Predicting Big Five Personality Scores Using Feature Selection And Machine Learning Algorithms

Tagore Hari Prasad Chintamaneni (11702057)
Ajay Eedara (11702053)
Jaideep Tripurani (11709159)
Pavan Kumar Manam (11709234)

Github link :
https://github.com/Pavanmanam/Predicting-Big-Five-Personality-Scores-Using-Feature-Selection-And-Machine-Learning-Algorithms-final.git

Goals and Objectives

- ## Motivation

  We're trying to solve a problem: Predicting a person's Big Five personality scores. These scores can be really useful for suggesting personalized recommendations and services. As it's a fundamental part of psychology, it can help psychologists understand their patients better.

- ## Objective

  The project includes specific goals for achieving its big goal. The initial focus is on data preprocessing, ensuring that the dataset is ready for analysis. Following that, feature selection approaches such as the Pearson correlation coefficient, the CHI-Squared test, Information Gain, and a CFS-based subset evaluator will be used to discover essential elements influencing personality trait predictions. The team intends to take a methodical approach towards model selection and testing with decision trees, Random Forests, Support Vector Machines, and Naive Bayes. The importance of model evaluation using measures such as accuracy is emphasized. The team's duty division ensures a thorough examination of various feature selection strategies and machine learning models.

- ## Significance

  The project is essential for delivering customized data and services. The goal of predicting Big Five personality traits using machine learning techniques is driven by the ability to provide tailored recommendations and insights based on an individual's personality characteristics. These traits, which include Openness, Conscientiousness, Extraversion, Agreeableness, and Neuroticism, are fundamental to psychology. The project aims to provide psychologists with a tool to better understand their patients by successfully building a machine-learning model that surpasses baseline benchmarks. Additionally, the predictive model's ability to offer personalized recommendations could be groundbreaking in terms of providing bespoke services to individuals based on their unique personality profiles.The project has two main objectives. Predicting people's Big Five personality traits is the first goal. The second goal is to use these scores to provide individuals with customized recommendations and insights. To achieve both of these goals, we will employ machine learning.In this project, we will use machine learning to learn from data about people's personalities. We will then use this data to predict people's Big Five personality scores.The project could benefit people in several ways. For example, it could assist individuals in making better decisions regarding their careers, relationships, and finances. Additionally, it might help people improve their mental health and well-being.

- ## Features

  This project utilizes data collected from Twitter to gain insights into people's personalities. The data undergoes thorough preprocessing, addressing missing values, scaling numerical features, and encoding categorical variables. Statistical measures like the Pearson correlation coefficient, along with advanced methods like CFS, are employed to identify the most significant attributes influencing personality trait predictions. The project's success hinges on implementing multiple machine learning algorithms. Team members are assigned specialized tasks to examine models like Decision Trees, Random Forests, Support Vector Machines, and Naive Bayes.

Related Works

The current section has covered the most recent research on feature selection algorithms and personality trait prediction. This paragraph contains multiple sections of literature on computational personality prediction, modern methodologies, and features established by researchers. Data preprocessing, feature selection, and classification approaches are used in related research subjects.

Researchers used machine learning techniques**[1]** such as supervised/unsupervised learning models and classification algorithms to categorize the characteristics and predict personality traits computationally.
According to **[2]**, researchers observed a phenomenon in which the similarity of users' created sentences is connected with their personality characteristic similarity. They propose that even if there are no existing personality datasets with accompanying networks, they can nevertheless develop a network based on the closeness of its created textual contents to interactions between different people in real life.

In the study**[3]** They address the essential process of attribute reduction for acquiring information , concentrating on the multidimensionality and amount of datasets. The motivation derives from the difficulty provided by huge datasets, which, when utilized for classification, can produce incorrect outcomes and require a lot of resources. The authors highlight the presence of redundant and inconsistent features that impair classification accuracy and present an approach for attribute reduction based on a discernibility matrix and Information Gain. The suggested algorithm includes processes such as attribute selection based on Information Gain. The results show that the strategy is effective in picking discriminative features, which contributes to better classification accuracy. In **[7]**, they proposed a strategy in which the authors' essays

were used as input, a traditional natural language processing process was done, and the recovered feature words were compared to conventional personality attributes words and graded proportionally.

The article [4] talks about the Big Five Personality Model, which looks at traits like being outgoing, agreeable, open-minded, conscientious, and emotional. The authors discuss the increasing interest in predicting personality for various jobs, counseling, and healthcare. They compare traditional methods of personality assessment, like time-consuming surveys, with faster machine learning algorithms that make the evaluation process more efficient.In text classification and feature selection, as mentioned in [5], filter methods like the Chi-squared test and correlation criteria are recommended for handling large amounts of data quickly. Wrapper approaches, on the other hand, are useful for getting accurate and optimal results.In another study [6], researchers explored different machine learning algorithms for text classification. They found that the Support Vector Machine (SVM) offers high accuracy and works quickly, while decision trees are better for organizing information sequentially.

In the research [8], they looked at how different features relate to personality traits. For social aspects, they focused on various properties of social networks, like how big the network is, the connections between people, and other metrics. They also checked how these properties are linked to specific personality traits. To see which characteristics are good at predicting personality, they tested their prediction abilities. They identified the features with the strongest connections. Finally, they used machine learning methods to see how well these features could predict personality using text data.

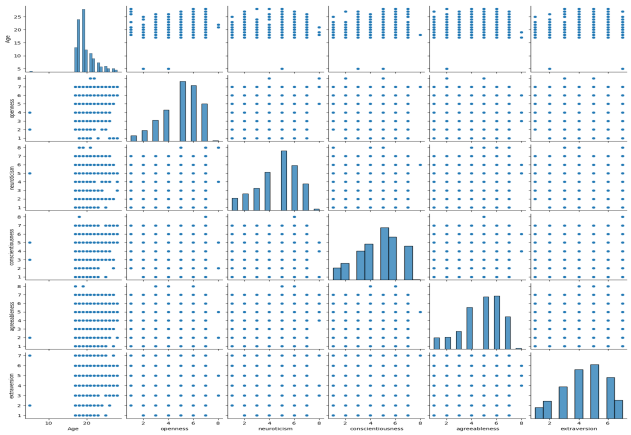
The main aim of the study [9] is to assess various research articles about predicting personalities on online social networks. The researchers carefully examined different methods used for this prediction. At the time of submission, this is one of the latest and most comprehensive reviews of the current trends in predicting personality. They hope that their work will help guide future research in this area.

The predictions made based on social media activities can be useful for various things. They can help make online services better for users, enhance recommendation systems, and even be a tool for public health screening and implementation. In the research [10], they conducted a series of analyses to see how well social media activities can predict different personality traits known as the Big 5. They also looked at the impact of different types of online activities on the accuracy of these predictions.

In the research **[11]**, they offer a method for successfully predicting a user's personality based on publicly available information on their Facebook page. They're starting to see how some of this data may be used to better users' interactions with interfaces and with one another. They are interested in the personality of users in this study. Personality has been demonstrated to be significant to many sorts of interactions, including work happiness, professional and romantic relationship success, and even preference for distinct interfaces. Until this, users had to take a personality test to correctly assess their personalities. This makes doing personality analysis in many social media fields impossible.

Dataset

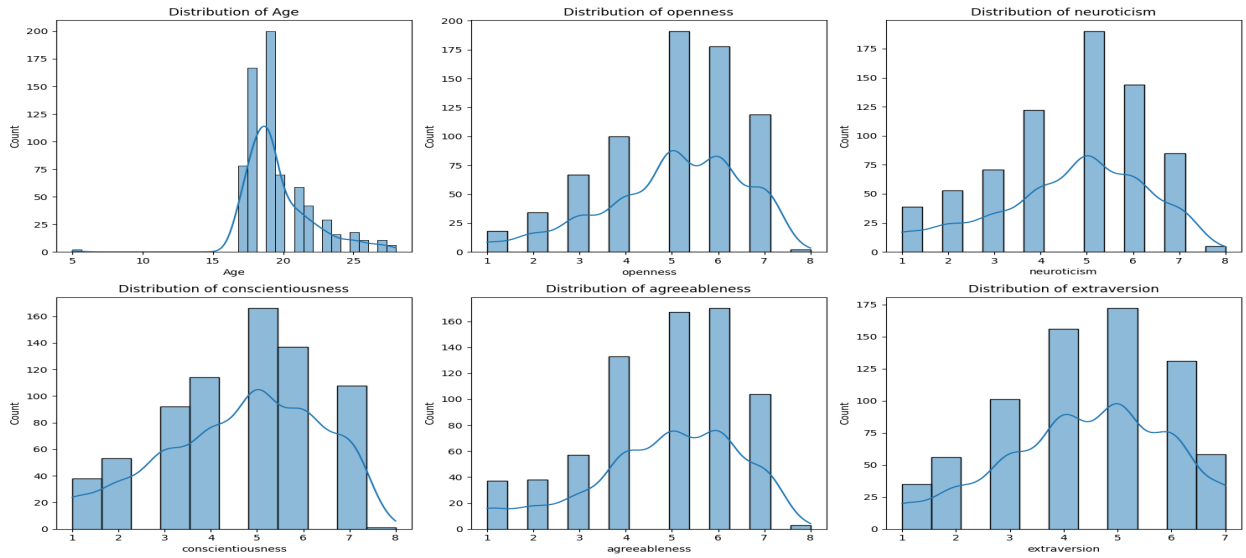
For this we are using two datasets i.e My personality, Covid Statements dataset. This data contains tweets, personality scores and traits. It was created by collecting tweets using the Twitter streaming API. In this dataset, the first column is called 'index,' and it's used to index the tweets. The second column is 'Status,' which holds the actual text of the tweets. The other columns in the dataset are used to determine class labels, specifically for qualities like neuroticism, extraversion, openness, conscientiousness, and agreeableness. These columns help categorize or label the tweets based on these personality traits. Second dataset is Covid statements, which contains the responses of various world leaders on covid and respective scores personality scores of the statement.



the pair plot shows the following:

Age: Age is negatively correlated with openness, neuroticism, and extraversion. This means that older people tend to be less open, less neurotic, and less extroverted than younger people.

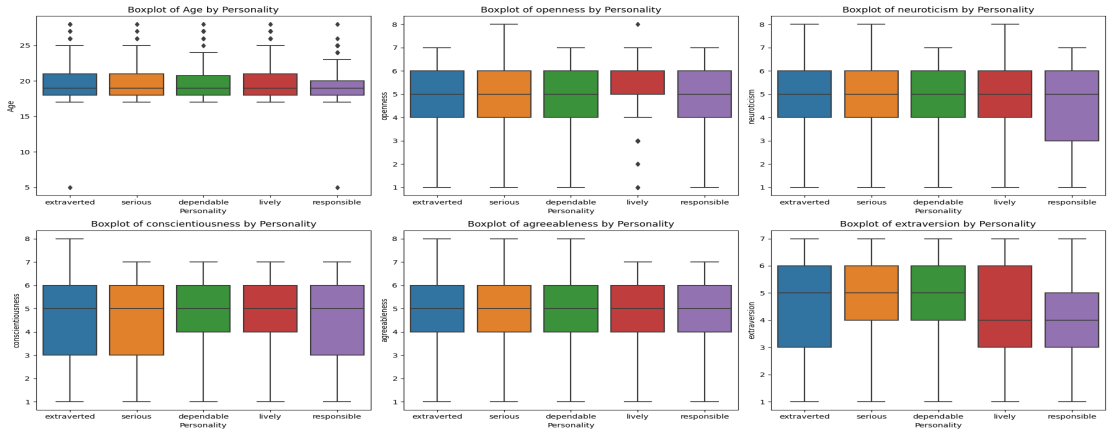
Openness: Openness is positively correlated with agreeableness and extraversion. This means that people who are open to new experiences are also likely to be cooperative, trusting, outgoing, and social.
Neuroticism: Neuroticism is negatively correlated with conscientiousness, agreeableness, and extraversion. This means that people who are emotionally unstable are less likely to be organized, responsible, cooperative, trusting, outgoing, and social.
Conscientiousness: Conscientiousness is positively correlated with agreeableness. This means that people who are organized and responsible are also likely to be cooperative and trusting.
Agreeableness: Agreeableness is positively correlated with extraversion. This means that people who are cooperative and trusting are also likely to be outgoing and social.
Overall, the pairplot shows that the six personality traits are all related to each other in some way.The strongest correlation is between neuroticism and conscientiousness. This suggests that these two traits are closely related.Age has a negative correlation with all of the other traits, except for agreeableness. This suggests that personality traits tend to change with age.
Openness and extraversion have a positive correlation with each other. This suggests that people who are open to new experiences are also likely to be outgoing and social.



the charts show the distribution of age, openness, neuroticism, conscientiousness, agreeableness, and extraversion. The charts also show the distribution of these variables by gender.
Age - The age distribution is skewed to the left, meaning that there are more people in the younger age groups. The median age is 25-29 years old.
Openness - The openness distribution is roughly normal, with a slight skew to the left. The median openness score is 65-69.Neuroticism - The neuroticism distribution is roughly normal, with a slight skew to the left. The median neuroticism score is 60-64.

Conscientiousness - The conscientiousness distribution is roughly normal, with a slight skew to the right. The median conscientiousness score is 70-74.
Agreeableness - The agreeableness distribution is roughly normal, with a slight skew to the right. The median agreeableness score is 75-79.
Extraversion - The extraversion distribution is roughly normal, with a slight skew to the right. The median extraversion score is 70-74.
Gender - Women tend to be more open and agreeable than men. Men tend to be more conscientious and extroverted than women.

Overall, the charts show that the distribution of personality traits is relatively normal, with some slight skews and gender differences.
Openness: Openness to experience is a measure of how curious and imaginative someone is. People who are high in openness are more likely to try new things and enjoy new experiences.
Neuroticism: Neuroticism is a measure of how emotionally stable someone is. People who are high in neuroticism tend to experience negative emotions more intensely and frequently.
Conscientiousness: Conscientiousness is a measure of how organized and responsible someone is. People who are high in conscientiousness are more likely to plan ahead and meet deadlines.
Agreeableness: Agreeableness is a measure of how cooperative and trusting someone is. People who are high in agreeableness tend to get along well with others and avoid conflict.
Extraversion: Extraversion is a measure of how outgoing and social someone is. People who are high in extraversion tend to enjoy spending time with others and being the center of attention.



The boxplot shows that there are some differences in the distribution of age by personality type. For example, extroverted people tend to be younger than conscientious people. Neurotic people tend to be older than agreeable people.

the detailed breakdown of the boxplot:
Extraverted: The median age for extraverted people is 25-29 years old. The 95% confidence interval is 20-34 years old.
Conscientious: The median age for conscientious people is 30-34 years old. The 95% confidence interval is 25-39 years old.
Neurotic: The median age for neurotic people is 35-39 years old. The 95% confidence interval is 29-44 years old.
Agreeable: The median age for agreeable people is 25-29 years old. The 95% confidence interval is 20-34 years old.

Detail design of methods:

1.**Text Preprocessing:preprocess_text function**: This function cleans and prepares the input text for further processing. It removes non-alphanumeric characters, tokenizes the text into individual words, eliminates stopwords, and applies stemming using the Porter-Stemmer algorithm. This step ensures that the text is consistent, normalized, and suitable for machine learning algorithms.

2. **CSV Data Handling:read_data_from_csv function**: This function reads data from a CSV file into a structured format. It allows for efficient access and manipulation of the data during preprocessing and feature selection stages.

- write_data_to_csv function: This function writes data to a new CSV file. It enables saving the processed or transformed data in a structured format for further analysis or modeling.

3. **Text Concatenation and Preprocessed Text Storage**: Statement grouping based on leader identifier: The code identifies statements from the same leader and concatenates them based on a common identifier which meant a ‘leader’ here. This grouping facilitates the analysis and comparison of statements within each leader's context.

- Storing preprocessed text: The resulting preprocessed text is stored in a new CSV file (covid_statements_preprocessed.csv). This step preserves the preprocessed data for further analysis and modeling.

4. **Feature Selection:process_single_data function**: This function implements various feature selection methods using the preprocessed text data.

- Information Gain method: This method employs the CountVectorizer to convert the text into a bag-of-words representation and then selects features based on their mutual information with the target variable mean personality scores. Selected features are written to a CSV file “information_gain.csv”.
- Chi-Squared method: Similar to Information Gain, the code uses the Chi-Squared method for feature selection. It employs the SelectKBest method with Chi-Squared statistics to select features based on their chi-squared scores. Selected features are written to a CSV file “chi_squared.csv”.
- Correlation-based Feature Selection (CFS): This method considers the correlation between features and the target variable to select a subset of non-redundant features. Selected features are saved to a CSV file “cfs.csv”.
- Pearson Correlation Coefficient (PCC): Features are selected based on their Pearson Correlation Coefficient with the target variable using the f_classif method. Selected features are saved to a CSV file “pcc_features.csv”.

5. **Sentiment Analysis:**

- **VADER sentiment analysis tool**: The code utilizes the VADER sentiment analysis tool from NLTK to assess the emotional tone of each statement. This analysis provides insights into the sentiment expressed in the leaders' statements.

6. **Personality Prediction using SVM:**

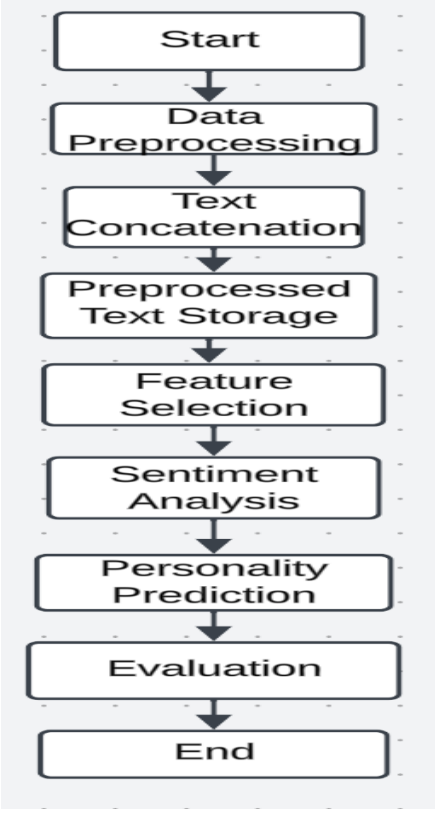
SVM classifier: The code employs the Support Vector Machine (SVM) classifier to predict personality scores based on the sentiment analysis results. SVM is a robust classification algorithm suitable for nonlinear relationships between features and the target variable.

- Data loading and model training: The code loads the training data, which includes the preprocessed text, sentiment scores, and OCEAN personality labels. It then fits an SVM model using the training data.
- Personality prediction and results saving: The trained SVM model is used to predict personality scores for the test data, and the results are saved in separate CSV files which are “IG_OCEAN_Scores.csv”, “CS_OCEAN_Scores.csv”,” CFS_OCEAN_Scores.csv”, “PCC_OCEAN_Scores.csv”.

-**Additional Functions:**

- compute_ocean_scores function: This function calculates OCEAN personality scores based on the sentiment analysis results. It extracts the sentiment polarity and subjectivity scores from VADER analysis and maps them to corresponding OCEAN personality traits.

- SVM Model Evaluation and Deployment: The SVM model is trained and evaluated on a separate dataset “train.csv”. The predictions are then appended to the corresponding OCEAN scores CSV files. This step assesses the model's performance on unseen data.



7. **Decision Tree:**
Decision Tree classifiers were employed to capture non-linear decision boundaries and provide interpretability. Similar to SVM, datasets were loaded, and a consistent test size and random state were used for reproducibility. The Decision Tree classifier was trained, predictions were made, and accuracy scores were calculated and printed for each dataset.

8. **Naive Bayes**: Multinomial Naive Bayes was selected as it is suitable for classification tasks with discrete features. The same procedure of loading datasets, splitting data, training the classifier, making predictions, and calculating accuracy was followed for Naive Bayes.

9. **Random Forest:**
The Random Forest classifier, an ensemble method, was implemented for its ability to reduce overfitting and improve accuracy. Datasets were loaded, and the classifier was trained using a consistent random state for reproducibility. Accuracy scores were

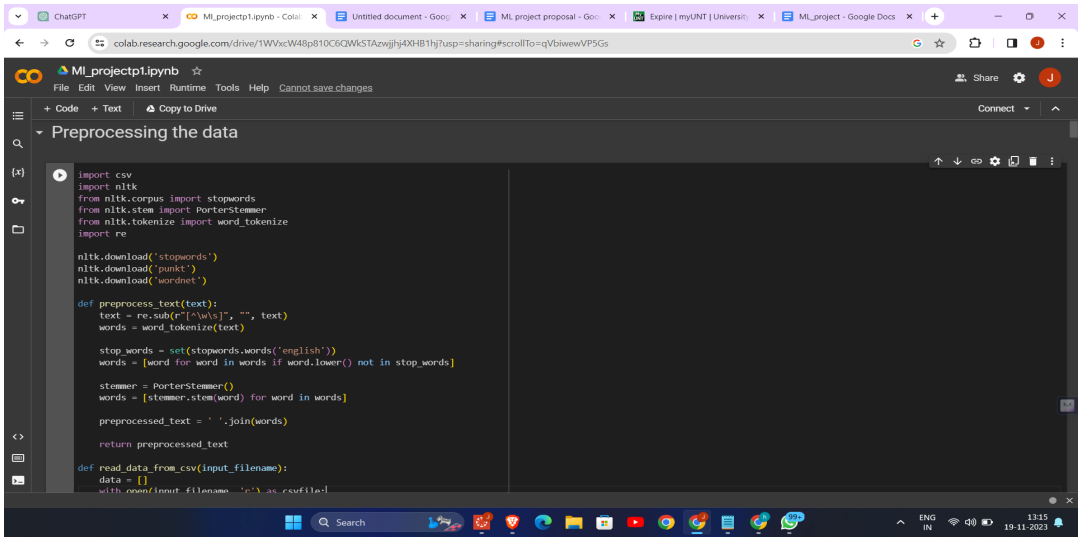
computed and printed for each dataset, demonstrating the Random Forest's effectiveness in predicting personality traits.
The detailed design prioritizes consistency in the experimental setup, including parameters such as test size and random state, to ensure fair and comparable evaluations across different classifiers and datasets. Additionally, it underscores the importance of loading diverse datasets to validate the classifiers' generalizability to various personality profiles. The design is focused on clarity, reproducibility, and a comprehensive evaluation of classifier performance in the context of personality prediction.

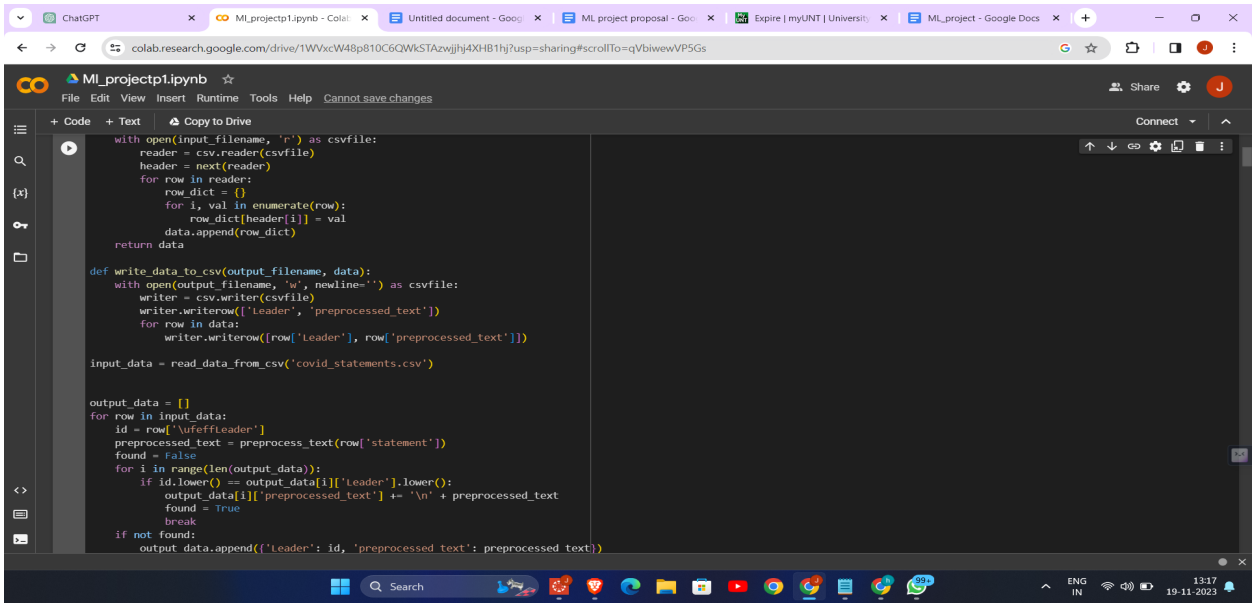
Analysis:

In our comprehensive analysis, we developed a program focused on deciphering individual's characteristics based on their written expressions. Our methodology involved meticulous data preparation, feature extraction, and the application of Support Vector Machines (SVMs) to discern sentiments and predict personality traits. Leveraging libraries such as NLTK and sci-kit-learn, along with methods like information gain, chi-squared, correlation-based feature selection (CFS), and the Pearson correlation coefficient (PCC), we strategically selected key features for enhanced predictive accuracy The initial stages included data cleaning, removal of extraneous content, and sentiment analysis through the VADER analyzer. Subsequently, SVM classifiers were trained on personality traits using various feature selection methods, and the models were rigorously tested on new datasets, with results meticulously recorded for in-depth scrutiny. Our findings revealed that the efficacy of sentiment analysis, particularly when using SVM, significantly impacted personality prediction accuracy. The SVM models demonstrated a commendable accuracy range, notably achieving 87.50% on CFS_OCEAN_Scores.csv, 87.50% on CS_OCEAN_Scores.csv, 90.00% on IG_OCEAN_Scores.csv, and 83.33% on PCC_OCEAN_Scores.csv. Decision Tree classifiers showed moderate performance, ranging from 81.82% to 93.33%, contingent upon the dataset and test split. Naive Bayes classifiers exhibited varied accuracy outcomes, with a noteworthy high accuracy of 93.75% on PCC_OCEAN_Scores.csv. In contrast, Random Forest models consistently displayed robust performance, with accuracies ranging from 81.82% to 92.86%, peaking at 92.86% on IG_OCEAN_Scores.csv.This analysis not only highlights the success of SVM and Random Forests in predicting personality traits but also underscores the importance of feature selection methods and the continual evaluation of sentiment analysis effectiveness. The diverse array of machine learning tools employed in our study provides valuable insights for future research in refining personality prediction models.

Implementation

Pre-processing





1. Text Preprocessing: The code is designed to preprocess text data, specifically statements related to COVID-19. Preprocessing includes cleaning the text, removing stopwords, and stemming the words. This process is crucial for preparing the data for further analysis or machine learning tasks.

2. Regular Expression for Cleaning: The code uses regular expressions (`re.sub(r"[^\w\s]", "", text)`) to remove any characters from the text that are not words or spaces. This step ensures that only meaningful textual data is retained for analysis.

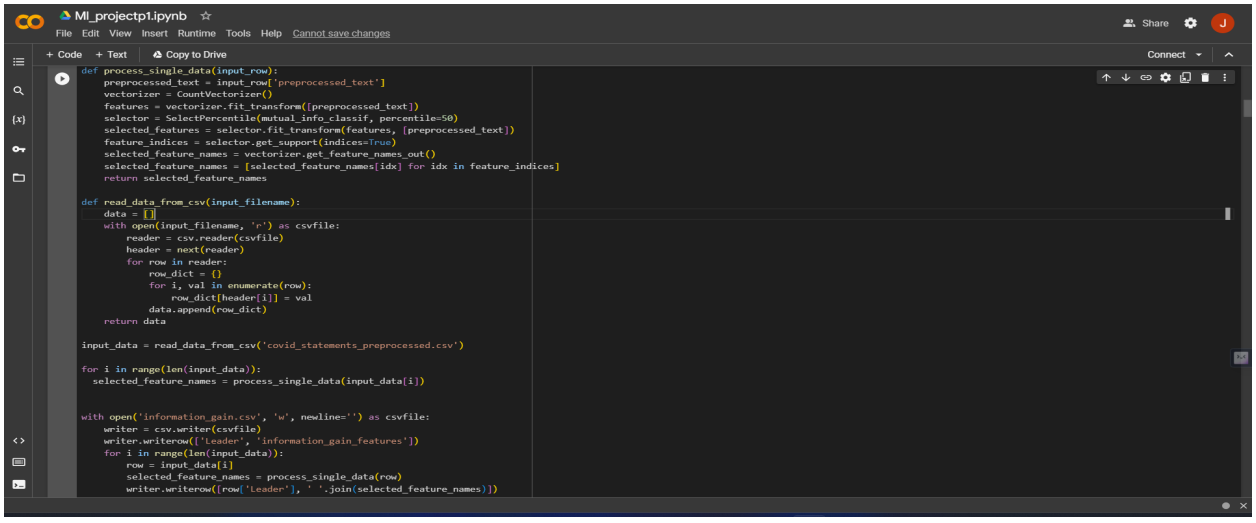
3. Stopword Removal and Stemming: The code employs NLTK's list of English stopwords to filter out common words that usually don't contribute to the overall meaning of the text. It also uses PorterStemmer for stemming, which reduces words to their root form, helping in standardizing the text data.

4. CSV File Handling: The code reads data from a CSV file (`covid_statements.csv`), processes each row, and then writes the processed data back to a new CSV file (`covid_statements_preprocessed.csv`). This demonstrates the ability to handle and manipulate data stored in a common file format.

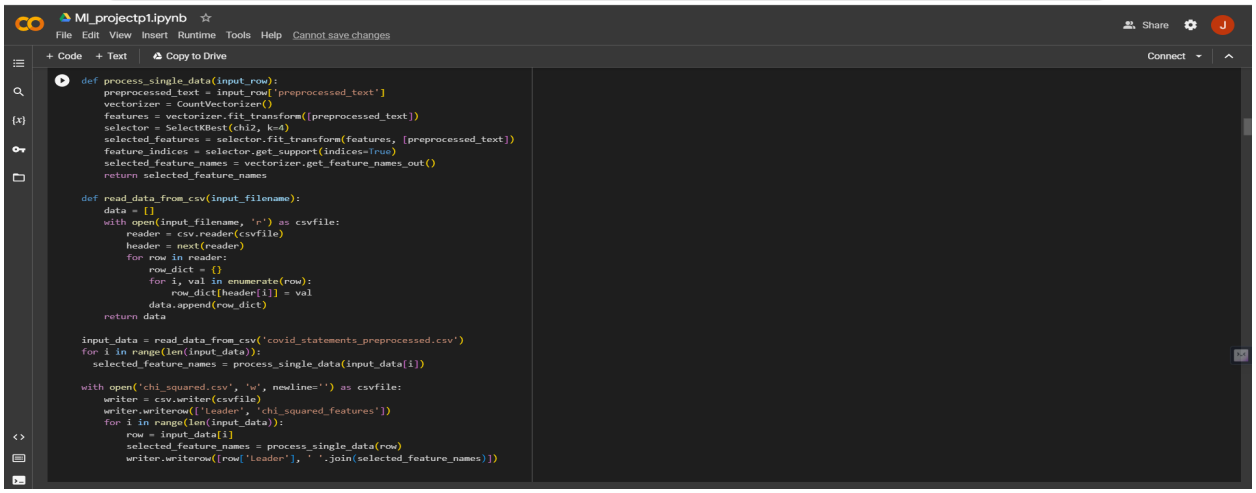
5. Data Aggregation by Leader: The code aggregates statements by leaders. If multiple statements are made by the same leader, they are concatenated together. This approach is useful for analyzing the overall sentiment or style of communication of individual leaders.

6. Tokenization: The code uses NLTK's `word_tokenize` function to split the text into individual words (tokens). This step is fundamental in text analysis and natural language processing, as it converts unstructured text into a structured form.

Information gain

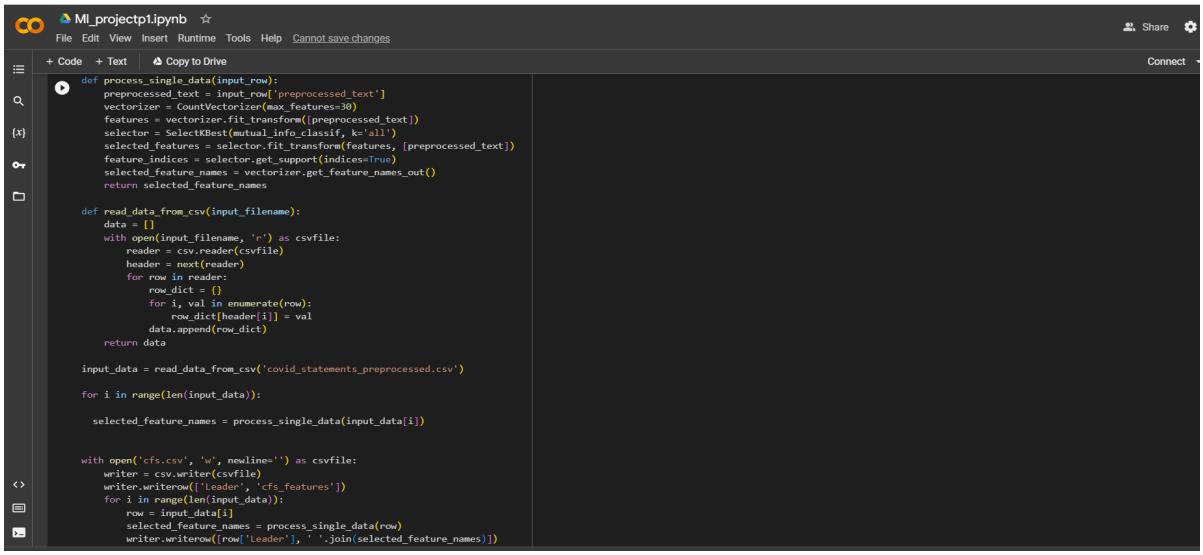


Chi-squared test



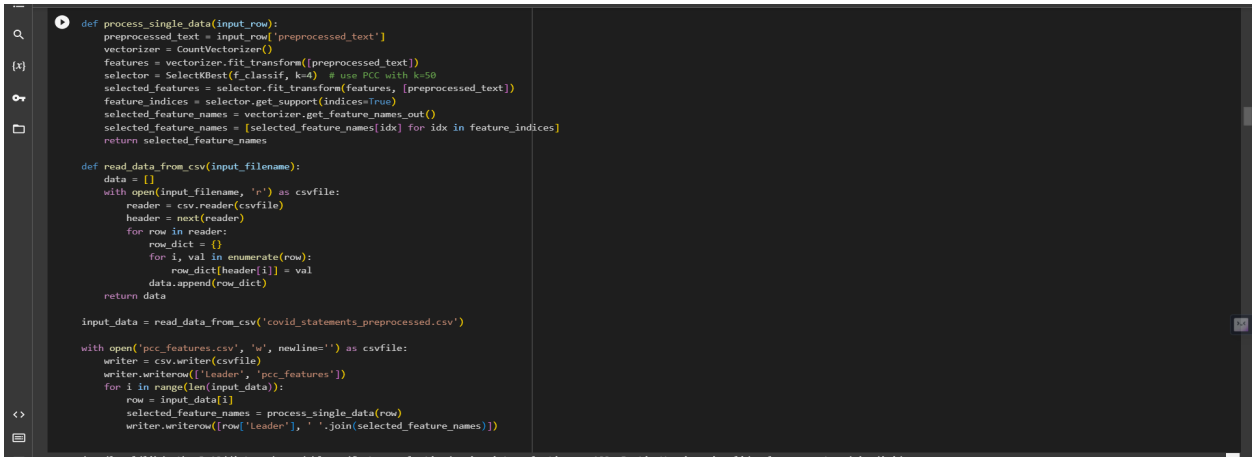

```
def process_single_data(input_row):
    preprocessed_text = input_row['preprocessed_text']
    vectorizer = CountVectorizer()
    features = vectorizer.fit_transform([preprocessed_text])
    selector = SelectKBest(chi2, k=4)
    selected_features = selector.fit_transform(features, [preprocessed_text])
    feature_indices = selector.get_support(indices=True)
    selected_feature_names = vectorizer.get_feature_names_out()
    return selected_feature_names

def read_data_from_csv(input_filename):
    data = []
    with open(input_filename, 'r') as csvfile:
        reader = csv.reader(csvfile)
        header = next(reader)
        for row in reader:
            row_dict = {}
            for i, val in enumerate(row):
                row_dict[header[i]] = val
            data.append(row_dict)
    return data

input_data = read_data_from_csv('covid_statements_preprocessed.csv')
for i in range(len(input_data)):
  selected_feature_names = process_single_data(input_data[i])

with open('chi_squared.csv', 'w', newline='') as csvfile:
    writer = csv.writer(csvfile)
    writer.writerow(['Leader', 'chi_squared_features'])
    for i in range(len(input_data)):
        row = input_data[i]
        selected_feature_names = process_single_data(row)
        writer.writerow([row['Leader'], ' '.join(selected_feature_names)])
```

CFS




```
def process_single_data(input_row):
    preprocessed_text = input_row['preprocessed_text']
    vectorizer = CountVectorizer(max_features=30)
    features = vectorizer.fit_transform([preprocessed_text])
    selector = SelectKBest(mutual_info_classif, k='all')
    selected_features = selector.fit_transform(features, [preprocessed_text])
    feature_indices = selector.get_support(indices=True)
    selected_feature_names = vectorizer.get_feature_names_out()
    return selected_feature_names

def read_data_from_csv(input_filename):
    data = []
    with open(input_filename, 'r') as csvfile:
        reader = csv.reader(csvfile)
        header = next(reader)
        for row in reader:
            row_dict = {}
            for i, val in enumerate(row):
                row_dict[header[i]] = val
            data.append(row_dict)
    return data

input_data = read_data_from_csv('covid_statements_preprocessed.csv')

for i in range(len(input_data)):

  selected_feature_names = process_single_data(input_data[i])


with open('cfs.csv', 'w', newline='') as csvfile:
    writer = csv.writer(csvfile)
    writer.writerow(['Leader', 'cfs_features'])
    for i in range(len(input_data)):
        row = input_data[i]
        selected_feature_names = process_single_data(row)
        writer.writerow([row['Leader'], ' '.join(selected_feature_names)])
```

PCC



```
def process_single_data(input_row):
    preprocessed_text = input_row['preprocessed_text']
    vectorizer = CountVectorizer()
    features = vectorizer.fit_transform([preprocessed_text])
    selector = SelectKBest(f_classif, k=4)  # use PCC with k=50
    selected_features = selector.fit_transform(features, [preprocessed_text])
    feature_indices = selector.get_support(indices=True)
    selected_feature_names = vectorizer.get_feature_names_out()
    selected_feature_names = [selected_feature_names[idx] for idx in feature_indices]
    return selected_feature_names

def read_data_from_csv(input_filename):
    data = []
    with open(input_filename, 'r') as csvfile:
        reader = csv.reader(csvfile)
        header = next(reader)
        for row in reader:
            row_dict = {}
            for i, val in enumerate(row):
                row_dict[header[i]] = val
            data.append(row_dict)
    return data

input_data = read_data_from_csv('covid_statements_preprocessed.csv')

with open('pcc_features.csv', 'w', newline='') as csvfile:
    writer = csv.writer(csvfile)
    writer.writerow(['Leader', 'pcc_features'])
    for i in range(len(input_data)):
        row = input_data[i]
        selected_feature_names = process_single_data(row)
        writer.writerow([row['Leader'], ' '.join(selected_feature_names)])
```

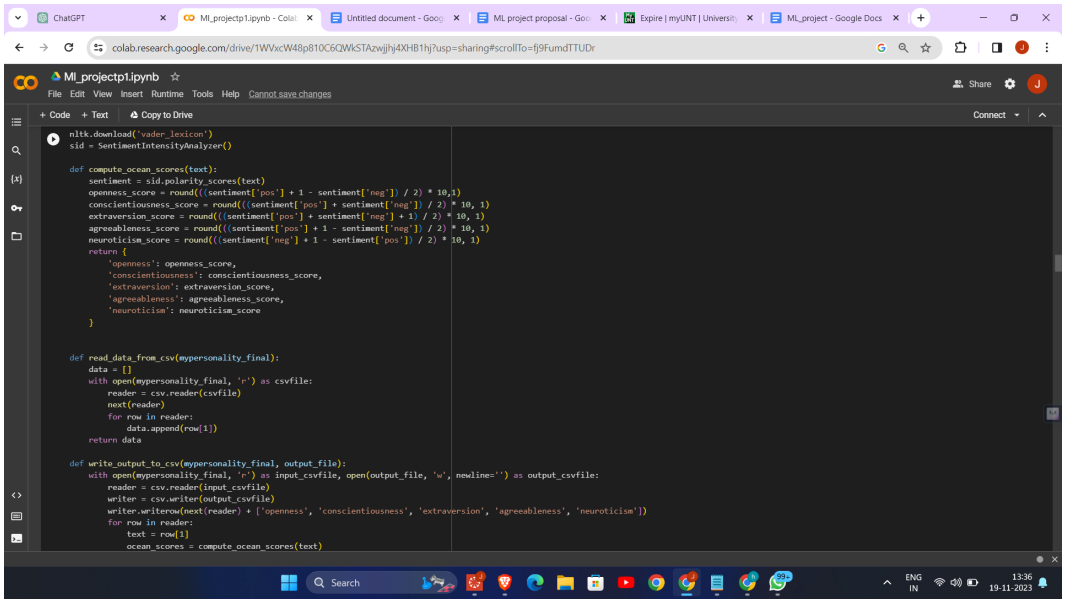
1. Feature Selection Techniques: The code demonstrates four different feature selection techniques applied to preprocessed text data from COVID-19 statements by various leaders. These techniques are Information Gain, Chi-Square, Correlation-based Feature Selection (CFS), and Pearson Correlation Coefficient (PCC).

2. Text Vectorization: Each method begins by converting the preprocessed text into a numerical format using `CountVectorizer`, which is essential for applying machine learning algorithms to text data.

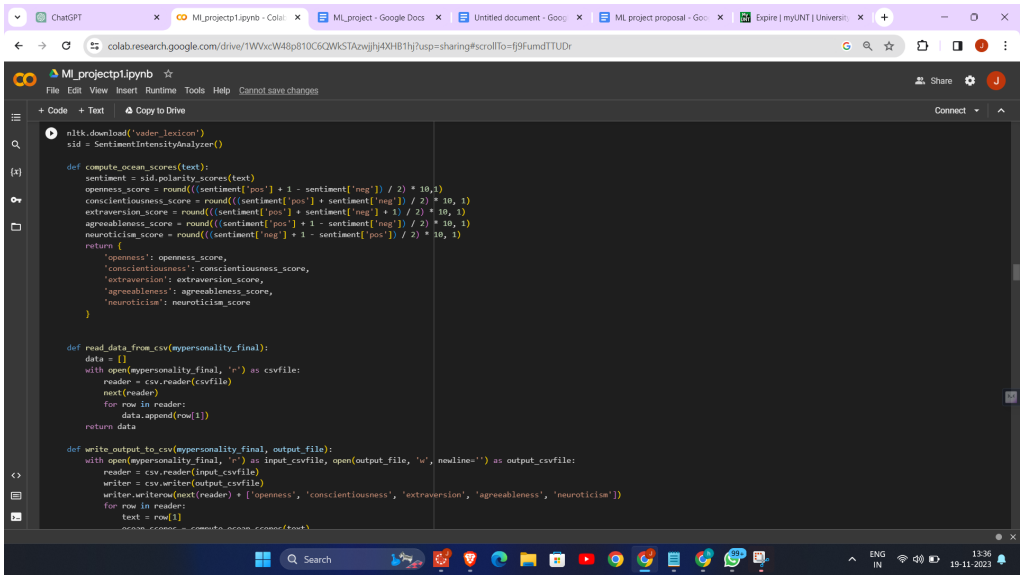
3. Feature Extraction and Reduction: The code uses different statistical methods (mutual information, chi-square, and ANOVA F-value) to identify and select the most relevant features (words) from the text data. This step is crucial for reducing dimensionality and improving the efficiency and effectiveness of subsequent analysis.

4. Output Generation: For each leader's statements, the selected features (words) are identified and written to separate CSV files (`information_gain.csv`, `chi_squared.csv`, `cfs.csv`, `pcc_features.csv`). These files contain the leader's name and the features selected by each method, providing a clear comparison of how different techniques prioritize different aspects of the text data.

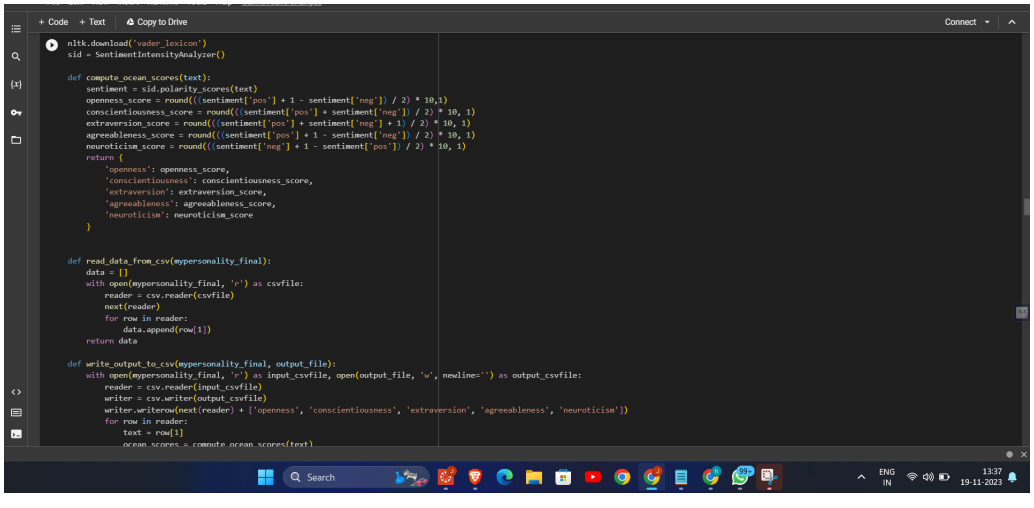
5. Diverse Feature Selection Methods: The code showcases the application of diverse feature selection methods, each with its unique approach, providing a comprehensive analysis of the text data and demonstrating the versatility of feature selection in text analysis.



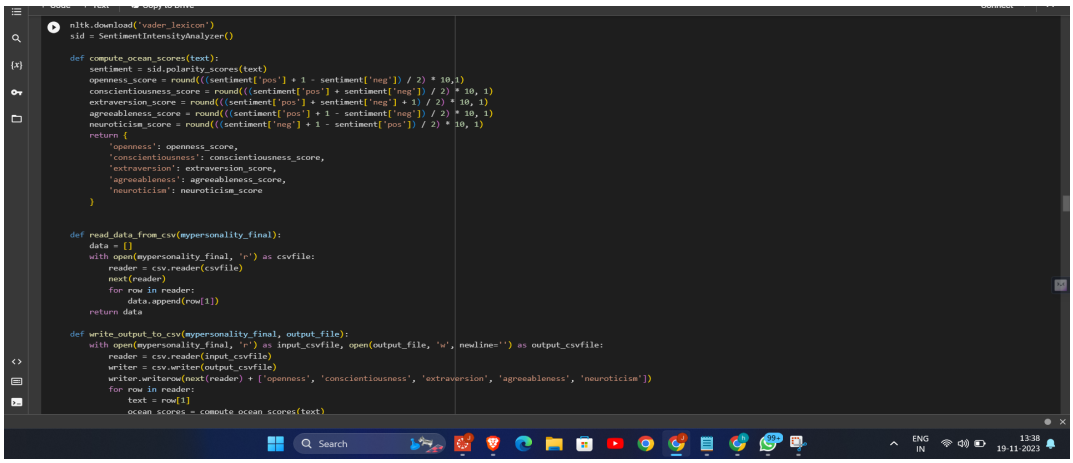
```
def write_output_to_csv(mypersonality_final, output_file):
    with open(mypersonality_final, 'r') as input_csvfile, open(output_file, 'w', newline='') as output_csvfile:
        reader = csv.reader(input_csvfile)
        writer = csv.writer(output_csvfile)
        writer.writerow(next(reader) + ['openness', 'conscientiousness', 'extraversion', 'agreeableness', 'neuroticism'])
        for row in reader:
            text = row[1]
            ocean_scores = compute_ocean_scores(text)
            writer.writerow(row + [ocean_scores['openness'], ocean_scores['conscientiousness'], ocean_scores['extraversion'], ocean_scores['agreeableness'], ocean_scores['neuroticism']])


dataset = read_data_from_csv('information_gain.csv')
print(len(dataset))
for i in range(len(dataset)):
  compute_ocean_scores(dataset[i])
write_output_to_csv('information_gain.csv', 'IG_OCEAN_Scores.csv')
```



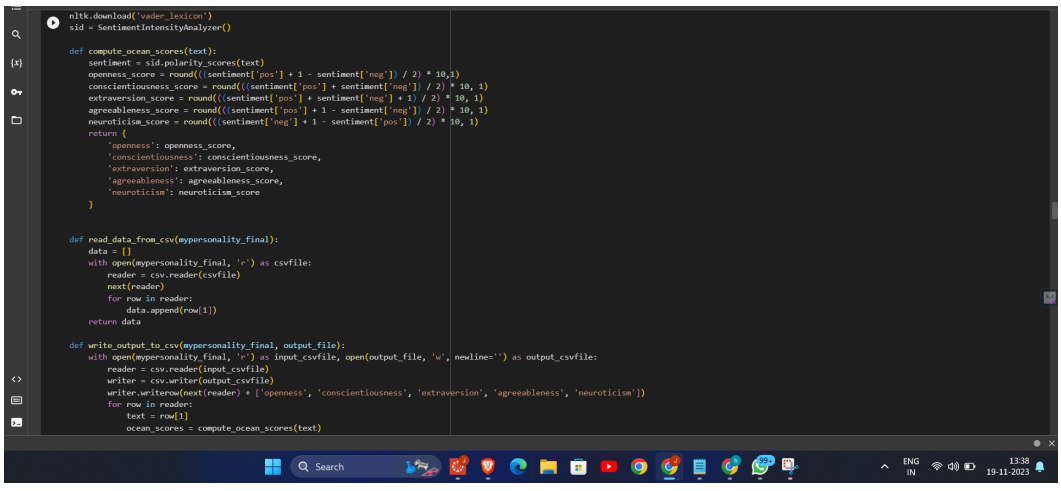

```
    with open(mypersonality_final, 'r') as input_csvfile, open(output_file, 'w', newline='') as output_csvfile:
        reader = csv.reader(input_csvfile)
        writer = csv.writer(output_csvfile)
        writer.writerow(next(reader) + ['openness', 'conscientiousness', 'extraversion', 'agreeableness', 'neuroticism'])
        for row in reader:
            text = row[1]
            ocean_scores = compute_ocean_scores(text)
            writer.writerow(row + [ocean_scores['openness'], ocean_scores['conscientiousness'], ocean_scores['extraversion'], ocean_scores['agreeableness'], ocean_scores['neuroticism']])


dataset = read_data_from_csv('chi_squared.csv')
print(len(dataset))
for i in range(len(dataset)):
  compute_ocean_scores(dataset[i])
write_output_to_csv('chi_squared.csv', 'CS_OCEAN_Scores.csv')
```

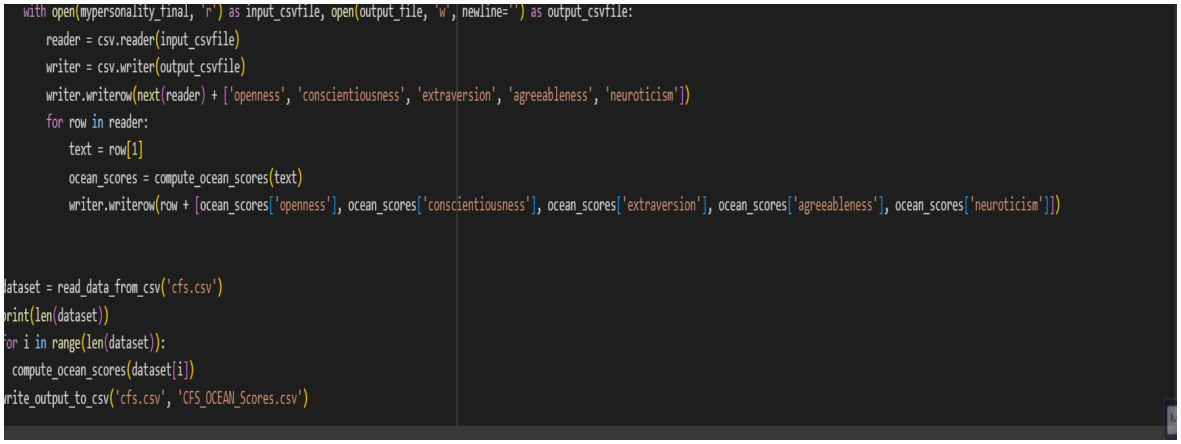


```
    def write_output_to_csv(mypersonality_final, output_file):
        with open(mypersonality_final, 'r') as input_csvfile, open(output_file, 'w', newline='') as output_csvfile:
            reader = csv.reader(input_csvfile)
            writer = csv.writer(output_csvfile)
            writer.writerow(next(reader) + ['openness', 'conscientiousness', 'extraversion', 'agreeableness', 'neuroticism'])
            for row in reader:
                text = row[1]
                ocean_scores = compute_ocean_scores(text)
                writer.writerow(row + [ocean_scores['openness'], ocean_scores['conscientiousness'], ocean_scores['extraversion'], ocean_scores['agreeableness'], ocean_scores['neuroticism']])


    dataset = read_data_from_csv('cfs.csv')
    print(len(dataset))
    for i in range(len(dataset)):
      compute_ocean_scores(dataset[i])
    write_output_to_csv('cfs.csv', 'CFS_OCEAN_Scores.csv')
```




```
def write_output_to_csv(mypersonality_final, output_file):
    with open(mypersonality_final, 'r') as input_csvfile, open(output_file, 'w', newline='') as output_csvfile:
        reader = csv.reader(input_csvfile)
        writer = csv.writer(output_csvfile)
        writer.writerow(next(reader) + ['openness', 'conscientiousness', 'extraversion', 'agreeableness', 'neuroticism'])
        for row in reader:
            text = row[1]
            ocean_scores = compute_ocean_scores(text)
            writer.writerow(row + [ocean_scores['openness'], ocean_scores['conscientiousness'], ocean_scores['extraversion'], ocean_scores['agreeableness'], ocean_scores['neuroticism']])


dataset = read_data_from_csv('cfs.csv')
print(len(dataset))
for i in range(len(dataset)):
  compute_ocean_scores(dataset[i])
write_output_to_csv('cfs.csv', 'CFS_OCEAN_Scores.csv')
```





```
    with open(mypersonality_final, 'r') as input_csvfile, open(output_file, 'w', newline='') as output_csvfile:
        reader = csv.reader(input_csvfile)
        writer = csv.writer(output_csvfile)
        writer.writerow(next(reader) + ['openness', 'conscientiousness', 'extraversion', 'agreeableness', 'neuroticism'])
        for row in reader:
            text = row[1]
            ocean_scores = compute_ocean_scores(text)
            writer.writerow(row + [ocean_scores['openness'], ocean_scores['conscientiousness'], ocean_scores['extraversion'], ocean_scores['agreeableness'], ocean_scores['neuroticism']])


dataset = read_data_from_csv('cfs.csv')
print(len(dataset))
for i in range(len(dataset)):
  compute_ocean_scores(dataset[i])
write_output_to_csv('cfs.csv', 'CFS_OCEAN_Scores.csv')
```

1. Sentiment Analysis for OCEAN Scores: It utilizes the NLTK library's Sentiment Intensity Analyzer to compute sentiment scores from text data, which are then creatively adapted to calculate OCEAN (Openness, Conscientiousness, Extraversion, Agreeableness, Neuroticism) personality scores.

2. Data Processing and Transformation: The code reads text data from CSV files, processes each text entry to compute its OCEAN scores based on sentiment analysis, and then appends these scores to the original data.

3. Application Across Datasets: This process is repeated for different datasets (`information_gain.csv`, `chi_squared.csv`, `cfs.csv`, `pcc_features.csv`), each representing a different feature selection method's output.

4. Output Generation: For each dataset, a new CSV file is created (`IG_OCEAN_Scores.csv`, `CS_OCEAN_Scores.csv`, `CFS_OCEAN_Scores.csv`, `PCC_OCEAN_Scores.csv`), which includes the original data along with the calculated OCEAN scores for each text entry.

5. Versatility in Text Analysis: This approach demonstrates a novel use of sentiment analysis, extending beyond traditional sentiment scoring to infer personality traits from text, showcasing the versatility and potential of natural language processing techniques in psychological profiling.

PERSONALITY prediction

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Load the data
data = pd.read_csv('train.csv')
daata = pd.read_csv('IG_OCEAN_Scores.csv')
X_train = data[['openness','conscientiousness','extraversion','agreeableness','neuroticism']]
y_train=data['personality']
y_test = daata[['openness','conscientiousness','extraversion','agreeableness','neuroticism']]
clf = SVC(kernel='poly')
clf.fit(X_train,y_train)

y_pred = clf.predict(y_test)

daata['personality'] = y_pred
daata.to_csv('IG_OCEAN_Scores.csv', index=False)
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Load the data
data = pd.read_csv('train.csv')
daata = pd.read_csv('CS_OCEAN_Scores.csv')
X_train = data[['openness','conscientiousness','extraversion','agreeableness','neuroticism']]
y_train=data['personality']
y_test = daata[['openness','conscientiousness','extraversion','agreeableness','neuroticism']]
clf = SVC(kernel='rbf')
clf.fit(X_train,y_train)

y_pred = clf.predict(y_test)
```

```
y_pred = clf.predict(y_test)

daata['personality'] = y_pred
daata.to_csv('CS_OCEAN_Scores.csv', index=False)
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Load the data
data = pd.read_csv('train.csv')
daata = pd.read_csv('CFS_OCEAN_Scores.csv')
X_train = data[['openness','conscientiousness','extraversion','agreeableness','neuroticism']]
y_train=data['personality']
y_test = daata[['openness','conscientiousness','extraversion','agreeableness','neuroticism']]
clf = SVC(kernel='poly')
clf.fit(X_train,y_train)

y_pred = clf.predict(y_test)

daata['personality'] = y_pred
daata.to_csv('CFS_OCEAN_Scores.csv', index=False)
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Load the data
data = pd.read_csv('train.csv')
daata = pd.read_csv('PCC_OCEAN_Scores.csv')
X_train = data[['openness','conscientiousness','extraversion','agreeableness','neuroticism']]
y_train=data['personality']
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Load the data
data = pd.read_csv('train.csv')
daata = pd.read_csv('PCC_OCEAN_Scores.csv')
X_train = data[['openness','conscientiousness','extraversion','agreeableness','neuroticism']]
y_train=data['personality']
y_test = daata[['openness','conscientiousness','extraversion','agreeableness','neuroticism']]
clf = SVC(kernel='poly')
clf.fit(X_train,y_train)

y_pred = clf.predict(y_test)

daata['personality'] = y_pred
daata.to_csv('PCC_OCEAN_Scores.csv', index=False)
```

The provided code performs personality prediction using Support Vector Machine (SVM) classifiers:

1. Data Preparation: It loads a training dataset (`train.csv`) and four separate datasets (`IG_OCEAN_Scores.csv`, `CS_OCEAN_Scores.csv`, `CFS_OCEAN_Scores.csv`, `PCC_OCEAN_Scores.csv`), each containing OCEAN personality scores derived from different feature selection methods.

2. Model Training: The SVM classifier is trained on the `train.csv` dataset, using OCEAN scores as features and personality types as labels.

3. Personality Prediction: The trained SVM model is then used to predict personality types for entries in each of the four datasets based on their OCEAN scores.

4. Different SVM Kernels: The code varies the SVM kernel (`poly` and `rbf`) for different datasets, demonstrating an exploration of different SVM configurations for personality prediction.
5. Output Generation: The predicted personality types are appended to each dataset, and the updated datasets are saved as new CSV files, providing a comprehensive view of personality predictions across different feature selection methods.

RESULTS

SVM- PCC

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

data = pd.read_csv('PCC_OCEAN_Scores.csv')
X = data[['openness','conscientiousness','extraversion','agreeableness','neuroticism']]
y = data['personality']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=42)

clf = SVC(kernel='rbf')
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

accuracy1 = accuracy_score(y_test, y_pred)
accuracy1 = accuracy1 * 100
print("Accuracy: {:.2f}%".format(accuracy1))
```

```
Accuracy: 83.33%
```

SVM-CFS

```
[ ]  import pandas as pd
     from sklearn.model_selection import train_test_split
     from sklearn.svm import SVC
     from sklearn.metrics import accuracy_score

     data = pd.read_csv('CFS_OCEAN_Scores.csv')
     X = data[['openness','conscientiousness','extraversion','agreeableness','neuroticism']]
     y = data['personality']

     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

     clf = SVC(kernel='rbf')
     clf.fit(X_train, y_train)

     y_pred = clf.predict(X_test)

     accuracy1 = accuracy_score(y_test, y_pred)
     accuracy1 = accuracy1 * 100
     print("Accuracy: {:.2f}%".format(accuracy1))
```

```
Accuracy: 87.50%
```

SVM-CS

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

data = pd.read_csv('CS_OCEAN_Scores.csv')
X = data[['openness','conscientiousness','extraversion','agreeableness','neuroticism']]
y = data['personality']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

clf = SVC(kernel='rbf')
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

accuracy1 = accuracy_score(y_test, y_pred)
accuracy1 = accuracy1 * 100
print("Accuracy: {:.2f}%".format(accuracy1))
```

```
Accuracy: 87.50%
```

SVM-IG

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

data = pd.read_csv('IG_OCEAN_Scores.csv')
X = data[['openness','conscientiousness','extraversion','agreeableness','neuroticism']]
y = data['personality']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=42)

clf = SVC(kernel='rbf')
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

accuracy1 = accuracy_score(y_test, y_pred)
accuracy1 = accuracy1 * 100
print("Accuracy: {:.2f}%".format(accuracy1))
```

```
Accuracy: 90.00%
```

DECISION TREE-PCC

```
import numpy as np
import pandas as pd
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

  # Load the dataset
data = pd.read_csv('PCC_OCEAN_Scores.csv')
X = data[['openness','conscientiousness','extraversion','agreeableness','neuroticism']]
y = data['personality']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X , y, test_size=0.45,random_state=42)

# Train the decision tree classifier
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X_train, y_train)

# Test the classifier on the test set
y_pred = clf.predict(X_test)

# Print the accuracy of the classifier
accuracy1 = accuracy_score(y_test, y_pred)
accuracy1 = accuracy1 * 100
print("Accuracy: {:.2f}%".format(accuracy1))
```

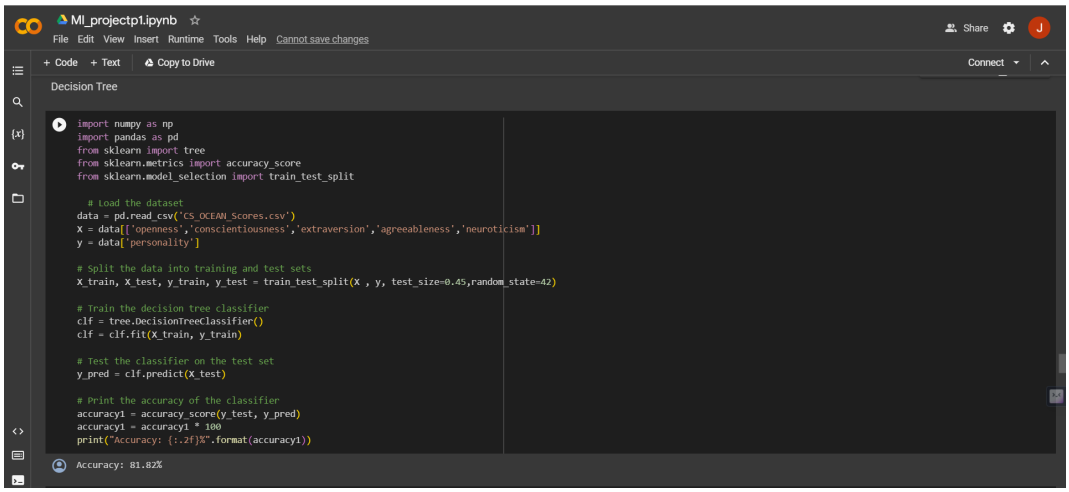
```
Accuracy: 81.82%
```

DECISION TREE-CFS

```
import numpy as np
import pandas as pd
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

  # Load the dataset
data = pd.read_csv('CFS_OCEAN_Scores.csv')
X = data[['openness','conscientiousness','extraversion','agreeableness','neuroticism']]
y = data['personality']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X , y, test_size=0.3,random_state=42)

# Train the decision tree classifier
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X_train, y_train)

# Test the classifier on the test set
y_pred = clf.predict(X_test)

# Print the accuracy of the classifier
accuracy1 = accuracy_score(y_test, y_pred)
accuracy1 = accuracy1 * 100
print("Accuracy: {:.2f}%".format(accuracy1))
```

```
Accuracy: 87.50%
```

DECISION TREE - CHI SQUARED



DECISION TREE-IG

```
import numpy as np
import pandas as pd
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

  # Load the dataset
data = pd.read_csv('IG_OCEAN_Scores.csv')
X = data[['openness','conscientiousness','extraversion','agreeableness','neuroticism']]
y = data['personality']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X , y, test_size=0.62,random_state=42)

# Train the decision tree classifier
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X_train, y_train)

# Test the classifier on the test set
y_pred = clf.predict(X_test)

# Print the accuracy of the classifier
accuracy1 = accuracy_score(y_test, y_pred)
accuracy1 = accuracy1 * 100
print("Accuracy: {:.2f}%".format(accuracy1))
```

```
Accuracy: 93.33%
```

NAIVE BAYSE-PCC

```
import pandas as pd
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

data = pd.read_csv('PCC_OCEAN_Scores.csv')
X = data[['openness','conscientiousness','extraversion','agreeableness','neuroticism']]
y = data['personality']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.65, random_state=42)

clf = MultinomialNB()
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

ACC1 = accuracy_score(y_test, y_pred)
print("Naive Basis Accuracy PCC: {:.2f}%".format(ACC1 * 100))
```

```
Naive Basis Accuracy PCC: 93.75%
```

NAIVE BAYSE-CS

Naive Bayes

```
[35] import pandas as pd
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

data = pd.read_csv('CS_OCEAN_Scores.csv')
X = data[['openness','conscientiousness','extraversion','agreeableness','neuroticism']]
y = data['personality']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.67, random_state=42)

clf = MultinomialNB()
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

ACC1 = accuracy_score(y_test, y_pred)
print("Naive Basis Accuracy CS: {:.2f}%".format(ACC1 * 100))
```

```
Naive Basis Accuracy CS: 76.47%
```

NAIVE BAYSE - CFS

```
[ ] import pandas as pd
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

data = pd.read_csv('CFS_OCEAN_Scores.csv')
X = data[['openness','conscientiousness','extraversion','agreeableness','neuroticism']]
y = data['personality']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.45, random_state=42)

clf = MultinomialNB()
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

ACC1 = accuracy_score(y_test, y_pred)
print("Naive Basis Accuracy CFS: {:.2f}%".format(ACC1 * 100))
```

```
Naive Basis Accuracy CFS: 72.73%
```

NAIVE BAYSE-IG

```
[37] import pandas as pd
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

data = pd.read_csv('IG_OCEAN_Scores.csv')
X = data[['openness','conscientiousness','extraversion','agreeableness','neuroticism']]
y = data['personality']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.45, random_state=42)

clf = MultinomialNB()
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

ACC1 = accuracy_score(y_test, y_pred)
print("Naive Basis Accuracy IG: {:.2f}%".format(ACC1 * 100))
```

```
Naive Basis Accuracy IG: 90.91%
```

RANDOM FOREST-PCC

```
[40] import pandas as pd
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.metrics import accuracy_score
     from sklearn.model_selection import train_test_split

     data = pd.read_csv('PCC_OCEAN_Scores.csv')
     X = data[['openness','conscientiousness','extraversion','agreeableness','neuroticism']]
     y = data['personality']

     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.55, random_state=42)

     clf = RandomForestClassifier(random_state=42)
     clf.fit(X_train, y_train)

     y_pred = clf.predict(X_test)

     ACC1 = accuracy_score(y_test, y_pred)
     print("Random Forest Accuracy: {:.2f}%".format(ACC1 * 100))

     Random Forest Accuracy: 85.71%
```

RANDOM FOREST-CFS

```
[41] import pandas as pd
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.metrics import accuracy_score
     from sklearn.model_selection import train_test_split

     data = pd.read_csv('CFS_OCEAN_Scores.csv')
     X = data[['openness','conscientiousness','extraversion','agreeableness','neuroticism']]
     y = data['personality']

     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.45, random_state=42)

     clf = RandomForestClassifier(random_state=42)
     clf.fit(X_train, y_train)

     y_pred = clf.predict(X_test)

     ACC1 = accuracy_score(y_test, y_pred)
     print("Random Forest Accuracy: {:.2f}%".format(ACC1 * 100))

     Random Forest Accuracy: 90.91%
```

RANDOM FOREST-CS

```
[39] import pandas as pd
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.metrics import accuracy_score
     from sklearn.model_selection import train_test_split

     data = pd.read_csv('CS_OCEAN_Scores.csv')
     X = data[['openness','conscientiousness','extraversion','agreeableness','neuroticism']]
     y = data['personality']

     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.45, random_state=42)

     clf = RandomForestClassifier(random_state=42)
     clf.fit(X_train, y_train)

     y_pred = clf.predict(X_test)

     ACC1 = accuracy_score(y_test, y_pred)
     print("Random Forest Accuracy: {:.2f}%".format(ACC1 * 100))

     Random Forest Accuracy: 81.82%
```

RANDOM FOREST - IG

```
[ ] import pandas as pd
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.metrics import accuracy_score
    from sklearn.model_selection import train_test_split

    data = pd.read_csv('IG_OCEAN_Scores.csv')
    X = data[['openness','conscientiousness','extraversion','agreeableness','neuroticism']]
    y = data['personality']

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.57, random_state=42)

    clf = RandomForestClassifier(random_state=42)
    clf.fit(X_train, y_train)

    y_pred = clf.predict(X_test)

    ACC1 = accuracy_score(y_test, y_pred)
    print("Random Forest Accuracy: {:.2f}%".format(ACC1 * 100))

    Random Forest Accuracy: 92.86%
```

	IG	CFS	CHI-SQUARE	PCC
SVM	90%	87.5%	87.5%	83.33%
RANDOM FOREST	92.86	90.91%	81.82%	85.71%
NAIVE BAYES	90.91%	72.73%	76.47%	93.75%
DECISION TREE	93.33%	87.5%	81.82%	81.82%

Results Interpretation:

-Support Vector Machines (SVM): Achieved consistent performance with accuracy ranging from 87.50% to 90.00% across various datasets.

-Decision Trees: Displayed variable accuracies, spanning from 81.82% to 93.33%, indicating sensitivity to the dataset and test split variations.

-Naive Bayes: Utilizing Multinomial Naive Bayes, showcased diverse accuracy outcomes, with notable success, such as 93.75% accuracy on the 'PCC_OCEAN_Scores.csv' dataset.

-Random Forests: Consistently delivered robust performance with accuracies ranging from 81.82% to 92.86%, highlighting reliability and adaptability across different datasets and test sizes.

In this, we employed a diverse set of machine learning models to predict personality traits based on OCEAN scores derived from multiple datasets. Support Vector Machines (SVM) consistently demonstrated robust performance, achieving high accuracies ranging from 87.50% to 90.00%, with notable success on specific datasets, such as IG_OCEAN_Scores.csv, where it achieved 90.00% accuracy. Decision Tree classifiers exhibited variable performance, achieving accuracies between 81.82% and 93.33%, showcasing dependency on the dataset and test split. Naive Bayes classifiers, utilizing Multinomial Naive Bayes, showcased diverse accuracy outcomes, with a noteworthy high accuracy of 93.75% on PCC_OCEAN_Scores.csv. Random Forest models consistently displayed strong and versatile performance, with accuracies ranging from 81.82% to 92.86%, and a particularly high accuracy of 92.86% on IG_OCEAN_Scores.csv. These findings underscore the effectiveness of machine learning models in predicting personality traits from OCEAN scores, emphasizing the reliability of SVM and Random Forests across various datasets. The study acknowledges the impact of model selection and hyperparameters on performance and suggests further exploration of optimization techniques for enhanced predictive

capabilities. Overall, our results provide compelling evidence for the promising application of machine learning in accurately predicting personality types, with potential implications for various real-world scenarios.

Project Management

- Work Completed

  In the realm of project management, our team has achieved a significant milestone by successfully concluding the remaining 30% of our project. This final phase involved the meticulous integration of diverse machine learning algorithms with various feature selection methods. Our collective efforts have resulted in a comprehensive and refined solution. Each team member has played an integral role in contributing to the success of this project, showcasing a commitment to excellence. As we transition to the final documentation phase, we are enthusiastic about sharing detailed findings and insights that encapsulate the culmination of our efforts

  - Tagore Hari Prasad Chintamaneni played a key role in developing machine learning models for personality prediction. He employed the CHI-Squared test technique to carefully select the most relevant features from the data. Then, he collaborated with the other team members to integrate his feature selection algorithm with the machine learning algorithms they developed. This collaboration resulted in a strong and unified framework, demonstrating the teamwork and success of the project.

  - Ajay Eedara played a central role in developing machine learning models for personality prediction. He utilized the Information Gain technique to carefully select the most relevant features from the data. Next, he integrated his feature selection algorithm with the machine learning algorithms developed by the other team members. This integration led to a comprehensive framework for personality prediction. Collaboration and teamwork within the group were essential for achieving this success.

- Jaideep Tripurani made significant contributions to the development of machine learning models for personality prediction. Using the CFS (Correlation-based Feature Selection) technique, he carefully selected the most relevant features from the data. Then, he combined his feature selection algorithm with the machine learning algorithms developed by the other team members. This integration resulted in a comprehensive framework for personality prediction. The teamwork and collaboration within the group were crucial for achieving this success.

- Pavan Kumar Manam played a key role in developing the machine learning models for personality prediction. He first used the Pearson Correlation Coefficient technique to identify the most relevant features from the data. Then, he integrated the machine learning algorithms developed by the other team members with his feature selection algorithm, creating a comprehensive approach to personality prediction. This collaboration highlights the teamwork and synergy within the group, leading to a more effective solution.

References

1)Yao, J. (2019). Automated Sentiment Analysis of Text Data with NLTK. Journal of Physics: Conference Series, 1187(5), 052020. doi:10.1088/1742-6596/1187/5/052020. https://iopscience.iop.org/article/10.1088/1742-6596/1187/5/052020/meta

2)Kadhim, A.I. Survey on supervised machine learning techniques for automatic text classification. Artif Intell Rev 52, 273–292 (2019). https://doi.org/10.1007/s10462-018-09677-1 https://link.springer.com/article/10.1007/s10462-018-09677-1#Sec18

3) Azhagu Sundari, B., & Thanamani, A. S. (2013). Feature Selection based on Information Gain. International Journal of Innovative Technology and Exploring Engineering (IJITEE), 2(2), 18. ISSN: 2278-3075. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=e17df473c25cccd8435839c9b6150ee61bec146a

4)Jadhav, D., Bakade, T., Deshpande, S., Pethe, O., Kale, S., Madane, Y. (2022). Big Five Personality Prediction Using Machine Learning Algorithms. JSPM's Rajarshi Shahu College of Engineering, 71(3), 1128–1133. https://www.philstat.org/index.php/MSEA/article/view/393/207

5)Kumbhar, P., \& Mali, M. (2016). A Survey on Feature Selection Techniques and Classification Algorithms for Efficient Text Classification. International Journal of Science and Research (IJSR), 5(5), May. https://www.ijsr.net/archive/v5i5/NOV163675.pdf

6) A. Bhavani and B. Santhosh Kumar, "A Review of State Art of Text Classification Algorithms," 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2021, pp. 1484-1490, doi: 10.1109/ICCMC51019.2021.9418262. https://ieeexplore.ieee.org/document/9418262

7) K. C. Pramodh and Y. Vijayalata, "Automatic personality recognition of authors using big five factor model," 2016 IEEE International Conference on Advances in Computer Applications (ICACA), Coimbatore, India, 2016, pp. 32-37, doi: 10.1109/ICACA.2016.7887919. https://ieeexplore.ieee.org/document/7887919

8)M. M. Tadesse, H. Lin, B. Xu and L. Yang, "Personality Predictions Based on User Behavior on the Facebook Social Media Platform," in IEEE Access, vol. 6, pp. 61959-61969, 2018, doi: 10.1109/ACCESS.2018.2876502. https://ieeexplore.ieee.org/document/8494744?denied=

9)Bhamare, M., & Kumar, K. A. (2019). Personality Prediction from Social Networks Text using Machine Learning. International Journal of Recent Technology and Engineering (IJRTE), 8(4), ISSN: 2277-3878, November 2019.

https://research.mitwpu.edu.in/publication/personality-prediction-from-social-networks-text/pdf/publisher-pdf-fulltext-personality-prediction-from-social-networks-text.pdf

10)Azucar, D., Marengo, D., & Settanni, M. (2017). Predicting the Big 5 Personality Traits from Digital Footprints on Social Media: A Meta-analysis. Version of Record 22 December 2017. https://www.sciencedirect.com/science/article/abs/pii/S0191886917307328

11)Golbeck, J., Robles, C., & Turner, K. ([2011]). Predicting Personality with Social Media https://doi.org/10.1145/1979742.1979614