Preparación de datos y modelado para el dataset Titanic

1) Importar librerías y paquetes

Además de hablar de las librerías y paquetes, me pareció interesante destacar ciertos conceptos que no fueron presentados en la clase y quizás alguna posible mejora para alguna parte del código en el caso de manejar datasets más pesados y más cómputo-dependientes.

Inicialmente, se importan las librerías necesarias como numpy y pandas para cálculos matemáticos y estructuras de datos además de matplotlib y seaborn para la parte de visualización de datos y gráficas.

Luego, se importan varias funcionalidades/modelos de la librería scikit learn. Los modelos importados y una breve explicación de ellos es la siguiente.

SVC, Linear SVC son algoritmos del tipo Support vector machines, los cuales buscan definir una recta, plano o hiperplano (hablando de un SVM lineal) tal que la distancia entre esta recta y el punto más cercano de cada clase sea máxima.

Dentro de los **algoritmos de árboles o ensembles**, se utilizan un decisionTreeClassifier, un randomForestClassifier y XGBoost. El decisionTreeClassifier construye un único árbol y lo utiliza para hacer las predicciones. El random forest construye múltiples árboles armados utilizando Bootstrap por ejemplo y luego al realizar una predicción se le da un peso al resultado de cada árbol del bosque y se decide qué predice el bosque (juntando las predicciones de cada uno y sus pesos). Por último, XGBoost utiliza una forma diferente de armar los árboles dado que son dependientes uno de otro, intentando mejorar las métricas con cada árbol nuevo. Como no es obligatoria la parte de modelado y predicción y por la dificultad extra de instalar XGBoost, el código para la creación del modelo utilizando este algoritmo se comenta en la solución incluída.

También se utiliza la técnica de Gridsearch para tunear los hiperparámetros de los distintos algoritmos. Una mejor alternativa para un caso computacionalmente complejo sería utilizar randomsearch para ubicar los puntos de interés donde es muy probable encontrar óptimos en la cercanía y aplicar gridsearch en espacios pequeños cercanos a dichos puntos de interés.

2) Cargar el dataset y mostrarlo

Para realizar estas dos tareas, se utilizan las funcionalidades de pandas llamadas read_csv para leer un archivo csv y cargarlo utilizando la estructura de datos llamada dataframe y head como forma de imprimir las primeras x filas (5 por defecto) del dataset de una forma prolija.

3) Gestionar los valores faltantes

Se utiliza la función pd.isnull para identificar los valores faltantes de cada columna. Se dropean las columnas Cabin y ticket por la gran cantidad de valores faltantes.

Se observa una distribución relativamente parecida a una normal para el atributo age, pero con un leve sesgo hacia la derecha, lo que determinará la estrategia de imputación utilizada. Debido a este sesgo, elegir la mediana es una mejor alternativa que elegir la media. Si no tuviéramos este sesgo, la estrategia de imputación elegida sería la media.

Para el atributo fare, la distribución está fuertemente sesgada hacia la derecha, por lo que se elige la mediana como valor a imputar. Alternativamente, se podría elegir aplicar una transformación a los datos para que su distribución se asemeje más a una normal. 2 estrategias para esto podrían ser aplicar el logaritmo natural a los valores del atributo o alguna de las variantes de la transformación "Box and Cox" (1964) presentada en applied-predictive-modeling, presente en la bibliografía del curso.

4) Graficar los datos

Se grafican varios atributos en relación al atributo a predecir para formar unas hipótesis iniciales sobre si algún atributo está correlacionado con la variable a predecir. Las librerías para hacer esto son seaborn y matplotlib.pyplot. Aquí se pueden observar ratios como la proporción de mujeres que sobrevivieron o indicios como qué clase es la que presentó más proporción de personas que sobrevivieron.

5) Feature Engineering

En la parte de feature engineering, lo primero que se hace es encodear los atributos categóricos a valores enteros. Se podría realizar mediante one hot encoding pero en este notebok, se decide no hacerlo.

Una hipótesis muy interesante provista por Samson Qian (creador del notebook en kaggle) es que el titulo de la persona puede influir en si esta sobrevivió o no. Desde mi punto de vista personal, quizás esto podría estar agregando un atributo fuertemente correlacionado con Sex. Luego de calcular la matriz de correlación, se ve que la correlación es muy baja, lo que no me parece intuitivo, dado que la gran mayoría de los ejemplos de títulos (más del 90%) son Mr, Mrs y Miss lols cuales se podrían relacionar directamente con el género.

(Imagen de correlación provista debajo)

	PassengerId	Survived	Pclass	Sex
PassengerId	1.000000	-0.005007	-0.035144	0.042939
Survived	-0.005007	1.000000	-0.338481	-0.543351
Pclass	-0.035144	-0.338481	1.000000	0.131900
Sex	0.042939	-0.543351	0.131900	1.000000
Age	0.034212	-0.064910	-0.339898	0.081163
SibSp	-0.057527	-0.035322	0.083081	-0.114631
Parch	-0.001652	0.081629	0.018443	-0.245489
Fare	0.012658	0.257307	-0.549500	-0.182333
Embarked	0.013128	-0.167675	0.162098	0.108262
FamSize	-0.040143	0.016639	0.065997	-0.200988
IsAlone	0.057462	-0.203367	0.135207	0.303646
Title	0.063978	-0.060604	-0.132099	0.023909

Luego, se normalizan los datos con el standard scaler (estandarización Z).

Pasos Opcionales

Como no se requiere, no voy a comentar todos los aspectos sobre la parte de modelado, predicción y evaluación de rendimiento, pero sí los incluí en el código y me interesaría remarcar algunos puntos.

Se puede ver que los mejores resultados a primera vista, pues la única métrica que estamos mirando es accuracy (no es suficiente para tomar una decisión sobre si el modelo es bueno o no), son los de SVC y random forest, alrededor de 0.82 – 0.83, pero varía cada vez, dado que el random forest tiene un grado de aleatoriedad en su construcción.

Cabe destacar que, aunque los resultados del random forest sean los mejores para este set de testeo, el modelo tardó 60 segundos en entrenar, mientras que SVC tardó 1.2 segundos y la regresión logística 3.4s. Para conjuntos de datos más grandes o muy cambiantes para los que se necesita reentrenar el modelo constantemente, habría que ver si los pocos puntos de mejora en la accuracy justifican el costo inmensamente mayor de entrenamiento del modelo de random forest. Probablemente los tiempos de inferencia también sean mucho mayores en el random forest.

	Model	Accuracy
2	Random Forest	0.826816
0	SVC	0.821229
3	Logistic Regression	0.804469
1	Linear SVC	0.793296
6	Decision Tree	0.787709
5	Gaussian Naive Bayes	0.776536
4	K Nearest Neighbors	0.765363