMC):**

$$\|c_s - c_t\|$$
 donde c_s y c_t son los centroides de los clusters s y t , respectivamente.
- **Mínimo energía (Minimum energy clustering):** $\frac{2}{nm} \sum_{i,j=1}^{n,m} \|a_i - b_j\|_2 - \frac{1}{n^2} \sum_{i,j=1}^n \|a_i - a_j\|_2 - \frac{1}{m^2} \sum_{i,j=1}^m \|b_i - b_j\|_2.$

Un aspecto interesante de este algoritmo es que pueden ser añadidas las limitaciones de conectividad, es decir, sólo grupos adyacentes pueden fusionarse entre sí, esto es, a través de una matriz de conectividad que define para cada muestra, las muestras de vecinos después de una estructura dada de los datos. Estas restricciones son útiles para imponer una cierta estructura local, así como para hacer que el algoritmo sea más rápido, especialmente cuando el número de las muestras es alta.

DBSCAN

El algoritmo DBSCAN ve agrupaciones como áreas de alta densidad separadas por zonas de baja densidad. Debido a esta visión bastante genérica, las agrupaciones que se encuentran pueden ser de cualquier forma, en lugar de k-medias que supone que los grupos tienen la forma convexa [16].

El componente central de DBSCAN es el concepto de muestras de núcleo, las cuales son las muestras que se encuentran en áreas de alta densidad. Por lo tanto, un clúster es un conjunto de muestras de núcleos, cada uno cerca del otro (medido por alguna medida de

distancia) y un conjunto de muestras no básicas que se encuentran cerca de una muestra básica. Hay dos parámetros necesarios para el algoritmo, min samples y EPS, los cuales definen formalmente la densidad deseada.

Más formalmente, se define una muestra del núcleo como una muestra del conjunto de datos de tal manera que existe una cantidad de muestra mínimas y a su vez otras muestras dentro de una distancia de EPS, que se definen como vecinos de la muestra del núcleo. Esto dice que la muestra de núcleo se encuentra en un área densa del espacio vectorial. Un clúster es un conjunto de muestras de núcleo que se puede construir mediante la adopción de forma recursiva de una muestra básica, la búsqueda de todos sus vecinos que son muestras de la base, la búsqueda de la totalidad de sus vecinos que son muestras de núcleo, y así sucesivamente. Un grupo también tiene un conjunto de muestras no básicas, que son las muestras que son vecinos de una muestra básica de la agrupación, pero no son en sí mismos muestras de núcleos. Intuitivamente, estas muestras están al margen de un clúster.

Cualquier muestra de núcleo es parte de un clúster, por definición. Cualquier muestra que no es una muestra del núcleo, y está al menos una distancia eps de cualquier muestra del núcleo, se considera un valor atípico por el algoritmo.

Normalmente, los resultados del algoritmo, pueden representarse tal como se expone en la Figura 1.9, el color indica la pertenencia al clúster, con grandes círculos que indican muestras de núcleos encontrados por el algoritmo, círculos más pequeños son muestras no básicas que todavía son parte de un clúster. Por otra parte, los valores atípicos se indican con puntos negros.

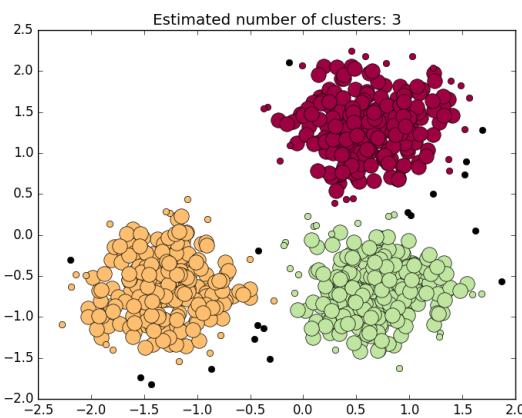


Figura 1.9: Representación de resultados al aplicar la clusterización por DBSCAN

Birch

Birch construye un árbol llamado Characteristic Feature Tree (CFT) de los datos correspondientes. Los datos están esencialmente con pérdida de información, comprimidos en un conjunto de nodos de rasgo característico denominados CF nodos. Estos tienen una serie de subgrupos llamados subclusters de rasgo característico, ubicados en los nodos CF no terminales los cuales pueden tener CF nodos como hijos.

Los subgrupos CF pueden contener la información necesaria para la agrupación que evite la necesidad de mantener los datos de entrada enteros en la memoria. Esta información incluye:

- Número de muestras en un subgrupo.
- Suma lineal, representada por un vector n-dimensional que sostiene la suma de todas las muestras.
- Suma al cuadrado, representada por la suma cuadrática de la norma L2 de todas las muestras.
- Centroides, para evitar un nuevo cálculo de sumas lineales con respecto al número de muestras.
- Norma al cuadrado de los centroides.

El algoritmo de Birch tiene dos parámetros, el umbral y el factor de branching. El factor de branching limita el número de subgrupos en un nodo y el umbral limita la distancia entre la muestra de entrada y los subclusters existentes.

Este algoritmo puede ser visto como un método de instancia o reducción de datos, ya que reduce los datos de entrada a un conjunto de subclusters que se obtienen directamente de las hojas de la CFT. Estos datos reducidos pueden ser procesados por la alimentación en un clúster global. Este clúster global puede ser establecido por n clusters. Si este parámetro se establece como valor 0 o ninguno, los subgrupos de las hojas se leen directamente, de lo contrario un paso global de la agrupación etiqueta estos subgrupos en grupos globales y las muestras se asignan a la etiqueta global del subgrupo más cercano.

Una descripción del algoritmo, es posible realizarla en los siguientes puntos:

- Una nueva muestra se inserta en la raíz del árbol CF que es un nodo CF. A continuación, se fusiona con el subgrupo de la raíz, el que tiene el radio más pequeño después de la fusión, limitada por el umbral de ramificación y condiciones de los

factores. Si el subcluster tiene algún nodo hijo, entonces esto se realiza repetidamente hasta que llega a una hoja. Después de encontrar el subcluster más cercano en la hoja, las propiedades de este subgrupo y los subclusters padres se actualizan de forma recursiva.

- Si el radio del subcluster obtenido mediante la fusión de la nueva muestra y el subgrupo más cercano es mayor que el cuadrado del umbral y si el número de subclusters es mayor que el factor de ramificación, a continuación, un espacio se asigna temporalmente a esta nueva muestra. Los dos subgrupos más lejanos se toman y de los subgrupos se dividen en dos grupos sobre la base de la distancia entre estos subgrupos.
- Si este nodo de división tiene un subgrupo de los padres y no hay espacio para un nuevo subgrupo, entonces el padre se divide en dos. Si no hay espacio, entonces este nodo se divide de nuevo en dos y el proceso se continúa de forma recursiva, hasta que llega a la raíz

Mixture Model

Los métodos de clustering basado en modelos tratan de optimizar el conjunto de datos a un modelo matemático. En general estos métodos se basan en la suposición que los datos han sido generados por una mezcla de distribuciones de probabilidad. Dentro de los más utilizados se encuentran **Gaussian Mixture** y **Expectation-Maximization**.

Expectation Maximization, supone que los datos emergen de una mezcla de distribuciones, donde cada distribución se denomina como *component distribution*, razón por la cual, los datos pueden agruparse usando un modelo de mezcla de densidades de k distribuciones de probabilidades, sin embargo, el problema reside en estimar los parámetros de estas distribuciones para proveer del mejor ajuste posible a los datos.

El algoritmo Expectation Maximization puede ser considerado como una extensión de k-medias , esto es debido a que: Si k-means asigna cada objeto a un clúster en función de su media, EM asigna cada objeto a un clúster en función de un peso que representa la probabilidad de pertenencia al clúster. Esto requiere que se defina una distribución de probabilidad para los clusters.

Por otro lado, un modelo de mezcla gaussiano como Gaussian Mixture Model (GMM) es una función de densidad de probabilidad representada por una suma de componentes gausianas, GMMs son usadas como modelos paramétricos de la distribución de probabilidad de medidas continuas, donde los parámetros de GMM son estimados

usando iterativamente el algoritmo Expectation-Maximization.

Un GMM es una suma con pesos de densidades gaussianas:

$$p(\vec{x}) = \sum_{i=1}^M w_i \times N(\vec{x}|\mu_i, \Sigma_i)$$

Donde \vec{x} es un vector D-dimensional de datos, w_i son los pesos con $\sum_{i=1}^M w_i = 1$, y $N(\vec{x}|\mu_i, \Sigma_i)$ es la densidad gaussiana, por lo tanto, la caracterización se completa con la media, la matriz de covarianza y el peso de cada componente gaussiana.

Una de las limitantes es que el número de componentes gausianos tiene que ser fijado al principio del algoritmo.

El GMM consta de los siguientes pasos:

1. **Iniciacilización:** para cada clase, un vector compuesto de la media y la matriz de covarianza es construido. Este vector representa las características de la distribución gaussiana usada para caracterizar las entidades del conjunto de datos. Inicialmente estos valores son generados aleatoriamente, posteriormente el algoritmo EM trata de aproximar los valores del vector de la distribución real de los datos.
2. Se estima la probabilidad de cada elemento de pertenecer a un clúster.
3. Se estiman los parámetros de la distribución de probabilidad para el próximo ciclo, primero se calcula la media de la clase a través de la media de todos los puntos en función del grado de relevancia de cada punto, continuando con el cálculo de la matriz de covarianza.
4. **Convergencia:** Después de cada ciclo se ejecuta un test de convergencia para verificar cuánto ha cambiado el vector de parámetros y si la diferencia es menor que un umbral de tolerancia el algoritmo se detiene, no obstante es posible detener el algoritmo debido al alcance de un número máximo de ciclos.

Una representación visual del modelo es posible observarla en la Figura 1.10, en ella se aprecia cómo a medida que se va iterando el algoritmo se generan los cambios y las separaciones en grupos de clúster definidos.

Cuadro Resumen

En la Tabla 1.1 se expone un resumen de las características de cada algoritmo expuesto, la escalabilidad que poseen, las distancias que ocupan, los casos de uso y los parámetros que poseen.

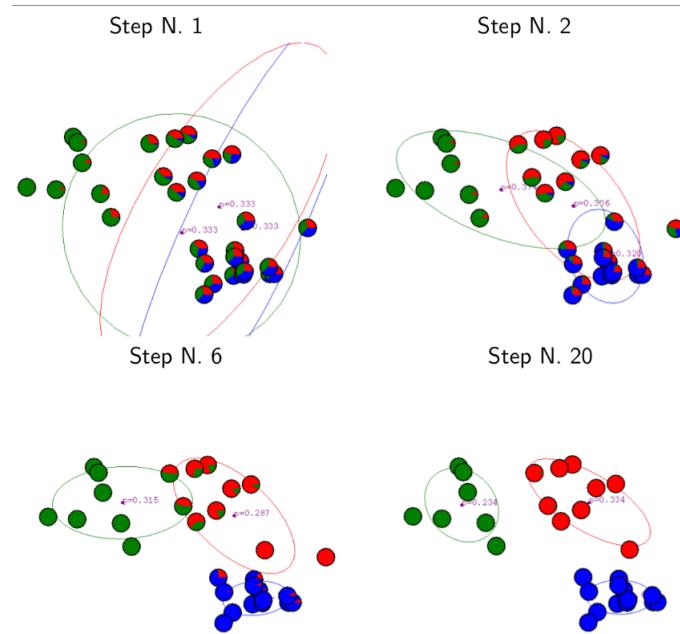


Figura 1.10: Esquema representativo de cambios durante las iteraciones en GMM

Tabla resumen de Algoritmos de Aprendizaje No Supervisado

Algoritmo	Parámetros	Escalabilidad	Usos	Métrica usada

K-Means	Número de clúster	Muchas muestras, mediana cantidad de clúster.	De propósito general, la geometría plana, no demasiados grupos	Distancia entre puntos
Affinity propagation	preferencia	No escalable con n ejemplos	Muchos clúster, tamaño de clúster desigual, geometría no plana	Distancia gráfica

Mean-shift	bandwidth	No escalable con n ejemplos	Muchos clúster, tamaño de clúster desigual, geometría no plana	Distancia entre puntos
Ward hierarchical clustering	Número de clúster	Mucha cantidad de ejemplos y de clusters	Cualquier clúster, es posible conexión de constraints	Distantia entre puntos

			Muchos clusters, posiblemente restricciones de conectividad, distancias no euclidianas	
Agglomerative clustering	Número de clúster, tipo de unión, distancia	Mucha cantidad de ejemplos y de clusters	Cualquier distancia pairwise	
DBSCAN	tamaño vecino	Mucha cantidad de ejemplos, mediana cantidad de clúster	Geometría no plana, tamaños de clusters distintos	Distancia entre puntos vecinos

Gaussian mixtures	variado	No escalable	Geometría plana, bueno para la estimación de la densidad	Distancia Mahalanobis para los centros
Birch	branching, umbral	Alto número de clúster y ejemplos	Largo set de datos, eliminación valores atípicos, reducción de datos	distancia euclidiana entre puntos

Tabla 1.1: Cuadro resumen de algoritmos de aprendizaje supervizado

Evaluación del desempeño de un clustering

Evaluar el desempeño de un algoritmo de clustering no es tan trivial como contar el número de errores o la precisión y la recuperación de un algoritmo de clasificación supervisada. En particular, cualquier métrica de evaluación no debe tomar los valores absolutos de las etiquetas de clúster en cuenta, sino más bien si estas agrupaciones definen

separaciones de los datos, de tal manera que los miembros que pertenecen a la misma clase son más similares que los miembros de diferentes clases de acuerdo con alguna similitud métrica.

Existen diversas medidas de similitud con el fin de evaluar el clustering, las cuales se explican a continuación:

Índice Rand ajustado (Adjusted Rand index)

Dado el conocimiento de las clases asignadas como verdaderas (etiquetas verdaderas) y las etiquetas obtenidas por el algoritmo de clustering (etiquetas predichas) el adjuster rand index es una función que mide la similaridad de las dos asignaciones, ignorando permutaciones, posee valores entre -1 y 1, siendo 1 el valor perfecto. Sin embargo, es imperante para evaluar el desempeño, conocer las etiquetas verdaderas de los datos.

Matemáticamente es posible definirlo como:

Sea C una asignación de clase real y dada la agrupación K , se define a y b como:

- a , el número de pares de elementos que están en el mismo set en C y en el mismo set en K .
- b , el número de pares de elementos que están en diferentes set en C y en diferentes set en K .

El valor del rand index no ajustado viene dado por:

$$RI = \frac{a+b}{C_2^{n_{samples}}}$$

Donde $C_2^{n_{samples}}$ es el número total de posibles pares en el set de datos.

Sin embargo, la puntuación de RI no garantiza que las asignaciones de etiquetas al azar conseguirán un valor cercano a cero, para contrarrestar este efecto se puede descartar la espera RI $E[RI]$ de etiquetas al azar mediante la definición del adjusted rand index:

$$ARI = \frac{RI - E[RI]}{\max(RI) - E[RI]}$$

Información mutua basada en scores

Dado el conocimiento de las etiquetas de las clases reales y las asignaciones obtenidas de algoritmos de agrupación de las mismas muestras, el mutual information es una función que mide el *acuerdo* de las dos asignaciones, ignorando las permutaciones. Existen dos versiones normalizadas diferentes de esta medida:

Normalized Mutual Information, NMI (Información mutua normalizada) y **Adjusted Mutual Information, AMI (Información mutua ajustada)**. NMI es a menudo usado en la literatura mientras que AMI fue propuesto más recientemente.

Matemáticamente, es posible definir esta forma de evaluación tal que: se asume dos etiquetas asignadas (de los mismos N objetos), U y V , su entropía es la cantidad de incertidumbre para un conjunto de particiones definido por:

$$H(U) = \sum_{i=1}^{|U|} P(i) \log(P(i))$$

donde $P(i) = |U_i|/N$ es la probabilidad que un objeto seleccionado aleatoriamente de la clase U sea asignado a la clase U_i , de igual manera para V :

$$H(V) = \sum_{j=1}^{|V|} P'(j) \log(P'(j))$$

Con $P'(j) = |V_j|/N$ el mutual information (MI) entre U y V es calculado por:

$$MI(U, V) = \sum_{i=1}^{|U|} \sum_{j=1}^{|V|} P(i, j) \log\left(\frac{P(i, j)}{P(i)P'(j)}\right)$$

donde $P(i, j) = |U_i \cap V_j|/N$ es la probabilidad de que un objeto seleccionado aleatoriamente sea asignado a ambas clases U_i y V_j .

El valor normalizado del mutual information es definido como:

$$NMI(U, V) = \frac{MI(U, V)}{\sqrt{H(U)H(V)}}$$

Este valor del mutual information y también la variante normalizada no se ajusta al azar y tiende a aumentar a medida que aumenta el número de diferentes etiquetas (clusters), independientemente de la cantidad real de *mutual information* entre las asignaciones de etiquetas.

El valor esperado para el mutual information puede ser calculado usando la ecuación descrita por Vinh, Epps, and Bailey, (2009) [17]. En esta ecuación, $a_i = |U_i|$ (el número de elementos en U_i) y $b_j = |V_j|$ (el número de elementos en V_j).

$$E[MI(U, V)] = \sum_{i=1}^{|U|} \sum_{j=1}^{|V|} \sum_{n_{ij}=(a_i+b_j-N)^+}^{\min(a_i, b_j)} \frac{n_{ij}}{N} \log\left(\frac{\frac{a_i!b_j!(N-a_i)!(N-b_j)!}{N!n_{ij}!(a_i-n_{ij})!(b_j-n_{ij})!(N-a_i-b_j+n_{ij})!}}{\frac{a_i!b_j!(N-a_i)!(N-b_j)!}{N!n_{ij}!(a_i-n_{ij})!(b_j-n_{ij})!(N-a_i-b_j+n_{ij})!}}\right)$$

Usando el valor esperado, el adjusted mutual information puede ser calculado usando una forma similar al ARI:

$$AMI = \frac{MI - E[MI]}{\max(H(U), H(V)) - E[MI]}$$

Homogeneidad, Completación y medida V (V-measure)

Para el caso en el que se conozca a ciencia cierta las etiquetas reales de las clases, es posible definir medidas de evaluación basándose en la entropía existente. En particular Rosenberg y Hirschberg (2007) [?] definen los siguientes dos objetivos deseables para cualquier asignación de clusters:

- **homogeneidad:** cada clúster contiene sólo miembros de una única clase.
- **Totalidad (completeness):** todos los miembros de una clase son asignados a un mismo clúster.

Los valores de estos score abarcan los rangos entre 0 y 1, siendo 1 el score perfecto.

Es posible definir la homogeneidad y el completeness como:

$$h = 1 - \frac{H(C|K)}{H(C)}$$

$$c = 1 - \frac{H(K|C)}{H(K)}$$

donde $H(C|K)$ es la entropía condicional de las clases dada la asignación del clúster y es definida por:

$$H(C|K) = - \sum_{c=1}^{|C|} \sum_{k=1}^{|K|} \frac{n_{c,k}}{n} \cdot \log\left(\frac{n_{c,k}}{n_k}\right)$$

y $H(C)$ es la entropía de la clase y cuyo valor viene dado por:

$$H(C) = - \sum_{c=1}^{|C|} \frac{n_c}{n} \cdot \log\left(\frac{n_c}{n}\right)$$

con n siendo el número total de muestras, n_c y n_k el número de muestras respectivamente pertenecientes a la clase c y al clúster k , y finalmente $n_{c,k}$ el número de muestra de la clase c asignados al clúster k .

Rosenberg y Hirschberg también definieron un **V-measure** como el score medio de la homogeneidad y completeness:

$$v = 2 \cdot \frac{h \cdot c}{h + c}$$

Coeficiente de silueta (Silhouette Coefficient)

Este coeficiente es posible utilizarlo cuando se desconocen las reales etiquetas de los ejemplos, una puntuación alta (por sobre 0.75) denota un modelo con grupos bien definidos. Este coeficiente se define para cada muestra y posee dos score [18]:

- **a:** la distancia media entre un ejemplo y todos los otros puntos en la misma clase.
- **b:** la distancia media entre un ejemplo y todos los otros puntos en el siguiente clúster vecino.

El coeficiente para una única muestra, viene dado por:

$$s = \frac{b-a}{\max(a,b)}$$

Calinski-Harabaz Index

Este índice es utilizado cuando las etiquetas son desconocidas, donde un mayor valor de éste implica un modelo mejor definido [19].

Para k clusters, el Calinski-Harabaz index s se da como la razón de la dispersión entre clusters y la dispersión dentro del grupo:

$$s(k) = \frac{\text{Tr}(B_k)}{\text{Tr}(W_k)} \frac{N-k}{k-1}$$

Donde B_K es la matriz de dispersión entre grupos y W_K es la matriz de dispersión dentro del clúster definida por:

$$\begin{aligned} W_k &= \sum_{q=1}^k \sum_{x \in C_q} (x - c_q)(x - c_q)^T \\ B_k &= \sum_q n_q (c_q - c)(c_q - c)^T \end{aligned}$$

Con N como el de puntos en el set de datos, C_q el set de puntos en el cluster q , c_q el centro del clúster q , c el centro de E , n_q el número de puntos en el clúster q .

Otros Métodos de Aprendizajes en Minería de Datos

Redes Neuronales

Redes neuronales es posible definirlas como una serie de modelos de aprendizaje que se basan en la forma de trabajo de las redes neuronales biológicas, es decir, se usa el concepto de *neurona* para estimar una función aproximada, la cual dependerá de un largo número de inputs, generalmente desconocidos.

En la imagen 1.11 se aprecia un sistema de red neuronal, en la cual se observa un sistema interconectado por neuronas, las cuales intercambian información en forma de mensaje entre ellas, además cada interconexión tiene un peso, el cual es un valor numérico, que puede ser obtenido en base a la experiencia, en resumen, una red neuronal es un conjunto de entradas y salidas regidas por capas intermedias que permiten evaluar la salida, dichas capas operan entre sí en base a funciones matemáticas y brindan un peso a la conexión, finalmente cada capa es usada para diseñar un modelo de aprendizaje supervisado o no.

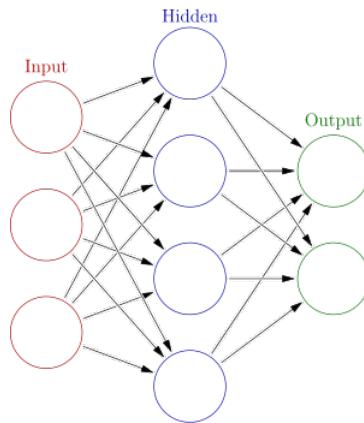


Figura 1.11: Representación esquemática de una Red Neuronal

Deep Learning

Deep Learning es una herramienta de Machine Learning la cual tiene por objetivo modelar abstracciones de alto nivel en los datos por medio del uso de múltiples capas de procesamiento, ya sea a través del uso de estructuras complejas a través de múltiples transformaciones [20][22][23].

La investigación en esta área tiene por objetivo generar mejores representaciones y crear modelos para aprender de estas representaciones a partir de datos no marcados a gran escala. En general las representaciones obtenidas se inspiran en los avances en la neurociencia y se basa libremente en la interpretación de los patrones de procesamiento y comunicación de información en un sistema nervioso, como la codificación neural que intenta definir una relación entre varios estímulos y respuestas neuronales asociadas en el cerebro[19-20-21].

Deep learning posee diversas arquitecturas, tales como: deep learning network, matrices de convoluciones, redes neuronales recurrentes, etc. las cuales han sido utilizadas en visión artificial para el reconocimiento de patrones, aprendizaje de escritura, etc. Deep Learning es una herramienta de Machine Learning la cual tiene por objetivo modelar abstracciones

alto nivel en los datos por medio del uso de múltiples capas de procesamiento, ya sea a través del uso de estructuras complejas a través de múltiples transformaciones no lineales [24][25][26].

Algoritmos Genéticos

Los algoritmos genéticos se basan en la evolución biológica y su base genético-molecular. Estos algoritmos hacen evolucionar una población de individuos sometiéndola a acciones aleatorias semejantes a las que actúan en la evolución biológica (mutaciones y recombinaciones genéticas), así como también a una selección de acuerdo con algún criterio, en función del cual se decide cuáles son los individuos más adaptados, que sobreviven, y cuáles los menos aptos, que son descartados [27] [28].

Los algoritmos genéticos se enmarcan dentro de los algoritmos evolutivos, que incluyen también las estrategias evolutivas, la programación evolutiva y la programación genética.

Los algoritmos genéticos funcionan entre el conjunto de soluciones de un problema llamado fenotipo, y el conjunto de individuos de una población natural, codificando la información de cada solución en una cadena, generalmente binaria, llamada cromosoma. Los símbolos que forman la cadena son llamados los genes.

Cuando la representación de los cromosomas se hace con cadenas de dígitos binarios se le conoce como genotipo. Los cromosomas evolucionan a través de iteraciones, llamadas generaciones. En cada generación, los cromosomas son evaluados usando alguna medida de aptitud. Las siguientes generaciones (nuevos cromosomas), son generadas aplicando los operadores genéticos repetidamente, siendo estos los operadores de selección, cruzamiento, mutación y reemplazo.

Un algoritmo genético puede presentar diversas variaciones, dependiendo de cómo se aplican los operadores genéticos (cruzamiento, mutación), de cómo se realiza la selección y de cómo se decide el reemplazo de los individuos para formar la nueva población. En general, el pseudocódigo consiste de los siguientes pasos:

- **Inicialización:**

Se genera aleatoriamente la población inicial, que está constituida por un conjunto de cromosomas los cuales representan las posibles soluciones del problema. En caso de no hacerlo aleatoriamente, es importante garantizar que dentro de la población inicial, se tenga la diversidad estructural de estas soluciones para tener una representación de la mayor parte de la población posible o al menos evitar la convergencia prematura.

- **Evaluación:**

A cada uno de los cromosomas de esta población se aplicará la función de aptitud para saber cómo *buena* es la solución que se está codificando.

■ **Condición de término:**

El Algoritmo Genético se deberá detener cuando se alcance la solución óptima, pero ésta generalmente se desconoce, por lo que se deben utilizar otros criterios de detención. Normalmente se usan dos criterios: correr el Algoritmo Genético un número máximo de iteraciones (generaciones) o detenerlo cuando no haya cambios en la población. Mientras no se cumpla la condición de término se hace lo siguiente:

- **Selección:** Después de saber la aptitud de cada cromosoma se procede a elegir los cromosomas que serán cruzados en la siguiente generación. Los cromosomas con mejor aptitud tienen mayor probabilidad de ser seleccionados.
- **Recombinación o Cruzamiento:** La recombinación es el principal operador genético, representa la reproducción sexual, opera sobre dos cromosomas a la vez para generar dos descendientes donde se combinan las características de ambos cromosomas padres.
- **Mutación:** modifica al azar parte del cromosoma de los individuos, y permite alcanzar zonas del espacio de búsqueda que no estaban cubiertas por los individuos de la población actual.
- **Reemplazo:** una vez aplicados los operadores genéticos, se seleccionan los mejores individuos para conformar la población de la generación siguiente.

1.2.3. Sistemas de Información y Modelo Vista Controlador

Los Sistemas de información son la base de toda disposición de información, ya sea a través de una herramienta o por medio de una plataforma web, en general consta con un sistema de almacenamiento persistente, un conjunto de herramientas que permiten extraer y manejar la información con respecto a dicho almacenamiento y módulos de consulta dispuestos en herramientas de visualización de los datos. Normalmente su desarrollo se basa en el conjunto de metodologías de ingeniería en software a través de las cuales es analizado, diseñado e implementado el sistema, mediante el desarrollo de diversos artefactos de software, tales como: casos de uso, diagramas de clases, arquitectura de software, etc. Dentro de las arquitecturas más utilizadas se encuentran: arquitectura 3 capas, cliente-servidor, modelo Vista Controlador, siendo esta última la que se utilizará para el desarrollo de la herramienta computacional.

El modelo vista controlador (MVC) es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son: el modelo, la vista y el controlador.

Por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario.[29] [30].

Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento [31] [32]

Se puede definir los componentes del modelo como sigue:

- **Modelo:** Es la representación de la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio). Envía a la *vista* aquella parte de la información que en cada momento se le solicita para que sea mostrada (típicamente a un usuario). Las peticiones de acceso o manipulación de información llegan al *modelo* a través del *controlador*.
- **Controlador:** Responde a eventos (usualmente acciones del usuario) e invoca peticiones al *modelo* cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos). También puede enviar comandos a su *vista* asociada si se solicita un cambio en la forma en que se presenta el *modelo* (por ejemplo, desplazamiento o scroll por un documento o por los diferentes registros de una base de datos), por tanto se podría decir que el *controlador* hace de intermediario entre la *vista* y el *modelo*.
- **Vista:** Presenta el *modelo* (información y lógica de negocio) en un formato adecuado para interactuar (usualmente la interfaz de usuario) por tanto requiere de dicho *modelo* la información que debe representar como salida.

2. Hipótesis

Se desea diseñar e implementar modelos de clasificación para la evaluación de relevancia clínica de mutaciones reportadas en proteína pVHL. Siendo la hipótesis:

Es factible el desarrollo de modelos de clasificación para mutaciones en pVHL, cuyas medidas de desempeño sean médicaamente aceptables y sea replicable a otras proteínas de interés.

Sin embargo. Durante el desarrollo del trabajo expuesto se quiere dar respuesta a las siguientes interrogantes, las cuales se exponen como hipótesis complementarias o sub hipótesis en este proyecto.

- La adición de información topológica, asociada a los sectores de interacción proteína-proteína en pVHL permite el aumento significativo en la medida de desempeño del modelo de clasificación, medido a través de la eficiencia global (*accuracy*) del modelo.²
- El uso de información topológica, proveniente de los sectores fuertemente conectados mediante interacciones electrostáticas en la proteína, permite inducir una subdivisión de ésta para aplicar modelos de clasificación a dichos sectores y obtener medidas de desempeño aceptables en base a la relevancia del problema.
- La generación de sub divisiones del set de datos mediante técnicas de clustering permite generar modelos de clasificación más eficientes y precisos en comparación con la utilización de una nube de datos compleja.
- Existen sub divisiones que favorecen la generación de modelos de clasificación con altas medidas de desempeño que se relacionan con propiedades fisiológicas y fisicoquímicas en la proteína.

Es importante destacar que son varias las preguntas que se planean responder a lo largo del desarrollo del proyecto. En particular, el uso de información topológica para la

generación de sub set de datos y cuyos modelos poseen medidas de desempeño relevantes al problema desarrollado.

3. Objetivos

Diseñar, implementar, validar y testear modelos de clasificación para la evaluación de la relevancia clínica de mutaciones puntuales en proteína pVHL.

Del objetivo general, además de lo expuesto en las hipótesis, se desglosan los siguientes objetivos específicos.

1. Aplicar técnicas de minería de datos y estadísticas para la evaluación de set de datos, extracción de información y análisis de características con relevancia para el modelo.
2. Diseñar, implementar, validar y testear modelo de clasificación para relevancia clínica de mutaciones puntuales en proteína pVHL, por medio del uso de algoritmos de aprendizaje supervisado.
3. Desarrollar modelos de clasificación en sub divisiones inducidas por sectores de interacción proteína-proteína en proteína pVHL.
4. Evaluar distribuciones de medidas de desempeño en set de datos aleatorios generados a partir de la información topológica de la proteína para la comparación de estas medidas con respecto a lo obtenido por la división de grupos en base a los sectores de interacción.
5. Crear método de sub división de la proteína en base a los sectores fuertemente conectados dada las interacciones electrostáticas entre los residuos y dichos grupos testearlos con los modelos de clasificación generados.
6. Implementar técnicas de clustering para inducir divisiones en el set de datos que sirvan como entradas para la implementación de modelos de clasificación.
7. Aplicar y evaluar meta-clasificadores como técnica de aumento de medida de desempeño para los modelos de clasificación generados.

4. Metodología y Herramientas

En el presente capítulo, se expone un resumen de las metodologías desarrolladas para poder cumplir cada uno de los objetivos propuestos, además de dar respuesta a las hipótesis planteadas.

La metodología, en conjunto con las herramientas desarrolladas, se exponen en base al objetivo que cumple, las cuales se exponen a continuación.

4.1. Preparación del Set de Datos

El set de datos a trabajar consta de atributos discretos y continuos, para los valores discretos, se generó un tratamiento de los datos, de tal manera que considera la distribución de los ejemplos en el set de datos. lo cual se expresa a continuación.

Sea X una variable discreta compuesta por $\{x_0, x_1, \dots, x_n\}$ con n elementos finitos, se tiene la siguiente transformación de la variable discreta a continua mediante:

$\forall x_i \in X$ se tiene que $x_i^T = \sum$ de todas las incidencias de x_i divididos por el total de ejemplos en el set de datos.

Con ese concepto en mente, se propone la preparación del set de datos, tal como se expone en la Figura 4.1.

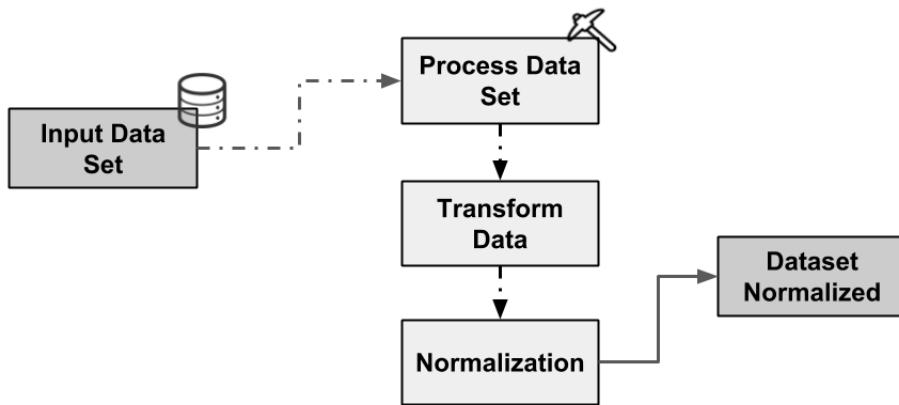


Figura 4.1: Esquema resumen metodología de preparación del set de datos.

Tal como se expone en la Figura 4.1, al set de datos inicial, se le aplica un procesamiento de los datos, donde se eliminan aquellos ejemplos que poseen valores nulos y perdidos, para luego trabajar con la transformación de atributos discretos a continuos, tal como se explicó previamente, finalmente se aplica la normalización al set de datos para generar las entradas a las siguientes etapas del desarrollo del proyecto.

4.2. Análisis estadísticos del set de datos

El desarrollo del análisis estadístico de los set de datos implica generar un conjunto de procesos que contemplan lo expuesto en la Figura 4.2.

En la Figura 4.2, se expone que a partir del set de datos, se aplica un conjunto de scripts asociados a los módulos de estadísticas, los cuales permiten generar: histogramas por atributos con distribución continua, box plot para el set de datos, generar la matriz de correlación a partir de los datos normalizados, visualizar estadísticos resúmenes y gráficos de torta para los atributos con distribución discreta, finalmente, trabaja con la información de los atributos para generar matrices de scatter plot, entre los atributos.

Todo lo anterior, permite evaluar las características de los set de datos a trabajar, generar visualizaciones a partir de la información que estos posean. Se destaca que el módulo desarrollado, es un módulo generalizado, es decir, soporta cualquier set de datos y facilita la aplicación de estos procesamientos.

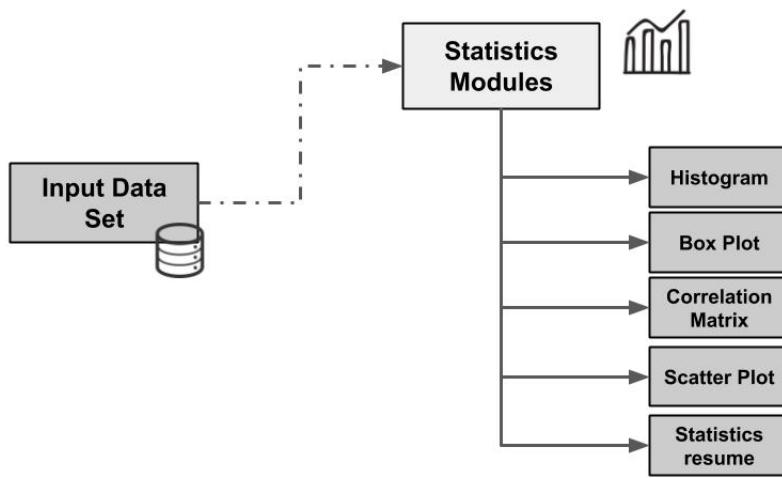


Figura 4.2: Esquema resumen metodología de desarrollo estadísticas para set de datos.

4.3. Entrenamiento de modelos de clasificación

El enfoque general de proyecto, radica en la generación de modelos de clasificación, los cuales, a partir de algoritmos de aprendizaje supervisado, permitan la clasificación de una nueva mutación, con el fin de poder evaluar si dicha instancia es clínicamente relevante o no, en base a la información asociada que ésta posee.

Se realiza un enfoque exploratorio para la generación de modelos, cada algoritmo se evalúa con sus diferentes parámetros, para cada evaluación, se registran diversas medidas de desempeño: *Accuracy*, *Precision*, *R-Call*, *F1 score*, *Neg log score*, *True Positive and Negative*, *False Positive and Negative*, donde cada una de éstas cumple un objetivo en particular y permiten seleccionar los mejores modelos para cada una de estas medidas de desempeño.

Una representación gráfica del proceso expuesto previamente, se expone en la Figura 4.3.

Es importante mencionar que para la validación del modelo y la evaluación de posible generalización, se utiliza validación cruzada. No obstante, en los casos en los que el set de datos posee un tamaño que no permite la sub división en diferentes grupos debido a la cantidad ínfima de elementos que poseen, se utiliza validación cruzada *Leave One Out*.

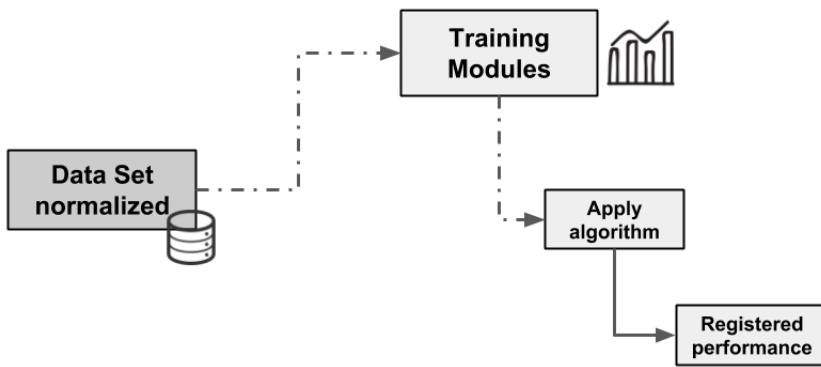


Figura 4.3: Esquema representativo metodología aplicación de entrenamiento de modelos para set de datos.

Una vez aplicado todo el proceso de entrenamiento de modelos en la fase exploratoria, se hace la selección de las mejores medidas de desempeño, con el fin de poder seleccionar el algoritmo y sus parámetros.

Los módulos desarrollados, permiten la aplicación de esta exploración de algoritmos, a diversos set de datos, debido a la generalización de los scripts.

4.4. Adición de Información Topológica

En la sección 2, se expone que se desea corroborar que la adición de información topológica, es benéfica a la hora de generar un aumento en las medidas de desempeño, con el fin de poder evaluar dicha característica, se presenta la metodología generada para la evaluación de adición de esta información, lo cual se expone en la Figura 4.4.

En la Figura 4.4, se expone que a partir del set de datos inicial, se hace una división de éste en base a las distribuciones de los valores del atributo *Superficie Sector* o el sector de interacción proteína-proteína, generando un sub conjunto de set de datos. Este conjunto de sub set de datos, es procesado para aplicarles los módulos de exploración de algoritmos de aprendizaje supervisado, lo cual permite generar los modelos de clasificación para cada sub set.

Al finalizar cada proceso, se selecciona el algoritmo y los parámetros que éste posee, con los cuales se obtuvo los mayores valores de medidas de desempeño.

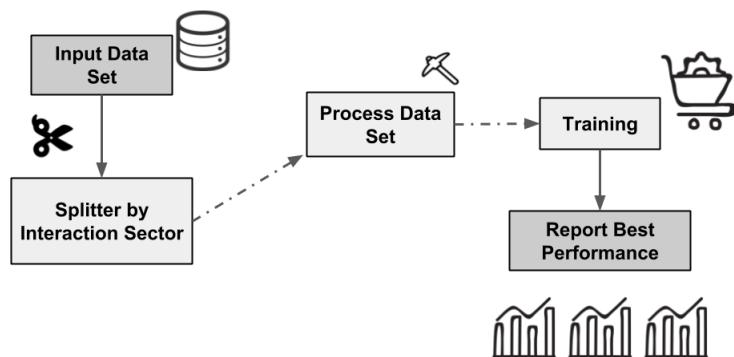


Figura 4.4: Esquema representativo metodología de evaluación de adición de información topológica.

4.5. Evaluación de Modelos

La fase exploratoria de entrenamiento de modelos, implica aplicar un conjunto de algoritmos de aprendizaje supervisado y un set de parámetros, los cuales dependen de cada algoritmo. Para evaluar sobreajuste y determinar la generalización del modelo. Se implementó validación cruzada, la cual implica una división del set de datos en set de entrenamiento y de testeo.

El proceso de validación cruzada fue implementado de diversas maneras, considerando la cantidad de elementos existentes en el set de datos, es decir, para aquellos casos en los que la cantidad de ejemplos fuera baja (inferior a 50) se aplica técnica *Leave one Out* y en el resto de los casos, una validación cruzada con una división de $k=10$. La primera hace referencia a que si el set de datos contiene n elementos, el entrenamiento contempla $n - 1$ ejemplos y se prueba con el restante. En la segunda se divide el set de datos en 10 grupos, se entrena con 9 y se valida con 1. En ambos casos, las medidas de desempeño se obtienen de la media de cada iteración.

Un punto importante a destacar, es que cada aplicación de algoritmo y sus parámetros, se ejecutó 100 veces, con el fin de poder obtener una distribución de las medidas de desempeño para cada evaluación, las métricas de evaluaciones finales, contemplan el promedio de la distribución de cada ejecución.

4.6. Selección de Modelos

La selección de modelos se basa en los algoritmos y sus parámetros que presentan mejores medidas de desempeño, es importante mencionar, que debido a que se generó una fase exploratoria, se contendría una distribución de métricas para cada proceso. Estos

valores asociados a una distribución de medidas de desempeño, se utilizó para la selección de los modelos.

Se tomó cada distribución y se obtuvieron aquellos valores cuyas medidas de desempeño eran mayores a 3 desviaciones estándar de la media, esto permitió obtener aquellos modelos con mejores valores, lo cual implica una selección de un conjunto de modelos que representan valores atípicos positivos dentro de la distribución.

Finalmente, estos modelos fueron sometidos a un proceso donde se obtuvieron: matriz de confusión, curvas de validación y de aprendizaje, gráficas de *precision v/s recall* y un resumen estadístico de sus medidas de desempeño.

Cada modelo seleccionado para cada división fue utilizado para predecir el resto de los grupos, con el fin de determinar si el modelo era específico para cada división, es decir, se espera que un modelo con buenos valores de medidas de desempeño para la división seleccionada, resultaba ineficiente con otras divisiones de sectores de interacción proteína-proteína.

4.7. Validación de las divisiones generadas

A la hora de evaluar si los valores de medidas de desempeños obtenidos eran significativos, se decidió hacer un estudio de la distribución de estas medidas por medio de la aplicación de un muestreo iterativo aleatorio, el cual contemplaba la constante división del set de datos en cantidades desde 2 hasta 13, generando el muestreo de 100 iteraciones por cada división, además de la posterior aplicación de los algoritmos de aprendizaje supervisado para el entrenamiento de modelos de clasificación.

Lo descrito anteriormente se expone de manera gráfica en la Figura 4.5.

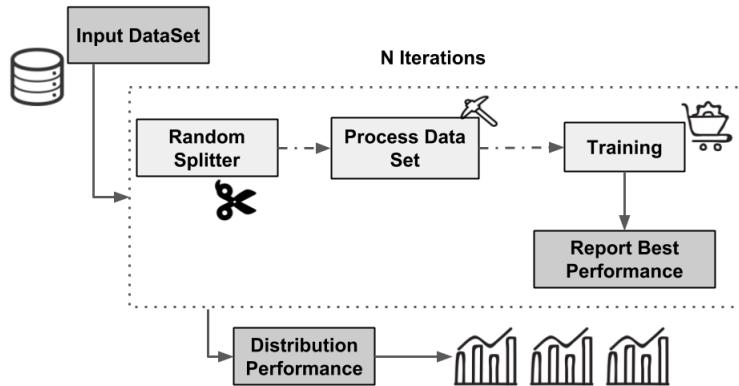


Figura 4.5: Esquema representativo metodología de validación de divisiones generadas

Tal como se observa en la Figura 4.5, el proceso es muy similar a los expuestos

previamente, la diferencia radica, en que con todas las medidas de desempeño obtenidas para cada iteración con cada sub división de elementos, se hacen histogramas con el fin de poder evaluar la distribución de estas medidas y generar una variable visual con respecto a éstas.

4.8. Unión de divisiones generadas

Debido a la sub división que se generó, fueron aplicados diversos algoritmos de clasificación. Para los casos en los que el algoritmo y sus parámetros se mantenían en diferentes grupos, se contempló una unión de dichas divisiones. Cada nueva unión se sometió al proceso de exploración de modelos.

La distribución de medidas de desempeño se evaluó y se obtuvieron los mejores clasificadores según la evaluación de los valores atípicos en dicha distribución. Se destaca además, que la unión de grupos se mantiene sí y sólo sí, los resultados obtenidos son mejores que en los grupos por separado.

4.9. Implementación de Meta-Learning

El Meta-learning, implica el uso de diferentes algoritmos y parámetros para generar un modelo de clasificación que se compone de diferentes algoritmos y parámetros. El objetivo general de la aplicación de esta técnica, radica en el hecho de aumentar los valores de las métricas de desempeño.

Se destaca que esta metodología fue aplicada a cada set de datos cuyas medidas de desempeño no tuvieron valores altos, el criterio de generación de acoplamiento de modelos fue como sigue.

En una primera instancia, se seleccionan los modelos cuyas tasas de verdaderos positivos son altas. continuando con los modelos cuyas tasas de verdaderos negativos también lo son. En una tercera instancia se contemplan aquellos modelos que presentaban una alta accuracy y finalmente aquellos que poseían una alta precisión.

Es importante mencionar, que la aplicación de meta-learning, es sólo hasta que las medidas de desempeño son lo suficientemente altas como para ser aceptables. Es decir, cuando se alcanzan valores superiores al 90 % de eficiencia, verdaderos positivos, o las medidas de desempeño que se estimen convenientes.

4.10. Aplicación de Clúster para la búsqueda de sub grupos en el set de datos

Otra forma de generar sub grupos dentro del set de datos, sin considerar las subdivisiones aleatorias generadas en el muestreo, es a través de la utilización de técnicas de clustering, las cuales permiten la generación de grupos, cuyos integrantes son similares entre sí y que son significativamente diferentes entre otros grupos.

El proceso generado para crear las subdivisiones de grupos en base a la información existente en el set de datos, es expuesto en la Figura 4.6.

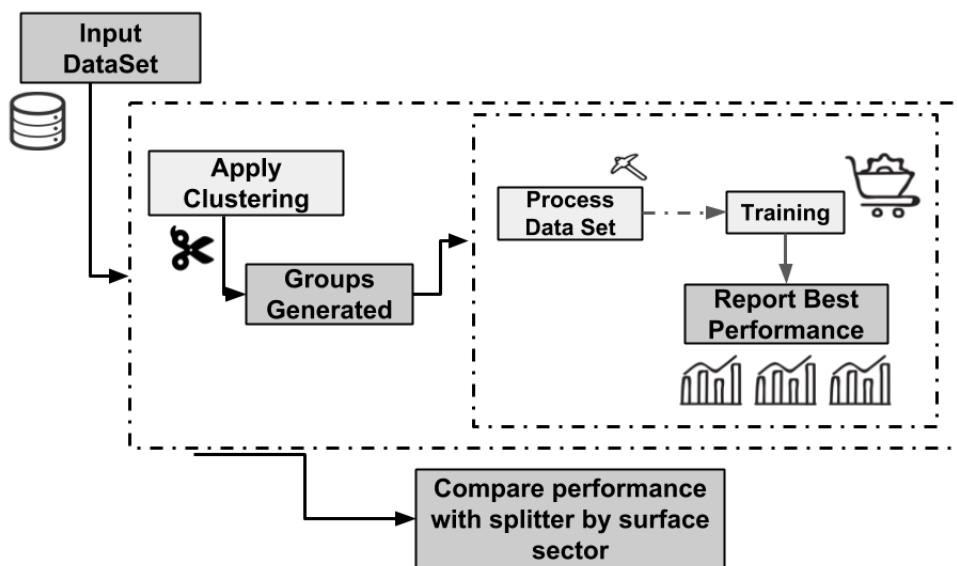


Figura 4.6: Esquema representativo metodología de aplicación de algoritmos de clustering

Tal como se expone en la Figura 4.6, a partir del set de datos inicial, se le aplican un conjunto de algoritmos de clustering, tales como: k-Means, Affinity propagation, mean shift, aglomerativos, DBScan, etc. a partir de los cuales son validados, es decir, se aplican los siguientes criterios para evaluar si es un grupo representativo:

- La cantidad de elementos por grupo debe ser mayor al 10 % de la muestra total.
- La evaluación del grupo, según el coeficiente de siluetas, debe ser por sobre un valor de 0.6.
- La distribución de clases en el grupo debe ser en las proporciones {50 : 50, 60 : 40, o 70 : 30}

Si la subdivisión completa cumplía con dichas condiciones se aplicaba el entrenamiento de modelos de aprendizaje supervisado de manera exploratorio, como se explicó

previamente y se registraron las mejores medidas de desempeño obtenidas para los grupos correctamente generados.

4.11. División inducida por clustering recursivo

Una nueva forma de inducir divisiones en los set de datos, es mediante la aplicación de algoritmos de clustering que permiten indicar la cantidad de grupos a obtener, los cuales se evalúan, mediante criterios de dispersión y pruebas de hipótesis. Una descripción general del algoritmo es como se expone a continuación.

Se toma un set de datos y se aplica un conjunto de algoritmos de clustering, induciendo una división de 2 grupos, en un segundo paso, se aplica criterios de coeficientes de siluetas y el índice de calinski harabasz, seleccionando aquellos grupos que poseían un mayor valor en cada índice. Luego se comparan estadísticamente los grupos mediante una prueba de hipótesis, utilizando el estadístico mann whitney para pruebas no paramétricas de comparación de medianas, quedándose con aquellas divisiones que estadísticamente fuesen diferentes entre sí y cuyos integrantes son semejantes entre sí.

El algoritmo explicado, se aplica de manera recursiva por cada división generada que sea válida, finaliza cuando la división implicada, no cumple con los criterios expuestos el algoritmo.

La implementación de este algoritmo entregaba como resultado un grafo con el que se visualizaba el proceso de división aplicado de manera recursiva al set de datos implicado.

Es importante mencionar, que no se contempla la proporción de las clases en la división de los grupos. No obstante, las divisiones obtenidas, debían cumplir con el criterio de que la cantidad de integrantes en ésta, debía ser superior al 10 % de la muestra total.

Cada división generada fue sometida a la fase de entrenamiento exploratorio, así como también a la evaluación de los grupos, la validación estadística y el resumen de características, entre todos los procesos que fueron aplicados al set de datos de inicial y las divisiones inducidas por la adición de información topológica asociada a los sectores de interacción proteína-proteína.

4.12. Generación de Grafos

Un enfoque diferente, pero que también tiene relación con la adición de información topológica de la proteína, se basa en la búsqueda de sectores en ésta que estén fuertemente conectados mediante interacciones electrostáticas, esto a través del uso de las teorías de grafos y algoritmos existentes para dicha búsqueda.

El proceso general, es factible resumirlo como se expone en la Figura 4.7.

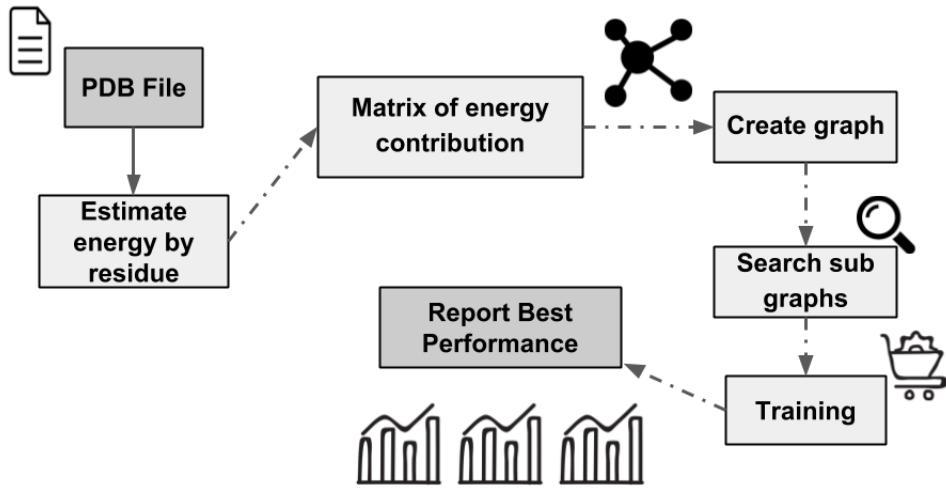


Figura 4.7: Esquema representativo metodología de búsqueda de sectores con alta interacción electrostática.

Todo el procesamiento de la información, es desarrollado tomando como entrada, la data existente en los archivo PDB.¹ A partir de éste se crea una matriz de interacciones electrostáticas entre residuos de la proteína, lo cual forma la matriz de adyacencia para la posterior generación del grafo. Una vez generado el grafo, se inicia la búsqueda de sub grafos, los cuales, se someten al proceso de exploración de clasificadores basados en algoritmos de aprendizaje supervisado.

Las energías de interacción que se calculan, en conjunto al procedimiento para calcularlas, son explicadas brevemente.

4.12.1. Cálculo de energías de Enlaces Covalente

El cálculo de energías covalentes se hace según lo expuesto en [1], para ello, es necesario contar con los valores de los ángulos *phi* y *psi*. De forma de facilitar el cálculo de estos enlaces, se utiliza el servicio web **WIWS**[2]. Ya con la información necesaria, se procede a utilizar los módulos generados para el procesamiento de los outputs de **WIWS** en conjunto con el cálculo de aporte de cada residuo calculado, por medio de scripts desarrollados en lenguaje de programación python.

¹<http://www.rcsb.org/>

4.12.2. Cálculo de energías de Enlaces de Hidrógeno

El cálculo de energías de enlaces de hidrógeno fue a través de la utilización del servicio web **WIWS**, el cual utiliza archivos protonados de la proteína de interés y evalúa los enlaces de hidrógenos más relevantes en ésta, mediante la utilización de un campo de fuerza y evaluando las probabilidades que presentan los átomos en la interacción.

Una vez se tenía la información, los archivos de salida de **WIWS** fueron procesados mediante lenguaje de programación Python, para poder hacer los cálculos efectivos.

4.12.3. Generación de Matriz de Adyacencia

Para generar la matriz de adyacencia, mediante lenguaje de programación Python, se procesan los valores previamente obtenidos de energías de enlace covalente y de enlaces de hidrógeno, se ordenan los residuos en una matriz y se completa cada columna según los elementos que correspondan.

4.12.4. Generación y Visualización del grafo

Para la generación del grafo se utilizó la librería *Networkx* de python y la visualización se generó mediante la utilización del plugin D3JS de JavaScript, lo cual permitía, a partir de la matriz de adyacencia, generar la visualización del grafo, simulando las interacciones de fuerzas electrostáticas, los movimientos de atracción y repulsión y a su vez la utilización de enfoques de campos de fuerza para dicha generación. El archivo *.js fue cargado a un navegador con el fin de generar la visualización dinámica y los efectos expuestos previamente. Finalmente, los sub grafos se estimaron mediante el uso de algoritmos de comunidades, contemplando valores energéticos y de distancia.

4.13. Desarrollo de estructura de proyecto, Diseño de Módulos y Ejecuciones Remotas

Con el fin de poder generar una estructura de proyecto organizada, fácil de manipular y altamente escalable, se utilizó el enfoque del paradigma de Programación Orientada a Objetos (POO), lo cual aprovecha las ventajas de dicho paradigma y facilita un diseño simple para el problema en cuestión.

La estructura del proyecto se basó en la organización dispuesta para python, donde se contemplan los siguientes componentes.

- **bin:** Directorio donde se disponen todos los archivos que presentan el carácter de ejecutables.

- **build**: Directorio donde se disponen todas las librerías que implica la instalación de los módulos generados.
- **docs**: Directorio donde se disponen todos los documentos asociados a la documentación del proyecto, los modelos, diagramas, etc.
- **resource**: Directorio donde se disponen todos aquellos scripts que son utilizados por los módulos de python mediante llamadas a sistemas, pero que no están escritos en el lenguaje python.
- **project**: Directorio donde se almacenan todos los módulos implementados en python.
- **README**: Archivo con la información general del proyecto y la descripción del mismo.
- **setup.py**: Archivo que permite la instalación de los módulos para ser accesibles desde cualquier entorno python de manera local.

Con el fin de exponer un poco el diseño generado para el desarrollo del proyecto, se expone en la Figura 4.8.

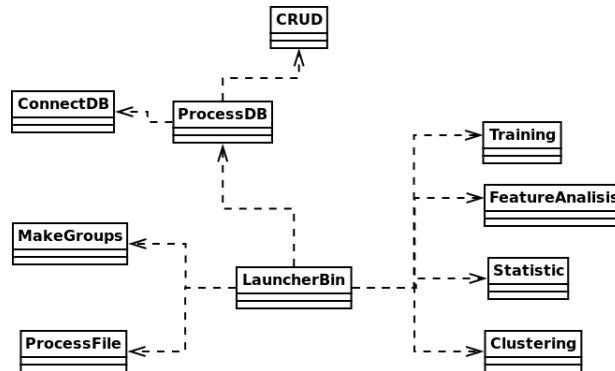


Figura 4.8: Representación de Módulos e Interacciones según POO.

Tal como es posible observar en la Figura 4.8, existe un módulo general, *LauncherBin*, el cual permite las interacciones entre los diferentes módulos. Se destaca además, que existen elementos con las funcionalidades asociadas al procesamiento de los archivos de entrada, la manipulación de la información, el almacenamiento de datos de manera persistente, etc. En cambio, existen otros, que tienen estrecha relación con la generación de información de salida, análisis estadísticos, evaluación de características, utilización de clustering y entrenamiento de modelos.

En general, cada módulo, tiene su responsabilidad y tiene estrecha relación con los otros existentes. Sin embargo, cada módulo en sí, es independiente de otro, es decir, puede tener funcionalidades independientes. por ejemplo, si se desea solamente hacer tratamientos estadísticos para un set de datos, es factible la utilización del módulo de *Statisctic* y así generar los resultados que se desean.

Es importante mencionar, que al ser un trabajo que implica un gran consumo de recursos computacionales, se desarrolló una metodología y sistemas de ejecuciones, que permitían la utilización de múltiples servidores remotos, los cuales eran accedidos de manera remota vía protocolo ssh, estas ejecuciones eran monitoreadas diariamente mediante un script *daemon*, el cual permitía, en una primera instancia, notificar si alguno de las rutinas dejaba de funcionar, y además, recopilar todas las instancias de ejecución y los resultados obtenidos.

Lo anterior fue aplicado en cada uno de los muestreos que se desarrollaron, ya sea con el objetivo de evaluar las distribuciones, o a su vez, corroborar la importancia de la división por la información topológica que implicaba el atributo *Superficie Sector*.

4.14. Herramientas Utilizadas

Las herramientas utilizadas pueden ser clasificadas según el lenguaje de programación utilizado, tal como se expone a continuación.

- Lenguaje de Programación **Python**
 - Scikit-Learn.
 - Pandas.
 - Numpy.
 - Seaborn.
 - Matplotlib.
 - BioPython.
 - Networkx.
- Lenguaje de Programación **R**
 - Ggplot2.
- Lenguaje de Programación **JavaScript**

- D3JS.

Adicional a las librerías utilizadas por cada lenguaje de programación, fueron consumidos los servicios web **WIWS** para las estimaciones de enlaces de hidrógenos y los cálculos de los ángulos ϕ y ψ .

También se expone que fueron utilizadas otras herramientas con respecto al desarrollo de notificaciones, sistemas de monitoreo y procesos de colas, las cuales son:

- SLURM como sistema gestor de colas.
- Screen como entornos de ejecuciones en múltiples sesiones.
- SMTP como gestor de correos.

5. Resultados Parciales

Durante el presente capítulo, se exponen los resultados parciales obtenidos a la hora de desarrollar cada una de las etapas expuestas en la metodología. Es importante mencionar además, que los resultados expuestos serán divididos en base al set de datos que lo conlleva. No obstante, se hará una breve explicación del set de datos, los atributos y sus características, partiendo inicialmente con el set de datos completo, luego aplicando la división por sector de interacción proteína-proteína, luego las divisiones aleatorias, las divisiones asociadas a los clustering, tanto inducidos o divisiones puras, así como también la data generada por la búsqueda de comunidades mediante sectores de interacción fuertemente conectados a través de energías electrostáticas o distancias de átomos.

5.1. Set de datos sin divisiones

El set de datos sin divisiones es aquel que contempla el 100 % de los datos, al cual, se desarrollaron el procesamiento de los datos, el análisis estadístico, el entrenamiento de modelos y la selección de estos, en base a la información de la metodología expuesta.

La data de interés contemplaba un total de 256 ejemplos y 15 atributos adicionales a la clasificación que estos poseen, la cual corresponde a **Clínicamente Relevante** con un **52.7 %** y **No Clínicamente Relevante 47.3 %**. Dentro de los atributos se encontraban valores discretos y continuos, los cuales fueron procesados como se explicó en la sección 4.1. Los atributos contemplaban información filogenética y estructural, asociada tanto a la mutación como a la data original.

5.1.1. Análisis Estadístico

El análisis estadístico se realizó contemplando box plots, matrices de correlación, histogramas, etc, además de resumen de estadísticos para cada set de datos, gráficos de

barras, etc. Los cuales se exponen a continuación¹.

Al visualizar algunas de las distribuciones de atributos discretos se observa, por ejemplo en la Figura 5.1 los diversos valores observados en el atributo SStructural, donde se expone que el 33 % de los ejemplos presenta valor **E**, 30 % **L** y 20 % **A**.

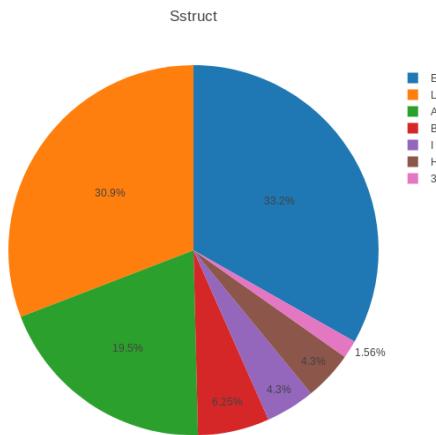


Figura 5.1: Distribución del Atributo SStructural

Se hicieron histogramas para evaluar posibles distribuciones en los atributos con valores continuos, a modo de ejemplo, se expone el histograma para el atributo **yDDG**, el cual se observa en la Figura 5.2. La distribución expone un comportamiento con tendencia a normal. No obstante, los valores se centran entre -0.5 y 0.5, esto es debido a que la data se normalizó previo a desarrollar el histograma, con el fin de poder generar visualizaciones escalables de los datos.

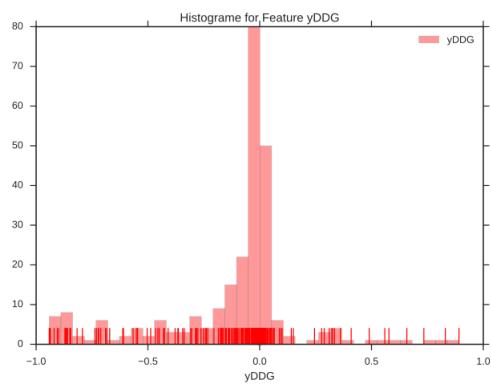


Figura 5.2: Histograma para el atributo yDDG.

¹Los restantes gráficos y resultados se encuentran disponibles en el material suplementario adicionado a este trabajo.

Por otro lado, se generaron box plot, para evaluar las dispersiones de los set de datos, dichas dispersiones, también fueron consideradas con la normalización de los datos, el cual se observa en la Figura 5.3.

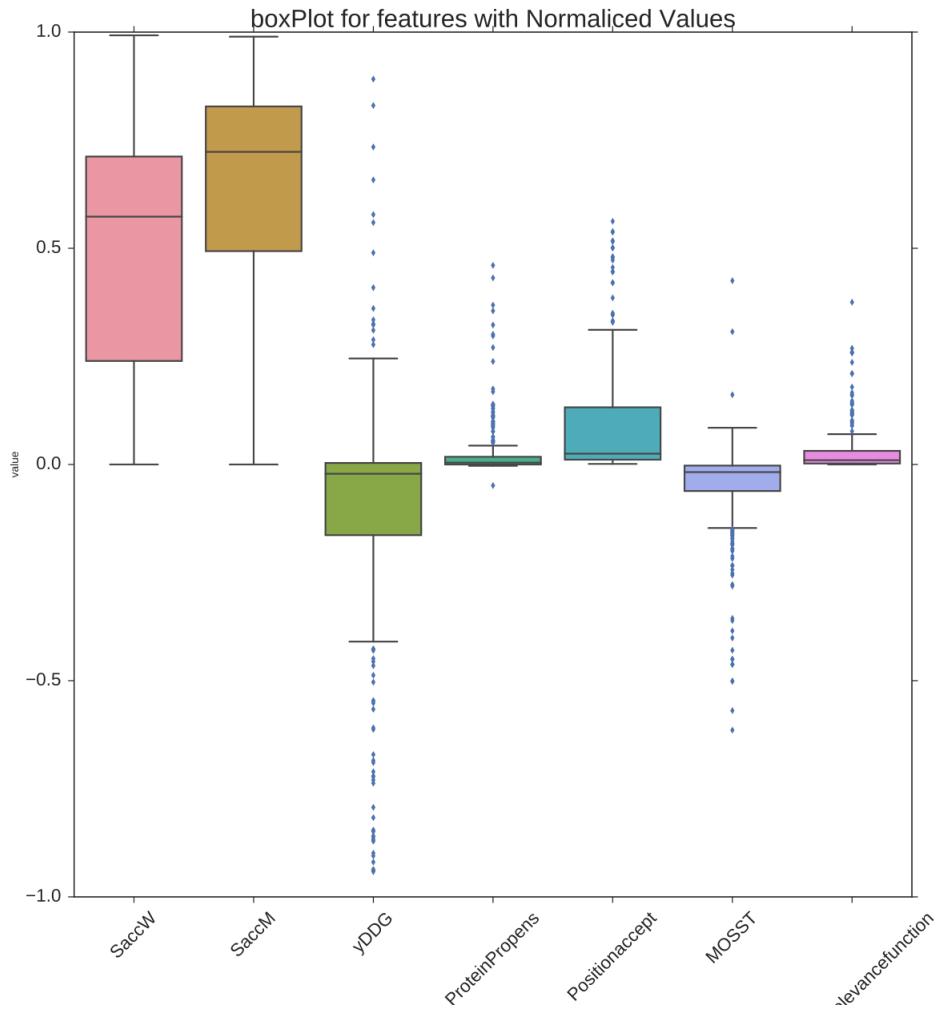


Figura 5.3: Box Plot para el set de datos.

Es importante mencionar, que el box plot, consideró sólo aquellos atributos que presentaban una distribución de datos continua. En la Figura 5.3 se aprecian gran cantidad de dispersiones en todos los atributos, además de un número significativo de puntos outliers en la muestra. Esto denota la gran dispersión que presentan los datos que se trabajaron.

Se evaluó la correlación existente entre los atributos del set de datos y los valores de ejemplos que estos poseían, con el fin de entender relaciones entre las características y determinar si era factible generar reducción de dimensionalidad evaluando coeficientes de correlación. La matriz de respuesta es la que se observa en la Figura 5.4.

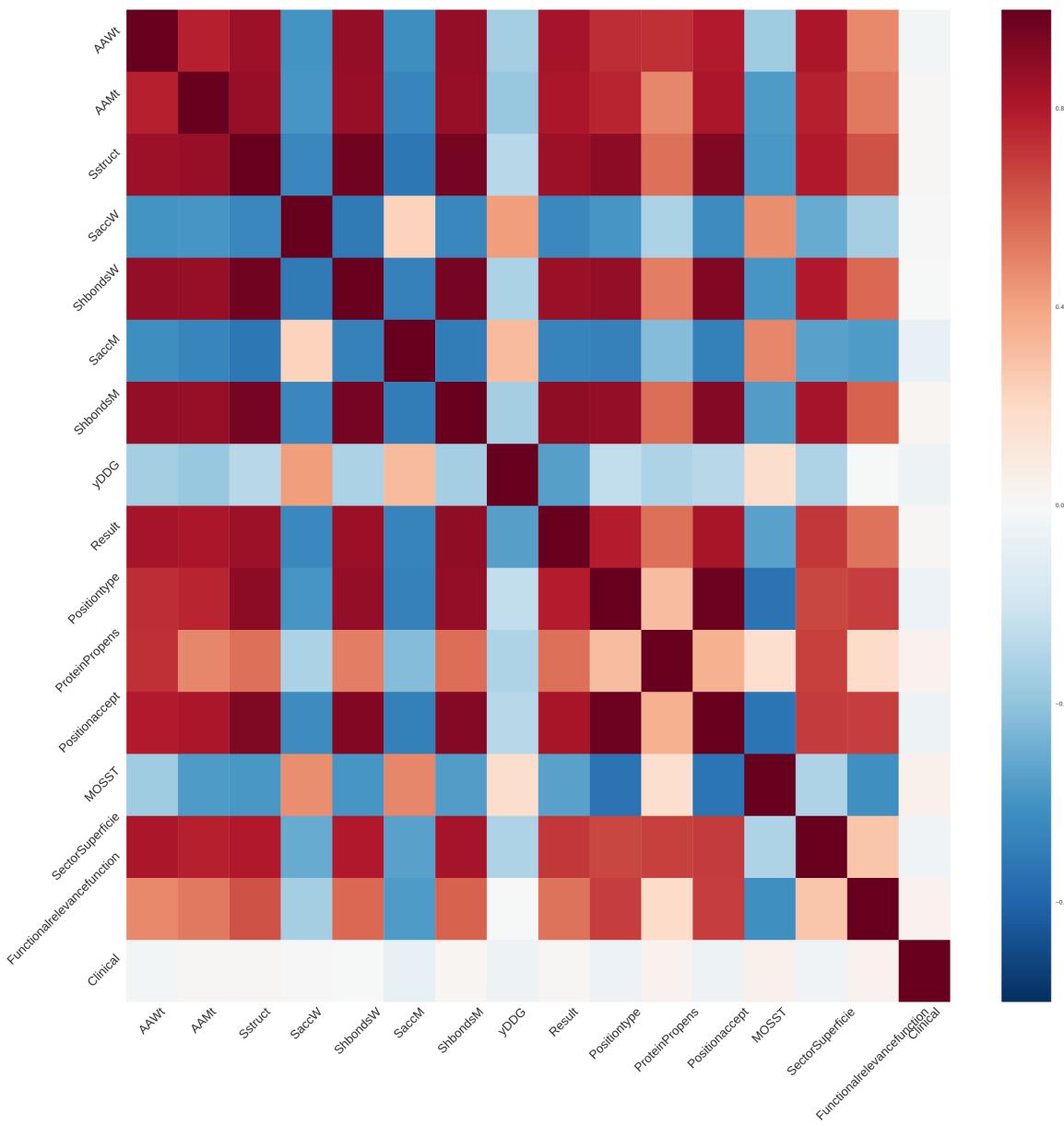


Figura 5.4: Matriz de Correlación para el set de datos.

Principalmente se observa en la Figura 5.4, que la clasificación de los ejemplos, no presentan ninguna relación con atributos en particular, es decir, la relación clínica no es influenciada por ningún atributo. Por otro lado, existen atributos cuyos coeficientes de correlación es cercano a 1, tales como *SaccW* y *ShbondsW* los cuales se encuentran directamente correlacionados, mientras que *SacCM* y *Sstruct* presentan correlación inversa.

5.1.2. Entrenamiento de Clasificadores

El entrenamiento de clasificadores fue mediante una fase exploratoria, la cual consistió en la aplicación de diferentes algoritmos de clasificación, tales como: Naive Bayes, Random Forest, SVM, KNN, Árboles de Decisión, Redes Neuronales, etc. Además de una variación de sus parámetros, tal como fue explicado en la sección 4.3. En total fueron **1578** ejecuciones para esta etapa. Resultados los cuales se sometieron a un proceso de selección.

La selección se hizo en consideración de los valores de tasas de verdaderos positivos, verdaderos negativos, precisión y eficiencia global. Con los resultados de la fase exploratoria, se desarrolló una distribución de medidas de desempeño para cada métrica en consideración, los cuales fueron gráficamente expuestos en histogramas. A modo de ejemplo, se expone en la Figura 5.5 el histograma para las tasas de verdaderos positivos obtenidas en la fase exploratoria.

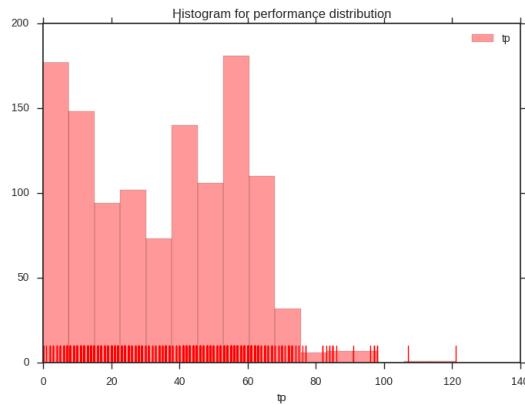


Figura 5.5: Histograma para distribución de Tasa de Verdaderos Positivos en Fase Exploratoria.

Se destaca que todo proceso de entrenamiento fue validado por medio de validación cruzada con un valor de $K = 10$, con el fin de evitar problemas de sobre ajustes y seleccionar modelos generalizados.

Para la selección de modelos, se evaluaron puntos en la distribución de medidas de desempeño, cuyos valores representaran puntos atípicos u outliers dentro de la muestra, considerándose como criterio de selección, modelos cuya medida de desempeño fuese mayor o igual a tres desviaciones estándar del promedio. Dado lo anterior, por cada métrica se seleccionaron un conjunto de modelos que cumplieron con el criterio implantado.

5.1.3. Selección de Modelos

Los modelos seleccionados y sus parámetros se exponen en las siguientes tablas 5.1, 5.2, resúmes por cada medida de desempeño.

Algoritmos seleccionados Según Tasa de Verdaderos Positivos		
Algoritmo	Descripción	Valor
MLP	MLPClassifier, activation: logistic, solver: sgd, learning_rate: invscaling, capas: 15-10-5	121
MLP	MLPClassifier, activation: tanh, solver: sgd, learning_rate: invscaling, capas: 15-5-15	107

Tabla 5.1: Modelos seleccionados según Tasa de verdaderos positivos.

Algoritmos seleccionados según valor de accuracy		
Algoritmo	Descripción	Accuracy
50	GradientBoostingClassifier	0.605874643875
poly	nuSVC	0.585435897436

entropy	RandomForest, n_estimators: 150	0.585851851852
MLP	MLPClassifier, activation: tanh, solver: lbfgs, learning_rate: adaptive, capas: 15-10-10	0.593435897436
MLP	MLPClassifier, activation: relu, solver: lbfgs, learning_rate: adaptive, capas: 5-5-10	0.594336182336
MLP	MLPClassifier, activation: relu, solver: lbfgs, learning_rate: adaptive, capas: 5-10-10	0.58558974359
MLP	MLPClassifier, activation: relu, solver: lbfgs, learning_rate: constant, capas: 10-10-10	0.585293447293

Tabla 5.2: Modelos seleccionados según Valor de Accuracy.

Al considerar la medida de Precisión, sólo el algoritmo Bernoulli, de Naive Bayes cumple con el criterio de selección, mientras que para la tasa de Verdaderos Positivos, Multi Layer Perceptron, del conjunto de algoritmos de Redes Neuronales, es el único que

cumple el criterio. No obstante, se adapta para diferentes tasas de aprendizaje y cantidades de capas.

Para todo modelo de clasificación seleccionado se aplicó un conjunto de procesos que permitieron obtener matrices de confusión, curvas de aprendizaje, áreas Roc, etc, los cuales, a modo de ilustración, se exponen a continuación.

La matriz de confusión para el modelo con algoritmo de clasificación GradientBoostingClassifier es como se expone en la Figura 5.6.

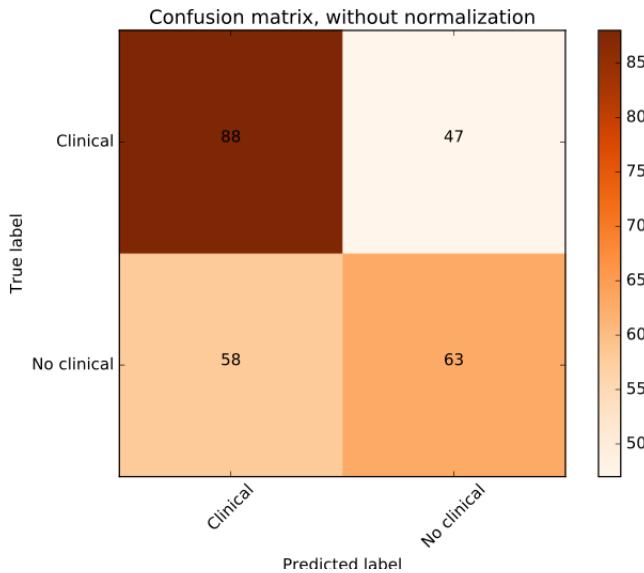


Figura 5.6: Matriz de Confusión para modelo de Clasificación con GradientBoostingClassifier.

En la Figura 5.6, se aprecia que la tasa de verdaderos positivos asociados a los elementos clasificados con relevancia clínica es de un **65 %**, mientras que la tasa de verdaderos negativos posee un valor de **52 %**, los cuales en particular, son valores que no indican una certeza significativa con respecto a un clasificador de la envergadura que se espera.

Por otro lado, se expone en la Figura 5.7 el proceso de evaluación de generalización del modelo mediante validación cruzada, generando la curva ROC del proceso, en donde en promedio, el área bajo la curva, es de un **0.58**, lo cual tienen relación con los valores de las tasas de verdaderos positivos y negativos, siendo un valor bajo, indicando un modelo azaroso a la hora de clasificar nuevas mutaciones.

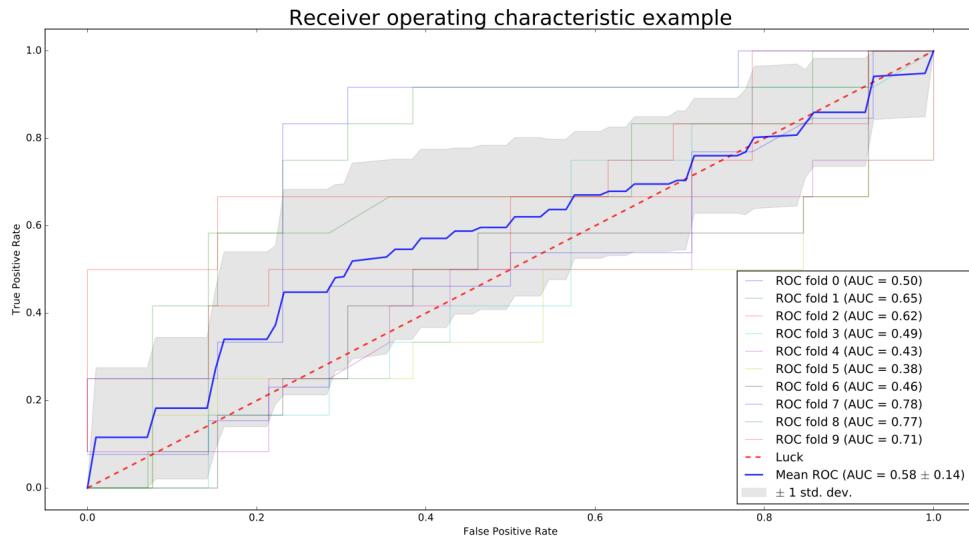


Figura 5.7: Curva ROC con evaluación mediante Validación Cruzada.

Otra de las gráficas de interés a la hora de validar los procesos de generalización de los modelos son las curvas de aprendizaje de éste y cómo se comportan los valores de entrenamiento con respecto al proceso de validación. La curva de aprendizaje para el modelo con GradientBoostingClassifier como algoritmo de clasificación es como se expone en la Figura 5.8.

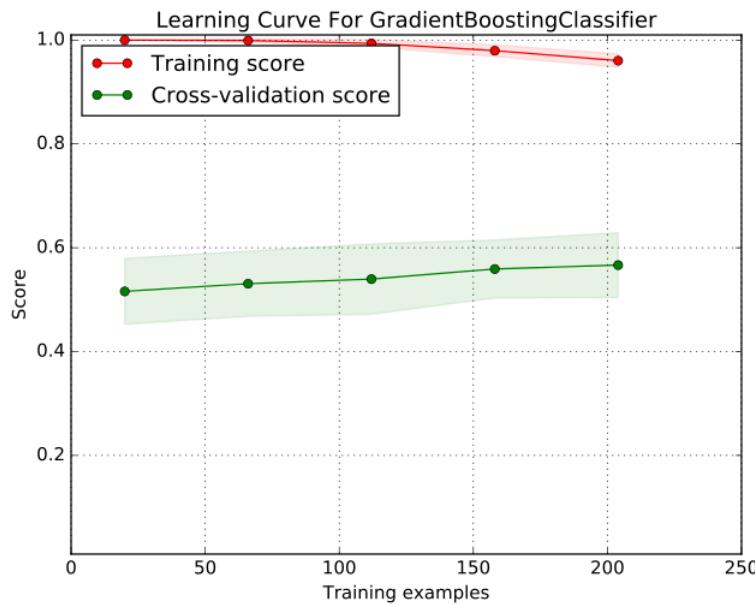


Figura 5.8: Curva de Aprendizaje para modelo de Clasificación con GradientBoostingClassifier.

Adicional a la curva de aprendizaje, todo proceso se evalúa mediante curvas de validación del algoritmo, en la cual se varía algún parámetro de distribución continua y determinar cómo afecta a las métricas obtenidas por el clasificador. En este caso de ejemplo, para el modelo de clasificación con algoritmo GradientBoostingClassifier, se expone su curva de evaluación con variación del parámetro de cantidad de iteraciones o repeticiones para el ensamble del modelo, lo cual se expone en la Figura 5.9.

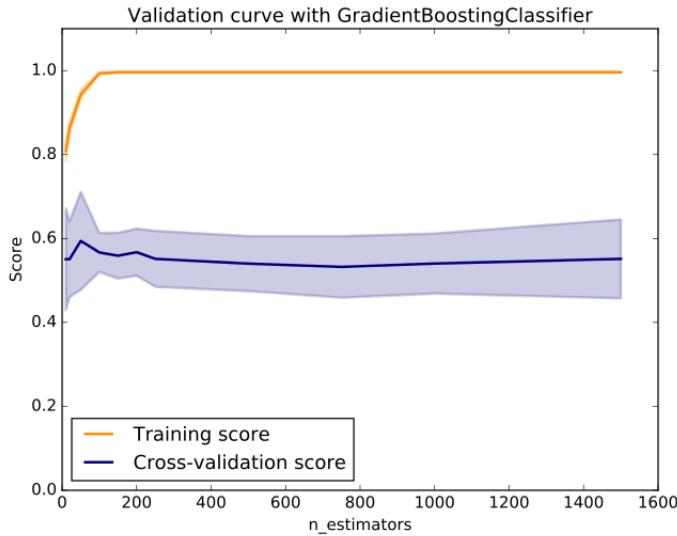


Figura 5.9: Curva de Aprendizaje para modelo de Clasificación con GradientBoostingClassifier.

Tal como se observa en la Figura 5.9, el proceso de validación y el de entrenamiento debiesen tener valores similares. No obstante, dicha característica no se aprecia en este modelo, además debido a las variaciones de los parámetros, estos debiesen variar en los resultados que exponen con el fin de denotar que el valor del parámetro resultado es representativo con respecto a las métricas que exponen sus variaciones.

Finalmente, otra de las gráficas que normalmente se realizan, es la curva de precisión v/s recall, para la evaluación del éxito de la relevancia del resultado dado a los que los elementos correctamente clasificados. Esta curva, permite evaluar una compensación entre el valor de la precisión y el recall. Si esta área bajo la curva es alta implica que una alta precisión se relaciona con una baja tasa de falsos positivos y un alto recall con una tasa de falsos negativos baja. En la Figura 5.10 se observa la curva para el modelo ejemplo expuesto.

El clasificador con algoritmo GradientBoostingClassifier, presenta un área bajo la curva de **0.58** para la curva de precisión v/s recall, lo cual denota prácticamente un modelo azaroso.

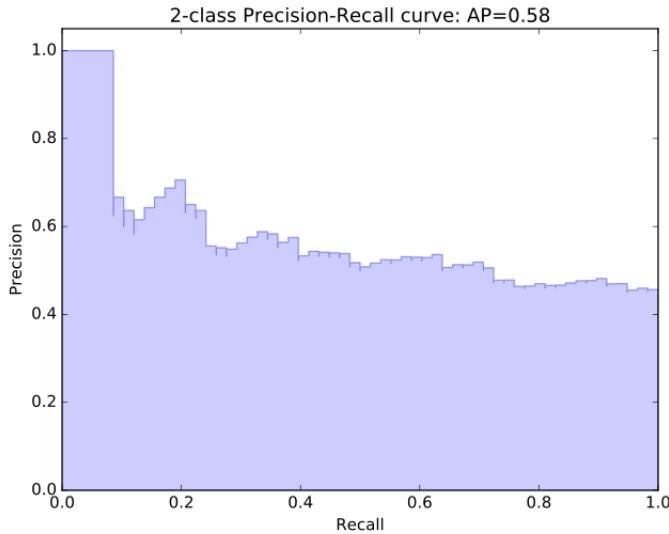


Figura 5.10: Curva de Precisión v/s Recall para modelo de Clasificación con GradientBoostingClassifier.

5.1.4. Análisis de Características

El análisis de características se desarrolló con el fin de poder evaluar cuales eran aquellas que entregan más importancia a la hora de evaluar el set de datos, es decir, corroborar aquellos atributos que entregan un mayor aporte a la varianza total de la muestra. Este enfoque, se hizo principalmente en base a técnicas de PCA y considerando elementos de relevancia asociado a las particiones que se generan en base al entrenamiento de modelos mediante algoritmo Random Forest.

Evaluación de Características con Random Forest

Tal como se expone en la Figura 5.11, existe una relevancia en las características que influyen en la división de elementos y árboles que permiten el entrenamiento mediante el algoritmo de Random Forest.

Atributos como *Result* y *Positionaccept*, son aquellos que entregan una mayor relevancia a la hora de entrenar. No obstante, el tener en consideración la evaluación de características en base a un algoritmo de entrenamiento, denota una forma de visualización en base a una deformación de espacio, la cual no puede realizarse en otros algoritmos debido a la forma en las que trabajan.

Por otro lado, la evaluación de características mediante PCA, implica cuántas características son las que aportan una mayor varianza. No obstante, no se identifican cuáles son dichas features.

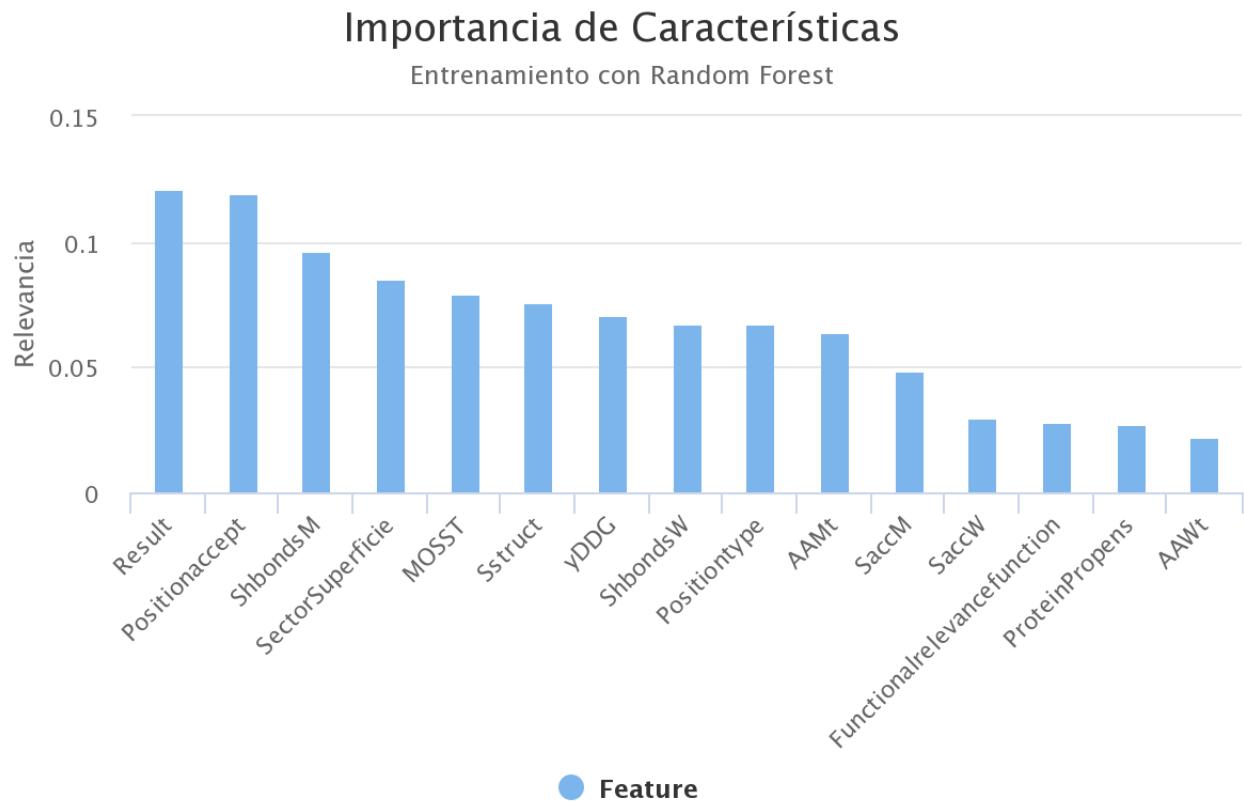


Figura 5.11: Importancia de características en entrenamiento Random Forest.

Un gráfico de aporte a varianza con respecto a número de componentes, es el que se expone en la Figura 5.12, en ella se destaca que 6 componentes principales entregan un 90 % de aporte a la varianza, mientras que 12 entregan cerca de un 98 %.

Uno de los grandes problemas que presenta la técnica de PCA es el famoso problema del *Scree Plot*, el cual consiste en la búsqueda de un punto de quiebre el cual permita hacer una selección de la cantidad de componentes a considerar versus la pérdida de información que se genere, esto mediante un análisis estadístico, matemático, diferencial, etc. Esto se ve reflejado en diversas áreas de investigación, en donde se necesita generar una reducción de la dimensionalidad, pero no se tiene un criterio matemático estadístico el cual permita entregar una cantidad definida, si no, que simplemente se basan en un número determinado por la expertís del investigador asociado al problema en sí.

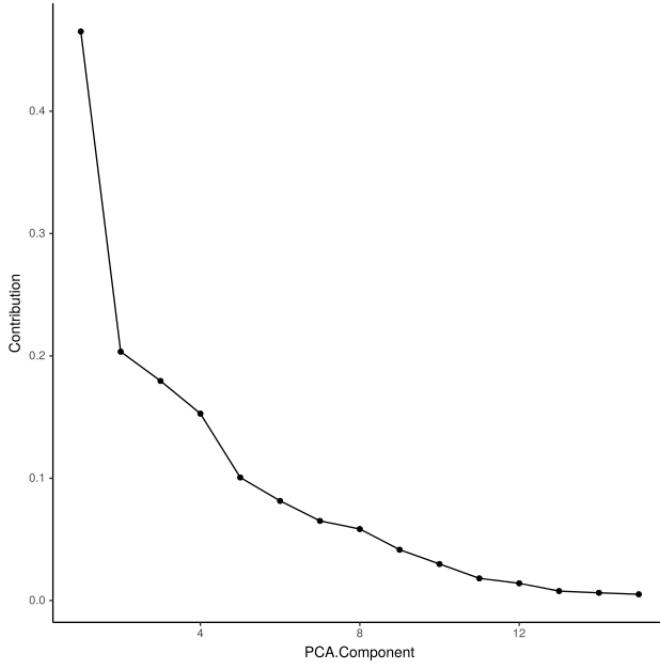


Figura 5.12: Aporte de varianza en base al número de componentes.

5.2. Divisiones inducidas por sectores de interacción Proteína-Proteína

La adición de información topológica, incluyó el conocimiento de los sectores de interacción que presenta VHL en interacciones proteína-proteína (PP). Se han reconocido 13 sectores de interacción, cuya distribución en el set de datos puede apreciarse en la Figura 5.13, en ella se observan los sectores correspondientes en conjunto con el porcentaje de ejemplos que abarcan en el set de datos.

Tal como se observa en la Figura 5.13, los sectores R, Z y O, son los que presentan una mayor proporción dentro de la muestra, mientras que H, U, N y C, no superan el 5 % del total de la muestra, razón por la cual, es posible que existan discusiones sobre la relevancia de los resultados en dichos grupos.

Una vez generada las divisiones, se generó el procesamiento estadístico de los datos para cada sector de interacción, el cual arrojó cambios significativos en cuanto a las dispersiones de los atributos con distribuciones continuas, lo cual denota una disminución de la desviación estándar, además de presentar una menor cantidad de outliers. Es posible observar este fenómeno en la totalidad de sectores estudiados. No obstante, a modo de ejemplo, esto se expone en la Figura 5.14.

Tal como fue expuesto en la sección 4.3, cada set de datos asociado a su sector de interacción fue sometido a los procesos de entrenamiento mediante fase exploratoria, la

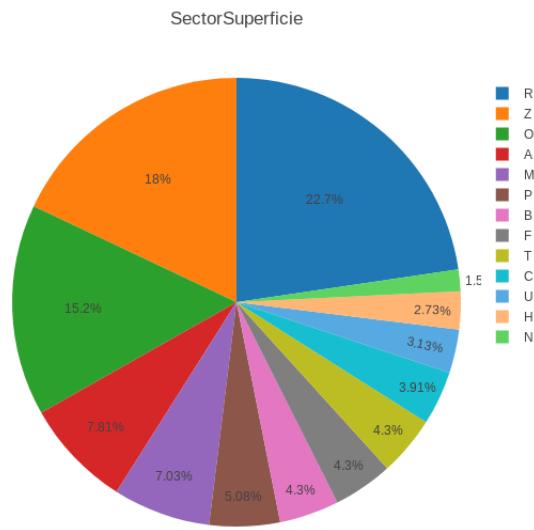


Figura 5.13: Distribución de ejemplos según los sectores de interacción PP en VHL.

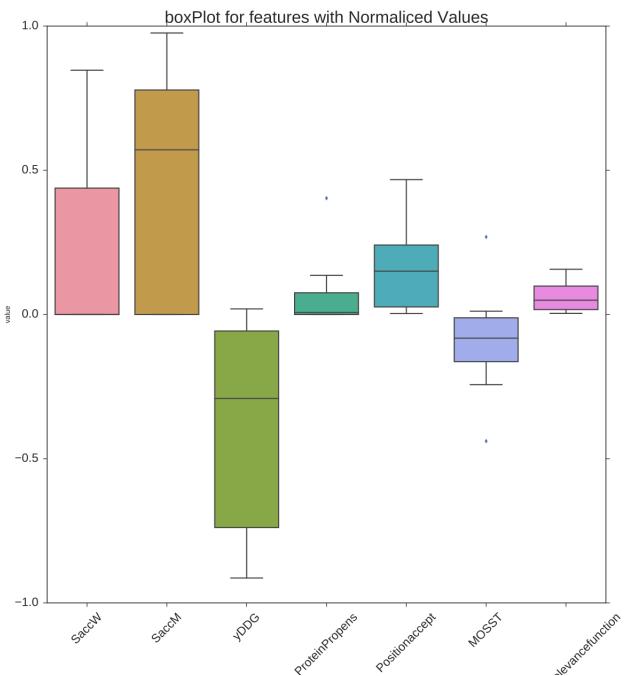


Figura 5.14: Box Plot para el sector de interacción A.

selección de modelos y la validación de estos mediante la generación de curvas de área ROC, validación, aprendizaje, precisión v/s recall y matrices de confusión.²

²Cada uno de estos resultados se encuentra disponible en el repositorio del proyecto, organizado según los sectores de interacción y los entrenamientos.

Es importante mencionar que la fase de validación del modelo fue mediante el uso de la técnica *Leave One Out*, lo cual permite entrenar y testear el modelo n veces, siendo n el número de ejemplos existentes en el set de datos.

Un resumen general de los modelos seleccionados por cada sector de interacción en base a las distintas distribuciones de medidas de desempeño, es como se expone en la Tabla 10.1³, en donde se denota la selección de modelos y los valores de métricas que estos poseen, además de las desviaciones estándar asociados a la distribución generada por el proceso iterativo de entrenamiento mediante Leave One Out.

5.3. Divisiones inducidas por clúster

Técnicas de clustering fueron aplicadas con el fin de poder inducir divisiones sólo con la información de los atributos y aplicar criterios de similitud con respecto a diferentes métricas de distancia. Se aplicaron diversos algoritmos de clustering y medidas de distancia, la selección de las mejores particiones implica el uso de los criterios expuestos en la sección 4.10.

Diversas particiones cumplieron los criterios infringidos, cada una de estas divisiones, se sometió al proceso de entrenamiento de modelos en fase exploratoria, mediante el uso de las metodologías expuestas previamente.

5.3.1. Generación de 2 particiones

Al inducir dos divisiones se obtiene un comportamiento de medidas de desempeño como se expone en la Figura 5.15.

En la Figura 5.15, se observa que los valores de recall son 1 y los de precisión cercanos a 0.8 en algunos casos, además de presentar un valor de accuracy en promedio cercano a 0.8 como lo es para el caso de la división inducida por distancias euclidianas mediante un linkage completo. Estas particiones se sometieron a los procesos de entrenamiento de modelos mediante validación cruzada con un valor de $k = 5$, posterior a ello se evaluaron los mejores modelos, obteniendo los resúmenes para cada modelo seleccionado según sus métricas de interés, en cada grupo.

5.3.2. Generación de 3 particiones

La generación de 3 particiones implicó una variación en las mejores medidas de desempeño obtenidas en los modelos de clasificación desarrollados, lo cual se aprecia en la

³Dicha tabla se encuentra en la sección de Anexos.

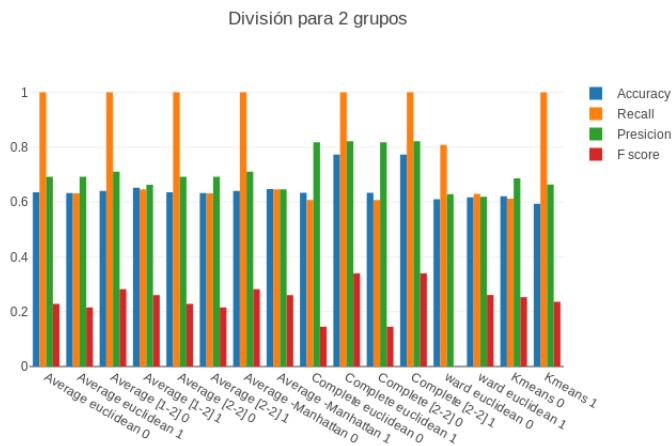


Figura 5.15: Medidas de Desempeño para dos divisiones.

Figura 5.16.

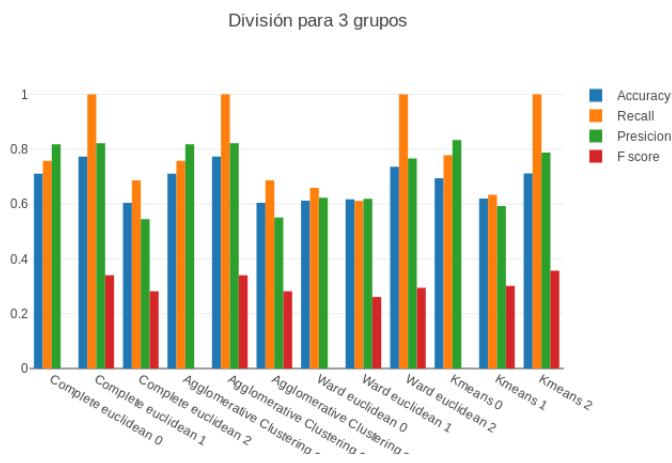


Figura 5.16: Medidas de Desempeño para tres divisiones.

Las divisiones inducidas por algoritmos aglomerativos, con un linkage completo y considerando distancia euclideana, fueron las que presentaron en su distribución mejores valores de medidas de desempeño, las particiones se sometieron a una fase de exploración y se seleccionaron los mejores modelos según metodología.

5.3.3. Generación de 4 particiones

La generación de 4 particiones implicó una variación en las mejores medidas de desempeño obtenidas en los modelos de clasificación desarrollados, lo cual se aprecia en la Figura 5.17.

Se obtuvieron las mejores divisiones inducidas por algoritmos aglomerativos, mientras

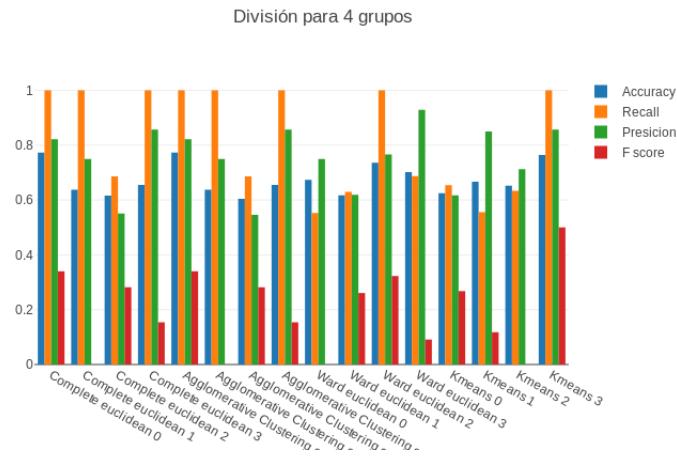


Figura 5.17: Medidas de Desempeño para cuatro divisiones.

que el linkage y la métrica de distancia se mantuvo con respecto a las anteriores, éstas, fueron las que presentaron en su distribución mejores valores de medidas de desempeño, las particiones se sometieron a una fase de exploración y se seleccionaron los mejores modelos según metodología.

5.3.4. Generación de 5 particiones

La generación de 5 particiones implicó una variación en las mejores medidas de desempeño obtenidas en los modelos de clasificación desarrollados, lo cual se aprecia en la Figura 5.18.

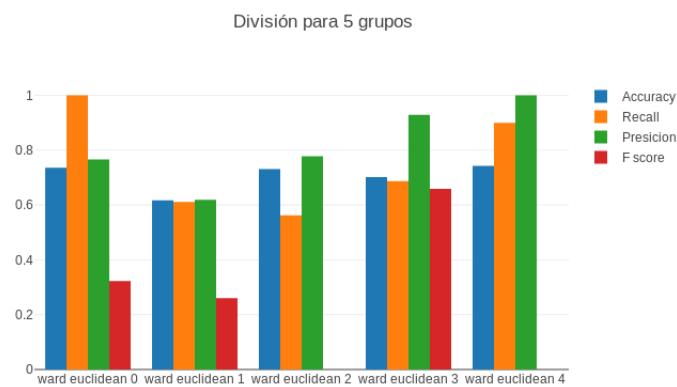


Figura 5.18: Medidas de Desempeño para cinco divisiones.

Las mejores divisiones se obtuvieron aplicando algoritmos ward y manteniendo las métricas de distancia con respecto a las anteriores, éstas, fueron las que presentaron en su

distribución mejores valores de medidas de desempeño, las particiones se sometieron a una fase de exploración y se seleccionaron los mejores modelos según metodología. Los valores de medidas de desempeño fueron menores con respecto a la data expuesta previamente. Sin embargo, se mantiene con respecto a los grupos, lo cual denota una persistencia de la medida en las distintas divisiones y no se ven favorecidas algunas con respecto a otras.

5.3.5. Selección de Particiones

La cantidad de particiones fue seleccionada en base a los resultados en promedio de los valores de medidas de desempeño que presentaba cada grupo en la partición, para ello se consideró la precisión, el recall y la accuracy del modelo. El resumen de este proceso, es decir, las mejores medidas de desempeño obtenidas para cada división, se exponen en la Tabla 5.3⁴.

Resumen Promedio medidas de desempeño según particiones				
Divisiones	Algoritmo	Accuracy	Precisión	Recall
2	Agglomerativo, Linkage Completo, Distancia euclideana	0.70	0.81	0.81
3	Agglomerativo, Linkage Completo, Distancia Euclideana.	0.69	0.81	0.72

⁴Un mejor resumen de dicha selección y valores medios además de desviaciones estándar 10.2.

4	Agglomerativo, Linkage Completo, Distancia Euclideana.	0.67	0.92	0.74
5	Agglomerativo, Linkage Completo, Distancia Euclideana.	0.70	0.75	0.83

Tabla 5.3: Resumen de medidas de desempeño promedios para las particiones generadas.

Tal como se expone en la Tabla 5.3, se observa que en promedio la accuracy es similar en casi todas las particiones. Sin embargo, la precisión y el recall, varían. En base a esto, y debido a que lo que se requiere es altos valores en las tasas de verdaderos positivos y negativos, se toma en consideración aquellas particiones que en promedio, presenten una alta precisión, razón por la cual se seleccionan 5 particiones.

Otro punto importante a destacar es que las medidas de desempeño en las particiones seleccionadas, presentaban menor desviación estándar intra grupos, en todas las métricas evaluadas, lo cual implica que las medidas estarán menos dispersas entre los grupos.

Se destaca además, que a cada una de las divisiones generadas se les aplicó la evaluación de los modelos y el procesamiento de las diferentes curvas de evaluación, adicional a ello se evaluó la independencia de las particiones mediante aplicación de modelos para predicción de valores diferentes a la partición que corresponde.

Al seleccionar 5 particiones, la cantidad de ejemplos por partición es como se expone en la Tabla 5.4.

Tal como se observa en la Tabla 5.4, la cantidad de ejemplos es similar en 4 particiones, abarcando una mayor proporción la primera partición con 120 ejemplos.

Dentro de los modelos obtenidos para dichas particiones se encuentran distribuidos principalmente entre KNN, Random Forest y Naive Bayes, cada uno con un conjunto de

Cantidad de ejemplos en Partición Generada		
#	Partición	Ejemplos
1.	Part. 0	38
2.	Part. 1	120
3.	Part. 2	41
4.	Part. 3	34
5.	Part. 4	23

Tabla 5.4: Cantidad de ejemplos en partición generada.

parámetros asociados a dichas métricas. Se destaca que la relevancia de dichos modelos apunta al hecho de que permiten generar explicaciones matemáticas o visuales para la clasificación que dichos entes generan. Es importante destacar, que también se obtuvo un conjunto de distribuciones de medidas de desempeño para los modelos, a partir de los cuales se seleccionaron aquellos con valores mayores a 3 desviaciones estándar por sobre el promedio. A modo de ejemplo, se expone el histograma que representa la distribución de resultados obtenidos para la Precision en el grupo 5 de las divisiones inducidas por el clustering utilizado, lo cual puede observarse en la Figura 5.19.

Tal como se expone en la Figura 5.19, una gran cantidad de elementos entregan valores de precisión de un 80 % aproximadamente. No obstante, existen algunos que permiten ostentar un 100 % en la etapa de entrenamiento, lo cual cumple en cierta parte, con las necesidades de los clasificadores a postular.

Un resumen completo de los modelos de clasificación, obtenidos por cada métrica, asociados a cada partición es como el que se expone en la Tabla 10.3 adjunta en la sección Anexos.

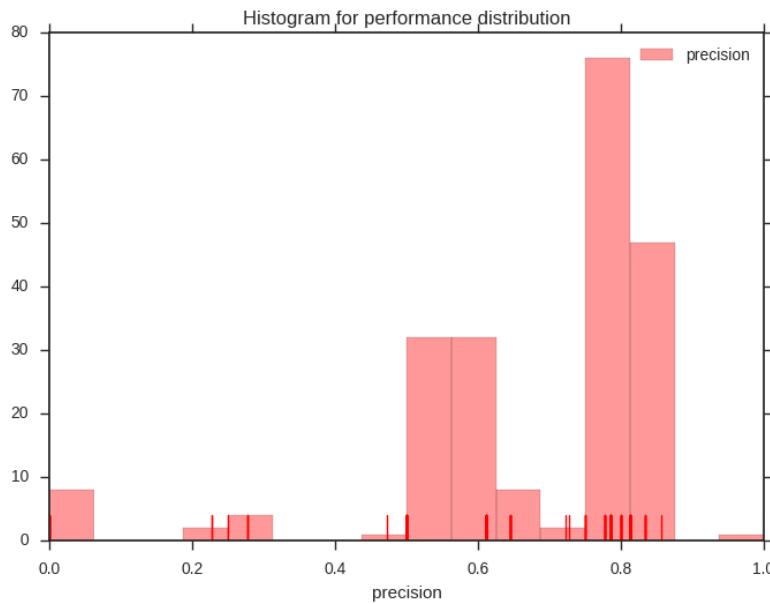


Figura 5.19: Histograma para Distribuciones de Precision en entrenamiento del grupo 5 en divisiones inducidas mediante clustering.

5.4. Divisiones inducidas por comunidades

La división inducida por comunidades, implica la utilización de teoría de grafos para la búsqueda de sectores fuertemente acoplados, estas conexiones vienen dados por las energías electrostáticas que presenta la proteína. Se generaron diversas matrices de adyacencia con respecto a diferentes métricas de interés, tales como: distancias con $C\beta$ como centroide, distancias con inducción de centroides, fuerzas electrostáticas mediante el cálculo de energías covalentes, puentes de hidrógeno y energías de repulsión o efectos de Van der Walls.

Cada una de estas matrices fue utilizada para implementar diversos grafos, dentro de los cuales, a modo de ejemplo se expone en la Figura 5.20, el grafo obtenido asociado a las matrices de adyacencia mediante $C\beta$ como centroide y aplicando pesos asociados a los valores energécticos que aportan cada residuo.

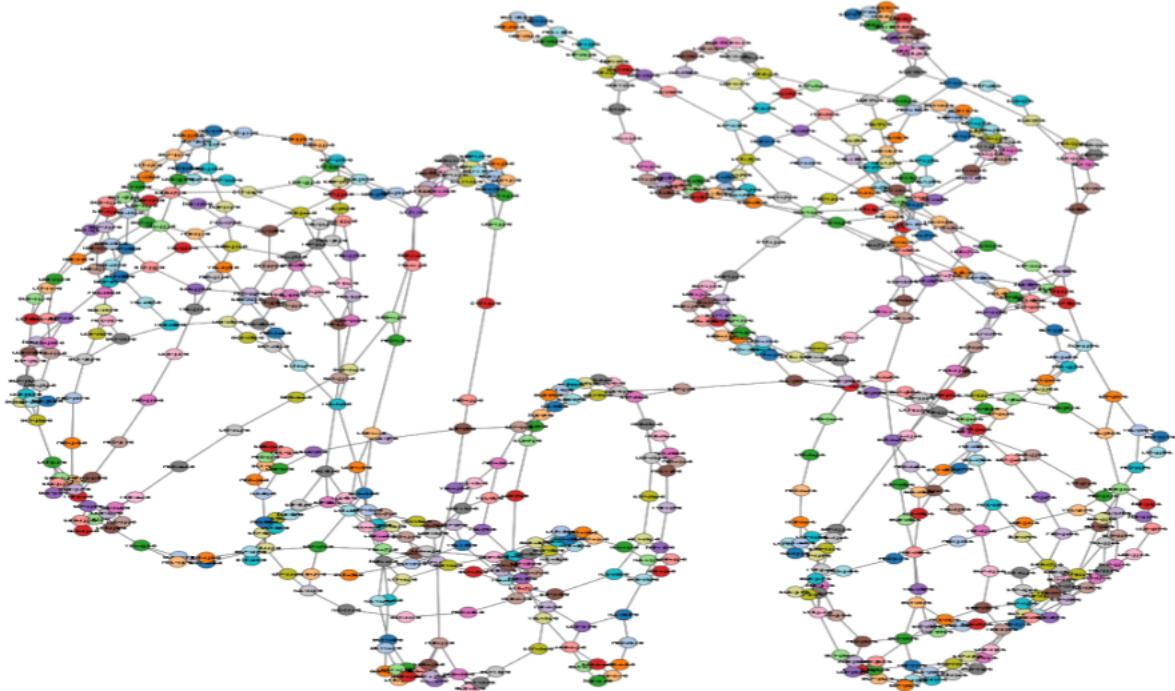


Figura 5.20: Grafo obtenido por matriz de adyacencia mediante energías de interacción.

A partir del grafo generado, se aplicaron diversos algoritmos de búsqueda de sectores fuertemente interconectados en la estructura, a partir de ello se aislaron dichos sitios obteniendo comunidades. Las comunidades se generaron con diferentes métricas de matrices de adyacencia y con la aplicación de diferentes técnicas de interés. Se obtuvieron alrededor de 20 comunidades en promedio. Prácticamente, sólo se reconoció como fuertemente conectados, aquellos sectores que presentaban enlaces covalentes.

Debido al gran número de comunidades encontradas, no fueron sometidas a las fases de exploración y evaluación de distribución de medidas de desempeño. En base a esto, se infiere que es necesario generar nuevos sistemas de escalamiento de los aportes energéticos o inducir nuevas formaciones de comunidades mediante evaluación de distancias, técnicas estadísticas, etc.

5.5. Divisiones inducidas por recursividad de clúster

Las divisiones inducidas por recursividad de clúster implicó el diseño y la implementación de un algoritmo de división recursiva con evaluación estadística y métricas de separación aplicadas a las particiones.

El proceso general se puede describir mediante lo que expone la Figura 5.21. En ella se observa que existen 7 nodos hojas, los cuales son las divisiones resultantes del proceso,

la cantidad de elementos para cada división es lo que expresa el valor del nodo, es decir, por ejemplo, existen particiones con 82, 33 y 62 elementos.

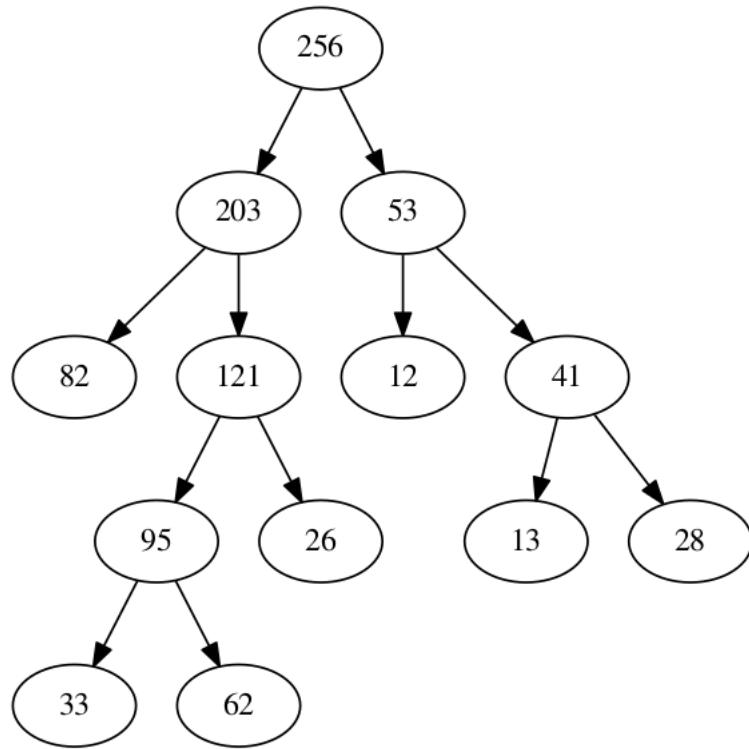


Figura 5.21: Esquema resumen del proceso de generación de particiones mediante recursividad de clúster.

Cada una de las particiones generadas, fue sometida a los procesos de exploración de modelos y posterior selección de estos en base a las métricas de interés. Además, para los mejores modelos, se obtuvieron los resúmenes expuestos en resultados anteriores y se evaluó que los modelos fuesen significativos, es decir, el o los modelos seleccionados para el grupo A, fueron utilizados para predecir los valores del resto de los grupos. Esto permitió evaluar la independencia de los grupos y que cada uno fuese tratado como un set de datos independiente.

Para cada partición obtenida, se obtuvo un conjunto de modelos asociados a sus respectivas distribuciones medidas de desempeño, los cuales representan puntos cuyos valores son por sobre tres desviaciones estándar del promedio. Las etapas de entrenamiento implicaron utilizar técnicas de validación asociadas a Leave One Out. No obstante, a la hora de entrenar con técnicas de validación cruzada, los resultados en términos de la eficiencia global no sufrían alteraciones.

Los principales modelos presentan diferentes algoritmos y parámetros. Sin embargo, los más recurrentes se asocian a KNN, SVM, Random Forest y MLP, presentando diferentes

valores de métricas de desempeño, los cuales varían entre 0.7 y 1 para el caso de la Precisión. Un resumen general de los modelos y sus valores, se expone en la Tabla 10.4 de la sección Anexos.

5.6. Comentarios finales sobre los resultados obtenidos

Se destaca que se obtuvieron diferentes modelos con distintas medidas de desempeño para cada situación. Si bien cada una de las particiones generadas entrega mejores resultados que los obtenidos al evaluar el modelo con el set de datos completo, son todos resultados parciales, cada uno los cuales deben ser validados con respecto a la metodología y la data obtenida. No obstante, estos resultados permiten vislumbrar de manera objetiva una metodología de desarrollo de modelos de clasificación para la relevancia clínica de mutaciones en VHL, el cual, pudiese ser expansible a otras proteínas de interés.

Si bien, las particiones indican que es necesario dividir el set de datos para obtener mejores resultados, es un gran desafío justificar cuáles particiones son las que vale la pena trabajar, además de determinar la mejor forma de identificar una nueva mutación y reconocer a qué partición reconoce, por lo que en la siguiente sección se abarcará cada una de estas temáticas y evaluarán las particiones, discutiendo los resultados obtenidos y corroborando las metodologías planteadas.

Se destaca además, que todos los resultados de los modelos, las bases de datos, las tablas de entrenamiento, análisis estadísticos y cualquier resultado relevante asociado al tema problema, se encuentran disponibles en el repositorio del proyecto, así como también los modelos, scripts, diseño de herramientas computacionales, esquemas de estructuras de servidor, y todo lo relevante en cuanto al proyecto en sí.

6. Discusiones

Se han generado diversos modelos, con respecto a los set de datos que se trabajaron, las particiones inducidas y el efecto de agregar información topológica. Cada uno de los puntos se exponen a continuación.

6.1. Modelos en Set de Datos Completo.

El set de datos completo, presentaba un total de 256 ejemplos, los cuales se encontraban proporcionalmente distribuidos en cuanto a sus valores de clases, lo que implica que no existe un desbalance de clases, además se descartaron problemas de maldiciones de dimensionalidad y redundancia de elementos, debido a que la cantidad de ejemplos era proporcional a la cantidad de atributos que éste poseía.

En cuanto al análisis estadístico, los atributos con distribución continua, presentaban un gran número de dispersiones y puntos outliers. No obstante, no fueron removidos debido a la poca cantidad de información existente en el set de datos, es decir, el número de ejemplos no era lo suficientemente significativo para efectuar pruebas de interés.

Al evaluar las matrices de correlación, existen atributos que se encuentran fuertemente correlacionados, tanto positiva como negativamente, esto implica que se presenta una dependencia entre ellos, la cual puede afectar en el entrenamiento de modelos y los valores de medidas de desempeño que se obtienen a partir de estos.

Las distribuciones de los atributos con carácter continuo, no presentaron rangos característicos de tendencia normal. Lo cual era esperable al visualizar los histogramas, además de las dispersiones existentes visualizadas en el box plot.

La fase exploratoria de entrenamiento de modelos de clasificación implicó la ejecución de un número significativo de algoritmos y una variación de parámetros de los cuales estos se componen. Sin embargo, tal como se pudo apreciar en la Figura 5.5. La gran mayoría de ejecuciones presentaban medidas de desempeño muy deficientes.

En base a dicho histograma, mediante técnicas de evaluaciones de distribuciones, se obtuvieron los mejores modelos, tal como se expuso previamente, acción que se llevó a cabo por cada medida de desempeño evaluada: Accuracy, Recall, Precision, Tasas de Verdaderos Positivos y Negativos.

Cada medida de desempeño presentaba modelos en particular, con diferentes parámetros, algunos tenían relación en cuanto al algoritmo, una intersección de dichos modelos según sus métricas, se expone en la Figura 6.1.

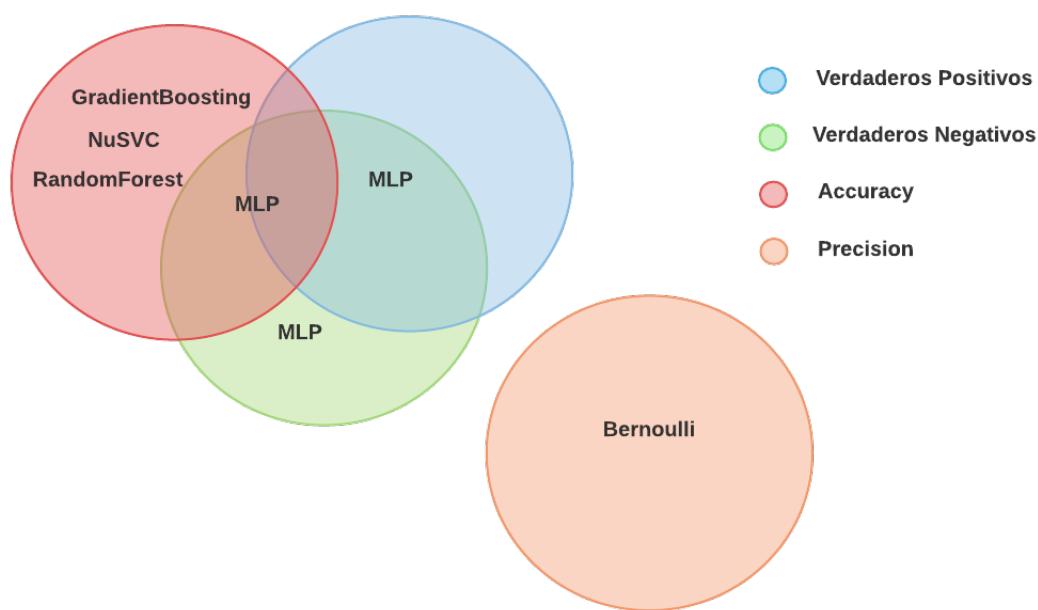


Figura 6.1: Diagrama de interacción entre modelos.

Tal como se aprecia en la Figura 6.1, Multi Layer Perceptron está en la interacción de diferentes medidas de desempeño, mientras que Bernoulli, GradientBoosting, NuSVC y Random Forest, aparecen sólo en alguna de las medidas de interés. Multi Layer Perceptron aparece como mejor algoritmo, dado a que se mantiene dentro de los mejores modelos para diferentes métricas de interés. No obstante, redes neuronales, al trabajar en forma de caja negra, no entrega una explicación plausible a la hora de evaluar la elección del clasificador, además, presenta problemas de sobre ajuste debido a que la cantidad de capas implica más y más complejidad en el problema, lo que al final de cuentas, a pesar de que se haya implementado una validación mediante *Cross Validation* puede que existan problemas de sobre ajuste, lo cual, a la hora de probar el modelo se vió reflejado dicho inconveniente. Además dicha evaluación se hace más notoria cuando se evalúa la generalización mediante *Leave One Out*.

Teniendo en consideración los puntos expuestos anteriormente, los modelos basados en

ensamble o utilización de transformación en base a kernel, implicarían mejores estrategias de modelos, además de permitir explicar de una manera visual, las clasificaciones que se generan. No obstante, Bernoulli aparece como un modelo con precisión de 0.65, lo cual implica que existe un 35 % de probabilidades de cometer un error, a pesar de ello, es el modelo con mejor precisión de los que se ejecutaron, por lo que, teniendo en cuenta esta consideración se propone en una primera instancia a Bernoulli como modelo de clasificación, continuando con métodos de ensamble como Random Forest y GradientSVC.

En base a esto, es posible comentar, que utilizar técnicas de Meta Learning para la implementación de modelos con características deseables (altas tasas de verdaderos positivos y negativos, un gran eficiencia global y un valor de precisión que implique altas probabilidades de acertar el resultado), aparecen como solución al problema presente en este set de datos.

La implementación de Meta Learning, se podría aplicar en el siguiente orden, en base a lo expuesto en los resultados.

1. Bernoulli.
2. Random Forest.
3. NuSVC.

Se espera que la aplicación de dichos algoritmos en conjunto, mediante una iteración secuencial de elementos, pueda aumentar el valor de las medidas de desempeño, sin dejar de lado la generalización del modelo y las características que esto conlleva.

Continuando con la selección de modelos, otro de los puntos que sería posible abarcar, implica la utilización de diversas métricas y diversos algoritmos, asociarles un peso a cada componente y generar la clasificación mediante votación, entregando una precisión ponderada, con respecto a los modelos utilizados y los valores que estos entregaban de manera individual.

El hecho de utilizar distintos modelos en un mismo problema de clasificación, permite suplir las deficiencias que en cierta manera, tienen los algoritmos, así, se producirían resultados más aceptables en cuanto a su valor de predicción. No obstante, la adición de pesos, contempla el hecho de generar una distribución de métricas individuales y acoplarlas al conjunto muestral. Dado esto, es imperante para dicha estrategia, considerar las importancias de cada medida y entregarle prioridades que se reflejen en la asignación de pesos.

Una posible representación, asociada a la generación de una clasificación, en base a lo expuesto previamente, es como se expone en la Figura 6.1.

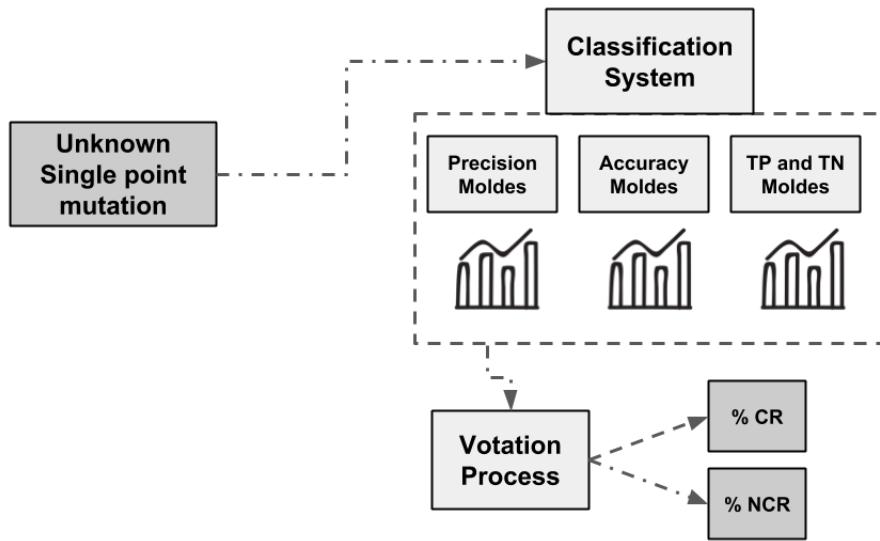


Figura 6.2: Sistema de clasificación y ponderación de elementos.

En la Figura 6.1, se expone que un conjunto de mutaciones puntuales son las entradas para el sistema de clasificación, dicha entidad, aplica los diversos modelos de clasificación que posee con respecto a las métricas expuestas y a la selección del conjunto de modelos que cumplieron con los criterios de consideración según metodología. Cada uno de estos modelos, generará un resultado, el cual será considerado en la ponderación final.

Dicho sistema de ponderación, contempla cada uno de los resultados y hace la sumatoria de ambas clasificaciones posibles (relevante clínicamente y no.), a las cuales les calcula el porcentaje que corresponde a su clasificación individual y en base a ésta, entrega un porcentaje del resultado, el cual está asociado a ambas clases, es decir, entrega un $x\%$ para la clase de relevancia clínica y un $y\%$ para aquella que presenta una no relevancia clínica. De esta forma, se genera una clasificación por votación de manera similar al funcionamiento de KNN.

Otro de los puntos importantes a discutir, es el análisis de las características, los cuales se realizaron en base a metodologías que implican una deformación de espacios, así como también transformaciones ortogonales de datos en base a cambios de base según técnica de PCA, los cuales, por un lado, arrojaron cuáles características eran de relevancia para generar un clasificador basado en random forest, mientras que la segunda implica la cantidad de componentes que resultan ser relevantes a la hora de considerar una reducción de la dimensionalidad. Ambas técnicas resultan ser efectivas, pero son influenciables por el modelo de clasificación o porque no entrega un valor específico de corte para determinar cuántas componentes son relevantes.

Dado a lo anterior, es posible utilizar otras técnicas que han sido implementadas

y estudiadas para el análisis de features y su selección, tales como: técnicas basadas en correlación de datos, Mutual Information, Evaluaciones Probabilísticas y teorías de conjuntos, etc.

Se espera que una disminución de la cantidad de atributos, asociados a técnicas de Meta Lerning, entreguen resultados más aceptables que los que se han encontrado. Es importante mencionar que, el trabajo relacionado con el set de datos completo, no está asociado a adición de información topológica, generar particiones, etc. Lo cual tiene relación con las hipótesis planteadas en el desarrollo de este proyecto.

6.2. Modelos en base a Set de Datos con particiones inducidas por Interacción Proteína-Proteína

La idea de generar particiones mediante la adición de información topológica, asociada a sitios de interacción proteína proteína, implicaba generar un conjunto de modelos de clasificación cuyas métricas fueran más altas que las obtenidas por los modelos asociados al set de datos full.

En la Figura 6.3, se observa una comparación del promedio de las medidas de desempeño obtenidas al utilizar los set de datos generados por las particiones mediante interacción proteína-proteína y el set de datos Full.



Figura 6.3: Comparación de Medidas de Desempeño en Full Data set v/s Divisiones inducidas por sectores de interacción

En general, las divisiones inducidas por la adición de la información topológica, presenta mejores resultados que los obtenidos por la utilización del set de datos full. Sin embargo,

es necesario justificar que dichas métricas no son debido solamente a la inducción de particiones, si no, más bien, es debido a que es por la adición de dicha información topológica.

Para probar lo anterior, en una primera instancia, se evaluaron que los modelos obtenidos para cada división eran propios de cada partición, para ello, se realizó una validación cruzada de los modelos, es decir, utilizar los modelos de la partición A para predecir el resto de particiones.

A modo de ejemplo, se expone en la Figura 6.4, la validación cruzada para el modelo Random Forest en la partición P, con respecto al resto de las divisiones.

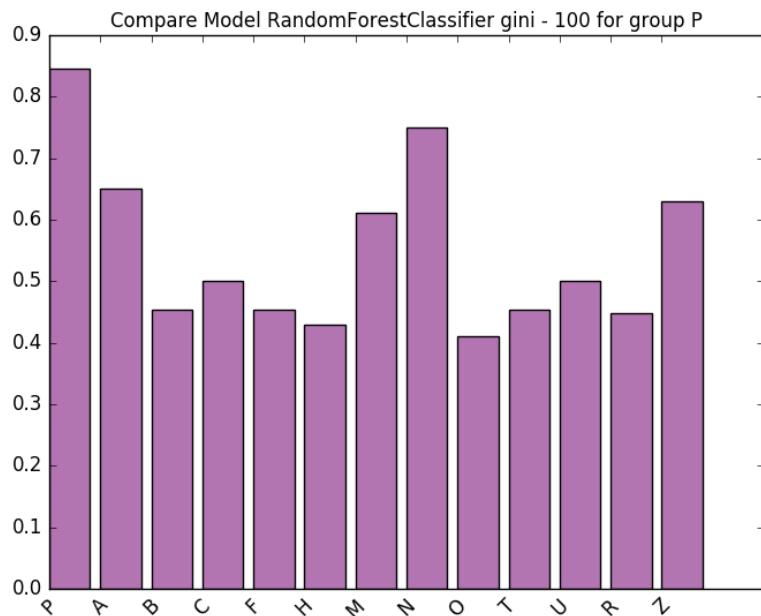


Figura 6.4: Validación cruzada para modelos P con respecto al resto de divisiones.

Tal como se aprecia en la Figura 6.4, el modelo utilizado para P, presenta resultados más bajos en la predicción con respecto a los restantes set de datos, lo cual demuestra, que en particular, el modelo es único para la división.

Un punto importante a destacar es que cada división implica un set de datos independientes, por lo que es normal pensar, que contemplan modelos independientes. Sin embargo, existen grupos, tales como N, U y P, que presentan modelos con las mismas características y algoritmos, lo cual implica que es posible generar una unión de dichos set de datos para crear nuevo set de datos y evaluar sus medidas de desempeño.

Otro de los puntos importantes a destacar, es que cada modelo seleccionado fue entrenado mediante la técnica Leave One Out. No obstante, se generó una distribución de

entrenamientos para cada algoritmo y sus parámetros, con el fin de evaluar la persistencia de los resultados y seleccionar aquellos modelos cuyas desviaciones estándar fuesen bajas y sus medias fuesen altas. Sin embargo, al utilizar esta técnica, las medidas de precisión y re-call disminuyeron drásticamente, lo que, a pesar, de tener valores de accuracy altos, las métricas relevantes consideradas previamente, fueron demasiado bajas.

Con el fin de poder evaluar si dicho comportamiento era afección sólo de la forma de entrenamiento, donde se contempla la distribución de un conjunto de set de pruebas, se utilizó la técnica de validación cruzada, para poder contrastar dichas métricas. Estos estudios se realizaron a diferentes valores de K, asociados a la validación y cantidad de iteraciones del entrenamiento. Al utilizar validación cruzada, los valores de accuracy se mantiene. No obstante, la precisión y el recall, subieron, razón por la cual, se denota que la metodología de entrenamiento, es la que afecta a los valores de dichas métricas.

Se destaca, que a pesar de ser set de datos con número bajo de ejemplos en algunos casos, se recomienda la validación cruzada para el entrenamiento de modelos y su testeo, de tal manera, es factible evaluar la generalización de modelos, sin verse afectada los valores obtenidos para la precisión y el recall, lo que se ve reflejado en tasas de verdaderos positivos y negativos.

También se evaluó que las herramientas utilizadas no tuvieron efecto en los resultados obtenidos, es decir, las implementaciones de modelos mediante Python utilizando la librería scikit-learn, no fuesen tan distantes a los obtenidos, por ejemplo, mediante la toolbox Classification-Learning de Matlab. Siendo los resultados similares a los obtenidos, por lo que la influencia de la implementación de los algoritmos no afecta a los resultados obtenidos.

Si bien, los modelos generados, en particular, presentan mejores resultados que los obtenidos asociados al set de datos full, en algunos casos, tales como las particiones A, M y R, no poseen medidas de desempeño deseadas, es decir, los valores de precisión debiesen estar cerca de los 0.9, lo cual en particular, casi ningún modelo lo logra, razón por la cual se necesita la implementación de técnicas asociadas a meta learning, o la generación de meta clasificadores, tal como se expuso en la sección 6.1 y en la Figura 6.1.

Todo lo anterior, denota, que si bien, la adición de información topológica implica que los modelos generados fueron mejores en comparación al set de datos full. No obstante, no son suficientes para cumplir las expectativas de modelos relacionados a esta área, lo cual denota que es necesario, generar nuevos estudios asociados a estos modelos. Además, es demostrable, que el problema no sólo contempla el hecho de hacer modelos de clasificación, si no que también, se torna a un nivel de complejidad mayor, debido a que existen ciertos factores que afectan los resultados, variables que denotan cambios significativos

y consideraciones que se requieren a la hora de desarrollar modelos en respecto a sistemas de clasificación en enfermedades.

6.3. Modelos en base a Set de Datos con particiones inducidas por Clustering

Los modelos obtenidos en base a particiones inducidas por técnicas de clustering, presentaron mejores resultados que al usar el set de datos completo. Las divisiones se implementaron según lo expuesto en la metodología y cumplían con el objetivo de no presentar problemas de desbalance de clases y maldición de la dimensionalidad.

Tal como se explicó en la sección de resultados, fueron diversos los algoritmos que cumplían con las características impuestas. No obstante, a la hora de seleccionar las particiones como tal, el principal criterio, fue que en promedio, la medida de precisión fuese alta y que a su vez, la desviación estándar que ésta posee, fuese baja, razón por la cual, fueron seleccionadas las divisiones generadas por 5 clustering, esto puede apreciarse en la Figura 6.5, en la que se pueden comparar las diferentes medidas de desempeño asociadas a cada partición.

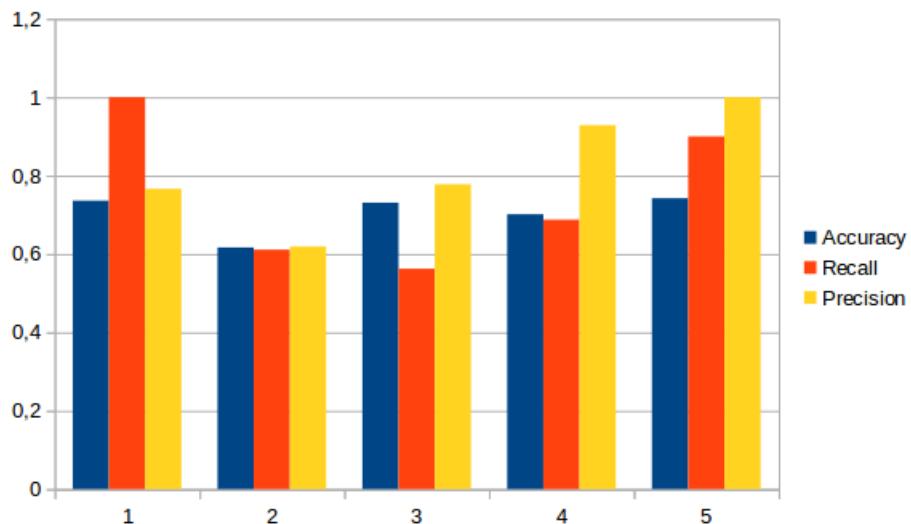


Figura 6.5: Resumen medias de medidas de desempeño para las particiones seleccionadas en inducción vía clustering.

En la Figura 6.5, la medida de Accuracy es la que presenta una menor desviación estándar, mientras que la precisión posee en promedio un valor de 0.8. Sin embargo, su valor mínimo es 0.62 y su máximo 1, lo cual denota que si bien, existe una partición un tanto azarosa, en general, los modelos presentan buenos comportamientos.

A pesar de que dichos modelos presentan comportamientos más favorables que utilizar el set de datos completo, el hecho de implementar esta metodología para nuevas mutaciones, requiere de la actualización de los centroides formados por cada partición, lo que implica que pueden generar cambios a la hora de procesar el modelo, además se crea la duda de que pueden presentar contradicciones o disparidades a la hora de evaluar las métricas de distancias, lo cual se vería reflejado en que una mutación esté a la misma distancia de dos o más grupos.

Pese a lo anterior, se cree que es necesario trabajar con la implementación del sistema de meta clasificación expuesto en secciones anteriores, esto con el fin de poder generar meta clasificadores que tengan en consideración diferentes características del sistema y contemple diversas evaluaciones asociados al modelo, lo cual presentaría grandes ventajas que tomar un modelo por si sólo y trabajar con éste.

6.4. Modelos en base a Set de Datos con particiones inducidas por Recursividad de Clustering

Las particiones generadas mediante clustering recursivo, se basan principalmente en el hecho de hacer constantes sub divisiones del set de datos, comparar estadísticamente dichas instancias y evaluar problemas asociados a desbalance de clases y dimensionalidad en el set de datos. Razón por la cual, a la hora de evaluar cada una de las particiones, se consideraron todos los factores expuestos en la sección de Metodología.

Se obtuvieron 7 particiones que cumplían con los requerimientos implantados, las cuales fueron entrenadas en fase exploratoria y sometidas a procesos de validación mediante técnicas de Leave One Out y Validación cruzada con $k=5$. Ambas validaciones entregaron resultados favorables a la hora de evaluar la medida de Accuracy. Sin embargo, como se expuso en secciones anteriores, la técnica de LOU, genera una influencia importante en las medidas de Precisión y Recall, impactando negativamente en dichos valores, causando bajas significativas, lo cual principalmente se debe al hecho de que la cantidad de entrenamientos es mayor y además se realizó una distribución de entrenamiento para cada algoritmo y parámetro, por lo que los resultados obtenidos son medias aritméticas de una muestra de resultados de entrenamiento. Considerando esto, se utiliza como medio de validación, la técnica de validación cruzada, debido a que genera resultados similares en cuanto a la Accuracy, pero tanto la Precisión como el Recall aumentan considerablemente y cumplen con el objetivo de evaluar sobreajustes y generalización en el modelo.

En base a lo expuesto anteriormente, se observa en la Figura 6.6, las medidas de desempeño obtenidas a través de las particiones inducidas mediante clustering recursivo.

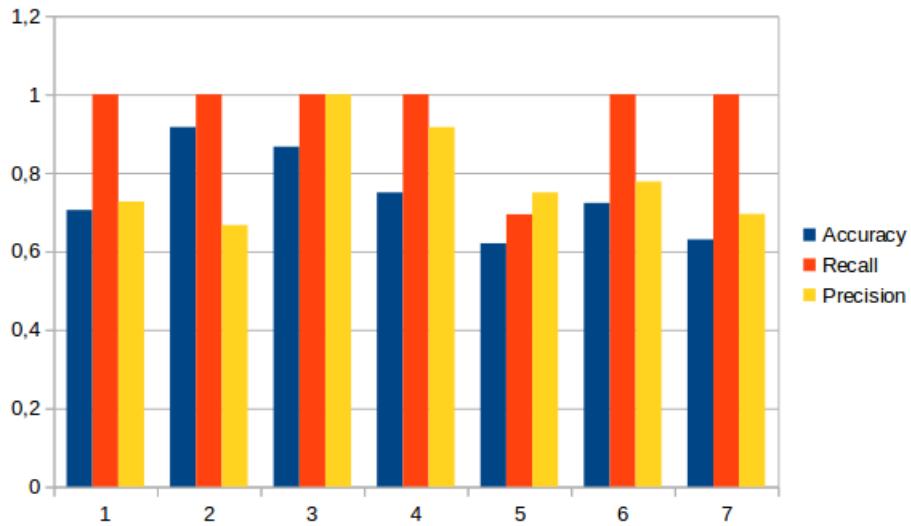


Figura 6.6: Resumen medias de medidas de desempeño para las particiones seleccionadas en inducción vía clustering recursivo.

Los resultados obtenidos son muy similares a los expuestos previamente al trabajar con particiones inducidas mediante técnicas de clustering, la razón es que, debido a que ambas trabajan de forma similar, es decir, utilizan técnicas de clustering para generar particiones, se hacen evaluaciones de distancia y se corroboran elementos con el fin de evaluar los problemas asociados a inconvenientes en el entrenamiento. Sin embargo, la principal diferencia, se expone en cómo se controlan las divisiones, ya que al utilizar técnicas convencionales, se evalúa un número finito de grupos al mismo tiempo, 2, 3, 4, etc, mientras que en el algoritmo de clustering recursivo, siempre se hace una evaluación en grupos de a dos y se hace la comparación entre ellos para determinar si es factible la correspondiente división. No obstante, la base de ambas técnicas, es similar, por lo que es factible que los resultados sean similares y que a su vez, los problemas para interpretar una nueva mutación sean similares.

No obstante, los resultados obtenidos por estas particiones, también indican que es necesario generar divisiones al set de datos de interés para obtener resultados más afines a lo que se espera de un modelo de clasificación de relevancia clínica en mutaciones. Sin embargo, se hace evidente la necesidad de implementar meta clasificadores para obtener resultados más favorables, considerar diversos modelos que atrapan, diferentes características deseables en un clasificador, razón por la cual, se piensa que aplicar la metodología propuesta para el set de datos de inicial entregaría resultados más confiables y con menor margen de error.

6.5. Evaluación de comunidades.

La adición de información topológica, asociada a la evaluación de sectores que se encuentran fuertemente interconectados mediante energías electrostáticas, implica trabajar en una primera instancia con la estructura 3D de la proteína y a partir de la información de coordenadas, lograr estimar energías de interacción que aporten a la estabilidad de la proteína y así disponer dicha información en estructuras de grafo.

La disposición de la información electrostática de una proteína en estructuras de grafos, permite aprovechar sus propiedades para encontrar sectores de interacción que se encuentren fuertemente conectados. Siendo una de las estrategias más relevantes para esta tarea, la búsqueda de comunidades.

Tal como se expuso en los resultados, las comunidades encontradas fueron del orden de alrededor de 20, lo cual, denota que es un número bastante elevado a la hora de entrenar modelos de clasificación para dichas comunidades, además, los elementos insertos en cada comunidad, estructuralmente hablando, hacían alusión que eran elementos que se encontraban asociadas covalentemente, lo que hace pensar, que, a pesar de que las matrices son normalizadas, la escalabilidad de energías entre electrostáticas y covalentes, hace referencia a que no fueron suficientes, por lo que falta una manera más inteligente de poder hacer dicha escalabilidad y poder disminuir el número de comunidades y hacer una reducción del peso que aporta la energía covalente a la estabilidad energética de la proteína.

Debido a que la cantidad de comunidades fue muy elevada con respecto a lo que se esperaba, no se tomó en consideración para generar modelos de clasificación según estas particiones, dado a que, por un lado, la cantidad de miembros era muy pequeño y el número de comunidades era muy elevado.

Uno de los posibles usos que es factible darle al desarrollo de comunidades, es sólo considerar la información existente en el set de datos y generar matrices de adyacencia a partir de métricas de distancias que presenten sus atributos en escalas normalizadas, es decir, considerar a cada ejemplo como un vector de dimensión 15 y calcular la distancia que existen entre cada uno de ellos. A partir de dicha matriz de distancia, sería posible considerar valores máximos, mínimos, promedios, medianas, etc, que permitan generar un punto de corte, con el fin de determinar si los sujetos de muestra se encuentran asociados o no y de dicha forma, generar matrices de adyacencia con uniones o con pesos, teniendo en considerar las características impuestas. De esta forma, servir como entrada para la búsqueda de comunidades y evaluar si existe un número de comunidades que sea de relevancia para estudios posteriores de desarrollo de modelos de clasificación.

6.6. Selección de particiones

Quedó demostrado que inducir particiones en el set de datos, aumenta cada una de las medidas de desempeño evaluadas para un clasificador, además de que es mucho más favorable utilizar conjuntos de clasificadores insertos en un meta clasificador para obtener predicciones más certeras, dado a que diversos algoritmos se complementan con respecto a las características que estos poseen, es decir, por ejemplo, SVM genera hiperplanos a través de transformaciones de kernels para generar separaciones en los datos y hacer la clasificación, lo cual denota una función con respecto a los datos en torno a su dimensión, caso contrario, es Random Forest, quien genera una deformación del espacio para evaluar logísticamente los atributos y concluir en modo recursivo a qué clase corresponde cada ejemplo, ambos permiten ser complementarios y facilitan anclar ambas respuestas.

Pese a que es necesario la utilización de sistemas de meta clasificación para el desarrollo de modelos de clasificadores en mutaciones, relacionadas con la relevancia clínica en VHL, también es menester utilizar particiones para aumentar las medidas de desempeño en las fases de entrenamiento, lo cual, si son altas, entrega una evaluación de robustez y persistencia ante cambios en los datos, lo cual se ve reflejado en las técnicas de validación de los modelos. Sin embargo, la gran interrogante se basa en cómo seleccionar la forma de hacer la partición.

En una primera instancia, simplemente se tomaría en consideración aquellas particiones que en promedio presentaran mayores valores en las medidas de desempeño que se están comparando. Una visualización de este comportamiento puede observarse en la Figura 6.7, en la cual se compara el promedio de las medidas de desempeño obtenidas en cada partición mediante cada técnica de interés, de izquierda a derecha: Clustering, Clustering Recursivo y Divisiones por Sector de Interacción PP.

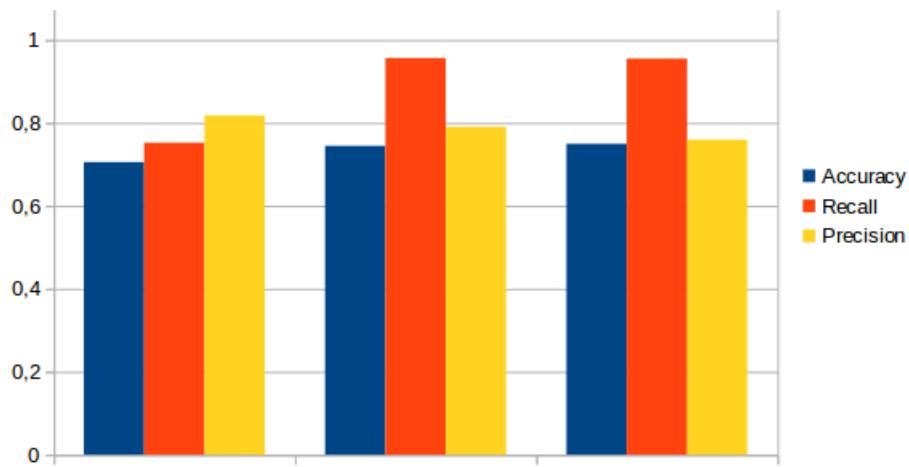


Figura 6.7: Comparación medidas de desempeño obtenidas para cada forma de partición.

Según lo observado en la Figura 6.7, simplemente se consideraría a la partición inducida por técnicas de clustering como la mejor división en promedio, debido a que presenta las tasas más altas de precisión. No obstante, los valores de recall y accuracy se ven disminuídos, por lo que sería prudente seleccionar las particiones generadas mediante clustering recursivo o a través de información topológica.

Si se considera las divisiones inducidas mediante clustering recursivo, a la hora de evaluar nuevas mutaciones, es necesario implementar la lógica de identificar a qué división pertenece, para ello, sería necesario considerar las distancias a los centroides, y en base a éstas determinar la partición y utilizar los modelos de ésta para la predicción. Sin embargo, nace la pregunta, qué pasa cuando la mutación sea validada por el usuario, el cual es el objetivo de interés, los grupos cambiarán?, los modelos serán alterados, etc. Al pensar en dicha instancia, es evidente reconocer que al agregar un nuevo elemento a un conjunto de elementos, el centroide de este grupo variará, además afectará a la homogeneidad del grupo y a la heterogeneidad del mismo con otros, lo que implicaría inclusive que en algunos casos, sea necesario un nuevo proceso de clustering recursivo, para hacer la actualización, razón que denota que en algún momento, la cantidad de grupos variará, los modelos se verán alterados y las medidas de desempeño puede que resulten perjudicadas, lo cual denota un problema de persistencia en el sistema de meta clasificación, al menos, a la hora de identificar a qué partición pertenece.

Caso contrario a lo que implicaría utilizar información topológica de la proteína, debido a que, ya se han registrado, reportado y analizado los trece sectores de interacción de proteína VHL con otras proteínas de interés, lo cual implica que los grupos entrenados en sí, no se vean afectados, además hace más fácil la detección de una nueva mutación y la identificación del sector de interacción al cual pertenece debido a que se tiene la información de la posición en la cual se detectó. Si bien, es completamente factible que los modelos se vean alterados al aceptar nuevas mutaciones y re entrenar los modelos, no se ve afectado el espacio muestral de las particiones en cuanto al número de éstas. Es de esperar que se afecte a los modelos, además es parte del proceso de entrenamiento del mismo, debido a que cumple con las necesidades de un sistema de meta clasificación con aprendizaje evolutivo, es decir, que constantemente va adquiriendo nueva información, la cual ha sido validada y en base a ella, se re entrena los modelos, se validan los nuevos conjuntos de clasificadores y se forman los sistemas de meta clasificación propuestos.

En base a lo anterior y denotando las ventajas que presentan algunas particiones por sobre otras, se tiene preferencia por las particiones basadas en información topológica asociada a sectores de interacción PP.

6.7. Unión de Sectores en Particiones inducidas por Interacción Proteína-Proteína

Tal como se expuso en secciones anteriores, existen grupos, cuyos modelos presentaron los mismos algoritmos y parámetros, lo cual hacía denotar que era posible unirlos en un único set de datos. Dado esto se sometió a fase exploratoria la unión de estos y se analizaron las medidas de desempeño obtenidas, así como también los algoritmos y sus respectivos parámetros.

Los grupos que fueron unidos son: B, N y U, quienes presentaban a Random Forest como algoritmo de aprendizaje supervisado, con diferentes parámetros asociados a dichas instancias.

Al hacer un análisis de los nuevos resultados, se aprecia que Random Forest, no permanece dentro de los algoritmos con mejores resultados. Sin embargo, los métodos basados en bagging presentan los valores más alto en Precisión con un 82 %, mientras que en Recall se obtiene el 100 % con métodos basados en SVM, además la eficiencia global alcanza un 80 % con algoritmos de Naive Bayes.

Los resultados asociados se pueden apreciar en la matriz de confusión, la cual se expone en la Figura 6.8, la cual denota valores de precisión altos, tal como se comentó previamente, además, las tasas de verdaderos positivos y negativos son altas.

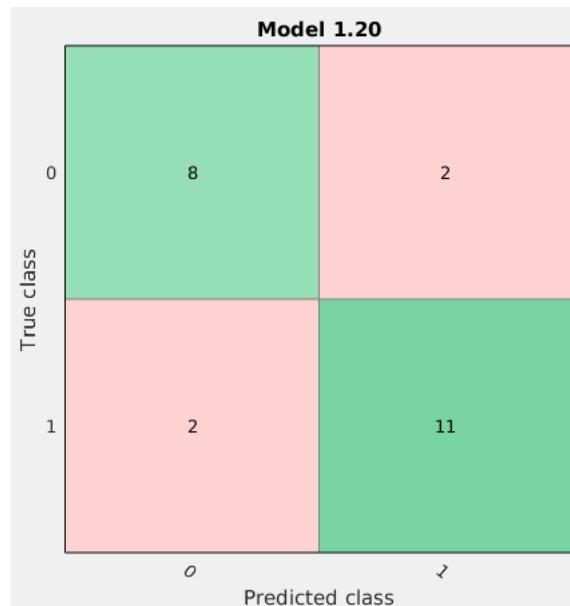


Figura 6.8: Matriz de confusión asociada a unión de grupos B-N-U en métodos asociados a Ensamble como algoritmo de entrenamiento

Por otro lado, en la Figura 6.9 se aprecia la curva ROC del proceso, en ella existe un

área general del 0.73 y no presenta muchas variaciones asociadas entre verdaderos y falsos positivos.

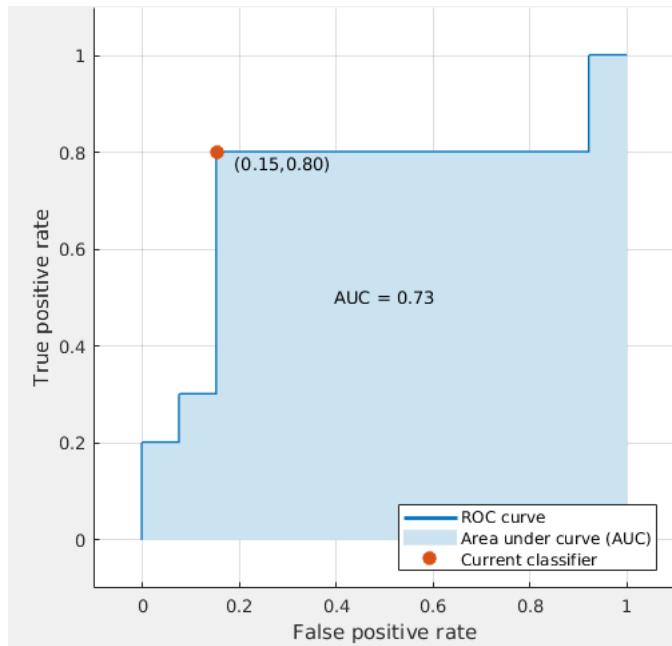


Figura 6.9: Curva ROC asociada a unión de grupos B-N-U en métodos asociados a Ensamble como algoritmo de entrenamiento

Finalmente, es discutible la utilización de la unión de estos grupos, en una primera instancia, se tendería a pensar que es mejor mantener los segmentos de interacción separados, debido a que los modelos presentan mejores medidas de precisión. Sin embargo, el hecho de poseer elementos con tan pocos integrantes, lo cual se pensaría que no es una muestra representativa y explicaría el porqué de los resultados obtenidos. Sin embargo, no se tienen más muestras asociadas a mutaciones en dichos sectores, lo cual implica que es muy complejo poseer más información para dichos sectores.

Por lo anterior, la razón de unir los sectores B, N y U, sólo considerando criterios estadísticos y matemáticos relacionados a la minería de datos y al desarrollo de modelos de clasificación se hace necesario, debido a que aumenta el espacio muestral y las medidas de desempeño no se ven afectadas de sobre manera. Sin embargo, si no se considera dicho hecho y se piensa que futuros casos es factible que aparezcan, es decir, es posible que se reporten nuevas mutaciones asociadas a dichos sectores de interacción, lo cual podría aportar nueva información a los set de datos correspondientes y establecer mejoras en los modelos de clasificación propuestos.

7. Conclusiones sobre resultados parciales

Ya con un porcentaje de desarrollo elevado en cuanto al trabajo y con algunos de los objetivos cumplidos y otros siendo pulidos, además siendo éste un trabajo de fase exploratoria, donde se han descubierto nuevas estrategias de desarrollos de modelos de clasificación, evaluaciones de características y análisis de resultados, se exponen algunas de las conclusiones más destacables, divididas por las temáticas expuestas.

7.1. Modelos en Set de Datos Completo.

Se obtuvieron conjuntos de modelos de clasificación para cada métrica de interés, los cuales representan puntos outliers en sus respectivos análisis de distribución de medidas de desempeño. Cada modelo por separado, no presenta grandes valores de métricas, denotando que dichos modelos, por sí solos, no cumplen las expectativas deseadas, razón por la cual, se necesita de la exploración de técnicas como meta learning, o el diseño e implementación de meta clasificadores que contemplen los resultados que entregan cada sub conjunto de modelos segun sus métricas de interés.

Los algoritmos basados en técnicas de ensamble, son mejores candidatos a la hora de trabajar set de datos con mayor número de elementos, además en conjunto con SVM, KNN y Naive Bayes, permiten generar una mejor explicación de los motivos de la clasificación que genera para cada mutación puntual, en contraste a MLP o cualquier otro algoritmo basado en redes neuronales.

El set de datos, no permite entregar información suficiente a los algoritmos para que las métricas de evaluación sean lo suficientemente buenas. No obstante, reducción de dimensionalidad, evaluación de componentes, análisis de importancia y/o otros, contempla la disminución de información y puede perjudicar al clasificador en vez de generar algún beneficio de éste.

Finalmente se espera, que el desarrollo de meta clasificadores, en base a meta lernaning o meta clasificación, satisfaga las necesidades de modelos de clasificación precisos y con alta eficiencia global para la evaluación de relevancia clínica en mutaciones asociadas a Vonn Hippel Lindau.

7.2. Modelos en base a Set de Datos con particiones inducidas por Interacción Proteína-Proteína

Se obtuvieron un conjunto de modelos asociados a cada set de datos independiente, cada sub conjunto representaba una serie de puntos outliers en la distribución de medidas de desempeño obtenidas, en particular, presentan 3 desviaciones estándar por sobre el promedio. Cada conjunto de modelos obtenidos para cada set de datos, es independiente entre sí. No obstante, existen sectores cuyos modelos poseen el mismo algoritmo de entrenamiento y los mismos parámetros, razón por la cual, se cree necesario que dichos set de datos se podrían juntar, lo cual se propone como trabajo a futuro para el área de desarrollo.

Los modelos obtenidos presentaron mejores resultados que la data obtenida al trabajar con el set de datos completo. Sin embargo, por sí sólos, en algunos sectores de interacción, no presentan los resultados esperados para los modelos de clasificación de esta envergadura, lo cual implica que es necesario desarrollar sistemas de meta clasificación en base a lo expuesto o planteado para el set de datos completo.

El hecho de trabajar con particiones asociadas a los sectores de interacción, contemplando información topológica de la proteína, aporta un mayor valor de medidas de desempeño, implica que el flujo de trabajo cambia un poco con respecto a la data para la clasificación, es decir, en una primera instancia, se evalúa a qué sector de interacción pertenece, luego se somete al sistema de meta clasificación, se procede a predecir la relevancia clínica para cada modelo propuesto y se genera la clasificación ponderada del sistema, obteniendo así la relevancia clínica a la cual se asocia dicha mutación.

7.3. Modelos en base a Set de Datos con particiones inducidas por Clustering

Los modelos desarrollados en base a las particiones inducidas por técnicas de clustering, por sí sólos presentan valores de precisión más altos con respecto a todas las formas de particiones que se realizaron. No obstante tanto la Accuracy como el Recall, son, en promedio, más bajo, aún así, son mejores en comparación al uso del set de datos completo.

A pesar de que los resultados, en términos de precisión son altos, en base a lo comentado en la sección 6.6 esta partición se descarta debido al hecho de que es posible que con nuevas mutaciones que se agreguen al sistema, se altere las cantidades de grupos, los espacios muestrales y las estructuras del proyecto en sí.

7.4. Modelos en base a Set de Datos con particiones inducidas por Recursividad de Clustering

Las particiones inducidas mediante clustering recursivo, presentan comportamientos similares a las obtenidas a través de clustering. No obstante, las medidas de Accuracy y Recall, presentan valores más altos a los obtenidos en las particiones por clustering y por supuesto a los obtenidos en el set de datos completo. Sin embargo, por las mismas razones expuestas en las particiones inducidas por técnicas de clustering, se rechaza la técnica utilizada.

7.5. Evaluación de comunidades.

La utilización de estructuras de grafos para la generación de particiones a partir de información estructural presentaba en la teoría grandes ventajas debido a que a partir de conocimiento topológico se podría haber inducido particiones que permitieran generar modelos de clasificación. No obstante, el uso de información energética se vió afectado por los valores de escala que entrega el uso de las energías covalente. Es decir, debido a que sus valores son varios factores de escala más altos que energías de interacción, estos son los que lideran principalmente las particiones.

Se obtuvieron cerca de 20 comunidades utilizando las técnicas de teorías de grafos, las cuales fueron descartadas por ser un número muy elevado, y por tener una cantidad reducida de integrantes.

Se espera que generar nuevas matrices de adyacencia basadas en las distancias entre los ejemplos en el set de datos, permita la generación de un número menor de comunidades y con una mayor cantidad de integrantes, los cuales sirvan para entrenar modelos de clasificación y obtener altos valores en sus respectivas medidas de desempeño.

7.6. Conclusiones finales

Se han desarrollado diversos modelos asociados a diferentes set de datos, contemplando distintas técnicas de particionar un set de datos y evaluando los resultados en la necesidad

final de un sistema de meta clasificación con base a las distribuciones de las medidas de desempeño y generando clasificaciones ponderadas según las métricas que se utilicen y los modelos implicados.

Se descartaron las particiones asociadas a distancias y se seleccionaron aquellas que trabajaban en base a la información topológica, finalmente según todo el proceso desarrollado en esta etapa exploratoria, se propone el siguiente flujo de trabajo para la evaluación de nuevas mutaciones.

1. Recibir como entrada: El residuo original, la mutación detectada y la posición en la que se encontró.
2. Calcular los atributos asociados a valores estructurales.
3. Calcular los atributos asociados a características filogenéticas.
4. Identificar el sector de interacción al cual pertenece la mutación a evaluar.
5. Predecir con los diferentes modelos asociados al sector de interacción entrenado.
6. Evaluar las ponderaciones de las clasificaciones y generar las probabilidades para caso de interés.
7. Exportar los resultados y esperar validación.

Con los pasos anteriores, es posible evaluar una nueva mutación reportada y determinar si es clínicamente relevante o no.

8. Estado del Trabajo y Desarrollos a Futuro

Actualmente, el trabajo se encuentra en obtención de resultados finales, discusiones sobre selección de modelos y particiones a utilizar y la evaluación de las diferentes ramas de trabajo a desarrollar. A continuación, se expone el estado y los trabajos a futuros por cada generación de modelos, según las particiones generadas.

8.1. Manejo de Set de Datos Completo

Dentro de los trabajos a futuro a desarrollar considerando el set de datos completo se encuentran el análisis de características mediante técnicas como: Mutual Information, análisis de matrices de correlaciones y covarianzas, evaluación de técnicas de conjuntos, además de analizar cómo afecta la eliminación de características en las medidas de desempeño de cada modelo.

Además se encuentra el manejo de meta learning sobre los modelos encontrados, los mejores en cada medida de desempeño y el diseño e implementación del sistema de clasificación en base a votaciones ponderadas, el cual fue explicado en la sección 6.1.

8.2. Particiones inducidas por Sectores de Interacción Proteína-Proteína

Los trabajos a futuro asociados a las particiones asociadas a los sectores de interacción, requieren de los siguientes desarrollos pendientes:

- Unir set de datos con algoritmos y parámetros en modelos de clasificación con mismo valor, someter a fase exploratoria de entrenamiento y evaluar el resultado y persistencia de los algoritmos y parámetros.

- Emplear técnicas de meta clasificación, asociadas a meta-learning para aumentar medidas de desempeño en algunas particiones con bajos resultados por sobre los esperados.
- Diseñar el proceso completo de implementación del flujo de trabajo.

8.3. Particiones inducidas por clúster

Debido a que las particiones inducidas por clustering no serán consideradas para efectos de desarrollo de sistemas de meta clasificación, no se necesitan, por ahora, más trabajos asociados a esta área de desarrollo.

8.4. Particiones inducidas por recursividad de clúster

Debido a que las particiones inducidas por clustering recursivo no serán consideradas para efectos de desarrollo de sistemas de meta clasificación, no se necesitan, por ahora, más trabajos asociados a esta área de desarrollo.

8.5. Particiones inducidas por comunidades

Al desarrollar particiones mediante comunidades, las que fueron generadas no resultaron óptimas, debido a los comentarios expuestos. En base a esto, como trabajos pendientes se tiene que, diseñar e implementar nuevas formas de desarrollo de matrices de adyacencia, sin considerar la información asociada a las coordenadas para estimar los cálculos energéticos, estas nuevas matrices trabajarán en base a la información existente en el set de datos y se evaluarán distintas métricas y formas de evaluación de unión de nodos. Las matrices serán testeadas mediante algoritmos de comunidades y se evaluarán con indicadores de grafos. Finalmente serán sometidas a fase exploratoria de modelos de clasificación, aquellas que no presenten un número elevado de comunidades, cuya cantidad de miembros sea mayor al 10 % con respecto al set de datos y que no presenten un desbalance de clases.

9. Exposiciones y Postulaciones

Uno de los objetivos implícitos en el desarrollo de una investigación es la publicación de diversos artículos científicos asociados a ésta, además de dar a conocer la temática en diferentes instancias y recibir opiniones y comentarios de feedback de entidades expertas en el área, con el fin de poder dar continuidad al desarrollo de esta iniciativa, enfocarla en puntos específicos y ahondar nuevas temáticas dentro de la misma línea de investigación o desarrollo de nuevas metodologías para la aplicación de distintas áreas de desarrollo.

Es por esto, que el proyecto fue presentado en:

- VHL-Hunter, Servicio web de clasificación de mutaciones en VHL, exposición oral en Workshop Científico CeBiB, Julio 2018.
- VHL-Hunter, Servicio web de clasificación de mutaciones en VHL, ponencia en forma de póster en Workshop Científico CeBiB, Julio 2018.

Adicional a ello, el proyecto ha sido postulado a congresos científicos en modalidad de exposición oral y como ponente en póster en los siguientes encuentros:

- 9th Argentinian Congress of Bioinformatics and Computational Biology (9CAB2C), Noviembre 2018.
- ISCB-LA SOIBIO EMBnet, Noviembre 2018, en el cual fue aceptada la postulación.
- Latin America SCS, Noviembre 2018.
- VII Encuentro de Investigación de Estudiantes de Postgrado UBB - 2018.

Bibliografía

- [1] Andres Irback, Simon Mitternacht, Sandipan Mohanty, *An effective all-atom potencial for proteins.*
- [2] M. L. Hekkelman, T. A. H. te Beek, S. R. Pettifer, D. Thorne, T. K. Attwood, G. Vriend, *WIWS: a protein structure bioinformatics Web service collection.*
- [3] Abdi. H., And Williams, L.J. (2010). “Principal component analysis”. Wiley Interdisciplinary Reviews: Computational Statistics. 2 (4): 433-459. doi:10.1002/wics.101.
- [4] Jolliffe I.T. Principal Component Analysis, Series: Springer Series in Statistics, 2nd ed., Springer, NY, 2002, XXIX, 487 p. 28 illus. ISBN 978-0-387-95442-4.
- [5] Bengio, Y.; et al. (2013). “Representation Learning: A Review and New Perspectives”(PDF). Pattern Analysis and Machine Intelligence. 35 (8): 1798-1828. doi:10.1109/TPAMI.2013.50.
- [6] S. Kotsiantis, supervisado Aprendizaje Automático: Una Revisión de la Clasificación de las técnicas de Informática Diario 31 (2007) 249-268.
- [7] “Five balltree construction algorithms”, Omohundro, S.M., International Computer Science Institute Technical Report (1989).
- [8] “The DISTANCE Procedure: Proximity Measures”. SAS/STAT 9.2 Users Guide. SAS Institute. Retrieved 2009-04-26.
- [9] “Automatic Capacity Tuning of Very Large VC-dimension Classifiers”, I. Guyon, B. Boser, V. Vapnik - Advances in neural information processing 1993.
- [10] Cortes, C.; Vapnik, V. (1995). “Support-vector networks”. Machine Learning. 20 (3): 273-297. doi:10.1007/BF00994018.

- [11] Pedregosa, F. Varoquaux, G. Gramfort, A. Michel, V. (2010-2016). Scikit-learn: Support Vector Machines. INRIA. Recuperado de <http://scikit-learn.org/stable/modules/svm.html/kernel-functions>.
- [12] Lebart,L.;Morineau,A. y Fenelon,J.P.: "Traitement des Donées Statistiques" Dunod. 1979.
- [13] "k-means++: The advantages of careful seeding." Arthur, David, and Sergei Vassilvitskii, Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics (2007).
- [14] Brendan J. Frey; Delbert Dueck (2007). "Clustering by passing messages between data points". Science. 315 (5814): 972-976. doi:10.1126/science.1136800. PMID 17218491.
- [15] Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2009). 14.3.12 Hierarchical clustering (PDF). The Elements of Statistical Learning (2nd edición). Nueva York: Springer. pp. 520-528. ISBN 0-387-84857-6. Consultado el 20 de diciembre de 2016.
- [16] "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise." Ester, M., H. P. Kriegel, J. Sander, and X. Xu, In Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, Portland, OR, AAAI Press, pp. 226-231. 1996.
- [17] Vinh, Epps, and Bailey, (2009). "Information theoretic measures for clusterings comparison". Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09. doi:10.1145/1553374.1553511. ISBN 9781605585161.
- [18] Peter J. Rousseeuw (1987). "Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis". Computational and Applied Mathematics 20: 53-65. doi:10.1016/0377-0427(87)90125-7.
- [19] Calinski, T., & Harabasz, J. (1974). "A dendrite method for cluster analysis". Communications in Statistics-theory and Methods 3: 1-27. doi:10.1080/03610926.2011.560741.
- [20] Bengio, Y.; Courville, A.; Vincent, P. (2013). "Representation Learning: A Review and New Perspectives". IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (8): 1798-1828. arXiv:1206.5538. doi:10.1109/tpami.2013.50.
- [21] Bengio, Y.; et al. (2013). "Representation Learning: A Review and New Perspectives"(PDF). Pattern Analysis and Machine Intelligence. 35 (8): 1798-1828. doi:10.1109/TPAMI.2013.50.

- [22] Bengio, Yoshua (2009). “Learning Deep Architectures for AI”(PDF). Foundations and Trends in Machine Learning 2 (1): 1-127. doi:10.1561/2200000006.
- [23] Deng, L.; Yu, D. (2014). “Deep Learning: Methods and Applications”(PDF). Foundations and Trends in Signal Processing 7: 3-4. doi:10.1561/2000000039.
- [24] Deep Machine Learning - A New Frontier in Artificial Intelligence Research - a survey paper by Itamar Arel, Derek C. Rose, and Thomas P. Karnowski. IEEE Computational Intelligence Magazine, 2013.
- [25] Ian Goodfellow, Yoshua Bengio, and Aaron Courville (2016). Deep Learning. MIT Press. Online.
- [26] Olshausen, B. A. (1996). “Emergence of simple-cell receptive field properties by learning a sparse code for natural images”. Nature 381 (6583): 607-609. doi:10.1038/381607a0.
- [27] J. H. Holland. University of Michigan Press, Ann Arbor. 1975. Adaptation in Natural and Artificial Systems.
- [28] D. E. Goldberg. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. 1989. Genetic Algorithms in Search, Optimization and Machine Learning.
- [29] “More deeply, the framework exists to separate the representation of information from user interaction.”The DCI Architecture: A New Vision of Object-Oriented Programming-Trygve Reenskaug and James Coplien, March 20, 2009.
- [30] “... the user input, the modeling of the external world, and the visual feedback to the user are explicitly separated and handled by three types of object.”, Applications.
- [31] Programming in Smalltalk-80(TM):How to use Model-View-Controller (MVC).
- [32] Simple Example of MVC (Model View Controller) Design Pattern for Abstraction
- [33] *Data Mining Curriculum*. ACM SIGKDD. 2006-04-30. Retrieved 2014-01-27.
- [34] Clifton, Christopher (2010). *Encyclopedia Britannica: Definition of Data Mining*. Retrieved 2010-12-09.
- [35] Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Retrieved 2012-08-07.
- [36] Fayyad, Usama; Piatetsky-Shapiro, Gregory; Smyth, Padhraic (1996). *From Data Mining to Knowledge Discovery in Databases* (PDF). Retrieved 17 December 2008.

- [37] Linehan WM, Zbar B, Klausner R. Renal carcinoma. In: Scriver CR, Beaudet AL, Sly WS, Valle D, eds. *The metabolic and molecular basis of inherited disease*. 8th edn. New York: McGraw-Hill, 2001:907-29.
- [38] Linehan WM, Zbar B, Klausner R. Renal Carcinoma. In: Vogelstein B, Kinzler K, eds. *The genetic basis of human cancer*. 2 edn. New York: McGraw-Hill, 2002.
- [39] Linehan WM, Lerman MI, Zbar B. Identification of the von Hippel-Lindau (VHL) gene. Its role in renal cancer. *JAMA* 1995; 273:564-70.
- [40] Clifford SC, Maher ER. Von Hippel-Lindau disease: clinical and molecular perspectives. *Adv Cancer Res* 2001;82: 85-105.
- [41] Maher ER, Kaelin WG Jr. von Hippel-Lindau disease. *Medicine (Baltimore)* 1997; 76: 381-91.
- [42] Kaelin WG Jr. Molecular basis of the VHL hereditary cancer syndrome. *Nat Rev Cancer* 2002; 2:673-82.
- [43] Russell R Lonser, Gladys M Glenn, McClellan Walther, Emily Y Chew, Steven K Libutti, W Marston Linehan, Edward H Oldfield, *von Hippel-Lindau disease*, *Lancet* 2003; 361: 2059-67.
- [44] Maher ER, Kaelin WG Jr, *von Hippel-Lindau disease*, *Medicine* [01 Nov 1997, 76(6):381-391], (PMID:9413424), DOI: 10.1097/00005792-199711000-00001.
- [45] Gossage, L., Pires, D. E. V., Olivera-Nappa, A., Asenjo, J., Bycroft, M., Blundell, T. L., and Eisen, T. (2014). *An integrated computational approach can classify VHL missense mutations according to risk of clear cell renal carcinoma*. *Human Molecular Genetics*, 23(22), 5976-5988. ISSN:0964-6906.
- [46] Christophe Béroud, Dominique Joly, Catherine Gallou, Frédéric Staroz, Marie Therése Orfanelli and Claudine Junien, *Software and database for the analysis of mutations in the VHL gene*, *Nucleic Acids Research*, 1998, Vol. 26, No. 1.
- [47] Duan DR, Pause A, Burgess WH, Aso T, Chen DY, Garrett KP, Conaway RC, Conaway JW, Linehan WM, Klausner RD. *Inhibition of transcription elongation by the VHL tumor suppressor protein*. *Science*. 1995 Sep 8;269(5229):1402-6.
- [48] Kishida T, Stackhouse TM, Chen F, Lerman MI, Zbar B. *Cellular proteins that bind the von Hippel-Lindau disease gene product: mapping of binding domains and the effect of missense mutations.*, *Cancer Res*. 1995 Oct 15;55(20):4544-8.

- [49] Schoenfeld AR, Davidowitz EJ, Burk RD. *Elongin BC complex prevents degradation of von Hippel-Lindau tumor suppressor gene products*. Proc Natl Acad Sci U S A. 2000 Jul 18;97(15):8507-12.
- [50] Lamiell JM, Salazar FG, Hsia YE. von Hippel-Lindau disease affecting 43 members of a single kindred. Medicine 1989; 68:1-29.
- [51] Escourolle R, Poirer J. Manual of Neuropathology. 2nd edn. Philadelphia: WB Saunders, 1978: 49-51.
- [52] Melmon KL, Rosen SW. Lindau s disease. Am J Med 1964; 36: 595-617.
- [53] Catherine L. Worth Robert Preissner Tom L. Blundell, *SDM-a server for predicting effects of mutations on protein stability and malfunction*, Nucleic Acids Research, Volume 39, Issue suppl_2, 1 July 2011, Pages W215-W222, <https://doi.org/10.1093/nar/gkr363>
- [54] Alvaro Olivera-Nappa, Barbara A Andrews and Juan A Asenjo, *Mutagenesis Objective Search and Selection Tool (MOSS): an algorithm to predict structure-function related mutations in proteins*, BMC Bioinformatics 2011 12:122.

10. Anexos

10.1. Tabla Modelos de clasificación generados según partición por sectores de interacción

Modelos para Sector A			
Modelo	Descripción	Métrica	Valor Métrica
Naive Bayes	Bernoulli	Accuracy	0.65
MLP	logistic-adam-adaptive (10-15-15)	Precision	0.55
MLP	logistic-adam-invscaling (10-15-15)	Re-call	0.5495
MLP	logistic-adam-adaptive (10-15-15)		0.5495
KNN	euclidean-uniform KNN: 4	TN	6
KNN	minkowski-uniform KNN: 2		6

KNN	minkowski-uniform KNN: 4		6
SVM	SVC-sigmoid		6
MLP	logistic-adam-invscaling (10-15-15)	TP	11
MLP	logistic-adam-adaptive (10-15-15)		11

Modelos para Sector B

Random Forest	Gini, 50		0.818181818182
Random Forest	Gini, 100		0.818181818182
Random Forest	Gini, 150		0.818181818182
Random Forest	Gini, 200		0.818181818182
Random Forest	Gini, 250		0.818181818182
Random Forest	Gini, 500		0.818181818182

Random Forest	Gini, 750		0.818181818182
Random Forest	Gini, 1000		0.818181818182
SVM	SVC-sigmoid	Precision	0.545454545455
SVM	NuSVC-sigmoid		0.545454545455
SVM	SVC-sigmoid	Re-call	0.545454545455
SVM	NuSVC-sigmoid		0.545454545455
Random Forest	Gini, 50	TN	4
Random Forest	Gini, 100		4
Random Forest	Gini, 150		4
Random Forest	Gini, 200		4
Random Forest	Gini, 250		4

Random Forest	Gini, 500		4
Random Forest	Gini, 750		4
Random Forest	Gini, 1000		4
SVM	SVC-sigmoid	TP	6
SVM	NuSVC-sigmoid		6

Modelos para Sector C

SVM	SVC-Linear	Accuracy	0.7
SVM	NuSVC-Linear		0.7
MLPClassifier	relu-sgd-invscaling (15-5-5)	Precision	224
MLPClassifier	Relu-lbfgs-adaptative (5-5-10)	Re-call	213
KNN	minkowski-uniform KNN: 2		6

TN

KNN	minkowski-uniform KNN: 3		6
KNN	minkowski-uniform KNN: 4		6
MLPClassifier	Relu-sgd-invscaling (10-10-10)	TP	2.22

Modelos para Sector F

SVM	NuSVC-kernel: poly	Accuracy	0.7272727272727276
SVM	NuSVC-kernel: rbf		0.7272727272727276
KNN	Auto-minkowski-distance KNN: 2		0.7272727272727276
KNN	Auto-euclidean-distance KNN: 2		0.7272727272727276
KNN	Ball-tree-minkowski-distance KNN: 2		0.7272727272727276
KNN	Ball-tree-euclidean-distance KNN: 2		0.7272727272727276
KNN	KD-tree-minkowski-distance KNN: 2		0.7272727272727276

KNN	KD-tree-euclidean-distance KNN: 2	Precision	0.7272727272727276
KNN	Brute-minkowski-distance KNN: 2		0.7272727272727276
KNN	Brute-euclidean-distance KNN: 2		0.7272727272727276
SVM	SVC-Linear		0.545454545455
SVM	NuSVC-kernel: poly	Precision	0.545454545455
SVM	NuSVC-kernel: rbf		0.545454545455
SVM	SVC-Linear		0.545454545455
SVM	NuSVC-kernel: poly	Re-call	0.545454545455
SVM	NuSVC-kernel: rbf		0.545454545455
KNN	Auto-minkowski-distance KNN: 2		4
KNN	Auto-euclidean-distance KNN: 2	TN	4

KNN	Ball-tree-minkowski-distance KNN: 2		4
SVM	SVC-Linear		6
SVM	NuSVC-kernel: poly	TP	6
SVM	NuSVC-kernel: rbf		6

Modelos para Sector H

SVM	NuSVC-kernel: sigmoid	Accuracy	0.571428571429
SVM	NuSVC-kernel: sigmoid	Precision	0.571428571429
SVM	NuSVC-kernel: sigmoid	Re-call	0.571428571429
MLP	relu-sgd-invscaling (5-10-5)	TN	1.6
SVM	NuSVC-kernel: sigmoid	TP	4

Modelos para Sector M

KNN	minkowski-distance KNN: 3	Accuracy	0.722222222222
KNN	euclidean-distance KNN: 3		0.722222222222
KNN	minkowski-distance KNN: 3		0.722222222222
KNN	euclidean-distance KNN: 3		0.722222222222
KNN	minkowski-distance KNN: 3		0.722222222222
KNN	euclidean-distance KNN: 3		0.722222222222
KNN	minkowski-distance KNN: 3		0.722222222222
KNN	euclidean-distance KNN: 3		0.722222222222
SVM	NuSVC-kernel: poly	Precision	0.277777777778
KNN	minkowski-distance KNN: 2		0.277777777778
KNN	euclidean-distance KNN: 2		0.277777777778

KNN	minkowski-distance KNN: 2		0.277777777778
KNN	euclidean-distance KNN: 2		0.277777777778
KNN	minkowski-distance KNN: 2		0.277777777778
KNN	euclidean-distance KNN: 2		0.277777777778
KNN	minkowski-distance KNN: 2		0.277777777778
KNN	euclidean-distance KNN: 2		0.277777777778
SVM	NuSVC-kernel: poly		0.277777777778
KNN	minkowski-distance KNN: 2		0.277777777778
KNN	euclidean-distance KNN: 2		0.277777777778
KNN	minkowski-distance KNN: 2		0.277777777778
KNN	euclidean-distance KNN: 2	Re-call	0.277777777778

KNN	minkowski-distance KNN: 2		0.277777777778
KNN	euclidean-distance KNN: 2		0.277777777778
KNN	minkowski-distance KNN: 2		0.277777777778
KNN	euclidean-distance KNN: 2		0.277777777778
SVM	SVC-Linear	TN	10
SVM	SVC-sigmoid		10
KNN	minkowski-distance KNN: 2	TP	5
KNN	euclidean-distance KNN: 2		5
KNN	minkowski-distance KNN: 2		5
KNN	euclidean-distance KNN: 2		5
KNN	minkowski-distance KNN: 2		5

KNN	euclidean-distance KNN: 2	5	
KNN	minkowski-distance KNN: 2	5	
KNN	euclidean-distance KNN: 2	5	
Modelos para Sector N			
Random Forest	Gini, 50	Accuracy	0.75
Random Forest	Gini, 100		0.75
Random Forest	Gini, 150		0.75
Random Forest	Gini, 200		0.75
Random Forest	Gini, 250		0.75
Random Forest	Gini, 500		0.75
Random Forest	Gini, 750		0.75

Random Forest	Gini, 1000	Precision	0.75
Random Forest	Gini, 50	Precision	0.75
Random Forest	Gini, 100	Precision	0.75
Random Forest	Gini, 150	Precision	0.75
Random Forest	Gini, 200	Precision	0.75
Random Forest	Gini, 250	Precision	0.75
Random Forest	Gini, 500	Precision	0.75
Random Forest	Gini, 750	Precision	0.75
Random Forest	Gini, 1000	Precision	0.75
Random Forest	Gini, 50	Re-call	0.75
Random Forest	Gini, 100	Re-call	0.75

Random Forest	Gini, 150		0.75
Random Forest	Gini, 200		0.75
Random Forest	Gini, 250		0.75
Random Forest	Gini, 500		0.75
Random Forest	Gini, 750		0.75
Random Forest	Gini, 1000		0.75
Random Forest	Gini, 50		0
Random Forest	Gini, 100		0
Random Forest	Gini, 150		0
Random Forest	Gini, 200	TN	0
Random Forest	Gini, 250		0

Random Forest	Gini, 500		0
Random Forest	Gini, 750		0
Random Forest	Gini, 1000		0
Random Forest	Gini, 50	TP	3
Random Forest	Gini, 100	TP	3
Random Forest	Gini, 150	TP	3
Random Forest	Gini, 200	TP	3
Random Forest	Gini, 250	TP	3
Random Forest	Gini, 500	TP	3
Random Forest	Gini, 750	TP	3
Random Forest	Gini, 1000	TP	3

Modelos para Sector O			
		Accuracy	0.717948717949
KNN	minkowski-distance KNN: 2		
KNN	euclidean-distance KNN: 2		
KNN	minkowski-distance KNN: 2		
KNN	euclidean-distance KNN: 2		
KNN	minkowski-distance KNN: 2		
KNN	euclidean-distance KNN: 2		
KNN	minkowski-distance KNN: 2		
SVM	NuSVC-kernel: sigmoid	Precision	0.333333333333
SVM	NuSVC-kernel: sigmoid	Re-call	0.333333333333

MLPClassifier	logistic-sgd-constant (5-5-15)		23.0
MLPClassifier	logistic-sgd-constant (10-5-15)		23.0
MLPClassifier	logistic-sgd-constant (15-15-15)		23.0
MLPClassifier	logistic-sgd-adaptive (5-5-15)		23.0
MLPClassifier	logistic-sgd-adaptive (5-10-15)	TN	23.0
MLPClassifier	logistic-sgd-adaptive (5-15-15)		23.0
MLPClassifier	logistic-sgd-adaptive (10-5-15)		23.0
MLPClassifier	logistic-sgd-adaptive (10-10-15)		23.0
MLPClassifier	logistic-sgd-adaptive (10-15-15)		23.0
SVM	NuSVC-kernel: sigmoid	TP	13
Modelos para Sector P			

Random Forest	Gini, 50		0.846153846154
Random Forest	Gini, 100		0.846153846154
Random Forest	Gini, 150		0.846153846154
Random Forest	Gini, 200		0.846153846154
Random Forest	Gini, 250		0.846153846154
Random Forest	Gini, 500		0.846153846154
Random Forest	Gini, 750		0.846153846154
Random Forest	Gini, 1000		0.846153846154
Naive Bayes	Bernoulli	Precision	0.38
Naive Bayes	Bernoulli	Re-call	0.38
Random Forest	Gini, 50		7

Random Forest	Gini, 100		7
Random Forest	Gini, 150		7
Random Forest	Gini, 200		7
Random Forest	Gini, 250		7
Random Forest	Gini, 500		7
Random Forest	Gini, 750		7
Random Forest	Gini, 1000		7
Naive Bayes	Bernoulli	TP	5

Modelos para Sector R

NuSVC	kernel: linear		0.655172413793
Naive Bayes	BernoulliNB	Accuracy	0.620689655172

NuSVC	kernel: sigmoid		0.603448275862
SVC	linear	Precision	0.568965517241
MLPClassifier	logistic-lbfgs-constant (5-5-15)		0.568965517241
MLPClassifier	logistic-lbfgs-constant (5-10-10)		0.568965517241
MLPClassifier	logistic-lbfgs-constant (5-10-15)		0.568965517241
MLPClassifier	logistic-lbfgs-constant (5-15-5)		0.568965517241
MLPClassifier	logistic-lbfgs-constant (5-15-10)		0.568965517241
MLPClassifier	logistic-lbfgs-constant (10-5-15)		0.568965517241
MLPClassifier	logistic-lbfgs-constant (10-10-15)		0.568965517241
MLPClassifier	logistic-lbfgs-constant (10-15-5)		0.568965517241
SVC	linear		0.568965517241

MLPClassifier	logistic-lbfgs-constant (5-5-15)		0.568965517241
MLPClassifier	logistic-lbfgs-constant (5-10-10)		0.568965517241
MLPClassifier	logistic-lbfgs-constant (5-10-15)		0.568965517241
MLPClassifier	logistic-lbfgs-constant (5-15-5)		0.568965517241
MLPClassifier	logistic-lbfgs-constant (5-15-10)		0.568965517241
MLPClassifier	logistic-lbfgs-constant (10-5-15)		0.568965517241
MLPClassifier	logistic-lbfgs-constant (10-10-15)		0.568965517241
MLPClassifier	logistic-lbfgs-constant (10-15-5)		0.568965517241
Naive Bayes	GaussianNB	TN	19
SVC	linear		33.0
MLPClassifier	logistic-lbfgs-constant (5-5-10)		33.0

MLPClassifier	logistic-lbfgs-constant (5-5-15)	33.0
MLPClassifier	logistic-lbfgs-constant (5-15-15)	33.0
MLPClassifier	logistic-lbfgs-constant (10-5-10)	33.0
MLPClassifier	logistic-lbfgs-constant (10-5-15)	33.0
MLPClassifier	logistic-lbfgs-constant (10-10-15)	33.0
MLPClassifier	logistic-lbfgs-constant (10-15-15)	33.0
MLPClassifier	logistic-lbfgs-constant (15-5-10)	33.0

Modelos para Sector T

KNN	minkowski-uniform KNN: 4	0.727272727273
KNN	euclidean-uniform KNN: 4	0.727272727273
KNN	minkowski-uniform KNN: 4	0.727272727273

Accuracy

KNN	euclidean-uniform KNN: 4		0.727272727273
KNN	minkowski-uniform KNN: 4		0.727272727273
KNN	euclidean-uniform KNN: 4		0.727272727273
KNN	minkowski-uniform KNN: 4		0.727272727273
KNN	euclidean-uniform KNN: 4		0.727272727273
SVC	sigmoid	Precision	0.454545454545
NuSVC	kernel: sigmoid	Precision	0.454545454545
SVC	sigmoid	Re-call	0.454545454545
NuSVC	kernel: sigmoid	Re-call	0.454545454545
KNN	minkowski-uniform KNN: 4		4.0
KNN	euclidean-uniform KNN: 4		4.0

KNN	minkowski-uniform KNN: 4		4.0
KNN	euclidean-uniform KNN: 4		4.0
KNN	minkowski-uniform KNN: 4		4.0
KNN	euclidean-uniform KNN: 4		4.0
KNN	minkowski-uniform KNN: 4		4.0
KNN	euclidean-uniform KNN: 4		4.0
SVC	sigmoid	TP	5
NuSVC	kernel: sigmoid		5

Modelos para Sector U

Random Forest	Gini, 50		1.0
Random Forest	Gini, 100		1.0

Random Forest	Gini, 150		1.0
Random Forest	Gini, 200		1.0
Random Forest	Gini, 250		1.0
Random Forest	Gini, 500		1.0
Random Forest	Gini, 750		1.0
Random Forest	Gini, 1000		1.0
Random Forest	Gini, 50		0.5
Random Forest	Gini, 100		0.5
Random Forest	Gini, 150		0.5
Random Forest	Gini, 200	Precision	0.5
Random Forest	Gini, 250		0.5

Random Forest	Gini, 500		0.5
Random Forest	Gini, 750		0.5
Random Forest	Gini, 1000		0.5
Random Forest	Gini, 50		0.5
Random Forest	Gini, 100		0.5
Random Forest	Gini, 150		0.5
Random Forest	Gini, 200	Re-call	0.5
Random Forest	Gini, 250		0.5
Random Forest	Gini, 500		0.5
Random Forest	Gini, 750		0.5
Random Forest	Gini, 1000		0.5

Random Forest	Gini, 50		4.0
Random Forest	Gini, 100		4.0
Random Forest	Gini, 150		4.0
Random Forest	Gini, 200	TN	4.0
Random Forest	Gini, 250		4.0
Random Forest	Gini, 500		4.0
Random Forest	Gini, 750		4.0
Random Forest	Gini, 1000		4.0
Random Forest	Gini, 50		4.0
Random Forest	Gini, 100		4.0
Random Forest	Gini, 150	TP	4.0

Random Forest	Gini, 200		4.0
Random Forest	Gini, 250		4.0
Random Forest	Gini, 500		4.0
Random Forest	Gini, 750		4.0
Random Forest	Gini, 1000		4.0

Modelos para Sector Z

SVC	linear	Accuracy	0.695652173913
SVC	rbf		0.695652173913
SVC	sigmoid		0.695652173913
Naive Bayes	GaussianNB	Precision	0.173913043478
Naive Bayes	GaussianNB	Re-call	0.173913043478

SVC	linear		32.0
SVC	rbf	TN	32.0
SVC	sigmoid		32.0
Naive Bayes	GaussianNB	TP	8

Tabla 10.1: Tabla resumen de modelos de clasificación para divisiones inducidas por sectores de interacción

10.2. Tabla resumen mejores resultados obtenidos para divisiones obtenidas mediante técnica de clustering

Resumen mejores resultados en particiones inducidas por clustering

Algoritmo	División	Accuracy	Recall	Precisión	Grupo	Integrantes
Agglomerative Clustering	Complete euclidean	0,6337018837	0,6069387755	0,8181818182	0	221
Agglomerative Clustering	Complete euclidean	0,772875817	1	0,8221153846	1	35
Dos Divisiones	Promedio	0,7032888503	0,8034693878	0,8201486014		
	Desviación	0,098410832	0,2779362573	0,0027814515		

Algoritmo	División	Accuracy	Recall	Precisión	Grupo	Integrantes

Agglomerative Clustering	Complete euclidean	0,7104166667	0,7571428571	0,8174603175	0	62
Agglomerative Clustering	Complete euclidean	0,772875817	1	0,8221153846	1	35
Agglomerative Clustering	Complete euclidean	0,6041139241	0,6857142857	0,5449657869	2	159
Tres Divisiones	Promedio	0,6958021359	0,8142857143	0,7281804963		
	Desviación	0,0853248645	0,1647508942	0,1586856633		

Algoritmo	División	Accuracy	Recall	Precisión	Grupo	Integrantes
Agglomerative Clustering	Complete euclidean	0,772875817	1	0,8221153846	0	35

Agglomerative Clustering	Complete euclidean	0,6378676471	1	0,75	1	33
Agglomerative Clustering	Complete euclidean	0,6163765823	0,6857142857	0,5507590133	2	159
Agglomerative Clustering	Complete euclidean	0,6547619048	1	0,8571428571	3	29
Cuatro Divisiones	Promedio	0,6704704878	0,9214285714	0,7450043138		
	Desviación	0,0700540431	0,1571428571	0,1369639415		

Algoritmo	División	Accuracy	Recall	Precisión	Grupo	Integrantes
Agglomerative Clustering	ward euclidean	0,7361111111	1	0,7662337662	0	38

Agglomerative Clustering	ward euclidean	0,6166666667	0,6111111111	0,6191176471	1	120
Agglomerative Clustering	ward euclidean	0,730952381	0,5625	0,7777777778	2	41
Agglomerative Clustering	ward euclidean	0,7013888889	0,6875	0,9285714286	3	34
Agglomerative Clustering	ward euclidean	0,7424242424	0,9	1	4	23
Cinco Divisiones	Promedio	0,705508658	0,7522222222	0,8183401239		
	Desviación	0,0520973259	0,1892933782	0,1493328688		

Tabla 10.2. Resumen medidas de desempeño obtenidas en divisiones inducidas por clustering

10.3. Resumen mejores modelos para cada división inducida mediante técnicas de clustering

Modelos para grupo 1			
Modelo	Descripción	Métrica	Valor
auto	KNN con 3 vecinos peso: uniform metrica: minkowski	Accuracy	0.736111111111
auto	KNN con 3 vecinos peso: uniform metrica: euclidean		0.736111111111
auto	KNN con 3 vecinos peso: distance metrica: minkowski		0.736111111111
auto	KNN con 3 vecinos peso: distance metrica: euclidean		0.736111111111
ball_tree	KNN con 3 vecinos peso: uniform metrica: minkowski		0.736111111111
ball_tree	KNN con 3 vecinos peso: uniform metrica: euclidean		0.736111111111
ball_tree	KNN con 3 vecinos peso: distance metrica: minkowski		0.736111111111
ball_tree	KNN con 3 vecinos peso: distance metrica: euclidean		0.736111111111
kd_tree	KNN con 3 vecinos peso: uniform metrica: minkowski		0.736111111111

kd_tree	KNN con 3 vecinos peso: uniform metrica: euclidean	0.736111111111
kd_tree	KNN con 3 vecinos peso: distance metrica: minkowski	0.736111111111
kd_tree	KNN con 3 vecinos peso: distance metrica: euclidean	0.736111111111
brute	KNN con 3 vecinos peso: uniform metrica: minkowski	0.736111111111
brute	KNN con 3 vecinos peso: uniform metrica: euclidean	0.736111111111
brute	KNN con 3 vecinos peso: distance metrica: minkowski	0.736111111111
brute	KNN con 3 vecinos peso: distance metrica: euclidean	0.736111111111
auto	KNN con 3 vecinos peso: uniform metrica: minkowski	0.766233766234
auto	KNN con 3 vecinos peso: uniform metrica: euclidean	0.766233766234
auto	KNN con 3 vecinos peso: distance metrica: minkowski	0.766233766234
auto	KNN con 3 vecinos peso: distance metrica: euclidean	0.766233766234

ball_tree	KNN con 3 vecinos peso: uniform metrica: minkowski	0.766233766234
ball_tree	KNN con 3 vecinos peso: uniform metrica: euclidean	0.766233766234
ball_tree	KNN con 3 vecinos peso: distance metrica: minkowski	0.766233766234
ball_tree	KNN con 3 vecinos peso: distance metrica: euclidean	0.766233766234
kd_tree	KNN con 3 vecinos peso: uniform metrica: minkowski	0.766233766234
kd_tree	KNN con 3 vecinos peso: uniform metrica: euclidean	0.766233766234
kd_tree	KNN con 3 vecinos peso: distance metrica: minkowski	0.766233766234
kd_tree	KNN con 3 vecinos peso: distance metrica: euclidean	0.766233766234
brute	KNN con 3 vecinos peso: uniform metrica: minkowski	0.766233766234
brute	KNN con 3 vecinos peso: uniform metrica: euclidean	0.766233766234
brute	KNN con 3 vecinos peso: distance metrica: minkowski	0.766233766234

brute	KNN con 3 vecinos peso: distance metrica: euclidean		0.766233766234
sigmoid	nuSVC		1
linear	SVC	Recall	1
rbf	SVC		1
sigmoid	SVC		1

Modelos para grupo 2

linear	nuSVC	Accuracy	0.625
gini	RandomForest, n_estimators: 250		0.625
gini	RandomForest, n_estimators: 250	Precision	0.619117647059
SAMME	AdaBoostClassifier con 20 estimadores		0.62962962963
SAMME	AdaBoostClassifier con 10 estimadores		0.574074074074

auto	KNN con 9 vecinos peso: distance metrica: minkowski	0.574074074074
auto	KNN con 9 vecinos peso: distance metrica: euclidean	0.574074074074
ball_tree	KNN con 9 vecinos peso: distance metrica: minkowski	0.574074074074
ball_tree	KNN con 9 vecinos peso: distance metrica: euclidean	0.574074074074
kd_tree	KNN con 9 vecinos peso: distance metrica: minkowski	0.574074074074
kd_tree	KNN con 9 vecinos peso: distance metrica: euclidean	0.574074074074
brute	KNN con 9 vecinos peso: distance metrica: minkowski	0.574074074074
brute	KNN con 9 vecinos peso: distance metrica: euclidean	0.574074074074
gini	DecisionTreeClassifier splitter:random	0.574074074074
entropy	DecisionTreeClassifier splitter:random	0.574074074074
20	GradientBoostingClassifier	0.574074074074

50	GradientBoostingClassifier	0.574074074074
poly	nuSVC	0.574074074074
poly	SVC	0.574074074074

Modelos para grupo 3

SAMME	AdaBoostClassifier con 20 estimadores	0.730952380952
SAMME.R	AdaBoostClassifier con 250 estimadores	0.707142857143
SAMME	AdaBoostClassifier con 10 estimadores	0.705952380952
SAMME.R	AdaBoostClassifier con 200 estimadores	0.683333333333
SAMME.R	AdaBoostClassifier con 500 estimadores	0.683333333333
SAMME.R	AdaBoostClassifier con 100 estimadores	0.682142857143
SAMME	AdaBoostClassifier con 750 estimadores	0.659523809524

SAMME	AdaBoostClassifier con 1000 estimadores	0.659523809524
SAMME.R	AdaBoostClassifier con 750 estimadores	0.659523809524
SAMME.R	AdaBoostClassifier con 1500 estimadores	0.659523809524
auto	KNN con 8 vecinos peso: uniform metrica: minkowski	0.659523809524
auto	KNN con 8 vecinos peso: uniform metrica: euclidean	0.659523809524
ball_tree	KNN con 8 vecinos peso: uniform metrica: minkowski	0.659523809524
ball_tree	KNN con 8 vecinos peso: uniform metrica: euclidean	0.659523809524
kd_tree	KNN con 8 vecinos peso: uniform metrica: minkowski	0.659523809524
kd_tree	KNN con 8 vecinos peso: uniform metrica: euclidean	0.659523809524
brute	KNN con 8 vecinos peso: uniform metrica: minkowski	0.659523809524
brute	KNN con 8 vecinos peso: uniform metrica: euclidean	0.659523809524

gini	RandomForest, n_estimators: 10	Precision	0.659523809524
entropy	RandomForest, n_estimators: 10		0.659523809524
auto	KNN con 7 vecinos peso: uniform metrica: minkowski		0.7777777777778
auto	KNN con 7 vecinos peso: uniform metrica: euclidean		0.7777777777778
ball_tree	KNN con 7 vecinos peso: uniform metrica: minkowski		0.7777777777778
ball_tree	KNN con 7 vecinos peso: uniform metrica: euclidean	Precision	0.7777777777778
kd_tree	KNN con 7 vecinos peso: uniform metrica: minkowski		0.7777777777778
kd_tree	KNN con 7 vecinos peso: uniform metrica: euclidean		0.7777777777778
brute	KNN con 7 vecinos peso: uniform metrica: minkowski		0.7777777777778
brute	KNN con 7 vecinos peso: uniform metrica: euclidean		0.7777777777778
sigmoid	nuSVC	Recall	875

Modelos para grupo 4			
entropy	RandomForest, n_estimators: 10	Accuracy	0.701388888889
entropy	RandomForest, n_estimators: 1000		0.701388888889
entropy	RandomForest, n_estimators: 1000	Precision	0.928571428571
entropy	RandomForest, n_estimators: 10	Recall	0.6875
Modelos para grupo 5			
BernoulliNB	Naive Bayes algoritmo	Accuracy	0.700757575758
SAMME	AdaBoostClassifier con 10 estimadores		0.700757575758
150	GradientBoostingClassifier		0.700757575758
gini	DecisionTreeClassifier splitter:random		0.6969696969697
BernoulliNB	Naive Bayes algoritmo		1.0

SAMME	AdaBoostClassifier con 10 estimadores		0.857142857143
SAMME	AdaBoostClassifier con 20 estimadores		0.833333333333
gini	DecisionTreeClassifier splitter:random		0.833333333333
20	GradientBoostingClassifier		0.833333333333
200	GradientBoostingClassifier		0.833333333333
1500	GradientBoostingClassifier		0.833333333333
sigmoid	nuSVC	Recall	0.9

Tabla 10.3: Resumen medidas de desempeño y modelos para cada división inducida por clustering.

10.4. Resumen mejores modelos obtenidos a partir de particiones inducidas mediante clustering recursivo

Modelos para grupo 1			
Modelo	Descripción	Métrica	Valor
auto	KNN con 3 vecinos peso: uniform metrica: minkowski		0.736111111111
auto	KNN con 3 vecinos peso: uniform metrica: euclidean		0.736111111111
auto	KNN con 3 vecinos peso: distance metrica: minkowski		0.736111111111
auto	KNN con 3 vecinos peso: distance metrica: euclidean		0.736111111111
ball_tree	KNN con 3 vecinos peso: uniform metrica: minkowski		0.736111111111
ball_tree	KNN con 3 vecinos peso: uniform metrica: euclidean		0.736111111111

ball_tree	KNN con 3 vecinos peso: distance metrica: minkowski	0.736111111111
ball_tree	KNN con 3 vecinos peso: distance metrica: euclidean	0.736111111111
kd_tree	KNN con 3 vecinos peso: uniform metrica: minkowski	0.736111111111
kd_tree	KNN con 3 vecinos peso: uniform metrica: euclidean	0.736111111111
kd_tree	KNN con 3 vecinos peso: distance metrica: minkowski	0.736111111111
kd_tree	KNN con 3 vecinos peso: distance metrica: euclidean	0.736111111111
brute	KNN con 3 vecinos peso: uniform metrica: minkowski	0.736111111111
brute	KNN con 3 vecinos peso: uniform metrica: euclidean	0.736111111111

brute	KNN con 3 vecinos peso: distance metrica: minkowski	0.736111111111
brute	KNN con 3 vecinos peso: distance metrica: euclidean	0.736111111111
auto	KNN con 3 vecinos peso: uniform metrica: minkowski	0.766233766234
auto	KNN con 3 vecinos peso: uniform metrica: euclidean	0.766233766234
auto	KNN con 3 vecinos peso: distance metrica: minkowski	0.766233766234
auto	KNN con 3 vecinos peso: distance metrica: euclidean	0.766233766234
ball_tree	KNN con 3 vecinos peso: uniform metrica: minkowski	0.766233766234
ball_tree	KNN con 3 vecinos peso: uniform metrica: euclidean	0.766233766234

ball_tree	KNN con 3 vecinos peso: distance metrica: minkowski	0.766233766234
ball_tree	KNN con 3 vecinos peso: distance metrica: euclidean	0.766233766234
kd_tree	KNN con 3 vecinos peso: uniform metrica: minkowski	0.766233766234
kd_tree	KNN con 3 vecinos peso: uniform metrica: euclidean	0.766233766234
kd_tree	KNN con 3 vecinos peso: distance metrica: minkowski	0.766233766234
kd_tree	KNN con 3 vecinos peso: distance metrica: euclidean	0.766233766234
brute	KNN con 3 vecinos peso: uniform metrica: minkowski	0.766233766234
brute	KNN con 3 vecinos peso: uniform metrica: euclidean	0.766233766234

Modelos para grupo 1				Modelos para grupo 2			
							Accuracy
brute	KNN con 3 vecinos peso: distance metrica: minkowski		0.766233766234				
brute	KNN con 3 vecinos peso: distance metrica: euclidean		0.766233766234				
sigmoid	muSVC		1				
linear	SVC		1	Recall			
rbf	SVC		1				
sigmoid	SVC		1				
linear	muSVC		0.625				

gini	RandomForest, n_estimators: 250		0.625
gini	RandomForest, n_estimators: 250	Precision	0.619117647059
SAMME	AdaBoostClassifier con 20 estimadores		0.62962962963
SAMME	AdaBoostClassifier con 10 estimadores		0.574074074074
auto	KNN con 9 vecinos peso: distance metrica: minkowski		0.574074074074
auto	KNN con 9 vecinos peso: distance metrica: euclidean		0.574074074074
ball_tree	KNN con 9 vecinos peso: distance metrica: minkowski		0.574074074074
ball_tree	KNN con 9 vecinos peso: distance metrica: euclidean		0.574074074074

kd_tree	KNN con 9 vecinos peso: distance metrica: minkowski	0.574074074074
kd_tree	KNN con 9 vecinos peso: distance metrica: euclidean	0.574074074074
brute	KNN con 9 vecinos peso: distance metrica: minkowski	0.574074074074
brute	KNN con 9 vecinos peso: distance metrica: euclidean	0.574074074074
gini	DecisionTreeClassifier splitter:random	0.574074074074
entropy	DecisionTreeClassifier splitter:random	0.574074074074
20	GradientBoostingClassifier	0.574074074074
50	GradientBoostingClassifier	0.574074074074

poly	nuSVC		0.574074074074
poly	SVC		0.574074074074
Modelos para grupo 3			
SAMME	AdaBoostClassifier con 20 estimadores		0.730952380952
SAMME.R	AdaBoostClassifier con 250 estimadores		0.707142857143
SAMME	AdaBoostClassifier con 10 estimadores		0.705952380952
SAMME.R	AdaBoostClassifier con 200 estimadores		0.683333333333
SAMME.R	AdaBoostClassifier con 500 estimadores		0.683333333333

SAMME.R	AdaBoostClassifier con 100 estimadores	0.682142857143
SAMME	AdaBoostClassifier con 750 estimadores	0.659523809524
SAMME	AdaBoostClassifier con 1000 estimadores	0.659523809524
SAMME.R	AdaBoostClassifier con 750 estimadores	0.659523809524
SAMME.R	AdaBoostClassifier con 1500 estimadores	0.659523809524
auto	KNN con 8 vecinos peso: uniform metrica: minkowski	0.659523809524
auto	KNN con 8 vecinos peso: uniform metrica: euclidean	0.659523809524
ball_tree	KNN con 8 vecinos peso: uniform metrica: minkowski	0.659523809524

ball_tree	KNN con 8 vecinos peso: uniform metrica: euclidean	0.659523809524
kd_tree	KNN con 8 vecinos peso: uniform metrica: minkowski	0.659523809524
kd_tree	KNN con 8 vecinos peso: uniform metrica: euclidean	0.659523809524
brute	KNN con 8 vecinos peso: uniform metrica: minkowski	0.659523809524
brute	KNN con 8 vecinos peso: uniform metrica: euclidean	0.659523809524
gini	RandomForest, n_estimators: 10	0.659523809524
entropy	RandomForest, n_estimators: 10	0.659523809524
auto	KNN con 7 vecinos peso: uniform metrica: minkowski	0.777777777778

auto	KNN con 7 vecinos peso: uniform metrica: euclidean	0.77777777777778
ball_tree	KNN con 7 vecinos peso: uniform metrica: minkowski	0.77777777777778
ball_tree	KNN con 7 vecinos peso: uniform metrica: euclidean	0.77777777777778
kd_tree	KNN con 7 vecinos peso: uniform metrica: minkowski	0.77777777777778
kd_tree	KNN con 7 vecinos peso: uniform metrica: euclidean	0.77777777777778
brute	KNN con 7 vecinos peso: uniform metrica: minkowski	0.77777777777778
brute	KNN con 7 vecinos peso: uniform metrica: euclidean	0.77777777777778
sigmoid	muSVC	Recall 875

Modelos para grupo 4			
entropy	RandomForest, n_estimators: 10	Accuracy	0.7013888888889
entropy	RandomForest, n_estimators: 1000		0.7013888888889
entropy	RandomForest, n_estimators: 1000	Precision	0.928571428571
entropy	RandomForest, n_estimators: 10	Recall	0.6875
Modelos para grupo 5			
BernoulliNB	Naive Bayes algoritmo		0.700757575758
SAMME	AdaBoostClassifier con 10 estimadores	Accuracy	0.700757575758

			0.700757575758
			0.69696969697
			1.0
			0.857142857143
		Precision	0.833333333333
			0.833333333333
			0.833333333333
			0.833333333333
150	GradientBoostingClassifier		
gini	DecisionTreeClassifier splitter:random		
BernoulliNB	Naive Bayes algoritmo		
SAMME	AdaBoostClassifier con 10 estimadores		
SAMME	AdaBoostClassifier con 20 estimadores		
gini	DecisionTreeClassifier splitter:random		
20	GradientBoostingClassifier		
200	GradientBoostingClassifier		

1500	GradientBoostingClassifier		0.833333333333333
sigmoid	nuSVC	Recall	0.9

Tabla 10.4: Resumen mejores modelos obtenidos para divisiones inducidas por clustering recursivo.