Ex 1: Titanic

In [1]:
```
# !pip install trelawney
```

In [2]:
```
# from google.colab import drive
# drive.mount("/content/gdrive", force_remount=True)
# %cd '/content/gdrive/My Drive/LDS6_MachineLearning/Practice_2023/Chapter5_Random_Forest/'
```

Cho dữ liệu titanic trong tập tin titanic_csv.csv

Yêu cầu: Hãy đọc dữ liệu từ tập tin này, áp dụng Random Forest để thực hiện việc xác định người trên tàu Titanic còn sống hay không dựa trên các thông tin được cung cấp.

Chi tiết:

1. Đọc dữ liệu. Chuẩn hóa dữ liệu. Từ dữ liệu tạo X gồm các thuộc tính 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare', 'embarked', y là 'survived'
2. Tạo X_train, X_test, y_train, y_test từ dữ liệu chuẩn hóa với tỷ lệ dữ liệu test là 0.3
3. Áp dụng Random Forest, Tìm kết quả
4. Kiểm tra độ chính xác
5. Tìm các thuộc tính quan trọng nhất trong tập dữ liệu
6. Trực quan hóa thuộc tính quan trọng
7. Áp dụng lại Random Forest dựa trên các thuộc tính quan trọng, tìm kết quả
8. Kiểm tra độ chính xác
9. Tự cho 1 dữ liệu X_test mới. Ví dụ như:

X_test =[[tuoi = 35, gia ve =50, gioi tinh = Male, tang lop = 3], [tuoi = 18, gia ve = 250, gioi tinh = Female, tang lop = 2]] , tìm kết quả Y test.

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import math
```

In [4]:
```
data = pd.read_csv("titanic_csv.csv", index_col=0)
```

In [5]:
```
type(data)
```

Out[5]:
```
pandas.core.frame.DataFrame
```

In [6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1309 entries, 1 to 1309
Data columns (total 12 columns):
pclass       1309 non-null int64
survived     1309 non-null int64
name         1309 non-null object
sex          1309 non-null object
age          1046 non-null float64
sibsp        1309 non-null int64
parch        1309 non-null int64
ticket       1309 non-null object
fare         1308 non-null float64
cabin        295 non-null object
embarked     1307 non-null object
home.dest    745 non-null object
dtypes: float64(2), int64(4), object(6)
memory usage: 132.9+ KB
```

In [7]:
```
data.head()
```

Out[7]:

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	home.dest
1	1	1	Allen, Miss. Elisabeth Walton	female	29.0000	0	0	24160	211.3375	B5	S	St Louis, MO
2	1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.5500	C22 C26	S	Montreal, PQ / Chesterville, ON
3	1	0	Allison, Miss. Helen Loraine	female	2.0000	1	2	113781	151.5500	C22 C26	S	Montreal, PQ / Chesterville, ON
4	1	0	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1	2	113781	151.5500	C22 C26	S	Montreal, PQ / Chesterville, ON
5	1	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1	2	113781	151.5500	C22 C26	S	Montreal, PQ / Chesterville, ON

In [8]:
```
data = data.interpolate()
```

In [9]:
```
X=data[['pclass', 'sex', 'age', 'sibsp', 'parch', 'fare', 'embarked']] # Features
y=data['survived']  # Labels
```

In [10]:
```
X = pd.get_dummies(X)
X.head()
```

Out[10]:

	pclass	age	sibsp	parch	fare	sex_female	sex_male	embarked_C	embarked_Q	embarked_S
1	1	29.0000	0	0	211.3375	1	0	0	0	1
2	1	0.9167	1	2	151.5500	0	1	0	0	1
3	1	2.0000	1	2	151.5500	1	0	0	0	1
4	1	30.0000	1	2	151.5500	0	1	0	0	1
5	1	25.0000	1	2	151.5500	1	0	0	0	1

In [11]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

In [12]:
```
from sklearn.ensemble import RandomForestClassifier
# rung bao nhieu cay?
clf=RandomForestClassifier(n_estimators=100)
```

In [13]:
```
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
```

In [14]:
```
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7786259541984732
```

In [15]:
```
# Kiểm tra độ chính xác
print("The Training prediction accuracy is: ",
      clf.score(X_train,y_train)*100,"%")
print("The Testing prediction accuracy is: ",
      clf.score(X_test,y_test)*100,"%")
# Co cach nao? giai phap nao de ket qua phu hop hon khong?
# (khi train va test chenh nhau nhieu?)
```

```
The Training prediction accuracy is:  98.58078602620087 %
The Testing prediction accuracy is:  77.86259541984732 %
```

Finding Important Features in Scikit-learn

In [16]:
```
import pandas as pd
feature_imp = pd.Series(clf.feature_importances_,
                        index = np.array(X.columns)).sort_values(ascending=False)
feature_imp
```

Out[16]:
```
age          0.303264
fare         0.263318
sex_male     0.124959
sex_female   0.110858
pclass       0.080454
sibsp        0.042827
parch        0.038482
embarked_C   0.017634
embarked_S   0.013234
embarked_Q   0.004971
dtype: float64
```

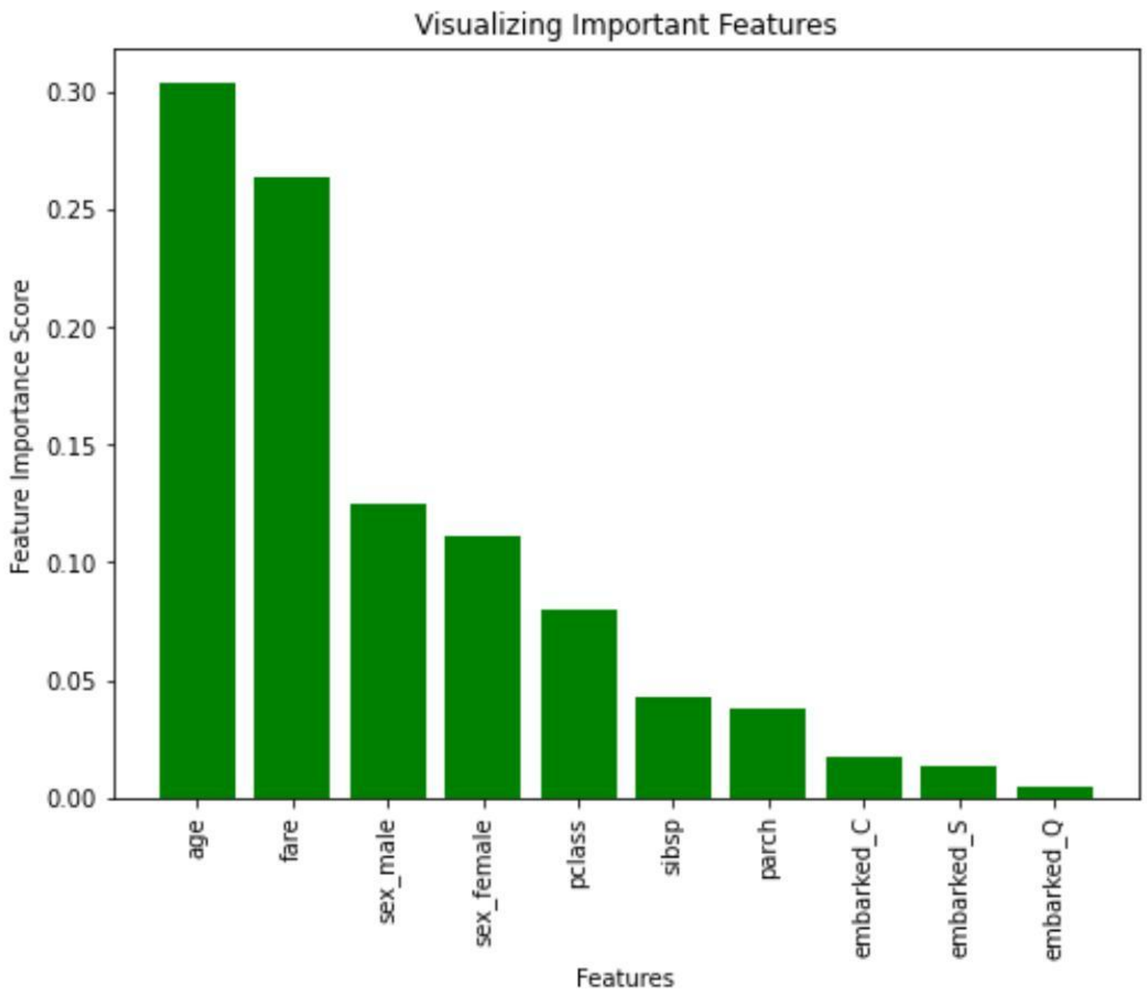
In [17]:
```
type(feature_imp)
```

Out[17]:
```
pandas.core.series.Series
```

In [18]:
```
feature_imp[feature_imp>=0.05].sum()
```

Out[18]:
```
0.882852184722256
```

In [19]:
```
import matplotlib.pyplot as plt
%matplotlib inline
# Creating a bar plot
plt.figure(figsize=(8,6))
plt.bar(feature_imp.index, feature_imp, color="g")
# Add labels to your graph
plt.xlabel('Features')
plt.ylabel('Feature Importance Score')
plt.title("Visualizing Important Features")
plt.xticks(feature_imp.index, rotation='vertical')
plt.show()
```



In [20]:
```
# Tạo lại dữ liệu huấn luyện và test sau khi bỏ đi các thuộc tính ít quan trọng hơn
X_now = X[['age', 'fare', 'sex_female', 'sex_male', 'pclass']]
y_now = data['survived']
```

In [21]:
```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X_now, y_now,
                                                    test_size=0.3)
```

In [22]:
```
clf_now=RandomForestClassifier(n_estimators=100)
clf_now.fit(X_train,y_train)
y_pred=clf_now.predict(X_test)
```

In [23]:
```
# Model Accuracy, how often is the classifier correct?
# => giảm đôi chút nhưng bỏ được các cột không liên quan
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7989821882951654
```

In [24]:
```
# Kiểm tra độ chính xác
print("The Training prediction accuracy is: ",
      clf_now.score(X_train,y_train)*100,"%")
print("The Testing prediction accuracy is: ",
      clf_now.score(X_test,y_test)*100,"%")
```

```
The Training prediction accuracy is:  98.58078602620087 %
The Testing prediction accuracy is:  79.89821882951654 %
```

In [25]:
```
from sklearn.metrics import confusion_matrix
```

In [26]:
```
confusion_matrix(y_test, y_pred, labels=[0, 1])
```

Out[26]:
```
array([[216,  32],
       [ 47,  98]], dtype=int64)
```

In [27]:
```
# Đánh giá model
from sklearn. metrics import classification_report, roc_auc_score, roc_curve
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.82      0.87      0.85       248
           1       0.75      0.68      0.71       145

    accuracy                           0.80       393
   macro avg       0.79      0.77      0.78       393
weighted avg       0.80      0.80      0.80       393
```

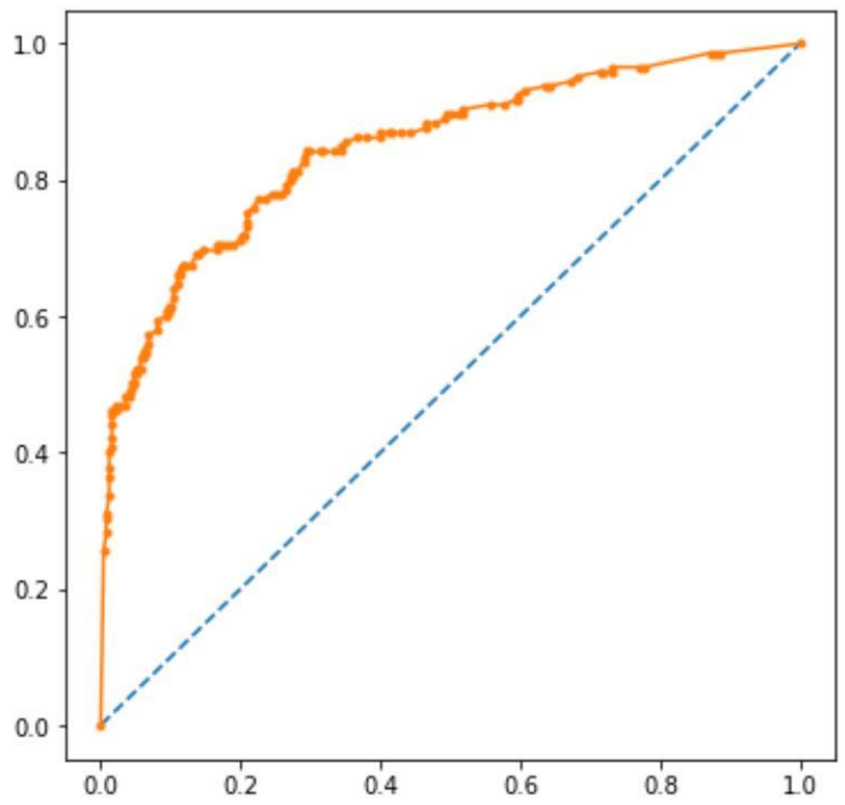
In [28]:
```
y_prob = clf_now.predict_proba(X_test)
y_probs = y_prob[:, 1]
#y_probs
```

In [29]:
```
roc_auc_score(y_test, y_probs)
```

Out[29]:
```
0.847983870967742
```

In [30]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(6,6))
# calculate roc curve
fpr, tpr, thresholds = roc_curve(y_test, y_probs)
```

```
# plot no skill
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(fpr, tpr, marker='.')
# show the plot
plt.show()
```



Nhận xét:

- Mô hình có Train R^2 cao hơn (0.98) so với Test (0.8) ở cả 2 thử nghiệm: full tính năng và bỏ bớt tính năng
- ROC AUC score 0.84
- => Mô hình tạm được, nhưng chưa phải là mô hình tốt nhất => Có giải pháp nào không?

In [31]:
```
# tuoi, gia ve, gioi tinh, tang lop
clf_now.predict([[35, 50, 0, 1, 3], [18, 250, 1, 0, 2]])
```

Out[31]:
```
array([0, 1], dtype=int64)
```

Surogate explainer

- Tham khảo thêm

In [32]:
```
from trelawney.surrogate_explainer import SurrogateExplainer
from sklearn.tree import DecisionTreeClassifier
```

In [33]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

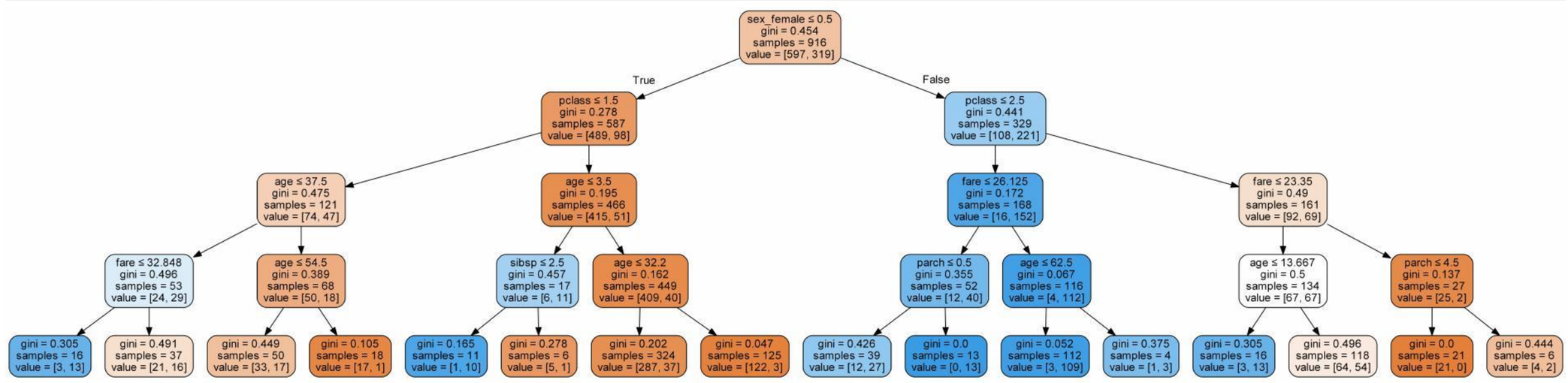
In [34]:
```
explainer = SurrogateExplainer(DecisionTreeClassifier(max_depth=4))
explainer.fit(clf, X_train, y_train)
```

Out[34]:
```
<trelawney.surrogate_explainer.SurrogateExplainer at 0x2caee133748>
```

In [35]:
```
from IPython.display import Image
```

In [36]:
```
explainer.plot_tree(out_path='tree_viz')
Image('tree_viz.png', width=1200, height=400)
```

Out[36]:



In [37]:
```
explainer.adequation_score()
```

Out[37]:
```
0.8318777292576419
```

In [38]:
```
from sklearn import metrics
```

In [39]:
```
explainer.adequation_score(metric=metrics.roc_auc_score)
```

Out[39]:
```
0.775407864820445
```