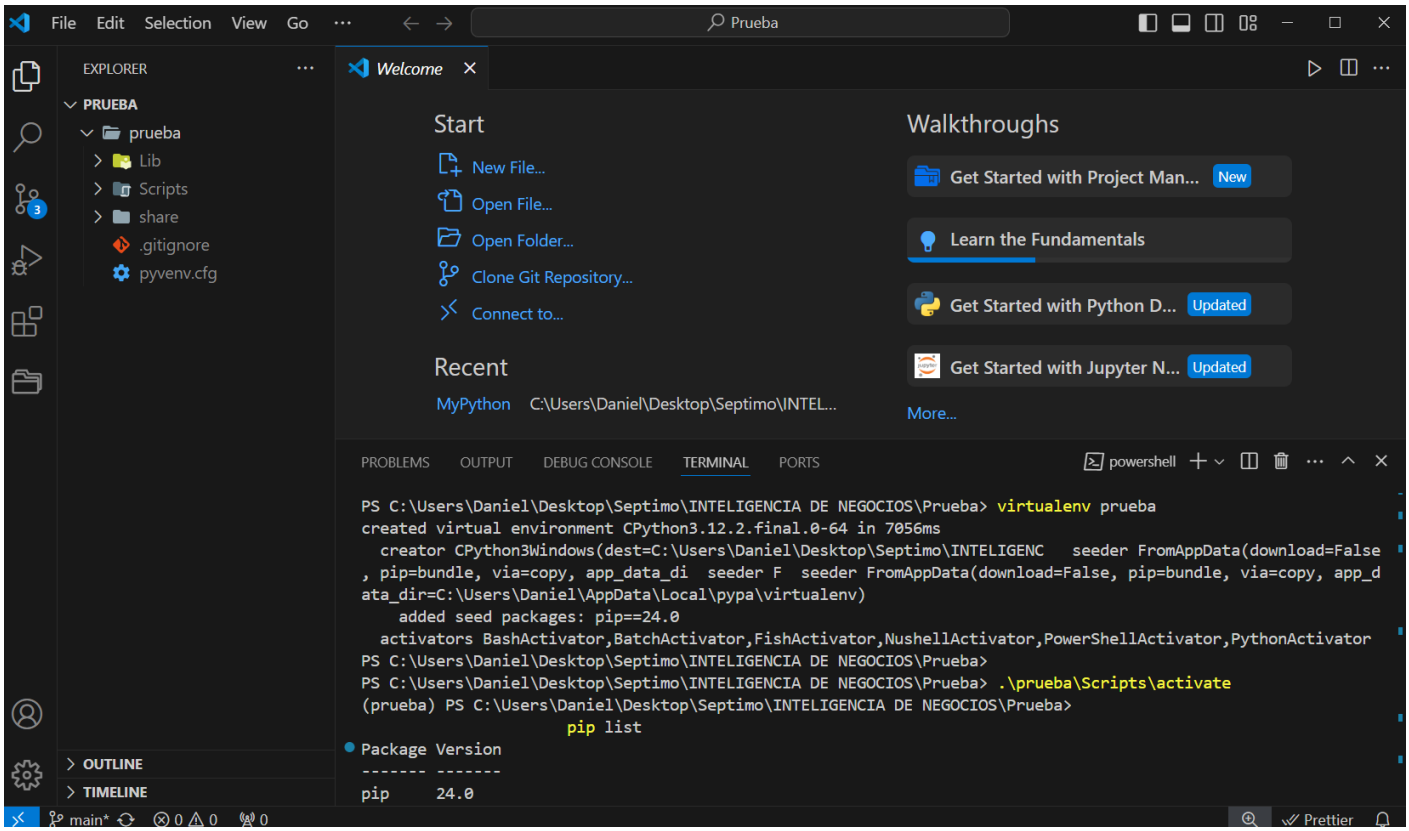
Prueba N1

Nombre: Daniel Galarza

Fecha: 09/04/2024

1. Creación del ambiente llamado Prueba



2. Use el [dataset](#) Titanic.xlsx

3. Aplique el algoritmo de clasificación

   Elimino las variables independientes que no me sirven

   Los elimino porque el Name no me sirve, la cabina tampoco y el bote salvavidas tiene demasiados valores nulos que no podre controlar solo usando la media.

```
X = dataset.drop(["Survived", "Name", "Cabin", "Life Boat", "Ticket Number"], axis=1)
y = pd.DataFrame(dataset["Survived"])
```

✓ 0.0s Python

```
X
```

✓ 0.0s Python

	Passenger Class	Sex	Age	No of Siblings or Spouses on Board	No of Parents or Children on Board	Passenger Fare	Port of Embarkation
0	First	Female	29.000000	0	0	211.3375	Southampton
1	First	Male	0.916700	1	2	151.5500	Southampton
2	First	Female	2.000000	1	2	151.5500	Southampton

Verificamos que existen valores nulos
Llenamos con la media esos valores para no que no exista sesgo

	Passenger Class	Sex	Age	No of Siblings or Spouses on Board	No of Parents or Children on Board	Ticket Number	Passenger Fare	Port of Embarkation
0	First	Female	29.0000	0	0	24160	211.3375	Southampton
1	First	Male	0.9167	1	2	113781	151.5500	Southampton
2	First	Female	2.0000	1	2	113781	151.5500	Southampton
3	First	Male	30.0000	1	2	113781	151.5500	Southampton
4	First	Female	25.0000	1	2	113781	151.5500	Southampton
...	...	...	...	...	...	...	...	...
1304	Third	Female	14.5000	1	0	2665	14.4542	Cherbourg
1305	Third	Female	NaN	1	0	2665	14.4542	Cherbourg
1306	Third	Male	26.5000	0	0	2656	7.2250	Cherbourg
1307	Third	Male	27.0000	0	0	2670	7.2250	Cherbourg
1308	Third	Male	29.0000	0	0	315082	7.8750	Southampton

```
# completar los valores que me faltan con la media
media = dataset['Age'].mean()
dataset['Age'] = dataset['Age'].fillna(media)
```

[25] ✓ 0.0s Python

	Passenger Class	Sex	Age	No of Siblings or Spouses on Board	No of Parents or Children on Board	Ticket Number	Passenger Fare	Port of Embarkation
0	First	Female	29.000000	0	0	24160	211.3375	Southampton
1	First	Male	0.916700	1	2	113781	151.5500	Southampton
2	First	Female	2.000000	1	2	113781	151.5500	Southampton
3	First	Male	30.000000	1	2	113781	151.5500	Southampton
4	First	Female	25.000000	1	2	113781	151.5500	Southampton
...	...	...	...	...	...	...	...	...
1304	Third	Female	14.500000	1	0	2665	14.4542	Cherbourg
1305	Third	Female	29.881135	1	0	2665	14.4542	Cherbourg
1306	Third	Male	26.500000	0	0	2656	7.2250	Cherbourg
1307	Third	Male	27.000000	0	0	2670	7.2250	Cherbourg
1308	Third	Male	29.000000	0	0	315082	7.8750	Southampton

1309 rows × 8 columns

Renombrar etiquetas

```
X = X.rename(columns={'Passenger Class':'Clase',
                      'Sex':'Sexo',
                      'Age':'Edad',
                      'No of Siblings or Spouses on Board':'noSibling',
                      'No of Parents or Children on Board':'noNiños',
                      'Ticket Number':'Tiquet',
                      'Passenger Fare':'Tarifa',
                      'Port of Embarkation':'Embarcacion'})
X
```

[33] ✓ 0.0s Python

	Clase	Sexo	Edad	noSibling	noNiños	Tiquet	Tarifa	Embarcacion
0	First	Female	29.000000	0	0	24160	211.3375	Southampton
1	First	Male	0.916700	1	2	113781	151.5500	Southampton
2	First	Female	2.000000	1	2	113781	151.5500	Southampton
3	First	Male	30.000000	1	2	113781	151.5500	Southampton
4	First	Female	25.000000	1	2	113781	151.5500	Southampton
...	...	...	...	...	...	...	...	...
1304	Third	Female	14.500000	1	0	2665	14.4542	Cherbourg
1305	Third	Female	29.881135	1	0	2665	14.4542	Cherbourg

Transformamos valores nominales a números

```
X.Embarcacion.value_counts()
```

[38] ✓ 0.0s

```
Embarcacion
Southampton    914
Cherbourg      270
Queenstown     123
Name: count, dtype: int64
```

```
# transformamos valores nominales a numeros
atributos = list(dataset.columns)
for atri in atributos:
    if atri == 'Clase':
        X[atri] = X[atri].map({'Third':0, 'First':1, 'Second':2})
    elif atri == 'Sexo':
        X[atri] = X[atri].map({'Male':0, 'Female':1})
    elif atri == 'Embarcacion':
        X[atri] = X[atri].map({'Southampton':0, 'Cherbourg':1, 'Queenstown':2})
X
```

[35] ✓ 0.0s

```
atributos = list(y.columns)
for atri in atributos:
    if atri == 'Survived':
        y[atri] = y[atri].map({'No':0, 'Yes':1})
y
```

[53] ✓ 0.0s

	Survived
0	1
1	1

División de datos de Prueba y Entrenamiento:
Aplicamos el algoritmo con un 25% de datos solamente sean Test y el resto con 75 de entrenamiento.

```
# Division de datos de Prueba y Entrenamiento
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
print("75% para train y 25% para Test")
print(X_train.shape)
print(X_test.shape)
```

[55] ✓ 0.0s Python

```
75% para train y 25% para Test
(981, 8)
(328, 8)
```

4. Aplique el algoritmo de clasificación

Para ello usamos las variables de x en entrenamiento y y en entrenamiento dado eso calculamos la predicción.

```
from sklearn.tree import DecisionTreeClassifier
clasificador = DecisionTreeClassifier(random_state=0)
clasificador.fit(X_train, y_train)
y_pred = clasificador.predict(X_test)
```

] ✓ 0.1s

4. Determine el score

```
# Score
clasificador.score(X_test, y_test)
                    # Test | resultado
```

[14] ✓ 0.0s

```
0.7591463414634146
```

5. Genere la Matriz de Confusión

6. Grafique el árbol

Obtención de variables independientes pata crear el árbol y creación del árbol usando max_depth para hacerlo más pequeño usando 3.

```
vi = [col for col in columnas if col != 'Survived']
vi
```

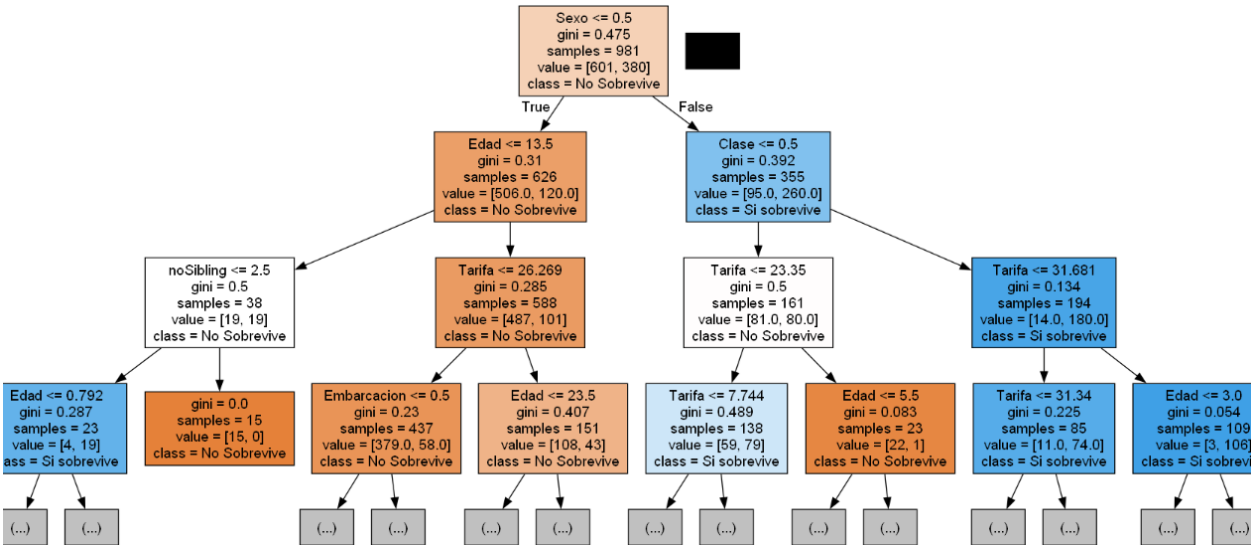
[16] ✓ 0.0s

```
['Clase', 'Sexo', 'Edad', 'noSibling', 'noNiños', 'Tarifa', 'Embarcacion']
```

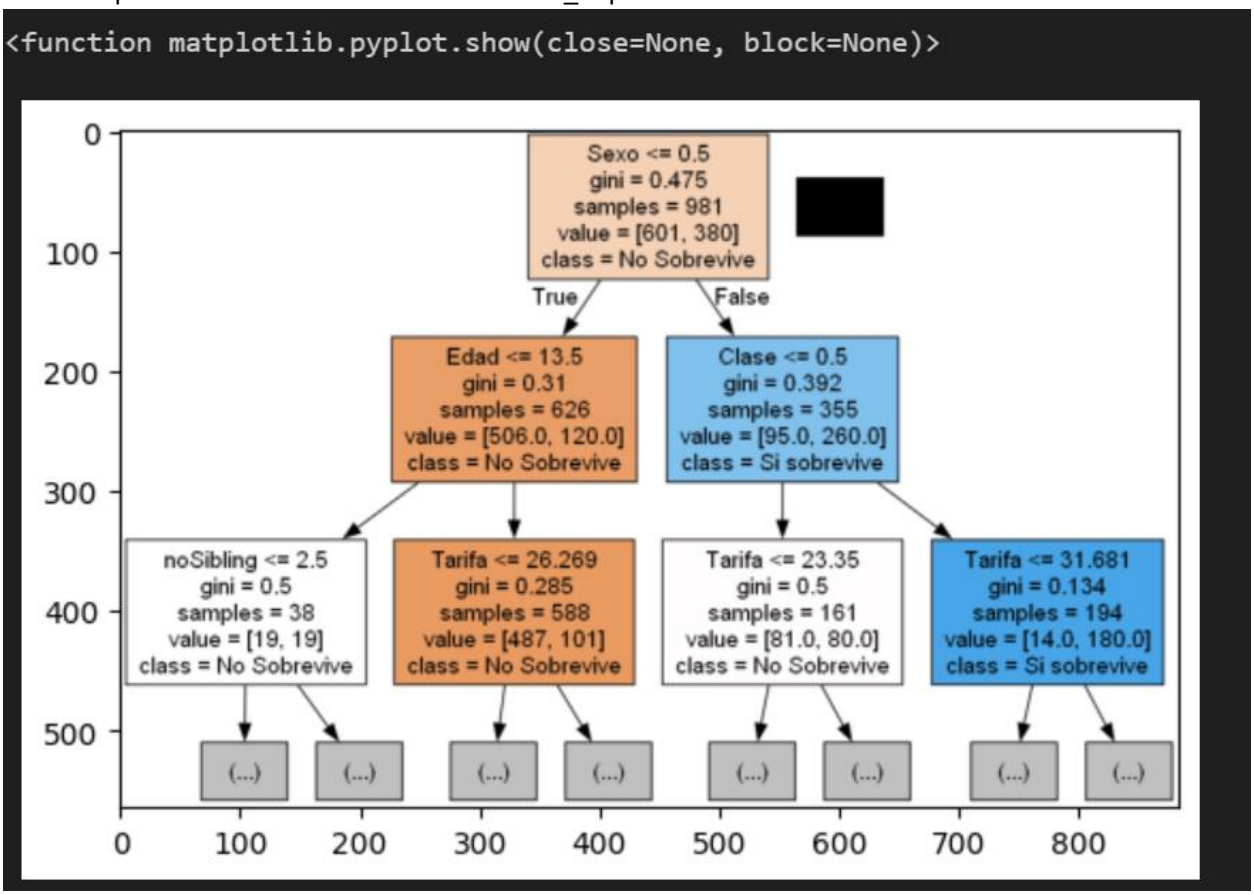
```
from sklearn import tree
import pydotplus
import matplotlib.pyplot as plt
import matplotlib.image as pltimg

clase = ['No', 'Si']

datos = tree.export_graphviz(clasificador,
                             class_names=clase,
                             feature_names=vi,
                             filled=True,
                             max_depth=3
                             )
graph = pydotplus.graph_from_dot_data(datos)
graph.write_png('arbol.png')
imagen = pltimg.imread('arbol.png')
plt.imshow(imagen)
plt.show
```

[18] ✓ 1.3s

Me sale un árbol grande:



Por ello aplicaremos una reducción con un max_depth = 2:



7. Utilice Weka y compare el resultado obtenido en Python