1. INTRODUCTION

1.1 ABOUT THE PROJECT

Data analytics (DA) is the process of examining data sets in order to find trends and draw conclusions about the information they contain. Increasingly, data analytics is done with the aid of specialized systems and software. Data analytics technologies and techniques are widely used in commercial industries to enable organizations to make more-informed business decisions. Scientists and researchers also use analytics tools to verify or disprove scientific models, theories and hypotheses.

Data analytics initiatives can help businesses increase revenue, improve operational efficiency, optimize marketing campaigns and bolster customer service efforts. Analytics also enable organizations to respond quickly to emerging market trends and gain a competitive edge over business rivals. The ultimate goal of data analytics, however, is boosting business performance. Depending on the particular application, the data that's analyzed can consist of either historical records or new information that has been processed for real-time analytics. In addition, it can come from a mix of internal systems and external data sources.

2. SYSTEM STUDY

2.1 EXISTING SYSTEM

The earlier system was a raw dataset which contains only the data in tabular form. It had various information of a player such as average,high score,best figures etc..,.It ttok around the data of batsmen of nearly 160+ players of around 10 teams and rectified the top players which are present in the dataset .Later on, few analysts worked on it to provide a visual analysis of dataset and find some of the insights.

2.2 DISADVANTAGES OF EXISTING SYSTEM

- It is not applied to visualize in various forms of Visualization charts that is available in Matplotlib and Seaborn Libraries
- Machine Learning algorithms are not applied and so we cannot predict the future of a player and the need of him in that team.
- Hypothesis testing and statistical analysis is npt done and so we cannot analyze a player's performance at nook and corner.

2.3 PROPOSED SYSTEM

The Analytics that I have done "CRICKET DATA ANALYTICS AND SCORE PREDICTION ON KAGGLE DATASET(IPL 2022 BATTERS.CSV) " has analyzed the player's data in various modules such as Data Cleaning,Exploratory Data analysis,statistical Analysis,hypothesis Testing.Machine learning algorithms have been applied to automate the system which helps us to find the value of a person and the need of him in the team which will automatically predict the salary/Auction price of him for the next season.

2.4 ADVANTAGES OF PROPOSED SYSTEM

- Insights can be analyzed visually through various charts
- Teams csn easily find the player through various angles such as average,strike rate etc..,.
- Simpler code
- Statistical measurements are performed
- Hypotheses Testing helps to check the user’s interpretation eith the actual result
- Reduction of analysis in text format
- Keep track of a player records ehich ould help a team in combinations that they are planning for upcoming seasons
- Increase in processing Throgh various Machine Learning algorithms

2.5 PROBLEM DEFINITION AND DESCRIPTION

In earlier days, it was very difficult for the team to analyze a player in various angles such as Strike Rate,Average etc..,.as a team squad consists of 20+ players and a player can be useful at any point of time for a team. So, the question that I have taken is to find a player who can be strong in vaious angles,to find a player who is strong in particular angles and to find the salary /auction price of a player for upcoming season and help them to get best core for the team and build a team to win and we can keep insight of auction price that is suitable for a player.

3. SYSTEM ANALYSIS

3.1 PACKAGES SELECTED

Front End : Python

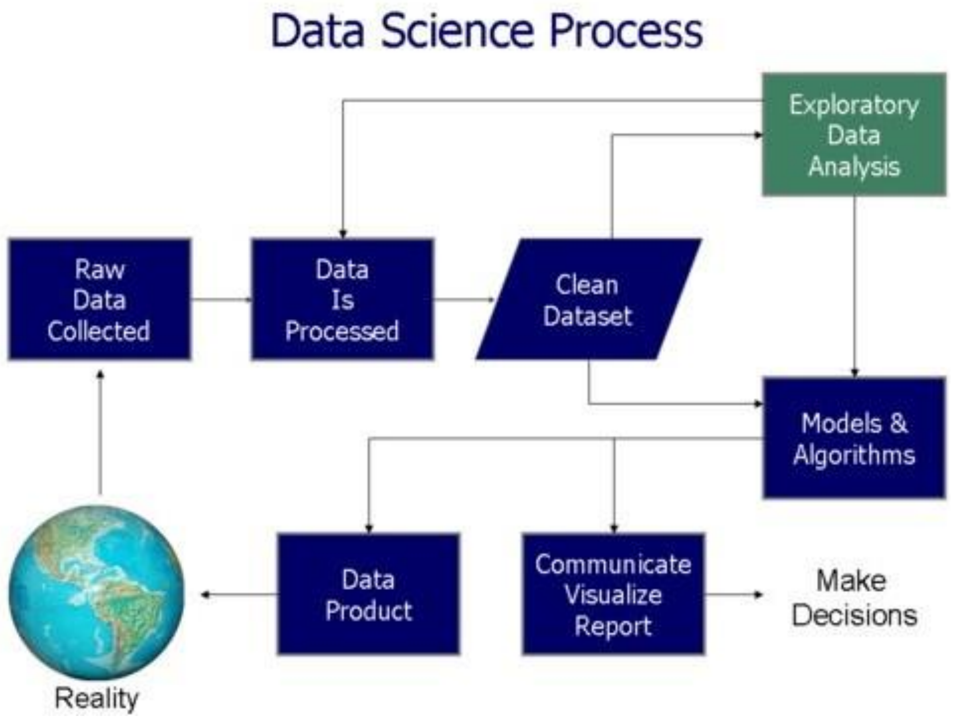
3.2 RESOURCES REQUIRED

Hardware Specification	Operating System	Windows 10 or higher
	Processor	Minimum - Intel Core i3 Recommended - Intel Core i5
	RAM	Minimum – 8 GB Recommended – 16 GB

Software Specification	IDE	Jupyter / Colab
	Libraries	NumPy, Pandas, Scikit Learn, Matplotbib, Seaborn, IpyWidgets etc.

3.3 DATA FLOW DIAGRAM

A two-dimensional diagram explains how data is processed and transferred in a system. The graphical depiction identifies each source of data and how it interacts with other data sources to reach a common output. Individuals seeking to draft a data flow diagram must identify external inputs and outputs, determine how the inputs and outputs relate to each other, and explain with graphics how these connections relate and what they result in. This type of diagram helps business development and design teams visualize how data is processed and identify or improve certain aspects.



4. SYSTEM DEVELOPMENT

4.1 FUNCTIONAL DOCUMENTATION

4.1.1 Data Collection

Data Collection is the process of gathering information on targeted variables identified as data requirements. The emphasis is on ensuring accurate and honest collection of data. Data Collection ensures that data gathered is accurate such that the related decisions are valid. Data Collection provides both a baseline to measure and a target to improve.

Data is collected from various sources ranging from organizational databases to the information in web pages. The data thus obtained, may not be structured and may contain irrelevant information. Hence, the collected data is required to be subjected to Data Processing and Data Cleaning.

4.1.2 Data Processing

The data that is collected must be processed or organized for analysis. This includes structuring the data as required for the relevant Analysis Tools. For example, the data might have to be placed into rows and columns in a table within a Spreadsheet or Statistical Application. A Data Model might have to be created.

4.1.3 Data Cleaning

Data cleaning is a crucial process in Data Mining. It carries an important part in the building of a model. Data Cleaning can be regarded as the process needed, but everyone often neglects it. Data quality is the main issue in quality information management. Data quality problems occur anywhere in information systems. These problems are solved by data cleaning.

4.1.4 Exploratory Data Analysis

Data that is processed, organized and cleaned would be ready for the analysis. Various data analysis techniques are available to understand, interpret, and derive conclusions based on the requirements. Data Visualization may also be used to examine the data in graphical format, to obtain additional insight regarding the messages within the data.

Statistical Data Models such as Correlation, Regression Analysis can be used to identify the relations among the data variables. These models that are descriptive of the data are helpful in simplifying analysis and communicate results.

The process might require additional Data Cleaning or additional Data Collection, and hence these activities are iterative in nature.

4. 1.5 Machine Learning

Machine Learning is the field of study that gives computers the capability to learn without being explicitly programmed. ML is one of the most exciting technologies that one would have ever come across. As it is evident from the name, it gives the computer that makes it more similar to humans: The ability to learn. Machine learning is actively being used today, perhaps in many more places than one would expect.

Machine learning is data driven technology. Large amount of data generated by organizations on daily bases. So, by notable relationships in data, organizations makes better decisions.

- Machine can learn itself from past data and automatically improve.
- From the given dataset it detects various patterns on data.
- For the big organizations branding is important and it will become more easy to target relatable customer base.
- It is similar to data mining because it is also deals with the huge amount of data.

4.2 SPECIAL FEATURES OF LANGUAGE/UTILITY

Python

Python is a popular, high-level programming language that is widely used for web development, scientific computing, data analysis, artificial intelligence, and more. It is known for its simplicity and readability, making it easy to learn and use.

Some of the key features of Python include:

Dynamic typing: Python uses dynamic typing, which means that variables do not have a fixed type, and the type of a variable is determined at runtime.

Interpreted language: Python is an interpreted language, which means that it does not need to be compiled before it is executed. This makes it easy to write and test code quickly.

Large and active community: Python has a large and active community, which means that there are many libraries and frameworks available for a wide range of tasks.

Object-oriented programming: Python supports object-oriented programming, which allows you to create reusable, modular code by defining classes and objects.

Large standard library: Python comes with a large standard library that provides functionality for many common programming tasks, such as connecting to web servers, reading and writing files, and working with data.

Cross-platform: Python runs on Windows, Mac OS, Linux and other operating systems, which makes it easy to write and run code on different platforms.

Python libraries:

Python is a popular programming language that has a large and active community, and as a result, there are many libraries available for a wide range of tasks. Here are a few of the most popular and widely used libraries:

NumPy:

A library for numerical computing and data manipulation. It provides support for large, multi-dimensional arrays and matrices, along with a large collection of high- level mathematical functions to operate on them.

Pandas:

A library for data manipulation and analysis. It provides data structures and operations for working with numerical tables and time series data. It's often used for data cleaning, transformation, and exploration.

Matplotlib:

A plotting library for creating static, animated, and interactive visualizations. It's commonly used for creating plots, histograms, bar charts, scatter plots, and more.

Seaborn :

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. For a brief introduction to the ideas behind the library, you can read the introductory notes or the paper.

SciPy:

The name “SciPy” stands for “Scientific Python”. It is an open-source library used for high-level scientific computations. This library is built over an extension of Numpy. It works with Numpy to handle complex computations. While Numpy allows sorting and indexing of array data, the numerical data code is stored in SciPy. It is also widely used by application developers and engineers.

Scikit-learn:

It is a famous Python library to work with complex data. Scikit-learn is an open-source library that supports machine learning. It supports variously supervised and unsupervised algorithms like linear regression, classification, clustering, etc. This library works in association with Numpy and SciPy.

4.3 PSEUDO CODE/ALGORITHM

```
Step 1:Start
Step 2:Import important Libraries
Step 3: Load the Dataset
Step 4:check the properties of dataset
Step 5: clean the data as per requirements
Step 6: perform Exploratory Data analysis
Step 7: Data Normalization and Transformation
Step 8: applying Machine Learning algorithms
Step 9: Finding Accuracy value an finding the peak value
Step 10: Stop.
```

5. USER MANUAL

5.1 HARDWARE /SOFTWARE REQUIREMENTS

Table 5.1 System Specification

Hardware Specification	Operating System	Windows 10 or higher
	Processor	Minimum - Intel Core i3 Recommended - Intel Core i5
	RAM	Minimum – 8 GB Recommended – 16 GB

Software Specification	IDE	Jupyter / Colab
	Libraries	NumPy, Pandas, Scikit Learn, Matplotbib, Seaborn, IpyWidgets etc.

5.2 INSTALLATION PROCEDURE

Python Installation Guide

Step 1: Go to https://www.python.org/ and under Downloads tab, select the latest version of Python for your operating system. In this case of Windows OS. The latest version of Python when creating this guide was Python 3.10.4. 2 Here, we have shown the installation for Python 3.8.0. Follow the same steps for the updated versions as well.

Step 2: Select the .exe file that you downloaded and click on Run. The python installation page will be open as shown below. Select the options “Install launcher for all users” and “Add Python 3.8 to PATH”. Then click on Install Now.

Step 3: Now Python is installed on your operating system. To verify, search for IDLE app on your system. Alternatively, you can also confirm the installation of Python on your system by opening command prompt and typing the command python --version. 3 This completes the installation of Python on your OS. •

6. CONCLUSION

6.1 SUMMARY OF THE PROJECT

Data set is taken from kaggle Dataset named“IPL2022BATTERS.CSV” and with the help of Inbuilt python libraries.The Dataset is analyzed in various forms such as numerical, stataistical,Testing and various insights are taken such as highest Run scorer,Player with highest 4’s and highest Strike Rate with the help of various inbuilt libraries such as Matplotlib and Seaborn and insights are visualized in the forms of pirechart, Bargraph,histogram and more on .

Machine Learning algorithms re applied on to predict the player with most Runs which is taken as Target Variable and other columns are entitled as feature vectors to predict the value of Target variable.Various supervised machine learning algorithms are applied and the predicted values are calculated and equated with the original values of target variable and founfdthe best algorithm in terms of accuracy rate.

Random Forest algorithm is best suited for this dataset and it can be applied onto the model .so, that it can predict values perfectly if it is automated in a way for future usage

6.2 FUTURE ENHANCEMENTS

In future we can boost our applications using Boosting algorithms such as GRADIENT BOOST, XGBOOST etc..,.and we can deploy the model using DJANGO,FLASK and we can make it as a user friendly and implement it in a better way and satisfy the client’s needs as to get accuracte values of a player.

BIBLIOGRAPHY

BOOK REFERENCE

1.Python for Data Analysis by Wes McKinney,Released October 2012 for Data Wrangling with Numpy,Pandas in Jupyter,O'Reilly 3rd edition.

2.Practical Statistics for DataScientists,O'Reilly 3rd edition,Peter Breau,Andrew Barace and Peter Gedeck

3.Introduction to Machine Learning with Python,Andreas.c.Muller and Sarah,published by O'Reilly

WEBSITE REFERENCE

https://towardsdatascience.com/

https://learning.edureka.co/my-classroom/data-science-training-program/coursecontent?courseId=2071&fromMasterCourse=0.

APPENDIX

FINDINGS AND INTERPRETATION

CODE CELLS WITH OUTPUT CELLS

```
df=pd.read_csv("/content/IPL 2022 Batters.csv")
```

```
df.head(10)
```

Player	Mat	Inns	NO	Runs	HS	Avg	BF	SR	100	50	4s
Jos Buttler	17	17	2	863	116	57.53	579	149.05	4	4	83
K L Rahul	15	15	3	616	103*	51.33	455	135.38	2	4	45
Quinton De Kock	15	15	1	508	140*	36.29	341	148.97	1	3	47
Hardik Pandya	15	15	4	487	87*	44.27	371	131.26	0	4	49
Shubman Gill	16	16	2	483	96	34.5	365	132.32	0	4	51
David Miller	16	16	9	481	94*	68.71	337	142.72	0	2	32
Faf Du Plessis	16	16	1	468	96	31.2	367	127.52	0	3	49
Shikhar Dhawan	14	14	2	460	88*	38.33	375	122.66	0	3	47
Sanju Samson	17	17	1	458	55	28.63	312	146.79	0	2	43
Deepak Hooda	15	14	0	451	59	32.21	330	136.66	0	4	36

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 162 entries, 0 to 161
Data columns (total 12 columns):
 #  Column  Non-Null Count  Dtype
--  ----  ---------  ----
 0   Player  162 non-null    object
 1   Mat     162 non-null    int64
 2   Inns    162 non-null    int64
 3   NO      162 non-null    int64
 4   Runs    162 non-null    int64
 5   HS      162 non-null    object
 6   Avg     162 non-null    object
 7   BF      162 non-null    int64
 8   SR      162 non-null    float64
 9   100     162 non-null    int64
 10  50      162 non-null    int64
 11  4s      162 non-null    int64
dtypes: float64(1), int64(8), object(3)
memory usage: 15.3+ KB
```

df.isnull().sum()

```
Player    0
Mat       0
Inns      0
NO        0
Runs      0
HS        0
Avg       0
BF        0
SR        0
100       0
50        0
4s        0
dtype: int64
```

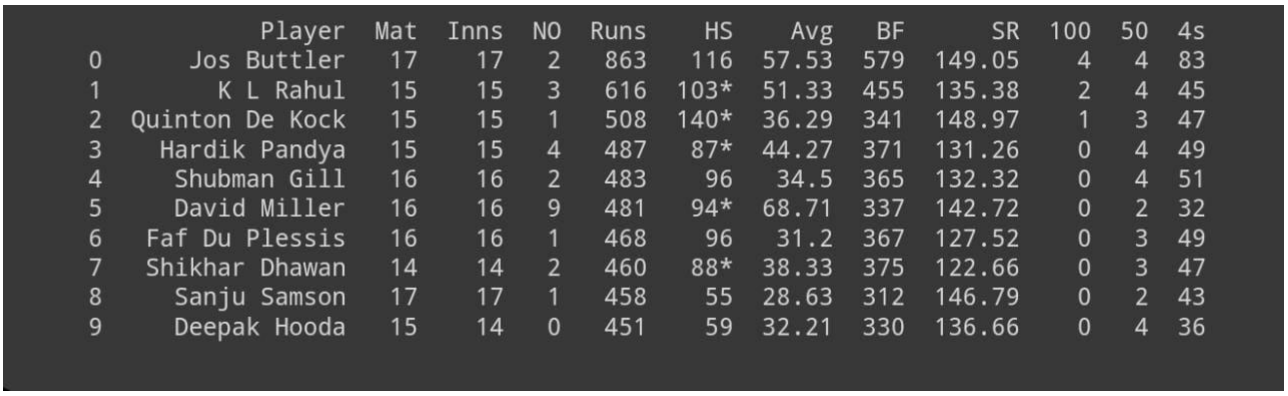
df.shape

```
(162, 12)
```

#piechart,Boxplot,Bargraph,scatterplot,heatmap,correlation matrix,Histogram,Countplot

import matplotlib.pyplot as plt
import seaborn as sns

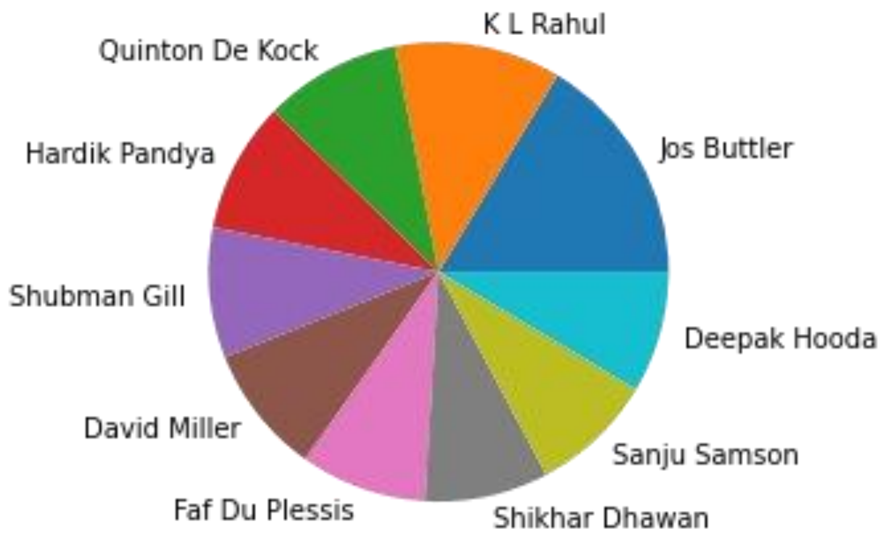
high=df.sort_values(by="Runs",ascending=False)
highest=high.head(10)
print(highest)



```
              Player  Mat  Inns  NO  Runs    HS    Avg   BF      SR  100  50  4s
0        Jos Buttler   17    17   2   863   116  57.53  579  149.05    4   4  83
1          K L Rahul   15    15   3   616  103*  51.33  455  135.38    2   4  45
2   Quinton De Kock    15    15   1   508  140*  36.29  341  148.97    1   3  47
3      Hardik Pandya   15    15   4   487   87*  44.27  371  131.26    0   4  49
4       Shubman Gill   16    16   2   483    96   34.5  365  132.32    0   4  51
5       David Miller   16    16   9   481   94*  68.71  337  142.72    0   2  32
6     Faf Du Plessis   16    16   1   468    96   31.2  367  127.52    0   3  49
7     Shikhar Dhawan   14    14   2   460   88*  38.33  375  122.66    0   3  47
8       Sanju Samson   17    17   1   458    55  28.63  312  146.79    0   2  43
9       Deepak Hooda   15    14   0   451    59  32.21  330  136.66    0   4  36
```

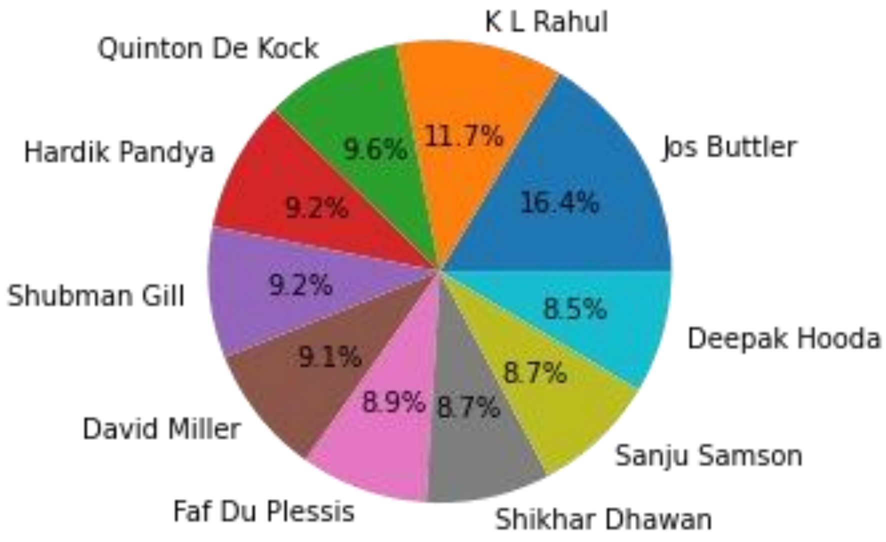
which player has scored more runs?

#normal pie chart
#parameter = runs
plt.pie(highest['Runs'],labels=highest['Player'])
plt.show()



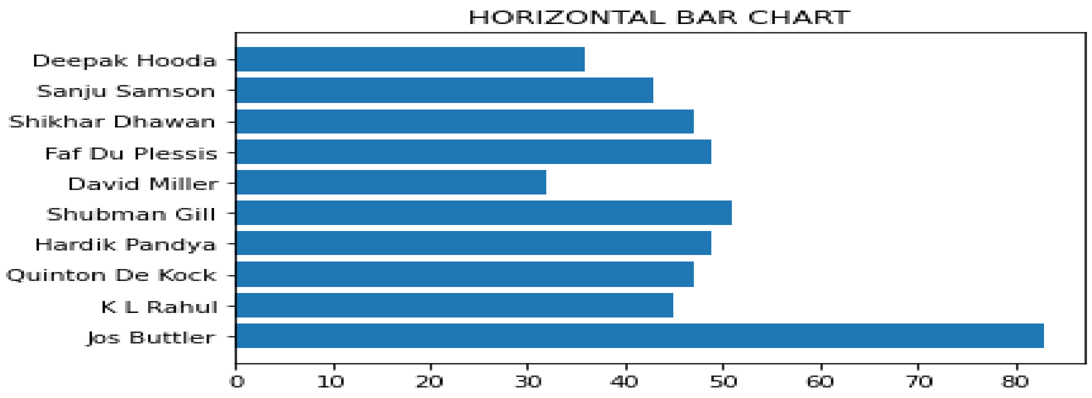
which player has scored more runs?

```
#pie chart with percentage values
#parameter = runs
plt.pie(highest['Runs'],labels=highest['Player'],autopct="%1.1f%%")
plt.show()
```



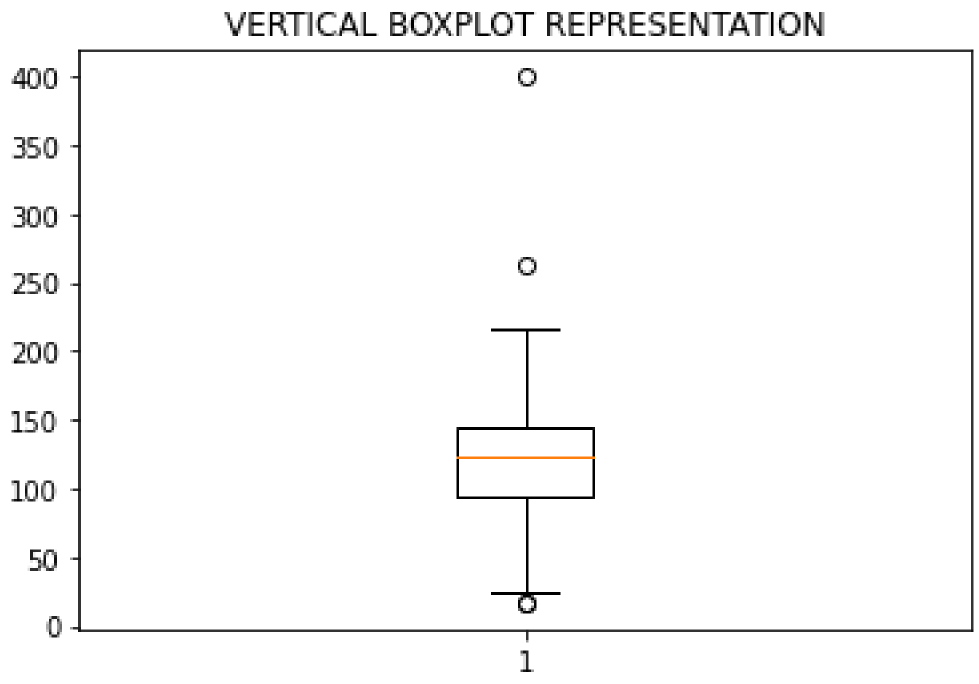
which player has scored more 4s?

```
plt.barh(highest['Player'],highest['4s'])
plt.title('HORIZONTAL BAR CHART')
plt.show()
```

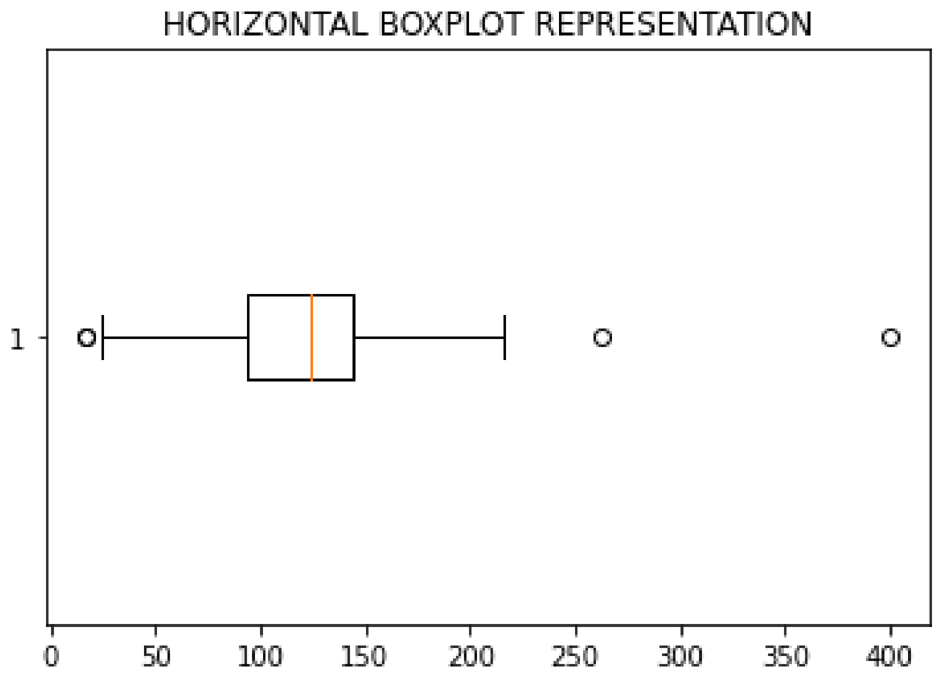


strike rate of all people(start-end)

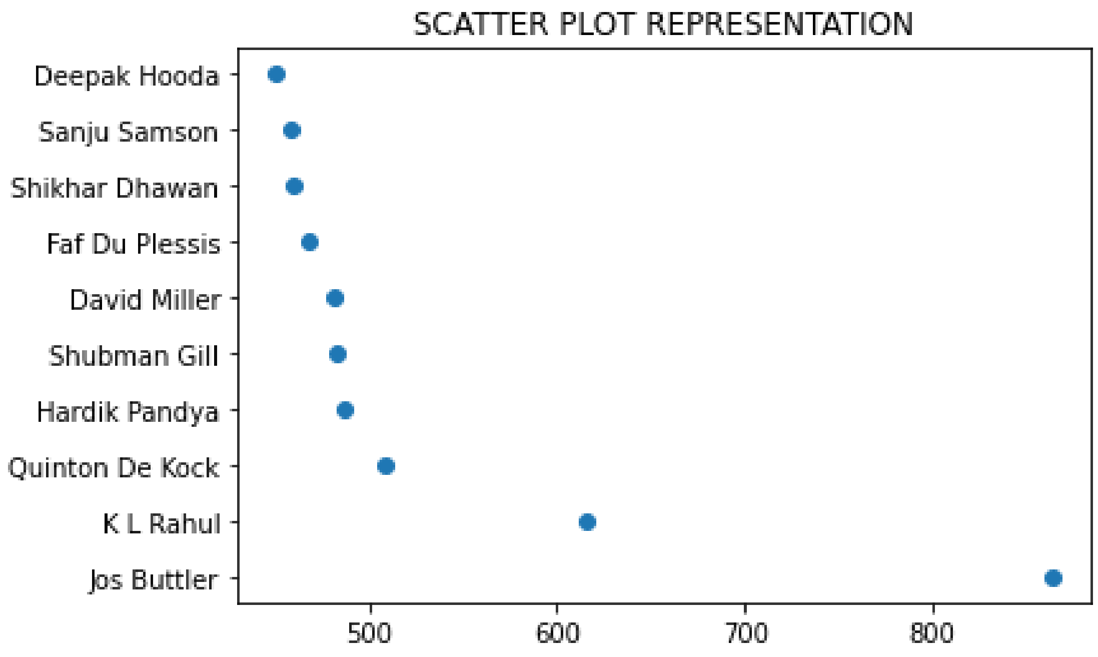
```
plt.boxplot(df['SR'])
plt.title('VERTICAL BOXPLOT REPRESENTATION')
plt.show()
```



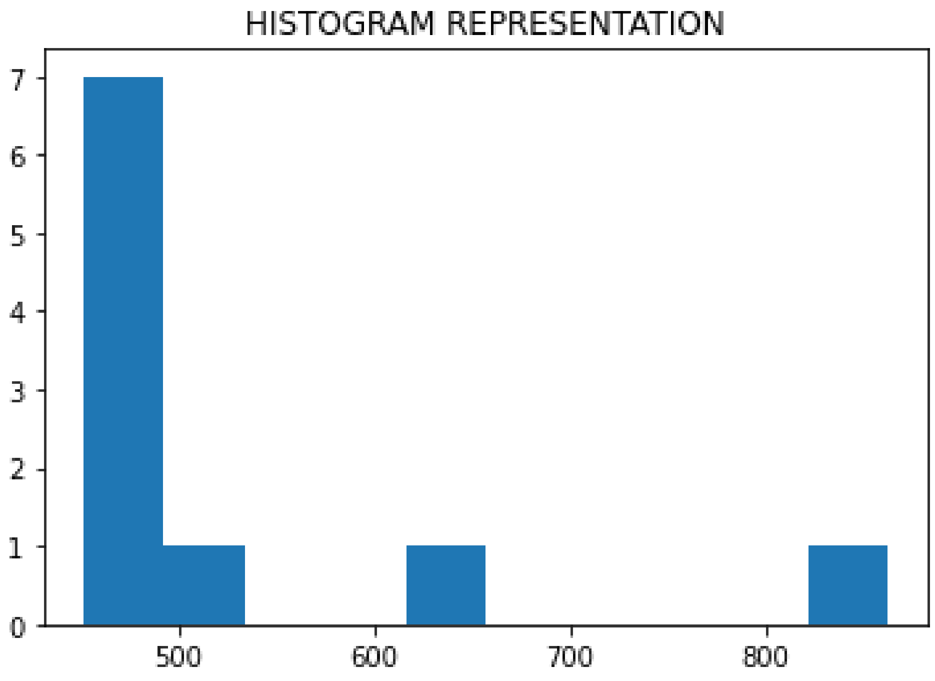
```
plt.boxplot(df['SR'],vert=False)
plt.title('HORIZONTAL BOXPLOT REPRESENTATION')
plt.show()
```



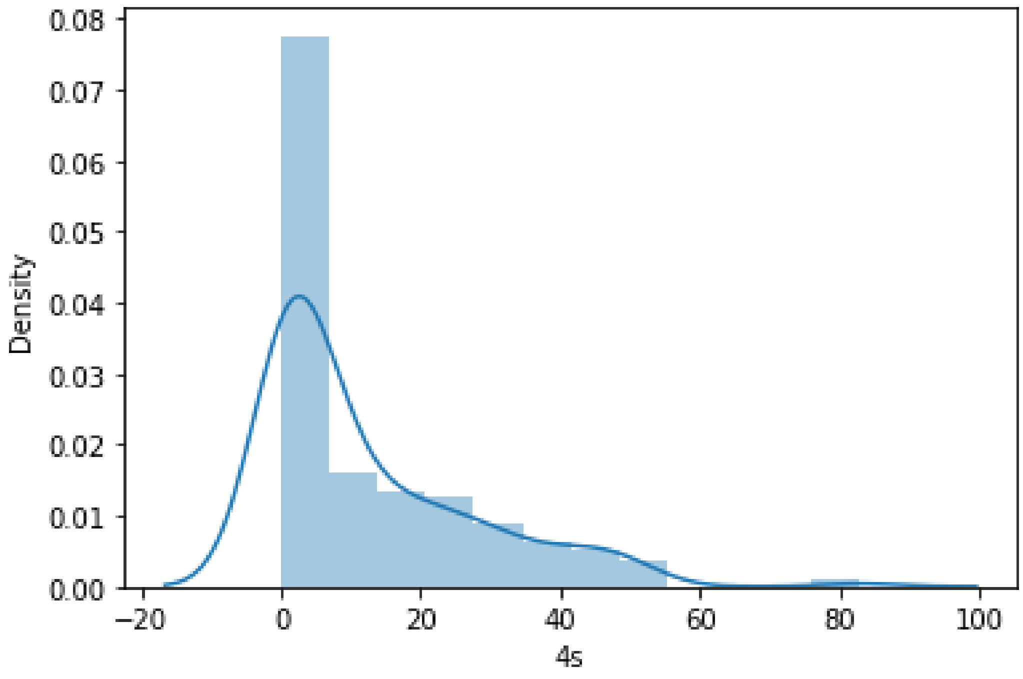
```
plt.scatter(highest['Runs'],highest['Player'])
plt.title('SCATTER PLOT REPRESENTATION')
plt.show()
```



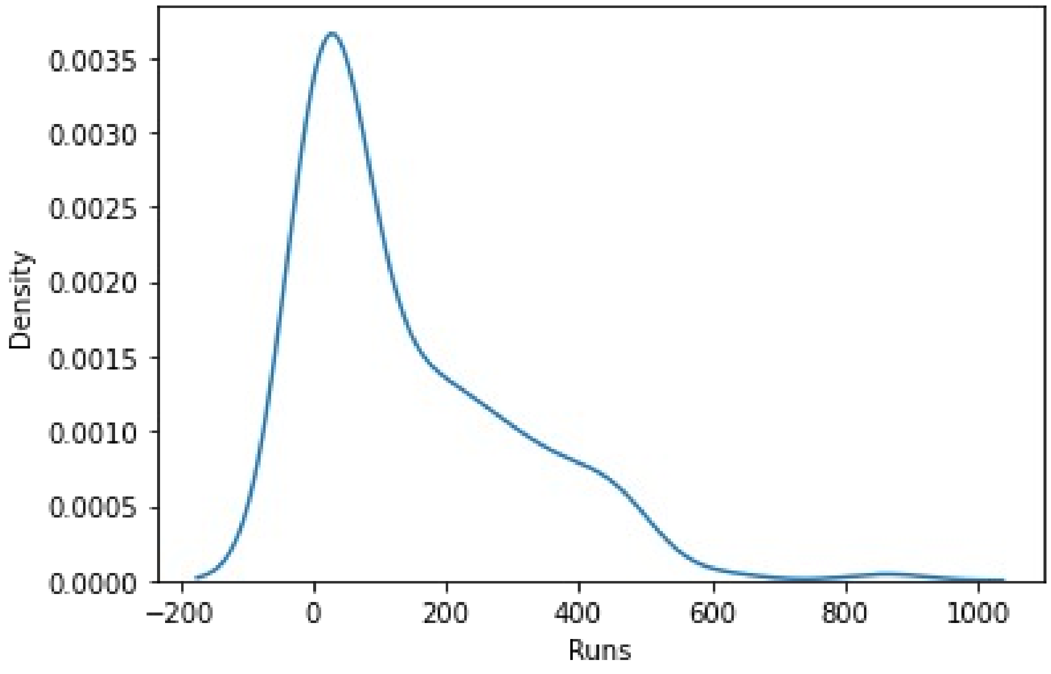
```
plt.hist(highest['Runs'])
plt.title("HISTOGRAM REPRESENTATION")
plt.show()
```



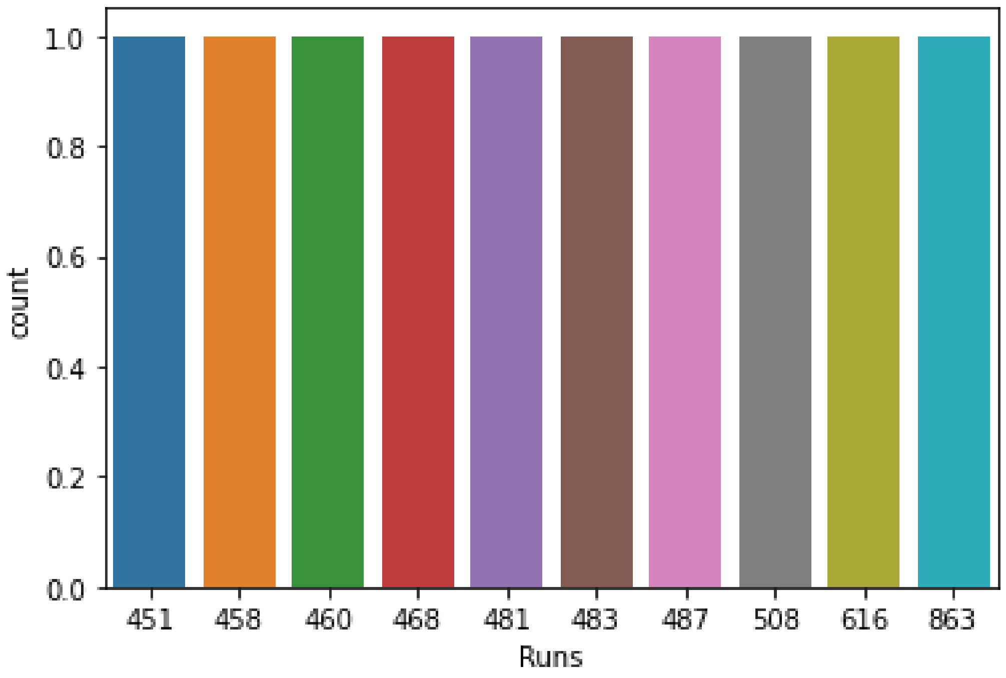
```
import seaborn as sns

sns.distplot(df['4s'])
plt.show()
```



```
sns.distplot(df['Runs'],hist=False)
plt.show()
```

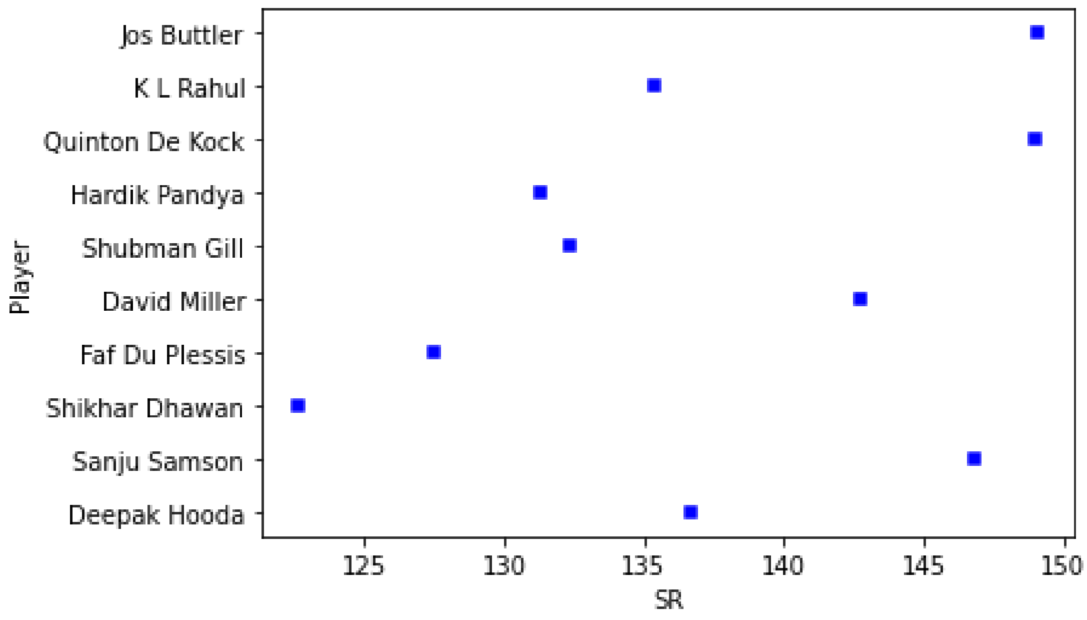


sns.countplot(x='Runs',data=highest)
plt.show()

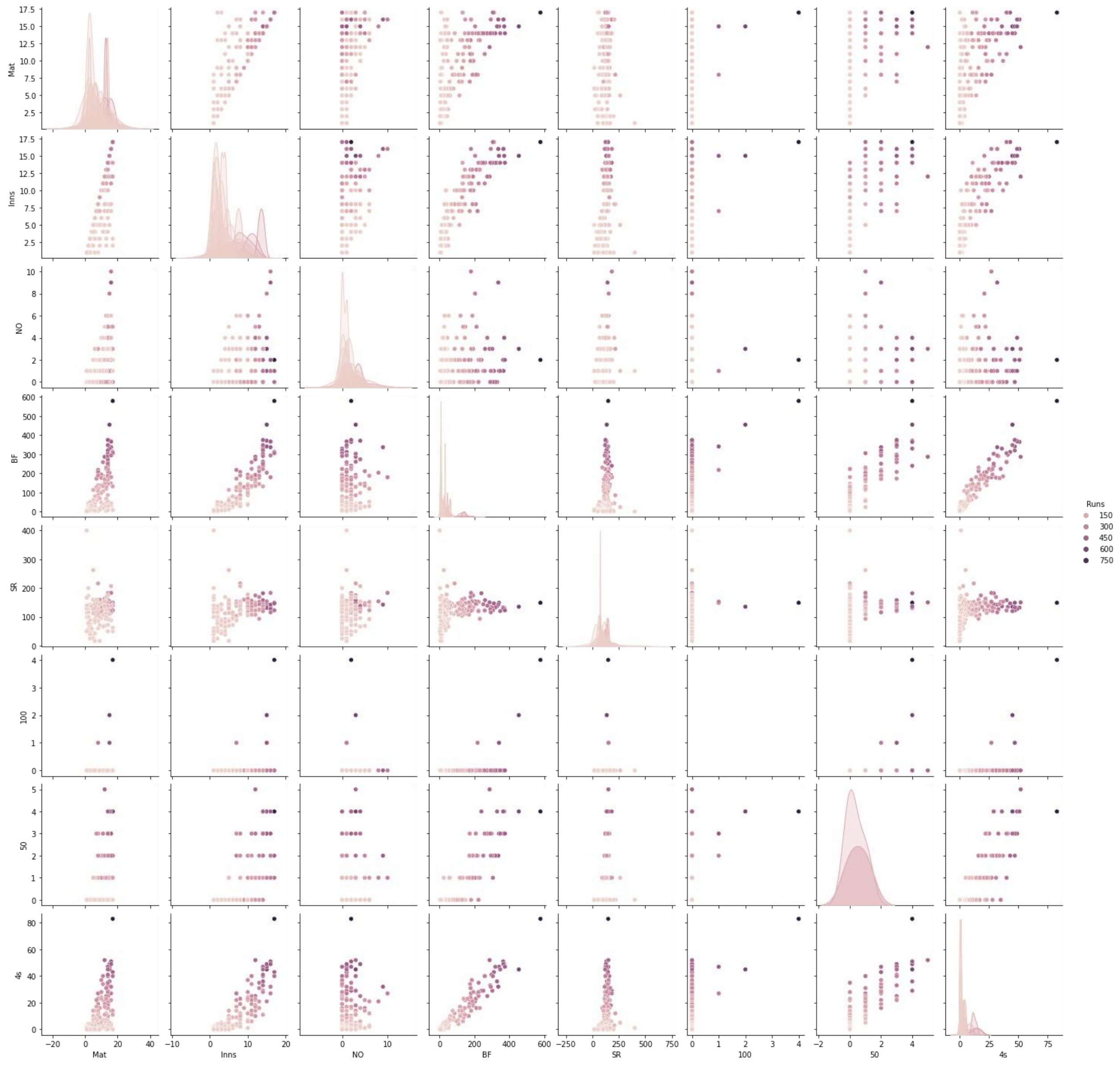


player with more strike rate

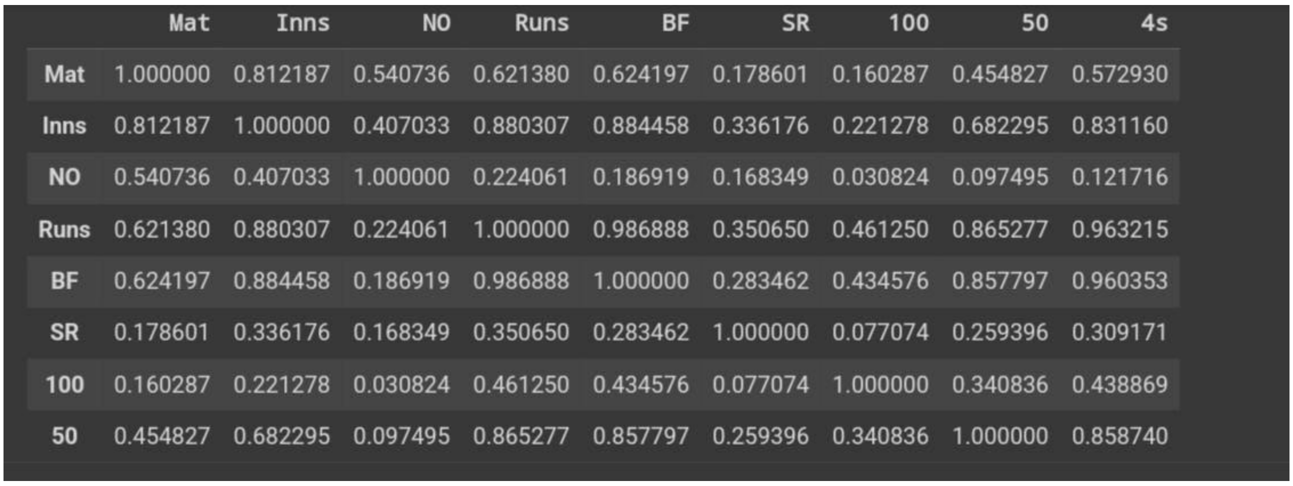
sns.scatterplot(x=highest["SR"],y=highest["Player"],marker='s',color='Blue')



```
sns.pairplot(df,hue='Runs')
plt.show()
```



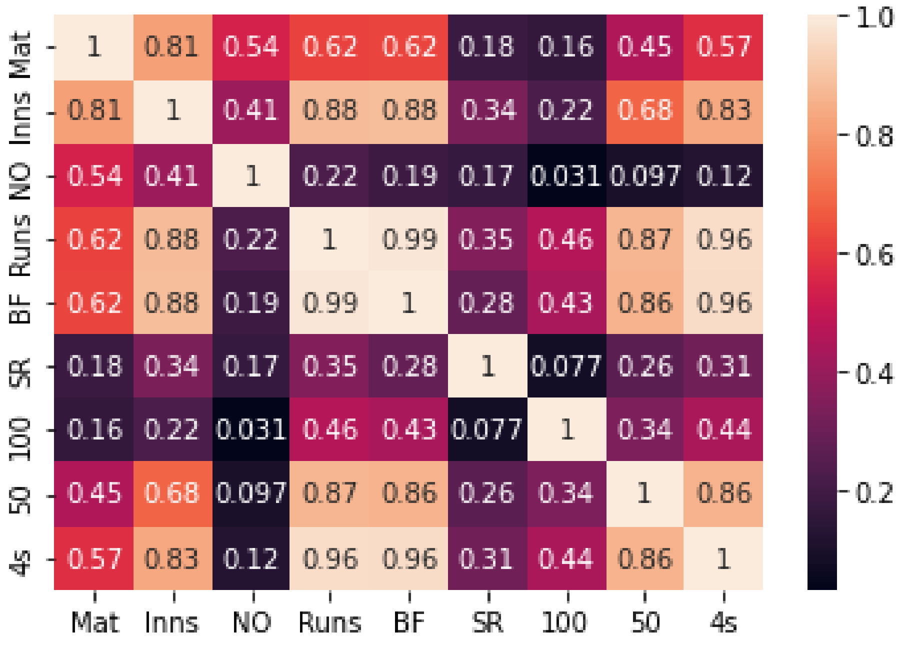
df.corr()



```
           Mat      Inns        NO      Runs        BF        SR       100        50        4s
Mat   1.000000  0.812187  0.540736  0.621380  0.624197  0.178601  0.160287  0.454827  0.572930
Inns  0.812187  1.000000  0.407033  0.880307  0.884458  0.336176  0.221278  0.682295  0.831160
NO    0.540736  0.407033  1.000000  0.224061  0.186919  0.168349  0.030824  0.097495  0.121716
Runs  0.621380  0.880307  0.224061  1.000000  0.986888  0.350650  0.461250  0.865277  0.963215
BF    0.624197  0.884458  0.186919  0.986888  1.000000  0.283462  0.434576  0.857797  0.960353
SR    0.178601  0.336176  0.168349  0.350650  0.283462  1.000000  0.077074  0.259396  0.309171
100   0.160287  0.221278  0.030824  0.461250  0.434576  0.077074  1.000000  0.340836  0.438869
50    0.454827  0.682295  0.097495  0.865277  0.857797  0.259396  0.340836  1.000000  0.858740
```

sns.heatmap(data=df.corr(),annot=True)

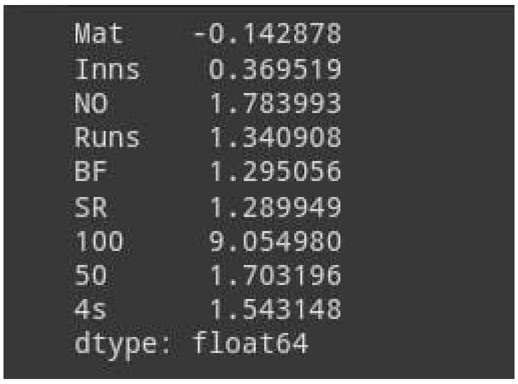
<Axes: >



various data visualization techniques are performed using seaborn and matplotlib libraries

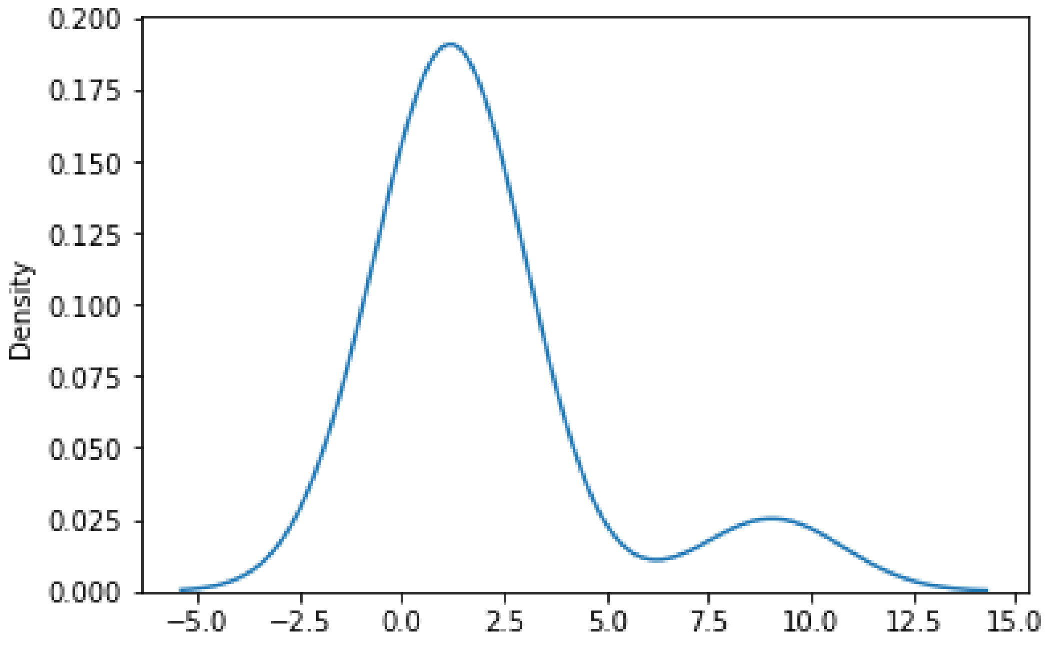
#Statistical analysis and EDA
import pandas as pd
import numpy as np
import math
from scipy import stats

s0=df.skew(axis=0)

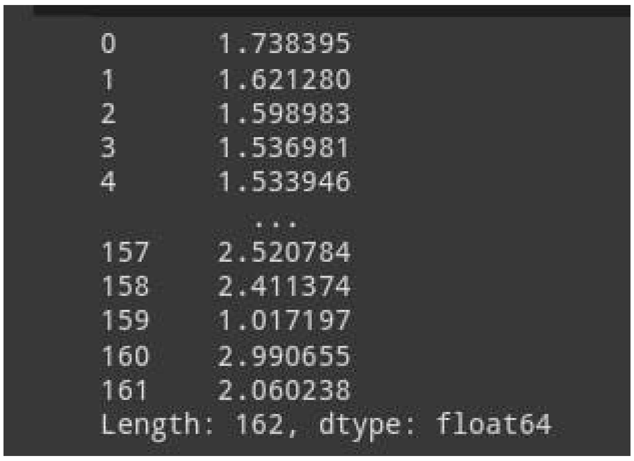


```
Mat     -0.142878
Inns     0.369519
NO       1.783993
Runs     1.340908
BF       1.295056
SR       1.289949
100      9.054980
50       1.703196
4s       1.543148
dtype: float64
```

sns.distplot(s0, hist = False)

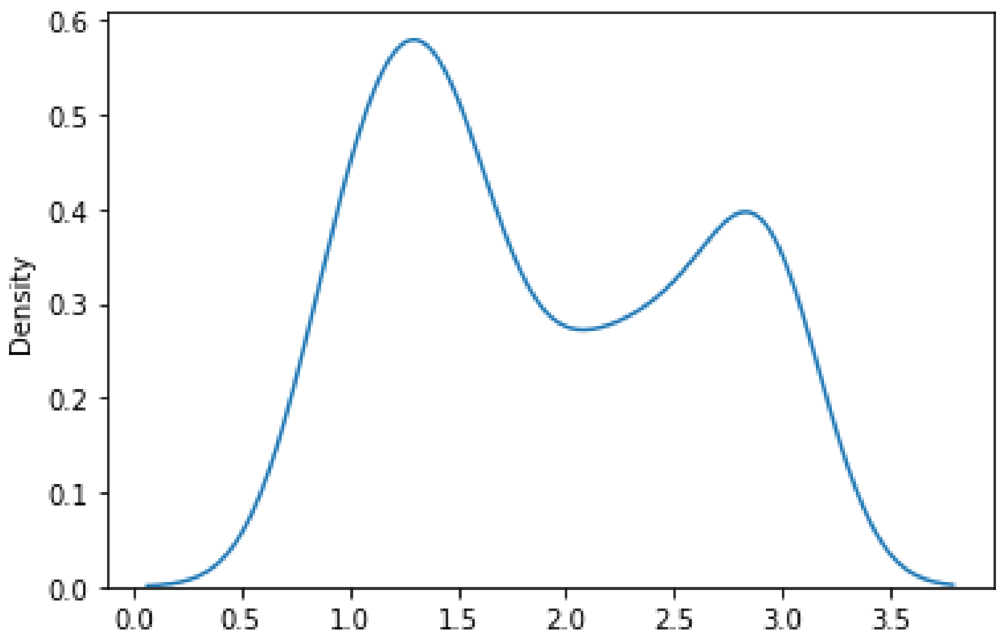


s1=df.skew(axis=1)

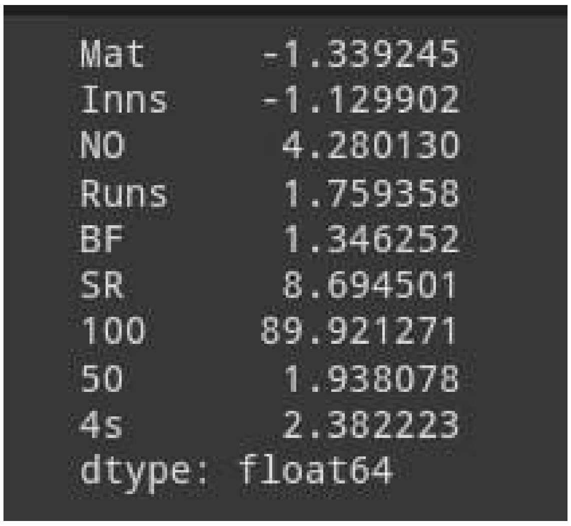


```
0      1.738395
1      1.621280
2      1.598983
3      1.536981
4      1.533946
         ...
157    2.520784
158    2.411374
159    1.017197
160    2.990655
161    2.060238
Length: 162, dtype: float64
```

sns.distplot(s1, hist = False)

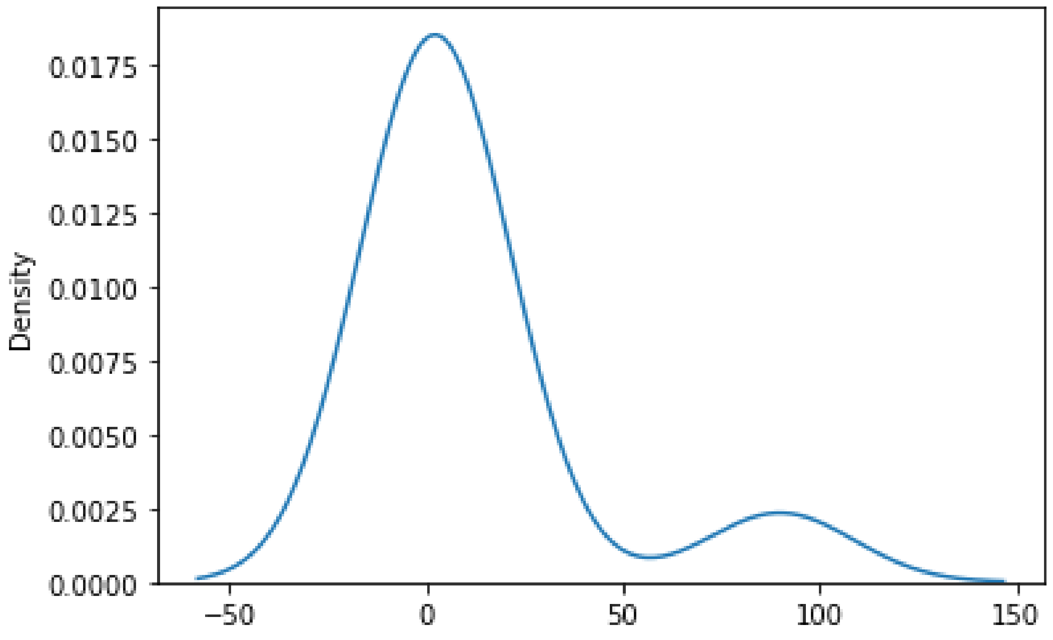


k0=df.kurtosis(axis=0)
k0

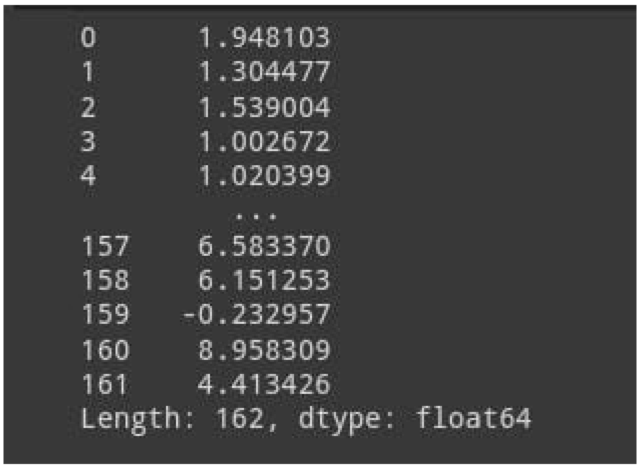


```
Mat      -1.339245
Inns     -1.129902
NO        4.280130
Runs      1.759358
BF        1.346252
SR        8.694501
100      89.921271
50        1.938078
4s        2.382223
dtype: float64
```

sns.distplot(k0, hist = False)

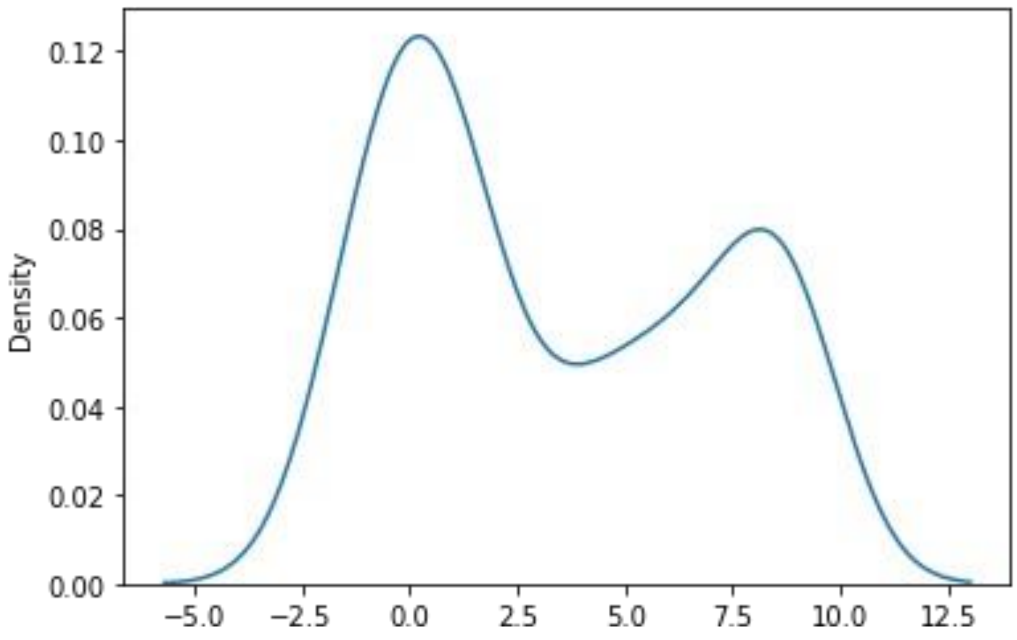


```
k1=df.kurtosis(axis=1)
k1
```



```
0       1.948103
1       1.304477
2       1.539004
3       1.002672
4       1.020399
          ...
157     6.583370
158     6.151253
159    -0.232957
160     8.958309
161     4.413426
Length: 162, dtype: float64
```

```
sns.distplot(k1, hist = False)
```



```
#STATISTICAL ANALYSIS

print(df['Runs'].var())
print(df['Runs'].mean())
print(df['Runs'].std())

26002.259489302967
142.2962962962963
161.25216119265804
```

SAMPLING DISTRIBUTION

```
#SAMPLE SIZE
#SAMPLE
#TARGET VARIABLE
#SAMPLE MEAN
#SAMPLE STD
#ZCRITICAL
#ZSTAT
#MARGIN OF ERROR
#CONFIDENCE INTERVAL


sample_size = 20
sample = df.sample(n=sample_size, random_state=2)

#TARGET = RUNS
sample_mean = sample.Runs.mean()
sample_std = sample.Runs.std()

Z_critical=stats.norm.ppf(q=0.975)
SE=sample_std/np.sqrt(20)
Z_stat=(sample_mean)/SE
print("Z_cririical value is:",Z_critical)
print("Z_stat value is:",Z_stat)

Z_cririical value is: 1.959963984540054
Z_stat value is: 4.4233325165476804

margin_of_error = Z_critical * (sample_std/math.sqrt(sample_size))
print(margin_of_error)

81.72918395711481
```

```
confidence_interval = (sample_mean - margin_of_error,
                       sample_mean + margin_of_error)
print(confidence_interval)
```

```
(102.72081604288518, 266.1791839571148)
```

```
#observations from SAMPLING DISTRIBUTIONS are:
print("mean values of sample is :",sample_mean)
print("standard deviation of sample is:",sample_std)
print("Z_crirical value is:",Z_critical)
print("Z_stat value is:",Z_stat)
print("margin of error is:",margin_of_error)
print("confifence interval is:",confidence_interval)
print("standard error is:",SE)
```

```
mean values of sample is : 184.45
standard deviation of sample is: 186.48507065968408
Z_crirical value is: 1.959963984540054
Z_stat value is: 4.4233325165476804
margin of error is: 81.72918395711481
confifence interval is: (102.72081604288518, 266.1791839571148)
standard error is: 41.69932947839052
```

```
#hypothesis testing
#t_test
#Z_test
#chisquare_test


from scipy import stats
from scipy.stats import t
from scipy.stats import ttest_1samp
from scipy.stats import chi2
from scipy.stats import chisquare



#t_test
accept="averagely a player contribute about 100+runs for his team"
reject="averagely a player does not contribute about 100+runs for his team"

avgt=df.Runs.sample(20,random_state=1)

avgt.mean=df["Runs"].mean()
avgt.mean

142.2962962962963

alphat=0.05
DOFt=19

pt=1-alphat/2
pt

0.975

critical_value_t=t.ppf(pt,DOFt)
print("critical value is:",critical_value_t)

critical value is: 2.093024054408263

t_statistic,p_value_t=ttest_1samp(avgt,avgt.mean)
print("t_statistic value is:",t_statistic)
print("p_value is:",p_value_t)

t_statistic value is: -0.6909694261046706
p_value is: 0.497940593993245

if (p_value_t<0.05):
    print("we reject null hypothesis")
    print(reject)
```

```
else:
   print("we accept null hypothesis")
   print(accept)
```

```
we accept null hypothesis
averagely a player contribute about 100+runs for his team
```

```
avgz=df.Runs.sample(30,random_state=4)
```

```
meanz=avgz.mean()
stdz=avgz.std()
```

```
meanz
```

```
219.5
```

```
stdz
```

```
172.81637653879912
```

```
z_critical_z=stats.norm.ppf(q=0.075)
N=30
SE=stdz/np.sqrt(N)
z_stat_z=(df['Runs'].mean()-meanz)/SE
print("Z_critical value is:",z_critical_z)
print("Z_stat value is:",z_stat_z)
```

```
Z_critical value is: -1.4395314709384563
Z_stat value is: -2.446886741197814
```

```
if -(z_critical_z)<z_stat_z>+(z_critical_z):
   print("we reject null hypothesis")
   print(reject)
else:
   print("we accept null hypothesis")
   print(accept)
```

```
we accept null hypothesis
averagely a player contribute about 100+runs for his team
```

```
#chisquare test
accept_c="all runs scored by highest player have good strike rate"
reject_c="all runs scored by highest player do not have good strike rate"

data_table=pd.crosstab(df.Runs.sample(50,random_state=1),df.SR.sample(50,random_s
tate=1))

alpha=0.05
COF=1

p=1-alpha/2

critical_value_c=chi2.ppf(p,COF)
print("critical value is:",critical_value_c)

critical value is: 5.023886187314888

chi2_statistic,p_c_value=chisquare([5,5])
print("statistic value is:",chi2_statistic)
print("p_value is:",p_c_value)

statistic value is: 0.0
p_value is: 1.0

if -(critical_value_c)<chi2_statistic>+(critical_value_c):
   print("we reject null hypothesis")
   print(reject_c)
else:
   print("we accept null hypothesis")
   print(accept_c)
```

```
we accept null hypothesis
all runs scored by highest player have good strike rate
```

```
#module 4(machine learning(supervised))


from sklearn.preprocessing import LabelEncoder
lb=LabelEncoder()


df['Player']=lb.fit_transform(df['Player'])
df['HS']=lb.fit_transform(df['HS'])
df['Avg']=lb.fit_transform(df['Avg'])


from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score


X=df.drop(['Runs'],axis=1)
Y=df['Runs']


X_train,X_test,Y_train,Y_test =train_test_split(X,Y,test_size=0.3,random_state=100)


sc=StandardScaler()


X_train=sc.fit_transform(X_train)
X_test=sc.fit_transform(X_test)
```

```
X_train=pd.DataFrame(X_train,columns=X.columns)
X_test=pd.DataFrame(X_test,columns=X.columns)


model=LogisticRegression()

model.fit(X_train,Y_train)

LogisticRegression()

pred_log=model.predict(X_test)
pred_log
```

```
array([161, 487,  4, 483,   2, 161,   7, 253,  12, 863,   4, 161, 483,
       258,  14, 230, 161,  2, 314,  22,  40,   7, 258,  18,  91,   2,
       341,   7, 183, 340,   2,  22, 458, 230, 314,   2,   2,  43, 376,
       341,   5,  5, 191,  12,  18,   7,  18,   4, 161])
```

```
print("accuracy score is:",accuracy_score(Y_test,pred_log))
```

```
accuracy score is: 0.061224489795918366
```

```
#decisiontree

from sklearn.tree import DecisionTreeClassifier

DT=DecisionTreeClassifier(criterion='gini',splitter='best',max_features=3,

max_depth=3)
DT.fit(X_train,Y_train)
DecisionTreeClassifier(max_depth=3,max_features=3)

DecisionTreeClassifier(max_depth=3, max_features=3)

DT_pred=DT.predict(X_test)

print(accuracy_score(Y_test,DT_pred))
```

```
0.02040816326530612
```

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import RandomForestClassifier

rf=RandomForestClassifier(n_estimators=100,criterion='entropy',max_depth=3)
rf.fit(X_train,Y_train)

RandomForestClassifier(criterion='entropy', max_depth=3)

rf_pred=rf.predict(X_test)
print(accuracy_score(Y_test,rf_pred))
```

```
0.12244897959183673
```

```
from sklearn.svm import SVC
svm=SVC(C=10,kernel='linear')

svm.fit(X_train,Y_train)

SVC(C=10, kernel='linear')

svm_pred=svm.predict(X_test)
print(accuracy_score(Y_test,svm_pred))
```

```
0.10204081632653061
```

```
from sklearn.neighbors import KNeighborsClassifier

neighbors=np.arange(1,9)
train_accuracy=np.empty(len(neighbors))
test_accuracy=np.empty(len(neighbors))

for i,k in enumerate(neighbors):
   knn=KNeighborsClassifier(n_neighbors=k)
   knn.fit(X_train,Y_train)

test_accuracy[i]=knn.score(X_test,Y_test)
knn2=KNeighborsClassifier(n_neighbors=2)
knn3=KNeighborsClassifier(n_neighbors=3)
knn4=KNeighborsClassifier(n_neighbors=4)
knn5=KNeighborsClassifier(n_neighbors=5)

knn3.fit(X_train,Y_train)
knn4.fit(X_train,Y_train)
knn5.fit(X_train,Y_train)

KNeighborsClassifier()

pred_knn=knn3.predict(X_test)
print(accuracy_score(Y_test,pred_knn))
```

```
0.10204081632653061
```

```
from sklearn.naive_bayes import GaussianNB,BernoulliNB
gaussian_nb=GaussianNB()
gaussian_nb.fit(X_train,Y_train)

GaussianNB()

bernoulli_nb=BernoulliNB()
bernoulli_nb.fit(X_train,Y_train)

BernoulliNB()

pred_gnb=gaussian_nb.predict(X_test)
print(accuracy_score(Y_test,pred_gnb))
```

```
0.02040816326530612
```

```
pred_bnb=bernoulli_nb.predict(X_test)
print(accuracy_score(Y_test,pred_bnb))
```

```
0.061224489795918366
```

```
#results of algorithms
print("accuracy of ML algorithms")
print("Logistic regression",round(100*accuracy_score(Y_test,pred_log)),"%")
print("Decision Tree",round(100*accuracy_score(Y_test,DT_pred)),"%")
print("Random forest",round(100*accuracy_score(Y_test,rf_pred)),"%")
print("svm",round(100*accuracy_score(Y_test,svm_pred)),"%")
print("KNN",round(100*accuracy_score(Y_test,pred_knn)),"%")
print("Bernoulli naive bayes",round(100*accuracy_score(Y_test,pred_bnb)),"%")
print("Gaussian naive bayes",round(100*accuracy_score(Y_test,pred_bnb)),"%")
```

```
accuracy of ML algorithms
Logistic regression 6 %
Decision Tree 2 %
Random forest 12 %
svm 10 %
KNN 6 %
Bernoulli naive bayes 6 %
Gaussian naive bayes 6 %
```