Titanic Dataset

Analysis and Prediction

By : JASSKARAN SINGH - 19 June 2020

INTRODUCTION

The goal of the project was to predict the survival of passengers based off a set of data. I used the Titanic Dataset to retrieve necessary data and evaluate accuracy of predictions made by the model designed by me.
I underwent this project as assigned to me under Summer Training programme provided by IIT, Kanpur. The titanic data has been split into two groups, a 'train set' and a 'test set' using sklearn's train_test_split method. For the training set, I provided with the outcome (whether or not a passenger survived). I used this set to build my model to generate predictions for the test set. For each passenger in the test set, I had to predict whether or not they survived the sinking.
My score was the percentage of correctly predictions.
In my work, I learned

1. Programming language Python and its libraries NumPy (to perform matrix operations), Pandas (to import dataset and apply changes to it), MatPlotLib and SeaBorn(to display graphs and charts) and SciKit-Learn (to apply machine learning algorithms).
2. Working on Jupyter files.
3. Several machine learning algorithms (decision tree, random forest, logistic regression).
4. Feature Engineering techniques.

During my work, I used:

1. Jupyter Notebook (.ipynb)
2. Python 3.7 version
3. Microsoft Excel

TASK DEFINATION

Given the dataset, with input columns namely:

1. PassengerId
2. Pclass
3. Name
4. Sex
5. Age
6. SibSp
7. Parch
8. Ticket
9. Fare
10. Cabin
11. Embarked

And their corresponding survival column, I had to predict the survival of passengers based off a set of data. Many input columns are here which didn't contribute to the fact that whether or not a person survived or not. My task was then to filter the data need to make predictions. I tried to establish relations between each input column to the survival column. I used the python codes and libraries to show statistical relations in form of bar charts, histobars etc. Next step was data cleaning as many rows of data were missing or not appropriate.

After cleaning up data and taking up useful input columns next step was to select an appropriate machine learning model to make predictions. To make results and predictions better and accurate, dataset was split into training set and test set, so as to avoid the problem of 'overfitting' of the model.

Following this brief summary of tasks-to-complete, I checked the accuracy of the machine learning model which came out to be around 84 percent. A confusion matrix was also employed to check truth values and predicted values.

METHODOLOGY

The useful libraries were imported into the jupyter notebook such as numpy for mathematical imputations, pandas for dataframe working, matplotlib and seaborn for visual representation of math. The 'titanic.csv' file containing the dataset was then loaded into the notebook.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sn
```

```
df = pd.read_csv('titanic.csv')
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Moving on, I checked that whether or not the dataset has missing values in it or not. Applying some functions it was clear that some columns contained nan values. So, next step was to clean the dataset having nan values. The following image shows which columns had those nan values along with the number of nan values corresponding to each column.

```
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
```

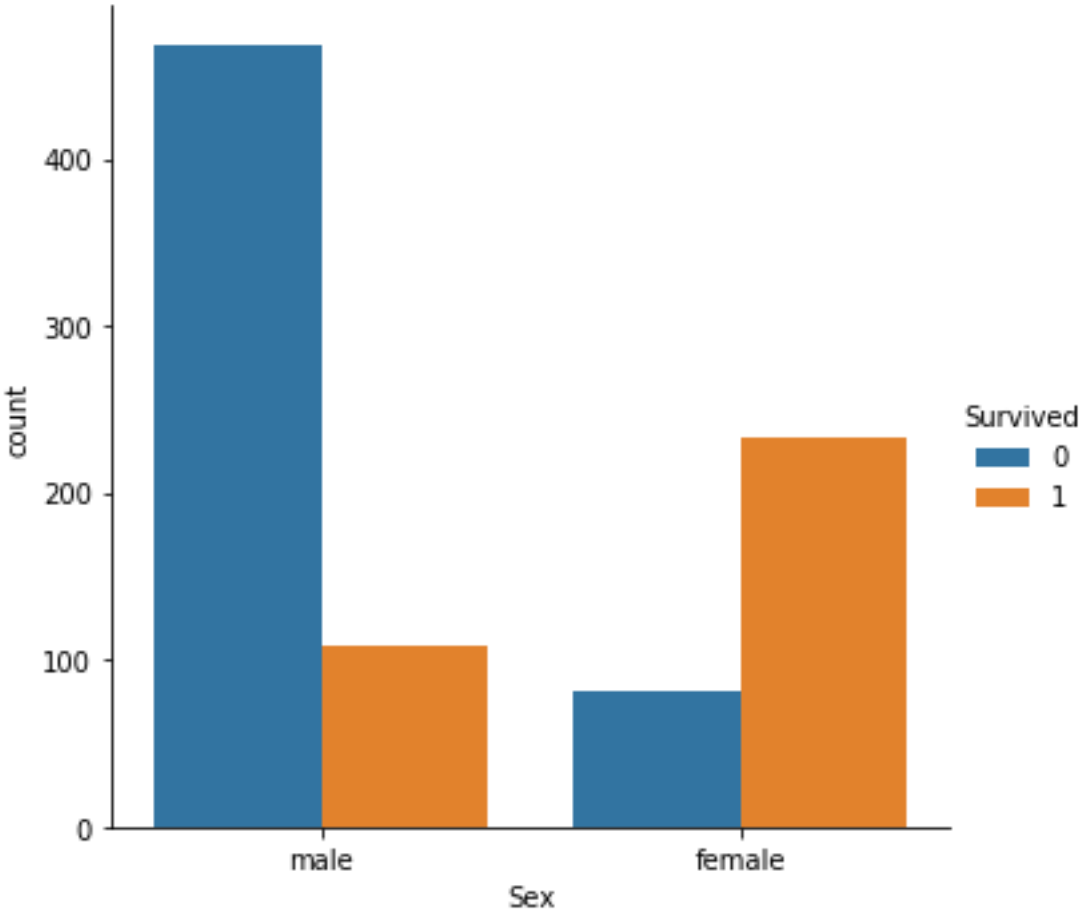
The ‘Cabin’ column had 687 nan values. The total dataset had 891 rows which meant that ‘Cabin’ column had maximum values as nan, so that could be dropped easily. ‘Age’ column had nan values which could be replaced with the median of all ages present in dataset. Further on, I observed that median and mean of all ages was nearly same so I chose median to fill the nan ‘Age’ entries. Same with ‘Embarked’ column as it had just 2 nan values, I decided to fill it up with one of the available Embarked locations.

By this time, I had no nan values in my dataset and was ready to establish relationships between input columns and the output ‘Survived’ column.

```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
dtype: int64
```

Defining and observing relations wrt Survival

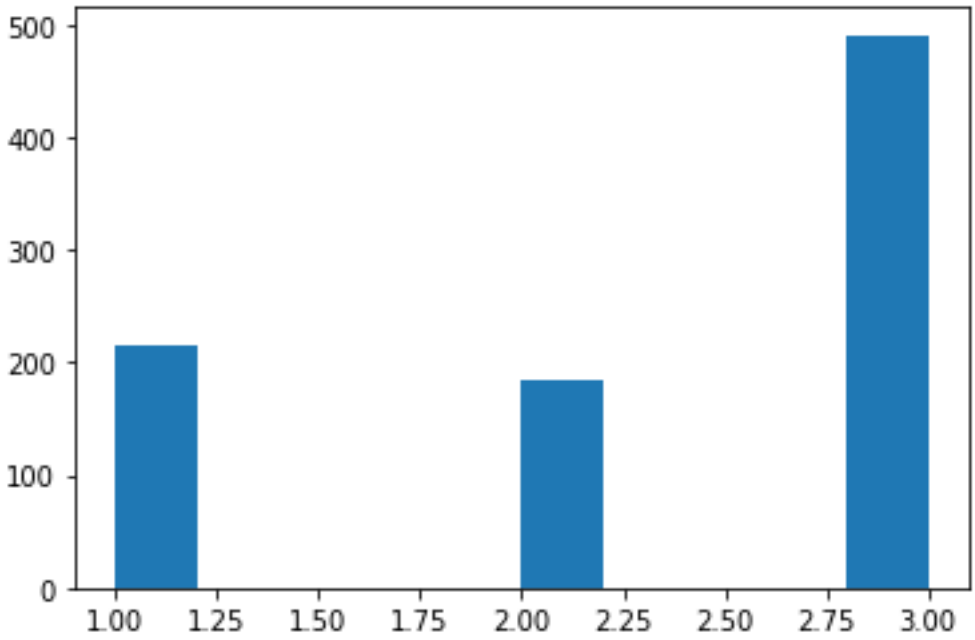
'Sex' column relation with 'Survived' column:



I observed that number of females who survived was more than the number of males who survived. This can be due to the reason that the females were given priority during the rescue when the incident occurred. The female causalities were much less than the male casualties.

So the 'Sex' columns play important role in survival prediction

'Pclass' column relation with 'Survived' column:



This graph shows distribution of people in the three different classes. Majority of people belonged to class 3. Further more we see that on average their survival ratio was the least as compared to people in class 1 or 2. It can be seen below. But ratios are different and can be important in predictions.

```
df[['Pclass', 'Survived']].groupby('Pclass').mean()
```

Pclass	Survived
1	0.629630
2	0.472826
3	0.242363

'Embarked' column relation with 'Survived' column:

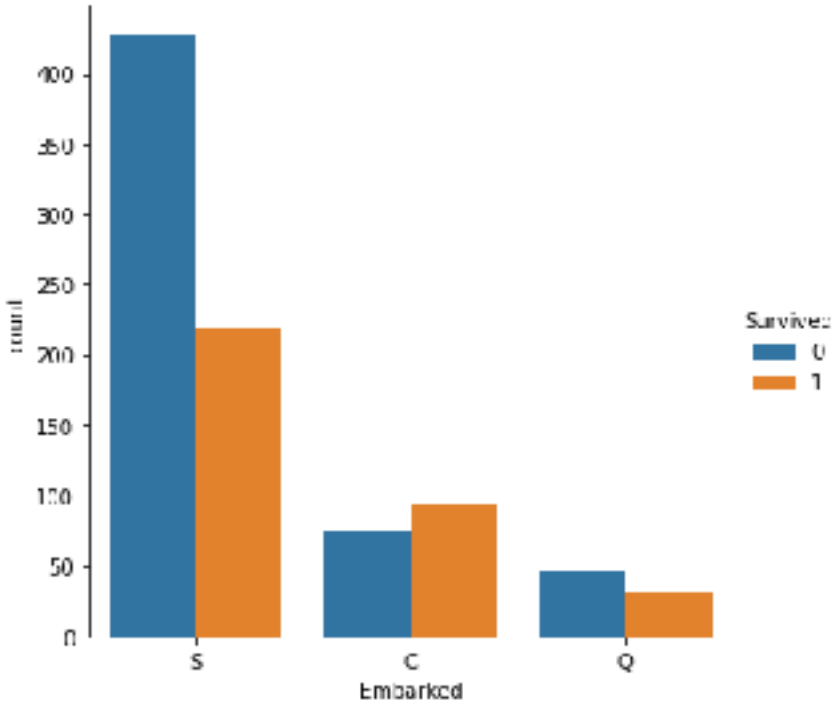
In total there were 3 embarking locations of the journey. I calculated the total passengers who embarked at each point and formed that 646, max people embarked at location 'S'. Further more I saw the survival ratio wrt to Embarking locations which can be seen below.

```
df['Embarked'].value_counts()
```

```
S    646
C    168
Q     77
Name: Embarked, dtype: int64
```

```
df[['Embarked', 'Survived']].groupby('Embarked').mean()
```

Embarked	Survived
C	0.553571
Q	0.389610
S	0.339009



'Fare' column relation with 'Survived' column:

I see that fare, too, played important role in prediction as survival was also dependent on it. There were different values of fare and accordingly their survival also varied. I calculated the average fare and performed analysis on it and came up with broadly two points:

1. Passengers who paid more than Average Fare survived more in number. Also they were less people who paid more than average.
2. Passengers who paid less than Average Fare survived less in number. Also they were more people who paid less than average.

```
df[df.Fare>avg_fare].Survived.value_counts()
# Passengers who paid more than Average Fare had more people survived. Also they were lesser in number
```

```
1    126
0     85
Name: Survived, dtype: int64
```

```
df[df.Fare<=avg_fare].Survived.value_counts()
#  Passengers who paid less than Average Fare had less people survived. Also they were greater in number.
```

```
0    464
1    216
Name: Survived, dtype: int64
```

This can be due to class difference or their on board location difference. So we see that 'Fare' column also plays important role in prediction.

So I chose these 4 input parameters, which according to me will provide high accuracy in survival prediction, namely:

1. Age
2. Sex
3. Fare
4. Embarked

ML Model Working

So, I chose the above mentioned 4 columns as input. A new dataframe was created, shown below, having all these input columns.

	Pclass	Fare	Survived	Sex_female	Sex_male	Embarked_C	Embarked_Q	Embarked_S
0	3	7.2500	0	0	1	0	0	1
1	1	71.2833	1	1	0	1	0	0
2	3	7.9250	1	1	0	0	0	1
3	1	53.1000	1	1	0	0	0	1
4	3	8.0500	0	0	1	0	0	1

Still to get better results a scaling was needed in numeric values of 'Pclass' and 'Fare' column. To tackle this I used MinMaxScaler method and added new updated values to another dataframe which looked as shown below:

	Pclass	Fare	Survived	Sex_female	Sex_male	Embarked_C	Embarked_Q	Embarked_S
0	1.0	0.014151	0	0	1	0	0	1
1	0.0	0.139136	1	1	0	1	0	0
2	1.0	0.015469	1	1	0	0	0	1
3	0.0	0.103644	1	1	0	0	0	1
4	1.0	0.015713	0	0	1	0	0	1

Now that inputs and output data-frames were ready. Next step was to split the data into training and testing dataset. The way I did this was using scikit-learn's library.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(inputs, target, test_size=0.2)
```

This helps to eliminate the problem of overfitting the dataset for predictions and better accuracy of results. I used various classification ML models like DecisionTreeClassifier(), RandomForrestClassifier and LogisticRegressor(), and found the following accuracies:

1. Random Forrest : 81.27
2. Decision Tree : 84.35
3. Logistic : 79.15

```
from sklearn import tree
model = tree.DecisionTreeClassifier()
```

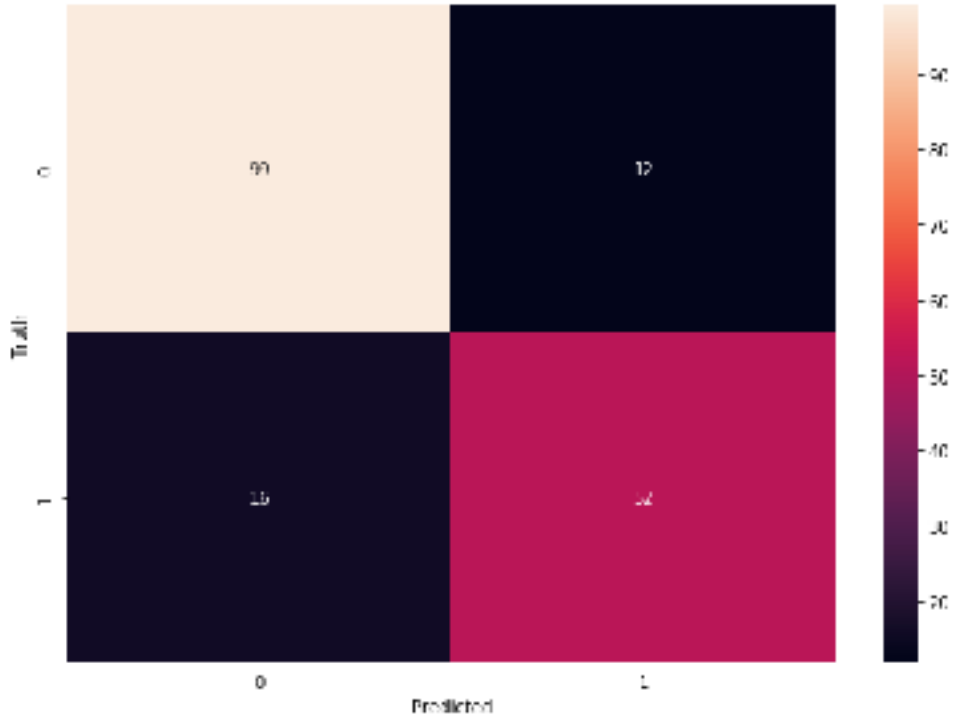
```
model.fit(X_train,y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

```
model.score(X_test, y_test)
```

```
0.8435754189944135
```

So I decided to make it with Decision Tree and achieved an accuracy of 84.35%.
I also printed a confusion matrix using heat-map for better visualisation.



SUMMARY

The input data containing 891 rows and 11 variables is imported. First, data cleaning is performed to impute missing values and convert data types to suitable ones. Next, visualize data to uncover patterns in survival. Passenger Class, Fare, Age and Sex seemed to impact survival. New variables are also created to extract more information from existing variables. For example dummy variables were created for Passenger Class and Sex. These dummy variables along with new columns were added to our inputs data-frame and previous columns were dropped.
Once data is cleaned and new features are created, next step is to try out various models using multiple variables selected. Classification ML models like DecisionTreeClassifier(), RandomForrestClassifier and LogisticRegressor() were tested and DecisionTreeClassifier() was the best outcome with highest accuracy of 84.35 %