Titanic Passenger Survival Prediction

In [42]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

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

taken data from Kaggle TiTaNic Challenge

In [44]:
```
train = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")
```

In [45]:
```
train.describe()
```

Out[45]:

	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

In [46]:
```
train.columns
```

Out[46]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

In [47]:
```
train.head(10)
```

Out[47]:

	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	Nal
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	Nal
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	Nal
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	Nal
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	Nal
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	Nal
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	Nal

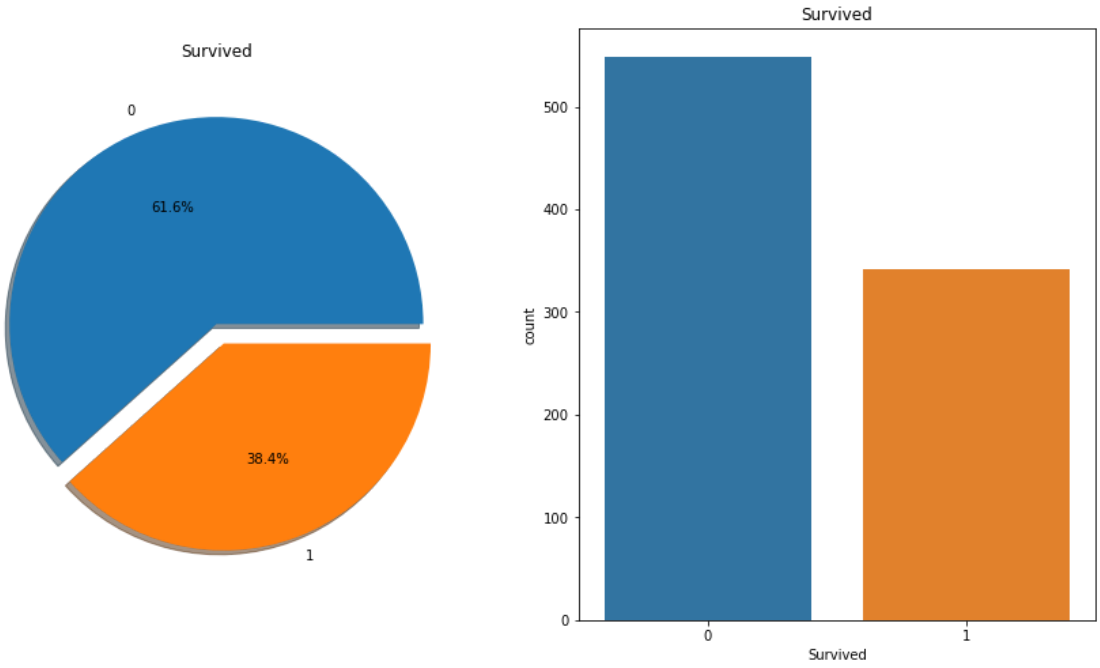
In [48]:
```
test.head(10)
```

Out[48]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S
5	897	3	Svensson, Mr. Johan Cervin	male	14.0	0	0	7538	9.2250	NaN	S
6	898	3	Connolly, Miss. Kate	female	30.0	0	0	330972	7.6292	NaN	Q
7	899	2	Caldwell, Mr. Albert Francis	male	26.0	1	1	248738	29.0000	NaN	S
8	900	3	Abrahim, Mrs. Joseph (Sophie Halaut Easu)	female	18.0	0	0	2657	7.2292	NaN	C
9	901	3	Davies, Mr. John Samuel	male	21.0	2	0	A/4 48871	24.1500	NaN	S

Data analysis by visualization

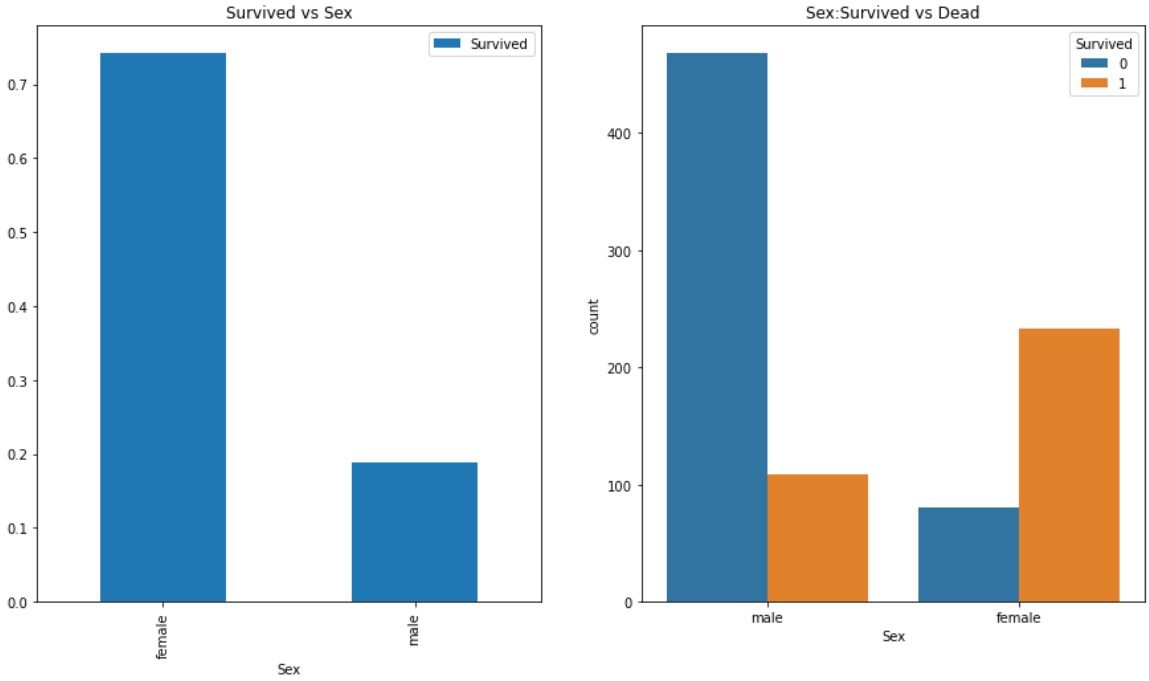
In [49]:
```
f,ax=plt.subplots(1,2,figsize=(15,8))
train['Survived'].value_counts().plot.pie(explode=[0,0.1],autopct='%1.1f%%',ax=a
ax[0].set_title('Survived')
ax[0].set_ylabel('')
sns.countplot('Survived',data=train,ax=ax[1])
ax[1].set_title('Survived')
plt.show()
```



Analysing the columns based on they survived or not

Sex -> Category wise survivability

In [50]:
```
f,ax=plt.subplots(1,2,figsize=(15,8))
train[['Sex','Survived']].groupby(['Sex']).mean().plot.bar(ax=ax[0])
ax[0].set_title('Survived vs Sex')
sns.countplot('Sex',hue='Survived',data=train,ax=ax[1])
ax[1].set_title('Sex:Survived vs Dead')
plt.show()
```



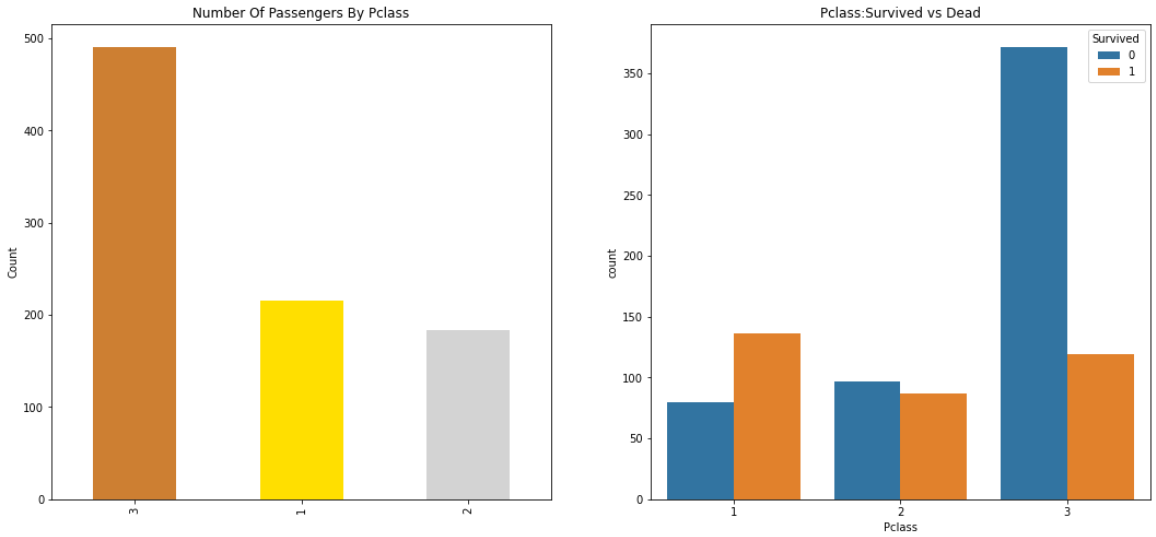
Passenger class ->

In [51]:
```
pd.crosstab(train.Pclass,train.Survived,margins=True).style.background_gradient(
```

Out[51]:

Survived	0	1	All
Pclass			
1	80	136	216
2	97	87	184
3	372	119	491
All	549	342	891

In [52]:
```
f,ax=plt.subplots(1,2,figsize=(18,8))
train['Pclass'].value_counts().plot.bar(color=['#CD7F32','#FFDF00','#D3D3D3'],ax
ax[0].set_title('Number Of Passengers By Pclass')
ax[0].set_ylabel('Count')
sns.countplot('Pclass',hue='Survived',data=train,ax=ax[1])
ax[1].set_title('Pclass:Survived vs Dead')
plt.show()
```



In [53]:
```
pd.crosstab([train.Sex,train.Survived],train.Pclass,margins=True).style.backgrou
```

Out[53]:

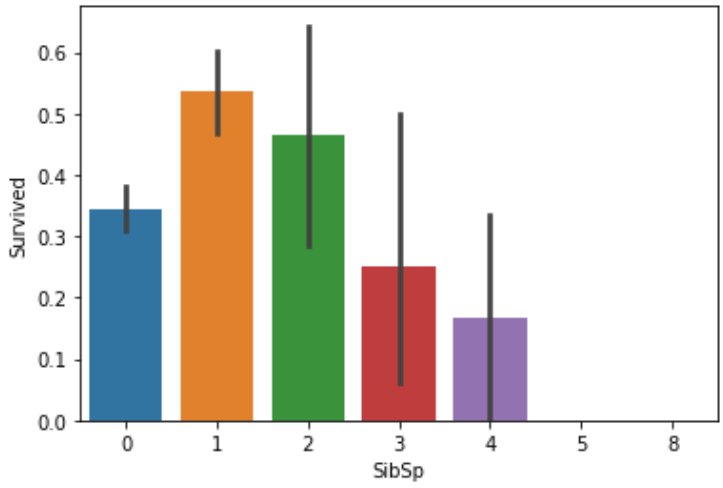
Sex	Survived	Pclass 1	2	3	All
female	0	3	6	72	81
	1	91	70	72	233
male	0	77	91	300	468
	1	45	17	47	109
All		216	184	491	891

from the above graph , we can say that the percentage of survival of first class is higher.

SibSp Feature

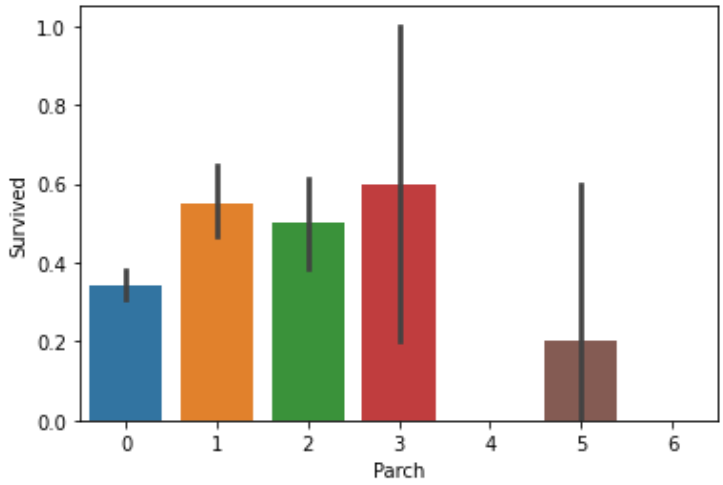
In [54]:
```
sns.barplot(x="SibSp", y="Survived", data=train)

print("Percentage of SibSp = 0 who survived:", train["Survived"][train["SibSp"]

print("Percentage of SibSp = 1 who survived:", train["Survived"][train["SibSp"]

print("Percentage of SibSp = 2 who survived:", train["Survived"][train["SibSp"]
```

```
Percentage of SibSp = 0 who survived: 34.53947368421053
Percentage of SibSp = 1 who survived: 53.588516746411486
Percentage of SibSp = 2 who survived: 46.42857142857143
```



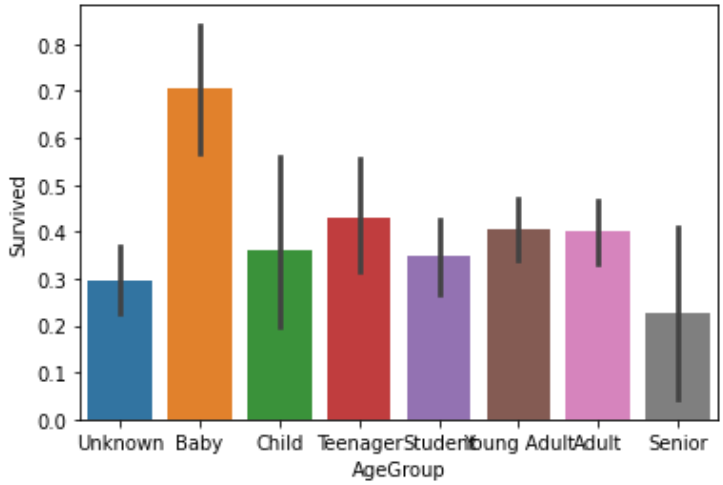
Parch Feature

In [55]:
```
sns.barplot(x="Parch", y="Survived", data=train)
plt.show()
```



Age Feature

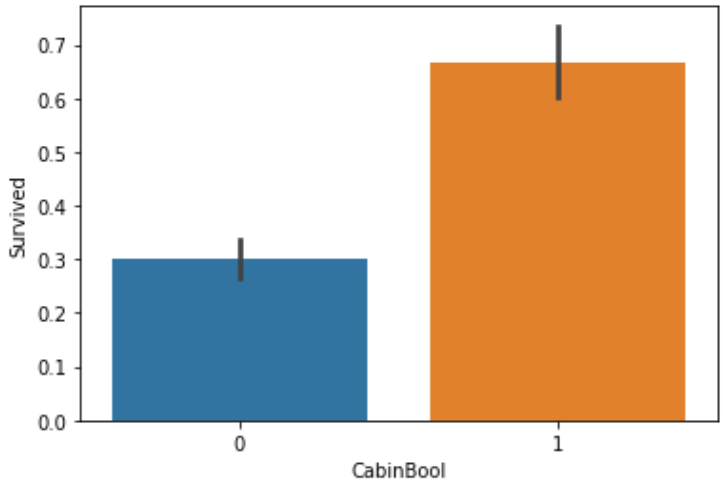
In [56]:
```
train["Age"] = train["Age"].fillna(-0.5)
test["Age"] = test["Age"].fillna(-0.5)
bins = [-1, 0, 5, 12, 18, 24, 35, 60, np.inf]
labels = ['Unknown', 'Baby', 'Child', 'Teenager', 'Student', 'Young Adult', 'Adu
train['AgeGroup'] = pd.cut(train["Age"], bins, labels = labels)
test['AgeGroup'] = pd.cut(test["Age"], bins, labels = labels)

sns.barplot(x="AgeGroup", y="Survived", data=train)
plt.show()
```



Cabin Feature

In [57]:
```
train["CabinBool"] = (train["Cabin"].notnull().astype('int'))
test["CabinBool"] = (test["Cabin"].notnull().astype('int'))


print("Percentage of CabinBool = 1 who survived:", train["Survived"][train["Cabi

print("Percentage of CabinBool = 0 who survived:", train["Survived"][train["Cabi

sns.barplot(x="CabinBool", y="Survived", data=train)
plt.show()
```

```
Percentage of CabinBool = 1 who survived: 66.66666666666666
Percentage of CabinBool = 0 who survived: 29.985443959243085
```



Cleaning Data

In [58]:
```
train = train.drop(['Cabin'], axis = 1)
test = test.drop(['Cabin'], axis = 1)
```

In [59]:
```
train = train.drop(['Ticket'], axis = 1)
test = test.drop(['Ticket'], axis = 1)
```

In [60]:
```
print("Number of people embarking in Southampton (S):")
southampton = train[train["Embarked"] == "S"].shape[0]
print(southampton)

print("Number of people embarking in Cherbourg (C):")
cherbourg = train[train["Embarked"] == "C"].shape[0]
print(cherbourg)

print("Number of people embarking in Queenstown (Q):")
queenstown = train[train["Embarked"] == "Q"].shape[0]
print(queenstown)
```

```
Number of people embarking in Southampton (S):
644
Number of people embarking in Cherbourg (C):
168
Number of people embarking in Queenstown (Q):
77
```

In [61]:
```
train = train.fillna({"Embarked": "S"})
```

In [62]:
```
combine = [train, test]

for dataset in combine:
    dataset['Title'] = dataset.Name.str.extract(' ([A-Za-z]+)\.', expand=False)

pd.crosstab(train['Title'], train['Sex'])
```

Out[62]:

Sex	female	male
Title		
Capt	0	1
Col	0	2
Countess	1	0
Don	0	1
Dr	1	6
Jonkheer	0	1
Lady	1	0
Major	0	2
Master	0	40
Miss	182	0
Mlle	2	0
Mme	1	0
Mr	0	517
Mrs	125	0
Ms	1	0
Rev	0	6
Sir	0	1

In [63]:
```
for dataset in combine:
    dataset['Title'] = dataset['Title'].replace(['Lady', 'Capt', 'Col',
    'Don', 'Dr', 'Major', 'Rev', 'Jonkheer', 'Dona'], 'Rare')

    dataset['Title'] = dataset['Title'].replace(['Countess', 'Lady', 'Sir'], 'Ro
    dataset['Title'] = dataset['Title'].replace('Mlle', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Ms', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Mme', 'Mrs')

train[['Title', 'Survived']].groupby(['Title'], as_index=False).mean()
```

Out[63]:

	Title	Survived
0	Master	0.575000
1	Miss	0.702703
2	Mr	0.156673
3	Mrs	0.793651
4	Rare	0.285714
5	Royal	1.000000

In [64]:
```
title_mapping = {"Mr": 1, "Miss": 2, "Mrs": 3, "Master": 4, "Royal": 5, "Rare":
for dataset in combine:
    dataset['Title'] = dataset['Title'].map(title_mapping)
    dataset['Title'] = dataset['Title'].fillna(0)

train.head()
```

Out[64]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Fare	Embarked	AgeG
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	7.2500	S	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	71.2833	C	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	7.9250	S	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	53.1000	S	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	8.0500	S	

fill missing age with mode age group for each title

In [65]:
```
mr_age = train[train["Title"] == 1]["AgeGroup"].mode() #Young Adult
miss_age = train[train["Title"] == 2]["AgeGroup"].mode() #Student
mrs_age = train[train["Title"] == 3]["AgeGroup"].mode() #Adult
master_age = train[train["Title"] == 4]["AgeGroup"].mode() #Baby
royal_age = train[train["Title"] == 5]["AgeGroup"].mode() #Adult
rare_age = train[train["Title"] == 6]["AgeGroup"].mode() #Adult

age_title_mapping = {1: "Young Adult", 2: "Student", 3: "Adult", 4: "Baby", 5: "

for x in range(len(train["AgeGroup"])):
    if train["AgeGroup"][x] == "Unknown":
        train["AgeGroup"][x] = age_title_mapping[train["Title"][x]]

for x in range(len(test["AgeGroup"])):
    if test["AgeGroup"][x] == "Unknown":
        test["AgeGroup"][x] = age_title_mapping[test["Title"][x]]
```

In [66]:
```
age_mapping = {'Baby': 1, 'Child': 2, 'Teenager': 3, 'Student': 4, 'Young Adult'
train['AgeGroup'] = train['AgeGroup'].map(age_mapping)
test['AgeGroup'] = test['AgeGroup'].map(age_mapping)

train.head()

#dropping the Age feature, just for now
train = train.drop(['Age'], axis = 1)
test = test.drop(['Age'], axis = 1)
```

In [67]:
```
train = train.drop(['Name'], axis = 1)
test = test.drop(['Name'], axis = 1)
```

In [68]:
```
sex_mapping = {"male": 0, "female": 1}
train['Sex'] = train['Sex'].map(sex_mapping)
test['Sex'] = test['Sex'].map(sex_mapping)

train.head()
```

Out[68]:

	PassengerId	Survived	Pclass	Sex	SibSp	Parch	Fare	Embarked	AgeGroup	CabinBool
0	1	0	3	0	1	0	7.2500	S	4.0	0
1	2	1	1	1	1	0	71.2833	C	6.0	1
2	3	1	3	1	0	0	7.9250	S	5.0	0
3	4	1	1	1	1	0	53.1000	S	5.0	1
4	5	0	3	0	0	0	8.0500	S	5.0	0

In [69]:
```
embarked_mapping = {"S": 1, "C": 2, "Q": 3}
train['Embarked'] = train['Embarked'].map(embarked_mapping)
test['Embarked'] = test['Embarked'].map(embarked_mapping)

train.head()
```

Out[69]:

	PassengerId	Survived	Pclass	Sex	SibSp	Parch	Fare	Embarked	AgeGroup	CabinBool
0	1	0	3	0	1	0	7.2500	1	4.0	0
1	2	1	1	1	1	0	71.2833	2	6.0	1
2	3	1	3	1	0	0	7.9250	1	5.0	0
3	4	1	1	1	1	0	53.1000	1	5.0	1
4	5	0	3	0	0	0	8.0500	1	5.0	0

In [70]:
```
for x in range(len(test["Fare"])):
    if pd.isnull(test["Fare"][x]):
        pclass = test["Pclass"][x] #Pclass = 3
        test["Fare"][x] = round(train[train["Pclass"] == pclass]["Fare"].mean(),

#map Fare values into groups of numerical values
train['FareBand'] = pd.qcut(train['Fare'], 4, labels = [1, 2, 3, 4])
test['FareBand'] = pd.qcut(test['Fare'], 4, labels = [1, 2, 3, 4])

#drop Fare values
train = train.drop(['Fare'], axis = 1)
test = test.drop(['Fare'], axis = 1)
```

In [71]:
```
train.head(10)
```

Out[71]:

	PassengerId	Survived	Pclass	Sex	SibSp	Parch	Embarked	AgeGroup	CabinBool	Title	Far
0	1	0	3	0	1	0	1	4.0	0	1	
1	2	1	1	1	1	0	2	6.0	1	3	
2	3	1	3	1	0	0	1	5.0	0	2	
3	4	1	1	1	1	0	1	5.0	1	3	
4	5	0	3	0	0	0	1	5.0	0	1	
5	6	0	3	0	0	0	3	5.0	0	1	
6	7	0	1	0	0	0	1	6.0	1	1	
7	8	0	3	0	3	1	1	1.0	0	4	
8	9	1	3	1	0	2	1	5.0	0	3	
9	10	1	2	1	1	0	2	3.0	0	3	

In [72]:
```
test.head(10)
```

Out[72]:

	PassengerId	Pclass	Sex	SibSp	Parch	Embarked	AgeGroup	CabinBool	Title	FareBand
0	892	3	0	0	0	3	5.0	0	1	1
1	893	3	1	1	0	1	6.0	0	3	1
2	894	2	0	0	0	3	7.0	0	1	2
3	895	3	0	0	0	1	5.0	0	1	2
4	896	3	1	1	1	1	4.0	0	3	2
5	897	3	0	0	0	1	3.0	0	1	2
6	898	3	1	0	0	3	5.0	0	2	1
7	899	2	0	1	1	1	5.0	0	1	3
8	900	3	1	0	0	2	3.0	0	3	1
9	901	3	0	2	0	1	4.0	0	1	3

Splitting the data and testing models

In [74]:
```
from sklearn.model_selection import train_test_split

predictors = train.drop(['Survived', 'PassengerId'], axis=1)
target = train["Survived"]
x_train, x_val, y_train, y_val = train_test_split(predictors, target, test_size
```

Random Forest

In [75]:
```
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier

randomforest = RandomForestClassifier()
randomforest.fit(x_train, y_train)
y_pred = randomforest.predict(x_val)
acc_randomforest = round(accuracy_score(y_pred, y_val) * 100, 2)
print(acc_randomforest)
```

```
83.25
```

Stochastic Gradient Descent

In [76]:
```
from sklearn.linear_model import SGDClassifier

sgd = SGDClassifier()
sgd.fit(x_train, y_train)
y_pred = sgd.predict(x_val)
acc_sgd = round(accuracy_score(y_pred, y_val) * 100, 2)
print(acc_sgd)
```

```
78.17
```

Gradient Boosting Classifier

In [77]:
```
from sklearn.ensemble import GradientBoostingClassifier

gbk = GradientBoostingClassifier()
gbk.fit(x_train, y_train)
y_pred = gbk.predict(x_val)
acc_gbk = round(accuracy_score(y_pred, y_val) * 100, 2)
print(acc_gbk)
```

```
84.77
```

prediction accuraccy of the GBC is highest as 84.77 percent

In [ ]:
```

```