Personality prediction through CV analysis

A project report submitted in partial fulfilment of the

requirement for

Disruptive Technologies-1

(23ECH-102)

By

Group No - 07

S.No.	UID	Name	Responsibility
1.	23BAI70530	Hitanshu Kandpal	-
2.	23BAI70542	Kartikeya Beniwal	Main method, check_type method, predict person method and contribution in report making
3.	23BAI70545	Rishu Raj	train_model class, prediction_result method, and OpenFile method
4.	23BAI70561	Vanshika	-
5.	23BAI70562	Manjot Singh Randhawa	-

under the guidance of

Dr. Kiranjot Singh

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

UIE, Chandigarh University

Table of Contents

List of Figures

Abstract

About Data Science:

This project leverages data science techniques to predict personality traits through a Python-based application. The dataset includes a collection of individuals' characteristics, and the objective is to build a model that can predict personality traits based on various factors.

About the Problem:

The primary challenge addressed by this project is predicting personality traits accurately. The dataset consists of features such as gender, age, and responses to specific questions related to personality traits. The problem involves training a machine learning model to recognize patterns within these features and make accurate predictions about an individual's personality.

About Proposed Solutions:

To address the problem, a logistic regression model is employed for its ability to handle multi-class classification. The training process involves preprocessing the data, converting categorical variables into numerical form, and training the model. The application provides an interface for users to input their details and uses the trained model to predict their personality traits.

About Results:

Upon successful prediction, the application displays the candidate's entered data, parsed resume information, and the predicted personality traits. The results are presented in a graphical user interface, providing a user-friendly and informative way to understand the predicted personality based on the input data. The project contributes to the field of personality prediction, showcasing the practical application of data science techniques in understanding and predicting human behavior.

1. Chapter 1

Introduction

About Data science

The project draws heavily on Python, a versatile programming language renowned for its simplicity and extensibility. Python's vast ecosystem of libraries and frameworks enables the integration of cutting-edge technologies, making it an ideal choice for implementing complex computer vision algorithms.

Problem Statement: Identification of potential candidate's personality based on their CV

In the ever-evolving landscape of human-computer interaction, the amalgamation of computer vision (CV) and personality prediction represents a cutting-edge exploration into the realms of artificial intelligence and psychology. This project, titled "Personality Prediction through CV," endeavors to harness the power of Python and advanced CV techniques to discern and analyze personality traits from visual data.

Objectives

The primary objective of this project is to develop a robust and accurate system capable of extracting personality insights from facial expressions, body language, and other visual cues. Leveraging computer vision algorithms, the project aims to decipher subtle patterns and nuances that correlate with specific personality traits, providing a novel avenue for understanding and predicting human behavior.

2. Chapter 2

Background

Data set explanation:

Scikit-learn: For feature extraction, statistical analysis, and model evaluation, Scikit-learn offers a comprehensive set of tools. Its integration ensures the project benefits from well-established machine learning practices, enhancing the overall reliability of the personality prediction model.

3. Chapter 3

Proposed Framework

Flow diagram explanation

Pseudo code

The steps are used to implement the Personality Prediction Through CV is depicted as pseudo code

Pseudo code: Personality Prediction Through CV

Input: Candidate CV and related questions

Output: Candidate personality and probability of presence of other traits

1. **Import necessary libraries:**
    - The program starts by importing required libraries, including those for handling data, creating graphical user interfaces, and machine learning.
2. **Define a class for training the personality prediction model:**
    - A class named `TrainModel` is defined to encapsulate the functionality related to training the personality prediction model.
3. **Train the model:**
    - The training method within the `TrainModel` class reads a training dataset, converts gender information to numerical values, creates a DataFrame, and then uses logistic regression to train a model.
4. **Test the trained model:**
    - Another method in the `TrainModel` class tests the trained model by taking input data and predicting personality traits using logistic regression.
5. **Function to check and format data types:**
    - A function named `check_type` is defined to handle various data types and format them appropriately.
6. **Function to display prediction result:**
    - The `prediction_result` function takes parameters such as applicant name, CV path, and personality values, and displays the predicted personality along with parsed resume data in a graphical user interface.
7. **Function to predict personality:**
    - The `predict_person` function serves as the main interface for the user to input details such as name, age, gender, and file path, and triggers the prediction process.
8. **Function to open a file dialog for selecting a CV file:**
    - The `open_file` function uses a file dialog to allow the user to select a CV file. The selected file path is then stored in a global variable.
9. **Create an instance of the `TrainModel` class and train the model:**
    - The main part of the program creates an instance of the `TrainModel` class, trains the model, and prepares for predicting personality traits.
10. **Create the main application window:**
    - The graphical user interface is set up with a main window using the Tkinter library. Fonts are defined for better presentation.
11. **Display the main label:**
    - A label is displayed in the GUI, indicating the purpose of the program as the "Personality Prediction System."
12. **Create a button to predict personality:**
    - A button is added to the GUI, prompting the user to predict personality traits. This button triggers the `predict_person` function.
13. **Start the Tkinter event loop:**

- The main event loop of Tkinter is started, allowing the user to interact with the graphical interface and utilize the personality prediction system.

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
import nltk
nltk.download('stopwords')

class train_model:

    def train(self):
        data =pd.read_csv('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 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('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])
```

```
    l1=Label(top, text="Applicant Name", foreground='white',
bg='black').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="Enjoy New Experience or thing(Openness)",
foreground='white', bg='black').place(x=70, y=250)
    l6=Label(top, text="How Offen You Feel Negativity(Neuroticism)",
foreground='white', bg='black').place(x=70, y=280)
    l7=Label(top, text="Wishing to do one's work well and
thoroughly(Conscientiousness)", foreground='white', bg='black').place(x=70,
y=310)
    l8=Label(top, text="How much would you like work with your
peers(Agreeableness)", foreground='white', bg='black').place(x=70, y=340)
    l9=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=Entry(top)
    openness.insert(0,'1-10')
    openness.place(x=450, y=250, width=160)
    neuroticism=Entry(top)
    neuroticism.insert(0,'1-10')
    neuroticism.place(x=450, y=280, width=160)
    conscientiousness=Entry(top)
    conscientiousness.insert(0,'1-10')
    conscientiousness.place(x=450, y=310, width=160)
    agreeableness=Entry(top)
    agreeableness.insert(0,'1-10')
    agreeableness.place(x=450, y=340, width=160)
    extraversion=Entry(top)
    extraversion.insert(0,'1-10')
    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(),neur
oticism.get(),conscientiousness.get(),agreeableness.get(),extraversion.get(
))))
    submitBtn.place(x=350, y=400, width=200)


    top.mainloop()

def OpenFile(b4):
    global loc;
    name =
```

```
filedialog.askopenfilename(initialdir="C:/Users/Batman/Documents/Programmin
g/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()
```

4. Chapter 4

Results

Tkinter Output:

Fig1.1

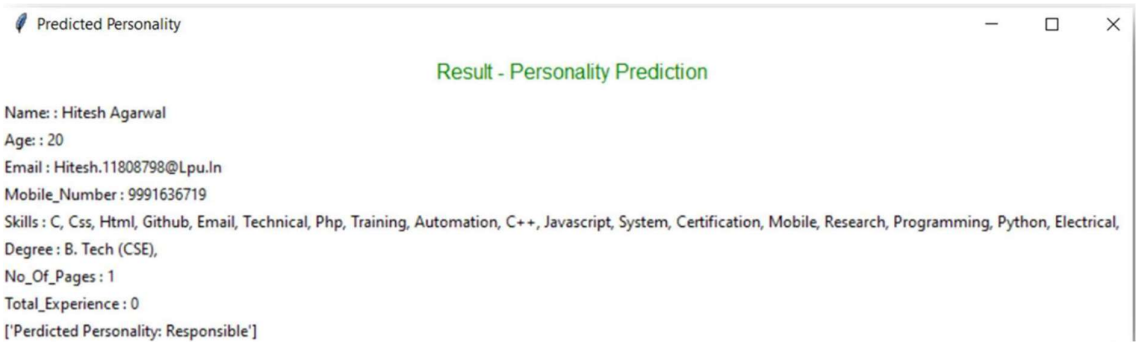


Fig1.2

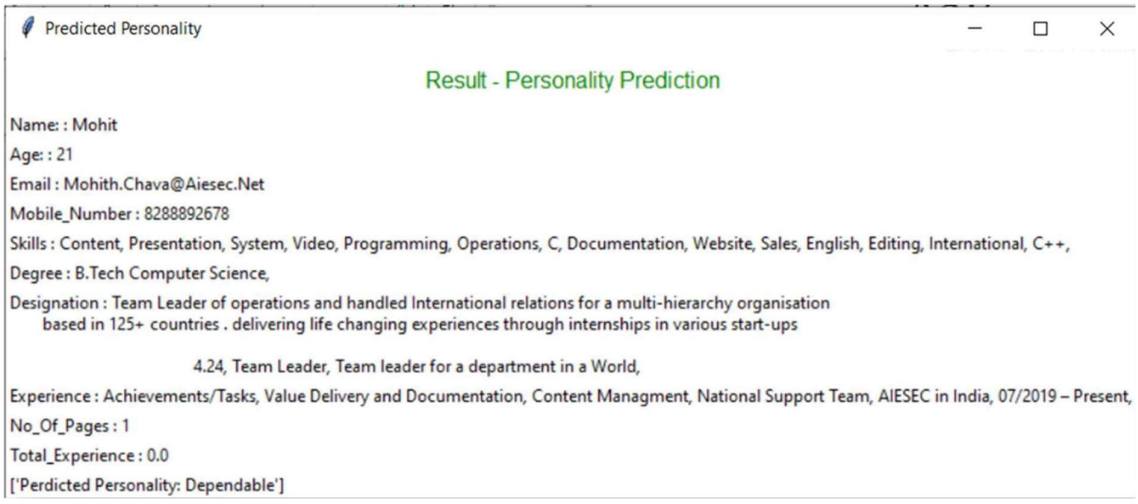
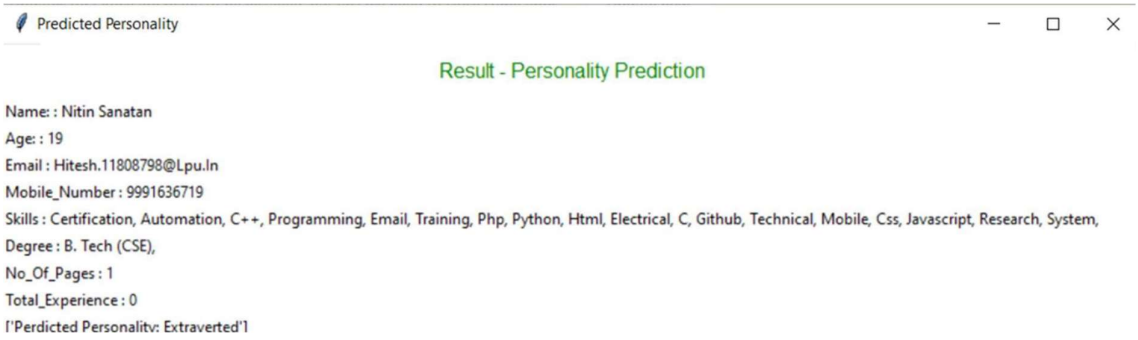


Fig1.3



SWOT analysis:

Strengths:

- Both engaging and simple to use.
- Extract all the important features of resume in seconds
- Easily predict the personality of applicant

Weakness:

- The anticipated personality data is not stored.
- A CV's bulk cannot be parsed in one sitting.

Opportunities:

- It can be extended for commercial uses
- It can be made more interactives where we can easily handle bulk data and represent it.
- It can improve the training model for various addition features that help us to predict more accurate result.
- We can add questionaries that ask some multiple-choice questions and automatically calculate the various values in place of directly asking the five characteristic values.

Threats:

1. There is no security added in the app yet that gives different rights to different users.
2. There are many businesses worldwide, and because their recruiting practices vary depending on the industry, it is necessary to make adjustments based on individual business demands, which can be difficult and expensive to maintain.

5. Chapter 5

Conclusion and Future Scope

In the realm of data science, the presented project stands as an intriguing exploration into the prediction of personality traits using a Python-based application. This initiative merges machine learning methodologies, data parsing techniques, and user interface design to delve into the intricate landscape of human behavior. As we navigate through the conclusion, we'll encapsulate the essence of the project, summarizing its key components, appealing to the emotional engagement it fosters, and charting a course for future enhancements.

The heart of this project lies in its training of a logistic regression model, an entity adept at discerning patterns within a dataset comprising various individual attributes. The journey begins with the preprocessing of data, transforming categorical variables into numerical form, and training the model on a well-curated dataset. The predictive interface allows users to input their details, resulting in the model offering insights into their personalities. The results are then visually presented, offering a comprehensive view that includes both entered and parsed resume data.

Engaging with the project goes beyond the technicalities; it delves into the profound realm of understanding and predicting human nature. Witnessing the application unravel the layers of an individual's personality, juxtaposed with the parsed details from their resume, creates an emotional connection. It invites reflection on the intersection of technology and personal insight, prompting contemplation on the potential of data science to unveil the intricacies of human behavior.

In drawing the curtain on this exploration, the application not only showcases the capabilities of data science but also underscores the significance of harnessing technology to decode the complexities of personality. It invites users to contemplate the power of algorithms in deciphering elements of our individuality, sparking curiosity about the insights that lie within the amalgamation of data and human experience.

As we stand at the crossroads of technology and human understanding, the path forward for this project is both promising and expansive. Future iterations could involve refining the model with additional features, expanding the dataset for greater diversity, and incorporating advanced natural language processing techniques for more nuanced predictions. The interface could evolve to provide more interactive and personalized experiences, fostering a deeper connection between users and the insights derived from their data. The journey towards the future envisions not just a predictive tool but a comprehensive platform that continuously adapts and learns, offering increasingly accurate and meaningful glimpses into the intricacies of human personality.

Github Repository Link

References

1. **Scikit-Learn: Machine Learning in Python.**
   - Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825-2830.
2. **Pandas: Powerful data structures for data analysis.**
   - McKinney, W. (2010). Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference (Vol. 445, pp. 56-61).
3. **Tkinter: Python's standard GUI (Graphical User Interface) package.**
   - Roseman, M., & Rhodes, J. (2000). Tkinter reference: a GUI for Python. New Mexico: New Mexico Tech Computer Center.
4. **Numpy: The fundamental package for scientific computing with Python.**
   - Van Der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The NumPy array: a structure for efficient numerical computation. Computing in Science & Engineering, 13(2), 22-30.
5. **NLTK: The Natural Language Toolkit.**
   - Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python. O'Reilly Media Inc.
6. **Pyresparser: Resume parser using Natural Language Processing.**
   - Pyresparser. (2021). GitHub Repository. Retrieved from https://github.com/OmkarPathak/pyresparser