MAJOR PROJECT REPORT

MAKAUT EVEN SEMESTER 2019-2020

MASTER OF COMPUTER APPLICATION

Lok Sabha 2009 vs 2014 vs 2019 Analysis and Prediction

UNDER THE SUPERVISION OF MR.SAUBHIK GOSWAMI

Done By:

NAME	ROLL NO
PRITAM MONDAL	14201017015
SAYAK PAL	14201017009
RINJU DANA	14201017014

MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY

Certificate

This is to certify that the project entitled "**Lok Sabha 2009 vs 2014 vs 2019 Analysis and Prediction** " has been prepared according to the regulation of the degree of Masters in Computer Application (MCA) under the University of Maulana Abul Kalam Azad University of Technology. The project is submitted by:

Students of Masters of Computer Application (MCA), 3rd-year 6th semester, of ***MEGHNAD SAHA INSTITUTE OF TECHNOLOGY*** (Affiliated to the Maulana Abul Kalam Azad University of Technology) have fulfilled the requirement for submission of this.

The whole procedure has been carried out under my supervision and guidance. I have gone through this project and have seen that it is fulfilling the requirements of Major Project under MAKAUT WB.

Signature of Students

Sayak Pal	Pritam Mondal	Rinju Dana
SAYAK PAL	PRITAM MONDAL	RINJU DANA

This is to certify that the above statement made by the students is correct to the best of my knowledge.

Date: - 22.06.2020

Signature of Guide
Mr. Saubhik Goswami
Department MCA

Signature of External Examiner

Head of the Department
Aparna Dutta
HOD-MCA

❖ ACKNOWLEDGMENT

Apart from the efforts of self-being, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project. We would like to show our greatest appreciation to WEBSKITTERS and respect to Sir Mr. Saubhik Goswami. We feel motivated and encouraged every time we attend his meeting. Without his encouragement and guidance, this project would not have materialized. The guidance and support received from all the members who contributed and who are contributing to this project were vital for the success of the project. We are grateful for the constant support and help.

We pay our sincere gratitude to the Head of the Department, Aparna Dutta for his constant support.

We have given efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. We would like to extend our sincere thanks to all of them.

We are highly indebted to WEBSKITTERS and respected Sir Saubhik Goswami for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

We would like to express our gratitude towards our parents & member of WEBSKITTERS and respected Sir Saubhik Goswami for their kind cooperation and encouragement which helped us in the completion of this project.

We would like to express our special gratitude and thanks to industry persons for giving us such attention and time.

Our thanks and appreciations also go to each one of the members in developing the project and people who have willingly helped us out with their abilities.

❖ CONTENTS

❖ Overview of the system

➢ Preface

Learning comes from doing. To learn something one has to go through Practical conditions. Recognizing this fact, the university has made it essential for MCA (MASTER OF COMPUTER APPLICATION) students to undergo major project Training. During this period, the students learn about the functioning of the organization and the actual business environment. Also, this training helps the student how to implement the theoretical knowledge into practical life in our day to day life.

This project was undertaken at **MEGHNAD SAHA INSTITUTE OF TECHNOLOGY, KOLKATA** during industrial training engineering to automate the system. The project is named as Lok Sabha 2009 vs 2014 vs 2019 Analysis and Prediction**.**

The purpose of this report is to assemble under one cover a sufficient body of knowledge about the management and development of a successful software engineering project. The following quotes outline the basic idea behind this technical report.

This report is about the adaptation of the techniques of project development and reflects the practice and methods of software engineering project this report is intended for:

- *Project managers*—the report delivers the necessary information of the process a software development project
- *Project coordinators* —the tutorial presents the state of the practice in software development and management techniques.
- *Software engineers, programmers, analysts, and other computer personnel* — the report contains a general description of—and problems in—software engineering project development, plus several methodologies and techniques for managing a software development project.

➢ Objective

- Develop an in-depth understanding of the key technologies in data science and business analytics: data mining, machine learning, visualization techniques, predictive modeling, and statistics.
- Practice problem analysis and decision-making.
- Gain practical, hands-on experience with statistics programming languages and big data tools through coursework and applied research experiences.
- Apply quantitative modeling and data analysis techniques to the solution of real-world business problems, communicate findings, and effectively present results using data visualization techniques.
- Demonstrate knowledge of statistical data analysis techniques utilized in business decision making.
- Apply principles of Data Science to the analysis of business problems. Apply algorithms to build machine intelligence.
- Demonstrate the use of teamwork, leadership skills, decision making, and organization theory.

❖ INTRODUCTION OF THE PROJECT

The project titled Lok Sabha 2009 vs 2014 vs 2019 Analysis and Prediction is a machine learning-based project where a detailed analysis has been done on the previous General Election of India of the year 2009, 2014, and 2019 respectively. The project has been done using predictive analysis methods with the algorithms related to predictive analysis. Here we analyse the data respectively on the candidate's list and electors list, following a look at the major historic change in majority in 2014 Lok Sabha from the preceding year 2009. A detailed comparison is done keeping in respect to the election results of the year 2009, 2014 and 2019. Through this model we look forward to the following points respectively :-

- **Implementation of Democracy:**

  If an individual has this app, then he or she understands which candidate stands form where and what amount of effort they are giving. We can know about education qualification, financial background, assets, etc. Any common individual can view the detailed analysis of the previous vote even Alliance wise, Candidate wise and State-wise. The simplicity of data visualization offers more understand ability and ease of use. On considering the detailed analysis of Lok Sabha vote 2009, 2014, and 2019 how people will vote in 2021. So, people can less worry about the propagandas of the biased news channels and give their votes to worthy candidates. Even as a point of view of a candidate, they can choose the right alliance group based on the analysis. They can get the prediction report about the number of votes upon total votes, how many seats they can get, candidate wise vote, state-wise votes, etc. These are the advantages for candidates before voting.

- **To Choose a Better Alliance of Political Party:**

  The project will provide a better way of getting a detailed result from the previous elections that have taken place and will be a help to the political parties of creating alliances before the elections. The number of seats and percentage of votes shared between the regional parties will help to grow alliances with the national parties across the country.

- **Choose a Better Candidate by Comparing the Previous Data:**

  It will be an eye-opener for the electors to choose the candidates based on their records. They can analyze into each candidate’s data, the amount of work done for the past years, they can simply check any criminal records concerning any candidates, and thus can get an idea or determine to vote for his or her choice without getting biased on media reports.

- **Properly Implement the Voting Rights to Build a Better Society:**

  Electing better candidates from the constituencies will help the constituencies and its electors to grow shortly, so through this app, it will help to analyze to choose a better candidate, which later will lead to building a better society, opting for linearization

➢ Scope of the Project

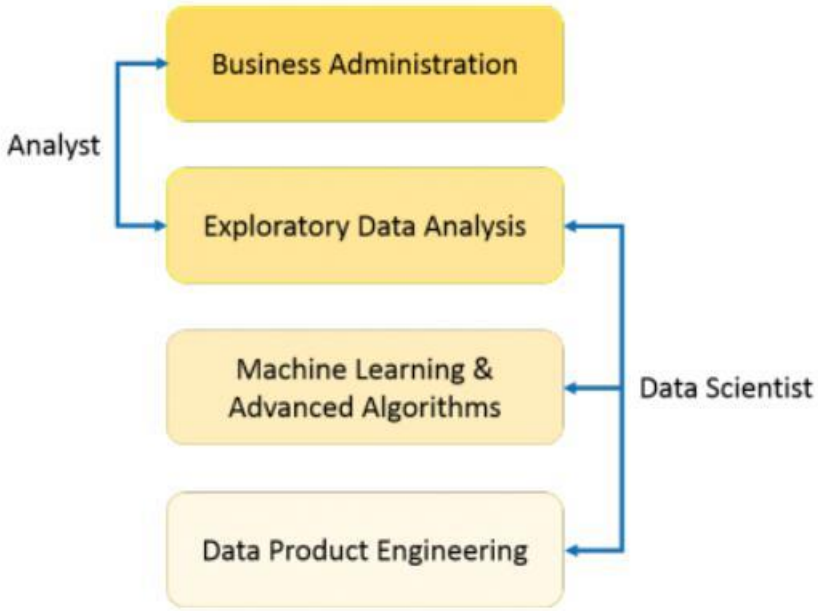
- The datasets of Lok Sabha 2009, 2014, and 2019 have approximately 5000 of electors and candidates data.
- This model analyses and predicts each of the datasets respectively and produce the desired result.
- The model follows the predictive analysis algorithm to analyze the datasets. The predictive analysis models are as follows:-

  - ✓ Logistic Regression
  - ✓ Decision Tree
  - ✓ Random-Forest Classifier

➢ ABOUT DATA SCIENCE AND PYTHON

Data Science is a blend of various tools, algorithms, and machine learning principles to discover hidden patterns from the raw data. How is this different from what statisticians have been doing for years?

The answer lies in the difference between explaining and predicting.



In the above image, a Data Analyst usually explains what is going on by processing the history of the data. On the other hand, Data Scientists not only do exploratory analysis to discover insights from it but also use various advanced machine learning algorithms to identify the occurrence of a particular event in the future. A Data Scientist will look at the data from many angles, sometimes angles not known earlier.

So, Data Science is primarily used to make decisions and predictions making use of predictive causal analytics, prescriptive analytics (predictive plus decision science), and machine learning.

- **Predictive causal analytics:** If a model which can predict the possibilities of a particular event in the future, then its need to apply predictive causal analytics. Say, if it is providing money on credit, then the probability of customers making future credit payments on time is a matter of concern. Here, it can build a model that can perform predictive analytics on the payment history of the customer to predict if the future payments will be on time or not.

- **Prescriptive analytics:** If a model which has the intelligence of taking its own decisions and the ability to modify it with dynamic parameters, then credit payments on time is a matter of concern. Here, it can build a model that can perform predictive analytics on the payment history of the customer to predict if the future payments will be on time or not. This relatively new field is all about providing advice. In other terms, it not only predicts but suggests a range of prescribed actions and associated outcomes.

  The best example of this is Google's self-driving car. The data gathered by vehicles can be used to train self-driving cars. You can run algorithms on this data to bring intelligence to it. This will enable your car to make decisions like when to turn, which path to take when to slow down or speed up.

Machine Learning

Machine learning is the science of getting computers to act without being explicitly programmed. In the past decade, machine learning has given us self-driving cars, practical speech recognition, effective web search, and a vastly improved understanding of the human genome. Machine learning is effective machine learning techniques, and gain practice implementing them and getting them to work for yourself. More importantly, you'll learn about not only the theoretical underpinnings of so pervasive today that you probably use it dozens of times a day without knowing it. Many researchers also think it is the best way to make progress towards human-level AI. In this class, you will learn about the most learning but also gain the practical know-how needed to quickly and powerfully apply these techniques to new problems. Finally, you'll learn about some of Silicon Valley's best practices in innovation as it pertains to machine learning and AI. This course provides a broad introduction to machine learning, data mining, and statistical pattern recognition. Topics include: (i) **Supervised learning** (parametric/non- parametric algorithms, support vector machines, kernels, neural networks).
(ii) **Unsupervised learning** (clustering, dimensionality reduction, recommendations systems, deep learning). (iii) Best practices in machine learning (bias/variance theory; innovation process in machine learning and AI). The course will also draw from numerous case studies and applications so that you'll also learn how to apply learning algorithms to building smart robots (perception, control), text understanding (web search, anti-spam), computer vision, medical informatics, audio, database mining, and other areas.

- **Machine learning for making predictions** — if the transactional data of a finance company and need to build a model to determine the future trend, then machine learning algorithms are the best bet. This falls under the paradigm of supervised learning. It is called supervised because data have already based on machines. For example, a fraud detection model can be trained using a historical record of fraudulent purchases.

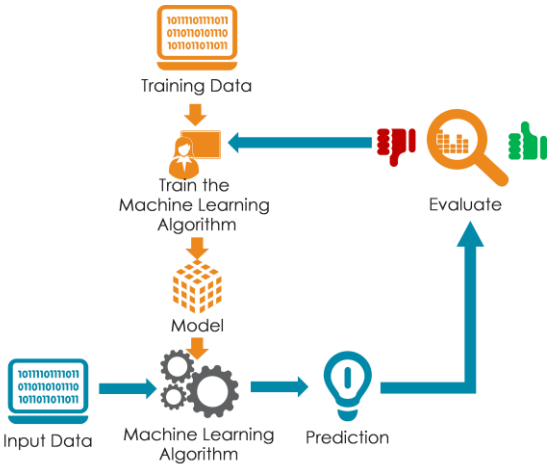
- **Machine learning for pattern discovery** — If you don't have the parameters based on which can make predictions, then you need to find out the hidden patterns within the dataset to be able to make meaningful predictions. This is nothing but the unsupervised model as you don't have any predefined labels for grouping. The most common algorithm used for pattern discovery is Clustering.

  Let's say you are working in a telephone company and you need to establish a network by putting towers in a region. Then, you can use the clustering technique to find those tower locations which will ensure that all the users receive optimum signal strength.

Working process of Machine Learning

Machine Learning algorithm is trained using a training data set to create a model. When new input data is introduced to the ML algorithm, it predicts based on the model.

The prediction is evaluated for accuracy and if the accuracy is acceptable, the Machine Learning algorithm is deployed. If the accuracy is not acceptable, the Machine Learning algorithm is trained again and again with an augmented training data set.



Types of Machine Learning

Machine learning is sub-categorized into three types:

Supervised Learning

Supervised Learning is the one, where you can consider the learning is guided by a teacher. We have a data-set that acts as a teacher and its role is to train the model or the machine. Once the model gets trained it can start making a prediction or decision when new data is given to it.

Unsupervised Learning

The model learns through observation and finds structures in the data. Once the model is given a data-set, it automatically finds patterns and relationships in the data-set by creating clusters in it. What it cannot do is add labels to the cluster, like it cannot say this a group of apples or mangoes, but it will separate all the apples from mangoes. Suppose we presented images of apples, bananas, and mangoes to the model, so what it does, based on some patterns and relationships it creates clusters and divides the dataset into those clusters. Now if a new data is fed to the model, it adds it to one of the created clusters.

Reinforcement Learning

It is the ability of an agent to interact with the environment and find out what is the best outcome. It follows the concept of the hit and trial method. The agent is rewarded or penalized with a point for a correct or a wrong answer, and based on the positive reward points gained the model trains itself. And again once trained it gets ready to predict the new data presented to it.

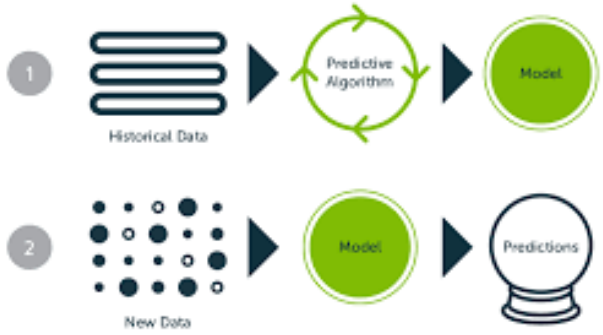
➢ Predictive Analysis

Well, how do we get to know about the upcoming election results or some kind of fraudulent transactions? Is it some magic, well actually no, but it's the use of Machine Learning-based predictive analysis that determines these results.

What is Predictive Analytics?

Predictive analytics describes the use of statistics and modeling to determine future performance based on current and historical data. Predictive analytics looks at patterns in data to determine if those patterns are likely to emerge again, which allows businesses and investors to adjust where they use their resources to take advantage of possible future events.

Predictive analytics is an area of statistics that deals with extracting information from data and using it to predict trends and behaviour patterns. The enhancement of predictive web analytics calculates statistical probabilities of future events online. Predictive analytics is often defined as predicting at a more detailed level of granularity, i.e., generating predictive scores (probabilities) for each organizational element. This distinguishes it from forecasting. For example, "Predictive analytics—Technology that learns from experience (data) to predict the future behaviour of individuals to drive better decisions. In future industrial systems, the value of predictive analytics will be to predict and prevent potential issues to achieve near-zero break-down and further be integrated into prescriptive analytics for decision optimization. Generally, the term predictive analytics is used to mean predictive modeling, "scoring" data with predictive models, and forecasting. However, people are increasingly using the term to refer to related analytical disciplines, such as descriptive modeling and decision modeling or optimization.



These disciplines also involve rigorous data analysis, and are widely used in business for segmentation and decision making, but have different purposes and the statistical techniques underlying them vary.

Approaches

Although machine learning (ML) is commonly used in designing predictive analytics, it doesn‘t mean it‘s the only solution. There are many ways to build a predictive analytics system. Simpler approaches, for example, may have very few data, or we may want to build a minimal solution fast, etc.

Types:-

- **Predictive models** – It uses predictive models to analyze the relationship between the specific performance of a unit in a sample and one or more known attributes or features of the unit. The objective of the model is to assess the likelihood that a similar unit in a different sample will exhibit a specific performance. This modeling systems have become capable of simulating human behavior or reactions to given stimuli or scenarios

- **Descriptive models** - Descriptive models quantify relationships in data in a way that is often used to classify customers or prospects. Unlike predictive models that focus on predicting single customer behavior (such as credit risk), descriptive models identify many different relationships between customers or products. Descriptive models do not rank-order customers by their likelihood of taking a particular action the way predictive models do. Instead, descriptive models can be used, for example, to categorize customers by their product preferences and life stage. Descriptive modeling tools can be utilized to develop further models that can simulate a large number of individualized agents and make predictions.

Analytical techniques

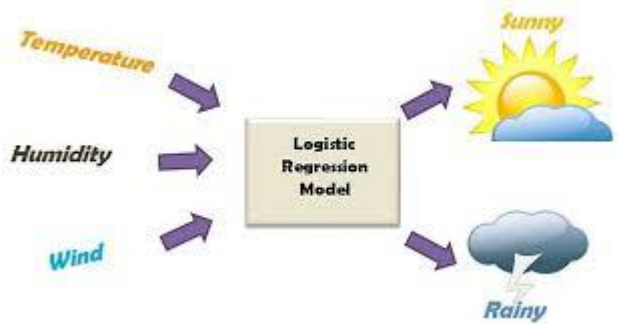
The approaches and techniques used to conduct predictive analytics can broadly be grouped into regression techniques and machine learning techniques are a part of Supervised Learning.

☞ Regression techniques

Regression models are the mainstay of predictive analytics. The focus lies in establishing a mathematical equation as a model to represent the interactions between the different variables in consideration. Depending on the situation, there are a wide variety of models that can be applied while performing predictive analytics. Some of them are briefly discussed below.

○ Logistic regression

Logistic Regression is one of the most commonly used Machine Learning algorithms that is used to model a binary variable that takes only 2 or more values – 0 and 1. The objective of Logistic Regression is to develop a mathematical equation that can give us a score in the range of 0 to 1. This score gives us the probability of the variable taking the value 1.



For example in the above figure, three inputs are given, where logistic regression is used to predict two binary values, in here the values or outcomes are either sunny or rainy from the given inputs temperature, humidity, and wind.

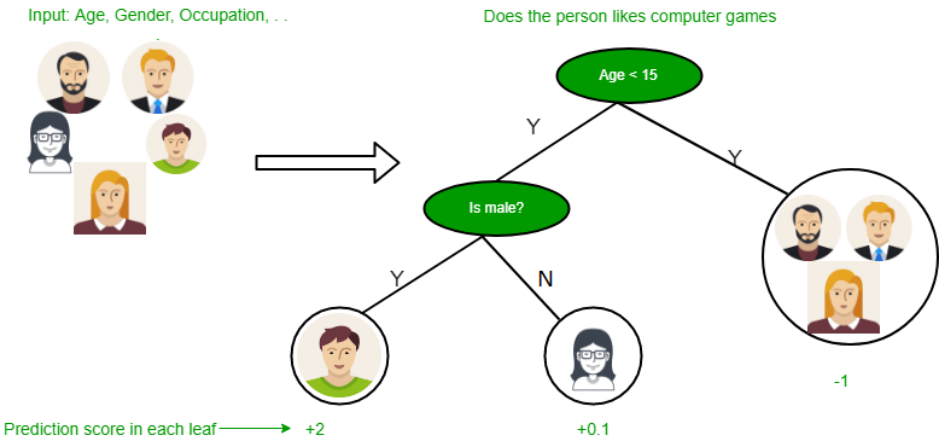
☞ Classification and regression trees (CART)

Classification and regression trees (CART) are a non-parametric decision tree learning technique that produces either classification or regression trees, depending on whether the dependent variable is categorical or numeric, respectively.

○ Decision Tree

Decision tree learning is one of the predictive modeling approaches used in statistics, data mining, and machine learning. It uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real

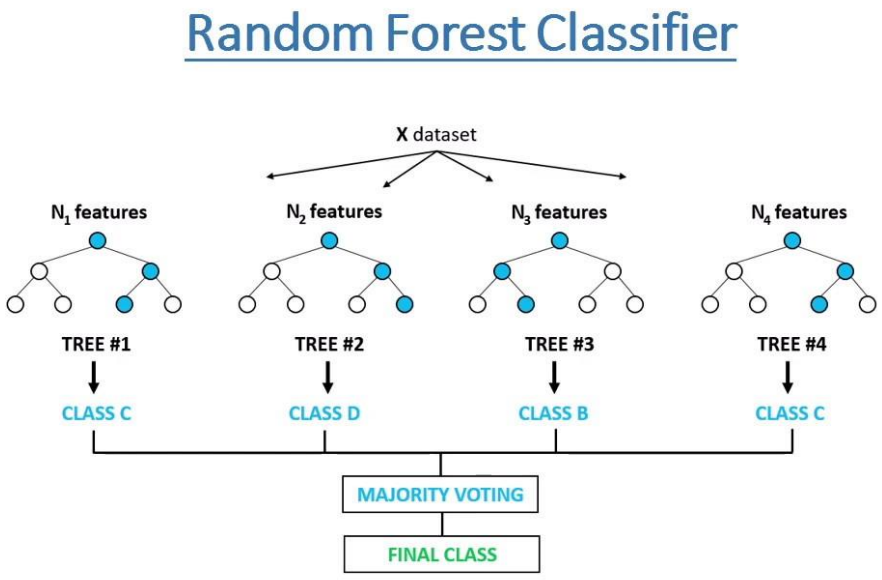
numbers) are called regression trees. Decision trees are among the most popular machine learning algorithms given their intelligibility and simplicity.



The above image will explain the application better. It shows the prediction that whether a person plays computer games, based on the inputs of age, gender, and occupation. A decision tree can be constructed that shows the attributes of this situation: gender, age, and level of income. A decision tree will identify which of these attributes has the highest predictive value and, ultimately determines whether the person likes computer games or not.

- ## Random Forest Classifier

Random forests or random decision forests are an ensemble learning method for classification, regression, and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.



❖ Advantages and Disadvantages of Machine Learning Language

➢ Advantages of Machine learning

1. Easily identifies trends and patterns

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviours and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

2. No human intervention needed (automation)

With ML, you don't need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus software's; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

3. Continuous Improvement

As ML algorithms gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

4. Handling multi-dimensional and multi-variety data

Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

5. Wide Applications

You could be an e-tailor or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

➢ Disadvantages of Machine Learning

With all those advantages to its powerfulness and popularity, Machine Learning isn't perfect. The following factors serve to limit it:

1. Data Acquisition

   Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

2. Time and Resources

   ML needs enough time to let the algorithms learn and develop enough to fulfil their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

3. Interpretation of Results

   Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

4. High error-susceptibility

   Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

❖ TOOLS AND TECHNOLOGIES

Python?

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping or production-ready software development.

Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc.).
- Python has a simple syntax similar to the English language.
- Python has a syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated procedurally, an object-orientated way, or a functional way.

Packages of Python

Packages are a way of structuring many packages and modules which helps in a well- organized hierarchy of data set, making the directories and modules easy to access. Just like there are different drives and folders in an OS to help us store files, similarly packages help us in storing other sub-packages and modules, so that it can be used by the user when necessary.

There are two types of packages:

I. User-Defined Packages
II. Import Packages

I. **User-Defined Packages:** To tell Python that a particular directory is a package, create a file named __init__.py inside it, and then it is considered as a package and may create other modules and sub-packages within it. This __init__.py file can be left blank or can be coded with the initialization code for the package.

To create a package in Python, we need to follow these three simple steps:

1. First, create a directory and give it a package name, preferably related to its operation.
2. Then put the classes and the required functions in it.
3. Finally create an __init__.py file inside the directory, to let Python know that the directory is a package.

II. **Import Packages:** Import in python is similar to #include header_file in C/C++. Python modules can get access to code from another module by importing the file/function using import. The import statement is the most common way of invoking the import machinery, but it is not the only way.

import module_name

When the import is used, it searches for the module initially in the local scope by calling __import__() function. The value returned by the function is then reflected in the output of the initial code.

```
import math
print(math.pi)
```

import module_name.member_name

In the above code module, math is imported, and its variables can be accessed by considering it to be a class and pi as its object.

The value of pi is returned by __import__(). pi as a whole can be imported into our initial code, rather than importing the whole module.

```
from math import pi

# Note that in the above example,
# we used math.pi. Here we have used
# pi directly.
print(pi)
```

Some import packages are-

❑ Numpy

Numpy is a general-purpose array-processing package. It provides a high- performance multidimensional array object and tools for working with these arrays. It is the fundamental package for scientific computing with Python.

It contains various features including these important ones:

- ➢ A powerful N-dimensional array object
- ➢ Sophisticated (broadcasting) functions

- ➢ Tools for integrating C/C++ and Fortran code
- ➢ Useful linear algebra, Fourier transform, and random number capabilities Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data.

  Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

The main object of numpy is the homogeneous multidimensional array:

- ➢ It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers.
- ➢ In Numpy dimensions are called axes. The number of axes is rank.
- ➢ Numpy's array class is called ndarray. It is also known by the alias array.

❑ Pandas

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with structured (tabular, multidimensional, potentially heterogeneous) and time-series data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open-source data analysis/manipulation tool available in any language. It is already well on its way toward this goal. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data.

Features of pandas:

- ➢ Fast and efficient DataFrame object with the default and customized indexing.
- ➢ Tools for loading data into in-memory data objects from different file formats.
- ➢ Data alignment and integrated handling of missing data.
- ➢ Reshaping and pivoting of data sets.
- ➢ Label-based slicing, indexing, and subsetting of large data sets.
- ➢ Columns from a data structure can be deleted or inserted.
- ➢ Group by data for aggregation and transformations.

❑ Matplotlib

Matplotlib is an excellent 2D and 3D graphics library for generating scientific figures.

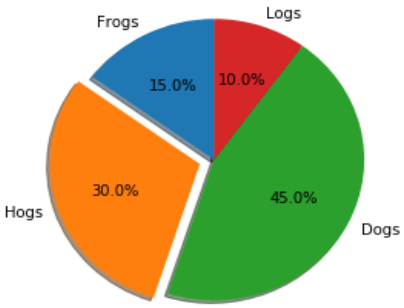
Some of the many advantages of this library include:

- ➢ Easy to get started
- ➢ Support for LATEX formatted labels and texts
- ➢ Great control of every element in a figure, including figure size and DPI.
- ➢ High-quality output in many formats, including PNG, PDF, SVG, EPS, and PGF.
- ➢ GUI for interactively exploring figures *and* support for headless generation of figure files (useful for batch jobs).

One of the key features of matplotlib that I would like to emphasize, and that I think makes matplotlib highly suitable for generating figures for scientific publications is that all aspects of the figure can be controlled *programmatically*. This is important for reproducibility and convenience when one needs to regenerate the figure with updated data or change its appearance.

Example:

```
import matplotlib.pyplot as plt

# Pie chart, where the slices will be ordered and plotted counter-clockwise:
labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
sizes = [15, 30, 45, 10]
explode = (0, 0.1, 0, 0)  # only "explode" the 2nd slice (i.e. 'Hogs')

fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

plt.show()
```



❑ Seaborn

Seaborn is a library for making statistical graphics in Python. It is built on top of matplotlib and closely integrated with pandas data structures.

Here is some of the functionality that seaborn offers:

- ➢ A dataset-oriented API for examining relationships between multiple variables
- ➢ Specialized support for using categorical variables to show observations or aggregate statistics
- ➢ Options for visualizing univariate or bivariate distributions and for comparing them between subsets of data
- ➢ Automatic estimation and plotting of linear regression models for different kinds of dependent variables
- ➢ Convenient views onto the overall structure of complex datasets
- ➢ High-level abstractions for structuring multi-plot grids that let you easily build complex visualizations
- ➢ Concise control over matplotlib figure styling with several built-in themes
- ➢ Tools for choosing color palettes that faithfully reveal patterns in your data

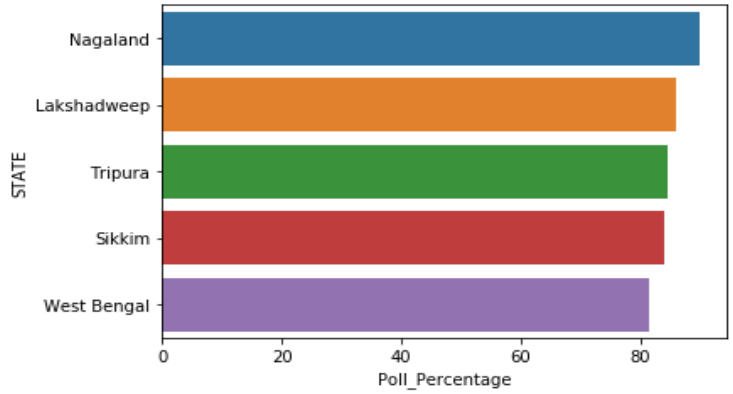
Seaborn aims to make visualization a central part of exploring and understanding data. Its dataset-oriented plotting functions operate on data frames and arrays containing whole

datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots.

Example:

```
import pandas as pd
import seaborn as sb

data = {'STATE': ['Nagaland','Lakshadweep','Tripura','Sikkim','West Bengal'],
        'Poll_Percentage': [90.000000,85.900000,84.475000,83.930000,81.551905],
        }
DF = pd.DataFrame(data, columns = ['STATE', 'Poll_Percentage'])

sb.barplot(x=DF.Poll_Percentage, y=DF.STATE)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1b21f637f88>
```



❑ Plotly

The plotly Python library (plotly.py) is an interactive, open-source plotting library that supports over 40 unique chart types covering a wide range of statistical, financial, geographic, scientific, and 3-dimensional use-cases.

Built on top of the Plotly JavaScript library (plotly.js), plotly.py enables Python users to create beautiful interactive web-based visualizations that can be displayed in Jupyter notebooks, saved to standalone HTML files, or served as part of pure Python-built web applications using Dash.

Plotly Express

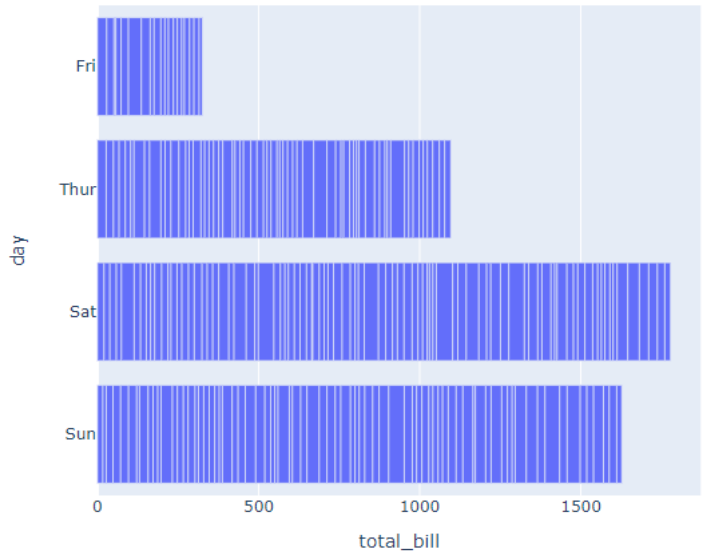
Plotly Express is the easy-to-use, high-level interface to Plotly, which operates on a variety of types of data and produces easy-to-style figures. Every Plotly Express function returns a graph_objects. Figure object whose data and layout have been pre-populated according to the provided arguments.

Here's the code to make a bar chart from a DataFrame, with the conventional Express import:

```
import plotly_express as px
px.bar(my_df, x='my_x_column', y='my_y_column')
```

Example:

```
import plotly.express as px
df = px.data.tips()
fig = px.bar(df, x="total_bill", y="day", orientation='h')
fig.show()
```



❑ Scikit-learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. The library is built upon the SciPy (Scientific Python) that must be installed before you can use scikit-learn. This stack that includes:

- ★ NumPy: Base n-dimensional array package
- ★ SciPy: Fundamental library for scientific computing
- ★ Matplotlib: Comprehensive 2D/3D plotting
- ★ IPython: Enhanced interactive console
- ★ Sympy: Symbolic mathematics
- ★ Pandas: Data structures and analysis

Extensions or modules for SciPy are conventionally named SciKits. As such, the module provides learning algorithms and is named scikit-learn.

The vision for the library is a level of robustness and support required for use in production systems. This means a deep focus on concerns such as ease of use, code quality, collaboration, documentation, and performance.

Although the interface is Python, c-libraries are leveraged for performance such as numpy for arrays and matrix operations, LAPACK, LibSVM, and the careful use of python.

Some popular groups of models provided by scikit-learn include:

- **Clustering:** for grouping unlabelled data such as KMeans.
- **Cross-Validation:** for estimating the performance of supervised models on unseen data.
- **Datasets:** for test datasets and for generating datasets with specific properties for investigating model behavior.
- **Dimensionality Reduction:** for reducing the number of attributes in data for summarization, visualization, and feature selection such as Principal component analysis.
- **Ensemble methods:** for combining the predictions of multiple supervised models.
- **Feature extraction:** for defining attributes in image and text data.
- **Feature selection:** for identifying meaningful attributes from which to create supervised models.
- **Parameter Tuning:** for getting the most out of supervised models.
- **Manifold Learning:** For summarizing and depicting complex multi-dimensional data.
- **Supervised Models:** a vast array not limited to generalized linear models, discriminant analysis, naive Bayes, lazy methods, neural networks, support vector machines, and decision trees.

❑ Anaconda

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS. It is developed and maintained by Anaconda, Inc., which was founded by Peter Wang and Travis Oliphant in 2012. As an Anaconda, Inc. product, it is also known as Anaconda Distribution or Anaconda Individual Edition, while other products from the company are Anaconda Team Edition and Anaconda Enterprise Edition, which are both not free.

Anaconda distribution comes with over 250 packages automatically installed, and over 7,500 additional open-source packages can be installed from PyPI as well as the conda package and virtual environment manager. It also includes a GUI, Anaconda Navigator, as a graphical alternative to the command-line interface (CLI).

The big difference between conda and the pip package manager is how to package dependencies are managed, which is a significant challenge for Python data science and the reason conda exists.

❑ Anaconda Navigator

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments, and channels without using command-line commands. Navigators can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages, and update them. It is available for Windows, macOS, and Linux.

The following applications are available by default in Navigator:

- JupyterLab
- Jupyter Notebook
- QtConsole
- Spyder
- Glue
- Orange
- RStudio
- Visual Studio Code

❑ Conda

Conda is an open-source, cross-platform, language-agnostic package manager and environment management system that installs, runs, and updates packages and their dependencies. It was created for Python programs, but it can package and distribute software for any language (e.g., R), including multi-language projects. The conda package and environment manager is included in all versions of Anaconda, Miniconda, and Anaconda Repository.

Anaconda Cloud is a package management service by Anaconda where users can find, access, store, and share public and private notebooks, environments, and conda and PyPI packages. Cloud hosts useful Python packages, notebooks, and environments for a wide variety of applications. Users do not need to log in or to have a Cloud account, to search for public packages, download, and install them.

Users can build new packages using the Anaconda Client command-line interface (CLI), then manually or automatically upload the packages to Cloud.

❑ Jupyter Notebook

Jupyter Notebook (formerly interactive computational IPython environment Notebooks) is creating a web-based Jupyter notebook for documents. The "notebook" term can colloquially refer to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on the context. A Jupyter Notebook document is a JSON document, following a versioned schema, and containing an ordered list of input/output cells that can contain code, text (using Markdown), mathematics, plots, and rich media, usually ending with the ".ipynb" extension.

A Jupyter Notebook can be converted to several open standard outputs formats(HTML, presentation slides, LaTeX, PDF, ReStructuredText, Markdown, Python) through "Download As" in the web interface, via the nbconvert library or "jupyter nbconvert" command-line interface in a shell. To simplify visualization of Jupyter notebook documents on the web, the nbconvert library is provided as a service through NbViewer which can take a URL to any publicly available notebook document, convert it to HTML on the fly and display it to the user.

Jupyter Notebook provides a browser-based REPL built upon a number of popular open-source libraries. Jupyter Notebook can connect to many kernels to allow programming in many languages. By default Jupyter Notebook ships with the IPython kernel.

❖ Project Overview:-

- **Project Name – Lok Sabha 2009 vs 2014 vs 2019 Analysis and Prediction**
- **Institute – Meghnad Saha Institute of Technology**
- **Project Type – Predictive Analysis with Machine Learning(ML)**
- **Back end – Python 3.8.3**
- **Platform – Windows 10 Pro Edition**

❖ SYSTEM AND SOFTWARE REQUIREMENT

Recommended System Requirements

Processors:

- Intel® Core™ i5 processor 4300M at 2.60 GHz or 2.59 GHz (1 socket, 2 cores, 2 threads per core), 8 GB of DRAM
- Intel® Xeon® processor E5-2698 v3 at 2.30 GHz (2 sockets, 16 cores each, 1 thread per core), 64 GB of DRAM
- Intel® Xeon Phi™ processor 7210 at 1.30 GHz (1 socket, 64 cores, 4 threads per core), 32 GB of DRAM, 16 GB of MCDRAM (flat mode enabled)
- Disk space: 2 to 3 GB
- Operating systems: Windows® 10, macOS*, and Linux*

Minimum System Requirements:

- Processors: Intel Atom® processor or Intel® Core™ i3 processor
- Disk space: 1 GB
- Operating systems: Windows* 7 or later, macOS, and Linux
- Python* versions: 2.7.X, 3.6.X
- Included development tools: conda*, conda-env, Jupyter Notebook* (IPython)
- Compatible tools: Microsoft Visual Studio*, PyCharm*
- Included Python packages: NumPy, SciPy, scikit-learn*, pandas, Matplotlib, Numba*, Intel® Threading Building Blocks, pyDAAL, Jupyter, mpi4py, PIP*, and others.

Software:

- PIP and NumPy: Installed with PIP, Ubuntu*, Python 3.6.2, NumPy 1.13.1, scikit-learn 0.18.2
- Windows: Python 3.8.1, PIP and NumPy 1.13.1, scikit-learn 0.18.2
- Intel® Distribution for Python* 2018

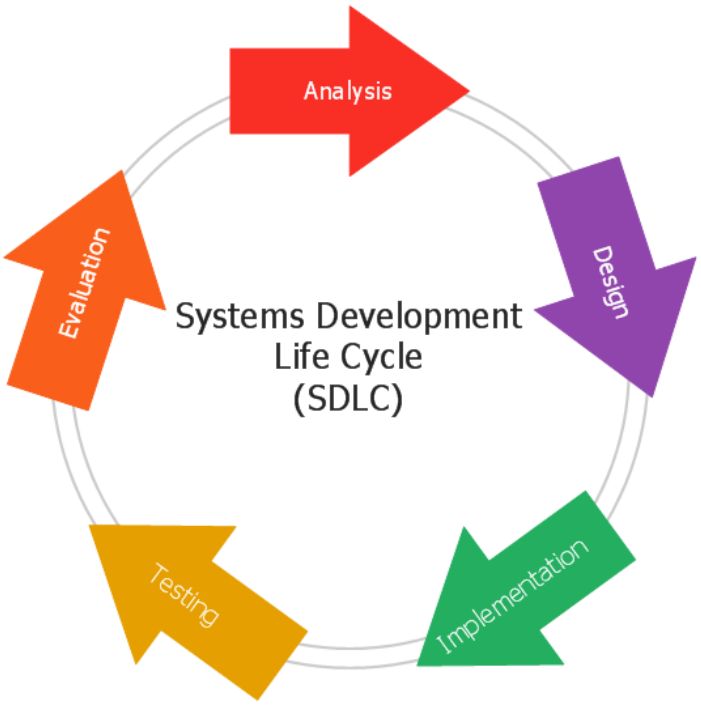
Modifications:

- Scikit-learn: Conda*-installed NumPy with Intel® Math Kernel Library (Intel® MKL) on Windows (PIP-installed SciPy on Windows contains Intel MKL dependency)
- Black-Scholes on Intel Core i5 processor and Windows: PIP-installed NumPy and Conda-installed SciPy

❖ SOFTWARE DEVELOPMENT LIFE CYCLE

What is SDLC:-

The software development life cycle (SDLC) is a series of phases that provide a common understanding of the software building process. How the software will be realized and developed from the business understanding and requirements elicitation phase to convert these business ideas and requirements into functions and features until its usage and operation to achieve the business needs. The good software engineer should have enough knowledge on how to choose the SDLC model based on the project context and the business requirements.



Phase 1: Requirement collection and analysis:

The requirement is the first stage in the SDLC process. It is conducted by the senior team members with inputs from all the stakeholders and domain experts in the industry. Planning for the quality assurance requirements and reorganization of the risks involved is also done at this stage. This stage gives a clearer picture of the scope of the entire project and the anticipated issues, opportunities, and directives that triggered the project.

Requirements Gathering stage needs teams to get detailed and precise requirements. This helps companies to finalize the necessary timeline to finish the work of that system.

Phase 2: Design:

In this third phase, the system and software design documents are prepared as per the requirement specification document. This helps define overall system architecture.

This design phase serves as input for the next phase of the model.

There are two kinds of design documents developed in this phase:

High-Level Design (HLD)

- Brief description and name of each module
- An outline of the functionality of every module
- Interface relationship and dependencies between modules
- Database tables identified along with their key elements
- Complete architecture diagrams along with technology details

Low-Level Design (LLD)

- Functional logic of the modules
- Database tables, which include type and size
- Complete detail of the interface
- Addresses all types of dependency issues
- Listing of error messages
- Complete input and outputs for every module

Phase 3: Implementation:

Once the system design phase is over, the next phase is coding. In this phase, developers start to build the entire system by writing code using the chosen programming language. In the coding phase, tasks are divided into units or modules and assigned to the various developers. It is the longest phase of the Software Development Life Cycle process.

In this phase, the developer needs to follow certain predefined coding guidelines. They also need to use programming tools like compiler, interpreters, debuggers to generate and implement the code.

Phase 4: Testing:

Once the software is complete, and it is deployed in the testing environment. The testing team starts testing the functionality of the entire system. This is done to verify that the entire application works according to the customer requirement.

During this phase, QA and testing team may find some bugs/defects which they communicate to developers. The development team fixes the bug and sends back to QA for a re-test. This process continues until the software is bug-free, stable, and working according to the business needs of that system.

Phase 5: Evaluation:

1. Installation/Deployment:

Once the software testing phase is over and no bugs or errors left in the system then the final deployment process starts. Based on the feedback given by the project manager, the final software is released and checked for deployment issues if any.

2. Maintenance:

Once the system is deployed, and customers start using the developed system, the following 3 activities occur

- Bug fixing - bugs are reported because of some scenarios which are not tested at all
- Upgrade - Upgrading the application to the newer versions of the Software
- Enhancement - Adding some new features into the existing software

The main focus of this SDLC phase is to ensure that needs continue to be met and that the system continues to perform as per the specification mentioned in the first phase.

SDLC model with respect to the Project:-

1. **Data Collection -** Collecting data allows you to capture a record of past events so that we can use data analysis to find recurring patterns. From those patterns, you build predictive models using machine learning algorithms that look for trends and predict future changes.
   Predictive models are only as good as the data from which they are built, so good data collection practices are crucial to developing high-performing models. The data need to be error-free (garbage in, garbage out) and contain relevant information for the task at hand.

2. **Feature Selection -** We all may have faced this problem of identifying the related features from a set of data and removing the irrelevant or less important features which do not contribute much to our target variable in order to achieve better accuracy for our model.
   Feature Selection is one of the core concepts in machine learning which hugely impacts the performance of your model. The data features that you use to train your machine learning models have a huge influence on the performance you can achieve. Irrelevant or partially relevant features can negatively impact model performance. Feature selection and Data cleaning should be the first and most important step of your model designing. In this post, you will discover feature selection techniques that you can use in Machine Learning. Feature Selection is the process where you automatically or manually select those features which contribute most to your prediction variable or output in which you are interested. Having irrelevant features in your data can decrease the accuracy of the models and make your model learn based on irrelevant features.
   How to select features and what are the Benefits of performing feature selection before modeling your data?

   - **Reduces Overfitting:** Less redundant data means less opportunity to make decisions based on noise.
   - **Improves Accuracy:** Less misleading data means modeling accuracy improves.

- **Reduces Training Time:** Fewer data points reduce algorithm complexity and algorithms train faster.

3. **Data Pre-Processing -** It is a data mining technique that transforms raw data into an understandable format. Raw data (real-world data) is always incomplete and that data cannot be sent through a model. That would cause certain errors. That is why we need to pre-process data before sending it through a model.
   Steps in Data Pre-processing
   1. Import libraries
   2. Read data
   3. Checking for missing values
   4. Checking for categorical data
   5. Standardize the data
   6. PCA transformation
   7. Data splitting

4. **Data Visualization -** Data visualization is a technique that uses an array of static and interactive visuals within a specific context to help people understand and make sense of large amounts of data. The data is often displayed in a story format that visualizes patterns, trends, and correlations that may otherwise go unnoticed. Data visualization is regularly used as an avenue to monetize data as a product.
   If data visualization pulls datasets out of your big data and visualizes a specific data story to monetize your data, machine learning can bring added value by streamlining your data visualization process for more accurate, predictive, and profitable data.

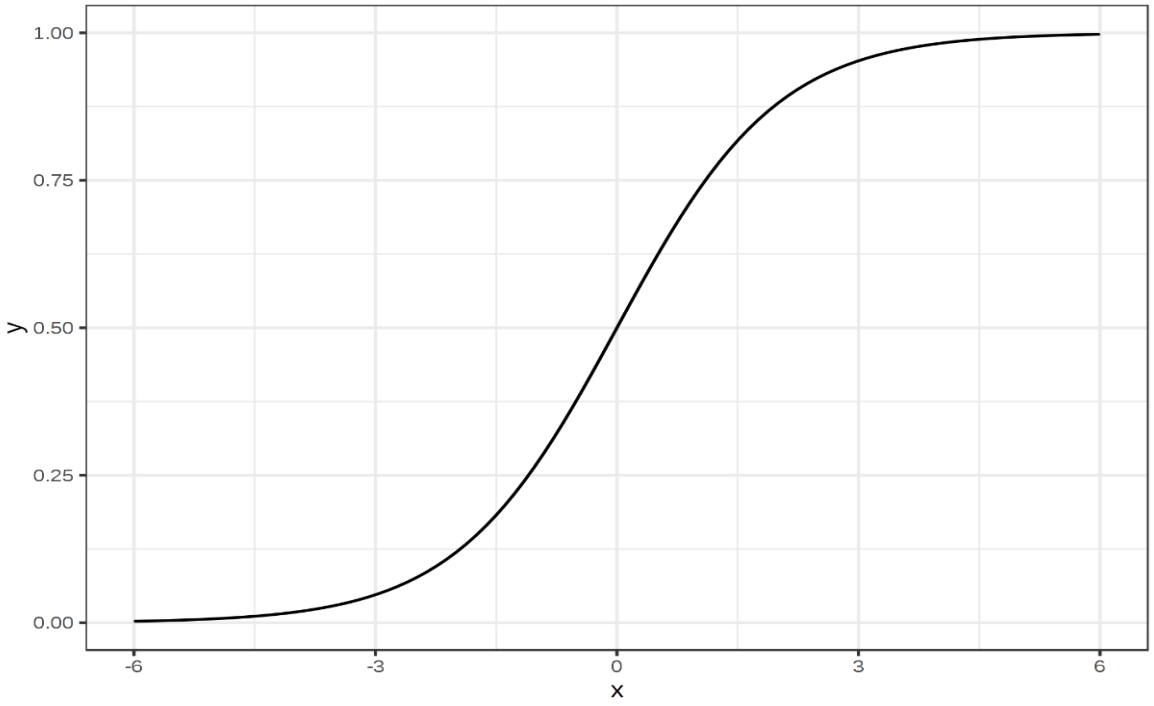
5. Model Selection –

I. Logistic Regression:

A solution for classification is logistic regression. Instead of fitting a straight line or hyperplane, the logistic regression model uses the logistic function to squeeze the output of a linear equation between 0 and 1. The logistic function is defined as:

$$\text{logistic}(\eta) = \frac{1}{1 + exp(-\eta)}$$

And it looks like this:



The logistic function. It outputs numbers between 0 and 1. At input 0, it outputs 0.5.

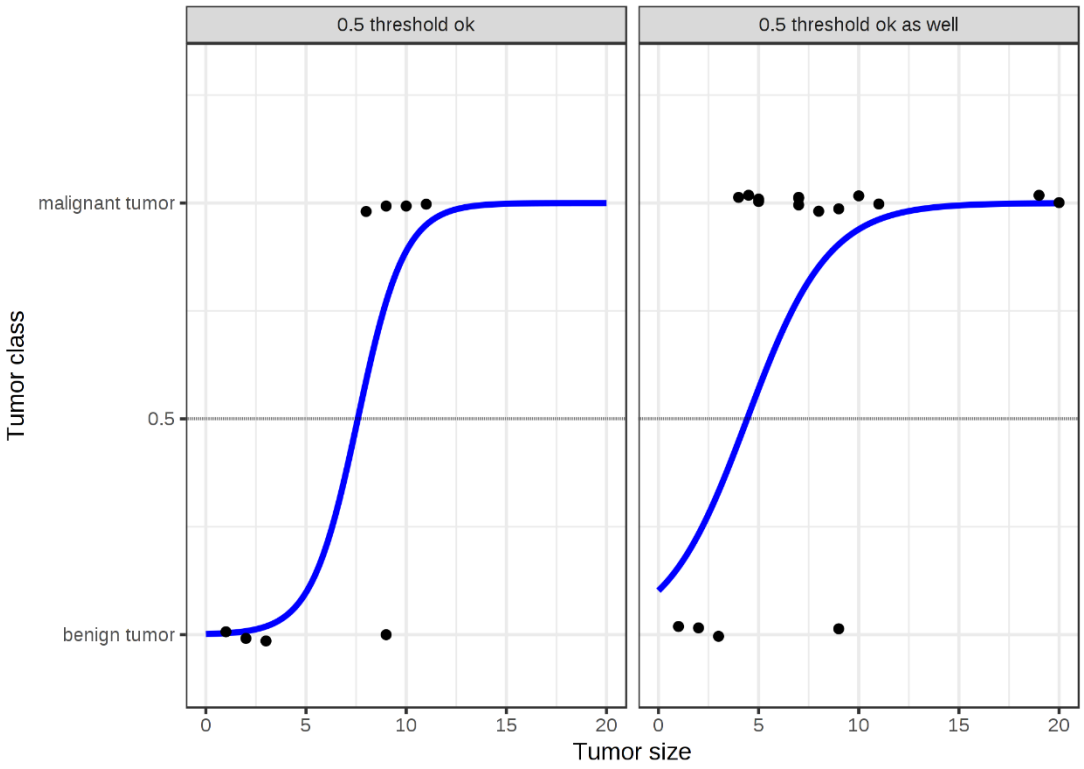
The step from linear regression to logistic regression is kind of straightforward. In the linear regression model, we have modeled the relationship between the outcome and features with a linear equation:

$$\hat{y}^{(i)} = \beta_0 + \beta_1 x_1^{(i)} + \ldots + \beta_p x_p^{(i)}$$

For classification, we prefer probabilities between 0 and 1, so we wrap the right side of the equation into the logistic function. This forces the output to assume only values between 0 and 1.

$$P(y^{(i)} = 1) = \frac{1}{1 + exp(-(\beta_0 + \beta_1 x_1^{(i)} + \ldots + \beta_p x_p^{(i)}))}$$

Let us revisit the tumor size example again. But instead of the linear regression model, we use the logistic regression model:



The logistic regression model finds the correct decision boundary between malignant and benign depending on tumour size. The line is the logistic function shifted and squeezed to fit the data.

The classification works better with logistic regression and we can use 0.5 as a threshold in both cases. The inclusion of additional points does not really affect the estimated curve.

Interpretation

The interpretation of the weights in logistic regression differs from the interpretation of the weights in linear regression since the outcome in logistic regression is a probability between 0 and 1. The weights do not influence the probability linearly any longer. The weighted sum is transformed by the logistic function to a probability.

Therefore we need to reformulate the equation for the interpretation so that only the linear term is on the right side of the formula.

$$log\left(\frac{P(y=1)}{1-P(y=1)}\right) = log\left(\frac{P(y=1)}{P(y=0)}\right) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$$

We call the term in the log () function "odds" (probability of event divided by the probability of no event) and wrapped in the logarithm it is called log odds.

This formula shows that the logistic regression model is a linear model for the log odds. Great! That does not sound helpful! With a little shuffling of the terms, you can figure out how the prediction changes when one of the features xj is changed by 1 unit. To do this, we can first apply the exp() function to both sides of the equation:

$$\frac{P(y=1)}{1-P(y=1)} = odds = exp\left(\beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p\right)$$

Then we compare what happens when we increase one of the feature values by but instead of looking at the difference, we look at the ratio of the two predictions:

$$\frac{odds_{x_j+1}}{odds} = \frac{exp\left(\beta_0 + \beta_1 x_1 + \ldots + \beta_j(x_j+1) + \ldots + \beta_p x_p\right)}{exp\left(\beta_j x_j + \ldots + \beta_p x_p\right)}$$

$$\frac{exp(a)}{exp(b)} = exp(a-b)$$

We apply the following rule:

And we remove many terms:

$$\frac{odds_{x_j+1}}{odds} = exp\left(\beta_j(x_j+1) - \beta_j x_j\right) = exp\left(\beta_j\right)$$

In the end, we have something as simple as exp() of a feature weight. A change in a feature by one unit changes the odds ratio (multiplicative) by a factor of exp(βj). We could also interpret it this way: A change in xj by one unit increases the log odds ratio by the value of the corresponding weight. Most people interpret the odds ratio because thinking about the log() of something is known to be hard on the brain. Interpreting the odds ratio already requires some getting used to. For example, if you have odds of 2, it means that the probability for y=1 is twice as high as y=0. If you have a weight (= log odds ratio) of 0.7, then increasing the respective feature by one unit multiplies the odds by exp(0.7) (approximately 2) and the odds change to 4. But usually, you do not deal with the odds and interpret the weights only as of the odds ratios. Because for actually calculating the odds you would need to set a value for each feature, which only makes sense if you want to look at one specific instance of your dataset.

These are the interpretations of the logistic regression model with different feature types:

- **Numerical feature:** If you increase the value of feature xj by one unit, the estimated odds change by a factor of exp(βj)
- **Binary categorical feature:** One of the two values of the feature is the reference category (in some languages, the one encoded in 0). Changing the feature xj from the reference category to the other category changes the estimated odds by a factor of exp(βj).
- **Categorical features with more than two categories:** One solution to deal with multiple categories is one-hot-encoding, meaning that each category has its own column. You only need L-1 columns for a categorical feature with L categories, otherwise, it is over-parameterized. The L-th category is then the reference category. You can use any other encoding that can be used in linear regression. The interpretation for each category then is equivalent to the interpretation of binary features.

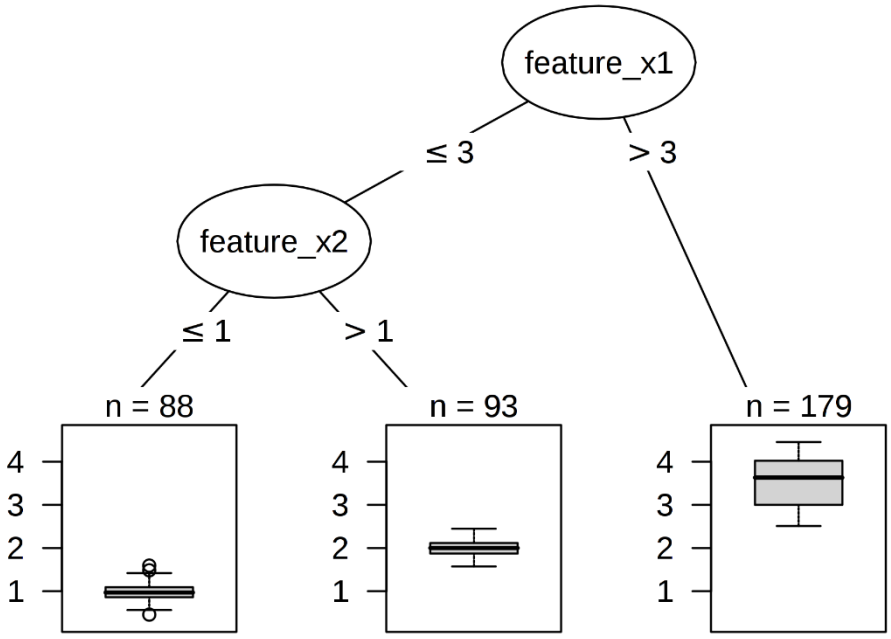
- **Intercept β0:** When all numerical features are zero and the categorical features are at the reference category, the estimated odds are exp(β0). The interpretation of the intercept weight is usually not relevant.

II. Decision Tree:

Linear regression and logistic regression models fail in situations where the relationship between features and outcome is nonlinear or where features interact with each other. Time to shine for the decision tree! Tree-based models split the data multiple times according to certain cutoff values in the features. Through splitting, different subsets of the dataset are created, with each instance belonging to one subset. The final subsets are called terminal or leaf nodes and the intermediate subsets are called internal nodes or split nodes. To predict the outcome in each leaf node, the average outcome of the training data in this node is used. Trees can be used for classification and regression.

There are various algorithms that can grow a tree. They differ in the possible structure of the tree (e.g. number of splits per node), the criteria for how to find the splits, when to stop splitting and how to estimate the simple models within the leaf nodes. The classification and regression trees (CART) algorithm is probably the most popular algorithm for tree induction. We will focus on CART, but the interpretation is similar

for most other tree types. I recommend the book 'The Elements of Statistical Learning' (Friedman, Hastie and Tibshirani 2009)17 for a more detailed introduction to CART.



Decision tree with artificial data. Instances with a value greater than 3 for feature x1 end up in node 5. All other instances are assigned to node 3 or node 4, depending on whether values of feature x2 exceed 1.

The following formula describes the relationship between the outcome y and features x.

$$\hat{y} = \hat{f}(x) = \sum_{m=1}^{M} c_m I\{x \in R_m\}$$

Each instance falls into exactly one leaf node (=subset Rm). I{x∈Rm} is the identity function that returns 1 if x is in the subset Rm and 0 otherwise. If an instance falls into a leaf node Rl , the predicted outcome is ŷ=cl, where cl is the average of all training instances in leaf node Rl.

But where do the subsets come from? This is quite simple: CART takes a feature and determines which cut-off point minimizes the variance of y for a regression task or the Gini index of the class distribution of y for classification tasks. The variance tells us how much the y values in a node are spread around their mean value. The Gini index tells us how "impure" a node is, e.g. if all classes have the same frequency, the node is impure, if only one class is present, it is maximally pure. Variance and Gini index are minimized when the data points in the nodes have very similar values for y. As a consequence, the best cut-off point makes the two resulting subsets as different as possible with respect to the target outcome. For categorical features, the algorithm tries to create subsets by trying different groupings of categories. After the best cutoff per feature has been determined, the algorithm selects the feature for splitting that would result in the best partition in terms of the variance or Gini index and adds this split to the tree. The algorithm continues this search-and-split recursively in both new nodes until a stop criterion is reached. Possible criteria are a minimum number of instances that have to be in a node before the split or the minimum number of instances that have to be in a terminal node.

Interpretation

The interpretation is simple: Starting from the root node, you go to the next node and the edges tell you which subsets you are looking at. Once you reach the leaf node, the node tells you the predicted outcome. All the edges are connected by 'AND'.

Template: If feature x is [smaller/bigger] than threshold c AND ... then the predicted outcome is the mean value of y of the instances in that node.

Feature importance

The overall importance of a feature in a decision tree can be computed in the following way: Go through all the splits for which the feature was used and measure how much it has reduced the variance or Gini index compared to the parent node. The sum of all importance is scaled to 100. This means that each importance can be interpreted as a share of the overall model's importance.

Tree decomposition

Individual predictions of a decision tree can be explained by decomposing the decision path into one component per feature. We can track a decision through the tree and explain a prediction by the contributions added at each decision node.

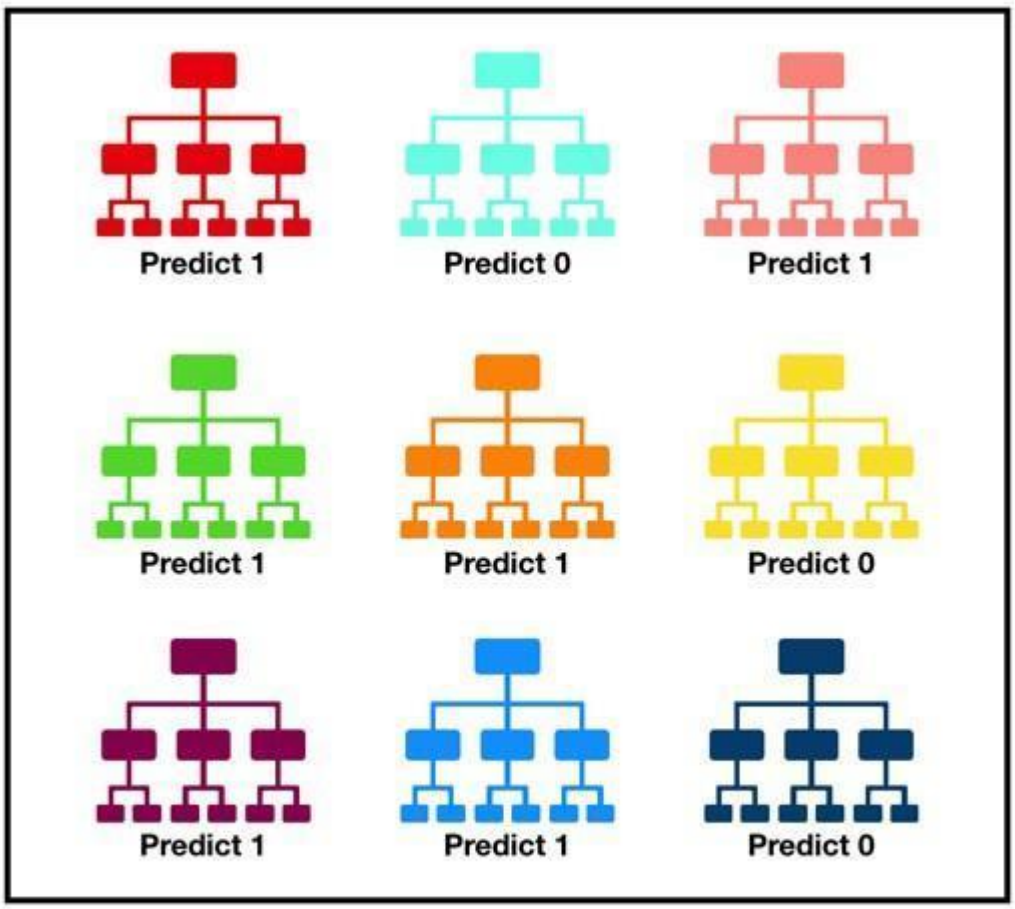
The root node in a decision tree is our starting point. If we were to use the root node to make predictions, it would predict the mean of the outcome of the training data. With the next split, we either subtract or add a term to this sum, depending on the next node in the path. To get to the final prediction, we have to follow the path of the data instance that we want to explain and keep adding to the formula.

The prediction of an individual instance is the mean of the target outcome plus the sum of all

$$\hat{f}(x) = \bar{y} + \sum_{d=1}^{D} \text{split.contrib(d,x)} = \bar{y} + \sum_{j=1}^{p} \text{feat.contrib(j,x)}$$

contributions of the D splits that occur between the root node and the terminal node where the instance ends up. We are not interested in the split contributions through but in the feature contributions. A feature might be used for more than one split or not at all. We can add the contributions for each of the p features and get an interpretation of how much each feature has contributed to a prediction.

II. **Random Forest:** Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the



Tally: Six 1s and Three 0s
Prediction: 1

random forest spits out a class prediction and the class with the most votes becomes our model's prediction (see figure below).

The fundamental concept behind random forest is a simple but powerful one — the wisdom of crowds. In data science-speak, the reason that the random forest model works so well is:

A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.

The low correlation between models is the key. Just like how investments with low correlations (like stocks and bonds) come together to form a portfolio that is greater than the sum of its parts, uncorrelated models can produce ensemble predictions that are more accurate than any of the individual predictions. The reason for this wonderful effect is that the trees protect each other from their individual errors (as long as they don't constantly all err in the same direction). While some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction. So the prerequisites for the random forest to perform well are:

a. There needs to be some actual signal in our features so that models built using those features do better than random guessing.
b. The predictions (and therefore the errors) made by the individual trees need to have low correlations with each other.

6. Train Test Split:

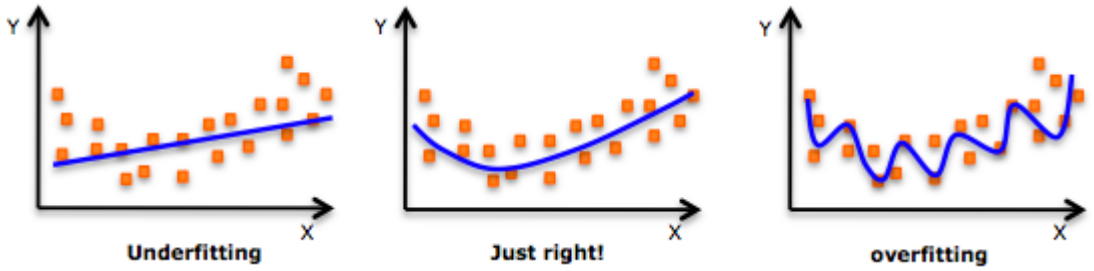
In statistics and machine learning we usually split our data into two subsets: training data and testing data (and sometimes to three: train, validate and test), and fit our model on the train data, in order to make predictions on the test data. When we do that, one of two things might happen: we overfit our model or we underfit our model. We don't want any of these things to happen, because they affect the predictability of our model — we might be using a model that has lower accuracy and/or is ungeneralized (meaning you can't generalize your predictions on other data). Let's see what under and overfitting actually mean:

Overfitting

Overfitting means that the model we trained has trained "too well" and is now, well, fit too closely to the training dataset. This usually happens when the model is too complex (i.e. too many features/variables compared to the number of observations). This model will be very accurate on the training data but will probably be very not accurate on untrained or new data. It is because this model is not generalized (or not AS generalized), meaning you can generalize the results and can't make any inferences on other data, which is, ultimately, what you are trying to do. Basically, when this happens, the model learns or describes the "noise" in the training data instead of the actual relationships between variables in the data. This noise, obviously, isn't part of any new dataset, and cannot be applied to it.

Underfitting

In contrast to overfitting, when a model is under fitted, it means that the model does not fit the training data and therefore misses the trends in the data. It also means the model cannot be generalized to new data. As you probably guessed (or figured out!), this is usually the result of a very simple model (not enough predictors/independent variables). It could also happen when, for example, we fit a linear model (like linear regression) to data that is not linear. It almost goes without saying that this model will have the poor predictive ability (on training data and can't be generalized to other data).



An example of overfitting, underfitting, and a model that's "just right!"

It is worth noting the underfitting is not as prevalent as overfitting. Nevertheless, we want to avoid both of those problems in data analysis. You might say we are trying to find the middle ground between under and overfitting our model. As you will see, train/test split and cross-validation help to avoid overfitting more than underfitting. Let's dive into both of them!

There are a few parameters that we need to understand before we use the class:

1. test_size - This parameter decides the size of the data that has to be split as the test dataset. This is given as a fraction. For example, if you pass 0.5 as the value, the dataset will be split 50% as the test dataset. If you're specifying this parameter, you can ignore the next parameter.
2. train_size - It has to specify this parameter only if it is not specifying the test_size This is the same as test_size, but instead tell the class what percent of the dataset wants to split as the training set.
3. random_state - Here they pass an integer, which will act as the seed for the random number generator during the split. Or, it can also pass an instance of the RandomState class, which will become the number generator. If it doesn't pass anything, the RandomState instance used by np.random will be used instead.

Train/Test Split:

As it is said before, the data we use is usually split into training data and test data. The training set contains a known output and the model learns on this data in order to be generalized to other data later on. We have the test dataset (or subset) in order to test our model's prediction on this subset.

7. Model Build:

Modification of column data:
As our dataset ‘LS_2.0.csv’ column ‘CRIMINAL CASES’ has float values. It looks like this:

```
In [54]: LS2019Cnd['CRIMINAL CASES']

Out[54]: 0       52.0
         1        0.0
         2        3.0
         3        0.0
         4        5.0
                 ...
         2258     0.0
         2259    18.0
         2260     0.0
         2261     3.0
         2262     0.0
         Name: CRIMINAL CASES, Length: 2263, dtype: float64
```

The Column ‘CRIMINAL CASES’ can be converted into a binary form eg: ‘HAS CASE’ and ‘NO CASE’. After converting it looks like this:

```
In [56]: LS2019Cnd['CRIMINAL CASES']

Out[56]: 0       HAS CASE
         1        NO CASE
         2       HAS CASE
         3        NO CASE
         4       HAS CASE
                   ...
         2258     NO CASE
         2259    HAS CASE
         2260     NO CASE
         2261    HAS CASE
         2262     NO CASE
         Name: CRIMINAL CASES, Length: 2263, dtype: object
```

Column ‘ASSETS’ has string values. So we have to convert these column values to float numbers before creating the model. Initially, the column looks like this:

```
In [62]: LS2019Cnd[['ASSETS']]
```

Out[62]:

	ASSETS
0	Rs 30,99,414\n ~ 30 Lacs+
1	Rs 1,84,77,888\n ~ 1 Crore+
2	Rs 3,64,91,000\n ~ 3 Crore+
3	Rs 0\n ~ 0
4	Rs 7,42,74,036\n ~ 7 Crore+
...	...
2258	Rs 48,90,000\n ~ 48 Lacs+
2259	Rs 1,28,78,51,556\n ~ 128 Crore+
2260	Rs 90,36,63,001\n ~ 90 Crore+
2261	Rs 5,85,77,327\n ~ 5 Crore+
2262	Rs 0\n ~ 0

2263 rows × 1 columns

We have to remove all impurities from data such as ‘\n ~’, ‘\n~’, ‘\n’, ‘Rs’, separator commas(,) and the second range(we don’t need that). But it remains a String datatype. So we have to convert it to a float value.

```
In [73]: LS2019Cnd[['ASSETS']]
```

Out[73]:

	ASSETS
0	3.099414e+06
1	1.847789e+07
2	3.649100e+07
3	0.000000e+00
4	7.427404e+07
...	...
2258	4.890000e+06
2259	1.287852e+09
2260	9.036630e+08
2261	5.857733e+07
2262	0.000000e+00

2263 rows × 1 columns

Now, ‘ASSETS’ can be converted into Different Economic Classes such as ‘NEAR TO BPL’, ‘LOWER CLASS’, ‘LOWER MIDDLE CLASS’, ‘MIDDLE CLASS’, ‘UPPER MIDDLE CLASS’, ‘ELITE CLASS’, ‘SUPER RICH’, ‘RICHEST OF RICH’ and ‘NO ASSETS’. Introducing a new column ‘STATUS’ which holds different economic classes.

```
In [78]: LS2019Cnd[['ASSETS','STATUS']]
```

Out[78]:

	ASSETS	STATUS
0	3.099414e+06	MIDDLE CLASS
1	1.847789e+07	UPPER MIDDLE CLASS
2	3.649100e+07	UPPER MIDDLE CLASS
3	0.000000e+00	NO ASSETS
4	7.427404e+07	UPPER MIDDLE CLASS
...	...	...
2258	4.890000e+06	MIDDLE CLASS
2259	1.287852e+09	RICHEST OF RICH
2260	9.036630e+08	SUPER RICH
2261	5.857733e+07	UPPER MIDDLE CLASS
2262	0.000000e+00	NO ASSETS

2263 rows × 2 columns

Creating an ‘AGE_GROUP’ column based on ‘AGE’.

```
In [86]: LS2019Cnd[['AGE','AGE_GROUP']]
```

Out[86]:

	AGE	AGE_GROUP
0	52.0	MIDDLE AGE
1	54.0	MIDDLE AGE
2	52.0	MIDDLE AGE
3	0.0	NOT KNOWN
4	58.0	MIDDLE AGE
...	...	...
2258	43.0	MIDDLE AGE
2259	63.0	OLD AGE
2260	49.0	MIDDLE AGE
2261	47.0	MIDDLE AGE
2262	0.0	NOT KNOWN

2263 rows × 2 columns

Label Encoding:

	STATE	CONSTITUENCY	NAME	PARTY	SYMBOL	GENDER	CRIMINAL CASES	CATEGORY	EDUCATION	TOTAL VOTES	TOTAL ELECTORS	STATUS	AGE_GROUP	WINNER
0	Telangana	ADILABAD	SOYAM BAPU RAO	BJP	Lotus	MALE	HAS CASE	ST	12th Pass	377374	1489790	MIDDLE CLASS	MIDDLE AGE	1
1	Telangana	ADILABAD	Godam Nagesh	TRS	Car	MALE	NO CASE	ST	Post Graduate	318814	1489790	UPPER MIDDLE CLASS	MIDDLE AGE	0
2	Telangana	ADILABAD	RATHOD RAMESH	INC	Hand	MALE	HAS CASE	ST	12th Pass	314238	1489790	UPPER MIDDLE CLASS	MIDDLE AGE	0
3	Telangana	ADILABAD	NOTA	NOTA	NO SYMBOL	NOT APPLICABLE	NO CASE	NOT APPLICABLE	NOT APPLICABLE	13036	1489790	NO ASSETS	NOT KNOWN	0
4	Uttar Pradesh	AGRA	Satyapal Singh Baghel	BJP	Lotus	MALE	HAS CASE	SC	Doctorate	646875	1937690	UPPER MIDDLE CLASS	MIDDLE AGE	1

Now, let's consider the following data:

In this example, the 'STATE', 'CONSTITUENCY', 'NAME', 'PARTY', 'SYMBOL', 'GENDER', 'CRIMINAL CASES', 'CATEGORY', 'EDUCATION', 'STATUS', 'AGE_GROUP', which is all text. As you might know by now, we can't have text in our data if we're going to run any kind of model on it. So before we can run a model, we need to make this data ready for the model.

And to convert this kind of categorical text data into model-understandable numerical data, we use the Label Encoder class. So all we have to do, to label encode the first column, is import the Label Encoder class from the sklearn library, fit and transform the first column of the data, and then replace the existing text data with the new encoded data. Let's have a look at the code.

In [91]:
```
from sklearn.preprocessing import LabelEncoder
labelEncoder_X = LabelEncoder()
LS2019Cnd['STATE'] = labelEncoder_X.fit_transform(LS2019Cnd['STATE'])
LS2019Cnd['CONSTITUENCY'] = labelEncoder_X.fit_transform(LS2019Cnd['CONSTITUENCY'])
LS2019Cnd['NAME'] = labelEncoder_X.fit_transform(LS2019Cnd['NAME'])
LS2019Cnd['PARTY'] = labelEncoder_X.fit_transform(LS2019Cnd['PARTY'])
LS2019Cnd['SYMBOL'] = labelEncoder_X.fit_transform(LS2019Cnd['SYMBOL'])
LS2019Cnd['GENDER'] = labelEncoder_X.fit_transform(LS2019Cnd['GENDER'])
LS2019Cnd['CRIMINAL CASES'] = labelEncoder_X.fit_transform(LS2019Cnd['CRIMINAL CASES'])
LS2019Cnd['CATEGORY'] = labelEncoder_X.fit_transform(LS2019Cnd['CATEGORY'])
LS2019Cnd['EDUCATION'] = labelEncoder_X.fit_transform(LS2019Cnd['EDUCATION'])
LS2019Cnd['STATUS'] = labelEncoder_X.fit_transform(LS2019Cnd['STATUS'])
LS2019Cnd['AGE_GROUP'] = labelEncoder_X.fit_transform(LS2019Cnd['AGE_GROUP'])
```

Output:

	STATE	CONSTITUENCY	NAME	PARTY	SYMBOL	GENDER	CRIMINAL CASES	CATEGORY	EDUCATION	TOTAL VOTES	TOTAL ELECTORS	STATUS	AGE_GROUP	WINNER
0	31	0	1713	26	80	1	0	3	1	377374	1489790	4	0	1
1	31	0	700	120	32	1	1	3	12	318814	1489790	9	0	0
2	31	0	1498	46	66	1	0	3	1	314238	1489790	9	0	0
3	31	0	1203	81	87	2	1	1	9	13036	1489790	6	1	0
4	33	1	1789	26	80	1	0	2	4	646875	1937690	9	0	1

8. Model Accuracy:

Confusion Matrix:

A confusion matrix is a tabular summary of the number of correct and incorrect predictions made by a classifier. It can be used to evaluate the performance of a classification model through the calculation of performance metrics like accuracy, precision, recall, and F1-score.

Suppose that a classifier produces the following results:

Predicted Outcomes	Actual Outcomes
class a	class a
class b	class b
class c	class c
class a	class c
class b	class a

Code

The following code snippet shows how to create a confusion matrix and calculate some important metrics using a Python library called scikit-learn.

```
# Importing the dependancies
from sklearn import metrics
# Predicted values
y_pred = ["a", "b", "c", "a", "b"]
# Actual values
y_act = ["a", "b", "c", "c", "a"]
# Printing the confusion matrix
# The columns will show the instances predicted for each label,
# and the rows will show the actual number of instances for each label.
print(metrics.confusion_matrix(y_act, y_pred, labels=["a", "b", "c"]))
# Printing the precision and recall, among other metrics
print(metrics.classification_report(y_act, y_pred, labels=["a",
"b","c"]))
```

Output

```
[[1 1 0]
 [0 1 0]
 [1 0 1]]
              precision    recall  f1-score   support

           a       0.50      0.50      0.50         2
           b       0.50      1.00      0.67         1
           c       1.00      0.50      0.67         2

    accuracy                           0.60         5
   macro avg       0.67      0.67      0.61         5
weighted avg       0.70      0.60      0.60         5
```

Explanation

y_pred is a list that holds the predicted labels. y_act contains the actual labels.

metrics.confusion_matrix() takes in the list of actual labels, the list of predicted labels, and an optional argument to specify the order of the labels. It calculates the confusion matrix for the given inputs.

metrics.classification_report() takes in the list of actual labels, the list of predicted labels, and an optional argument to specify the order of the labels. It calculates performance metrics like precision, recall, and support.

❖ SOURCE CODE

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score
, recall_score, f1_score, average_precision_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
import plotly.express as px
import seaborn as sb
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
%matplotlib inline


#Logistic Regression
from sklearn.linear_model import LogisticRegression

#Decision Tree
from sklearn.tree import DecisionTreeClassifier

#Random Forest
from sklearn.ensemble import RandomForestClassifier


# Reading 2009 candidate dataset

LS09Cand = pd.read_csv('../Project/LS2009Candidate.csv')
print(LS09Cand.shape)
LS09Cand.head()
```

Out[2]:

	ST_CODE	State name	Month	Year	PC Number	PC name	PC Type	Candidate Name	Candidate Sex	Candidate Category	Candidate Age	Party Abbreviation	Total Votes Polled	Position
0	S01	Andhra Pradesh	3	2009	1	Adilabad	ST	RATHOD RAMESH	M	ST	43.0	TDP	372268.0	1.0
1	S01	Andhra Pradesh	3	2009	1	Adilabad	ST	KOTNAK RAMESH	M	ST	39.0	INC	257181.0	2.0
2	S01	Andhra Pradesh	3	2009	1	Adilabad	ST	MESRAM NAGO RAO	M	ST	59.0	PRAP	112930.0	3.0
3	S01	Andhra Pradesh	3	2009	1	Adilabad	ST	ADE TUKARAM	M	ST	55.0	BJP	57931.0	4.0
4	S01	Andhra Pradesh	3	2009	1	Adilabad	ST	RATHOD SADASHIV NAIK	M	ST	50.0	BSP	16471.0	5.0

```
# Reading 2014 candidate dataset

LS14Cand = pd.read_csv('../Project/LS2014Candidate.csv')
print(LS14Cand.shape)
LS14Cand.head()
```

(8794, 14)

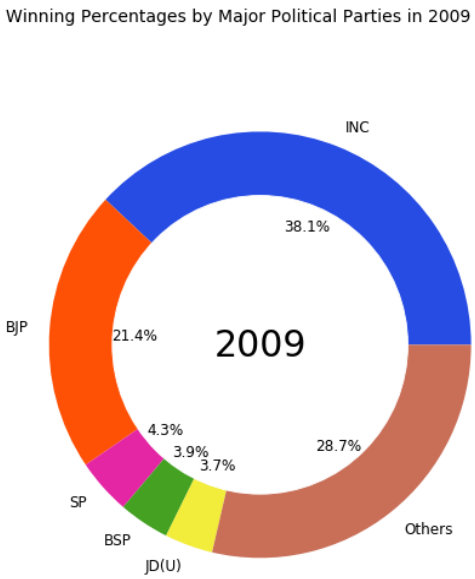
Out[3]:

	ST_CODE	State name	Month	Year	PC Number	PC name	PC Type	Candidate Name	Candidate Sex	Candidate Category	Candidate Age	Party Abbreviation	Total Votes Polled	Position
0	S01	Andhra Pradesh	5	2014	1	Adilabad	ST	GODAM NAGESH	M	ST	49.0	TRS	430847	1
1	S01	Andhra Pradesh	5	2014	1	Adilabad	ST	NARESH	M	ST	37.0	INC	259557	2
2	S01	Andhra Pradesh	5	2014	1	Adilabad	ST	RAMESH RATHOD	M	ST	48.0	TDP	184198	3
3	S01	Andhra Pradesh	5	2014	1	Adilabad	ST	RATHOD SADASHIV	M	ST	55.0	BSP	94420	4
4	S01	Andhra Pradesh	5	2014	1	Adilabad	ST	NETHAWATH RAMDAS	M	ST	44.0	IND	41032	5

```
# 2009 Party wise seat winners

Seatswin09 = LS09Cand[LS09Cand['Position'] == 1]
#Seatswin09
df09 = Seatswin09['Party Abbreviation'].value_counts().head().to_dict()
#df09
totalvote09 = sum(Seatswin09['Party Abbreviation'].value_counts().tolist()
)
#sum09
df09['Others'] = totalvote09 - sum(Seatswin09['Party Abbreviation'].value_
counts().head().tolist())
df09
```

Out[4]: {'INC': 206, 'BJP': 116, 'SP': 23, 'BSP': 21, 'JD(U)': 20, 'Others': 155}

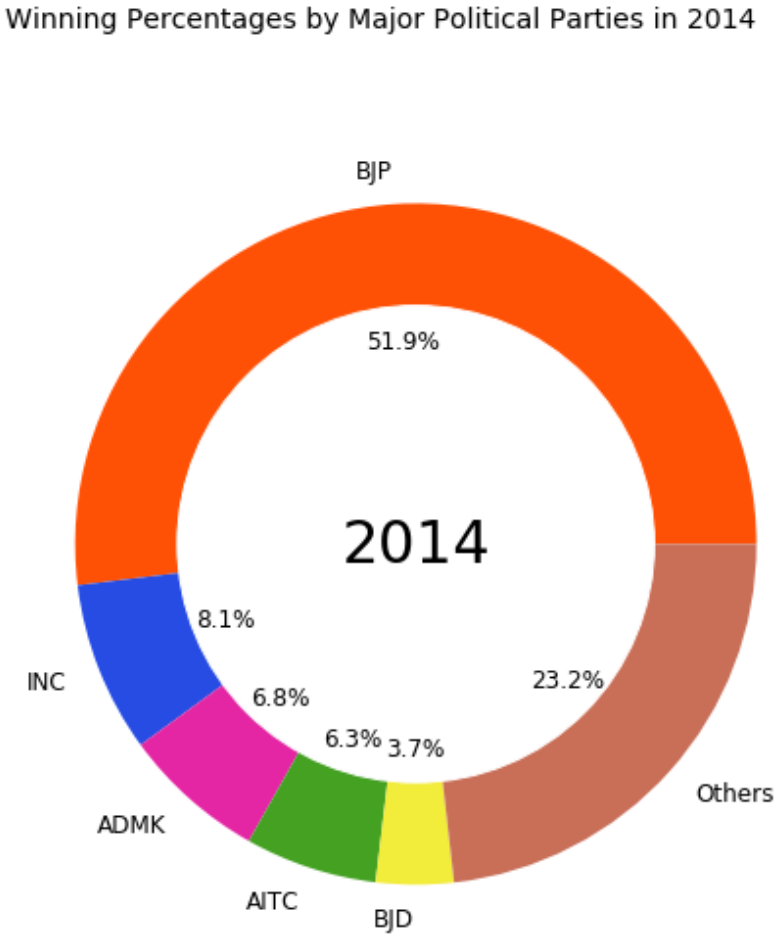
```
# 2009 Winning Percentages by Major Political Parties

colors = ("#264CE4", "#FF5106", "#E426A4", "#44A122", "#F2EC3A", "#C96F58")
plt.figure(figsize=(20,8))
plt.pie(df09.values(), labels=df09.keys(), colors=colors, autopct='%1.1f%%', text
props={'fontsize': 12})
my_circle1=plt.Circle( (0,0), 0.7, color='white')
fig = plt.gcf()
fig.suptitle("Winning Percentages by Major Political Parties in 2009", fontsize=1
4) # Adding supertitle with pyplot import
ax = fig.gca()
ax.add_patch(my_circle1)

label = ax.annotate("2009", xy=(0, 0), fontsize=30, ha="center",va="center")

ax.axis('off')
ax.set_aspect('equal')
ax.autoscale_view()
```




```
# 2014 Party wise seat winners

Seatswin14 = LS14Cand[LS14Cand['Position'] == 1]
#Seatswin14
df14 = Seatswin14['Party Abbreviation'].value_counts().head().to_dict()
#df14
totalvote14 = sum(Seatswin14['Party Abbreviation'].value_counts().tolist())
#sum14
df14['Others'] = totalvote14 - sum(Seatswin14['Party Abbreviation'].value_cou
nts().head().tolist())
df14
```

```
Out[6]: {'BJP': 282, 'INC': 44, 'ADMK': 37, 'AITC': 34, 'BJD': 20, 'Others': 126}
```

```
# 2014 Winning Percentages by Major Political Parties

colors = ("#FF5106", "#264CE4", "#E426A4", "#44A122", "#F2EC3A", "#C96F58")
plt.figure(figsize=(20,8))
plt.pie(df14.values(), labels=df14.keys(), colors=colors, autopct='%1.1f%%',
textprops={'fontsize': 12})
my_circle2=plt.Circle( (0,0), 0.7, color='white')
fig = plt.gcf()
fig.suptitle("Winning Percentages by Major Political Parties in 2014", fontsi
ze=14) # Adding supertitle with pyplot import
ax = fig.gca()
ax.add_patch(my_circle2)

label = ax.annotate("2014", xy=(0, 0), fontsize=30, ha="center",va="center")

ax.axis('off')
ax.set_aspect('equal')
ax.autoscale_view()
```




```
# Concatenating 2009 & 2014 Candidate datasets on one another

LS0914Cand = pd.concat([LS09Cand,LS14Cand])
print(LS0914Cand.shape)
LS0914Cand.head()
```

```
(16864, 14)
```

Out[8]:

	ST_CODE	State name	Month	Year	PC Number	PC name	PC Type	Candidate Name	Candidate Sex	Candidate Category	Candidate Age	Party Abbreviation	Total Votes Polled	Position
0	S01	Andhra Pradesh	3	2009	1	Adilabad	ST	RATHOD RAMESH	M	ST	43.0	TDP	372268.0	1.0
1	S01	Andhra Pradesh	3	2009	1	Adilabad	ST	KOTNAK RAMESH	M	ST	39.0	INC	257181.0	2.0
2	S01	Andhra Pradesh	3	2009	1	Adilabad	ST	MESRAM NAGO RAO	M	ST	59.0	PRAP	112930.0	3.0
3	S01	Andhra Pradesh	3	2009	1	Adilabad	ST	ADE TUKARAM	M	ST	55.0	BJP	57931.0	4.0
4	S01	Andhra Pradesh	3	2009	1	Adilabad	ST	RATHOD SADASHIV NAIK	M	ST	50.0	BSP	16471.0	5.0

```
#Checking all political parties abbreviation so that we can make it conscise
by including alliance & significant parties

LS0914Cand['Party Abbreviation'].unique()
```

Out[9]:
```
array(['TDP', 'INC', 'PRAP', 'BJP', 'BSP', 'IND', 'TRS', 'MCPI(S)',
       'PPOI', 'RPI(A)', 'LSP', 'TPPP', 'IJP', 'BSP(AP)', 'BPD', 'SP',
       'RKSP', 'MUL', 'ANC', 'UWF', 'SAP', 'AJBP', 'AIMIM', 'JD(S)',
       'MANP', 'GRIP', 'SJP(R)', 'CPI', 'CPM', 'RJD', 'BSSP', 'BHSASP',
       'JD(U)', 'RPI', 'RDHP', 'RDMP', 'CPI(ML)(L)', 'PBHP', 'RPI(KH)',
       'RPC(S)', 'BCUF', 'BJSH', 'RRS', nan, 'LB', 'AC', 'PPA', 'AUDF',
       'ASDC', 'NCP', 'RWS', 'BOPF', 'AGP', 'RVNP', 'LJP', 'RCPI(R)',
       'RSPS', 'AIMF', 'JMM', 'BVM', 'AIFB', 'SHS', 'LTSD', 'BJKVP',
       'BLPGL', 'BJJD', 'RSP', 'RPP', 'JGP', 'KSVP', 'RKJP', 'RJJM',
       'ABAS', 'ABJS', 'RSWD', 'AD', 'RMEP', 'PTSS', 'LPSP', 'SBSP',
       'SJTP', 'BMF', 'SLP(L)', 'KVSP', 'ABDBM', 'NBNP', 'BHJAP',
       'BSP(K)', 'JVM', 'BSKP', 'LM', 'JPS', 'EKSP', 'BUDM', 'BJKD',
       'JKM', 'PMSP', 'SSD', 'AJSP', 'RLD', 'STPI', 'MAG', 'UGDP', 'SGF',
       'BNJD', 'MJP', 'LSWP', 'VHS', 'NLHP', 'KKJHS', 'NSCP', 'RSP(S)',
       'ABMSD', 'SVPP', 'RPIE', 'ADSP', 'HJCBL', 'INLD', 'JKNPP', 'RASJP',
       'SMBHP', 'BHBP', 'BRPP', 'NELU', 'VAJP', 'RASAP', 'JCP', 'BHC',
       'NSSP', 'AIFB(S)', 'RND', 'JJJKMC', 'RJAP', 'AIRP', 'JUP', 'RRD',
       'JKN', 'JKPDP', 'JPC', 'JKANC', 'BCDP', 'BSKRP', 'RNSP', 'DGPP',
       'BBP', 'SKP', 'PRCP', 'BHPP', 'NDEP', 'ICSP', 'ABHM', 'AIJMK',
       'BPJP', 'BSC', 'KTMK', 'KCVP', 'THPI', 'KEC', 'VCK', 'KEC(M)',
       'SWJP', 'AITC', 'BSA', 'SVSP', 'BMM', 'IVD', 'ASP', 'RSMD', 'PRSP',
       'GMS', 'GGP', 'BJBP', 'ABHKP', 'AIC', 'YVP', 'LKSP', 'RPI(D)',
       'BMSM', 'BBM', 'LKSGM', 'HJP', 'KM', 'SVRP', 'DESEP', 'RP(K)',
       'IUML', 'PRBP', 'ARP', 'BREM', 'DPI', 'STBP', 'ABMP', 'JSS', 'JP',
       'LVKP', 'MNS', 'BVA', 'PRPI', 'PPIS', 'IBSP', 'PG', 'RPPI', 'SWP',
       'MPP', 'RBCP', 'PDA', 'UDP', 'HSPDP', 'MDP', 'ACNC', 'NPF', 'BJD',
       'KOKD', 'JDP', 'SAMO', 'JHKP', 'OMM', 'KS', 'BOP', 'BGTD', 'DCP',
       'SAD', 'AIDWC', 'DBSP', 'ARWP', 'LBP', 'SAD(M)', 'BVP', 'PLP',
       'MB(S)P', 'ABSR', 'ABJP', 'RJVP', 'RMGLMP', 'RBD', 'BCP', 'BSSPA',
       'FCI', 'RDSD', 'ABCD(A)', 'ABMSKP', 'SDF', 'SHRP', 'SJEP', 'SGPP',
       'ADMK', 'DMK', 'DMDK', 'MMKA', 'PKMK', 'DPK', 'JJ', 'KDC', 'MAMAK',
       'YSP', 'ABKMM', 'PMK', 'AIVP', 'PNK', 'ADSMK', 'NMK', 'KNMK',
       'UMK', 'MDMK', 'CDF', 'PT', 'AMB', 'PECP', 'BD', 'RSBP', 'JSP',
       'UNLP', 'LD', 'RMSP', 'MKUP', 'JKD', 'BPC', 'ABLTP', 'BKLJP', 'JM',
       'AP', 'RYS', 'ABRS', 'RTKP', 'RVP', 'BSK', 'MC', 'MADP', 'BSKPB',
```

```
#Introducing Alliance column for optimized substitution of Winning Party Abbr
eviation column

UPA_list = ['INC','NCP', 'RJD', 'DMK', 'IUML', 'JMM','JD(s)','KC(M)','RLD','R
SP','CMP(J)','KC(J)','PPI','MD']
NDA_list = ['BJP','SS', 'LJP', 'SAD', 'RLSP', 'AD','PMK','NPP','AINRC','NPF',
'RPI(A)','BPF','JD(U)','SDF','NDPP','MNF','RIDALOS','KMDK','IJK','PNK','JSP',
'GJM','MGP','GFP','GVP','AJSU','IPFT','MPP','KPP','JKPC','KC(T)','BDJS','AGP'
,'JSS','PPA','UDP','HSPDP','PSP','JRS','KVC','PNP','SBSP','KC(N)','PDF','MDPF
']
Others_list = ['YSRCP','AAAP', 'IND', 'AIUDF', 'BLSP', 'JKPDP', 'JD(S)', 'INL
D', 'CPI', 'AIMIM', 'KEC(M)','SWP', 'NPEP', 'JKN', 'AIFB', 'MUL', 'AUDF', 'BO
PF', 'BVA', 'HJCBL', 'JVM','MDMK']
LS0914Cand['Alliance']=LS0914Cand['Party Abbreviation']
LS0914Cand['Alliance']=LS0914Cand['Alliance'].replace(to_replace=UPA_list,val
ue='UPA')
LS0914Cand['Alliance']=LS0914Cand['Alliance'].replace(to_replace=NDA_list,val
ue='NDA')
LS0914Cand['Alliance']=LS0914Cand['Alliance'].replace(to_replace=Others_list,
value='Others')
```

```
LS0914Cand
```

Out[10]:

ST_CODE	State name	Month	Year	PC Number	PC name	PC Type	Candidate Name	Candidate Sex	Candidate Category	Candidate Age	Party Abbreviation	Total Votes Polled	Position	Alliance
S01	Andhra Pradesh	3	2009	1	Adilabad	ST	RATHOD RAMESH	M	ST	43.0	TDP	372268.0	1.0	TDP
S01	Andhra Pradesh	3	2009	1	Adilabad	ST	KOTNAK RAMESH	M	ST	39.0	INC	257181.0	2.0	UPA
S01	Andhra Pradesh	3	2009	1	Adilabad	ST	MESRAM NAGO RAO	M	ST	59.0	PRAP	112930.0	3.0	PRAP
S01	Andhra Pradesh	3	2009	1	Adilabad	ST	ADE TUKARAM	M	ST	55.0	BJP	57931.0	4.0	NDA
S01	Andhra Pradesh	3	2009	1	Adilabad	ST	RATHOD SADASHIV NAIK	M	ST	50.0	BSP	16471.0	5.0	BSP
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
U07	Puducherry	5	2014	1	Puducherry	GEN	PUVALA NAGESWARA RAO	M	GEN	60.0	IND	465.0	27.0	Others
U07	Puducherry	5	2014	1	Puducherry	GEN	G. PALANI	M	GEN	59.0	CPI(ML)(L)	438.0	28.0	CPI(ML)(L)
U07	Puducherry	5	2014	1	Puducherry	GEN	MARIE UTHRIANATHAN	M	GEN	32.0	SAP	366.0	29.0	SAP
U07	Puducherry	5	2014	1	Puducherry	GEN	S. CHITRAKALA	F	GEN	37.0	JD(U)	309.0	30.0	NDA
U07	Puducherry	5	2014	1	Puducherry	GEN	P. DHANARASU @ MATHIMAHARAJA	M	GEN	50.0	IND	198.0	31.0	Others

```
# Winning seats distribution by Major Political Parties & Alliances for 2009 & 2014

SeatsWin = LS0914Cand[(LS0914Cand.Position==1)].groupby(['Alliance','Year'])['Position'].sum().reset_index().pivot(index='Alliance', columns='Year',values='Position').reset_index().fillna(0).sort_values([2014,2009], ascending=False).reset_index(drop = True)
SeatsWin
```

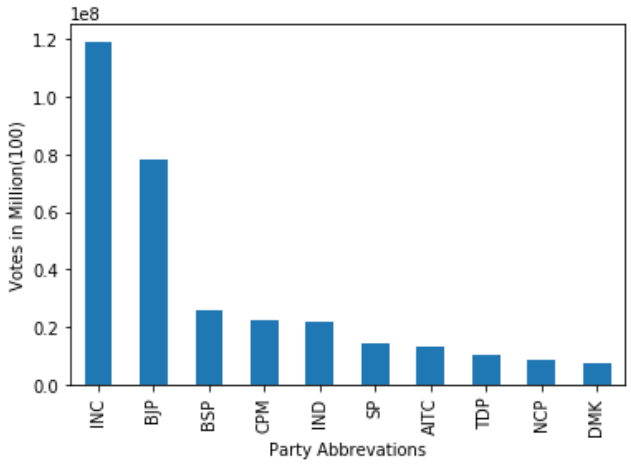
Out[11]:

Year	Alliance	2009	2014
0	NDA	143.0	300.0
1	UPA	245.0	59.0
2	ADMK	9.0	37.0
3	Others	32.0	34.0
4	AITC	19.0	34.0
5	BJD	14.0	20.0
6	SHS	11.0	18.0
7	TDP	6.0	16.0
8	TRS	2.0	11.0
9	CPM	16.0	9.0
10	SP	23.0	5.0
11	BSP	21.0	0.0

```
#VOTE SHARE BY MAJOR POLITICAL PARTIES IN 2009
votespartywise09 = LS09Cand.groupby('Party Abbreviation')['Total Votes Polled'].sum()
x09 = votespartywise09.sort_values(ascending=False)[:10].plot(kind="bar")
x09.set_xlabel('Party Abbrevations')
```

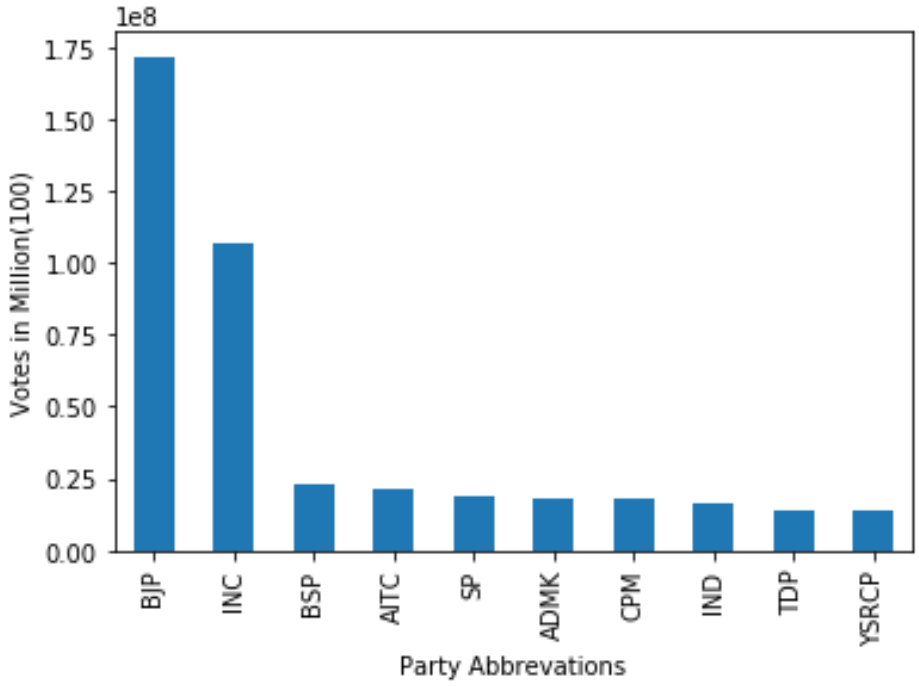
```
x09.set_ylabel('Votes in Million(100)')
votespartywise09.sort_values(ascending=False)[:10]
```

```
Out[13]: Party Abbreviation
         INC     119111019.0
         BJP      78175832.0
         BSP      25718188.0
         CPM      22219111.0
         IND      21577849.0
         SP       14283708.0
         AITC     13356510.0
         TDP      10481659.0
         NCP       8521502.0
         DMK       7226655.0
         Name: Total Votes Polled, dtype: float64
```



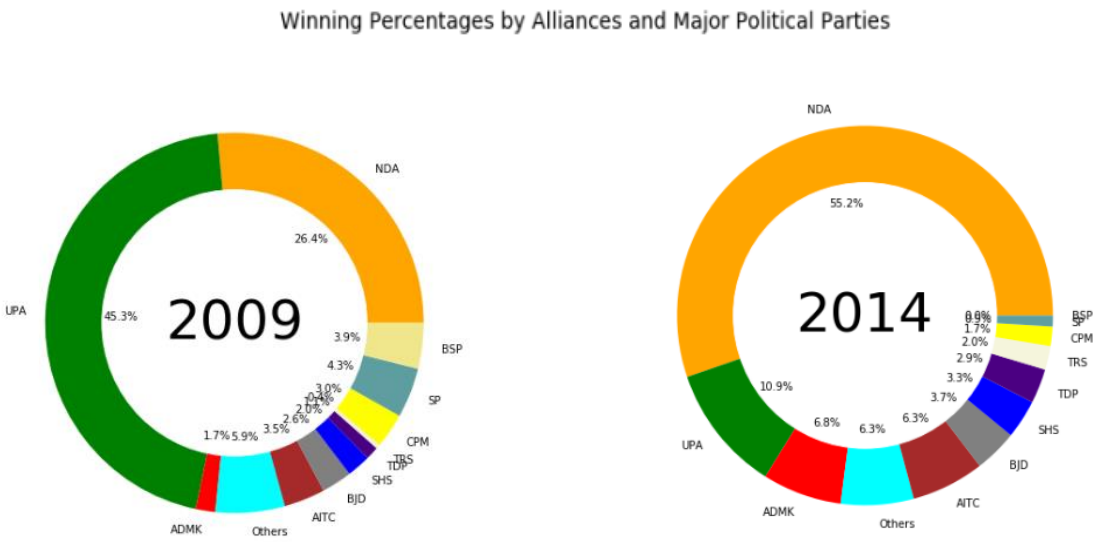
```
#VOTE SHARE BY MAJOR POLITICAL PARTIES IN 2014
votespartywise14 = LS14Cand.groupby('Party Abbreviation')['Total Votes Polled
'].sum()
x14 = votespartywise14.sort_values(ascending=False)[:10].plot(kind="bar")
x14.set_xlabel('Party Abbrevations')
x14.set_ylabel('Votes in Million(100)')
votespartywise14.sort_values(ascending=False)[:10]
```

```
Out[14]: Party Abbreviation
         BJP      171660230
         INC      106935942
         BSP       22946346
         AITC      21262665
         SP        18673089
         ADMK      18111579
         CPM       17988955
         IND       16737720
         TDP       14099230
         YSRCP     13995435
         Name: Total Votes Polled, dtype: int64
```




```
#Seats won by Alliances and Major Political Parties

colors  = ("orange", "green", "red", "cyan", "brown", "grey", "blue", "indigo
", "beige", "yellow","cadetblue","khaki")
plt.figure(figsize=(20,8))
plt.subplot(1,2,1)
plt.pie(SeatsWin[2009], labels=SeatsWin['Alliance'], colors=colors,autopct='%
1.1f%%')
my_circle1=plt.Circle( (0,0), 0.7, color='white')
fig = plt.gcf() # get current figure
fig.suptitle("Winning Percentages by Alliances and Major Political Parties",
fontsize=14) # Adding supertitle with pyplot import
ax = fig.gca() # get current axis
ax.add_patch(my_circle1)

label = ax.annotate("2009", xy=(0,0), fontsize=50, ha="center",va="center")
```

```
ax.axis('off')
ax.set_aspect('equal')
ax.autoscale_view()
plt.subplot(1,2,2)
plt.pie(SeatsWin[2014], labels=SeatsWin['Alliance'], colors=colors,autopct='%
1.1f%%')
my_circle2=plt.Circle( (0,0), 0.7, color='white')
fig = plt.gcf()
ax = fig.gca()
ax.add_patch(my_circle2)

label = ax.annotate("2014", xy=(0, 0), fontsize=50, ha="center",va="center")

ax.axis('off')
ax.set_aspect('equal')
ax.autoscale_view()

plt.show();
```



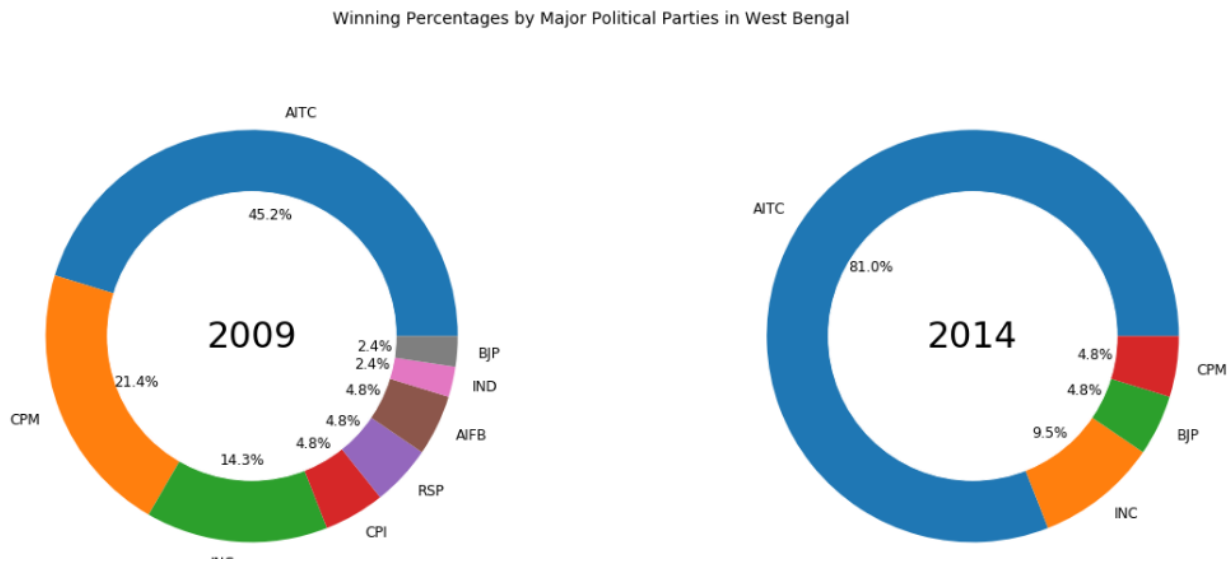

```
Seatswin09 = LS09Cand[LS09Cand['State name']=='West Bengal']
Seatswin09 = Seatswin09[Seatswin09['Position']==1]
df09 = Seatswin09['Party Abbreviation'].value_counts().to_dict()

Seatswin14 = LS14Cand[LS14Cand['State name']=='West Bengal']
Seatswin14 = Seatswin14[Seatswin14['Position']==1]
df14 = Seatswin14['Party Abbreviation'].value_counts().to_dict()

df09
```

```
Out[14]: {'AITC': 19,
          'CPM': 9,
          'INC': 6,
          'CPI': 2,
          'RSP': 2,
          'AIFB': 2,
          'IND': 1,
          'BJP': 1}
```

```
df14
```

```
Out[15]: {'AITC': 34, 'INC': 4, 'BJP': 2, 'CPM': 2}
```

```
# 2009 West Bengal

plt.figure(figsize=(20,8))
plt.subplot(1,2,1)
plt.pie(df09.values(), labels=df09.keys(), autopct='%1.1f%%', textprops={'fon
tsize': 12})
my_circle1=plt.Circle( (0,0), 0.7, color='white')
fig = plt.gcf()
fig.suptitle("Winning Percentages by Major Political Parties in West Bengal",
 fontsize=14) # Adding supertitle with pyplot import
ax = fig.gca()
ax.add_patch(my_circle1)

label = ax.annotate("2009", xy=(0, 0), fontsize=30, ha="center",va="center")

ax.axis('off')
ax.set_aspect('equal')
ax.autoscale_view()
# 2014 West Bengal

plt.subplot(1,2,2)
plt.pie(df14.values(), labels=df14.keys(), autopct='%1.1f%%', textprops={'fon
tsize': 12})
my_circle2=plt.Circle( (0,0), 0.7, color='white')
fig = plt.gcf()
ax = fig.gca()
ax.add_patch(my_circle2)

label = ax.annotate("2014", xy=(0, 0), fontsize=30, ha="center",va="center")

ax.axis('off')
ax.set_aspect('equal')
ax.autoscale_view()
```




```
Seatswin09 = LS09Cand[LS09Cand['State name']=='Uttar Pradesh']
Seatswin09 = Seatswin09[Seatswin09['Position']==1]
dfup09 = Seatswin09['Party Abbreviation'].value_counts().to_dict()

Seatswin14 = LS14Cand[LS14Cand['State name']=='Uttar Pradesh']
Seatswin14 = Seatswin14[Seatswin14['Position']==1]
dfup14 = Seatswin14['Party Abbreviation'].value_counts().to_dict()

dfup09
```

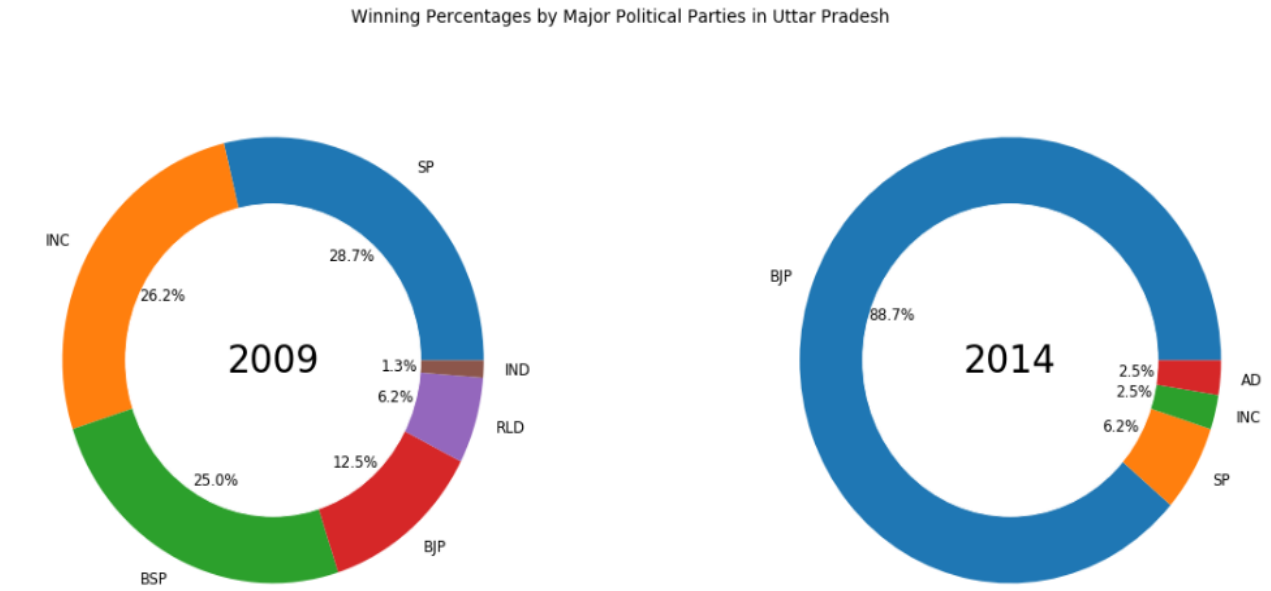
```
Out[18]: {'SP': 23, 'INC': 21, 'BSP': 20, 'BJP': 10, 'RLD': 5, 'IND': 1}
```

```
dfup14
```

```
Out[19]: {'BJP': 71, 'SP': 5, 'INC': 2, 'AD': 2}
```

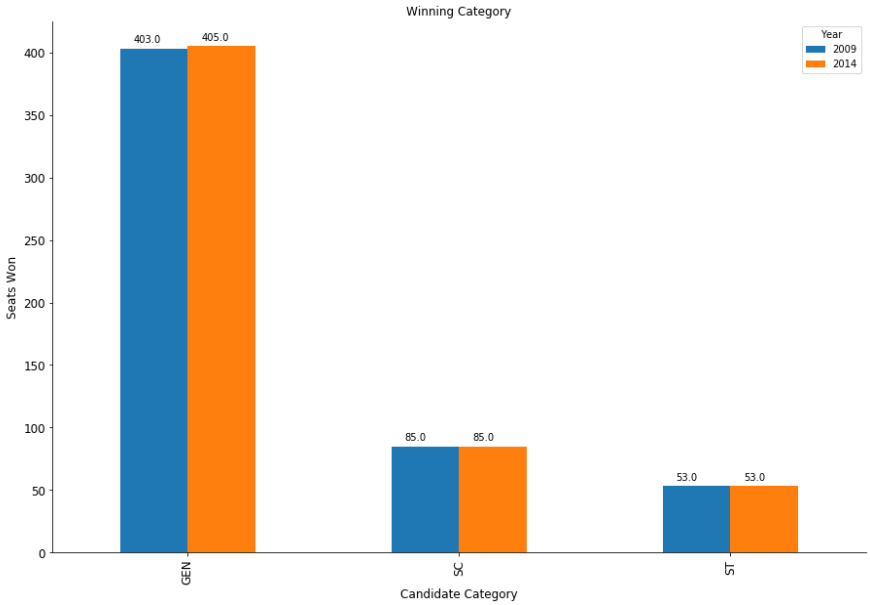
```
# 2009 Uttar Pradesh

colors = ("#FF0000", "blue", "#44A122", "orange", "cyan","green")
plt.figure(figsize=(20,8))
plt.subplot(1,2,1)
plt.pie(dfup09.values(), labels=dfup09.keys(), autopct='%1.1f%%',textprops={'
fontsize': 12})
my_circle1=plt.Circle( (0,0), 0.7, color='white')
fig = plt.gcf()
fig.suptitle("Winning Percentages by Major Political Parties in Uttar Pradesh
", fontsize=14) # Adding supertitle with pyplot import
ax = fig.gca()
ax.add_patch(my_circle1)
```

```
label = ax.annotate("2009", xy=(0, 0), fontsize=30, ha="center",va="center")

ax.axis('off')
ax.set_aspect('equal')
ax.autoscale_view()


# 2014 Uttar Pradesh

colors = ("orange", "#FF0000", "blue", "#44A122")
plt.subplot(1,2,2)
plt.pie(dfup14.values(), labels=dfup14.keys(), autopct='%1.1f%%', textprops={
'fontsize': 12})
my_circle2=plt.Circle( (0,0), 0.7, color='white')
fig = plt.gcf()
ax = fig.gca()
ax.add_patch(my_circle2)

label = ax.annotate("2014", xy=(0, 0), fontsize=30, ha="center",va="center")

ax.axis('off')
ax.set_aspect('equal')
ax.autoscale_view()
```



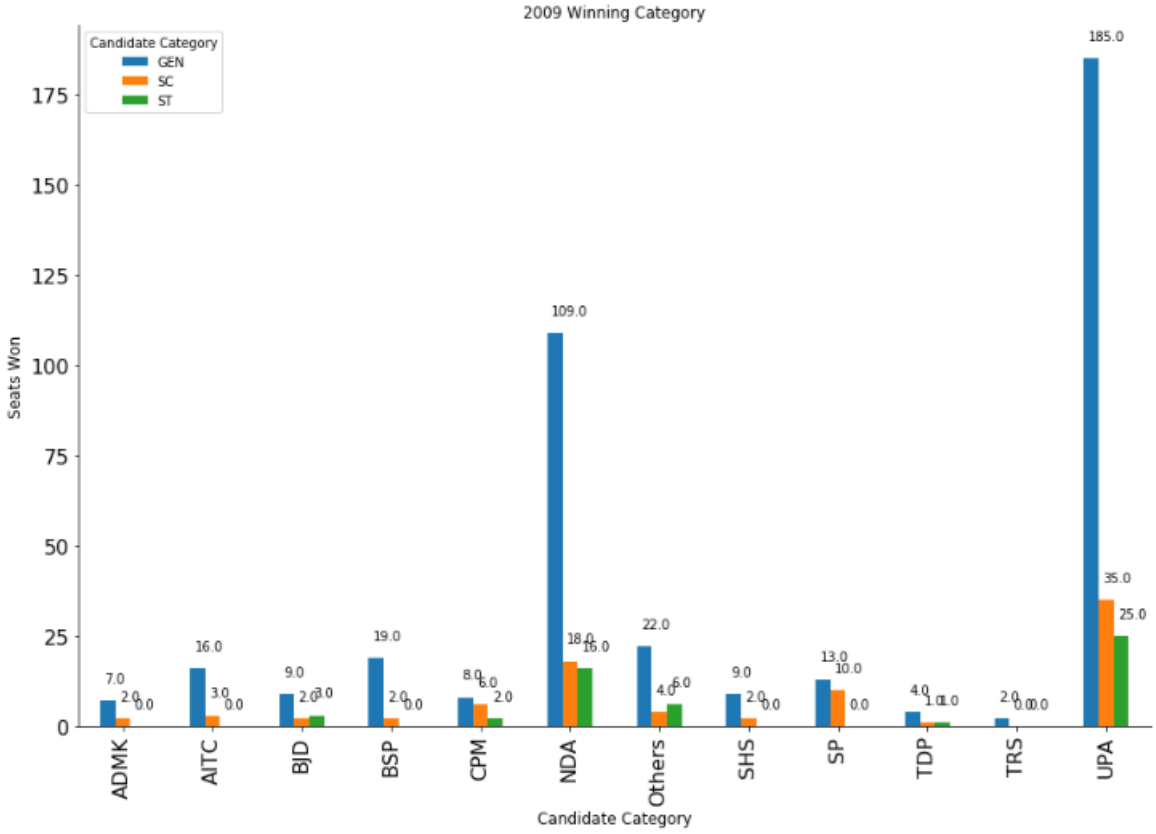

```
# function to add value label to plot
def annot_plot(ax,w,h):
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    for p in ax.patches:
        ax.annotate('{}'.format(p.get_height()), (p.get_x()+w, p.get_height()
+h))
```

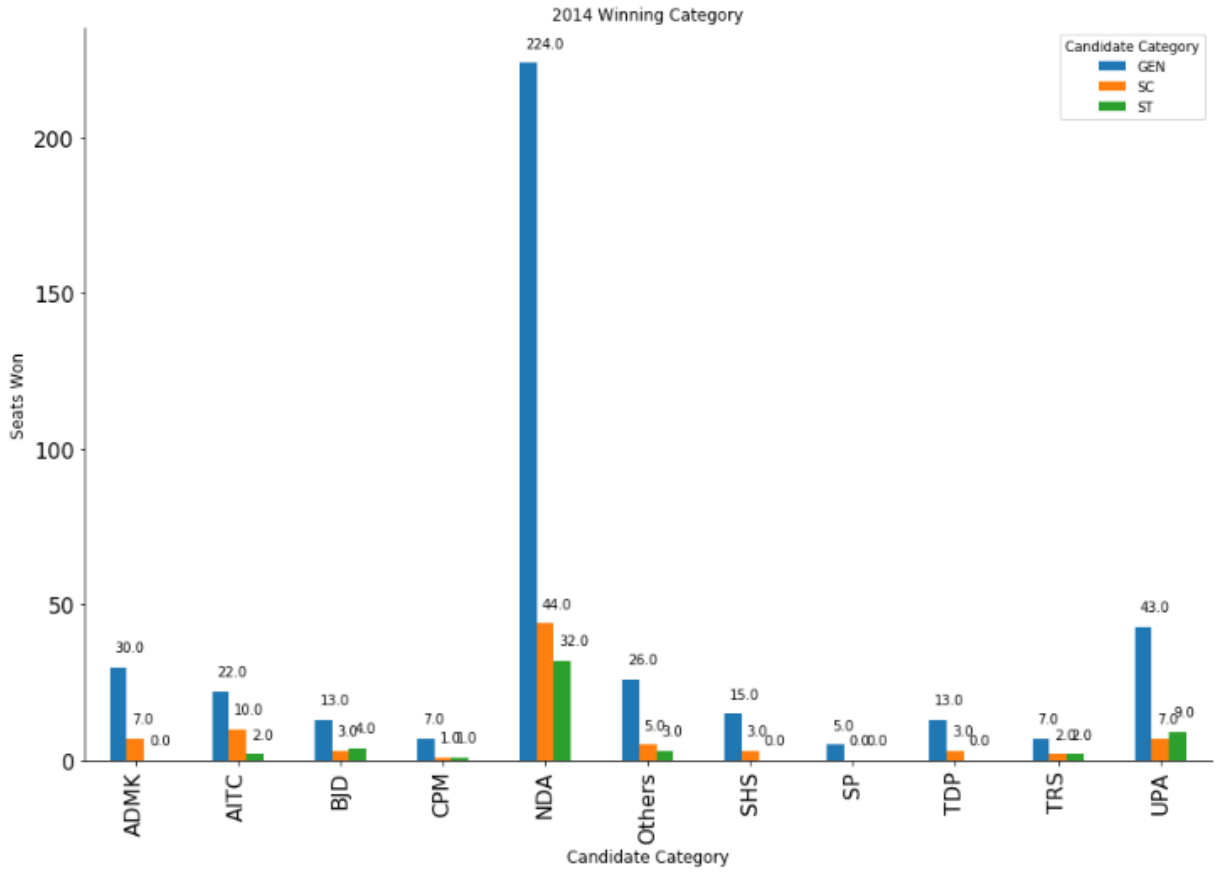
```
CatWin = LS0914Cand[(LS0914Cand.Position==1)].groupby(['Candidate Category',
'Year'])['Position'].sum().reset_index().pivot(index='Candidate Category', co
lumns='Year',values='Position').reset_index().fillna(0).sort_values([2014,200
9], ascending=False).reset_index(drop = True)


nx = CatWin.plot(kind='bar', title ="Winning Category", figsize=(15, 10), leg
end=True, fontsize=12)
nx.set_xlabel("Candidate Category", fontsize=12)
nx.set_ylabel("Seats Won", fontsize=12)

# Modifying Axis Labels
labels = [item.get_text() for item in nx.get_xticklabels()]
labels[0] = 'GEN'
labels[1]= 'SC'
labels[2]='ST'
nx.set_xticklabels(labels)
annot_plot(nx,0.05,5)
```



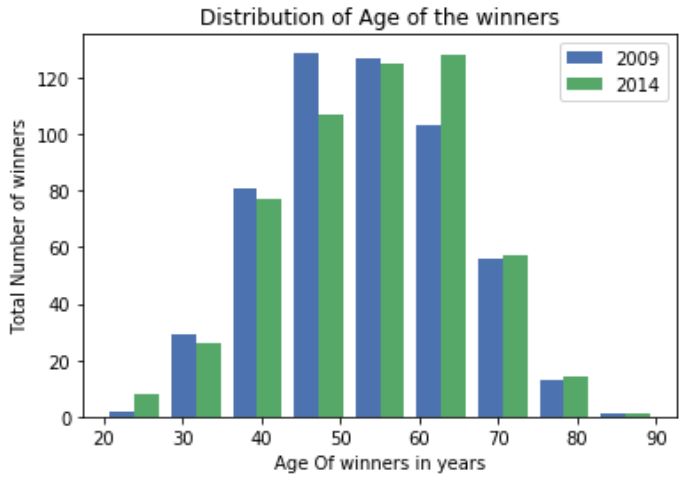

```
CatAlliance09 = LS0914Cand[(LS0914Cand.Position==1) & (LS0914Cand.Year==2009)
].groupby(['Alliance','Candidate Category'])['Position'].sum().unstack().rese
t_index().fillna(0)
CatAlliance14 = LS0914Cand[(LS0914Cand.Position==1) & (LS0914Cand.Year==2014)
].groupby(['Alliance','Candidate Category'])['Position'].sum().unstack().rese
t_index().fillna(0)

nx = CatAlliance09.plot(kind='bar', title ="2009 Winning Category", figsize=(
15, 10), legend=True, fontsize=16)
nx.set_xlabel("Candidate Category", fontsize=12)
nx.set_ylabel("Seats Won", fontsize=12)
# Modifying Axis Labels
```

```
labels = [item.get_text() for item in nx.get_xticklabels()]
labels[0:11] = CatAlliance09['Alliance']
nx.set_xticklabels(labels)

annot_plot(nx,0.05,5)

nx = CatAlliance14.plot(kind='bar', title ="2014 Winning Category", figsize=(
15, 10), legend=True, fontsize=16)
nx.set_xlabel("Candidate Category", fontsize=12)
nx.set_ylabel("Seats Won", fontsize=12)
# Modifying Axis Labels
labels = [item.get_text() for item in nx.get_xticklabels()]
labels[0:11] = CatAlliance14['Alliance']
nx.set_xticklabels(labels)

annot_plot(nx,0.05,5)
```

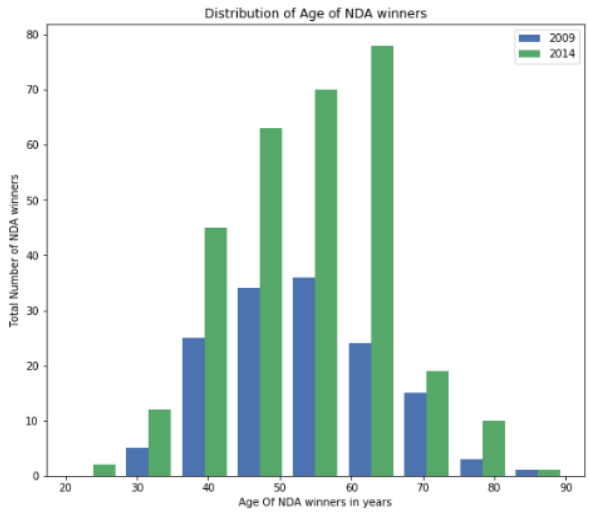


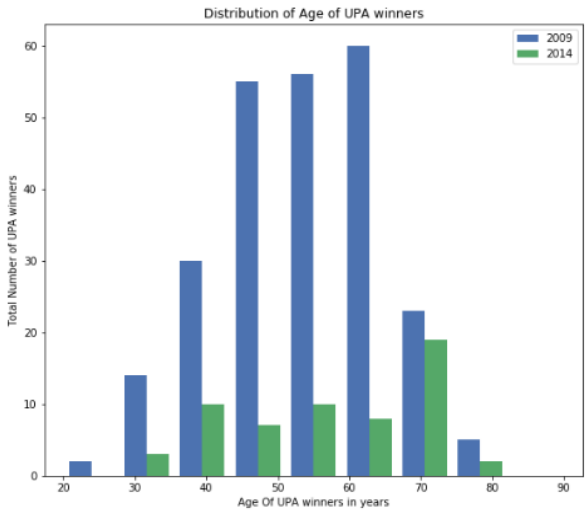



```
plt.style.use('seaborn-deep')

Age09=LS0914Cand[(LS0914Cand.Position==1) & (LS0914Cand.Year==2009)]['Candida
te Age'].tolist()
Age14=LS0914Cand[(LS0914Cand.Position==1) & (LS0914Cand.Year==2014)]['Candida
te Age'].tolist()
bins = np.linspace(20, 90, 10)
plt.hist([Age09, Age14], bins, label=['2009', '2014'])
plt.legend(loc='upper right')
plt.xlabel('Age Of winners in years')
plt.ylabel('Total Number of winners')
plt.title('Distribution of Age of the winners')
plt.show()
```



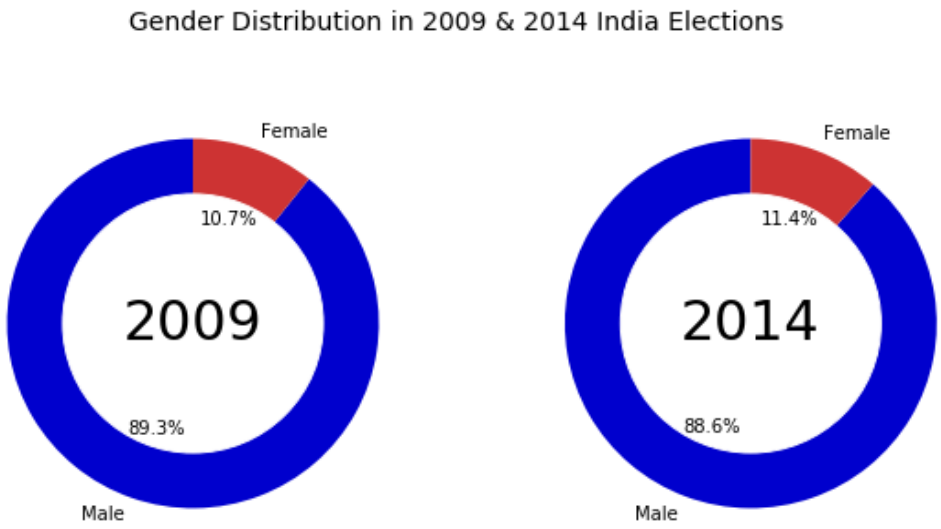
```
# Age Distribution of Winning Candidates in 2009 & 2014 for NDA & UPA in Indi
a Elections

plt.figure(figsize=(20,8))
plt.subplot(1,2,1)
plt.style.use('seaborn-deep')

Age09UPA=LS0914Cand[(LS0914Cand.Position==1) & (LS0914Cand.Year==2009)& (LS09
14Cand.Alliance=='UPA')]['Candidate Age'].tolist()
Age14UPA=LS0914Cand[(LS0914Cand.Position==1) & (LS0914Cand.Year==2014)& (LS09
14Cand.Alliance=='UPA')]['Candidate Age'].tolist()
Age09NDA=LS0914Cand[(LS0914Cand.Position==1) & (LS0914Cand.Year==2009)& (LS09
14Cand.Alliance=='NDA')]['Candidate Age'].tolist()
Age14NDA=LS0914Cand[(LS0914Cand.Position==1) & (LS0914Cand.Year==2014)& (LS09
14Cand.Alliance=='NDA')]['Candidate Age'].tolist()

bins = np.linspace(20, 90, 10)
plt.hist([Age09NDA, Age14NDA], bins, label=['2009', '2014'])
plt.legend(loc='upper right')
plt.xlabel('Age Of NDA winners in years')
plt.ylabel('Total Number of NDA winners')
plt.title('Distribution of Age of NDA winners')


plt.subplot(1,2,2)
bins = np.linspace(20, 90, 10)
plt.hist([Age09UPA, Age14UPA], bins, label=['2009', '2014'])
plt.legend(loc='upper right')
plt.xlabel('Age Of UPA winners in years')
plt.ylabel('Total Number of UPA winners')
plt.title('Distribution of Age of UPA winners')

plt.show();
```



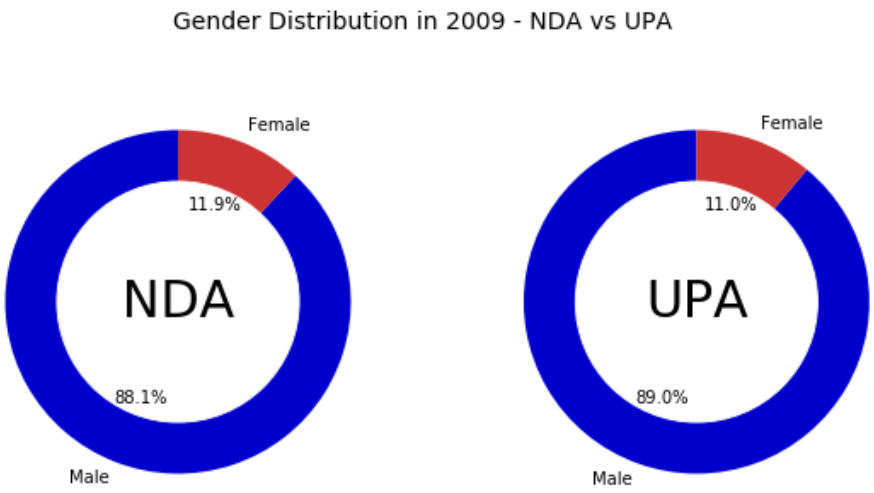



```
# Gender Distribution of Winning Candidates in 2009 & 2014 India Elections
colors = ['#0000CD','#CD3333']
```

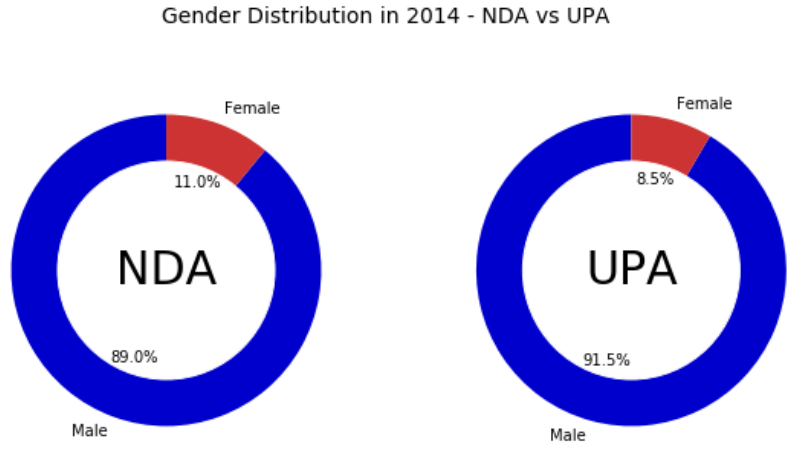
```
plt.figure(figsize=(10,5))
plt.subplot(1,2,1)
plt.pie(LS0914Cand[(LS0914Cand.Position==1) & (LS0914Cand.Year==2009)]['Candi
date Sex'].value_counts(), labels=['Male','Female'],autopct='%1.1f%%',colors=
colors, startangle=90)
my_circle1=plt.Circle( (0,0), 0.7, color='white')
fig = plt.gcf()
fig.suptitle("Gender Distribution in 2009 & 2014 India Elections", fontsize=1
4) # Adding supertitle with pyplot import
ax = fig.gca()
ax.add_patch(my_circle1)
label = ax.annotate("2009", xy=(0, 0), fontsize=30, ha="center",va="center")
ax.axis('off')
ax.set_aspect('equal')
ax.autoscale_view()

plt.subplot(1,2,2)
plt.pie(LS0914Cand[(LS0914Cand.Position==1) & (LS0914Cand.Year==2014)]['Candi
date Sex'].value_counts(), labels=['Male','Female'],autopct='%1.1f%%',colors=
colors, startangle=90)
my_circle2=plt.Circle( (0,0), 0.7, color='white')
fig = plt.gcf() #gcf means get current figure

ax = fig.gca() # gca means get current axis
ax.add_patch(my_circle2)

label = ax.annotate("2014", xy=(0, 0), fontsize=30, ha="center",va="center")
ax.axis('off')
ax.set_aspect('equal')
ax.autoscale_view()
plt.show();
```



```
# Gender Distribution of Winning Candidates in 2009 - NDA vs UPA in India Ele
ctions
colors = ['#0000CD','#CD3333']
plt.figure(figsize=(10,5))
plt.subplot(1,2,1)
plt.pie(LS0914Cand[(LS0914Cand.Position==1) & (LS0914Cand.Year==2009)& (LS091
4Cand.Alliance=='NDA')]['Candidate Sex'].value_counts(), labels=['Male','Fema
le'],autopct='%1.1f%%',colors=colors, startangle=90)
my_circle1=plt.Circle( (0,0), 0.7, color='white')
fig = plt.gcf()
fig.suptitle("Gender Distribution in 2009 - NDA vs UPA", fontsize=14) # Addin
g supertitle with pyplot import
ax = fig.gca()
ax.add_patch(my_circle1)
label = ax.annotate("NDA", xy=(0, 0), fontsize=30, ha="center",va="center")
ax.axis('off')
ax.set_aspect('equal')
ax.autoscale_view()

plt.subplot(1,2,2)
plt.pie(LS0914Cand[(LS0914Cand.Position==1) & (LS0914Cand.Year==2009)& (LS091
4Cand.Alliance=='UPA')]['Candidate Sex'].value_counts(), labels=['Male','Fema
le'],autopct='%1.1f%%',colors=colors, startangle=90)
my_circle2=plt.Circle( (0,0), 0.7, color='white')
fig = plt.gcf() #gcf means get current figure
ax = fig.gca() # gca means get current axis
ax.add_patch(my_circle2)
label = ax.annotate("UPA", xy=(0, 0), fontsize=30, ha="center",va="center")

ax.axis('off')
ax.set_aspect('equal')
ax.autoscale_view()
plt.show();
```



```
# Gender Distribution of Winning Candidates in 2014 - NDA vs UPA in India Ele
ctions

colors = ['#0000CD','#CD3333']
plt.figure(figsize=(10,5))
plt.title('2014')
plt.subplot(1,2,1)
plt.pie(LS0914Cand[(LS0914Cand.Position==1) & (LS0914Cand.Year==2014)& (LS091
4Cand.Alliance=='NDA')]['Candidate Sex'].value_counts(), labels=['Male','Fema
le'],autopct='%1.1f%%',colors=colors, startangle=90)
my_circle1=plt.Circle( (0,0), 0.7, color='white')
fig = plt.gcf()
fig.suptitle("Gender Distribution in 2014 - NDA vs UPA", fontsize=14) # Addin
g supertitle with pyplot import
ax = fig.gca()
ax.add_patch(my_circle1)
label = ax.annotate("NDA", xy=(0, 0), fontsize=30, ha="center",va="center")
ax.axis('off')
ax.set_aspect('equal')
ax.autoscale_view()

plt.subplot(1,2,2)
plt.pie(LS0914Cand[(LS0914Cand.Position==1) & (LS0914Cand.Year==2014)& (LS091
4Cand.Alliance=='UPA')]['Candidate Sex'].value_counts(), labels=['Male','Fema
le'],autopct='%1.1f%%',colors=colors, startangle=90)
my_circle2=plt.Circle( (0,0), 0.7, color='white')
fig = plt.gcf() #gcf means get current figure
ax = fig.gca() # gca means get current axis
ax.add_patch(my_circle2)

label = ax.annotate("UPA", xy=(0, 0), fontsize=30, ha="center",va="center")
ax.axis('off')
ax.set_aspect('equal')
ax.autoscale_view()
plt.show();
```




```
# Reading 2009 Electors dataset
LS09Elec = pd.read_csv('../Project/LS2009Electors.csv')
```

```
print(LS09Elec.shape)
LS09Elec.head()
```

```
(543, 8)
```

Out[28]:

	STATE CODE	STATE	PC NO	PARLIAMENTARY CONSTITUENCY	Total voters	Total_Electors	TOT_CONTESTANT	POLL PERCENTAGE
0	S01	Andhra Pradesh	1	Adilabad	864165	1131211	9	76.39
1	S01	Andhra Pradesh	2	Peddapalle	905332	1315642	15	68.81
2	S01	Andhra Pradesh	3	Karimnagar	990646	1496211	15	66.21
3	S01	Andhra Pradesh	4	Nizamabad	891508	1333271	12	66.87
4	S01	Andhra Pradesh	5	Zahirabad	1017372	1359566	10	74.83

```
LS09Elec.STATE.unique()
```

Out[29]:
```
array(['Andhra Pradesh', 'Arunachal Pradesh', 'Assam', 'Bihar', 'Goa',
       'Gujarat', 'Haryana', 'Himachal Pradesh', 'Jammu & Kashmir',
       'Karnataka', 'Kerala', 'Madhya Pradesh', 'Maharashtra', 'Manipur',
       'Meghalaya', 'Mizoram', 'Nagaland', 'Orissa', 'Punjab',
       'Rajasthan', 'Sikkim', 'Tamil Nadu', 'Tripura', 'Uttar Pradesh',
       'West Bengal', 'Chattisgarh', 'Jharkhand', 'Uttarakhand',
       'Andaman & Nicobar Islands', 'Chandigarh', 'Dadra & Nagar Haveli',
       'Daman & Diu', 'NCT OF Delhi', 'Lakshadweep', 'Puducherry'],
      dtype=object)
```

```
# Reading 2014 Electors dataset
LS14Elec = pd.read_csv('../Project/LS2014Electors.csv')
print(LS14Elec.shape)
LS14Elec.head()
```

```
(543, 7)
```

Out[30]:

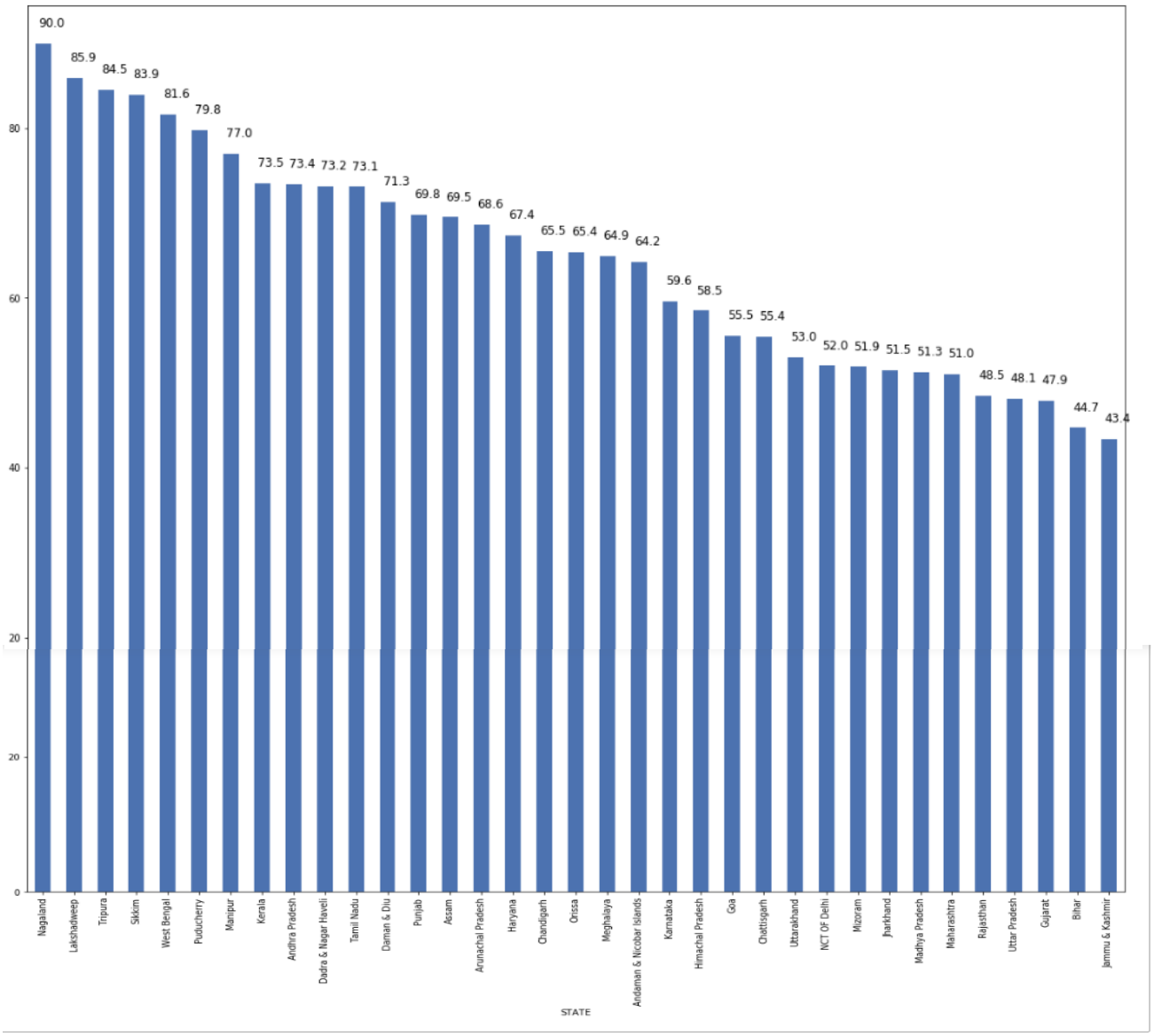
	STATE CODE	STATE	PC NO	PARLIAMENTARY CONSTITUENCY	Total voters	Total_Electors	POLL PERCENTAGE
0	S01	Andhra Pradesh	1	Adilabad	1055593	1386282	76.15
1	S01	Andhra Pradesh	2	Peddapalle	1025194	1425355	71.93
2	S01	Andhra Pradesh	3	Karimnagar	1127225	1550810	72.69
3	S01	Andhra Pradesh	4	Nizamabad	1034032	1496193	69.11
4	S01	Andhra Pradesh	5	Zahirabad	1099784	1445354	76.09

```
LS14Elec.STATE.unique()
```

Out[31]:
```
array(['Andhra Pradesh', 'Arunachal Pradesh', 'Assam', 'Bihar', 'Goa',
       'Gujarat', 'Haryana', 'Himachal Pradesh', 'Jammu & Kashmir',
       'Karnataka', 'Kerala', 'Madhya Pradesh', 'Maharashtra', 'Manipur',
       'Meghalaya', 'Mizoram', 'Nagaland', 'Odisha', 'Punjab',
       'Rajasthan', 'Sikkim', 'Tamil Nadu', 'Tripura', 'Uttar Pradesh',
       'West Bengal', 'Chhattisgarh', 'Jharkhand', 'Uttarakhand',
       'Andaman & Nicobar Islands', 'Chandigarh', 'Dadra & Nagar Haveli',
       'Daman & Diu', 'NCT OF Delhi', 'Lakshadweep', 'Puducherry'],
      dtype=object)
```

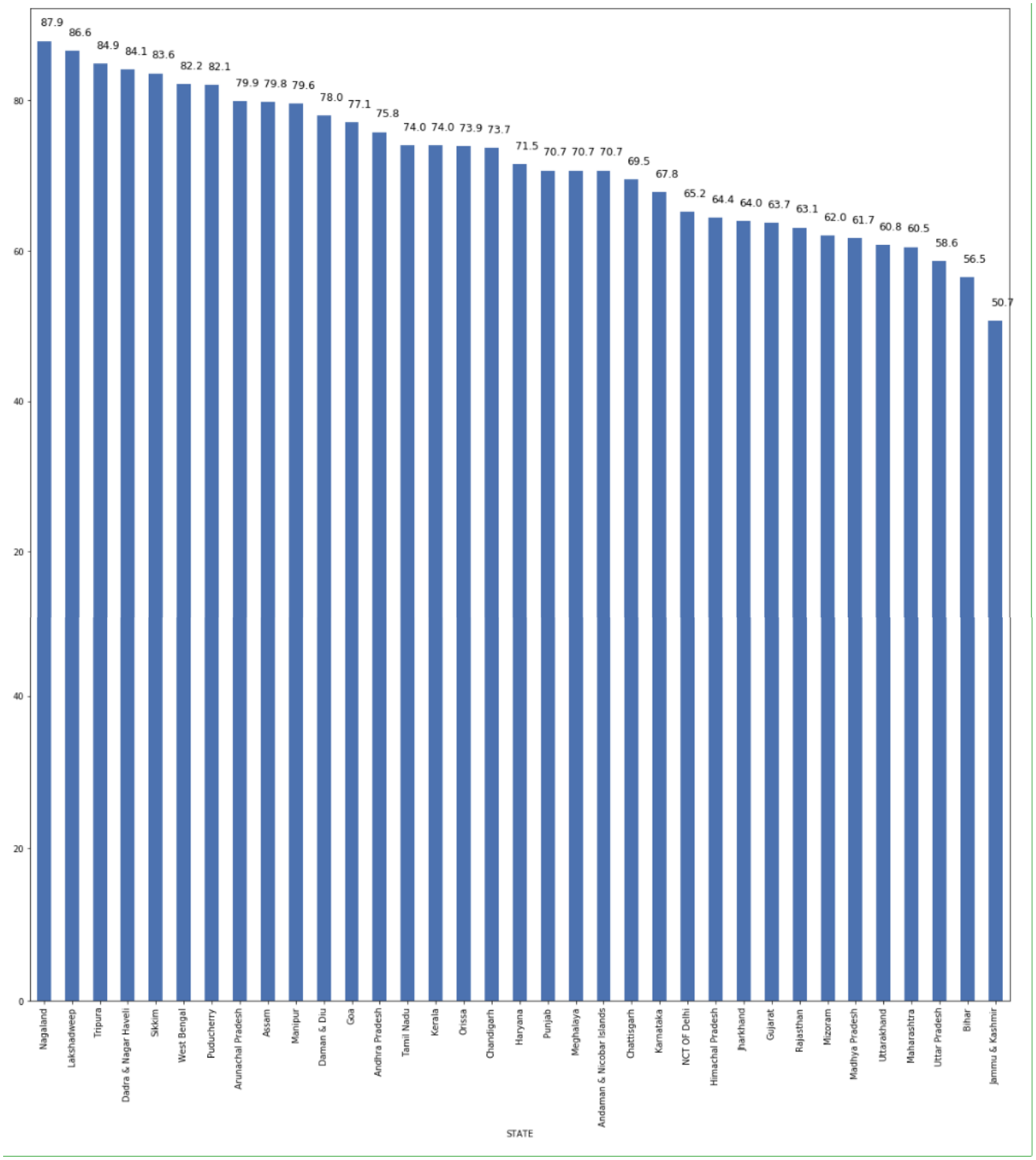
```
LS14Elec['STATE']=LS14Elec['STATE'].replace(to_replace=['Odisha'],value='Orissa')
```

```
LS14Elec['STATE']=LS14Elec['STATE'].replace(to_replace=['Chhattisgarh'],value
='Chattisgarh')
pollper09 = LS09Elec.groupby("STATE").mean()
LS09 = pollper09[['POLL PERCENTAGE']].round(1).sort_values('POLL PERCENTAGE',
ascending=False)
ax1 =LS09['POLL PERCENTAGE'].plot(kind='bar',figsize=(20, 15))
for p in ax1.patches:
    ax1.annotate(format(p.get_height()), (p.get_x()+0.1, p.get_height()+2),fo
ntsize=12)
```




```
pollper14 = LS14Elec.groupby("STATE").mean()
LS14 = pollper14[['POLL PERCENTAGE']].round(1).sort_values('POLL PERCENTAGE',
ascending=False)
```

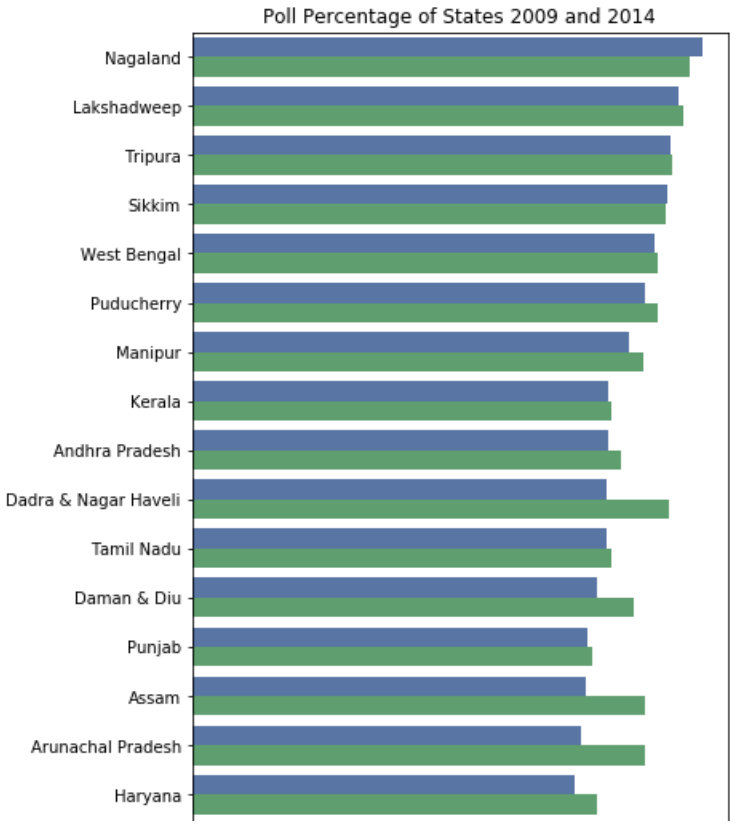
```
ax1 =LS14['POLL PERCENTAGE'].plot(kind='bar',figsize=(20, 15))
for p in ax1.patches:
    ax1.annotate(format(p.get_height()), (p.get_x()+0.1, p.get_height()+2),fo
ntsize=12)
```

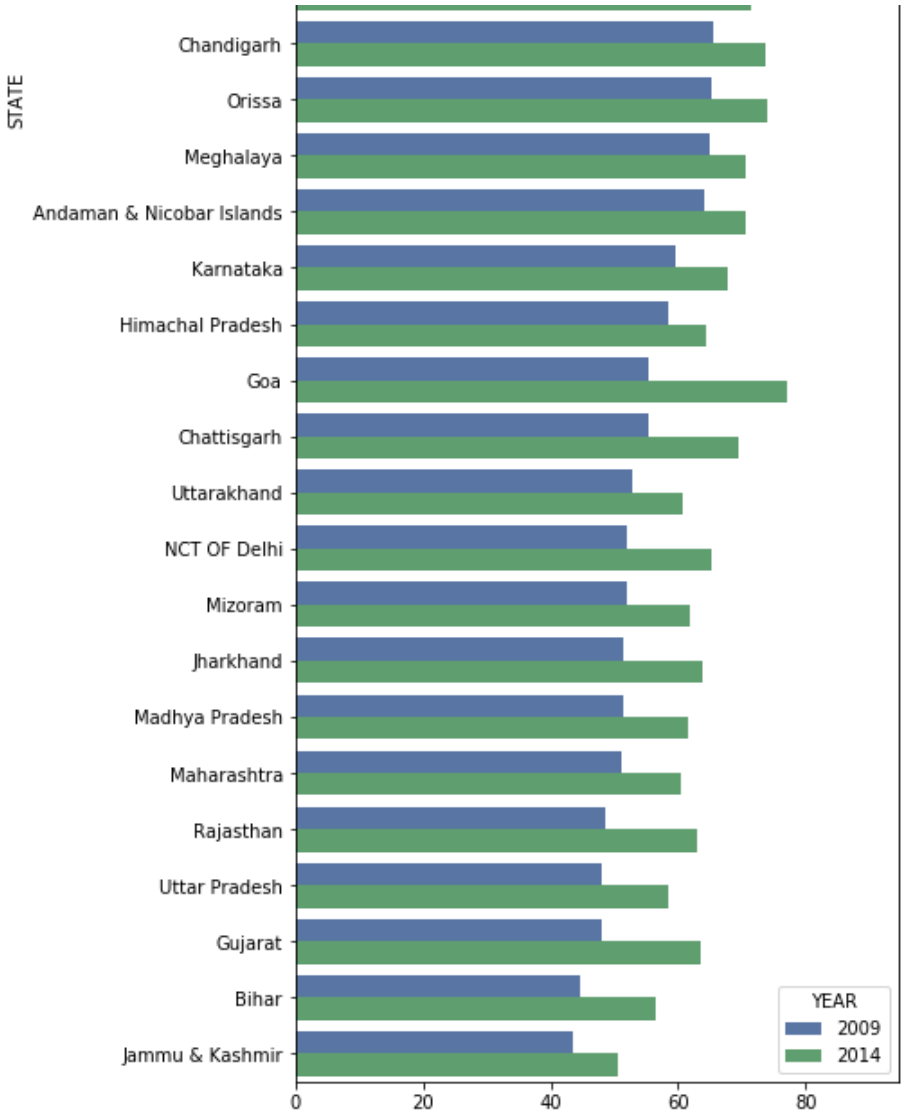



```
LS09Elec = LS09Elec.groupby('STATE').mean()
LS09 = LS09Elec[['POLL PERCENTAGE']].sort_values('POLL PERCENTAGE',ascending=
False).to_dict()
Y09=[2009 for i in range(35)]
```

```
S09=list(LS09['POLL PERCENTAGE'].keys())
P09=list(LS09['POLL PERCENTAGE'].values())


LS14Elec = LS14Elec.groupby('STATE').mean()
LS14 = LS14Elec[['POLL PERCENTAGE']].sort_values('POLL PERCENTAGE',ascending=
False).to_dict()
Y14=[2014 for i in range(35)]
S14=list(LS14['POLL PERCENTAGE'].keys())
P14=list(LS14['POLL PERCENTAGE'].values())
Data = {'YEAR':Y09+Y14,'STATE':S09+S14,'Poll_Percentage':P09+P14}
DF = pd.DataFrame(data=Data)
ax = plt.subplots(figsize=(6, 20))
sb.barplot(x=DF.Poll_Percentage,y=DF.STATE,hue=DF.YEAR)
plt.title('Poll Percentage of States 2009 and 2014')
```

```
Out[33]: Text(0.5, 1.0, 'Poll Percentage of States 2009 and 2014')
```






```
LS09Cand = pd.read_csv('../Project/LS2009Candidate.csv')
LS09Elec = pd.read_csv('../Project/LS2009Electors.csv')
```

```
LS09Cand.head()
```

Out[2]:

	ST_CODE	State name	Month	Year	PC Number	PC name	PC Type	Candidate Name	Candidate Sex	Candidate Category	Candidate Age	Party Abbreviation	Total Votes Polled	Position
0	S01	Andhra Pradesh	3	2009	1	Adilabad	ST	RATHOD RAMESH	M	ST	43.0	TDP	372268.0	1.0
1	S01	Andhra Pradesh	3	2009	1	Adilabad	ST	KOTNAK RAMESH	M	ST	39.0	INC	257181.0	2.0
2	S01	Andhra Pradesh	3	2009	1	Adilabad	ST	MESRAM NAGO RAO	M	ST	59.0	PRAP	112930.0	3.0
3	S01	Andhra Pradesh	3	2009	1	Adilabad	ST	ADE TUKARAM	M	ST	55.0	BJP	57931.0	4.0
4	S01	Andhra Pradesh	3	2009	1	Adilabad	ST	RATHOD SADASHIV NAIK	M	ST	50.0	BSP	16471.0	5.0

```
LS09Elec.head()
```

Out[36]:

	STATE CODE	STATE	PC NO	PARLIAMENTARY CONSTITUENCY	Total voters	Total_Electors	TOT_CONTESTANT	POLL PERCENTAGE
0	S01	Andhra Pradesh	1	Adilabad	864165	1131211	9	76.39
1	S01	Andhra Pradesh	2	Peddapalle	905332	1315642	15	68.81
2	S01	Andhra Pradesh	3	Karimnagar	990646	1496211	15	66.21
3	S01	Andhra Pradesh	4	Nizamabad	891508	1333271	12	66.87
4	S01	Andhra Pradesh	5	Zahirabad	1017372	1359566	10	74.83

```
LS09DF = pd.merge(left=LS09Cand, right=LS09Elec, left_on='PC name', right_on='PARLIAMENTARY CONSTITUENCY')
LS09DF.head()
```

Out[37]:

	ST_CODE	State name	Month	Year	PC Number	PC name	PC Type	Candidate Name	Candidate Sex	Candidate Category	...	Total Votes Polled	Position	STATE CODE	STATE	PC NO	PARLIAMENTARY CONSTITUENCY
0	S01	Andhra Pradesh	3	2009	1	Adilabad	ST	RATHOD RAMESH	M	ST	...	372268.0	1.0	S01	Andhra Pradesh	1	Adilaba
1	S01	Andhra Pradesh	3	2009	1	Adilabad	ST	KOTNAK RAMESH	M	ST	...	257181.0	2.0	S01	Andhra Pradesh	1	Adilaba
2	S01	Andhra Pradesh	3	2009	1	Adilabad	ST	MESRAM NAGO RAO	M	ST	...	112930.0	3.0	S01	Andhra Pradesh	1	Adilaba
3	S01	Andhra Pradesh	3	2009	1	Adilabad	ST	ADE TUKARAM	M	ST	...	57931.0	4.0	S01	Andhra Pradesh	1	Adilaba
4	S01	Andhra Pradesh	3	2009	1	Adilabad	ST	RATHOD SADASHIV NAIK	M	ST	...	16471.0	5.0	S01	Andhra Pradesh	1	Adilaba

5 rows × 22 columns

```
LS09DF = LS09DF.drop(['ST_CODE','Month','Year','PC Number','PC Type','STATE CODE','STATE','PC NO','PARLIAMENTARY CONSTITUENCY','Total voters','POLL PERCENTAGE','TOT_CONTESTANT'], axis=1)
LS09DF.columns
```

Out[38]:
```
Index(['State name', 'PC name', 'Candidate Name', 'Candidate Sex',
       'Candidate Category', 'Candidate Age', 'Party Abbreviation',
       'Total Votes Polled', 'Position', 'Total_Electors'],
      dtype='object')
```

```
LS09DF.isnull().sum().sort_values(ascending=False)
```

Out[39]:
```
Position              43
Total Votes Polled    43
Party Abbreviation    43
Candidate Age         43
Candidate Category    43
Candidate Sex         43
Total_Electors         0
Candidate Name         0
PC name                0
State name             0
dtype: int64
```

```
LS09DF = LS09DF.fillna({
    'Position':0.0,
```

```
    'Total Votes Polled':0.0,
    'Party Abbreviation':'NOT APPLICABLE',
    'Candidate Age':0.0,
    'Candidate Category':'NA',
    'Candidate Sex':'NA',
})
```

```
LS09DF.isnull().sum().sort_values(ascending=False)
```

Out[41]:
```
Total_Electors          0
Position                0
Total Votes Polled      0
Party Abbreviation      0
Candidate Age           0
Candidate Category      0
Candidate Sex           0
Candidate Name          0
PC name                 0
State name              0
dtype: int64
```

```
LS14Cand = pd.read_csv('../Project/LS2014Candidate.csv')
LS14Elec = pd.read_csv('../Project/LS2014Electors.csv')
```

```
LS14DF = pd.merge(left=LS14Cand, right=LS14Elec, left_on='PC name',
right_on='PARLIAMENTARY CONSTITUENCY')
```

```
LS14DF.head()
```

Out[43]:

	ST_CODE	State name	Month	Year	PC Number	PC name	PC Type	Candidate Name	Candidate Sex	Candidate Category	...	Party Abbreviation	Total Votes Polled	Position	STATE CODE	STATE	PC NO	P
0	S01	Andhra Pradesh	5	2014	1	Adilabad	ST	GODAM NAGESH	M	ST	...	TRS	430847	1	S01	Andhra Pradesh	1	
1	S01	Andhra Pradesh	5	2014	1	Adilabad	ST	NARESH	M	ST	...	INC	259557	2	S01	Andhra Pradesh	1	
2	S01	Andhra Pradesh	5	2014	1	Adilabad	ST	RAMESH RATHOD	M	ST	...	TDP	184198	3	S01	Andhra Pradesh	1	
3	S01	Andhra Pradesh	5	2014	1	Adilabad	ST	RATHOD SADASHIV	M	ST	...	BSP	94420	4	S01	Andhra Pradesh	1	
4	S01	Andhra Pradesh	5	2014	1	Adilabad	ST	NETHAWATH RAMDAS	M	ST	...	IND	41032	5	S01	Andhra Pradesh	1	

5 rows × 21 columns

```
LS14DF = LS14DF.drop(['ST_CODE','Month','Year','PC Number','PC Type','STATE C
ODE','STATE','PC NO','PARLIAMENTARY CONSTITUENCY','Total voters','POLL PERCEN
TAGE'], axis=1)
LS14DF.columns
```

Out[44]:
```
Index(['State name', 'PC name', 'Candidate Name', 'Candidate Sex',
       'Candidate Category', 'Candidate Age', 'Party Abbreviation',
       'Total Votes Polled', 'Position', 'Total_Electors'],
      dtype='object')
```

```
LS14DF.isnull().sum().sort_values(ascending=False)
```

```
Out[45]: Candidate Age          548
         Candidate Category     548
         Candidate Sex          548
         Total_Electors           0
         Position                 0
         Total Votes Polled       0
         Party Abbreviation       0
         Candidate Name           0
         PC name                  0
         State name               0
         dtype: int64
```

```
LS14DF = LS14DF.fillna({
    'Candidate Age':0.0,
    'Candidate Category':'NA',
    'Candidate Sex':'NA',
})
LS14DF.isnull().sum().sort_values(ascending=False)
```

```
Out[41]: Total_Electors          0
         Position                0
         Total Votes Polled      0
         Party Abbreviation      0
         Candidate Age           0
         Candidate Category      0
         Candidate Sex           0
         Candidate Name          0
         PC name                 0
         State name              0
         dtype: int64
```

```
In [48]: LS09DF.shape
Out[48]: (8160, 10)

In [49]: LS14DF.shape
Out[49]: (8898, 10)

In [50]: LS09DF.columns
Out[50]: Index(['State name', 'PC name', 'Candidate Name', 'Candidate Sex',
                'Candidate Category', 'Candidate Age', 'Party Abbreviation',
                'Total Votes Polled', 'Position', 'Total_Electors'],
               dtype='object')

In [51]: LS14DF.columns
Out[51]: Index(['State name', 'PC name', 'Candidate Name', 'Candidate Sex',
                'Candidate Category', 'Candidate Age', 'Party Abbreviation',
                'Total Votes Polled', 'Position', 'Total_Electors'],
               dtype='object')
```

```
for i in range(len(LS09DF)):
    LS09DF.iloc[i,0] = LS09DF.iloc[i,0].upper()    #Statename
    LS09DF.iloc[i,1] = LS09DF.iloc[i,1].upper()    #PC name
    LS09DF.iloc[i,2] = LS09DF.iloc[i,2].upper()    #Candidate Name
    LS09DF.iloc[i,6] = LS09DF.iloc[i,6].upper()    #Party Abbreviation

for i in range(len(LS14DF)):
```

```
    LS14DF.iloc[i,0] = LS14DF.iloc[i,0].upper()    #Statename
    LS14DF.iloc[i,1] = LS14DF.iloc[i,1].upper()    #PC name
    LS14DF.iloc[i,2] = LS14DF.iloc[i,2].upper()    #Candidate Name
    LS14DF.iloc[i,6] = LS14DF.iloc[i,6].upper()    #Party Abbreviation
```

LS09DF

Out[53]:

	State name	PC name	Candidate Name	Candidate Sex	Candidate Category	Candidate Age	Party Abbreviation	Total Votes Polled	Position	Total_Electors
0	ANDHRA PRADESH	ADILABAD	RATHOD RAMESH	M	ST	43.0	TDP	372268.0	1.0	1131211
1	ANDHRA PRADESH	ADILABAD	KOTNAK RAMESH	M	ST	39.0	INC	257181.0	2.0	1131211
2	ANDHRA PRADESH	ADILABAD	MESRAM NAGO RAO	M	ST	59.0	PRAP	112930.0	3.0	1131211
3	ANDHRA PRADESH	ADILABAD	ADE TUKARAM	M	ST	55.0	BJP	57931.0	4.0	1131211
4	ANDHRA PRADESH	ADILABAD	RATHOD SADASHIV NAIK	M	ST	50.0	BSP	16471.0	5.0	1131211
...	...	...	...	...	...	...	...	...	...	...
8155	PUDUCHERRY	PUDUCHERRY	DR. R. NARAYANAN	M	GEN	59.0	IND	422.0	24.0	762440
8156	PUDUCHERRY	PUDUCHERRY	MUPPADAI MATHIMAHARAJA	M	GEN	45.0	IND	378.0	25.0	762440
8157	PUDUCHERRY	PUDUCHERRY	MURUGAIYAN. K.S	M	GEN	42.0	IND	378.0	26.0	762440
8158	PUDUCHERRY	PUDUCHERRY	I.M. SEKAR	M	GEN	46.0	IND	375.0	27.0	762440
8159	PUDUCHERRY	PUDUCHERRY	MOHAMED YOUSUF. S.A	M	GEN	46.0	IND	298.0	28.0	762440

8160 rows × 10 columns

LS14DF

Out[54]:

	State name	PC name	Candidate Name	Candidate Sex	Candidate Category	Candidate Age	Party Abbreviation	Total Votes Polled	Position	Total_Electors
0	ANDHRA PRADESH	ADILABAD	GODAM NAGESH	M	ST	49.0	TRS	430847	1	1386282
1	ANDHRA PRADESH	ADILABAD	NARESH	M	ST	37.0	INC	259557	2	1386282
2	ANDHRA PRADESH	ADILABAD	RAMESH RATHOD	M	ST	48.0	TDP	184198	3	1386282
3	ANDHRA PRADESH	ADILABAD	RATHOD SADASHIV	M	ST	55.0	BSP	94420	4	1386282
4	ANDHRA PRADESH	ADILABAD	NETHAWATH RAMDAS	M	ST	44.0	IND	41032	5	1386282
...	...	...	...	...	...	...	...	...	...	...
8893	PUDUCHERRY	PUDUCHERRY	PUVALA NAGESWARA RAO	M	GEN	60.0	IND	465	27	901357
8894	PUDUCHERRY	PUDUCHERRY	G. PALANI	M	GEN	59.0	CPI(ML)(L)	438	28	901357
8895	PUDUCHERRY	PUDUCHERRY	MARIE UTHRIANATHAN	M	GEN	32.0	SAP	366	29	901357
8896	PUDUCHERRY	PUDUCHERRY	S. CHITRAKALA	F	GEN	37.0	JD(U)	309	30	901357
8897	PUDUCHERRY	PUDUCHERRY	P. DHANARASU @ MATHIMAHARAJA	M	GEN	50.0	IND	198	31	901357

8898 rows × 10 columns

```
def AGE_CONV(df):
    AGE_GROUP = []
    for i in df['Candidate Age']:
        if i >= 24 and i <=35:
            AGE_GROUP.append('YOUNG AGE')
        elif i >= 36 and i<=60:
            AGE_GROUP.append('MIDDLE AGE')
        elif i >=60:
```

```
            AGE_GROUP.append('OLD AGE')
        else:
            AGE_GROUP.append('NOT KNOWN')
    return AGE_GROUP

LS09DF['AGE GROUP'] = AGE_CONV(LS09DF)
LS14DF['AGE GROUP'] = AGE_CONV(LS14DF)


def WIN(df):
    WINNER = []
    for i in df['Position']:
        if i == 1:
            WINNER.append(1)
        else:
            WINNER.append(0)
    return WINNER
LS09DF['Position'] = WIN(LS09DF)
LS14DF['Position'] = WIN(LS14DF)


LS09DF.to_csv('../Project/LS09DF.csv',index=False)
LS14DF.to_csv('../Project/LS14DF.csv',index=False)


from sklearn.preprocessing import LabelEncoder
labelEncoder_X = LabelEncoder()
LS09DF['State name'] = labelEncoder_X.fit_transform(LS09DF['State name'])
LS09DF['PC name'] = labelEncoder_X.fit_transform(LS09DF['PC name'])
LS09DF['Candidate Name'] = labelEncoder_X.fit_transform(LS09DF['Candidate Nam
e'])
LS09DF['Candidate Sex'] = labelEncoder_X.fit_transform(LS09DF['Candidate Sex'
])
LS09DF['Candidate Category'] = labelEncoder_X.fit_transform(LS09DF['Candidate
 Category'])
LS09DF['Party Abbreviation'] = labelEncoder_X.fit_transform(LS09DF['Party Abb
reviation'])
LS09DF['AGE GROUP'] = labelEncoder_X.fit_transform(LS09DF['AGE GROUP'])


from sklearn.preprocessing import LabelEncoder
labelEncoder_X = LabelEncoder()
LS14DF['State name'] = labelEncoder_X.fit_transform(LS14DF['State name'])
LS14DF['PC name'] = labelEncoder_X.fit_transform(LS14DF['PC name'])
LS14DF['Candidate Name'] = labelEncoder_X.fit_transform(LS14DF['Candidate Nam
e'])
LS14DF['Candidate Sex'] = labelEncoder_X.fit_transform(LS14DF['Candidate Sex'
])
LS14DF['Candidate Category'] = labelEncoder_X.fit_transform(LS14DF['Candidate
 Category'])
LS14DF['Party Abbreviation'] = labelEncoder_X.fit_transform(LS14DF['Party Abb
reviation'])
LS14DF['AGE GROUP'] = labelEncoder_X.fit_transform(LS14DF['AGE GROUP'])
```

```
X09=LS09DF.drop(['Candidate Age','Position'],axis=1)
Y09=LS09DF['Position']

X09.columns
```

```
Out[61]: Index(['State name', 'PC name', 'Candidate Name', 'Candidate Sex',
                'Candidate Category', 'Party Abbreviation', 'Total Votes Polled',
                'Total_Electors', 'AGE GROUP'],
               dtype='object')
```

```
X14=LS14DF.drop(['Candidate Age','Position'],axis=1)
Y14=LS14DF['Position']

X14.columns
```

```
Out[61]: Index(['State name', 'PC name', 'Candidate Name', 'Candidate Sex',
                'Candidate Category', 'Party Abbreviation', 'Total Votes Polled',
                'Total_Electors', 'AGE GROUP'],
               dtype='object')
```

```
X_train,X_test,Y_train,Y_test = train_test_split(X09,Y09,test_size=0.3,random
_state=42)
Scaler_X = StandardScaler()
X_train = Scaler_X.fit_transform(X_train)
X_test = Scaler_X.transform(X_test)


#Logistic Regression
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression()
lr.fit(X_train,Y_train)
Y_pred = lr.predict(X_test)

print(accuracy_score(Y_test,Y_pred))
print(confusion_matrix(Y_test,Y_pred))
```

```
0.9558823529411765
[[2232   48]
 [  60  108]]
```

```
#Decision Tree
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier()
dtree.fit(X_train,Y_train)
predictions = dtree.predict(X_test)
print(accuracy_score(Y_test,predictions ))
print(confusion_matrix(Y_test,predictions ))
```

```
0.9530228758169934
[[2218   62]
 [  53  115]]
```

```
#Random Forest
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=200)
rfc.fit(X_train,Y_train)
rfc_pred = rfc.predict(X_test)
print(accuracy_score(Y_test,rfc_pred))
print(confusion_matrix(Y_test,rfc_pred))
```

```
0.9599673202614379
[[2232   48]
 [  50  118]]
```

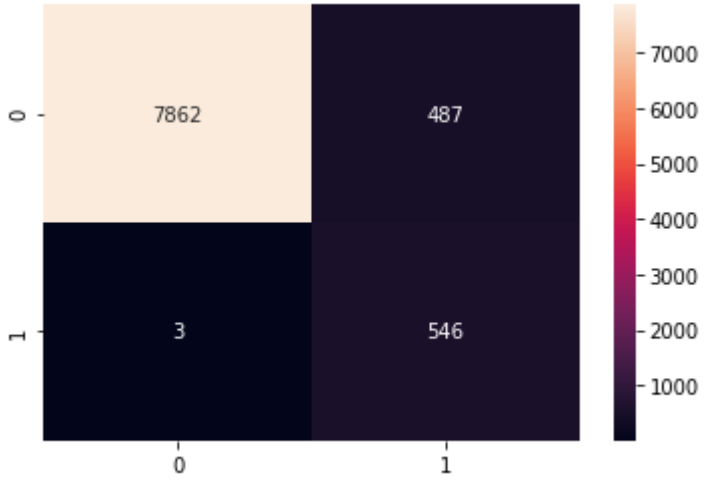
```
# Logistic Regression
PRED_Y14 = lr.predict(X14)
print(accuracy_score(Y14, PRED_Y14))
cm = confusion_matrix(Y14, PRED_Y14)
print(cm)
print(classification_report(Y14, PRED_Y14))
```

```
0.9449314452685997
[[7862  487]
 [   3  546]]
              precision    recall  f1-score   support

           0       1.00      0.94      0.97      8349
           1       0.53      0.99      0.69       549

    accuracy                           0.94      8898
   macro avg       0.76      0.97      0.83      8898
weighted avg       0.97      0.94      0.95      8898
```

```
print("Precision score: {}".format(precision_score(Y14, PRED_Y14)))
```

```
Precision score: 0.5285575992255567
```

```
sb.heatmap(cm, annot=True, fmt="d")
plt.show()
```




```
print("Recall score: {}".format(recall_score(Y14, PRED_Y14)))
```

```
Recall score: 0.994535519125683
```

```
print("F1 Score: {}".format(f1_score(Y14, PRED_Y14)))
```

```
   F1 Score: 0.6902654867256638
```

```
print('Average precision-
recall score: {0:0.2f}'.format(average_precision_score(Y14, PRED_Y14)))
```

```
 Average precision-recall score: 0.53
```

```
LS14 = pd.read_csv('../Project/LS14DF.csv')
LS14['Predicted Winner'] = PRED_Y14.tolist()
LS14.head()
```

Out[69]:

	State name	PC name	Candidate Name	Candidate Sex	Candidate Category	Candidate Age	Party Abbreviation	Total Votes Polled	Position	Total_Electors	AGE GROUP	Predicted Winner
0	ANDHRA PRADESH	ADILABAD	GODAM NAGESH	M	ST	49.0	TRS	430847	1	1386282	MIDDLE AGE	1
1	ANDHRA PRADESH	ADILABAD	NARESH	M	ST	37.0	INC	259557	0	1386282	MIDDLE AGE	1
2	ANDHRA PRADESH	ADILABAD	RAMESH RATHOD	M	ST	48.0	TDP	184198	0	1386282	MIDDLE AGE	0
3	ANDHRA PRADESH	ADILABAD	RATHOD SADASHIV	M	ST	55.0	BSP	94420	0	1386282	MIDDLE AGE	0
4	ANDHRA PRADESH	ADILABAD	NETHAWATH RAMDAS	M	ST	44.0	IND	41032	0	1386282	MIDDLE AGE	0

```
Seatswin14actual = LS14[LS14['Position'] == 1]
df14actual = Seatswin14actual['Party Abbreviation'].value_counts().head().to_
dict()
totalvote14actual = sum(Seatswin14actual['Party Abbreviation'].value_counts()
.tolist())
df14actual['Others'] = totalvote14actual - sum(Seatswin14actual['Party Abbrev
iation'].value_counts().head().tolist())

Seatswin14pred = LS14[LS14['Predicted Winner'] == 1]
df14pred = Seatswin14pred['Party Abbreviation'].value_counts().head().to_dict
()
```

```
totalvote14pred = sum(Seatswin14pred['Party Abbreviation'].value_counts().tol
ist())
df14pred['Others'] = totalvote14pred - sum(Seatswin14pred['Party Abbreviation
'].value_counts().head().tolist())
```

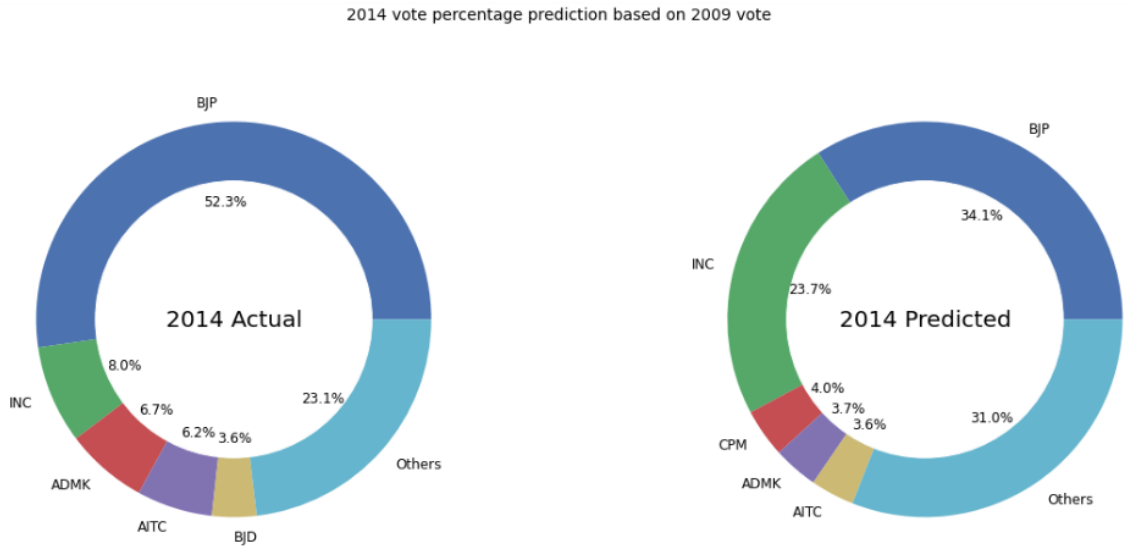
```
df14actual
```

```
Out[71]: {'BJP': 287, 'INC': 44, 'ADMK': 37, 'AITC': 34, 'BJD': 20, 'Others': 127}
```

```
df14pred
```

```
Out[72]: {'BJP': 352, 'INC': 245, 'CPM': 41, 'ADMK': 38, 'AITC': 37, 'Others': 320}
```

```
# 2014 Actual Winning Percentages
#colors = ("#FF5106", "#264CE4", "#E426A4", "#44A122", "#F2EC3A", "#C96F58")
plt.figure(figsize=(20,8))
plt.subplot(1,2,1)
plt.pie(df14actual.values(), labels=df14actual.keys(), autopct='%1.1f%%', tex
tprops={'fontsize': 12})
my_circle1=plt.Circle( (0,0), 0.7, color='white')
fig = plt.gcf()
fig.suptitle("2014 vote percentage prediction based on 2009 vote", fontsize=1
4)
ax = fig.gca()
ax.add_patch(my_circle1)

label = ax.annotate("2014 Actual", xy=(0, 0), fontsize=20, ha="center",va="ce
nter")

ax.axis('off')
ax.set_aspect('equal')
ax.autoscale_view()

# 2014 Predicted Winning Percentages
#colors = ("red", "#264CE4", "#E426A4", "#44A122", "#F2EC3A", "#C96F58")
plt.subplot(1,2,2)
plt.pie(df14pred.values(), labels=df14pred.keys(), autopct='%1.1f%%', textpro
ps={'fontsize': 12})
my_circle2=plt.Circle( (0,0), 0.7, color='white')
fig = plt.gcf()
ax = fig.gca()
ax.add_patch(my_circle2)

label = ax.annotate("2014 Predicted", xy=(0, 0), fontsize=20, ha="center",va=
"center")

ax.axis('off')
ax.set_aspect('equal')
ax.autoscale_view()
```




```
# 2019 Candidate Data Pre-processing
LS19Cand = pd.read_csv('../Project/LS_2.0.csv')
LS19Cand = LS19Cand.drop(['SYMBOL','CRIMINAL\nCASES','EDUCATION','ASSETS','LI
ABILITIES','GENERAL\nVOTES','POSTAL\nVOTES','OVER TOTAL ELECTORS \nIN CONSTIT
UENCY','OVER TOTAL VOTES POLLED \nIN CONSTITUENCY'],axis=1)
```

```
LS19Cand.head ()
```

Out[75]:

	STATE	CONSTITUENCY	NAME	WINNER	PARTY	GENDER	AGE	CATEGORY	TOTAL\nVOTES	TOTAL ELECTORS
0	Telangana	ADILABAD	SOYAM BAPU RAO	1	BJP	MALE	52.0	ST	377374	1489790
1	Telangana	ADILABAD	Godam Nagesh	0	TRS	MALE	54.0	ST	318814	1489790
2	Telangana	ADILABAD	RATHOD RAMESH	0	INC	MALE	52.0	ST	314238	1489790
3	Telangana	ADILABAD	NOTA	0	NOTA	NaN	NaN	NaN	13036	1489790
4	Uttar Pradesh	AGRA	Satyapal Singh Baghel	1	BJP	MALE	58.0	SC	646875	1937690

```
LS19Cand.isnull().sum().sort_values(ascending=False)
```

Out[76]:

```
CATEGORY          245
AGE               245
GENDER            245
TOTAL ELECTORS      0
TOTAL\nVOTES        0
PARTY               0
WINNER              0
NAME                0
CONSTITUENCY        0
STATE               0
dtype: int64
```

```
LS19Cand.dtypes
```

```
Out[77]: STATE              object
         CONSTITUENCY       object
         NAME               object
         WINNER              int64
         PARTY              object
         GENDER             object
         AGE               float64
         CATEGORY           object
         TOTAL\nVOTES        int64
         TOTAL ELECTORS      int64
         dtype: object
```

```
LS19Cand = LS19Cand.fillna({
    'CATEGORY':'NA',
    'AGE':0.0,
    'GENDER':'NA',
})
```

```
LS19Cand.isnull().sum().sort_values(ascending=False)
```

```
Out[79]: TOTAL ELECTORS    0
         TOTAL\nVOTES      0
         CATEGORY          0
         AGE               0
         GENDER            0
         PARTY             0
         WINNER            0
         NAME              0
         CONSTITUENCY      0
         STATE             0
         dtype: int64
```

```
AGE = []
for i in LS19Cand['AGE']:
    if i >= 24 and i <=35:
        AGE.append('YOUNG AGE')
    elif i >= 36 and i<=60:
        AGE.append('MIDDLE AGE')
    elif i >=60:
        AGE.append('OLD AGE')
    else:
        AGE.append('NOT KNOWN')
LS19Cand['AGE GROUP'] = AGE
```

```
LS19Cand=LS19Cand.drop(['AGE'],axis=1)
LS19Cand.head()
```

Out[81]:

	STATE	CONSTITUENCY	NAME	WINNER	PARTY	GENDER	CATEGORY	TOTAL\nVOTES	TOTAL ELECTORS	AGE GROUP
0	Telangana	ADILABAD	SOYAM BAPU RAO	1	BJP	MALE	ST	377374	1489790	MIDDLE AGE
1	Telangana	ADILABAD	Godam Nagesh	0	TRS	MALE	ST	318814	1489790	MIDDLE AGE
2	Telangana	ADILABAD	RATHOD RAMESH	0	INC	MALE	ST	314238	1489790	MIDDLE AGE
3	Telangana	ADILABAD	NOTA	0	NOTA	NA	NA	13036	1489790	NOT KNOWN
4	Uttar Pradesh	AGRA	Satyapal Singh Baghel	1	BJP	MALE	SC	646875	1937690	MIDDLE AGE

```
for i in range(len(LS19Cand)):
    LS19Cand.iloc[i,0] = LS19Cand.iloc[i,0].upper()    #STATE
    LS19Cand.iloc[i,1] = LS19Cand.iloc[i,1].upper()    #CONSTITUENCY
    LS19Cand.iloc[i,2] = LS19Cand.iloc[i,2].upper()    #NAME
    LS19Cand.iloc[i,4] = LS19Cand.iloc[i,4].upper()    #PARTY
    LS19Cand.iloc[i,5] = LS19Cand.iloc[i,5].upper()    #GENDER
```

```
LS19Cand.head()
```

Out[83]:

	STATE	CONSTITUENCY	NAME	WINNER	PARTY	GENDER	CATEGORY	TOTAL\nVOTES	TOTAL ELECTORS	AGE GROUP
0	TELANGANA	ADILABAD	SOYAM BAPU RAO	1	BJP	MALE	ST	377374	1489790	MIDDLE AGE
1	TELANGANA	ADILABAD	GODAM NAGESH	0	TRS	MALE	ST	318814	1489790	MIDDLE AGE
2	TELANGANA	ADILABAD	RATHOD RAMESH	0	INC	MALE	ST	314238	1489790	MIDDLE AGE
3	TELANGANA	ADILABAD	NOTA	0	NOTA	NA	NA	13036	1489790	NOT KNOWN
4	UTTAR PRADESH	AGRA	SATYAPAL SINGH BAGHEL	1	BJP	MALE	SC	646875	1937690	MIDDLE AGE

```
for i in range(len(LS19Cand)):
    if LS19Cand.iloc[i,5] == 'MALE':
        LS19Cand.iloc[i,5]='M'
    elif LS19Cand.iloc[i,5] == 'FEMALE':
        LS19Cand.iloc[i,5]='F'
```

```
LS19Cand.head()
```

Out[85]:

	STATE	CONSTITUENCY	NAME	WINNER	PARTY	GENDER	CATEGORY	TOTAL\nVOTES	TOTAL ELECTORS	AGE GROUP
0	TELANGANA	ADILABAD	SOYAM BAPU RAO	1	BJP	M	ST	377374	1489790	MIDDLE AGE
1	TELANGANA	ADILABAD	GODAM NAGESH	0	TRS	M	ST	318814	1489790	MIDDLE AGE
2	TELANGANA	ADILABAD	RATHOD RAMESH	0	INC	M	ST	314238	1489790	MIDDLE AGE
3	TELANGANA	ADILABAD	NOTA	0	NOTA	NA	NA	13036	1489790	NOT KNOWN
4	UTTAR PRADESH	AGRA	SATYAPAL SINGH BAGHEL	1	BJP	M	SC	646875	1937690	MIDDLE AGE

```
LS19Cand = LS19Cand.rename(columns={'STATE':'State name','CONSTITUENCY':'PC name','NAME':'Candidate Name','WINNER':'Position','PARTY':'Party Abbreviation','GENDER':'Candidate Sex','CATEGORY':'Candidate Category','TOTAL\nVOTES':'Total Votes Polled','TOTAL ELECTORS':'Total_Electors'})
```

```
LS19Cand.columns
```

Out[44]:
```
Index(['State name', 'PC name', 'Candidate Name', 'Candidate Sex',
       'Candidate Category', 'Candidate Age', 'Party Abbreviation',
       'Total Votes Polled', 'Position', 'Total_Electors'],
      dtype='object')
```

```
LS19Cand.to_csv('../Project/LS19DF.csv',index=False)
from sklearn.preprocessing import LabelEncoder
labelEncoder_X = LabelEncoder()
LS19DF = LS19Cand
LS19DF['State name'] = labelEncoder_X.fit_transform(LS19DF['State name'])
LS19DF['PC name'] = labelEncoder_X.fit_transform(LS19DF['PC name'])
LS19DF['Candidate Name'] = labelEncoder_X.fit_transform(LS19DF['Candidate Nam
e'])
LS19DF['Candidate Sex'] = labelEncoder_X.fit_transform(LS19DF['Candidate Sex'
])
LS19DF['Candidate Category'] = labelEncoder_X.fit_transform(LS19DF['Candidate
 Category'])
LS19DF['Party Abbreviation'] = labelEncoder_X.fit_transform(LS19DF['Party Abb
reviation'])
LS19DF['AGE GROUP'] = labelEncoder_X.fit_transform(LS19DF['AGE GROUP'])
```

```
LS19DF.head()
```

Out[90]:

	State name	PC name	Candidate Name	Position	Party Abbreviation	Candidate Sex	Candidate Category	Total Votes Polled	Total_Electors	AGE GROUP
0	31	0	1776	1	26	1	3	377374	1489790	0
1	31	0	685	0	121	1	3	318814	1489790	0
2	31	0	1553	0	46	1	3	314238	1489790	0
3	31	0	1224	0	81	2	1	13036	1489790	1
4	33	1	1674	1	26	1	2	646875	1937690	0

```
LS14DF.head()
```

Out[91]:

	State name	PC name	Candidate Name	Candidate Sex	Candidate Category	Candidate Age	Party Abbreviation	Total Votes Polled	Position	Total_Electors	AGE GROUP
0	1	0	2319	1	4	49.0	448	430847	1	1386282	0
1	1	0	4548	1	4	37.0	178	259557	0	1386282	0
2	1	0	5788	1	4	48.0	442	184198	0	1386282	0
3	1	0	5933	1	4	55.0	127	94420	0	1386282	0
4	1	0	4632	1	4	44.0	179	41032	0	1386282	0

```
X14.head()
```

Out[92]:

	State name	PC name	Candidate Name	Candidate Sex	Candidate Category	Party Abbreviation	Total Votes Polled	Total_Electors	AGE GROUP
0	1	0	2319	1	4	448	430847	1386282	0
1	1	0	4548	1	4	178	259557	1386282	0
2	1	0	5788	1	4	442	184198	1386282	0
3	1	0	5933	1	4	127	94420	1386282	0
4	1	0	4632	1	4	179	41032	1386282	0

```
Y14.head()
```

```
Out[93]: 0    1
         1    0
         2    0
         3    0
         4    0
         Name: Position, dtype: int64
```

```
X19=LS19DF.drop(['Position'],axis=1)
Y19=LS19DF['Position']

X19.head()
```

Out[95]:

	State name	PC name	Candidate Name	Party Abbreviation	Candidate Sex	Candidate Category	Total Votes Polled	Total_Electors	AGE GROUP
0	31	0	1776	26	1	3	377374	1489790	0
1	31	0	685	121	1	3	318814	1489790	0
2	31	0	1553	46	1	3	314238	1489790	0
3	31	0	1224	81	2	1	13036	1489790	1
4	33	1	1674	26	1	2	646875	1937690	0

```
Y19.head()
```

```
Out[96]: 0    1
         1    0
         2    0
         3    0
         4    1
         Name: Position, dtype: int64
```

```
X_train,X_test,Y_train,Y_test =
train_test_split(X14,Y14,test_size=0.3,random_state=42)
Scaler_X = StandardScaler()
X_train = Scaler_X.fit_transform(X_train)
X_test = Scaler_X.transform(X_test)

#Logistic Regression
lr = LogisticRegression()
lr.fit(X_train,Y_train)
Y_pred = lr.predict(X_test)

print(accuracy_score(Y_test,Y_pred))
print(confusion_matrix(Y_test,Y_pred))
```

```
0.9767790262172285
[[2486   34]
 [  28  122]]
```

```
#Decision Tree
dtree = DecisionTreeClassifier()
dtree.fit(X_train,Y_train)
predictions = dtree.predict(X_test)
print(accuracy_score(Y_test,predictions ))
```

```
print(confusion_matrix(Y_test,predictions ))
```

```
0.9801498127340824
[[2495   25]
 [  28  122]]
```

```
#Random Forest
rfc = RandomForestClassifier(n_estimators=200)
rfc.fit(X_train,Y_train)
rfc__pred = rfc.predict(X_test)
print(accuracy_score(Y_test,rfc__pred))
print(confusion_matrix(Y_test,rfc__pred))
print(rfc__pred)
```

```
0.9887640449438202
[[2508   12]
 [  18  132]]
[0 0 0 ... 0 0 0]
```

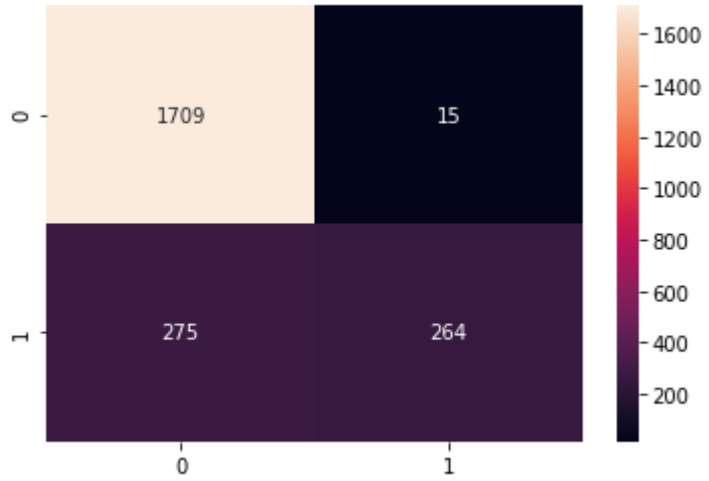
```
#Logistic Regression

Y_pred = lr.predict(X19)
print(accuracy_score(Y19, Y_pred))
cm = confusion_matrix(Y19, Y_pred)
print(cm)
print(classification_report(Y19, Y_pred))
```

```
0.8718515245249668
[[1709   15]
 [ 275  264]]
              precision    recall  f1-score   support

           0       0.86      0.99      0.92      1724
           1       0.95      0.49      0.65       539

    accuracy                           0.87      2263
   macro avg       0.90      0.74      0.78      2263
weighted avg       0.88      0.87      0.86      2263
```

```
print("Precision score: {}".format(precision_score(Y19, Y_pred)))
```

```
Precision score: 0.946236559139785
```

```
sb.heatmap(cm,annot = True, fmt = "d")
plt.show()
```




```
print("Recall score: {}".format(recall_score(Y19, Y_pred)))
```

```
Recall score: 0.4897959183673469
```

```
print("F1 Score: {}".format(f1_score(Y19, Y_pred)))
```

```
F1 Score: 0.6454767726161369
```

```
print('Average precision-
recall score: {0:0.2f}'.format(average_precision_score(Y19, Y_pred)))
```

```
Average precision-recall score: 0.58
```

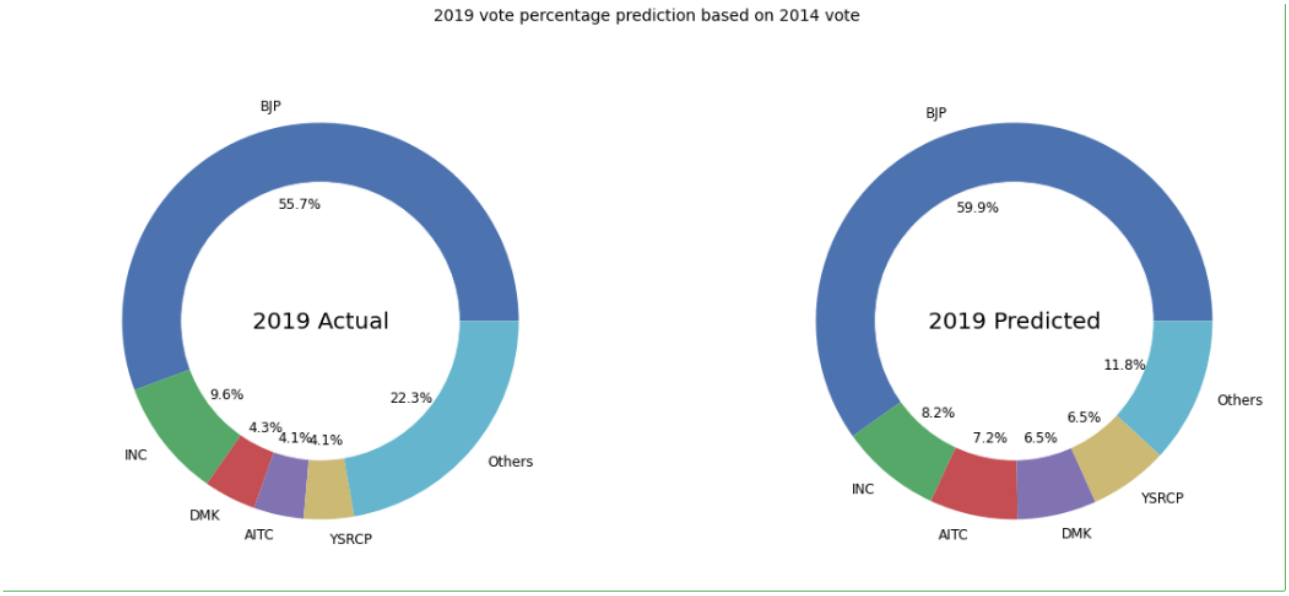
```
LS19Cand = pd.read_csv('../Project/LS19DF.csv')
LS19Cand['Predicted Winner'] = Y_pred.tolist()
LS19Cand.head()
```

Out[102]:

	State name	PC name	Candidate Name	Position	Party Abbreviation	Candidate Sex	Candidate Category	Total Votes Polled	Total_Electors	AGE GROUP	Predicted Winner
0	TELANGANA	ADILABAD	SOYAM BAPU RAO	1	BJP	M	ST	377374	1489790	MIDDLE AGE	0
1	TELANGANA	ADILABAD	GODAM NAGESH	0	TRS	M	ST	318814	1489790	MIDDLE AGE	0
2	TELANGANA	ADILABAD	RATHOD RAMESH	0	INC	M	ST	314238	1489790	MIDDLE AGE	0
3	TELANGANA	ADILABAD	NOTA	0	NOTA	NaN	NaN	13036	1489790	NOT KNOWN	0
4	UTTAR PRADESH	AGRA	SATYAPAL SINGH BAGHEL	1	BJP	M	SC	646875	1937690	MIDDLE AGE	0

```
Seatswin19actual = LS19Cand[LS19Cand['Position'] == 1]
df19actual = Seatswin19actual['Party
Abbreviation'].value_counts().head().to_dict()
totalvote19actual = sum(Seatswin19actual['Party
Abbreviation'].value_counts().tolist())
df19actual['Others'] = totalvote19actual - sum(Seatswin19actual['Party
Abbreviation'].value_counts().head().tolist())

Seatswin19pred = LS19Cand[LS19Cand['Predicted Winner'] == 1]
```

```
df19pred = Seatswin19pred['Party
Abbreviation'].value_counts().head().to_dict()
totalvote19pred = sum(Seatswin19pred['Party
Abbreviation'].value_counts().tolist())
df19pred['Others'] = totalvote19pred - sum(Seatswin19pred['Party
Abbreviation'].value_counts().head().tolist())

# 2019 Actual Winning Percentages

plt.figure(figsize=(20,8))
plt.subplot(1,2,1)
plt.pie(df19actual.values(), labels=df19actual.keys(), autopct='%1.1f%%',
textprops={'fontsize': 12})
my_circle1=plt.Circle( (0,0), 0.7, color='white')
fig = plt.gcf()
fig.suptitle("2019 vote percentage prediction based on 2014 vote",
fontsize=14)
ax = fig.gca()
ax.add_patch(my_circle1)

label = ax.annotate("2019 Actual", xy=(0, 0), fontsize=20,
ha="center",va="center")

ax.axis('off')
ax.set_aspect('equal')
ax.autoscale_view()

# 2019 Predicted Winning Percentages

plt.subplot(1,2,2)
plt.pie(df19pred.values(), labels=df19pred.keys(), autopct='%1.1f%%',
textprops={'fontsize': 12})
my_circle2=plt.Circle( (0,0), 0.7, color='white')
fig = plt.gcf()
ax = fig.gca()
ax.add_patch(my_circle2)

label = ax.annotate("2019 Predicted", xy=(0, 0), fontsize=20,
ha="center",va="center")

ax.axis('off')
ax.set_aspect('equal')
ax.autoscale_view()
```




```
#Detail analysis of 2019 general election(Lok Sabha Vote) in India using 3 Al
gorithms
#Prediction on 2019 Lok Sabha Vote in India(Prediction Models)  using 3 Algor
ithms
```

```
LS2019Cnd = pd.read_csv("../Project/LS_2.0.csv")
pd.set_option('display.max_rows',20000, 'display.max_columns',100)
LS2019Cnd.shape
```

```
Out[36]: (2263, 19)
```

```
LS2019Cnd. head()
```

Out[37]:

	STATE	CONSTITUENCY	NAME	WINNER	PARTY	SYMBOL	GENDER	CRIMINAL\nCASES	AGE	CATEGORY	EDUCATION	ASSETS	LIABILITIES
0	Telangana	ADILABAD	SOYAM BAPU RAO	1	BJP	Lotus	MALE	52	52.0	ST	12th Pass	Rs 30,99,414\n ~ 30 Lacs+	Rs 2,31,450\n ~ 2 Lacs+
1	Telangana	ADILABAD	Godam Nagesh	0	TRS	Car	MALE	0	54.0	ST	Post Graduate	Rs 1,84,77,888\n ~ 1 Crore+	Rs 8,47,000\n ~ 8 Lacs+
2	Telangana	ADILABAD	RATHOD RAMESH	0	INC	Hand	MALE	3	52.0	ST	12th Pass	Rs 3,64,91,000\n ~ 3 Crore+	Rs 1,53,00,000\n ~ 1 Crore+
3	Telangana	ADILABAD	NOTA	0	NOTA	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	Uttar Pradesh	AGRA	Satyapal Singh Baghel	1	BJP	Lotus	MALE	5	58.0	SC	Doctorate	Rs 7,42,74,036\n ~ 7 Crore+	Rs 86,06,522\n ~ 86 Lacs+

```
LS2019Cnd.isnull().sum().sort_values(ascending=False).head(30)
```

```
Out[38]: CATEGORY                                    245
         GENDER                                      245
         LIABILITIES                                 245
         ASSETS                                      245
         EDUCATION                                   245
         SYMBOL                                      245
         AGE                                         245
         CRIMINAL\nCASES                             245
         CONSTITUENCY                                  0
         NAME                                          0
         WINNER                                        0
         PARTY                                         0
         TOTAL ELECTORS                                0
         OVER TOTAL VOTES POLLED \nIN CONSTITUENCY     0
         GENERAL\nVOTES                                0
         POSTAL\nVOTES                                 0
         TOTAL\nVOTES                                  0
         OVER TOTAL ELECTORS \nIN CONSTITUENCY         0
         STATE                                         0
         dtype: int64
```

```
LS2019Cnd.columns
```

```
Out[39]: Index(['STATE', 'CONSTITUENCY', 'NAME', 'WINNER', 'PARTY', 'SYMBOL', 'GENDER',
                'CRIMINAL\nCASES', 'AGE', 'CATEGORY', 'EDUCATION', 'ASSETS',
                'LIABILITIES', 'GENERAL\nVOTES', 'POSTAL\nVOTES', 'TOTAL\nVOTES',
                'OVER TOTAL ELECTORS \nIN CONSTITUENCY',
                'OVER TOTAL VOTES POLLED \nIN CONSTITUENCY', 'TOTAL ELECTORS'],
               dtype='object')
```

```
LS2019Cnd = LS2019Cnd.rename({'CRIMINAL\nCASES':'CRIMINAL CASES','GENERAL\nVOTES':'GENERAL VOTES','POSTAL\nVOTES':'POSTAL VOTES','TOTAL\nVOTES':'TOTAL VOTES','OVER TOTAL ELECTORS \nIN CONSTITUENCY':'OVER TOTAL ELECTORS IN CONSTITUENCY','OVER TOTAL VOTES POLLED \nIN CONSTITUENCY':'OVER TOTAL VOTES POLLED IN CONSTITUENCY'}, axis=1)
```

```
LS2019Cnd.columns
```

```
Out[41]: Index(['STATE', 'CONSTITUENCY', 'NAME', 'WINNER', 'PARTY', 'SYMBOL', 'GENDER',
                'CRIMINAL CASES', 'AGE', 'CATEGORY', 'EDUCATION', 'ASSETS',
                'LIABILITIES', 'GENERAL VOTES', 'POSTAL VOTES', 'TOTAL VOTES',
                'OVER TOTAL ELECTORS IN CONSTITUENCY',
                'OVER TOTAL VOTES POLLED IN CONSTITUENCY', 'TOTAL ELECTORS'],
               dtype='object')
```

```
LS2019Cnd[LS2019Cnd['NAME']=='NOTA'].head()
```

Out[42]:

	STATE	CONSTITUENCY	NAME	WINNER	PARTY	SYMBOL	GENDER	CRIMINAL CASES	AGE	CATEGORY	EDUCATION	ASSETS	LIABILITIES	GENERAL VOTES
3	Telangana	ADILABAD	NOTA	0	NOTA	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	13030
14	Gujarat	AHMEDABAD WEST	NOTA	0	NOTA	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	14580
39	West Bengal	ALIPURDUARS	NOTA	0	NOTA	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	21147
46	Uttarakhand	ALMORA	NOTA	0	NOTA	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	15311
54	Andhra Pradesh	AMALAPURAM	NOTA	0	NOTA	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	16427

```
LS2019Cnd = LS2019Cnd.fillna({
    'SYMBOL': 'NO SYMBOL',
    'GENDER': 'NOT APPLICABLE',
    'CRIMINAL CASES': 0,
    'AGE': 0.0,
    'CATEGORY': 'NOT APPLICABLE',
    'EDUCATION': 'NOT APPLICABLE',
    'ASSETS': 'Rs 0\n ~ 0',
    'LIABILITIES': 'Rs 0\n ~ 0'
})
```

```
LS2019Cnd.isnull().sum()
```

Out[44]:
```
STATE                                0
CONSTITUENCY                         0
NAME                                 0
WINNER                               0
PARTY                                0
SYMBOL                               0
GENDER                               0
CRIMINAL CASES                       0
AGE                                  0
CATEGORY                             0
EDUCATION                            0
ASSETS                               0
LIABILITIES                          0
GENERAL VOTES                        0
POSTAL VOTES                         0
TOTAL VOTES                          0
OVER TOTAL ELECTORS IN CONSTITUENCY  0
OVER TOTAL VOTES POLLED IN CONSTITUENCY  0
TOTAL ELECTORS                       0
dtype: int64
```

```
LS2019Cnd['PARTY'].value_counts()
```

```
sum(LS2019Cnd['PARTY'].value_counts())
```

Out[46]: 2263

```
# Percentage of candidate (Partywise)
```

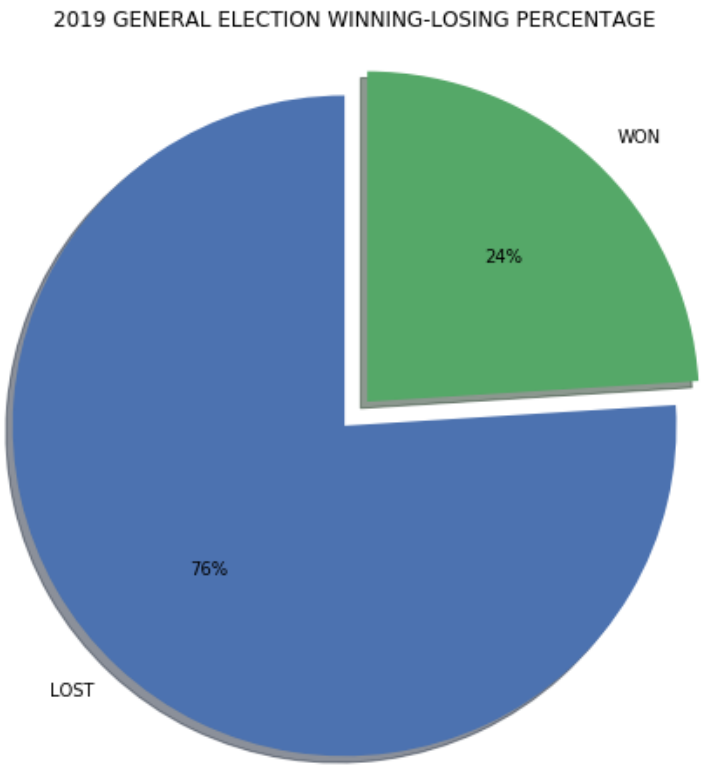
```
(LS2019Cnd['PARTY'].value_counts()/sum(LS2019Cnd['PARTY'].value_counts()))*100
```

```
Out[47]: BJP          18.559434
         INC          18.250110
         NOTA         10.826337
         IND           8.882015
         BSP           7.202828
         CPI(M)        4.418913
         AITC          2.076889
         VBA           2.076889
         SP            1.723376
         NTK           1.679187
         MNM           1.590809
         SHS           1.148917
         YSRCP         1.104728
         AAP           1.104728
         TDP           1.104728
         DMK           1.016350
         RJD           0.927972
         BJD           0.927972
         AIADMK        0.927972
```

```
# Winner and Loser Persentage

List = []
List = ((LS2019Cnd['WINNER'].value_counts()/sum(LS2019Cnd['WINNER'].value_cou
nts()))*100).to_list()
List
```

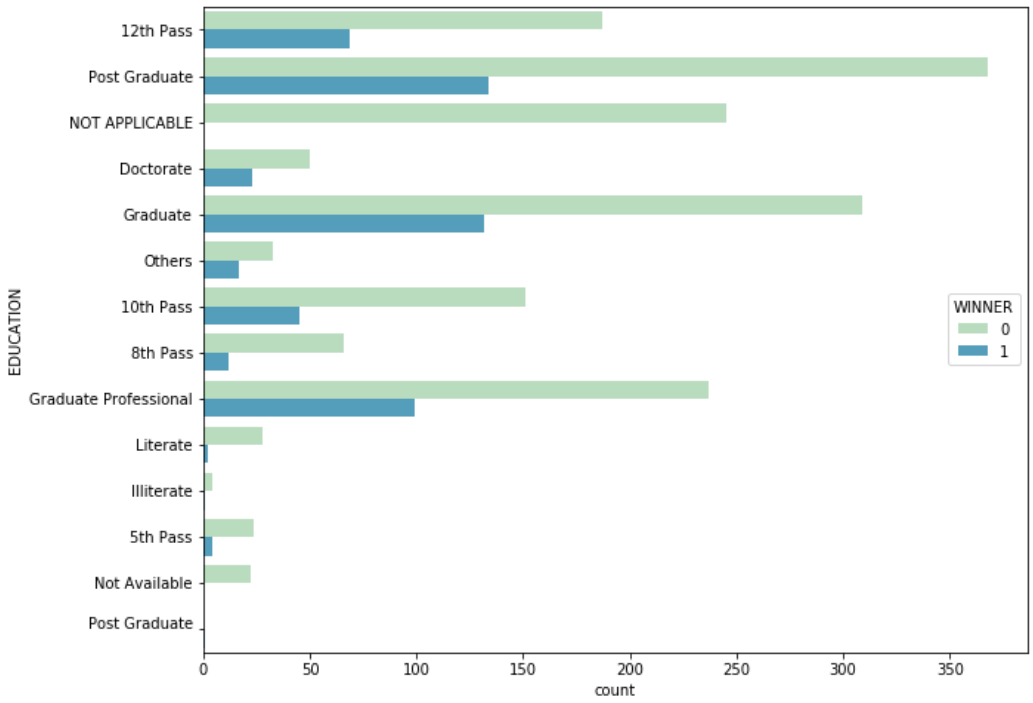
```
Out[48]: [76.18205921343349, 23.817940786566506]
```

```
plt.rcParams['figure.figsize'] = (12,8)
labels = 'LOST', 'WON'
sizes = List
explode = (0, 0.1)

fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%0.0f%%',
        shadow=True, startangle=90,center=(0, 0))
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle
.
plt.title('2019 GENERAL ELECTION WINNING-LOSING PERCENTAGE')
plt.show()
```




```
plt.figure(figsize=(10,8))
sb.countplot(y='EDUCATION',hue='WINNER',data=LS2019Cnd,palette="GnBu")
```

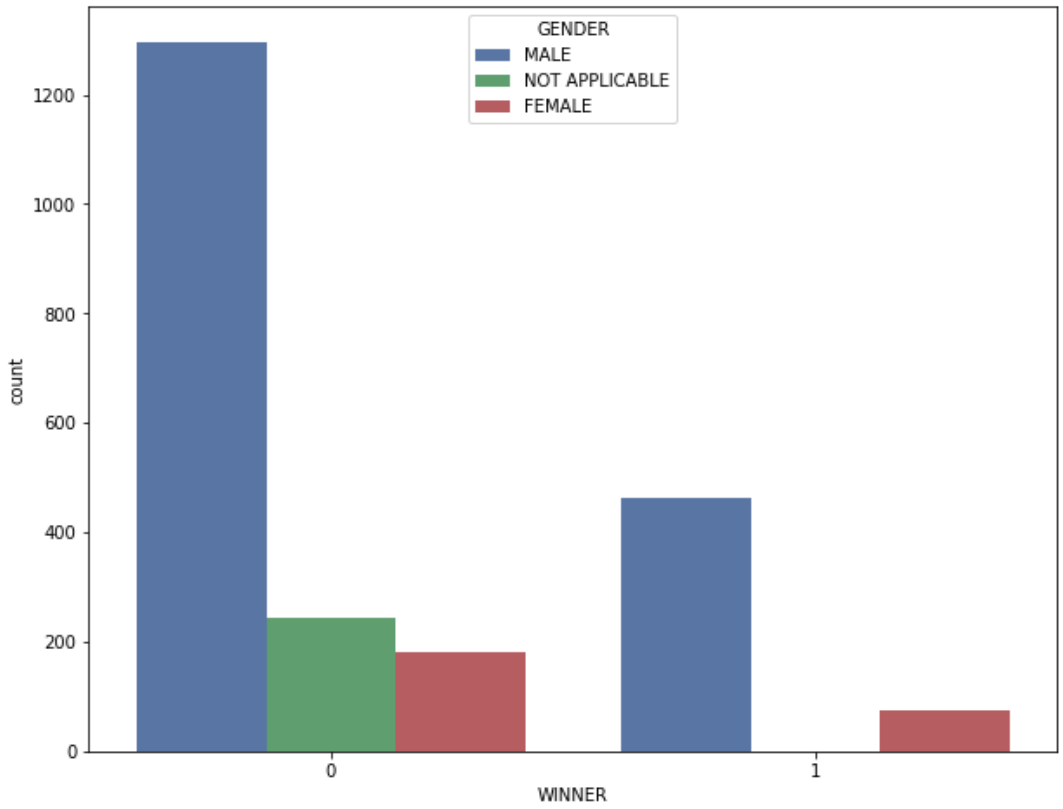
```
Out[50]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb39a9d1550>
```




```
plt.figure(figsize=(10,8))
```

```
sb.countplot(x='WINNER',hue='GENDER',data=LS2019Cnd)
```

```
Out[51]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb39aa9e390>
```




```
#Filling Nan Values in CRIMINAL CASE COLUMN
LS2019Cnd['CRIMINAL CASES']=LS2019Cnd['CRIMINAL CASES'].replace(to_replace ="
Not Available",value ="0")
LS2019Cnd['CRIMINAL CASES'] = LS2019Cnd['CRIMINAL CASES'].astype(float)


# Removing Spacial Charecters from Education column

LS2019Cnd['EDUCATION']=LS2019Cnd['EDUCATION'].replace(to_replace ="Post Gradu
ate\n", value ="Post Graduate")

LS2019Cnd.head()
```

Out[54]:

	STATE	CONSTITUENCY	NAME	WINNER	PARTY	SYMBOL	GENDER	CRIMINAL CASES	AGE	CATEGORY	EDUCATION	ASSETS	LIABILITIES	G
0	Telangana	ADILABAD	SOYAM BAPU RAO	1	BJP	Lotus	MALE	52.0	52.0	ST	12th Pass	Rs 30,99,414\n ~ 30 Lacs+	Rs 2,31,450\n ~ 2 Lacs+	
1	Telangana	ADILABAD	Godam Nagesh	0	TRS	Car	MALE	0.0	54.0	ST	Post Graduate	Rs 1,84,77,888\n ~ 1 Crore+	Rs 8,47,000\n ~ 8 Lacs+	
2	Telangana	ADILABAD	RATHOD RAMESH	0	INC	Hand	MALE	3.0	52.0	ST	12th Pass	Rs 3,64,91,000\n ~ 3 Crore+	Rs 1,53,00,000\n ~ 1 Crore+	
3	Telangana	ADILABAD	NOTA	0	NOTA	NO SYMBOL	NOT APPLICABLE	0.0	0.0	NOT APPLICABLE	NOT APPLICABLE	Rs 0\n ~ 0	Rs 0\n ~ 0	
4	Uttar Pradesh	AGRA	Satyapal Singh Baghel	1	BJP	Lotus	MALE	5.0	58.0	SC	Doctorate	Rs 7,42,74,036\n ~ 7 Crore+	Rs 86,06,522\n ~ 86 Lacs+	

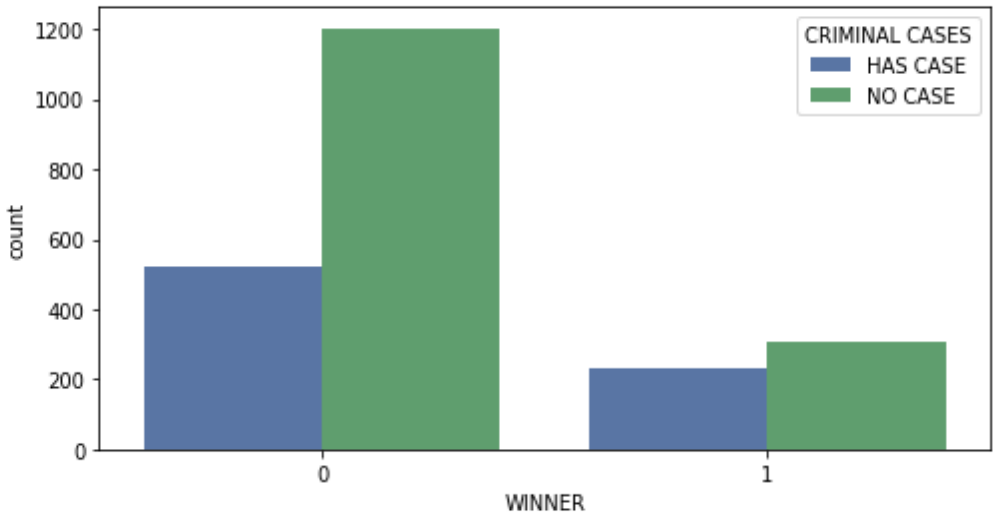
```
#Checking if the candidate has criminal case or not
for i in range(len(LS2019Cnd)):
    if LS2019Cnd.iloc[i,7] > 0:
        LS2019Cnd.iloc[i,7]='HAS CASE'
    else:
        LS2019Cnd.iloc[i,7]='NO CASE'

LS2019Cnd.head()
```

Out[56]:

	STATE	CONSTITUENCY	NAME	WINNER	PARTY	SYMBOL	GENDER	CRIMINAL CASES	AGE	CATEGORY	ED
0	Telangana	ADILABAD	SOYAM BAPU RAO	1	BJP	Lotus	MALE	HAS CASE	52.0	ST	
1	Telangana	ADILABAD	Godam Nagesh	0	TRS	Car	MALE	NO CASE	54.0	ST	
2	Telangana	ADILABAD	RATHOD RAMESH	0	INC	Hand	MALE	HAS CASE	52.0	ST	
3	Telangana	ADILABAD	NOTA	0	NOTA	NO SYMBOL	NOT APPLICABLE	NO CASE	0.0	NOT APPLICABLE	APF
4	Uttar Pradesh	AGRA	Satyapal Singh Baghel	1	BJP	Lotus	MALE	HAS CASE	58.0	SC	

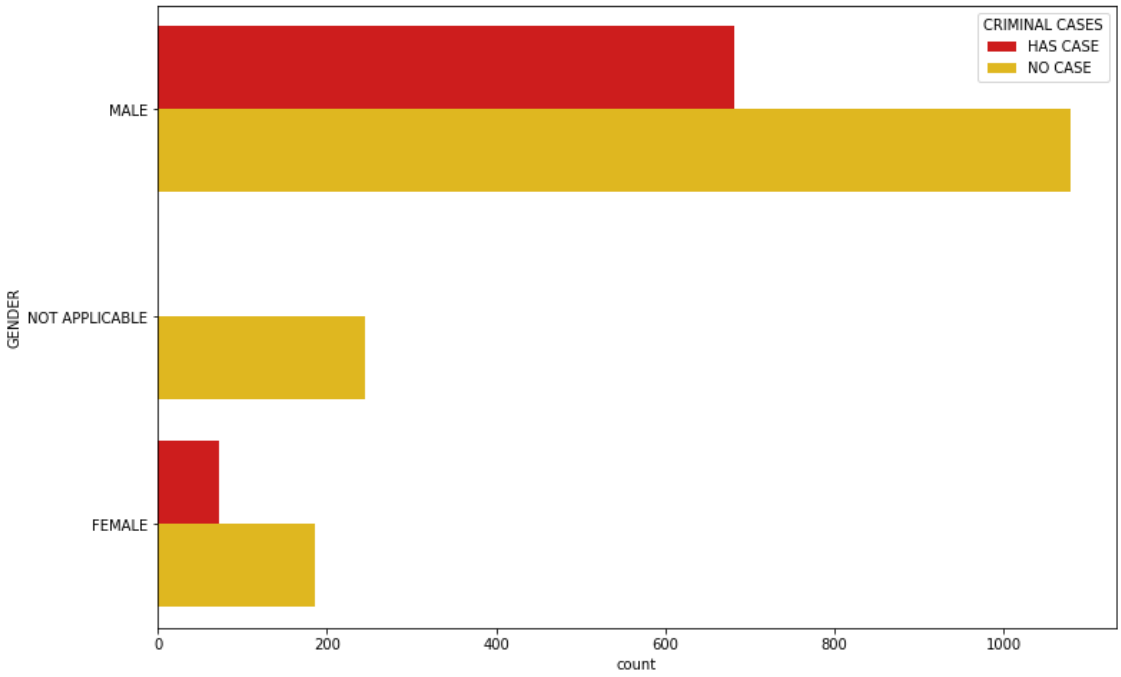
```
plt.figure(figsize=(8,4))
sb.countplot(x='WINNER',hue='CRIMINAL CASES',data=LS2019Cnd)
```

Out[57]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb39a5e9310>




```
sb.countplot(y='GENDER',hue='CRIMINAL CASES',data=LS2019Cnd,palette='hot')
```

```
Out[58]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb39a85b1d0>
```




```
# fill Not Available Assets to -1

for i in range(len(LS2019Cnd)):
    if LS2019Cnd.iloc[i,11]=='Not Available':
        LS2019Cnd.iloc[i,11] = -1
    if LS2019Cnd.iloc[i,12]=='Not Available':
        LS2019Cnd.iloc[i,12] = -1

LS2019Cnd[LS2019Cnd.isnull().any(axis=1)].sum()
```

```
Out[60]: STATE                                    0.0
         CONSTITUENCY                             0.0
         NAME                                     0.0
         WINNER                                   0.0
         PARTY                                    0.0
         SYMBOL                                   0.0
         GENDER                                   0.0
         CRIMINAL CASES                           0.0
         AGE                                      0.0
         CATEGORY                                 0.0
         EDUCATION                                0.0
         ASSETS                                   0.0
         LIABILITIES                              0.0
         GENERAL VOTES                            0.0
         POSTAL VOTES                             0.0
         TOTAL VOTES                              0.0
         OVER TOTAL ELECTORS IN CONSTITUENCY      0.0
         OVER TOTAL VOTES POLLED IN CONSTITUENCY  0.0
         TOTAL ELECTORS                           0.0
         dtype: float64
```

```
LS2019Cnd[['ASSETS','LIABILITIES']].head()
```

Out[61]:

	ASSETS	LIABILITIES
0	Rs 30,99,414\n ~ 30 Lacs+	Rs 2,31,450\n ~ 2 Lacs+
1	Rs 1,84,77,888\n ~ 1 Crore+	Rs 8,47,000\n ~ 8 Lacs+
2	Rs 3,64,91,000\n ~ 3 Crore+	Rs 1,53,00,000\n ~ 1 Crore+
3	Rs 0\n ~ 0	Rs 0\n ~ 0
4	Rs 7,42,74,036\n ~ 7 Crore+	Rs 86,06,522\n ~ 86 Lacs+

```
LS2019Cnd['ASSETS'] = LS2019Cnd['ASSETS'].str.split('\n ~', 1, expand=True)[0
]
LS2019Cnd['LIABILITIES'] = LS2019Cnd['LIABILITIES'].str.split('\n ~', 1, expa
nd=True)[0]

LS2019Cnd['ASSETS'] = LS2019Cnd['ASSETS'].str.split('\n~', 1, expand=True)[0]
LS2019Cnd['LIABILITIES'] = LS2019Cnd['LIABILITIES'].str.split('\n~', 1, expan
d=True)[0]

LS2019Cnd['ASSETS'] = LS2019Cnd['ASSETS'].str.split('\n', 1, expand=True)[0]
LS2019Cnd['LIABILITIES'] = LS2019Cnd['LIABILITIES'].str.split('\n', 1, expand
=True)[0]


LS2019Cnd[['ASSETS','LIABILITIES']].head()
```

Out[63]:

	ASSETS	LIABILITIES
0	Rs 30,99,414	Rs 2,31,450
1	Rs 1,84,77,888	Rs 8,47,000
2	Rs 3,64,91,000	Rs 1,53,00,000
3	Rs 0	Rs 0
4	Rs 7,42,74,036	Rs 86,06,522

```
LS2019Cnd['ASSETS'] = LS2019Cnd['ASSETS'].str.split('Rs ', 1, expand=True)[1]
LS2019Cnd['LIABILITIES'] = LS2019Cnd['LIABILITIES'].str.split('Rs ', 1, expan
d=True)[1]


LS2019Cnd['ASSETS'] = LS2019Cnd['ASSETS'].str.split(',').str.join('')
LS2019Cnd['LIABILITIES'] = LS2019Cnd['LIABILITIES'].str.split(',').str.join('
')

LS2019Cnd[['ASSETS','LIABILITIES']].head()
```

Out[67]:

	ASSETS	LIABILITIES
0	3099414	231450
1	18477888	847000
2	36491000	15300000
3	0	0
4	74274036	8606522

```
LS2019Cnd['ASSETS']=LS2019Cnd['ASSETS'].str.replace(' ', '')
LS2019Cnd['LIABILITIES'] = LS2019Cnd['LIABILITIES'].str.replace(' ', '')

LS2019Cnd[['ASSETS','LIABILITIES']].head()
```

Out[69]:

	ASSETS	LIABILITIES
0	3099414	231450
1	18477888	847000
2	36491000	15300000
3	0	0
4	74274036	8606522

```
LS2019Cnd['ASSETS'] = LS2019Cnd['ASSETS'].astype(float)
LS2019Cnd['LIABILITIES'] = LS2019Cnd['LIABILITIES'].astype(float)

LS2019Cnd[['ASSETS','LIABILITIES']].head()
```

Out[71]:

	ASSETS	LIABILITIES
0	3099414.0	231450.0
1	18477888.0	847000.0
2	36491000.0	15300000.0
3	0.0	0.0
4	74274036.0	8606522.0

```
LS2019Cnd.to_csv('../Project/LS_2.1.csv',index=False)
# ASSETS can be converted into Different Economic Classes

STATUS = []
for i in LS2019Cnd['ASSETS']:
```

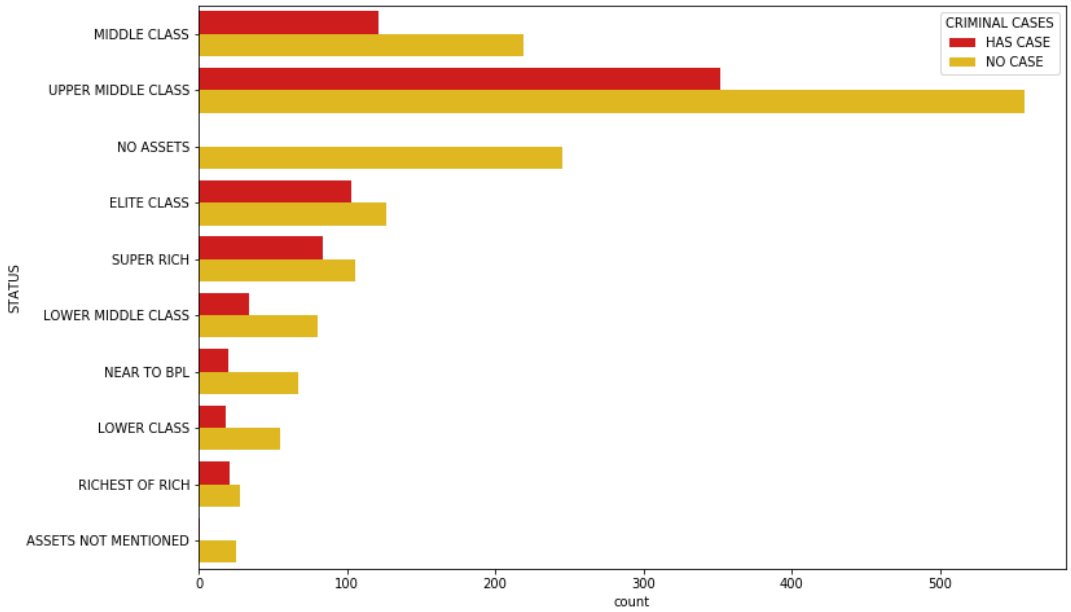
```
    if i >0.0 and i < 500000.0:
        STATUS.append('NEAR TO BPL')
    elif i >= 500000.0 and i <= 1000000.0:
        STATUS.append('LOWER CLASS')
    elif i >= 1000001.0 and i <= 2500000.0:
        STATUS.append('LOWER MIDDLE CLASS')
    elif i >= 2500001.0 and i <= 10000000.0:
        STATUS.append('MIDDLE CLASS')
    elif i >= 10000001.0 and i <= 100000000.0:
        STATUS.append('UPPER MIDDLE CLASS')
    elif i >= 100000000.0 and i <= 250000000.0:
        STATUS.append('ELITE CLASS')
    elif i >= 250000001.0 and i <= 1000000000.0:
        STATUS.append('SUPER RICH')
    elif i >= 1000000001.0:
        STATUS.append('RICHEST OF RICH')
    elif i == 0.0:
        STATUS.append('NO ASSETS')
    else:
        STATUS.append('ASSETS NOT MENTIONED')

LS2019Cnd['STATUS'] = STATUS


LS2019Cnd.head ()
```

Out[74]:

	STATE	CONSTITUENCY	NAME	WINNER	PARTY	SYMBOL	GENDER	CRIMINAL CASES	AGE	CATEGORY	EDUCATION	ASSETS	LIABILITIES	GEN V
0	Telangana	ADILABAD	SOYAM BAPU RAO	1	BJP	Lotus	MALE	HAS CASE	52.0	ST	12th Pass	3099414.0	231450.0	3
1	Telangana	ADILABAD	Godam Nagesh	0	TRS	Car	MALE	NO CASE	54.0	ST	Post Graduate	18477888.0	847000.0	3
2	Telangana	ADILABAD	RATHOD RAMESH	0	INC	Hand	MALE	HAS CASE	52.0	ST	12th Pass	36491000.0	15300000.0	3
3	Telangana	ADILABAD	NOTA	0	NOTA	NO SYMBOL	NOT APPLICABLE	NO CASE	0.0	NOT APPLICABLE	NOT APPLICABLE	0.0	0.0	
4	Uttar Pradesh	AGRA	Satyapal Singh Baghel	1	BJP	Lotus	MALE	HAS CASE	58.0	SC	Doctorate	74274036.0	8606522.0	6

```
sb.countplot(y='STATUS',hue='CRIMINAL CASES',data=LS2019Cnd,palette='hot')
```

```
Out[75]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb39a49e2d0>
```

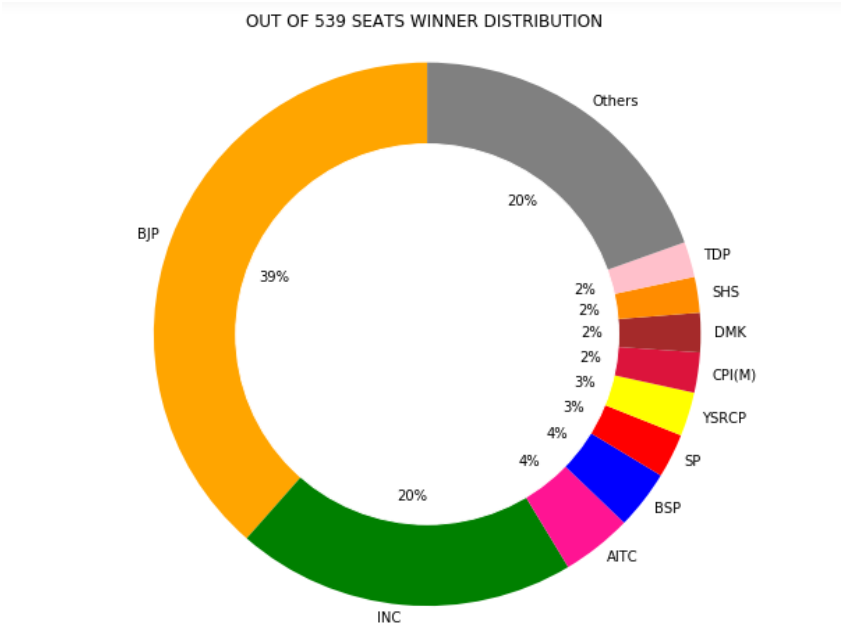



```
#Using Groupby on Party with Total Votes(TO CHECK HIGHEST PERCENTAGE OF VOTES
 FOR TOP 10 PARTIES)
top10 = LS2019Cnd.groupby('PARTY')['TOTAL VOTES'].sum().sort_values(ascending
=False).head(10).to_dict()
total = LS2019Cnd['TOTAL VOTES'].sum()
top10['Others'] = total - sum(LS2019Cnd.groupby('PARTY')['TOTAL VOTES'].sum()
.sort_values(ascending=False).head(10))
top10
```

```
Out[76]: {'BJP': 228938556,
          'INC': 119418722,
          'AITC': 24832104,
          'BSP': 20808194,
          'SP': 15616282,
          'YSRCP': 15537006,
          'CPI(M)': 14180942,
          'DMK': 13877992,
          'SHS': 12618927,
          'TDP': 12515345,
          'Others': 115896633}
```

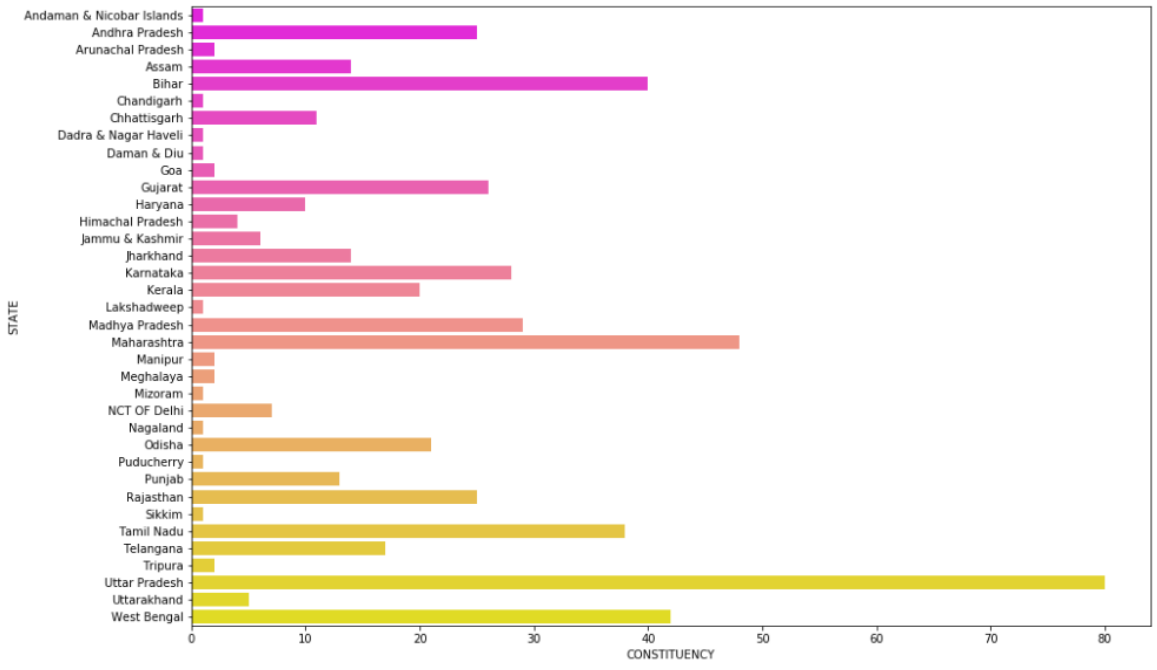
```
plt.rcParams['figure.figsize'] = (10,8)
colors=('orange', 'green', 'deeppink', 'blue', 'red', 'yellow', 'crimson', 'b
rown','darkorange','pink','gray')

my_circle = plt.Circle((0, 0), 0.7, color='white')
```

```
d = plt.pie(top10.values(), labels=top10.keys(), autopct='%0.0f%%', startangl
e=90,colors=colors, labeldistance=1.05)
plt.axis('equal')
plt.gca().add_artist(my_circle)
plt.title('OUT OF 539 SEATS WINNER DISTRIBUTION')
plt.show()
```

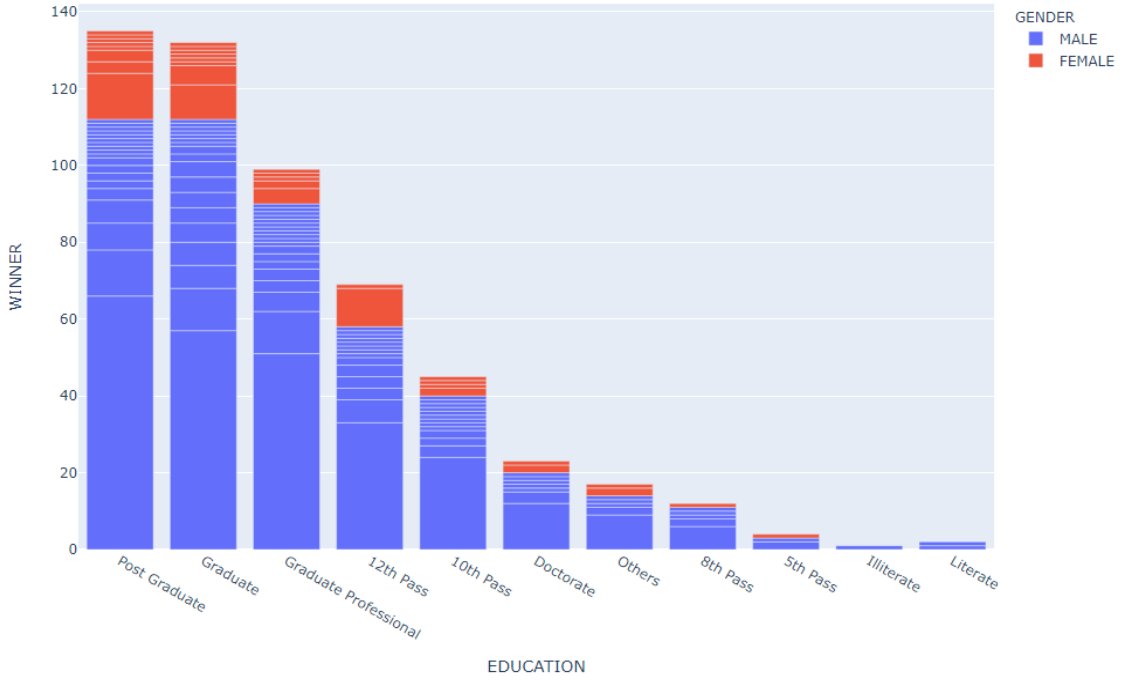



```
p = LS2019Cnd.groupby('STATE')['CONSTITUENCY'].nunique().reset_index()
plt.figure(figsize=(15,10))
sb.barplot(y='STATE',x='CONSTITUENCY',data=p,palette='spring')
```

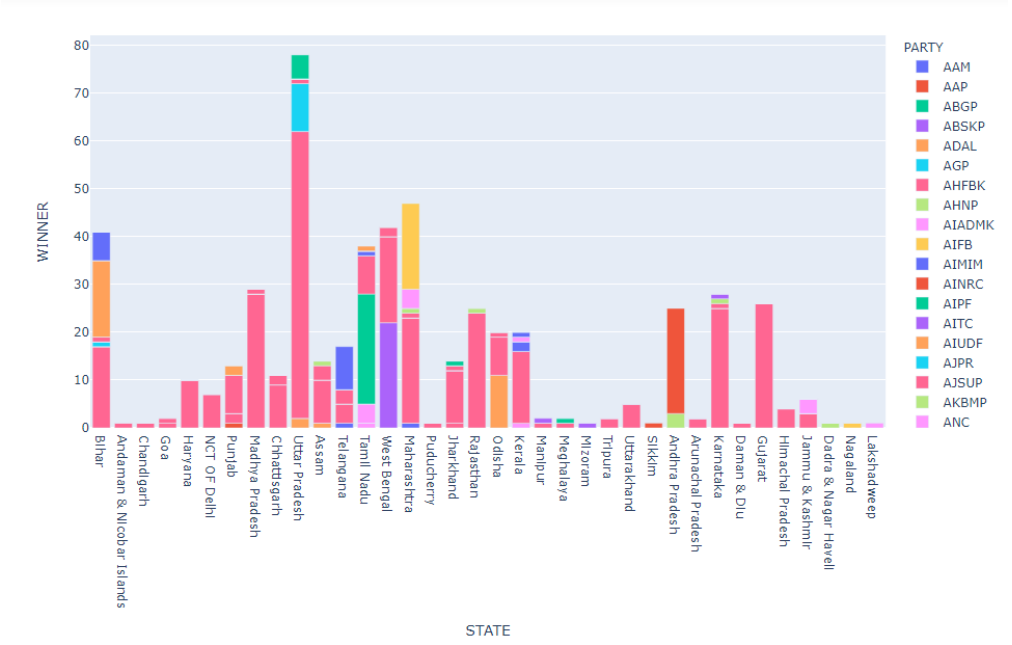
```
Out[78]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb39a76b950>
```



```
p = LS2019Cnd.groupby(['PARTY','EDUCATION','GENDER'])['WINNER'].sum().reset_index().sort_values('WINNER',ascending = False)
p = p[p['WINNER']!=0]
fig = px.bar(p, x='EDUCATION', y='WINNER',hover_data =['PARTY'], color='GENDER', height=650)
fig.show()
```

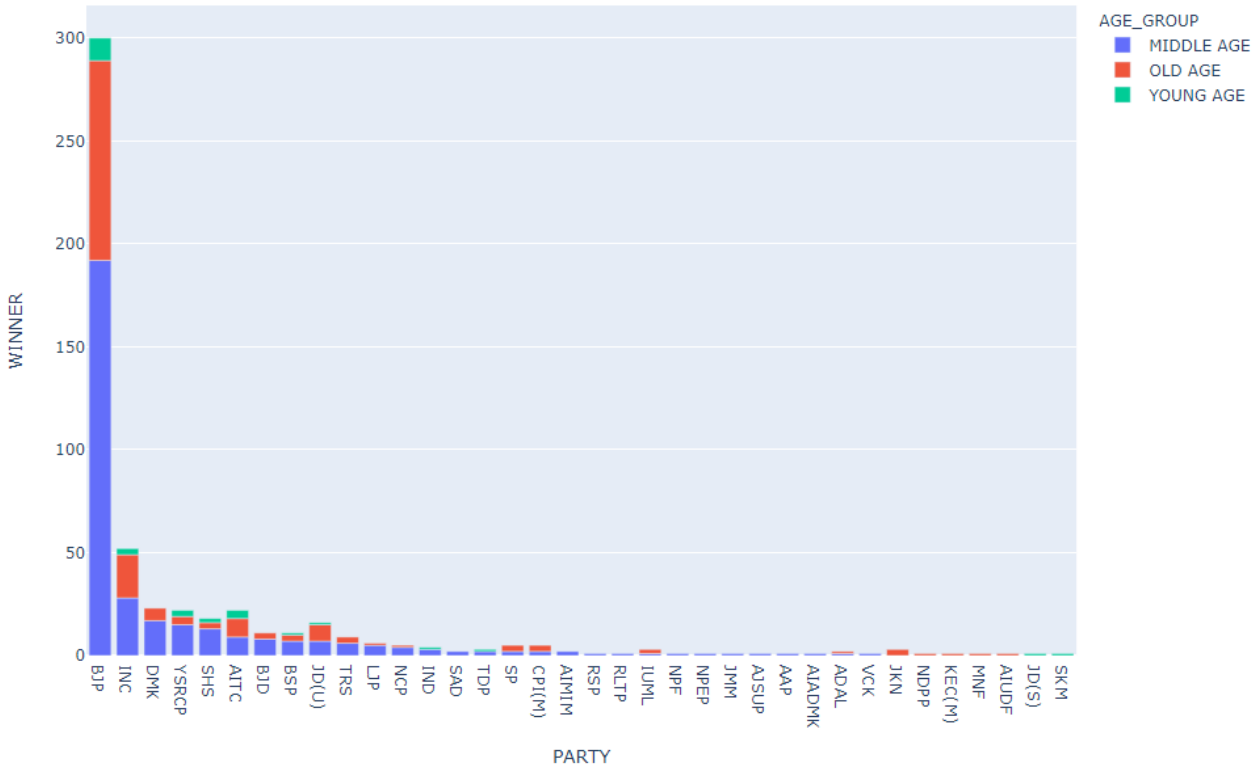



```
p = LS2019Cnd.groupby(['PARTY','STATE']).sum().reset_index().sort_values('PARTY',ascending =True)
fig = px.bar(p, x='STATE', y='WINNER',hover_data =['PARTY'], color='PARTY', height=650)
fig.show()
```



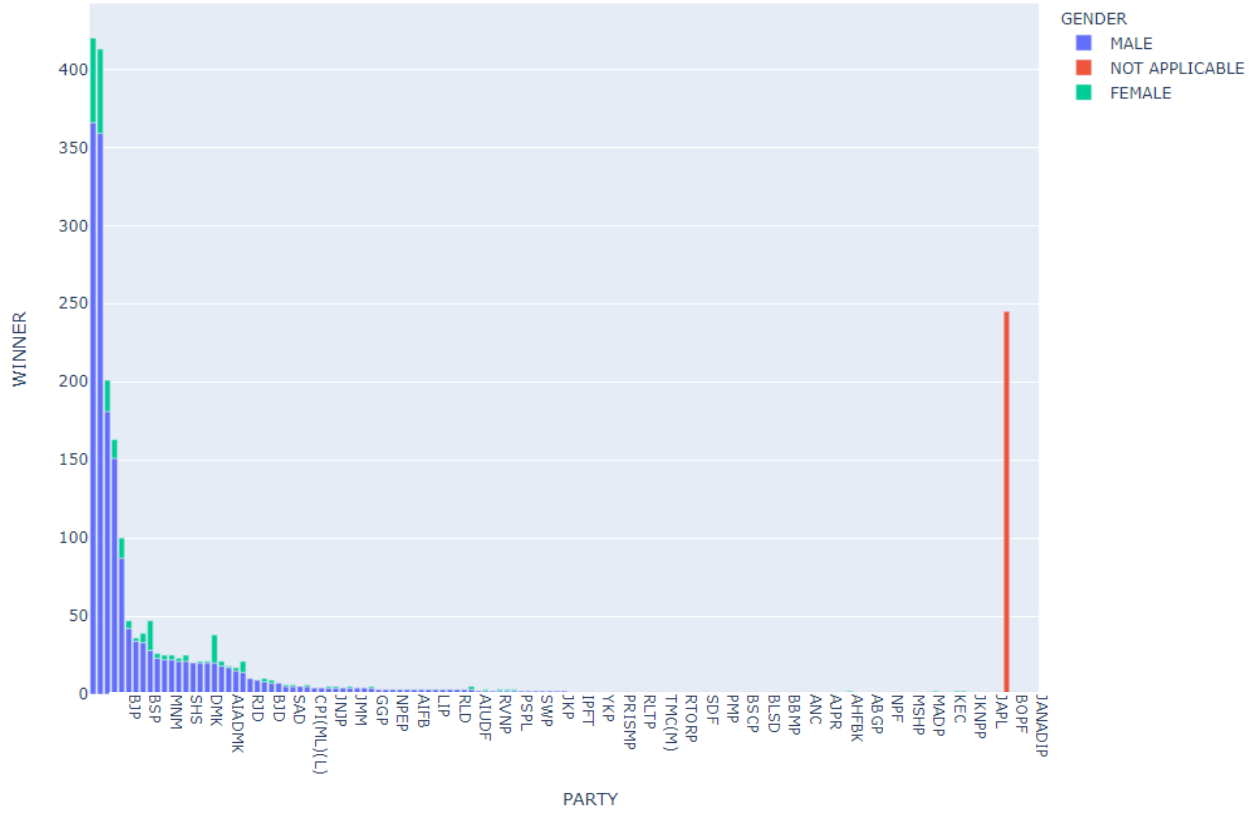

```
#Creating Age_Group from Age column
AGE_GROUP = []
```

```
for i in LS2019Cnd['AGE']:
    if i >= 24 and i <=35:
        AGE_GROUP.append('YOUNG AGE')
    elif i >= 36 and i<=60:
        AGE_GROUP.append('MIDDLE AGE')
    elif i >=60:
        AGE_GROUP.append('OLD AGE')
    else:
        AGE_GROUP.append('NOT KNOWN')
LS2019Cnd['AGE_GROUP'] = AGE_GROUP


p = LS2019Cnd.groupby(['PARTY','AGE_GROUP'])['WINNER'].sum().reset_index().so
rt_values('WINNER',ascending = False)
p = p[p['WINNER']!=0]
fig = px.bar(p, x='PARTY',y='WINNER',hover_data =['AGE_GROUP'], color='AGE_GR
OUP', height=650)
fig.show()
```

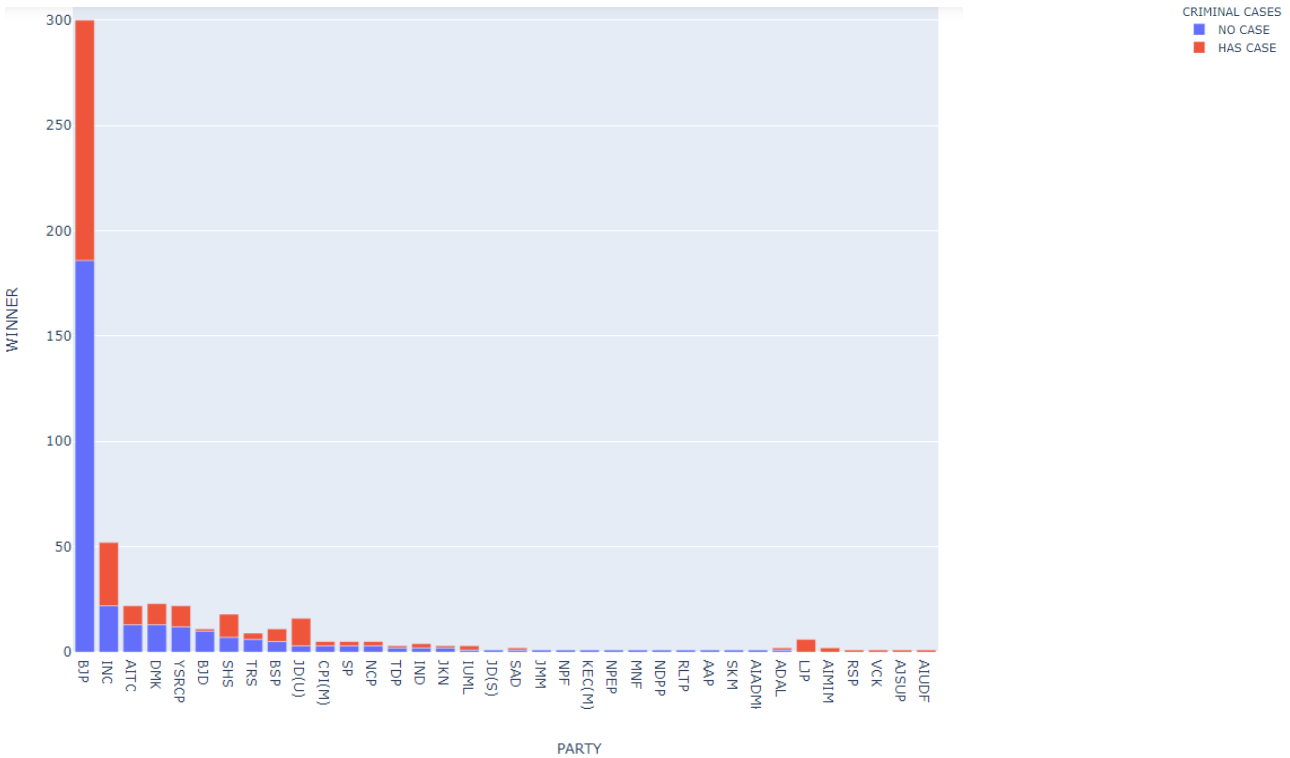



```
p = LS2019Cnd.groupby(['PARTY','GENDER'])['WINNER'].count().reset_index().sor
t_values('WINNER',ascending = False)

#p = p[p['WINNER']!=0]
fig = px.bar(p, x='PARTY',y='WINNER',hover_data =['GENDER'], color='GENDER',
height=700)
```

```
fig.show()
```




```
p = LS2019Cnd.groupby(['PARTY','CRIMINAL CASES'])['WINNER'].sum().reset_index
().sort_values('WINNER',ascending = False)
p = p[p['WINNER']!=0]
fig = px.bar(p, x='PARTY',y='WINNER',hover_data =['CRIMINAL CASES'], color='C
RIMINAL CASES', height=750)
fig.show()
```



```
from sklearn.preprocessing import LabelEncoder
labelEncoder_X = LabelEncoder()
LS2019Cnd['STATE'] = labelEncoder_X.fit_transform(LS2019Cnd['STATE'])
LS2019Cnd['CONSTITUENCY'] = labelEncoder_X.fit_transform(LS2019Cnd['CONSTITUE
NCY'])
LS2019Cnd['NAME'] = labelEncoder_X.fit_transform(LS2019Cnd['NAME'])
LS2019Cnd['PARTY'] = labelEncoder_X.fit_transform(LS2019Cnd['PARTY'])
LS2019Cnd['SYMBOL'] = labelEncoder_X.fit_transform(LS2019Cnd['SYMBOL'])
LS2019Cnd['GENDER'] = labelEncoder_X.fit_transform(LS2019Cnd['GENDER'])
LS2019Cnd['CRIMINAL CASES'] = labelEncoder_X.fit_transform(LS2019Cnd['CRIMINA
L CASES'])
LS2019Cnd['CATEGORY'] = labelEncoder_X.fit_transform(LS2019Cnd['CATEGORY'])
LS2019Cnd['EDUCATION'] = labelEncoder_X.fit_transform(LS2019Cnd['EDUCATION'])
LS2019Cnd['STATUS'] = labelEncoder_X.fit_transform(LS2019Cnd['STATUS'])
LS2019Cnd['AGE_GROUP'] = labelEncoder_X.fit_transform(LS2019Cnd['AGE_GROUP'])
X=LS2019Cnd.drop(['WINNER','ASSETS','LIABILITIES','GENERAL VOTES','POSTAL VOT
ES','AGE','OVER TOTAL ELECTORS IN CONSTITUENCY','OVER TOTAL VOTES POLLED IN C
ONSTITUENCY'],axis=1)
Y=LS2019Cnd['WINNER']
```

X

Out[88]:

	STATE	CONSTITUENCY	NAME	PARTY	SYMBOL	GENDER	CRIMINAL CASES	CATEGORY	EDUCATION	TOTAL VOTES	TOTAL ELECTORS	STATUS	AGE_GROUP
0	31	0	1713	26	80	1	0	3	1	377374	1489790	4	0
1	31	0	700	120	32	1	1	3	12	318814	1489790	9	0
2	31	0	1498	46	66	1	0	3	1	314238	1489790	9	0
3	31	0	1203	81	87	2	1	1	9	13036	1489790	6	1
4	33	1	1789	26	80	1	0	2	4	646875	1937690	9	0
5	33	1	1118	35	49	1	1	2	12	435329	1937690	1	0
6	33	1	1375	46	66	0	1	2	12	45149	1937690	9	0
7	19	2	609	26	80	1	1	0	4	704660	1861396	1	0
8	19	2	1617	78	37	1	0	0	5	423186	1861396	9	3
9	19	2	1728	123	43	1	1	0	5	31807	1861396	9	2
10	10	3	1350	26	80	1	1	0	11	749834	1811851	9	0

Y

Out[89]:

0	1
1	0
2	0
3	0
4	1
5	0
6	0
7	1
8	0
9	0
10	1
11	0
12	1
13	0
14	0
15	0
16	1
17	0
18	0

```
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.3,random_sta
te=42)


from sklearn.preprocessing import StandardScaler
Scaler_X = StandardScaler()
X_train = Scaler_X.fit_transform(X_train)
X_test = Scaler_X.transform(X_test)


from sklearn.metrics import confusion_matrix, accuracy_score

#Logistic Regression
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression()
lr.fit(X_train,Y_train)
Y_pred = lr.predict(X_test)

print(accuracy_score(Y_test,Y_pred))
print(confusion_matrix(Y_test,Y_pred))
```

```
0.9101620029455081
[[464  26]
 [ 35 154]]
```

```
#Decision Tree
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier()
dtree.fit(X_train,Y_train)
predictions = dtree.predict(X_test)
print(accuracy_score(Y_test,predictions ))
print(confusion_matrix(Y_test,predictions ))
```

```
0.8571428571428571
[[448  42]
 [ 55 134]]
```

```
#Random Forest
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=200)
rfc.fit(X_train,Y_train)
rfc_pred = rfc.predict(X_test)
print(accuracy_score(Y_test,rfc_pred))
print(confusion_matrix(Y_test,rfc_pred))
print(rfc_pred)
```

```
0.9131075110456554
[[470  20]
 [ 39 150]]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1
 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 1 0 0 1 0 1 0 1 1 1 0 0 0 0 0 0 1
 0 0 0 0 0 1 0 0 0 1 0 1 1 0 0 1 0 1 0 0 1 0 1 0 1 0 1 0 0 1 0 0 0 0 0 1 0
 0 0 1 0 1 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1
 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0
 0 1 1 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 0 0 0 1 0 1 0 0 1
 0 1 0 0 0 1 1 0 0 1 0 0 1 0 0 0 1 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 0
 0 0 0 1 0 0 0 0 0 1 1 1 1 1 0 1 1 0 1 0 0 1 0 0 1 0 0 0 0 0 1 1 0 0 0 1 0
 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0
 0 1 0 1 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 1 1 0 1 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 1 1 0 0 1 1 0 0 1 0 1 1 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1
 0 0 0 1 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 1 0
 1 0 0 0 0 0 1 0 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0
 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0
 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 1 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 1
 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 1 0 0 0 1 0 1 1 0 0 0 0 0 1 1 0 1 0 0 1 1
 1 0 0 0 0 0 1 0 0 1 0 0 1]
```

```
#Logistic Regression

lr = LogisticRegression()
lr.fit(X_train,Y_train)
Y_pred = lr.predict(X_test)

print(accuracy_score(Y_test,Y_pred))
cm = confusion_matrix(Y_test,Y_pred)
print(cm)
print(classification_report(Y_test,Y_pred))
```

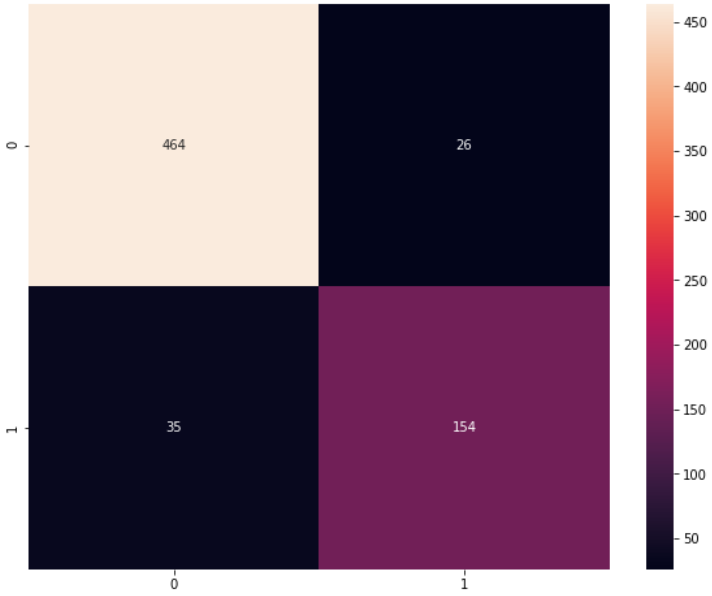
```
0.9101620029455081
[[464  26]
 [ 35 154]]
              precision    recall  f1-score   support

           0       0.93      0.95      0.94       490
           1       0.86      0.81      0.83       189

    accuracy                           0.91       679
   macro avg       0.89      0.88      0.89       679
weighted avg       0.91      0.91      0.91       679
```

```
print("Precision score: {}".format(precision_score(Y_test,Y_pred)))
```

```
Precision score: 0.8555555555555555
```

```
sb.heatmap(cm,annot=True,fmt="d")
```

```
plt.show()
```




```
print("Recall score: {}".format(recall_score(Y_test,Y_pred)))
```

```
Recall score: 0.8148148148148148
```

```
print("F1 Score: {}".format(f1_score(Y_test, Y_pred)))
```

```
F1 Score: 0.8346883468834688
```

```
print('Average precision-recall score: {0:0.2f}'.format(
    average_precision_score(Y_test, Y_pred)))
```

```
Average precision-recall score: 0.75
```

❖ Evaluation

The model uses predictive analysis methods to analyze data based on the datasets and calculate the predictive results. Here the model predicts using the precision, recall methods respectively and uses f1-score to measure the accuracy of the model predicting the voting results.

1. f1-score to measure the accuracy of the prediction of the year 2014.

```
print("F1 Score: {}".format(f1_score(Y14, PRED_Y14)))
```

```
F1 Score: 0.6902654867256638
```

2. f1-score to measure the a**cc**uracy of the prediction of the year 2019.

```
print("F1 Score: {}".format(f1_score(Y19, Y_pred)))
```

```
F1 Score: 0.6454767726161369
```

❖ Future Scope

- We can implement a scope to add new election dataset to our model.
- The model focuses on regression and CART techniques of predictive analytics. Time Series Analysis can also be used daily political opinion polls.
- The model can be further developed into an application for future election prediction and analysis
- The model can be further improvised for better experience.

❖ Limitations

- The more reliable the data is the more precise the expectation. The data used to form an expectation in an analytical procedure. The source of the information available is particularly important. Without a proper reliable dataset, the expected results may hamper.

- There is a direct correlation between the predictability of the data and the quality of the expectation derived from the data. Generally, the more precise an expectation is for an analytical procedure, the greater will be the potential reliability of that procedure. However, the information is subject to data reliability considerations mentioned above.

❖ Conclusion

This project that we undertook was truly a very rewarding experience for us more than one way. It has given a big thrust to our technical knowledge as prospective Software professional. It has also helped us enhance our skills on the personal front. The system has been developed for the given condition and is found working effectively. The developed system is flexible and changes whenever can be made easy. The system was verified with valid as well as invalid data in each manner. The system is run with an insight into the necessary modifications that may require in the future. Hence the system can be maintained successfully without much network.

And we feel extremely satisfied by the fact that we have managed to develop the project of course with equal contribution from all the team members. We think we have exploited the opportunity that came my way to the fullest extent by increasing our technical knowledge and also gaining the valuable work experience apart from studying the other subjects in our curriculum.

❖Bibliography

System analysis and Design: - by Elias M.

Data Analysis with Open Source Tools: - by Philipp K. Janert

Websites:

- https://data.gov.in/
- https://www.google.com/
- https://www.ieee.org/
- https://www.youtube.com/
- https://www.wikipedia.org/
- https://scikit-learn.org/
- https://pandas.pydata.org/
- https://numpy.org/
- https://towardsdatascience.com/