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

```
data = pd.read_csv("gender_submission.csv")
```

```
test = pd.read_csv("test.csv")
train = pd.read_csv("train.csv")
```

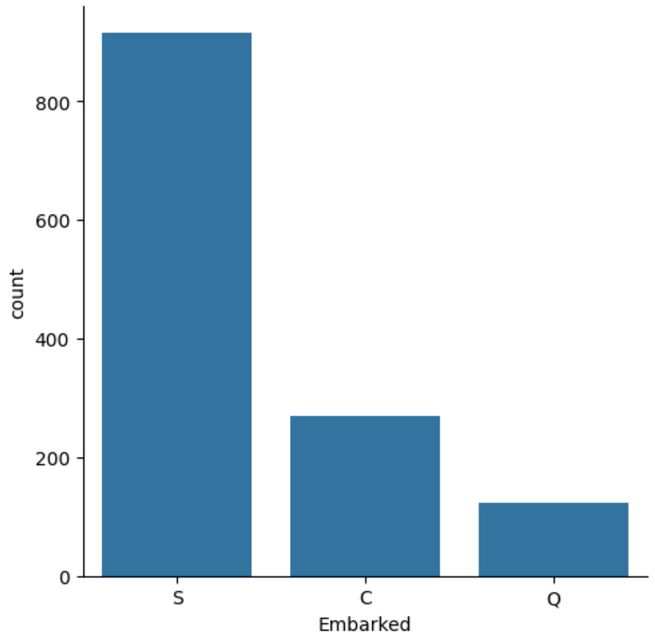
```
all = pd.concat([train, test], sort = False)
all.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1309 entries, 0 to 417
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  1309 non-null   int64
 1   Survived     891 non-null    float64
 2   Pclass       1309 non-null   int64
 3   Name         1309 non-null   object
 4   Sex          1309 non-null   object
 5   Age          1046 non-null   float64
 6   SibSp        1309 non-null   int64
 7   Parch        1309 non-null   int64
 8   Ticket       1309 non-null   object
 9   Fare         1308 non-null   float64
 10  Cabin        295 non-null    object
 11  Embarked     1307 non-null   object
dtypes: float64(3), int64(4), object(5)
memory usage: 132.9+ KB
```

```
#Fill Missing numbers with median
all['Age'] = all['Age'].fillna(value=all['Age'].median())
all['Fare'] = all['Fare'].fillna(value=all['Fare'].median())
```

```
all = all.reset_index()
sns.catplot(x = 'Embarked', kind = 'count', data = all)
```

```
<seaborn.axisgrid.FacetGrid at 0x7ef3cfeb4b80>
```



```
all.loc[ all['Age'] <= 16, 'Age'] = 0
all.loc[(all['Age'] > 16) & (all['Age'] <= 32), 'Age'] = 1
all.loc[(all['Age'] > 32) & (all['Age'] <= 48), 'Age'] = 2
all.loc[(all['Age'] > 48) & (all['Age'] <= 64), 'Age'] = 3
all.loc[ all['Age'] > 64, 'Age'] = 4
```

```
import re
def get_title(name):
    title_search = re.search(' ([A-Za-z]+\.)', name)

    if title_search:
        return title_search.group(1)
    return ""


all['Title'] = all['Name'].apply(get_title)
all['Title'].value_counts()
```

```
Mr.          757
Miss.        260
Mrs.         197
Master.       61
Rev.           8
Dr.            8
Col.           4
Mlle.          2
Major.         2
Ms.            2
Lady.          1
Sir.           1
Mme.           1
Don.           1
Capt.          1
Countess.      1
Jonkheer.      1
Dona.          1
Name: Title, dtype: int64
```

```
all['Title'] = all['Title'].replace(['Capt.', 'Dr.', 'Major.', 'Rev.'], 'Officer.')
all['Title'] = all['Title'].replace(['Lady.', 'Countess.', 'Don.', 'Sir.', 'Jonkheer.', 'Dona.'], 'Royal.')
all['Title'] = all['Title'].replace(['Mlle.', 'Ms.'], 'Miss.')
all['Title'] = all['Title'].replace(['Mme.'], 'Mrs.')
all['Title'].value_counts()
```

```
Mr.         757
Miss.       264
Mrs.        198
Master.      61
Officer.     19
Royal.        6
Col.          4
Name: Title, dtype: int64
```

```
all['Cabin'] = all['Cabin'].fillna('Missing')
all['Cabin'] = all['Cabin'].str[0]
all['Cabin'].value_counts()
```

```
M    1014
C      94
B      65
D      46
E      41
A      22
F      21
G       5
T       1
Name: Cabin, dtype: int64
```

```
all['Family_Size'] = all['SibSp'] + all['Parch'] + 1
all['IsAlone'] = 0
all.loc[all['Family_Size']==1, 'IsAlone'] = 1
all.head()
```

	index	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Title	Family_Size
0	0	1	0.0	3	Braund, Mr. Owen Harris	male	1.0	1	0	A/5 21171	7.2500	M	S	Mr.	2
1	1	2	1.0	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	2.0	1	0	PC 17599	71.2833	C	C	Mrs.	2
2	2	3	1.0	3	Heikkinen, Miss. Laina	female	1.0	0	0	STON/O2. 3101282	7.9250	M	S	Miss.	1
					Futrelle, Mrs.										

Next steps: View recommended plots

```
all_1 = all.drop(['Name', 'Ticket'], axis = 1)
all_dummies = pd.get_dummies(all_1, drop_first = True)
all_train = all_dummies[all_dummies['Survived'].notna()]
all_test = all_dummies[all_dummies['Survived'].isna()]
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(all_train.drop(['PassengerId','Survived'],axis=1),
                                                    all_train['Survived'], test_size=0.30,
                                                    random_state=101, stratify = all_train['Survived'])
```

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

```
ada = AdaBoostClassifier(DecisionTreeClassifier(),n_estimators=100, random_state=0)
ada.fit(X_train,y_train)
```

```
      AdaBoostClassifier
estimator: DecisionTreeClassifier
   DecisionTreeClassifier
```

```
predictions = ada.predict(X_test)
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

         0.0       0.78      0.82      0.80       165
         1.0       0.68      0.63      0.66       103

    accuracy                           0.75       268
   macro avg       0.73      0.72      0.73       268
weighted avg       0.74      0.75      0.74       268
```

```
print (f'Train Accuracy - : {ada.score(X_train,y_train):.3f}')
print (f'Test Accuracy - : {ada.score(X_test,y_test):.3f}')
```

```
Train Accuracy - : 1.000
Test Accuracy - : 0.746
```

```
TestForPred = all_test.drop(['PassengerId', 'Survived'], axis = 1)
```

```
t_pred = ada.predict(TestForPred).astype(int)


PassengerId = all_test['PassengerId']


adaSub = pd.DataFrame({'PassengerId': PassengerId, 'Survived':t_pred })
```