INT-404

Artificial Intelligence

Cricket Team Selection For Next World Cup

End Term Report

Name	Reg. No	Roll No
K. Sreesai Sameera	11802425	42
P. Sai Swetha Reddy	11802632	41
G. Ranadeep Reddy	11802499	65
P. Sai Charan	11802401	22

Section: K18HV

Department of Intelligent Systems

School of Computer Science Engineering

Lovely Professional University, Jalandhar

April-2020

STUDENT DECLARATION

This is to declare that this report has been written by us. No part of the report is copied from other sources. All information included from sources have been duly acknowledged. We aver that if any part of the report is found to be copied, we shall take full responsibility for it.

Name of student1:-

K. Sreesai Sameera

Reg. No: 11802425

Roll NO: 42

Name of Student3:-

G. Ranadeep Reddy

Reg. No: 11802499

Roll No: 65

Name of Student2:-

P. Sai Swetha Reddy

Reg. No: 11802632

Roll No: 41

Name of Student4:-

P. Sai Charan

Reg. No: 11802401

Roll No: 22

TABLE OF CONTENTS

BONAFIDE CERTIFICATE

Certified that this project report “Cricket Team Selection For Next Year” is the bonafide work of “K.Sreesai Sameera, P. Sai Swetha Reddy, G. Ranadeep Reddy, P. Sai Charan” who carried out the project the project work under my supervision.

Mr. Dipen Saini
Assistant Professor
Dept: Computer Science & Engineering
LPU
Phagwara
PUNJAB

1. <u>Background and Objective of the Project:</u>

1.1 Introduction:

We present a predictive analysis model for 2019 men's Cricket World Cup. We believe this predictive analysis strategy would be very useful for viewers, sponsors, and team strategists. This would also give insights to various cricket analysts and commentators about the features that play a crucial role in the statistical analysis. This model is developed based on the historical data collected for the 10 participating teams (Afghanistan, Australia, Bangladesh, England, India, New Zealand, Pakistan, South Africa, Sri Lanka, and West Indies). In addition, we test our model on 2015 world cup data and measure the accuracy of predictions. We are planning to expand this model as the tournament is close by and once the final squads are announced. This model is developed based on the players who were a part of their respective teams/squads in the recently 5 concluded tournaments.

1.2 Objective:

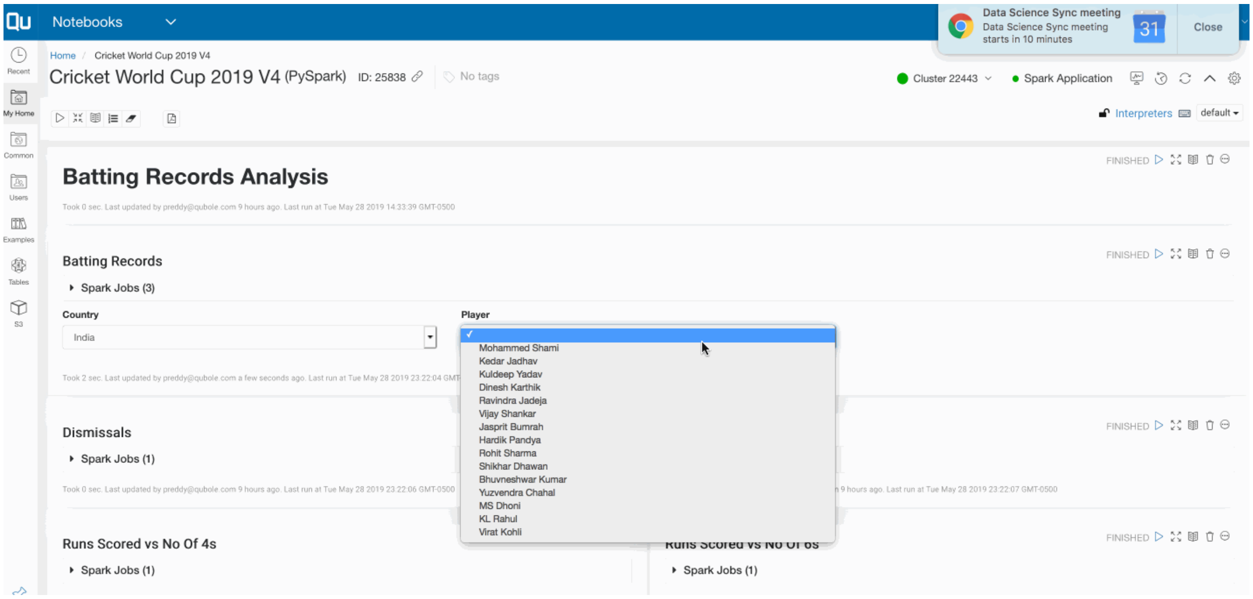
This project is useful for investors to invest in stock market based on the various factors. The project target is to create a program that analyses cricket team selection for next world cup. This also includes factors of various data descriptions.

The main feature of this project is to generate an approximate forecasting output and create a general idea of future values based on the previous data by generating a pattern. The scope of this project does not exceed more than a generalized suggestion tool.

2. Description of the Project:

2.1 Data Acquisition and Wrangling

As a first step, the goal here is to analyze player performance using an interactive web form built using Qubole Notebook's **dynamic input forms** capability with drop-downs that respond to a selection change by collecting/filtering the data associated with the selected player and refreshing the respective charts as demonstrated below,



Cricket World Cup 2019 V4 (PySpark) ID: 25838

Batting Records Analysis

Took 0 sec. Last updated by preddy@qubole.com 9 hours ago. Last run at Tue May 28 2019 14:33:39 GMT-0500

Batting Records

Spark Jobs (3)

Country: India

Player:
✓
Mohammed Shami
Kedar Jadhav
Kuldeep Yadav
Dinesh Karthik
Ravindra Jadeja
Vijay Shankar
Jasprit Bumrah
Hardik Pandya
Rohit Sharma
Shikhar Dhawan
Bhuvneshwar Kumar
Yuzvendra Chahal
MS Dhoni
KL Rahul
Virat Kohli

Took 2 sec. Last updated by preddy@qubole.com a few seconds ago. Last run at Tue May 28 2019 23:22:04 GMT

Dismissals

Spark Jobs (1)

Took 0 sec. Last updated by preddy@qubole.com 9 hours ago. Last run at Tue May 28 2019 23:22:06 GMT-0500

9 hours ago. Last run at Tue May 28 2019 23:22:07 GMT-0500

Runs Scored vs No Of 4s

Spark Jobs (1)

Runs Scored vs No Of 6s

Spark Jobs (1)

every player that has ever played international cricket is chronicled with a unique profile (player ID). for example: if you search for Virat Kohli, you will be linked to his **profile page**. The unique profile (player ID) of Virat Kohli can be inferred from the profile page as 253802. The idea here is to start from the tournament **squads page** and acquire player metadata as well as historical batting/bowler data. For this purpose, we will use a simple and elegant Python package.

The below code block demonstrates the use of Python's multiprocessing capabilities to independently and asynchronously acquire all of the player historical batting/bowling records. This fast-tracks the data acquisition and all of the historical data associated with every participating player in the ICC Cricket World Cup. On my Spark cluster master with 4 vCPU’s and 30 gigabytes of memory, the average processing time was observed to be a little over three minutes on average, which is quite impressive.

```
import requests
import sys
import re
from lxml import html
from bs4 import BeautifulSoup
import pandas as pd
from multiprocessing import Pool
from pyspark.sql.types import *
from multiprocessing.pool import ThreadPool

url="https://raw.githubusercontent.com/Pradeep39/cricket_analytics/
master/utilities/cricket_data_wrangling.py"
sc.addPyFile(url)
exec(requests.get(url).text)

batting_dfs=list()
bowling_dfs=list()

if __name__ == '__main__':
    pool=ThreadPool()
    #fetch worldcup squad list
    r =
requests.get("http://www.espncricinfo.com/ci/content/squad/index.ht
ml?object=1144415")
    soup = BeautifulSoup(r.content, "html.parser")
    for ul in soup.findAll("ul", class_="squads_list"):
     for a_team in ul.findAll("a", href=True):
      team_squad_df = getCWCTeamData(a_team.text,\
                  "http://www.espncricinfo.com"+a_team['href'])

      for index, row in team_squad_df.iterrows():
        try:
            def getBatsmanCallback(resultDF):
                if not resultDF.empty:
                    batting_dfs.append(resultDF)
            pool.apply_async(getPlayerData, args = (row,"batting",
),\
                                callback = getBatsmanCallback)

            def getBowlerCallback(resultDF):
                if not resultDF.empty:
                    bowling_dfs.append(resultDF)
            pool.apply_async(getPlayerData, args = (row, "bowling",
),\
                                callback = getBowlerCallback)

        except Exception as ex:
          print("Exception in Main:"+str(ex))
          pass
    pool.close()
    pool.join()
```

2.2 Data Cleansing/ETL

All of the independently collected data associated with each player is not clean, so let’s use the distributed data processing power of Apache Spark by combining all of the data and projecting it to Apache Spark distributed memory for doing data cleansing in a distributed manner.

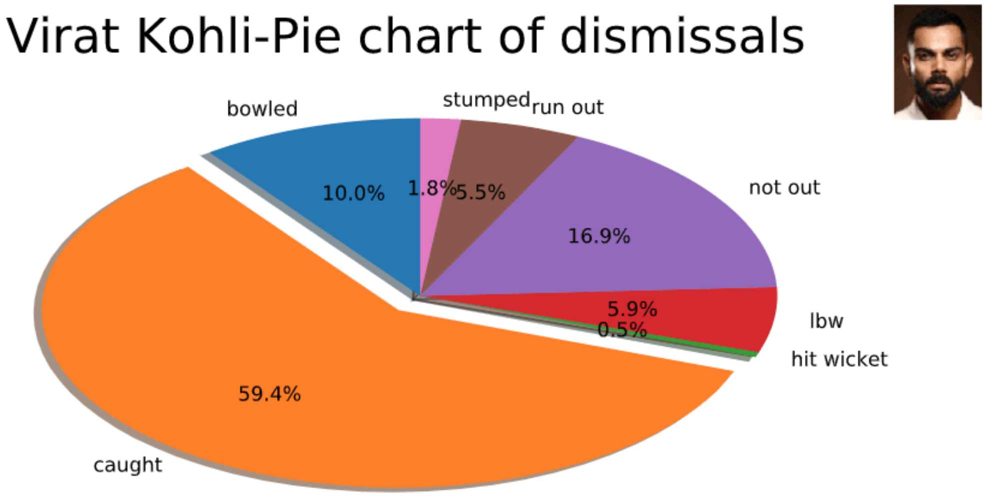
```
from pyspark.sql.types import *
from pyspark.sql.functions import *

batting_df = pd.DataFrame({})
bowling_df = pd.DataFrame({})

for df in batting_dfs:
    batting_df=batting_df.append(df)

for df in bowling_dfs:
    bowling_df=bowling_df.append(df)

batting_schema = StructType([StructField("Runs", StringType(),
True),StructField("Mins", StringType(), True),StructField("BF",
StringType(), True),StructField("4s", StringType(),
True),StructField("6s", StringType(), True),StructField("SR",
StringType(), True),StructField("Pos", StringType(),
True),StructField("Dismissal", StringType(),
True),StructField("Inns", StringType(),
True),StructField("Opposition", StringType(),
True),StructField("Ground", StringType(), True),StructField("Start
Date", StringType(), True),StructField("Country", StringType(),
True),StructField("PlayerID", StringType(),
True),StructField("PlayerName", StringType(),
True),StructField("RecordType", StringType(),
True),StructField("PlayerImg", StringType(),
True),StructField("PlayerMainRole", StringType(),
True),StructField("Age", StringType(), True),StructField("Batting",
StringType(), True),StructField("Bowling", StringType(),
True),StructField("PlayerRole", StringType(), True)])

batting_spark_df=sqlContext.createDataFrame(batting_df,schema=batti
ng_schema)

bowling_schema = StructType([ StructField("Overs", StringType(),
True),StructField("Mdns", StringType(), True),StructField("Runs",
StringType(), True),StructField("Wkts", StringType(),
True),StructField("Econ", StringType(), True),StructField("Pos",
StringType(), True),StructField("Inns", StringType(),
True),StructField("Opposition", StringType(),
True),StructField("Ground", StringType(), True),StructField("Start
Date", StringType(), True),StructField("Country", StringType(),
True),StructField("PlayerID", StringType(),
True),StructField("PlayerName", StringType(),
True),StructField("RecordType", StringType(),
True),StructField("PlayerImg", StringType(),
True),StructField("PlayerMainRole", StringType(),
True),StructField("Age", StringType(), True),StructField("Batting",
StringType(), True),StructField("Bowling", StringType(),
True),StructField("PlayerRole", StringType(), True)])

bowling_spark_df=sqlContext.createDataFrame(bowling_df,schema=bowli
ng_schema)
```

```
#distributed data cleansing operations to prepare data for analysis
using matplotlib
clean_batting_spark_df = batting_spark_df\
     .filter("Runs!= 'DNB' AND Runs!='sub' AND Runs!='absent' AND
Runs!='TDNB'")\
     .withColumn('Runs', regexp_replace('Runs', '[*]', ''))\
     .withColumn('Mins', regexp_replace('Mins', '[-]', '0'))

clean_bowling_spark_df = bowling_spark_df.filter(\
                          "Overs!= 'DNB' AND Overs!='TDNB'")
```

2.3 Batsman performance analysis

Now that we have a clean data set, any given batsman and bowler data can be filtered and collected back to the cluster master's Python process for analyzing and plotting visualizations to provide better insights on player traits and trends. The below code block filters on Virat Kohli and plots a pie chart of his dismissals.

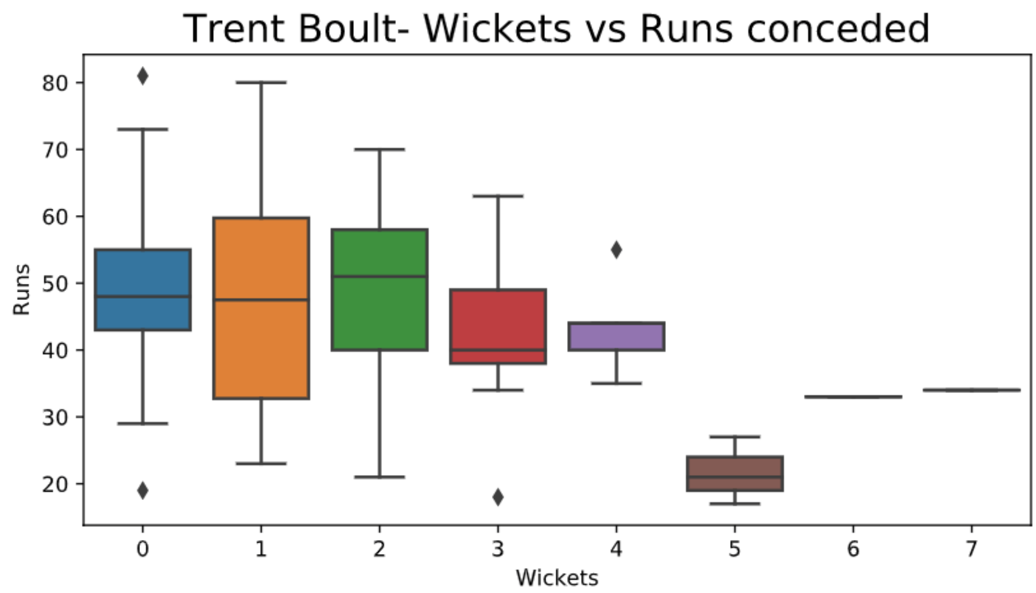
```
matplotlib.pyplot as plt
import seaborn as sns
from pylab import rcParams
from skimage import io

selectedPlayerProfile=253802

# Set figure size
rcParams['figure.figsize'] = 8,4

if selectedCountry==selectedPlayerCountry:
    batsman = clean_batting_spark_df\
             .filter(batting_spark_df.PlayerID ==
selectedPlayerProfile)\
             .toPandas()

    d = batsman['Dismissal']
    # Convert to data frame
    df = pd.DataFrame(d)
    df1=df['Dismissal'].groupby(df['Dismissal']).count()
    df2 = pd.DataFrame(df1)
    df2.columns=['Count']
    df3=df2.reset_index(inplace=False)

    image = io.imread(selectedPlayerImg)
    fig, ax = plt.subplots()
    explode = [0] * len(df3['Dismissal'])
    explode[1] = 0.1
    ax.pie(df3['Count'], explode= explode,
labels=df3['Dismissal'],autopct='%1.1f%%',shadow=True,
startangle=90)

    atitle = selectedPlayerName +  "-Pie chart of dismissals"
    plt.suptitle(atitle, fontsize=24)
    newax = fig.add_axes([0.8, 0.8, 0.2, 0.2], anchor='NE',
zorder=-1)
    newax.imshow(image)
    newax.axis('off')
    z.showplot(plt)
    plt.gcf().clear()
```



2.4 Bowler performance analysis

Let’s now focus on a bowler and analyze his historical performance. The below code block filters one of the top 10 bowlers — Trent Boult from New Zealand’s cricket team — and plots a box plot of the number of runs conceded per wickets taken.

```
import seaborn as sns
from pylab import rcParams
from skimage import io

selectedPlayerProfile=277912

bowler = clean_bowling_spark_df\
         .filter(clean_bowling_spark_df.PlayerID ==
selectedBowlerProfile)\
         .toPandas()

if not bowler.empty:
   bowler['Runs']=pd.to_numeric(bowler['Runs'])
   bowler['Wkts']=pd.to_numeric(bowler['Wkts'])
   # Set figure size
   rcParams['figure.figsize'] = 8,4
   image = io.imread(selectedBowlerImg)
   fig, ax  = plt.subplots()
   atitle = selectedBowlerName + "- Wickets vs Runs conceded"
   ax = sns.boxplot(x='Wkts', y='Runs', data=bowler)
   plt.title(atitle,fontsize=18)
   plt.xlabel('Wickets')
   newax = fig.add_axes([0.8, 0.8, 0.2, 0.2], anchor='NE', zorder=-
1)
   newax.imshow(image)
   newax.axis('off')
   z.showplot(plt)
   plt.gcf().clear()
```



2.5 Predictive Analytics: Predicting a player performance using time series analysis

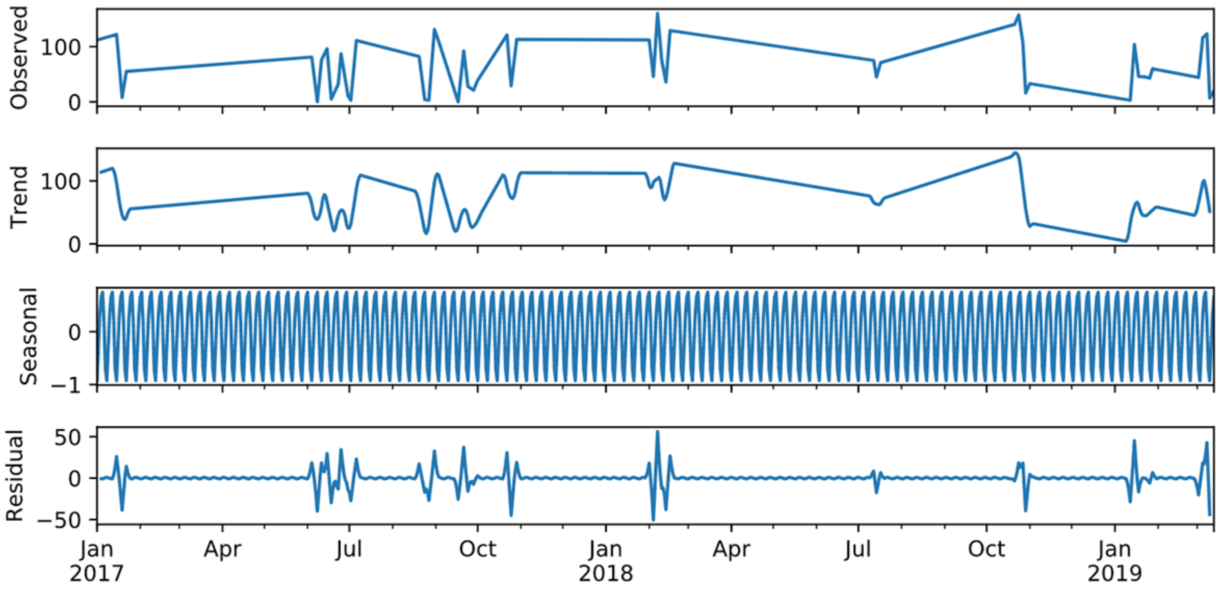
Lets dive into predicting/forecasting the future performance of a player. As a disclaimer, these predictions may be completely wrong and meant only to be referenced/used for educational purposes. Several variables impact player performance. Today we will explore a simple forecasting technique called “Time Series Analysis”. Time Series generally don’t require distributed scale processing as the series data is summarized and not of a big data class. While there are distributed time series libraries like spark-ts, they lack the breadth of what the Python Statsmodels package offers. So we shall stick with the Python Statsmodels package to complete this solution.

We will first focus on Time Series Analysis for one of the top batsmen in the world, Virat Kohli, to predict the runs he will score in the group stage of the tournament (round-robin stage). To cut to the chase, the time series model created produced the following predictions of runs scored by Virat Kohli in the group stages of the ICC Cricket World Cup:

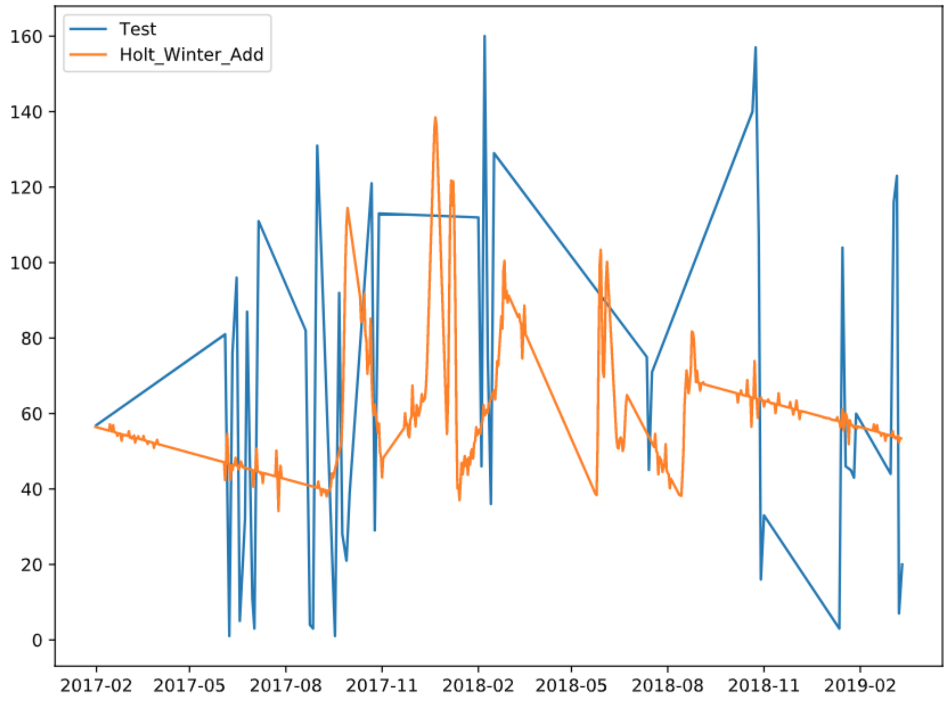
```
########Virat Kohli world cup remaining group stage match runs
predictions#######
#Bangladesh vs India on July 2nd: 67
#Srilanka vs India on July 6th: 67
#India vs Pakistan on July 9th: 67
#India vs Australia on July 14th: 66
##################################################################
```

```
idx=pd.date_range(start=virat_df.index.min(), \
              end=virat_df.index.max(), freq='D')

virat_df = pd.DataFrame(data=virat_df,index=idx, columns=['runs'])

virat_df =
virat_df.assign(RunsInterpolated=df.interpolate(method='time'))
```

```
decomposition = seasonal_decompose(\
               virat_df.loc['2017-01-
01':df.index.max()].RunsInterpolated)
decomposition.plot()
z.showplot(plt)
```



```
import matplotlib
from statsmodels.tsa.holtwinters import ExponentialSmoothing
rcParams['figure.figsize'] = 8,6
series=df.RunsInterpolated
series[series==0]=1
train_size=int(len(series)*0.8)
train_size
train=series[0:train_size]
test=series[train_size+1:]
y_hat = test.copy()
fit = ExponentialSmoothing( train ,seasonal_periods=730,\
                             trend=None, seasonal='add',).fit()
y_hat['Holt_Winter_Add']= fit.forecast(len(test))
plt.plot(test, label='Test')
plt.plot(y_hat['Holt_Winter_Add'], label='Holt_Winter_Add')
plt.legend(loc='best')
z.showplot(plt)
plt.gcf().clear()
```



2.6 Root Mean Squared Error: (RMSE)

The below block of code computes the RMSE (Root Mean Squared Error) for the fitted model. RMSE indicates the performance of the model in the test set.

```
from sklearn.metrics import mean_squared_error
from math import sqrt
rms_add = sqrt(mean_squared_error(test, y_hat.Holt_Winter_Add))
print('RMSE for Holts Winter Additive Model:%f'%rms_add)

#####Output#####
# RMSE for Holts Winter Additive Model:40.490023
```

2.7 Final forecast for 180 days beyond the last ODI

Now let’s finally forecast out for 180 days beyond the last ODI game that Virat Kohli played and print the predictions for the runs that he will score in the group stage.

```
y_hat['Holt_Winter_Add']= fit.forecast(len(test)+180)
print('########Virat Kohli world cup match by match runs prediction
1 (Holts Winter Additive)#######')
print('########Virat Kohli world cup match by match runs
predictions#######')
print('England vs India on June 30th: %d' \
                  %int(y_hat['Holt_Winter_Add'].loc['2019-06-30']))
print('Bangladesh vs India on July 2nd: %d' \
                  %int(y_hat['Holt_Winter_Add'].loc['2019-07-02']))
print('Srilanka vs India on July 6h: %d' \
                  %int(y_hat['Holt_Winter_Add'].loc['2019-07-06']))
print('############################################################
###')

########Virat Kohli world cup match by match runs
predictions#######
#Bangladesh vs India on July 2nd: 67
#Srilanka vs India on July 6th: 67
#India vs Pakistan on July 9th: 67
```

```
#India vs Australia on July 14th: 66
##################################################################

##################################################################
```

3. Work Division:

This project can’t be done by dividing the work of the project among us individually separately. So, everyone revised the every concept and methodology used/related in making this project successful.

4. Technologies and Framework used:

When it comes to the technologies and frameworks used in this project we choose to use python as programming language to program the project which is quite easy to use, portable, extensible and much efficient as of because it offers many libraries and frameworks like Tensorflow, matplotlib, pandas etc which plays a crucial role in this project.

And also we used a tool called ‘COLABORATORY’ which also called as ‘GOOGLE COLAB’ for doing this project. This thing behind using this google colab tool is that it supports GPU which is totally for free.

5. Conclusion:

This report demonstrated how to acquire data from a seemingly semi-structured source, transform/analyze player performance, and apply the time series predictive analytics technique to predict future player performance. Using these elements, we built a predictive model that forecast who will win the world cup. For all of those cricket fans who have read this article thus far, enjoy the world cup and may the best team win.