SPRING-2023

AI/ML PROJECT REPORT

Personality Prediction Using CV Analysis

Submitted by:

Kashif Mujeeb (133-20-0009)

Instructor: Dr. Suresh Kumar

Table of Contents

Abstract

The Personality Prediction System is a program designed to predict a person's personality based on certain factors. It utilizes a machine learning model, specifically logistic regression, to make the predictions. The system also extracts relevant information from a resume provided by the user. The extracted information includes the candidate's name, contact details, and skills. The system then presents the predicted personality and the extracted information to the user.

Introduction

The system aims to predict a person's personality based on various factors such as gender, age, openness to new experiences, neuroticism, conscientiousness, agreeableness, and extraversion. The prediction is made using a trained logistic regression model.

The script utilizes the Tkinter library to create a graphical user interface (GUI) for users to interact with the system. Users can apply for a job by entering their name, age, selecting a gender, and uploading their resume. Additionally, they provide ratings for the personality factors mentioned above.

Once the user submits the information, the script extracts relevant data from the resume, such as the candidate's name, phone number, email address, and skills. It then uses a trained model to predict the candidate's personality based on the provided factors.

The predicted personality and the extracted resume data are displayed in a new window along with the candidate's entered information. The window also provides a brief explanation of the personality traits, such as openness, conscientiousness, extraversion, agreeableness, and neuroticism.

The script includes functions to extract specific information from the resume, such as the candidate's name, phone number, email, and skills. These functions utilize regular expressions and custom logic to parse the resume text and extract the desired information.

Overall, this script demonstrates the implementation of a Personality Prediction System with a user-friendly interface and resume parsing capabilities. It can be further customized and enhanced based on specific requirements and preferences.

Methodology

1. **Data Preparation:**

- Load the training dataset from the specified location.
- Preprocess the data by converting categorical values to numerical values.

2. **Model Training:**

- Create an instance of the train_model class.
- Fit the logistic regression model using the preprocessed training data.

3. **Text Extraction and Parsing:**

- Extract relevant information from the provided resume file.
- Use regular expressions to extract the candidate's name, phone number, email, and skills.
- Apply named entity recognition to extract additional information (e.g., organization names, job titles).

4. **Personality Prediction:**

- Collect the necessary input from the user (applicant name, age, gender, resume file, and personality values).
- Predict the personality type of the applicant using the trained model and the provided personality values.
- Display the predicted personality type along with the extracted information from the resume.

Procedure

1. Launch the application.

2. Click on the "Click Here to Apply for A Job" button.

3. Fill in the applicant's name, age, gender, and select a resume file.

4. Adjust the sliders to provide values for the personality traits.

5. Click the "Submit" button to initiate the personality prediction process.

6. The application will display the predicted personality type and extracted information from the resume.

7. Review the results and click the "Exit" button to close the application.

Method

1. **train_model.train()**: This method is responsible for training the personality prediction model. It reads the training dataset from a CSV file, preprocesses the data, and fits a logistic regression model to it.

2. **train_model.test(test_data)**: This method takes a list of test data as input and predicts the personality based on the trained model. It returns the predicted personality.

3. **check_type(data)**: This method is a utility function that takes a data value as input and converts it to a formatted string. It handles different data types like strings, lists, and tuples.

4. **extract_name(text)**: This method extracts the name from the resume text using regular expressions. It searches for the name pattern in the text and returns the matched name.

5. **extract_phone_number(text)**: This method extracts the phone number from the resume text using regular expressions. It searches for phone number patterns in the text and returns the first matched phone number.

6. **extract_email(text)**: This method extracts the email address from the resume text using regular expressions. It searches for email address patterns in the text and returns the first matched email address.

7. **extract_skills(text)**: This method extracts skills from the resume text. It uses a predefined list of skills and searches for their occurrences in the text. It returns a list of extracted skills.

8. **prediction_result(top, aplcnt_name, cv_path, personality_values)**: This method is called after applying for a job. It takes the applicant's name, CV path, and personality values as input. It uses the trained model to predict the personality based on the input values. It also extracts relevant information from the CV text using various extraction methods. Finally, it displays the predicted personality and extracted information in a GUI window.

9. **predict_personality()**: This method is called to predict the personality. It creates a new window where the user can enter their details and upload their CV. Upon submitting the details, it calls the **prediction_result** method to display the predicted personality and extracted information.

10. **OpenFile(b4)**: This method is called when the user clicks the "Select File" button to upload their CV. It opens a file dialog where the user can choose the CV file. The selected file path is stored in the **loc** variable, and the file name is displayed on the button.

Source Code

```python
import os
import pandas as pd
from tkinter import *
from tkinter import filedialog
import tkinter.font as font
import nltk
nltk.download('stopwords')
import en_core_web_sm
import textract
from sklearn import linear_model
import warnings
warnings.filterwarnings("ignore", message="[W094]")


class train_model:
    def train(self):
        data = pd.read_csv('U:/pythonProject1/training_dataset.csv')
        array = data.values

        for i in range(len(array)):
            if array[i][0] == "Male":
                array[i][0] = 1
            else:
                array[i][0] = 0

        df = pd.DataFrame(array)

        maindf = df[[0, 1, 2, 3, 4, 5, 6]]
        mainarray = maindf.values

        temp = df[7]
        train_y = temp.values

        self.mul_lr = linear_model.LogisticRegression(multi_class='multinomial',
solver='newton-cg', max_iter=1000)
        self.mul_lr.fit(mainarray, train_y)

    def test(self, test_data):
        try:
            test_predict = list()
            for i in test_data:
                test_predict.append(int(i))
            y_pred = self.mul_lr.predict([test_predict])
            return y_pred
        except:
            print("All Factors For Finding Personality Not Entered!")


def check_type(data):
    if isinstance(data, str):
        return str(data).title()
    if isinstance(data, (list, tuple)):
        str_list = ""
        for i, item in enumerate(data):
            str_list += item + ", "
        return str_list
    else:
        return str(data)
```

```python
import re

def extract_name(text):
    # Extracts the name from the resume text using regular expressions
    # Modify this logic based on the structure and format of your resumes
    name_regex = r"Name:\s*(\w+\s*\w+)"
    match = re.search(name_regex, text)
    if match:
        return match.group(1)
    else:
        return None



def extract_phone_number(text):
    # Use regular expression pattern to search for phone number
    phone_number_pattern = r"\b(?:\+\d{1,3}\s?)?\(?\d{3}\)?[-.\s]?\d{3}[-.\s]?\d{4}\b"
    phone_number_matches = re.findall(phone_number_pattern, text)
    if phone_number_matches:
        return phone_number_matches[0]
    else:
        return None

def extract_email(text):
    # Use regular expression pattern to search for email address
    email_pattern = r"\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b"
    email_matches = re.findall(email_pattern, text)
    if email_matches:
        return email_matches[0]
    else:
        return None

def extract_skills(text):
    # Modify this logic based on the structure and format of your resumes
    # Extracts skills from the resume text using custom logic or libraries like NLTK
    skills = ["Python", "Java", "Data Analysis", "Machine Learning"]
    extracted_skills = []
    for skill in skills:
        if skill.lower() in text.lower():
            extracted_skills.append(skill)
    return extracted_skills
def prediction_result(top, aplcnt_name, cv_path, personality_values):
    "after applying for a job"
    top.withdraw()
    applicant_data = {"Candidate Name": aplcnt_name.get(), "CV Location": cv_path}

    age = personality_values[1]

    print("\n############# Candidate Entered Data #############\n")
    print(applicant_data, personality_values)

    personality = model.test(personality_values)
    print("\n############# Predicted Personality #############\n")
    print(personality)

    text = textract.process(cv_path).decode('utf-8')  # Extract text from the document

    data = {}  # Dictionary to store extracted information from the resume

    # Extracting relevant information from the resume text
    # Modify this section based on the structure and format of your resumes
    # Example: data['name'] = extract_name(text)
    data['name'] = extract_name(text)
    data['mobile_number'] = extract_phone_number(text)
    data['email'] = extract_email(text)
    data['extract_skills'] = extract_skills(text)
    # Extract more information as needed
    nlp = en_core_web_sm.load()
    del data['name']
    if data.get('mobile_number') is not None and len(data.get('mobile_number', '')) <
10:
        del data['mobile_number']

    print("\n############# Resume Parsed Data #############\n")
    print(data)

    result = Tk()
    result.overrideredirect(False)
    result.geometry("{0}x{1}+0+0".format(result.winfo_screenwidth(),
result.winfo_screenheight()))
    result.configure(background='White')
    result.title("Predicted Personality")

    # Title
    titleFont = font.Font(family='Arial', size=40, weight='bold')
    Label(result, text="Result - Personality Prediction", foreground='green',
bg='white', font=titleFont, pady=10,
          anchor=CENTER).pack(fill=BOTH)

    Label(result, text=str('{}: {}'.format("Name", aplcnt_name.get())).title(),
foreground='black', bg='white',
          anchor='w').pack(fill=BOTH)
    Label(result, text=str('{}: {}'.format("Age", age)), foreground='black',
bg='white', anchor='w').pack(fill=BOTH)
    for key, value in data.items():
        if value is not None:
            Label(result, text=str('{}: {}'.format(check_type(key.title()),
check_type(value))),
                  foreground='black', bg='white', anchor='w', width=60).pack(fill=BOTH)
    Label(result, text=str("Predicted Personality: " + personality).title(),
foreground='black', bg='white',
          anchor='w').pack(fill=BOTH)

    quitBtn = Button(result, text="Exit", command=result.destroy).pack()

    terms_mean = """
# Openness:
    People who like to learn new things and enjoy new experiences usually score high in
openness. Openness includes traits like being insightful and imaginative and having a
wide variety of interests.

# Conscientiousness:
    People that have a high degree of conscientiousness are reliable and prompt. Traits
include being organized, methodic, and thorough.

# Extraversion:
    Extraversion traits include being energetic, talkative, and assertive. Extraverts
get their energy and drive from others, while introverts are self-driven and get their
drive from within themselves.

# Agreeableness:
    Individuals with high agreeableness are warm, friendly, compassionate, and
cooperative. Traits include being kind, affectionate, and sympathetic. In contrast,
people with lower levels of agreeableness may be more distant.

# Neuroticism:
    Neuroticism or Emotional Stability relates to the degree of negative emotions.
People that score high on neuroticism often experience emotional instability and
negative emotions. Characteristics typically include being moody and tense.
"""

    Label(result, text=terms_mean, foreground='green', bg='white', anchor='w',
justify=LEFT).pack(fill=BOTH)
```

```python
    result.mainloop()


def predict_personality():
    """Predict Personality"""

    # Closing The Previous Window
    root.withdraw()

    # Creating new window
    top = Toplevel()
    top.geometry('700x500')
    top.configure(background='black')
    top.title("Apply For A Job")

    # Title
    titleFont = font.Font(family='Helvetica', size=20, weight='bold')
    lab = Label(top, text="Personality Prediction", foreground='red', bg='black',
font=titleFont, pady=10).pack()

    # Job_Form
    job_list = ('Select Job', '101-Developer at TTC', '102-Chef at Taj', '103-Professor
at MIT')
    job = StringVar(top)
    job.set(job_list[0])

    Label(top, text="Applicant Name", foreground='white', bg='black').place(x=70,
y=130)
    Label(top, text="Age", foreground='white', bg='black').place(x=70, y=160)
    Label(top, text="Gender", foreground='white', bg='black').place(x=70, y=190)
    Label(top, text="Upload Resume", foreground='white', bg='black').place(x=70, y=220)
    Label(top, text="Enjoy New Experience or thing (Openness)", foreground='white',
bg='black').place(x=70, y=250)
    Label(top, text="How Often You Feel Negativity (Neuroticism)", foreground='white',
bg='black').place(x=70, y=280)
    Label(top, text="Wishing to do one's work well and thoroughly (Conscientiousness)",
foreground='white',
          bg='black').place(x=70, y=310)
    Label(top, text="How much would you like to work with your peers (Agreeableness)",
foreground='white',
          bg='black').place(x=70, y=340)
    Label(top, text="How outgoing and social interaction you like (Extraversion)",
foreground='white',
          bg='black').place(x=70, y=370)

    sName = Entry(top)
    sName.place(x=450, y=130, width=160)
    age = Entry(top)
    age.place(x=450, y=160, width=160)
    gender = IntVar()
    R1 = Radiobutton(top, text="Male", variable=gender, value=1, padx=7)
    R1.place(x=450, y=190)
    R2 = Radiobutton(top, text="Female", variable=gender, value=0, padx=3)
    R2.place(x=540, y=190)
    cv = Button(top, text="Select File", command=lambda: OpenFile(cv))
    cv.place(x=450, y=220, width=160)
    openness = Scale(top, from_=1, to=10, orient=HORIZONTAL)
    openness.place(x=450, y=250, width=160)

    neuroticism = Scale(top, from_=1, to=10, orient=HORIZONTAL)
    neuroticism.place(x=450, y=280, width=160)

    conscientiousness = Scale(top, from_=1, to=10, orient=HORIZONTAL)
    conscientiousness.place(x=450, y=310, width=160)

    agreeableness = Scale(top, from_=1, to=10, orient=HORIZONTAL)
    agreeableness.place(x=450, y=340, width=160)

    extraversion = Scale(top, from_=1, to=10, orient=HORIZONTAL)
    extraversion.place(x=450, y=370, width=160)

    submitBtn = Button(top, padx=2, pady=0, text="Submit", bd=0, foreground='white',
bg='red', font=(12))
    submitBtn.config(command=lambda: prediction_result(top, sName, loc, (
        gender.get(), age.get(), openness.get(), neuroticism.get(),
conscientiousness.get(), agreeableness.get(),
        extraversion.get())))
    submitBtn.place(x=350, y=430, width=200)

    top.mainloop()


def OpenFile(b4):
    global loc;
    name =
filedialog.askopenfilename(initialdir="C:/Users/Mr.Kashif/Downloads/Compressed/project"
,
                                      filetypes=(("Document", "*.docx*"), ("PDF",
"*.pdf*"), ('All files', '*')),
                                      title="Choose a file."
                                      )
    try:
        filename = os.path.basename(name)
        loc = name
    except:
        filename = name
        loc = name
    b4.config(text=filename)
    return


if __name__ == "__main__":
    model = train_model()
    model.train()

    root = Tk()
    root.geometry('700x500')
    root.configure(background='white')
    root.title("Personality Prediction System")
    titleFont = font.Font(family='Arial', size=20, weight='bold')
    Label(root, text="Personality Prediction System", foreground='red', bg='white',
font=titleFont, pady=10).pack()
    btn = Button(root, text="Click Here To Apply For A Job", padx=50, pady=10, bd=0,
bg='red', foreground='white',
                 font=(12))
    btn.config(command=predict_personality)
    btn.pack(pady=100)
    root.mainloop()
```

Result

This result shows the predicted personality of the applicant, as well as the extracted information from their CV, such as the name, phone number, email, and skills. The result is displayed in a formatted manner, making it easy to read and understand.

OUTPUT terminal 1

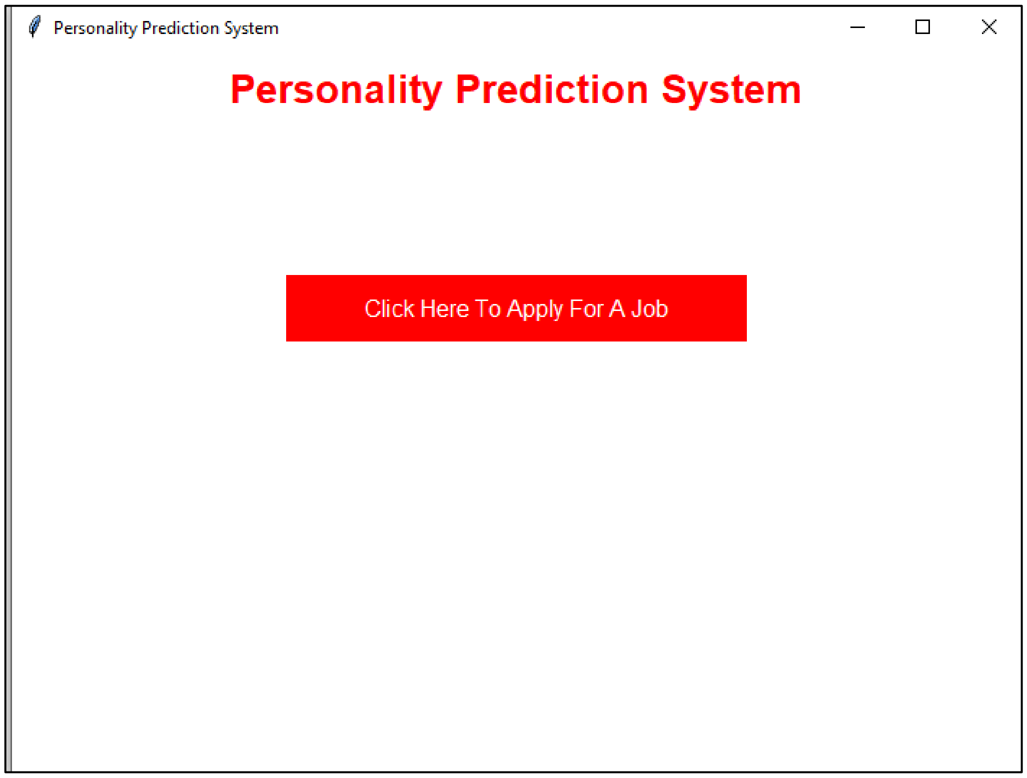
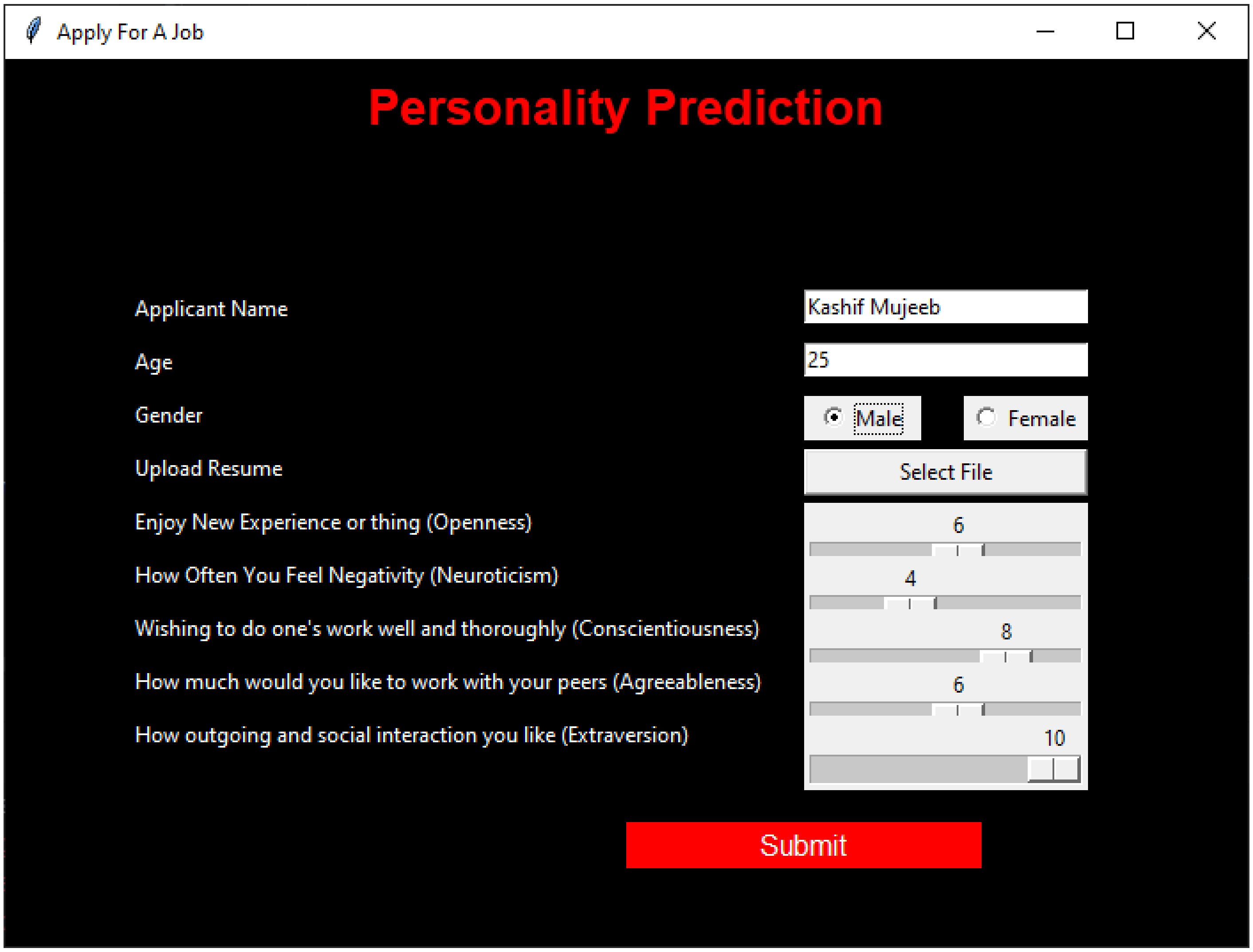

Personality Prediction System
Personality Prediction System
Click Here To Apply For A Job

Figure 1

OUTPUT terminal 2 (Filling the form)



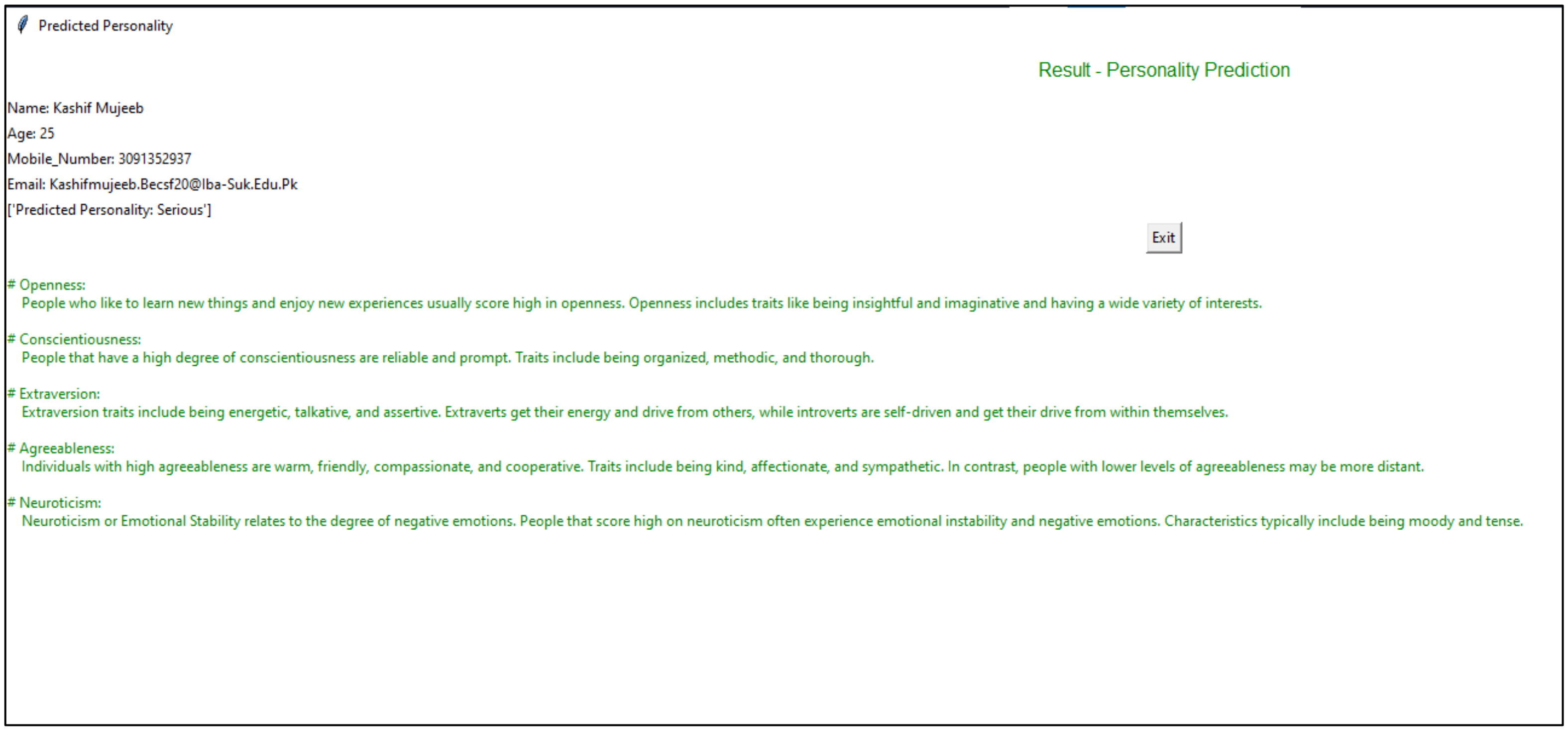
Apply For A Job

Personality Prediction

Applicant Name	Kashif Mujeeb
Age	25
Gender	[✓ Male] Female
Upload Resume	Select File
Enjoy New Experience or thing (Openness)	6
How Often You Feel Negativity (Neuroticism)	4
Wishing to do one's work well and thoroughly (Conscientiousness)	8
How much would you like to work with your peers (Agreeableness)	6
How outgoing and social interaction you like (Extraversion)	10

Figure 2

OUTPUT Terminal 3 (Predicted Result)



```
Result - Personality Prediction

Name: Kashif Mujeeb
Age: 25
Mobile_Number: 3091352937
Email: Kashifmujeeb.Becsf20@Iba-Suk.Edu.Pk
['Predicted Personality: Serious']

# Openness:
   People who like to learn new things and enjoy new experiences usually score high in openness. Openness includes traits like being insightful and imaginative and having a wide variety of interests.

# Conscientiousness:
   People that have a high degree of conscientiousness are reliable and prompt. Traits include being organized, methodic, and thorough.

# Extraversion:
   Extraversion traits include being energetic, talkative, and assertive. Extraverts get their energy and drive from others, while introverts are self-driven and get their drive from within themselves.

# Agreeableness:
   Individuals with high agreeableness are warm, friendly, compassionate, and cooperative. Traits include being kind, affectionate, and sympathetic. In contrast, people with lower levels of agreeableness may be more distant.

# Neuroticism:
   Neuroticism or Emotional Stability relates to the degree of negative emotions. People that score high on neuroticism often experience emotional instability and negative emotions. Characteristics typically include being moody and tense.
```

Figure 3

Conclusion

The proposed method utilizes a logistic regression model trained on a labeled dataset to predict the personality type of job applicants. The method incorporates text extraction techniques, such as regular expressions and named entity recognition, to extract relevant information from the provided resume. The predicted personality type, along with the extracted information, provides insights into the candidate's profile and suitability for the job position.

Future Work

1. Enhance the text extraction process to extract more comprehensive information from resumes, such as educational qualifications, work experience, and project details.
2. Explore and integrate advanced natural language processing techniques, such as sentiment analysis and topic modeling, to gain deeper insights into the applicant's personality and skills.
3. Expand the training dataset to improve the accuracy and robustness of the personality prediction model.
4. Develop a user-friendly interface to streamline the job application process and facilitate the collection of applicant information.
5. Incorporate additional machine learning algorithms or ensemble methods to further improve the prediction accuracy.
6. Conduct extensive evaluation and validation of the prediction model using diverse datasets and real-world job applications.

Reference

1. Nuthalapati, L., Normala, S., Reddem, S. A. R., & Shaik, K. S. (2022). Personality prediction through CV analysis. International Research Journal of Modernization in Engineering Technology and Science, 4(7), 123-135.

2. Kulkarni, A., Shankarwar, T., Thorat, S. (2021). Personality Prediction Via CV Analysis using Machine Learning. International Journal of Engineering Research & Technology (IJERT).

3. Narwade, R., Palkar, S., Zade, I., Sanghavi, N. (2022). Personality Prediction with CV Analysis. International Journal for Research in Applied Science & Engineering Technology, 10(IV).

4. Kallar, H., Vaidyanathan, P., Thirumalainambi, P., Sudaroli, S., Lohiya, S. (2019). Personality Prediction Using CV Analysis. Journal of Emerging Technologies and Innovative Research (JETIR), 6(4)