Predicción de enfermedades del corazón usando el algoritmo K-Nearest Neighbors

Maria Isabel Arango Palacio, María José Bernal Vélez

Martín Díaz Vélez, Isabella Montoya Henao

Universidad EAFIT

Medellín, Colombia

Resumen

Las enfermedades del corazón son la principal causa de muerte a nivel mundial, provocando que cuatro de cada cinco defunciones se deban a este tipo de enfermedades, en especial a cardiopatías coronarias y accidentes cerebrovasculares. No obstante, la atención oportuna y el diagnóstico adecuado de estas enfermedades generaría un gran impacto en el sistema de salud, puesto que se reduciría la tasa de mortalidad. Es por esto que con las nuevas tecnologías que han surgido en los últimos años, el análisis de patrones en los conjuntos de datos relacionados con las enfermedades del corazón han facilitado la predicción temprana de estas enfermedades. Por consiguiente, el objetivo de este trabajo es hacer uso de uno de los algoritmos más utilizados y eficaces para el reconocimiento de patrones, el KNN, para predecir si una persona va a sufrir de una enfermedad del corazón o no. Así pues, basándose en las métricas de desempeño, se obtuvo que el modelo realizado hace buenas predicciones sobre las personas que presentan enfermedades del corazón, minimizando en gran medida el error de predecir que el paciente está sano cuando en realidad está enfermo.

1. Introducción

Las enfermedades cardiacas son las más comunes a nivel mundial y las mayores causantes de muertes súbitas en la actualidad. La mayoría de las personas no son lo suficientemente conscientes de los síntomas ni de aquellos factores clave que llevan a ocasionarlas. Es por esto que, la atención oportuna y el diagnostico adecuado de este tipo de enfermedades reducirían la tasa de mortalidad en muchos países y evitarían que muchas personas combatieran con este tipo de enfermedades.

Para poder detectar aquellas posibles personas que pueden llegar a sufrir de una enfermedad del corazón es necesario analizar ciertas condiciones médicas de cada individuo para poder así realizar una clasificación que permita identificar que probabilidad tendría una persona, dadas sus condiciones de salud, de tener una enfermedad cardiaca.

Es por esto que, en el presente trabajo utilizamos un algoritmo de Machine Learning basado en el aprendizaje supervisado para estudiar las condiciones médicas de una población y poder predecir si tendrán o no una enfermedad cardiaca. El algoritmo implementado es el de los K-vecinos más cercanos (KNN), uno de los más utilizados en la industria a la hora de reconocer patrones y realizar clasificaciones. Luego de ser implementado, se expondrán según los análisis realizados, los resultados óptimos para cada uno de los componentes y métricas que influyen en que el modelo tenga la mejor predicción posible y que permiten evaluar que tan eficaz esta siendo.

2. Planteamiento del problema

Estados Unidos es uno de los países en los que las enfermedades del corazón afectan a la mayor parte de la población. Alrededor del 47 % de la población presentan al menos uno de los tres factores de riesgo clave de las enfermedades cardíacas, los cuales son presión arterial alta, colesterol alto y tabaquismo. Ahora bien, detectar y prevenir los factores que más influyen en las enfermedades del corazón es muy importante en la asistencia sanitaria.

Así pues, en este proyecto se utilizó el algoritmo KNN para predecir el estado de salud del corazón de una persona expresado como una variable dicotómica (enfermedad cardíaca: sí/no). Para tal fin, se hace uso de una base de datos de Kaggle que fue respondida por 319795 estadounidenses y son consideradas 18 diferentes variables donde están incluidas la edad, condición de diabético, la obesidad (IMC elevado), la falta de actividad física, el consumo excesivo de alcohol o de cigarrillo, entre otras. Las cuales pueden influir en si una persona presenta enfermedades del corazón como cardiopatía coronaria o infarto de miocardio [1].

El conjunto de datos más reciente es del año 2020, el cual en un principio fue recolectado por el Centro para el Control y Prevención de Enfermedades de Estados Unidos (CDC), siendo este una parte importante del Sistema de Vigilancia de los Factores de Riesgo en el Comportamiento (BRFSS), ya que realiza encuestas telefónicas anuales para recopilar datos sobre el estado de salud de los residentes en los 50 estados [1].

3. Marco teórico

3.1. Algoritmo K-vecinos más cercanos (KNN)

Para empezar, es importante introducir el concepto de Aprendizaje supervisado (Supervised learning). El aprendizaje supervisado en Machine Learning es un proceso que permite entrenar algoritmos que clasifican datos o predicen resultados con precisión, usando conjuntos de datos etiquetados. Esto, le permite al algoritmo aprender y mejorar una función que producirá un output apropiado cuando se le dan datos sin etiqueta [2]. El aprendizaje supervisado se puede dividir en dos tipos de problemas: Problemas de clasificación y problemas de regresión. Los problemas de clasificación utilizan un algoritmo para asignar con precisión los datos de prueba en categorías específicas; reconocen características específicas dentro del conjunto de datos e intenta sacar algunas conclusiones

sobre cómo se deben etiquetar o definir esas características. Y los problemas de regresión lineal se utilizan para comprender la relación entre las variables dependientes e independientes [2].

El algoritmo de los K- vecinos más cercanos (KNN) fue desarrollado en primera instancia por los estadísticos Evelyn Fix y Joseph Hodge en 1951 y años más tarde fue divulgado por el teórico informático Thomas Cover. El KNN es un algoritmo no paramétrico basado en el aprendizaje supervisado, utilizado principalmente para resolver problemas de clasificación [3]. Este algoritmo clasifica cada dato basado en la cercanía y asociación que tiene con el conjunto de datos ya clasificado, asumiendo que puntos con características similares del conjunto de datos se pueden encontrar uno cerca del otro [2]. Además, el KNN hace parte de los algoritmos con aprendizaje perezoso (lazy learning), los cuales no aprenden durante el entrenamiento sino que el aprendizaje sucede a medida que prueba los datos del test [4].

A continuación se describen las fases del algoritmo KNN, en las cuales se realiza todo el proceso de clasificación.

1. **Fase de entrenamiento**

   La fase de entrenamiento del algoritmo consiste básicamente en identificar, aprender y almacenar las características y las clases de los datos de entrenamiento. Estos datos son vectores multidimensionales en donde cada uno de ellos contiene una etiqueta (tipo) de clase, las cuales serán utilizadas más adelante para hacer la clasificación y en las cuales estarán agrupados los datos.

2. **Fase de clasificación**

   En esta fase se determina primero el parámetro k, que va a significar el numero de vecinos que se van a tener en cuenta a la hora de evaluar la distancia y determinar la clasificación de un dato respecto a los demás. Seleccionar este parámetro depende mucho de los datos que se tengan, generalmente grandes valores de k reducen el efecto de ruido a la hora de clasificar, pero hace que los limites en las clases sean menos claros [5].

   En segundo lugar, se calcula la distancia del punto de prueba a los k puntos vecinos, se determina cuales de ellos están más cerca y a que clase pertenecen. Con esto se clasifica el punto de prueba a la clase del punto más cercano, es decir, el que presenta menor distancia. La métrica más utilizada para calcular las distancias entre puntos en este algoritmo es la distancia euclídea dada por $d(p,q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}$ donde p es un punto con coordenadas (p_1, p_2) y q otro punto con (q_1, q_2) respectivamente [3]. Sin embargo, otro tipo de métricas como la distancia de Manhatthan, la distancia de Mahalanobis, coeficientes de correlación, entre otras, también pueden ser utilizadas.

   En pocas palabras, dado un punto X_q que debe ser clasificado y $X_1, X_2, ..., X_k$ los k vecinos más cercanos, el algoritmo de clasificación esta dado por

$$f(\hat{X}_q) = arg \max_{v \in V} \sum_{i=1}^{k} [v = f(X_i)]$$

en donde v representa las etiquetas de clase y el $f(\hat{X}_q)$ devuelto por el algoritmo de clasificación un estimador de $f(X_q)$ que corresponde al valor más común de f entre los k vecinos más cercanos al punto que se está clasificando X_q [6].

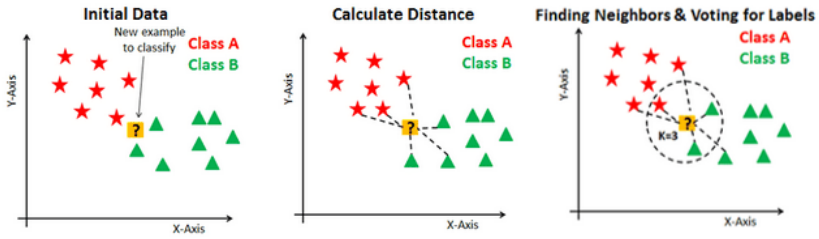

Figura 1: Ejemplo de clasificación del K-nn.

3.2. Ventajas y desventajas

En este apartado se mencionan algunas de las ventajas y desventajas más importantes del algoritmo de Machine Learning K-NN.

3.2.1. Ventajas

Entre las ventajas de implementar el KNN para realizar clasificaciones sobre datos están las siguientes [7]:

- Es un de los algoritmos de aprendizaje supervisado más simples; es fácil de entender e implementar.
- Tiene un limite de decisión bastante flexible ajustando el valor del parámetro K, esto le permite aprender limites de decisión no lineales.
- Es un algoritmo versátil, útil para diversos problemas como clasificación, regresión, búsqueda.
- Como su etapa de aprendizaje lo realiza a la hora de predecir, entre más datos nuevos tenga, más puede evolucionar.
- El algoritmo no necesita que se haga ninguna suposición frente a los datos por su estilo no paramétrico.

3.2.2. Desventajas

A pesar de lo sencillo que pueda ser el algoritmo, hay ciertos aspectos que hacen que no se sea la mejor elección a la hora de elegir un método de ML para clasifica y son los siguientes:

- El KNN tiene una complejidad alta a la hora de predecir grandes conjuntos de datos. Es muy lento cuando el número de dimensiones y de variables aumentan [7].
- Es sensible ante los datos atípicos, un solo dato poco común puede cambiar los limites de las clases.

- El algoritmo de los K-vecinos más cercanos asume que todas las características tienen igual importancia. Es otras palabras, es sensible si diferentes distancias tienen diferentes rangos.

3.3. K-vecinos más cercanos con distancia ponderada

El método clásico del KNN descrito anteriormente supone que los vecinos más cercanos al punto de prueba nos la mejor clasificación, esto se hace teniendo en cuenta las características del conjunto de datos. Sin embargo, el problema de suponer esto es que es posible que se tengan gran cantidad de atributos que son irrelevantes y que pueden afectar la clasificación. Adicionalmente, otro problema que se presenta en el KNN convencional es que si los k vecinos más cercanos varían mucho en su distancia, entonces la clasificación del punto de prueba x_q va a realizarse a una clase a la que probablemente no debería pertenecer [8].

Es por esto que, surge una variante del algoritmo, el KNN ponderado, el cual busca corregir el sesgo asignando pesos a las distancias, lo que permite darle mayor importancia a los vecinos más cercanos. En otras palabras, el algoritmo de K-vecinos más cercanos con distancia ponderada, como su nombre lo indica, pondera la contribución de cada vecino con respecto a la distancia entre el y el punto de prueba x_q a ser clasificado, dando mayor peso a los más cercanos [3]. Esta asignación de pesos a los k puntos más cercanos se realiza mediante una función Kernel; cualquier función se puede utilizar como función kernel para el clasificador KNN cuyo valor disminuye a medida que aumenta la distancia [8]. La función más simple y la más común es la función de distancia inversa, es decir,

$$W_i = \frac{1}{d(X_q, X_i)^2} \qquad i = 1, 2, ..., n$$

Y finalmente la predicción y clasificación del punto X_q ya teniendo los pesos y los k vecinos más cercanos $X_1, X_2, ..., X_k$, se sintetiza en la la función que se describe a continuación:

$$f(\hat{X}_q) = arg \max_{v \in V} \sum_{i=1}^{k} w_i [v = f(X_i)]$$

la cual funciona de manera similar a la descrita en el KNN clásico, solo que en esta se tienen en cuenta los pesos entre los k vecinos más cercanos para hacer la clasificación ponderada.

Esta mejora del KNN es más robusta ante los ruidos de los datos, ya que al tomar promedios ponderados de los k vecinos más cercanos el algoritmo puede evitar el impacto de datos con ruido aislados. Por otro lado, es bastante efectivo en conjuntos de datos grandes.

3.4. Análisis de componentes principales

El análisis de componentes principales, por sus siglas en inglés (PCA), es un modelo estadístico que expresa un conjunto de variables en combinaciones lineales de componentes no correlacionadas entre sí, con el fin de simplificar la complejidad de espacios muéstrales con muchas dimensiones, conservando al mismo tiempo la mayor cantidad de

información [9].

Para calcular las componentes principales de los datos, se obtienen la combinación lineal de las variables originales que tengan máxima varianza. Para esto, lo que se hace en primer lugar es calcular la matriz de covarianza definida como se muestra a continuación, la cual registra la correlación entre cada par de elementos y en la diagonal principal están contenidas las varianzas.

$$S = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})$$

Por lo tanto, como el espacio que mejor representa a los puntos viene definido por los vectores propios asociados a los r mayores autovalores de la matriz de covarianza S, van existir entonces tantas componentes principales como variables que se obtendrán calculando los valores propios o raíces características,$\lambda_1, ..., \lambda_n$, de la matriz de varianzas y covarianzas de las variables, S, mediante:

$$|S - \lambda I| = 0$$

Y sus vectores asociados son:

$$(S - \lambda_i I)a_i = 0$$

Los términos λ_i son reales, al ser la matriz S simétrica, y positivos, ya que S es definida positiva. Llamando Z a la matriz cuyas columnas son los valores de los p componentes en los n individuos, estas nuevas variables están relacionadas con las originales mediante:

$$Z = XA$$

donde A'A=I.

Finalmente, ordenando los vectores propios por valores propios en forma decreciente, se eligen los p vectores propios con los mayores valores propios para formar una matriz de n $\times$ p dimensional W. Utilizando esa matriz de vectores propios n $\times$ p para transformar las muestras en el nuevo subespacio de dimensión p [10].

No obstante, como para calcular el análisis de componentes principales se trabaja con varianzas, hace que el método PCA sea altamente sensible a datos atípicos (outliers), por lo que es altamente recomendable estudiar si los hay. La detección de valores atípicos con respecto a una determinada dimensión es algo relativamente sencillo de hacer mediante comprobaciones gráficas. Sin embargo, cuando se trata con múltiples dimensiones el proceso se complica.

3.5. Distancia de Mahalanobis

Como fue mencionado anteriormente, diferentes métricas se pueden implementar para el cálculo de las distancias entre los puntos. Una de las métricas que se utilizará más adelante será la distancia de Mahalanobis. Esta métrica calcula distancias teniendo en cuenta la relación entre las variables de los vectores, por lo que forma curvas de nivel para las distancias que tienen formas parecidas a la que presentan los datos en si. Para comprender esto mejor, se puede ver la siguiente figura:

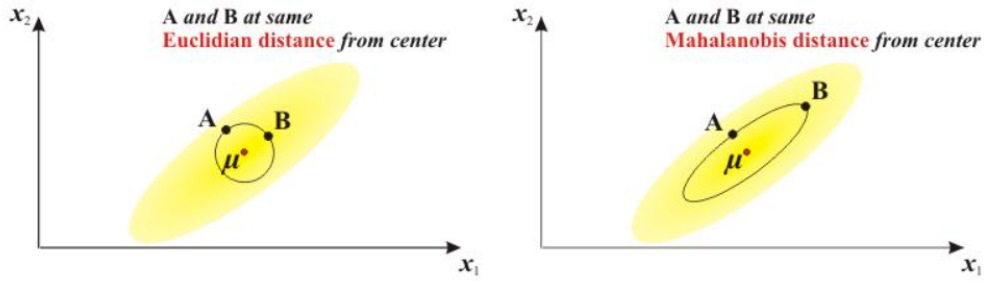

Figura 2: Curvas de nivel distancia Euclidea vs. distancia de Mahalanobis

Mientras que las curvas de nivel de la distancia euclídea siempre son en forma circular, las curvas de nivel de la distancia de Mahalanobis se adapta a la forma de los datos. El calculo de la distancia de Mahalanobis se puede llevar a cabo de la siguiente manera:

$$d(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T \Sigma^{-1} (\vec{x} - \vec{y})}$$

donde Σ es la matriz de covarianza de todos los datos disponibles y $\vec{x}, \vec{y}$ son datos puntuales [21].

3.6. Métricas para evaluar el desempeño del clasificador

En los modelos de clasificación existen diferentes métricas que permiten evaluar el desempeño del algoritmo, las cuales tienen focos diferentes, dependiendo del parámetro que se quiera analizar. A continuación se presentarán algunas de ellas, determinando cómo se calculan y qué representan del modelo.

3.6.1. Matriz de confusión

La matriz de confusión es una herramienta en el campo de la inteligencia artificial y el aprendizaje automático que permite visualizar el desempeño del algoritmo [15]. En esta, se presentan las siguientes posibilidades:

		Valor predecido	
		Falso	Verdadero
Valor real	Falso	Verdaderos negativos	Falsos positivos
	Verdadero	Falsos negativos	Verdaderos positivos

Como se puede ver, en esta matriz se tienen en las filas los valores reales y en las columnas los resultados de la predicción. En el caso de que el tanto el valor real como el predecido sean verdaderos o falsos, se tiene que el modelo logró clasificar de manera adecuada. Para este trabajo, esto se dará cuando una persona que tiene enfermedades de corazón se predice que está enferma o se obtiene que una persona está sana cuando en realidad sí lo está. Estas dos opciones son las deseadas, pero esto no siempre ocurre. Por un lado, el modelo puede predecir que cierta muestra cumple una determinada condición cuando en realidad no lo hace, el cual es el caso de un falso positivo, también conocido como el error tipo I. Por otro lado, cuando el modelo predice falso cuando la respuesta es verdadera, se tiene un falso negativo, o error tipo II. Específicamente, para las enfermedades de corazón, el error tipo I se obtendrá cuando se dice que cierto individuo está enfermo cuando en realidad está sano, y el error tipo II cuando se predice que una persona está sana pero lo cierto es que tiene alguna enfermedad del corazón.

Por tanto, la matriz de confusión contendrá la frecuencia de cada una de estas cuatro posibilidades, para la cual se esperaría que los falsos positivos y falsos negativos sean mucho menores que los verdaderos positivos y verdaderos negativos.

Esta matriz permite calcular varias métricas que se presentarán a continuación:

- **Exactitud (accuracy):** se encarga de calcular qué tan cerca está el resultado de una medición a su valor verdadero, el cual, en términos estadísticos, se relaciona con el sesgo de la estimación. Este se calcula con la siguiente fórmula:

$$\text{Accuracy} = \frac{\text{Verdaderos positivos} + \text{Verdaderos negativos}}{\text{Total de predicciones}}$$

  Es decir, la exactitud calcula la proporción de predicciones correctas que fueron realizadas [16]. Esta no es una buena métrica cuando se trabaja con bases de datos desbalanceadas, pues puede tener muy buenos resultados cuando el modelo es muy malo, o viceversa [17].

- **Precisión (precision):** se refiere a la dispersión del conjunto de resultados obtenidos, pues si hay menor dispersión, habrá mayor precisión. Esta se calcula como sigue:

$$\text{Precision} = \frac{\text{Verdaderos positivos}}{\text{Verdaderos positivos} + \text{Falsos positivos}}$$

  Lo cual representa el porcentaje de casos positivos que fueron detectados en el modelo [16].

- **Sensibilidad (recall):** indica la proporción de verdaderos positivos que fueron correctamente estimados por el algoritmo, calculada de la siguiente manera:

$$\text{Recall} = \frac{\text{Verdaderos positivos}}{\text{Verdaderos positivos} + \text{Falsos negativos}}$$

Esta indica la capacidad del algoritmo de poder predecir correctamente los casos verdaderos y se conoce también como tasa de verdaderos positivos [16].

- **Especificidad (specificity):** es la tasa de verdaderos negativos, que se encuentra como la proporción entre los casos negativos bien clasificados por el modelo, respecto al total de negativos. Esto es:

$$\text{Specificity} = \frac{\text{Verdaderos negativos}}{\text{Verdaderos negativos} + \text{Falsos positivos}}$$

Debido a que todas estas medidas son proporciones, siempre tomaran valores en el intervalo [0,1], las cuales, entre más cercanas a 1, indicarán que el modelo es mejor en este aspecto [16].

3.6.2. F1 score

El F1 score es una muy buena métrica cuando se trabaja con datos desbalanceados, como es el caso de este trabajo, debido a que este combina la precisión y la sensibilidad en una única métrica.

Esta se calcula como la media harmónica de la precisión y la sensibilidad, es decir:

$$\text{F1 score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

Debido a que la precisión y la sensibilidad toman valores entre 0 y 1, esta métrica también lo hará, alcanzando su mejor valor en el extremo superior de este intervalo [18].

3.6.3. Curva ROC y AUC

La curva Característica Operativa del Receptor (ROC) es una representación gráfica que muestra la capacidad de diagnóstico de los clasificadores binarios. Dicha curva se crea trazando la tasa de verdaderos positivos (TPR), es decir la sensibilidad, contra la tasa de falsos positivos (FPR), en varias configuraciones de umbral (valor a partir del cual se decide que un caso es un positivo) [19]. En general, la curva de ROC se puede generar graficando la función de distribución acumulada de la sensibilidad (Eje y) para diferentes valores de umbral, frente a la función de distribución acumulada de la tasa de falsos positivos (Eje X). En la siguiente gráfica se puede observar diferentes formas de la curva ROC:

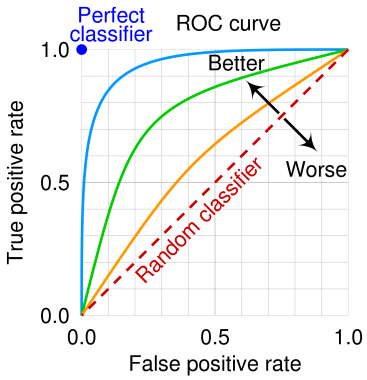

Figura 3: Curva ROC

De la anterior se puede ver que el modelo será mejor al clasificador aleatorio siempre y cuando la curva ROC esté por encima de la recta $y = x$, y será peor siempre que este por debajo de esta curva.

Nótese que para el calculo de la curva ROC se itera sobre diferentes umbrales para calcular los verdaderos positivos y los falsos positivos; sin embargo, el algoritmo de KNN simplemente devuelve, para cada punto, si este es positivo o no. Para tener la probabilidad de que un punto sea positivo, se puede calcular entonces la cantidad de datos positivos encontrados por el KNN dividido entre k. Con estas probabilidades se puede entonces establecer los diferentes umbrales para decidir si un punto se tomará como positivo o no y así calcular la curva ROC.

Por otro lado, el área bajo la curva ROC (AUC), es la métrica que cuantifica el rendimiento del clasificador para distinguir entre clases. Este valor se encuentra más comúnmente utilizando el método de integración numérica de regla trapezoidal en donde el área del trapezoide esta definida como:

$$\frac{(X_{i+1} - X_i) * (Y_i - Y_{i+1})}{2}$$

en donde X representa la probabilidad de falsos positivos en diferentes intervalos de probabilidad; y Y representa la probabilidad de verdaderos positivos en diferentes intervalos. Así,

$$\frac{(FPRi + 1 - FPR_i) * (TPR_i - TPR_{i+1})}{2}$$

Al final, el valor del AUC es la suma de todas las áreas trapezoidales encontradas en cada intervalo definido en donde en resultado de la suma tomará valores entre 0 y 1, siendo 1 el mejor valor que este puede tomar [20].

4. Metodología

4.1. Limpieza de la base de datos

Para lograr una buena predicción del estado de salud del corazón de una persona, se hace primeramente una limpieza o depuración de los datos con el objetivo de detectar, corregir o eliminar registros imprecisos de un conjunto de datos [11]. Así pues, se hizo uso de Python para facilitar dicho proceso, en el cual las variables dicotómicas fueron reemplazadas por variables binarias. Adicionalmente, como la variable edad está registrada en 13 intervalos de tamaño 4 que empiezan desde los 18 años hasta los 80 o más años, se considera el punto medio de cada intervalo y se reemplaza en la base de datos que se usará para la predicción.

Respuesta	Yes	Yes, during pregnancy	No	No, borderline diabetes	Male	Female
Reemplazo	1	0.8	0	0.7	1	0

Respuesta	Excellent	Very good	Good	Fair	Poor
Reemplazo	1	0.75	0.5	0.25	0

Por ende, cada uno de las variables de la base de datos que cuenta con alguna de las respuestas anteriores en sus respectivas filas, fue reemplazada por el valor respectivo mostrado en la tabla, esto con el fin de facilitar el procesamiento de los datos.

4.2. PCA

Dado que la base de datos cuenta con 18 variables, el análisis de los resultados obtenidos en las predicciones del modelo se dificultaba, ya que no se pueden representar gráficamente los datos y se hace más complejo encontrar tendencias en ellos. Adicionalmente, al hacer la predicción con todas las variables de la base de datos la complejidad en tiempo del programa es muy alta, de hecho puede demorarse en correr hasta más de 100 segundos, pero al aplicar el análisis de componentes principales al programa este problema se mejora notablemente ya que el tiempo de ejecución del programa se reduce a más del 50 %.

Es por esta razón, que se decide aplicar el método PCA sobre los datos para reducir la dimensión del dataset y la complejidad en tiempo del programa. Siguiendo cada uno de los pasos explicados con anterioridad se obtuvo que los tres vectores que maximizan la varianza son los autovectores de la matriz de covarianza asociados al máximo autovalor, en este caso $\lambda_1 = 0{,}29634$ y $\lambda_2 = 0{,}26506$ y $\lambda_3 = 0{,}19504$. Siendo estos los que mayor porcentaje de los datos explican, por lo que fueron los que se tomaron para reducir la dimensión de los datos.

4.3. Entrenamiento del modelo

Determinar la cantidad de datos que se van a designar para entrenar el modelo y para probarlo es una de las partes más fundamentales para lograr obtener buenos resultados a la hora de probar la calidad de predicciones que

hace el modelo. Por una parte, los datos de prueba son aquellos que se reservan para comprobar que el modelo tenga el funcionamiento esperado, y sea capaz de predecir de manera correcta lo que se le indica. Aunque es importante que los datos de prueba tengan un volumen suficiente como para lograr hacer análisis estadísticos significativos, la mayor cantidad de los datos va a ser usada para entrenar al modelo.

Normalmente, el conjunto de datos se suele repartir en un 80 % para el proceso de entrenamiento y un 20 % para la etapa de prueba. No obstante, estos porcentajes pueden variar según sea el caso, dependiendo de la complejidad del algoritmo y de la capacidad de aprendizaje que tenga [12].

Para este modelo, se designó el 90 % de los datos para entrenarlo, que corresponde a 287800 observaciones, y el 10 % de estos para validarlo, siendo estos 31980 registros. Sin embargo, como los datos estaban desbalanceados se utilizó un método de undersampling que hizo que la cantidad de datos de entrenamiento se redujera a 49246, con los cual la proporción de datos de entrenamiento y ensayo resultó ser finalmente de aproximadamente 60:40.

4.4. Undersampling

La mayoría de los algoritmos de clasificación están diseñados para trabajar sobre conjuntos de datos balanceados. Esto quiere decir que si el algoritmo de clasificación desea predecir a qué clase pertenece un individuo, lo óptimo es que la cantidad de individuos que pertenecen a cada clase sean similares.

Sin embargo, esto hace que se generen muchos problemas a la hora de pronosticar un nuevo individuo cuando las clases que se quieren predecir estén desbalanceadas. Esto se debe a que estos algoritmos se suelen basar en tratar de predecir la mayor cantidad de individuos de la forma correcta, o minimizar las clasificaciones malas, pero cuando los datos están desbalanceados es normal que este no sea el objetivo, pues muchas veces se quiere minimizar solo ciertos errores de clasificación, mientras que otros errores no tendrán tanta importancia [13].

En el presente caso, se quiere predecir si un paciente tiene enfermedades del corazón o no. Es importante notar que predecir que el paciente no está enfermo cuando en realidad si lo está es algo muy malo, pues ignorar esta enfermedad puede llegar a tener peores consecuencias. Por el otro lado, si se estima que un paciente está enfermo cuando no lo está, esto no es tan grave, pues igualmente se pueden tomar precauciones sobre la enfermedad o hacer estudios más profundos para revisar si sí lo está o no. En otras palabras, el objetivo principal será reducir principalmente los errores tipo II (predecir que no está enfermo cuando sí lo está).

El algoritmo de por si, no logrará reducir lo suficiente los erres tipo II, lo cual se verá a continuación.

Primero, en la siguiente gráfica se puede ver la cantidad de pacientes con enfermedades del corazón en comparación a la cantidad de pacientes sin la enfermedad:

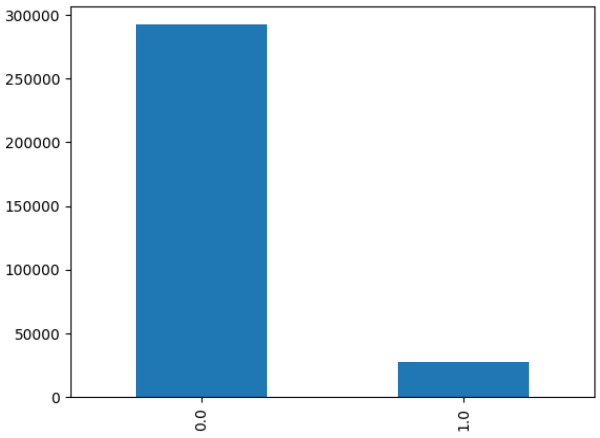

Figura 4: Proporción de clases

En la figura anterior, los $0's$ representan las personas sin enfermedad y los $1's$ la cantidad de personas enfermas, lo cual evidencia entonces que hay muchos más pacientes sanos que pacientes enfermos.

En primer lugar, es importante revisar que pasaría si se utiliza el dataset desbalanceado para el entrenamiento del algoritmo KNN. Los resultados obtenidos se pueden ver en la siguiente figura:

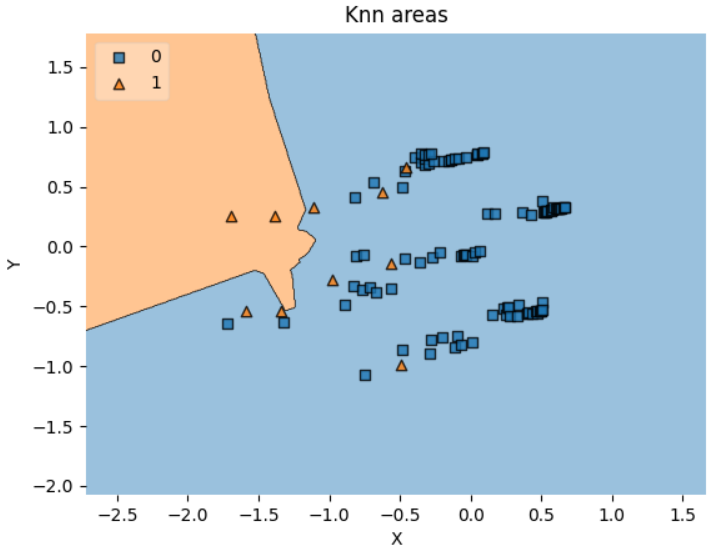

Figura 5: Regiones de clasificación data desbalanceada con k=4

En la figura anterior, los puntos que se pueden ver son las observaciones de 110 individuos de los datos originales, los cuales fueron utilizados como pacientes. Como fue mencionado anteriormente, la información de cada paciente se redujo a únicamente dos variables que expresan la máxima variabilidad posible de todas las originales, las cuales se representan en la gráfica sobre el eje x y y, esto se realizo para poder graficar y ver las tendencias de los datos. Además, los cuadrados representan que el paciente está sano, y los triángulos que está enfermo, de lo cual se puede

notar que en el entrenamiento se utilizaron muchos mas pacientes sanos que enfermos. También se pueden ver las dos regiones de predicción, una azul y la otra naranja, las cuales sirven para predecir los resultados de otros individuos sobre los cuales no se conoce su clase. Si el individuo está ubicado en la región azul, se predecirá que este está sano, de lo contrario se dirá que está enfermo. Nótese que la región azul es mucho mayor y la mayoría de los individuos están ubicadas en esta. Ahora, la siguiente matriz de confusión muestra los resultados de la predicción los resultados de 80 individuos de prueba:

Real\Predicción	Sano	Enfermo
Sano	72	1
Enfermo	7	0

De la anterior, se puede ver que, aunque predijo muchos individuos sanos de forma adecuado, solo logró predecir uno enfermo, el cual no lo estaba. En adición, cometió 7 veces el error tipo II para las 7 personas enfermas, los cuales son los más importantes de evitar, mostrando que la predicción fue muy mala.

Por tanto, para mejorar esta predicción, se utilizará el método de *undersampling*, el cual se basa en reducir la cantidad de los individuos de la clase dominante para balancear ambos [13]. Es importante notar que esta predicción solo se realizará para el entrenamiento, pues para la prueba se debe ensayar con casos reales, es decir, con las muestras desbalanceadas. La siguiente gráfica muestra el entrenamiento tras utilizar este método:

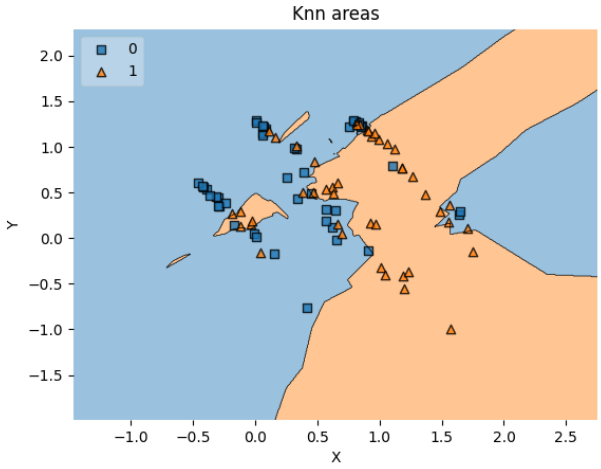

Figura 6: Regiones de clasificación tras undersampling con k=4

Se puede notar que ahora la región naranja es mucho mayor y que los datos están mejor distribuidos en ambas regiones, la azul y la naranja. Es posible ver que en el entrenamiento se están usando muchos más individuos enfermos que antes. La matriz de confusión generada por la clasificación de los 80 individuos del testeo es:

Real\Predicción	Sano	Enfermo
Sano	56	11
Enfermo	8	7

Anteriormente se estaban cometiendo todos los errores tipo II, es decir, todas las personas enfermas se estaban prediciendo como sanas. Ahora, la mitad de las personas enfermas se están prediciendo correctamente, lo cual presenta una buena mejora. No obstante, esta mejora se dio a costo de aumentar la cantidad de errores tipo I, indicando que más personas que no están enfermas se van a predecir como si lo estuvieran. Sin embargo, como se mencionó anteriormente, con tal de reducir el error tipo II, se aceptará aumentar hasta cierto punto el error tipo I.

Así, es posible entender la importancia del undersampling cuando las observaciones están desbalanceadas, logrando que el algoritmo prediga algo mucho más cercano al objetivo. Ahora, se explicará como se realiza el undersampling.

En primer lugar, existen diferentes métodos para hacer esto. El más sencillo es elegir muestras aleatorias de la clase mayor y eliminarlas. No obstante, para el algoritmo KNN se ha encontrado que existen otras técnicas más eficientes para realizar el undersampling. Para el algoritmo de este modelo, se decidió realizar pruebas con el undersampling aleatorio y con el algoritmo de undersampling NearMiss-1, el cual busca los tres individuos más cercanos de la clase menor para cada muestra de la clase mayor y calcula la distancia media a estos. Luego, se eliminan los datos de la clase mayor que tienen menor distancia media a los datos de la clase menor, con el propósito de eliminar los datos invasores de la clase mayor, ya que cuando están muy cerca a la clase menor, dificulta la capacidad del algoritmo para separar ambas clases. La desventaja de este undersampling siendo que distorsiona la geometría real de los datos de la clase mayor. [14].

4.5. Métrica de distancia Mahalanobis

Para este algoritmo se decidió implementar la distancia de Mahalanobis para revisar si esta daba mejores resultados. Sin embargo se presentó el problema de que el algoritmo utilizado para KNN no funciona adecuadamente al pasarle como parámetro la métrica de Mahalanobis (esto se debe a que los algoritmos de KNN mas eficientes usan estructuras más avanzadas como el kd-tree para calcular los puntos mas cercanos y estas estructuras de datos no estaban funcionando adecuadamente con la métrica de Mahalanobis).

Por tanto, la idea de solución se basa en que el algoritmo continúe usando la distancia euclídea pero mediante una transformación de los datos, para que la nueva distancia entre estos fuera la distancia Mahalanobis de los datos originales. Para lograr esto, se debe recordar la expresión de la distancia de Mahalanobis:

$$d(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T \Sigma^{-1} (\vec{x} - \vec{y})}$$

Por un lado, es importante recordar que la matriz de covarianzas Σ es semidefinida positiva, por lo que su inversa Σ^{-1} también lo es debido a las propiedades que esta tiene. Por otro lado, a este tipo de matrices se les puede aplicar la factorización de Cholesky, obteniendo la inversa de la matriz de covarianza como la multiplicación de una matriz cuadrada por su transpuesta, es decir, $\Sigma^{-1} = L^T * L$. Ahora, reemplazando en la fórmula de la distancia de

Mahalanobis se obtiene:

$$\begin{aligned} d(\vec{x},\vec{y}) &= \sqrt{(\vec{x}-\vec{y})^T L^T * L(\vec{x}-\vec{y})} \\ &= \sqrt{(L(\vec{x}-\vec{y}))^T * L(\vec{x}-\vec{y})} \\ &= \sqrt{(L\vec{x}-L\vec{y})^T * L\vec{x}-L\vec{y}} \end{aligned} \tag{1}$$

Nótese que ($L\vec{x}$ y $L\vec{y}$ son transformaciones lineales de los datos. Por ende, suponga que se tienen nuevos vectores de variables al multiplicar L por cada vector de variables original. Específicamente, sean $L\vec{x} = z$ y $L\vec{y} = w$, entonces:

$$d(\vec{x},\vec{y}) = \sqrt{(z-w)^T * (z-w)}$$

Esta última fórmula es la distancia euclídea par alos vectores z y w, por lo que la distancia de Mahalanobis de los datos originales es simplemente la distancia euclídea de los transformados. Por tanto, para implementar esta métrica en el algoritmo, se encuentra la descomposición de Cholesky de la inversa de la matriz de covarianza de los datos, y estos últimos se multiplican por la matriz generada tras esta factorización para seguir utilizando la misma distancia euclídea para el KNN.

Por otro lado, la distancia de Mahalanobis también fue utilizada tras haber realizado el PCA para proyectar a una dimensión menor, pero luego de utilizar este, la distancia de Mahalanobis no cambia en comparación a la distancia euclídea. Esto se debe a que al utilizar el método del PCA para reducir las dimensiones de los datos, estos quedan incorrelados y por ende, su matriz de covarianza es la identidad. En este caso específico, la fórmula de la distancia de Mahalanobis se vuelve la misma que la de la distancia euclídea. Esto es:

$$d(\vec{x},\vec{y}) = \sqrt{(\vec{x}-\vec{y})^T I^{-1}(\vec{x}-\vec{y})} = \sqrt{(\vec{x}-\vec{y})^T(\vec{x}-\vec{y})}$$

Por tanto, tras el uso del algoritmo del PCA no tiene mucho sentido el uso de la métrica de Mahalanobis.

5. Resultados

En esta sección se presentan los resultados óptimos obtenidos para nuestro modelo y los respectivos análisis de cada componente que influye en su efectividad.

5.1. Datos completos

5.1.1. Undersampling aleatorio

En esta parte se muestran los resultados obtenidos al tomar el conjunto de datos completos e implementar el algoritmo KNN sin realizarle ningún variación. Para este modelo, se realiza el undersampling aleatorio y se analizan tres parámetros que son: el valor óptimo que debe tomar K para conseguir el mayor valor para el AUC, el valor resultante del AUC y la matriz de confusión que se obtiene usando el K óptimo encontrado. Así pues, en primer lugar para determinar el K-óptimo se prueban diferentes valores de K comenzando en 50 hasta 1000, con un tamaño de paso de 50, y para cada valor de K se obtiene el respectivo AUC. Los resultados obtenidos se muestran a continuación:

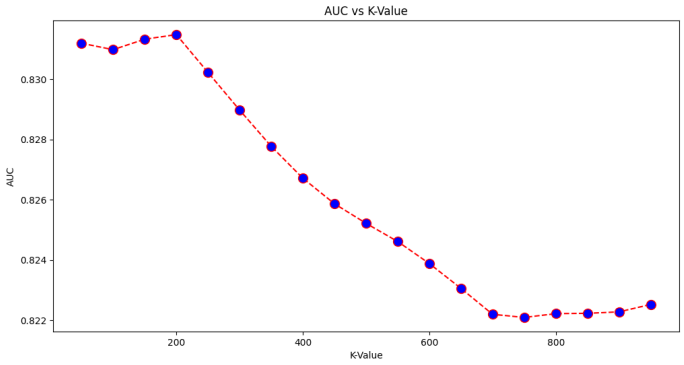

Figura 7: AUC con diferentes valores de k para los datos completos

Por lo tanto, se determina que el K óptimo tiene un valor de 200, para un valor máximo del AUC de 0.831. Para este K, los resultados que se obtienen para la matriz de confusión son:

Real\Predicción	Sano	Enfermo
Sano	21140	8090
Enfermo	590	2160

De los resultados anteriores se observa que de los 319975 personas estadounidenses, el modelo predijo 21140 como sanas cuando en realidad están sanas y 2160 como enfermas cuando en realidad tienen enfermedades del corazón. Ahora bien, el modelo predijo que 8090 personas están enfermas cuando en realidad se encuentran sanas, y el error tipo II se logra minimizar a 590, indicando que el modelo predijo 590 personas como sanas cuando en realidad están enfermas.

5.1.2. Undersampling NearMiss-1

En esta parte se muestran los resultados obtenidos al tomar el conjunto de datos completos e implementar el algoritmo KNN sin realizarle ningún variación. Para este modelo, se realiza otro tipo de undersampling (Near-Miss

I) y se analizan los mismo tres parámetros que se consideraron en el numeral anterior. De lo cual, se obtiene lo siguiente:

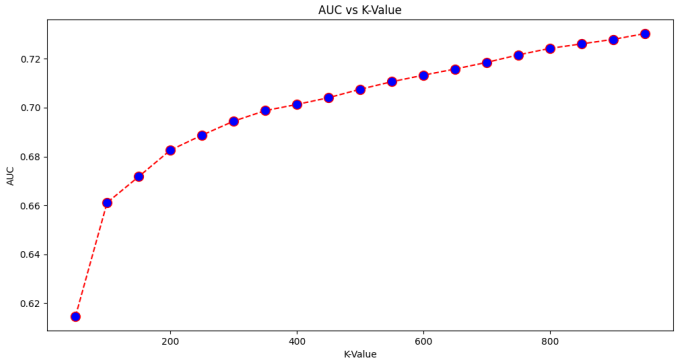

Figura 8: AUC con diferentes valores de k para los datos completos con Near-Miss I

De la gráfica anterior se evidencia que el AUC toma su máximo valor (AUC=0.823) cuando K tiene un valor de 989. Para este K, los resultados que se obtienen para la matriz de confusión son:

Real\Predicción	Sano	Enfermo
Sano	22389	6841
Enfermo	763	1987

La matriz de confusión anterior, permite observar que la magnitud del error tipo I y II al hacer el balanceo de los datos con NearMiss-1. De esto se observa que, se aumenta el error tipo II en comparación con el modelo anterior, pues en este caso predice que 763 personas están sanas cuando en realidad están enfermas. Sin embargo, al aplicar este tipo de undersampling el error tipo I se reduce a comparación del obtenido anteriormente, pues se obtiene que el modelo predice 6841 pacientes como enfermos cuando en realidad están sanos.

5.2. Distancia de Mahalanobis

Por otra parte, al utilizar la distancia de Mahalanobis para implementar el algoritmo KNN se realizó la siguiente gráfica para determinar cuál es el valor de K óptimo para esta variación al algoritmo. Esta se encarga de implementar el algoritmo KNN para diferentes valores del parámetro K, comenzando desde 21 y terminando en 381 con un tamaño de paso de 20, y para cada uno de ellos calcula el AUC. Así, los resultados obtenidos son los siguientes:

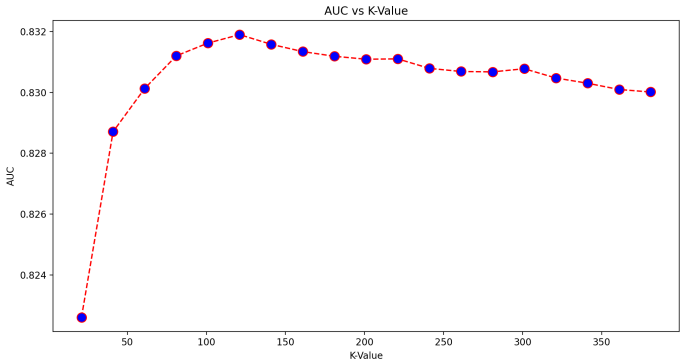

Figura 9: AUC con diferentes valores de k con distancia de Mahalanobis

De la anterior, se puede ver que el AUC se maximiza cuando el valor del parámetro k es 121, obtiendo un valor del AUC de 0.83189. Luego, la matriz de confusión obtenida a partir de utilizar el valor de k óptimo es:

Real\Predicción	Sano	Enfermo
Sano	20636	8594
Enfermo	541	2209

De la matriz de confusión anterior es posible notar que la cantidad de personas que se predijeron como sanas cuando en realidad estaban enfermas es de 541, que corresponde a una baja probabilidad.

5.3. Análisis del PCA

En esta sección, se realiza el análisis de componentes principales para reducir la dimensión de los datos a 2 y 3 variables. Para cada una de ellas se encuentra el valor de k que maximiza el AUC al tomar diferentes valores de k, desde 21 hasta 381, con un tamaño de paso de 20. Luego, con el valor óptimo encontrado para este parámetro, se realiza su respectiva matriz de confusión.

5.3.1. PCA a dos dimensiones

Al reducir la dimensión del conjunto de datos a 2 variables, se calculo en primera instancia el valor óptimo de K que permitía maximizar el AUC del modelo. En la siguiente gráfica se puede ver cada valor de k y el AUC que se obtiene

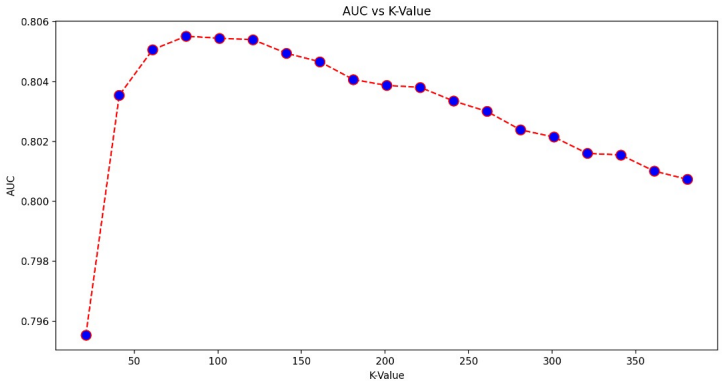

Figura 10: AUC para diferentes valores de k con PCA 2

Así entonces, resulta que el valor de k que maximiza el rendimiento del clasificador es de $k = 81$ para un AUC = 0.8055. Ya con este valor para el parámetro k, se puede observar la siguiente matriz de confusión:

real\predicción	Sano	Enfermo
Sano	19914	9316
Enfermo	571	2176

Dicha matriz permite observar la magnitud de los errores tipo I y tipo II al reducir el número de variables a 2. De esta manera se puede evidenciar que el error tipo II, que es el que se tiene como objetivo minimizar, resulto ser bajo en comparación con todas las personas que se están estudiando. Es decir que, de las 319975 personas estudiadas, se predijo que 571 están sanas cuando en realidad están enfermas.

5.3.2. PCA a tres dimensiones

Luego de reducir la dimensión de los datos a 3 variables, la gráfica obtenida para encontrar el valor de k óptimo es la siguiente:

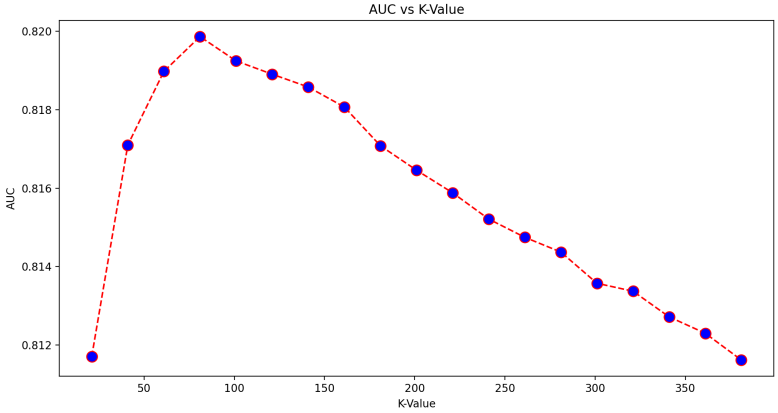

Figura 11: AUC para diferentes valores de k con PCA 3

De la anterior, es posible notar que el valor de k que maximiza el AUC en este caso es 81, para el cual se tiene un AUC de 0.81986. Además, la matriz de confusión para este parámetro es:

real\predicción	Sano	Enfermo
Sano	19643	9587
Enfermo	474	2276

De la anterior es posible notar que la cantidad de errores tipo II que se van a realizar en las predicciones con este método es 474, un valor relativamente bajo comparado con el total de individuos predichos.

Se puede concluir después de analizar las diferentes reducciones realizadas a los datos que, la reducción de las dimensiones con la que se tienen mejores resultados para el AUC y el error tipo II es p=3.

5.4. Resultados óptimos

Luego de los análisis realizados sobre los dos tipos de undersampling, el cambio en las métricas de distancia y la reducción de las dimensiones mediante PCA, se determina que los resultados óptimos se obtienen al usar el undersampling aleatorio debido a que con este tipo de balanceo se presenta un menor error tipo II y además se obtiene un AUC muy bueno. Adicionalmente, se opta por no modificar la métrica de distancia, es decir, se trabaja con la distancia euclídea debido a que se obtienen resultados muy similares al usar la distancia de Mahalanobis, y hacer esta modificación hace que la implementación del modelo sea un poco más tediosa. Por último, en cuanto a la reducción de la dimensión de los datos se decide explicar los datos en términos de tres variables ya que, aunque el AUC bajara un poco, el algoritmo con PCA en tres dimensiones fue el que menos cometió errores tipo 2, además la complejidad en tiempo de el programa se reduce mucho y se facilita el análisis de los datos.

De esta manera, la matriz de confusión obtenida con estos resultados óptimos es la siguiente:

real\predicción	Sano	Enfermo
Sano	19643	9587
Enfermo	474	2276

Sobre esta última, se obtienen las siguientes métricas:

$$\text{Exactitud} = \frac{19643 + 2276}{19643 + 9587 + 474 + 2276} = 0{,}68539$$
$$\text{Precisión} = \frac{19643}{19643 + 9587} = 0{,}67201$$
$$\text{Sensibilidad} = \frac{19643}{19643 + 474} = 0{,}97643$$
$$\text{Especificidad} = \frac{2276}{2276 + 9587} = 0{,}19185$$

Como fue mencionado anteriormente, para el caso de este trabajo, la métrica de exactitud no es la más adecuada, ya que esta no es una buena métrica para datos desbalanceados. Además, debido a que el enfoque de este trabajo

es reducir el error tipo II, la métrica más importante de las anteriores es la sensibilidad, ya que esta considera la cantidad de falsos negativos obtenidos. De esta manera, se tiene que el modelo obtenido tiene una muy buena sensibilidad, ya que toma un valor del 97.64 %, indicando que la capacidad del algoritmo de predecir correctamente los casos verdaderos es muy buena.

Por otro lado, a partir de la precisión y la sensibilidad se puede calcular el F1 score de la siguiente manera:

$$\text{F1 score} = 2 * \frac{0{,}67201 * 0{,}97643}{0{,}67201 + 0{,}97643} = 0{,}79611$$

La métrica anterior se encarga de combinar la sensibilidad y la precisión en una sola métrica, asignándole el mismo peso a ambas, la cual indica que el rendimiento del algoritmo es bueno.

Por otro lado, es importante calcular la probabilidad de cometer los errores tipo I y II, ya que estos son una parte crucial en el buen funcionamiento del modelo. Los obtenidos son los siguientes:

$$P(\text{error tipo I}) = \frac{9587}{19643 + 9587} = 0{,}32798$$
$$P(\text{error tipo II}) = \frac{474}{474 + 2276} = 0{,}17236$$

Así, se tiene que las probabilidades de cometer los errores tipo I y II son del 32.79 % y 17.24 %, respectivamente. Es posible notar que la probabilidad de diagnosticar una persona como sana cuando en realidad está enferma es baja, lo cual indica que el modelo funciona bien para su propósito principal.

Finalmente, la curva ROC obtenida con este modelo es la siguiente:

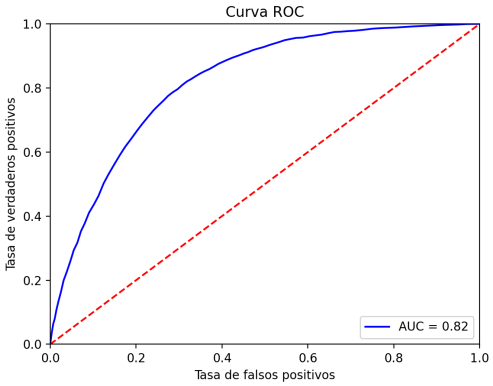

Figura 12: Curva ROC para resultados óptimos

Debido a que la curva obtenida (en azul) está por encima a la función identidad (en rojo), se puede decir que el algoritmo tiene un buen desempeño, ya que este funciona mejor que el método aleatorio. Luego, el AUC es igual a 0.81986, lo que indica que el modelo es capaz de distinguir correctamente si una persona tiene enfermedades del corazón o no con una probabilidad del 81.98 %.

6. Conclusión

En definitiva, el algoritmo KNN es una muy buen alternativa a la hora de hacer predicción de datos debido a su fácil implementación, y su flexibilidad con los límites de decisión, lo cual se logra con las variaciones que toma el parámetro k. Así pues, en el proyecto se implementan diferentes variaciones para este modelo, de lo cual se obtuvo que al reemplazar la métrica de la distancia del algoritmo original (euclídea) por la distancia de Mahalanobis se observa que no hay cambios significativos frente al modelo normal, ya que al calcular las métricas de desempeño de mayor relevación para nuestro modelo (el AUC y la matriz de confusión), se obtuvieron resultados bastante similares, determinando así que no vale la pena cambiar la distancia euclídea por la distancia de Mahalanobis para nuestra clasificación.

Por otra parte, los resultados obtenidos al hacer PCA y reducir a 3 dimensiones los datos fueron satisfactorios, debido a que aunque el AUC bajaba un poco (tomando los datos sin hacer PCA el AUC=0.831 y reduciendo a tres dimensiones el AUC=0.819), al explicar los datos en términos de tres variables la complejidad en tiempo del programa se reducía, se cometian menos errores tipo 2 y adicional a esto se facilitaba el análisis de los resultados.

Por último, sobre los resultados del modelo, se puede notar que genera un buen resultado, ya que se logra reducir la probabilidad del error tipo I a un porcentaje bajo. No obstante, cuando se hacen predicciones con datos imbalanceados es sumamente importante determinar el tipo de underdampling que presente mejores resultados en el modelo. Si bien, en un principio los datos se les hizo un tipo de undersampling llamado Near-Miss I, con el cual la capacidad de diagnostico es similar a la que se obtiene haciendo el undesampling aleatorio, AUC=0.823 y AUC=0.831, respectivamente. Sin embargo, el error tipo II que es el que nos interesa minimizar toma un menor valor cuando se hace el undersampling aleatorio por lo que se decide que es una mejor forma de balancear los datos. Por lo tanto con las modificaciones realizadas, se puede concluir que el algoritmo si tuvo un desempeño bueno en la clasificación de pacientes con enfermedades del corazón.

Referencias

[1] K, Pytlak, *Personal Key Indicators of Heart Disease*, Kaggle, 2020 [En línea]. Disponible en: https://www.kaggle.com/datasets/kamilpytlak/personal-key-indicators-of-heart-disease [Accedido: Mayo 10, 2022]

[2] IBM Cloud Education, *Supervised Learning*, IBM Cloud Learn Hub, 2020 [En línea]. Disponible en: https://www.ibm.com/cloud/learn/supervised-learning#:~:text=Supervised%20learning%2C%20also%20known%20as,data%20or%20predict%20outcomes%20accurately [Accedido: Mayo 15, 2022]

[3] Wikipedia, *k-nearest neighbors algorithm*, 2022 [En línea]. Disponible en: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#Statistical_setting [Accedido: Mayo 15, 2022]

[4] Merkle, *El algoritmo K-NN y su importancia en el modelado de datos*, 2020 [En línea]. Disponible en: https://www.merkleinc.com/es/es/blog/algoritmo-knn-modelado-datos [Accedido: Mayo 17, 2022]

[5] L. Gonzalez, *K Vecinos más Cercanos – Teoría*, 2019 [En línea]. Disponible en: https://aprendeia.com/k-vecinos-mas-cercanos-teoria-machine-learning/ [Accedido: Mayo 15, 2022]

[6] Wikipedia, *k vecinos más próximos*, 2021 [En línea]. Disponible en: https://es.wikipedia.org/wiki/K_vecinos_m%C3%A1s_pr%C3%B3ximos [Accedido: Mayo 16, 2022]

[7] O. Harrison, *Machine Learning Basics with the K-Nearest Neighbors Algorithm*, Towards Data Science, 2018 [En línea]. Disponible en: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761 [Accedido: Mayo 15, 2022]

[8] kanakalathav99, *Weighted K-NN*, Geeks for Geeks, 2020 [En línea]. Disponible en: https://www.geeksforgeeks.org/weighted-k-nn/ [Accedido: Mayo 17, 2022]

[9] J. Amat Rodrigo, *Análisis de Componentes Principales (Principal Component Analysis, PCA) y t-SNE*, Ciencia de Datos, 2017 [En línea]. Disponible en: https://www.cienciadedatos.net/documentos/35_principal_component_analysis [Accedido: Mayo 18, 2022]

[10] D. Peña, *Analisis de Datos Multivariantes*, Universidad Carlos III de Madrid, 2002.

[11] Kyocera, *La limpieza de datos como activo empresarial*, KYOCERA Document Solutions, 2021 [En línea]. Disponible en: https://www.kyoceradocumentsolutions.es/es/smarter-workspaces/insights-hub/articles/la-limpieza-de-datos-como-activo-empresarial.html [Accedido: Mayo 18, 2022]

[12] J. Brownlee, *Impact of Dataset Size on Deep Learning Model Skill And Performance Estimates*, Machine Learning Mastery, 2019 [En línea]. Disponible en: https://machinelearningmastery.com/impact-of-dataset-size-on-deep-learning-model-skill-and-performance-estimates/ [Accedido: Mayo 18, 2022]

[13] J. Brownlee, *Why Is Imbalanced Classification Difficult?*, Machine Learning Mastery, 2020 [En línea]. Disponible en: https://machinelearningmastery.com/imbalanced-classification-is-hard/ [Accedido: Mayo 17, 2022]

[14] B. Madhukar, *Using Near-Miss Algorithm For Imbalanced Datasets*, Analytics India Magazine, 2020 [En línea]. Disponible en: https://analyticsindiamag.com/using-near-miss-algorithm-for-imbalanced-datasets/ [Accedido: Mayo 17, 2022]

[15] A. Bhandari, *Everything you Should Know about Confusion Matrix for Machine Learning*, Analytics Vidhya, 2020 [En línea]. Disponible en: https://www.analyticsvidhya.com/blog/2020/04/confusion-matrix-machine-learning/ [Accedido: Mayo 20, 2022]

[16] J. I. Barrios Arce, *La matriz de confusión y sus métricas*, Health Big Data, 2019 [En línea]. Disponible en: https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/ [Accedido: Mayo 20, 2022]

[17] Aprende Machine Learning, *Clasificación con datos desbalanceados*, 2019 [En línea] Disponible en: https://www.aprendemachinelearning.com/clasificacion-con-datos-desbalanceados/ [Accedido: Mayo 20, 2022]

[18] J Korstanje, *The F1 score*, 2021 [En línea]. Disponible en: https://towardsdatascience.com/the-f1-score-bec2bbc38aa6 [Accedido: Mayo 20, 2022]

[19] Machine Learning Crash Course, *Classification: ROC Curve and AUC*, Google Developers, 2020 [En línea]. Disponible en: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc [Accedido: Mayo 20, 2022]

[20] D. Bhalla , *A complete guide to Area Under Curve (AUC)*, Listen Data, 2014 [En línea]. Disponible en: https://www.listendata.com/2014/08/learn-area-under-curve-auc.html [Accedido: Mayo 29, 2022]

[21] M. T Escobedo Portillo y J. Salas Plata, "P. Ch. Mahalanobis y las aplicaciones de su distancia estadística", Cultura Científica y Tecnológica, 2008 [En línea]. Disponible en: https://www.researchgate.net/publication/28249208_P_Ch_Mahalanobis_y_las_aplicaciones_de_su_distancia_estadistica