```
#import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import OneHotEncoder
from sklearn.impute import SimpleImputer



# Load the dataset
titanic_df = pd.read_csv('titanic.csv')


#View the data
titanic_df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence	female	38.0	1	0	PC 17599	71.2833

```
#Basic information

titanic_df.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
```

```
#Describe the data

titanic_df.describe()
```

	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

```
#Find the duplicates

titanic_df.duplicated().sum()
```

```
0
```

```
#unique values

titanic_df['Pclass'].unique()
```

```
array([3, 1, 2])
```

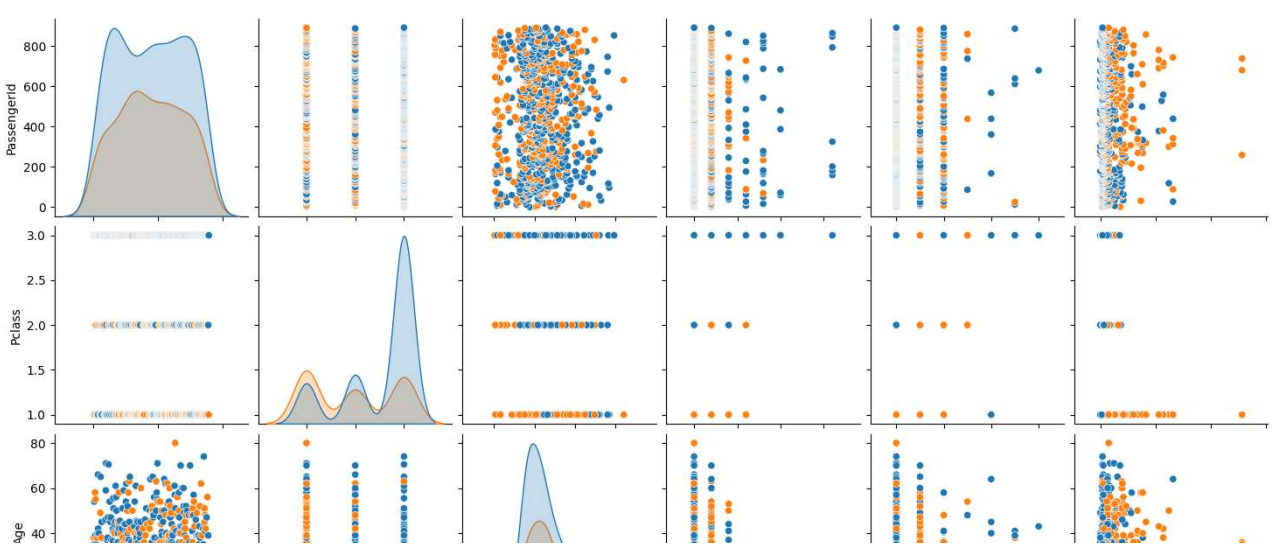
```
titanic_df['Survived'].unique()
```

```
array([0, 1])
```

```
titanic_df['Sex'].unique()
```

```
array(['male', 'female'], dtype=object)
```

```
# Visualize the data
sns.pairplot(titanic_df, hue='Survived')
plt.show()
```




```
#Find null values

titanic_df.isnull().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

```
#Replace null values

titanic_df.replace(np.nan,'0',inplace = True)
```

```
#Check the changes now
titanic_df.isnull().sum()
```

```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
dtype: int64
```

```
#cheking the Datatypes

titanic_df.dtypes
```

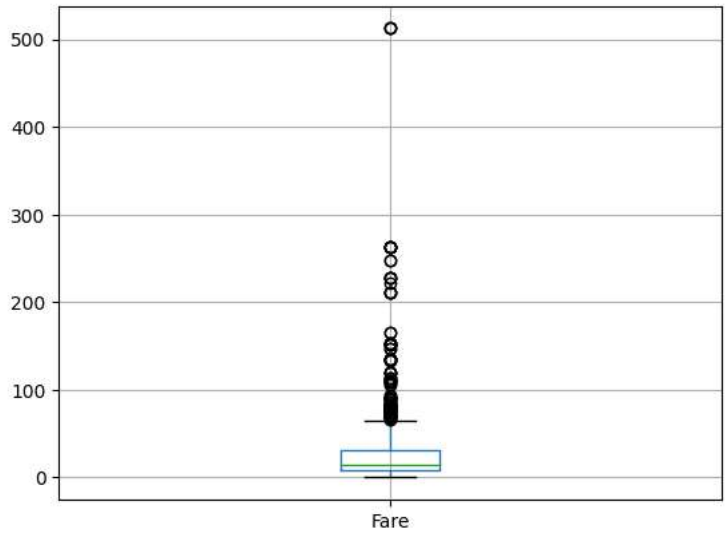
```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age             object
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

```
#Filter data

titanic_df[titanic_df['Pclass']==1].head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500	C103	S
23	24	1	1	Sloper, Mr. William Thompson	male	28.0	0	0	113788	35.5000	A6	S

```
#Boxplot

titanic_df[['Fare']].boxplot()
```

```
<Axes: >
```



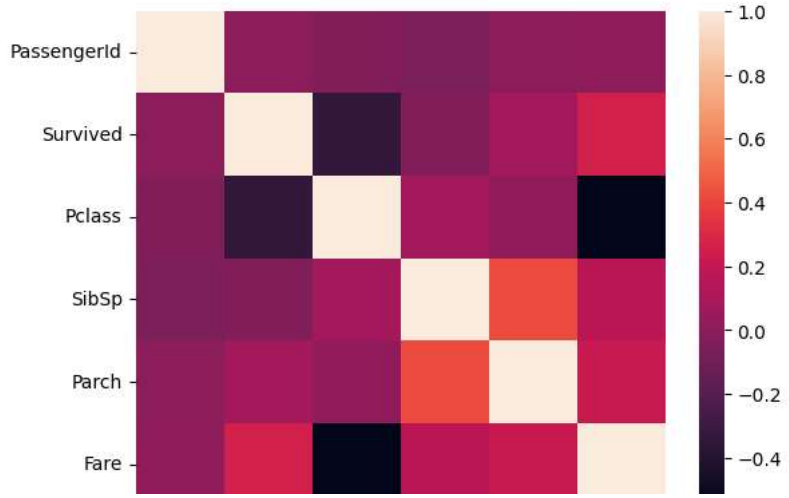

```
#Correlation

titanic_df.corr()
```

```
<ipython-input-35-682f5432db94>:3: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future versior
  titanic_df.corr()
```

	PassengerId	Survived	Pclass	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	0.083081	0.018443	-0.549500
SibSp	-0.057527	-0.035322	0.083081	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.159651	0.216225	1.000000

```
#Correlation plot

sns.heatmap(titanic_df.corr())
```

```
<ipython-input-37-dec87e11b1d9>:3: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future versior
  sns.heatmap(titanic_df.corr())
<Axes: >
```




```
# Select features and target variable
features = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked']
target = 'Survived'
X = titanic_df[features]
y = titanic_df[target]
```

```
# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


# One-hot encode the categorical variables
cat_vars = ['Sex', 'Embarked']
enc = OneHotEncoder(handle_unknown='ignore')
X_train_enc = enc.fit_transform(X_train[cat_vars])
X_test_enc = enc.transform(X_test[cat_vars])
feature_names = enc.get_feature_names_out(cat_vars)


# Handle missing values
imp = SimpleImputer(strategy='median')
X_train_num = pd.DataFrame(X_train.select_dtypes(include='number'))
X_test_num = pd.DataFrame(X_test.select_dtypes(include='number'))
X_train_imputed = imp.fit_transform(X_train_num)
X_test_imputed = imp.transform(X_test_num)


# Concatenate the numeric and encoded categorical variables
X_train_final = pd.concat([pd.DataFrame(X_train_imputed, columns=X_train_num.columns), pd.DataFrame(X_train_enc.toarray(), columns=feature_na
X_test_final = pd.concat([pd.DataFrame(X_test_imputed, columns=X_test_num.columns), pd.DataFrame(X_test_enc.toarray(), columns=feature_names)


# Train a logistic regression model on the training data
model = LogisticRegression()
model.fit(X_train_final, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

▾ LogisticRegression
LogisticRegression()

```
# Evaluate the model on the test data and print the accuracy score
y_pred = model.predict(X_test_final)
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy:', accuracy)
```

```
Accuracy: 0.7653631284916201
```

✓ 0s completed at 3:23 PM