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.

```
VARIABLE       DESCRIPTION            KEY

survival       Survival               0 = No, 1 = Yes
pclass         Ticket class           1 = 1st, 2 = 2nd, 3 = 3rd
sex            Sex
Age            Age in years
sibsp          # of siblings / spouses aboard the Titanic
parch          # of parents / children aboard the Titanic
ticket         Ticket number
fare           Passenger fare
cabin          Cabin number
embarked       Port of Embarkation    C = Cherbourg, Q = Queenstown, S =
Southampton



Variable Notes

pclass: A proxy for socio-economic status (SES)
1st    = Upper
2nd    = Middle
3rd    = Lower

age: Age is fractional if less than 1. If the age is estimated, is it in the
form of xx.5

sibsp: The dataset defines family relations in this way...
Sibling = brother, sister, stepbrother, stepsister
Spouse = husband, wife (mistresses and fiancés were ignored)

parch: The dataset defines family relations in this way...
Parent = mother, father
Child = daughter, son, stepdaughter, stepson
Some children travelled only with a nanny, therefore parch=0 for them.
```

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

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

Loading Dataset

In [2]:
```
data = pd.read_csv(r'C:\Users\vamsi\Desktop\M.Tech\ML\19 Projects\titanic data.cs
```

In [3]:
```
data.head(10)
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabi
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	Na
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C8
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	Na
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C12
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	Na
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	Na
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E4
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	Na
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	Na
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	Na

Types of Features :

- **Categorical** - Sex, and Embarked.
- *_Continuous_ * - Age, Fare
- **Discrete -** SibSp, Parch.

- **Alphanumeric -** Cabin

In [4]:
```
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 [5]:
```
data.isnull().sum()
```

Out[5]:
```
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
```

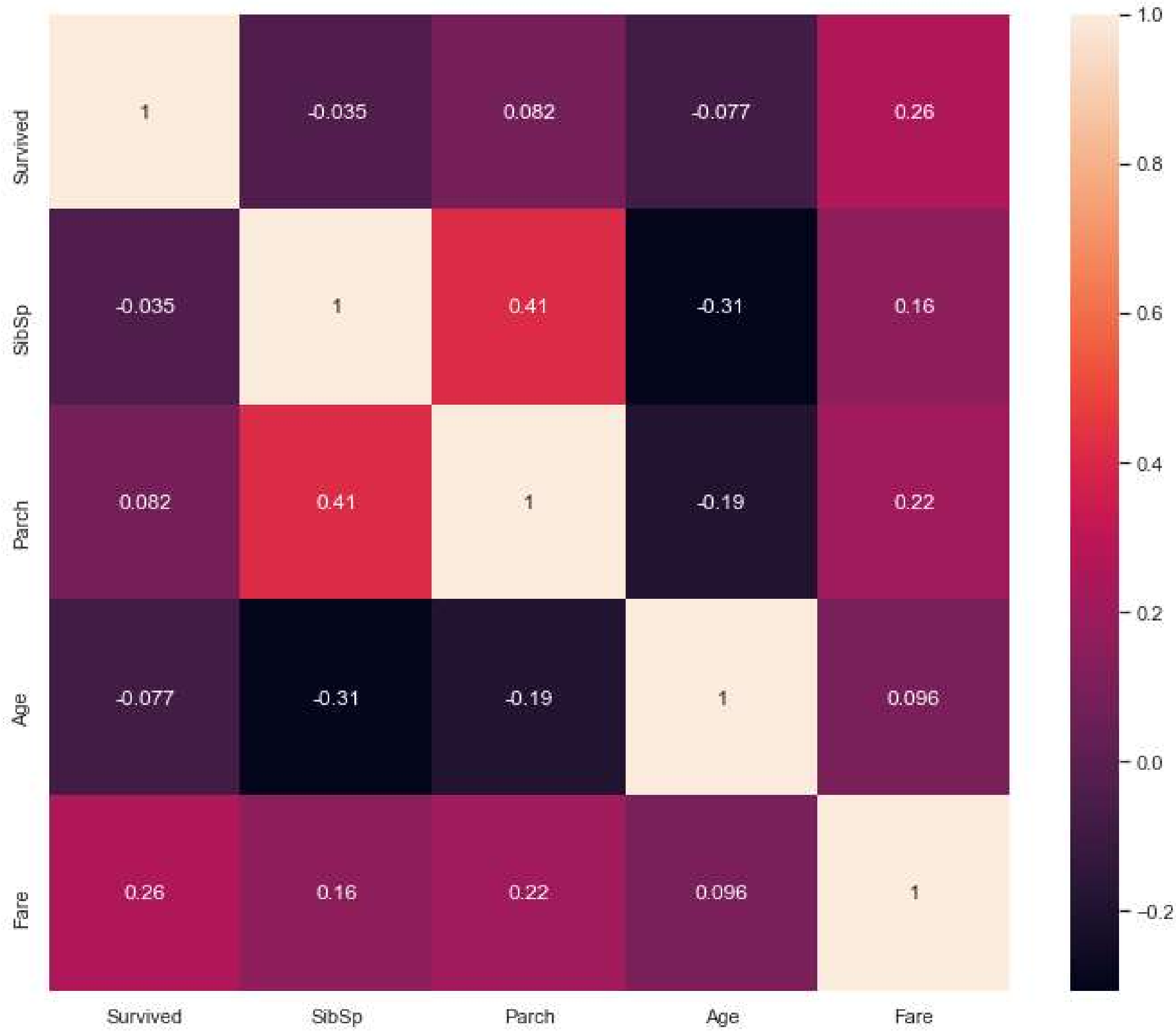
In [6]:
```
data.describe()
```

Out[6]:

	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 [7]:
```
plt.figure(figsize=(12, 10))
heatmap = sns.heatmap(data[["Survived","SibSp","Parch","Age","Fare"]].corr(), anr
```



*Conclusion : *

Only Fare feature seems to have a significative correlation with the survival probability.

It doesn't mean that the other features are not usefull. Subpopulations in these features can be correlated with the survival. To determine this, we need to explore in detail these features

sibsp - Number of siblings / spouses aboard the Titanic

In [8]:
```
data['SibSp'].nunique()
```

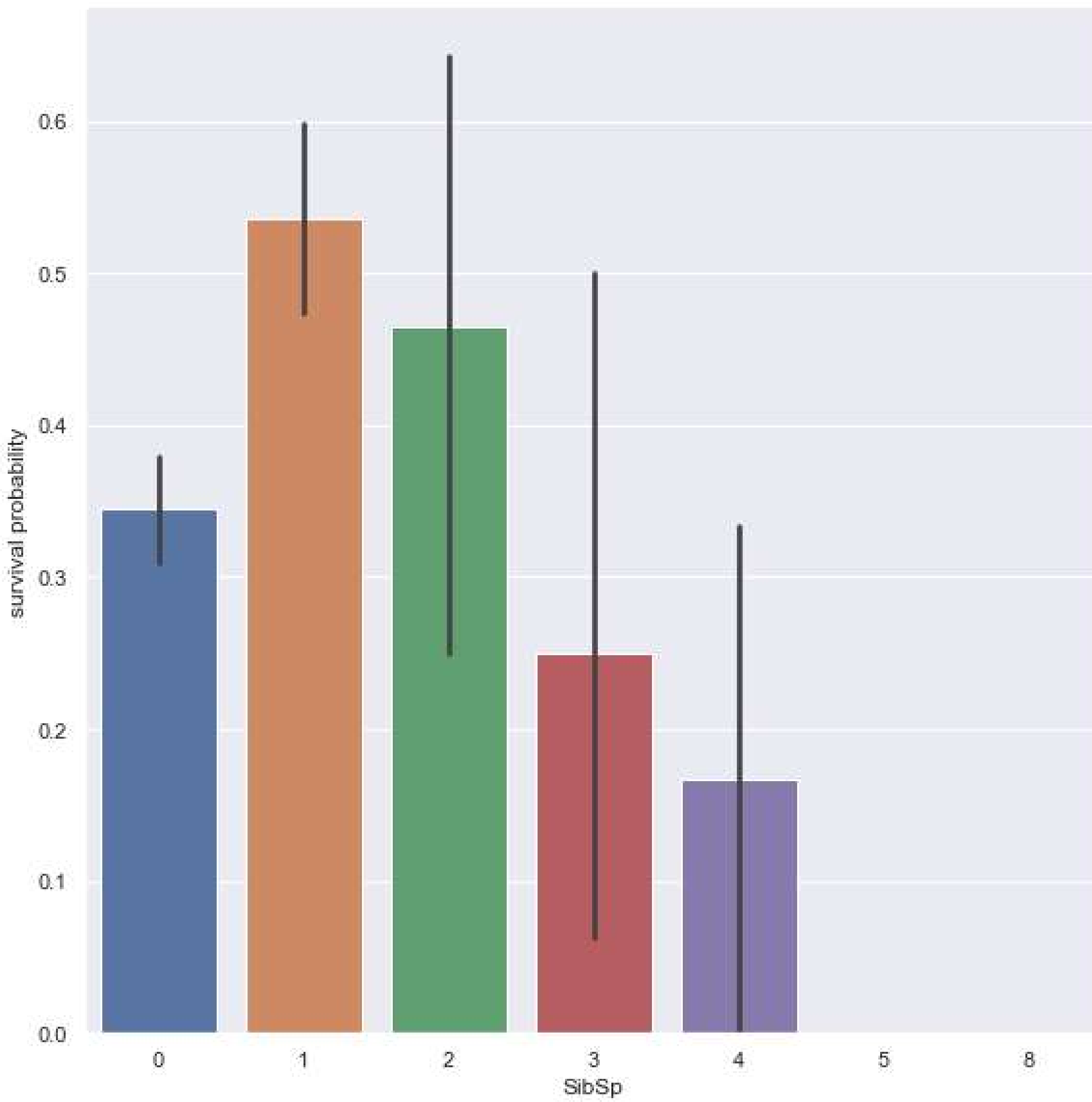
Out[8]:
```
7
```

In [9]:
```
data['SibSp'].unique()
```

Out[9]:
```
array([1, 0, 3, 4, 2, 5, 8], dtype=int64)
```

In [10]:
```
bargraph_sibsp = sns.factorplot(x = "SibSp", y = "Survived", data = data, kind =
bargraph_sibsp = bargraph_sibsp.set_ylabels("survival probability")
```

```
C:\Users\vamsi\anaconda3\lib\site-packages\seaborn\categorical.py:3704: UserWar
ning: The `factorplot` function has been renamed to `catplot`. The original nam
e will be removed in a future release. Please update your code. Note that the d
efault `kind` in `factorplot` (`'point'`) has changed `'strip'` in `catplot`.
  warnings.warn(msg)
C:\Users\vamsi\anaconda3\lib\site-packages\seaborn\categorical.py:3710: UserWar
ning: The `size` parameter has been renamed to `height`; please update your cod
e.
  warnings.warn(msg, UserWarning)
```

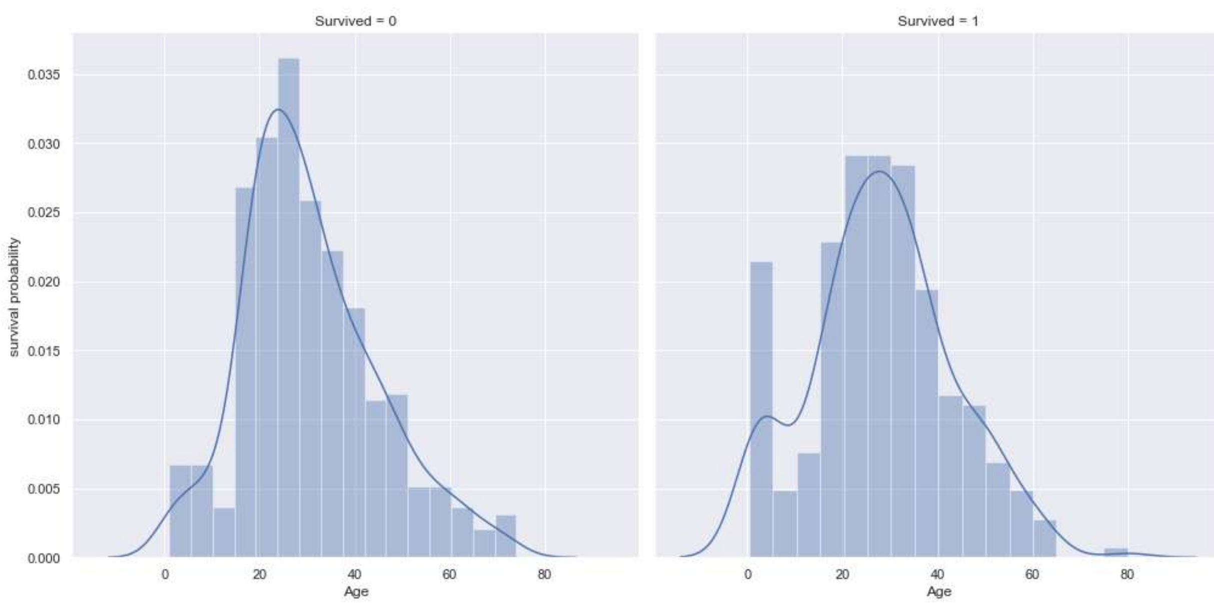


It seems that passengers having a lot of siblings/spouses have less chance to survive. Single passengers (0 SibSP) or with two other persons (SibSP 1 or 2) have more chance to survive.

Age

In [11]:
```
age_visual = sns.FacetGrid(data, col = 'Survived', size=7)
age_visual = age_visual.map(sns.distplot, "Age")
age_visual = age_visual.set_ylabels("survival probability")
```

```
C:\Users\vamsi\anaconda3\lib\site-packages\seaborn\axisgrid.py:316: UserWarnin
g: The `size` parameter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
C:\Users\vamsi\anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
C:\Users\vamsi\anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```



Age distribution seems to be a tailed distribution, maybe a gaussian distribution.

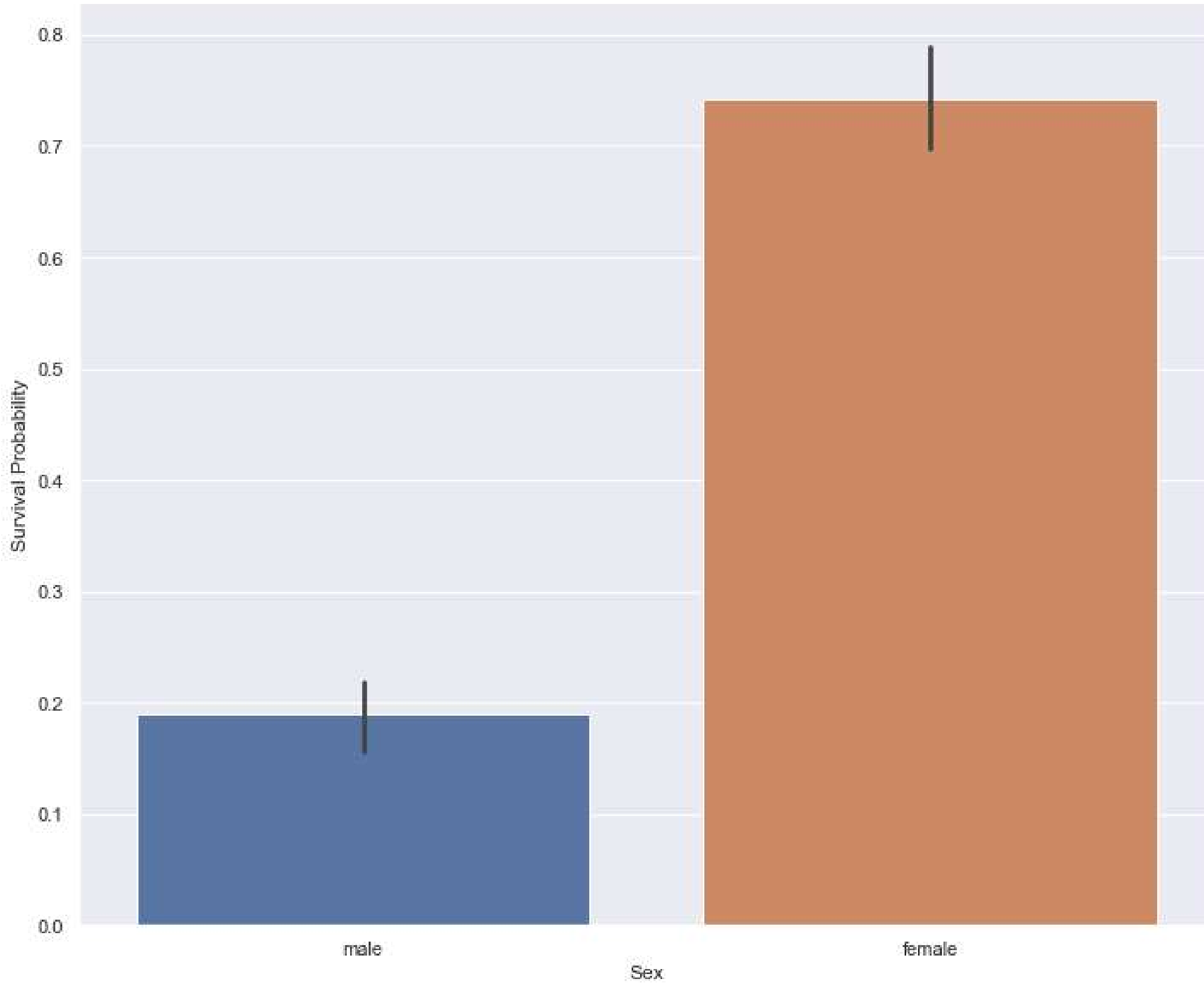
We notice that age distributions are not the same in the survived and not survived subpopulations. Indeed, there is a peak corresponding to young passengers, that have survived. We also see that passengers between 60-80 have less survived.

So, even if "Age" is not correlated with "Survived", we can see that there is age categories of passengers that of have more or less chance to survive.

It seems that very young passengers have more chance to survive.

Sex

In [12]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 10))
age_plot = sns.barplot(x = "Sex",y = "Survived", data = data)
age_plot = age_plot.set_ylabel("Survival Probability")
```



In [13]:
```
data[["Sex","Survived"]].groupby('Sex').mean()
```

Out[13]:

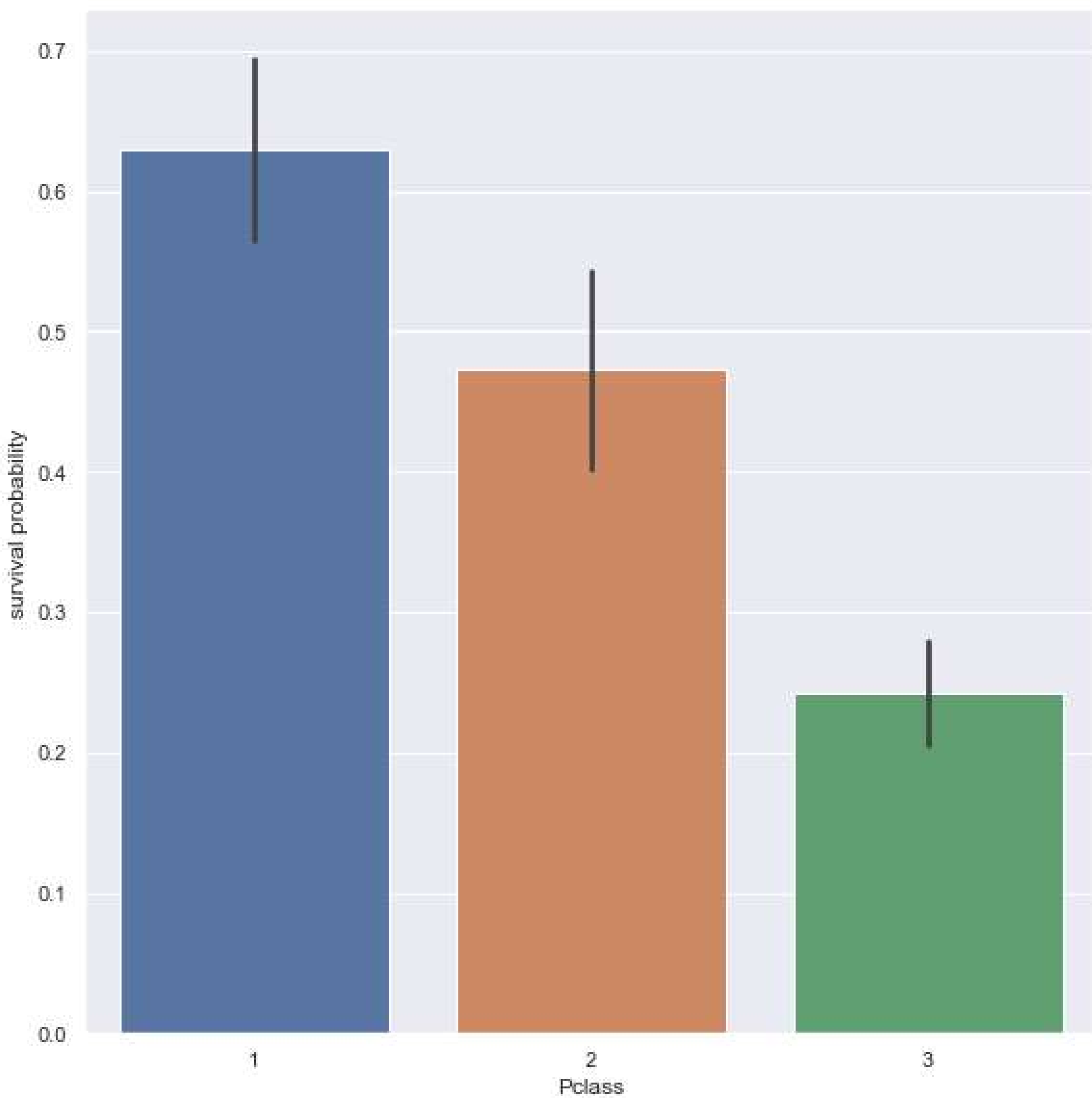
	Survived
Sex	
female	0.742038
male	0.188908

It is clearly obvious that Male have less chance to survive than Female. So Sex, might play an important role in the prediction of the survival. For those who have seen the Titanic movie (1997), I am sure, we all remember this sentence during the evacuation **- Women and children first**

PClass

In [14]:
```
pclass = sns.factorplot(x = "Pclass", y = "Survived", data = data, kind = "bar",
pclass = pclass.set_ylabels("survival probability")
```

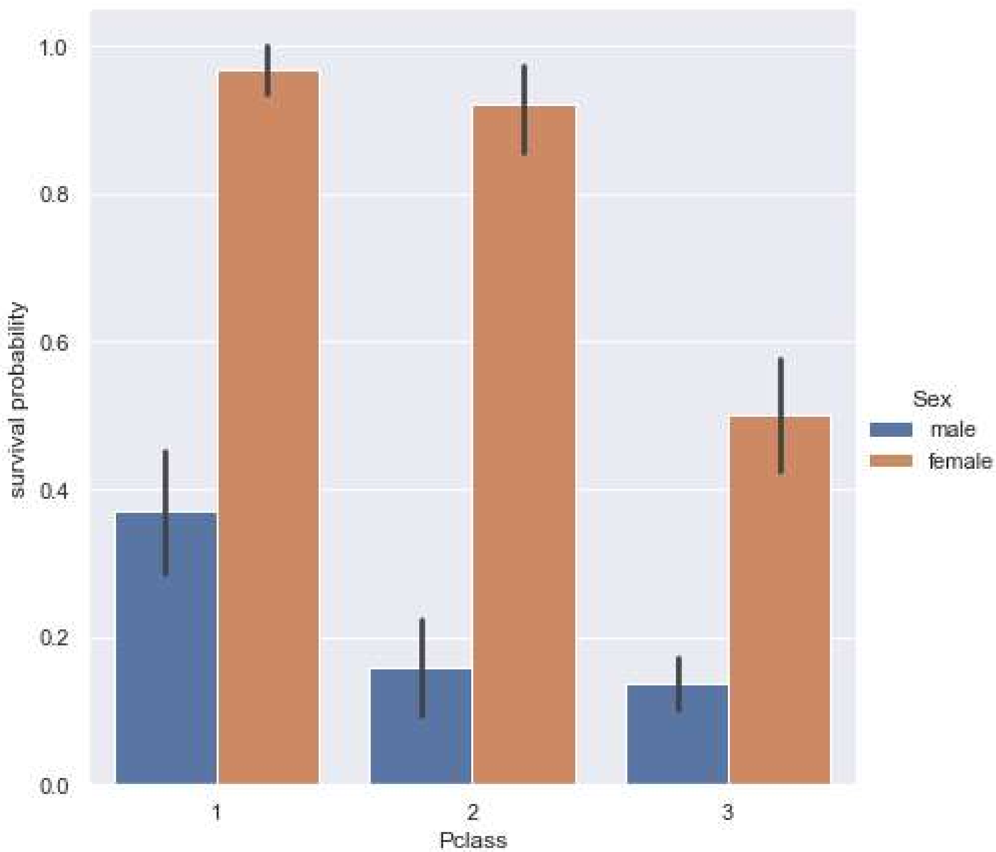
```
C:\Users\vamsi\anaconda3\lib\site-packages\seaborn\categorical.py:3704: UserWar
ning: The `factorplot` function has been renamed to `catplot`. The original nam
e will be removed in a future release. Please update your code. Note that the d
efault `kind` in `factorplot` (`'point'`) has changed `'strip'` in `catplot`.
  warnings.warn(msg)
C:\Users\vamsi\anaconda3\lib\site-packages\seaborn\categorical.py:3710: UserWar
ning: The `size` parameter has been renamed to `height`; please update your cod
e.
  warnings.warn(msg, UserWarning)
```



Pclass vs Survived by Sex

In [15]:
```
g = sns.factorplot(x="Pclass", y="Survived", hue="Sex", data=data, size=6, kind='
g = g.set_ylabels("survival probability")
```

```
C:\Users\vamsi\anaconda3\lib\site-packages\seaborn\categorical.py:3704: UserWar
ning: The `factorplot` function has been renamed to `catplot`. The original nam
e will be removed in a future release. Please update your code. Note that the d
efault `kind` in `factorplot` (`'point'`) has changed `'strip'` in `catplot`.
  warnings.warn(msg)
C:\Users\vamsi\anaconda3\lib\site-packages\seaborn\categorical.py:3710: UserWar
ning: The `size` parameter has been renamed to `height`; please update your cod
e.
  warnings.warn(msg, UserWarning)
```



Embarked

In [16]:
```
data["Embarked"].isnull().sum()
```

Out[16]: 2

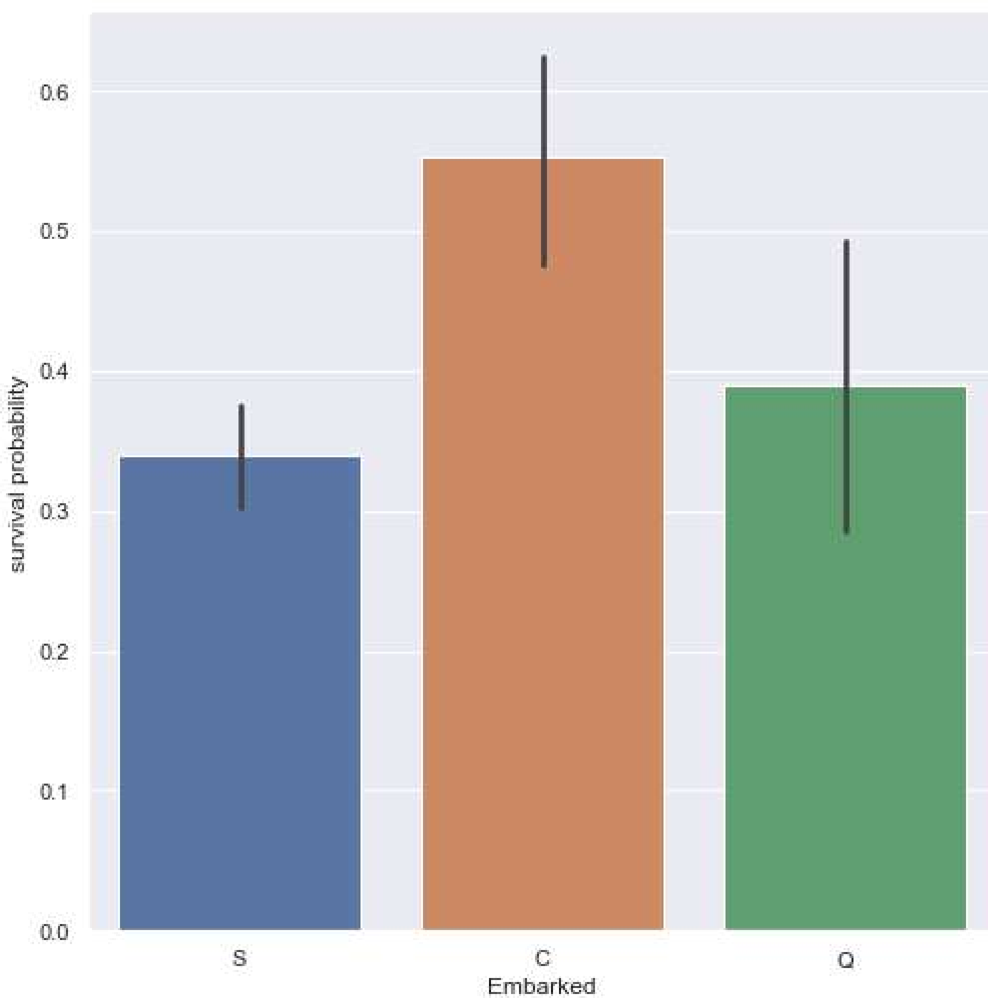
In [17]:
```
data["Embarked"].value_counts()
```

Out[17]:
```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

In [18]:
```
#Fill Embarked with 'S' i.e. the most frequent values
data["Embarked"] = data["Embarked"].fillna("S")
```

In [19]:
```
g = sns.factorplot(x="Embarked", y="Survived", data=data, size=7, kind="bar")
g = g.set_ylabels("survival probability")
```

```
C:\Users\vamsi\anaconda3\lib\site-packages\seaborn\categorical.py:3704: UserWar
ning: The `factorplot` function has been renamed to `catplot`. The original nam
e will be removed in a future release. Please update your code. Note that the d
efault `kind` in `factorplot` (`'point'`) has changed `'strip'` in `catplot`.
  warnings.warn(msg)
C:\Users\vamsi\anaconda3\lib\site-packages\seaborn\categorical.py:3710: UserWar
ning: The `size` parameter has been renamed to `height`; please update your cod
e.
  warnings.warn(msg, UserWarning)
```

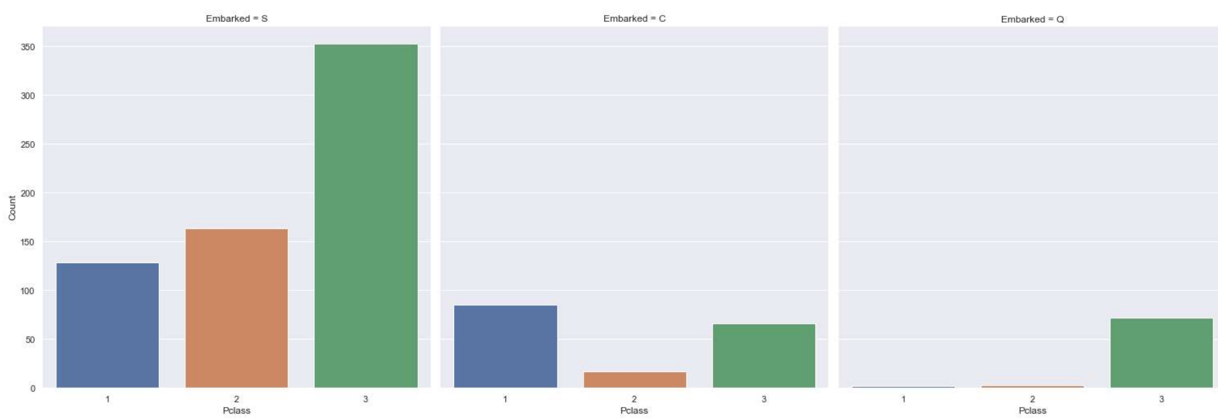


Passenger coming from Cherbourg (C) have more chance to survive.

Let's find the reason

In [20]:
```
# Explore Pclass vs Embarked
g = sns.factorplot("Pclass", col="Embarked",  data=data, size=7, kind="count")
g.despine(left=True)
g = g.set_ylabels("Count")
```

```
C:\Users\vamsi\anaconda3\lib\site-packages\seaborn\categorical.py:3704: UserWar
ning: The `factorplot` function has been renamed to `catplot`. The original nam
e will be removed in a future release. Please update your code. Note that the d
efault `kind` in `factorplot` (`'point'`) has changed `'strip'` in `catplot`.
  warnings.warn(msg)
C:\Users\vamsi\anaconda3\lib\site-packages\seaborn\categorical.py:3710: UserWar
ning: The `size` parameter has been renamed to `height`; please update your cod
e.
  warnings.warn(msg, UserWarning)
C:\Users\vamsi\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variable as a keyword arg: x. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments witho
ut an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



Cherbourg passengers are mostly in first class which have the highest survival rate.
Southampton (S) and Queenstown (Q) passangers are mostly in third class.

Preparing data

In [21]:
```
data.head()
```

Out[21]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabi
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	Na
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C8
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	Na
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C12
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	Na

In [24]:
```
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 [25]:
```
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 [26]:
```
data["Age"].isnull().sum()
```

Out[26]:
```
0
```

In [27]:
```
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 [28]:
```
data["Embarked"].isnull().sum()
```

Out[28]:
```
0
```

In [29]:
```
#Fill Embarked with 'S' i.e. the most frequent values
data["Embarked"] = data["Embarked"].fillna("S")
```

In [30]:
```
col_to_drop = ['PassengerId','Cabin', 'Ticket','Name']
data.drop(col_to_drop, axis=1, inplace = True)
```

In [31]:
```
data.head()
```

Out[31]:

	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 [32]:
```
genders = {"male": 0, "female": 1}
data['Sex'] = data['Sex'].map(genders)
```

In [33]:
```
data.head()
```

Out[33]:

	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 [34]:
```
ports = {"S": 0, "C": 1, "Q": 2}

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

In [35]:
```
data.head()
```

Out[35]:

	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 [36]:
```
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
```

Splitting data

In [37]:
```
# input and output data

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

In [38]:
```
x.head()
```

Out[38]:

	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 [39]:
```
y.head()
```

Out[39]:
```
0    0
1    1
2    1
3    1
4    0
Name: Survived, dtype: int64
```

In [40]:
```
# 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_st
```

Feature Scaling

In [41]:
```
from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
xtrain = sc_x.fit_transform(xtrain)
xtest = sc_x.transform(xtest)
```

Classification

In [42]:
```
logreg = LogisticRegression()
svc_classifier = SVC()
dt_classifier = DecisionTreeClassifier()
knn_classifier = KNeighborsClassifier(5)
rf_classifier = RandomForestClassifier(n_estimators=1000, criterion = 'entropy',
```

In [44]:
```
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[44]:
```
RandomForestClassifier(criterion='entropy', n_estimators=1000, random_state=0)
```

In [45]:
```
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 [46]:
```
# 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 [47]:
```
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       :  79.48
K-NN Classifier     :  79.48
Random Forest       :  84.7
```

In [ ]:

In [ ]: