Titanic – Machine Learning from Disaster

Shubham Deshmukh (#202006307), *x2020gfv@stfx.ca*

Introduction

The Project is based on an online competition "Titanic: Machine Learning from Disaster" posted in Kaggle (https://www.kaggle.com/c/titanic) The details of the competition are posted in the linked mentioned. The aim of the project is to predict the survival of the passengers boarded on the iconic Titanic ship during the infamous shipwreck in 1912.

The data used in the project is also available in the Kaggle competition page as CSV files, Information of 1309 travellers are given in the data sets, the data however are available in two files, first “train.csv” which contains data of 891 passengers whose survival information is also present. On the other hand, “test.csv” contains information of 418 passengers whose survival are to be predicted in this project.

In this project I have used these data of multiple passengers and build my machine learning model to predict weather the traveller survived or not during the disaster.

The Figure below shows the columns in the combined dataset and their type.

```
print(merged_train_test_data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1309 entries, 0 to 417
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  1309 non-null   int64
 1   Survived     1309 non-null   object
 2   Pclass       1309 non-null   int64
 3   Name         1309 non-null   object
 4   Sex          1309 non-null   object
 5   Age          1046 non-null   float64
 6   SibSp        1309 non-null   int64
 7   Parch        1309 non-null   int64
 8   Ticket       1309 non-null   object
 9   Fare         1308 non-null   float64
 10  Cabin        295 non-null    object
 11  Embarked     1307 non-null   object
 12  Type         1309 non-null   int64
dtypes: float64(2), int64(5), object(6)
memory usage: 143.2+ KB
None
```

A “Type” column is added, and I have given it a value of 1 for all the train data (819 rows) and 0 for all the testing data (418 rows) to Identify the data while prediction in the Model.

```
train_rawdata['Type'] = 1   #TRAINING DATA
test_rawdata['Type'] = 0    #TESTING DATA
```

Since the data type of most the columns are either object or float type, we need to cast them to integer type for better predictability by the model.

Data Cleaning & Processing

First, we need to check if the data columns contain any *NULL/NA/NaN* values, if yes, then we need to fill them using several ways like median, mean etc.

```
: print(merged_train_test_data.isnull().sum())
```

```
PassengerId       0
Survived          0
Pclass            0
Name              0
Sex               0
Age             263
SibSp             0
Parch             0
Ticket            0
Fare              1
Cabin          1014
Embarked          2
Type              0
dtype: int64
```

Embarked

The "Embarked" column only contains 'S', 'C' and 'Q' values and contains two null/NaN values. So, before filling the null values I have replaced 'S', 'C' and 'Q' as 0, 1 and 2 respectively for casting them to integer value and better understanding by the model.

Old Values: S, C, Q

New Values: 0, 1, 2

I have filled the null value with mode (most frequent) value of the column.

```
#Replacing S=0, C=1, Q=2 & Filling value to empty cells in embarked by most frequent(mode) value.
merged_train_test_data["Embarked"] = merged_train_test_data['Embarked'].replace(to_replace=['S', 'C','Q'], value=[0, 1, 2])
freq_embarked_value= merged_train_test_data['Embarked'].mode()
merged_train_test_data["Embarked"] = merged_train_test_data['Embarked'].fillna(int(freq_embarked_value)).astype(int)
```

Fare

There is only a single null value in column "Fare". So, in order to fill it I have calculated the median of "Fare" column and filled it at the place of null value. Also, I have changed the datatype of the column to integer.

```
#Converting fare to int and replace NA with 0
merged_train_test_data['Fare'].fillna( merged_train_test_data['Fare'].median()).astype(int)
```

Cabin

The “Cabin” contains 1064 null values in total, there are only 263 values in the column, however, it contains cabin number as its first digit which can be useful while predicting survival data. So I have extra the cabin number from the column while filling the null values with “U0” which stands for unkown cabin information. Also, I have replaced the cabin letter with numeric values as shown in the below figure.

```
# Remove Cabin letter from Cabin and dropping the other suffix values from it
cabinCode = {"A": 1, "B": 2, "C": 3, "D": 4, "E": 5, "F": 6, "G": 7, "U": 8}

merged_train_test_data['Cabin'] = merged_train_test_data['Cabin'].fillna("U0")
merged_train_test_data['Cabin'] = merged_train_test_data['Cabin'].map(
                lambda x: re.compile("([a-zA-Z]+)").search(x).group())

merged_train_test_data['Cabin'] = merged_train_test_data['Cabin'].map(cabinCode)
merged_train_test_data['Cabin'] = merged_train_test_data['Cabin'].astype(int)
```

Age

The Age column has 263 null values which needs to be filled in order to use it for prediction. I have used median to fill all null values of column “Age”, however I have made a grouping of age by “Sex” column and computed median values of “Age” column respective of their “Sex” value.

```
#Filling absent values of 'Age' Column in the entire data set using median based on Sex of the Passenger
merged_train_test_data["Age"].fillna(merged_train_test_data.groupby('Sex')['Age'].transform("median"), inplace=True)
```

The “Age” column contains data which is scattered between 0 to 80, In order to compress the data I used qcut to find a range which can be used to encode the data into numeric values.

New Values : 0, 1, 2, 3, 4, 5 and 6 as per the condition of the age range.

```
merged_train_test_data['Age'] = merged_train_test_data['Age'].astype(int)
merged_train_test_data.loc[ merged_train_test_data['Age'] <= 11, 'Age'] = 0
merged_train_test_data.loc[(merged_train_test_data['Age'] > 11) & (merged_train_test_data['Age'] <= 18), 'Age'] = 1
merged_train_test_data.loc[(merged_train_test_data['Age'] > 18) & (merged_train_test_data['Age'] <= 22), 'Age'] = 2
merged_train_test_data.loc[(merged_train_test_data['Age'] > 22) & (merged_train_test_data['Age'] <= 27), 'Age'] = 3
merged_train_test_data.loc[(merged_train_test_data['Age'] > 27) & (merged_train_test_data['Age'] <= 33), 'Age'] = 4
merged_train_test_data.loc[(merged_train_test_data['Age'] > 33) & (merged_train_test_data['Age'] <= 40), 'Age'] = 5
merged_train_test_data.loc[(merged_train_test_data['Age'] > 40) & (merged_train_test_data['Age'] <= 66), 'Age'] = 6
merged_train_test_data.loc[ merged_train_test_data['Age'] > 66, 'Age'] = 6
```

Sex

The “Sex” column only contains two types of values i.e. ‘male’ and ‘female’. I have replaced the values ‘male’ and ‘female’ with 1 and 0 respectively for casting to integer datatype.

Old Values: male, female

New Values: 1, 0

```
#Transform Sex data as Male = 1 and Female = 0
merged_train_test_data['Sex'] = merged_train_test_data['Sex'].replace(to_replace=['male', 'female'], value=[1, 0])
```

Name

The “Name” column does not contain much information which can be used directly. However, it contains the title of the passenger which can be useful for prediction. So, I have extracted the titles of the passengers.

Mr, Miss, Mrs., Master were the most frequent titles the other infrequent titles are replaced as RareFemale and RareMale titles. Also, I have replaced the Titles with integer values as shown in the below figure.

A new column “Title” is added for this and “Name” column is dropped afterwards.

```
#Extracting Title out of name column and dropping the name column
titleCode = {"Mr": 1, "Miss": 2, "Mrs": 3, "Master": 4, "RareFemale": 5, "RareMale": 6}
```

Ticket

The “Ticket” column contains random values and is of no use to create any pattern for prediction and hence its dropped form the dataset.

```
#Dropping the Ticket Column since it has random values which cannot be used to create any pattern
merged_train_test_data = merged_train_test_data.drop('Ticket',axis = 1)
```

Pclass

The “Pclass” column contains passenger’s class in which they are travelling, it has integer values 1, 2 and 3 and doesn’t contain any null values.

SibSp

The “SibSp” column contains number of siblings and spouses travelling with the passenger. It has integer values and has no null values.

Parch

The “Parch” column contains number of parents and children travelling with the passenger. It has integer values and has no null values.

Feature Engineering

FamilySize

A new column “FamilySize” is added to the dataset, It contains sum all the travellers (Parents, Children, Spouse, Siblings) travelling the main passenger (including the main passenger).

FamilySize = SibSp + Parch + 1

```
#Add familysize column
merged_train_test_data['FamilySize'] = merged_train_test_data['SibSp'] + merged_train_test_data['Parch'] + 1
```

FarePerPassenger

A new column “FarePerPassenger” is added to the dataset, it contains the fare per passenger. Here the total fare is divided by the size of the family. below if the formula for “FamilySize”

FarePerPassenger = Fare / FamilySize

```
#Adding Fare per passanger
merged_train_test_data['FarePerPassanger'] = (merged_train_test_data['Fare']/(merged_train_test_data['FamilySize']))
```

The data is divided into training and testing data after dropping all the irrelevant columns from the dataset which does not contribute much to the prediction model.

Dropped Column: PassengerId, Survived, Type and Fare

Machine Learning Model

I have used ***Random Forest Classifier*** as the model for predicting survival value for the passengers. I achieved my best score **0.78708** on Kaggle using this method after Cross validation, Parameter selection using GridSearch and Feature selection.

Model Selections

- Used Decision Tree and Linear Regression Model to predict the survival data however I achieved scores of **0.74** and **0.75** respectively.

- Later, used Gradient Boosting Classifier with that I achieved 0.72 as my Kaggle score.

- Used parameter selection and cross validation in Gradient Boosting Classifier, I score improved marginally to **0.75**.

- Lastly used Random Forest Classifier and find the most optimum score. In the initial submission my score was in the range **0.755 - 0.777**.

- Applied cross validation, parameter search and feature selection, also varied random seed and other parameters to find my best score of **0.78708** with Random Forest Classifier

Below are some snippets from Kaggle of Leaderboard, My Score and Submission Details

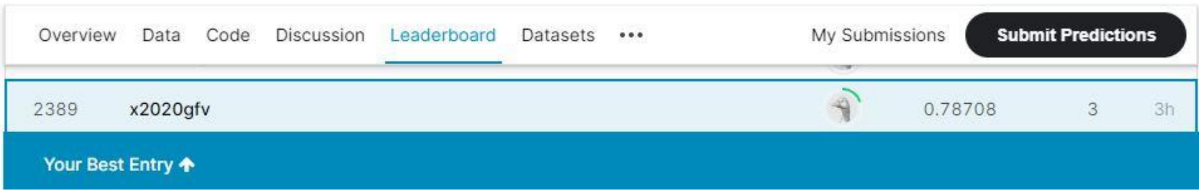

Figure. Leaderboad

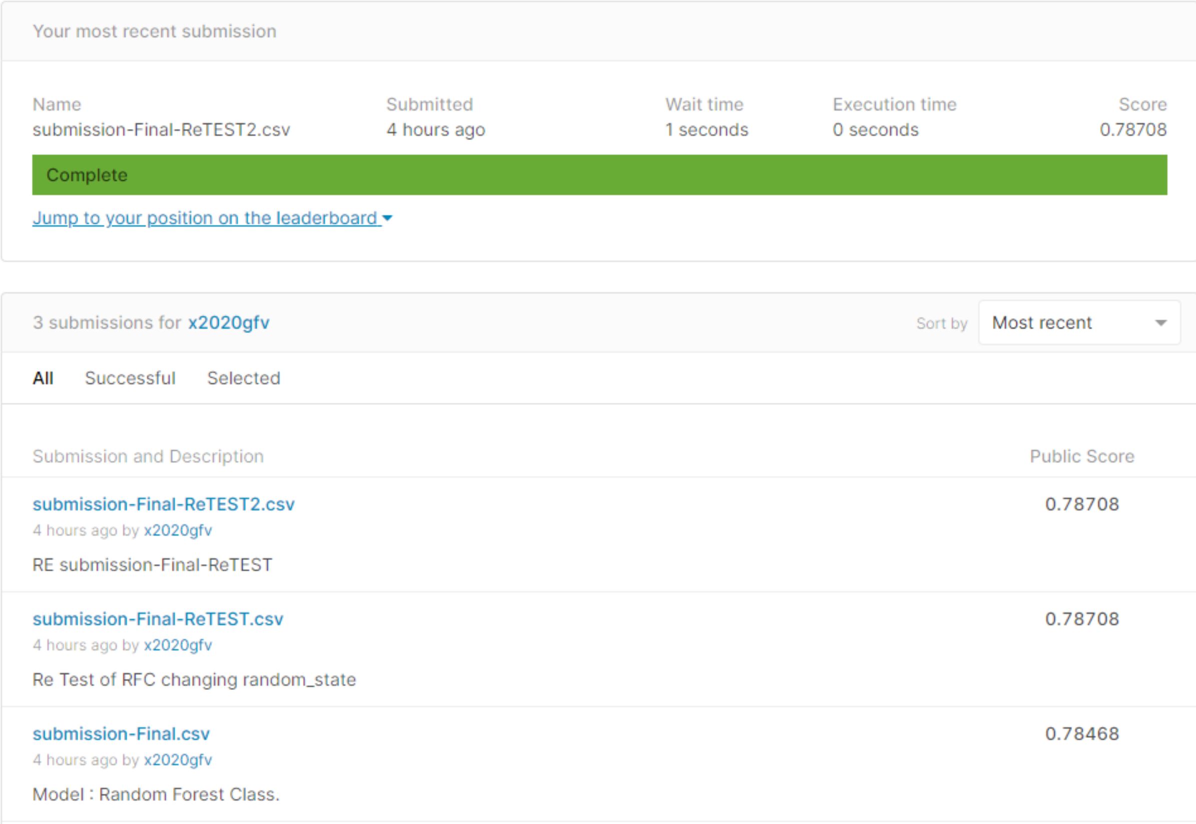

Figure. Submissions

Note: All submission are not displayed.