Instituto Tecnológico de Estudios Superiores de Monterrey

Reto - Titanic

Inteligencia artificial avanzada para ciencia de datos

Grupo 102

Equipo 2

Luis Gabriel Martínez Rentería	A01651812
Miguel Alejandro Salas Reyna	A00827219
Héctor Francisco Marin Garrido	A00827714
Emilio Fernando Prado Chible	A01570318

Resumen

“El hundimiento del Titanic es uno de los naufragios más infames de la historia.

El 15 de abril de 1912, durante su viaje inaugural, el RMS Titanic, ampliamente considerado como "insumergible", se hundió después de chocar con un iceberg. Desafortunadamente, no había suficientes botes salvavidas para todos a bordo, lo que resultó en la muerte de 1502 de los 2224 pasajeros y tripulantes.

Si bien hubo algún elemento de suerte involucrado en la supervivencia, parece que algunos grupos de personas tenían más probabilidades de sobrevivir que otros.

En este desafío, le pedimos que construya un modelo predictivo que responda a la pregunta: "¿Qué tipo de personas tenían más probabilidades de sobrevivir?" utilizando datos de pasajeros (es decir, nombre, edad, sexo, clase socioeconómica, etc.)”.

Información obtenida de Kaggle.

Introducción (Datos)

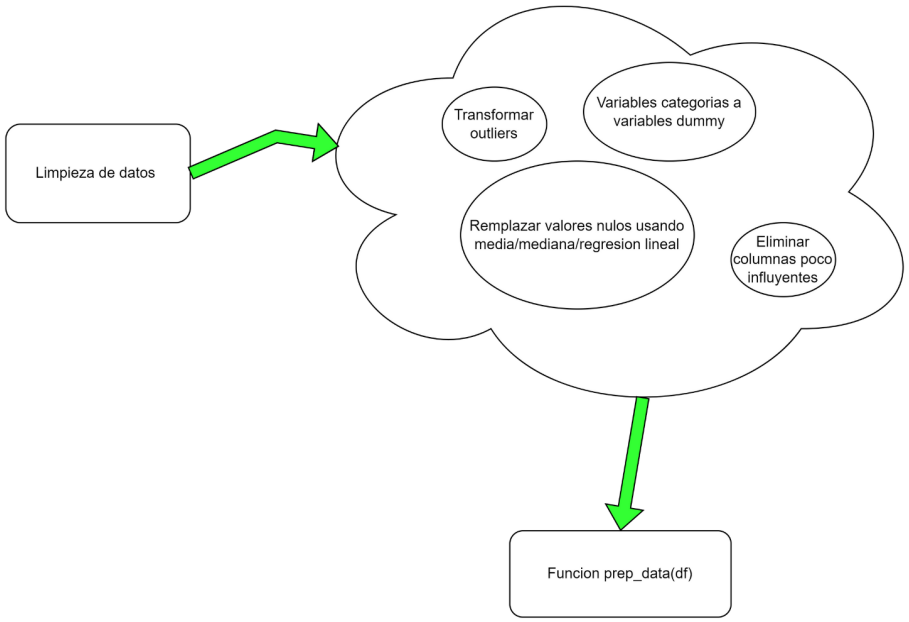
Para la solución del reto planteado, lo primero que tenemos que hacer es conocer el problema. En este caso se nos presenta el reto más famoso de Kaggle y uno de los favoritos para introducirse a la ciencia de datos, ya que con este datasets podemos hacer manipulación de datos sencilla que nos permita conocer los pasos para desarrollar una solución mediante ciencia de datos. Además que nos permite conocer modelos sencillos de clasificación como lo es la regresión logística.

Los datos presentados en este reto se clasifican en las siguientes variables:

Variable	Definition	Key
survival	Survival	0 = No, 1 = Yes
pclass	Ticket class	1 = 1st, 2 = 2nd, 3 = 3rd
sex	Sex	
Age	Age in years	

sibsp	# of siblings / spouses aboard the Titanic	
parch	# of parents / children aboard the Titanic	
ticket	Ticket number	
fare	Passenger fare	
cabin	Cabin number	
embarked	Port of Embarkation	C = Cherbourg, Q = Queenstown, S = Southampton

Una vez que entendimos cada variable, el siguiente paso fue realizar las manipulaciones correspondientes para la limpieza de datos. De esta manera utilizamos una función que nos permitía manipular el dataset completo, el proceso se muestra en la imagen siguiente.

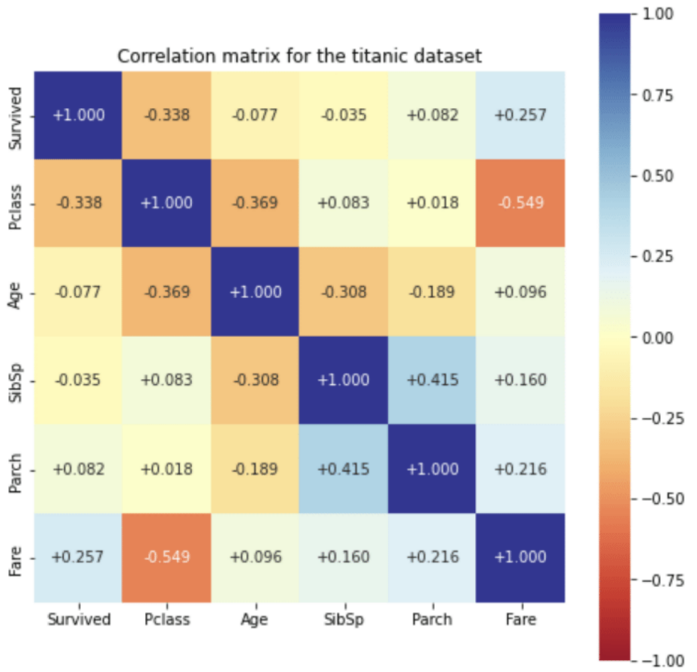


El procedimiento consistió en reemplazar valores nulos con el uso de la media, la mediana o mediante la implementación de una regresión lineal. Los outliers se eliminaron, se utilizaron variables dummies para variables categóricas y eliminamos columnas que no tenían un valor considerable para el desarrollo del modelo.

Female	Male	Embarked_C	Embarked_Q	Embarked_S
0.0	1.0	0.0	0.0	1.0
1.0	0.0	1.0	0.0	0.0
1.0	0.0	0.0	0.0	1.0
1.0	0.0	0.0	0.0	1.0
0.0	1.0	0.0	0.0	1.0

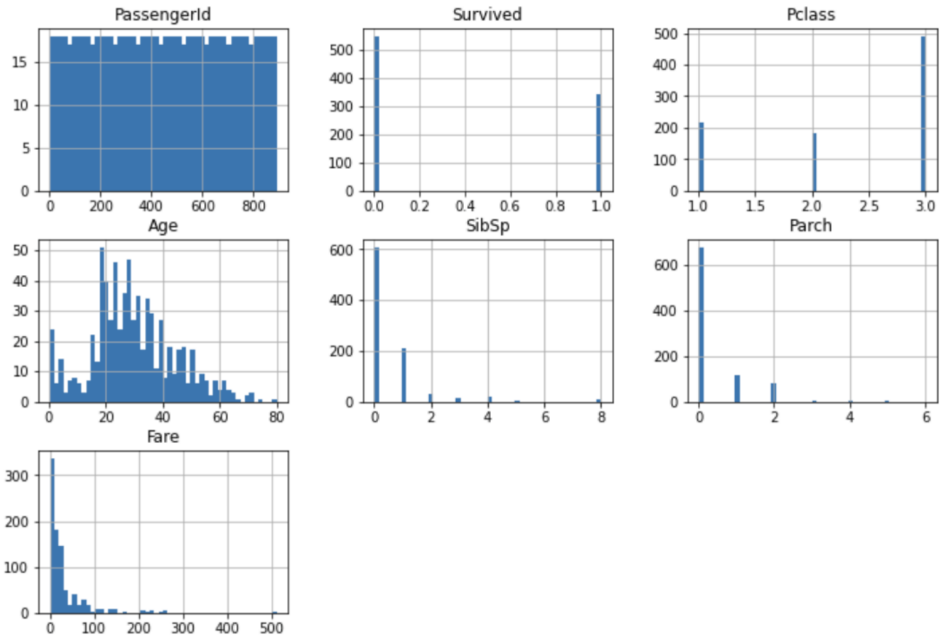
Imagen: Valores Dummies

En el caso de los valores nulos, la columna Embarked contenía dos, los cuales fueron sustituidos por la letra S que representa el puerto de Southampton, ya que se investigó el nombre de la persona y se encontró información histórica que nos permitió determinar el puerto por el que abordó. Para el caso de Age, se encontró una alta correlación entre la edad y la clase, por lo que se usó la media por clase y sexo para llenar los valores nulos. De esta manera, formamos grupos que nos permitió diversificar los valores obtenidos y no quedarnos con una media general.



Una vez que solucionamos los problemas de la limpieza, nos enfocamos en realizar una análisis de nuestras variables. Encontramos la correlación entre ellas, y nos dimos cuenta qué variables tienen altas influencias entre sí.

Como se muestra en las gráficas siguientes, podemos observar ciertas distribuciones de las variables. Algunas de las variables categóricas se aprecian a simple vista, la variable de Age también destaca a simple vista ya que es lo más cercano a una distribución normal, sin embargo la normalidad no es clara, ya que presenta cierta inclinación hacia la izquierda y una alta concentración en edades de adultos jóvenes. La variable Fare se concentra altamente a la izquierda en valores menores a 50.



Análisis de los resultados (Modelo, Evaluación y Refinamiento)

Para la selección del modelo se hicieron varias pruebas con diferentes métodos de limpieza y diferentes modelos, usando las librerías de Scikit-learn y Keras (keras para redes neuronales y sklearn para otros modelos).

Primero se usaron los modelos con una limpieza sencilla, en la que al final de la limpieza teníamos 8 columnas con las siguientes variables:

- Survived
- Pclass: Clase en la que viajó la persona
- Sex: Sexo de la persona
- Age: Edad de la persona
- Fare: Precio del boleto de la persona
- Embarked: Puerto en el que embarcó la persona
- Title: Título de la persona (Se ecnontraba en su nombre)
- FamilySize: Total de miembros de la familia de la persona a bordo

Se hizo un escalado MinMax para las variables “Age” y “Fare” y se separó el conjunto de valores de la base de datos de entrenamiento en 2 conjuntos, conjunto de entrenamiento y

conjunto de prueba, ya que solo de estas conocíamos los valores de Survived correctos y podríamos validar correctamente la precisión de los modelos entrenados.

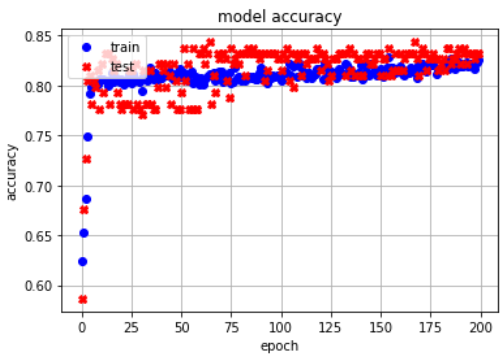
Se entrenaron los siguientes modelos de machine learning usando el conjunto de entrenamiento y usando el conjunto de prueba para validar que nos estuviera regresando una precisión correcta.

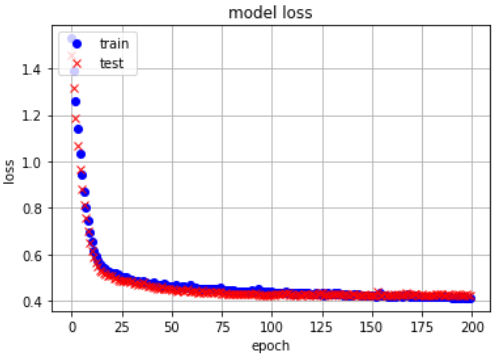
	Model	Score
3	Random Forest	0.826816
0	Support Vector Machines	0.821229
1	KNN	0.810056
2	Logistic Regression	0.782123
4	Decision Tree	0.776536

También se entrenó una red neuronal con la siguiente estructura:

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 dense (Dense)                (None, 8)                 64

 dense_1 (Dense)              (None, 16)                144

 dense_2 (Dense)              (None, 32)                544

 dense_3 (Dense)              (None, 16)                528

 dense_4 (Dense)              (None, 8)                 136

 dense_5 (Dense)              (None, 1)                 9

=================================================================
Total params: 1,425
Trainable params: 1,425
Non-trainable params: 0
_________________________________________________________________
```

Esta red nos regresó un accuracy de 0.8324, por lo que fue el modelo que nos regresó la mejor precisión de todos los modelos entrenados con este conjunto, teniendo las siguientes curvas de precisión y pérdida:





Lo siguiente fue probar los modelos que nos dieron las mejores precisiones con el set de prueba que nos da Kaggle para que lo subamos a la plataforma y veamos el score real de los modelos.
Los resultados en la plataforma de Kaggle fueron los siguientes:

- Random Forest: 0.75837
- KNN: 0.74880
- Support Vector Machine: 0.77272
- Neural Network: 0.76794

No nos sentimos contentos con estos resultados por lo que probamos con otra limpieza un poco más compleja y nos quedamos probando solo modelos de redes neuronales.

NOTA: Todo lo mencionado anteriormente de modelaje se puede encontrar en el siguiente notebook:
https://colab.research.google.com/drive/1WlsZKU1aYyF6zU5b7bc65NUe5FpCzUaX?usp=sharing

Se hizo una nueva limpieza con la que se mantuvieron 9 variables de columnas:

- Survived
- Pclass: Clase en la que viajó la persona
- Sex: Sexo de la persona
- Age: Edad de la persona
- Fare: Precio del boleto de la persona
- Embarked: Puerto en el que embarcó la persona
- Title: Título de la persona (Se encontraba en su nombre)
- SibSp: Total de hermanos y esposos/esposas a bordo
- Parch: Total de padres e hijos a bordo

Se hizo el mismo escalado MinMax a las variables Age, Fare, SibSp y Parch y se configuró una red neuronal con las siguientes características:

```
model = Sequential()
model.add(Dense(16, input_shape=(8,), activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(4, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 16)                144

 dense_1 (Dense)             (None, 32)                544

 dense_2 (Dense)             (None, 16)                528

 dense_3 (Dense)             (None, 4)                 68

 dense_4 (Dense)             (None, 1)                 5

=================================================================
```

El accuracy de este modelo fue medido de manera incorrecta ya que estuvo usando un dataset diferente del correcto, por lo que los resultados de accuracy salían mucho más elevados de lo que deberían de haber salido. De igual manera se subió una predicción hecha con esta red a la plataforma de Kaggle y la precisión en la plataforma fue de 0.78229, por lo que salió mejor que las otras predicciones hechas.

NOTA: Todo lo mencionado anteriormente de modelaje se puede encontrar en el siguiente notebook:
https://colab.research.google.com/drive/1QkX9wdzCWXPvEtMJWxxA8aWTaHr6jane?usp=sharing

Aun así se trabajaron otros modelos con las siguientes características:

El siguiente modelo utilizado fue construido utilizando una librería llamada Keras, la cual usa tensorflow por debajo, con esta librería armamos un modelo de 6 capas. Este mismo modelo utiliza las funciones de activación relu en sus 5 de sus 6 capas y en la capa final se utilizó la función sigmoid para obtener las predicciones entre 0 y 1. A continuacion el codigo del modelo inicial:

```
# Model 0 Sin Optimizar
model = Sequential()
model.add(Dense(20, activation = 'relu', input_shape=(10,)))
model.add(Dense(40, activation = 'relu'))
model.add(Dense(20, activation = 'relu'))
model.add(Dense(10, activation = 'relu'))
model.add(Dense(5, activation = 'relu'))
model.add(Dense(1, activation = 'sigmoid'))
model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics=['accuracy'])
```

Este modelo fue comparado con otros modelos implementados en sklearn como el de Logistic Regression o Random Forest Decision Tree, sin embargo decidimos quedarnos con este modelo ya que no solo fue el que nos dio mejores resultados desde un inicio, si no que tambien teniamos la posibilidad de controlar a más bajo nivel nuestra red de neuronas de una manera que no está disponible en las implementaciones de sklearn.

También cabe mencionar que para este modelo se utilizaron las siguientes variables:

- Survived
- Pclass: Clase en la que viajó la persona
- Sex: Sexo de la persona
- Age: Edad de la persona (Los valores faltantes fueron llenados utilizando Regresión Lineal)
- Fare: Precio del boleto de la persona (Los valores faltantes fueron llenados utilizando la agrupación de Pclass y Sex y en base a esto sacando la mediana para cada grupo.)
- Embarked: Puerto en el que embarcó la persona
- SibSp: Total de hermanos y esposos/esposas a bordo
- Parch: Total de padres e hijos a bordo

Se hizo el mismo escalado MinMax a las variables Age, Fare, SibSp y Parch y se convirtieron en variables dummies las columnas Sex y Embarked.

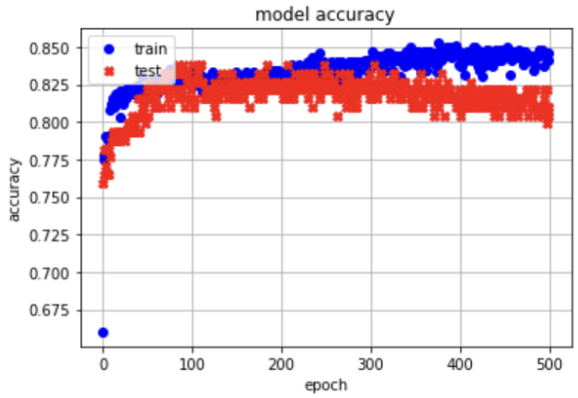
Para la etapa de evaluación, utilizamos varias métricas y visualizaciones como evaluaciones del modelo en train y test mostrando el loss y accuracy en cada uno de los dataset.

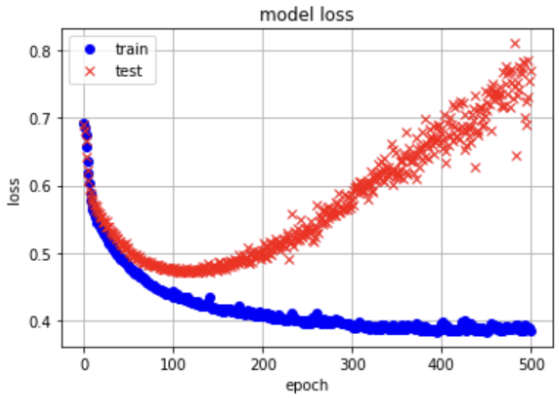
Ejemplo a continuación:

```
6/6 [==============================] - 0s 4ms/step - loss: 0.4410 - accuracy: 0.8045
score on test: 0.8044692873954773
23/23 [==============================] - 0s 2ms/step - loss: 0.3399 - accuracy: 0.8596
score on train: 0.8595505356788635
```

A su vez hicimos usos de gráficos para lograr visualizar estas mismas métricas en cada punto del entrenamiento al graficar la pérdida y la precisión en contra de las épocas.

Ejemplo a continuación:





Gracias a las anteriores métricas fuimos capaces de ir ajustando los diferentes hiperparametros como el número de épocas, el batch size, número de capas, entre otros… Con esto fuimos mejorando nuestro accuracy medido tanto en entrenamiento como en el momento de realizar la predicción contra el dataset de Kaggle.

Con este procedimiento un tanto empírico de prueba y error, fue que fuimos optimizando el modelo al cambiar las dimensiones del output en las capas densas, agregando o removiendo datos con la finalidad de mejorar la precisión del modelo, agregar capas de Dropout para lograr reducir el overfit y lograr aumentar las épocas de manera efectiva para lograr una mayor precisión así como agregar regularizados en cada una de las capas para lograr optimizar el modelo y obtener mayor precisión.

A continuación una imagen del modelo final ya optimizado:

```
# Best Model 0.79665
model = Sequential()
model.add(Dense(20, activation = 'relu', input_shape=(10,), kernel_regularizer=regularizers.l2(0.0001)))
model.add(Dropout(0.2))
model.add(Dense(40, activation = 'relu', kernel_regularizer=regularizers.l2(0.0001)))
model.add(Dropout(0.2))
model.add(Dense(20, activation = 'relu', kernel_regularizer=regularizers.l2(0.0001)))
model.add(Dense(10, activation = 'relu', kernel_regularizer=regularizers.l2(0.0001)))
model.add(Dense(5, activation = 'relu', kernel_regularizer=regularizers.l2(0.0001)))
model.add(Dense(1, activation = 'sigmoid'))
model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics=['accuracy'])
```

Con este último modelo terminamos consiguiendo un score de 0.79665 tal y como se muestra en la siguiente imagen:

NOTA: La información de este modelo asi como el trabajo realizado puede ser encontrada en la siguiente libreta
https://colab.research.google.com/drive/1PtpXafKCJGptShKguAPi7BYJZC4n9XdW?usp=sharing

Conclusión (Solución).
Para atacar esta situación problema se hizo uso de varios modelos y métodos para predecir si los individuos sobrevivieron o no al desastre del titanic, si bien en un principio consideramos que el mejor modelo fue el de Support Machine Vector no nos sentimos satisfechos con los resultados obtenidos por lo que hicimos uso de una red neuronal de 6 capas y mejoramos los métodos utilizados para realizar la limpieza de los datos, con este modelo conseguimos mejores resultados en la plataforma de kaggle pasando de una precisión de 0.77 con el modelo anterior a una de 0.78 con nuestra red, finalmente armamos un modelo de 6 capas con una función sigmoid en la capa final para obtener valores de entre 1 y 0 (vivo o muerto). Este último modelo no solo daba mejores resultados pero también nos dio más control que los modelos de la librería sklearn y finalmente después de modificar los hiperparametros obtuvimos una precisión de 0.79 en kaggle siendo este el valor más alto que conseguimos y nuestro modelo definitivo.