Proyecto # 2 APIs para ML

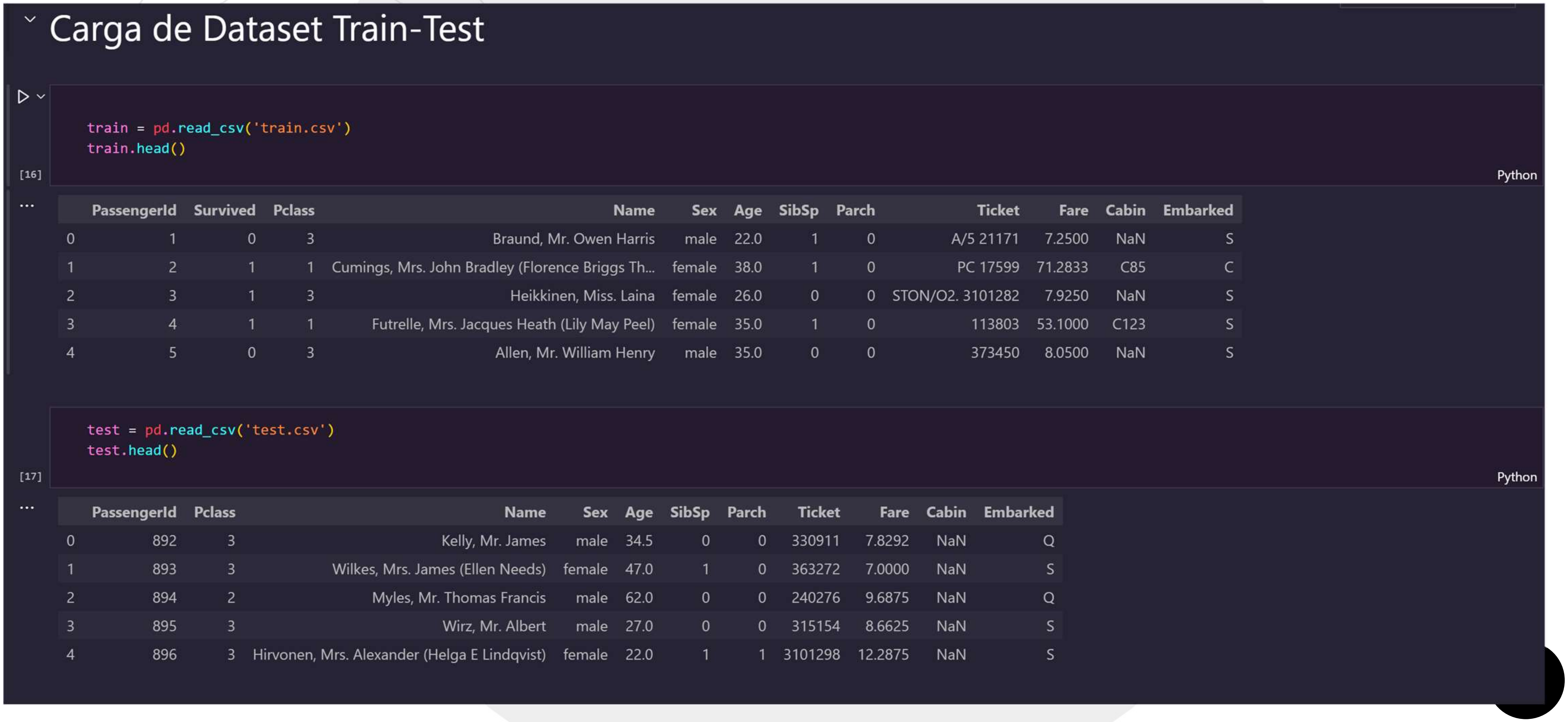
Equipo de trabajo

21000958	215002659	20023915
Osiel Gutierrez Herrera	Jose Andres Salguero Palomo	Enrique Antonio Viau Najarro

Objetivo

Aplicar todos los conceptos y métodos aprendidos durante el curso para resolver un problema de clasificación.

1.Análisis exploratorio del dataset Titanic



Carga de Dataset Train-Test

[16]

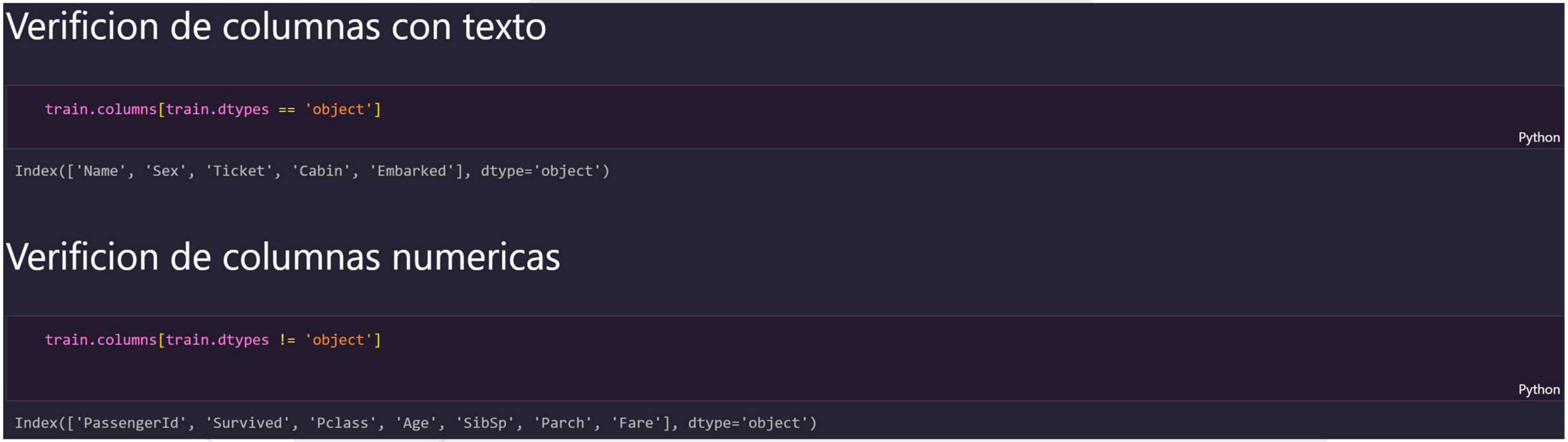
```python
train = pd.read_csv('train.csv')
train.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

[17]

```python
test = pd.read_csv('test.csv')
test.head()
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S



Verificion de columnas con texto

```
train.columns[train.dtypes == 'object']
```

Python

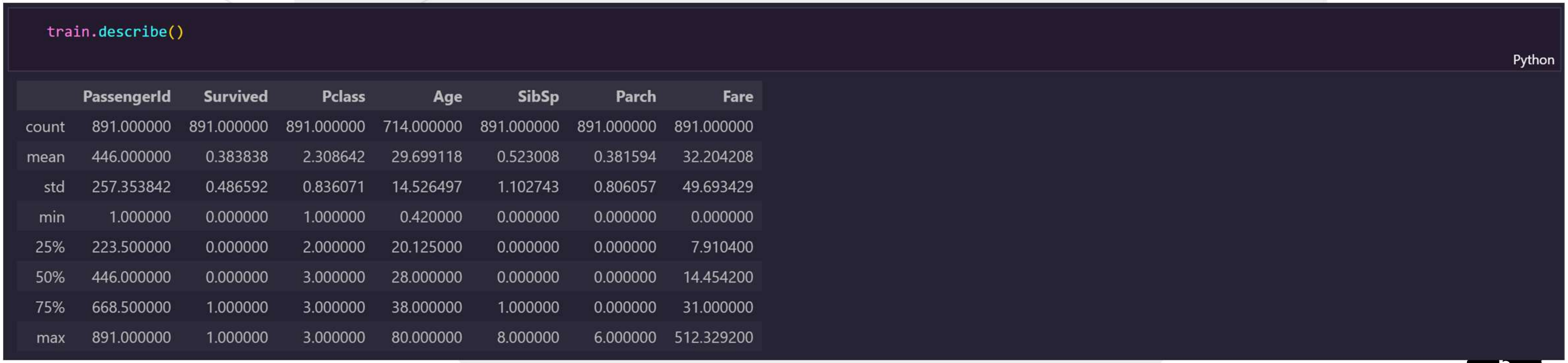
```
Index(['Name', 'Sex', 'Ticket', 'Cabin', 'Embarked'], dtype='object')
```

Verificion de columnas numericas

```
train.columns[train.dtypes != 'object']
```

Python

```
Index(['PassengerId', 'Survived', 'Pclass', 'Age', 'SibSp', 'Parch', 'Fare'], dtype='object')
```

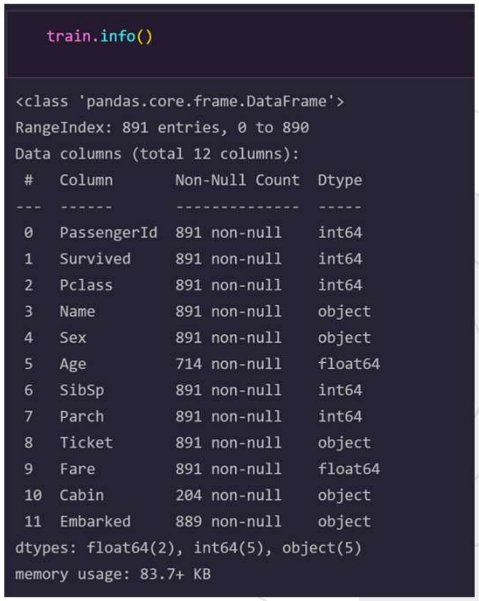


```
train.describe()
```

Python

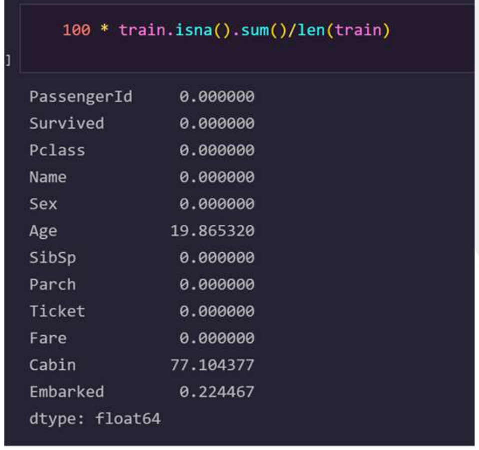
	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

Exploracion de los datos



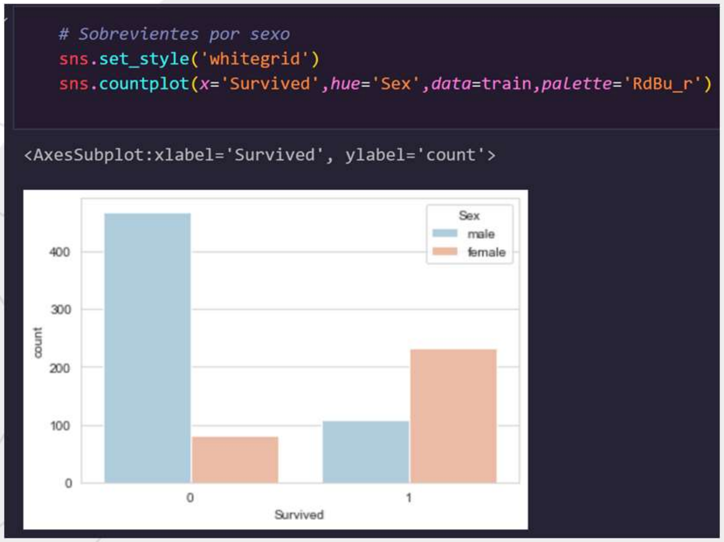
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```



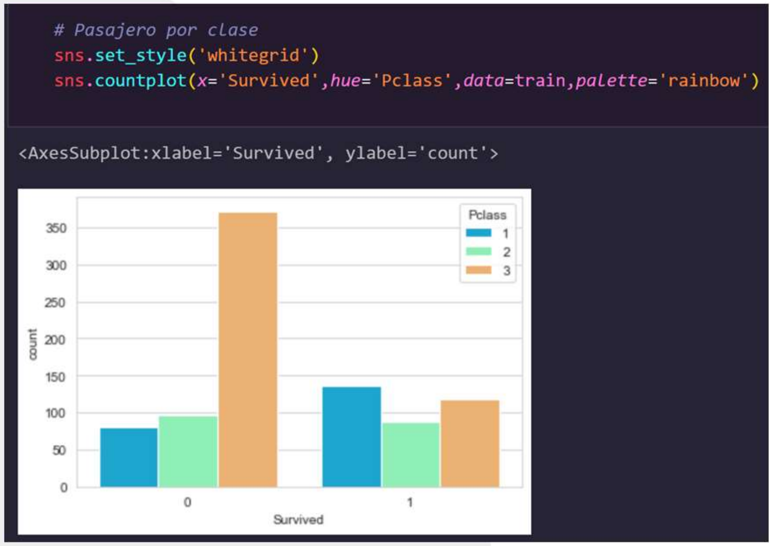
```
100 * train.isna().sum()/len(train)
```

```
PassengerId     0.000000
Survived        0.000000
Pclass          0.000000
Name            0.000000
Sex             0.000000
Age            19.865320
SibSp           0.000000
Parch           0.000000
Ticket          0.000000
Fare            0.000000
Cabin          77.104377
Embarked        0.224467
dtype: float64
```



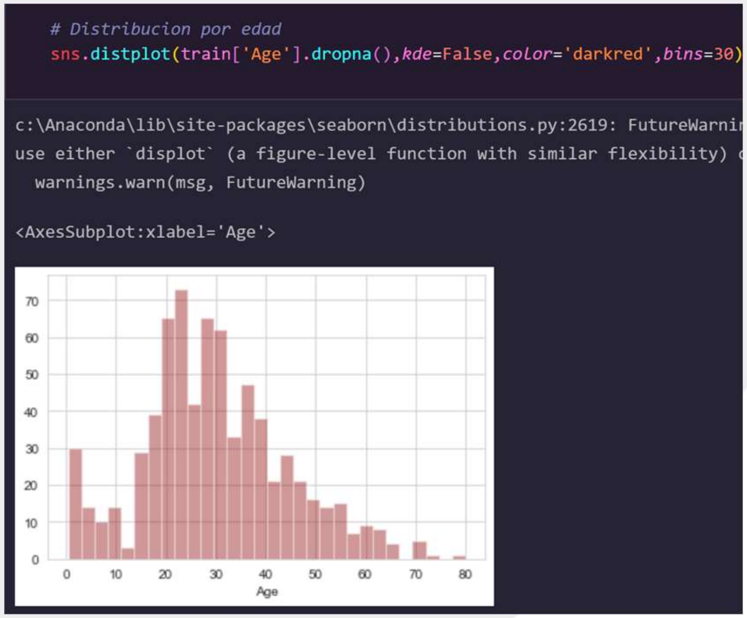
```
# Sobrevientes por sexo
sns.set_style('whitegrid')
sns.countplot(x='Survived',hue='Sex',data=train,palette='RdBu_r')
```

```
<AxesSubplot:xlabel='Survived', ylabel='count'>
```



```
# Pasajero por clase
sns.set_style('whitegrid')
sns.countplot(x='Survived',hue='Pclass',data=train,palette='rainbow')
```

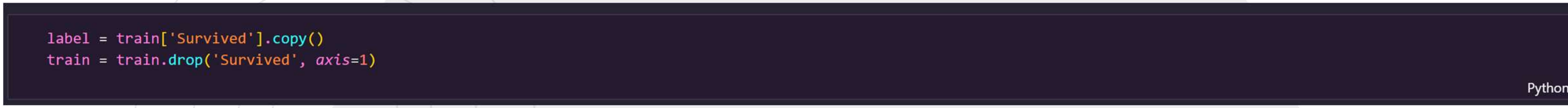
```
<AxesSubplot:xlabel='Survived', ylabel='count'>
```



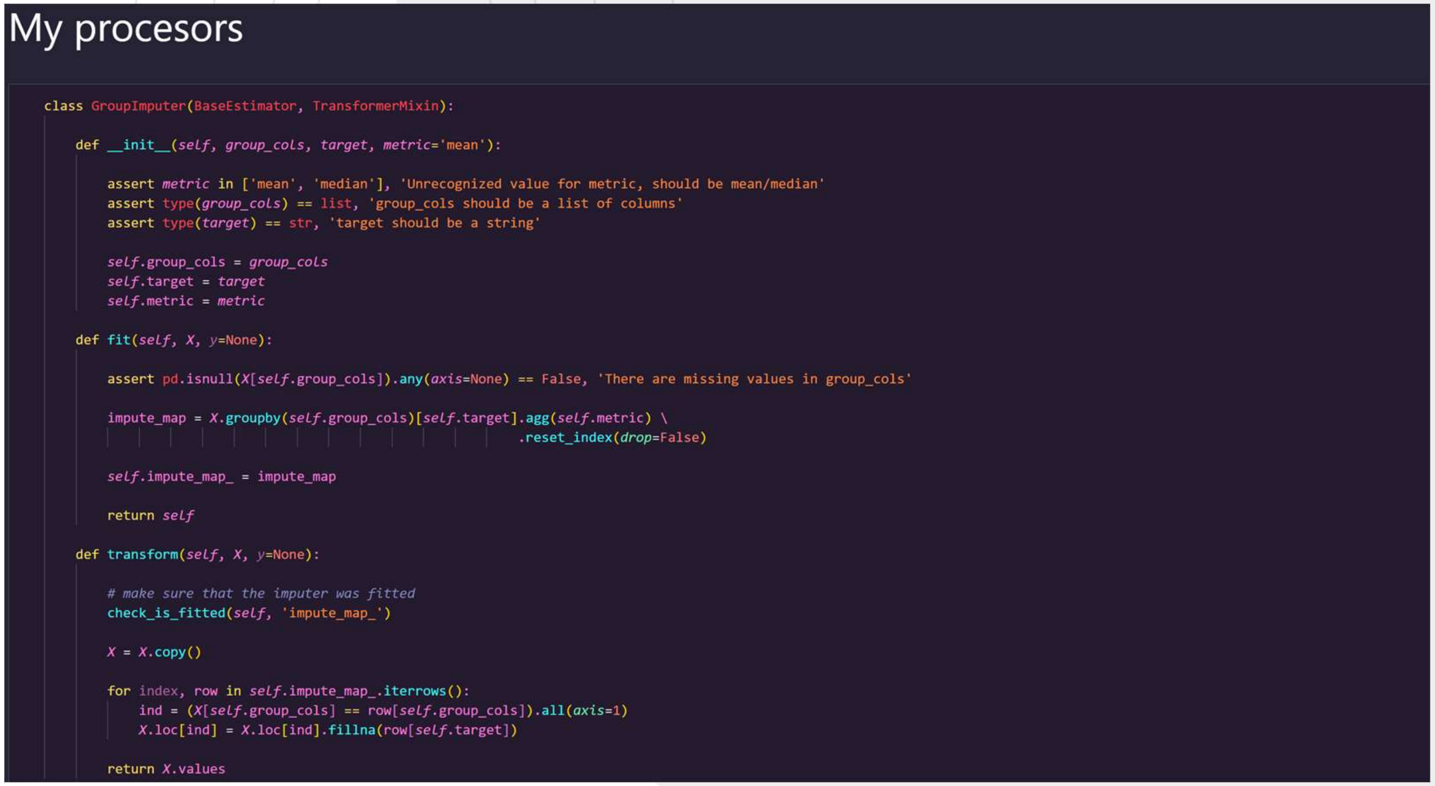
```
# Distribucion por edad
sns.distplot(train['Age'].dropna(),kde=False,color='darkred',bins=30)
```

```
c:\Anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarnin
use either `displot` (a figure-level function with similar flexibility) c
  warnings.warn(msg, FutureWarning)

<AxesSubplot:xlabel='Age'>
```

2. Limpieza de Datos



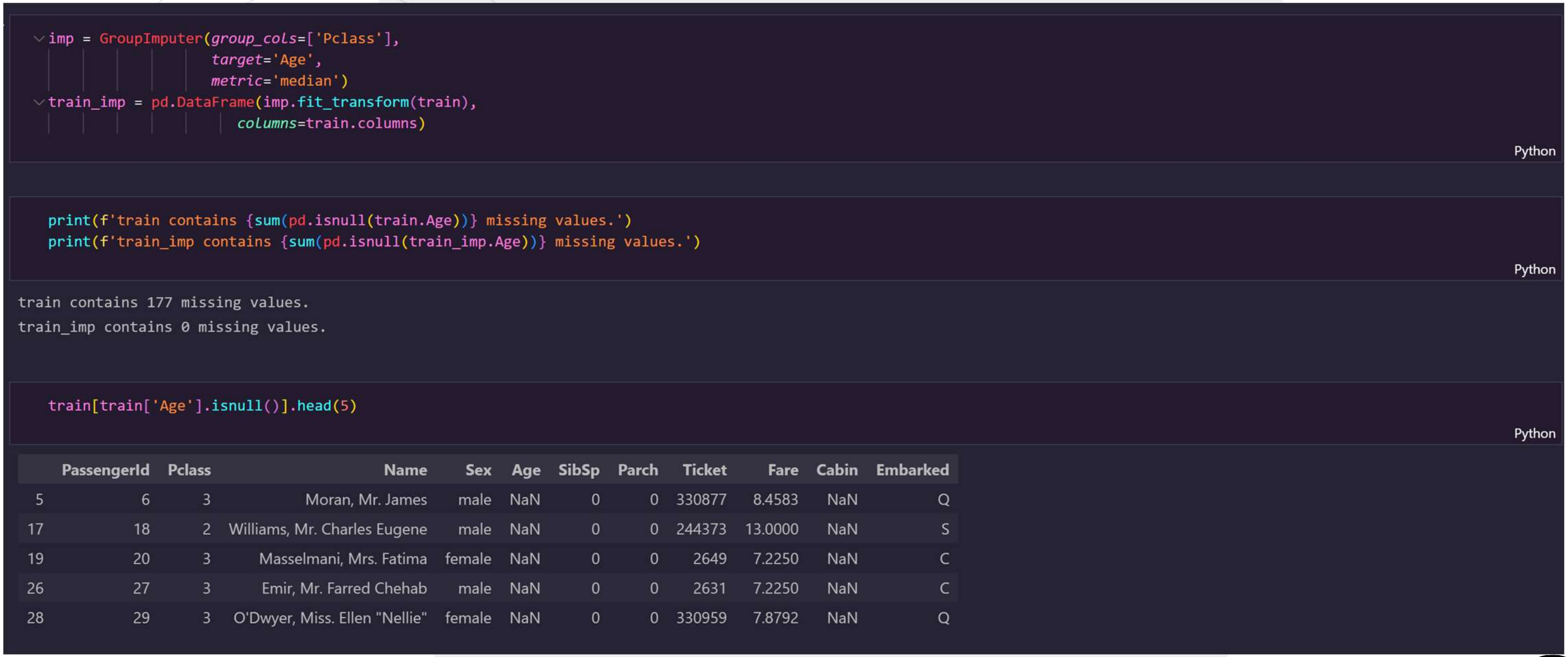
```
label = train['Survived'].copy()
train = train.drop('Survived', axis=1)
```



My procesors

```
class GroupImputer(BaseEstimator, TransformerMixin):

    def __init__(self, group_cols, target, metric='mean'):

        assert metric in ['mean', 'median'], 'Unrecognized value for metric, should be mean/median'
        assert type(group_cols) == list, 'group_cols should be a list of columns'
        assert type(target) == str, 'target should be a string'

        self.group_cols = group_cols
        self.target = target
        self.metric = metric

    def fit(self, X, y=None):

        assert pd.isnull(X[self.group_cols]).any(axis=None) == False, 'There are missing values in group_cols'

        impute_map = X.groupby(self.group_cols)[self.target].agg(self.metric) \
                                                            .reset_index(drop=False)

        self.impute_map_ = impute_map

        return self

    def transform(self, X, y=None):

        # make sure that the imputer was fitted
        check_is_fitted(self, 'impute_map_')

        X = X.copy()

        for index, row in self.impute_map_.iterrows():
            ind = (X[self.group_cols] == row[self.group_cols]).all(axis=1)
            X.loc[ind] = X.loc[ind].fillna(row[self.target])

        return X.values
```

3. Imputaciones



```
imp = GroupImputer(group_cols=['Pclass'],
                   target='Age',
                   metric='median')
train_imp = pd.DataFrame(imp.fit_transform(train),
                      columns=train.columns)
```

Python

```
print(f'train contains {sum(pd.isnull(train.Age))} missing values.')
print(f'train_imp contains {sum(pd.isnull(train_imp.Age))} missing values.')
```

Python

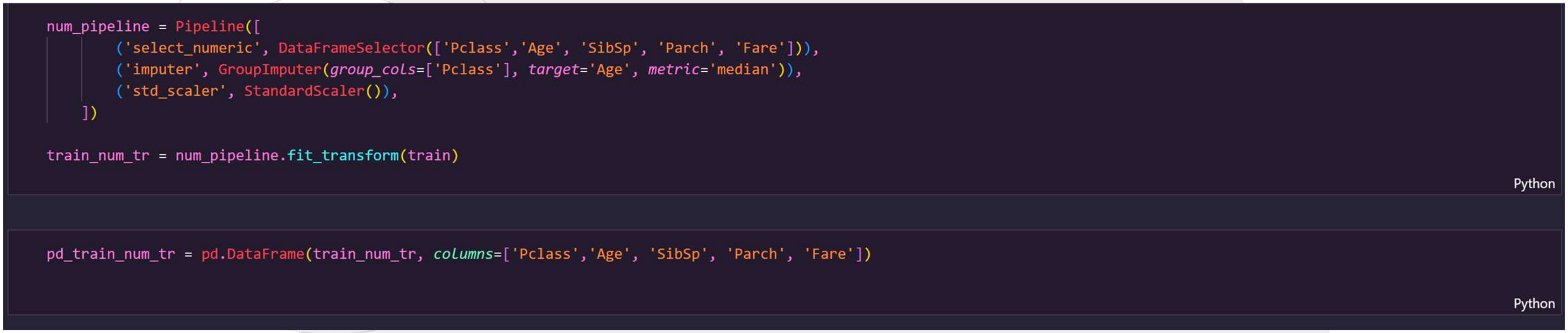
```
train contains 177 missing values.
train_imp contains 0 missing values.
```

```
train[train['Age'].isnull()].head(5)
```

Python

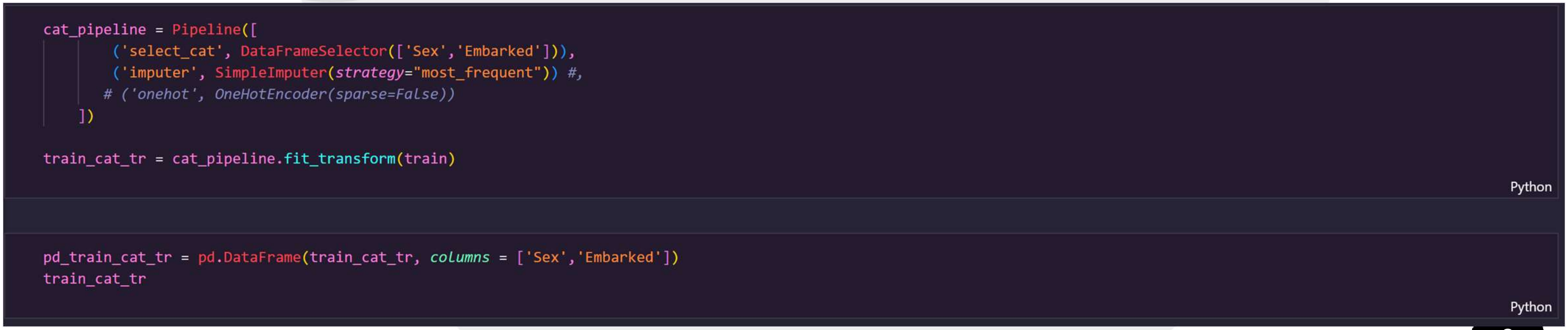
	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
5	6	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
17	18	2	Williams, Mr. Charles Eugene	male	NaN	0	0	244373	13.0000	NaN	S
19	20	3	Masselmani, Mrs. Fatima	female	NaN	0	0	2649	7.2250	NaN	C
26	27	3	Emir, Mr. Farred Chehab	male	NaN	0	0	2631	7.2250	NaN	C
28	29	3	O'Dwyer, Miss. Ellen "Nellie"	female	NaN	0	0	330959	7.8792	NaN	Q

4. Pipeline



```
num_pipeline = Pipeline([
        ('select_numeric', DataFrameSelector(['Pclass','Age', 'SibSp', 'Parch', 'Fare'])),
        ('imputer', GroupImputer(group_cols=['Pclass'], target='Age', metric='median')),
        ('std_scaler', StandardScaler()),
    ])

train_num_tr = num_pipeline.fit_transform(train)
```

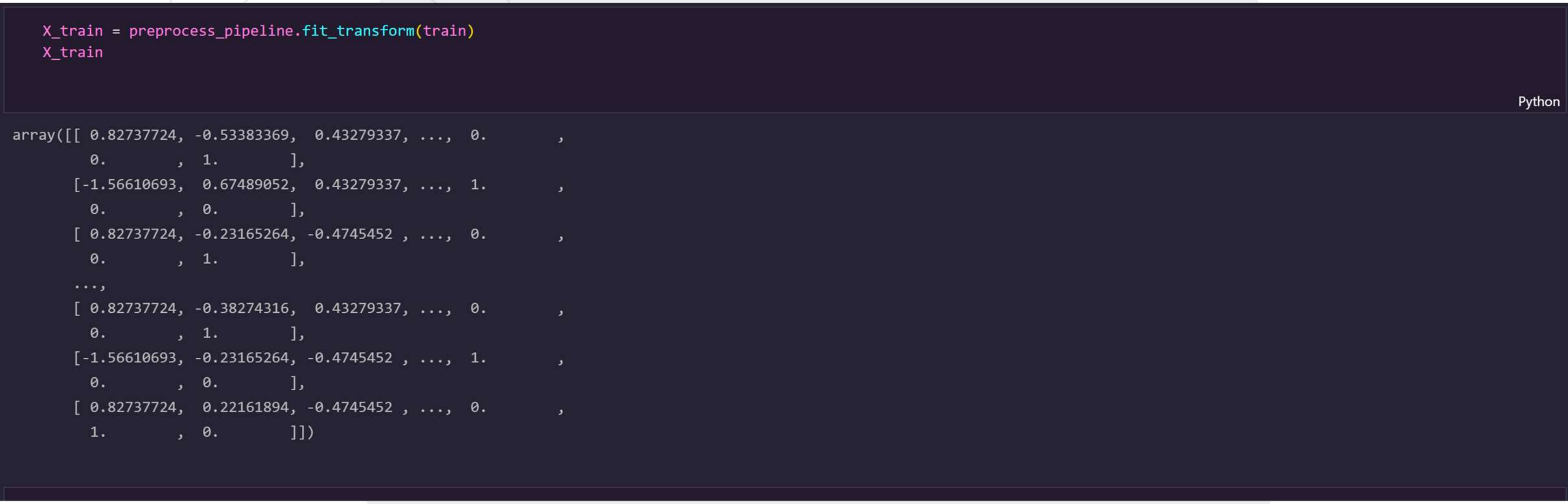
```
pd_train_num_tr = pd.DataFrame(train_num_tr, columns=['Pclass','Age', 'SibSp', 'Parch', 'Fare'])
```



```
cat_pipeline = Pipeline([
        ('select_cat', DataFrameSelector(['Sex','Embarked'])),
        ('imputer', SimpleImputer(strategy="most_frequent")) #,
       # ('onehot', OneHotEncoder(sparse=False))
    ])

train_cat_tr = cat_pipeline.fit_transform(train)
```

```
pd_train_cat_tr = pd.DataFrame(train_cat_tr, columns = ['Sex','Embarked'])
train_cat_tr
```

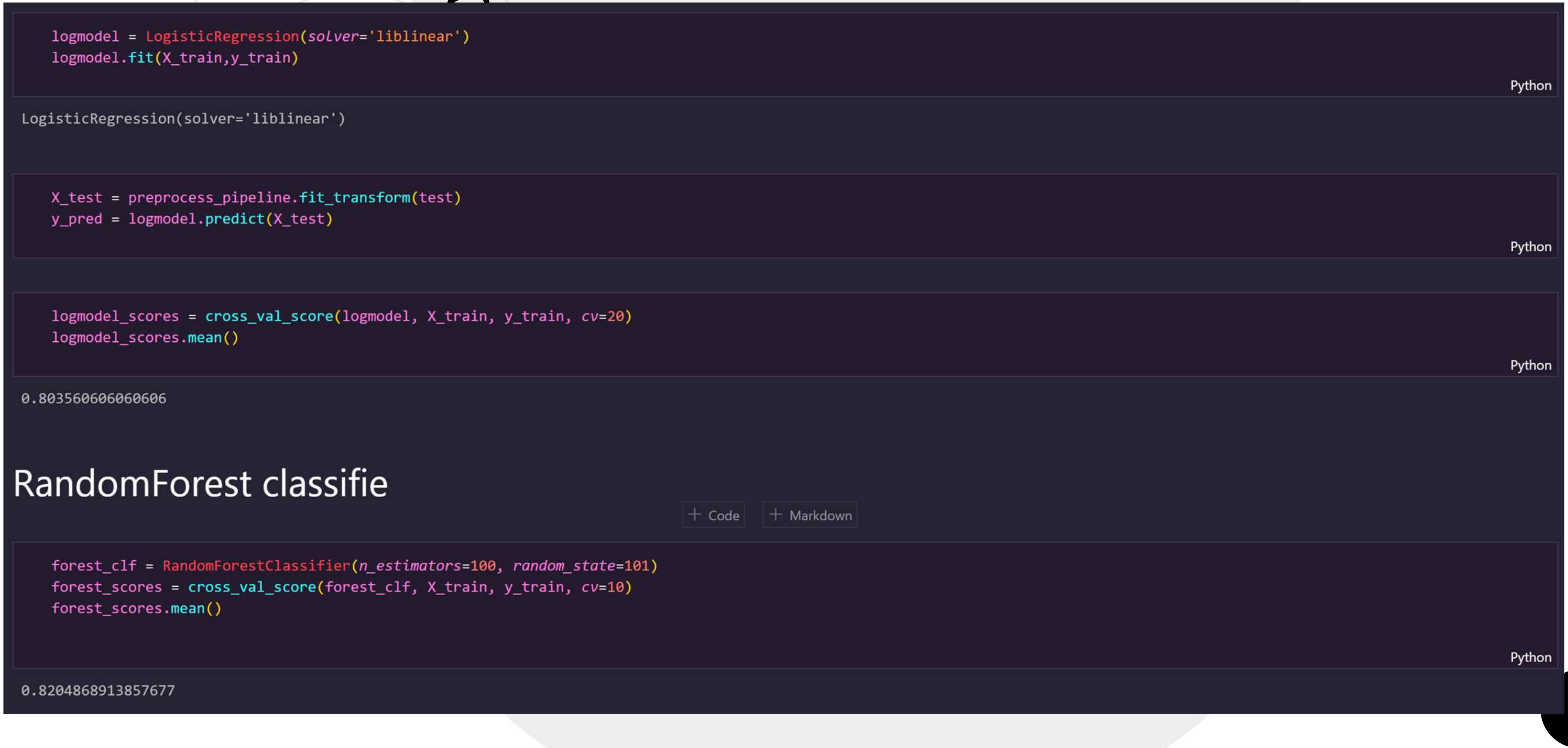
5. Preparación de datos para entrenamiento



```
X_train = preprocess_pipeline.fit_transform(train)
X_train
```

```
array([[ 0.82737724, -0.53383369,  0.43279337, ...,  0.        ,
         0.        ,  1.        ],
       [-1.56610693,  0.67489052,  0.43279337, ...,  1.        ,
         0.        ,  0.        ],
       [ 0.82737724, -0.23165264, -0.4745452 , ...,  0.        ,
         0.        ,  1.        ],
       ...,
       [ 0.82737724, -0.38274316,  0.43279337, ...,  0.        ,
         0.        ,  1.        ],
       [-1.56610693, -0.23165264, -0.4745452 , ...,  1.        ,
         0.        ,  0.        ],
       [ 0.82737724,  0.22161894, -0.4745452 , ...,  0.        ,
         1.        ,  0.        ]])
```

6. Entrenamiento y predicción



```
logmodel = LogisticRegression(solver='liblinear')
logmodel.fit(X_train,y_train)
```

```
LogisticRegression(solver='liblinear')
```

```
X_test = preprocess_pipeline.fit_transform(test)
y_pred = logmodel.predict(X_test)
```

```
logmodel_scores = cross_val_score(logmodel, X_train, y_train, cv=20)
logmodel_scores.mean()
```

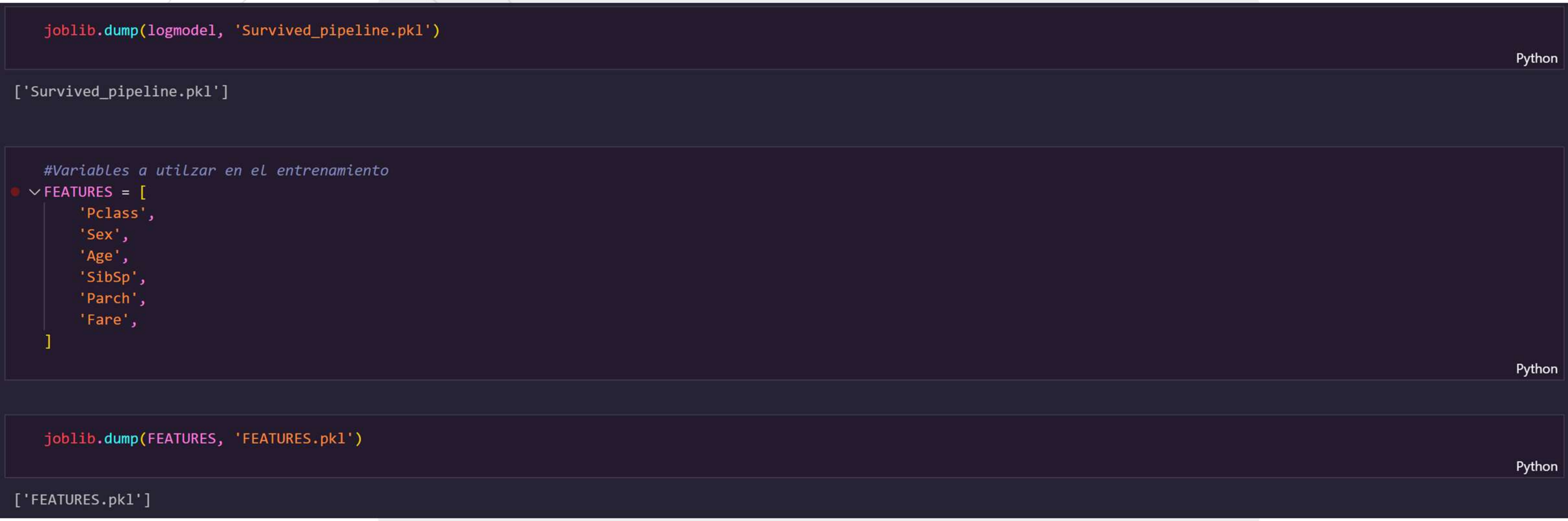
```
0.803560606060606
```

RandomForest classifie

```
forest_clf = RandomForestClassifier(n_estimators=100, random_state=101)
forest_scores = cross_val_score(forest_clf, X_train, y_train, cv=10)
forest_scores.mean()
```

```
0.8204868913857677
```

7. Generación de archivo pkl



```python
joblib.dump(logmodel, 'Survived_pipeline.pkl')
```

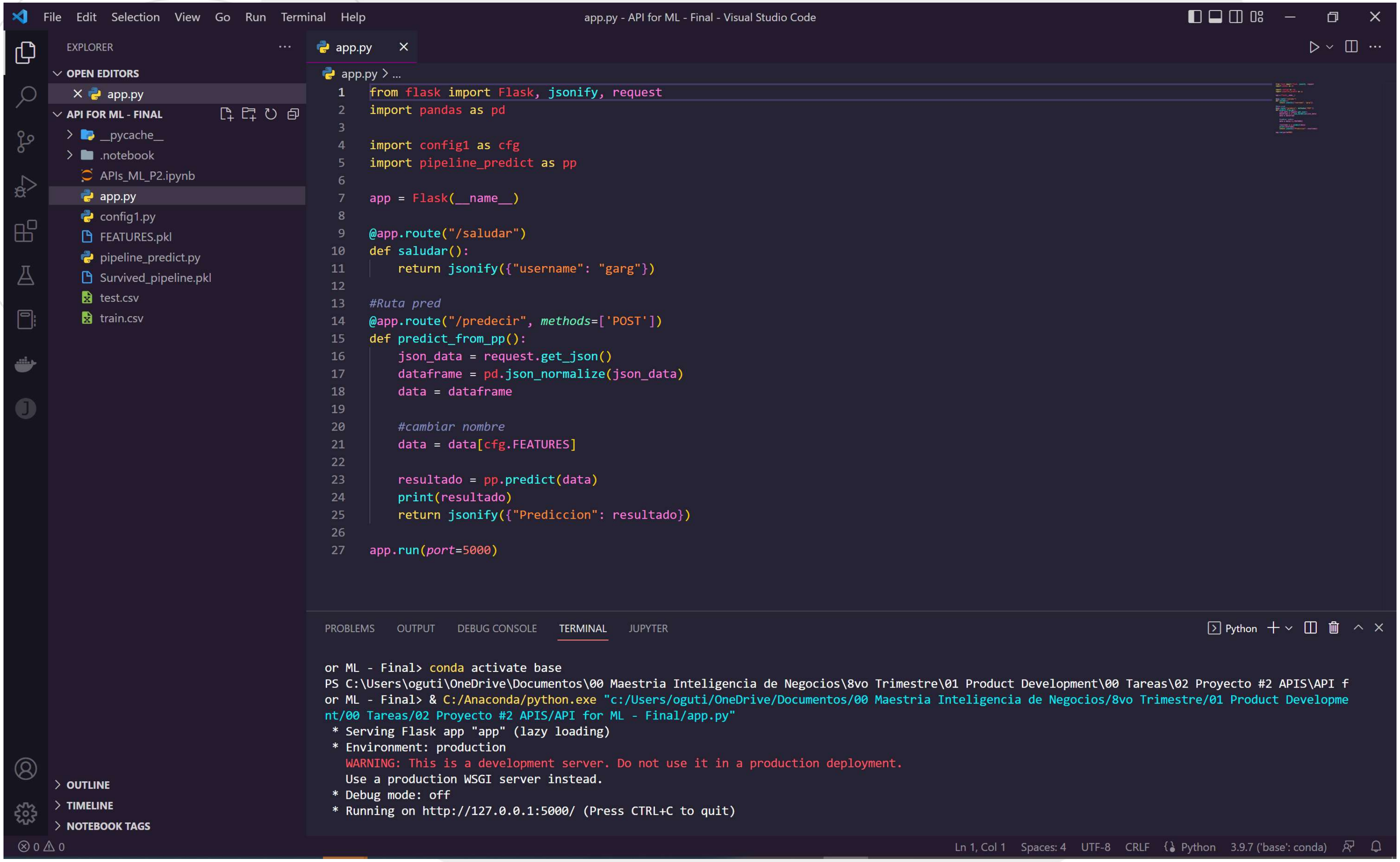
```
['Survived_pipeline.pkl']
```

```python
#Variables a utilzar en el entrenamiento
FEATURES = [
    'Pclass',
    'Sex',
    'Age',
    'SibSp',
    'Parch',
    'Fare',
]
```

```python
joblib.dump(FEATURES, 'FEATURES.pkl')
```

```
['FEATURES.pkl']
```

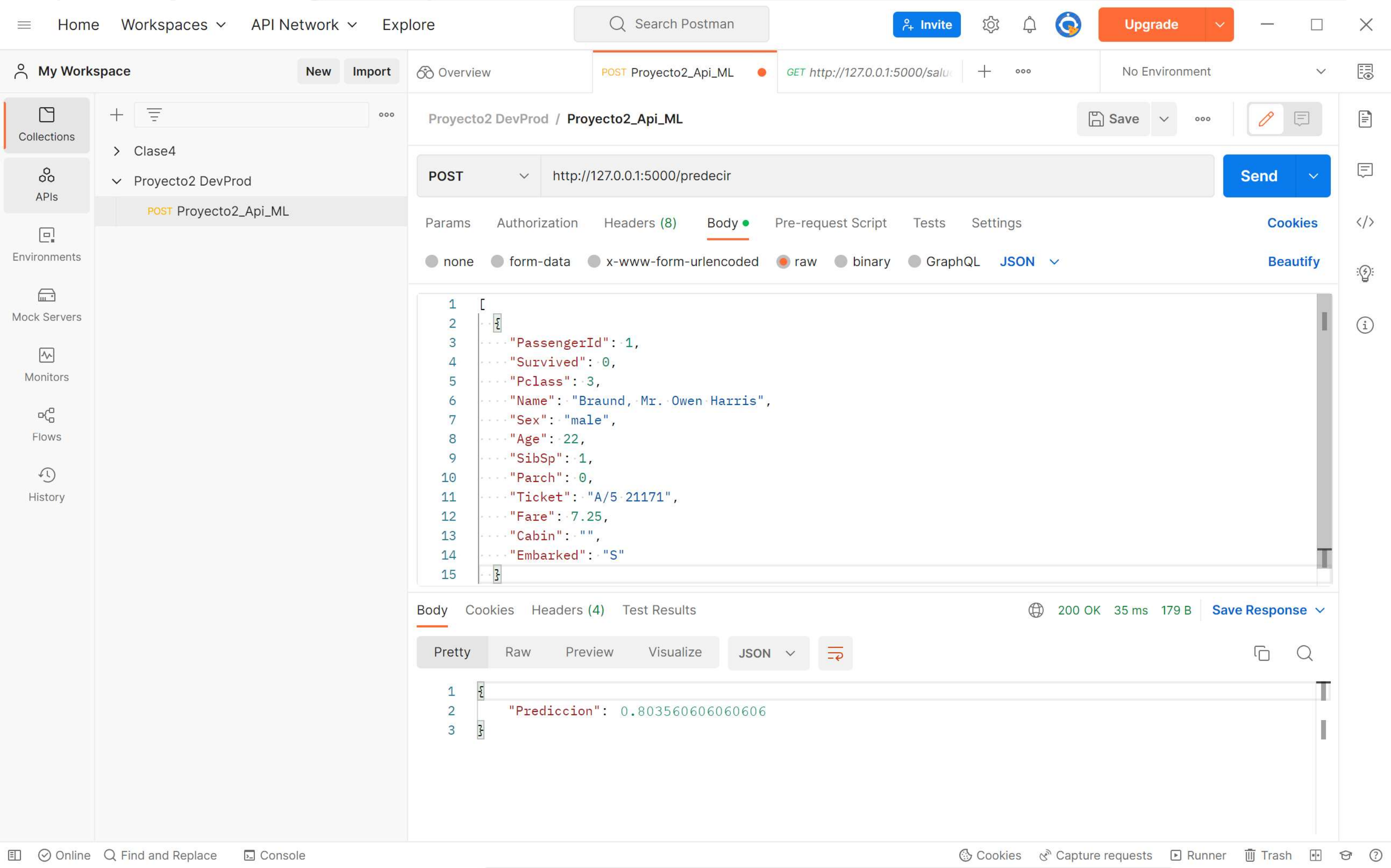
8. APIs



```python
from flask import Flask, jsonify, request
import pandas as pd

import config1 as cfg
import pipeline_predict as pp

app = Flask(__name__)

@app.route("/saludar")
def saludar():
    return jsonify({"username": "garg"})

#Ruta pred
@app.route("/predecir", methods=['POST'])
def predict_from_pp():
    json_data = request.get_json()
    dataframe = pd.json_normalize(json_data)
    data = dataframe

    #cambiar nombre
    data = data[cfg.FEATURES]

    resultado = pp.predict(data)
    print(resultado)
    return jsonify({"Prediccion": resultado})

app.run(port=5000)
```

```
or ML - Final> conda activate base
PS C:\Users\oguti\OneDrive\Documentos\00 Maestria Inteligencia de Negocios\8vo Trimestre\01 Product Development\00 Tareas\02 Proyecto #2 APIS\API f
or ML - Final> & C:/Anaconda/python.exe "c:/Users/oguti/OneDrive/Documentos/00 Maestria Inteligencia de Negocios/8vo Trimestre/01 Product Developme
nt/00 Tareas/02 Proyecto #2 APIS/API for ML - Final/app.py"
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

9. Conexión con Postman



POST http://127.0.0.1:5000/predecir

Body: ● raw JSON

```
[
  {
    "PassengerId": 1,
    "Survived": 0,
    "Pclass": 3,
    "Name": "Braund, Mr. Owen Harris",
    "Sex": "male",
    "Age": 22,
    "SibSp": 1,
    "Parch": 0,
    "Ticket": "A/5 21171",
    "Fare": 7.25,
    "Cabin": "",
    "Embarked": "S"
  }
```

200 OK 35 ms 179 B

```
{
    "Prediccion": 0.803560606060606
}
```

Concluisones

- Estas herramientas nos ayudan a facilitar la entraga de un modelo de prediction, en este caso la Plataforma de phyton con Sklearn nos ayuda a elaborar modelos de predicción y poder visualizer y consultar la inforamcion para futuros analisis.

- Nos permite ampliar nuestro portafolio de herramientas para solventar problemas futuros.