

# Outstanding Project 2

By:

Name: Jatin Jindal

E-mail Id: jatinjindal1199@gmail.com

## 1. Load the dataset from the csv file.

Ans:

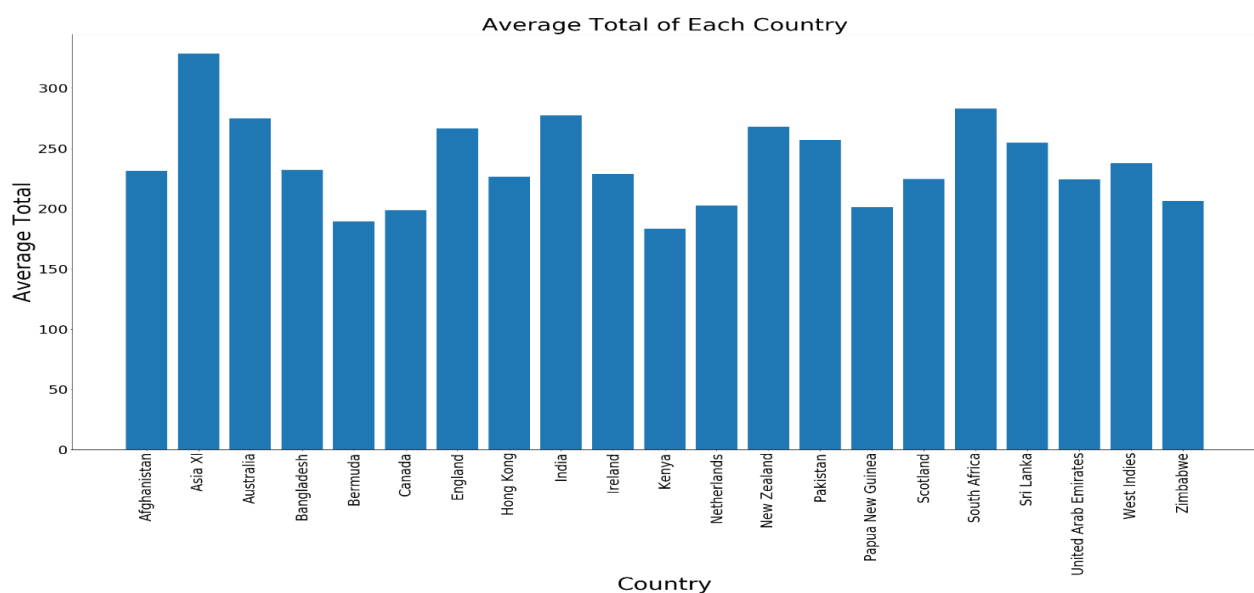
```
1 df=pd.read_csv('/content/drive/My Drive/Colab Notebooks/DataSets/odi.csv')
2 df
```

## 2. Use “groupby” operation, to find the average number of runs, scored by each country, and represent it on a bar graph.

Ans:

```
1 gr1=df.groupby(by=['bat_team'],as_index=False)['total'].mean()
2 gr1
```

```
1 plt.figure(figsize=(40,20))
2 plt.bar(gr1['bat_team'],gr1['total'],align='center')
3 plt.xticks(fontsize=30,rotation=90)
4 plt.yticks(fontsize=30)
5 plt.xlabel('Country',fontsize=45)
6 plt.ylabel('Average Total',fontsize=45)
7 plt.title('Average Total of Each Country',fontsize=45)
8 plt.show()
```



### 3. Handle Missing values:

a. If there are null values in continuous numerical column, replace the null values by

the mean of that column

b. If there are null values in ordinal numerical column, replace the null values by the

mode of that column

c. If there are null values in categorical column, replace the null values by the mode

of that column

d. If more than 50% the values in a column are null, then drop that entire column

**Ans:**

```
[ ] 1 df.isnull().sum()

mid      0
date     0
venue    0
bat_team 0
bowl_team 0
batsman   0
bowler    0
runs      0
wickets   0
overs     0
runs_last_5 0
wickets_last_5 0
striker   0
non-striker 0
total     0
dtype: int64
```

As we can see that there is no null value in this dataset. So no need to fill the nan values.

**4. Remove the columns, that you think, do not contribute to the total score, in the first innings.**

**Ans:**

```
1 df=df.drop(columns=['mid', 'date'])  
2 df
```

As the match id and date had no significance to the in the predicting the score of Cricket Match.

**5. Convert the categorical columns (if any), to numeric, using one hot encoding/ dummy encoding.**

**Ans:**

```
[ ] 1 from sklearn.preprocessing import LabelEncoder  
    2 le=LabelEncoder()  
    3 for i in ['batsman','bowler','venue','bat_team','bowl_team']:  
    4     df[i]=le.fit_transform(df[i])  
    5 df
```

Since the number of unique values are over 1000 so using dummy encoding can lead to huge increase in size of input data so we will be using One Hot Encoding to encode each categorical value to a specific number

**6. Pick “total” column, as the target variable**

**Ans:**

```
[ ] 1 y=df['total']  
    2 x=df.drop(columns=['total'])
```

Here the y is our Dependent Variable having data of total score and x is our Independent Variable .

## 7. Select the relevant features.

Ans:

```
[ ] 1 plt.figure(figsize=(13,13))
     2 sns.heatmap(x.corr(),annot=True)
```



We have used Heat Map to find correlation between all the columns.

Even though there are column which are highly corelated to each other but still we know that there dependence is due to flow of the game of cricket and each of these variables is necessary to predict the Total Score at the ed of the innings.

## 8. Perform train-test-split

Ans:

```
[ ] 1 from sklearn.model_selection import train_test_split
    2 x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.2,random_state=0)
    3 print(x_train.shape)
    4 print(x_test.shape)
    5 print(y_train.shape)
    6 print(y_test.shape)
```

```
(280719, 12)
(70180, 12)
(280719,)
(70180,)
```

x\_train, y\_train is the data we used for training our model and x\_test and y\_test is the data used for the testing of model.

We have taken the split ratio as 2:8 i.e the train data will have 80% of data and test data has 20% of data.

## 9. Perform Feature scaling

Ans:

```
[ ] 1 from sklearn.preprocessing import StandardScaler
    2 sc=StandardScaler()
    3 sc.fit(x)
    4 x_train_sc=sc.transform(x_train)
    5 x_test_sc=sc.transform(x_test)
    6 print(x_train_sc)
```

```
([ [ 0.09425697  1.32447013 -0.5017735 ... -0.803236 -0.50435767
    -0.49456454]
  [ 0.35354945  1.48610057 -1.33296354 ...  1.59514986 -1.03787645
    -0.76089103]
  [-1.46149793  1.00120924 -0.33553549 ...  0.39595693 -0.21981432
    -0.49456454]
  ...
  [-1.56521492 -0.45346473  0.82813057 ... -0.803236  2.02096457
    2.56819008]
  [ 1.10549765  0.35468748 -0.5017735 ...  0.39595693 -0.93117269
    -0.76089103]
  [-0.89105447  0.35468748 -0.83424951 ... -0.803236 -0.7533331
    -0.02849318]])
```

It is used to scale the values to a common scale as there were some columns having values in range upto 300 and some columns having value ranging only to a single digit number.

x\_train\_sc and x\_test\_sc is our scaled data.

## 10. Use

Ans:

### a. Use Linear Regression

```
[ ] 1 from sklearn.linear_model import LinearRegression
     2 lr_sc=LinearRegression()
     3 lr_sc.fit(x_train_sc,y_train)
     4 lr_sc.score(x_test_sc,y_test)
```



0.532113209510528

When linear Regression is used to Train the Model it gives the Accuracy of only 53.21%

### b. Use Decision Tree Regression

```
[ ] 1 from sklearn.tree import DecisionTreeRegressor
     2 tree_reg_sc=DecisionTreeRegressor()
     3 tree_reg_sc.fit(x_train_sc,y_train)
     4 tree_reg_sc.score(x_test_sc,y_test)
```




0.9418492707031377

When the Decision Tree is used to train the data it gives us the accuracy of 94.18% which is too good compared to Linear Regression

### c. Use Random Forest Regression

```
[ ] 1 from sklearn.ensemble import RandomForestRegressor
    2 forest_reg_sc=RandomForestRegressor(n_estimators=20)
    3 forest_reg_sc.fit(x_train_sc,y_train)
    4 forest_reg_sc.score(x_test_sc,y_test)
```

 0.9749269886773223


When the Random Forest is used to train the data it gives us the accuracy of 97.49%.

As we can see that from all the above models the Random Forest is giving us the Highest accuracy so we will be using this model to for predicting the 1<sup>st</sup> Innings ODI score.

## 11. Evaluate the model

**Ans:**

```
[ ] 1 from sklearn.model_selection import cross_val_score
    2 accuracies_forest = cross_val_score(estimator = forest_reg_sc, X = x_train, y = y_train, cv = 10)
    3 print('Accuracy={:f} and Standard Deviation={:f}'.format(accuracies_forest.mean()*100,accuracies_forest.std()*100))
    4 accuracies_forest
```

 Accuracy=97.025560 and Standard Deviation=0.119596  
array([0.9724282 , 0.97041491, 0.96801528, 0.9703441 , 0.97074815,  
 0.97054813, 0.96933768, 0.97029012, 0.97151 , 0.96891945])

Now we use the Cross validation to find that the accuracy we got is right or not or weather our model is overfitted or not.

As we can see that most of the accuracies are near to the 97 and giving the average accuracy of 97% so our model is well trained.

## 12. Apply prediction

Ans:

```
[ ] 1 y_pred=forest_reg_sc.predict(x_test_sc)
     2 y_pred
```

```
array([ 98.15, 220. , 278.4 , ..., 148. , 209.1 , 294. ])
```

```
[ ] 1 n=3834
     2 print(forest_reg_sc.predict(x_train_sc[[n]]))
     3 print(y_train[n])
```

```
[301.]
344
```

Use of Predict command to predict the Total score corresponding to the input data.