Title: G13ODI

Shubham Bhandari shubhambhandari13@gmail.com

This is project report for the subject 'CS1305: Business Intelligence' at JK Lakshmipat University, Jaipur

ABSTRACT

Abstract Content

Please note: Abbreviations should be introduced at the first mention in the main text – no abbreviations lists. Suggested structure of main text (not enforced) is provided below.

1 Introduction

2 Background

2.1 The Game of Cricket

'Cricket' also known as 'Gentlemen's game' is a bat and ball team sport played between two teams of eleven players each. The game is played on a ground at the centre of which is a 22-yard pitch with three wooden stumps at both the ends known as wickets. The wickets has also has two bails balanced on them. The game has fixed pitch size and variable ground size. One of the team act as a batting side in what is called as a inning while the other team at the same time act as the bowline or fielding side. The batting side score runs by striking the ball bowled to them using bat, and the fielding side tries to catch and stop the ball. Team scoring more runs in the specified balls wins the game. The possible scoring options in cricket include runs ranging from one to six and the possible dissmisal options include clean bowled, run out, caught, leg before wickets and stumped. The fielding side tries to restrict the batting side to minimum runs by taking wickets and exhausting the specified balls. A cluster of six such balls is known as an over. The batting side always plays in pair, in which one of the batsman known as striker stands at the further side of the pitch to the bowler and other known as non striker stand near the bowler.When ten players have been dismissed, the innings ends and the teams swap roles. The game has three umpires, two on the ground and one in a room with the ability of giving decision with the help of cameras. There is also a match refree in a cricket match.

The game has three popular formats the Twenety20, the One Day International and Test comprising of 20, 50 and unlimited overs respectiverly. Test matches are played for a duration of five days, where each team gets batting twice. The ball is a hard, solid spheroid made of compressed leather with a slightly raised sewn seam enclosing a cork core which is layered with tightly wound string. The bat is a piece of wooden crafted for hitting the ball. The size of bat has some restrictions, exact size depends on the comfort of the player.

Cricket's origins are uncertain, but the earlies reference is in South-East England in the middle of 16th century. Cricket is a highly popular sport in the British Colonies.

2.2 The Cricket Dataset

The dataset has been obtained from cricsheet.org. The data has ball by ball data of nearly 4000 matches played between 2006-2019. This data has been then processed to create scorecard file and match information file. We are using 1,788 ODI matches dataset for this project. The dataset provides various parameters for each match divided in two files. Match information file comprises of 1788 rows and 25 columns and the scorcard file has 37,963 rows and 23 columns. This dataset comprises of parameters such as match date, venue umpire, winner, winning runs/wickets etc alongwith scorecard of each match. The scorecard provides each player's performance in every match. The players who have contributed either through bowling or batting have been included in the dataset.

2.3 Cricket Betting

Cricket betting works in two ways, first one is to bet on the outcome of the match and the other one is betting on the outcome of six-overs. In the case of six-over betting, bets are placed on how many runs can be scored by a team. In betting one needs to analyse the situation and make an educated guess about the outcome and earn money from it. In a cricket match, there are several factors that are considered before betting, these factors include weather,pitch report,team combination,strengths and weakness of players,spin and pace combination,past records,player form etc. Betting is all mathematics. The ratios or odds offered depend on the situation of the match at a particular instance and the amount of money put on that team.

2.4 Statistical Models

2.5 Machine Learning Models

3 Past Work/Related work/Motivation

With the onset of the era of data and better computing systems, data from practically any field can be used for analysis and improve the performance or increase the efficiency of humans in that field. Similarly several studies have already been conducted to prove some popular hypothesis or analyse performance of the players. These analysis can actively predict the rising star in the field of cricket or can predict the future performance of a player. Such studies also act as an aid for team selection procedure by creating mathematical models for selection of players. These models can also help us in fantasy cricket. On similar lines we are trying to dismissal method in a match by away team which can again come handy in fantasy cricket and also team selection for a match.

4 Evaluation

4.1 Methodology

4.1.1 Objective of this work

The objective is to read the provided dataset and analyse the dismissal methods of the players by both the teams.

4.1.2 Followed Methodology

- Identified the objective and prepared scorecard and match information files accordingly.
- Descriptive Analysis of the thus generated data.
- Statistical Analysis of the data.
- Either accept the null hypothesis or suggest alternate hypothesis also predict the future outcome.

4.2 Descriptive Data Analysis

The dataset comprises of 1,788 ODI matches data collected from the ODI match held between 16/09/2006-11/7/2019 around the globe. The dataset describes player wise statistics of each match. This data is further accumulated to find out the method of dismissal method of the player by away team. The dataset also has each match's essential data including its venue, city, umpires, teams etc.

4.3 Statistical Modelling

4.4 Machine Learning Model

5 Result and Discussion

6 Future Work

A Information of Dataset

The dataset comprises of two files, one file comprises of each ODI match description and the other file has the scorecard of every ODI match. These matches can be uniquely identified using match id. Attributes in both the files are as follows:

- Scorecard:

    1. match-id: Unique id of each match, that can uniquely identify a match between scorecard and match information file.
    2. innings: Innings number (Can be 0 or 1)
    3. name: Name of the player
    4. batting-position: Batting position of the player (0 if the player didn't bat)
    5. over-batsman: Over at which said batsman came out to play
    6. runs-scored: Runs scored by the player
    7. balls-played: Number of balls played by the player as a batsman.
    8. dots: Number of dot balls played by the player.

9. ones: Number of balls when the player scored a single run.
10. twos: Number of balls when the player scored two runs.
11. threes: Number of balls when the player scored three runs.
12. fours: Number of balls when the player scored four runs.
13. sixes: Number of balls when the player scored six runs.
14. wicket-method: Dismissal method of the player (0 if player remained not out or didn't come out to bat)
15. balls-bowled: Number of balls that the player bowled as a bowler (0 if the player didn't bowled at all)
16. maiden-overs: Number of overs in which the player didn't give a single run as a bowler.
17. runs-given: Number of runs that the batsman scored on the said player's balls
18. wickets: Wickets taken by the player
19. extras: Extras given by the player as a bowler.
20. fall-of-wicket-score: Score at which the player got out.
21. fall-of-wicket-over: Over at which the player got out.
22. fall-of-wicket-no: Wicket number at which the player got out.
23. fall-of-wicket-bowler: Bowler who got the wicket (0 in case of run out).

- Match Information:

    1. city: City in which match was held
    2. competition: Competition name
    3. date: Date of match
    4. match-id: Unique id of each match, that can uniquely identify a match between scorecard and match information file.
    5. gender: Gender of the teams playing the match. (Either male or female)
    6. match-number: Number of the match in the respective series or competition
    7. match-referee: Match refree name
    8. method: D/L if match ended by D/L rule
    9. neutralvenue: true or false based upon the home venue of both the teams
    10. outcome: (No result or tie), if none of the two team win the match
    11. player-of-match: Name of the player of the match
    12. reserve-umpire: Name of reserved umpire of the match
    13. season: Year in which match was played
    14. series: Series name of which the match was a part
    15. team-0: First team name
    16. team-1: Second team name
    17. toss-decision: (Fielding or batting) Toss decision by toss winner team

18. toss-winner: Toss winner team name
19. tv-umpire: TV umpire name
20. umpire-0: First umpire name
21. umpire-1: Second umpire name
22. venue: Ground name where match is being held
23. winner: Winner of the match
24. winner-runs: Winner score difference
25. winner-wickets: Winner wicket difference

B Source Code of Implementation

bi_cricket (1)

November 24, 2019

[281]:
```
# from google.colab import drive
# drive.mount('/gdrive')
# %cd /gdrive
```

[282]:
```
import pandas as pd
import os
```

[283]:
```
files = [file for dirpath, directory, file in os.walk('./all_csv/')][0]
# files=[file for dirpath,directory,file in os.walk(r'/gdrive/My Drive/all_csv/
 ↪')][0]
```

[284]:
```
match_data = pd.DataFrame(data=None)
odi_scorecard = pd.DataFrame(data=None)
ttwenty_scorecard = pd.DataFrame(data=None)
odi_info = pd.DataFrame(data=None)
ttwenty_info = pd.DataFrame(data=None)
```

[285]:
```
def rename_date_umpire(index_list):
    n = 0
    for i in range(index_list.__len__()):
        if str.lower(index_list[i]).strip() == 'date':
            index_list[i] += '_'+str(n)
            n += 1
    n = 0
    for i in range(index_list.__len__()):
        if str.lower(index_list[i]).strip() == 'umpire':
            index_list[i] += '_'+str(n)
            n += 1
    n = 0
    for i in range(index_list.__len__()):
        if str.lower(index_list[i]).strip() == 'team':
            index_list[i] += '_'+str(n)
            n += 1
    return index_list
```

[286]:
```
def find_game(df_game,df_info):
    if 'series' in df_info.columns:
        if 'odi' in str.lower(df_info.iloc[0]['series']):
            return 'odi'
        if 't20i' in str.lower(df_info.iloc[0]['series']) or 't20' in str.
↪lower(df_info.iloc[0]['series']) or 'indian premier league' in str.
↪lower(df_info.iloc[0]['series'])  or 'indian premier league' in str.
↪lower(df_info.iloc[0]['competition']):
            return 'twenty'
    if max(df_game['balls-bowled'])<=24:
        return 'twenty'
    if 24<max(df_game['balls-bowled'])<=60:
        return 'odi'
    return
```

[287]:

```
def append_file(temp_df, temp_info_df, type_game):
    global odi_scorecard
    global ttwenty_scorecard
    global odi_info
    global ttwenty_info
    if type_game == 'odi':
        odi_scorecard = odi_scorecard.append(temp_df, ignore_index=True)
        odi_info = odi_info.append(temp_info_df, ignore_index=True)
    elif type_game == 'twenty':
        ttwenty_scorecard = ttwenty_scorecard.append(
            temp_df, ignore_index=True)
        ttwenty_info = ttwenty_info.append(temp_info_df, ignore_index=True)
```

[288]:

```
def get_extras_type(match_data):
    list_extras = []
    for index, row in match_data.iterrows():
        ov = str(row['over'])
        if '.' in ov:
            ov = str(row['over']).split('.')
            ball_no = int(ov[1])
            over_no = int(ov[0])
        else:
            continue
        if row['extras'] != 0:
            if row['runs'] != 0:
                match_data.loc[index, 'extras_type'] = 'w'
            list_extras.append(index)
        if ball_no > 6:
            if len(list_extras) > 0:
                match_data.loc[list_extras.pop(-1), 'extras_type'] = 'w'
    for i in list_extras:
        match_data.loc[i, 'extras_type'] = 'b'
    return match_data
```

[289]:

```
def prepare_scorecard(match_data,team_0,team_1):
    match_data = get_extras_type(match_data)
#     print(match_data[match_data['bowler']==        'Mashrafe Mortaza'])
```

```
    teams=['','']
    players = list((match_data['striker'].append(
        match_data['non-striker']).append(match_data['bowler'])).unique())
    #     to make 22 players if any player has not played
#     for i in range(len(players),22):
#         players.append('p_'+str(i))
#         fow
#    ␣
 ↪player_stats=['match-id','innings','name','batting-position','over-batsman','runs-scored','
    player_stats = ['match-id', 'team-name', 'innings', 'name',␣
 ↪'batting-position', 'over-batsman', 'runs-scored', 'balls-played', 'dots',␣
 ↪'ones', 'twos', 'threes', 'fours', 'sixes',
                    'wicket-method', 'balls-bowled', 'maiden-overs',␣
 ↪'runs-given', 'wickets', 'extras', 'fall-of-wicket-score',␣
 ↪'fall-of-wicket-overs', 'fall-of-wicket-no', 'fall-of-wicket-bowler']
    player_data = {key: {key_type: 0 for key_type in player_stats}
                   for key in players}
    for p in players:
        player_data[p]['match-id'] = match_data.loc[0, 'file_no']
        player_data[p]['name'] = p
    team_score = 0
    balls = 0
    pos = 1
    inning = False
    w = 1
    p_no = 1
    w_no = 1
    extras_over = 0
    for index, row in match_data.iterrows():
        ov = str(row['over'])
        if '.' in ov:
            ov = ov.split('.')
            ball_no = int(ov[1])
            over_no = int(ov[0])
        else:
            continue
        if over_no > 50:
            player_data = [value for key, value in player_data.items()]
            scorecard = pd.DataFrame(data=player_data)
            scorecard = scorecard[player_stats]
            return scorecard
        if ball_no == 1 and over_no == 0:
            pos = 1
            w = 1
            team_score = 0
            w_no = 1
            runs_over = 0
            if row['innings']==1 and teams[0]=='':
                teams[0]=row['batting-team']
                if teams[0]==team_0:
                    teams[1]=team_1
                elif teams[0]==team_1:
                    teams[1]=team_0
            # if row['innings']!=1:
            #     p_no=12
        if ball_no == 1:
            extras_over = 0
        if row['runs'] == 1:
            player_data[row['striker']]['ones'] += 1
        elif row['runs'] == 2:
            player_data[row['striker']]['twos'] += 1
        elif row['runs'] == 3:
            player_data[row['striker']]['threes'] += 1
        elif row['runs'] == 4:
            player_data[row['striker']]['fours'] += 1
        elif row['runs'] == 6:
            player_data[row['striker']]['sixes'] += 1
        elif row['extras'] == 0:
            player_data[row['striker']]['dots'] += 1
        if player_data[row['striker']]['batting-position'] == 0:
            player_data[row['striker']]['batting-position'] = pos
#             print(type(row['over']),type(extras_over))
#             print(row['over'])
            player_data[row['striker']
                        ]['over-batsman'] = float(row['over'])-extras_over
            pos += 1
        if player_data[row['non-striker']]['batting-position'] == 0:
            player_data[row['non-striker']]['batting-position'] = pos
            player_data[row['non-striker']
                        ]['over-batsman'] = float(row['over'])-extras_over
            pos += 1
    # wicket
#         print(row['out-player'])
        if not pd.isna(row['out-player']):
            player_data[row['out-player']]['wicket-method'] = row['out']
#             fow
#             player_data[players[p_no-1]]['fow']=w
#             player_data[players[p_no-1]]['fow_runs']=team_score
#             player_data[players[p_no-1]]['fow_overs']=row['over']
#             player_data[players[p_no-1]]['fow_batsman']=row['out-player']
#             player_data[players[p_no-1]]['fow_bowler']=row['bowler']
            p_no += 1
            w += 1
            if row['out'] != 'run out':
                player_data[row['bowler']]['wickets'] += 1
            player_data[row['out-player']]['fall-of-wicket-score'] = team_score
            player_data[row['out-player']
                        ]['fall-of-wicket-overs'] =␣
 ↪float(row['over'])-extras_over
            player_data[row['out-player']]['fall-of-wicket-no'] = w_no
            player_data[row['out-player']
                        ]['fall-of-wicket-bowler'] = row['bowler']
            w_no += 1
        team_score += row['runs']+row['extras']
        runs_over += row['runs']
        if row['extras'] != 0 and row['extras_type'] == 'w':
            runs_over += 1
            player_data[row['bowler']]['extras'] += 1
            player_data[row['striker']]['runs-scored'] += row['extras']-1
            player_data[row['bowler']]['runs-given'] += row['extras']-1
            extras_over += 0.1
        elif row['extras'] != 0:
            #             print(row)
            player_data[row['bowler']]['balls-bowled'] += 1
            player_data[row['striker']]['balls-played'] += 1
        else:
            player_data[row['striker']]['balls-played'] += 1
            player_data[row['bowler']]['balls-bowled'] += 1
            player_data[row['bowler']]['runs-given'] += row['runs']
        player_data[row['striker']]['runs-scored'] += row['runs']
        if ball_no >= 6:
            if ball_no == 6 and runs_over == 0:
                player_data[row['bowler']]['maiden-overs'] += 1
            runs_over = 0
        player_data[row['striker']]['innings'] = row['innings']
        player_data[row['striker']]['team-name']=row['batting-team']
        player_data[row['non-striker']]['team-name']=row['batting-team']
        if row['innings'] == 1:
            player_data[row['bowler']]['innings'] = 2
            player_data[row['bowler']]['team-name']=teams[1]
#             print(teams,row['bowler'])
        elif row['innings']==2:
            player_data[row['bowler']]['innings'] = 1
            player_data[row['bowler']]['team-name']=teams[0]
#             print(teams,row['bowler'])
#     print(player_data)

    player_data = [value for key, value in player_data.items()]
    scorecard = pd.DataFrame(data=player_data)
    scorecard = scorecard[player_stats]
    return scorecard
```

[290]:
```
i = 0
for file in files[:5]:
    print(i, '--', file)
    i += 1
    count = 0
    df_index = []
    df_row = []
    add = r'./all_csv/'+file
#     add=r'/gdrive/My Drive/all_csv/'+file
    df_index = ['file_no']
    file_no = file.split('.')[0]
    df_row = [file_no]
    with open(add) as f:
        new_f = f.readlines()
        for line in new_f:
            if 'version' in line:
                count += 1
            elif 'info' in line:
                count += 1
                line = line.strip().split(',')
                df_index.append(line[1])
                df_row.append(line[2])
            else:
                df_index = rename_date_umpire(df_index)
                df_dic = dict(zip(df_index, df_row), index=[0])
                temp_info_df = pd.DataFrame(df_dic)
#                 df_info=df_info.append(temp_info_df,ignore_index=True)
                # gender=df_info['gender'].iloc[0]
                # gender=str.lower(gender.strip())
                break
    temp_df = pd.read_csv(add, skiprows=count, names=[
                          0, 'innings', 'over', 'batting-team', 'striker',␣
 ↪'non-striker', 'bowler', 'runs', 'extras', 'out', 'out-player'])
    temp_df = temp_df.drop([0], axis=1)
    temp_df['file_no'] = [file_no]*(temp_df.shape[0])
    temp_sc = prepare_scorecard(temp_df,temp_info_df['team_0'].
 ↪values[0],temp_info_df['team_1'].values[0])
    # print(temp_sc)
#     append_file(temp_df,gender,type_game)
    append_file(temp_sc, temp_info_df, find_game(temp_sc, temp_info_df))
```

```
0 -- 1019975.csv
1 -- 682919.csv
2 -- 952191.csv
3 -- 1043961.csv
4 -- 565820.csv
```

[291]:

```
odi_info.columns
```

[291]:

```
Index(['city', 'competition', 'date_0', 'file_no', 'gender', 'index',
       'match_number', 'match_referee', 'player_of_match', 'reserve_umpire',
       'season', 'series', 'team_0', 'team_1', 'toss_decision', 'toss_winner',
       'tv_umpire', 'umpire_0', 'umpire_1', 'venue', 'winner', 'winner_runs'],
      dtype='object')
```

[292]:

```
ttwenty_info.columns
```

[292]:

```
Index(['city', 'competition', 'date_0', 'file_no', 'gender', 'index',
       'match_number', 'match_referee', 'neutralvenue', 'player_of_match',
       'reserve_umpire', 'season', 'series', 'team_0', 'team_1',
       'toss_decision', 'toss_winner', 'tv_umpire', 'umpire_0', 'umpire_1',
       'venue', 'winner', 'winner_runs', 'winner_wickets'],
      dtype='object')
```

[280]:

```
odi_info.to_csv('./odi_info.csv', index=False)
ttwenty_info.to_csv('./ttwenty_info.csv', index=False)
odi_scorecard.to_csv('./odi_scorecard.csv', index=False)
ttwenty_scorecard.to_csv('./ttwenty_scorecard.csv', index=False)
```

cricket-data-modification

November 24, 2019

```
[ ]: import pandas as pd
```

```
[ ]: df_info=pd.read_csv(r'./full/odi_info.csv')
```

```
[ ]: df_scorecard=pd.read_csv(r'./full/odi_scorecard.csv')
```

```
[ ]: df_scorecard = df_scorecard.astype({"match-id": str,'innings':str})
```

0.0.1 Delete Attributes

```
[ ]: df_info=df_info.drop(['eliminator','date-1','date-2','date-3','date-4'],axis=1)
```

```
[ ]: df_info=df_info.drop(['winner-innings'],axis=1)
```

```
[ ]: df_info=df_info.drop(['index'],axis=1)
```

0.0.2 Delete matches

```
[ ]: for i in df_info['match-id'].index:
         if '(' in df_info['match-id'][i]:
             print(i)
```

```
[ ]: df_info.drop(df_info[df_info['match-id']=='300438 (1)'].index,inplace=True)
     df_info.drop(df_info[df_info['match-id']=='812777 (1)'].index,inplace=True)
```

```
[ ]: df_info.drop(df_info[df_info['match-id']=='1050229'].index,inplace=True)
```

```
[ ]: df_info.drop(df_info[df_info['match-id']=='1022599'].index,inplace=True)
```

```
[ ]: df_info.drop(df_info[df_info['match-id']=='426423'].index,inplace=True)
```

```
[ ]: df_info.drop(df_info[df_info['match-id']=='915773'].index,inplace=True)
```

```
[ ]: df_scorecard.drop(df_scorecard[df_scorecard['match-id']=='1050229'].
       ↪index,inplace=True)
```

```
[ ]: df_scorecard.drop(df_scorecard[df_scorecard['match-id']=='1022599'].
       ↪index,inplace=True)
```

```
[ ]: df_scorecard.drop(df_scorecard[df_scorecard['match-id']=='426423'].
       ↪index,inplace=True)
```

[ ]:
```
df_scorecard.drop(df_scorecard[df_scorecard['match-id']=='915773'].
 ↪index,inplace=True)
```

[ ]:
```
df_scorecard.drop(df_scorecard[df_scorecard['match-id']=='300438 (1)'].
 ↪index,inplace=True)
df_scorecard.drop(df_scorecard[df_scorecard['match-id']=='812777 (1)'].
 ↪index,inplace=True)
```

[ ]:
```
match_id=df_info[(df_info['competition']=='Indian Premier League') |␣
 ↪(df_info['competition']=='ICC World Twenty20') |␣
 ↪(df_info['competition']=='Big Bash League')]['match-id']
for i in match_id:
    df_scorecard.drop(df_scorecard[df_scorecard['match-id']==i].
 ↪index,inplace=True)
    df_info.drop(df_info[df_info['match-id']==i].index,inplace=True)
```

[ ]:
```
for i in match_id:
    df_scorecard.drop(df_scorecard[df_scorecard['match-id']==i].
 ↪index,inplace=True)
    df_info.drop(df_info[df_info['match-id']==i].index,inplace=True)
```

[ ]:
```
# df_scorecard.drop(['Unnamed: 0','Unnamed: 0.1'],axis=1,inplace=True)
```

0.0.3 Additional Formatting

[ ]:
```
df_zero=df_scorecard[df_scorecard['innings']=='0']
for i, v in df_zero.iterrows():
    print(v['match-id'])
```

[ ]:
```
df_info['competition']=df_info['competition'].apply(lambda x: str(x).
 ↪replace(r'"',''))
```

[ ]:
```
df_info['venue']=df_info['venue'].apply(lambda x: str(x).replace(r'"',''))
```

[ ]:
```
df_info['year']=df_info['date'].apply(lambda x: str(x).split(r'/')[0])
```

0.0.4 Aggregate scorecard

[ ]:
```
df_scorecard_agg=df_scorecard.groupby(['match-id','team-name'],as_index=False).
 ↪sum()
```

[ ]:
```
match_id=df_scorecard_agg[df_scorecard_agg['runs-scored']==0]['match-id']
```

[ ]:
```
cancelled_matches=df_scorecard_agg[(df_scorecard_agg['runs-scored']<=50) &␣
 ↪(df_scorecard_agg['batting-position']<66) &␣
 ↪(df_scorecard_agg['balls-played']<300)]
match_id=match_id.
 ↪append(cancelled_matches[cancelled_matches['runs-scored']<cancelled_matches['runs-given']][
```

[ ]:

```
for i in match_id:
    df_scorecard.drop(df_scorecard[df_scorecard['match-id']==i].
 ↪index,inplace=True)
    df_info.drop(df_info[df_info['match-id']==i].index,inplace=True)
    df_scorecard_agg.drop(df_scorecard_agg[df_scorecard_agg['match-id']==i].
 ↪index,inplace=True)
```

0.0.5 Save files

[ ]:

```
df_info.sort_values(['match-id'],inplace=True)
df_info.to_csv(r'./full/odi_info.csv',index=False)
```

[ ]:

```
df_scorecard.sort_values(['match-id'],inplace=True)
df_scorecard.to_csv(r'./full/odi_scorecard.csv',index=False)
```

[ ]:

```
df_scorecard_agg.sort_values(['match-id'],inplace=True)
df_scorecard_agg.to_csv(r'./full/odi_scorecard_agg.csv',index=False)
```

[ ]:

The interactive visualizations cannot be shown in LaTex therfore, here is the link for all the Jupyter notebook: vis-cricket.ipynb.

vis-cricket

November 24, 2019

[3]:
```
import pandas as pd
import plotly.figure_factory as ff
import plotly.graph_objects as go
import plotly.io as pio
import math
# renderer for jupyter notebook
pio.renderers.default='notebook'
%%latex
```

```
UsageError: Line magic function `%%latex` not found.
```

[17]:
```
pio.templates.default = "plotly_dark"
```

[18]:
```
df_scorecard=pd.read_csv(r'./full/odi_scorecard.csv')
```

[19]:
```
df_info=pd.read_csv(r'./full/odi_info.csv')
```

[20]:
```
df_scorecard_agg=pd.read_csv(r'./full/odi_scorecard_agg.csv')
```

[21]:
```
df_total=pd.merge(df_info,df_scorecard,on='match-id')
```

[22]:
```
df_total_agg=pd.merge(df_info,df_scorecard_agg,on='match-id')
```

[23]:
```
df_info.columns
```

[23]:
```
Index(['city', 'competition', 'date', 'match-id', 'gender', 'match-number',
       'match-referee', 'method', 'neutralvenue', 'outcome', 'player-of-match',
       'reserve-umpire', 'season', 'series', 'team-0', 'team-1',
       'toss-decision', 'toss-winner', 'tv-umpire', 'umpire-0', 'umpire-1',
       'venue', 'winner', 'winner-runs', 'winner-wickets', 'year'],
      dtype='object')
```

[24]:
```
df_scorecard_agg.columns
```

[24]:
```
Index(['match-id', 'team-name', 'batting-position', 'over-batsman',
       'runs-scored', 'balls-played', 'dots', 'ones', 'twos', 'threes',
       'fours', 'sixes', 'balls-bowled', 'maiden-overs', 'runs-given',
       'wickets', 'extras', 'fall-of-wicket-score', 'fall-of-wicket-overs',
       'fall-of-wicket-no'],
      dtype='object')
```

[25]:
```
df_scorecard.columns
```

[25]:

```
Index(['match-id', 'team-name', 'innings', 'name', 'batting-position',
       'over-batsman', 'runs-scored', 'balls-played', 'dots', 'ones', 'twos',
       'threes', 'fours', 'sixes', 'wicket-method', 'balls-bowled',
       'maiden-overs', 'runs-given', 'wickets', 'extras',
       'fall-of-wicket-score', 'fall-of-wicket-overs', 'fall-of-wicket-no',
       'fall-of-wicket-bowler'],
      dtype='object')
```

0.1 Visualizations

0.1.1 Wickets

Wickets Methods

[175]:

```
wickets_method=df_scorecard[df_scorecard['wicket-method']!='0']['wicket-method']
```

[176]:

```
wicket_method=wickets_method.value_counts().index
wicket_method_value=wickets_method.value_counts().values
fig=go.Figure(data=[go.
 ↪Bar(x=wicket_method,y=wicket_method_value,text=wicket_method_value,textposition='auto')],la
 ↪of wickets per dismissal method")
fig.update_layout(xaxis_title='Dismissal Method',yaxis_title='Count')
fig.show()
```

Dismissal Method Distribution

[177]:

```
fig=go.Figure()
fig.add_trace(go.Pie(labels=wicket_method,values=wicket_method_value))
fig.update_layout(title='Dismissal method distribution')
fig.show()
```

Fall of Wicket by Runs

[27]:

```
fow_score=df_scorecard[(df_scorecard['fall-of-wicket-score']!
 ↪=0)]['fall-of-wicket-score'].
 ↪append(df_scorecard[(df_scorecard['fall-of-wicket-score']==0) &␣
 ↪df_scorecard['fall-of-wicket-no']!=0]['fall-of-wicket-score'])
```

[28]:

```
fig = go.Figure(data=[go.Histogram(x=fow_score)])
fig.update_layout(title='Fall of wicket by␣
 ↪runs',xaxis_title='Runs',yaxis_title='Wickets Fallen')
fig.show()
```

Probability Distribution of fall of wicket by runs

[29]:

```
fig=ff.create_distplot([fow_score],group_labels=['Fall of wicket Runs'])
fig.update_layout(title='Probability Distribution of wickets fall by␣
 ↪runs',xaxis_title='runs',yaxis_title='Probability')
fig.show()
```

Fall of Wickets by overs

[143]:
```
fow_overs=df_scorecard[df_scorecard['fall-of-wicket-overs']>0.
 ↪0]['fall-of-wicket-overs'].apply(lambda x:str(x).split('.')[0])
```

[144]:
```
fig = go.Figure(data=[go.Histogram(x=fow_overs)])
fig.show()
```

Probability distribution of Fall of wickets by overs

[145]:
```
fow_overs=fow_overs.astype('int64')
```

[146]:
```
fig=ff.create_distplot([fow_overs],group_labels=['Fall of wicket Overs'])
fig.update_layout(title='Probability Distribution of wickets fall by␣
 ↪over',xaxis_title='Over',yaxis_title='Probability')
fig.show()
```

0.1.2 Team Statistics

Teamwise runs scored

[147]:
```
team_scores={team:[] for team in df_scorecard_agg['team-name']}
for index,row in df_scorecard_agg.iterrows():
    team_scores[row['team-name']].append(row['runs-scored'])
fig=go.Figure()
for index, value in team_scores.items():
    fig.add_trace(go.Box(y=value,name=index,boxmean='sd'))
fig.update_layout(title='Teamwise Runs scored',xaxis_title='Team␣
 ↪Name',yaxis_title='Runs Scored')
fig.show()
```

Teamwise Wickets Taken

[148]:
```
team_scores={team:[] for team in df_scorecard_agg['team-name']}
for index,row in df_scorecard_agg.iterrows():
    team_scores[row['team-name']].append(row['wickets'])
fig=go.Figure()
for index, value in team_scores.items():
    fig.add_trace(go.Box(y=value,name=index,boxmean='sd'))
fig.update_layout(title='Teamwise Wickets Taken',xaxis_title='Team␣
 ↪Name',yaxis_title='Wickets Taken Scored')
fig.show()
```

Teamwise Extras Given

[14]:
```
team_scores={team:[] for team in df_scorecard_agg['team-name']}
for index,row in df_scorecard_agg.iterrows():
    team_scores[row['team-name']].append(row['extras'])
fig=go.Figure()
for index, value in team_scores.items():
    fig.add_trace(go.Box(y=value,name=index,boxmean='sd'))
fig.update_layout(title='Teamwise Extras Given',xaxis_title='Team␣
 ↪Name',yaxis_title='Wickets Taken Scored')
fig.show()
```

Team performance over the years

[20]:

```
year=[]
team=[]
matches=[]
team_year_wise_total=df_info.groupby('year')['team-0'].value_counts()
for index,value in team_year_wise_total.iteritems():
    year.append(index[0])
    team.append(index[1])
    matches.append(value)
temp1=pd.DataFrame({'year':year,'team':team,'matches0':matches})
year=[]
team=[]
matches=[]
team_year_wise_total=df_info.groupby('year')['team-1'].value_counts()
for index,value in team_year_wise_total.iteritems():
    year.append(index[0])
    team.append(index[1])
    matches.append(value)
temp2=pd.DataFrame({'year':year,'team':team,'matches1':matches})
year=[]
team=[]
wins=[]
team_year_wise_wins=df_info.groupby('year')['winner'].value_counts()
for index,value in team_year_wise_wins.iteritems():
    year.append(index[0])
    team.append(index[1])
    wins.append(value)
temp3=pd.DataFrame({'year':year,'team':team,'wins':wins})
df_matches_year=pd.merge(temp1,temp2,on=['year','team'])
df_matches_year=pd.merge(df_matches_year,temp3,on=['year','team'])
```

[12]:

```
df_matches_year['matches']=df_matches_year['matches0']+df_matches_year['matches1']
df_matches_year['win-ratio']=round(df_matches_year['wins']/
 ↪df_matches_year['matches'],3)
```

[13]:

```
df_matches_year_dict={i:{'year':[],'ratio':[]} for i in df_matches_year['team'].
 ↪unique()}
for index,row in df_matches_year.iterrows():
    df_matches_year_dict[row['team']]['year'].append(row['year'])
    df_matches_year_dict[row['team']]['ratio'].append(row['win-ratio'])
```

[14]:

```
fig=go.Figure()
for i in df_matches_year['team'].unique():
    fig.add_trace(go.
  ↪Scatter(x=df_matches_year_dict[i]['year'],y=df_matches_year_dict[i]['ratio'],name=i))
fig.update_layout(
    title_text='Win ratio of teams in ODI yearwise',
    xaxis_rangeslider_visible=True
)
fig.show()
```

Team wise performance over different venues

[76]:

```
venue=[]
team=[]
matches=[]
team_year_wise_total=df_info.groupby('venue')['team-0'].value_counts()
for index,value in team_year_wise_total.iteritems():
    venue.append(index[0])
    team.append(index[1])
    matches.append(value)
temp1=pd.DataFrame({'venue':venue,'team':team,'matches0':matches})
venue=[]
team=[]
matches=[]
team_year_wise_total=df_info.groupby('venue')['team-1'].value_counts()
for index,value in team_year_wise_total.iteritems():
    venue.append(index[0])
    team.append(index[1])
    matches.append(value)
temp2=pd.DataFrame({'venue':venue,'team':team,'matches1':matches})
venue=[]
team=[]
wins=[]
team_year_wise_wins=df_info.groupby('venue')['winner'].value_counts()
for index,value in team_year_wise_wins.iteritems():
    venue.append(index[0])
    team.append(index[1])
    wins.append(value)
temp3=pd.DataFrame({'venue':venue,'team':team,'wins':wins})
df_matches_venue=pd.merge(temp1,temp2,on=['venue','team'])
df_matches_venue=pd.merge(df_matches_venue,temp3,on=['venue','team'])
```

[77]:

```
df_matches_venue['matches']=df_matches_venue['matches0']+df_matches_venue['matches1']
df_matches_venue['win-ratio']=round(df_matches_venue['wins']/
  ↪df_matches_venue['matches'],3)
```

[78]:

```
matches_count=df_matches_venue.groupby('team',as_index=False).sum()
matches_count=matches_count.sort_values(by='matches',ascending=False)
```

```
matches_count=matches_count.iloc[:8,:]
```

[79]:

```
df_matches_venue_dict={i:{'venue':[],'ratio':[]} for i in␣
 ↪df_matches_venue['team'].unique()}
for index,row in df_matches_venue.iterrows():
    df_matches_venue_dict[row['team']]['venue'].append(row['venue'])
    df_matches_venue_dict[row['team']]['ratio'].append(row['win-ratio'])
```

[80]:

```
fig=go.Figure()
color_v=["rgb(37,102,118)", "rgb(98,240,101)", "rgb(154,72,174)",␣
 ↪"rgb(184,228,80)", "rgb(209,48,255)", "rgb(101,161,14)", "rgb(46,33,208)",␣
 ↪"rgb(241,192,57)"]
j=0
for i in matches_count['team'].unique():
#     print(i)
    fig.add_trace(go.Scatter(
        x=df_matches_venue_dict[i]['venue'],
        y=df_matches_venue_dict[i]['ratio'],
        marker=dict(color=color_v[j],size=12),
        mode='markers',
        name=i))
    j+=1
fig.update_layout(
    title_text='Win ratio of teams in ODI yearwise',
    xaxis_rangeslider_visible=True
)
fig.show()
```

0.1.3 Venue Statistics

Venue wise Runs Scored

[150]:

```
venues=df_total_agg['venue'].value_counts()
venue_scores={venue:[] for venue in venues.index[:15]}
for index,row in df_total_agg.iterrows():
#     print(venue_scores.get(row['venue'],-1),row['venue'])
    if venue_scores.get(row['venue'],-1)!=-1:
        venue_scores[row['venue']].append(row['runs-scored'])
fig=go.Figure()
for index, value in venue_scores.items():
    fig.add_trace(go.Box(y=value,name=index,boxmean='sd'))
fig.update_layout(title='Venue wise Runs scored',xaxis_title='Venue␣
 ↪Name',yaxis_title='Runs Scored')
fig.show()
```

Venue wise Wickets Fallen

[151]:
```
venues=df_total_agg['venue'].value_counts()
venue_scores={venue:[] for venue in venues.index[:15]}
for index,row in df_total_agg.iterrows():
#     print(venue_scores.get(row['venue'],-1),row['venue'])
    if venue_scores.get(row['venue'],-1)!=-1:
        venue_scores[row['venue']].append(row['wickets'])
fig=go.Figure()
for index, value in venue_scores.items():
    fig.add_trace(go.Box(y=value,name=index,boxmean='sd'))
fig.update_layout(title='Venue wise Wickets Fallen',xaxis_title='Venue␣
 ↪Name',yaxis_title='Wickets Fallen')
fig.show()
```

ODI matches distribution among grounds

[152]:
```
venues=df_total_agg['venue'].value_counts().iloc[:50]
fig=go.Figure()
fig.add_trace(go.Pie(labels=venues.index,values=venues.values))
fig.update_layout(title='ODI matches distribution among top 50 grounds'
                 ,margin=dict(l=20,r=20,t=40,b=20),
                 autosize=False,
                 width=1000,
                 height=1000)
fig.show()
```

0.1.4 Player Statistics

Matches distibution between genders

[153]:
```
gender=df_info['gender'].value_counts()
fig=go.Figure()
fig.add_trace(go.Pie(labels=gender.index.unique(),values=gender.values))
fig.update_layout(title='Gender wise matches distribution')
fig.show()
```

Top 10 Batsmen

[109]:
```
df_scorecard_batsman_agg=df_scorecard.groupby(['name'],as_index=False).sum()
df_scorecard_batsman_agg=df_scorecard_batsman_agg.
 ↪sort_values(by=['runs-scored'],ascending=False)
df_scorecard_batsman_agg=df_scorecard_batsman_agg.reset_index(drop=True)
```

[110]:
```
df_scorecard_batsman_agg=df_scorecard_batsman_agg.drop(['batting-position',
       'over-batsman','extras',
       'fall-of-wicket-score', 'fall-of-wicket-overs',␣
 ↪'fall-of-wicket-no'],axis=1)
```

[111]:
```
batsman_innings=df_scorecard[df_scorecard['over-batsman']>0.0]['name']
batsman_innings=batsman_innings.value_counts().to_dict()
df_scorecard_batsman_agg['innings']=df_scorecard_batsman_agg['name'].
 ↪map(batsman_innings)
df_scorecard_batsman_agg['strike-rate']=df_scorecard_batsman_agg.apply(lambda␣
 ↪row: round((row['runs-scored']/row['balls-played'])*100,3) if␣
 ↪row['balls-played']>0 else 0 ,axis=1)
df_scorecard_batsman_agg['avg']=df_scorecard_batsman_agg.apply(lambda row:␣
 ↪round(row['runs-scored']/row['innings'],3) if row['innings']>0 else 0,axis=1)
```

[112]:
```
df_scorecard_batsman_agg_sub=df_scorecard_batsman_agg.iloc[:10]
fig=go.Figure()
fig.add_trace(go.Table(
    header=dict(
        values=['Batsman Name','Innings','Runs Scored','Balls␣
 ↪Played','Fours','Sixes','Batting Strike Rate','Batting Average'],
        fill_color='paleturquoise',
        align='left',
        font=dict(color='black',size=14)
    ),
    cells=dict(values=
                 ␣
 ↪[df_scorecard_batsman_agg_sub['name'],df_scorecard_batsman_agg_sub['innings'],df_scorecard_
                  ␣
 ↪df_scorecard_batsman_agg_sub['balls-played'],df_scorecard_batsman_agg_sub['fours'],df_score
                 ␣
 ↪df_scorecard_batsman_agg_sub['strike-rate'],df_scorecard_batsman_agg_sub['avg']],
               align='left'
    )
))
fig.update_layout(title='Top 10 Batsmen')
fig.show()
```

Batsmen Performance

[113]:
```
df_scorecard_batsman_agg_sub=df_scorecard_batsman_agg.iloc[:]
hover_text=[]
bubble_size=[]
for index,row in df_scorecard_batsman_agg_sub.iterrows():
    hover_text.append(
        ('Name: {name}<br>'+'Balls Played: {balls}<br>'+'Average:␣
 ↪{avg}<br>'+'Strike Rate: {strike}<br>').format(
            ␣
 ↪name=row['name'],avg=row['avg'],strike=row['strike-rate'],balls=row['balls-played']))
    bubble_size.append(math.sqrt(row['strike-rate'])*2)
fig=go.Figure()
fig.add_trace(
    go.Scatter(
    x=df_scorecard_batsman_agg_sub['balls-played'],
    y=df_scorecard_batsman_agg_sub['runs-scored'],
    text=hover_text,
        mode='markers',
        marker=dict(
        color=df_scorecard_batsman_agg_sub['innings'],
        colorbar=dict(
        title='Innings Played'
        ),
            colorscale='Viridis',
        size=bubble_size,
            showscale=True
        )
    )
)
fig.update_layout(title='Batsmen Performance',xaxis_title='Balls␣
 ↪Played',yaxis_title='Runs scored')
fig.show()
```

Top 10 Bowlers

[158]:

```
df_scorecard_bowler_agg=df_scorecard.groupby(['name'],as_index=False).sum()
df_scorecard_bowler_agg=df_scorecard_bowler_agg.
 ↪sort_values(by=['wickets'],ascending=False)
df_scorecard_bowler_agg=df_scorecard_bowler_agg.reset_index(drop=True)
```

[159]:

```
df_scorecard_bowler_agg=df_scorecard_bowler_agg.drop(['batting-position',␣
 ↪'over-batsman', 'runs-scored',
       'balls-played', 'dots', 'ones', 'twos', 'threes', 'fours', 'sixes',␣
 ↪'extras',
       'fall-of-wicket-score', 'fall-of-wicket-overs',␣
 ↪'fall-of-wicket-no'],axis=1)
```

[160]:

```
bowler_innings=df_scorecard[df_scorecard['balls-bowled']>0]['name']
bowler_innings=bowler_innings.value_counts().to_dict()
df_scorecard_bowler_agg['innings']=df_scorecard_bowler_agg['name'].
 ↪map(bowler_innings)
df_scorecard_bowler_agg['strike-rate']=df_scorecard_bowler_agg.apply(lambda row:
 ↪ round((row['balls-bowled']/row['wickets']),3) if row['wickets']>0 else 0␣
 ↪,axis=1)
df_scorecard_bowler_agg['avg']=df_scorecard_bowler_agg.apply(lambda row:␣
 ↪round(row['runs-given']/row['wickets'],3) if row['wickets']>0 else 0,axis=1)
df_scorecard_bowler_agg['eco']=df_scorecard_bowler_agg.apply(lambda row:␣
 ↪round(row['runs-given']/(row['balls-bowled']/6),3) if row['balls-bowled']>0␣
 ↪else 0,axis=1)
```

```
[161]: df_scorecard_bowler_agg_sub=df_scorecard_bowler_agg.iloc[:10]
fig=go.Figure()
fig.add_trace(go.Table(
    header=dict(
        values=['Bowler Name','Innings','Balls','Maiden Overs','Runs␣
 ↪Conceded','Wickets','Economy','Bowling Strike Rate','Bowling Average'],
        fill_color='paleturquoise',
        align='left',
        font=dict(color='black',size=14)
    ),
    cells=dict(values=
                ␣
 ↪[df_scorecard_bowler_agg_sub['name'],df_scorecard_bowler_agg_sub['innings'],df_scorecard_bo
                 ␣
 ↪df_scorecard_bowler_agg_sub['maiden-overs'],df_scorecard_bowler_agg_sub['runs-given'],df_sc
                df_scorecard_bowler_agg_sub['eco'],
                ␣
 ↪df_scorecard_bowler_agg_sub['strike-rate'],df_scorecard_bowler_agg_sub['avg']],
               align='left'
    )
))
fig.update_layout(title='Top 10 Bowlers')
fig.show()
```

Bowlers Performance

```
[162]: df_scorecard_bowler_agg_sub=df_scorecard_bowler_agg.iloc[:]
hover_text=[]
bubble_size=[]

for index,row in df_scorecard_bowler_agg_sub.iterrows():
    hover_text.append(
        ('Name: {name}<br>'+'Economy: {eco}<br>'+'Average: {avg}<br>'+'Strike␣
 ↪Rate: {strike}<br>').format(
           ␣
 ↪name=row['name'],avg=row['avg'],strike=row['strike-rate'],eco=row['eco']))
    bubble_size.append(math.sqrt(row['eco'])*6)
fig=go.Figure()
fig.add_trace(
    go.Scatter(
    x=df_scorecard_batsman_agg_sub['balls-bowled'],
    y=df_scorecard_batsman_agg_sub['wickets'],
    text=hover_text,
        mode='markers',
        marker=dict(
        color=df_scorecard_batsman_agg_sub['innings'],
        colorbar=dict(
        title='Innings Played'
        ),
            colorscale='Viridis',
        size=bubble_size,
            showscale=True
        )
    )
)
fig.update_layout(title='Bowlers Performance',xaxis_title='Balls Bowled',yaxis_title='Wickets')
fig.show()
```

Batsmen performance over the years: Runs scored

[183]:

```
df_player_year=df_total.groupby(by=['year','name'],as_index=False).sum()
```

[164]:

```
top_batsmen=df_scorecard_batsman_agg.iloc[:10]['name'].tolist()
df_batsmen_year=df_player_year[df_player_year['name'].isin(top_batsmen)]
df_batsman_year_grouped=df_batsmen_year.groupby('name')
```

[151]:

```
fig=go.Figure()
for name,group in df_batsman_year_grouped:
    fig.add_trace(go.Scatter(x=group['year'],y=group['runs-scored'],name=name))
fig.update_layout(
    title_text='Runs in a calendar year of top 10 ODI batsmen',
    xaxis_rangeslider_visible=True,
    xaxis_title='Year',
    yaxis_title='Runs'
)
fig.show()
```

Batsmen performance over the years: Strike rate

[152]:

```
df_batsmen_year['strike-rate']=round(df_batsmen_year['runs-scored']/df_batsmen_year['balls-played'],4)*100
df_batsman_year_grouped=df_batsmen_year.groupby('name')
```

[153]:

```
fig=go.Figure()
for name,group in df_batsman_year_grouped:
    fig.add_trace(go.Scatter(x=group['year'],y=group['strike-rate'],name=name))
fig.update_layout(
    title_text='Strike rate of top 10 batsmen in ODI yearwise',
    xaxis_rangeslider_visible=True,
    xaxis_title='Year',
    yaxis_title='Strike Rate'
)
fig.show()
```

Bowlers performance over the years: Wickets taken

[171]:
```
top_bowlers=df_scorecard_bowler_agg.iloc[:10]['name'].tolist()
df_bowlers_year=df_player_year[df_player_year['name'].isin(top_bowlers)]
df_bowlers_year_grouped=df_bowlers_year.groupby('name')
```

[172]:
```
fig=go.Figure()
for name,group in df_bowlers_year_grouped:
    fig.add_trace(go.Scatter(x=group['year'],y=group['wickets'],name=name))
fig.update_layout(
    title_text='Runs in a calendar year of top 10 ODI bowlers',
    xaxis_rangeslider_visible=True,
    xaxis_title='Year',
    yaxis_title='Wickets'
)
fig.show()
```

Bowlers performance over the years: Economy

[181]:
```
df_bowlers_year['economy']=round(df_bowlers_year['runs-given']/
 ↪(df_bowlers_year['balls-bowled']/6),2)
df_bowlers_year_grouped=df_bowlers_year.groupby('name')
```

[182]:
```
fig=go.Figure()
for name,group in df_bowlers_year_grouped:
    fig.add_trace(go.Scatter(x=group['year'],y=group['economy'],name=name))
fig.update_layout(
    title_text='Economy of top 10 bowlers in ODI yearwise',
    xaxis_rangeslider_visible=True,
    xaxis_title='Year',
    yaxis_title='Strike Rate'
)
fig.show()
```

The interactive visualizations cannot be shown in LaTex therfore, here is the link for all the Jupyter notebook: hypothesis.ipynb.

hypothesis

November 24, 2019

```
[ ]: import pandas as pd
     import plotly.figure_factory as ff
     import plotly.graph_objects as go
     import plotly.io as pio
     import math
     from scipy import stats
     # renderer for jupyter notebook
     from sklearn.metrics import mean_absolute_error
     pio.renderers.default='notebook'
     from sklearn.model_selection import train_test_split
     from sklearn.linear_model import LinearRegression
```

```
[ ]: pio.templates.default = "plotly_dark"
```

```
[ ]: df_scorecard=pd.read_csv(r'./full/odi_scorecard.csv')
     df_info=pd.read_csv(r'./full/odi_info.csv')
```

0.0.1 Hypothesis

1

- H(0):Mean value of batsman bowled is equal to mean value of batsman dismissed by lbw in ODI
- H(A):Mean value of batsman bowled is not equal to mean value batsman dismissed by lbw

Data

```
[ ]: df_first=df_scorecard[(df_scorecard['wicket-method']=='bowled')|(df_scorecard['wicket-method']
```

```
[ ]: df_first['lbw']=df_first['wicket-method'].apply(lambda x: 0 if x=='bowled' else␣
       ↪1 )
     df_first['bowled']=df_first['wicket-method'].apply(lambda x: 1 if x=='bowled'␣
       ↪else 0 )
```

```
[ ]: df_first=df_first.groupby(['match-id'],as_index=False).sum()
```

```
[ ]: df_first=df_first[['lbw','bowled']]
```

```
[ ]: df_first
```

Visualizations

```
[ ]: fig = go.Figure()
     fig.add_trace(go.Histogram(x=df_first['lbw'], histnorm='probability',␣
      ↪name='lbw'))
     fig.add_trace(go.Histogram(x=df_first['bowled'],␣
      ↪histnorm='probability',name='bowled'))
     fig.update_layout(title='Probability distribution for lbw and␣
      ↪bowled',xaxis_title='Number of wickets',yaxis_title='Probability')
     fig.show()
```

```
[ ]: fig=ff.
      ↪create_distplot([df_first['lbw'],df_first['bowled']],['LBW','Bowled'],bin_size=1,curve_type
     fig.update_layout(title_text='Distribution of dimissal␣
      ↪methods',xaxis_title='Number of wickets',yaxis_title='Density')
     fig.show()
```

Hypothesis Testing

Paired T test

```
[ ]: df_first[['lbw','bowled']].describe()
```

```
[ ]: ttest,pval=stats.ttest_rel(df_first['lbw'],df_first['bowled'])
```

```
[ ]: print(pval)
     if pval<0.05:
         print("reject null hypothesis")
     else:
         print("accept null hypothesis")
```

2

- H(0):Wickets fallen in the first 70% of the first innings is equal to the wickets fallen in the last 30% of the first innings
- H(A):Wickets fallen in the first 70% of the first innings is not equal to the wickets fallen in the last 30% of the first innings
- H(0):Wickets fallen in the first 71% of the first innings is equal to the wickets fallen in the last 29% of the first innings
- H(A):Wickets fallen in the first 71% of the first innings is not equal to the wickets fallen in the last 29% of the first innings
- H(0):Wickets fallen in the first 50% of the first innings is equal to the wickets fallen in the last 50% of the first innings
- H(A):Wickets fallen in the first 50% of the first innings is not equal to the wickets fallen in the last 50% of the first innings
- H(0):Wickets fallen in the first 10 overs of the first innings is equal to the wickets fallen in the last 10 overs of the first innings
- H(A):Wickets fallen in the first 10 overs of the first innings is not equal to the wickets fallen in the last 10 overs of the first innings

Data

[ ]:

```
temp=pd.DataFrame(data=None)
temp=df_scorecard[df_scorecard['innings']==1]
temp=temp.groupby('match-id',as_index=False).sum()
temp['total-overs']=round(temp['balls-played']/6)
temp=temp[['match-id','total-overs']]
temp['first-seventy']=round(temp['total-overs']*0.70)
temp['first-seventyone']=round(temp['total-overs']*0.71)
temp['first-fifty']=round(temp['total-overs']*0.5)
temp['last-ten']=round(temp['total-overs']-10)
temp=temp.merge(df_scorecard[(df_scorecard['innings']==1)␣
 ↪&(df_scorecard['fall-of-wicket-overs']>0.0)],on=['match-id'])
temp['fall-of-wicket-overs']=temp['fall-of-wicket-overs'].apply(lambda x:␣
 ↪int(x)+1)
```

[ ]:

```
df_second=pd.DataFrame({'match-id':temp['match-id']})
df_second['first-seventy-wickets']=temp[(temp['fall-of-wicket-overs']<=temp['first-seventy'])␣
 ↪& (temp['fall-of-wicket-overs']>0)]['fall-of-wicket-overs']
df_second['last-thirty-wickets']=temp[temp['fall-of-wicket-overs']>temp['first-seventy']]['fal
df_second['first-seventy-wickets']=df_second['first-seventy-wickets'].
 ↪apply(lambda x:1 if x>0 else 0)
df_second['last-thirty-wickets']=df_second['last-thirty-wickets'].apply(lambda␣
 ↪x:1 if x>0 else 0)

df_second['first-seventyone-wickets']=temp[(temp['fall-of-wicket-overs']<=temp['first-seventyo
 ↪& (temp['fall-of-wicket-overs']>0)]['fall-of-wicket-overs']
df_second['last-twentynine-wickets']=temp[temp['fall-of-wicket-overs']>temp['first-seventyone'
df_second['first-seventyone-wickets']=df_second['first-seventyone-wickets'].
 ↪apply(lambda x:1 if x>0 else 0)
df_second['last-twentynine-wickets']=df_second['last-twentynine-wickets'].
 ↪apply(lambda x:1 if x>0 else 0)

df_second['first-fifty-wickets']=temp[(temp['fall-of-wicket-overs']<=temp['first-fifty'])␣
 ↪& (temp['fall-of-wicket-overs']>0)]['fall-of-wicket-overs']
df_second['last-fifty-wickets']=temp[temp['fall-of-wicket-overs']>temp['first-fifty']]['fall-o
df_second['first-fifty-wickets']=df_second['first-fifty-wickets'].apply(lambda␣
 ↪x:1 if x>0 else 0)
df_second['last-fifty-wickets']=df_second['last-fifty-wickets'].apply(lambda x:
 ↪1 if x>0 else 0)

df_second['first-ten-overs-wickets']=temp[(temp['fall-of-wicket-overs']<=10) &␣
 ↪(temp['fall-of-wicket-overs']>0)]['fall-of-wicket-overs']
df_second['last-ten-overs-wickets']=temp[temp['fall-of-wicket-overs']>temp['last-ten']]['fall-
df_second['first-ten-overs-wickets']=df_second['first-ten-overs-wickets'].
 ↪apply(lambda x:1 if x>0 else 0)
df_second['last-ten-overs-wickets']=df_second['last-ten-overs-wickets'].
 ↪apply(lambda x:1 if x>0 else 0)
```

```
df_second=df_second.groupby(['match-id'],as_index=False).sum()
df_second=df_second.drop(['match-id'],axis=1)
```

[ ]:
```
df_second
```

Visualizations

[ ]:
```
fig = go.Figure()
fig.add_trace(go.Histogram(x=df_second['first-seventy-wickets'],␣
 ↪histnorm='probability', name='First 70%'))
fig.add_trace(go.Histogram(x=df_second['last-thirty-wickets'],␣
 ↪histnorm='probability',name='Last 30%'))
fig.update_layout(title='Probability distribution for wickets fallen in first␣
 ↪70% and last 30% of first innings',xaxis_title='Number of␣
 ↪wickets',yaxis_title='Probability')
fig.show()
```

[ ]:
```
fig = go.Figure()
fig.add_trace(go.Histogram(x=df_second['first-seventyone-wickets'],␣
 ↪histnorm='probability', name='First 71%'))
fig.add_trace(go.Histogram(x=df_second['last-twentynine-wickets'],␣
 ↪histnorm='probability',name='Last 29%'))
fig.update_layout(title='Probability distribution for wickets fallen in first␣
 ↪71% and last 29% of first innings',xaxis_title='Number of␣
 ↪wickets',yaxis_title='Probability')
fig.show()
```

[ ]:
```
fig = go.Figure()
fig.add_trace(go.Histogram(x=df_second['first-fifty-wickets'],␣
 ↪histnorm='probability', name='First 50%'))
fig.add_trace(go.Histogram(x=df_second['last-fifty-wickets'],␣
 ↪histnorm='probability',name='Last 50%'))
fig.update_layout(title='Probability distribution for wickets fallen in first␣
 ↪50% and last 50% of first innings',xaxis_title='Number of␣
 ↪wickets',yaxis_title='Probability')
fig.show()
```

[ ]:
```
fig = go.Figure()
fig.add_trace(go.Histogram(x=df_second['first-ten-overs-wickets'],␣
 ↪histnorm='probability', name='First 10'))
fig.add_trace(go.Histogram(x=df_second['last-ten-overs-wickets'],␣
 ↪histnorm='probability',name='Last 10'))
fig.update_layout(title='Probability distribution for wickets fallen in first␣
 ↪10 overs and last 10 oversof first innings',xaxis_title='Number of␣
 ↪wickets',yaxis_title='Probability')
fig.show()
```

[ ]:
```
fig=ff.
 ↪create_distplot([df_second['first-seventy-wickets'],df_second['last-thirty-wickets']],['Fir
 ↪70%','Last 30%'],curve_type='normal')
fig.update_layout(title='Distribution for wickets fallen in first 70% and last␣
 ↪30% of first innings',xaxis_title='Number of wickets',yaxis_title='Density')
fig.show()
```

[ ]:
```
fig=ff.
 ↪create_distplot([df_second['first-fifty-wickets'],df_second['last-fifty-wickets']],['First␣
 ↪50%','Last 50%'],curve_type='normal')
fig.update_layout(title='Distribution for wickets fallen in first 50% and last␣
 ↪50% of first innings',xaxis_title='Number of wickets',yaxis_title='Density')
fig.show()
```

[ ]:
```
fig=ff.
 ↪create_distplot([df_second['first-ten-overs-wickets'],df_second['last-ten-overs-wickets']],
 ↪10 overs','Last 10 overs'],curve_type='normal')
fig.update_layout(title='Distribution for wickets fallen in first 10 overs and␣
 ↪last 10 overs of first innings',xaxis_title='Number of␣
 ↪wickets',yaxis_title='Density')
fig.show()
```

Hypothesis Testing

Paired T Test

[ ]:
```
df_second.describe()
```

[ ]:
```
ttest,pval=stats.
 ↪ttest_rel(df_second['first-seventy-wickets'],df_second['last-thirty-wickets'])
```

[ ]:
```
print(pval)
if pval<0.05:
    print("reject null hypothesis")
else:
    print("accept null hypothesis")
```

[ ]:
```
ttest,pval=stats.
 ↪ttest_rel(df_second['first-seventyone-wickets'],df_second['last-twentynine-wickets'])
```

[ ]:
```
print(pval)
if pval<0.05:
    print("reject null hypothesis")
else:
    print("accept null hypothesis")
```

[ ]:
```
ttest,pval=stats.
 ↪ttest_rel(df_second['first-fifty-wickets'],df_second['last-fifty-wickets'])
```

[ ]:
```
print(pval)
if pval<0.05:
    print("reject null hypothesis")
else:
    print("accept null hypothesis")
```

[ ]:
```
ttest,pval=stats.
 ↪ttest_rel(df_second['first-ten-overs-wickets'],df_second['last-ten-overs-wickets'])
```

[ ]:
```
print(pval)
if pval<0.05:
    print("reject null hypothesis")
else:
    print("accept null hypothesis")
```

3

- H(0):There is an equal probability of wicket by the first category of dismissal and second category of dismissal
- H(A):There is an equal probability of wicket by the first category of dismissal and second category of dismissal

Data

[ ]:
```
df_third=df_scorecard[df_scorecard['wicket-method']!='0']
```

[ ]:
```
first_cat=['run out','hit wicket','obstructing the field','retired␣
 ↪out','stumped']
second_cat=['caught','bowled','lbw','caught and bowled']
```

[ ]:
```
df_third['first-category']=df_third['wicket-method'].apply(lambda x: 1 if x in␣
 ↪first_cat else 0 )
df_third['sec-category']=df_third['wicket-method'].apply(lambda x: 1 if x in␣
 ↪second_cat else 0 )
```

[ ]:
```
df_third=df_third[['match-id','first-category','sec-category']]
df_third=df_third.groupby(['match-id'],as_index=False).sum()
df_third['wickets']=df_third['first-category']+df_third['sec-category']
```

[ ]:
```
# df_third.loc[:,'wic_batsman']=round(df_third['wic_batsman']/
 ↪df_third['wickets'],3)
# df_third.loc[:,'wic_bowler']=round(df_third['wic_bowler']/
 ↪df_third['wickets'],3)
df_third=df_third[['first-category','sec-category']]
```

[ ]:
```
df_third
```

Visualizations

[ ]:
```
fig = go.Figure()
```

```
fig.add_trace(go.Histogram(x=df_third['first-category'],␣
 ↪histnorm='probability', name='First Category'))
fig.add_trace(go.Histogram(x=df_third['sec-category'],␣
 ↪histnorm='probability',name='Second Category'))
fig.update_layout(title='Probability distribution for wickets fallen by First␣
 ↪category of dismissal methods and second category',xaxis_title='Number of␣
 ↪wickets',yaxis_title='Probability')
fig.show()
```

[ ]:
```
fig=ff.
 ↪create_distplot([df_third['first-category'],df_third['sec-category']],['First␣
 ↪Category','Second Category'],curve_type='normal')
fig.update_layout(title='Density of wickets fallen by first and second category␣
 ↪of dismissal methods',xaxis_title='Number of wickets',yaxis_title='Density')
fig.show()
```

Hypothesis Testing

Paired T Test

[ ]:
```
df_third.describe()
```

[ ]:
```
ttest,pval=stats.ttest_rel(df_third['first-category'],df_third['sec-category'])
```

[ ]:
```
print(pval)
if pval<0.05:
    print("reject null hypothesis")
else:
    print("accept null hypothesis")
```

Current

[ ]:
```
# df_sample_mean_bats=[]
# df_sample_mean_bowler=[]
# for i in range(1000):
#     df_sample=df_third.sample(n=100)
#     mean_v=df_sample.mean(axis=0)
#     df_sample_mean_bats.append(mean_v[0])
#     df_sample_mean_bowler.append(mean_v[1])
# fig=ff.
 ↪create_distplot([df_sample_mean_bats,df_sample_mean_bowler],['Batsman','Bowler'])
# fig.show()
```

0.0.2 ML

Correlation

[ ]:
```
df_kohli=df_scorecard[df_scorecard['name']=='V Kohli']
```

[ ]:

```
corr_val=df_kohli.drop(['match-id'],axis=1).corr()
corr_list=[]
for i in range(corr_val.shape[0]):
    corr_list.append(corr_val.iloc[:,i])
fig = go.Figure(data=go.Heatmap(
                   z=corr_list,
                   x=corr_val.columns,
                   y=corr_val.columns))
fig.show()
```

[ ]:

```
columns = np.full((corr_val.shape[0],), True, dtype=bool)
for i in range(corr_val.shape[0]):
    for j in range(i+1, corr_val.shape[0]):
        if corr_val.iloc[i,j] >= 0.9:
            if columns[j]:
                columns[j] = False
```

[ ]:

```
selected_columns = corr_val.columns[columns]
data = df_scorecard[selected_columns]
```

[ ]:

```
data.sort_values(by='innings')
```

[ ]:

```
x = df_kohli.loc[:,'balls-played'].values
y = df_kohli.loc[:, 'runs-scored'].values
```

[ ]:

```
xTrain, xTest, yTrain, yTest = train_test_split(x, y, test_size = 1/3,␣
 ↪random_state = 0)
```

[ ]:

```
linearRegressor = LinearRegression()
```

[ ]:

```
yTrain = yTrain.reshape(1, -1)
xTrain = xTrain.reshape(1, -1)
yTest = yTest.reshape(1, -1)
xTest = xTest.reshape(1, -1)
```

[ ]:

```
linearRegressor.fit(xTrain, yTrain)
```

[ ]:

```
print(xTest.shape)
print(yTest.shape)
print(xTrain.shape)
print(yTrain.shape)
```

[ ]:

```
yPrediction = linearRegressor.predict(xTest)
```

[ ]:

```
```