TABLE OF CONTENTS

TITLE **PAGE NO.**

OBJECTIVES

This system can be used in many business parts/areas that may require expert candidates. This system will reduce the workload of the (workers in general/hiring, training, and firing department). This system will help the (related to workers in general) to select the right candidate for the desired job profile, which in turn provide the expert (all the workers in a company or country) for the organization. Admin can easily shortlist a candidate based on their personality scores and select the appropriate candidate for a particular job profile.

Using Natural Language Processing (NLP) can be defined as a process that enables a machine to become more like a human, because of this deeply cutting the distance between machines and humans. This system will focus not only on qualification and inexperience but also focuses on other important aspects, which are needed/demanded for a particular job position. Admin can store the data in excel sheet for further comparison and sorting of data.

BACKGROUND OF PERSONALITY PERCEPTION

The Big Five Personality Traits model is based on findings from several independent researchers, and it dates back to the late 1950s. But the model as we know it now began to take shape in the 1990s.

Lewis Goldberg, a researcher at the Oregon Research Institute, is credited with naming the model "The Big Five." It is now considered to be an accurate and respected personality scale, which is routinely used by businesses and in psychological research.

The Big Five Personality Traits Model measures five key dimensions of people's personalities:

Openness: sometimes called "Intellect" or "Imagination," this measures your level of creativity, and your desire for knowledge and new experiences.

Conscientiousness: this looks at the level of care that you take in your life and work. If you score highly in conscientiousness, you'll likely be organized and thorough, and know how to make plans and follow them through. If you score low, you'll likely be lax and disorganized.

Extraversion/Introversion: this dimension measures your level of sociability. Are you outgoing or quiet, for instance? Do you draw energy from a crowd, or do you find it difficult to work and communicate with other people?

Agreeableness: this dimension measures how well you get on with other people. Are you considerate, helpful and willing to compromise? Or do you tend to put your needs before others'?

Natural Reactions: sometimes called "Emotional Stability" or "Neuroticism," this measure emotional reactions. Do you react negatively or calmly to bad news? Do you worry obsessively about small details, or are you relaxed in stressful situations?

DEPENDENCIES OF SYSTEM

Python Modules/Libraries:

1. **OS**: For accessing the files and data from internal storage.
2. **Pandas**: For accessing and manipulating datasheets.
3. **Numpy**: For working on arrays and other data manipulation.
4. **Tkinter**: For building the GUI.
5. **Functools**: Tools for Manipulating Functions. Purpose: Functions that operate on other functions.
6. **Pyresparser**: Module for extracting information from resume.
7. **Sklearn**: It features various classification, regression and clustering algorithms. We used sklearn to make the model learn on various characteristic values using logical regression.

DESCRIPTION

The system built in this project predicts personality of peoples by using their gender, age, score of openness, conscientiousness, extraversion, agreeableness, neuroticism and experience. It parses all the data from CV/resume and on the result page, it shows all the information from the entered data and uploaded resume. This system uses *logistic regression* for training the model and *pyresparser* module for parsing the information from resume which is built using *nltk* and *spaCy* module in python.

Description of Methods and Flow in the System:

1. ***train_model class***: It contains two method which train the model and predict the result by giving the various values.
    a. ***train method*:** It read the dataset for training the model from a csv file and build a model using *Logistic Regression.* It uses different 7 values for training the model.

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

self.mul_lr.fit(mainarray, train_y)
```

    b. ***test method:*** It predict the personality of a person by passing an array of values that contains gender, age and other 5 personality characteristics.

```
test_predict=list()
for i in test_data:
    test_predict.append(int(i))
y_pred = self.mul_lr.predict([test_predict])
return y_pred
```

2. ***main method***: We start with creating an object of *train_model* class and train the model by calling *train* method of class. Then we initialize a variable with *Tk* object and design the landing page of system using labels and button. A button with name *Predict Personality* is designed which calls *predict_person* method.

```
if __name__ == "__main__":
    '''initialize system with training model'''
    model = train_model()
    model.train()

    root = Tk()
    root.geometry('700x500')
    root.configure(background='white')
    root.title("Personality Prediction System")
    titleFont = font.Font(family='Helvetica', size=25, weight='bold')
    homeBtnFont = font.Font(size=12, weight='bold')
    lab=Label(root, text="Personality Prediction System", bg='white', font=titleFont, pady=30).pack()
    b2=Button(root, padx=4, pady=4, width=30, text="Predict Personality", bg='black', foreground='white',
    root.mainloop()
```

3. ***predict_person method***: We withdraw the root tkinter window and create a new toplevel window and configure its size and attributes. We label the heading of window followed by various labels and their entries. For selecting of a resume file, user needs to press choose file button which then calls *Openfile* method that takes an argument of button. In *predict_person* method, various entries are taken for predicting the personality. Submit button pass all the values to prediction_result.

4. ***OpenFile method***: It tries to open the directory with default address name and file types and except if file not chosen. After try except block, the method changes the name of choose file button in *predict_person* method with the base name of file so that user can know about the chosen file.

```
name = filedialog.askopenfilename(initialdir="C:/Users/Batman/Documents/Programming/tkinter/",
                       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
```

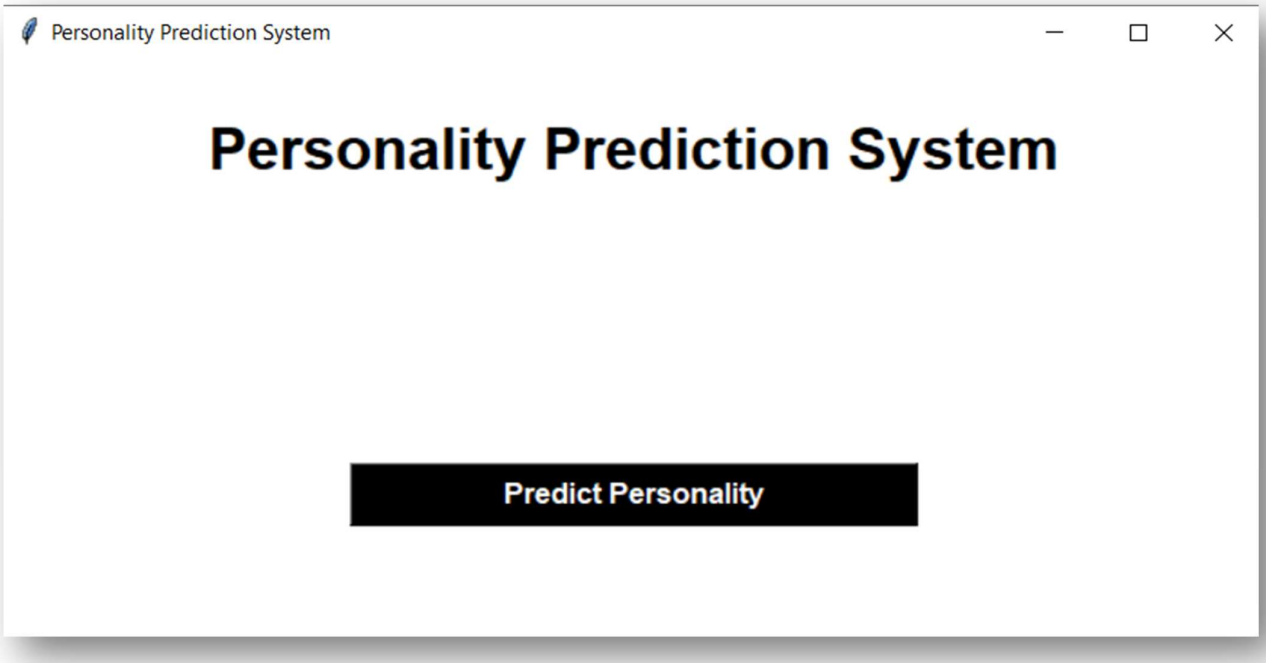
5. ***prediction_result method:*** This method firstly closes the previous tkinter window which was used to take the data from user. After this, it calls *test* method of model object and stores the result returned by method. After this it parse all the information from resume and stores in a variable followed by a try except block which try to delete name and validate mobile number from fetched information from resume. Then it prints all the data submitted by user on console. After this, the method popup a full screen window which shows all the parsed information and predicted personality on GUI window along with the definition of each personality characteristic's definition.

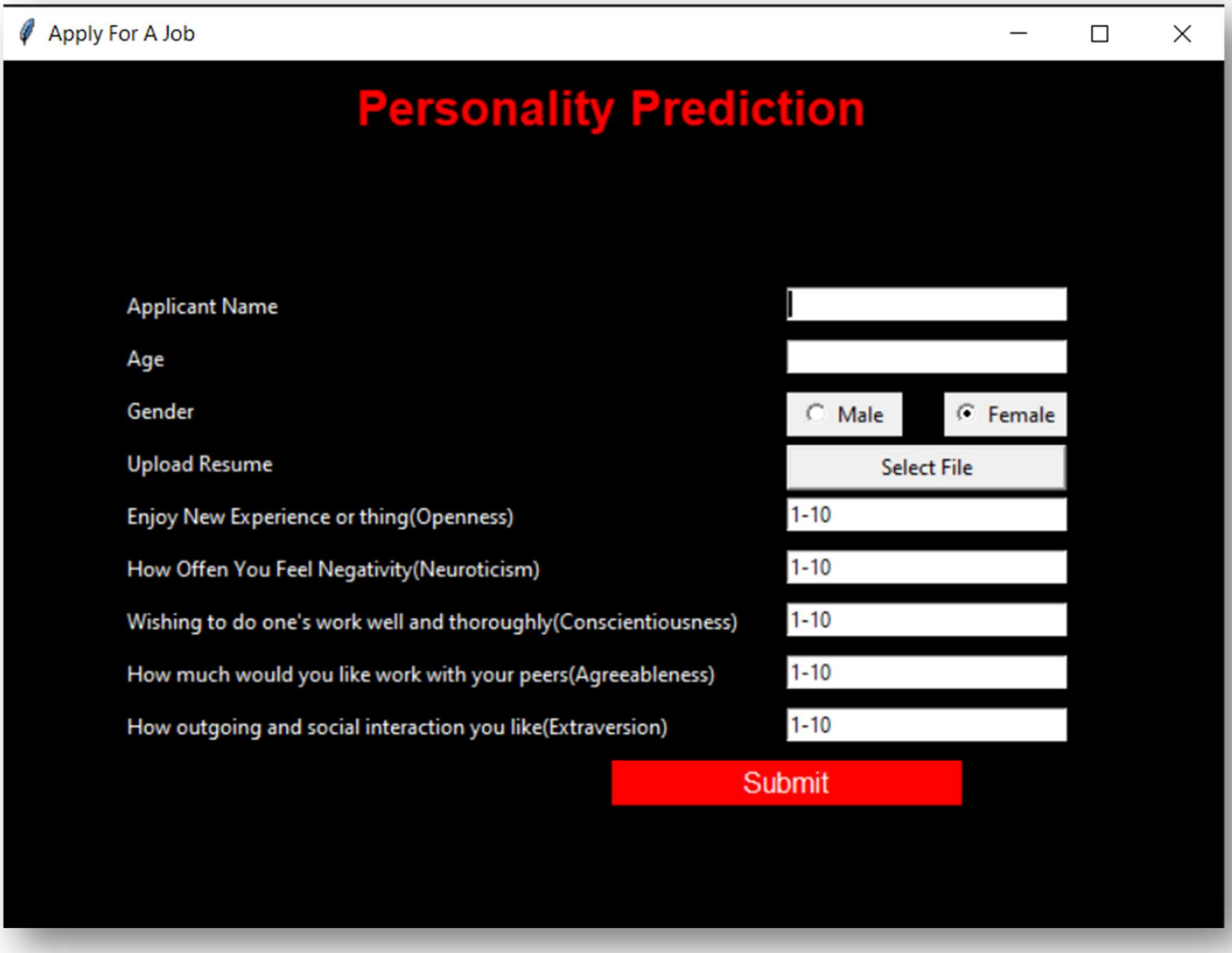
6. ***check_type method:*** It converts various strings and numbers into desired format and converts lists and tuples in string.

```
def check_type(data):
    """Check datatype of string and convert and return"""
    if type(data)==str or type(data)==str:
        return str(data).title()
    if type(data)==list or type(data)==tuple:
        str_list=""
        for i,item in enumerate(data):
            str_list+=item+", "
        return str_list
    else:   return str(data)
```

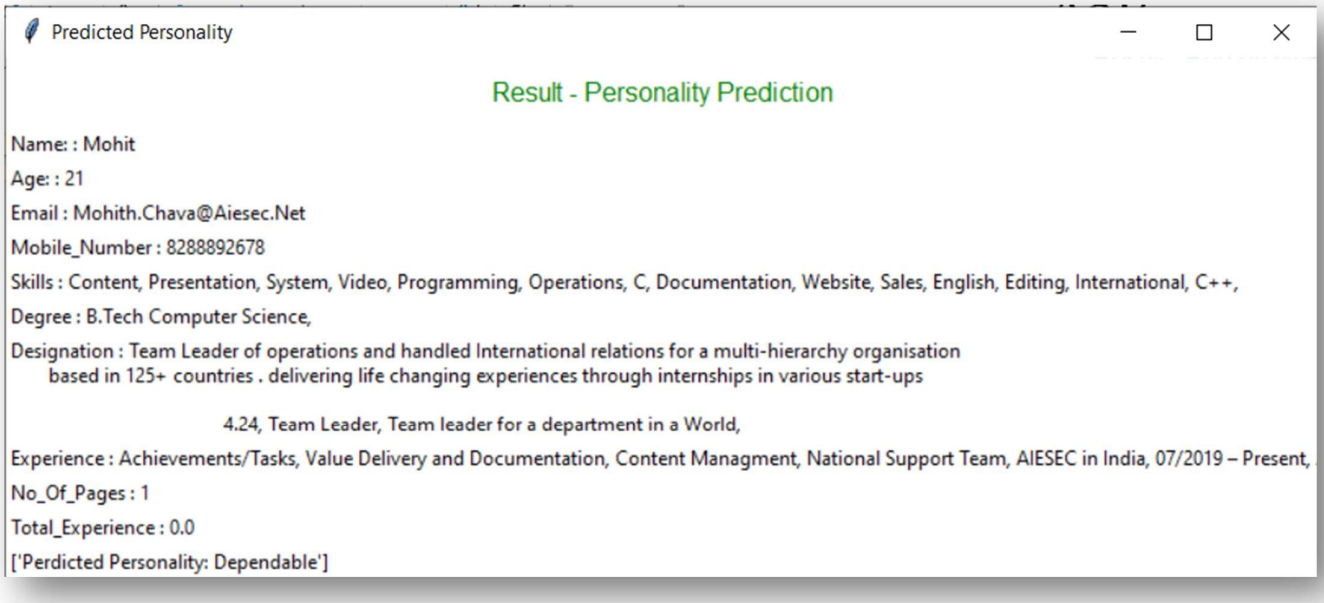
IMPLEMENTATION

On landing page, *'Predict Personality'* button pops up a new window for taking various inputs from user and submit it prediction model which will predict the personality.





On result page, all the manipulated information and predicted result will be displayed



Division of Work

Hitesh - train_model class and prediction_result method.
I have used sklearn module for training the model that can predict the personality such as serious, lively, responsible, extraverted, dependable etc. The model learns from age, gender, openness, extraverted and other personality characteristics. The model uses logistic regression approach. Model trained from dataset of more than 700 people and predict the personality when a list of required values is passed to *test* function of class train_model. In prediction_result method, number of input values are taken from predict_person and it manipulate the data and calls the test function for predicting result and parse all the information from resume using pyresparser. After this, all raw values are printed on console and information and results printed to a Tkinter GUI window.
Shagun – predict_person method and OpenFile method
In predict_person method, I have created a GUI window on various labels, entries, radio buttons and buttons are created. The function takes name, age, resume file and five various personality characteristics values. Choose file button fetch the location of resume by calling OpenFile method. OpenFile method tries to open the directory and select the file and returns the location to a variable. After entering all the values by user, on clicking submit button, the GUI window closes, and all the collected data passed to prediction_result method.
Nitin –main method, check_type method and contribution in report making
In main method, first I create a instance of class train_model and calls the method of class for training the model. Then a GUI windows created using Tk method from tkinter module which contains a button which take us to data entering method. In check type method, various operations are designed for string manipulation. It Capitalize the string using title function of strings, convert tuple and list to string and returns the manipulated data. It also converts the numbers and other data types to string and returns.

CODE

```
import os
import pandas as pd
import numpy as np
from tkinter import *
from tkinter import filedialog
import tkinter.font as font
from functools import partial
from pyresparser import ResumeParser
from sklearn import datasets, linear_model

class train_model:

    def train(self):
        data =pd.read_csv('train 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 type(data)==str or type(data)==str:
        return str(data).title()
    if type(data)==list or type(data)==tuple:
        str_list=""
        for i,item in enumerate(data):
            str_list+=item+", "
        return str_list
    else:   return str(data)

def prediction_result(top, aplcnt_name, cv_path, personality_values):
    "after applying 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)
    data = ResumeParser(cv_path).get_extracted_data()

    try:
        del data['name']
        if len(data['mobile_number'])<10:
            del data['mobile_number']
    except:
        pass

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

    for key in data.keys():
        if data[key] is not None:
            print('{} : {}'.format(key,data[key]))

    result=Tk()
  #  result.geometry('700x550')
    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 in data.keys():
        if data[key] is not None:
            Label(result, text = str('{} : {}'.format(check_type(key.title()),
                  check_type(data[key]))), foreground='black',bg='white',
                              anchor='w', width=60).pack(fill=BOTH)
    Label(result, text = str("perdicted personality: "+personality).title(),
          foreground='black', bg='white', anchor='w').pack(fill=BOTH)

    quitBtn = Button(result, text="Exit", command =lambda:  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 organised, methodic, and thorough.

    # Extraversion:
    Extraversion traits include being; energetic, talkative, and assertive (sometime seen
    as outspoken by Introverts). Extraverts get their energy and drive from others, while
    introverts are self-driven get their drive from within themselves.

    # Agreeableness:
    As it perhaps sounds, these individuals are warm, friendly, compassionate and
    cooperative and 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 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)

    result.mainloop()


def perdict_person():
    """Predict Personality"""

    # Closing The Previous Window
    root.withdraw()
```

```
    # Creating new window
    top = Toplevel()
    top.geometry('500x500')
    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])

    l1=Label(top, text="Applicant Name", foreground='white', bg='black')
    l1.place(x=70, y=130)
    l2=Label(top, text="Age", foreground='white', bg='black').place(x=70, y=160)
    l3=Label(top, text="Gender", foreground='white', bg='black').place(x=70, y=190)
    l4=Label(top, text="Upload Resume", foreground='white', bg='black').place(x=70, y=220)
    l5=Label(top, text="Openness", foreground='white', bg='black').place(x=70, y=250)
    l6=Label(top, text="Neuroticism", foreground='white', bg='black').place(x=70, y=280)
    l7=Label(top, text="Conscientiousness", foreground='white', bg='black')
    l7.place(x=70, y=310)
    l8=Label(top, text="Agreeableness", foreground='white', bg='black').place(x=70, y=340)
    l9=Label(top, text="Extraversion", foreground='white', bg='black').place(x=70, y=370)

    sName=Entry(top)
    sName.place(x=300, y=130, width=160)
    age=Entry(top)
    age.place(x=300, y=160, width=160)
    gender = IntVar()
    R1 = Radiobutton(top, text="Male", variable=gender, value=1, padx=7)
    R1.place(x=300, y=190)
    R2 = Radiobutton(top, text="Female", variable=gender, value=0, padx=3)
    R2.place(x=390, y=190)
    cv=Button(top, text="Select File", command=lambda:  OpenFile(cv))
    cv.place(x=300, y=220, width=160)
    openness=Entry(top)
    openness.place(x=300, y=250, width=160)
    neuroticism=Entry(top)
    neuroticism.place(x=300, y=280, width=160)
    conscientiousness=Entry(top)
    conscientiousness.place(x=300, y=310, width=160)
    agreeableness=Entry(top)
    agreeableness.place(x=300, y=340, width=160)
    extraversion=Entry(top)
    extraversion.place(x=300, 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=150, y=400, width=200)


    top.mainloop()

def OpenFile(b4):
    global loc;
    name = filedialog.askopenfilename(
              initialdir="C:/Users/Batman/Documents/Programming/tkinter/",
              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='Helvetica', size=25, weight='bold')
    homeBtnFont = font.Font(size=12, weight='bold')
    lab=Label(root, text="Personality Prediction System", bg='white',
              font=titleFont, pady=30).pack()
    b2=Button(root, padx=4, pady=4, width=30, text="Predict Personality",
              bg='black', foreground='white', bd=1, font=homeBtnFont,
              command=perdict_person).place(relx=0.5, rely=0.5, anchor=CENTER)
    root.mainloop()
```

PREDICTED PERSONALITY ON VARIOUS TEST CASES