ELECTION ANALYSIS WEBSITE

A Project Report

Submitted in partial fulfillment of the

Requirement for award of the degree of

BACHELOR OF SCIENCE (COMPUTER SCIENCE)

Submitted By

Anshu Dayashankar Tiwari
CS21005

Under the esteemed guidance of

Ms. C. Kalpana

Assistant.Professor

DEPARTMENT OF COMPUTER SCIENCE

S.S.T. COLLEGE OF ARTS & COMMERCE

(Affiliated to University of Mumbai)

ULHASNAGAR, 421004

MAHARASHTRA

2023-2024

S.S.T. COLLEGE OF ARTS & COMMERCE

(Affiliated to University of Mumbai)

ULHASNAGAR, 421004
MAHARASHTRA

2023-2024

<u>CERTIFICATE</u>

This is certify that the project entitled, “**PHISHING URLS DETECTOR”**, is bonafied work of **Anshu Dayashankar Tiwari** bearing Seat No.**(CS21005)** submitted in partial fulfillment of the requirements for the award of degree of **BACHELOR OF SCIENCE IN COMPUTER SCIENCE** from University Of Mumbai.

Internal Guide **Coordinator**

External Examiner

Date: **College Seal**

THE APPROVAL PROJECT PROPOSAL

(Note: All entries of the proforma of approval should be filled up with appropriate and complete information. Incomplete proforma of approval in any respect will be summarily rejected.)

PNR No. _ Roll no. :CS21005

1. Name of the Student

Anshu Dayashankar Tiwari

2. Title of the Project

PHISHING URLS DETECTOR

3. Name of the Guide

Asst.Prof. Ms.C.Kalpana

4. Teaching experience of the Guide ______________________________

5. Is this your first submission? Yes □ No □

Signature of the Student Signature of the Guide

Date: ………………… Date: ………………

Signature of the coordinator Date: …………………

ABSTRACT

This project entails the development of an election analysis website using Python, aimed at providing an in-depth examination of electoral data. Leveraging Python's rich ecosystem of libraries, including pandas for data manipulation, matplotlib and seaborn for visualization, and scikit-learn for machine learning, the website will offer a comprehensive suite of analytical tools. Users will have access to features such as interactive dashboards, trend analysis, demographic breakdowns, and predictive modeling, all presented through a user-friendly interface.

The website will empower users to explore electoral data from various angles, enabling insights into voting patterns, candidate performance, and demographic influences on election outcomes. Machine learning algorithms will be employed to generate predictive models, allowing users to forecast election results based on historical data and current trends.

Furthermore, the website will facilitate data-driven discussions and decision-making processes for stakeholders such as political analysts, journalists, campaign strategists, and policymakers. By providing accessible access to complex electoral data through intuitive visualizations and analysis tools, the website aims to enhance transparency, understanding, and engagement with the democratic process.

Through continuous updates and improvements, the election analysis website will serve as a valuable resource for studying past elections, monitoring ongoing campaigns, and predicting future electoral outcomes, contributing to informed decision-making and public discourse.

.

ACKNOWLEDGEMENT

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them. It gives me pleasure in presenting this project report, undertaken by myself as per our 5th Semester Curriculum “**ELECTION ANALYSIS**” on having completed this project. I personally want to thank to my project guide **Asst. Prof. Mr. C Kalpana**. She has provided valuable guidance for me.

I would like to thanks to the **Asst.Porf.Ms. C. Kalpana** H.O.D. DEPT **of B.Sc. INFORMATION TECHNOLOGY**. Especial thanks to **Asst.Prof. Mr. Yogesh Patil** and **Asst.Prof. Ms. Mayra Lachhani** who have given full support for making this project effective. They both have given their precious time for helping me. This project would not have been possible without the efforts of the discrimination stood with me whenever any difficulty came to my way and provided grate support.

DECLARATION

I hereby declare that the project entitled “**ELECTION ANALYSIS”** is done at S.S.T. College in Ulhasnagar has not in been any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than us, no one has submitted to any other university.

The project is done in partial fulfillment of the requirements for theaward of degree of (**BACHELOR OF SCIENCE IN COMPUTER SCIENCE)** to be submitted as final semester project as part of our curriculum.

Name and Signature of the Students

Anshu Dayashankar Tiwari

TABLE OF CONTENT

CHAPTER - 1
INTRODUCTION

This chapter provides a thorough overview of our project, explaining its background, goals, purpose, application, scope, expected outcomes, and report format. Our project, "Election Analysis," aims to provide comprehensive insights and visualizations of electoral data to inform voters and researchers. It facilitates understanding and interpretation of election trends through data-driven analysis and presentation.

In the modern democracy, elections serve as the cornerstone of political participation, decision-making, and governance. Understanding the dynamics, trends, and outcomes of elections is crucial for policymakers, analysts, and the general public alike. With the advent of technology, the process of election analysis has been greatly enhanced, allowing for the extraction of valuable insights from vast amounts of electoral data.

The election analysis website project typically involves collecting, processing, and analyzing election data to provide insightful visualizations and summaries. Initially, data is gathered from various sources, such as official election records or APIs, and organized into a structured format. Statistical analysis and visualization libraries such as Matplotlib or Plotly are employed to generate graphs, charts, and maps illustrating voting patterns, demographics, and electoral trends. These visualizations are integrated into a web application using frameworks like Flask or Django, allowing users to interactively explore and understand the election data. Additionally, machine learning techniques may be applied for predictive modeling or sentiment analysis based on social media data.

The “Election Analysis” project aims to offer a comprehensive and user-friendly platform for analyzing and interpreting election results using Python's robust data processing and visualization capabilities. It aims to empower users with the tools and insights necessary to understand, analyze, and predict political elections effectively. By leveraging Python's capabilities for data analysis and visualization, the website will serve as a valuable resource for election enthusiasts, researchers, journalists, and policymakers alike.

1.1 BACKGROUND

The Election Analysis Website project is a comprehensive platform designed to cater to a wide range of users interested in understanding and analyzing political elections. It aims to offer a multifaceted approach to election data analysis by incorporating various Python libraries and tools for data manipulation, visualization, and interpretation.

At its core, the project involves the aggregation of election data from diverse sources, including official government databases, independent surveys, historical records, and potentially user-contributed datasets. The data collection process will involve web scraping, API integration, and data cleaning techniques to ensure the integrity and accuracy of the information.

Once the data is collected and preprocessed, the website will utilize Python libraries such as Pandas for data manipulation and analysis. This will include tasks such as calculating voter turnout, demographic breakdowns, regional voting patterns, and identifying correlations between different variables. The project will leverage Matplotlib and Seaborn for data visualization, offering users interactive and informative charts, graphs, and maps to explore election trends visually. This will allow users to gain insights into voting behavior, analyze historical election results, and identify emerging patterns and trends.

In addition to descriptive analysis, the website will incorporate predictive modeling using machine learning algorithms. This will enable users to forecast election outcomes based on historical data, polling trends, and other relevant factors. Algorithms such as logistic regression, random forests, and gradient boosting may be employed to develop predictive models and assess the likelihood of different electoral scenarios. On the frontend, the website will provide an intuitive user interface with interactive dashboards and customizable tools for exploring election data. Users will have the ability to filter data, compare different elections, and drill down into specific regions or demographics of interest.

1.1 RESEARCH OBJECTIVES

The main objective of this project is as follows:

- Data Collection and Processing: Gather election data from official sources, clean and integrate it for analysis.
- Exploratory Data Analysis (EDA): Analyze election data to find trends and patterns, using statistical methods and visualization tools.
- Predictive Modeling: Develop machine learning models to predict election outcomes, evaluating accuracy and feature importance.
- Sentiment Analysis and Social Media Mining: Analyze social media sentiment towards candidates, tracking changes and impact on elections.
- Interactive Visualization and User Experience (UX): Design user-friendly dashboards to present election insights effectively, incorporating feedback for improvement.
- Ethical Considerations and Bias Mitigation: Address privacy concerns and algorithmic biases, ensuring transparency and fairness in analysis.
- Scalability and Performance Optimization: Architect website infrastructure for handling varying traffic and data volume, optimizing algorithms and database queries for efficiency.
- User Engagement and Impact Assessment: Assess website's impact on public discourse and civic engagement, refining features based on user interactions and feedback.

1.2 PURPOSE, SCOPE, APPLICABILITY

1.2.1 PURPOSE:

The purpose of the Election Analysis Website project is to provide a comprehensive platform for analyzing election data using Python. Through this website, users can access detailed insights, trends, and visualizations derived from electoral information, fostering a deeper understanding of political dynamics. The website will offer features such as real-time updates, historical data comparison, constituency-wise analysis, demographic breakdowns, and sentiment analysis of social media trends. By leveraging Python's powerful data analysis libraries like Pandas, Matplotlib, and Seaborn, the website aims to deliver accurate and interactive visual representations of election data. Additionally, users can engage with predictive models to forecast electoral outcomes based on various factors. This project serves not only as a tool for political enthusiasts, journalists, and researchers to scrutinize election patterns but also as an educational resource for promoting data literacy and civic engagement. Through accessible and informative analysis, the Election Analysis Website aims to contribute to a more informed electorate and foster transparency in democratic processes..

1.2.2 SCOPE:

The scope of this project **:-**

A Python-based election analysis website project holds immense potential in providing comprehensive insights into electoral processes. Through data scraping techniques, it can gather real-time data from various sources, including official election commissions, social media platforms, and news outlets. Utilizing data visualization libraries like Matplotlib and Plotly, the website can present intricate trends, voter demographics, and constituency-wise analysis, empowering users to grasp the political landscape effectively. Implementing machine learning algorithms such as sentiment analysis on social media feeds can offer sentiment-based predictions and gauge public opinion. Integration of interactive features like live result tracking, candidate profiles, and historical data comparison can enhance user engagement. Additionally, incorporating security measures to prevent data manipulation and ensuring user privacy would be critical. Overall, a Python-based election analysis website has the potential to be a powerful tool for political analysts, journalists, and voters alike, fostering informed decision-making and promoting transparency in democratic processes.

1.2.3 APPLICABILITY:

The Election Analysis Website Project, developed using Python, offers a multifaceted toolset with broad applicability across various domains. This project enables in-depth scrutiny and visualization of election data, fostering insights crucial for political campaigns, academic research, and public awareness. By leveraging Python's robust libraries such as Pandas, Matplotlib, and Plotly, the website delivers dynamic visualizations like interactive maps, charts, and graphs, facilitating comprehensive analyses of voting trends, demographic patterns, and candidate performances. Its user-friendly interface empowers policymakers, journalists, and citizens alike to explore election dynamics effortlessly. Furthermore, the project's flexibility allows for customization, accommodating diverse data sources and research objectives. From predicting electoral outcomes to evaluating campaign strategies, this Python-based platform serves as a valuable resource for understanding and navigating the complexities of democratic processes.

1.3 ORGANISATION OF REPORT

Throughout the remainder of this report, I describe the process I used to create the website, from research to development. Each chapter focuses on a specific aspect of the process.

Chapter 2 discusses the survey of technologies and websites and presents a high-level overview of ideas related to proposed system.

Chapter 3 requirements and analysis and the process involved in creating the different user interface for user as well as analysis of proposed system.

Chapter 4 describes the system design and how system will going to working future with algorithm and detailed explanation about proposed system.

CHAPTER - 2
SURVEY OF TECHNOLOGIES

2.1 EXISTING SYSTEM

- Election Analysis with Django: Leveraging Django, a Python web framework, this system allows users to input election data, perform various analyses such as voter demographics, candidate performance, and trends over time. It utilizes Django's ORM for database management and integrates libraries like Matplotlib for data visualization. Users can generate reports and visualize insights through interactive charts and graphs.

- Election Analysis with Flask and Pandas: Built with Flask for the web framework and Pandas for data manipulation, this system offers comprehensive election analysis functionalities. Users can upload election datasets in various formats, conduct statistical analyses, and extract meaningful insights. It employs Flask's lightweight structure for rapid development and integrates Pandas for efficient data handling, enabling users to explore election data with ease.

- Election Analysis with Scrapy and Dash: Utilizing Scrapy for web scraping and Dash for building interactive web applications, this system automates data collection from multiple sources such as election commission websites and social media platforms. It provides real-time updates on election-related news, sentiment analysis of public opinions, and visualization of candidate performance. By combining Scrapy's web scraping capabilities with Dash's interactive visualization, users can gain valuable insights into election dynamics and trends.

Limitations of the Existing System :-

- One common limitation among these existing systems is scalability. As election datasets grow larger or as the user base expands, the systems may struggle to handle increased traffic and data volume. This could lead to performance issues, longer processing times, or even system crashes under heavy loads.

- Another common limitation is the level of customization available to users. While these systems offer pre-built functionalities for election analysis, they may lack flexibility in allowing users to tailor the analysis according to specific requirements or to integrate additional data sources easily. This limitation could hinder users who need specialized analyses or wish to incorporate unique data sources into their election analysis projects.

2.2 PROPOSED SYSTEM

An election analysis website project using Python could involve several key components. Firstly, data retrieval and preprocessing would be crucial. Python libraries like BeautifulSoup or Scrapy could be utilized for web scraping to gather election data from various sources such as official government websites or reputable news outlets. Once the data is collected, it would need to be cleaned and organized, which could be accomplished using libraries like Pandas for data manipulation and cleaning. Next, analysis and visualization tools such as Matplotlib, Seaborn, or Plotly could be employed to uncover trends, patterns, and insights from the data. This could involve examining voter demographics, candidate performance, geographic voting patterns, and more. Additionally, machine learning techniques could be applied for predictive modeling or clustering analysis to gain deeper insights into voter behavior. Finally, the results of the analysis could be presented to users through an interactive web interface built using frameworks like Django or Flask, allowing users to explore the data and findings dynamically. This system would provide a comprehensive platform for election analysis, offering both data-driven insights and user-friendly interaction..

2.3 TECHNOLOGY SURVEY

We are using 9th generation Intel core i3 processor

The Operating system which is used for our project is Windows 10 because in this operating system is stable and supports more features and is more user friendly.

Ram 4GB is used is used as it will provide fast reading and writing capabilities and will in turn support in processing.

2.3.1 Visual Studio Code:-

Visual Studio Code is a free source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.

Instead of a project system, it allows users to open one or more directories, which can then be saved in workspaces for future reuse. This allows it to operate as a language-agnostic code editor for any language. Itsupports a number of programming languages and a set of features that differs per language. Unwanted files and folders can be excluded from the project tree via the settings. Many Visual Studio Code features are not exposed through menus or the user interface, but can be accessed via the command palette.

2.3.2 React JS:-

React is a JavaScript library for building user interfaces. It is declarative, efficient, and flexible. React makes it easy to create interactive UIs by using a component-based approach. React components are reusable pieces of code that can be combined to create complex UIs.

Components can be nested to create hierarchies, and they can be updated independently of each other. This makes React UIs very efficient and easy to maintain.

React is also very fast. It uses a virtual DOM to efficiently update the UI. The virtual DOM is a lightweight representation of the real DOM, and it is used to calculate the changes that need to be made to the real DOM. This makes React UIs very responsive and smooth.

2.3.3 Python Flask:-

Flask is a lightweight and versatile Python web framework known for its simplicity and flexibility. It provides the essential tools to build web applications, making it ideal for developers to create web services quickly. Flask follows a micro-framework design, allowing developers to add libraries and components as needed.

It offers URL routing, request handling, and templating, simplifying the development process. Flask's minimalist structure enables developers to focus on their specific project requirements, making it an excellent choice for developing small to medium-sized web applications, APIs, and prototypes. Its open-source nature, extensive documentation, and a thriving community make Flask a popular choice for web development tasks.

2.3.3 Firebase Real-time Database:-

Firebase Real-time Database is a cloud-hosted NoSQL database provided by Google as part of the Firebase platform. It allows developers to store and synchronize data in real-time across multiple clients, including web, mobile, and server applications.

Firebase Real-time Database uses JSON as its data format and offers real-time synchronization, offline data access, and robust security rules to control data access.

Developers can easily integrate it into their applications using Firebase SDKs, making it a powerful tool for building collaborative and responsive apps. The database's server less nature removes the need for complex server setups, making it an attractive choice for mobile and web app development.

2.3.5 VirusTotal API:-

Virus Total is a service that analyzes files and URLs for malicious activity. It uses a variety of antivirus engines and other security tools to identify malware, viruses, and other threats. Virus Total also provides information about the reputation of files and URLs.

The Virus Total API allows developers to integrate Virus Total functionality into their own applications. The API provides a number of endpoints that can be used to scan files and URLs, get reports on scanned files and URLs, and get information about the reputation of files and URLs.

The Virus Total API is a valuable tool for developers who want to add virus scanning and malware detection capabilities to their applications. It is also a valuable tool for security researchers who want to analyze files and URLs for malicious activity.

2.3.6 Tailwind CSS:-

Tailwind CSS is an extremely adaptable utility-first CSS framework made to make web development more efficient and straightforward. It offers an extensive collection of atomic, pre-defined CSS classes that are ready to use and can be applied straight to HTML elements. By creating these classes with Tailwind CSS, developers can quickly style and organize web interfaces without the need for custom CSS.

This utility-centric methodology promotes consistency, expedites development, and improves maintainability. Tailwind CSS is a flexible and effective option for creating contemporary, responsive, and aesthetically pleasing websites and web apps since it can be expanded and adjusted to meet particular project requirements.

CHAPTER – 3
REQUIREMENTS AND ANALYSIS

In this chapter we will discuss about project requirement and analysis in that we are going to explain the Function, Software and hardware requirement, Planning and Scheduling of the project, Preliminary Product Description and Conceptual Models of project.

3.1 PROBLEM STATEMENT

Problem statement :-

- In the era of democratic processes, understanding election data is crucial for political analysts, policymakers, and the general public. However, navigating through vast amounts of election data can be overwhelming and time-consuming. Hence, there is a need for a user-friendly and efficient platform to analyze and visualize election data.

- The objective of this project is to create a user-friendly web platform that aggregates, analyzes, and presents election data in a comprehensive and visually appealing manner. The website will serve as a one-stop destination for users to access valuable insights, trends, and statistics related to past and ongoing elections at various levels of government.

3.2 REQUIRENMENT SPECIFICATION

3.2.1 Functional Requirements :-

- User Authentication and Authorization:

  Users should be able to register, log in, and manage their accounts. Different levels of access should be provided based on user roles (admin, analyst, regular user).

- Data Collection and Processing:

  The system should be able to collect election data from various sources (e.g., official government websites, APIs, CSV files). Data should be validated, cleaned, and stored securely in a database.

- Data Analysis and Visualization:

  Implement algorithms for analyzing election data, such as calculating voter turnout, candidate performance, and trends over time. Provide interactive visualizations (e.g., charts, graphs, maps) to present the analyzed data in a user-friendly manner.

- Search and Filtering:

  Allow users to search for specific election results based on criteria like location, election type, candidate name, etc. Provide filtering options to refine search results according to user preferences.

- Documentation and Training:

  Document the system architecture, design decisions, and technical specifications for future reference. Provide training sessions or tutorials to help users understand how to use the website effectively.

3.2.1 Non-Functional Requirements :-

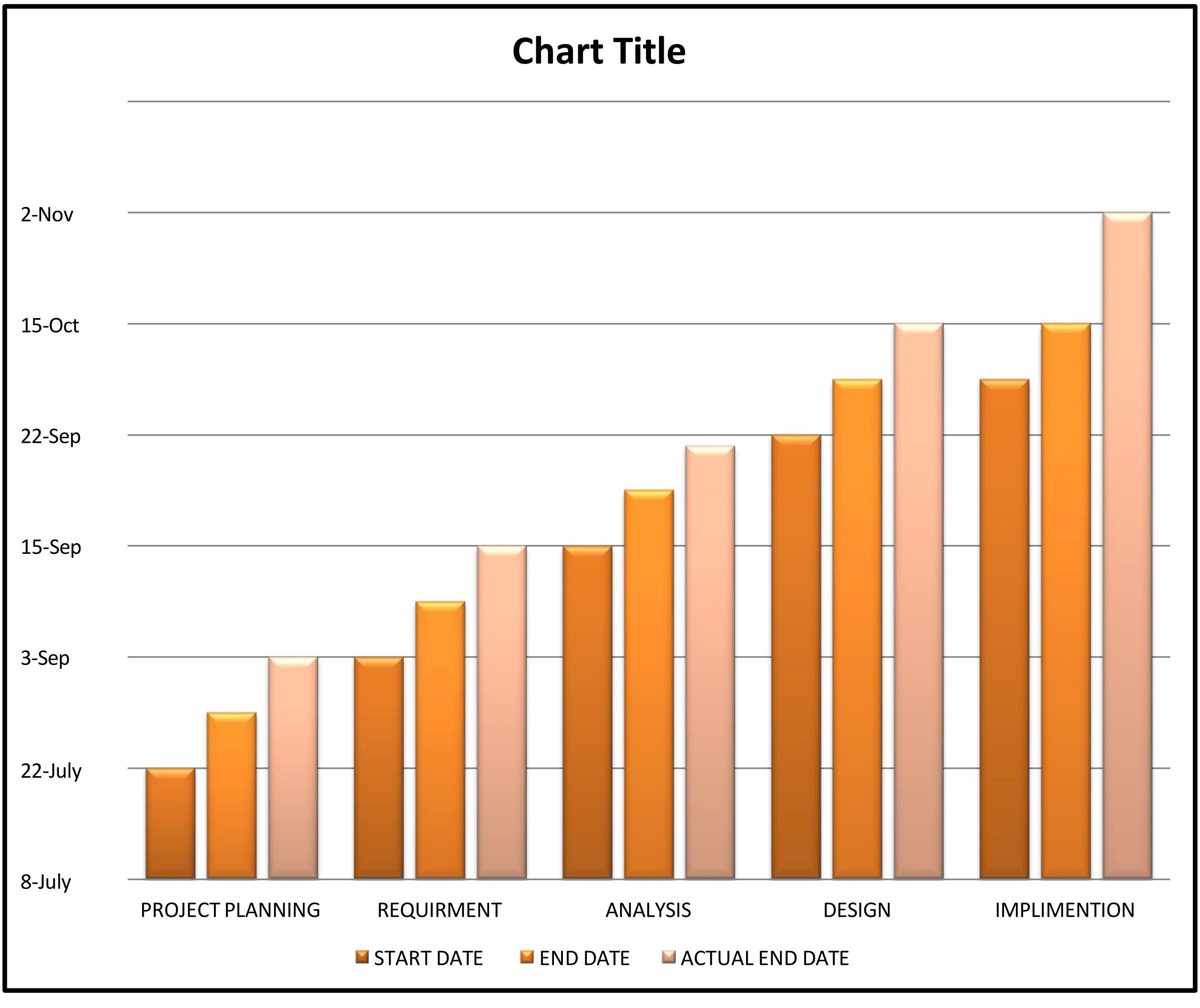
- Performance:
  The website should be able to handle concurrent user requests efficiently, ensuring fast response times even during peak usage periods. Response times for data queries and visualization rendering should be optimized to provide a seamless user experience.

- Scalability:
  The system should be designed to scale horizontally to accommodate increasing data volumes and user traffic over time. Scalability should be achieved through the use of scalable infrastructure components such as cloud services or containerization.

- Security:
  The website should implement robust security measures to protect sensitive data, including user authentication, data encryption, and protection against common web security threats (e.g., SQL injection, cross-site scripting). Compliance with relevant security standards and regulations (e.g., GDPR, HIPAA) should be ensured, especially when handling personally identifiable information (PII).

- Reliability and Availability:
  The website should have high availability, with minimal downtime for maintenance or unexpected issues. Measures such as load balancing, redundancy, and failover mechanisms should be implemented to ensure continuous availability of services. Regular backups of data should be performed to prevent data loss in case of system failures or disasters.

3.3 SCHEDULIMG AND PLANNING GANTT CHART OF ELECTION ANALYSIS



3.3 SOFTWARE AND HARDWARE REQUIREMENT

3.3.1 Hardware Requirement

- Processor : Intel i3/i5/i7
- Accessories: LCD/LED Display, Mouse, Keyboard.
- RAM : 4GB
- Hard Disk : 100GB

3.3.2 Software Requirements

- Operating System : Windows 10(ultimate, enterprise).
- Front-End Development: HTML, CSS.
- Back-End Development: Python.
- Database: Firebase Real-time Database.
- Package Manager: NPM(Node Package Manager) And PIP 3.
- IDE : Visual Studio Code, Jupyter Notebook.

3.4 PRELIMINARY PRODUCT DESCRIPTION

The Election Analysis Website is a platform designed to provide comprehensive analysis and visualization of election data. Leveraging Python, it offers users intuitive tools to explore electoral trends, demographics, and outcomes. With a user-friendly interface and powerful backend processing, the website aims to be a go-to resource for political analysts, journalists, and the general public interested in understanding election dynamics.

3.4.1 Key Features

- Data Import and Processing:

  Utilize Python libraries (such as pandas) to import and process election data from various sources, including official electoral commission websites, APIs, and CSV files. Implement data cleaning and normalization techniques to ensure accuracy and consistency.

- Election Results Analysis:

  Provide detailed analysis of election results at various levels (national, regional, local). Enable users to compare historical election data and track trends over time. Implement statistical tools for forecasting and predicting future election outcomes based on historical patterns.

- Real-time Updates:

  Implement functionality to fetch and update election results in real-time as they become available. Notify users of significant developments or changes in the electoral landscape via email alerts or push notifications.

- User Authentication and Permissions:

  Implement user authentication mechanisms to control access to sensitive data and features. Define user roles and permissions to tailor the experience for different types of users (e.g., admins, analysts, guests).

3.5 CONCEPTUAL MODELS

The Data Flow Diagram (DFD) Server two purpose:-

- To provide an indication Of How data are transformed as they move through the system.

- To Depict the Function that transformed the data flow the DFD provide additional information domain and server as a basic for modeling of function It is graphical Representation that depicts information flow transform that are applied as data move from input to output to basic flow of data diagram is also known as data graph or bubble chart.

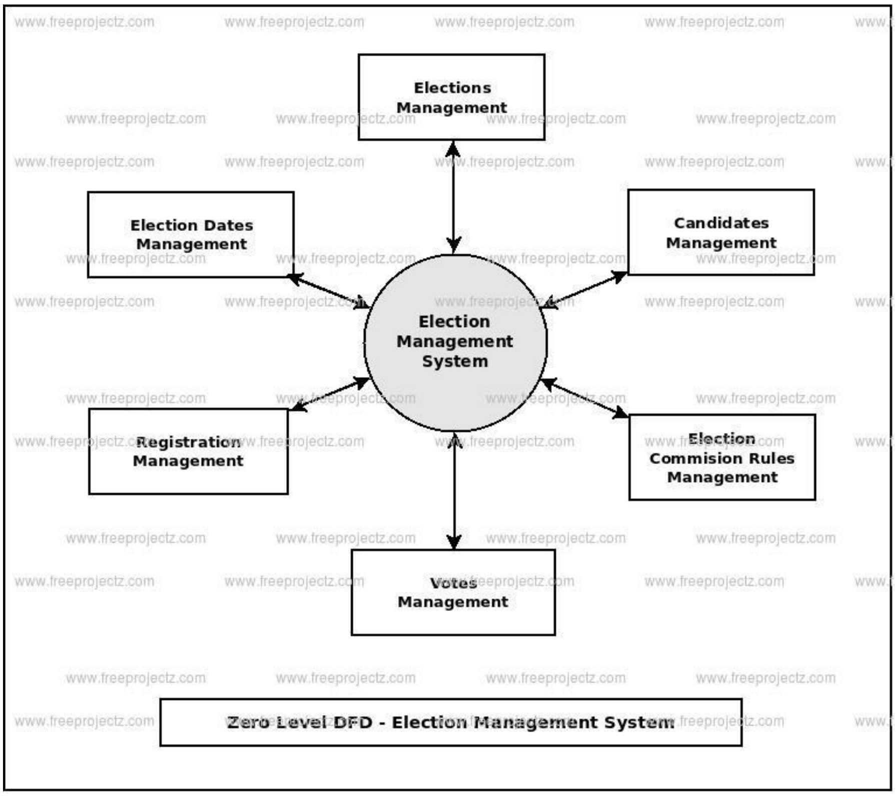

Elections Management
Election Dates Management
Candidates Management
Election Management System
Registration Management
Election Commision Rules Management
Votes Management
Zero Level DFD - Election Management System

Fig.3.5.1 DFD Level 0

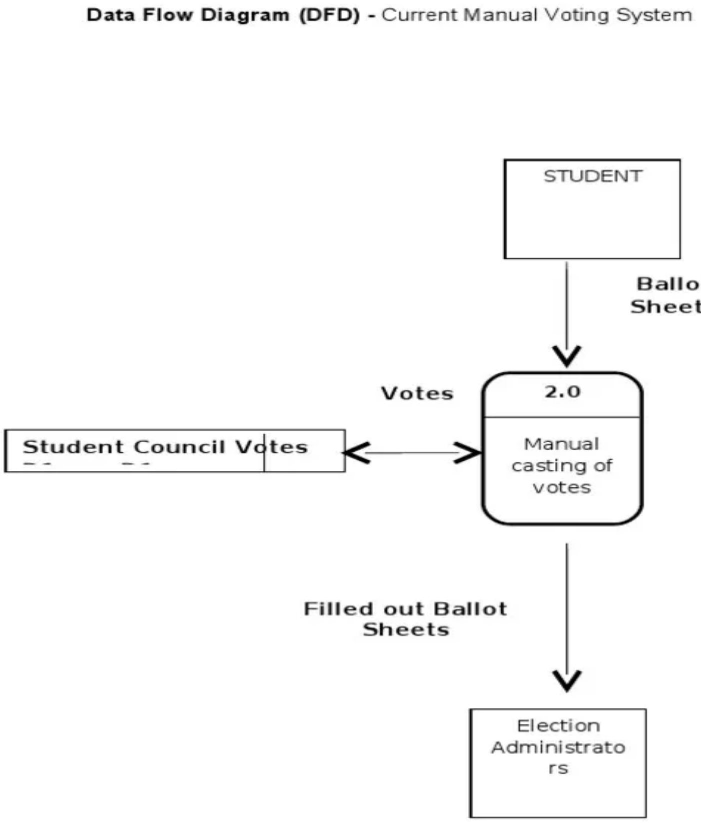


Fig.3.5.2 DFD Level 1

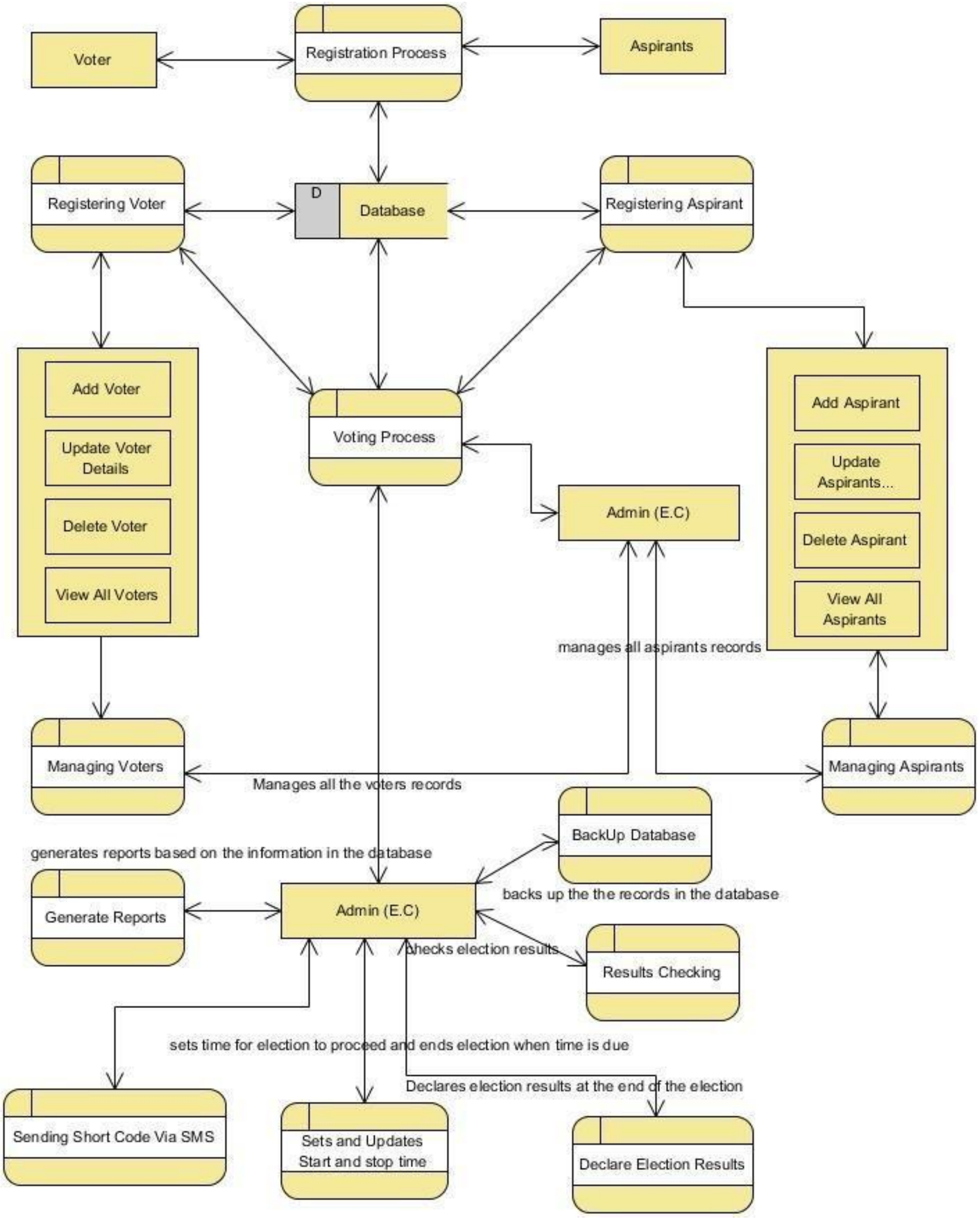
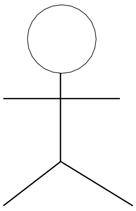
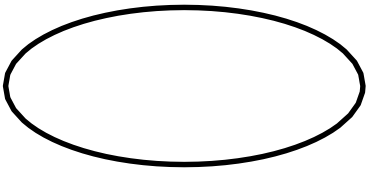
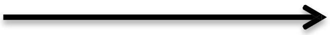


Fig.3.5.4 Flow-Chart

Use Case Diagram

The use case Diagram is used to identify the primary elements & processesthat form the system the primary elements are termed as actors & processer are called use case

Notation	Description
Actor 	Basically the actors are not part of a use case diagram but to beable to create a user case diagram itneeds some actors
Use case 	Function from the use case is to give an illustration of system that is easily to be understand
Relation includes <<include>> 	Behavior that must be met before an event can occur in condition where this use case is partof another use case

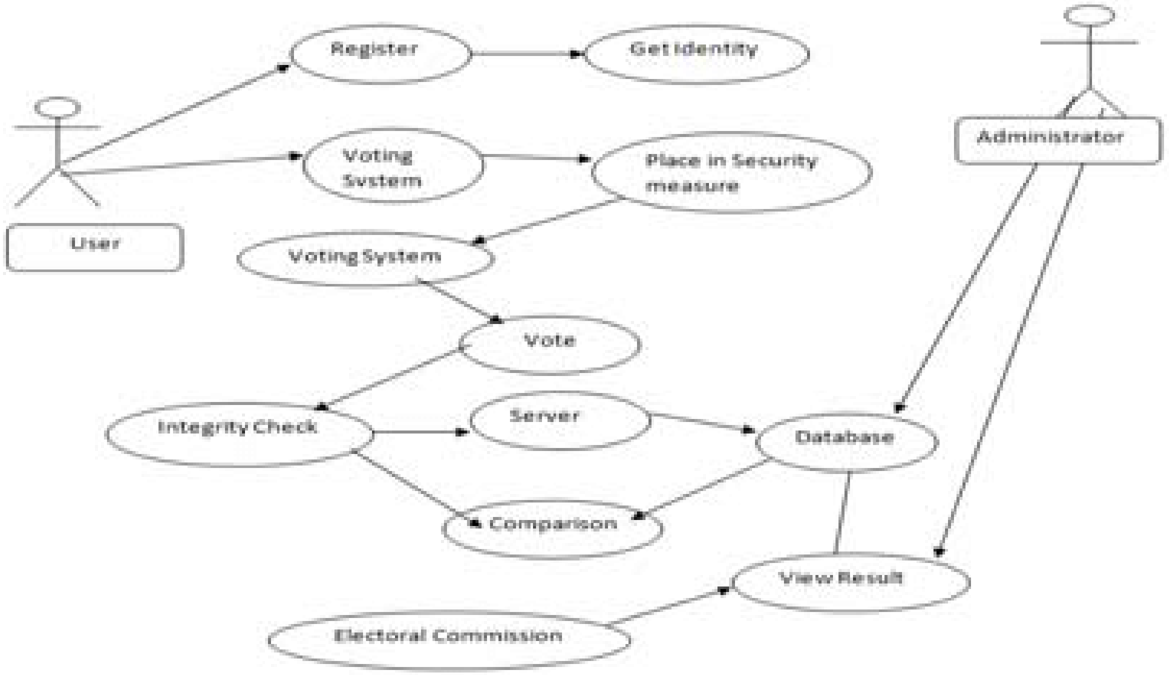


Fig.3.5.5 Use Case Diagram

CHAPTER – 4
SYSTEM DESIGN

4.1 DESIGN DETAILS

In this section, we delve into the intricate details of the Election Analysis website design, outlining key components and their functionalities.

4.1.1 Data Analyzing

- Data Gathering:
  Web Scraping: Use libraries like BeautifulSoup or Scrapy to scrape election data from official government websites, news sources, or reliable databases. You can extract data such as election results, candidate information, demographics, etc.
  APIs: Some organizations provide APIs to access election data. You can use libraries like requests to interact with these APIs and retrieve the required data.
  Public Datasets: Explore publicly available datasets related to elections on platforms like Kaggle or data.gov. These datasets may include historical election results, voter turnout, demographic information, etc.

- Data Storage:
  Choose a suitable database system like SQLite, MySQL, or PostgreSQL to store the collected data. SQLite is a good option for smaller projects, while MySQL or PostgreSQL can handle larger datasets and more complex queries.

- Data Analysis:
  Data Cleaning: Before performing analysis, clean the collected data to handle missing values, inconsistencies, and outliers. Libraries like Pandas

can be used for data cleaning tasks.

Exploratory Data Analysis (EDA): Use descriptive statistics, visualizations, and hypothesis testing techniques to explore the data. Matplotlib, Seaborn, and Plotly are popular libraries for data visualization in Python.

Statistical Analysis: Conduct statistical analysis to identify patterns, correlations, and trends in the data. You can use libraries like SciPy and StatsModels for statistical analysis.

- Machine Learning:

  If you want to predict election outcomes or analyze voter behavior, you can build machine learning models. Libraries like scikit-learn and TensorFlow provide tools for building predictive models and performing advanced analytics.

4.1.2 Phishing Detection

Creating a predictive model to forecast state wise candidates' performance based on collected election data for an election analysis website can be an interesting and challenging task. Here's a general roadmap you can follow using Python:

- Data Preprocessing:

  Clean the data by handling missing values, outliers, and inconsistencies. Encode categorical variables using techniques like one-hot encoding or label encoding. Split the data into training and testing sets.

- Feature Engineering:

  Create new features if necessary. For example, you can derive features like percentage of votes won by a candidate in the previous election, demographic characteristics of the constituency, etc. Select relevant features that contribute significantly to the prediction.

- Regression Model for Vote Share:

  Use regression techniques like Linear Regression, Decision Trees, Random Forest, or Gradient Boosting to predict the percentage of votes each candidate will receive in a particular constituency/state. Train separate models for each state or region to capture regional variations.

- Classification Model for Winner Prediction:
  Convert the problem into a classification task where the goal is to predict the winning candidate in each constituency/state. Train classifiers like Logistic Regression, Random Forest, Support Vector Machines, or Gradient Boosting for this task. You may need to handle class imbalance if one party dominates in certain regions.

4.2 PROCEDURE

This section provides a step-by-step procedure for utilizing the Phishing URL Detector, from inputting URLs to receiving scan results.

4.2.1 Data Collection Procedure

- Data Collection and Preparation:
  Gather historical election data from reliable sources. This data should include information such as candidates' names, parties, vote counts, demographics, previous election results, etc. Clean and preprocess the collected data. This involves handling missing values, removing duplicates, standardizing data formats, and performing any necessary feature engineering.

- Exploratory Data Analysis (EDA):
  Conduct EDA to gain insights into the dataset. This includes visualizations, statistical summaries, and identifying patterns or trends in the data. Analyze the relationships between different variables such as candidate performance, demographics, political parties, etc.

4.2.2 Fea Phishing Detection Procedure

- Feature Engineering:
  Create relevant features such as voter turnout rate, demographics of the constituency, past election results, etc. Feature selection techniques like correlation analysis, PCA, or feature importance can be applied to select the most predictive features.

- Classification Model for Winning/Losing Prediction:
  Train a classification model to predict whether a candidate will win or lose in a given

constituency. Algorithms like Random Forest, Gradient Boosting, or Logistic Regression can be used for this task. Split the dataset into training and testing sets for model evaluation.

- Regression Model for Vote Share Prediction:

  Train a regression model to predict the vote share of each candidate in a constituency. Use algorithms like Linear Regression, Ridge Regression, or Gradient Boosting Regression. Again, split the dataset into training and testing sets for model evaluation.

- State-wise Aggregation:

  Aggregate the predictions from the constituency level to predict state-wise outcomes. For classification, count the number of winning candidates predicted for each party in each state. For regression, calculate the average predicted vote share for each party in each state.

- Evaluation:

  Evaluate the classification model using metrics like accuracy, precision, recall, and F1-score. Evaluate the regression model using metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), or R-squared. Use cross-validation techniques to ensure robustness of the models.

- Visualization and Reporting:

  Visualize the predicted outcomes using plots such as bar charts, heatmaps, or choropleth maps. Present the analysis findings and predictions on an election analysis website. Provide interactive features for users to explore the data and predictions.

- Model Deployment:

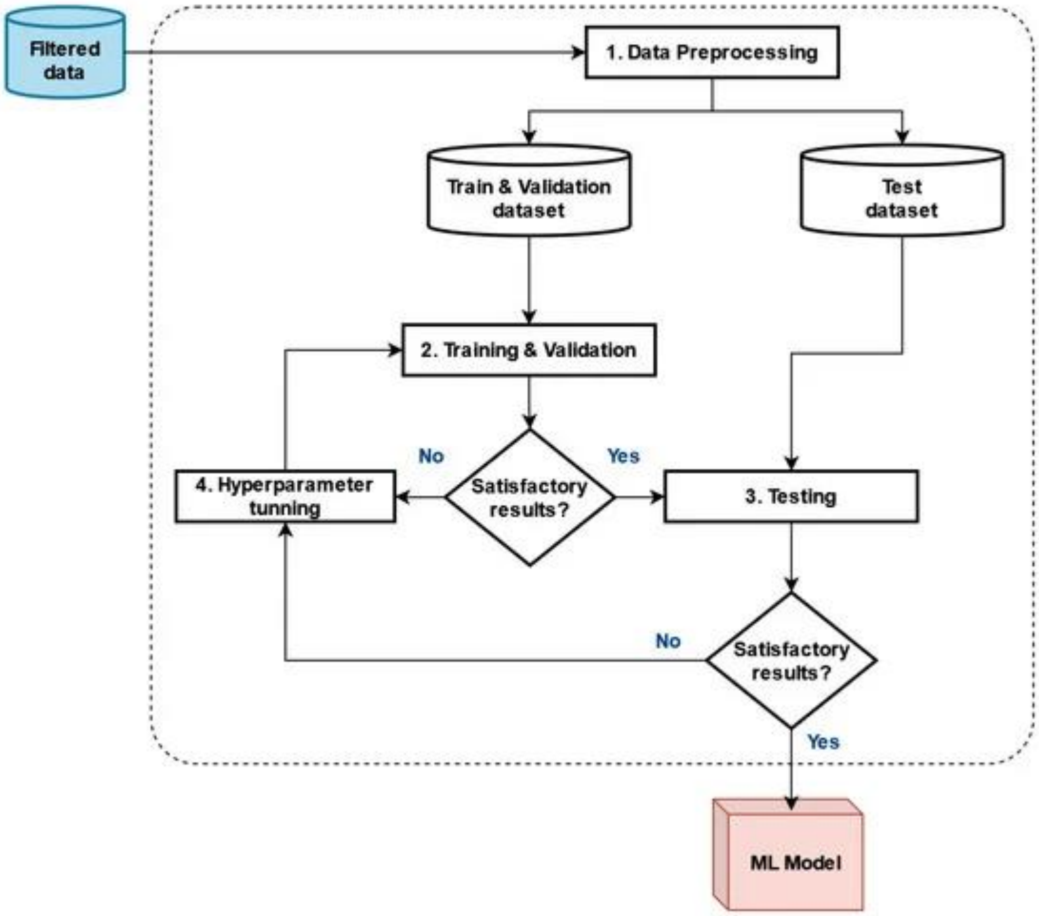
  Deploy the trained models on a web server using frameworks like Flask or Django. Integrate the prediction functionality into the website, allowing users to input new data and get predictions in real-time.

- Maintenance and Updates:

  Regularly update the models with new election data to keep the predictions accurate. Monitor the website for any issues and make necessary improvements based on user feedback.

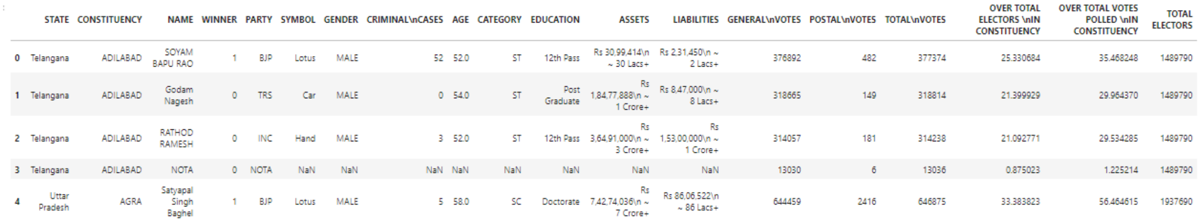
4.3 SYSTEM ARCHITECTURE

The system architecture provides a high-level overview of the Election Analysis Website, illustrating the major components and their interactions.



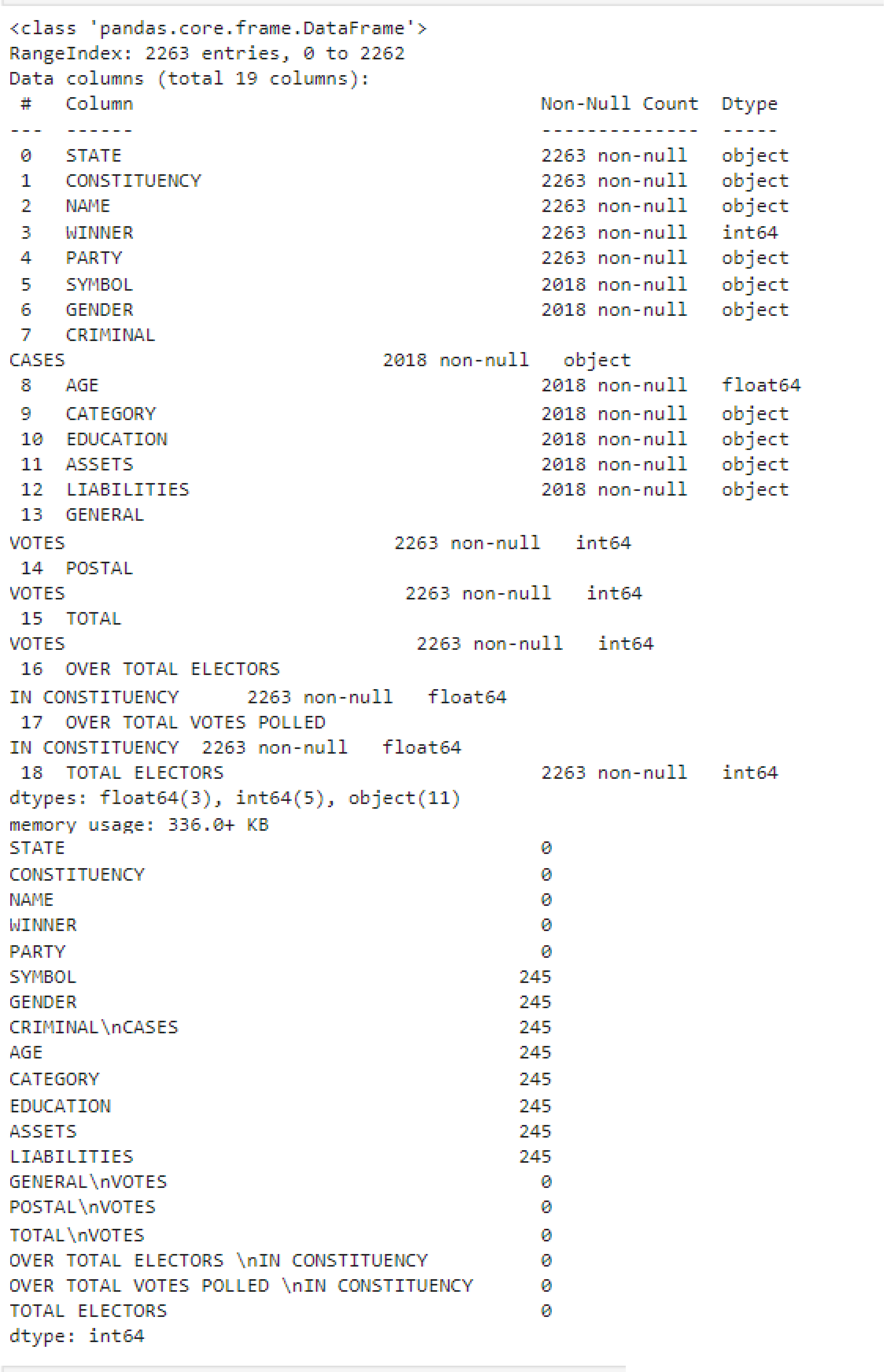
CHAPTER – 5
Implementation & Testing

```
#importing necessary libraries
import numpy as np
import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import random
import plotly.express as px
import plotly.graph_objects as go
import os

for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
election_df=pd.read_csv('LS_2.0.csv')
election_df.head()
```



	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
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+	376892	482	377374	25.330684	35.468248	1489790
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+	318665	149	318814	21.399929	29.964370	1489790
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+	314057	181	314238	21.092771	29.534285	1489790
3	Telangana	ADILABAD	NOTA	0	NOTA	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	13030	6	13036	0.875023	1.225214	1489790
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+	644459	2416	646875	33.383823	56.464615	1937690

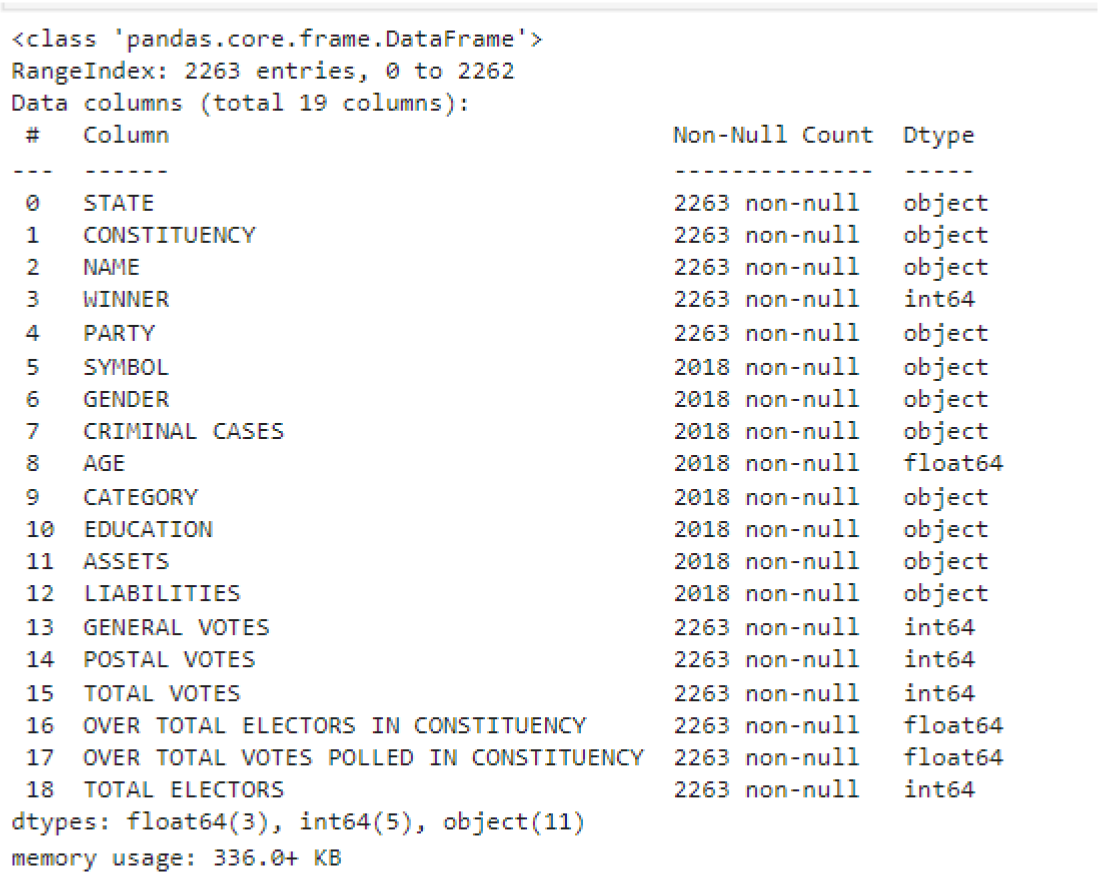
```
election_df.info()
# Checking For Null Values
election_df.isnull().sum()
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2263 entries, 0 to 2262
Data columns (total 19 columns):
 #   Column                                        Non-Null Count  Dtype
---  ------                                        --------------  -----
 0   STATE                                         2263 non-null   object
 1   CONSTITUENCY                                  2263 non-null   object
 2   NAME                                          2263 non-null   object
 3   WINNER                                        2263 non-null   int64
 4   PARTY                                         2263 non-null   object
 5   SYMBOL                                        2018 non-null   object
 6   GENDER                                        2018 non-null   object
 7   CRIMINAL
CASES                            2018 non-null   object
 8   AGE                                           2018 non-null   float64
 9   CATEGORY                                      2018 non-null   object
 10  EDUCATION                                     2018 non-null   object
 11  ASSETS                                        2018 non-null   object
 12  LIABILITIES                                   2018 non-null   object
 13  GENERAL
VOTES                             2263 non-null   int64
 14  POSTAL
VOTES                              2263 non-null   int64
 15  TOTAL
VOTES                               2263 non-null   int64
 16  OVER TOTAL ELECTORS
IN CONSTITUENCY      2263 non-null   float64
 17  OVER TOTAL VOTES POLLED
IN CONSTITUENCY  2263 non-null   float64
 18  TOTAL ELECTORS                                2263 non-null   int64
dtypes: float64(3), int64(5), object(11)
memory usage: 336.0+ KB
STATE                                           0
CONSTITUENCY                                    0
NAME                                            0
WINNER                                          0
PARTY                                           0
SYMBOL                                        245
GENDER                                        245
CRIMINAL\nCASES                               245
AGE                                           245
CATEGORY                                      245
EDUCATION                                     245
ASSETS                                        245
LIABILITIES                                   245
GENERAL\nVOTES                                  0
POSTAL\nVOTES                                   0
TOTAL\nVOTES                                    0
OVER TOTAL ELECTORS \nIN CONSTITUENCY           0
OVER TOTAL VOTES POLLED \nIN CONSTITUENCY       0
TOTAL ELECTORS                                  0
dtype: int64
```

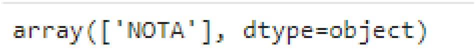
election_df.rename(columns={"CRIMINAL\nCASES": "CRIMINAL CASES", "GENERAL\nVOTES": "GENERAL VOTES", "POSTAL\nVOTES": "POSTAL VOTES","TOTAL\nVOTES": "TOTAL

```
VOTES","OVER TOTAL ELECTORS \nIN CONSTITUENCY": "OVER TOTAL ELECTORS IN CONSTITUENCY","OVER TOTAL VOTES POLLED \nIN CONSTITUENCY": "OVER TOTAL VOTES POLLED IN CONSTITUENCY"}, inplace=True)
election_df.info()
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2263 entries, 0 to 2262
Data columns (total 19 columns):
 #   Column                                   Non-Null Count  Dtype  
---  ------                                   --------------  -----  
 0   STATE                                    2263 non-null   object 
 1   CONSTITUENCY                             2263 non-null   object 
 2   NAME                                     2263 non-null   object 
 3   WINNER                                   2263 non-null   int64  
 4   PARTY                                    2263 non-null   object 
 5   SYMBOL                                   2018 non-null   object 
 6   GENDER                                   2018 non-null   object 
 7   CRIMINAL CASES                           2018 non-null   object 
 8   AGE                                      2018 non-null   float64
 9   CATEGORY                                 2018 non-null   object 
 10  EDUCATION                                2018 non-null   object 
 11  ASSETS                                   2018 non-null   object 
 12  LIABILITIES                              2018 non-null   object 
 13  GENERAL VOTES                            2263 non-null   int64  
 14  POSTAL VOTES                             2263 non-null   int64  
 15  TOTAL VOTES                              2263 non-null   int64  
 16  OVER TOTAL ELECTORS IN CONSTITUENCY      2263 non-null   float64
 17  OVER TOTAL VOTES POLLED IN CONSTITUENCY  2263 non-null   float64
 18  TOTAL ELECTORS                           2263 non-null   int64  
dtypes: float64(3), int64(5), object(11)
memory usage: 336.0+ KB
```

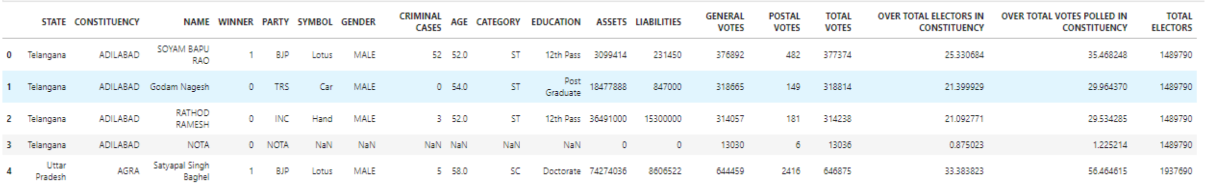
```
#election_df.loc[election_df['SYMBOL'].isnull()==True,['NAME'] ] = 0
election_df[election_df['SYMBOL'].isnull()==True]['NAME'].unique()
```



```
array(['NOTA'], dtype=object)
```

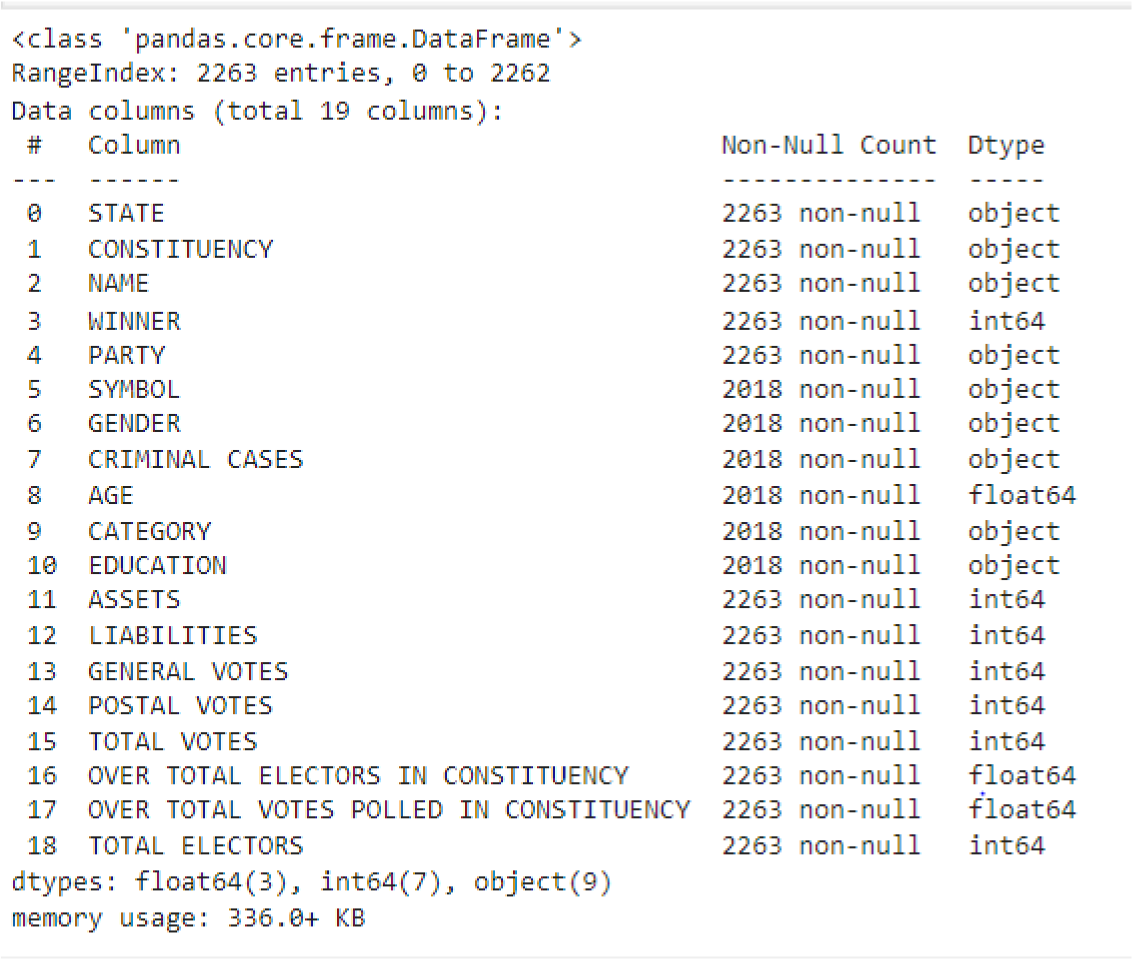
```
# Clean the ASSETS and LIABILITIES col which has input like below:
# Rs 30,99,414
# ~ 30 Lacs+
def data_cleaner(x):
    try:
        x = x.split('\n')[0]
        x = x.replace('Rs', '')
        x = x.replace(',', '')
        x = x.strip()
        return int(x)
    except:
        x = 0
    return int(x)
```

```
election_df['ASSETS'] = election_df['ASSETS'].apply((data_cleaner))
election_df['LIABILITIES'] = election_df['LIABILITIES'].apply((data_cleaner))
election_df.head()
```



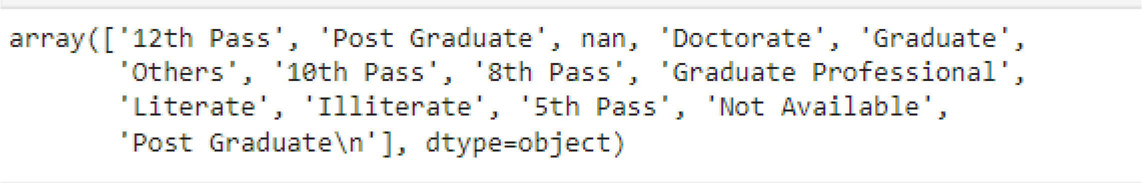
	STATE	CONSTITUENCY	NAME	WINNER	PARTY	SYMBOL	GENDER	CRIMINAL CASES	AGE	CATEGORY	EDUCATION	ASSETS	LIABILITIES	GENERAL VOTES	POSTAL VOTES	TOTAL VOTES	OVER TOTAL ELECTORS IN CONSTITUENCY	OVER TOTAL VOTES POLLED IN CONSTITUENCY	TOTAL ELECTORS
0	Telangana	ADILABAD	SOYAM BAPU RAO	1	BJP	Lotus	MALE	52	52.0	ST	12th Pass	3099414	231450	376892	482	377374	25.330684	35.468248	1489790
1	Telangana	ADILABAD	Godam Nagesh	0	TRS	Car	MALE	0	54.0	ST	Post Graduate	18477888	847000	318665	149	318814	21.399929	29.964370	1489790
2	Telangana	ADILABAD	RATHOD RAMESH	0	INC	Hand	MALE	3	52.0	ST	12th Pass	36491000	15300000	314057	181	314238	21.092771	29.534285	1489790
3	Telangana	ADILABAD	NOTA	0	NOTA	NaN	NaN	NaN	NaN	NaN	NaN	0	0	13030	6	13036	0.875023	1.225214	1489790
4	Uttar Pradesh	AGRA	Satyapal Singh Baghel	1	BJP	Lotus	MALE	5	58.0	SC	Doctorate	74274036	8606522	644459	2416	646875	33.383823	56.464615	1937690

```
election_df.info()
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2263 entries, 0 to 2262
Data columns (total 19 columns):
 #   Column                                   Non-Null Count  Dtype  
---  ------                                   --------------  -----  
 0   STATE                                    2263 non-null   object 
 1   CONSTITUENCY                             2263 non-null   object 
 2   NAME                                     2263 non-null   object 
 3   WINNER                                   2263 non-null   int64  
 4   PARTY                                    2263 non-null   object 
 5   SYMBOL                                   2018 non-null   object 
 6   GENDER                                   2018 non-null   object 
 7   CRIMINAL CASES                           2018 non-null   object 
 8   AGE                                      2018 non-null   float64
 9   CATEGORY                                 2018 non-null   object 
 10  EDUCATION                                2018 non-null   object 
 11  ASSETS                                   2263 non-null   int64  
 12  LIABILITIES                              2263 non-null   int64  
 13  GENERAL VOTES                            2263 non-null   int64  
 14  POSTAL VOTES                             2263 non-null   int64  
 15  TOTAL VOTES                              2263 non-null   int64  
 16  OVER TOTAL ELECTORS IN CONSTITUENCY      2263 non-null   float64
 17  OVER TOTAL VOTES POLLED IN CONSTITUENCY  2263 non-null   float64
 18  TOTAL ELECTORS                           2263 non-null   int64  
dtypes: float64(3), int64(7), object(9)
memory usage: 336.0+ KB
```

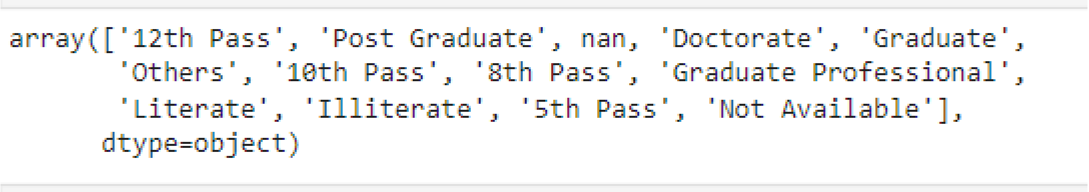
```
election_df['EDUCATION'].unique()
```



```
array(['12th Pass', 'Post Graduate', nan, 'Doctorate', 'Graduate',
       'Others', '10th Pass', '8th Pass', 'Graduate Professional',
       'Literate', 'Illiterate', '5th Pass', 'Not Available',
       'Post Graduate\n'], dtype=object)
```

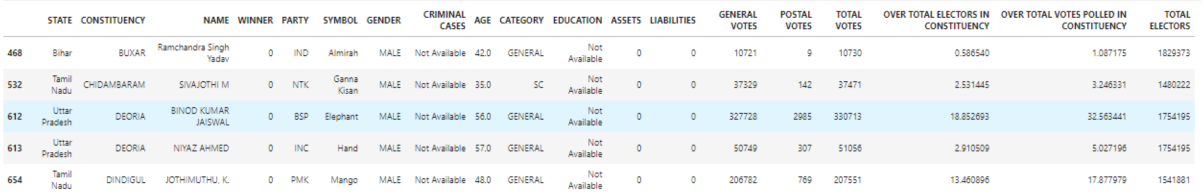
```
election_df['EDUCATION'].replace({'Post Graduate\n':'Post Graduate'},inplace=True)
```

```
election_df['EDUCATION'].unique()
```



```
array(['12th Pass', 'Post Graduate', nan, 'Doctorate', 'Graduate',
       'Others', '10th Pass', '8th Pass', 'Graduate Professional',
       'Literate', 'Illiterate', '5th Pass', 'Not Available'],
      dtype=object)
```

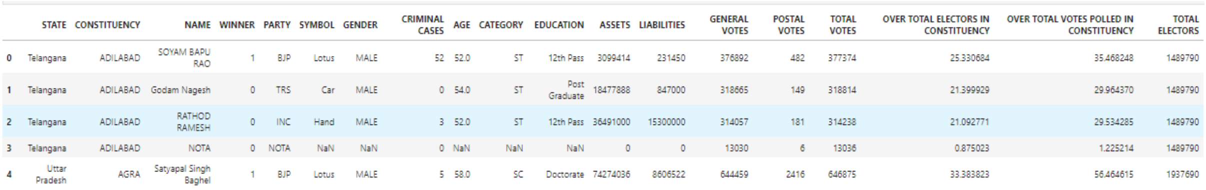
```
election_df[election_df['CRIMINAL CASES']=='Not Available'].head()
```



	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
468	Bihar	BUXAR	Ramchandra Singh Yadav	0	IND	Almirah	MALE	Not Available	42.0	GENERAL	Not Available	0	0	10721	9	10730	0.586540	1.087175	1829373
532	Tamil Nadu	CHIDAMBARAM	SIVAJOTHI M	0	NTK	Ganna Kisan	MALE	Not Available	35.0	SC	Not Available	0	0	37329	142	37471	2.531445	3.246331	1480222
612	Uttar Pradesh	DEORIA	BINOD KUMAR JAISWAL	0	BSP	Elephant	MALE	Not Available	56.0	GENERAL	Not Available	0	0	327728	2985	330713	18.852693	32.563441	1754195
613	Uttar Pradesh	DEORIA	NIYAZ AHMED	0	INC	Hand	MALE	Not Available	57.0	GENERAL	Not Available	0	0	50749	307	51056	2.910509	5.027196	1754195
654	Tamil Nadu	DINDIGUL	JOTHIMUTHU, K.	0	PMK	Mango	MALE	Not Available	48.0	GENERAL	Not Available	0	0	206782	769	207551	13.460896	17.877979	1541881

```
election_df['ASSETS']=pd.to_numeric(election_df['ASSETS'])
election_df['LIABILITIES']=pd.to_numeric(election_df['LIABILITIES'])
election_df['CRIMINAL CASES'].replace({np.NaN:0})
election_df['CRIMINAL CASES'] = pd.to_numeric(election_df['CRIMINAL CASES'],
errors='coerce').fillna(0).astype(np.int64)]
```

```
election_df.head()
```

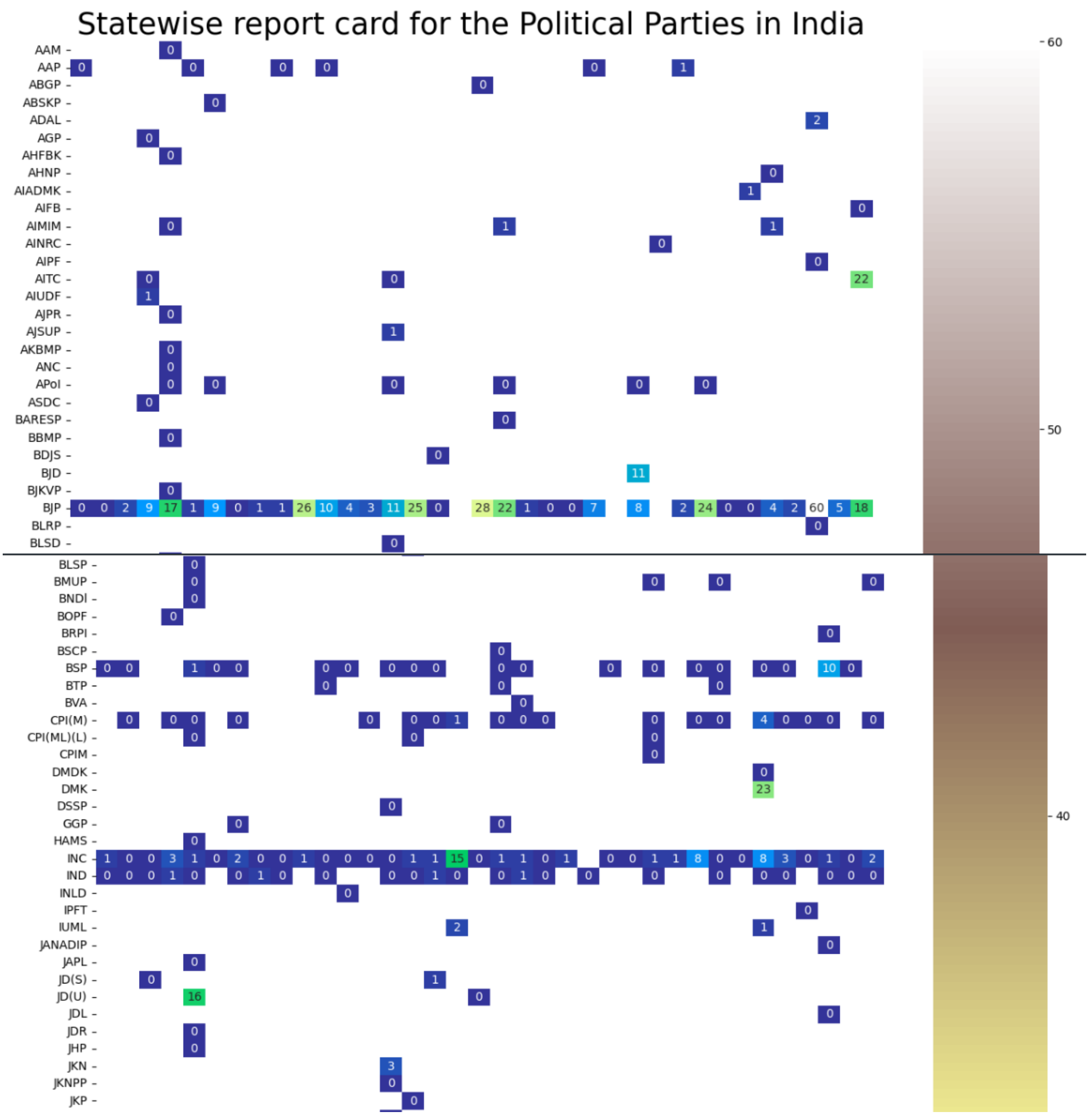


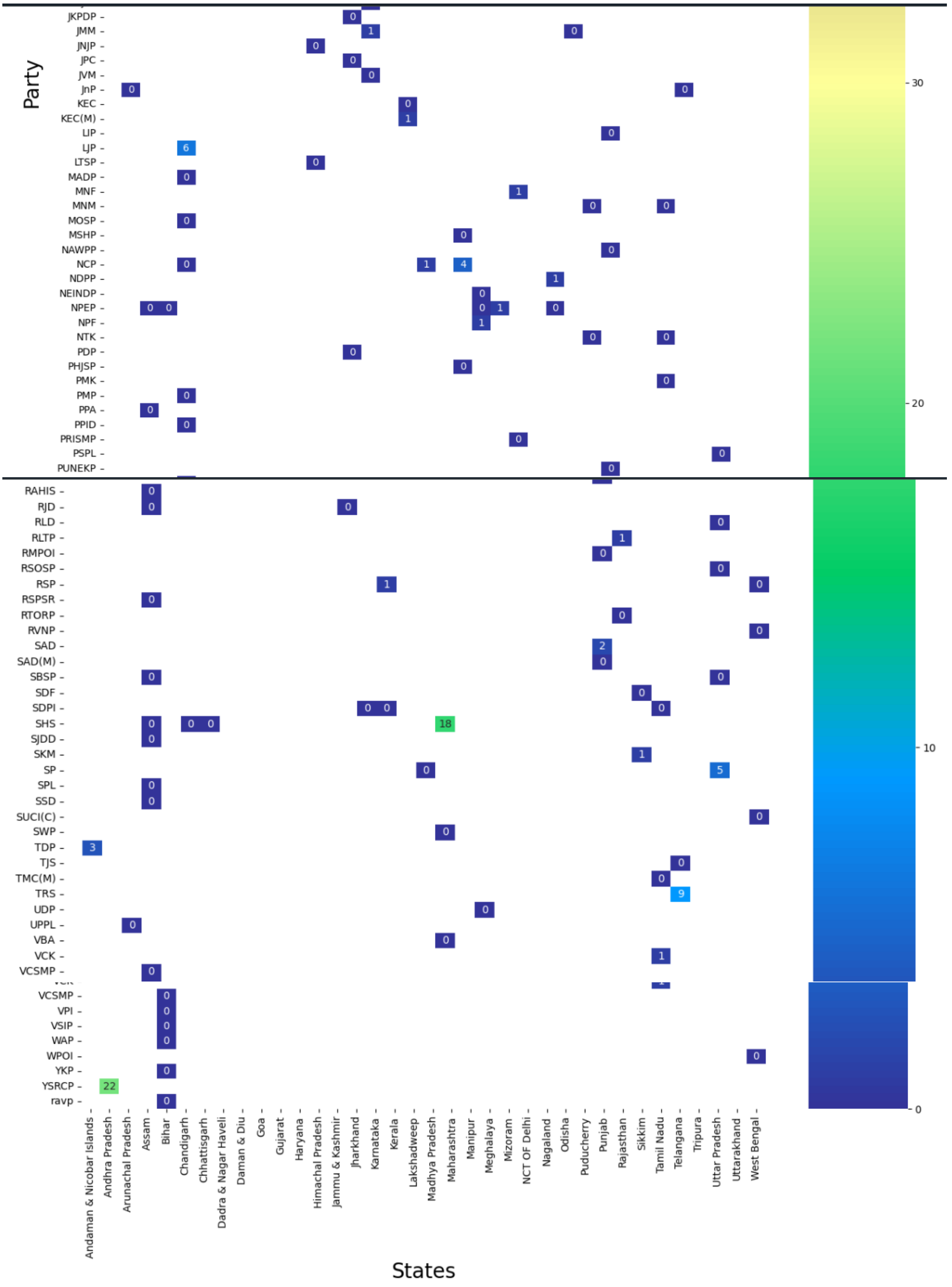
	STATE	CONSTITUENCY	NAME	WINNER	PARTY	SYMBOL	GENDER	CRIMINAL CASES	AGE	CATEGORY	EDUCATION	ASSETS	LIABILITIES	GENERAL VOTES	POSTAL VOTES	TOTAL VOTES	OVER TOTAL ELECTORS IN CONSTITUENCY	OVER TOTAL VOTES POLLED IN CONSTITUENCY	TOTAL ELECTORS
0	Telangana	ADILABAD	SOYAM BAPU RAO	1	BJP	Lotus	MALE	52	52.0	ST	12th Pass	3099414	231450	376892	482	377374	25.330684	35.468248	1489790
1	Telangana	ADILABAD	Godam Nagesh	0	TRS	Car	MALE	0	54.0	ST	Post Graduate	18477888	847000	318665	149	318814	21.399929	29.964370	1489790
2	Telangana	ADILABAD	RATHOD RAMESH	0	INC	Hand	MALE	3	52.0	ST	12th Pass	36491000	15300000	314057	181	314238	21.092771	29.534285	1489790
3	Telangana	ADILABAD	NOTA	0	NOTA	NaN	NaN	0	NaN	NaN	NaN	0	0	13030	6	13036	0.875023	1.225214	1489790
4	Uttar Pradesh	AGRA	Satyapal Singh Baghel	1	BJP	Lotus	MALE	5	58.0	SC	Doctorate	74274036	8606522	644459	2416	646875	33.383823	56.464615	1937690

```
state_const = election_df.groupby(['STATE']).apply(lambda
x:x['CONSTITUENCY'].nunique()).reset_index(name='# Constituency')
#state_shp = gpd.read_file('/kaggle/input/india-states/Igismap/Indian_States.shp')
#shp_state_joined = state_shp.set_index('st_nm').join(state_const.set_index('STATE'))

#fig, ax = plt.subplots(1, figsize=(10, 10))
#ax.axis('off')
#ax.set_title('State-wise Distribution of Indian Constituencies',
#             fontdict={'fontsize': '15', 'fontweight' : '3'})
#fig = shp_state_joined.plot(column='# Constituency', cmap='inferno_r',linewidth=0.5, ax=ax,
edgecolor='0.2',legend=True)

state_const.sort_values(by='# Constituency',ascending=False,inplace=True)
fig2 = px.bar(state_const, x='STATE', y='# Constituency',
                     color='# Constituency',
             labels={'pop':'Constituencies of India'})
```

```
fig2.update_layout(title_text='Statewise distribution of the Constituencies all over
India',template='plotly_white')
fig2.show()

# Percentage of Voting by State

df_votes_perct_constituency = election_df.groupby(['STATE','CONSTITUENCY','TOTAL
ELECTORS'])['TOTAL VOTES'].sum().reset_index()
df_votes_perct_constituency['% VOTED IN CONSTITUENCY'] =
round(df_votes_perct_constituency['TOTAL VOTES']*100/df_votes_perct_constituency['TOTAL
ELECTORS'],2)
df_voters_state = election_df[['STATE','CONSTITUENCY','TOTAL ELECTORS']].drop_duplicates()
df_voters_state = df_voters_state.groupby('STATE')['TOTAL ELECTORS'].sum().reset_index()
df_votes_state = election_df.groupby('STATE')['TOTAL
VOTES'].sum().reset_index().sort_values('TOTAL VOTES',ascending = False)
df_votes_perct_state = pd.merge(df_votes_state,df_voters_state, on ='STATE',how = 'left')
df_votes_perct_state['% VOTED IN STATE'] = round(df_votes_perct_state['TOTAL
VOTES']*100/df_votes_perct_state['TOTAL ELECTORS'],2)
df_votes_perct_state = df_votes_perct_state.sort_values('% VOTED IN STATE',ascending = False)
fig = px.bar(df_votes_perct_state, x='STATE', y='% VOTED IN STATE', color='% VOTED IN STATE',
height=600, width=800)
fig.update_layout(title_text='Statewise distribution of voting percentage all over
India',template='plotly_dark')
fig.show()

# Which Parties have been present in most constituencies and States?

vote_prty=election_df[election_df['PARTY']!='NOTA']
prty_cnt=vote_prty.groupby('PARTY').apply(lambda
x:x['CONSTITUENCY'].count()).reset_index(name='# Constituency')
prty_st=vote_prty.groupby('PARTY').apply(lambda x:x['STATE'].nunique()).reset_index(name='# State')
prty_cnt.sort_values(by='# Constituency',ascending=False,inplace=True)
prty_top_cn=prty_cnt[:20]
prty_top_all=pd.merge(prty_top_cn,prty_st,how='inner',left_on='PARTY',right_on='PARTY')
#fig = px.scatter(prty_top_all, x='# Constituency', y='# State', color='# State',
#                size='# Constituency', hover_data=['PARTY'])
fig = px.bar(prty_top_all, x='PARTY', y='# Constituency',
                 color='# State',title='Win Counts by a Political Party in 2019')
fig.update_layout(title_text='Constituency vs Statewise participation for the most contesting Political
Parties',template='ggplot2')
fig.show()

#What has been the performance of the Parties Statewise?

st_prty=vote_prty.groupby(['PARTY','STATE']).apply(lambda
x:x['WINNER'].sum()).reset_index(name='Wins')
pvt_st_prty=pd.pivot(st_prty,index='PARTY',columns='STATE',values='Wins')
plt.figure(figsize=(15,35))
sns.heatmap(pvt_st_prty,annot=True,fmt='g',cmap='terrain')
```

```
#fig = px.imshow(pvt_st_prty, text_auto=True, aspect="auto")
#fig.show()
plt.xlabel('States',size=20)
plt.ylabel('Party',size=20)
plt.title('Statewise report card for the Political Parties in India',size=25)
```





```
# Which party has won the most constituencies?

part_win=election_df.groupby('PARTY').apply(lambda x:x['WINNER'].sum()).reset_index(name='#
Wins')
part_win.sort_values(by='# Wins',ascending=False,inplace=True)
```

```
top_part_win=part_win[0:15]
fig = px.bar(top_part_win, x='PARTY', y='# Wins',
             color='# Wins',title='Win Counts by a Political Party in 2019')
fig.update_layout(title_text='Win Counts by a Political Party in 2019',template='plotly_white')
fig.show()

# Party Wise vote share

vote_share_top5 = election_df.groupby('PARTY')['TOTAL VOTES'].sum().nlargest(10).index.tolist()


def vote_share(row):
    if row['PARTY'] not in vote_share_top5:
        return 'Other'
    else:
        return row['PARTY']
election_df['Party New'] = election_df.apply(vote_share,axis =1)
counts = election_df.groupby('Party New')['TOTAL VOTES'].sum()
labels = counts.index
values = counts.values
pie = go.Pie(labels=labels, values=values, marker=dict(line=dict(color='#000000', width=1)))
layout = go.Layout(title='Partywise Vote Share')
fig = go.Figure(data=[pie], layout=layout)
fig.show()

# What has been the general Win vs Loss relationship for the Parties in 2019?

prty_cnt_win=pd.merge(prty_cnt,part_win,how='inner',left_on='PARTY',right_on='PARTY')
prty_cnt_win['Lost']=prty_cnt_win['# Constituency']-prty_cnt_win['# Wins']
prty_wins_cnt=prty_cnt_win[['PARTY','# Wins']]
prty_wins_cnt['Verdict']='Constituency Won'
prty_loss_cnt=prty_cnt_win[['PARTY','Lost']]
prty_loss_cnt['Verdict']='Constituency Lost'
prty_wins_cnt.columns=['Party','Counts','Verdict']
prty_loss_cnt.columns=['Party','Counts','Verdict']
top_prty_wins_cnt=prty_wins_cnt[:15]
prty_loss_cnt_cnt=prty_loss_cnt[:15]
prt_win_loss=pd.concat([top_prty_wins_cnt,prty_loss_cnt_cnt])
fig = px.bar(prt_win_loss, x='Party', y='Counts', color='Verdict')
fig.update_layout(title_text='Win vs Loss Analysis for the Top Parties',template='plotly_dark')
fig.show()
```

```
C:\Users\Asus\AppData\Local\Temp\ipykernel_32820\396968807.py:6: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
C:\Users\Asus\AppData\Local\Temp\ipykernel_32820\396968807.py:8: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

# What is the Gender Ratio of the Contestants? Also the Gender Ratio of the Winners?

```
vote_gndr=election_df[election_df['PARTY']!='NOTA']
gndr_overall=vote_gndr.groupby('GENDER').apply(lambda
x:x['NAME'].count()).reset_index(name='Counts')
gndr_overall['Category']='Overall Gender Ratio'
winners=vote_gndr[vote_gndr['WINNER']==1]
gndr_winner=winners.groupby('GENDER').apply(lambda
x:x['NAME'].count()).reset_index(name='Counts')
gndr_winner['Category']='Winning Gender Ratio'
gndr_overl_win=pd.concat([gndr_winner,gndr_overall])
fig = px.bar(gndr_overl_win, x='GENDER', y='Counts',
             color='Category', barmode='group')
fig.update_layout(title_text='Participation vs Win Counts analysis for the Genders',template='seaborn')
fig.show()
```

# What is the Educational Qualification of our politicians?

```
ed_valid=election_df[election_df['PARTY']!="NOTA"]
ed_cnt=ed_valid.groupby('EDUCATION').apply(lambda
x:x['PARTY'].count()).reset_index(name='Counts')
fig = go.Figure(data=[go.Pie(labels=ed_cnt['EDUCATION'], values=ed_cnt['Counts'], pull=[0.1, 0.2, 0,
0.1, 0.2, 0,0.1, 0.2, 0,0.1, 0.2, 0.1])])
fig.update_layout(title_text='Overall Education Qualification of all the Nominees',template='plotly_white')
fig.show()
ed_won=ed_valid[ed_valid['WINNER']==1]
ed_win_cnt=ed_won.groupby('EDUCATION').apply(lambda
x:x['PARTY'].count()).reset_index(name='Counts')
fig2 = go.Figure(data=[go.Pie(labels=ed_win_cnt['EDUCATION'], values=ed_win_cnt['Counts'],
pull=[0.1, 0.2, 0, 0.1, 0.2, 0,0.1, 0.1, 0.2,0, 0.1, 0.2],title='Education Qualification of the Winners')])
fig2.update_layout(title_text='Education Qualification of the Winners',template='plotly_white')
fig2.show()
```

# What is the relationship of Age and Politics?

```
age_cnt=ed_valid.groupby(['AGE','GENDER']).apply(lambda
x:x['NAME'].count()).reset_index(name='Counts')
fig = px.histogram(age_cnt, x="AGE",y='Counts',color='GENDER',title='Age Counts Distribution among
the politicians')
fig.update_layout(title_text='Age Counts Distribution among the politicians',template='plotly_white')
fig.show()
```

```
# What relation does the Politician category have with the election results?


vote_cat=election_df[election_df['PARTY']!='NOTA']
cat_overall=vote_cat.groupby('CATEGORY').apply(lambda
x:x['NAME'].count()).reset_index(name='Counts')
cat_overall['Category']='Overall Category Counts'
winners_cat=vote_gndr[vote_gndr['WINNER']==1]
cat_winner=winners_cat.groupby('CATEGORY').apply(lambda
x:x['NAME'].count()).reset_index(name='Counts')
cat_winner['Category']='Winning Category Ratio'
cat_overl_win=pd.concat([cat_winner,cat_overall])
fig = px.bar(cat_overl_win, x='CATEGORY', y='Counts',
             color='Category', barmode='group')
fig.update_layout(title_text='Participation vs Win Counts for the Category in
Politics',template='plotly_white')
fig.show()

# Have the politicians been involved with criminal activities?

crim_cnt=ed_valid.groupby('CRIMINAL CASES').apply(lambda
x:x['NAME'].count()).reset_index(name='Counts')
fig = px.histogram(crim_cnt, x='CRIMINAL CASES',y='Counts',marginal='violin')
fig.update_layout(title_text='Criminal Cases Counts Distribution among the
politicians',template='plotly_white')
fig.show()

# Candidate and Winner Assets

#Candidates
df_assets = pd.read_csv('LS_2.0.csv')
df_assets[['ASSETS2','ASSETS_VALUE']] = df_assets['ASSETS'].str.split('~',expand=True)
df_assets.drop(['ASSETS2'],axis =1,inplace=True)
df_assets = df_assets[df_assets['ASSETS_VALUE'].notnull()]
def asset_range(row):
    if row['ASSETS_VALUE'].endswith('Crore+'):
        return 'Crore+'
    elif row['ASSETS_VALUE'].endswith('Lacs+'):
        return 'Lakh+'
    elif row['ASSETS_VALUE'].endswith('Thou+'):
        return 'Thousand+'
    else:
        return 'NAN'

df_assets['ASSETS_RANGE'] = df_assets.apply(asset_range,axis =1)
df_assets['COUNT'] = 1
df_assets = df_assets[df_assets['ASSETS_RANGE'] != 'NAN']
counts = df_assets.groupby('ASSETS_RANGE')['COUNT'].sum()
```

```
labels = counts.index
values = counts.values
pie = go.Pie(labels=labels, values=values, marker=dict(line=dict(color='#000000', width=1)))
layout = go.Layout(title='Assests of Candidates')
fig = go.Figure(data=[pie], layout=layout)
fig.show()



# Winners
df_assets = pd.read_csv('LS_2.0.csv')
df_assets[['ASSETS2','ASSETS_VALUE']] = df_assets['ASSETS'].str.split('~',expand=True)
df_assets.drop(['ASSETS2'],axis =1,inplace=True)
df_assets = df_assets[df_assets['ASSETS_VALUE'].notnull()]
def asset_range(row):
    if row['ASSETS_VALUE'].endswith('Crore+'):
        return 'Crore+'
    elif row['ASSETS_VALUE'].endswith('Lacs+'):
        return 'Lakh+'
    elif row['ASSETS_VALUE'].endswith('Thou+'):
        return 'Thousand+'
    else:
        return 'NAN'

df_assets['ASSETS_RANGE'] = df_assets.apply(asset_range,axis =1)
df_assets['COUNT'] = 1
df_assets = df_assets[df_assets['ASSETS_RANGE'] != 'NAN']
df_assets = df_assets[df_assets['WINNER'] == 1]
counts = df_assets.groupby('ASSETS_RANGE')['COUNT'].sum()
labels = counts.index
values = counts.values
pie = go.Pie(labels=labels, values=values, marker=dict(line=dict(color='#000000', width=1)))
layout = go.Layout(title='Assests of Winners')
fig = go.Figure(data=[pie], layout=layout)
fig.show()

# Category wise Candidates
df_category = election_df['CATEGORY'].value_counts().reset_index()
df_category.columns = ['CATEGORY','COUNT']
fig = px.bar(df_category, x='CATEGORY', y='COUNT', color='CATEGORY', height=500,
title="Category wise Candidates")
fig.show()

# Category wise winners
df_gender_won =election_df[election_df['WINNER'] == 1]
df_category = df_gender_won['CATEGORY'].value_counts().reset_index()
df_category.columns = ['CATEGORY','COUNT']
fig = px.bar(df_category, x='CATEGORY', y='COUNT', color='CATEGORY', height=500,
title="Category wise winners")
fig.show()
```

CONCLUSION

The election analysis website project developed in Python provides a comprehensive platform for understanding and predicting statewise candidates' performance based on collected election data. Leveraging classification and regression techniques, the system offers valuable insights into electoral outcomes, aiding stakeholders in making informed decisions. Through meticulous data collection, cleaning, and feature engineering, the platform ensures the reliability and accuracy of predictions.

By employing classification models, it accurately forecasts winning candidates in each constituency/state, while regression models provide estimates of vote shares. The website's user-friendly interface facilitates easy input of data and access to predictions, enhancing usability for diverse stakeholders, including politicians, policymakers, and the general public. With a focus on scalability, security, and performance, the system is designed for seamless deployment and maintenance.

Regular monitoring and updates ensure the continued relevance and effectiveness of the models, thereby empowering users with reliable tools for electoral analysis. In essence, the election analysis website project serves as a valuable resource for understanding electoral dynamics, fostering transparency, and facilitating evidence-based decision-making in the democratic process.

REFERENCES

[1] Stephanie Tawa Lama, “Studying Elections in India: Scientific and Political Debates”, November 2009, South Asia Multidisciplinary Academic Journal

[2] https://resources.infosecinstitute.com/category/enterprise /phishing/the-phishing landscape/phishing-data-attack-statistics/#gref

[3] I.Vasudevan, A.P.V.Raghavendra, “Election Analysis and Pedagogy Forecast Using Big Data Analytics”, 2007, IJEDR

[4] Mohammad R., Thabtah F. McCluskey L., (2015) Phishing websites dataset. Available: https://archive.ics.uci.edu/ml/datasets/Phishing+Websites Accessed January 2016

[5] http://dataaspirant.com/2017/01/30/how-decision-tree- algorithm-works/

[6] http://dataaspirant.com/2017/05/22/random-forest- algorithm-machine-learing/

[7] https://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html

[8] www.Virustoolapi.com

[9] www.phishtank.com

[10] www.googlesafebrowesring.com