Titanic Survivor Prediction

Using Random Forest Classifier

Presented By:

Mr.Aditya Viswabhusan
Mr.Ashutosh Kar

Submitted To:

ICT Cell
IIT Kanpur

INTRODUCTION

- The sinking of the RMS Titanic is one of the most infamous shipwrecks in history.
- In this project, we were asked to analyse what sorts of people were likely to survive in the disaster.
- The project was completed by applying different tools of machine learning to predict which passengers survived the tragedy.
- Before feeding on the data to our machine learning model, different pre-processing techniques were applied.
- Once the data was ready, different machine learning models were tried upon to find the best accuracy.

Goals And Objective:

- Purpose of the Project is to generate predictions for Titanic Passengers.

- Objective of the Project is to build a Classification model and determine whether a passenger has survived or not.

Software Required:

1. Python v3.8
2. Anaconda Navigator v1.9.12
3. Jupyter Notebook v6.0.3

Libraries Used:

- Pandas
- Numpy
- Scikit Learn

Random Forest Classifier

- Random Forest Classifier is a set of decision trees from randomly selected subset of training set. It aggregates the votes from different decision trees to decide the final class of the test object.
- It runs efficiently on large databases and produces highly accurate classifier
- It generates an internal unbiased estimate of the generalization error as the forest building progresses.
- It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.

Implementation:

- Importing necessary libraries.
- Importing the datasets.
- Cleaning and Analyzing the dataset
- Building the model.
- Using Random Forest for making the prediction.

Importing Necessary Libraries:

In [1]:

```
import pandas as pd;
import numpy as np
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.feature_selection import SelectKBest, chi2, f_regression
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
```

Reading The Data from dataset:

```
In [3]: df=pd.read_csv('train.csv');
```

```
In [3]: df.head()
```

Out[3]:

	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

Analyzing the Data:

```
In [4]: df.describe()
```

Out[4]:

	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

Pclass Vs. Survived

The passengers having Pclass as 1,2 had higher Survival rate.

```
In [10]: df[['Pclass','Survived']].groupby('Pclass',as_index=False).mean()
```

Out[10]:

	Pclass	Survived
0	1	0.629630
1	2	0.472826
2	3	0.242363

Sex vs. Survived:

The Female passengers had a higher rate of survival.

```
In [5]: df.Sex.value_counts()

Out[5]: male      577
        female    314
        Name: Sex, dtype: int64
```

```
In [6]: df[['Sex','Survived']].groupby('Sex',as_index=False).mean()
```

Out[6]:

	Sex	Survived
0	female	0.742038
1	male	0.188908

SibSp vs. Survived

The passenger having less SibSp had higher rate of survival.

```
In [7]: df[['SibSp','Survived']].groupby('SibSp',as_index=False).mean()
```

Out[7]:

	SibSp	Survived
0	0	0.345395
1	1	0.535885
2	2	0.464286
3	3	0.250000
4	4	0.166667
5	5	0.000000
6	8	0.000000

Parch Vs. Survived

The passengers having Parch as 1,2,3 had a higher rate of Survival.

```
In [11]: df[['Parch','Survived']].groupby('Parch',as_index=False).mean()
```

Out[11]:

	Parch	Survived
0	0	0.343658
1	1	0.550847
2	2	0.500000
3	3	0.600000
4	4	0.000000
5	5	0.200000
6	6	0.000000

Embarked Vs. Survived

The passengers Embarked from ‘C’ had a higher rate of Survival.

```
In [17]: df[['Embarked','Survived']].groupby('Embarked',as_index=False).mean()
```

Out[17]:

	Embarked	Survived
0	C	0.553571
1	Q	0.389610
2	S	0.336957

Findings:

From the above analysis , we determine that ‘Pclass’ , ‘Sex’, ‘Age’, ‘Fare’ , ‘Embarked’ majorly determine the survival outcome.

```
In [43]: df1 = df[['Pclass','Sex','Age','Fare','Embarked']]
```

```
In [44]: df1.head()
```

Out[44]:

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

Finding the NULL values in our dataset:

- The Null values found in the dataset were majorly from two classes i.e. Age and Embarked
- The Age had 177 Null values and the Embarked had 2 Null values.

```
In [45]: df1.isnull().sum()

Out[45]: Pclass        0
         Sex           0
         Age         177
         Fare          0
         Embarked      2
         dtype: int64
```

Filling up the NULL values:

- **The Embarked feature was substituted with ‘S’ as many passengers had embarked from ‘S’.**
- **The Age was substituted with the mean value.**

```
In [27]: df1.Embarked.value_counts()

Out[27]: S    644
         C    168
         Q     77
         Name: Embarked, dtype: int64

In [28]: df1.fillna({'Age':df1.Age.mean(),'Embarked':'S'},inplace=True)
```

Final Attributes Considered:

- The final attributes considered were : ‘Sex’, ‘Pclass’, ‘Age’, ‘Fare’ and ‘Embarked’
- Encoding was done on ‘Sex’ and ‘Embarked’ attributes.

```
In [57]: final_data.head()
```

Out[57]:

	Sex_female	Sex_male	Embarked_C	Embarked_Q	Embarked_S	Pclass	Age	Fare
0	0	1	0	0	1	3	22.0	7.2500
1	1	0	1	0	0	1	38.0	71.2833
2	1	0	0	0	1	3	26.0	7.9250
3	1	0	0	0	1	1	35.0	53.1000
4	0	1	0	0	1	3	35.0	8.0500

```
In [58]: final_data.isnull().sum()
```

```
Out[58]: Sex_female    0
         Sex_male      0
         Embarked_C    0
         Embarked_Q    0
         Embarked_S    0
         Pclass        0
         Age           0
         Fare          0
         dtype: int64
```

Selecting The K-Best Features and Scaling the data:

- To select the best features, SelectKBest method was used.
- To scale the data, Standard scaler was implemented.

```
In [64]: x=SelectKBest(score_func=chi2,k=8).fit_transform(x,y)

In [65]: scaler=StandardScaler()
         x=scaler.fit_transform(x)
```

Building The Model:

- **The train size is considered to be 80% and the test size is considered to be 20%.**
- **n_estimators were fixed at 40.**

```
In [66]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)

In [67]: classifier = RandomForestClassifier(n_estimators = 40, criterion = 'entropy', random_state = 0)
         classifier.fit(x_train, y_train)

Out[67]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
                                max_depth=None, max_features='auto', 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, n_estimators=40,
                                n_jobs=None, oob_score=False, random_state=0, verbose=0,
                                warm_start=False)
```

Accuracy Score and Confusion Matrix:

Accuracy was found to be 86.59% for the following prediction

```
In [38]: classifier.score(xtest,ytest)
Out[38]: 0.8659217877094972

In [39]: pred=classifier.predict(xtest)

In [40]: from sklearn.metrics import confusion_matrix

In [41]: confusion_matrix(ytest,pred)
Out[41]: array([[104,  12],
                [ 12,  51]], dtype=int64)
```

Conclusion:

- All previous classifications model gave accuracy which was less than 82.00%.
- No algorithm was able to give accuracy of 100.00%.
- Certain attributes like ‘PassengerId’, ’Ticket’, ’Cabin’, ‘Name’, ‘SibSp’, ‘Parch’ were not considered as they were not much affecting the target variable.
- From the analysis we found that Women,Children and First Class Passengers had a higher chance of survival, as compared to others.

Thank You !