```
#import all libraries
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, roc_auc_score, roc_curve
from sklearn.impute import SimpleImputer
import matplotlib.pyplot as plt
```

```
#load the datasets
trainingData = pd.read_csv("/content/train.csv")
```

```
# viewing the data sample
trainingData.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

Next steps: Generate code with trainingData | View recommended plots | New interactive sheet

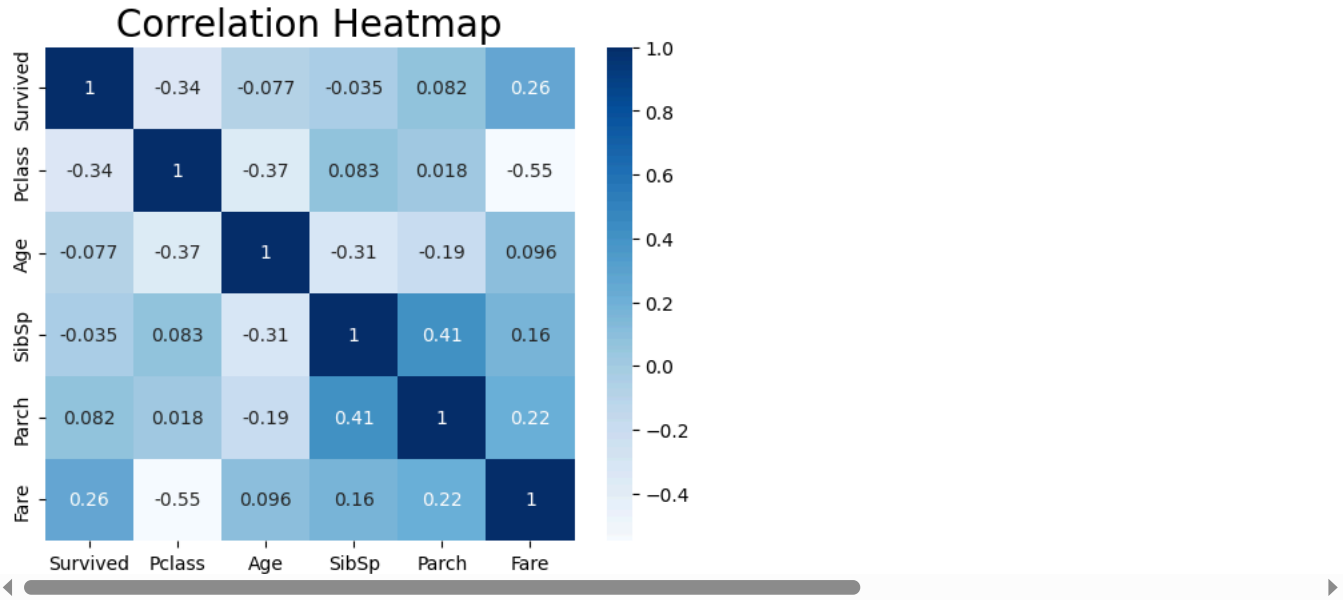
```
# show all columns

X = ['Survived', 'Pclass', 'Age', 'SibSp',
       'Parch', 'Fare']
X_features = trainingData[X]
trainingData.columns
```

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

```
#correlation heatmap
heatmap =  sns.heatmap(X_features.corr(), cmap = "Blues", annot = True)
heatmap.set_title("Correlation Heatmap", fontdict={'fontsize':20})
```

```
Text(0.5, 1.0, 'Correlation Heatmap')
```




```
#feature selection
"""
    Pclass : rejected
    Age : Rejected
    SibSp : Rejected
    Fare : Accepted
    Parch: Accepted
"""
```

```
#
Final_X = ['Fare','Parch']
Final_X_features = trainingData[Final_X]
Y = trainingData['Survived']
```

```
#seperate the testing data
testingData = pd.read_csv("/content/test.csv")
testX = testingData[Final_X]

testX.describe()
```

	Fare	Parch
count	417.000000	418.000000
mean	35.627188	0.392344
std	55.907576	0.981429
min	0.000000	0.000000
25%	7.895800	0.000000
50%	14.454200	0.000000
75%	31.500000	0.000000
max	512.329200	9.000000

```
imputer = SimpleImputer(strategy = 'mean')

testX = pd.DataFrame(imputer.fit_transform(testX), columns = testX.columns)

print("Null Values after imputation: ")
testX
```

```
Null Values after imputation:
```

	Fare	Parch
0	7.8292	0.0
1	7.0000	0.0
2	9.6875	0.0
3	8.6625	0.0
4	12.2875	1.0
...	...	...
413	8.0500	0.0
414	108.9000	0.0
415	7.2500	0.0
416	8.0500	0.0
417	22.3583	1.0

418 rows × 2 columns

Next steps: Generate code with testX | View recommended plots | New interactive sheet

```
#classifier model
model2 = DecisionTreeClassifier(random_state=1)
model2.fit(Final_X_features, Y)

preds2 = model2.predict(testX)
preds2
```

```
array([0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0,
       1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0,
       0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1,
       1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0,
       0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1,
       1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0,
```

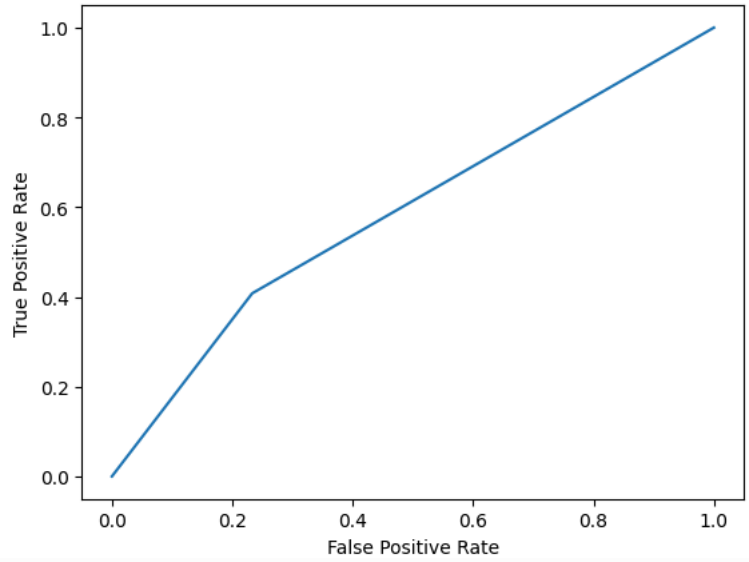
```
       1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0,
       0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0,
       1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1])
```

```
#check accuracy
trueResults = pd.read_csv("/content/gender_submission.csv")
testY = trueResults['Survived']
conf = confusion_matrix(testY, preds2)
accu = accuracy_score(testY, preds2)

#roc_aoc score
rScore = roc_auc_score(testY, preds2)

#roc curve
fpr, tpr, thresholds = roc_curve(testY, preds2)
plt.plot(fpr,tpr)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
```

```
Text(0, 0.5, 'True Positive Rate')
```




```
print(f'Accuracy Score: {accu}')
print(f"ROC_AUC score : {rScore}")
print(f"Confusion Matrix : \n{conf}")
```

```
Accuracy Score: 0.6363636363636364
ROC_AUC score : 0.587406015037594
Confusion Matrix :
[[204  62]
 [ 90  62]]
```