UNIVERSIDAD TECNOLÓGICA METROPOLITANA
del Estado de Chile

Facultad de Ingeniería
Departamento de Informática y Computación
Minería de Datos

Modelo Predictivo de Enfermedades Coronarias

Integrantes:
Alexandra Olivares S.
Daniela Galleguillos D.
Profesor:
Pablo Figueroa

Minería de datos - 1° Semestre 2020

Índice

1.- Introducción

Todos saben en que puede derivar una enfermedad coronaria, pero no muchos saben cuales son los parámetros para diagnosticar una enfermedad coronaria (bueno aparte de los doctores).

Este proyecto trata de identificar enfermedades coronarias según la información recolectada de pacientes que ingresan a la UCI, parámetros como tipos de dolor de pecho, presión sanguínea, colesterol, resultados de electrocardiograma y otros 10 parámetros evaluados.

El dataset contenía originalmente 76 atributos, pero la limpieza que se le hizo la redujo a 14. Donde los datos que se tienen son provenientes de un hospital de Cleveland.

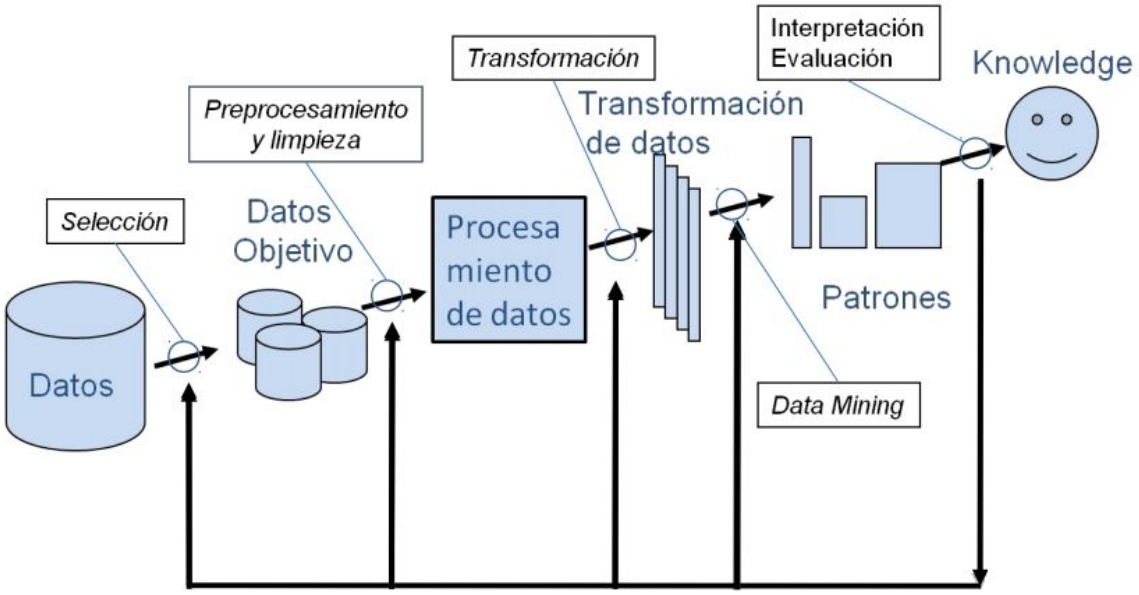
2.- Marco teórico

A continuación se definirán ciertos conceptos que son importantes en el desarrollo del proyecto, de estos se dará paso a la elaboración del proyecto.

2.1.- Metodologías

2.1.1.- Metodologías de Minería de datos - KDD

"Knowledge Discovery in Databases" es una metodología propuesta por Fayyad en 1996, que consta de 5 fases: Selección, preprocesamiento, transformación, minería de datos y evaluación e implementación. Es un proceso iterativo e interactivo.



Los pasos de "Knowledge Discovery in Databases" son :

1.- Desarrollar un entendimiento de la aplicación de dominio y los conocimientos previos y la identificación de la meta del proceso de kdd desde el punto de vista del cliente.

2.- Crear un conjunto de datos objetivo: la selección de un conjunto de datos, o que se centren en un subconjunto de variables o datos de muestras, el descubrimiento que se llevará a cabo.

3.- Limpieza y preprocesamiento de datos. Operaciones básicas incluyen la eliminación de ruido, campos de datos vacíos, etc.

4.- Reducción de datos y la proyección: la búsqueda de características útiles para representar los datos en función del objetivo de la tarea. (reducción de dimensionalidad)
5.- Colocar el objetivo del KDD (paso 1) a un método de minería de datos para luego

6.-el análisis exploratorio y de hipótesis y el modelo de selección: la elección del algoritmo de minería de datos que se utilizará para la búsqueda de patrones de datos

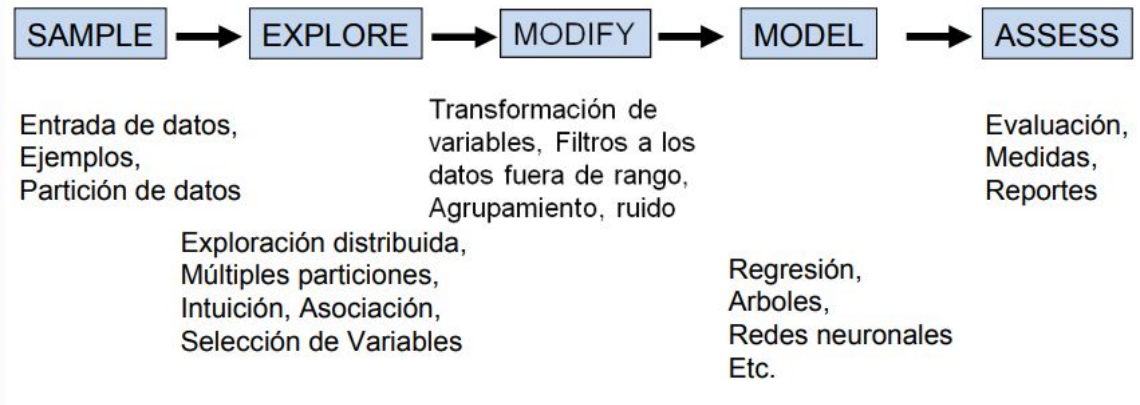
7.-. Séptimo es la minería de datos: la búsqueda de patrones de interés en una determinada forma de representación o de un conjunto de tales representaciones.

8.- Interpretación de los patrones minados, posiblemente se puede regresar a cualquiera de los pasos 1 a 7 para más iteración. Este paso puede implicar también la visualización de los patrones y modelos extraídos o visualización de los datos que figuran extraído modelos

9.- Está actuando sobre el conocimiento descubierto: el uso del conocimiento directamente, incorporado el conocimiento en otro sistema para la adopción de nuevas medidas o, simplemente, documentación y presentación de informes a las partes interesadas. Este proceso también incluye la comprobación y la solución de posibles conflictos con creían (o extrae) los conocimientos.

2.1.2.- Metodologías de Minerías de Datos - SEMMA

SEMMA es el acrónimo a las cinco fases: (Sample, Explore, Modify, Model, Assess) La metodología es propuesta por SAS Institute Inc, la define como: "... proceso de selección, exploración y modelamiento de grandes cantidades de datos para descubrir patrones de negocios desconocidos…"



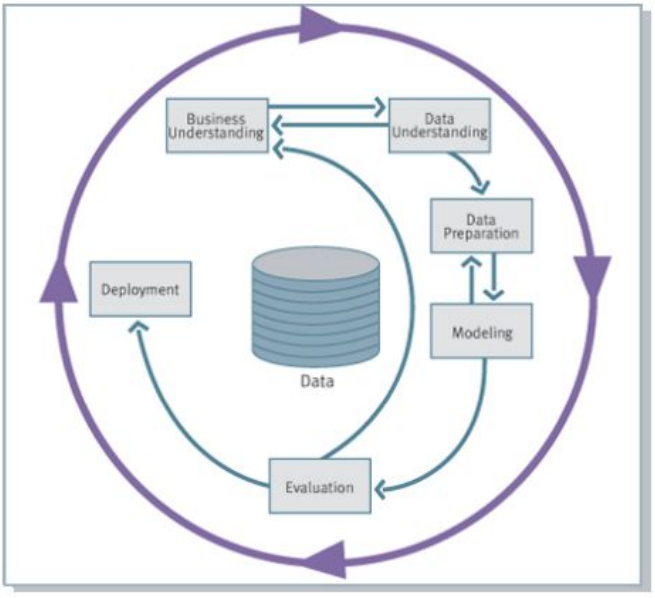
2.1.3.- Metodologías de Minería de datos - CRISP-DM

Cross-Industry Standard Process for Data Mining (CRISP-DM) es una iniciativa financiada por la Comunidad Europea se ha unido para desarrollar una plataforma para Minería de Datos. Los objetivos de esa iniciativa son:

-Fomentar la interoperabilidad de las herramientas a través de todo el proceso de minería de datos.

-Eliminar la experiencia misteriosa y costosa de las tareas simples de minería de datos.

La plataforma está creada para almacenar la experiencia para que los proyectos sean replicados, y ayuda a la planeación y gerencia del proyecto, también tiene este "factor de comodidad" para los nuevos usuarios, y esto demuestra la madurez de la minería de datos y reduce la dependencia en "estrellas"



Las fases de CRISP-DM:

1. Comprensión del negocio:
    - Entendimiento de los objetivos y requerimientos del proyecto.
    - Definición del problema de Minería de Datos
2. Comprensión de los datos
    - Obtención del conjunto inicial de datos.
    - Exploración del conjunto de datos.
    - Identificar las características de calidad de los datos
    - Identificar los resultados iniciales obvios.

3. Preparación de Datos
    Selección de datos
    Limpieza de datos
4. Modelamiento
    Implementación en herramientas de Minería de Datos
5. Evaluación
    Determinar si los resultados coinciden con los objetivos del negocio
    Identificar las temas de negocio que deberían haberse abordado
6. Despliegue
    Instalar los modelos resultantes en la práctica
    Configuración para minería de datos de forma repetida o continua

2.2.- Modelos

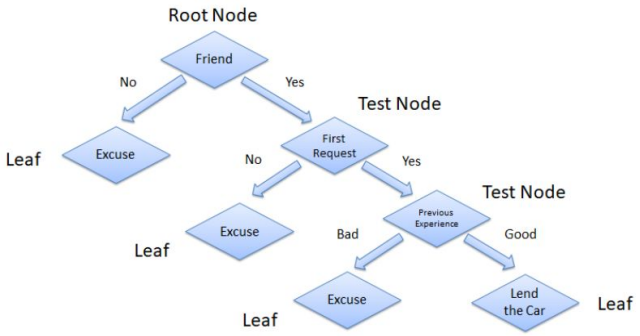
2.2.1.- Aprendizaje supervisado

Simon Haykin(2008) ve al lenguaje supervisado como un lenguaje con un maestro que le supervisa, el lo explica como "En términos conceptuales, podemos pensar que el profesor tiene conocimiento del entorno, y que ese conocimiento está representado por un conjunto de ejemplos de entrada y salida, pero el entorno es desconocido para la red neuronal. Supongamos ahora que el profesor y la red neuronal están ambos expuestos a un vector de formación (es decir, un ejemplo) extraído del mismo entorno. En virtud de los conocimientos incorporados, el profesor es capaz de proporcionar a la red neuronal una respuesta deseada para ese vector de entrenamiento. En efecto, la respuesta deseada representa la acción "óptima" que debe realizar la red neuronal. Los parámetros de la red se ajustan bajo la influencia combinada del vector de formación y la señal de error. El error se define como la diferencia entre la respuesta deseada y la respuesta real de la red. Este ajuste se lleva a cabo de forma iterativa en un paso a paso con el objetivo de hacer que la red neural emulara al maestro; se presume que la emulación es óptima en algún sentido estadístico. De esta manera, el conocimiento del entorno disponible para el profesor se transfiere a la red mediante el entrenamiento y almacenados en forma de pesos sinápticos "fijos", que representan la memoria a largo plazo. Cuando se alcanza esta condición, podemos entonces dispensar con el profesor y dejar que la red neural se ocupe completamente del entorno por sí mismo."

El vector de entrenamiento nombrado en la explicación y el maestro corresponden a lo que conocemos como data de entrenamiento, esta es una muestra de datos etiquetados con su resultado esperado (etiquetas), los resultados guían a la red como si fuera un profesor enseñando el camino óptimo, para que esta se adapte y se ajuste para emular los resultados esperados.

2.2.1.1.- Árboles de decisión

Los creadores de la metodología de los árboles de decisiones fueron Leo Breiman, Jerome Friedman, Richard Olshen y Charles Stone ,esta metodología suele ser representada de manera gráfica como se observa en la siguiente ilustración



Estos árboles funcionan a través del descarte con una serie de preguntas encuentran el grupo de datos adecuados para hacer una predicción.
Pedro Almargo-Blanco y Fernando Sancho-Caparrini (2017) definen a los árboles de decisión como “un modelo de clasificacion (y regresión) que, a partir de las caracterısticas de un objeto dado, y aplicando una serie de reglas, es capaz de asociar una clase a dicho objeto (o un valor continuo en el caso de regresión).”, estas características que nos dicen el árbol de decisión las identifica dentro de los datos ingresados con etiquetas, o así llamada data de prueba.

2.2.1.2.- Random Forest

El método de random forest es una modificación del proceso de bagging que consigue mejores resultados gracias a que correlaciona los árboles generados en el proceso.

Supongase un set de datos en el que hay un predictor muy influyente junto con otros moderadamente influyentes. En este escenario, todos o casi todos los árboles creados en el proceso de bagging estarán dominados por el mismo predictor y serán muy parecidos entre ellos. Como consecuencia de la alta correlación entre los árboles, el proceso de bagging apenas conseguirá disminuir la varianza y, por lo tanto, tampoco mejorar el modelo. Random forest evita este problema haciendo una selección aleatoria de m predictores antes de evaluar cada división. De esta forma, un promedio de $(p-m)/p$ divisiones no contemplarán el predictor influyente, permitiendo que otros predictores puedan ser seleccionados. Solo con añadir este paso extra se consigue decorrelacionar los árboles, por lo que su agregación consigue una mayir reducción de la varianza.

2.2.1.3.- AdaBoost

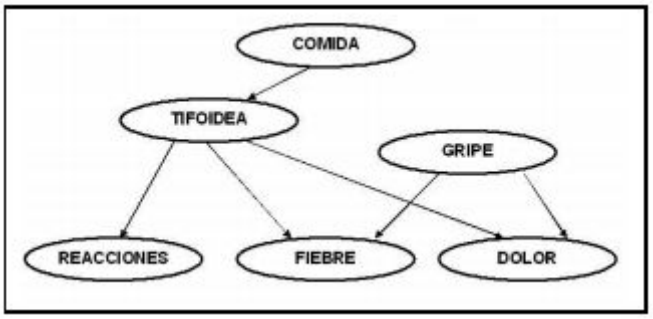
La publicación en 1999 del algoritmo AdaBoost (*Adaptative Boosting)* por parte de Yoav Freund y Robert Schapire supuso un avance muy importante en el campo del aprendizaje estadístico, ya que hizo posible aplicar la estrategia de boosting a multitud de problemas. A continuación, se introduce AdaBoost con un problema de clasificación binaria.

Para el funcionamiento de AdaBoost es necesario establecer:

- Un tipo de modelo, normalmente llamado weak learner o base learner, que sea capaz de predecir la variable respuesta con un porcentaje de acierto ligeramente superior a lo esperado por azar. En el caso de los árboles de regresión, este weak learner suele ser un árbol con apenas unos pocos nodos.
- Codificar las dos clases de la variable respuesta como +1 y -1.
- Un peso inicial e igual para todas las observaciones que forman el set de entrenamiento.

2.2.1.4.- Redes Bayesianas

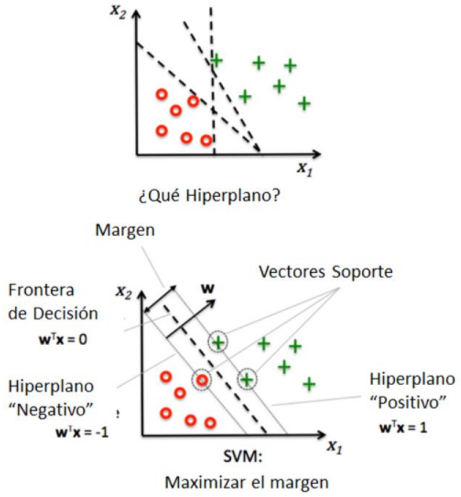
Las redes bayesianas son una representación gráfica de dependencias para razonamiento probabilístico, en la cual los nodos representan variables aleatorias y los arcos representan relaciones de dependencia directa entre las variables. La Figura muestra un ejemplo hipotético de una red bayesiana (RB) que representa cierto conocimiento sobre medicina. En este caso, los nodos representan enfermedades, síntomas y factores que causan algunas enfermedades. La variable a la que apunta un arco es dependiente de la que está en el origen de éste, por ejemplo fiebre depende de tifoidea y gripe en la red de la Figura. La topología o estructura de la red nos da información sobre las dependencias probabilísticas entre las variables. La red también representa las independencias condicionales de una variable (o conjunto de variables) dada(s) otra(s) variable(s). Por ejemplo, en la red de la Figura, las reacciones son cond. indep. de C, G, F, D dado tifoidea. (Donde: C es comida, T es tifoidea, G es gripe, R es reacciones, F es fiebre y D es Dolor).



Ejemplo de una red bayesiana. Los nodos representan variables aleatorias y los arcos relaciones de dependencia

2.2.1.5.- SVM

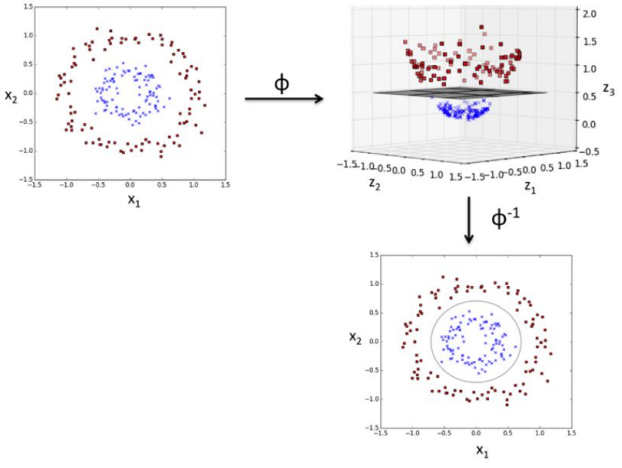
Las Máquinas Vectoriales de Soporte o su acrónimo en inglés SVM fueron desarrolladas entre el 1962-1964 por V.N.Vapnik y su equipo del Instituto de Ciencias de Control de la Academia Rusa de Ciencias, Moscú, Rusia, el cual tenía como objetivo " Mapear los vectores de entrada x en un espacio de características de alta dimensión Z a través de un mapeo no lineal, elegido a priori. En este espacio, se construye un hiperplano de separación óptimo ".



Como se observa en la Ilustración para establecer los márgenes máximos, se añaden dos rectas paralelas(hiperplano positivo y negativo) y se intenta maximizar sus distancias a la línea de decisión original o la frontera de decisión.

2.2.1.6.- SVM Polinomial

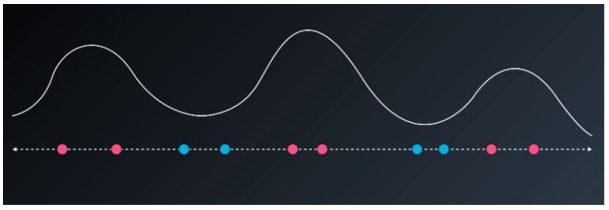
La idea básica del Kernel, cuando tratamos con combinaciones no lineales, es proyectarlas en un espacio con más dimensiones vías una función de correspondencia ϕ, de forma que los datos sean linealmente separables, una vez separados, este vuelve a su forma original, como se puede observar en la siguiente Ilustración.

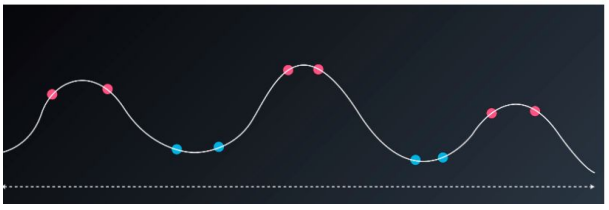


2.2.1.7.- SVM Gaussian

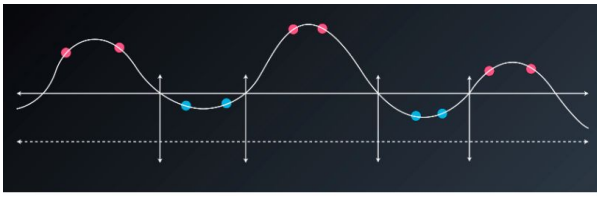
El núcleo RBF es una función cuyo valor depende de la distancia del origen o de algún punto. Es conocido como Kernel Gaussiano y tiene matemáticamente el siguiente formato: $k(x_1, x_2) = e^{(-\gamma||x_1 - x_2||^2)}$, siendo $\gamma > 0$.

El funcionamiento del Kernel RBF, es más entendible de forma gráfica, como se observa en las siguientes Ilustraciones. En la parte superior de la primera se identifica como se encuentran los datos se maneta inicial, es decir de forma lineal, sin embargo la curvas generadas por el Kernel concuerdan en la diferencias de clases, es decir las montañas conforman una clase (puntos rojos) y los valles conforman otra (puntos azules), En la parte inferior de la primera imagen se observa como la data es cambiada para que coincida con el análisis planteado.





En la siguiente ilustración se comprueba que estas clases pueden ser separadas por este sistema de valles y montañas, para luego al inferior de la imagen pasar a su formato original, es decir, lineal pero con la separación ya realizada



Parámetro Gamma(γ), Se utiliza para modificar la anchura de las montañas y valles, si γ es grande se estrecharán las montañas y valles que tiendan al sobre ajuste, mientras que si γ es pequeña cuando tiendan al sobreajuste las montañas y valles se ensanchan.

2.2.1.8.- SVM Sigmoid

El núcleo hiperbólico tangente también se conoce como núcleo sigmoide y como núcleo de perceptrón multicapa (MLP). El Núcleo Sigmoide proviene del campo de las Redes Neuronales, donde la función sigmoide bipolar se utiliza a menudo como una función de activación para las neuronas artificiales.

$$k(x,\ y)\ =\ tanh\ (alfa\ \hat{x}T\ y\ +\ c)$$

Es interesante observar que un modelo SVM que utiliza una función de núcleo sigmoide equivale a una red neuronal de dos capas, perceptron. Este núcleo fue muy popular para las máquinas de vectores de apoyo debido a su origen en la teoría de redes neuronales. Además, a pesar de ser sólo condicionalmente positivo definitivo, se ha encontrado que funciona bien en la práctica.

Hay dos parámetros ajustables en el núcleo sigmoide, la pendiente alfa y la constante de intercepción c. Un valor común para alfa es 1/N, donde N es la dimensión de los datos. Un estudio más detallado sobre los núcleos sigmoides se puede encontrar en los trabajos de Hsuan-Tien y Chih-Jen.

2.2.2.- Aprendizaje no supervisado

Juan Zambrano(2018) nos resalta"A diferencia del aprendizaje supervisado, en el no supervisado solo se le otorgan las características, sin proporcionarle al algoritmo ninguna etiqueta. Su función es la agrupación, por lo que el algoritmo debería catalogar por similitud y poder crear grupos, sin tener la capacidad de definir cómo es cada individualidad de cada uno de los integrantes del grupo." A lo que hace referencia es a que el algoritmo logra identificar dentro de la data patrones que la diferencia para separarla en grupos, sin necesidad de una guía. Algunos de sus algoritmos más destacados incluyen clasificación o clustering, K-mean y reglas de asociación.

2.2.2.1.- K-means

El algoritmo K-Means fue propuesto por MacQueen, J. (1967) , quien lo define com como "un proceso de partición de un Población N-dimensional en conjuntos k sobre la base de una muestra" y lo explica como "el proceso otorga particiones que son razonablemente eficientes en el sentido de la varianza dentro de una clase. Es decir, si p es la función de masa de probabilidad para la población, $S\ =\ \{S_1, S_2,\ ...,\ S_k\}$ es una partición de E_N, y $u_i,\ i=\ 1,\ 2,\ ...,\ k$, es la media condición de p sobre el conjunto S_i, entonces

$w^2(S) = \sum_{i=1}^{K} f S_i \left| z - u_i \right|^2 dp(z)$ tiende a ser baja para las particiones S generadas por el método. Decimos "tiende a ser baja", principalmente por consideraciones intuitivas, corroboradas en cierta medida por el análisis matemático y la experiencia práctica en computación. ”

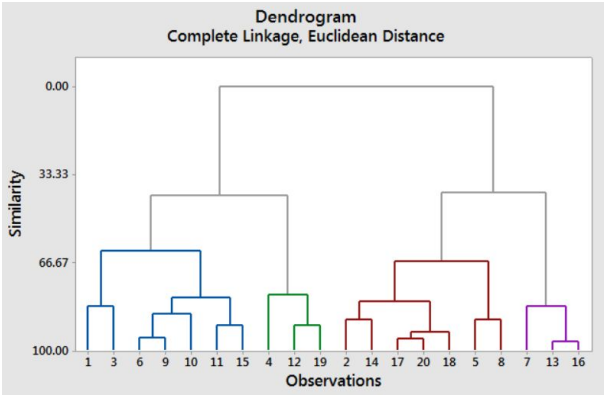
En el algoritmo de K-medias, se debe ingresar el número K(número de cluster), para encontrar este número se utilizan distintos métodos como, Conocimiento de campo, Decisión de negocios o el método del codo.

2.2.2.2.- Agrupación jerárquica

Según la Enciclopedia cubana. (2012) “Los algoritmos jerárquicos producen una secuencia anidada de particiones del conjunto de objetos, es decir, los grupos se organizan de forma jerárquica y cada grupo (cluster) puede verse como la unión de otros grupos (clusters), obteniendo así distintos niveles de jerarquía de grupos. Esta organización jerárquica es representada tradicionalmente por un árbol llamado dendrograma, el cual proporciona una taxonomía o índice jerárquico de la información procesada.”

Los dendrogramas son gráficos en los que se pueden observar los grados de similitud de sus elementos a través de sus distancias como se puede observar según el gráfico, se fusionan en la parte inferior son similares, mientras que las que están en la parte superior son muy diferentes. Con los dendrogramas, las conclusiones se hacen basándose en la ubicación del eje vertical y no en el horizontal.

Analizando la ilustración a continuación podemos observar 4 grandes grupos en diferentes colores, sin embargo dentro de estos grupos pueden identificar elementos que son más similares a otros, por ejemplo, los elementos 13 y 16 al igual que 17 y 20 son muy similares, mientras que 11 y 15, 5 y 8 son similares pero no tanto como los grupos anteriores.



A pesar que la separación pre-hecha en la ilustración corresponde a 4 grupos, este se puede separar en 2 grandes grupos o en 6 grupos o hasta 8 grupos pequeños según su grado de relación.

2.3.- Herramientas

2.3.1.- Jupyter

Jupyter permite desarrollar softwares de código abierto, estándares abiertos y servicios de computación interactiva en docena de lenguajes de programación.

2.3.2.- Anaconda

Es el juego de herramientas que lo equipa para trabajar con miles de paquetes y bibliotecas de código abierto.

2.3.3.- Colab

Colab es un entorno interactivo denominado notebook de Colab, que permite escribir y ejecutar código.

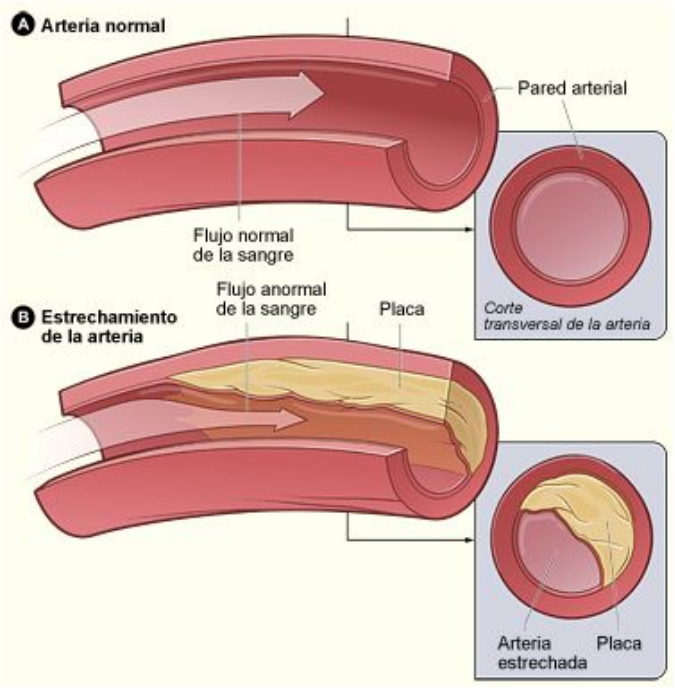
2.3.4.- Python

Python corresponde a un lenguaje de programación que le permite trabajar rápidamente e integrar sistemas de manera más efectiva, este lenguaje cuenta con varias librerías que facilitan el análisis de los datos entre esta se encuentran:

- Panda (Python Data Analysis Library) la librería que permite el manejo de datos con el formato de tablas, la cual se usa de manera esencial en la limpieza y manejo de datos
- Panda Profiling, librería la cual realiza un análisis exploratorio para analizar los datos rápidamente.
- Sklearn es un herramientas simples y eficientes para el análisis predictivo de datos construida sobre NumPy, SciPy y matplotlib, y de código abierto

3.- Comprensión del negocio

La enfermedad coronaria, conocida también como enfermedad de las arterias coronarias, es una afección en la que la placa se deposita dentro de las arterias coronarias. Estas arterias suministran sangre rica en oxígeno al músculo cardíaco, que es el músculo del corazón.
La placa está formada por grasa, colesterol, calcio y otras sustancias que se encuentran en la sangre. Cuando la placa se deposita en las arterias produce una enfermedad llamada aterosclerosis. El depósito de placa se produce en el transcurso de muchos años.



La figura A muestra una arteria normal con flujo normal de sangre. La ilustración del recuadro muestra un corte transversal de una arteria normal. La figura B muestra una arteria con depósito de placa. La ilustración del recuadro muestra un corte transversal de la arteria que se ha vuelto más estrecha por la acumulación de placa.

Con el tiempo, la placa endurece y estrecha las arterias coronarias, con lo cual se limita el flujo de sangre rica en oxígeno que llega al músculo cardíaco.
A la larga, una parte de la placa puede romperse. Al hacerlo, se puede formar un coágulo de sangre en la superficie de la placa. Si el coágulo crece lo suficiente, puede bloquear en su mayor parte o en su totalidad el flujo de sangre que pasa por la arteria coronaria.

3.1.- Definición del problema

El diagnóstico oportuno de una enfermedad coronaria, daría tiempo valioso a los profesionales de la salud para poder salvar vidas, por lo que el modelo predictivo de enfermedades coronarias pretende dar ese “tiempo” para alcanzar a diagnosticar a tiempo y que se puedan tratar los pacientes.

Nos encontramos con una base de datos con una serie de variables de personas con problemas cardíacos, esta data está creada con el objetivo de encontrar una forma rápida de identificar si la persona en cuestión tiene problemas cardíacos o no.

3.2.- Solución propuesta

Implementar diferentes formas de IA, con el objetivo de encontrar la mejor forma de identificar a los pacientes con enfermedades cardiacas.

3.3.- Objetivos

3.3.1.- Objetivos generales

Encontrar la mejor forma de identificar a los pacientes con enfermedades cardiacas a través de IA.

3.3.2.- Objetivos específicos

-Identificar los modelos que puedan predecir un enfermo coronario.

-Comparar el comportamiento de los distintos modelos que caracterizan enfermedades coronarias.

-Elaborar un modelo predictivo que caracterice enfermedades coronarias de un paciente.

-Evaluar y realizar pruebas al sistema, identificar sus falencias y posibles mejoras.

3.4.- Hipótesis

Los algoritmos de aprendizaje supervisado serán más eficientes a la hora de identificar a los pacientes con enfermedades cardiacas

4.- Comprensión de los datos

4.1.- Descripción de base de datos

La base de datos en cuestión contiene datos estructurados, dado que contienen un formato y campos fijos. Se puede observar que la data está claramente organizada y sin ruido por lo que no necesita una limpieza exhaustiva.

4.2.- Descripción de variables

4.2.1.- Input Candidato

1. (edad)
2. (sexo) 1 = hombre y 0 = mujer
3. (dolorP) tipos de dolor de pecho
4. (Presion) presión sanguínea en reposo
5. (Colesterol) colesterol sérico en mg/dl
6. (azucar) azúcar en la sangre en ayunas > 120 mg/dl(1 = verdadero; 0 = falso)
7. (electro) resultados electrocardiográficos en reposo (valores 0,1,2)
8. (maximaFc) frecuencia cardíaca máxima alcanzada
9. (angina) angina inducida por el ejercicio
10. (depresionST) depresión ST inducida por el ejercicio en relación con el descanso (1 = si; 0 = no)
11. (pendienteST) la pendiente del segmento ST del ejercicio máximo
12. (busques) número de buques principales (0-3) coloreados por la flotación
13. (thal) 3 = normal; 6 = defecto fijo; 7 = defecto reversible

4.2.2.- Target Candidatos

Esta base de datos se encuentra un apartado llamado target, la cual se encuentra 0 y 1 que representan el estado de la enfermedad angiográfica, en los cuales 0(menor al 50% reducción de diámetro) indica que no presenta enfermedad y 1(mayor al 50% reducción de diámetro) que si presenta enfermedad

5.- Preparación de los datos

5.1.- Tratamiento de valores perdidos

Se aplicaría el comando para panda df.dropna(), donde df corresponde a la base de datos tratada para eliminar los datos nan o perdidos junto a la fila donde se encuentre ese dato.

	edad	sexo	dolorP	presion	colesterol	azucar	electro	maximaFc	angina	depresionST	pendienteST	busques	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

303 rows × 14 columns

```
#Eliminar nan o valores predidos
corazon.dropna()
corazon
```

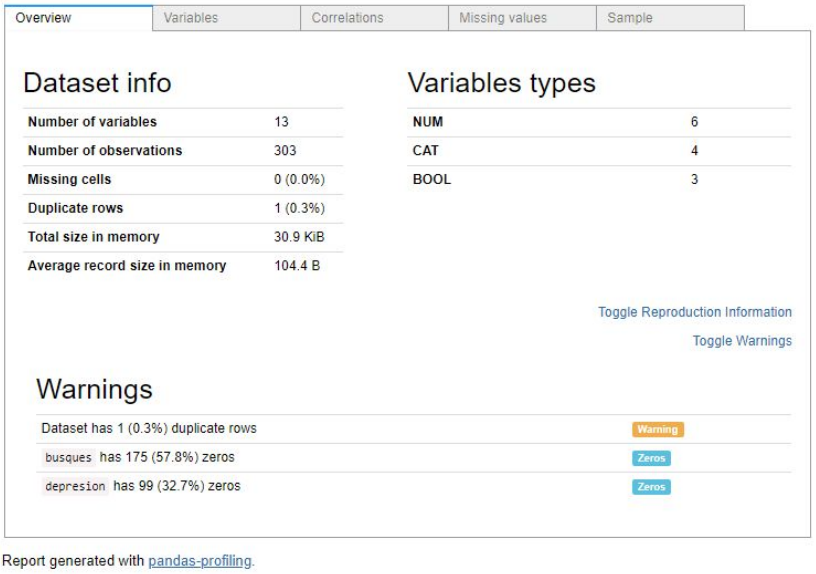
	edad	sexo	dolorP	presion	colesterol	azucar	electro	maximaFc	angina	depresionST	pendienteST	busques	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

303 rows × 14 columns

Como se observa el la imagen la data no cuenta con datos perdidos(nan)

5.2.- Análisis descriptivo de los datos

Una vez la data separada del dato target se analiza con pandas profiling, el cual da una vista detallada y amplia de los datos.



Overview | Variables | Correlations | Missing values | Sample

Dataset info

Number of variables	13
Number of observations	303
Missing cells	0 (0.0%)
Duplicate rows	1 (0.3%)
Total size in memory	30.9 KiB
Average record size in memory	104.4 B

Variables types

NUM	6
CAT	4
BOOL	3

Toggle Reproduction Information

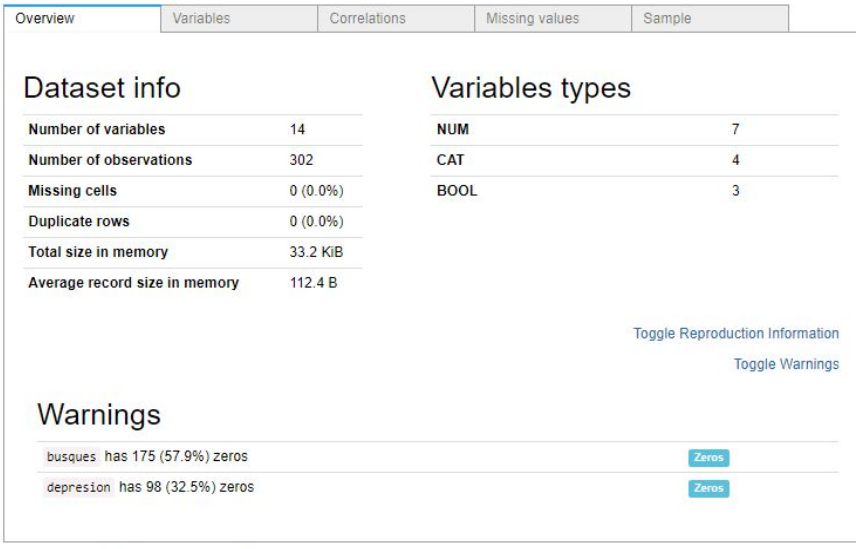
Toggle Warnings

Warnings

Dataset has 1 (0.3%) duplicate rows	Warning
busques has 175 (57.8%) zeros	Zeros
depresion has 99 (32.7%) zeros	Zeros

Report generated with pandas-profiling.

Principalmente se puede observar que en la base de datos tenemos algunos datos repetidos por los cual serán eliminados y realizó el análisis nuevamente (se eliminan los duplicados antes de separar los datos en data y target para que queden con los mismos índices).



Overview | Variables | Correlations | Missing values | Sample

Dataset info

Number of variables	14
Number of observations	302
Missing cells	0 (0.0%)
Duplicate rows	0 (0.0%)
Total size in memory	33.2 KiB
Average record size in memory	112.4 B

Variables types

NUM	7
CAT	4
BOOL	3

Toggle Reproduction Information

Toggle Warnings

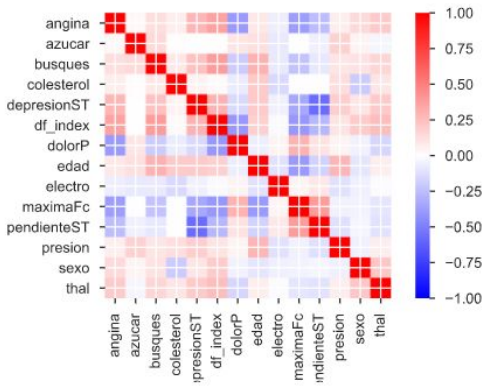
Warnings

busques has 175 (57.9%) zeros	Zeros
depresion has 98 (32.5%) zeros	Zeros

Report generated with pandas-profiling.

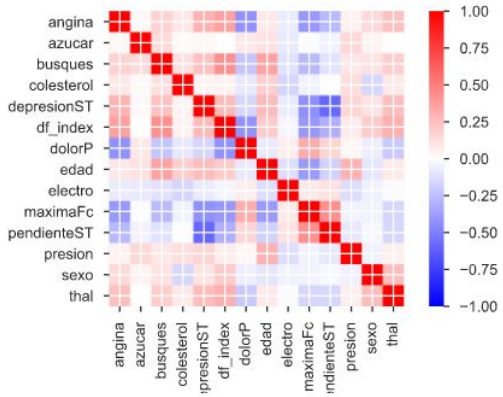
Podemos observar cómo este reporte nos muestra las correlaciones de las variables de distintas maneras.

Pearson's



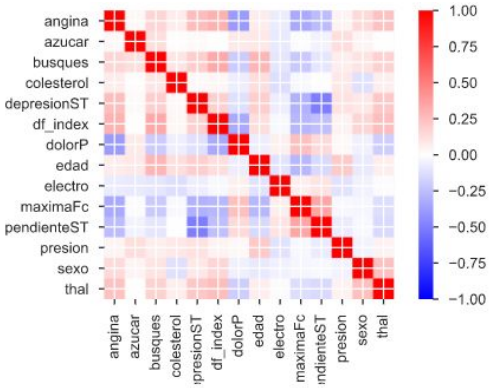
Como se puede observar que no existe una dependencia positiva o negativa total de las variables entre sí, de hecho en muchas se puede observar que no existe ninguna relación lineal.

Spearman's p



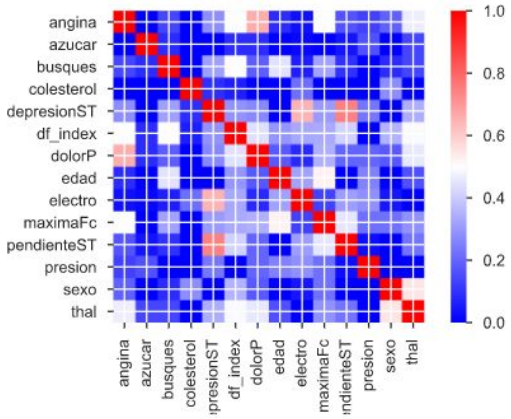
Al igual que en la correlación de Pearson's se puede observar que no existe una dependencia positiva o negativa total de las variables entre sí, de hecho en muchas se puede observar que no existe ninguna relación lineal.

Kendall's T



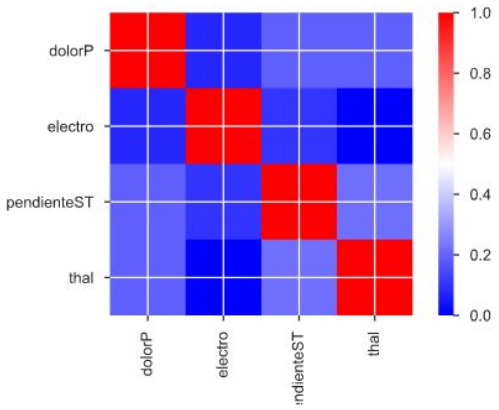
Confirmando los índices anteriores el índice Kendall's nos dice que las variables en su mayoría son independientes el uno del otro y solo unas pocas tienden a una leve dependencia.

Phik



El índice Phi nos indica que en la mayoría de las variables no contienen relación entre sí y algunas cuentan con una pequeña relación, ninguna variable contiene una relación perfecta.

Cramer's V



El índice Cramer's nos indica que no existe relación entre las variables.

5.3.- Transformación de variables

Al contar con la data ordenada no se transforma, como podemos observar todas las variables cualitativas están ya han sido cambiadas a binario para su fácil utilización.
Por ejemplo
sexo → 1 = hombre y 0 = mujer
depresión ST inducida por el ejercicio en relación con el descanso → 1 = si y 0 = no
resultados electrocardiográficos en reposo → valores 0,1,2
azúcar en la sangre en ayunas > 120 mg/dl → 1 = verdadero y 0 = falso
thal: 3 = normal; 6 = defecto fijo; 7 = defecto reversible

5.4.- Agrupamiento de variables

Dado que las variables son independientes unas de otras no se ha efectuado una agrupación de variables y se ha trabajado con las variables como son dadas de la base de datos.

5.5.- Análisis Factorial

Para el análisis factorial analizamos los datos para identificar si el análisis factorial es factible.

```
chi_square_value, p_value = calculate_bartlett_sphericity(corazon)
chi_square_value, p_value
```

```
(562.4278578491244, 8.20713233355246e-75)
```

```
kmo_all, kmo_model = calculate_kmo(corazon)
kmo_model
```

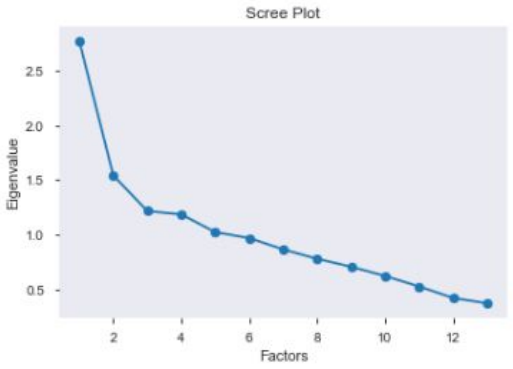
```
0.666246137902862
```

El test de bartlett nos indica que es posible el análisis factorial y el test kmol nos indica que es posible pero se encuentra el límite de su realización.

```
fa = FactorAnalyzer()
fa.analyze(corazon, 12, rotation = None)
ev, v = fa.get_eigenvalues()
ev
```

	Original_Eigenvalues
0	2.770752
1	1.541732
2	1.216776
3	1.185743
4	1.023144
5	0.969278
6	0.865050
7	0.779523
8	0.706230
9	0.622433
10	0.525867
11	0.421128
12	0.372342

```
plt.scatter(range(1,corazon.shape[1]+1),ev)
plt.plot(range(1,corazon.shape[1]+1),ev)
plt.title('Scree Plot')
plt.xlabel('Factors')
plt.ylabel('Eigenvalue')
plt.grid()
plt.show()
```



Se puede observar que los valores propios que son mayor a 1, son 5 valores por lo cual necesitaremos solo 5 factores en la factorización.

```
QAfa = FactorAnalyzer()
fa.analyze(corazon, 5, rotation="varimax")
fa.loadings
```

	Factor1	Factor2	Factor3	Factor4	Factor5
edad	-0.449013	-0.130724	0.581447	0.057528	0.036866
sexo	0.031976	0.883655	0.034589	0.020434	0.164400
dolorP	0.162880	0.049549	0.071143	-0.029580	-0.661879
presion	0.002925	-0.070181	0.442827	0.150144	-0.036453
colesterol	0.070047	-0.266707	0.319167	-0.040655	0.179020
azucar	-0.033535	0.074445	0.274661	0.019347	-0.106789
electro	-0.016318	-0.011716	-0.234774	-0.038758	-0.077949
maximaFc	0.747634	-0.016112	-0.048503	-0.308932	-0.282759
angina	-0.173350	0.064247	0.030593	0.239996	0.515411
depresionST	-0.079084	0.054664	0.215156	0.707517	0.215102
pendienteST	0.167648	-0.003262	-0.072554	-0.739937	-0.091364
busques	-0.184674	0.093317	0.348849	0.056042	0.229490
thal	0.061117	0.156257	0.151528	0.118945	0.325341

Analizando la data encontramos

	factor 1	factor 2	factor 3	factor 4	factor 5
edad	-0,449013	-0,130724	0,581447	0,057528	0,036866
sexo	0,031976	0,883655	0,034589	0,020434	0,164400
dolorP	0,162880	0,049549	0,071143	-0,029580	-0,661879
presion	0,002925	-0,070181	0,442827	0,150144	-0,036453
colesterol	0,070047	-0,266707	0,319167	-0,040655	0,179020
azucar	-0,033535	0,074445	0,274661	0,019347	-0,106789
electro	-0,016318	-0,011716	-0,234774	-0,038758	-0,077949
maximaFc	0,747634	-0,016112	-0,048503	-0,308932	-0,282759
angina	-0,173350	0,064247	0,030593	0,239996	0,515411
depresionST	-0,079084	0,054664	0,215156	0,707517	0,215102
pendienteST	0,167648	-0,003262	-0,072554	-0,739937	-0,091364
busques	-0,184674	0,093317	0,348849	0,056042	0,229490
thal	0,061117	0,156257	0,151528	0,118945	0,325341

En el Factor 1 tiene altas cargas de factor para MaximaFc
En el Factor 2 tiene altas cargas de factor para sexo
En el Factor 3 tiene altas cargas de factor para edad
En el Factor 4 tiene altas cargas de factor para depresiónST y pendienteST
En el Factor 5 tiene altas cargas de factor para angina

5.6.- Definición de variables

Dado el set de datos utilizado en conjunto con los resultados anteriores y tomando en consideración que la cantidad de variables desde el principio son solo 13, se mantendrá el set de datos intactos.

6.- Modelado y evaluación

6.1.- Segmentación data en grupo de entrenamiento y evaluación.

La separación de los datos se realiza gracias a la siguiente librería de python **'sklearn.model_selection'**, de esta importamos **train_test_split.** Esta aplicación permite separar la data total en forma lineal o aleatoria y con porcentaje a elección, en este caso se ocupará la manera aleatoria con 30% data de entrenamiento y 70% de data de evaluación con el comando: $entrenamiento, evaluacion = train\_test\_split(data, test\_size = 0.30)$

```
corazon = pd.read_csv('heart.csv')
corazon.columns = ['edad', 'sexo', 'dolorP', 'presion','colesterol','azucar', 'electro', 'maximaFc', 'a
corazon.head()
```

edad	sexo	dolorP	presion	colesterol	azucar	electro	maximaFc	angina	depresionST	pendienteST	busques	thal	target
63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

```
entrenamiento, evaluacion= train_test_split(corazon, test_size = 0.30)
```

```
entrenamiento_y = pd.DataFrame(entrenamiento['target'])
entrenamiento_x = entrenamiento.drop(['target'], axis=1)

evaluacion_y = pd.DataFrame(evaluacion['target'])
evaluacion_x = evaluacion.drop(['target'], axis=1)
```

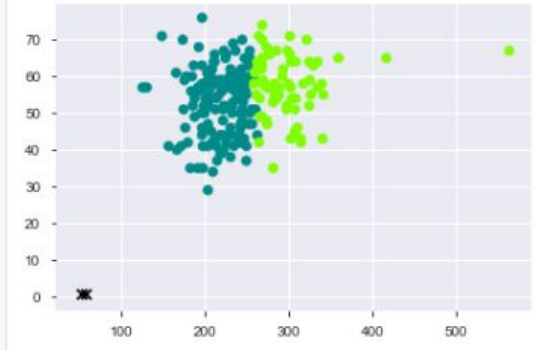
En la imagen anterior podemos observar como se lee la data para luego ser separado en evaluación y entrenamiento, siendo posterior la separación de x e y o data y objetivo.

6.2.- Aplicación de diversos modelos

6.2.1.-K-means(Cluster)

Con todos los datos a disposición:

```
kmeans = cluster.KMeans(2)
kmeans.fit(entrenamiento_x)
labels = kmeans.labels_
centroids = kmeans.cluster_centers_
plt.scatter(entrenamiento_x['colesterol'], entrenamiento_x['edad'], c=labels,cmap=ListedColormap(colors
plt.scatter(centroids[:,0], centroids[:,1], marker='x', c='black')
plt.show()
```

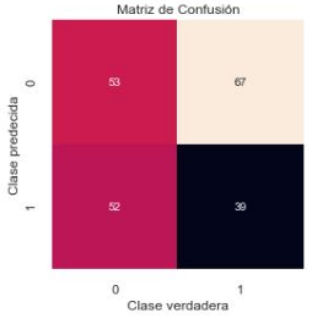


Se puede observar claramente la división aunque los centroides se encuentren alejados de los datos.

Luego analizamos los resultado asumiendo que tienen los mismos nombre, es decir cluster_0 corresponde a 0 y cluster_1 corresponde a 1.

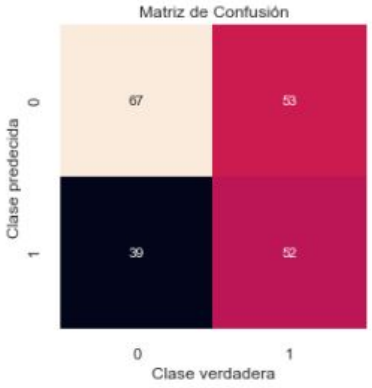
```
# Matriz de confusión
y_pred= lista_knn2
cnf_matrix = confusion_matrix(y_entrenamiento, y_pred)

print("Cantidad de errores de clasificación sobre un total de {0} casos: {1}"
      .format(y_entrenamiento.shape[0],(y_entrenamiento != y_pred).sum()))
print("Efectividad del algoritmo: {0: .2f}"
      .format(1 - (y_entrenamiento != y_pred).sum()/y_entrenamiento.shape[0]))

# Graficando la matriz de confusión
sns.heatmap(cnf_matrix.T, square=True, annot=True, fmt='d', cbar=False)
plt.xlabel('Clase verdadera')
plt.ylabel('Clase predecida')
plt.title('Matriz de Confusión')
plt.show()
```

```
Cantidad de errores de clasificación sobre un total de 211 casos: 119
Efectividad del algoritmo:  0.44
```



Luego se analiza con el contrario, es decir cluster _0 = 1 y cluster_1 = 0

```
Cantidad de errores de clasificación sobre un total de 211 casos: 92
Efectividad del algoritmo:  0.56
```



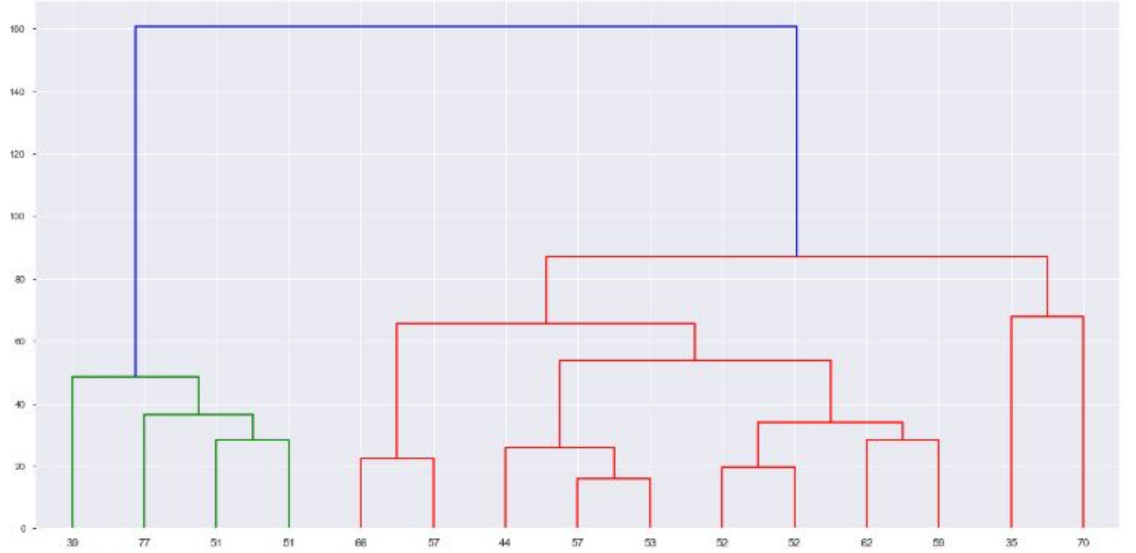
6.2.1.-Agrupación jerárquica(Cluster)

Se identifica una muestra de la data para trabajar de mejor manera el algoritmo jerárquico.

```
data = corazon.sample(15)
target_data = pd.DataFrame(data['target'])
data_x = data.drop(['target'], axis=1)
```

```
distancias = linkage(data_x.values, 'complete', metric='euclidean')
etiquetas = data_x.edad.values
```

```
plt.figure(figsize=(20, 10))
dendrogram(distancias, orientation='top',labels = etiquetas, distance_sort='descending',show_leaf_count
plt.show()
```



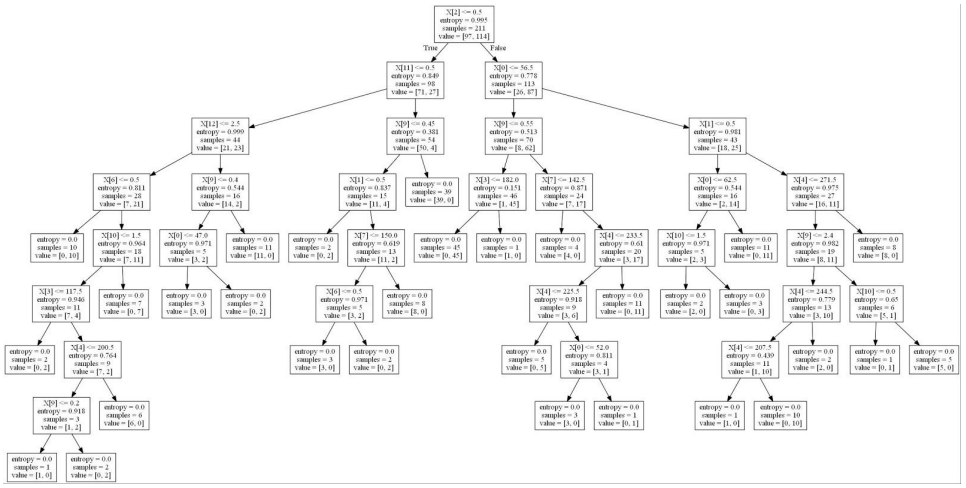
Este algoritmo logra dividir la data en 2, lo cual da 8 aciertos de observables de 15, lo cual se podría asumir como un 53.33% de acierto o una probabilidad de 0.53

6.2.2.-Árboles de decisión (Aprendizaje Supervisado)

```
ad = DecisionTreeClassifier(criterion='entropy', max_depth=10) # Creando el modelo
ad.fit(entrenamiento_x, lista_entrenamiento_y) # Ajustando el modelo

#generando archivo para graficar el arbol
with open("mi_arbol.dot", 'w') as archivo_dot:
    tree.export_graphviz(ad, out_file = archivo_dot)
```

```
from graphviz import Digraph
# utilizando el lenguaje dot para graficar el arbol.
!dot -Tjpeg mi_arbol.dot -o arbol_decision1.jpeg
```

Utilizando toda la data de entrenamiento obtenemos el siguiente árbol

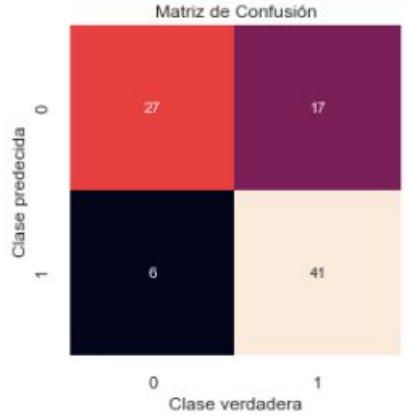


Este modelo nos dice que obtiene una precisión de

```
: # verificando la precisión
  print("precisión del modelo: {0: .2f}".format((lista_entrenamiento_y == ad.predict(entrenamiento_x)).me
```

```
precisión del modelo:  1.00
```

```
Cantidad de errores de clasificación sobre un total de 91 casos: 23
Efectividad del algoritmo:  0.75
```



6.2.2.-Random Forest (Aprendizaje Supervisado)

Se entrena el modelo y se evalúa su precisión

```
rf = RandomForestClassifier() # Creando el modelo
rf.fit(entrenamiento_x, entrenamiento_y) # Ajustando el modelo
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

```
# verificando la precisión
print("precisión del modelo: {0: .2f}".format((y_entrenamiento == rf.predict(entrenamiento_x)).mean()))
```

```
precisión del modelo:  1.00
```

```
# verificando la precisión test
print("precisión del modelo: {0: .2f}".format((y_evaluacion == rf.predict(evaluacion_x)).mean()))
```

```
precisión del modelo:  0.84
```

```
# verificando la precisión test
print("precisión del modelo: {0: .2f}".format((y_evaluacion == ad.predict(evaluacion_x)).mean()))
```

```
precisión del modelo:  0.79
```

```
# Armando un simple arbol de decisión
tree = DecisionTreeClassifier(max_depth=3, random_state=0)
tree.fit(entrenamiento_x, y_entrenamiento)
print('Precisión modelo inicial train/test  {0:.3f}/{1:.3f}'
      .format(tree.score(entrenamiento_x, y_entrenamiento), tree.score(evaluacion_x, y_evaluacion)))
```

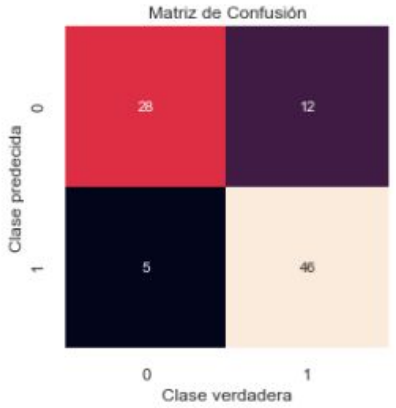
```
Precisión modelo inicial train/test  0.839/0.835
```

Se arma el modelo final

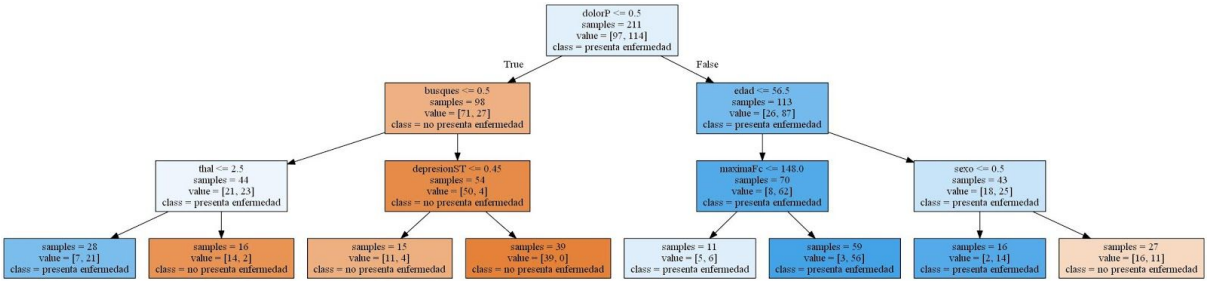
```
# Armando un simple arbol de decisión
tree = DecisionTreeClassifier(max_depth=3, random_state=0)
tree.fit(entrenamiento_x, y_entrenamiento)
print('Precisión modelo inicial train/test  {0:.3f}/{1:.3f}'
      .format(tree.score(entrenamiento_x, y_entrenamiento), tree.score(evaluacion_x, y_evaluacion)))
```

```
Precisión modelo inicial train/test  0.839/0.835
```

```
Cantidad de errores de clasificación sobre un total de 91 casos: 17
Efectividad del algoritmo:  0.81
```



Como se observa su precisión final es 0.839 en el entrenamiento y 0.835 en el test, el cual se puede observar en la siguiente imagen.



Una manera de aumentar la precisión es ocupar AdaBoost, la cual identifica las variables más importantes.

```
from sklearn.ensemble import AdaBoostClassifier
# Utilizando AdaBoost para aumentar la precisión
ada = AdaBoostClassifier(base_estimator=tree, n_estimators=500,
                         learning_rate=1.5, random_state=1)
# Ajustando los datos
ada = ada.fit(entrenamiento_x, y_entrenamiento)
```

```
ada.score(evaluacion_x, y_evaluacion)
```

```
0.8131868131868132
```

```
ada.score(entrenamiento_x, y_entrenamiento)
```

```
1.0
```

```
ada.feature_importances_
```

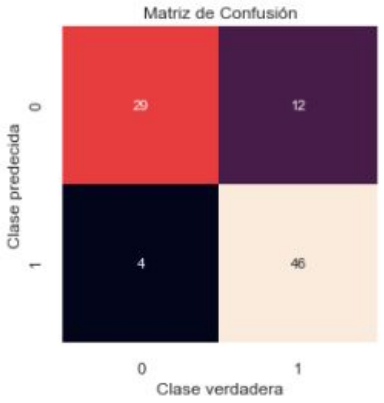
```
array([0.12168609, 0.02428683, 0.05944098, 0.13180385, 0.19015053,
       0.0073171 , 0.03292147, 0.1792512 , 0.02660971, 0.10255279,
       0.02297624, 0.05206466, 0.04893856])
```

```
imp = pd.DataFrame(zip(caracterictica, ada.feature_importances_), columns = ['Caracteristica', 'Importa
imp = imp.sort_values(by = 'Importancia', ascending = False)
imp.head()
```

	Caracteristica	Importancia
4	colesterol	0.190151
7	maximaFc	0.179251
3	presion	0.131804
0	edad	0.121686
9	depresionST	0.102553

Se puede identificar como características más importantes el colesterol y la maximaFc.

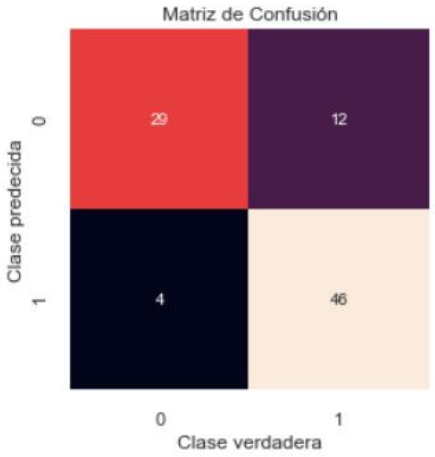
```
Cantidad de errores de clasificación sobre un total de 91 casos: 16
Efectividad del algoritmo:  0.82
```



6.2.3.-Redes bayesianas (Aprendizaje Supervisado)

Al aplicar redes bayesianas encontramos una gran probabilidad de aciertos, dado que tiene una probabilidad de acertar de 0.82.

```
Cantidad de errores de clasificación sobre un total de 91 casos: 16
Efectividad del algoritmo:  0.82
```



Cómo se logra apreciar en la matriz de confusión nos encontramos que del total de datos, 91 el algoritmo nos presenta 4 falsos negativos y 12 falsos positivos.

6.2.3.-SVM (Aprendizaje Supervisado)

6.2.3.1.-SVM, Kernel Lineal

Se entrena la al SVM

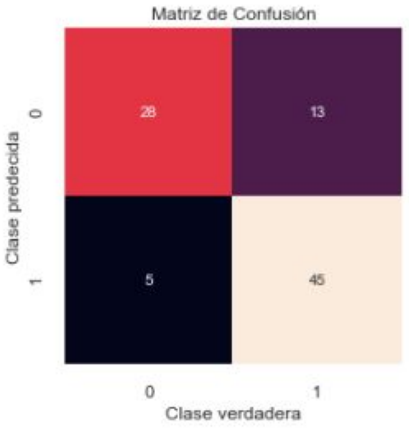
```
svclassifier = SVC(kernel='linear')
svclassifier.fit(entrenamiento_x, y_entrenamiento)
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

Luego se prueba con la data de evaluación o test y se comparan los resultados midiendo la efectividad, y comprobando la matriz de confusión.

```
y_pred = svclassifier.predict(evaluacion_x)

# Matriz de confusión
cnf_matrix = confusion_matrix(y_evaluacion, y_pred)

print("Cantidad de errores de clasificación sobre un total de {0} casos: {1}"
      .format(y_evaluacion.shape[0],(y_evaluacion != y_pred).sum()))
print("Efectividad del algoritmo: {0: .2f}"
      .format(1 - (y_evaluacion != y_pred).sum()/y_evaluacion.shape[0]))

# Graficando la matriz de confusión
sns.heatmap(cnf_matrix.T, square=True, annot=True, fmt='d', cbar=False)
plt.xlabel('Clase verdadera')
plt.ylabel('Clase predecida')
plt.title('Matriz de Confusión')
plt.show()
```

```
Cantidad de errores de clasificación sobre un total de 91 casos: 18
Efectividad del algoritmo:  0.80
```



6.2.3.2.-SVM, Kernel Polynomial

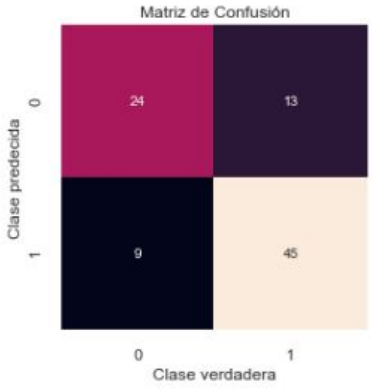
Se entrena la al SVM

```
svclassifier = SVC(kernel='poly', degree=2)
svclassifier.fit(entrenamiento_x, y_entrenamiento)
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=2, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

Se evalúan los resultados con la data de evaluación

```
Cantidad de errores de clasificación sobre un total de 91 casos: 22
Efectividad del algoritmo:  0.76
```



6.2.3.3.-SVM, Kernel Gaussian

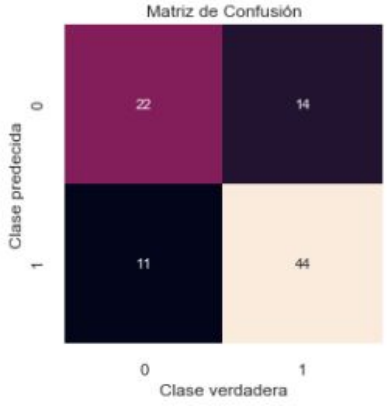
Se entrena la data

```
svclassifier = SVC(kernel='rbf')
svclassifier.fit(entrenamiento_x, y_entrenamiento)
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

Se evalúan los resultados con la data de evaluación

```
Cantidad de errores de clasificación sobre un total de 91 casos: 25
Efectividad del algoritmo:  0.73
```



6.2.3.4.-SVM, Kernel Sigmoid

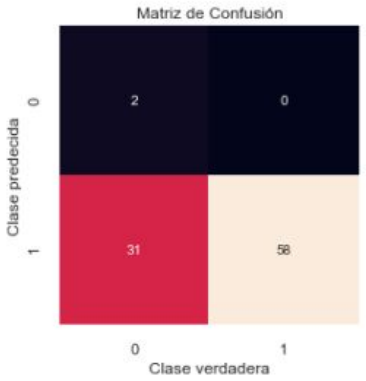
Se entrena la data

```
svclassifier = SVC(kernel='sigmoid')
svclassifier.fit(entrenamiento_x, y_entrenamiento)
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoid',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

Se evalúan los resultados con la data de evaluación

```
Cantidad de errores de clasificación sobre un total de 91 casos: 31
Efectividad del algoritmo:  0.66
```



6.3.- Validación de los diversos modelo

Juntamos los resultados de los modelos en término de error y efectividad:

Modelo	Efectividad	Error
K-means	0,56	92/211
Agrupación jerárquica	0,53	7/15
Arbol de decision	0,75	23/91
Random forest	0,81	17/91
Random forest AdaBoost	0,82	16/91
Redes Bayesianas	0,82	16/91
SVM Lineal	0,80	18/91
SVM Polinomial	0,76	22/91
SVM Gaussian	0,73	25/91
SVM Sigmoid	0,66	31/91

A través de estos resultados podemos identificar que los mejores modelos a utilizar corresponden a Random Forest con AdaBoost junto a Redes Bayesianas.

7.- Conclusiones

7.1.- Resumen del trabajo efectuado

Bueno, este proyecto se basó en un dataset de enfermedades coronarias, donde aplicamos distintos métodos. Los resultados obtenidos con los modelos utilizados fueron bastante variados aunque todos sobre el 50% de efectividad, aunque los modelos que más acertaron para este dataset son redes bayesianas y random forest adaboost.

7.2.- Explicación del modelo seleccionado y los motivos

A través del trabajo se analizó los datos con diferentes modelos, los cuales con mayor precisión corresponde a Random forest con AdaBoost y Redes Bayesianas ambos con un 82% de precisión. De estos modelos se ha seleccionado Random forest con AdaBoost dado su forma de trabajar con decisiones en base a la data que se asemeja de mejor manera a la forma en que trabajan los médicos con sus diagnósticos, se le agrega AdaBoost como una forma de optimizar el modelo.

7.3.- Cómo aplicar el modelo y extensión del trabajo

El modelo está pensado para ser aplicado en el ámbito médico, ayudando con los análisis efectuados una vez ingresado a un hospital por orden de urgencia, este modelo servirá para agilizar el diagnóstico, encontrando prioridad a los pacientes a tratar por enfermedades coronarias.