```
from google.colab import files


uploaded = files.upload()
```

Choose Files 3 files

- **train.csv**(text/csv) - 61194 bytes, last modified: 7/12/2024 - 100% done
- **test.csv**(text/csv) - 28629 bytes, last modified: 7/12/2024 - 100% done
- **gender_submission.csv**(text/csv) - 3258 bytes, last modified: 7/12/2024 - 100% done

```
Saving train.csv to train.csv
Saving test.csv to test.csv
Saving gender_submission.csv to gender_submission.csv
```

```
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.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_validate
from sklearn.model_selection import cross_val_score
from sklearn import metrics

from sklearn import preprocessing

from yellowbrick.classifier import ConfusionMatrix
from yellowbrick.classifier import ClassificationReport
from yellowbrick.classifier import ROCAUC
plt.style.use('ggplot')

import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

```
dataset = pd.read_csv('train.csv')
dataset.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 `dataset` View recommended plots

```
dataset.describe(include = "all")
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
count	891.000000	891.000000	891.000000	891	891	714.000000	891.000000	891.000000	891	891.000000	204	889
unique	NaN	NaN	NaN	891	2	NaN	NaN	NaN	681	NaN	147	3
top	NaN	NaN	NaN	Braund, Mr. Owen Harris	male	NaN	NaN	NaN	347082	NaN	B96 B98	S
freq	NaN	NaN	NaN	1	577	NaN	NaN	NaN	7	NaN	4	644
mean	446.000000	0.383838	2.308642	NaN	NaN	29.699118	0.523008	0.381594	NaN	32.204208	NaN	NaN
std	257.353842	0.486592	0.836071	NaN	NaN	14.526497	1.102743	0.806057	NaN	49.693429	NaN	NaN
min	1.000000	0.000000	1.000000	NaN	NaN	0.420000	0.000000	0.000000	NaN	0.000000	NaN	NaN
25%	223.500000	0.000000	2.000000	NaN	NaN	20.125000	0.000000	0.000000	NaN	7.910400	NaN	NaN
50%	446.000000	0.000000	3.000000	NaN	NaN	28.000000	0.000000	0.000000	NaN	14.454200	NaN	NaN

```
dataset.shape
```

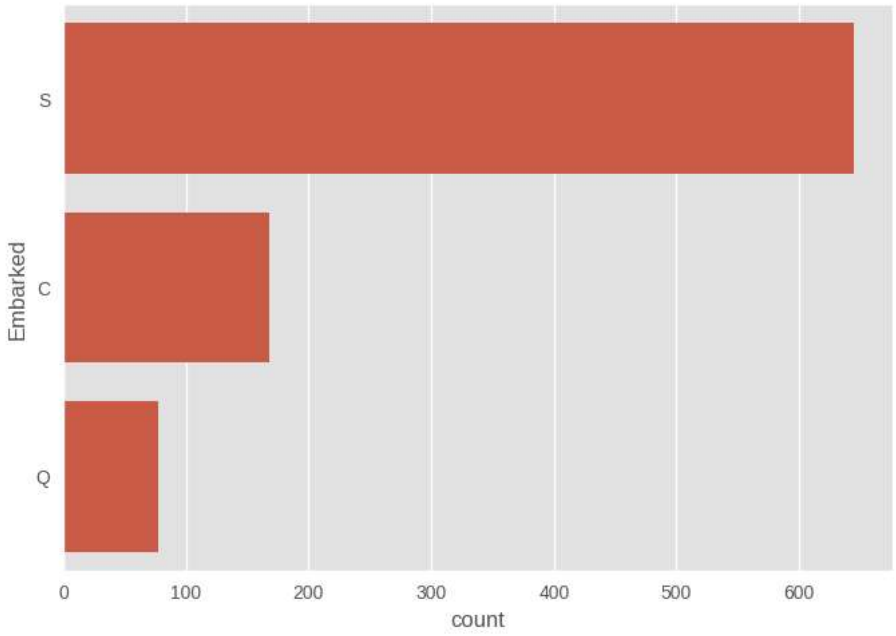
```
(891, 12)
```

```
dataset.isnull().sum(axis=0)
```

```
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
```

```
sns.countplot(dataset['Embarked'])
```

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




```
dataset = dataset.fillna({"Embarked": "S"})
```

```
dataset = pd.get_dummies(dataset, columns=['Sex'])
dataset.head()
```

	PassengerId	Survived	Pclass	Name	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Sex_female	Sex_male
0	1	0	3	Braund, Mr. Owen Harris	22.0	1	0	A/5 21171	7.2500	NaN	S	False	True
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	38.0	1	0	PC 17599	71.2833	C85	C	True	False
2	3	1	3	Heikkinen, Miss. Laina	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	True	False

Next steps: Generate code with dataset View recommended plots

```
dataset = pd.get_dummies(dataset, columns=['Embarked'])
dataset.head()
```

	PassengerId	Survived	Pclass	Name	Age	SibSp	Parch	Ticket	Fare	Cabin	Sex_female	Sex_male	Embarked_C	Embarked_
0	1	0	3	Braund, Mr. Owen Harris	22.0	1	0	A/5 21171	7.2500	NaN	False	True	False	Fals
1	2	1	1	Cumings, Mrs. John Bradley (Florence	38.0	1	0	PC 17599	71.2833	C85	True	False	True	Fals

Next steps: Generate code with dataset View recommended plots

```
feat_names = ['Pclass', 'Sex_male', 'Sex_female', 'Embarked_C', 'Embarked_Q', 'Embarked_S', 'Parch', 'SibSp', 'Fare']
targ_names = ['Dead (0)', 'Survived (1)'] # 0 - Dead, 1 - Survived

train_class = dataset[['Survived']]
train_feature = dataset[feat_names]
train_feature.head()
```

	Pclass	Sex_male	Sex_female	Embarked_C	Embarked_Q	Embarked_S	Parch	SibSp	Fare
0	3	True	False	False	False	True	0	1	7.2500
1	1	False	True	True	False	False	0	1	71.2833
2	3	False	True	False	False	True	0	0	7.9250
3	1	False	True	False	False	True	0	1	53.1000
4	3	True	False	False	False	True	0	0	8.0500

Next steps: Generate code with `train_feature` View recommended plots

```
clf = DecisionTreeClassifier(random_state=0)
scoring = {'acc': 'accuracy',
           'prec_macro': 'precision_macro',
           'rec_macro': 'recall_macro',
           'f1_macro': 'f1_macro'}
scores = cross_validate(clf, train_feature, train_class, cv=10, scoring=scoring)
# print(scores.keys())

print ('Accuracy score : %.3f' % scores['test_acc'].mean())
print ('Precisoin score : %.3f' % scores['test_prec_macro'].mean())
print ('Recall score : %.3f' % scores['test_rec_macro'].mean())
print ('F1 score : %.3f' % scores['test_f1_macro'].mean())
```

```
Accuracy score : 0.805
Precisoin score : 0.798
Recall score : 0.785
F1 score : 0.789
```

```
para_grid = {
    'min_samples_split' : range(10,500,20),
    'max_depth': range(1,20,2),
    'criterion': ("gini", "entropy")
}

clf_tree = DecisionTreeClassifier()
clf_cv = GridSearchCV(clf_tree,
                      para_grid,
                      scoring='accuracy',
                      cv=5,
                      n_jobs=-1)
clf_cv.fit(train_feature,train_class)

best_parameters = clf_cv.best_params_
print(best_parameters)
```

```
{'criterion': 'gini', 'max_depth': 11, 'min_samples_split': 50}
```

```
clf = clf_cv.best_estimator_
scoring = {'acc': 'accuracy',
           'prec_macro': 'precision_macro',
           'rec_macro': 'recall_macro',
           'f1_macro': 'f1_macro'}
scores = cross_validate(clf, train_feature, train_class, cv=10, scoring=scoring)
#print(scores.keys())

print ('Accuracy score : %.3f' % scores['test_acc'].mean())
print ('Precisoin score : %.3f' % scores['test_prec_macro'].mean())
print ('Recall score : %.3f' % scores['test_rec_macro'].mean())
print ('F1 score score : %.3f' % scores['test_f1_macro'].mean())
```

```
Accuracy score : 0.800
Precisoin score : 0.804
Recall score : 0.770
F1 score score : 0.778
```

```
X_train, X_test, y_train, y_test = train_test_split(train_feature, train_class, test_size=0.33)
print (str(X_train.shape) +","+ str(y_train.shape))
print (str(X_test.shape) +","+ str(y_test.shape))
```

```
(596, 9),(596, 1)
(295, 9),(295, 1)
```

```
clf2 = clf_cv.best_estimator_
clf2.fit(X_train,y_train)
predictions = clf2.predict(X_test)
print(metrics.classification_report(y_test,predictions, target_names=targ_names, digits=3))
```

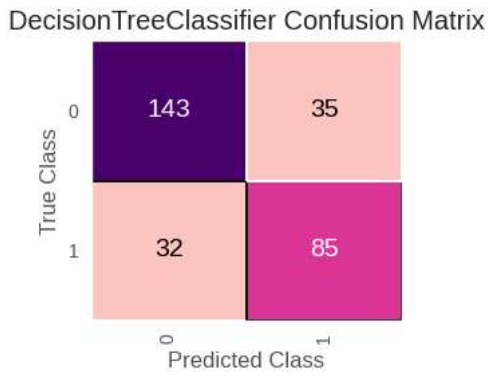
```
              precision    recall  f1-score   support

        Dead (0)      0.817     0.803     0.810       178
    Survived (1)      0.708     0.726     0.717       117

        accuracy                          0.773       295
       macro avg      0.763     0.765     0.764       295
    weighted avg      0.774     0.773     0.773       295
```

```
fig, ax = plt.subplots(figsize=(7,3))
visualizer = ClassificationReport(clf2, classes=targ_names, support=True, cmap='RdPu')
visualizer.score(X_test, y_test)
for label in visualizer.ax.texts:
    label.set_size(14)
g = visualizer.poof()
```

```
/usr/local/lib/python3.10/dist-packages/yellowbrick/classifier/base.py:232: YellowbrickWarning: could not determine class_counts_ fr
  warnings.warn(
```


```
fig, ax = plt.subplots(figsize=(3,3))
cm = ConfusionMatrix(clf2, classes=[0, 1], cmap='RdPu')
cm.score(X_test, y_test)
for label in cm.ax.texts:
    label.set_size(14)
cm.poof()
```

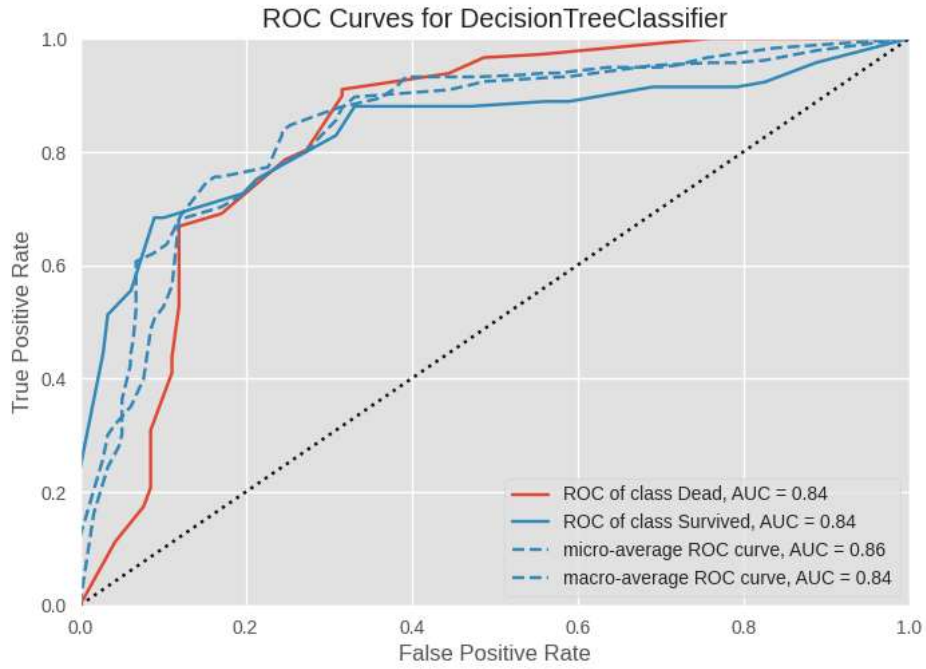
```
/usr/local/lib/python3.10/dist-packages/yellowbrick/classifier/base.py:232: YellowbrickWarning: could not determine class_counts_ fr
  warnings.warn(
```




```
<Axes: title={'center': 'DecisionTreeClassifier Confusion Matrix'}, xlabel='Predicted Class', ylabel='True Class'>
```

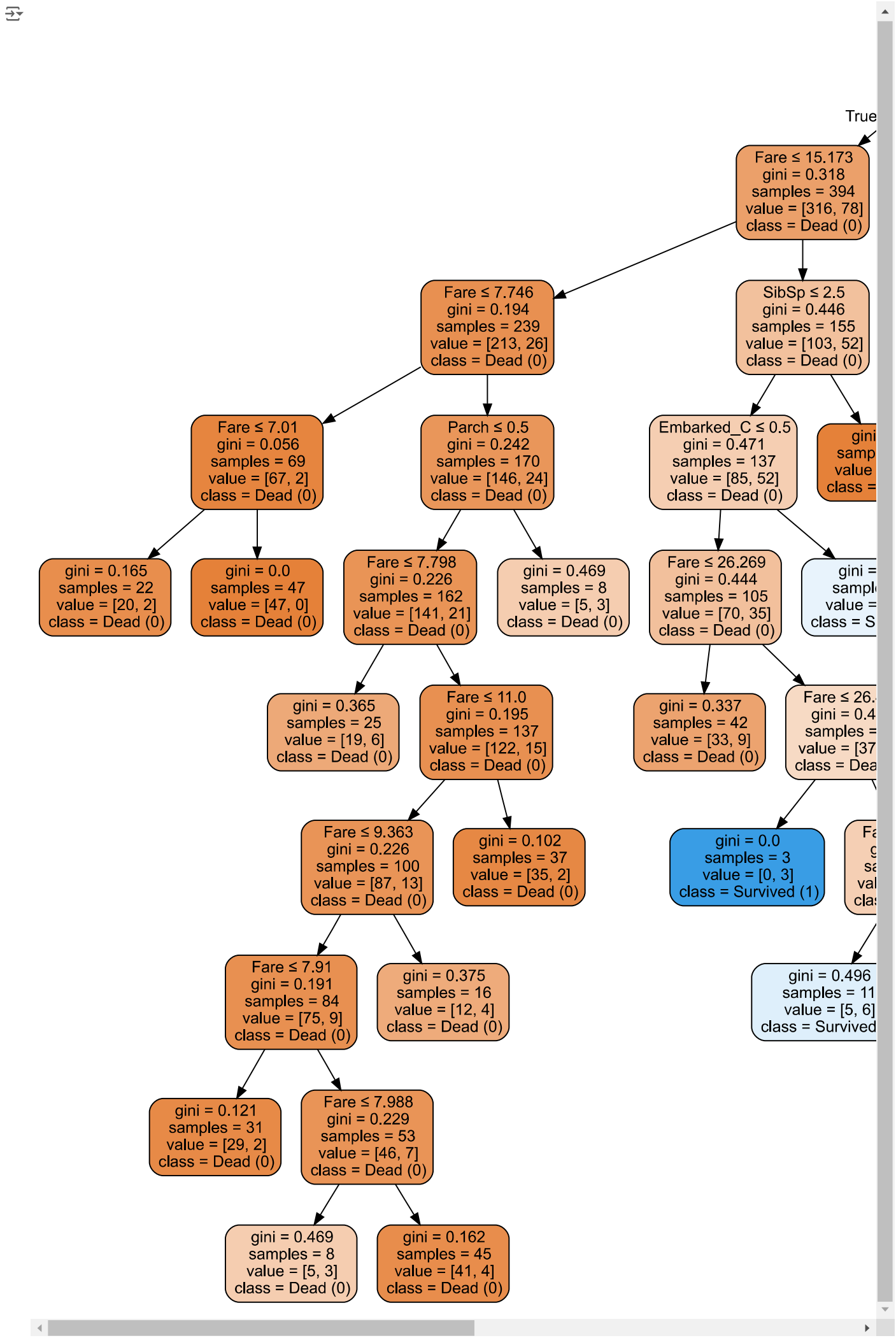
```
modelviz = clf_cv.best_estimator_
visualizer = ROCAUC(modelviz, classes=["Dead", "Survived"])

visualizer.fit(X_train, y_train)
visualizer.score(X_test, y_test)        # Evaluate the model on the test data
visualizer.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but DecisionTreeClass
  warnings.warn(
```

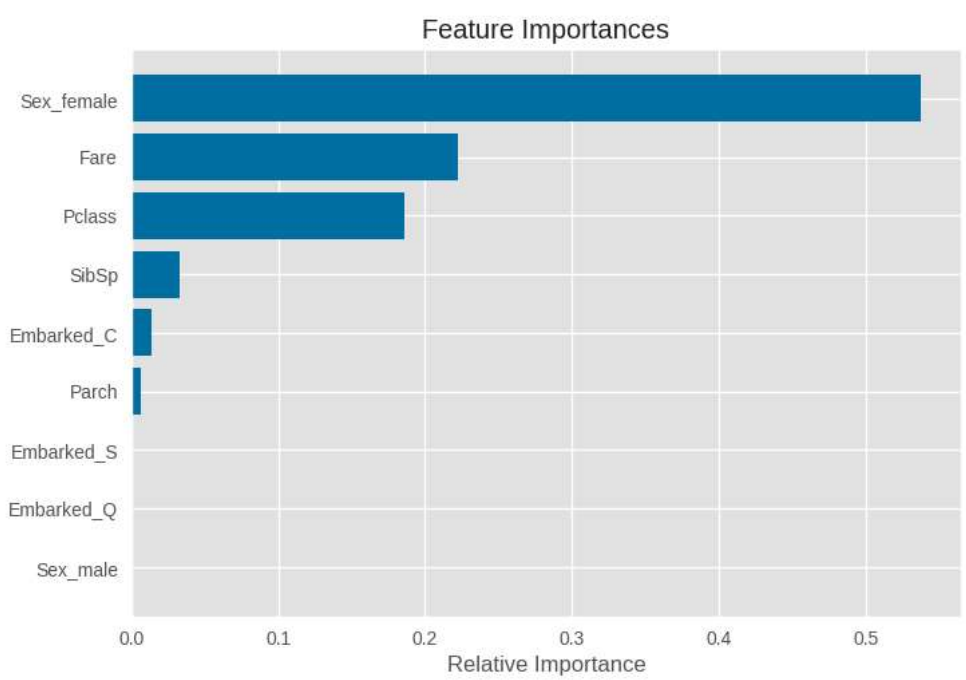



```
<Axes: title={'center': 'ROC Curves for DecisionTreeClassifier'}, xlabel='False Positive Rate', ylabel='True Positive Rate'>
```

```
import graphviz
from sklearn.tree import DecisionTreeClassifier, export_graphviz

data = export_graphviz(clf,out_file=None,feature_names=feat_names,class_names=targ_names,
                       filled=True, rounded=True,
                       special_characters=True)
graph = graphviz.Source(data)
graph
```


True
Fare ≤ 15.173
gini = 0.318
samples = 394
value = [316, 78]
class = Dead (0)
Fare ≤ 7.746
gini = 0.194
samples = 239
value = [213, 26]
class = Dead (0)
SibSp ≤ 2.5
gini = 0.446
samples = 155
value = [103, 52]
class = Dead (0)
Fare ≤ 7.01
gini = 0.056
samples = 69
value = [67, 2]
class = Dead (0)
Parch ≤ 0.5
gini = 0.242
samples = 170
value = [146, 24]
class = Dead (0)
Embarked_C ≤ 0.5
gini = 0.471
samples = 137
value = [85, 52]
class = Dead (0)
gini = 0.165
samples = 22
value = [20, 2]
class = Dead (0)
gini = 0.0
samples = 47
value = [47, 0]
class = Dead (0)
Fare ≤ 7.798
gini = 0.226
samples = 162
value = [141, 21]
class = Dead (0)
gini = 0.469
samples = 8
value = [5, 3]
class = Dead (0)
Fare ≤ 26.269
gini = 0.444
samples = 105
value = [70, 35]
class = Dead (0)
gini = 0.365
samples = 25
value = [19, 6]
class = Dead (0)
Fare ≤ 11.0
gini = 0.195
samples = 137
value = [122, 15]
class = Dead (0)
gini = 0.337
samples = 42
value = [33, 9]
class = Dead (0)
Fare ≤ 9.363
gini = 0.226
samples = 100
value = [87, 13]
class = Dead (0)
gini = 0.102
samples = 37
value = [35, 2]
class = Dead (0)
gini = 0.0
samples = 3
value = [0, 3]
class = Survived (1)
Fare ≤ 7.91
gini = 0.191
samples = 84
value = [75, 9]
class = Dead (0)
gini = 0.375
samples = 16
value = [12, 4]
class = Dead (0)
gini = 0.496
samples = 11
value = [5, 6]
class = Survived
gini = 0.121
samples = 31
value = [29, 2]
class = Dead (0)
Fare ≤ 7.988
gini = 0.229
samples = 53
value = [46, 7]
class = Dead (0)
gini = 0.469
samples = 8
value = [5, 3]
class = Dead (0)
gini = 0.162
samples = 45
value = [41, 4]
class = Dead (0)

```
importances = clf.feature_importances_
indices = np.argsort(importances)

plt.title('Feature Importances')
plt.barh(range(len(indices)), importances[indices], color='b', align='center')
plt.yticks(range(len(indices)), [feat_names[i] for i in indices])
plt.xlabel('Relative Importance')
plt.show()
```




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