

**HMR INSTITUTE OF TECHNOLOGY & MANAGEMENT**

**HAMIDPUR , DELHI - 110036**

**Affiliated to**

**GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY**

**Sector - 16C Dwarka, Delhi - 110075, India 2008-12**

# Summer Training Project Report On IPL DATA ANALYSIS

Submitted in partial fulfilment of the requirements for

The award of the degree of

**Bachelor of Technology  
In**

**(ELECTRONICS AND COMMUNICATION ENGINEERING)**

**Guide(s): Submitted by:**

**Mr. Chandan SWATI SHUKLA**

**Roll No: 44913302817**

**DECLARATION**

We, students of B.Tech (ECE 4th sem) hereby declare that the minor project entitled “IPL DATA ANALYSIS” which is submitted to the Department of ECE, HMR Institute of Technology & Management, Hamidpur Delhi, affiliated to Guru Gobind Singh Indraprastha University, Dwarka(New Delhi) in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in ECE, has not been previously formed the basis for the award of any degree, diploma or other similar title or recognition. The list of member(s)involved in the project is listed below: -

* Yash Chauhan(44913302817)

This is to certify that the above statement made by the candidate(s) is correct to the best of my knowledge.

**Mr. A.K. Sharma**

**(Head of Department)**

**(Assistant Professor of Electronics Department)**

**HMRITM Hamidpur, New Delhi-110036**

**CERTIFICATE**

This is to certify that the project entitled **“ IPL DATA ANALYSIS ”** is the bonafide work carried out by **YASH CHAUHAN** student of **B.Tech , HMR Institute of Technology & Management, Hamidpur Delhi, affiliated to Guru Gobind Singh Indraprastha University, Dwarka(New Delhi) ,** during the year **2ND YEAR,** in partial fulfillment of the requirements for the award of the Degree of Bachelor of **Electronic and Communication Engineering** and that the project has not formed the basis for the award previously of any degree, diploma, associateship, fellowship or any other similar title.

**Signature of the Guide : Chandan**

**Place: HMRITM**

**Date: 06 Sept,2019**

**ACKNOWLEDGEMENT**

**It is my pleasure to be indebted to various people, who directly or indirectly contributed in the development of this work and who influenced my thinking, behavior, and acts during the course of study.**

**I express my sincere gratitude to V C PANDEY, worthy Principal for providing me an opportunity to undergo summer training at HMR Institute of Technology**

**I am thankful to Mr Chandan for his support, cooperation, and motivation provided to me during the training for constant inspiration, presence and blessings.**

**I also extend my sincere appreciation to Mr/Ms Chandan who provided his valuable suggestions and precious time in accomplishing my project report.**

**Lastly, I would like to thank the almighty and my parents for their moral support and my friends with whom I shared my day-to-day experience and received lots of suggestions that improved my quality of work.**

**Name of the student:**

* **Swati shukla**
* **Neha**
* **Aashish baliyan**
* **Yash chauhan**

**Contents**

## 1. Cover Page & Title page

## 2. Declaration by the Student

## 3. Certificate by the Guide

## 4. Acknowledgment

## 5.Team details

## 6. Abstract

## 7. Table of Contents

## 8. Codes

## 9. Appendices

## 10. References

**Team Details**

**SWATI SHUKLA**

**Roll No: 44913302817**

**Contribution: Coding**

**NEHA**

**Roll No: 42713302817**

**Contribution: Coding**

**AASHISH BALIYAN**

**Roll No:45713302817**

**Contribution: Report File**

**YASH CHAUHAN**

**Roll No: 01613302817**

**Contribution: Report file**

**Abstract**

This project aims at IPL data analysis according to their choices of teams. Prediction uses machine learning and go over all of the options and learn what we like and then recommend us best option that would we like. Data Science / Analytics is all about finding valuable insights from the given dataset. In short, Finding answers that could help business. In this tutorial, We will see how to get started with Data Analysis in Python. The Python packages that we use in this notebook are: numpy, pandas, matplotlib, and seaborn

Since usually such tutorials are based on in-built datasets like iris, It becomes harder for the learner to connect with the analysis and hence learning becomes difficult. To overcome this, The dataset that we use in this notebook is IPL (Indian Premier League) Dataset posted on Kaggle Datasets sourced from cricsheet. IPL is one of the most popular cricket tournaments in the world, thus the problems we try to solve and the questions that we try to answer should be familiar to anyone who knows Cricket.

**Algorithm Used**

Linear Regression:

In statistics, linear regression is a linear approach to modelling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables). The case of one explanatory variable is called simple linear regression. For more than one independent variable, the process is called multiple linear regression. This term is distinct from multivariate linear regression, where multiple correlated dependent variables are predicted, rather than a single scalar variable.

In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models. Most commonly, the conditional mean of the response given the values of the explanatory variables (or predictors) is assumed to be an affine function of those values; less commonly, the conditional median or some other quantile is used. Like all forms of regression analysis, linear regression focuses on the conditional probability distribution of the response given the values of the predictors, rather than on the joint probability distribution of all of these variables, which is the domain of multivariate analysis.

Linear regression was the first type of regression analysis to be studied rigorously, and to be used extensively in practical applications.This is because models which depend linearly on their unknown parameters are easier to fit than models which are non-linearly related to their parameters and because the statistical properties of the resulting estimators are easier to determine.

Linear regression has many practical uses. Most applications fall into one of the following two broad categories:

* If the goal is prediction, or forecasting, or error reduction, linear regression can be used to fit a predictive model to an observed data set of values of the response and explanatory variables. After developing such a model, if additional values of the explanatory variables are collected without an accompanying response value, the fitted model can be used to make a prediction of the response.
* If the goal is to explain variation in the response variable that can be attributed to variation in the explanatory variables, linear regression analysis can be applied to quantify the strength of the relationship between the response and the explanatory variables, and in particular to determine whether some explanatory variables may have no linear relationship with the response at all, or to identify which subsets of explanatory variables may contain redundant information about the response.

Fig. 3.1 (Figure Description)

3.2. Assumptions

Standard linear regression models with standard estimation techniques make a number of assumptions about the predictor variables, the response variables and their relationship. Numerous extensions have been developed that allow each of these assumptions to be relaxed (i.e. reduced to a weaker form), and in some cases eliminated entirely. Generally these extensions make the estimation procedure more complex and time-consuming, and may also require more data in order to produce an equally precise model.

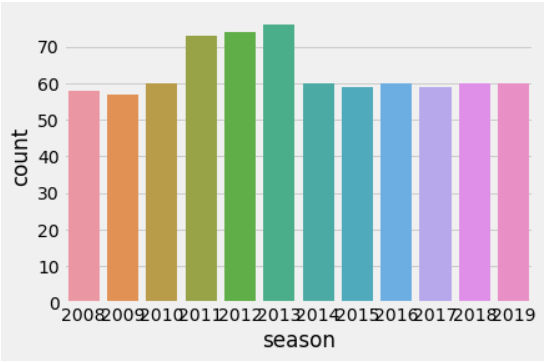


Fig. 3.2 (Figure Description)

Example of a cubic polynomial regression, which is a type of linear regression.

The following are the major assumptions made by standard linear regression models with standard estimation techniques (e.g. ordinary least squares):

* Weak exogeneity. This essentially means that the predictor variables x can be treated as fixed values, rather than random variables. This means, for example, that the predictor variables are assumed to be error-free that is, not contaminated with measurement errors. Although this assumption is not realistic in many settings, dropping it leads to significantly more difficult errors in variables models.
* Linearity. This means that the mean of the response variable is a linear combination of the parameters (regression coefficients) and the predictor variables. Note that this assumption is much less restrictive than it may at first seem. Because the predictor variables are treated as fixed values (see above), linearity is really only a restriction on the parameters. The predictor variables themselves can be arbitrarily transformed, and in fact multiple copies of the same underlying predictor variable can be added, each one transformed differently. This trick is used, for example, in polynomial regression, which uses linear regression to fit the response variable as an arbitrary polynomial function (up to a given rank) of a predictor variable. This makes linear regression an extremely powerful inference method. In fact, models such as polynomial regression are often "too powerful", in that they tend to overfit the data. As a result, some kind of regularization  must typically be used to prevent unreasonable solutions coming out of the estimation process. Common examples are ridge regression and lasso regression. Bayesian linear regression can also be used, which by its nature is more or less immune to the problem of overfitting. (In fact, ridge regression and lasso regression can both be viewed as special cases of Bayesian linear regression, with particular types of prior distribution  placed on the regression coefficients.)
* Constant variance (a.k.a. homoscedasticity). This means that different values of the response variables have the same variance in their errors, regardless of the values of the predictor variables. In practice this assumption is invalid (i.e. the errors are heteroscedastic) if the response variable can vary over a wide scale. In order to check for heterogeneous error variance, or when a pattern of residuals violates model assumptions of homoscedasticity (error is equally variable around the 'best-fitting line' for all points of x), it is prudent to look for a "fanning effect" between residual error and predicted values. This is to say there will be a systematic change in the absolute or squared residuals when plotted against the predictive variables. Errors will not be evenly distributed across the regression line. Heteroscedasticity will result in the averaging over of distinguishable variances around the points to get a single variance that is inaccurately representing all the variances of the line. In effect, residuals appear clustered and spread apart on their predicted plots for larger and smaller values for points along the linear regression line, and the mean squared error for the model will be wrong. Typically, for example, a response variable whose mean is large will have a greater variance than one whose mean is small. For example, a given person whose income is predicted to be $100,000 may easily have an actual income of $80,000 or $120,000 (a standard deviation of around $20,000), while another person with a projected income of $10,000 is unlikely to have the same $20,000 standard deviation, which would imply their actual income would vary anywhere between -$10,000 and $30,000. (In fact, as this shows, in many cases—often the same cases where the assumption of normally distributed errors fails—the variance or standard deviation should be predicted to be proportional to the mean, rather than constant.) Simple linear regression estimation methods give less precise parameter estimates and misleading inferential quantities such as standard errors when substantial heteroscedasticity is present. However, various estimation techniques and heteroscedasticity-consistent standard errors can handle heteroscedasticity in a quite general way.bayesian linear regression techniques can also be used when the variance is assumed to be a function of the mean. It is also possible in some cases to fix the problem by applying a transformation to the response variable (e.g. fit the logarithm of the response variable using a linear regression model, which implies that the response variable has a log normal distribution  rather than a normal logarithm).
* Independence of errors. This assumes that the errors of the response variables are uncorrelated with each other. (Actual statistical independence is a stronger condition than mere lack of correlation and is often not needed, although it can be exploited if it is known to hold.) Some methods are capable of handling correlated errors, although they typically require significantly more data unless some sort of regularization is used to bias the model towards assuming uncorrelated errors Bayseian linear regression  is a general way of handling this issue.
* Lack of perfect multicollinearity in the predictors. For standard least squares estimation methods, the design matrix X must have full  otherwise, we have a condition known as perfect multicollinearity in the predictor variables. This can be triggered by having two or more perfectly correlated predictor variables (e.g. if the same predictor variable is mistakenly given twice, either without transforming one of the copies or by transforming one of the copies linearly). It can also happen if there is too little data available compared to the number of parameters to be estimated (e.g. fewer data points than regression coefficients). In the case of perfect multicollinearity, the parameter vector β will be non identifiable it has no unique solution. At most we will be able to identify some of the parameters, i.e. narrow down its value to some linear subspace of R. Methods for fitting linear models with multicollinearity have been developed some require additional assumptions such as "effect sparsity"—that a large fraction of the effects are exactly zero.  
  Note that the more computationally expensive iterated algorithms for parameter estimation, such as those used in, do not suffer from this problem.

Beyond these assumptions, several other statistical properties of the data strongly influence the performance of different estimation methods:

* The statistical relationship between the error terms and the regressors plays an important role in determining whether an estimation procedure has desirable sampling properties such as being unbiased and consistent.
* The arrangement, or probability distribution of the predictor variables x has a major influence on the precision of estimates of β. sampling and design of experiments  are highly developed subfields of statistics that provide guidance for collecting data in such a way to achieve a precise estimate of β.

.

**Libraries Used**

**Numpy**

NumPy is a library for the [Python programming language](https://en.wikipedia.org/wiki/Python_(programming_language)), adding support for large, multi-dimensional [arrays](https://en.wikipedia.org/wiki/Array_data_structure) and [matrices](https://en.wikipedia.org/wiki/Matrix_(math)) , along with a large collection of [ematical](https://en.wikipedia.org/wiki/High-level_programming_language) [functions](https://en.wikipedia.org/wiki/Function_(mathematics)) to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by [Jim Hugunin](https://en.wikipedia.org/wiki/Jim_Hugunin) with contributions from several other developers. In 2005, [Travis Oliphant](https://en.wikipedia.org/wiki/Travis_Oliphant) created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is [open-source software](https://en.wikipedia.org/wiki/Open-source_software) and has many contributors.

* History

The Python programming language was not initially designed for numerical computing, but attracted the attention of the scientific and engineering community early on, so that a special interest group called matrix-sig was founded in 1995 with the aim of defining an array computing package. Among its members was Python designer and maintainer, who implemented extensions to [Python's syntax](https://en.wikipedia.org/wiki/Python_syntax_and_semantics) (in particular the indexing syntax) to make array computing easier.There was a desire to get Numeric into the Python standard library, but Guido van Rossum decided that the code was not maintainable in its state then. In early 2005, NumPy developer [Travis Oliphant](https://en.wikipedia.org/wiki/Travis_Oliphant) wanted to unify the community around a single array package and ported Numarray's features to Numeric, releasing the result as NumPy 1.0 in 2006.[[6]](https://en.wikipedia.org/wiki/NumPy) This new project was part of SciPy. To avoid installing the large SciPy package just to get an array object, this new package was separated and called NumPy. Support for Python 3 was added in 2011 with NumPy version 1.5.0.In 2011, PyPy started development on an implementation of the NumPy API for PyPy. It is not yet fully compatible with NumPy.

**Pandas:**

In [computer programming](https://en.wikipedia.org/wiki/Computer_programming), pandas is a [software library](https://en.wikipedia.org/wiki/Software_library) written for the [Python programming language](https://en.wikipedia.org/wiki/Python_(programming_language)) for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and [time series](https://en.wikipedia.org/wiki/Time_series). It is [free software](https://en.wikipedia.org/wiki/Free_software) released under the [three-clause BSD license](https://en.wikipedia.org/wiki/3-clause_BSD_license). The name is derived from the term "[panel data](https://en.wikipedia.org/wiki/Panel_data)", an [econometrics](https://en.wikipedia.org/wiki/Econometrics) term for data sets that include observations over multiple time periods for the same individuals.

**Pandas** is the most popular **python** library that is used for data analysis. It provides highly optimized performance with back-end source code is purely written in C or **Python**. We can analyze data in **pandas** with: Series

**Matplotlib**

Matplotlib. Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+.There is also a [procedural](https://en.wikipedia.org/wiki/Procedural_programming) "pylab" interface based on a [state machine](https://en.wikipedia.org/wiki/State_machine) (like [OpenGL](https://en.wikipedia.org/wiki/OpenGL)), designed to closely resemble that of [MATLAB](https://en.wikipedia.org/wiki/MATLAB), though its use is discouraged.[[3]](https://en.wikipedia.org/wiki/Matplotlib#cite_note-3) [SciPy](https://en.wikipedia.org/wiki/SciPy) makes use of Matplotlib.

Matplotlib was originally written by [John D. Hunter](https://en.wikipedia.org/wiki/John_D._Hunter), has an active development community, and is distributed under a [BSD-style license](https://en.wikipedia.org/wiki/BSD_licenses). Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in August 2012, and further joined by Thomas Caswell.

**Seaborn:**

Statistical data visualization. Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Seaborn is a Python data visualization library based on [matplotlib](https://matplotlib.org/). 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](https://seaborn.pydata.org/introduction.html#introduction). Visit the [installation page](https://seaborn.pydata.org/installing.html#installing) to see how you can download the package. You can browse the [example gallery](https://seaborn.pydata.org/examples/index.html#example-gallery) to see what you can do with seaborn, and then check out the [tutorial](https://seaborn.pydata.org/tutorial.html#tutorial) and [API reference](https://seaborn.pydata.org/api.html#api-ref) to find out how.

To see the code or report a bug, please visit the [github repository](https://github.com/mwaskom/seaborn). General support issues are most at home on [stackoverflow](https://stackoverflow.com/), where there is a seaborn tag.

**CODES**

**import** **pandas** **as** **pd**

matches = pd.read\_csv('matches.csv')

matches.head()

Matches.columns

*#no of matches in every season*

pd.value\_counts(matches['season'], sort=**False**)

pd.value\_counts(matches['season'])

*#no of times a team won*

pd.value\_counts(matches['winner'])

*#no of times a player was awarded man of match award*

pd.value\_counts(matches['player\_of\_match']).head()

*#find the winner of the ipl seasons*

final\_matches = matches.drop\_duplicates('season', keep='last')

pd.value\_counts(final\_matches['winner'])

*#find winner of every season*

temp\_df = final\_matches.reset\_index()

temp\_df[['season','winner']]

*#find win\_by\_runs margin and exclude where win\_by\_runs = 0*

pd.value\_counts(matches['win\_by\_runs']).head()

temp\_df = matches[matches['win\_by\_runs'] != 0]

pd.value\_counts(temp\_df['win\_by\_runs']).head()

*#find how many matches a team won in a season*

seasons = pd.unique(matches['season'])

winners\_in\_each\_season = []

**for** season **in** seasons:

matches\_in\_a\_season = matches[matches['season']==season]

winners\_in\_each\_season.append(pd.value\_counts(matches\_in\_a\_season['winner']))

**for** i **in** range(len(winners\_in\_each\_season)):

print(f'Season - **{seasons[i]}**')

print(winners\_in\_each\_season[i])

print('**\n**')

seasons

*#find matches where CSK won by more than 50 runs*

*#Print three columns - team1, team2, win\_by\_runs margin*

matches[(first condition) & (second condition)]

*#1. find 5 players who took most number of catches*

*#2. find 5 players with most wickets*

*#3. find the player who was given out max. no of times*

*#by the following way -*

*# . - caught*

*# . - bowled*

*# . - run out*

*# . - lbw*

*# . - caught and bowled*

*# . - stumped*

*# . - hit wicket*

*# . - obstructing the field*

*#1. find 5 players who took most number of catches*

*#2. find 5 players with most wickets*

*#3. find the player who was given out max. no of times*

*#by the following way -*

*# . - caught*

*# . - bowled*

*# . - run out*

*# . - lbw*

*# . - caught and bowled*

*# . - stumped*

*# . - hit wicket*

*# . - obstructing the field*

pd.unique(deliveries['dismissal\_kind'])

deliveries.columns

deliveries.shape

temp\_df = deliveries['dismissal\_kind']

temp\_df.isna()

deliveries.max()

deliveries = deliveries.dropna()

deliveries

temp\_df = deliveries[['bowler', 'dismissal\_kind']].dropna()

pd.unique(deliveries['dismissal\_kind'])

temp\_df = temp\_df[(temp\_df['dismissal\_kind']!='run out')&(temp\_df['dismissal\_kind']!='retired hurt')&(temp\_df['dismissal\_kind']!='obstructing the field')]

pd.value\_counts(temp\_df['bowler']).head()

deliveries[deliveries['dismissal\_kind']=='hit wicket']

toss\_decision = pd.value\_counts(matches['toss\_decision'])

*#labels = ['field', 'bat']*

toss\_decision

type(toss\_decision)

labels = []

**for** key **in** toss\_decision.keys():

labels.append(key)

labels

**import** **matplotlib.pyplot** **as** **plot**

plot.pie(toss\_decision, labels=labels, autopct='%.f**%%**', startangle=90, radius=0.8)

deliveries.columns

x = deliveries[deliveries['dismissal\_kind']

!='obstructing the field']

dismissal\_kind = pd.value\_counts(

x['dismissal\_kind'],sort=**False**)

dismissal\_kind = dismissal\_kind.sort\_index()

labels = []

**for** key **in** dismissal\_kind.keys():

labels.append(key)

plot.pie(dismissal\_kind, radius=1.5,

autopct='**%.2f%%**', pctdistance=1,

startangle=90, counterclock=**False**,

explode=[0.5,0,0,0,0,0,0,0], shadow=**True**)

plot.legend(labels)

matches.head()

win = matches[matches['toss\_winner'] == matches['winner']]

win\_count = pd.value\_counts(win['toss\_decision'])

lose = pd.value\_counts(matches[

matches['toss\_winner'] != matches['winner']]

['toss\_decision'])

win\_count

labels = ['field\_win', 'bat\_win', 'field\_lose', 'bat\_lose']

data = []

**for** value **in** win\_count:

data.append(value)

**for** value **in** lose:

data.append(value)

data

plot.pie(data, labels=labels, autopct='**%.2f%%**', startangle=90, counterclock=**False**)

plot.hist(matches['season'],color='yellow',rwidth=0.9,bins=[2008,2009,2010,2011,2012,2013,2014,2015,2016,2017,2018,2019])

seasons = pd.unique(matches['season'])

seasons.sort()

plot.bar(seasons,pd.value\_counts(matches['season'],sort=**False**),color='yellow')

pd.value\_counts(matches['season'],sort=**False**)

seasons = pd.unique(matches['season'])

seasons.sort()

seasons

pd.value\_counts(matches['season'],sort=**False**)

**from** **matplotlib** **import** style

plot.bar(seasons,pd.value\_counts(matches['season'],sort=**False**))

*#style.use('ggplot')*

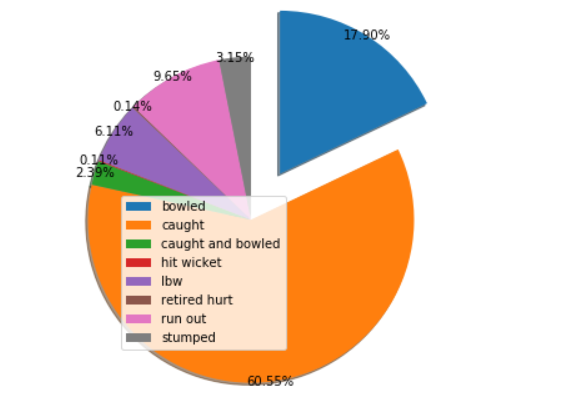
style.use('fivethirtyeight')

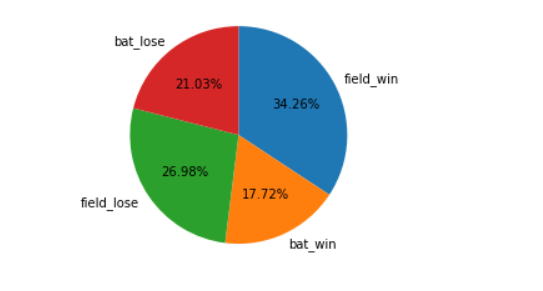
*#style.use('classic')*

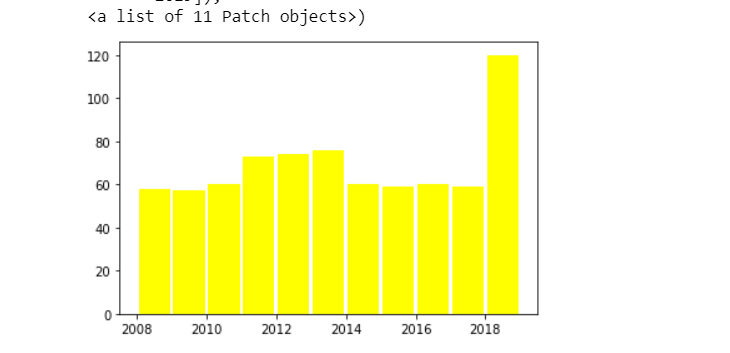
**import** **seaborn** **as** **sns**

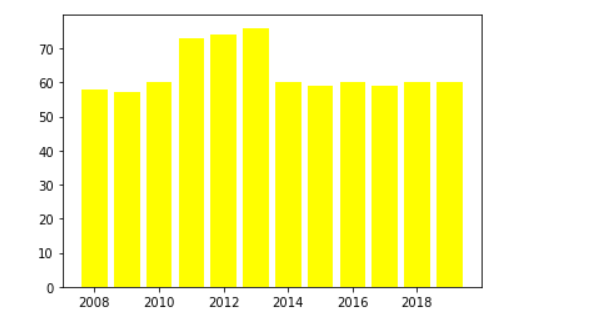
sns.countplot(x=matches['season'], data=matches)

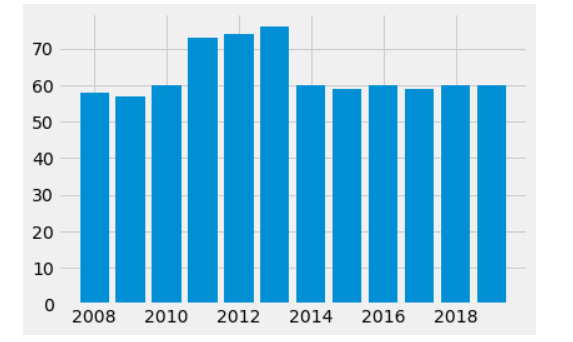
**OUTPUTS**

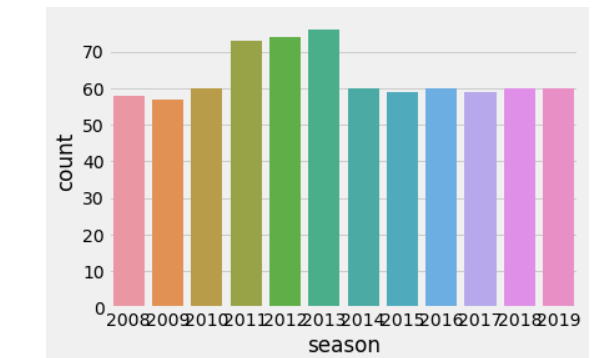












**Summary**

Data analysis is the process of evaluating data using analytical and statistical tools to discover useful information and aid in business decision making. There are several data analysis methods including data mining, text analytics, business intelligence and data visualization. In this we are analysing the data the data in every aspect such as in the form of runs or in the form of form of matches and many more so that we can predict the assumption for the future values.

**References**

[1]https://www.kaggle.com/nulldata/begin-your-data-analysis-in-python-with-ipl-data

[2]https://towardsdatascience.com/analysing-ipl-data-to-begin-data-analytics-with-python-5d2f610126a

[3]https://github.com/amrrs/iq18\_workshop/blob/master/data\_analysis/Data%20Analysis%20in%20Python%20with%20IPL%20Dataset.ipynb

[4] https://www.meetup.com/BangPypers/events/qxlhwkyxfbwb/