Overview

The sinking of the RMS Titanic is one of the most infamous shipwrecks in history. On April 15, 1912, during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew. This sensational tragedy shocked the international community and led to better safety regulations for ships.

One of the reasons that the shipwreck led to such loss of life was that there were not enough lifeboats for the passengers and crew. Although there was some element of luck involved in surviving the sinking, some groups of people were more likely to survive than others, such as women, children, and the upper-class.

In this challenge, we target to complete the analysis of what sorts of people were likely to survive.

Importing Libraries

In [1]:

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

from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC

import warnings
warnings.filterwarnings("ignore")

sns.set(rc={'figure.figsize':(12, 10)})
```

In [4]:

```
data=pd.read_csv("titanic_data.csv")
data.head(10)
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Emb
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	

Types of Features :

Categorical - Sex, and Embarked.

Continuous - Age, Fare

Discrete - SibSp, Parch.

Alphanumeric - Cabin

In [5]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [8]:

```
data.isnull().sum()
```

Out[8]:

```
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
```

In [9]:

```
data.describe()
```

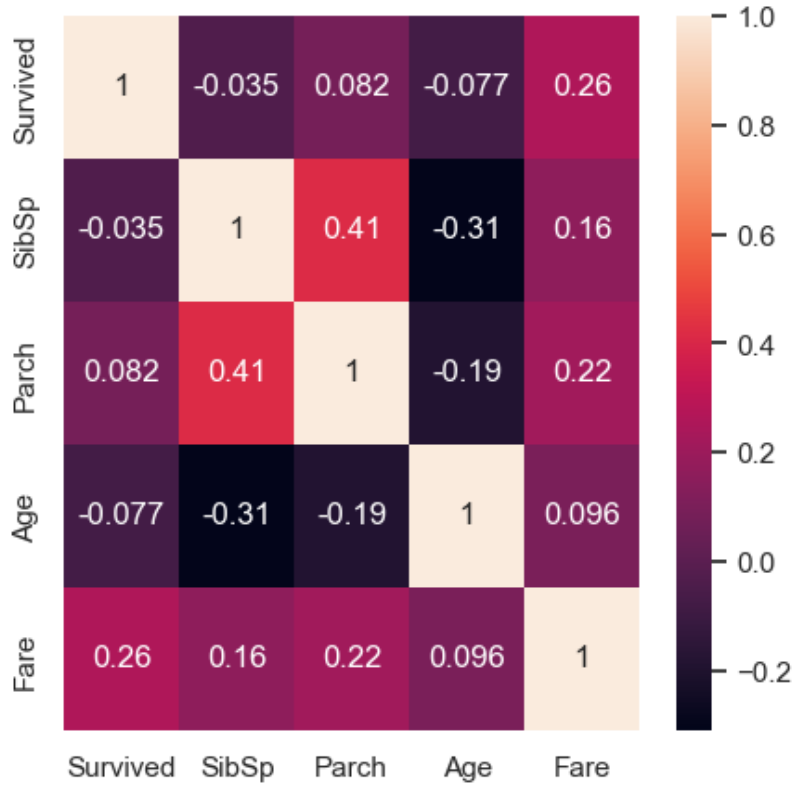
Out[9]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

Numerical value analysis

In [11]:

```
plt.figure(figsize=(5,5))
heatmap = sns.heatmap(data[["Survived","SibSp","Parch","Age","Fare"]].corr(), annot=True)
```



sibsp - Number of siblings / spouses aboard the Titanic

In [12]:

```
data['SibSp'].nunique()
```

Out[12]:

```
7
```

In [13]:

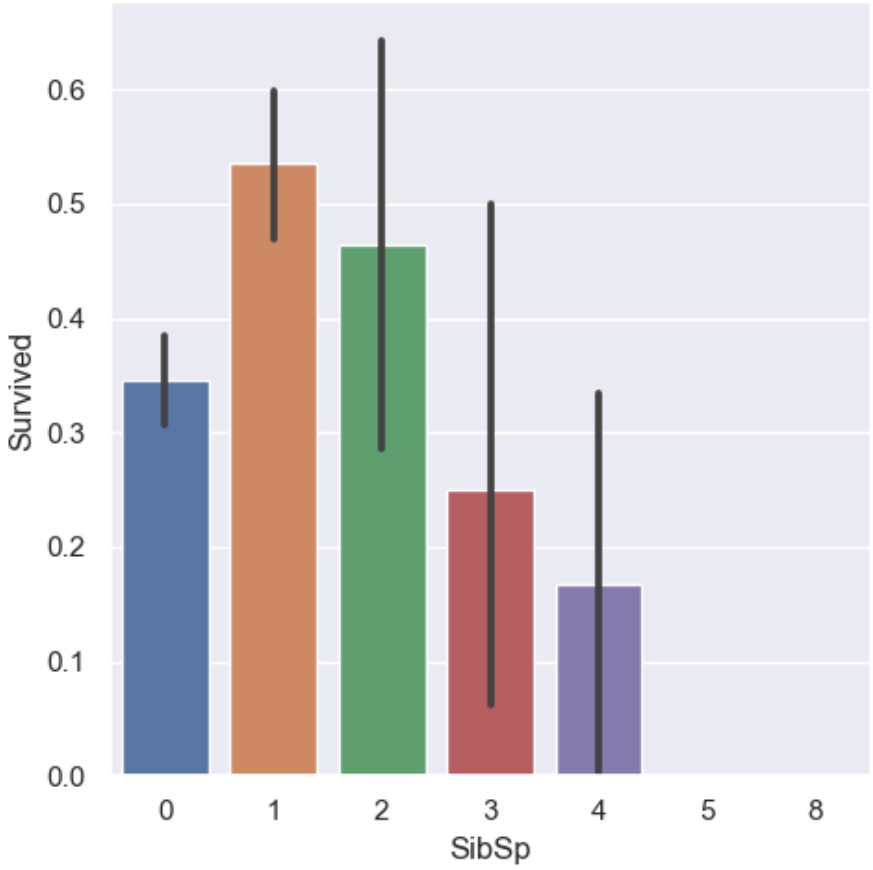
```
data['SibSp'].unique()
```

Out[13]:

```
array([1, 0, 3, 4, 2, 5, 8], dtype=int64)
```

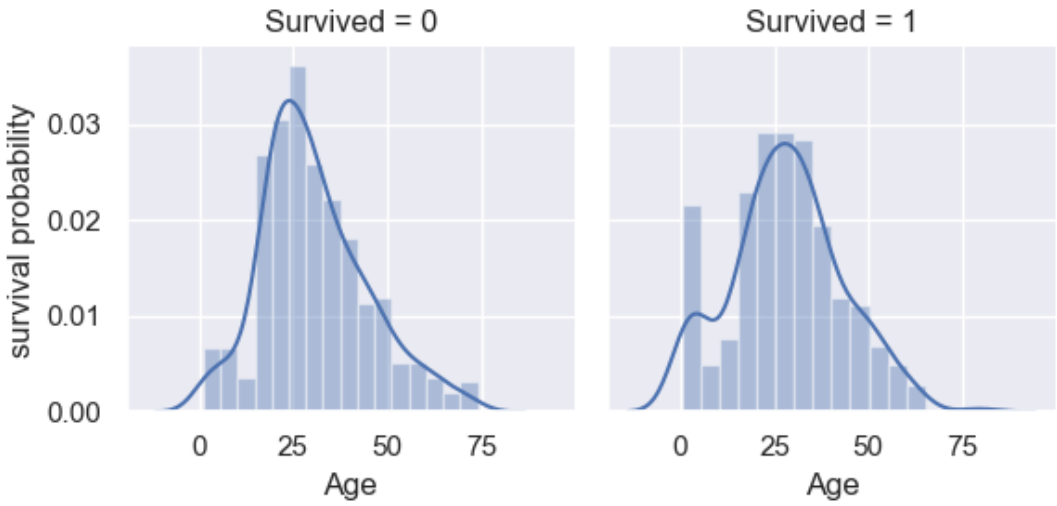
In [24]:

```
bargraph_sibsp = sns.catplot(x = "SibSp", y = "Survived", data = data,kind="bar")
```



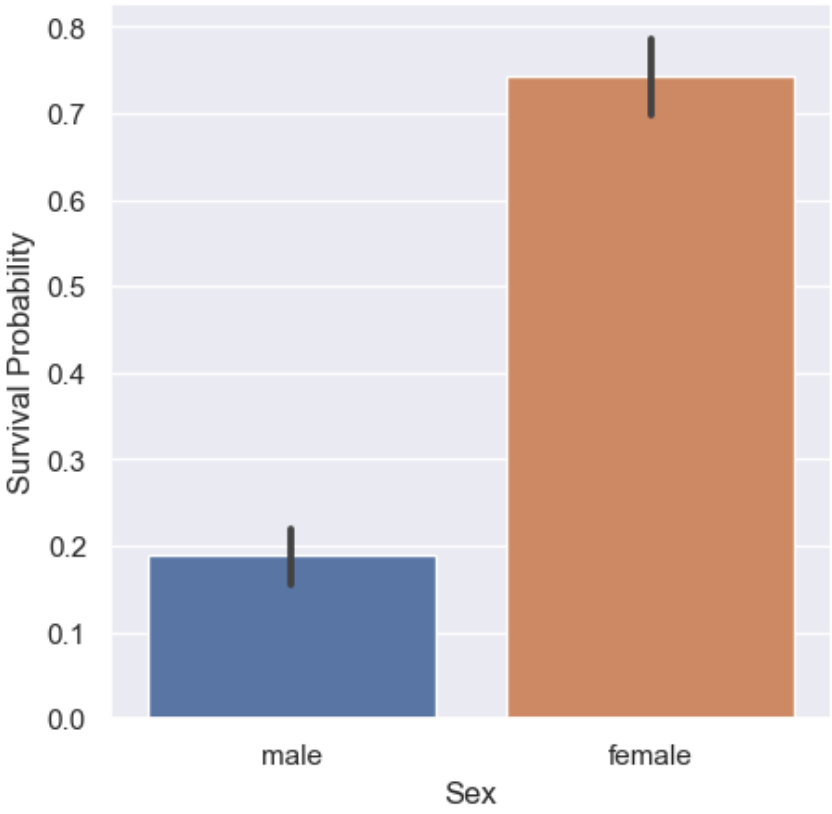
In [26]:

```
age_visual = sns.FacetGrid(data, col = 'Survived')
age_visual = age_visual.map(sns.distplot, "Age")
age_visual = age_visual.set_ylabels("survival probability")
```



In [27]:

```
import matplotlib.pyplot as plt
plt.figure(figsize=(5,5))
age_plot = sns.barplot(x = "Sex",y = "Survived", data = data)
age_plot = age_plot.set_ylabel("Survival Probability")
```



In [28]:

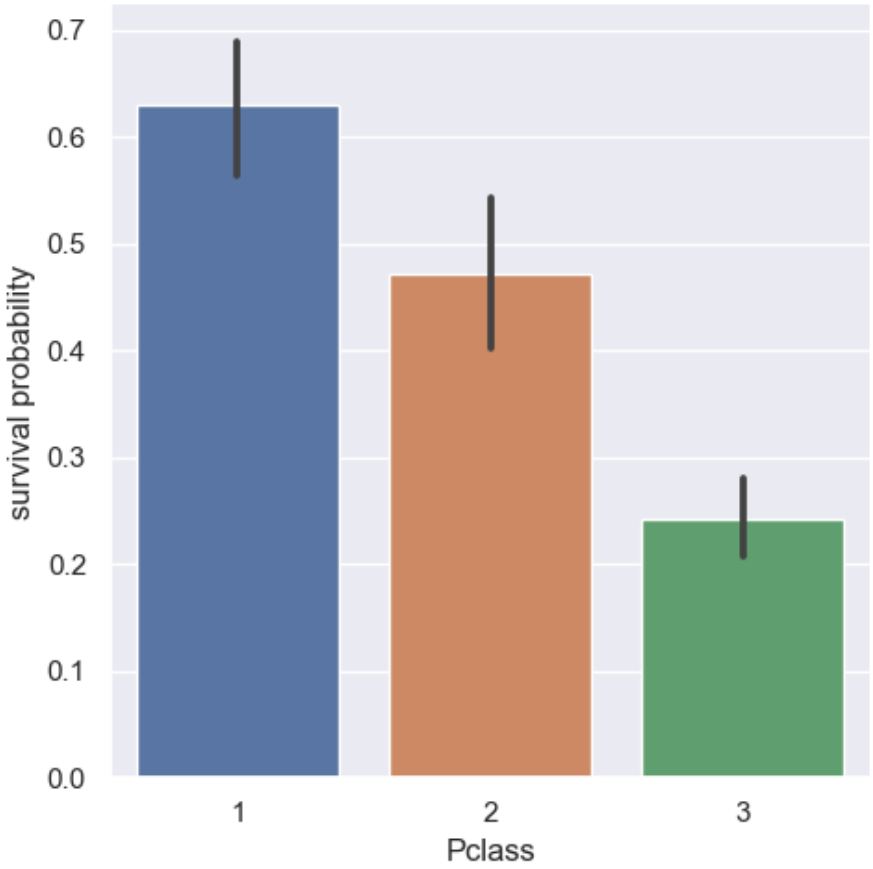
```
data[["Sex","Survived"]].groupby('Sex').mean()
```

Out[28]:

Sex	Survived
female	0.742038
male	0.188908

In [30]:

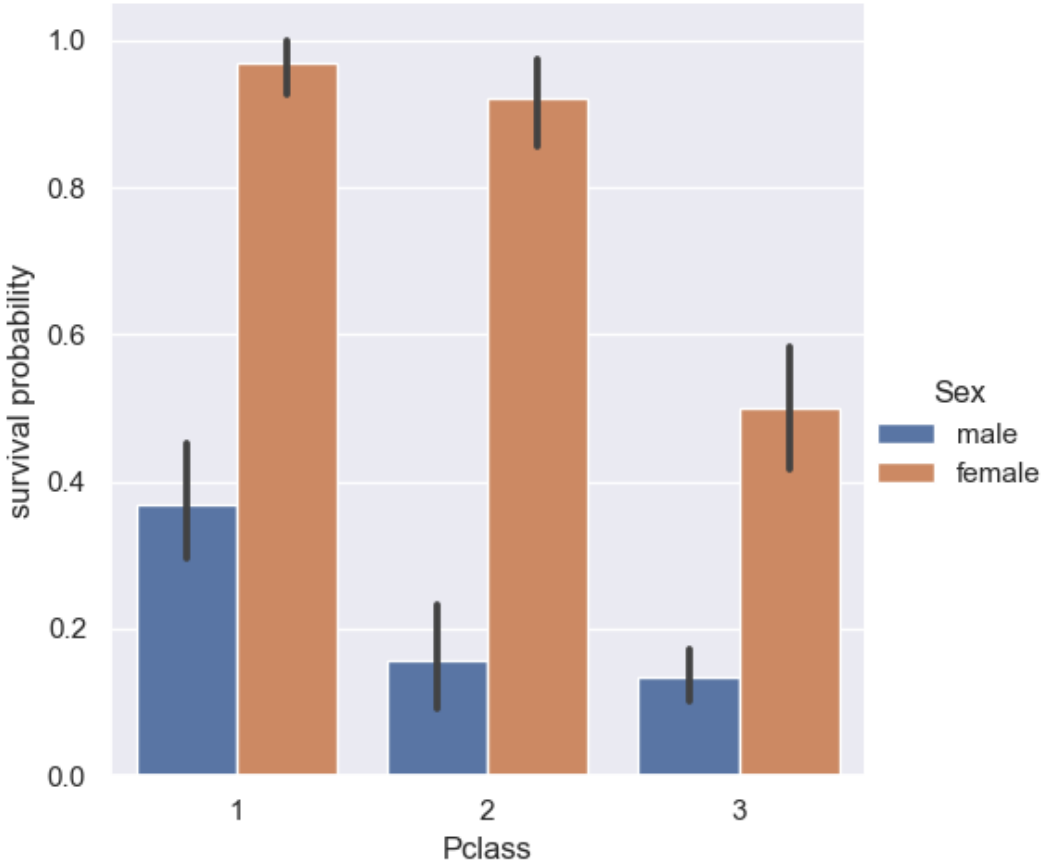
```
pclass = sns.catplot(x = "Pclass", y = "Survived", data = data, kind = "bar")
pclass = pclass.set_ylabels("survival probability")
```



In [35]:

```
g = sns.catplot(x="Pclass", y="Survived", hue="Sex", data=data, kind="bar")
g = g.set_ylabels("survival probability")

import warnings
warnings.filterwarnings("ignore")
```



In [36]:

```
data["Embarked"].isnull().sum()
```

Out[36]:

```
2
```

In [38]:

```
data["Embarked"].value_counts()
```

Out[38]:

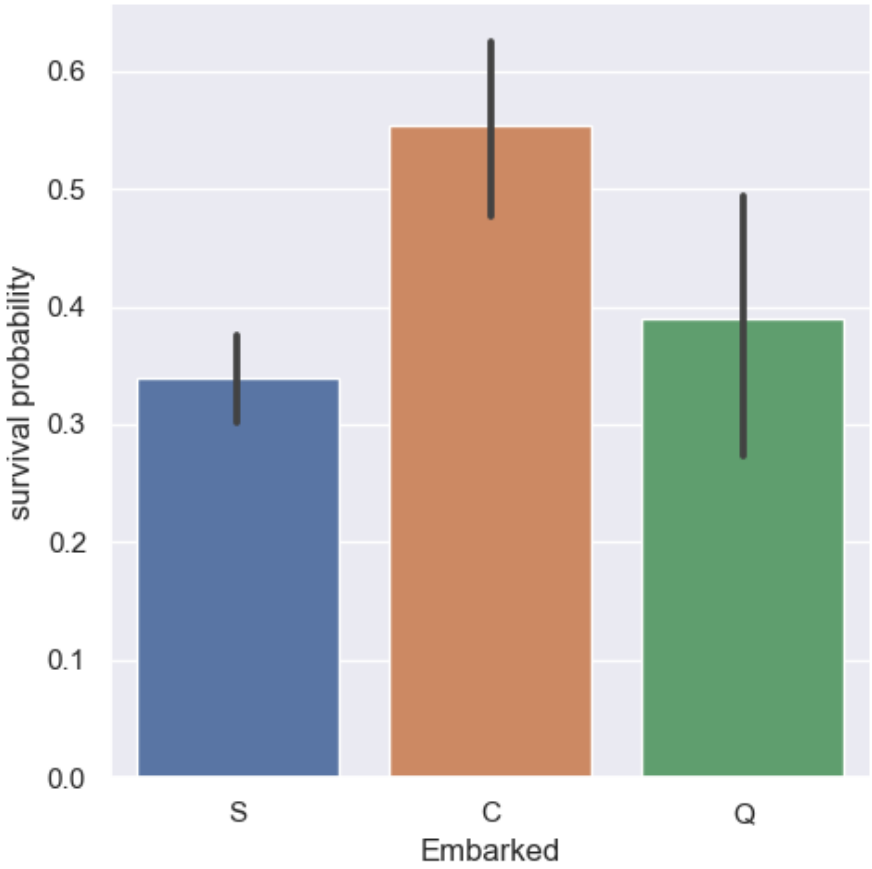
```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

In [39]:

```
data["Embarked"] = data["Embarked"].fillna("S")
```

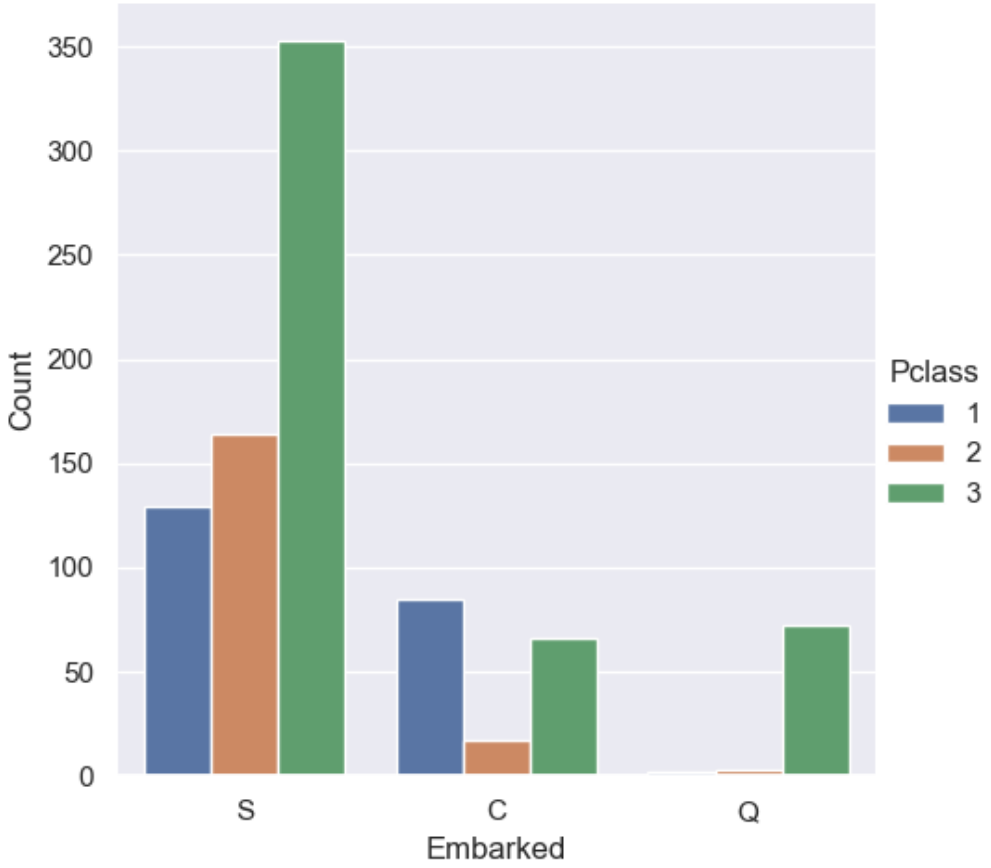
In [40]:

```
g = sns.catplot(x="Embarked", y="Survived", data=data, kind="bar")
g = g.set_ylabels("survival probability")
```



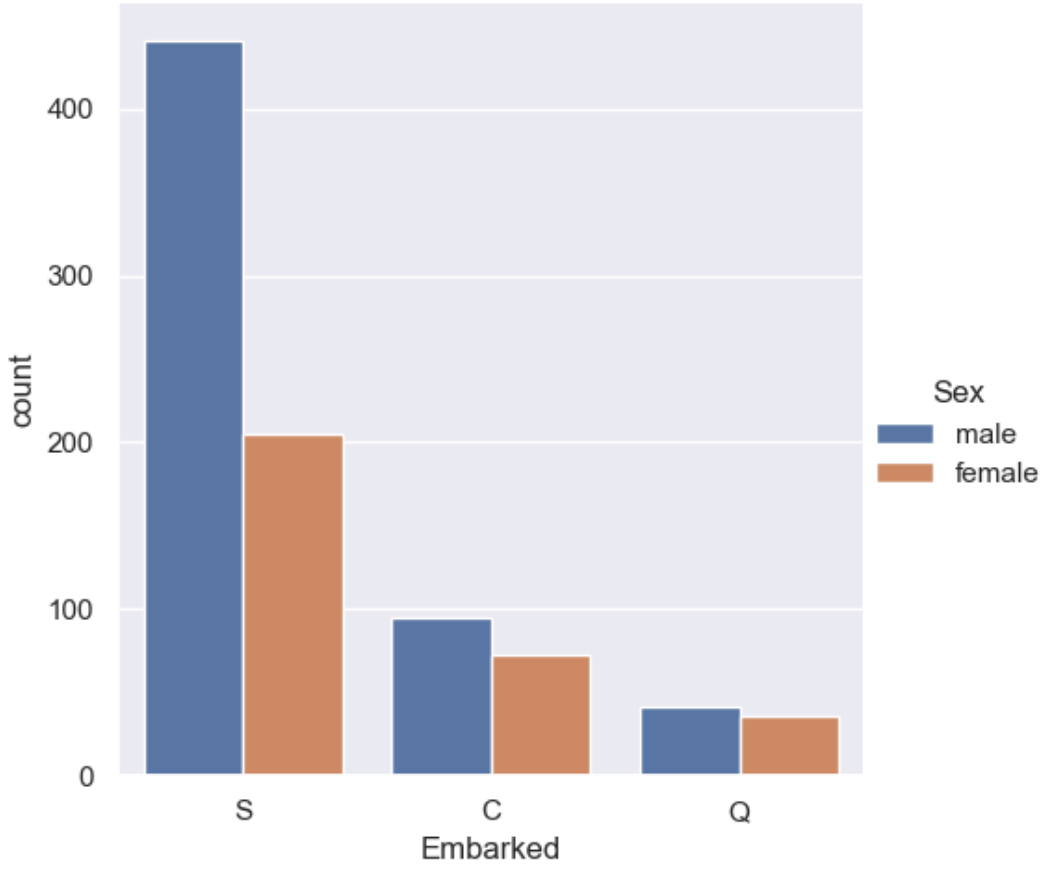
In [66]:

```
g = sns.catplot(x="Embarked",  data=data,hue='Pclass', kind="count")
g.despine(left=True)
g = g.set_ylabels("Count")
```



In [70]:

```
g = sns.catplot(hue="Sex", x="Embarked",  data=data, kind="count")
```



Preparing data

In [71]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     891 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [76]:

```
mean = data["Age"].mean()
std = data["Age"].std()
is_null = data["Age"].isnull().sum()

# compute random numbers between the mean, std and is_null
rand_age = np.random.randint(mean - std, mean + std, size = is_null)

# fill NaN values in Age column with random values generated
age_slice = data["Age"].copy()
age_slice[np.isnan(age_slice)] = rand_age
data["Age"] = age_slice
```

In [77]:

```
data["Age"].isnull().sum()
```

Out[77]:

```
0
```

In [79]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          891 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     891 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [80]:

```
#Fill Embarked with 'S' i.e. the most frequent values
data["Embarked"] = data["Embarked"].fillna("S")
```

In [81]:

```
col_to_drop = ['PassengerId','Cabin', 'Ticket','Name']
data.drop(col_to_drop, axis=1, inplace = True)
```

In [83]:

```
data.head()
```

Out[83]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.0	1	0	7.2500	S
1	1	1	female	38.0	1	0	71.2833	C
2	1	3	female	26.0	0	0	7.9250	S
3	1	1	female	35.0	1	0	53.1000	S
4	0	3	male	35.0	0	0	8.0500	S

In [84]:

```
genders = {"male": 0, "female": 1}
data['Sex'] = data['Sex'].map(genders)
```

In [85]:

```
data.head()
```

Out[85]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	0	22.0	1	0	7.2500	S
1	1	1	1	38.0	1	0	71.2833	C
2	1	3	1	26.0	0	0	7.9250	S
3	1	1	1	35.0	1	0	53.1000	S
4	0	3	0	35.0	0	0	8.0500	S

In [86]:

```
ports = {"S": 0, "C": 1, "Q": 2}

data['Embarked'] = data['Embarked'].map(ports)
```

In [87]:

```
data.head()
```

Out[87]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	0	22.0	1	0	7.2500	0
1	1	1	1	38.0	1	0	71.2833	1
2	1	3	1	26.0	0	0	7.9250	0
3	1	1	1	35.0	1	0	53.1000	0
4	0	3	0	35.0	0	0	8.0500	0

In [88]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 8 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Survived  891 non-null    int64  
 1   Pclass    891 non-null    int64  
 2   Sex       891 non-null    int64  
 3   Age       891 non-null    float64
 4   SibSp     891 non-null    int64  
 5   Parch     891 non-null    int64  
 6   Fare      891 non-null    float64
 7   Embarked  891 non-null    int64  
dtypes: float64(2), int64(6)
memory usage: 55.8 KB
```

Train-Test split

In [89]:

```
x = data.drop(data.columns[[0]], axis = 1)
y = data['Survived']
```

In [90]:

```
x.head()
```

Out[90]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	0	22.0	1	0	7.2500	0
1	1	1	38.0	1	0	71.2833	1
2	3	1	26.0	0	0	7.9250	0
3	1	1	35.0	1	0	53.1000	0
4	3	0	35.0	0	0	8.0500	0

In [91]:

```
y.head()
```

Out[91]:

```
0    0
1    1
2    1
3    1
4    0
Name: Survived, dtype: int64
```

In [92]:

```
# splitting into training and testing data
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size = 0.30, random_state =0)
```

Feature scaling

In [94]:

```
from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
xtrain = sc_x.fit_transform(xtrain)
xtest = sc_x.transform(xtest)
```

Classification

In [95]:

```
logreg = LogisticRegression()
svc_classifier = SVC()
dt_classifier = DecisionTreeClassifier()
knn_classifier = KNeighborsClassifier(5)
rf_classifier = RandomForestClassifier(n_estimators=1000, criterion = 'entropy', random_state = 0
```

In [96]:

```
logreg.fit(xtrain, ytrain)
svc_classifier.fit(xtrain, ytrain)
dt_classifier.fit(xtrain, ytrain)
knn_classifier.fit(xtrain, ytrain)
rf_classifier.fit(xtrain, ytrain)
```

Out[96]:

```
RandomForestClassifier
RandomForestClassifier(criterion='entropy', n_estimators=1000, random_state=0)
```

In [97]:

```
logreg_ypred = logreg.predict(xtest)
svc_classifier_ypred = svc_classifier.predict(xtest)
dt_classifier_ypred = dt_classifier.predict(xtest)
knn_classifier_ypred = knn_classifier.predict(xtest)
rf_classifier_ypred = rf_classifier.predict(xtest)
```

In [98]:

```
# finding accuracy
from sklearn.metrics import accuracy_score

logreg_acc = accuracy_score(ytest, logreg_ypred)
svc_classifier_acc = accuracy_score(ytest, svc_classifier_ypred)
dt_classifier_acc = accuracy_score(ytest, dt_classifier_ypred)
knn_classifier_acc = accuracy_score(ytest, knn_classifier_ypred)
rf_classifier_acc = accuracy_score(ytest, rf_classifier_ypred)
```

In [99]:

```
print ("Logistic Regression : ", round(logreg_acc*100, 2))
print ("Support Vector      : ", round(svc_classifier_acc*100, 2))
print ("Decision Tree       : ", round(dt_classifier_acc*100, 2))
print ("K-NN Classifier     : ", round(knn_classifier_acc*100, 2))
print ("Random Forest       : ", round(rf_classifier_acc*100, 2))
```

```
Logistic Regression :  80.22
Support Vector      :  81.34
Decision Tree       :  80.22
K-NN Classifier     :  80.97
Random Forest       :  84.7
```

In [ ]:

```
P
```