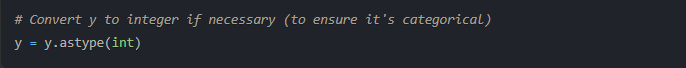
Claro, con mucho gusto explicaré este código paso a paso como si no supieras nada de programación. Déjame saber si necesitas alguna aclaración adicional en cualquier momento.

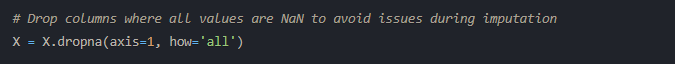
A black background with white text

Description automatically generated

Aquí se separan las características (features) y el objetivo (target) del conjunto de datos. `X` contendrá todas las columnas excepto la última, que se asigna a `y`. Esto se hace porque en un problema de aprendizaje automático supervisado, generalmente tienes un conjunto de características que se utilizan para predecir un objetivo o etiqueta.



Esta línea convierte el objetivo `y` a tipo entero. Esto es necesario si el objetivo es una variable categórica, ya que el aprendizaje automático a menudo trata las variables categóricas como enteros.



Esta línea elimina las columnas de `X` (características) donde todos los valores son NaN (Not a Number, o valor faltante). Esto se hace para evitar problemas durante la imputación, que es el proceso de rellenar los valores faltantes.

A screen shot of a computer code

Description automatically generated

Estas líneas identifican las características numéricas y categóricas en `X`. Las características numéricas son aquellas que contienen números, mientras que las características categóricas son aquellas que contienen valores no numéricos (por ejemplo, cadenas de texto).

A computer screen shot of a program

Description automatically generated

Estas líneas crean dos tuberías (pipelines) separadas para el preprocesamiento de datos numéricos y categóricos. Cada tubería es una secuencia de transformaciones que se aplicarán a los datos correspondientes.

Para los datos numéricos, la tubería realiza dos pasos:

1. `SimpleImputer` con la estrategia `'median'` rellena los valores faltantes con la mediana de esa columna.

2. `StandardScaler` estandariza los datos numéricos restando la media y dividiendo por la desviación estándar.

Para los datos categóricos, la tubería también realiza dos pasos:

1. `SimpleImputer` con la estrategia `'most\_frequent'` rellena los valores faltantes con el valor más frecuente en esa columna.

2. `OneHotEncoder` codifica las características categóricas en una representación numérica utilizando codificación one-hot.

A screen shot of a computer code

Description automatically generated

Esta línea combina las dos tuberías de preprocesamiento en un solo `ColumnTransformer`. El `ColumnTransformer` aplica la tubería numérica a las características numéricas y la tubería categórica a las características categóricas.



Esta línea realiza el preprocesamiento de los datos aplicando las transformaciones definidas en el `preprocessor` a las características `X`. El resultado es una matriz `X\_preprocessed` con las características numéricas escaladas y las características categóricas codificadas en one-hot.

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_preprocessed, y, test\_size=0.3, random\_state=42)**

Esta línea divide los datos en conjuntos de entrenamiento y prueba. `train\_test\_split` es una función de scikit-learn que divide los datos en dos subconjuntos: uno para entrenar el modelo (`X\_train`, `y\_train`) y otro para evaluar el modelo entrenado (`X\_test`, `y\_test`). `test\_size=0.3` significa que el 30% de los datos se utilizarán para pruebas, y `random\_state=42` establece una semilla para obtener resultados reproducibles.

A black background with white text

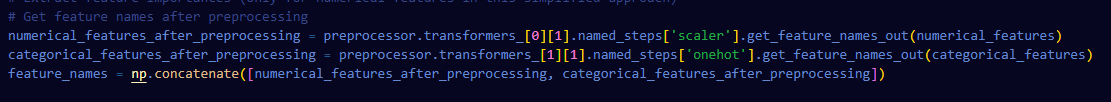
Description automatically generated

Estas líneas inicializan un clasificador Random Forest (Bosque Aleatorio) y lo entrenan con los datos de entrenamiento. `RandomForestClassifier` es un algoritmo de aprendizaje automático que construye múltiples árboles de decisión y combina sus predicciones. `n\_estimators=100` especifica que se construirán 100 árboles de decisión, y `random\_state=42` establece una semilla para obtener resultados reproducibles.

A screenshot of a computer code

Description automatically generated

Estas líneas realizan predicciones en el conjunto de prueba utilizando el modelo entrenado, y evalúan el rendimiento del modelo. `y\_pred` contiene las predicciones del modelo en el conjunto de prueba. `classification\_report` imprime una matriz de clasificación que muestra métricas como precisión, recuperación (recall), y puntaje F1 para cada clase. `confusion\_matrix` imprime una matriz de confusión que muestra las predicciones correctas e incorrectas del modelo.



Esta línea obtiene los nombres de las características numéricas después del preprocesamiento.

Veamos detenidamente cómo funciona:

1. `preprocessor.transformers\_` es una lista de tuplas, donde cada tupla contiene un nombre y un Pipeline. En este caso, `preprocessor.transformers\_[0]` es la tupla que contiene el Pipeline para las características numéricas.

2. `preprocessor.transformers\_[0][1]` accede al Pipeline de características numéricas.

3. `named\_steps` es un diccionario que contiene los pasos del Pipeline con sus respectivos nombres.

4. `named\_steps['scaler']` accede al paso llamado `'scaler'` en el Pipeline, que es el `StandardScaler`.

5. `get\_feature\_names\_out(numerical\_features)` es un método del `StandardScaler` que devuelve los nombres de las características numéricas después de la estandarización.

Así, `numerical\_features\_after\_preprocessing` contiene los nombres de las características numéricas después del preprocesamiento (estandarización).

Esta línea es similar a la anterior, pero obtiene los nombres de las características categóricas después del preprocesamiento (codificación one-hot).

1. `preprocessor.transformers\_[1]` es la tupla que contiene el Pipeline para las características categóricas.

2. `named\_steps['onehot']` accede al paso llamado `'onehot'` en el Pipeline, que es el `OneHotEncoder`.

3. `get\_feature\_names\_out(categorical\_features)` es un método del `OneHotEncoder` que devuelve los nombres de las características categóricas después de la codificación one-hot.

Así, `categorical\_features\_after\_preprocessing` contiene los nombres de las características categóricas después del preprocesamiento (codificación one-hot).

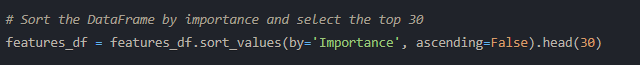
Esta línea combina los nombres de las características numéricas y categóricas en una sola lista llamada `feature\_names`. `np.concatenate` es una función de NumPy que concatena dos o más arrays.

En resumen, estas líneas obtienen los nombres de las características numéricas y categóricas después del preprocesamiento (estandarización y codificación one-hot, respectivamente), y luego los combina en una sola lista llamada `feature\_names`. Esto es necesario porque el modelo Random Forest trabaja con todas las características juntas, tanto numéricas como categóricas, después del preprocesamiento.

A black background with white text

Description automatically generated

Esta línea crea un DataFrame de pandas `features\_df` que contiene los nombres de las características y sus respectivas importancias según el modelo Random Forest. `rf\_classifier.feature\_importances\_` es un atributo que proporciona la importancia de cada característica en el modelo.



Aquí, el DataFrame `features\_df` se ordena de forma descendente según la importancia de las características, y se seleccionan las 30 características más importantes utilizando el método `head(30)`.

A screen shot of a computer program

Description automatically generated

Estas líneas crean un gráfico de barras horizontales que muestra las 30 características más importantes y sus respectivas importancias según el modelo Random Forest.

1. `plt.figure(figsize=(10, 8))` crea una nueva figura con un tamaño de 10 pulgadas de ancho y 8 pulgadas de alto.

2. `plt.title('Top 30 Feature Importances from RFC (numerical and categorical columns)')` establece el título del gráfico.

3. `plt.barh(features\_df['Feature'], features\_df['Importance'], color='b')` crea las barras horizontales, donde las etiquetas en el eje y son los nombres de las características (`features\_df['Feature']`), y las longitudes de las barras corresponden a las importancias de las características (`features\_df['Importance']`). El parámetro `color='b'` establece el color de las barras en azul.

4. `plt.xlabel('Importance')` y `plt.ylabel('Feature')` establecen las etiquetas de los ejes x e y, respectivamente.

5. `plt.gca().invert\_yaxis()` invierte el eje y para que las características más importantes aparezcan en la parte superior.

6. `plt.show()` muestra el gráfico.

Este código crea un gráfico de barras horizontales que permite visualizar fácilmente las características más importantes según el modelo Random Forest. Esto puede ser útil para entender mejor cómo el modelo está tomando decisiones y qué características son más relevantes para la tarea de clasificación.