Employability Prediction Based on Personality Traits and Cognitive Analysis obtained from Social Media Profiles

Submitted by

Pinaki Banerjee (13000121003)

Anuvab Chakravarty (13000121036)

Debanjan Konar (13000121006)

Soumyajit Dey Sarkar (13000221080)

Group 26

Final Year 7th Semester

October, 2025

Submitted for the partial fulfillment for the degree of
Bachelor of Technology in
Computer Science and Engineering

Techno Main Salt Lake,
EM 4/1, Salt lake, Sector - V, Kolkata - 700091

Department of Computer Science and Engineering
Techno Main Salt Lake
Kolkata - 700 091
West Bengal, India

APPROVAL

This is to certify that the project entitled **"Employability Prediction Based on Personality Traits and Cognitive Analysis obtained from Social Media Profiles"** prepared by **Pinaki Banerjee** *(13000121003)*, **Debanjan Konar** *(13000121006)*, **Anuvab Chakravarty** *(13000121036)* and **Soumyajit Dey Sarkar** *(13000221080)* be accepted in partial fulfillment for the degree of Bachelor of Technology in Computer Science and Engineering.

It is to be understood that by this approval, the undersigned does not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn thereof, but approves the report only for the purpose for which it has been submitted.

. . . .
(Signature of the Internal Guide)

. . . .
(Signature of the HOD)

. . . .
(Signature of the External Examiner)

. . . .

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

. . . .

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our project guide in the department of Computer Science and Engineering. We are extremely thankful for the keen interest our guide took in advising us, for the books, reference materials and support extended to us.

Last but not the least we convey our gratitude to all the teachers for providing us the technical skill that will always remain as our asset and to all non-teaching staffs for the gracious hospitality they offered us.

Place: Techno Main Salt Lake
Date: 06.10.2024

Pinaki Banerjee (13000121003)

Debanjan Konar (13000121006)

Anuvab Chakravarty (13000121036)

Soumyajit Dey Sarkar (13000221080)

Table of Content

Abstract

This project aims to simplify the hiring process by developing a Software-as-a-Service (SaaS) tool that leverages social media profiles to analyze employability. The tool will use machine learning models to classify personality traits (based on Big 5 and MBTI frameworks) and predict an individual's suitability for specific job roles. By automating the analysis of social media activity, this solution addresses inefficiencies in traditional hiring methods, providing a detailed hiring metric based on personality and cognitive abilities. Ultimately, this tool enhances decision-making for employers and career counselors while reducing manual labor in early-stage hiring.

1 Introduction

1.1 Project Overview

This project addresses inefficiencies in the current hiring process by introducing a prediction tool that automates employability analysis using machine learning and social media data. It falls under the web development domain, with a focus on building a SaaS-based prediction platform. The tool will primarily rely on cloud-hosted software, machine learning algorithms, and web automation to analyze personality traits and cognitive abilities.

1.2 Project Purpose

The primary objective of this project is to develop a Software-as-a-Service (SaaS) tool that utilizes machine learning to predict employability based on personality traits derived from social media profiles. By analyzing user activities and posts, the tool can classify individuals' personality traits using models like the Big 5 and Myers-Briggs Type Indicator (MBTI), mapping these traits to specific job roles.
The project aims to address inefficiencies in traditional hiring processes by providing early-stage personality and cognitive ability insights. The system incorporates classification models such as Logistic Regression for accurate trait classification. This approach enables predictive analytics for employers and career counselors, offering data-driven insights into a candidate's suitability for various roles.
Additionally, the project involves overcoming challenges in large-scale data processing, integrating multiple social media APIs, and optimizing classification algorithms to ensure robust, reliable predictions.

1.3 Technical Domain Specifications

The proposed solution operates at the intersection of Web Development and Machine Learning, as it aims to create a comprehensive, user-friendly software tool designed for employers to streamline the employability analysis process. By leveraging social media profile links of individuals, the tool will enable quick and efficient assessment of candidates. The core functionality lies in automating personality and cognitive ability evaluations based on social media activity.
The system will employ advanced machine learning algorithms to generate a detailed hiring metric. This metric will be grounded in established personality models such as the Big 5 and MBTI frameworks, allowing for precise classification of individuals into personality types. These classifications will then be correlated with job roles, enabling the tool to predict an individual's suitability for specific positions based on their inherent traits and behavioral patterns.

- **Hardware** : The project does not require specialized hardware beyond a standard machine with adequate processing power. The project can be run on any system with at least 8GB of RAM and a multi-core processor.
- **Operating System** : The project is cross-platform and can be developed and executed on any modern operating system, including:
  - Windows 10/11
  - MacOS
  - Linux distributions (Ubuntu, Kali, etc.)
- **Software** :
  - **Programming Languages** : Python 3.x will be the primary programming language for our project due to the presence of ML libraries and other data analysis tools.
  - **Libraries / Frameworks** :
    - **Scikit-learn** : Used for machine learning algorithms (RandomForestClassifier, LogisticRegression, SGDClassifier), data preprocessing, and evaluation metrics like accuracy score.
    - **XGBoost** : For implementing the XGBClassifier model, a powerful gradient-boosting algorithm for enhanced classification performance.
    - **Pandas** : Used for data manipulation and preprocessing tasks, including handling datasets and preparing them for machine learning models.
    - **NumPy** : To handle large arrays and matrices for efficient numerical computations, particularly in preprocessing and model input preparation.

    - **NLTK** : For natural language processing tasks like stopword removal and word lemmatization, enabling effective text preprocessing.
    - **Matplotlib and Seaborn** : For data visualization, including plotting graphs and visualizing model performance and dataset characteristics.
    - **Pickle** : Used for saving and loading machine learning models, allowing for efficient reuse of trained models in future processes.
    - **Warnings** : Used for suppressing warning messages during model training and evaluation, ensuring cleaner output during experimentation.
  - **Development Environment** :
    - **Jupyter Notebook** : For interactive development, experimentation, and visualization.
    - **Google Colab** : For cloud-based execution when working with larger datasets or GPU-based model training.
    - **VS Code** : An integrated development environment developed by Microsoft for Windows, Linux, macOS and web browsers.

1.4 Business Domain Specifications

From a business standpoint, this project offers significant potential across a variety of industries, particularly those involved in human resources, talent acquisition, and digital profiling. The growing reliance on data-driven decision-making in recruitment and employee management has led organizations to seek innovative solutions for evaluating candidates beyond traditional metrics like resumes and interviews. By integrating machine learning with social media analysis, this project provides a powerful tool for personality-driven employability analysis. Below is an exploration of the potential business impact across various domains:

- **Human Resource Management (HRM)**: Human Resource departments across industries are constantly searching for ways to enhance the recruitment process and improve the accuracy of hiring decisions. This project introduces a machine learning-driven solution that provides HR professionals with detailed personality and cognitive assessments of candidates, derived from their social media activity. This method offers deeper insights into a candidate's potential fit for a specific role, allowing HR teams to make more informed decisions about hiring, employee engagement, and career development. The system's ability to predict an individual's alignment with job roles based on their traits ensures a more streamlined and efficient hiring process.

- **Recruitment and Talent Acquisition**: Recruitment agencies and talent acquisition specialists can leverage this tool to enhance candidate profiling and reduce the time spent on the early stages of hiring. By automating the initial candidate screening process and delivering personality-based hiring metrics, recruitment professionals can quickly assess a candidate's employability and role fit. This solution mitigates the limitations of traditional CV-based assessments and allows recruiters to focus on more targeted interviews with candidates who demonstrate strong predictive suitability for the required positions. Furthermore, the solution can be integrated into existing Applicant Tracking Systems (ATS) for seamless recruitment workflows.

- **Digital Marketing and Customer Profiling**: Companies engaged in digital marketing and customer profiling can benefit from this tool by utilizing it to assess personality traits for tailored advertising and customer engagement strategies. By analyzing personality traits from social media profiles, businesses can create more personalized marketing campaigns that resonate with specific personality types. This deeper understanding of consumer behavior allows brands to enhance their customer segmentation, improving conversion rates and building long-term relationships with customers.

- **Career Counseling and Educational Institutions**: Career counseling services and educational institutions can adopt this tool to guide students and job seekers in identifying roles that align with their personality and cognitive strengths. By mapping an individual's personality traits to suitable career paths, the tool provides valuable insights into how individuals can best utilize their strengths. Career counselors can offer more personalized guidance, helping individuals select career paths where they are most likely to succeed and thrive.

1.5 Glossary / Keywords

Term	Definition
Software-as-a-Service (SaaS)	A cloud-based software delivery model where applications are hosted by a service provider and made available to customers over the internet.
Machine Learning (ML)	A subset of artificial intelligence that focuses on training algorithms to learn patterns from data and make decisions without explicit programming.
Personality Traits	Enduring characteristics and patterns of behavior, thought, and emotion, used in models like Big 5 and MBTI to describe individuals.
Big 5 Model	A widely accepted framework in psychology that includes five personality traits: Openness, Conscientiousness, Extraversion, Agreeableness, and Neuroticism (OCEAN).

Term	Definition
Myers-Briggs Type Indicator (MBTI)	A personality classification system that categorizes individuals into 16 different personality types based on four dichotomies: Extraversion vs. Introversion, Sensing vs. Intuition, Thinking vs. Feeling, and Judging vs. Perceiving.
Logistic Regression	A statistical model used in machine learning for binary or multi-class classification tasks, predicting the probability of a categorical outcome.
Cognitive Abilities	Mental skills like reasoning, memory, problem-solving, and planning, which are often linked to an individual's ability to perform tasks or jobs effectively.
Social Media APIs	Application Programming Interfaces provided by social media platforms (such as Facebook, LinkedIn, Twitter) that allow access to user data, posts, and activity for analysis.
Data Preprocessing	The act of cleaning, organizing, and transforming raw data into a structured format suitable for machine learning models. This may include tasks like tokenization, normalization, and vectorization.
Employability Analysis	The process of assessing a person's suitability for employment based on various factors such as skills, personality traits, and cognitive abilities.
Personality Classification	The use of algorithms and models to categorize individuals based on personality traits derived from data such as social media posts.
Predictive Analytics	A branch of analytics that uses historical data and machine learning algorithms to predict future outcomes, in this case, predicting job suitability based on personality traits.
Cross-Validation	A model validation technique used to evaluate how a machine learning model performs on different subsets of a dataset by training and testing on multiple folds of the data.
Classification Model	A machine learning model that predicts which category (class) a given input belongs to, such as job role suitability based on personality traits.
Applicant Tracking System (ATS)	Software used by employers and recruiters to manage job applications and streamline the recruitment process. Integration with the tool could enhance candidate screening through personality insights.
Role Fit	The extent to which an individual's personality traits and cognitive abilities align with the requirements of a specific job role.

2 Related Studies

In recent years, the prediction of personality traits from social media posts has become a critical area of research, largely due to the rapid rise of digital content creation and sharing across platforms. This research often utilizes established personality classification schemes such as the Big 5 and the Myers-Briggs Type Indicator (MBTI) to infer users' personalities based on their online behavior [1][2][3]. Numerous studies, such as those by Bakry et al. [1], Souri et al. [2], and Hernandez and Knight [3], have demonstrated how data from social media profiles, including posts, likes, and comments, can be analyzed through machine learning models to predict personality traits with remarkable accuracy. The Big 5 model, which classifies personality into five dimensions—Openness, Conscientiousness, Extraversion, Agreeableness, and Neuroticism—has been widely recognized as one of the most reliable frameworks for this purpose [1][2]. Similarly, the MBTI, which divides personality into 16 types based on four dichotomies (Extraversion vs. Introversion, Sensing vs. Intuition, Thinking vs. Feeling, Judging vs. Perceiving), has been successfully applied to social media data to map users into personality types [3].

The significance of this research becomes more apparent when considering the strong correlations between personality traits and job engagement [4][5]. Studies have found that personality plays a critical role in influencing an individual's level of engagement in their work, impacting factors such as productivity, satisfaction, and long-term career success [4][5]. For example, Öngöre [4] argues that personality traits such as Conscientiousness and Extraversion are positively correlated with job engagement, while lower levels of Agreeableness and Neuroticism may hinder workplace performance. Additionally, research by Nwogu and Momoh [5] suggests that understanding personality preferences is vital for developing employability qualities that align with the evolving demands of the job market. As the world of work continues to change, these studies emphasize that personality traits must be considered when determining employability qualities, particularly in job roles that require high levels of adaptability and cognitive ability [4][5].

Moreover, this intersection of personality and employability is further underscored by the increasing use of personality prediction models in employment settings, particularly for recruitment and role fit assessments [5]. The research by Nwogu [5] has shown that personality traits are significant predictors of job performance, especially when employers seek to match candidates with job roles that align with their psychological profiles. In such a context, personality classification models like those described by Bakry et al. [1], Souri et al. [2], and Hernandez and Knight [3] could play a pivotal role in future recruitment strategies, enabling employers to assess potential candidates based on their digital footprints, thus improving hiring decisions. The integration of personality prediction tools into Applicant Tracking Systems (ATS) could lead to a more data-driven, precise selection process, enhancing both job fit and organizational efficiency [1][2][3][5]. Furthermore, this approach ties into the broader field of predictive analytics, where personality traits can be used alongside other behavioral data to forecast job performance and organizational success [1][2][3][4][5].

In conclusion, the convergence of personality analysis, social media data, and machine learning opens up new possibilities for understanding employability in a digitally-driven world. Research from multiple studies underscores the importance of personality traits not only in personal development but also as key determinants of job engagement and long-term employability [1][2][3][4][5]. By leveraging tools and models derived from this research, organizations can better align their recruitment processes with the psychological profiles of potential employees, fostering a more engaged, productive workforce [1][2][3][4][5].

3 Problem Definition and Preliminaries

The hiring process within organizations, regardless of their size, remains a significant topic of debate and discussion. Accurately understanding "employability," or what constitutes an individual's "employability," presents a daunting challenge. Despite advancements in technology, including online resumes, profile links, virtual interviews, and take-home assignments, the hiring mechanism continues to exhibit fundamental inefficiencies and is susceptible to severe errors in judgment.

3.1 Problem Statement

This project aims to tackle these inefficiencies by introducing a prediction tool that aids in assessing employability. The tool is designed to reduce the search space for potential candidates, streamline the hiring process, and improve the overall quality of hires by focusing on specific traits relevant to job roles.

3.2 Project Boundaries

3.2.1 Scope

- The project encompasses the development of a predictive model that analyzes personality traits and cognitive abilities based on social media data and other relevant metrics.
- It includes the creation of a comprehensive employability assessment tool that career counselors, employers, and students can utilize for informed decision-making regarding career choices and hiring processes.
- The model will provide tailored recommendations for various job roles and skill sets, enhancing the specificity of hires and addressing diverse hiring use cases.

3.2.2 Exclusions

- The project does not cover the entire hiring process, such as the final selection stages, negotiation of job offers, or onboarding processes.
- It does not address external factors affecting employment, such as economic conditions or industry-specific trends.
- The model's effectiveness in real-world scenarios will require further validation beyond this initial implementation phase.

3.3 Research Methods

To conduct this research and complete the project, a combination of methods and techniques was employed:

- **Data Collection and Preprocessing:** Data was gathered from various social media platforms using APIs, focusing on user activity, posts, and interactions to derive personality traits.
- **Machine Learning Models:** The project utilized several machine learning libraries, including Scikit-learn and XGBoost, to build and evaluate classification models. Algorithms such as Random Forest Classifier, Logistic Regression, and Support Vector Machines (SVM) were implemented to enhance the accuracy of predictions.
- **Natural Language Processing (NLP):** The NLTK and spaCy libraries were used for text preprocessing and natural language understanding, allowing the analysis of text data to extract relevant features.
- **Data Visualization:** Libraries like Matplotlib and Seaborn facilitated data visualization, helping to interpret results and communicate findings effectively.

By integrating these methodologies, the project aims to revolutionize the hiring process, ensuring that organizations can make better hiring decisions while reducing the manpower required for entry-level positions. Ultimately, this prediction tool is envisioned as a valuable resource for students, professionals, and employers alike, enhancing the alignment between candidates and organizational culture.

4 Proposed Solution

- **Classification of Personality Traits**: The process of determining employability and predicting cognitive levels will often involve identifying an individual's personality type. This project aims to classify personality traits using established models like the Big Five and the Myers-Briggs Type Indicator (MBTI). These models provide a framework for understanding the complexities of personality and how they relate to employability.
- **Social Media Analysis**: To classify these traits, social media posts will be analyzed. A significant portion of the population shares their thoughts, feelings, and behaviors on platforms like Facebook and Twitter. By utilizing natural language processing (NLP) techniques, we can extract valuable insights from these posts, which will serve as indicators of personality traits. An extension will be implemented to capture posts from a person's Facebook profile seamlessly while browsing.
- **Automation of Data Collection**: To streamline the post collection process, automation tools such as Selenium or Puppeteer will be employed. These tools will facilitate the automatic retrieval of posts from users' social media accounts, ensuring a continuous flow of data for analysis. This step will significantly reduce manual effort and enhance the efficiency of data collection.
- **Model Training and Prediction**: The collected posts will be processed and passed to a machine learning model trained on sample data to predict personality traits. A dictionary will be initialized to store trained models for various personality classifications. This approach will allow for quick retrieval and application of models, ensuring adaptability to different datasets and contexts.
- **Software-as-a-Service (SaaS) Implementation**: The proposed solution will be implemented as a Software-as-a-Service (SaaS) tool. This will provide employers with an easy and efficient way to conduct employability analyses using social media profile links of target individuals. The SaaS model will facilitate easy access and integration into existing recruitment processes, allowing employers to make data-driven hiring decisions.
- **Development of a Hiring Metric**: A detailed hiring metric based on personality traits and cognitive abilities will be devised. This metric will enable individuals to be mapped to specific job roles based on their predictive suitability. By considering both personality traits and cognitive skills, employers can better align candidates with the requirements of various positions, ultimately improving hiring outcomes and job satisfaction.

5 Project Planning

5.1 Software Life Cycle Model

We have chosen the iterative waterfall model for this project because of the many benefits that it offers. The iterative nature of the model allows for changes and adjustments to be made as the project progresses, which can be beneficial in addressing unexpected issues or requirements.

The project is divided into multiple distinct phases:

- **Requirement Gathering and Analysis** : This phase involves gathering requirements from stakeholders through interviews, surveys, and documentation reviews. The goal is to understand the functional and non-functional requirements, define the project's scope, and identify any technical or business constraints. This is crucial for aligning stakeholder expectations and laying the foundation for the design and development phases.It spanned approximately three weeks,at the end of which we had a clearer idea of project requirements.

- **Data Collection and Preparation** : Over the course a week, the data collection phase was carried out using a personality characteristics dataset from Kaggle. This process involved analyzing structure of the dataset, and conducting data cleaning and preprocessing to ensure it was ready for analysis.

- **Model Training** : Lasting over three weeks in this phase, a machine learning model was developed for personality classification using the MBTI (Myers-Briggs Type Indicator) dataset sourced from Kaggle. The dataset comprises textual data derived from users' online interactions, which were analyzed to identify patterns associated with different personality types. The data went extensive preprocessing, to convert raw text into a format suitable for machine learning algorithms.

  Various algorithms were tested, including Logistic Regression, Support Vector Machines, and Random Forests, with their performances evaluated based on accuracy, precision, and recall. The model was trained on a portion of the dataset, allowing it to learn the characteristics of different personality types. After optimization, the final model demonstrated a commendable accuracy rate, effectively classifying users' personality types based on their textual data.

- **Model Evaluation** : The machine learning model for personality classification was rigorously tested to ensure its robustness and reliability. Initially, the dataset was split into training and testing subsets, typically using a 70-30 ratio, allowing the model to learn from a diverse set of examples while reserving a portion for validation.

To further enhance the testing process, k-fold cross-validation was employed. This technique involved dividing the training dataset into k subsets, training the model k times, each time using a different subset as the validation set while the remaining k-1 subsets were used for training. This approach ensured that every data point was used for both training and validation, providing a comprehensive evaluation of the model's performance.

Performance metrics such as accuracy, precision, recall, and F1-score were calculated to assess the model's effectiveness in classifying personality types accurately. Additionally, confusion matrices were generated to visualize classification results, allowing for the identification of specific areas where the model performed well or needed improvement. By iterating through this testing phase, the model was fine-tuned, optimizing its parameters to achieve a higher level of accuracy and generalization for unseen data. This phase lasted for two weeks.

- **Final Deployment and Documentation** : Once a suitable model with acceptable accuracy is found, Comprehensive documentation will be created to support both us(developers) and end users. This will cover system architecture, model functionality, API endpoints, usage instructions, troubleshooting guidelines, and maintenance procedures. Proper documentation ensures that future developers can manage updates or fixes efficiently, and users can fully understand how to utilize the model's capabilities. This phase marks the transition from development to real-world application.

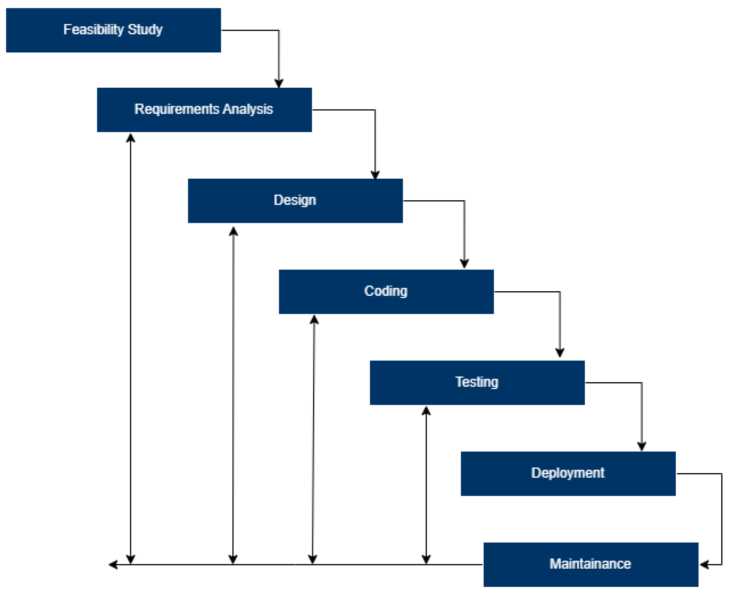


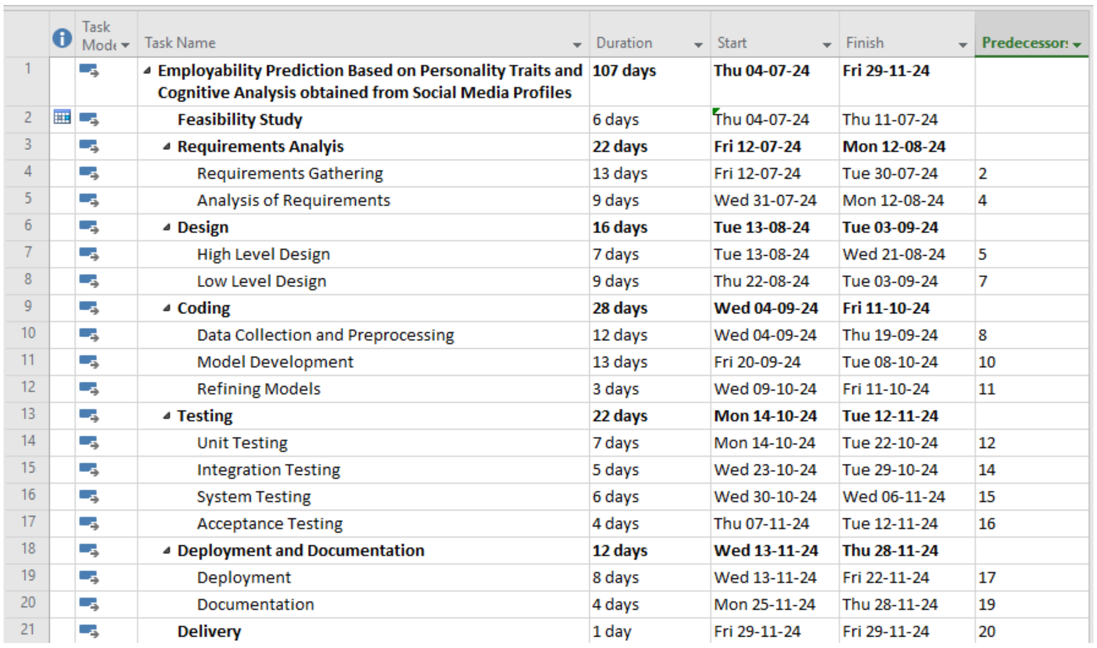
Figure 1: Iterative Waterfall Model

5.2 Dependencies

Critical dependencies were outlined to facilitate the project's smooth advancement. For instance, finding suitable dataset needed to happen before pre-processing and model training, data preparation needed to be finalized before moving on to model development. To ensure accountability and track progress, milestones were defined at the conclusion of each phase. The first key milestone was the completion of the requirement gathering phase, followed by data preparation, and continued through subsequent phases in an organized progression.

5.3 Timeline

Project timeline and tentative schedule has been provided below.



	Task Mode	Task Name	Duration	Start	Finish	Predecessors
1		**Employability Prediction Based on Personality Traits and Cognitive Analysis obtained from Social Media Profiles**	**107 days**	**Thu 04-07-24**	**Fri 29-11-24**	
2		**Feasibility Study**	6 days	Thu 04-07-24	Thu 11-07-24	
3		**Requirements Analysis**	**22 days**	**Fri 12-07-24**	**Mon 12-08-24**	
4		Requirements Gathering	13 days	Fri 12-07-24	Tue 30-07-24	2
5		Analysis of Requirements	9 days	Wed 31-07-24	Mon 12-08-24	4
6		**Design**	**16 days**	**Tue 13-08-24**	**Tue 03-09-24**	
7		High Level Design	7 days	Tue 13-08-24	Wed 21-08-24	5
8		Low Level Design	9 days	Thu 22-08-24	Tue 03-09-24	7
9		**Coding**	**28 days**	**Wed 04-09-24**	**Fri 11-10-24**	
10		Data Collection and Preprocessing	12 days	Wed 04-09-24	Thu 19-09-24	8
11		Model Development	13 days	Fri 20-09-24	Tue 08-10-24	10
12		Refining Models	3 days	Wed 09-10-24	Fri 11-10-24	11
13		**Testing**	**22 days**	**Mon 14-10-24**	**Tue 12-11-24**	
14		Unit Testing	7 days	Mon 14-10-24	Tue 22-10-24	12
15		Integration Testing	5 days	Wed 23-10-24	Tue 29-10-24	14
16		System Testing	6 days	Wed 30-10-24	Wed 06-11-24	15
17		Acceptance Testing	4 days	Thu 07-11-24	Tue 12-11-24	16
18		**Deployment and Documentation**	**12 days**	**Wed 13-11-24**	**Thu 28-11-24**	
19		Deployment	8 days	Wed 13-11-24	Fri 22-11-24	17
20		Documentation	4 days	Mon 25-11-24	Thu 28-11-24	19
21		**Delivery**	1 day	Fri 29-11-24	Fri 29-11-24	20

Figure 2: Project Plan

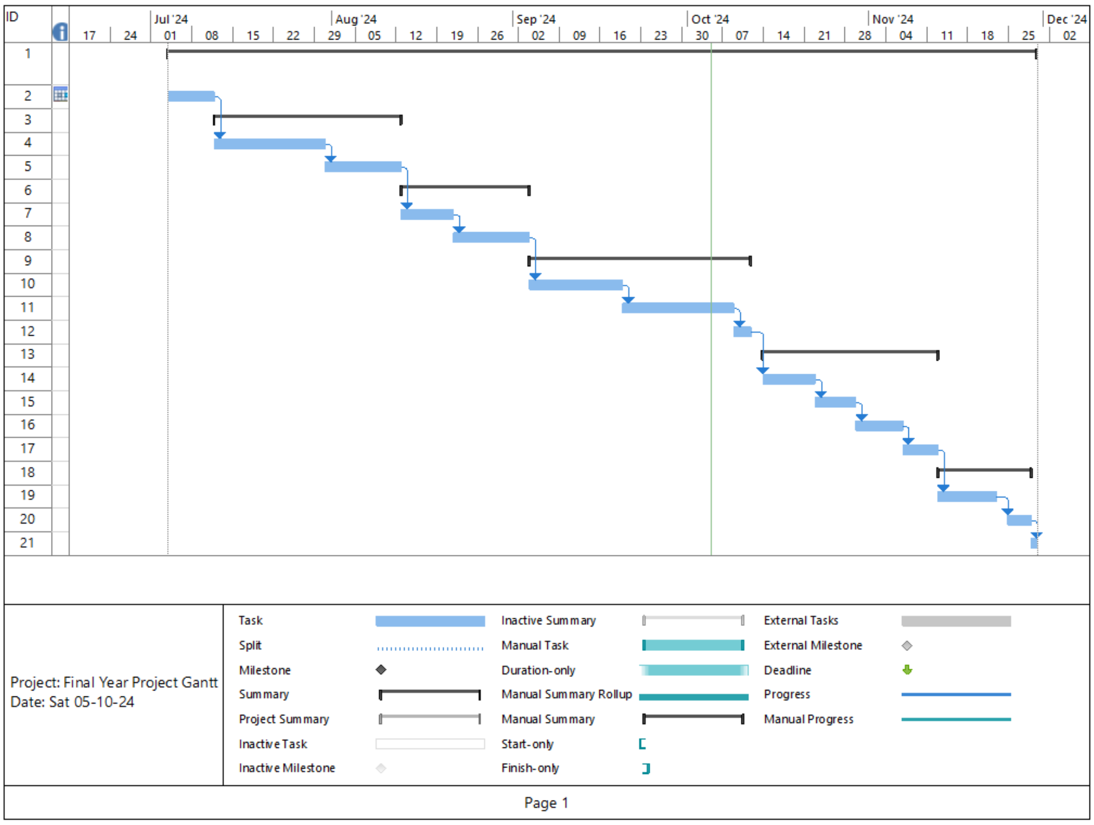
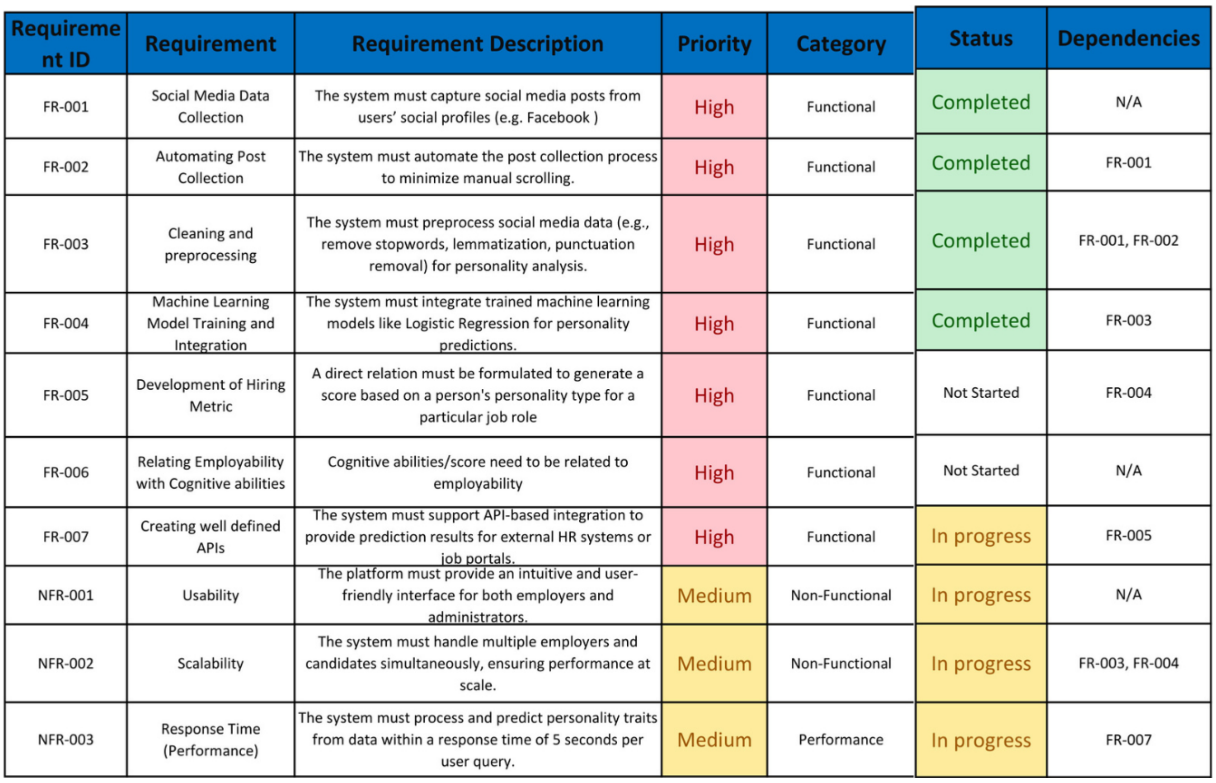


Figure 3: Gantt Chart

6 Requirement Analysis

6.1 Requirement Matrix

The Requirement Matrix is a comprehensive tool used to track and manage the key requirements of a project. It systematically organizes each requirement with a unique identifier, description, priority, and category (such as functional or non-functional). The matrix also records the source of the requirement, its current status (e.g., in progress, completed), any dependencies on other requirements, and the team or individual responsible for fulfilling it. Additionally, it includes a verification method to ensure the requirement is met, such as through testing or review. This structured format helps ensure that all requirements are clearly defined, prioritized, and tracked, enabling effective project management and ensuring alignment with stakeholder expectations.



Requirement ID	Requirement	Requirement Description	Priority	Category	Status	Dependencies
FR-001	Social Media Data Collection	The system must capture social media posts from users' social profiles (e.g. Facebook)	High	Functional	Completed	N/A
FR-002	Automating Post Collection	The system must automate the post collection process to minimize manual scrolling.	High	Functional	Completed	FR-001
FR-003	Cleaning and preprocessing	The system must preprocess social media data (e.g., remove stopwords, lemmatization, punctuation removal) for personality analysis.	High	Functional	Completed	FR-001, FR-002
FR-004	Machine Learning Model Training and Integration	The system must integrate trained machine learning models like Logistic Regression for personality predictions.	High	Functional	Completed	FR-003
FR-005	Development of Hiring Metric	A direct relation must be formulated to generate a score based on a person's personality type for a particular job role	High	Functional	Not Started	FR-004
FR-006	Relating Employability with Cognitive abilities	Cognitive abilities/score need to be related to employability	High	Functional	Not Started	N/A
FR-007	Creating well defined APIs	The system must support API-based integration to provide prediction results for external HR systems or job portals.	High	Functional	In progress	FR-005
NFR-001	Usability	The platform must provide an intuitive and user-friendly interface for both employers and administrators.	Medium	Non-Functional	In progress	N/A
NFR-002	Scalability	The system must handle multiple employers and candidates simultaneously, ensuring performance at scale.	Medium	Non-Functional	In progress	FR-003, FR-004
NFR-003	Response Time (Performance)	The system must process and predict personality traits from data within a response time of 5 seconds per user query.	Medium	Performance	In progress	FR-007

Figure 4: Requirement Matrix

6.2 Requirement Elaboration

6.2.1 Functional Requirements

Requirement ID: FR-001
Description: Social Media Data Collection
Priority: High
Category: Functional
The goal is to gather user-posted social media data, which serves as the foundational input for personality analysis for each user. This enables the system to access posts from platforms like Facebook for further processing.

Requirement ID: FR-002
Description: Automating Post Collection
Priority: High
Category: Functional
We aim to streamline the process of collecting social media data by automating it with browser automation tools. This removes manual effort, ensuring that data collection happens consistently, quickly, and without user intervention.

Requirement ID: FR-003
Description: Cleaning and Preprocessing
Priority: High
Category: Functional
This ensures that the raw social media data is transformed into a clean, standardized format. Data cleaning improves accuracy by removing noise (e.g., irrelevant words) and making the data usable for personality analysis models.

Requirement ID: FR-004
Description: Machine Learning Model Training and Integration
Priority: High
Category: Functional
The aim is to integrate trained machine learning models for personality predictions based on the processed data and well established personality classification systems like BIG5 and MBTI. This allows the system to make informed and automated decisions about one's personality type, improving the efficiency of the analysis process.

Requirement ID: FR-005
Description: Development of Hiring Metric
Priority: High
Category: Functional
We seek to establish a measurable hiring metric that links personality traits to job roles. This score-based system will help recruiters evaluate candidates based on their personality profiles and match them to suitable job roles.

Requirement ID: FR-006
Description: Relating Employability with Cognitive Abilities
Priority: High
Category: Functional
The goal is to connect cognitive abilities with employability, allowing the system to quantify how specific mental aptitudes impact a candidate's job suitability.

Requirement ID: FR-007
Description: Creating Well-Defined APIs
Priority: High
Category: Functional
The system aims to provide clear, well-documented APIs to facilitate integration with external

services or applications, ensuring smooth data exchange and system interoperability.

Requirement ID: NFR-001
Description: Usability
Priority: Medium
Category: Non-Functional
The system must provide an intuitive, user-friendly interface that makes navigation and interaction seamless for users. It should be accessible to individuals with varying levels of technical expertise, ensuring that key functions (like data input, analysis, and report generation) are easy to perform. Usability also includes adherence to accessibility standards.

Requirement ID: NFR-002
Description: Scalability
Priority: Medium
Category: Non-Functional
The system must be able to handle increasing volumes of data and a growing number of users without degradation in performance. As the project expands, the infrastructure should be capable of scaling both horizontally (adding more servers) and vertically (upgrading server capabilities) to accommodate higher loads, ensuring that the system can support future growth.

Requirement ID: NFR-003
Description: Response Time
Priority: Medium
Category: Non-Functional
The system must deliver results promptly, with minimal delay between data submission and output. Ideally, core tasks such as social media data processing, model predictions, and report generation should complete within seconds, providing users with near-instant feedback. Maintaining quick response times is crucial for user satisfaction and operational efficiency.

7 Design

7.1 Technical Environment

The technical environment for our project includes a blend of hardware, software, and tools designed to facilitate efficient data analysis, machine learning model training, and deployment. Below is a comprehensive overview of the minimum hardware requirements, software tools, and package specifications essential for the successful execution of this project.

Minimum Hardware Configuration
Given the nature of the project, which involves processing textual data and training machine learning models, the hardware requirements are modest but significant enough to ensure optimal performance. The minimum configuration needed is:

- **Processor** : Intel Core i5 (or equivalent) with a base clock speed of at least 2.5 GHz. A multi-core processor is preferred as it helps in parallel processing, which is essential during model training and data preprocessing steps.
- **RAM** : 8 GB of RAM is recommended to handle the operations of data loading, cleaning, and transformation. Large datasets, like those used in this project, may require more memory to prevent memory overflow errors and reduce delays during processing. For larger datasets, 16 GB of RAM would be ideal.
- **Storage** : At least 256 GB of SSD storage is recommended. Faster storage access significantly impacts loading time for datasets and dependencies. SSD is preferred over traditional HDD because of its faster read/write speeds, which benefit large datasets like the Reddit-based social media posts used in this project.
- **Graphics Processing Unit (GPU)** : For basic machine learning tasks like Logistic Regression or SVM, a dedicated GPU is not necessary. However, if deep learning models or more complex neural networks were introduced later, a GPU like NVIDIA GTX 1060 with 4 GB VRAM or higher would be advantageous.
- **Operating System** : Windows 10 (64-bit) or higher, macOS 10.13 (High Sierra) or higher, or any stable Linux distribution (e.g., Ubuntu 18.04 or higher). The operating system should support all necessary machine learning libraries and be compatible with the tools required for the project.

- **Programming Languages** : Python 3.x will be the primary programming language for our project due to the presence of ML libraries and other data analysis tools.
- **Libraries / Frameworks** :
  - **Scikit-learn** : Used for machine learning algorithms (RandomForestClassifier, LogisticRegression, SGDClassifier), data preprocessing, and evaluation metrics like accuracy score.
  - **XGBoost** : For implementing the XGBClassifier model, a powerful gradient-boosting algorithm for enhanced classification performance.

- **Pandas** : Used for data manipulation and preprocessing tasks, including handling datasets and preparing them for machine learning models.
- **NumPy** : To handle large arrays and matrices for efficient numerical computations, particularly in preprocessing and model input preparation.
- **NLTK** : For natural language processing tasks like stopword removal and word lemmatization, enabling effective text preprocessing.
- **Matplotlib and Seaborn** : For data visualization, including plotting graphs and visualizing model performance and dataset characteristics.
- **Pickle** : Used for saving and loading machine learning models, allowing for efficient reuse of trained models in future processes.
- **Warnings** : Used for suppressing warning messages during model training and evaluation, ensuring cleaner output during experimentation.

• **Development Environment** :

- **Jupyter Notebook** : For interactive development, experimentation, and visualization.
- **Google Colab** : For cloud-based execution when working with larger datasets or GPU-based model training.
- **VS Code** : An integrated development environment developed by Microsoft for Windows, Linux, macOS and web browsers.

7.2 Detailed Design

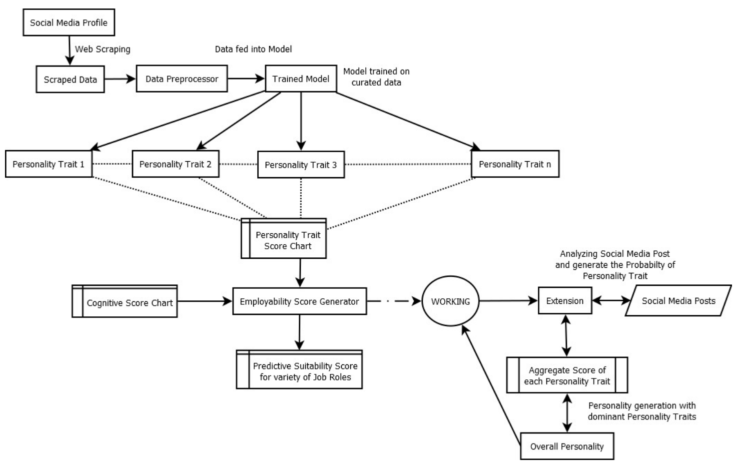

Social Media Profile
Web Scraping
Data fed into Model
Scraped Data
Data Preprocessor
Trained Model
Model trained on curated data
Personality Trait 1
Personality Trait 2
Personality Trait 3
Personality Trait n
Personality Trait Score Chart
Analyzing Social Media Post and generate the Probabilty of Personality Trait
Cognitive Score Chart
Employability Score Generator
WORKING
Extension
Social Media Posts
Aggregate Score of each Personality Trait
Predictive Suitability Score for variety of Job Roles
Personality generation with dominant Personality Traits
Overall Personality

Figure 5: High level design

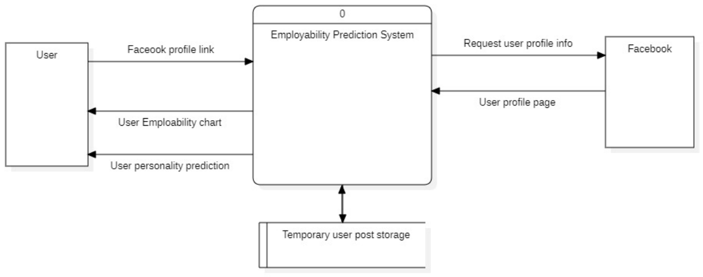

0
Employability Prediction System
User
Faceook profile link
User Emploability chart
User personality prediction
Request user profile info
User profile page
Facebook
Temporary user post storage

Figure 6: Level 0 Data Flow Diagram

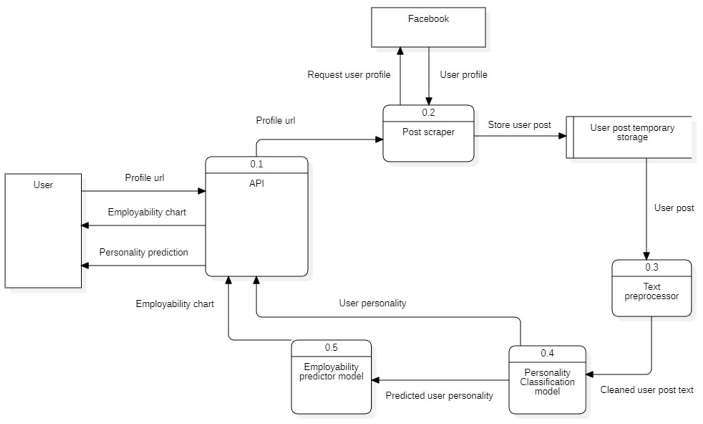


Figure 7: Level 1 Data Flow Diagram

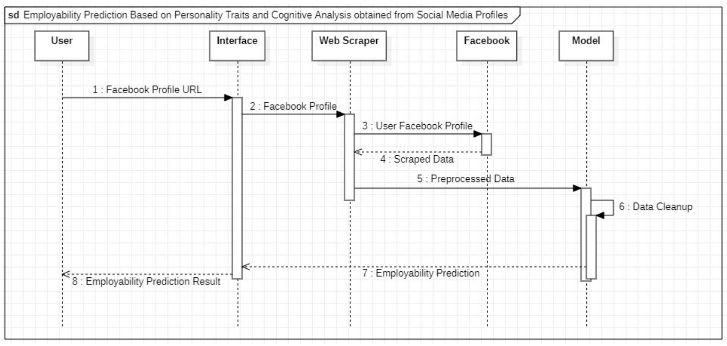

sd Employability Prediction Based on Personality Traits and Cognitive Analysis obtained from Social Media Profiles
User
Interface
Web Scraper
Facebook
Model
1 : Facebook Profile URL
2 : Facebook Profile
3 : User Facebook Profile
4 : Scraped Data
5 : Preprocessed Data
6 : Data Cleanup
7 : Employability Prediction
8 : Employability Prediction Result

Figure 8: Sequence Diagram

8 Implementation

For the initial prototype, a subset of high-priority requirements from the Requirement Matrix (RM) was chosen to implement core functionalities and demonstrate proof of concept. The focus is on data collection, data preprocessing, model training, and evaluation, as these are fundamental to the project's success. By selecting these critical features, the aim is to ensure that the data pipeline and model work seamlessly together. This approach provides a solid foundation for future integration with additional components and more complex functionalities, allowing for validation of key processes before expanding the project's scope. The below subsections highlight the details of features implemented.

8.1 Automated Post Collection

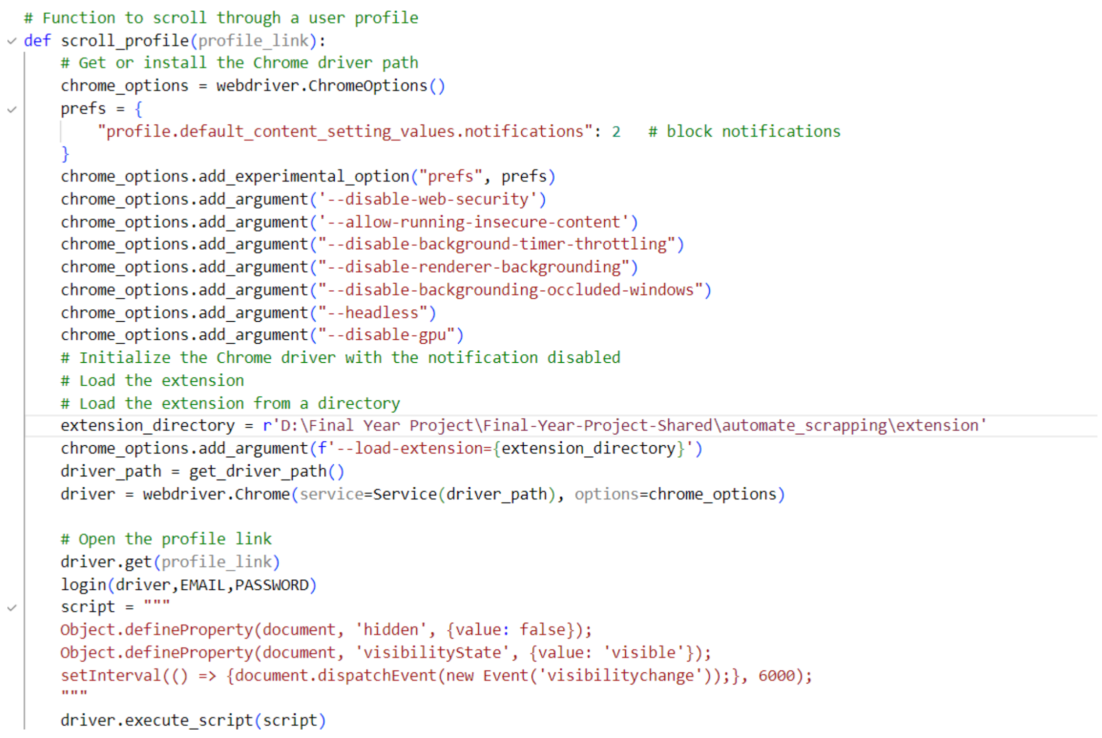
The scroll_profile function automates scrolling through a Facebook user's profile and extracts posts, sending the data to an API endpoint. It begins by setting up a Chrome browser instance using Selenium WebDriver, configured to block notifications and disable certain features like background throttling, ensuring smooth execution even when the browser is minimized or inactive. A custom Chrome extension is loaded, for bypassing restrictions or handling dynamic content.

The function logs into Facebook with provided credentials and navigates to the user's profile. To keep the page active during scrolling, it executes a JavaScript snippet that overrides the browser's visibility properties, tricking the site into thinking the tab is always in focus.

Once on the profile, the function starts scrolling. It calculates the page's scroll height and moves downward in increments, pausing briefly between scrolls to let additional content load. This process continues until a certain scroll limit or until no new content is found. If any pop-up alerts appear, they are dismissed to prevent interruptions.

As the profile is scrolled, posts are identified, and their text is extracted (though this part is not detailed in the function). The extracted text is then sent to a predefined API endpoint for further processing. After reaching the scroll limit or loading all content, the function waits for a short time and then closes the browser, completing the task.

This automated process allows for efficient data collection from Facebook profiles without manual intervention.



```python
# Function to scroll through a user profile
def scroll_profile(profile_link):
    # Get or install the Chrome driver path
    chrome_options = webdriver.ChromeOptions()
    prefs = {
        "profile.default_content_setting_values.notifications": 2   # block notifications
    }
    chrome_options.add_experimental_option("prefs", prefs)
    chrome_options.add_argument('--disable-web-security')
    chrome_options.add_argument('--allow-running-insecure-content')
    chrome_options.add_argument("--disable-background-timer-throttling")
    chrome_options.add_argument("--disable-renderer-backgrounding")
    chrome_options.add_argument("--disable-backgrounding-occluded-windows")
    chrome_options.add_argument("--headless")
    chrome_options.add_argument("--disable-gpu")
    # Initialize the Chrome driver with the notification disabled
    # Load the extension
    # Load the extension from a directory
    extension_directory = r'D:\Final Year Project\Final-Year-Project-Shared\automate_scrapping\extension'
    chrome_options.add_argument(f'--load-extension={extension_directory}')
    driver_path = get_driver_path()
    driver = webdriver.Chrome(service=Service(driver_path), options=chrome_options)

    # Open the profile link
    driver.get(profile_link)
    login(driver,EMAIL,PASSWORD)
    script = """
    Object.defineProperty(document, 'hidden', {value: false});
    Object.defineProperty(document, 'visibilityState', {value: 'visible'});
    setInterval(() => {document.dispatchEvent(new Event('visibilitychange'));}, 6000);
    """
    driver.execute_script(script)
```

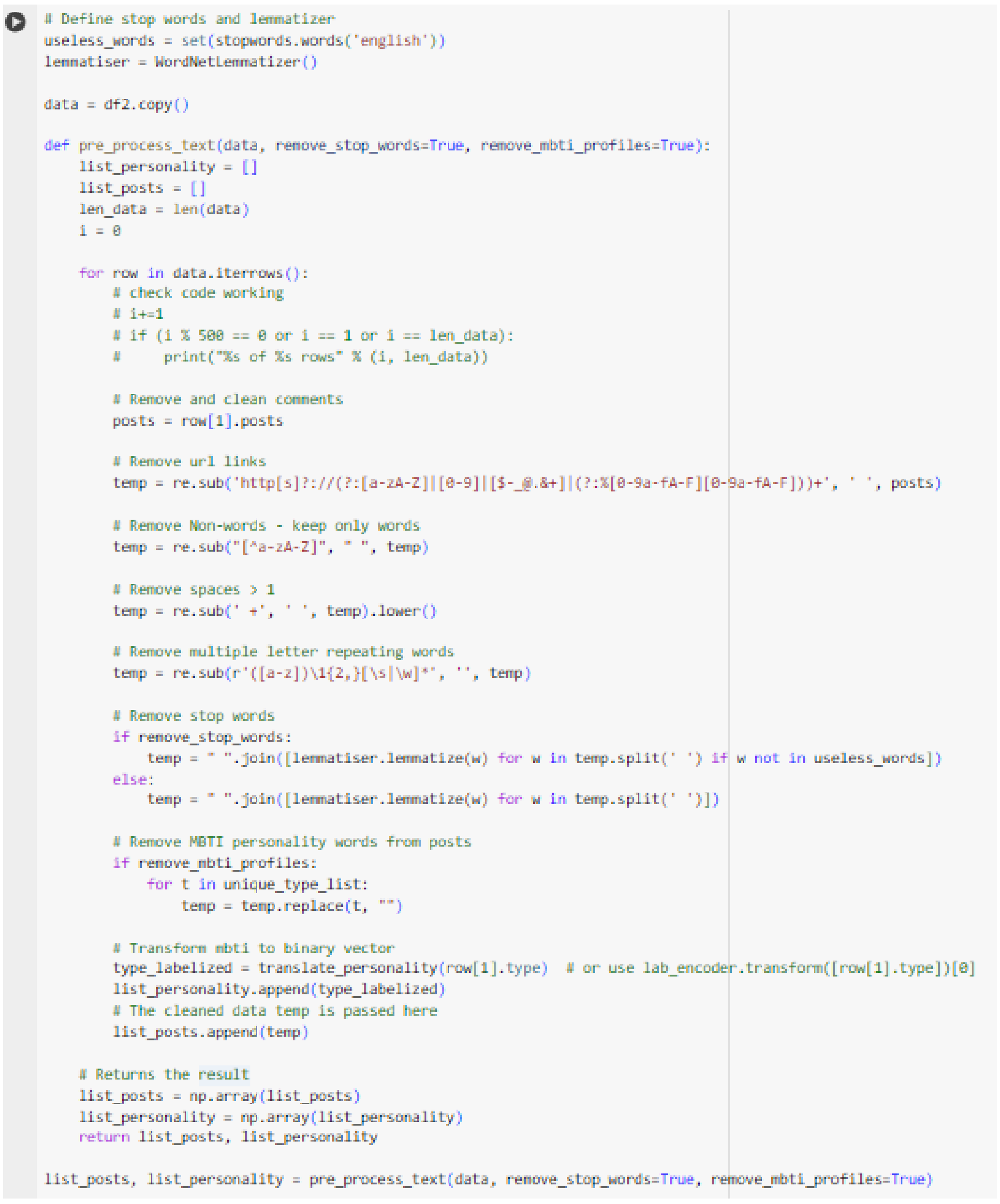
Figure 9: Automated Post Collection

8.2 Text Preprocessing

The function `preprocess_posts` is used to clean the text of user posts, that it receives from the above step, by performing the following:

- Removal of URLs
- Elimination of non-word characters
- Removal of extra spaces and repeated letters
- Optionally removes MBTI personality types if `remove_mbti_profiles` is set to `True`.
- Stopwords are removed if `remove_stop_words` is set to `True`.
- Lemmatization is applied to normalize words into their base form.

Now this data is ready for further processing.



```python
# Define stop words and lemmatizer
useless_words = set(stopwords.words('english'))
lemmatiser = WordNetLemmatizer()

data = df2.copy()

def pre_process_text(data, remove_stop_words=True, remove_mbti_profiles=True):
    list_personality = []
    list_posts = []
    len_data = len(data)
    i = 0

    for row in data.iterrows():
        # check code working
        # i+=1
        # if (i % 500 == 0 or i == 1 or i == len_data):
        #     print("%s of %s rows" % (i, len_data))

        # Remove and clean comments
        posts = row[1].posts

        # Remove url links
        temp = re.sub('http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|(?:%[0-9a-fA-F][0-9a-fA-F]))+', ' ', posts)

        # Remove Non-words - keep only words
        temp = re.sub("[^a-zA-Z]", " ", temp)

        # Remove spaces > 1
        temp = re.sub(' +', ' ', temp).lower()

        # Remove multiple letter repeating words
        temp = re.sub(r'([a-z])\1{2,}[\s|\w]*', '', temp)

        # Remove stop words
        if remove_stop_words:
            temp = " ".join([lemmatiser.lemmatize(w) for w in temp.split(' ') if w not in useless_words])
        else:
            temp = " ".join([lemmatiser.lemmatize(w) for w in temp.split(' ')])

        # Remove MBTI personality words from posts
        if remove_mbti_profiles:
            for t in unique_type_list:
                temp = temp.replace(t, "")

        # Transform mbti to binary vector
        type_labelized = translate_personality(row[1].type)  # or use lab_encoder.transform([row[1].type])[0]
        list_personality.append(type_labelized)
        # The cleaned data temp is passed here
        list_posts.append(temp)

    # Returns the result
    list_posts = np.array(list_posts)
    list_personality = np.array(list_personality)
    return list_posts, list_personality

list_posts, list_personality = pre_process_text(data, remove_stop_words=True, remove_mbti_profiles=True)
```

Figure 10: Data Preprocessing

8.3 Personality Classification System

The system designed for personality classification utilizes several machine learning and natural language processing techniques. This section provides a detailed overview of the different components, including the technical environment, and model training processes.

8.3.1 Technical Environment

Unique Type List and Personality Dictionary

The system uses a `unique_type_list` containing the 16 MBTI types and a `b_Pers` dictionary to map MBTI dichotomies to binary values:

- **I/E**: 0 for Introverted (I), 1 for Extroverted (E)
- **N/S**: 0 for Intuitive (N), 1 for Sensing (S)
- **F/T**: 0 for Feeling (F), 1 for Thinking (T)
- **J/P**: 0 for Judging (J), 1 for Perceiving (P)

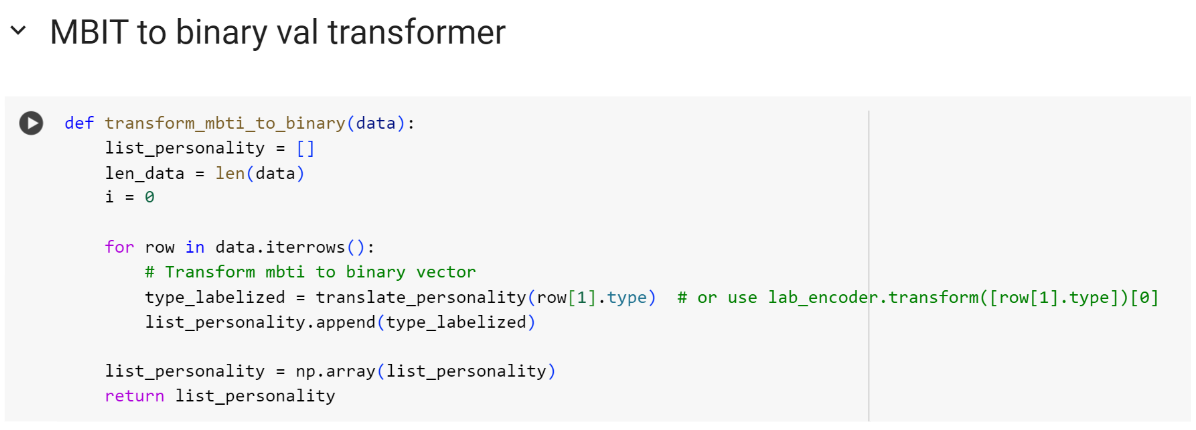
8.3.2 Personality Translation Functions

Two functions are implemented for converting between MBTI types and binary vectors:

- `translate_personality`: Translates an MBTI type (e.g., 'INFJ') into its corresponding binary vector (e.g., [0, 0, 0, 0]).
- `translate_back`: Converts a binary vector (e.g., [0, 0, 0, 0]) back into its corresponding MBTI type using the `b_Pers` dictionary.

8.3.3 Binary Transformation for MBTI

The function `transform_mbti_to_binary` is responsible for converting MBTI personality types into their binary vector representation. It iterates over the dataset rows and applies `translate_personality` to achieve this transformation.


MBIT to binary val transformer

```python
def transform_mbti_to_binary(data):
    list_personality = []
    len_data = len(data)
    i = 0

    for row in data.iterrows():
        # Transform mbti to binary vector
        type_labelized = translate_personality(row[1].type)  # or use lab_encoder.transform([row[1].type])[0]
        list_personality.append(type_labelized)

    list_personality = np.array(list_personality)
    return list_personality
```

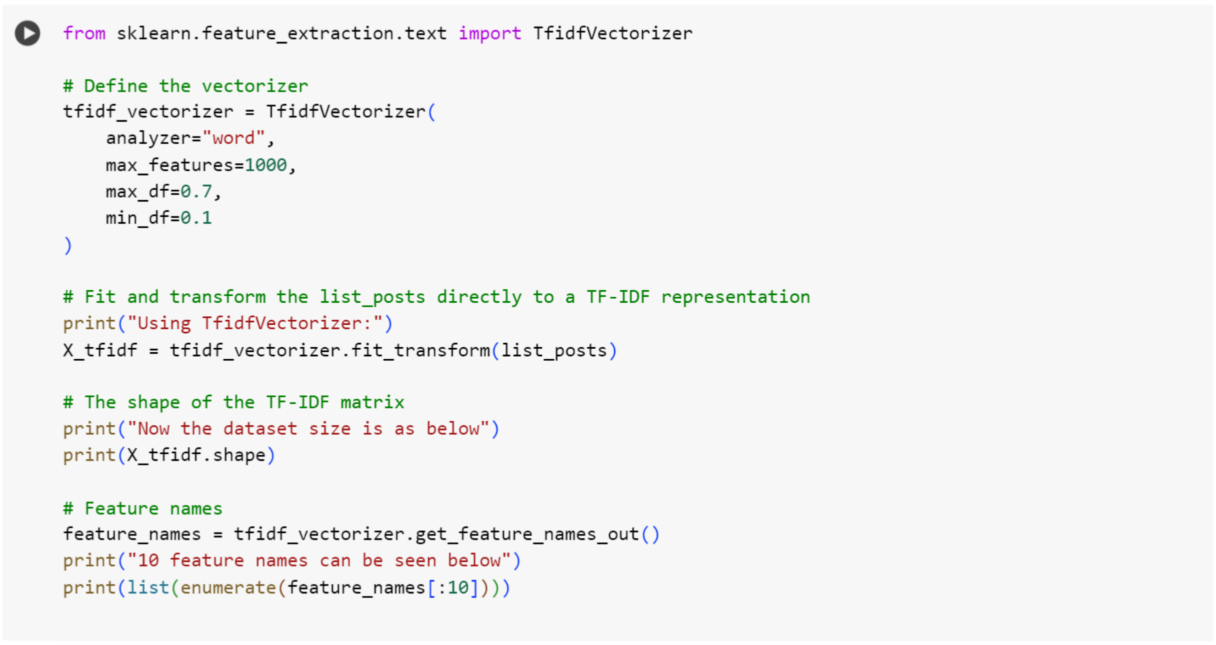
Figure 11: Binary Transformation for MBTI

8.3.4 TF-IDF Vectorization

The text data is vectorized using the `TfidfVectorizer`, which is initialized with the following parameters:

- **max_features=1000**: Limits the number of features to the top 1000 words.
- **max_df=0.7**: Ignores terms that appear in more than 70% of the documents.
- **min_df=0.1**: Ignores terms that appear in less than 10% of the documents.

The `fit_transform` method is applied to `list_posts`, generating the TF-IDF matrix `X_tfidf`.



```python
from sklearn.feature_extraction.text import TfidfVectorizer

# Define the vectorizer
tfidf_vectorizer = TfidfVectorizer(
    analyzer="word",
    max_features=1000,
    max_df=0.7,
    min_df=0.1
)

# Fit and transform the list_posts directly to a TF-IDF representation
print("Using TfidfVectorizer:")
X_tfidf = tfidf_vectorizer.fit_transform(list_posts)

# The shape of the TF-IDF matrix
print("Now the dataset size is as below")
print(X_tfidf.shape)

# Feature names
feature_names = tfidf_vectorizer.get_feature_names_out()
print("10 feature names can be seen below")
print(list(enumerate(feature_names[:10])))
```

Figure 12: TF-IDF Vectorization

8.3.5 Personality Type Dichotomies

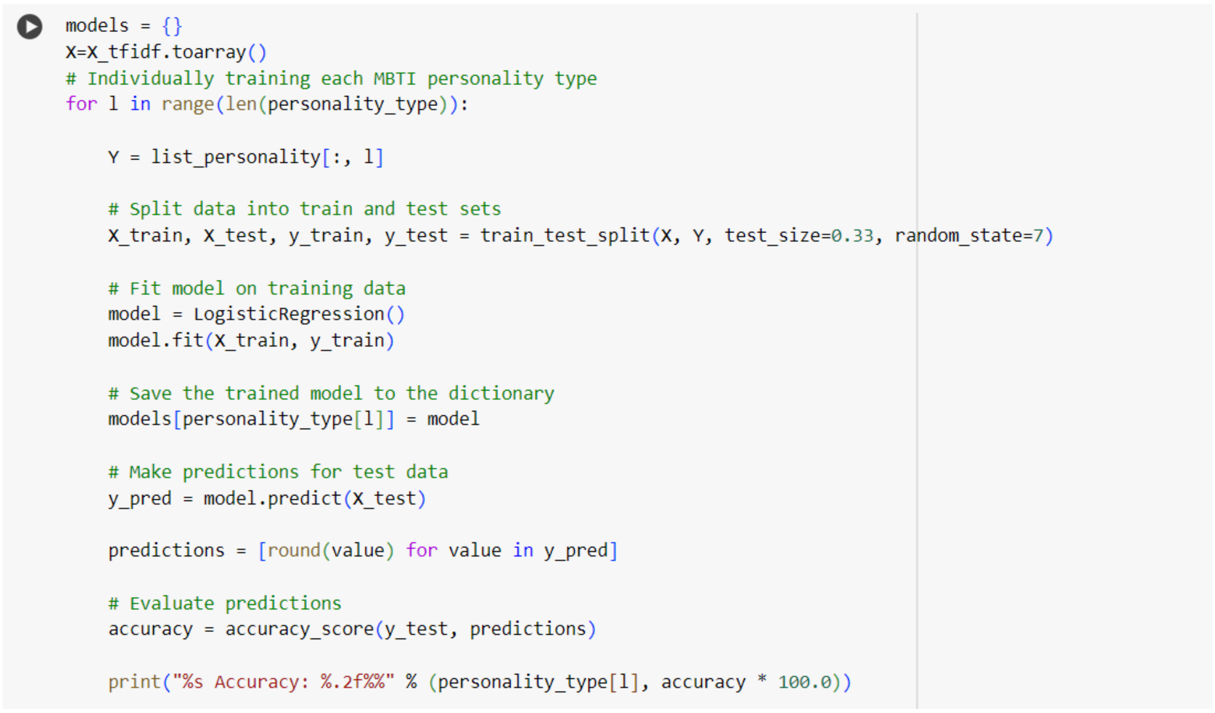
The four MBTI dichotomies are stored in the `personality_type` list, representing:

- **IE**: Introversion (I) / Extroversion (E)
- **NS**: Intuition (N) / Sensing (S)
- **FT**: Feeling (F) / Thinking (T)
- **JP**: Judging (J) / Perceiving (P)

8.3.6 Training Logistic Regression Models

For each dichotomy (IE, NS, FT, JP), the system:

- Splits the data into training and test sets using `train_test_split`.
- Trains a `LogisticRegression` model.
- Saves the trained models in a dictionary with the dichotomy as the key.
- Evaluates the model's accuracy using `accuracy_score` and prints the results.



```python
models = {}
X=X_tfidf.toarray()
# Individually training each MBTI personality type
for l in range(len(personality_type)):

    Y = list_personality[:, l]

    # Split data into train and test sets
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.33, random_state=7)

    # Fit model on training data
    model = LogisticRegression()
    model.fit(X_train, y_train)

    # Save the trained model to the dictionary
    models[personality_type[l]] = model

    # Make predictions for test data
    y_pred = model.predict(X_test)

    predictions = [round(value) for value in y_pred]

    # Evaluate predictions
    accuracy = accuracy_score(y_test, predictions)

    print("%s Accuracy: %.2f%%" % (personality_type[l], accuracy * 100.0))
```

Figure 13: Model Training

9 Test Plans, Results and Analysis

9.1 Report on Model Accuracies for MBTI Personality Type Classification

The purpose of this report is to evaluate the performance of six machine learning models in classifying the MBTI personality types. The models tested include RandomForest, XGBoost, Stochastic Gradient Descent (SGD), Logistic Regression, Artificial Neural Networks (ANN), and XGBoost Hypertuned. These models were evaluated on their ability to predict the four MBTI dichotomies:

- Introversion (I) / Extroversion (E)
- Intuition (N) / Sensing (S)
- Feeling (F) / Thinking (T)
- Judging (J) / Perceiving (P)

Each model's performance was measured using classification accuracy. The findings are discussed below, followed by the corresponding test plan, results, and analysis.

9.2 Test Plans

The test cases were designed to evaluate the accuracy of each model across the four MBTI dichotomies. Each test case was assigned a unique identifier based on the module being tested.

9.3 Test Results

The expected test results for each model across the dichotomies are presented in the following pages.

9.4 Performance Metrics

Classification Accuracy: The primary performance metric for this analysis is classification accuracy, which measures the proportion of correct predictions over the total number of predictions for each dichotomy. The results show how well each model performs for specific MBTI dichotomies.

9.5 Test Results Analysis

Key Observations:

1. **Best Performance for N/S Dichotomy:** Logistic Regression and SGD models perform the best for the Intuition/Sensing (N/S) dichotomy, with accuracies of 86.06% and 86.03%, respectively. RandomForest also performs well with an accuracy of 85.52%, while XGBoost achieves 85.19% accuracy.

2. **Strong I/E Classification:** RandomForest, SGD, Logistic Regression, and XGBoost Hypertuned show similar performances in classifying Introversion/Extroversion (I/E), with accuracies around 77%. ANN performs comparatively lower, achieving 71.71% accuracy.

3. **Weaker Performance for F/T and J/P Dichotomies:** Across all models, the Feeling/Thinking (F/T) and Judging/Perceiving (J/P) dichotomies have the lowest accuracies. The highest accuracy for F/T classification is achieved by Logistic Regression (72.44%), followed closely by SGD (72.34%). For J/P classification, the best accuracy is also from Logistic Regression (64.51%), followed closely by SGD (64.41%).

4. **XGBoost Hypertuned Improvement:** The XGBoost Hypertuned model slightly improves accuracy for the F/T and J/P dichotomies compared to the regular XGBoost model, showing better performance in capturing the distinctions in the more complex personality traits.

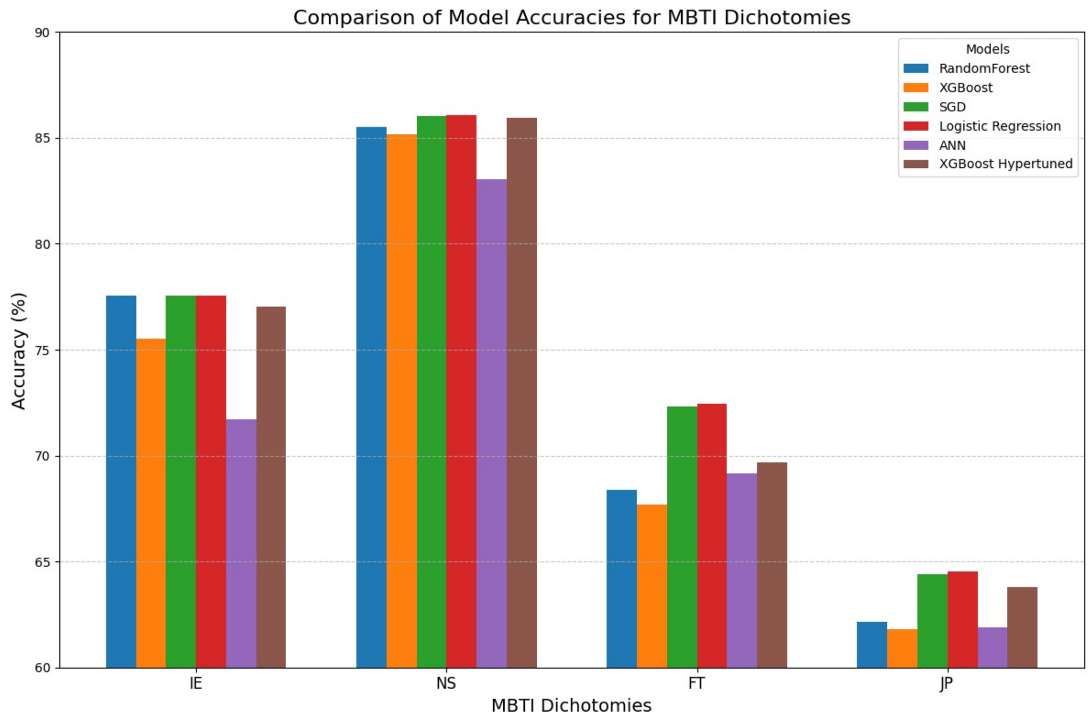


Figure 14: Model accuracy comparison for each Dichotomy

9.6 Findings

Logistic Regression and SGD are the top-performing models overall, especially for the N/S dichotomy, where they achieved the highest accuracies. The models generally perform better on the I/E and N/S dichotomies, while the F/T and J/P dichotomies proved to be more challenging. XGBoost Hypertuned showed marginal improvements in the F/T and J/P dichotomies compared to the base XGBoost model.

For future improvements, a focus on enhancing performance for the F/T and J/P dichotomies may require more feature engineering, deeper models, or alternative learning techniques.

10 Conclusion

The benefits brought about by the project will be plentiful.

- The tool will assist career counselors and students in making more informed career-related decisions.
- It will enable employers to enhance their hiring practices and add greater specificity to the recruitment process.
- The model or hiring metric can function as an independent product, allowing for customization based on different job roles, skill sets, and various hiring scenarios.
- Given that personality traits and cognitive abilities can be evaluated over an extended period, these tools can be utilized to direct efforts and offer suitable training programs for both students and professionals.

While the Employability Prediction Tool serves as a direct resource for students, professionals, employers, and career counselors, the model and hiring metric are distinct deliverables designed for diverse applications. Ultimately, the project seeks to narrow the hiring search scope, facilitate more targeted hiring for specific job roles, minimize the resources required for entry-level recruitment, and ensure that the selected candidates align better with the organization's work culture.

11 References

[1] M. R. Bakry, M. M. Nasr, and F. K. Alsheref, ”Personality Classification Model of Social Network Profiles based on their Activities and Contents,” *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 7, 2022. [Online]. Available: https://thesai.org/Downloads/Volume13No7/Paper_3-Personality_Classification_Model_of_Social_Network.pdf

[2] A. Souri, S. Hosseinpour, and A. M. Rahmani, ”Personality classification based on profiles of social networks' users and the five-factor model of personality,” *Hum. Cent. Comput. Inf. Sci.*, vol. 8, p. 24, 2018. [Online]. Available: https://doi.org/10.1186/s13673-018-0147-4

[3] R. Hernandez and I. S. Knight, ”Predicting Myers-Briggs Type Indicator with Text Classification,” 2018. [Online]. Available: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1184/reports/6839354.pdf

[4] Ö. Öngöre, ”A Study of Relationship between Personality Traits and Job Engagement,” *Procedia - Social and Behavioral Sciences*, vol. 141, pp. 1315-1319, 2014. [Online]. Available: https://doi.org/10.1016/j.sbspro.2014.05.226

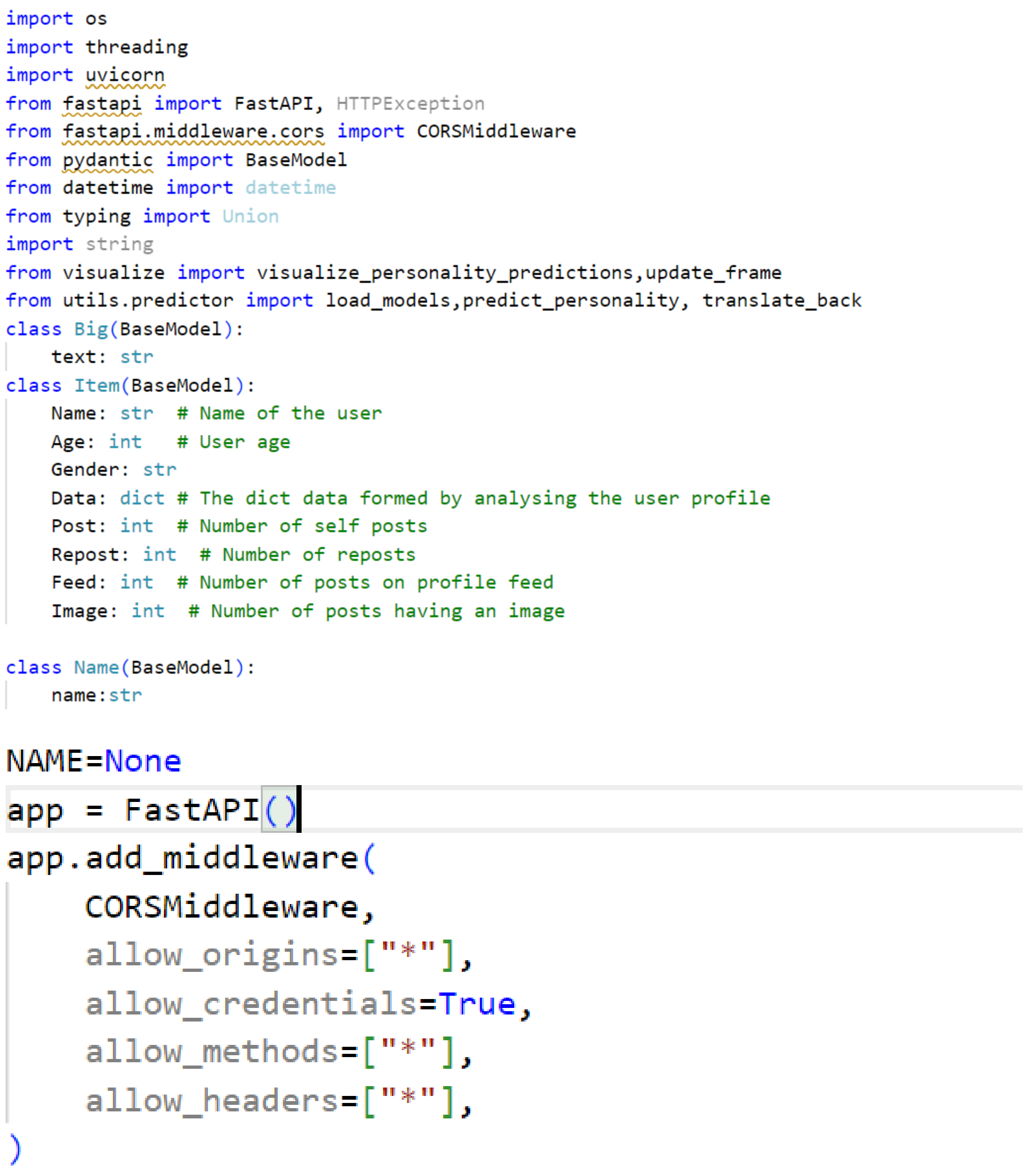
[5] M. Nwogu, ”Graduate Employability Qualities and Personality Preference as Determinants of Job Performance in Nigeria,” *European Scientific Journal*, vol. 11, no. 25, Sep. 2015. [Online]. Available: https://core.ac.uk/download/pdf/236411658.pdf

12 APPENDIX A - Prototypes (optional)

Current version of our Prototype is capable of Automated post collection, Data preprocessing, and then Personality classification using the pretrained model based on MBTI. The steps to generate a user’s personalized personality classification report is as follows:

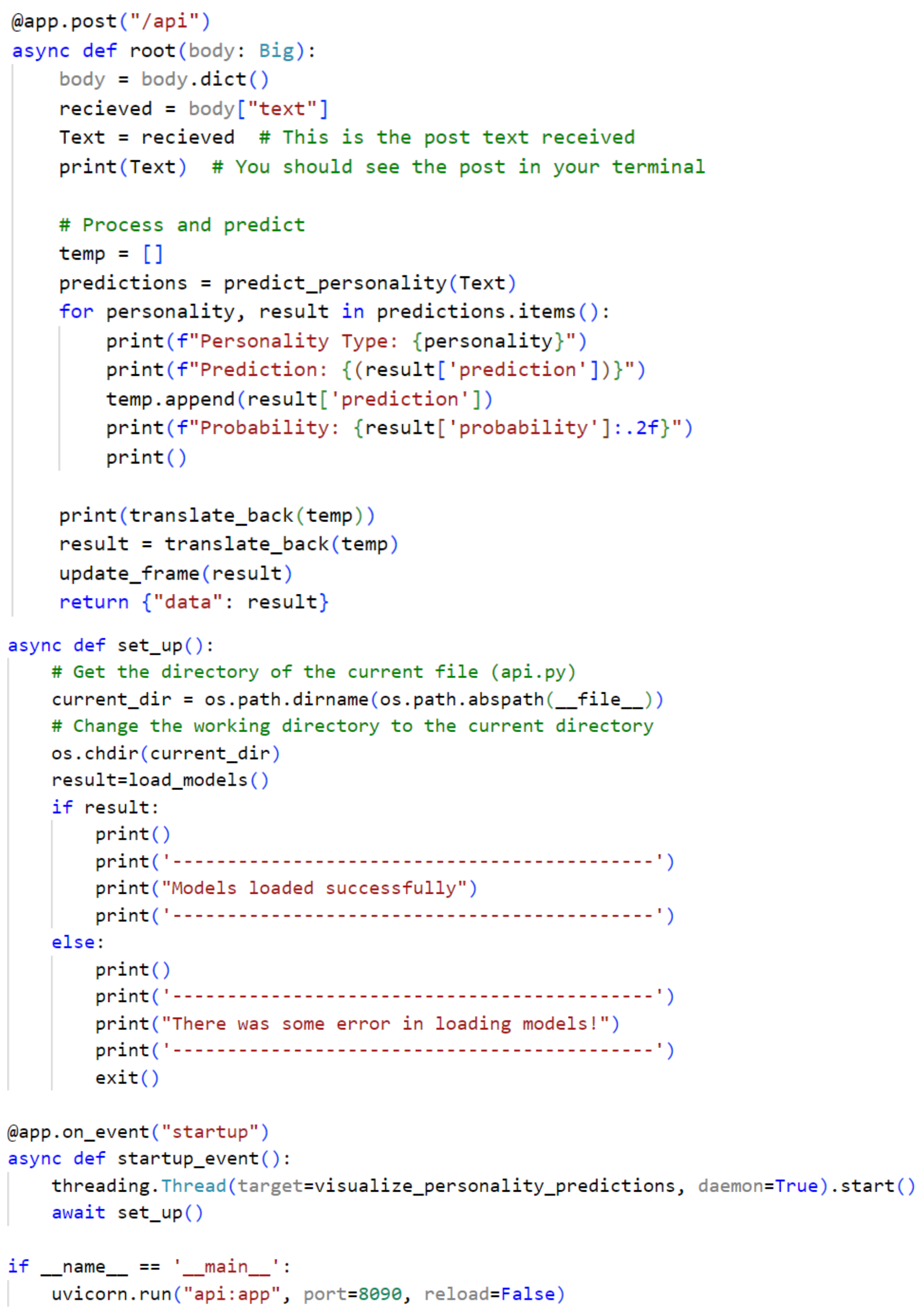
- We start by running the post collection API process. This will collect the user post that will be sent by the Post Scraper.
- Then we start the Post scraper, which in turn asks for the user’s Faceook profile link.
- Then the Post Scraper loads user profile in a automated browser environment.
- We load an extension to bypass the content security policies of the page.
- The extension automatically scrolls the user’s profile page.
- Each post gets sent to the API end-point.
- The post gets cleaned and gets sent to the Personality classifier.
- The pretrained model then classifies the text.
- We are currently using matplotlib to visualize user’s personality results.
- In future this personality classification result will be used to create a mapping to employability matrix.

Code snippets are provided below:



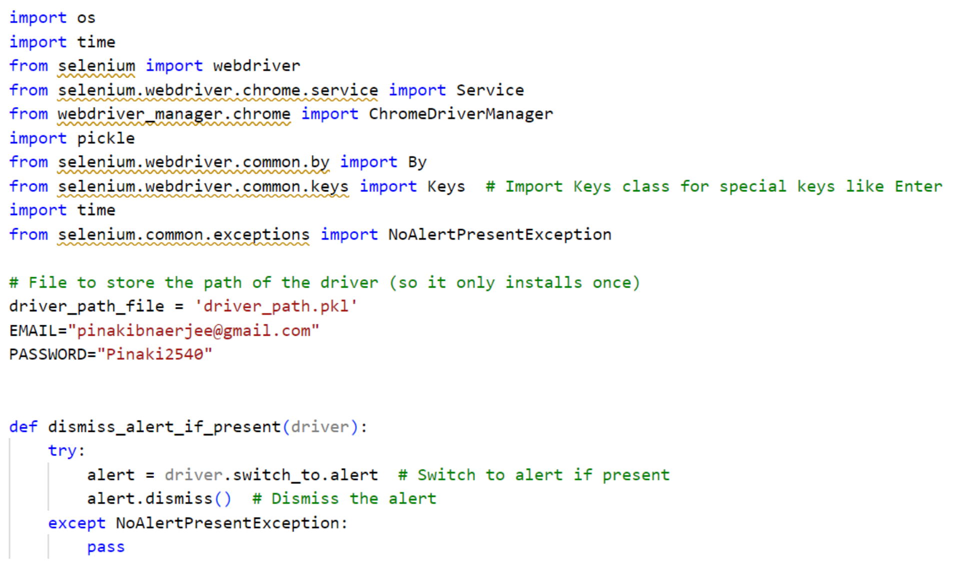
```
import os
import threading
import uvicorn
from fastapi import FastAPI, HTTPException
from fastapi.middleware.cors import CORSMiddleware
from pydantic import BaseModel
from datetime import datetime
from typing import Union
import string
from visualize import visualize_personality_predictions,update_frame
from utils.predictor import load_models,predict_personality, translate_back
class Big(BaseModel):
    text: str
class Item(BaseModel):
    Name: str  # Name of the user
    Age: int   # User age
    Gender: str
    Data: dict # The dict data formed by analysing the user profile
    Post: int  # Number of self posts
    Repost: int  # Number of reposts
    Feed: int  # Number of posts on profile feed
    Image: int  # Number of posts having an image

class Name(BaseModel):
    name:str

NAME=None
app = FastAPI()
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)
```

Figure 15: API part:1

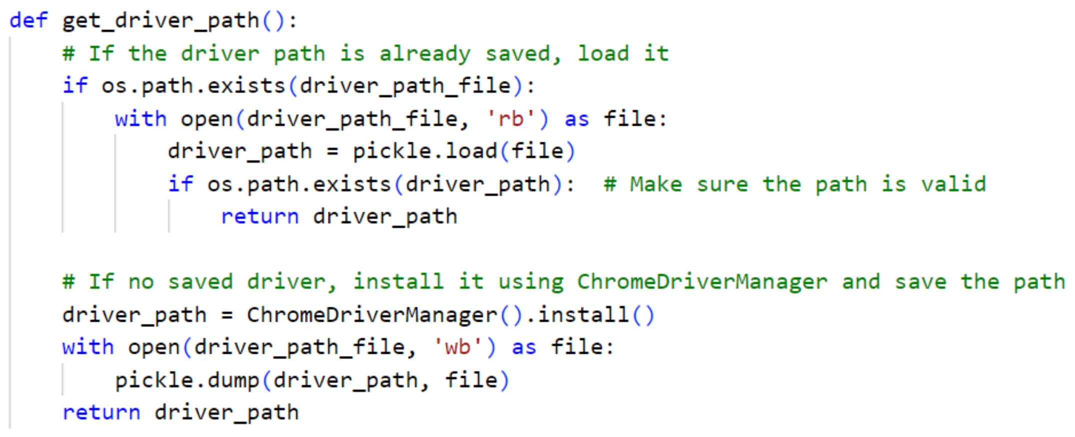


```
@app.post("/api")
async def root(body: Big):
    body = body.dict()
    recieved = body["text"]
    Text = recieved  # This is the post text received
    print(Text)  # You should see the post in your terminal

    # Process and predict
    temp = []
    predictions = predict_personality(Text)
    for personality, result in predictions.items():
        print(f"Personality Type: {personality}")
        print(f"Prediction: {(result['prediction'])}")
        temp.append(result['prediction'])
        print(f"Probability: {result['probability']:.2f}")
        print()

    print(translate_back(temp))
    result = translate_back(temp)
    update_frame(result)
    return {"data": result}

async def set_up():
    # Get the directory of the current file (api.py)
    current_dir = os.path.dirname(os.path.abspath(__file__))
    # Change the working directory to the current directory
    os.chdir(current_dir)
    result=load_models()
    if result:
        print()
        print('--------------------------------------------')
        print("Models loaded successfully")
        print('--------------------------------------------')
    else:
        print()
        print('--------------------------------------------')
        print("There was some error in loading models!")
        print('--------------------------------------------')
        exit()

@app.on_event("startup")
async def startup_event():
    threading.Thread(target=visualize_personality_predictions, daemon=True).start()
    await set_up()

if __name__ == '__main__':
    uvicorn.run("api:app", port=8090, reload=False)
```

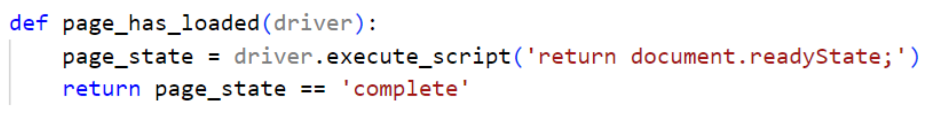
Figure 16: API part:2



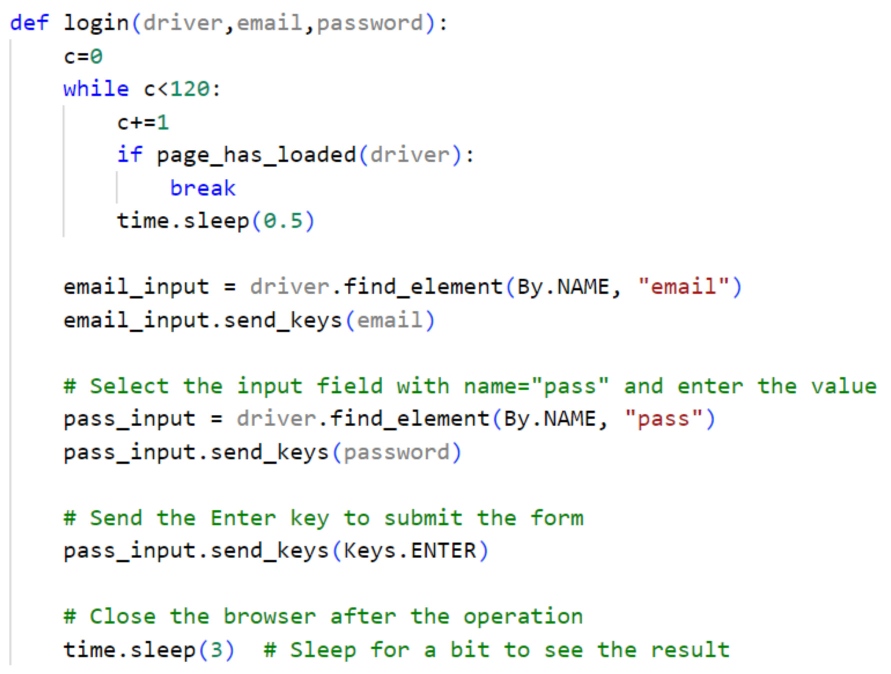
```python
import os
import time
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
import pickle
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys  # Import Keys class for special keys like Enter
import time
from selenium.common.exceptions import NoAlertPresentException

# File to store the path of the driver (so it only installs once)
driver_path_file = 'driver_path.pkl'
EMAIL="pinakibnaerjee@gmail.com"
PASSWORD="Pinaki2540"


def dismiss_alert_if_present(driver):
    try:
        alert = driver.switch_to.alert  # Switch to alert if present
        alert.dismiss()  # Dismiss the alert
    except NoAlertPresentException:
        pass
```



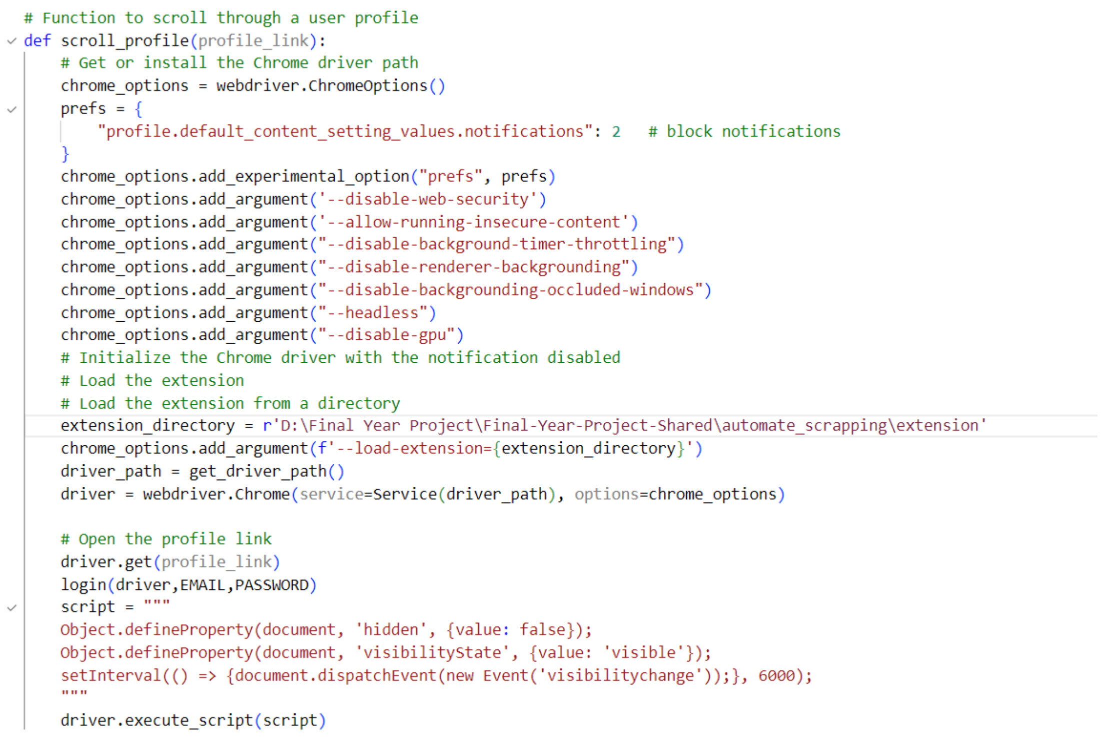
```python
def get_driver_path():
    # If the driver path is already saved, load it
    if os.path.exists(driver_path_file):
        with open(driver_path_file, 'rb') as file:
            driver_path = pickle.load(file)
            if os.path.exists(driver_path):  # Make sure the path is valid
                return driver_path

    # If no saved driver, install it using ChromeDriverManager and save the path
    driver_path = ChromeDriverManager().install()
    with open(driver_path_file, 'wb') as file:
        pickle.dump(driver_path, file)
    return driver_path
```



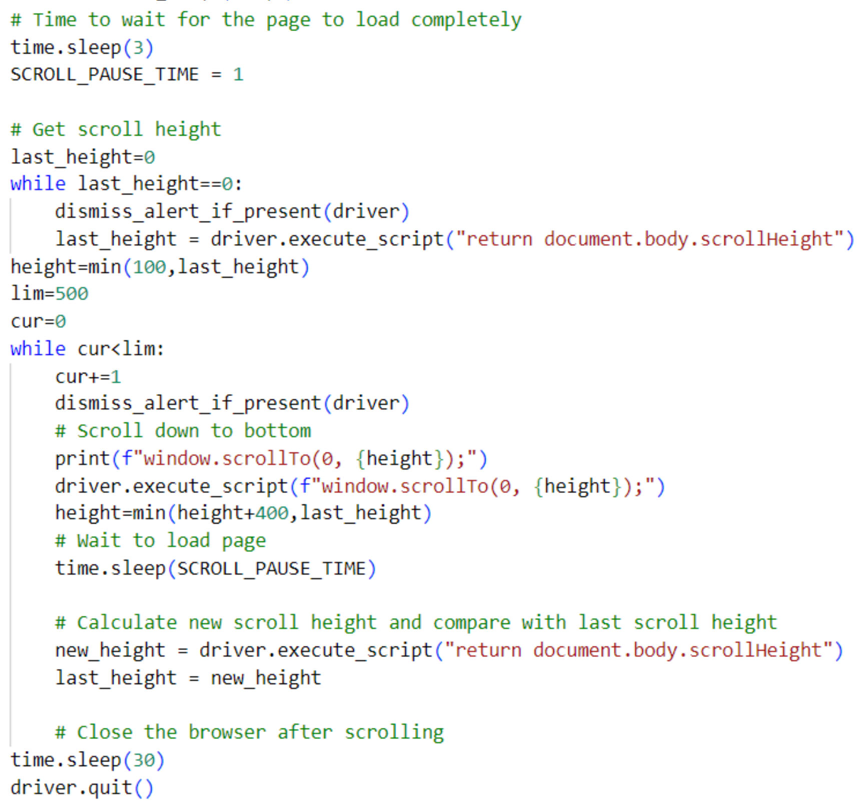
```python
def page_has_loaded(driver):
    page_state = driver.execute_script('return document.readyState;')
    return page_state == 'complete'
```



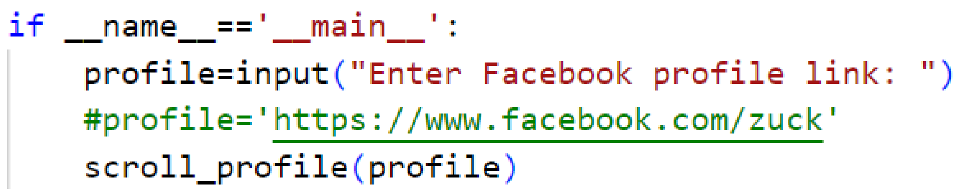
```python
def login(driver,email,password):
    c=0
    while c<120:
        c+=1
        if page_has_loaded(driver):
            break
        time.sleep(0.5)

    email_input = driver.find_element(By.NAME, "email")
    email_input.send_keys(email)

    # Select the input field with name="pass" and enter the value
    pass_input = driver.find_element(By.NAME, "pass")
    pass_input.send_keys(password)

    # Send the Enter key to submit the form
    pass_input.send_keys(Keys.ENTER)

    # Close the browser after the operation
    time.sleep(3)  # Sleep for a bit to see the result
```



```python
# Function to scroll through a user profile
def scroll_profile(profile_link):
    # Get or install the Chrome driver path
    chrome_options = webdriver.ChromeOptions()
    prefs = {
        "profile.default_content_setting_values.notifications": 2   # block notifications
    }
    chrome_options.add_experimental_option("prefs", prefs)
    chrome_options.add_argument('--disable-web-security')
    chrome_options.add_argument('--allow-running-insecure-content')
    chrome_options.add_argument("--disable-background-timer-throttling")
    chrome_options.add_argument("--disable-renderer-backgrounding")
    chrome_options.add_argument("--disable-backgrounding-occluded-windows")
    chrome_options.add_argument("--headless")
    chrome_options.add_argument("--disable-gpu")
    # Initialize the Chrome driver with the notification disabled
    # Load the extension
    # Load the extension from a directory
    extension_directory = r'D:\Final Year Project\Final-Year-Project-Shared\automate_scrapping\extension'
    chrome_options.add_argument(f'--load-extension={extension_directory}')
    driver_path = get_driver_path()
    driver = webdriver.Chrome(service=Service(driver_path), options=chrome_options)

    # Open the profile link
    driver.get(profile_link)
    login(driver,EMAIL,PASSWORD)
    script = """
    Object.defineProperty(document, 'hidden', {value: false});
    Object.defineProperty(document, 'visibilityState', {value: 'visible'});
    setInterval(() => {document.dispatchEvent(new Event('visibilitychange'));}, 6000);
    """
    driver.execute_script(script)
```

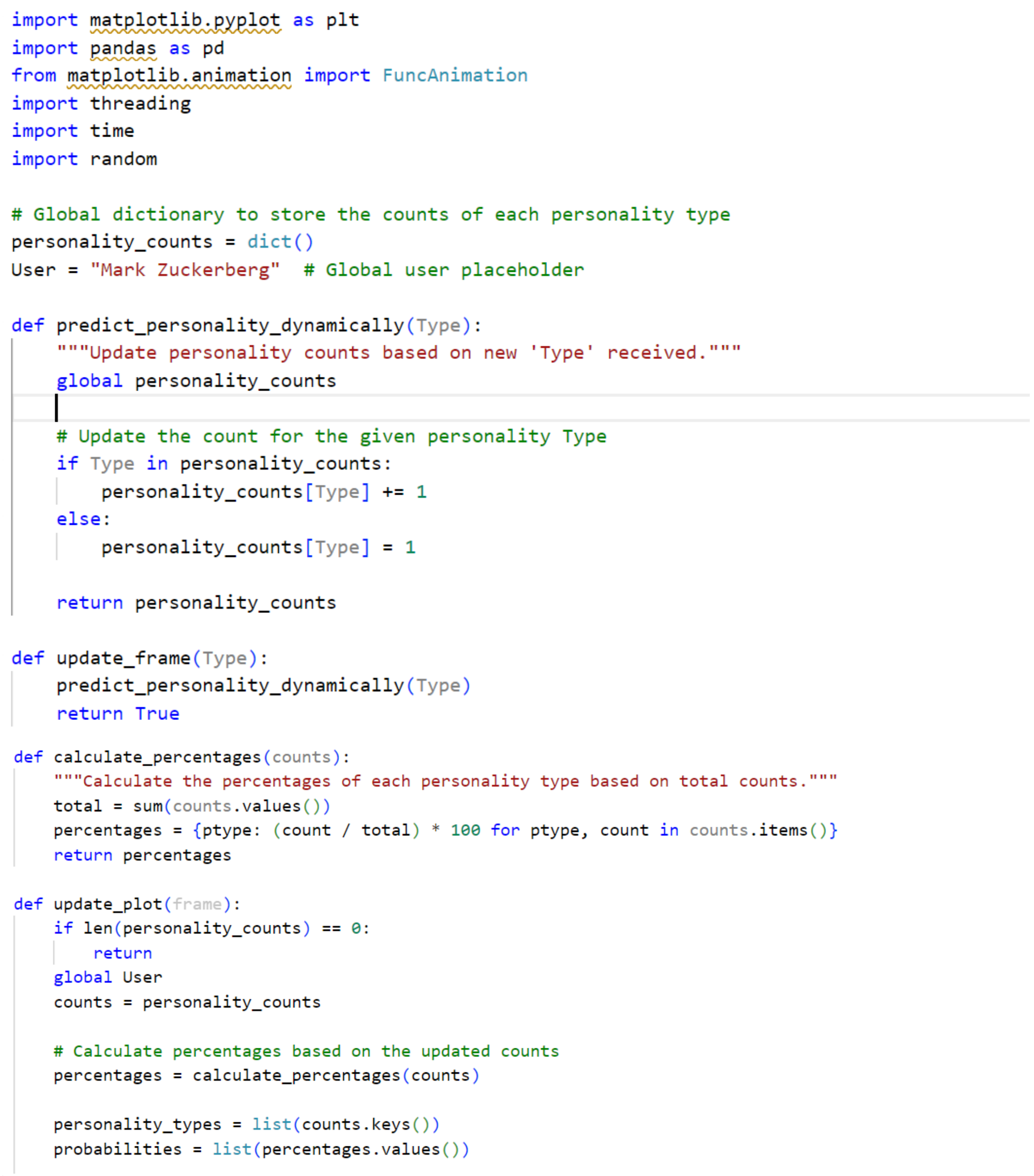


```python
    # Time to wait for the page to load completely
    time.sleep(3)
    SCROLL_PAUSE_TIME = 1

    # Get scroll height
    last_height=0
    while last_height==0:
        dismiss_alert_if_present(driver)
        last_height = driver.execute_script("return document.body.scrollHeight")
    height=min(100,last_height)
    lim=500
    cur=0
    while cur<lim:
        cur+=1
        dismiss_alert_if_present(driver)
        # Scroll down to bottom
        print(f"window.scrollTo(0, {height});")
        driver.execute_script(f"window.scrollTo(0, {height});")
        height=min(height+400,last_height)
        # Wait to load page
        time.sleep(SCROLL_PAUSE_TIME)

        # Calculate new scroll height and compare with last scroll height
        new_height = driver.execute_script("return document.body.scrollHeight")
        last_height = new_height

        # Close the browser after scrolling
    time.sleep(30)
    driver.quit()
```

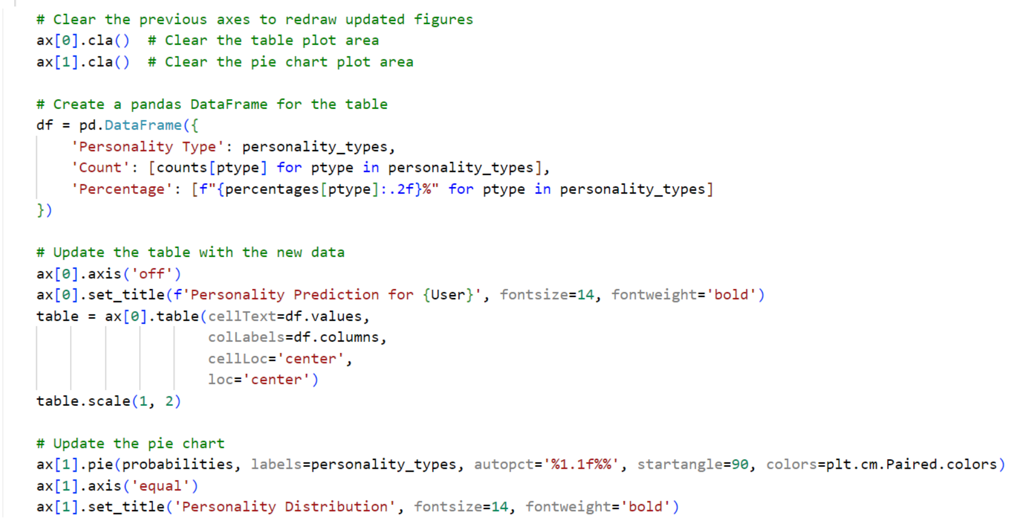


```python
if __name__=='__main__':
    profile=input("Enter Facebook profile link: ")
    #profile='https://www.facebook.com/zuck'
    scroll_profile(profile)
```

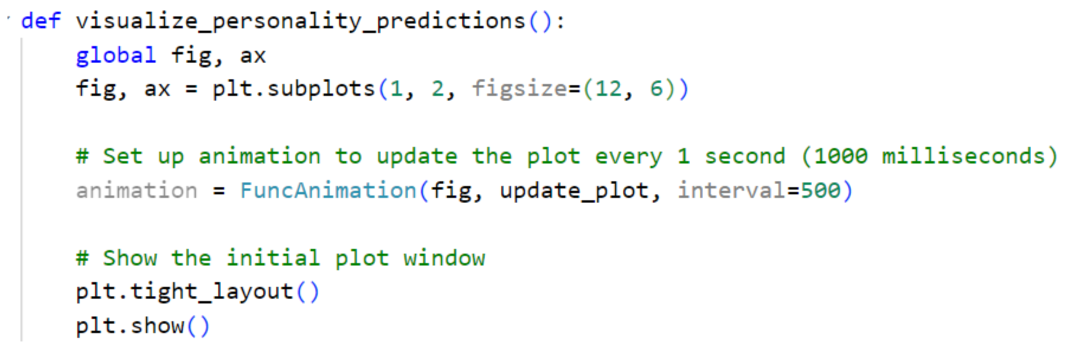
Figure 17: Facebook Post Scraper



```python
import matplotlib.pyplot as plt
import pandas as pd
from matplotlib.animation import FuncAnimation
import threading
import time
import random

# Global dictionary to store the counts of each personality type
personality_counts = dict()
User = "Mark Zuckerberg"  # Global user placeholder

def predict_personality_dynamically(Type):
    """Update personality counts based on new 'Type' received."""
    global personality_counts

    # Update the count for the given personality Type
    if Type in personality_counts:
        personality_counts[Type] += 1
    else:
        personality_counts[Type] = 1

    return personality_counts

def update_frame(Type):
    predict_personality_dynamically(Type)
    return True

def calculate_percentages(counts):
    """Calculate the percentages of each personality type based on total counts."""
    total = sum(counts.values())
    percentages = {ptype: (count / total) * 100 for ptype, count in counts.items()}
    return percentages

def update_plot(frame):
    if len(personality_counts) == 0:
        return
    global User
    counts = personality_counts

    # Calculate percentages based on the updated counts
    percentages = calculate_percentages(counts)

    personality_types = list(counts.keys())
    probabilities = list(percentages.values())
```



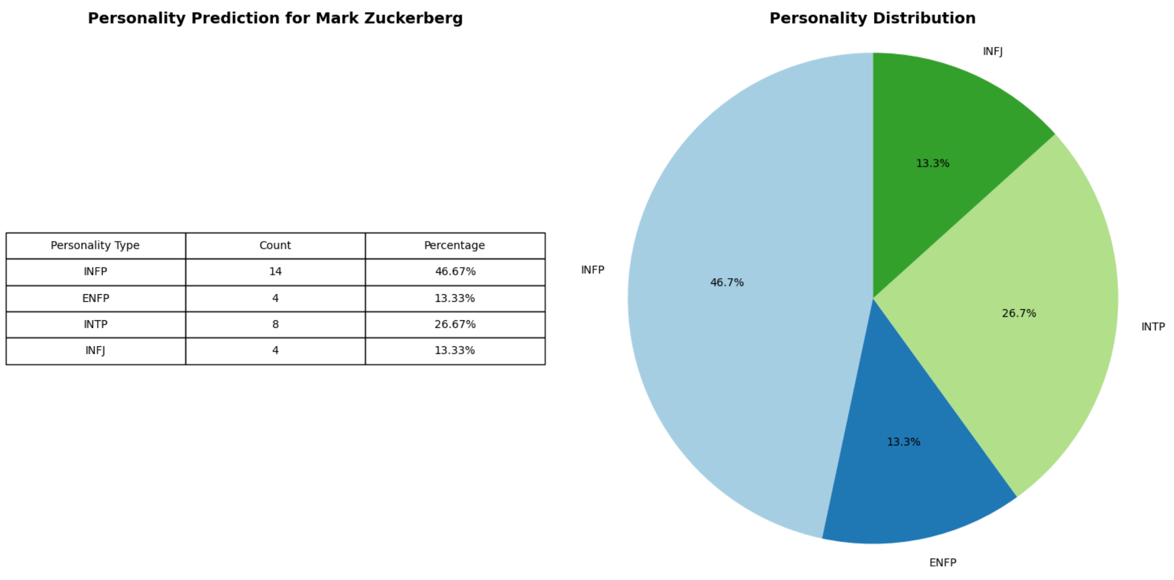
```python
    # Clear the previous axes to redraw updated figures
    ax[0].cla()  # Clear the table plot area
    ax[1].cla()  # Clear the pie chart plot area

    # Create a pandas DataFrame for the table
    df = pd.DataFrame({
        'Personality Type': personality_types,
        'Count': [counts[ptype] for ptype in personality_types],
        'Percentage': [f"{percentages[ptype]:.2f}%" for ptype in personality_types]
    })

    # Update the table with the new data
    ax[0].axis('off')
    ax[0].set_title(f'Personality Prediction for {User}', fontsize=14, fontweight='bold')
    table = ax[0].table(cellText=df.values,
                        colLabels=df.columns,
                        cellLoc='center',
                        loc='center')
    table.scale(1, 2)

    # Update the pie chart
    ax[1].pie(probabilities, labels=personality_types, autopct='%1.1f%%', startangle=90, colors=plt.cm.Paired.colors)
    ax[1].axis('equal')
    ax[1].set_title('Personality Distribution', fontsize=14, fontweight='bold')
```



```
def visualize_personality_predictions():
    global fig, ax
    fig, ax = plt.subplots(1, 2, figsize=(12, 6))

    # Set up animation to update the plot every 1 second (1000 milliseconds)
    animation = FuncAnimation(fig, update_plot, interval=500)

    # Show the initial plot window
    plt.tight_layout()
    plt.show()
```

Figure 18: Result Visualization

12.1 Final Output



Personality Prediction for Mark Zuckerberg

Personality Type	Count	Percentage
INFP	14	46.67%
ENFP	4	13.33%
INTP	8	26.67%
INFJ	4	13.33%

Figure 19: Final Personality classification result