TC2034 Proyecto de Aprendizaje Supervisado

Equipo 6

Adrián Landaverde Nava A01745052

Naomi Padilla Mora A01745914

Sabrina Nicole Rodríguez Salgado A01745197

Índice

Introducción al conjunto de datos (dataset) seleccionado

El conjunto de datos seleccionado fue el *Heart Failure Prediction Dataset,* obtenido de la plataforma *Kaggle*. El *heart.csv* está compuesto por 12 columnas y 918 filas. Las columnas conforman las siguientes variables de estudio y de los siguientes tipos.

- *Age:* Edad del paciente. *Cuantitativa-int.*
- *Sex:* Sexo del paciente. F:Female, M:Male. *Cualitativa-object.*
- *ChestPain Type:* Tipo de dolor de pecho que presenta el paciente. TA:Typical Angina, ATA:Atypical Angina, NAP:Non-Anginal Pain, ASY:Asymptomatic. *Cualitativa-object.*
- *RestingBP:* Presión en la sangre cuando el paciente está en reposo (mm Hg). *Cuantitativa-int.*
- *Cholesterol:* Nivel de colesterol en el paciente (mm Hg). *Cuantitativa-int*.
- *FastingBS:* Prueba de glucosa en ayunas donde es 1 si *FastingBS > 120 mg/dl,* de lo contrario, 0. *Cuantitativa-int*.
- *RestingECG:* Resultados del electrocardiograma realizado al paciente en reposo. Normal:Normal, ST:con anomalías en la onda ST-T (inversión de la onda T y/o elevación o depresión del ST > 0.05 mV), LVH:muestra hipertrofia ventricular izquierda probable o definitiva según los criterios de *Estes*. *Cualitativa-object*.
- *MaxHR:* frecuencia cardiaca máxima alcanzada por el paciente (60 a 202). *Cuantitativa-int.*
- *ExerciseAngina:* Si el paciente presenta una *Angina* inducida por el ejercicio. Y:Yes, N:No. *Cualitativa-object.*
- *Oldpeak:* depresión del ST inducida por el ejercicio en relación con el reposo que se observa en el electrocardiograma del paciente. *Cuantitativa-float.*
- *ST_Slope:* la pendiente del segmento ST del ejercicio máximo observada en el electrocardiograma del paciente. Up: upsloping(ascendiente), Flat:flat(plana), Down:downsloping(descendiente). *Cualitativa-object.*
- *HeartDisease:* sí se observa una enfermedad cardiaca en el paciente. 1:heart disease (tiene una enfermedad cardiaca), 0:Normal (no tiene una enfermedad cardiaca). *Cuantitativa-int.*

De lo anterior, identificamos a la variable *HeartDisease* como la variable **dependiente** a predecir. Ya que para conocer si el paciente sufre o no de una enfermedad cardiaca es necesario analizar sus resultados en el resto de las variables, las cuales, se identificarán como variables **independientes.** El conjunto de datos tiene un número razonable de observaciones que permitirán entrenar a los algoritmos correspondientes de forma adecuada.

La razón por la que se eligió este dataset centrado en la insuficiencia cardiaca es porque este tipo de condición, además de que puede ser mortal por sí sola (Mayo Clinic, 2022), suele ser indicador de otros tipos de enfermedades en un paciente. Por ejemplo, la insuficiencia cardíaca puede ser causada por hipertensión, diabetes, obesidad, enfermedad coronaria, o enfermedades cardiovasculares en general (Mayo Clinic, 2022), las cuales son la causa número uno de muerte en el mundo según la Organización Mundial de la Salud

(2020). Por lo tanto, la insuficiencia cardiaca (*Heart Disease* en inglés) es un indicador importante de la presencia de otras condiciones potencialmente fatales. Es por esto que es vital detectar los casos de insuficiencia cardiaca y los factores asociados a ésta que pudieran contribuir a detectarla.

Para poder realizar el análisis y la aplicación de modelos de predicción en este *dataset* se tuvo que realizar un *pre-tratamiento* de los datos para garantizar un análisis adecuado y eficiente. Dicho preprocesamiento de los datos se hizo una sola vez y se usó para todos los algoritmos. Posterior a la familiarización con el *dataset*, donde se observó su dimensión y composición de variables, se llevó a cabo una búsqueda de valores nulos, con esta nos aseguramos de que el *dataset* no contaba con valores nulos que obstaculizaran el análisis.

Una vez realizado esto, se consultó de qué tipo (cualitativa-object o cuantitativa-int-float) era cada variable. Con base en esto, las variables que resultaron cualitativas se cambiaron a cuantitativas redefiniendo sus componentes con números, por ejemplo: Para la variable *Sex, F:Female, M:Male*, se redefinió como *Sex, F:0, M:1.* Esto con el objetivo de poder aplicar los modelos de predicción, los cuales requieren únicamente de variables cuantitativas. En el siguiente gráfico podemos observar las distribuciones de cada una de estas variables.

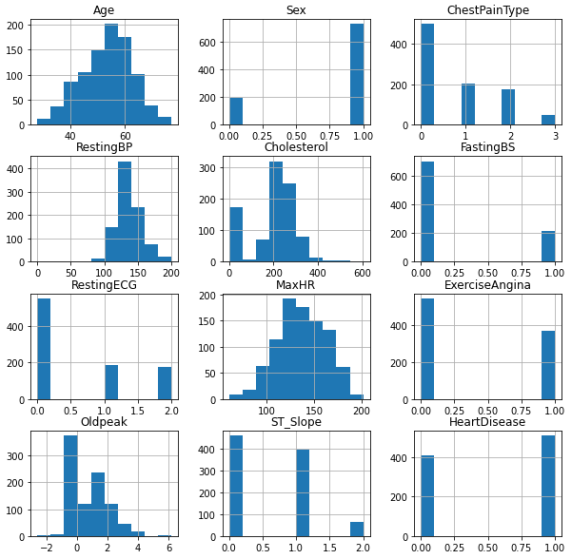

Figura 1: *Gráficos de distribución de variables.*

Destacando que en el gráfico se puede observar una distribución favorable (equilibrada) en los datos, particularmente hablando del número de observaciones de cada clase de la variable objetivo *HeartDisease.* Este tipo de distribución *normal* permite un análisis más eficaz ya que disminuye las probabilidades de presentar *overfitting*. Además, se realizó un análisis de correlación para observar qué tanto se relacionan todas las variables **independientes** entre sí y con la variable **dependiente**, *HeartDisease*. Se busca que las variables independientes no estén correlacionadas entre sí. De esta manera, podemos comprobar que todas las 11 variables restantes tienen un efecto significativo en la predicción de la variable *HeartDisease* y por ello, se utilizarán todas en los modelos.

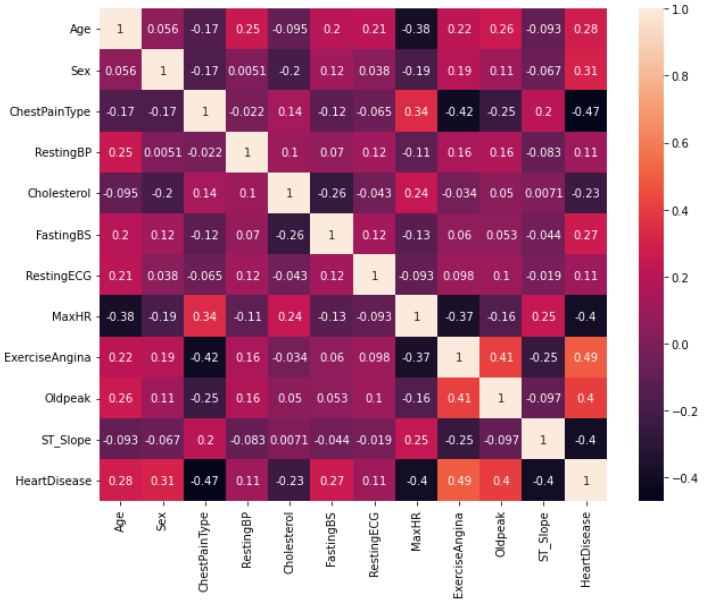

Figura 2: *Gráfico de análisis de correlación.*

Una vez llevada a cabo esta preparación de datos, se aplicaron los siguientes métodos de predicción:

- Métodos vistos en clase
    - Árbol de decisión.
    - Máquina de soporte vectorial (SVM).
    - Redes Neuronales (RNN).
- Métodos investigados
    - Regresión logística. Naomi Padilla Mora.
    - Random Forest. Sabrina Nicole Rodríguez Salgado.
    - K-Nearest Neighbors. Adrián Landaverde Nava.

Métodos de predicción

Nota: La preparación de los datos fue la misma para todos los algoritmos y sólo se hizo una vez. Dicha preparación se explica más arriba en este documento.

Liga al Google Colab con el código:

https://colab.research.google.com/drive/1NKG7-AEvCs0RUMWAezCDfAf3sRnzLIFa?usp=sharing

Métodos vistos en clase

Árbol de decisión

El árbol de decisión es un modelo predictivo que consiste en un mapeo de las observaciones sobre un elemento a las conclusiones sobre su valor objetivo. En las estructuras de árbol, tenemos las hojas, *leaves*, que representan clasificaciones o niveles, los nodos *nonleaf* son características y las ramas representan conjunciones de características que conducen a las clasificaciones (Tan, 2015).

a) Persona encargada del método: Adrián Landaverde Nava.

b) Evaluación del modelo entrenado

```
[[ 92  17]
[ 39 128]]
              precision    recall  f1-score   support

           0       0.70      0.84      0.77       109
           1       0.88      0.77      0.82       167

    accuracy                           0.80       276
   macro avg       0.79      0.81      0.79       276
weighted avg       0.81      0.80      0.80       276
```

El modelo presenta una *accuracy* de 0.87, mismo que indica que el modelo logra clasificar correctamente valores reales, tanto positivos como negativos. La matriz de confusión muestra 98 valores reales para la clase 0 y 141 valores reales para la clase 1. En cuanto a las métricas de *precision*, *recall* y *F1-score* (las cuales se calculan individualmente para cada clase y también en conjunto), el menor valor se encontró para la *precision* de la clase 0, y el valor más alto fue el de la *recall* para esta misma clase. Lo anterior permite ver que el modelo en general hace un buen trabajo en la clasificación. Los promedios de dichas medidas para ambas clases son de 0.

c) Conclusiones del método con la base de patrones utilizada.

- Particularidades del método con el dataset utilizado

  Este método, como su nombre lo indica, se basa en decisiones, por lo que es capaz de analizar el dataset de manera eficiente a pesar de tener muchas variables, lo cual implica que se trata de un análisis de muchas dimensiones. A pesar de no poder visualizarlo gráficamente, este método encuentra los patrones que crean tal clasificación de los datos.

- Comportamiento para predecir

Con este método se puede observar que este método funciona mejor para predecir valores positivos (1), mismos que corresponden a que el paciente presente una enfermedad cardíaca. Esto se puede observar a través del puntaje de *precision* de cada una de las variables, donde el de valor 1 es mayor que la otra clase. Sin embargo, dado que el *recall* de la clase 0 es mayor que el de la clase 1, este modelo podría mejorar, ya que lo que más interesa es poder identificar la mayor cantidad de valores de la clase 1, ya que puede prevenir más casos tener un valor falso de no tener una enfermedad cardíaca que tener un valor falso de sí tener una enfermedad cardíaca

Máquina de soporte vectorial (SVM)

El SVM pertenece al conjunto de métodos de aprendizaje supervisado. Al igual que otros métodos, es reconocido por realizar análisis de clasificación y de regresión a través del aprendizaje de un conjunto de datos y el reconocimiento de patrones. Una definición más precisa indicaría que una SVM construye un hiperplano o conjunto de hiperplanos para clasificar todas las entradas en un espacio de varias dimensiones o incluso, en un espacio infinito. El objetivo del SVM es maximizar el margen entre el hiperplano y los vectores de soporte, los cuales son los valores más cercanos al margen de clasificación (Gove, & Faytong, 2012).

a) Persona encargada del método: Sabrina Nicole Rodríguez Salgado.

b) Evaluación del modelo entrenado:

```
[[ 98  11]
 [ 26 141]]
              precision    recall  f1-score   support

           0       0.79      0.90      0.84       109
           1       0.93      0.84      0.88       167

    accuracy                           0.87       276
   macro avg       0.86      0.87      0.86       276
weighted avg       0.87      0.87      0.87       276
```

El modelo presenta una *accuracy* de 0.87, lo cual es indicador de qué tanto el modelo logra clasificar correctamente valores reales, tanto positivos como negativos. La matriz de confusión indica que de la clase 0 (ausencia de insuficiencia cardiaca) hubo 98 instancias detectadas correctamente, mientras que para la clase 1 (presencia de insuficiencia cardiaca) hubo 141 predicciones correctas. En cuanto a las métricas de *precision*, *recall* y *F1-score* (las cuales se calculan individualmente para cada clase y también en conjunto), el

menor valor se encontró para la *precision* de la clase 0, y el valor más alto fue el de la *recall* para esta misma clase. Lo anterior permite ver que el modelo en general hace un buen trabajo en la clasificación. Los promedios de dichas medidas para ambas clases se encuentran entre 0.86 y 0.87.

c) Conclusiones del método con la base de patrones utilizada.

- Particularidades del método con el dataset utilizado

El algoritmo de máquina de soporte vectorial busca generar un hiperplano lineal con la máxima distancia posible (margen) entre los datos. Al tratar de clasificar los datos en dos clases únicamente (0 y 1, ausencia y presencia de insuficiencia cardiaca, respectivamente), el SVM se vuelve un algoritmo ideal. Algo importante para poder aplicar dicho algoritmo es que los patrones en cuestión sean linealmente separables, y si no lo son, se debe usar un *kernel*, el cual transforma los datos a una dimensión mayor para poder encontrar un hiperplano lineal en este espacio. Para demostrar cómo cambian los resultados y la evaluación del modelo, se usó SVM simple (usando un parámetro de kernel lineal), y se usó también SVM con kernels polinomial de grado 2, gaussiano y sigmoide. El SVM simple fue el que mejores resultados obtuvo, lo cual puede atribuirse a que los datos sí eran linealmente separables en un inicio, por lo que la transformación con kernels no fue necesaria.

- Comportamiento para predecir

Como se mencionó previamente, se usaron kernels para ejemplificar las diferentes formas de aplicar SVM. El SVM simple (equivalente a usar un kernel lineal) fue el que obtuvo una mejor evaluación del modelo, con una *accuracy* de 0.87; mientras que para el kernel polinomial de grado 2 fue 0.73, para el gaussiano 0.72, y para el sigmoide 0.53. Definitivamente, el mejor modelo para clasificar fue el SVM simple y es probable que las transformaciones con kernel no hayan sido necesarias. En general, el SVM simple es lo suficientemente eficiente para clasificar en las clases 0 y 1.

Redes neuronales

Las redes neuronales se crearon con el objetivo de representar el proceso cerebral de reconocimiento y clasificación de patrones. Por lo tanto, es capaz de esto mismo, aprender y clasificar patrones. En la red neuronal se divide el espacio de vectores de entrada por un hiperplano. Si los cascos convexos de dos subconjuntos de entradas de entrenamiento no se cruzan, un hiperplano puede realizar la separación. Se puede obtener un hiperplano separador mediante los procedimientos de entrenamiento o mediante métodos de programación lineal (Nilsson, 1998).

a) Persona encargada del método: Naomi Padilla Mora.

b) Evaluación del modelo entrenado

```
Evaluating on training set…
```

```
21/21 [==============================] - 0s 2ms/step - loss:
0.3507 - accuracy: 0.8489

loss=0.3507, accuracy: 84.8910%

Evaluating on testing set…

9/9 [==============================] - 0s 2ms/step - loss: 0.3587
- accuracy: 0.8659

loss=0.3587, accuracy: 86.5942%
```

Como se observa, el *accuracy* obtenido en el método-prueba es de 86.5942, el cual es un gran puntaje y podemos interpretarlo como un buen modelo. Sin embargo, comparando los puntajes de entrenamiento y de test y siendo que el test tiene un mayor puntaje, esto puede indicar que el método está generando un poco de *overfitting,* lo cual implica que el método de adapta demasiado bien a este conjunto de datos pero puede disminuir su precisión al tratar de predecir con otra serie de datos. Afortunadamente, este *overfitting* es pequeño y se puede considerar casi normal.

c) Conclusiones del método con la base de patrones utilizada.

- Particularidades del método con el dataset utilizado

Como se mencionó anteriormente, una de las particularidades del método con el dataset utilizado es el grado de *overfitting* que presenta, aunque sea pequeño. Este es un factor que también se puede explicar con el batch_size que es el número de ejemplos de entrenamiento tomados que mientras mayor sea el tamaño del lote, más espacio de memoria necesitará. Por lo tanto, al realizar varias pruebas, decidimos utilizar un *batch_size = 200*, por el tamaño de la muestra y porque a mayor batch_size, menor era el puntaje en *accuracy* del modelo. Afectando la eficiencia del mismo.

- Comportamiento para predecir

Por los puntajes obtenidos, podemos afirmar que el comportamiento para predecir de las Redes neuronales es bastante bueno para el dataset. Por otro lado, teniendo en cuenta lo que se presentó sobre el *overfitting* e incluso la cuestión del impacto que está generando el *batch_size* se considera que no es el modelo más confiable de predicción.

Métodos de investigación

Regresión logística

La regresión logística, a pesar de su nombre, es un modelo de clasificación. Es un método simple y eficiente para problemas de clasificación binarios y lineales. Este modelo es muy

fácil de aplicar y genera un muy buen rendimiento con clases linealmente separables. En Scikit-learn hay una versión optimizada de *Regresión Logística*, con la cual es posible hacer la clasificación multiclase (Subasi, 2020).

a) Persona encargada del método: Naomi Padilla Mora.

b) Evaluación del modelo entrenado

```
[[ 97  12]
 [ 26 141]]
              precision    recall  f1-score   support

           0       0.79      0.89      0.84       109
           1       0.92      0.84      0.88       167

    accuracy                           0.86       276
   macro avg       0.86      0.87      0.86       276
weighted avg       0.87      0.86      0.86       276
```

El *accuracy* es de 0.86, este resultado denota que el modelo Regresión Logística tiene una calificación bastante buena para la predicción de la variable *HeartDisease* que expresa si el paciente sufre (clase 1) o no (clase 0) de una insuficiencia cardiaca. Observando la matriz de confusión podemos determinar que 97 casos de la clase 0 y 141 casos para la clase 1, fueron predichos de manera correcta. Lo que significa que 97 casos no presentan insuficiencia cardiaca, mientras que 141 casos sí presentan insuficiencia cardiaca. Además, 12 casos son falsos positivos a insuficiencia cardiaca y 26 casos son falsos negativos a este padecimiento. Por otra parte, observamos que la predicción de los casos de clase 1 tiene una *precision* mucho mayor que la de la clase 0, implicando que es más probable ser un falso positivo que un falso negativo. Por los puntajes obtenidos, podemos decir que la Regresión Logística nos ofrece un buen modelo de clasificación pero no el mejor.

c) Conclusiones del método con la base de patrones utilizada.

- Particularidades del método con el dataset utilizado

  Ya que con el dataset utilizado se nos plantea una clasificación binaria (clase 0 y clase 1) el método de Regresión Logística cuenta con la eficiencia necesaria requerida para dar una solución bastante acertada al tratarse de un método simple ya que, no se requiere trabajar en más dimensiones por ser clases linealmente separables.

- Comportamiento para predecir

  A pesar de que el método de Regresión Logística presenta un puntaje de prueba bastante alto se considera que no es el mejor método para la predicción del

padecimiento o ausencia de una insuficiencia cardiaca. Es muy destacable que para la predicción de la clase 1 (presenta insuficiencia cardiaca) el valor de precisión es bastante elevado, acertando en 141 casos de 167. Sin embargo, aún son 24 casos de falso negativo los cuales pueden sufrir grandes consecuencias al no ser detectado este padecimiento. Además, para la clase 0 (no presenta insuficiencia cardiaca) el nivel de precisión disminuye considerablemente obteniendo que 12 casos de 109 resultaron falsos positivos, lo cual, no es tan grave ya que en realidad son pacientes que no tienen un padecimiento que atente contra su salud pero que podrían verse perjudicados por tratamientos innecesarios.

Random forest

El clasificador de *Random Forest* consiste en un gran número de árboles de decisión individuales que operan en conjunto. Cada árbol llega a una predicción de clase y la clase con el mayor número de "votos" se convierte en la predicción de todo el modelo. El *Random Forest* es un modelo altamente efectivo porque los árboles individuales que lo conforman no están correlacionados, y cada uno de dichos árboles "diversifica" al otro y los errores de cada árbol se contrarrestan entre sí (Yiu, 2019). Adicionalmente, los árboles de decisión que conforman al "bosque" aprenden usando diferentes conjuntos de entrenamiento, así como diferentes variables (*features).*

a) Persona encargada del método: Sabrina Nicole Rodríguez Salgado.

b) Evaluación del modelo entrenado

```
[[ 91  18]
 [ 14 153]]
              precision    recall  f1-score   support

           0       0.87      0.83      0.85       109
           1       0.89      0.92      0.91       167

    accuracy                           0.88       276
   macro avg       0.88      0.88      0.88       276
weighted avg       0.88      0.88      0.88       276
```

Las métricas de evaluación para el *Random Forest* resultaron altas en general, con la menor de ellas siendo la *recall* para la clase 0: 0.83, y la métrica con mayor valor la *recall* para la clase 1. La *accuracy* de este modelo es 0.88, lo cual indica una muy buena capacidad del modelo para identificar correctamente valores reales en ambas clases. Las predicciones correctas para la clase 0 fueron 91, mientras que las predicciones correctas para la clase 0 fueron 153, con 14 y 18 predicciones erróneas para cada clase, respectivamente. Los promedios de la *precision,* la *recall* y el *F1-score* para las dos clases son todos 0.88, de nuevo indicador del buen funcionamiento del modelo.

c) Conclusiones del método con la base de patrones utilizada.

- Particularidades del método con el dataset utilizado

El *Random Forest* consiste en juntar varios árboles de decisión individuales y, con el factor aleatorio, tratar de evitar los errores y sesgos que presentaría un solo método de estos. Recordando que los árboles de decisión que conforman al bosque aleatorio aprenden empleando diferentes conjuntos de entrenamiento y diferentes variables (*features).* Con 918 observaciones y 11 *features*, es posible usar un bosque aleatorio con 100 árboles (en el código no se especifica, pero el algoritmo empleado genera 100 árboles por default) para clasificar. Ya que el método *Random Tree Classifier* tiene un atributo llamado "*feature_importances_*" o importancia de las variables, el cual da un valor de qué tan relevante es cada variable para determinar el *output* del modelo. De acuerdo con esto, las variables más importantes para la clasificación fueron *ST_Slope* (la pendiente del segmento ST del ejercicio máximo observada en el electrocardiograma del paciente), *ChestPainType* (tipo de dolor de pecho que presenta el paciente), y *ExerciseAngina* (si el paciente presenta una *Angina* inducida por el ejercicio).

Hablando de la variable con mayor importancia (*ST_Slope*), en electrocardiografía, el segmento ST en condiciones normales es plano en un electrocardiograma, pero si se hace esfuerzo físico, el segmento ST se eleva rápidamente (Myekg, s.f.), aumentando su pendiente. De hecho, sí hay una relación importante entre la capacidad aeróbica, la pendiente ST y el desarrollo de desenlaces cardiovasculares (García-Peña, et al., 2007). Por lo que los resultados dados por este algoritmo son consistentes con demás investigaciones médicas.

- Comportamiento para predecir

El *Random Forest* fue muy eficiente a la hora de clasificar, ya que el total de predicciones correctas para este método fue de 244 de 276 datos, lo que da la *accuracy* antes mencionada de 0.88. Estos resultados del bosque aleatorio han sido muy buenos, lo cual puede deberse a que usar este método es equivalente a emplear varios métodos "simples" de aprendizaje supervisado, en este caso, de árboles de decisión.

K-Nearest Neighbors (KNN)

K-nearest Neighbors (KNN, K Vecinos más cercanos) es un clasificador no paramétrico, el cual usa proximidad para hacer clasificaciones o predicciones sobre el agrupamiento de un dato individual (IBM, 2022). Para clasificación, se usa asumiendo que datos o puntos similares pueden encontrarse uno cerca de otro. KNN requiere determinar la forma en la se calcularán las distancias entre los puntos (puede ser con medidas como distancia euclidiana o distancia Manhattan, entre otras), así como definir *K,* el número de vecinos que se usará para determinar la clasificación de un dato en específico.

a) Persona encargada del método: Adrián Landaverde Nava.

b) Evaluación del modelo entrenado

```
[[ 75  34]
 [ 42 125]]
              precision    recall  f1-score   support

           0       0.64      0.69      0.66       109
           1       0.79      0.75      0.77       167

    accuracy                           0.72       276
   macro avg       0.71      0.72      0.72       276
weighted avg       0.73      0.72      0.73       276
```

El modelo presenta una *accuracy* de 0.72, mismo que indica que el modelo no es tan bueno para clasificar estos valores. La matriz de confusión muestra 75 valores reales para la clase 0 y 125 valores reales para la clase 1. En cuanto a las métricas de *precision*, *recall* y *F1-score*), el menor valor se encontró para la *precision* de la clase 0, y el valor más alto fue el de la *precision* para la clase 1. Lo anterior permite ver que el modelo en general no hace un buen trabajo en la clasificación. Los valores de dichas clases son de 0.71, 0.72 y 0.73

c) Conclusiones del método con la base de patrones utilizada.

- Particularidades del método con el dataset utilizado

  Este método lo que hace es buscar los vecinos más próximos, por lo que busca las observaciones que más se parezcan a la predicción a realizar. Este método tiene la facilidad de poder realizar una búsqueda multidimensional sin tantos problemas. Y para nuestro método, se tomó en cuenta los 5 vecinos más próximos, de tal manera que analiza las 5 observaciones cuyas variables más se parezcan y así realiza la predicción

- Comportamiento para predecir

  Con este método se puede observar que este método funciona mejor para predecir valores positivos (1), mismos que corresponden a que el paciente presente una enfermedad cardíaca. Esto se puede observar a través del puntaje de *precision* de cada una de las variables, donde el de valor 1 es mayor que la otra clase. Sin embargo, dado que el recall de la clase 0 es mayor que el de la clase 1, este modelo podría mejorar, ya que lo que más interesa es poder identificar la mayor cantidad de valores de la clase 1, ya que puede prevenir más casos tener un valor falso de no tener una enfermedad cardíaca que tener un valor falso de sí tener una enfermedad cardíaca. Aún así, de manera general, este método no es tan bueno porque sus valores para cada una de las métricas son relativamente bajos. Por lo tanto, no es tan bueno para predecir.

Comparación de todos los métodos implementados

El mejor método para clasificar fue el *Random Forest*, con la mayor *accuracy,* de 0.88; mientras que el algoritmo que peor se comportó fue K-Nearest Neighbors, con una *accuracy* de 0.72. El comportamiento de los algoritmos SVM simple y la red neuronal fueron parecidos, los dos fueron aproximadamente igual de eficientes en la clasificación, ambos con 0.87 de *accuracy* a dos cifras significativas. La regresión logística, algoritmo de clasificación para problemas binarios como el de este trabajo, tuvo una *accuracy* de 0.86, un valor cercano a los dos métodos mencionados antes. Los métodos restantes, árbol de decisión y el ya mencionado KNN, tuvieron los valores de *accuracy* más bajos y que más difirieron con el resto, con 0.80 y 0.72 cada uno. Además, el KNN no resultó tan eficiente como el resto de los modelos debido a los datos no se encuentran normalizados, ya que hay valores que se encuentran entre 0 a 2 o 0 a 1 y otras variables con valores de 0 a 200, por ejemplo, este método suele ser muy sensible a esta clase de distribución de datos. El rendimiento del árbol de decisión fue deficiente en comparación con el del *Random Forest,* lo que puede deberse al sobreajuste que puede llegar a presentar este modelo. Sin embargo, el bosque aleatorio está diseñado de manera que se disminuya el error individual de cada árbol y que éstos se diversifiquen entre sí. Luego, tiene sentido que el SVM y la red neuronal hayan tenido rendimientos similares, ya que el SVM es un tipo de red neuronal, pero con un enfoque ligeramente diferente.

Conclusiones

Posibles mejoras de cada método y en general

De manera general, una muy importante mejora es poder encontrar un dataset más grande, ya que, aunque se tiene una gran cantidad de observaciones, dado que también se tienen muchas variables, sería mejor tener muchos más datos para poder hacer un análisis más completo y robusto que pueda predecir de mejor manera. Asimismo, otra mejora de manera general es hacer una estandarización o normalización de los datos para que los modelos no tengan ningún sesgo sobre la distribución de los datos y todos estén en el mismo rango de valores.

En el método de Decision Tree, dado que pareciera ser que puede predecir mejor los valores para una clase mucho más que la otra, una posible mejora sería probar diferentes combinaciones de cantidad de ramas. Asimismo, dado que se tienen muchas variables y el resultado a predecir es muy disperso, también sería idóneo usar diferentes valores del número mínimo de hojas y de decisiones en una sola rama, esto con el fin de poder analizar más variables.

En el caso de Support Vector Machine, al usar los diferentes tipos de kernels, el kernel lineal claramente fue el que mejor rendimiento tuvo, lo que posiblemente indique que los datos fueron linealmente separables desde un principio. Además, el resto de los algoritmos implementados también funcionaron correctamente. De cualquier manera, una posible mejora a este método podría hacer un análisis más cuidadoso de los datos en este aspecto

y tal vez visualizar de otras formas los efectos de los kernels no lineales. Esto permitiría entender mejor por qué algunos tienen un mejor comportamiento que otros.

Las mejoras que se pueden realizar en para el método de Redes neuronales son, familiarizarse mejor con sus componentes y la estructura del código, como se mencionó anteriormente, para definir el *batch_size* se realizaron varias pruebas y así se identificó el comportamiento. Sin embargo, hay algunos otros parámetros de los cuales no se tiene la total comprensión de por qué definirlos con esos valores, como por ejemplo, los valores asignados a los hiper parámetros. De igual forma, investigar la forma de implementar en la cuestión de código, una matriz de confusión pudo haber enriquecido el análisis del método. Con esto, los resultados podrían ser más precisos.

Para el método de Regresión Logística podría resultar benéfico probar con diversos puntajes para el *split* de los datos, ya que al entrenarlo de manera diferente, el puntaje de test puede mejorar o empeorar, por ende, se tendrían que hacer varias pruebas, aunque la combinación utilizada para este análisis es una de las más recomendadas, 30% para el entrenamiento y 70% para la prueba. Otro factor es el *random_state,* probarlo con diversas semillas que mantienen el split con los mismos porcentajes pero va cambiando de manera aleatoria la selección de estos datos. Incluso, el *random_state* ofrece más parámetros que valdrían la pena investigar ya que ayuda a realizar una selección de datos más específica acorde al objetivo. Remarcando que, la regresión logística fue uno de los modelos con mejor puntaje para la predicción de la clase 1, pero con un puntaje bastante deficiente en la predicción de la clase 0.

En el caso de *Random Forest*, una mejora posible podría ser haber implementado más árboles en el bosque aleatorio. Ya que los bosques aleatorios toman subconjuntos de datos para trabajar, así como variables diferentes, mientras más árboles hay en un bosque aleatorio, más posibilidades se cubren para distintas observaciones. Si bien el número de árboles que se usó (100) funcionó correctamente, y no necesariamente más árboles resulta en algo mejor, hubiera estado bien experimentar más con esta parte, o investigar más acerca de los efectos del número de árboles.

Para K-Nearest Neighbours, una de las mejoras de este método es que hay que tener una mayor consideración de la cantidad de vecinos cercanos, esto porque en cada base de datos, este valor puede variar, y puede ser mejor o peor el incrementar o disminuir este número. Asimismo, se tiene que considerar una normalización o estandarización de los datos para poder conseguir mejores resultados, ya que no todas las variables eran binarias, por lo que el rango de números es muy variable, por lo que los resultados pueden estar sesgados por esta diferencia en el rango de los valores de todas las variables.

Conclusiones generales

El mejor método, como se menciona anteriormente, fue el *Random Forest,* cuyas métricas de evaluación fueron las mejores. Este método tuvo la *accuracy* más alta, de 0.88, y por lo tanto, el número de predicciones correctas que hizo fue el más alto de todos los métodos. El buen rendimiento del bosque aleatorio se puede atribuir a que emplear este método es equivalente a usar varios métodos regulares de aprendizaje supervisado, específicamente de árboles de decisión.

Conclusiones individuales

Adrián Landaverde Nava

Los métodos que yo implementé fue el de Decision Tree y K-Nearest Neighbours. Al aplicar estos 2 métodos me sorprendió mucho descubrir que el KNN tuvo un desempeño mucho peor que de Decision Tree, e incluso que todos los demás métodos, ya que el KNN al buscar los vecinos más próximos en múltiples dimensiones puede hacer un análisis más extenso, sin embargo, no fue así. Por lo tanto, el árbol de decisión tuvo un desempeño muy bueno al compararlo con los otros métodos. Asimismo, un aspecto que no me gustó inicialmente del proyecto, fue que teníamos más de 3 variables, por lo que no sería posible graficar las variables. Sin embargo, una vez que decidimos reducir las dimensiones de las variables al realizar un algoritmo de Principal Component Analysis (PCA), fue posible graficar estos datos y así poder visualizar una versión simplificada del dataset. Por lo tanto, esto nos permite poder tener un entendimiento más profundo sobre la clasificación de los datos.

Asimismo, al hacer el gráfico de la comparación de cada par de variables, se pudo ver que los datos no están completamente separados, por lo que estos métodos de Machine Learning son muy necesarios para poder encontrar patrones dentro de los datos. Es por esto que a partir de los métodos que empleamos fue posible generar un buen modelo que pueda predecir estos valores, ya que a simple vista no es posible identificar tendencias absolutas o que siempre se cumplan. Por lo tanto, usar este tipo de algoritmos son muy eficientes para encontrar este tipo de resultados.

Finalmente, otra posible área de oportunidad es que tengamos una base de datos más grade para poder tener una mayor cantidad de observaciones que puedan ser usadas como datos de entrenamiento y así poder obtener mejores resultados que generalicen estos patrones y así poder predecir de una mejor manera la condición cardíaca de las personas.

Naomi Padilla Mora

Para el desarrollo de este análisis decidí implementar los métodos de *Redes neuronales* y *Regresión Logística*, el primero porque suele ser uno de los métodos más reconocidos por su eficiencia y precisión y el segundo, porque al tratarse de un método sencillo se adapta muy bien al objetivo, que es la clasificación de clases binarias. De ambos métodos, el de Redes neuronales fue el que mejor puntaje de predicción obtuvo, incluso, se aproxima bastante al puntaje del *Random Forest,* que se concluyó fue el mejor método de clasificación para el dataset. Incluso, podemos observar su gran similitud con el SVM, ya que son métodos bastante similares. Sin embargo, me gustaría destacar que para el método de Regresión Logística se pudo hacer un análisis un poco más amplio y detallado por la matriz de confusión y que esto me benefició para comprender más la situación y generar un análisis más específico. Considero que aplicar la matriz de confusión puede proporcionar información muy valiosa que explica la razón de los puntajes como el *accuracy.*

Considero que es muy enriquecedor para el análisis someter el dataset a diversos métodos de clasificación. Esto me permitió no solo tener un modelo más preciso de predicción, sino también me permitió familiarizarme con más métodos y conocer cómo es que estos funcionan y para casos futuros, poder identificar de manera más sencilla que método se puede utilizar dependiendo del dataset sin tener que realizar tantas pruebas. De igual forma, pude aprender más sobre el método de Redes neuronales, el cual considero que es uno de los más complejos, cuestión que se puede ver claramente en el desarrollo de código, pero también es uno de los más robustos y con mayor potencial para distintos tipos de soluciones.

Algo que me gustó mucho del proyecto fue que se nos dió la facilidad de elegir los métodos a implementar ya que, al contar con esta libertad realicé mayor investigación y siento que me involucré un poco más en la comprensión de los mismos métodos. Además, me sentí con la libertad para desarrollar elementos de visualización que apoyaran a la comprensión de las soluciones, permitiendo también aprender más sobre las librerías e instrucciones en Python. Finalmente, lo que menos me gustó fue que este análisis se concentró principalmente en la eficiencia de los modelos pero que no se generó un análisis sobre el impacto médico que estos modelos pueden tener y lo que implican sus resultados. Considero que sería interesante poner a prueba uno de estos modelos con otra base de datos, para así, analizar más resultados.

Sabrina Nicole Rodríguez Salgado

Los métodos que yo implementé fueron Máquina de Soporte Vectorial (SVM) y *Random Forest.* En el caso de SVM me pareció adecuado incluir las diversas formas en las que se puede implementar dicho método, específicamente con el uso de kernels, los cuales se usan para transformar datos que no son linealmente separables, aunque dichos kernels no fueran completamente necesarios. Al final, el SVM de mejor rendimiento fue el SVM simple, es decir, el equivalente a emplear un kernel “lineal”. En comparación con el resto de los algoritmos, el SVM tuvo un rendimiento similar al de la red neuronal, y esto coincide con el hecho de que son métodos similares, el SVM es un tipo de red neuronal. Este primer método tuvo una *accuracy* de 0.87, un valor alto.

Sin embargo, el mejor método, no sólo de los que implementé yo, sino de todos los métodos usados, fue el *Random Forest*, con una *accuracy* de 0.88. La manera en la que está diseñado el bosque aleatorio es para contrarrestar los errores cometidos por cada árbol individual en el bosque y al mismo tiempo generar varias “opciones” distintas que permitan llegar a un resultado de clasificación de diferentes formas. El factor aleatorio es lo que hace tan poderoso al *Random Forest*, pues toma los resultados de muchos árboles de decisión individuales, distintos, aleatorios y no correlacionados entre sí y junta dichos resultados para llegar a un resultado más general y confiable. Probablemente es por esto que este algoritmo tuvo el mayor número de predicciones correctas. Con este algoritmo también se descubrieron las variables con mayor importancia para clasificar a un paciente como con o sin insuficiencia cardiaca, dichas variables fueron *ST_Slope*, *ChestPainType* y *ExerciseAngina.*

En este proyecto aprendí sobre el funcionamiento de varios métodos de aprendizaje supervisado y sobre las ventajas y desventajas de cada uno. Aprendí también que al usar aprendizaje de máquina, una gran parte del proceso siempre queda a criterio de quien lleva a cabo el análisis. Me pareció interesante aplicar todos los algoritmos al mismo tiempo, al mismo conjunto de datos, para ver el efecto real del diseño de cada método en la clasificación. Aprendí a interpretar mejor la matriz de confusión y métricas como la *precision* y la *recall,* y cómo todas estas medidas se relacionan entre sí. Lo que más me gustó del proyecto es que tuvimos bastante libertad a la hora de desarrollarlo, pudimos elegir el conjunto de datos a utilizar, la forma en la que implementamos los métodos a la hora de programar y qué otras cosas podíamos agregarle al análisis. Lo que menos me gustó fue que no hubo mucha más investigación médica a raíz de los resultados obtenidos, esto hubiera permitido darle más interpretación a éstos, por ejemplo, hablando de las variables con mayor importancia a la hora de clasificar. Me parece que esta falta de investigación posterior podría ser una posible mejora del proyecto que haría más aplicables los resultados en la vida real.

Referencias

Datahacker.rs. (2019, 26 octubre). #K An implementation of a Shallow Neural Network in Keras - Circles dataset. Master Data Science. https://datahacker.rs/how-to-implement-shallow-neural-network-keras/

Dernoncourt, F. (2015, 22 mayo). What is batch size in neural network? Cross Validated. Recuperado 2 de junio de 2022, de https://stats.stackexchange.com/questions/153531/what-is-batch-size-in-neural-network

García-Peña, et al. (2007). *Relationship of ST segment/heart rate slope index and ST segment change index scores on the heart rate change during conventional stress test with the presentation of new cardiovascular events.* Recuperado de https://doi.org/10.1016/j.rccar.2016.09.007

Gove, R., Faytong, J. (2012). *Advances in computers.* Science Direct. Recuperado de https://www.sciencedirect.com/topics/computer-science/support-vector-machine

IBM. (2022). *What is the k-nearest neighbors algorithm?* Recuperado de https://www.ibm.com/topics/knn

Mayo Clinic. (2022). *Heart failure.* Recuperado de https://www.mayoclinic.org/diseases-conditions/heart-failure/symptoms-causes/syc-20373142

Myekg. (s.f.). *Valoración del segmento ST.* Recuperado de https://www.my-ekg.com/como-leer-ekg/segmento-st.html

Nilsson, N. J. (1998). *Neural Networks.* Science Direct. Recuperado de https://www.sciencedirect.com/topics/computer-science/neural-networks

Subasi, A. (2020). *Logistic Regression.* Science Direct. Recuperado de https://www.sciencedirect.com/topics/computer-science/logistic-regression

Tan, L. (2015). *The Art and Science of Analyzing Software Data*. Science Direct. Recuperado de https://www.sciencedirect.com/topics/computer-science/decision-tree

World Health Organization. (2020). *The top 10 causes of death.* Recuperado de https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death

Yiu, T. (2019). *Understanding Random Forest*. Towards Data Science. Recuperado de https://towardsdatascience.com/understanding-random-forest-58381e0602d2