Topic - Titanic: Machine Learning from Disaster

Problem Description: On April 15, 1912, Royal Mail Ship Titanic was a British Luxury ship which sank during its first voyage after colliding with an iceberg. Because of this 1502 out of 2224 passengers and crew died. The reason which led to so many death was mentioned as not having so many lifeboats for the passengers and the most of people who survived were women, children, and first class passengers.

This problem is a classification problem, because here the outcome will only be Survived or Not Survived in numerical values it will be 1 or 0.

Using Machine learning different models i am going to predict the survival.

Data Description: The data has been provided by Kaggle. The dataset has two different categories of data.

1. Training data,
2. Test Data.

There are 891 records and the features has been described below:

Variables	**Definition**	**Key**
survival	survival	0 = No, 1 = Yes
pclass	Ticket Class	1 = 1st, 2 = 2nd, 3 = 3rd
sex	Sex	1 = Male, 0 = Female **(Modified in Numbers)**
Age	Age in years	
sibsp	Number of Siblings/ spouses aboard the Titanic	
parcg	Number of Parents/ Children aboard the Titanic	
ticket	Ticket Number	
fare	Ticket 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.

1. Operations Performed on Titanic Dataset

I loaded the training and test dataset using pd.read_cv where pd is the pandas library.

```
In [2]: #Load training data
        train_data = pd.read_csv("/Users/pankaj/Desktop/titanic/train.csv")
        test_data = pd.read_csv("/Users/pankaj/Desktop/titanic/test.csv")
```

I found the total number of records in Training and Test data set is 891 and 418.

```
In [56]: #find the length of training data and test data

         length_trainData = len(train_data)
         length_testData = len(test_data)
         print(length_trainData)
         print(length_testData)

         891
         418
```

When i described the training dataset using python, i found out that only 38% of the passengers survived on Titanic.

```
In [7]: train_data.describe()
```

Out[7]:

	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

I checked which columns contains any missing values and found out that AGE, CABIN and EMBARKED has missing values and we need to fill these missing values with some meaning values. We can do two things, either we can remove or replace the null values. If we have more than 80% of the values which are null, then we can delete that particular column. We found that Cabin has 687 values which are null, Age has 177 values as null and Embarked has only 2 values which are null.

In [14]:
```
total = train_data.isnull().sum().sort_values(ascending=False)
percent_1 = train_data.isnull().sum()/train_data.isnull().count()*100
percent_2 = (round(percent_1, 1)).sort_values(ascending=False)
missing_data = pd.concat([total, percent_2], axis=1, keys=['Total', '%'])
missing_data.head(5)
```

Out[14]:

	Total	%
Cabin	687	77.1
Age	177	19.9
Embarked	2	0.2
Fare	0	0.0
Ticket	0	0.0

We can see that Cabin has 77.1% values are null, so we can remove this column. For Age and Embarked, we can replace null with some meaning value.

For Age, we can find the mean and standard deviation and fill these null values with these. And For Embarked, we found that there are three different values for Embarked which are S, Q and C. and S is most frequent enough.

In [16]:
```
train_data['Embarked'].describe()
```

Out[16]:
```
count      889
unique       3
top          S
freq       644
Name: Embarked, dtype: object
```

We can replace the null value with the most frequent value of Embarked which is S.

In [19]:
```
common_value = 'S'
data = [train_data, test_data]

for dataset in data:
    dataset['Embarked'] = dataset['Embarked'].fillna(common_value)

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

Out[19]:
```
0
```

We have to replace the Male and Female categorical values with the numerical values. So for Male, i have used 1 and for Female i have used 0.

```
# change male to 1 and female to 0
gender = {'male':1, 'female':0}
data = [train_data, test_data]
for dataset in data:
    dataset['Sex'] = dataset['Sex'].map(gender)
train_data.head(10)
```

And I change the categorical values of Embarked to numerical.

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

for dataset in data:
    dataset['Embarked'] = dataset['Embarked'].map(ports)
train_data.head(10)
```

I changed the fare type from Float to Int. And I have deleted Ticket, Cabin, And Name columns as these features were not contributing much to the decisions. So after cleaning all the data, I got this dataset.

```
In [25]: train_data.head(10)
```

Out[25]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	1	22	1	0	7	0
1	1	1	0	38	1	0	71	1
2	1	3	0	26	0	0	7	0
3	1	1	0	35	1	0	53	0
4	0	3	1	35	0	0	8	0
5	0	3	1	28	0	0	8	2
6	0	1	1	54	0	0	51	0
7	0	3	1	2	3	1	21	0
8	1	3	0	27	0	2	11	0
9	1	2	0	14	1	0	30	1

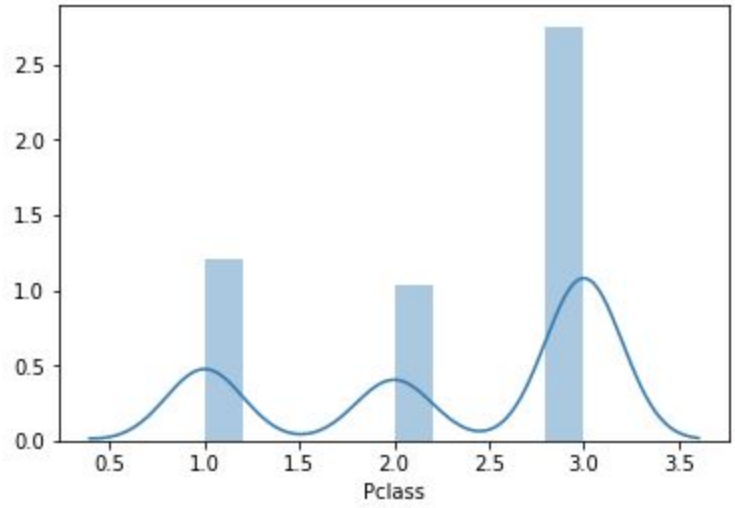
Data Visualization:

I plotted a histogram for the Pclass feature and found out that most of the passengers were from 3rd Class.

```
sns.distplot(train_data['Pclass'])
```

```
/anaconda3/lib/python3.6/site-packages/matplotlib/axes/_
and has been replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and h
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x10a3b0ac8>
```



I checked for the correlation coefficient for the features. Correlation is an index between that range from -1 to 1. When the value is near zero there is no linear relationship. As the correlation gets closer to plus or minus one, the relationship is stronger. A value of one (or negative one) indicates a perfect linear relationship between two variables.

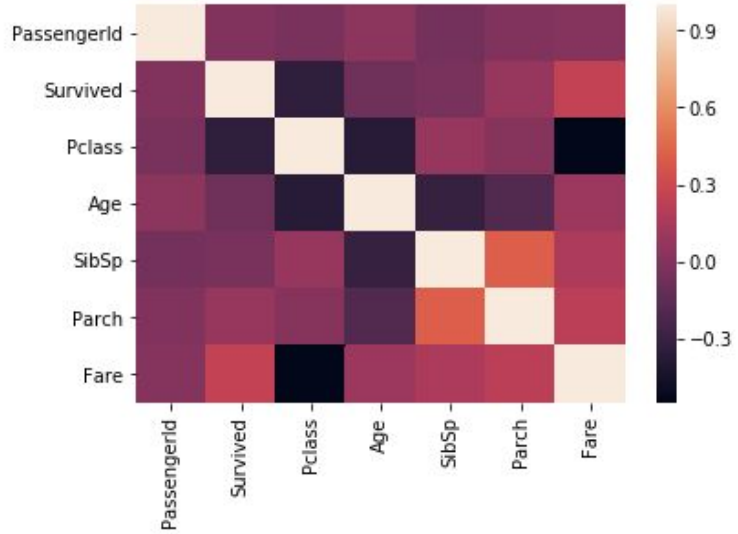
```
train_data.corr()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067
SibSp	-0.057527	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000

This is the heatmap of the correlation coefficient of the features.

```
sns.heatmap(corr,
        xticklabels=corr.columns,
        yticklabels=corr.columns)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a168f9128>
```

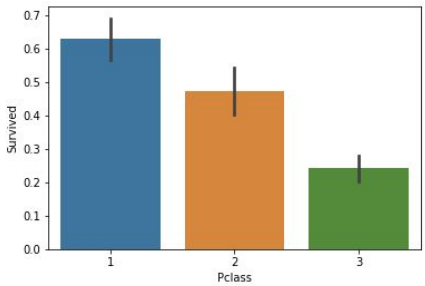


When i compared the number of survival based on the Passenger class, i found out that most of the survivals are from the First class.

```
# Class vs Survived
SurvicedvsClass = train_data[['Pclass', 'Survived']].groupby(['Pclass'], as_index=False).mean().sort_values(by='Survive
print(SurvicedvsClass)
```

```
   Pclass  Survived
0       1  0.629630
1       2  0.472826
2       3  0.242363
```

I also prepared the bar plot of the Passenger class and survival.

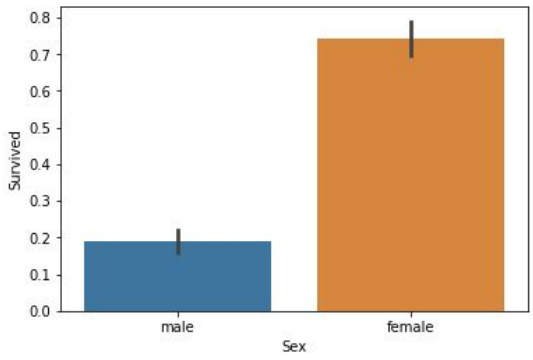


And when the survival was compared with the sex. It showed that Female were survived more than the male.

```
# sex vs Survived
print(train_data[["Sex", "Survived"]].groupby(['Sex'], as_index=False).mean().sort_values(by='Survived', ascending=Fals
```

```
   Sex  Survived
0    0  0.742038
1    1  0.188908
```

I again, plotted the bar graph for the Sex and Survival and it's clear in the bar graph that females survived more than male.



Machine Learning Models:

I used different machine learning models to compare the results.

Initially, i used **Perceptron**, and my accuracy was about 65-75% on the model (not kaggle).

```
perceptron = Perceptron(max_iter=5)
perceptron.fit(X_train, Y_train)

Y_pred = perceptron.predict(X_test)

acc_perceptron = round(perceptron.score(X_train, Y_train) * 100, 2)
```

```
acc_perceptron
```

```
65.88
```

Later i used **Multi-Level Perceptron** and my accuracy on the model increased upto 96% and on kaggle it increased upto 78%.

```
mlp_perceptron = MLPClassifier(hidden_layer_sizes = (784), max_iter = 1000, alpha = 0.00001,
                               solver ='lbfgs', verbose=True, random_state=20, tol = 0.00000001, activation ='tanh')
#mlp_perceptron = Perceptron(max_iter=5)
mlp_perceptron.fit(X_train, Y_train)
Y_pred = mlp_perceptron.predict(X_test)
acc_perceptron = round(mlp_perceptron.score(X_train, Y_train) * 100, 2)
acc_perceptron

#relu - 84.96
#tanh - 96
```

```
96.86
```

I used Logistic Regression and the accuracy which i got was 80%.

```
logreg = LogisticRegression()
logreg.fit(X_train, Y_train)

Y_pred = logreg.predict(X_test)

acc_log = round(logreg.score(X_train, Y_train) * 100, 2)

acc_log
```

```
80.25
```

Also, i used **Random forest** and the accuracy which i got is is 96.86% which is almost same as the multi level perceptron.

```
random_forest = RandomForestClassifier(n_estimators=100)
random_forest.fit(X_train, Y_train)

Y_prediction = random_forest.predict(X_test)

random_forest.score(X_train, Y_train)
acc_random_forest = round(random_forest.score(X_train, Y_train) * 100, 2)

acc_random_forest
```

```
96.86
```

I also used, **Gaussian Naive Bayes** model and i got 79% accuracy level.

```
gaussian = GaussianNB()
gaussian.fit(X_train, Y_train)
Y_pred = gaussian.predict(X_test)
acc_gaussian = round(gaussian.score(X_train, Y_train) * 100, 2)
acc_gaussian
```

```
79.35
```

Conclusion:

Out of all the models which i applied, Multi-level Perceptron and Random forest shows really great performance than Logistic Regression and Gaussian Naive Bayes. I checked the confusion matrix and Precision and Recall of random forest and found out the result below.

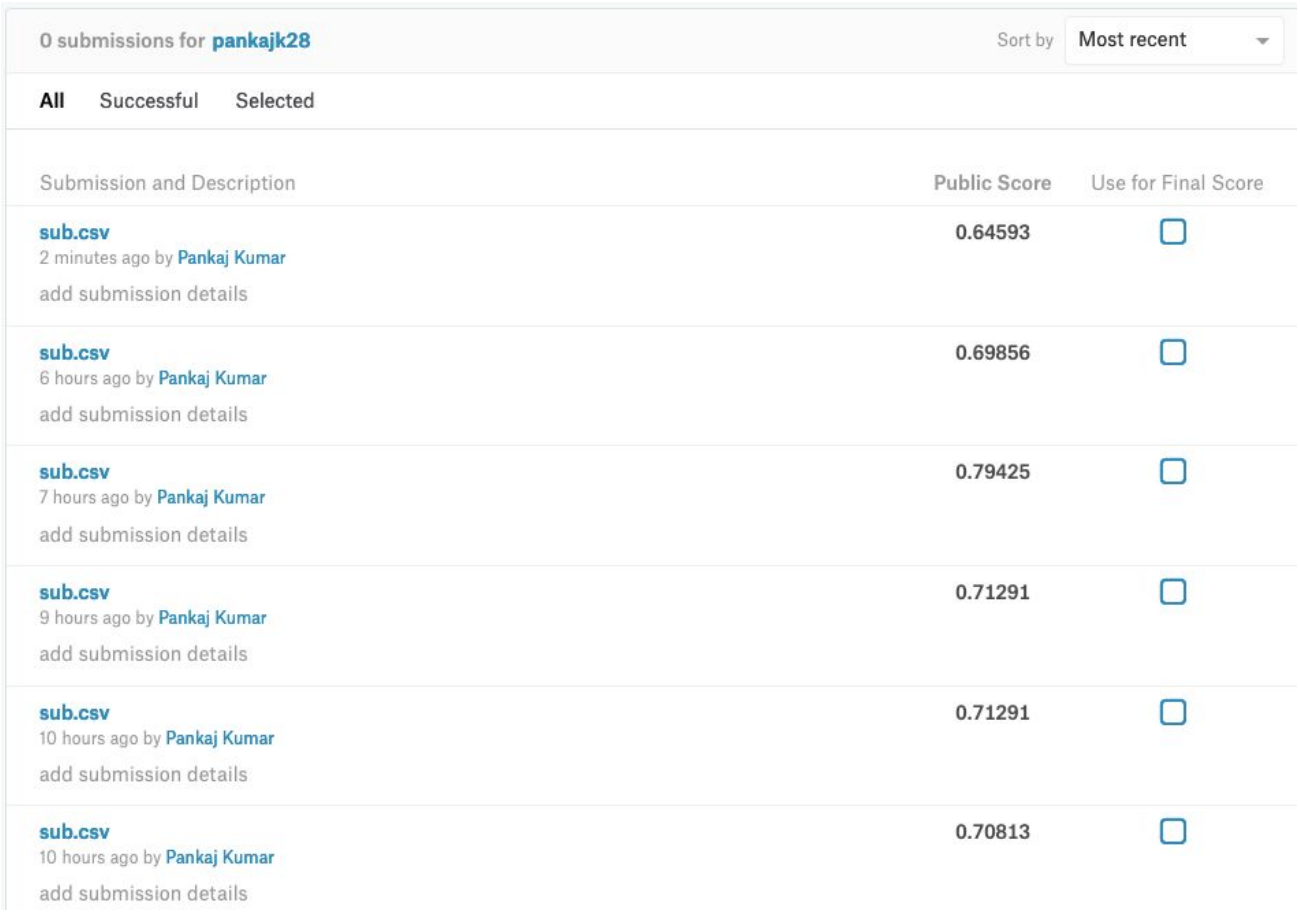
```
predictions = cross_val_predict(random_forest, X_train, Y_train, cv=3)
confusion_matrix(Y_train, predictions)
```

```
array([[460,  89],
       [ 96, 246]])
```

```
print("Precision:", precision_score(Y_train, predictions))
print("Recall:",recall_score(Y_train, predictions))
```

```
Precision: 0.7343283582089553
Recall: 0.7192982456140351
```

When i uploaded my csv file on kaggle, based on different models, i got different different accuracy, but the highest accuracy which i got was 79% using multi level perceptron.



Reference:

1. https://www.kaggle.com/c/titanic/
2. https://scikit-learn.org/stable/modules/naive_bayes.html#gaussian-naive-bayes
3. https://scikit-learn.org/stable/modules/neural_networks_supervised.html#multi-layer-perceptron
4. https://scikit-learn.org/stable/modules/ensemble.html#forest