Data Science Exploration 3 - Continuous random variable

November 16, 2022

0.1 ECE 131A Data science exploration 3: Please complete this jupyter notebook by filling out the code blocks. Once you have completed the notebook, generate a PDF of the completed notebook and upload the PDF to Gradescope by 11:59 PM on 11/15/2022.

0.2 In this data science exploration, we will be performing basic statistical analysis on a real world datasets. The analysis will consist of plotting the empirical distribution of the quantities of interest in the dataset and visualizing how well the empirical distribution matches with continuous probability distributions learned in the class.

[ ]:
```
## Importing the necessary packages

import math
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy import stats
```

0.3 Indian premier league match dataset: data on matches played in the Indian premier league from 2008-2018. The data has many attributes but in this assignment we will only focus on some of the attributes

[ ]:
```
## Loading the dataset as a pandas dataframe and printing the header
matches = pd.read_csv('matches.csv')
matches.head()
```

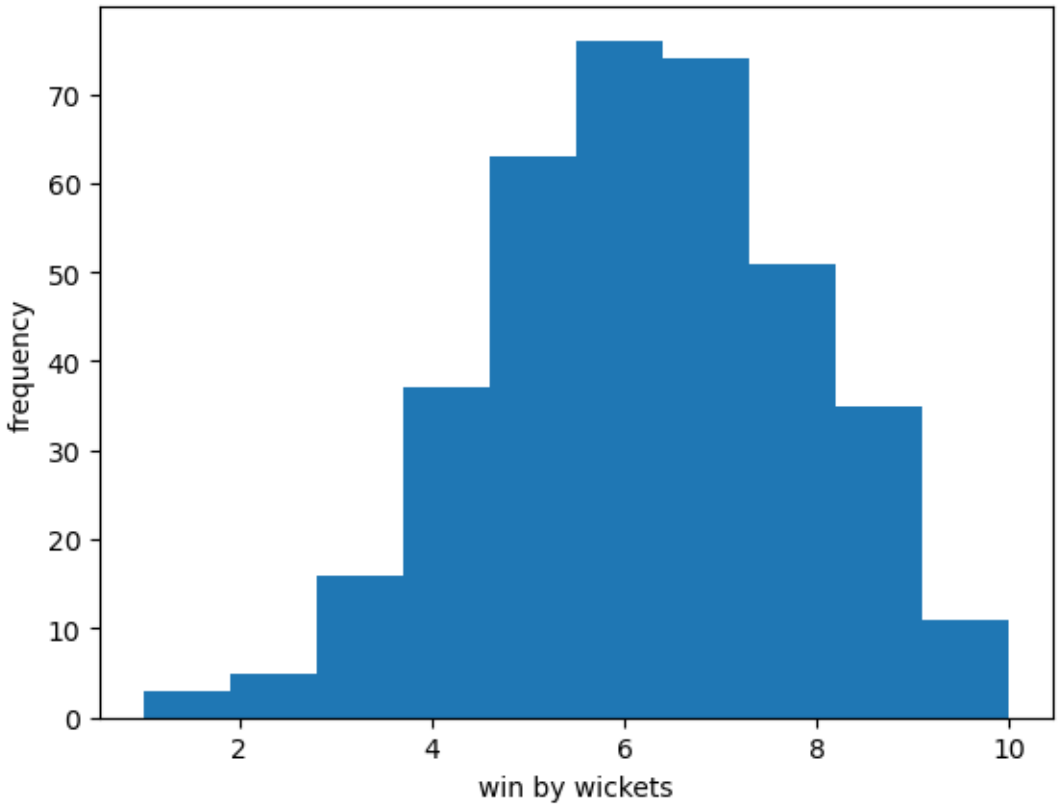
[ ]:

	id	season	city	date	team1	team2	toss_winner	toss_decision
0	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field
1	2	2017	Pune	2017-04-06	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field
2	3	2017	Rajkot	2017-04-07	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field
3	4	2017	Indore	2017-04-08	Rising Pune Supergiant			
4	5	2017	Bangalore	2017-04-08	Royal Challengers Bangalore			

```
3              Kings XI Punjab              Kings XI Punjab         field
4            Delhi Daredevils  Royal Challengers Bangalore           bat

   result  dl_applied                       winner  win_by_runs  \
0  normal           0          Sunrisers Hyderabad           35
1  normal           0       Rising Pune Supergiant            0
2  normal           0        Kolkata Knight Riders            0
3  normal           0              Kings XI Punjab            0
4  normal           0  Royal Challengers Bangalore           15

   win_by_wickets player_of_match                                      venue  \
0               0    Yuvraj Singh  Rajiv Gandhi International Stadium, Uppal
1               7       SPD Smith    Maharashtra Cricket Association Stadium
2              10         CA Lynn     Saurashtra Cricket Association Stadium
3               6      GJ Maxwell                     Holkar Cricket Stadium
4               0       KM Jadhav                      M Chinnaswamy Stadium

          umpire1        umpire2 umpire3
0     AY Dandekar       NJ Llong     NaN
1  A Nand Kishore         S Ravi     NaN
2     Nitin Menon      CK Nandan     NaN
3    AK Chaudhary  C Shamshuddin     NaN
4             NaN            NaN     NaN
```

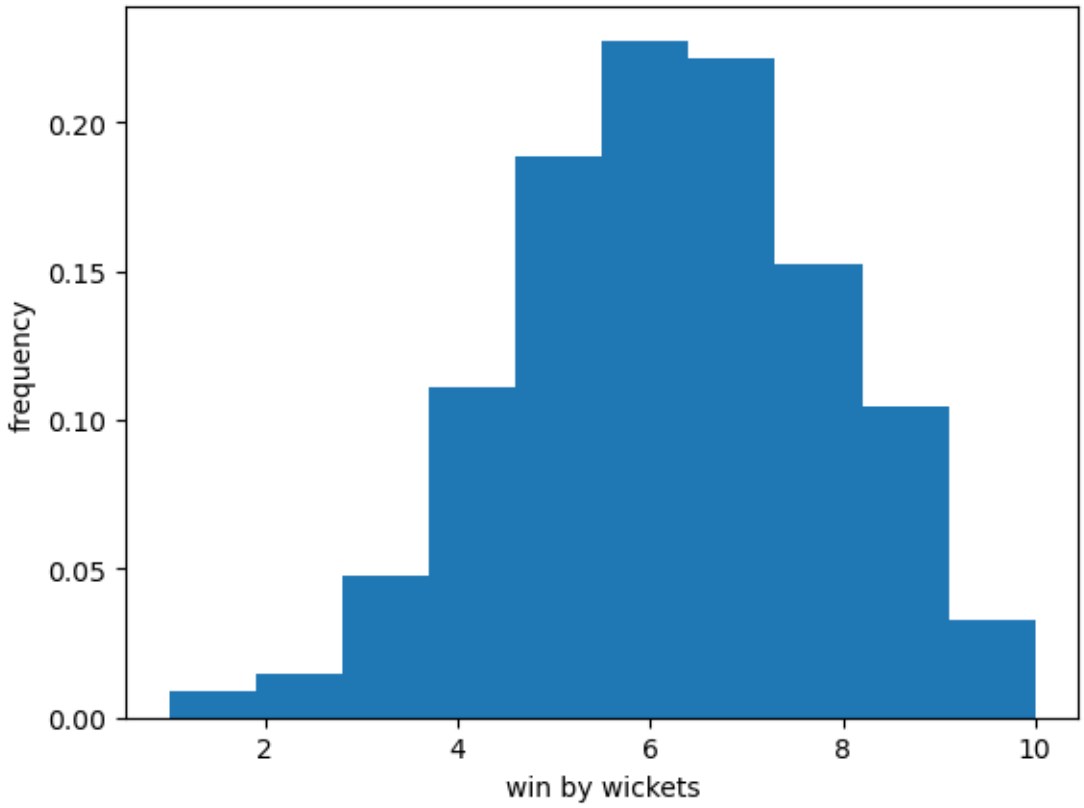
0.4 Plot a un-normalized bar plot of the winning margin in terms of the number of wickets: win by wickets in the horizontal axis and the un-normalized frequency in the vertical axis. Observe that the winning margin can be either in terms of wickets or in terms of runs but not both.

[ ]:
```
# Start your code here
wins_by_wicket=matches['win_by_wickets']
wins_by_wicket=wins_by_wicket[wins_by_wicket>0]
plt.hist(wins_by_wicket,bins=10)
plt.xlabel("win by wickets")
plt.ylabel("frequency")
plt.show()

# End your code here
```



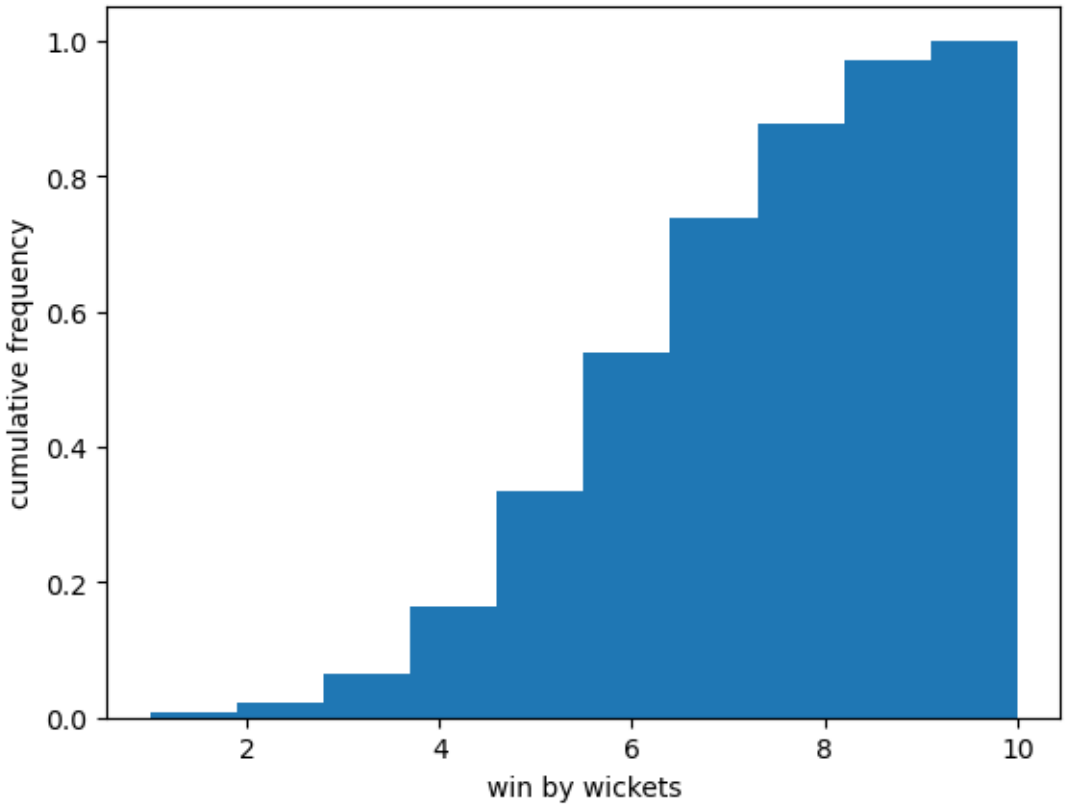
0.5 Plot a normalized bar plot of the winning margin in terms of the number of wickets: win by wickets in the horizontal axis and the normalized frequency in the vertical axis. Recall, that this will generate a plot of the empirical Probability Density Function (PDF).

[ ]:
```
# Start your code here

plt.hist(wins_by_wicket,bins=10,density=True)
plt.xlabel("win by wickets")
plt.ylabel("frequency")
plt.show()

# End your code here
```



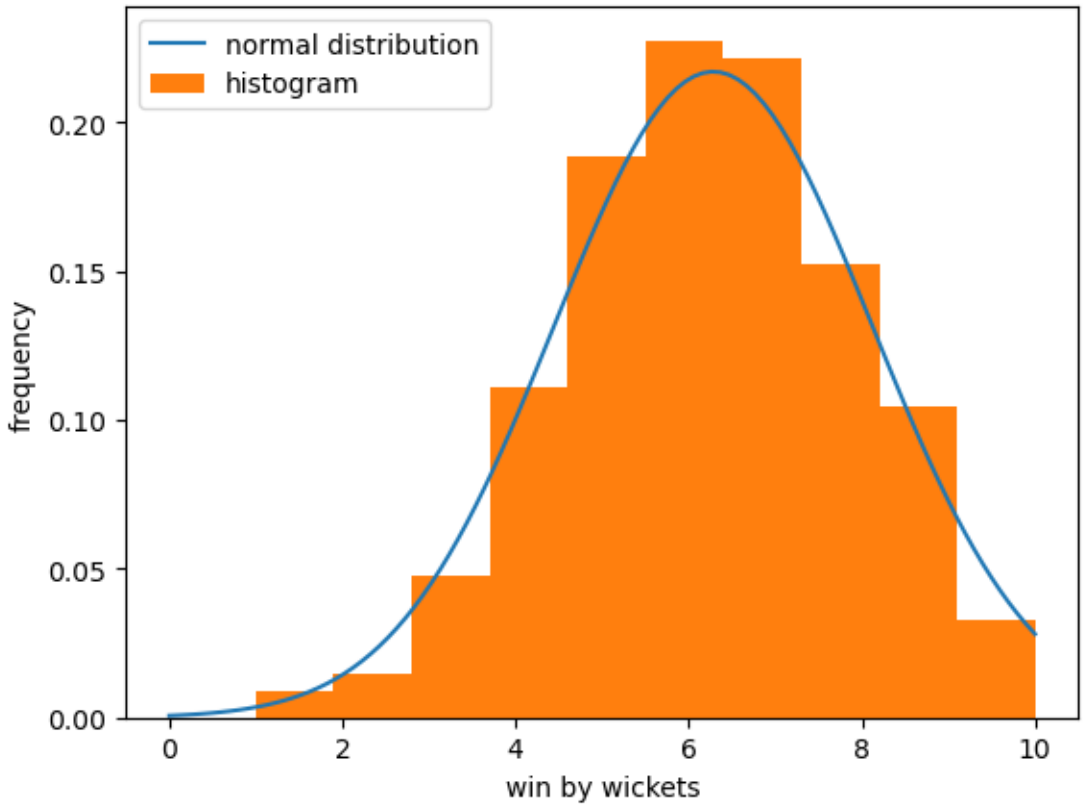
0.6 Plot the cumulative distribution of the winning margin in terms of the number of wickets: win by wickets in the horizontal axis and cumulative frequency in the vertical axis. Recall, that this will generate a plot of the empirical Cumulative Distribution Function (CDF).

[ ]:
```
# Start your code here

plt.hist(wins_by_wicket,bins=10,density=True,cumulative=True)
plt.xlabel("win by wickets")
plt.ylabel("cumulative frequency")
plt.show()

# End your code here
```



0.7 Fit a gaussian distribution to the empirical PDF: plot the empirical PDF and the gaussian distribution fitting the emprical PDF on the same plot

[ ]:
```
# Start your code here

mu=wins_by_wicket.mean()
sigma=wins_by_wicket.std()
x=np.linspace(0,10,100)
y=stats.norm.pdf(x,mu,sigma)
plt.plot(x,y)
plt.hist(wins_by_wicket,bins=10,density=True)
plt.xlabel("win by wickets")
plt.ylabel("frequency")
plt.legend(['normal distribution','histogram'])
plt.show()

# End your code here
```



0.8 Indian premier league deliveries dataset: data on balls bowled in the Indian premier league from 2008-2018. The data has many attributes but in this assignment we will only focus on some of the attributes

[ ]:
```
## Loading the dataset as a pandas dataframe and printing the header
deliveries = pd.read_csv('deliveries.csv')
deliveries.head()
```

[ ]:

	match_id	inning	batting_team	bowling_team	over	ball	batsman	non_striker	bowler	is_super_over	…	bye_runs
0	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	1	DA Warner	S Dhawan	TS Mills	0	…	0
1	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	2	DA Warner	S Dhawan	TS Mills	0	…	0
2	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	3	DA Warner	S Dhawan	TS Mills	0	…	0
3	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	4	DA Warner	S Dhawan	TS Mills	0	…	0
4	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	5	DA Warner	S Dhawan	TS Mills	0	…	0

```
   legbye_runs  noball_runs  penalty_runs  batsman_runs  extra_runs  \
0            0            0             0             0           0
1            0            0             0             0           0
2            0            0             0             4           0
3            0            0             0             0           0
4            0            0             0             0           2

   total_runs  player_dismissed dismissal_kind fielder
0           0               NaN            NaN     NaN
1           0               NaN            NaN     NaN
2           4               NaN            NaN     NaN
3           0               NaN            NaN     NaN
4           2               NaN            NaN     NaN

[5 rows x 21 columns]
```

[ ]:
```
deliveries.columns
```

[ ]:
```
Index(['match_id', 'inning', 'batting_team', 'bowling_team', 'over', 'ball',
       'batsman', 'non_striker', 'bowler', 'is_super_over', 'wide_runs',
       'bye_runs', 'legbye_runs', 'noball_runs', 'penalty_runs',
       'batsman_runs', 'extra_runs', 'total_runs', 'player_dismissed',
       'dismissal_kind', 'fielder'],
      dtype='object')
```

0.9 Process the dataframe and create a new dataframe with the following structure:

0.10 - Rows correspond to the match ID

0.11 - Column 1 correspond to the total number of runs scored in the match

0.12 - Column 2 correspond to the total number of runs scored in the match by the batsman

0.13 After you have created the dataframe, print the header of the dataframe.

[ ]:
```
# Start your code here
mod_df=deliveries.groupby('match_id')['batsman_runs','total_runs'].sum()
mod_df.head()
# End your code here
```

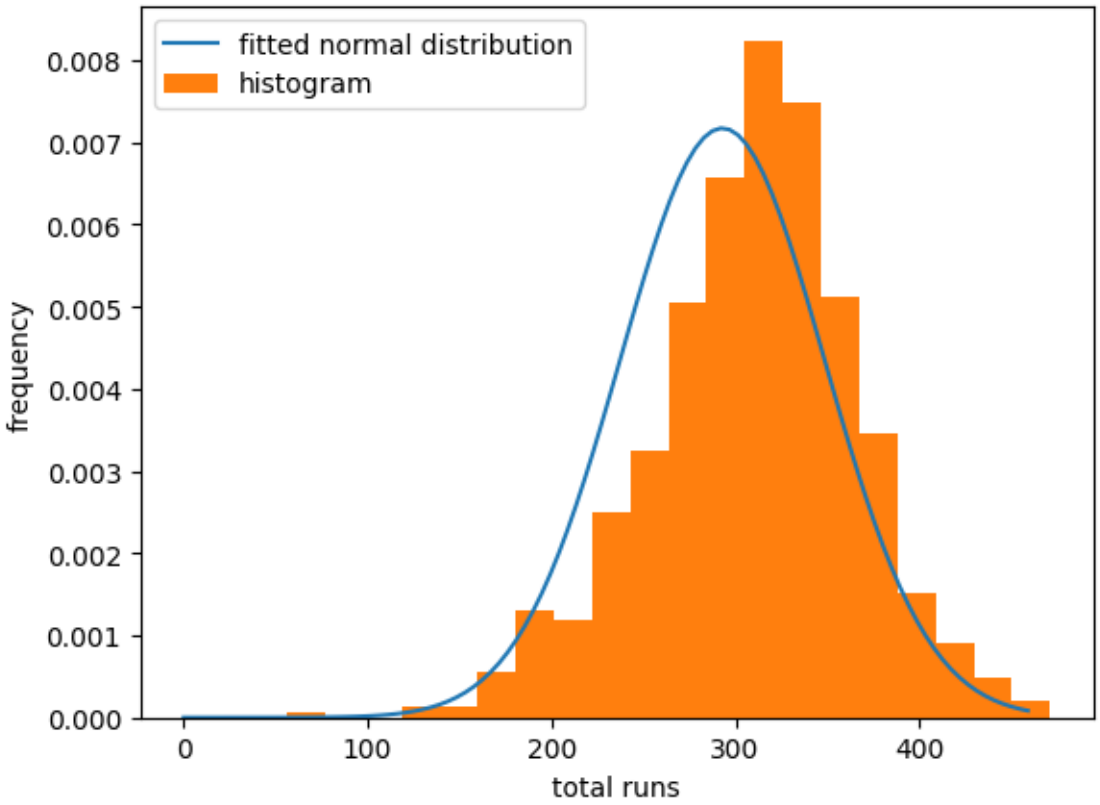
```
/tmp/ipykernel_4156/192070405.py:2: FutureWarning: Indexing with multiple keys
(implicitly converted to a tuple of keys) will be deprecated, use a list
instead.
  mod_df=deliveries.groupby('match_id')['batsman_runs','total_runs'].sum()
```

[ ]:

match_id	batsman_runs	total_runs
1	366	379
2	359	371
3	348	367
4	311	327
5	288	299

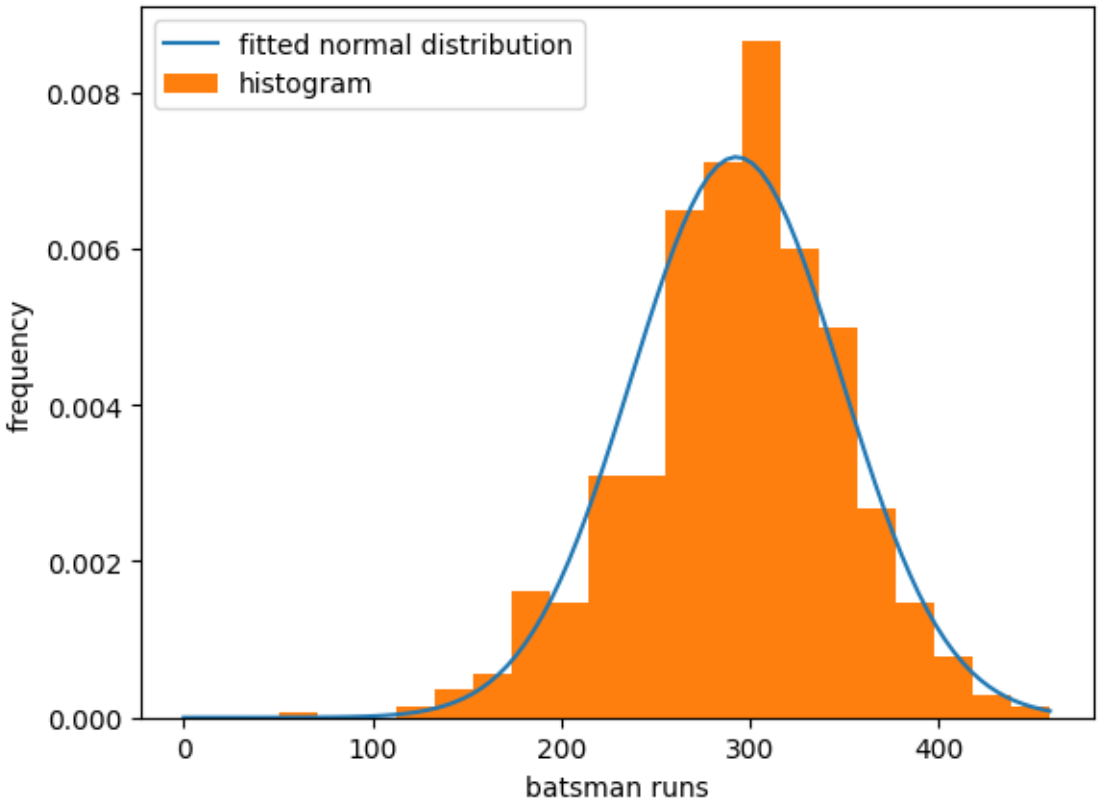
0.14 Fitting a gaussian distribution to the total number of runs scored in a match: plot the empirical PDF of the total number of runs scored in a match and the gaussian distribution fitting the emprical PDF on the same plot

[ ]:
```
# Start your code here
mu=mod_df['batsman_runs'].mean()
sigma=mod_df['batsman_runs'].std()
x=np.linspace(0,max(mod_df['batsman_runs']),100)
y=stats.norm.pdf(x,mu,sigma)
plt.plot(x,y)

plt.hist(mod_df['total_runs'],bins=20,density=True)
plt.xlabel("total runs")
plt.ylabel("frequency")
plt.legend(['fitted normal distribution','histogram'])
plt.show()

# End your code here
```



0.15 Fitting a gaussian distribution to the total number of runs scored by batsman in a match: plot the empirical PDF of the total number of runs scored by batsman in a match and the gaussian distribution fitting the emprical PDF on the same plot

[ ]:
```
# Start your code here

mu=mod_df['batsman_runs'].mean()
sigma=mod_df['batsman_runs'].std()
x=np.linspace(0,max(mod_df['batsman_runs']),100)
y=stats.norm.pdf(x,mu,sigma)

plt.plot(x,y)
plt.hist(mod_df['batsman_runs'],bins=20,density=True)
plt.xlabel("batsman runs")
plt.ylabel("frequency")
plt.legend(['fitted normal distribution','histogram'])
plt.show()

# End your code here
```


fitted normal distribution
histogram
0.008
0.006
0.004
0.002
0.000
frequency
0
100
200
300
400
batsman runs

[ ]: