Import Libraries

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

```
from google.colab import drive
drive.mount ('/content/drive')
```

```
Mounted at /content/drive
```

```
train_data = pd.read_csv("/content/drive/MyDrive/TitanicDataset/train.csv")
test_data = pd.read_csv("/content/drive/MyDrive/TitanicDataset/test.csv")
```

EDA

```
train_data.head(5)
```

	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

```
train_data.shape, test_data.shape
```

```
((891, 12), (418, 11))
```

```
train_data.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
```

```
train_data.isnull().sum()*100/train_data.shape[0]
```

```
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
```

```
test_data.isnull().sum()*100/test_data.shape[0]
```

```
PassengerId     0.000000
Pclass          0.000000
Name            0.000000
```

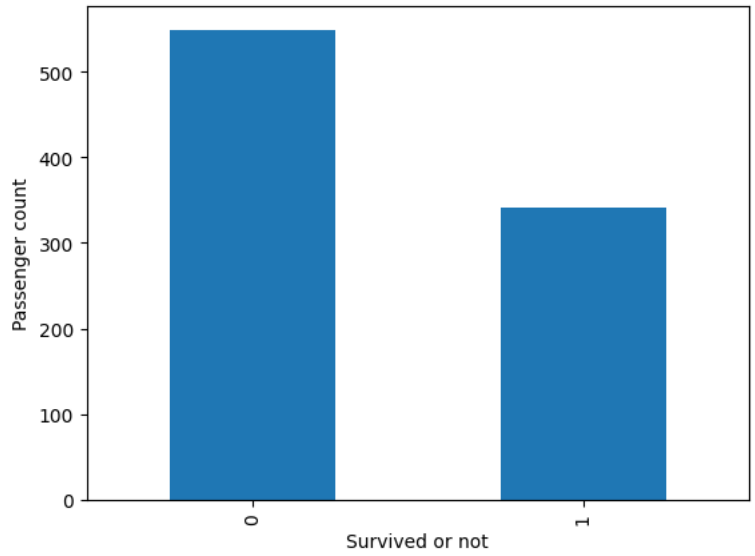
```
Sex            0.000000
Age           20.574163
SibSp          0.000000
Parch          0.000000
Ticket         0.000000
Fare           0.239234
Cabin         78.229665
Embarked       0.000000
dtype: float64
```

```
train_data.Survived.value_counts()*100/train_data.shape[0]
```

```
0    61.616162
1    38.383838
Name: Survived, dtype: float64
```

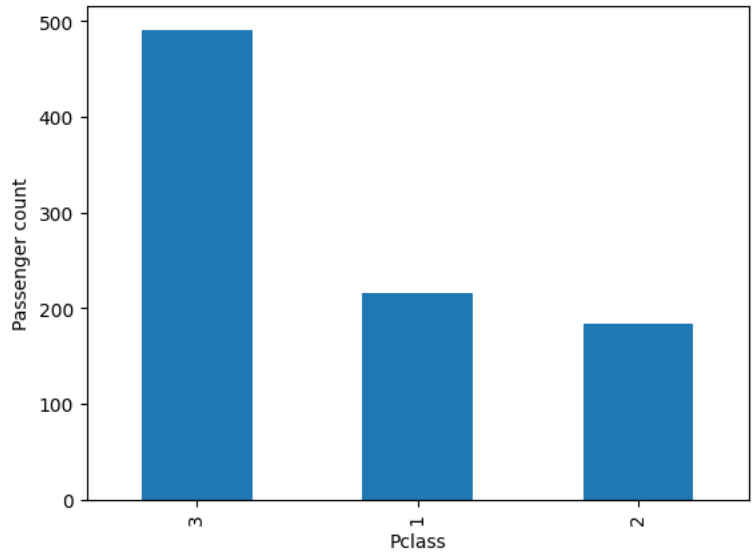
```
plt = train_data.Survived.value_counts().plot(kind='bar')
plt.set_xlabel('Survived or not')
plt.set_ylabel("Passenger count")
```

```
Text(0, 0.5, 'Passenger count')
```



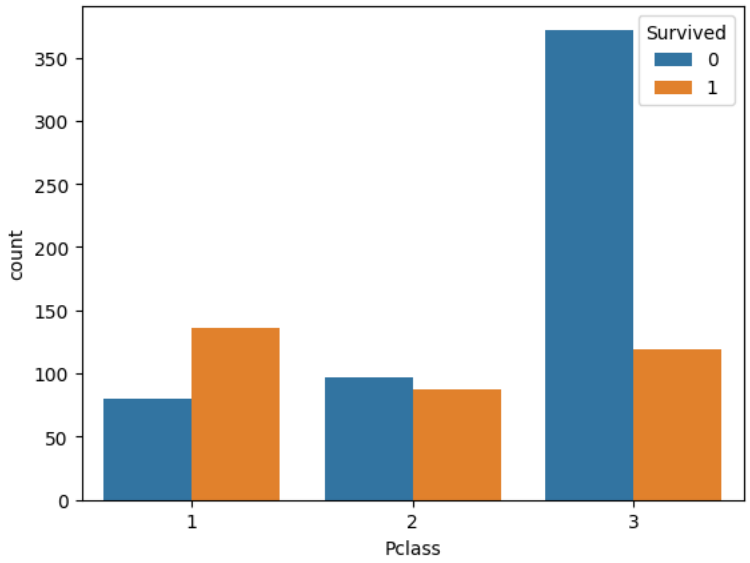

```
plt = train_data.Pclass.value_counts().plot(kind='bar')
plt.set_xlabel('Pclass')
plt.set_ylabel("Passenger count")
```

```
Text(0, 0.5, 'Passenger count')
```



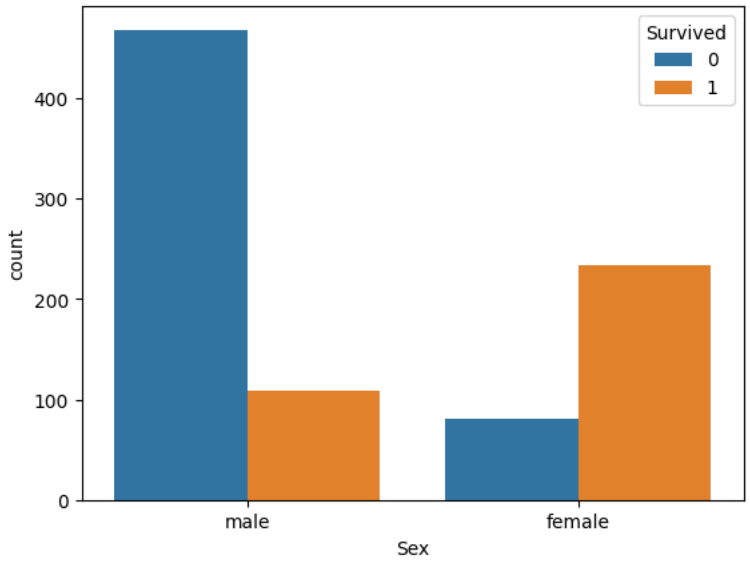

```
sns.countplot(x='Pclass', hue='Survived', data=train_data)
```

```
<Axes: xlabel='Pclass', ylabel='count'>
```




```
sns.countplot(x='Sex', hue='Survived', data=train_data)
```

```
<Axes: xlabel='Sex', ylabel='count'>
```

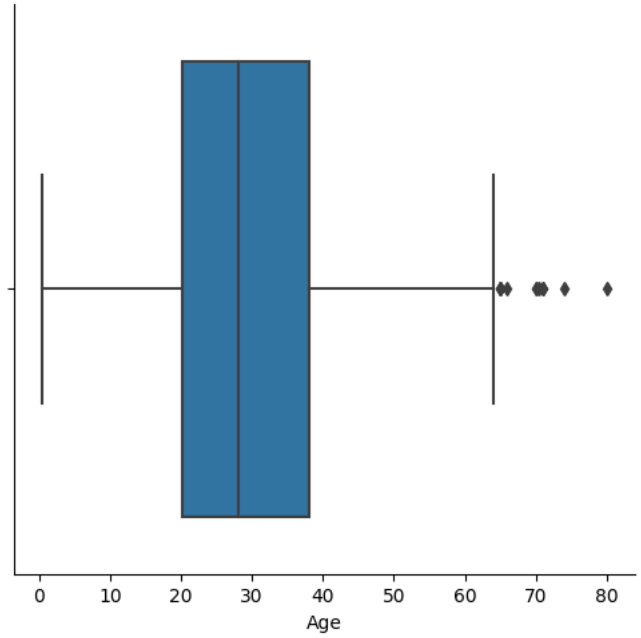



```
sns.distplot(train_data['Age'])
```

```
<ipython-input-29-24cded17e1bf>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(train_data['Age'])
<Axes: xlabel='Age', ylabel='Density'>
```

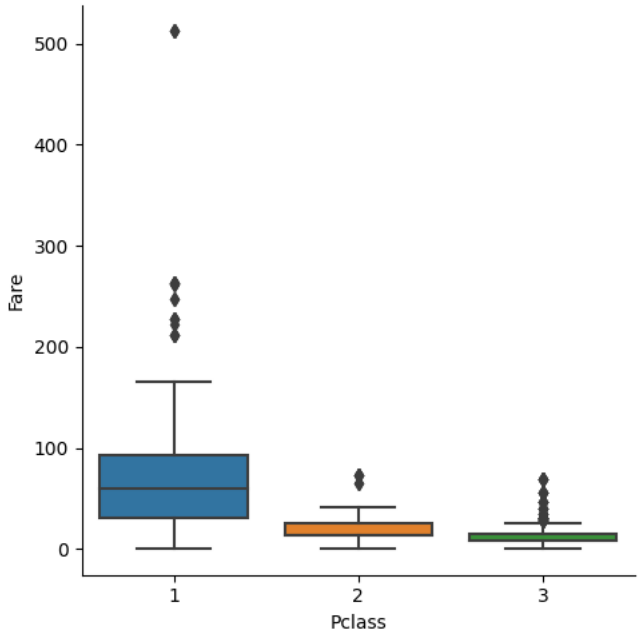
```
sns.catplot(x='Age', data = train_data, kind='box')
```

```
<seaborn.axisgrid.FacetGrid at 0x7b0fed007c40>
```



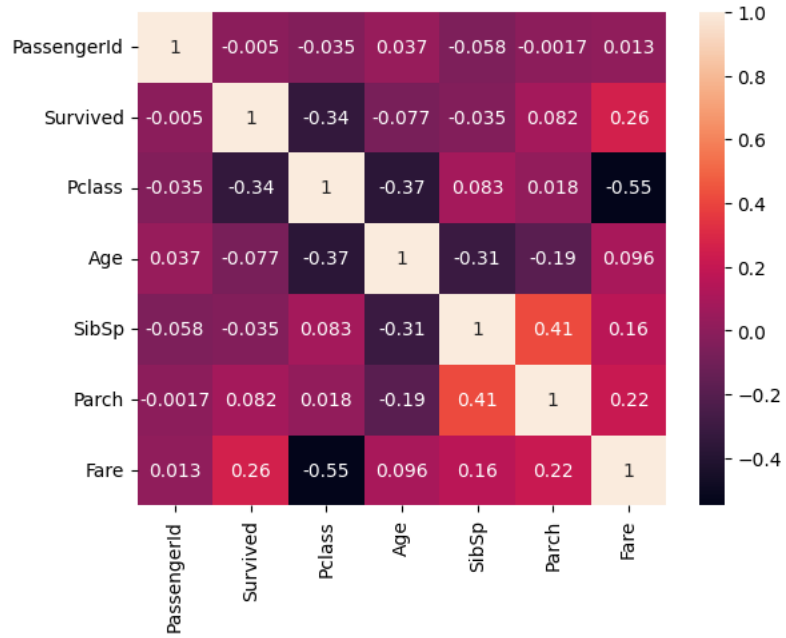

```
sns.catplot(x='Pclass', y='Fare', data = train_data, kind='box')
```

```
<seaborn.axisgrid.FacetGrid at 0x7b0fece34ac0>
```




```
sns.heatmap(train_data.corr(), annot=True)
```

```
<ipython-input-33-dc3a8a176a50>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version
  sns.heatmap(train_data.corr(), annot=True)
<Axes: >
```



Data Pre-processing

```
train_data.head(10)
```

	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 Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
				Futrelle, Mrs. Jacques						

```
drop_cols = ['Name', 'PassengerId', 'Ticket', 'Cabin']
train_data.drop(drop_cols, axis=1, inplace=True)
test_data.drop(['Name', 'Ticket', 'Cabin'], axis=1, inplace=True)
```

```
train_data.columns
```

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

```
train_data.isnull().sum()
```

```
Survived      0
Pclass        0
Sex           0
Age         177
SibSp         0
Parch         0
Fare          0
Embarked      2
dtype: int64
```

```
age_mean = int(train_data['Age'].mean())
embarked_mode = train_data['Embarked'].mode()[0]
train_data['Age'].fillna(age_mean, inplace=True)
train_data['Embarked'].fillna(embarked_mode, inplace=True)

test_data['Age'].fillna(age_mean, inplace=True)
test_data['Embarked'].fillna(embarked_mode, inplace=True)
test_data['Fare'].fillna(train_data['Fare'].mean(), inplace=True)


q1 = train_data['Age'].quantile(0.25)
q3 = train_data['Age'].quantile(0.75)
iqr = q3-q1
ul = q3 + 1.5*iqr
ll = q1 - 1.5*iqr

train_data = train_data[((train_data.Age >= ll) &
                    (train_data.Age <= ul))]
max_age_train = train_data['Age'].max()
min_age_train = train_data['Age'].min()

def cap_age(age):
    if age >= max_age_train:
        return max_age_train
    if age <= min_age_train:
        return min_age_train
    return age
test_data['Age'] = test_data['Age'].apply(lambda x: cap_age(x))


df_c1_train = train_data[train_data.Pclass == 1]
df_c2_train = train_data[train_data.Pclass == 2]
df_c3_train = train_data[train_data.Pclass == 3]

df_c1_test = test_data[test_data.Pclass == 1]
df_c2_test = test_data[test_data.Pclass == 2]
df_c3_test = test_data[test_data.Pclass == 3]

def cap_fare(fare, max_fare, min_fare):
    if fare >= max_fare:
        return max_fare
    if fare <= min_fare:
        return min_fare
    return fare

def celan_fare(df_train, df_test):
    q1 = df_train['Fare'].quantile(0.25)
    q3 = df_train['Fare'].quantile(0.75)
    iqr = q3-q1
    ul = q3 + 1.5*iqr
    ll = q1 - 1.5*iqr
    df_train = df_train[((df_train.Fare >= ll) &
                    (df_train.Fare <= ul))]
    max_fare, min_fare = df_train['Fare'].max(), df_train['Fare'].min()
    df_test['Fare'] = df_test['Fare'].apply(lambda x: cap_fare(x,
                                                               max_fare,
                                                               min_fare))
    return df_train, df_test

df1, df2 = celan_fare(df_c1_train, df_c1_test)
df3, df4 = celan_fare(df_c2_train, df_c2_test)
df5, df6 = celan_fare(df_c3_train, df_c3_test)

train_data = pd.concat([df1, df3, df5])
test_data = pd.concat([df2, df4, df6])


train_data['Sex'] = train_data['Sex'].apply(lambda x: 1 if x == "male" else 0)
train_data['Embarked'] = train_data['Embarked'].map({'S': 0, 'C': 1, 'Q':3})

test_data['Sex'] = test_data['Sex'].apply(lambda x: 1 if x == "male" else 0)
test_data['Embarked'] = test_data['Embarked'].map({'S': 0, 'C': 1, 'Q':3})


temp = train_data
temp['Embarked'] = train_data['Embarked'].map({'S': 0, 'C': 1, 'Q':3})
```

```
temp['Embarked']
```

```
1     NaN
3     NaN
6     NaN
23    NaN
30    NaN
      ..
881   NaN
882   NaN
884   NaN
888   NaN
890   NaN
Name: Embarked, Length: 753, dtype: float64
```

```
from sklearn.model_selection import train_test_split
features = train_data.drop("Survived", axis=1)#x
labels = train_data["Survived"] #y, class
x_train, x_test, y_train, y_test = train_test_split(features, labels, test_size=0.2,
                                                    random_state=123)
```

ML Models

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import cohen_kappa_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import confusion_matrix

def evaluate(model, X_test, y_test):
    pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, pred)
    precision = precision_score(y_test, pred, average='macro')
    recall = recall_score(y_test, pred, average='macro')
    f1 = f1_score(y_test, pred, average='macro')
    print(classification_report(y_test, pred))
    print('Accuracy: %f' % accuracy)
    print('Precision: %f' % precision)
    print('Recall: %f' % recall)
    print('F1 score: %f' % f1)
    print("================================================================")
```

```
DT_clf = DecisionTreeClassifier()
RF_clf = RandomForestClassifier()
```

```
# Assuming RF_clf and DT_clf are instances of RandomForestClassifier and DecisionTreeClassifier

models = {
    "RF": RF_clf,
    "DT": DT_clf
}

for name, model in models.items():
    print(name)
    # Fit the model first
    model.fit(x_train.drop(['Embarked'], axis=1), y_train)

    # Then perform hyperparameter tuning or any other evaluation
    evaluate(model, x_test.drop(['Embarked'], axis=1), y_test)
```

```
 RF
              precision    recall  f1-score   support

           0       0.69      0.69      0.69        91
```

```
                1       0.53      0.53      0.53        60

         accuracy                           0.63       151
        macro avg       0.61      0.61      0.61       151
     weighted avg       0.63      0.63      0.63       151

     Accuracy: 0.629139
     Precision: 0.612821
     Recall: 0.612821
     F1 score: 0.612821
     ============================================================
     DT
                   precision    recall  f1-score   support

                0       0.65      0.66      0.66        91
                1       0.47      0.47      0.47        60

         accuracy                           0.58       151
        macro avg       0.56      0.56      0.56       151
     weighted avg       0.58      0.58      0.58       151

     Accuracy: 0.582781
     Precision: 0.563375
     Recall: 0.563004
     F1 score: 0.563163
     ============================================================
```

```
x_test.drop(['Embarked'], axis=1).columns
```

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

```
test_data.drop(['Embarked'], axis=1).columns
```

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

```
final_model = RandomForestClassifier()
final_model.fit(x_train.drop(['Embarked'], axis=1), y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

Prediction

```
pred = final_model.predict(test_data.drop(['PassengerId', 'Embarked'], axis=1))
```

```
test_data['Survived'] = pred
```

```
submission = test_data[['PassengerId', 'Survived']]
```

```
submission.to_csv("submission.csv", index=False)
```