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)

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 holds significant importance for providing personalized data and services. The aim of predicting Big Five personality scores using machine learning techniques is motivated by the ability to provide individualized recommendations and insights based on an individual's personality characteristics. These scores, which include Openness, Conscientiousness, Extraversion, Agreeableness, and Neuroticism, are a basic part of psychology. The project hopes to equip psychologists with a tool to better understand their patients by successfully constructing a machine-learning model that outperforms baseline benchmarks. Furthermore, the predictive model's ability to provide personalized recommendations could be game-changing in terms of providing bespoke services to individuals based on their unique personality profiles.

- ### Features

This project is built on the datasets "my_personality", " Covid statements," which were obtained using the Twitter streaming API. The dataset, which consists of tweet text and class labels indicating personality qualities, is carefully preprocessed, including handling missing values, scaling numerical features, and encoding categorical variables. Statistical measurements such as the Pearson correlation coefficient, as well as more advanced methods like CFS, are used to identify the most relevant attributes for predicting personality trait scores. The implementation of multiple machine learning algorithms is critical to the project's success, with team members allocated specialized tasks to investigate models such as 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.

In The article **[4],** it focuses on the Big Five Personality Model, which includes traits such as Extraversion, Agreeableness, Openness, Conscientiousness, and Neuroticism. The authors draw attention to the growing interest in personality prediction in a variety of professions, including job development, counseling, and healthcare. Traditional techniques of personality assessment, which involve time-consuming surveys, are compared to the efficiency gained by machine learning algorithms, which streamline the evaluation process. For text classification and feature selection, According to **[5]**, among the feature selection approaches, filter methods such as the Chi-squared test and correlation criteria will help to handle vast amounts of data in less time. Wrapper approaches, on the other hand, can be effective for accurate and optimal output. In **[6]**, they studied multiple algorithms using machine learning as classifiers for text classification and stated that when comparing different machine learning algorithms, the Support vector machine (SVM) provides more accuracy and computes in less time, whereas a decision tree is used for sequential categorization.

According to the study **[8]**, the relationships between each feature set and personality attributes. In the case of social characteristics, they concentrate on many types of structural network properties, including network size, betweenness, density, brokerage, and transitivity metrics, as well as their correlations to specific attributes. They then tested the characteristics' prediction potential by predicting each personality attribute. They look at the characteristics that have the highest correlations. Finally, they define machine learning methods and include them into the prediction model to investigate the extent to which they can predict personality using text dataset.

The primary goal of this research **[9]** is to evaluate the many studies published in the literature on the identification of personality prediction of online social networks. They conducted a precise and relative examination of several methods used for personality prediction in this research.This is frequently the major review article on newest trends utilized for personality prediction at the time of submission for the most effective of our data. They hope that their dedication will aid in differentiating the headings for future research here.

These social media-based predictions may subsequently be utilized for a number of applications, such as adapting online services to improve user experience, improving recommender systems, and perhaps serving as a screening and implementation tool for public health. In the study **[10]**, they undertake a series of meta-analyses to investigate the predictive potential of social media digital footprints across Big 5 personality characteristics In addition, they study the effect of various forms of digital footprints on prediction accuracy.

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.

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.

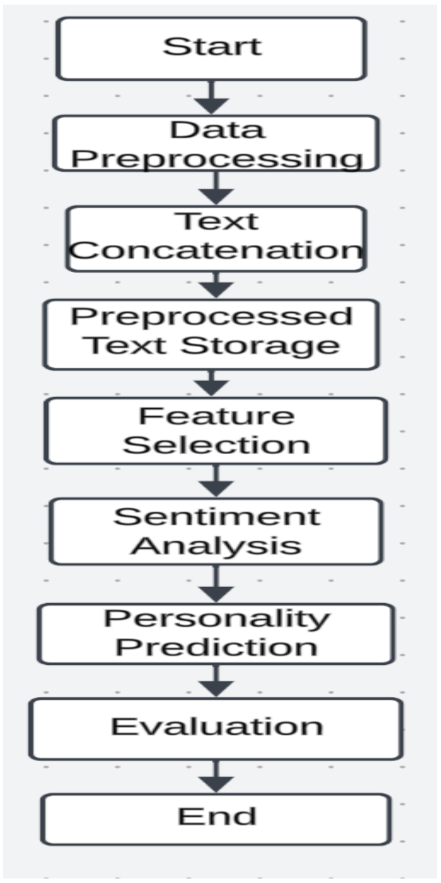
- Information Gain method: This method employs the CountVectorizer to convert the text into a bag-of-words representation and then selects features based on their mutual

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

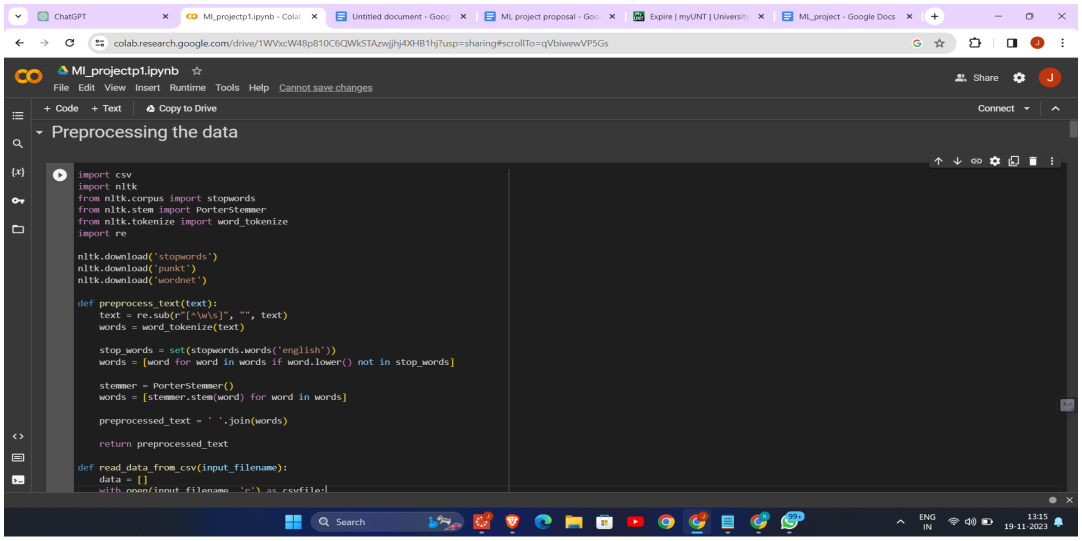
clarity, reproducibility, and a comprehensive evaluation of classifier performance in the context of personality prediction.

Analysis:

We made a program to figure out what people are like based on what they write. First, we get the data ready, pick the important parts, and use Support Vector Machines (SVMs) to analyze feelings and predict personalities. We use libraries like NLTK and sci-kit-learn and methods like information gain, chi-squared, correlation-based feature selection (CFS), and Pearson correlation coefficient (PCC) to choose the best features. After cleaning up the writing, taking out the extra stuff, and finding feelings using the VADER analyzer, the program trains SVM classifiers on personality traits using different feature selection methods. Then, we test the models on new data, and the results are saved for more studying. To make our personality-predicting model work well, we need to pick good features and use the right machine-learning method. How well sentiment analysis works can change, and other measures like accuracy can give us more details on how well the model is doing. We tried different machine learning tools, like SVM, Decision Tree, Naive Bayes, and Random Forest, to predict personalities on different sets of data. The study examined the effectiveness of sentiment analysis, emphasizing metrics such as accuracy, to comprehensively evaluate model performance. Several machine learning tools, including SVM, Decision Tree, Naive Bayes, and Random Forest, were tested on distinct datasets. SVM emerged as the most successful method, achieving an accuracy range of 83.3%. Decision Tree demonstrated moderate performance 81.82% , However, it is noteworthy that Naive Bayes shows a relatively lower accuracy level of 72.73%.Random Forest consistently shown accuracy rate of 92.86%

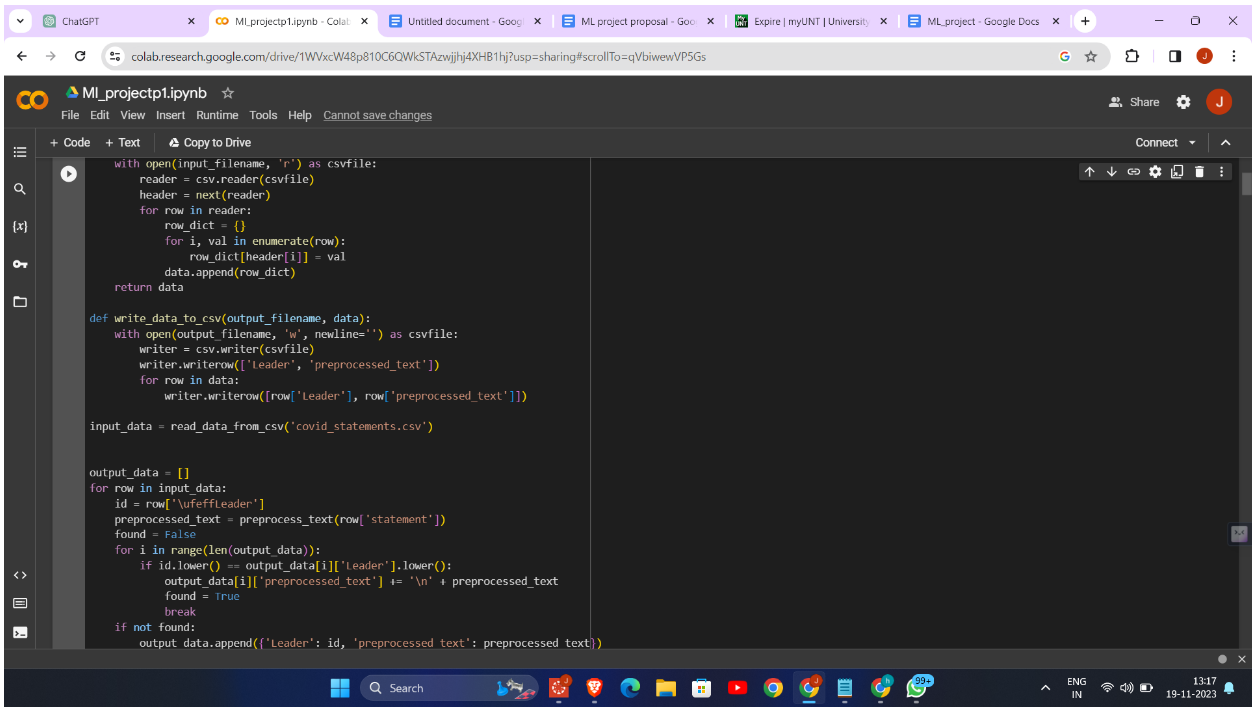
Implementation

Pre processing



ChatGPT
Ml_projectp1.ipynb - Colab
Untitled document - Goog
ML project proposal - Goo
Expire | myUNT | University
ML_project - Google Docs
colab.research.google.com/drive/1WVxcW48p810C6QWkSTAzwjjhj4XHB1hj?usp=sharing#scrollTo=qVbiwewVP5Gs
Ml_projectp1.ipynb
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
Share
Connect
Preprocessing the data
Search
ENG IN
13:15 19-11-2023

```python
import csv
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
import re

nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')

def preprocess_text(text):
    text = re.sub(r"[^\w\s]", "", text)
    words = word_tokenize(text)

    stop_words = set(stopwords.words('english'))
    words = [word for word in words if word.lower() not in stop_words]

    stemmer = PorterStemmer()
    words = [stemmer.stem(word) for word in words]

    preprocessed_text = ' '.join(words)

    return preprocessed_text

def read_data_from_csv(input_filename):
    data = []
    with open(input_filename, 'r') as csvfile:
```



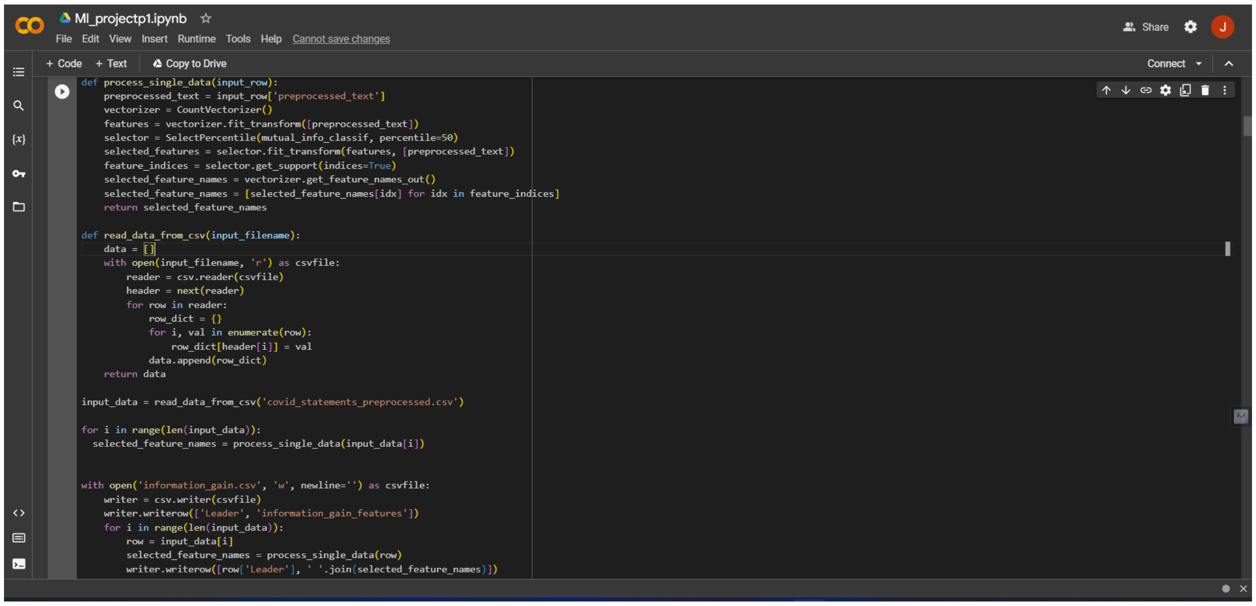
ChatGPT
Ml_projectp1.ipynb - Colab
Untitled document - Goog
ML project proposal - Goo
Expire | myUNT | University
ML_project - Google Docs
colab.research.google.com/drive/1WVxcW48p810C6QWkSTAzwjjhj4XHB1hj?usp=sharing#scrollTo=qVbiwewVP5Gs
Ml_projectp1.ipynb
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
Share
Connect
Search
ENG IN
13:17 19-11-2023

```python
    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

def write_data_to_csv(output_filename, data):
    with open(output_filename, 'w', newline='') as csvfile:
        writer = csv.writer(csvfile)
        writer.writerow(['Leader', 'preprocessed_text'])
        for row in data:
            writer.writerow([row['Leader'], row['preprocessed_text']])

input_data = read_data_from_csv('covid_statements.csv')


output_data = []
for row in input_data:
    id = row['\ufeffLeader']
    preprocessed_text = preprocess_text(row['statement'])
    found = False
    for i in range(len(output_data)):
        if id.lower() == output_data[i]['Leader'].lower():
            output_data[i]['preprocessed_text'] += '\n' + preprocessed_text
            found = True
            break
    if not found:
        output_data.append({'Leader': id, 'preprocessed_text': preprocessed_text})
```

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.

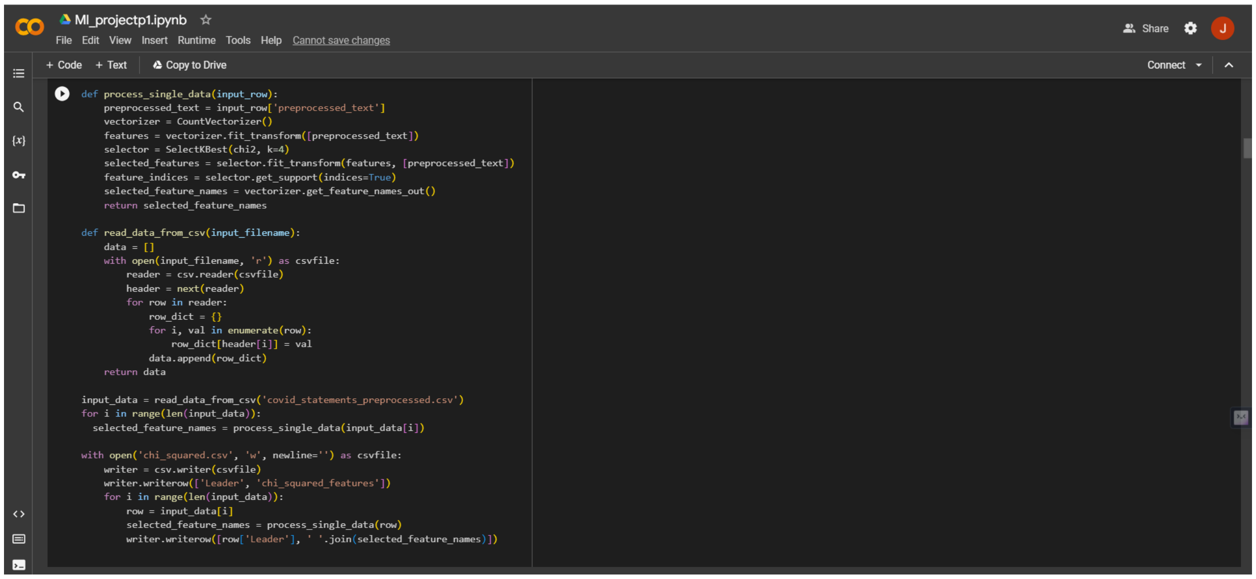
2. Regular Expression for Cleaning: The code uses regular expressions (`re.sub(r"[^\w\s]", "", text)`) to remove any characters from the text that are not words or spaces. This step ensures that only meaningful textual data is retained for analysis.
3. Stopword Removal and Stemming: The code employs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


MI_projectp1.ipynb
File Edit View Insert Runtime Tools Help Cannot save changes
Share
+ Code + Text
Copy to Drive
Connect

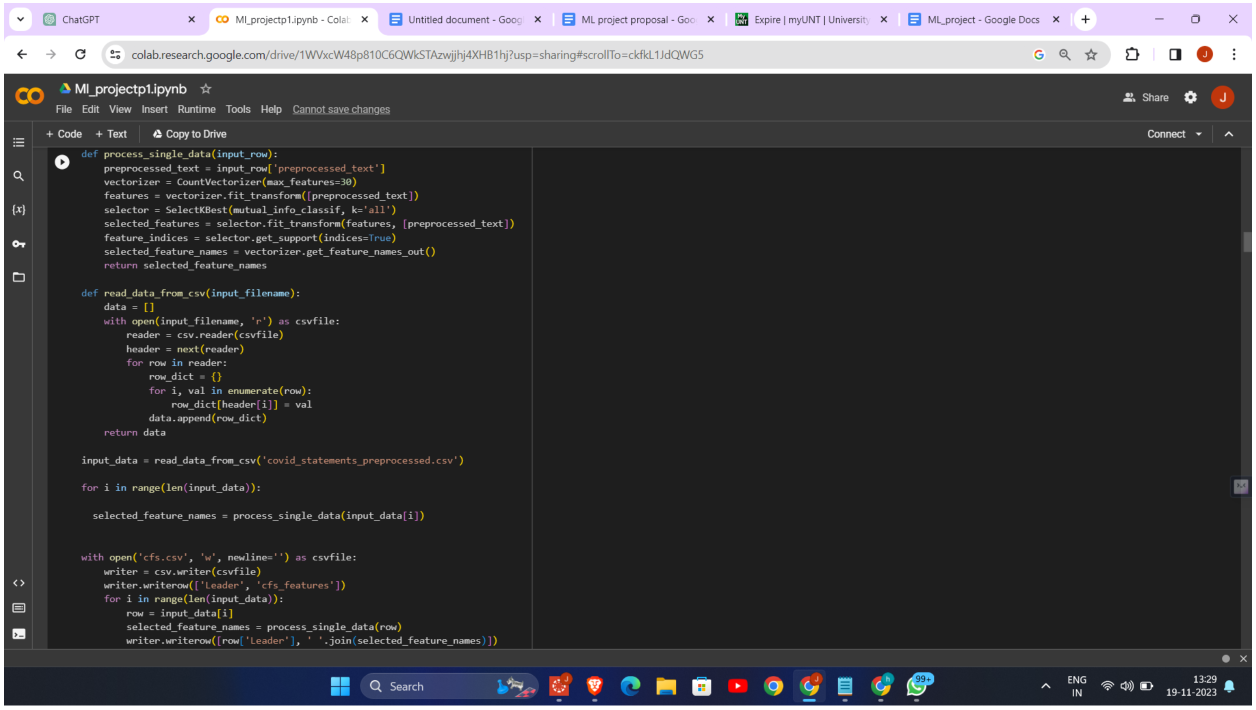
```python
def process_single_data(input_row):
    preprocessed_text = input_row['preprocessed_text']
    vectorizer = CountVectorizer()
    features = vectorizer.fit_transform([preprocessed_text])
    selector = SelectPercentile(mutual_info_classif, percentile=50)
    selected_features = selector.fit_transform(features, [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')

for i in range(len(input_data)):
  selected_feature_names = process_single_data(input_data[i])


with open('information_gain.csv', 'w', newline='') as csvfile:
    writer = csv.writer(csvfile)
    writer.writerow(['Leader', 'information_gain_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)])
```

Chi - squared test


MI_projectp1.ipynb
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
Share
Connect

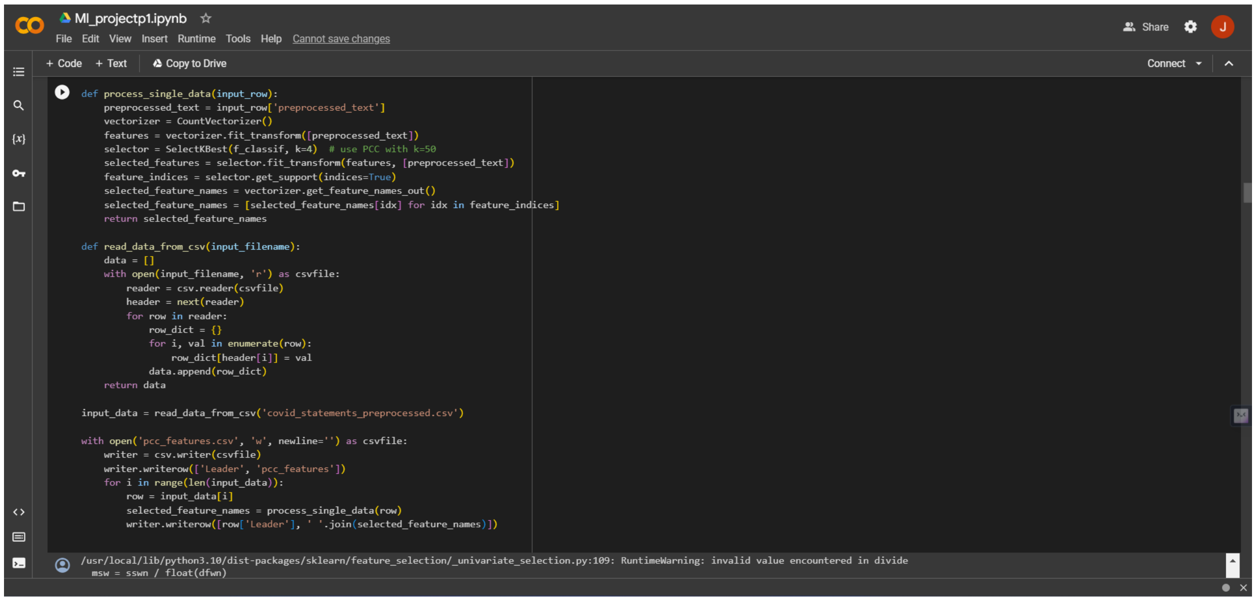
```python
def process_single_data(input_row):
    preprocessed_text = input_row['preprocessed_text']
    vectorizer = CountVectorizer()
    features = vectorizer.fit_transform([preprocessed_text])
    selector = SelectKBest(chi2, k=4)
    selected_features = selector.fit_transform(features, [preprocessed_text])
    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


ChatGPT
MI_projectp1.ipynb - Colab
Untitled document - Goog
ML project proposal - Goo
Expire | myUNT | University
ML_project - Google Docs
colab.research.google.com/drive/1WVxcW48p810C6QWkSTAzwjjhj4XHB1hj?usp=sharing#scrollTo=ckfkL1JdQWG5
MI_projectp1.ipynb
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
Share
Connect
Search
ENG
IN
13:29
19-11-2023

```python
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


MI_projectp1.ipynb
File Edit View Insert Runtime Tools Help Cannot save changes
Share
+ Code + Text Copy to Drive
Connect

```python
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)])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/feature_selection/_univariate_selection.py:109: RuntimeWarning: invalid value encountered in divide
  msw = sswn / float(dfwn)
```

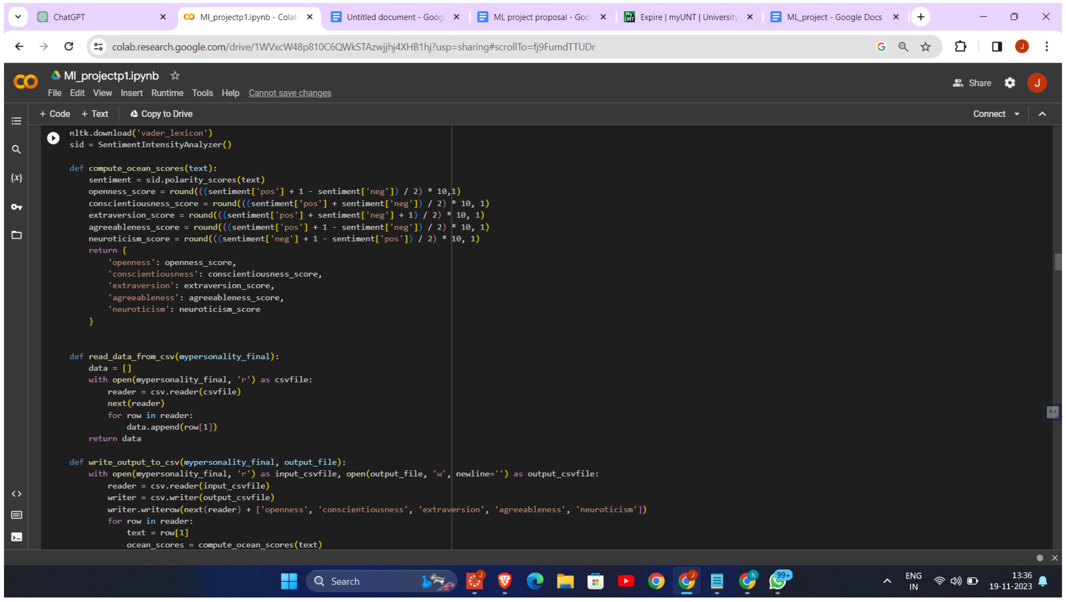
1. Feature Selection Techniques: The code demonstrates four different feature selection techniques applied to preprocessed text data from COVID-19 statements by various leaders. These techniques are Information Gain, Chi-Square, Correlation-based Feature Selection (CFS), and Pearson Correlation Coefficient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

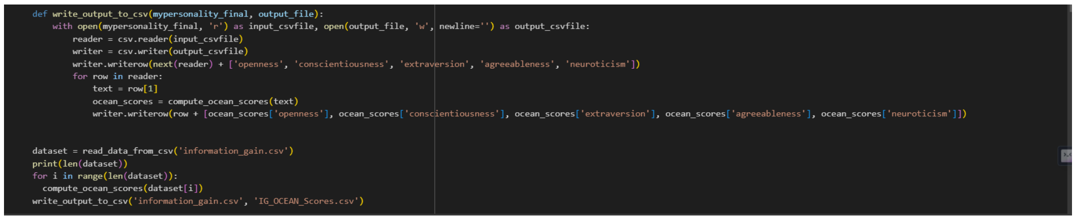
3. Feature Extraction and Reduction: The code uses different statistical methods (mutual information, chi-square, and ANOVA F-value) to identify and select the most relevant features (words) from the text data. This step is crucial for reducing dimensionality and improving the efficiency and effectiveness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

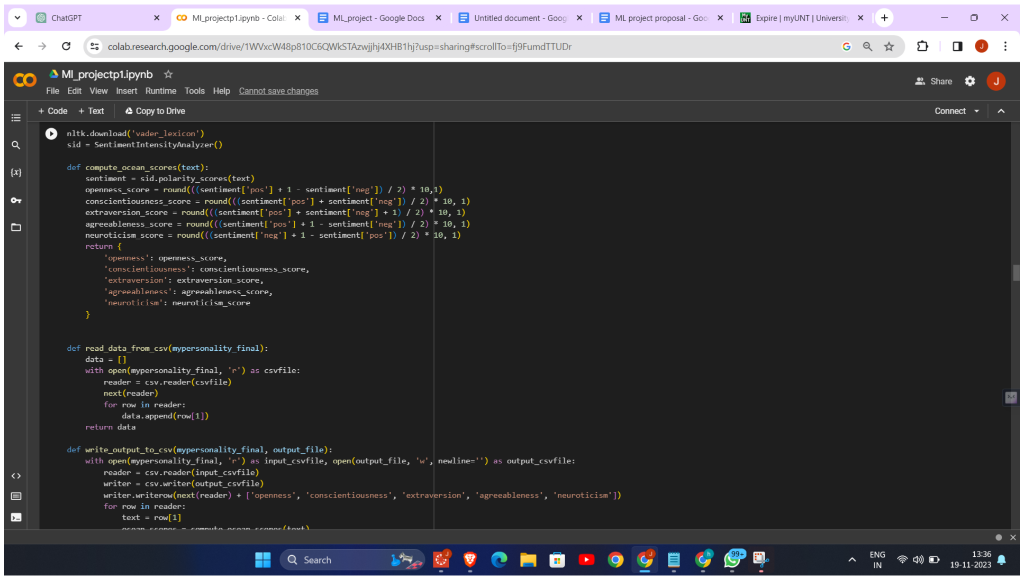
5. Diverse Feature Selection Methods: The code showcases the application of diverse feature selection methods, each with its unique approach, providing a comprehensive analysis of the text data and demonstrating the versatility of feature selection in text analysis.



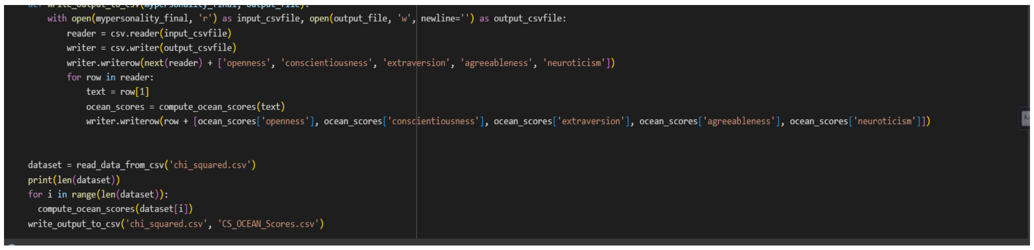
```python
nltk.download('vader_lexicon')
sid = SentimentIntensityAnalyzer()

def compute_ocean_scores(text):
    sentiment = sid.polarity_scores(text)
    openness_score = round(((sentiment['pos'] + 1 - sentiment['neg']) / 2) * 10,1)
    conscientiousness_score = round(((sentiment['pos'] + sentiment['neg']) / 2) * 10, 1)
    extraversion_score = round(((sentiment['pos'] + sentiment['neg'] + 1) / 2) * 10, 1)
    agreeableness_score = round(((sentiment['pos'] + 1 - sentiment['neg']) / 2) * 10, 1)
    neuroticism_score = round(((sentiment['neg'] + 1 - sentiment['pos']) / 2) * 10, 1)
    return {
        'openness': openness_score,
        'conscientiousness': conscientiousness_score,
        'extraversion': extraversion_score,
        'agreeableness': agreeableness_score,
        'neuroticism': neuroticism_score
    }


def read_data_from_csv(mypersonality_final):
    data = []
    with open(mypersonality_final, 'r') as csvfile:
        reader = csv.reader(csvfile)
        next(reader)
        for row in reader:
            data.append(row[1])
    return data

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)
```



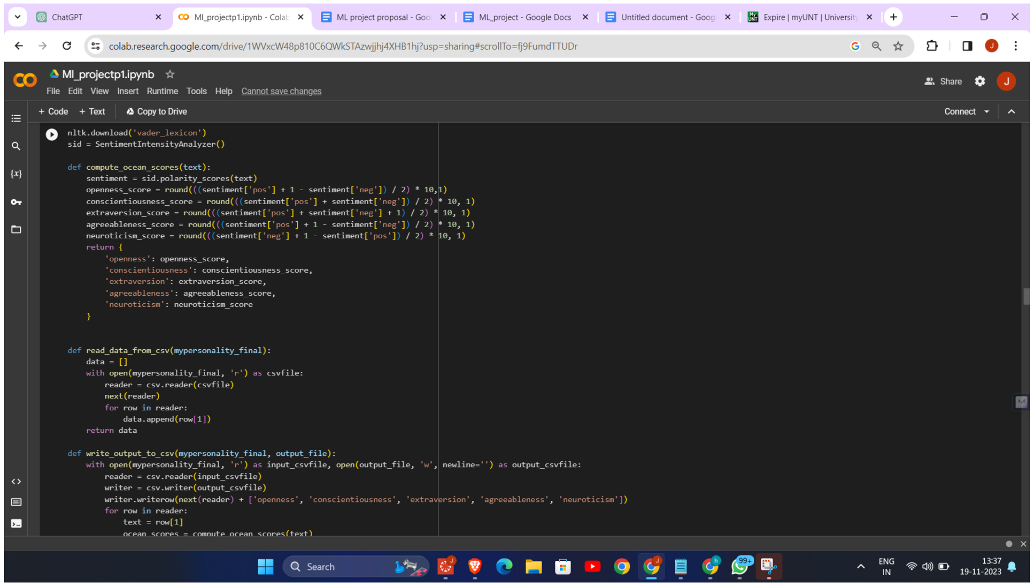
```python
def write_output_to_csv(mypersonality_final, output_file):
    with open(mypersonality_final, 'r') as input_csvfile, open(output_file, 'w', newline='') as output_csvfile:
        reader = csv.reader(input_csvfile)
        writer = csv.writer(output_csvfile)
        writer.writerow(next(reader) + ['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')
```



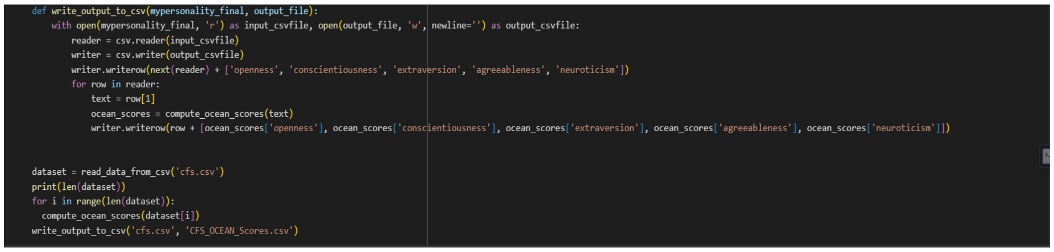
```python
nltk.download('vader_lexicon')
sid = SentimentIntensityAnalyzer()

def compute_ocean_scores(text):
    sentiment = sid.polarity_scores(text)
    openness_score = round(((sentiment['pos'] + 1 - sentiment['neg']) / 2) * 10,1)
    conscientiousness_score = round(((sentiment['pos'] + sentiment['neg']) / 2) * 10, 1)
    extraversion_score = round(((sentiment['pos'] + sentiment['neg'] + 1) / 2) * 10, 1)
    agreeableness_score = round(((sentiment['pos'] + 1 - sentiment['neg']) / 2) * 10, 1)
    neuroticism_score = round(((sentiment['neg'] + 1 - sentiment['pos']) / 2) * 10, 1)
    return {
        'openness': openness_score,
        'conscientiousness': conscientiousness_score,
        'extraversion': extraversion_score,
        'agreeableness': agreeableness_score,
        'neuroticism': neuroticism_score
    }


def read_data_from_csv(mypersonality_final):
    data = []
    with open(mypersonality_final, 'r') as csvfile:
        reader = csv.reader(csvfile)
        next(reader)
        for row in reader:
            data.append(row[1])
    return data

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]
```



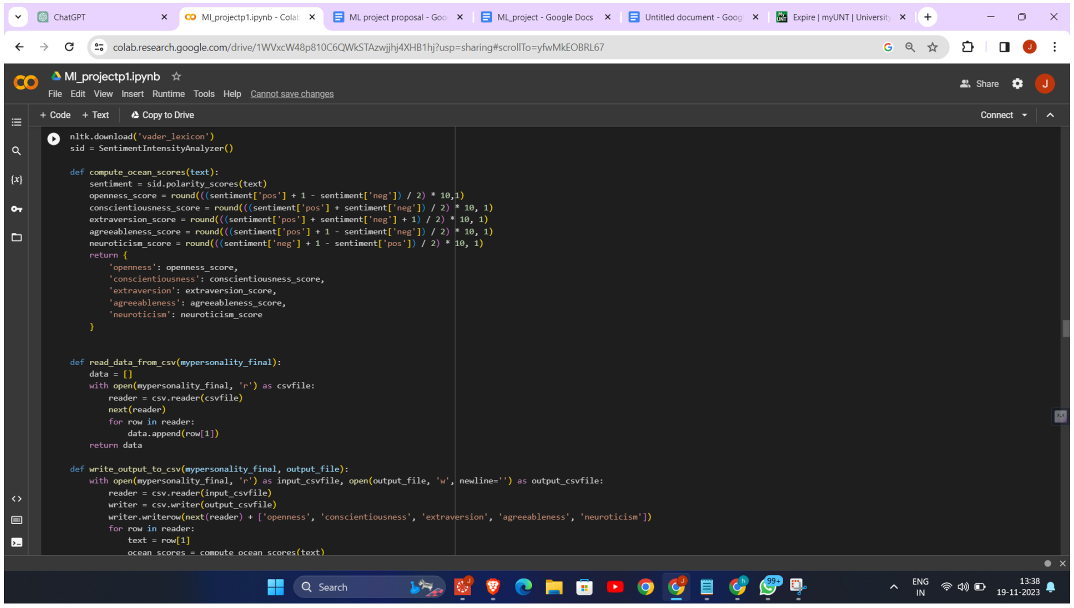
```python
    with open(mypersonality_final, 'r') as input_csvfile, open(output_file, 'w', newline='') as output_csvfile:
        reader = csv.reader(input_csvfile)
        writer = csv.writer(output_csvfile)
        writer.writerow(next(reader) + ['openness', 'conscientiousness', 'extraversion', 'agreeableness', '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')
```



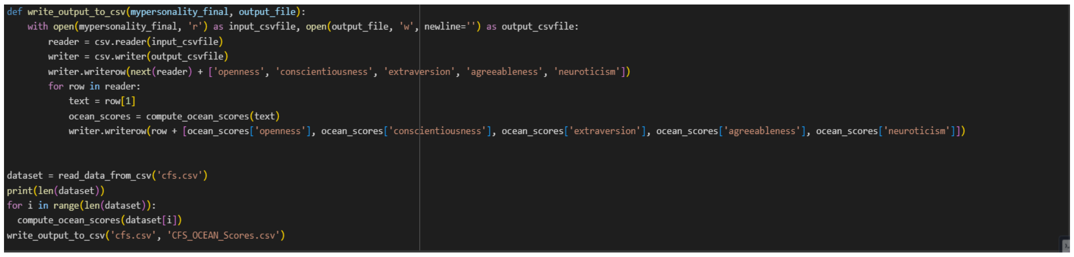
```python
nltk.download('vader_lexicon')
sid = SentimentIntensityAnalyzer()

def compute_ocean_scores(text):
    sentiment = sid.polarity_scores(text)
    openness_score = round(((sentiment['pos'] + 1 - sentiment['neg']) / 2) * 10,1)
    conscientiousness_score = round(((sentiment['pos'] + sentiment['neg']) / 2) * 10, 1)
    extraversion_score = round(((sentiment['pos'] + sentiment['neg'] + 1) / 2) * 10, 1)
    agreeableness_score = round(((sentiment['pos'] + 1 - sentiment['neg']) / 2) * 10, 1)
    neuroticism_score = round(((sentiment['neg'] + 1 - sentiment['pos']) / 2) * 10, 1)
    return {
        'openness': openness_score,
        'conscientiousness': conscientiousness_score,
        'extraversion': extraversion_score,
        'agreeableness': agreeableness_score,
        'neuroticism': neuroticism_score
    }


def read_data_from_csv(mypersonality_final):
    data = []
    with open(mypersonality_final, 'r') as csvfile:
        reader = csv.reader(csvfile)
        next(reader)
        for row in reader:
            data.append(row[1])
    return data

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)
```



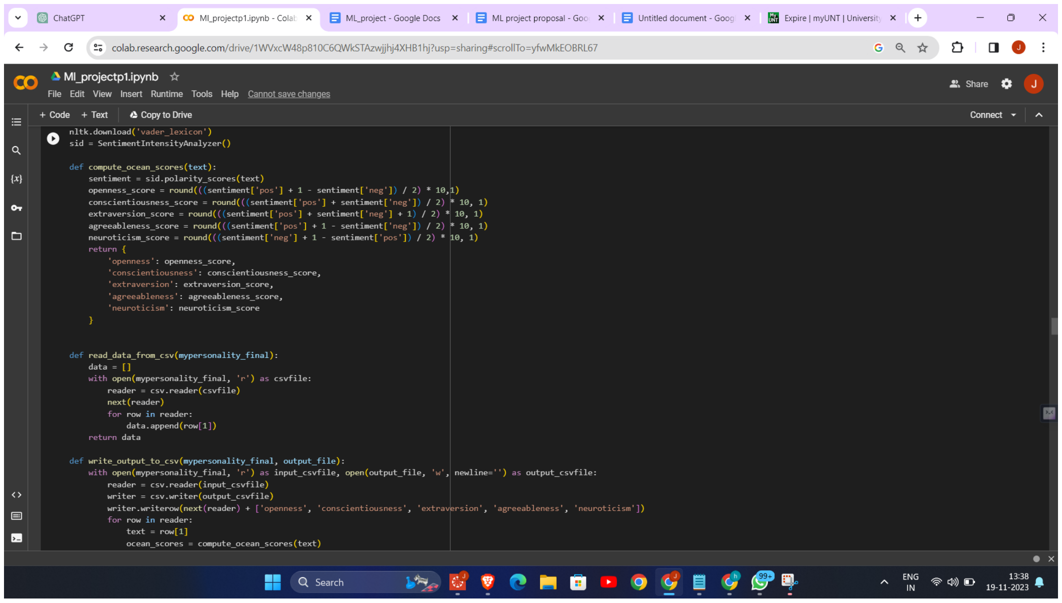
```python
def write_output_to_csv(mypersonality_final, output_file):
    with open(mypersonality_final, 'r') as input_csvfile, open(output_file, 'w', newline='') as output_csvfile:
        reader = csv.reader(input_csvfile)
        writer = csv.writer(output_csvfile)
        writer.writerow(next(reader) + ['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')
```



```python
nltk.download('vader_lexicon')
sid = SentimentIntensityAnalyzer()

def compute_ocean_scores(text):
    sentiment = sid.polarity_scores(text)
    openness_score = round(((sentiment['pos'] + 1 - sentiment['neg']) / 2) * 10,1)
    conscientiousness_score = round(((sentiment['pos'] + sentiment['neg']) / 2) * 10, 1)
    extraversion_score = round(((sentiment['pos'] + sentiment['neg'] + 1) / 2) * 10, 1)
    agreeableness_score = round(((sentiment['pos'] + 1 - sentiment['neg']) / 2) * 10, 1)
    neuroticism_score = round(((sentiment['neg'] + 1 - sentiment['pos']) / 2) * 10, 1)
    return {
        'openness': openness_score,
        'conscientiousness': conscientiousness_score,
        'extraversion': extraversion_score,
        'agreeableness': agreeableness_score,
        'neuroticism': neuroticism_score
    }


def read_data_from_csv(mypersonality_final):
    data = []
    with open(mypersonality_final, 'r') as csvfile:
        reader = csv.reader(csvfile)
        next(reader)
        for row in reader:
            data.append(row[1])
    return data

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)
```

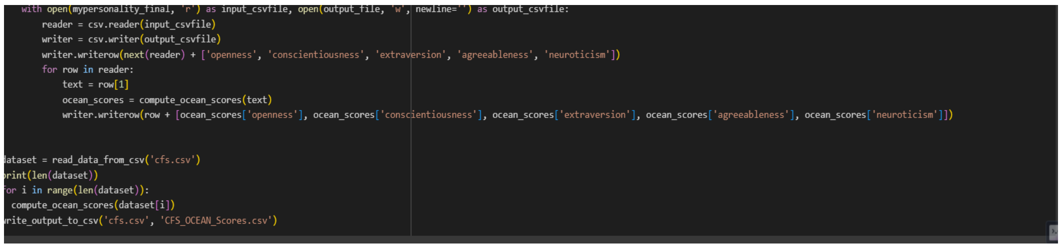


```
def write_output_to_csv(mypersonality_final, output_file):
    with open(mypersonality_final, 'r') as input_csvfile, open(output_file, 'w', newline='') as output_csvfile:
        reader = csv.reader(input_csvfile)
        writer = csv.writer(output_csvfile)
        writer.writerow(next(reader) + ['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')
```



ChatGPT
Ml_projectp1.ipynb - Colab
ML_project - Google Docs
ML project proposal - Goo
Untitled document - Goog
Expire | myUNT | University
colab.research.google.com/drive/1WVxcW48p810C6QWkSTAzwjjhj4XHB1hj?usp=sharing#scrollTo=yfwMkEOBRL67
Ml_projectp1.ipynb
File Edit View Insert Runtime Tools Help Cannot save changes
Share
+ Code + Text
Copy to Drive
Connect
Search
ENG IN
13:38
19-11-2023

```
nltk.download('vader_lexicon')
sid = SentimentIntensityAnalyzer()

def compute_ocean_scores(text):
    sentiment = sid.polarity_scores(text)
    openness_score = round(((sentiment['pos'] + 1 - sentiment['neg']) / 2) * 10,1)
    conscientiousness_score = round(((sentiment['pos'] + sentiment['neg']) / 2) * 10, 1)
    extraversion_score = round(((sentiment['pos'] + sentiment['neg'] + 1) / 2) * 10, 1)
    agreeableness_score = round(((sentiment['pos'] + 1 - sentiment['neg']) / 2) * 10, 1)
    neuroticism_score = round(((sentiment['neg'] + 1 - sentiment['pos']) / 2) * 10, 1)
    return {
        'openness': openness_score,
        'conscientiousness': conscientiousness_score,
        'extraversion': extraversion_score,
        'agreeableness': agreeableness_score,
        'neuroticism': neuroticism_score
    }


def read_data_from_csv(mypersonality_final):
    data = []
    with open(mypersonality_final, 'r') as csvfile:
        reader = csv.reader(csvfile)
        next(reader)
        for row in reader:
            data.append(row[1])
    return data

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)
```



```
  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']])


ataset = read_data_from_csv('cfs.csv')
rint(len(dataset))
or i in range(len(dataset)):
 compute_ocean_scores(dataset[i])
rite_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.

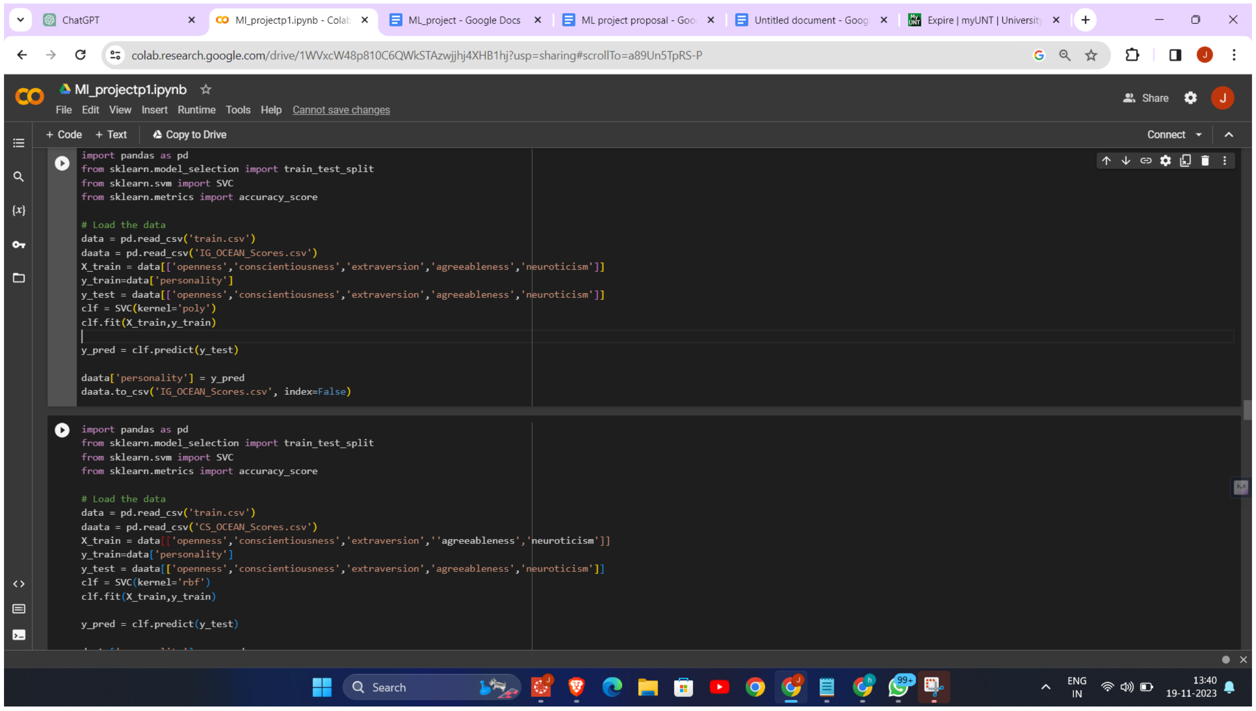
2. Data Processing and Transformation: The code reads text data from CSV files, processes each text entry to compute its OCEAN scores based on sentiment analysis, and then appends these scores to the original data.

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.

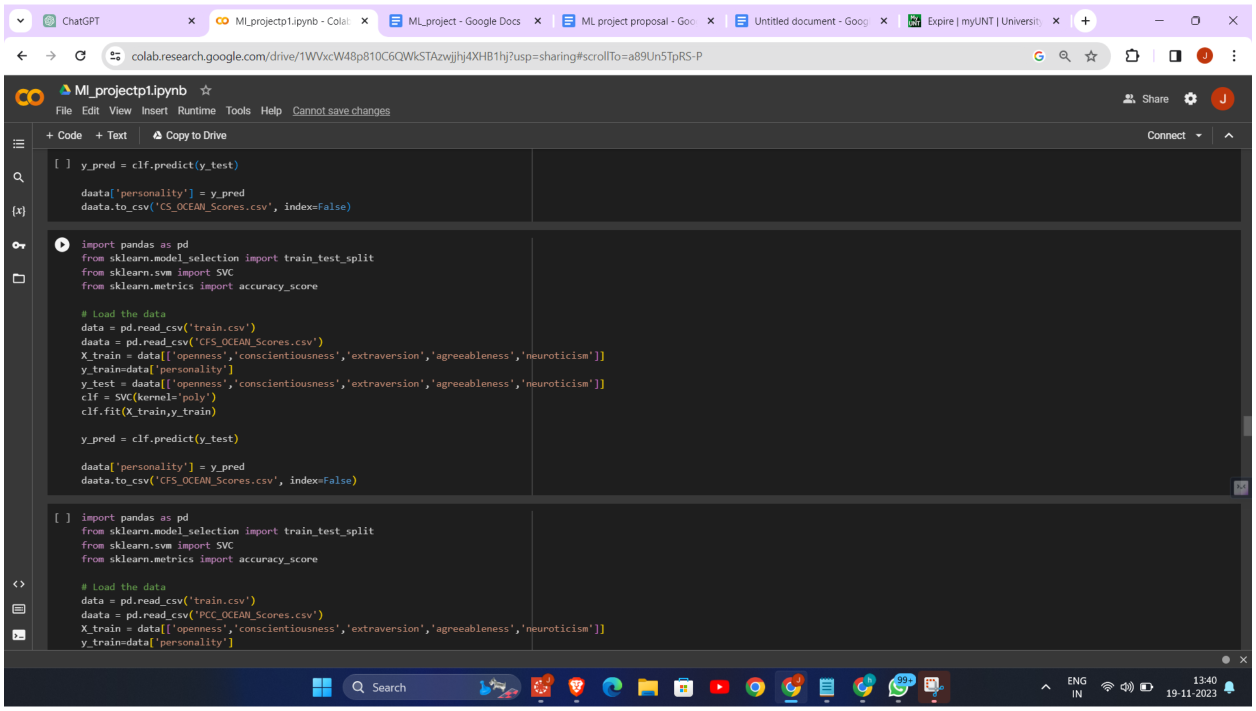
5. Versatility in Text Analysis: This approach demonstrates a novel use of sentiment analysis, extending beyond traditional sentiment scoring to infer personality traits from text, showcasing the versatility and potential of natural language processing techniques in psychological profiling.

PERSONALITY prediction


ChatGPT
Ml_projectp1.ipynb - Colab
ML_project - Google Docs
ML project proposal - Goo
Untitled document - Goog
Expire | myUNT | University
colab.research.google.com/drive/1WVxcW48p810C6QWkSTAzwjjhj4XHB1hj?usp=sharing#scrollTo=a89Un5TpRS-P
Ml_projectp1.ipynb
File Edit View Insert Runtime Tools Help Cannot save changes
Share
+ Code + Text Copy to Drive
Connect
Search
ENG IN
13:40 19-11-2023

```
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)
```

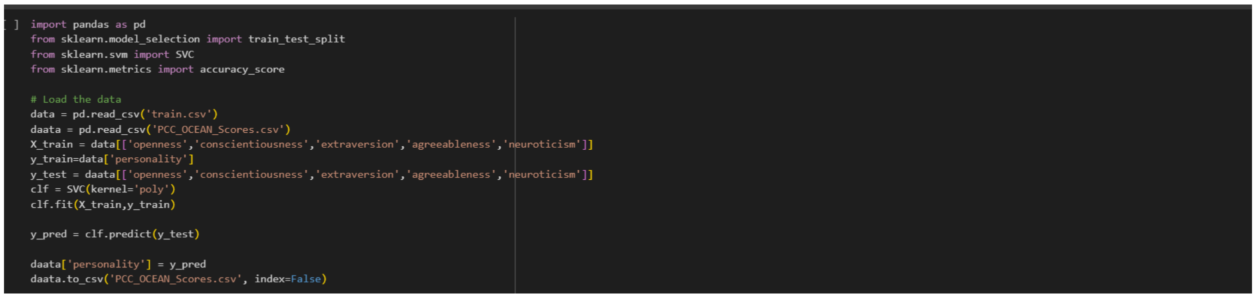


ChatGPT
Ml_projectp1.ipynb - Colab
ML_project - Google Docs
ML project proposal - Goo
Untitled document - Goog
Expire | myUNT | University
colab.research.google.com/drive/1WVxcW48p810C6QWkSTAzwjjhj4XHB1hj?usp=sharing#scrollTo=a89Un5TpRS-P
Ml_projectp1.ipynb
File Edit View Insert Runtime Tools Help Cannot save changes
Share
+ Code + Text
Copy to Drive
Connect
Search
ENG IN
13:40 19-11-2023

```
[ ] 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.

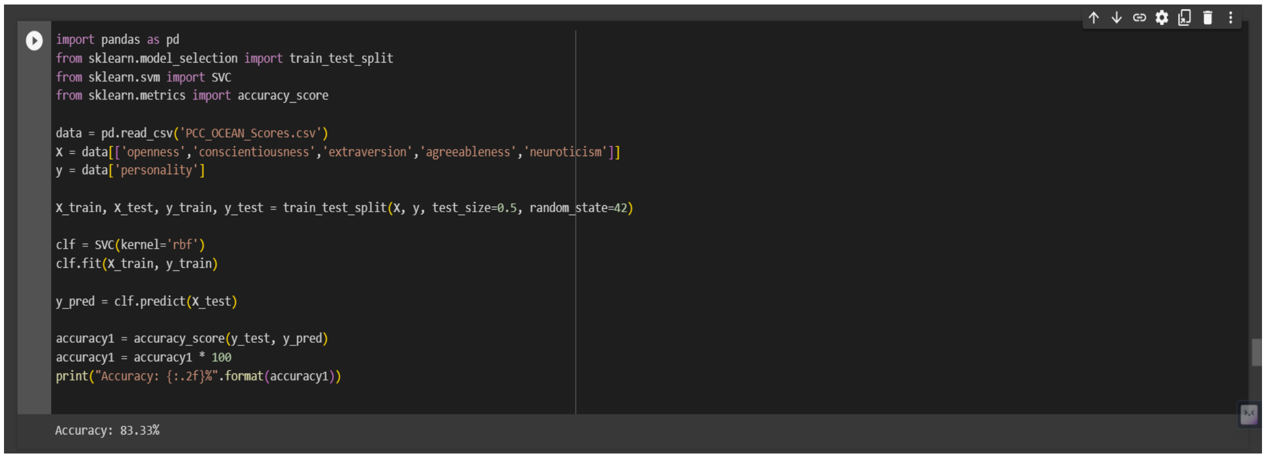
3. Personality Prediction: The trained SVM model is then used to predict personality types for entries in each of the four datasets based on their OCEAN scores.

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.

PRELIMINARY RESULTS

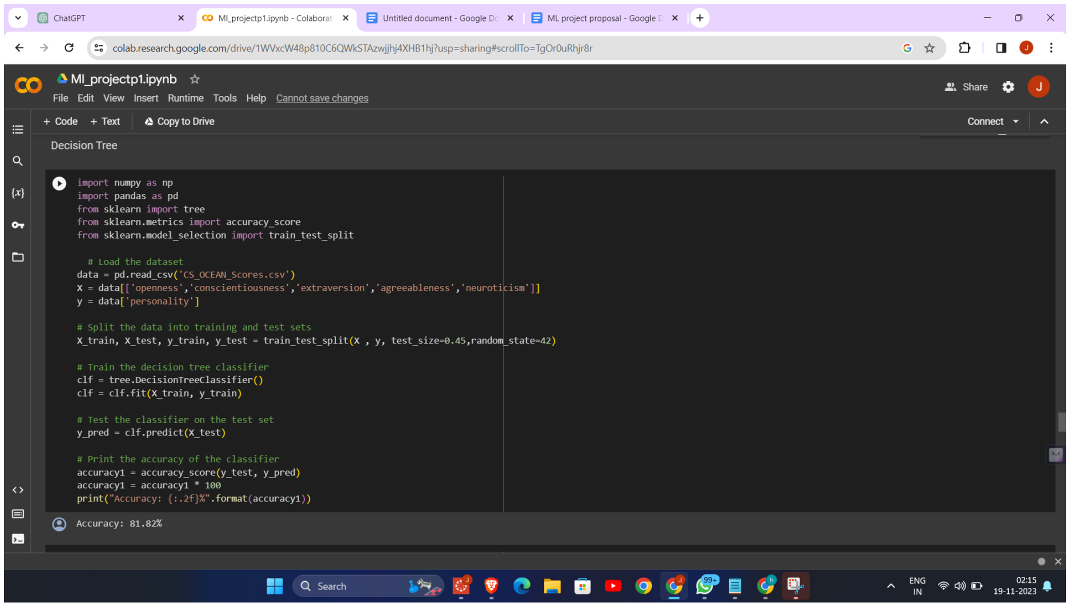
SVM- PCC



```
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%
```

DECISION TREE - CHI SQUARED

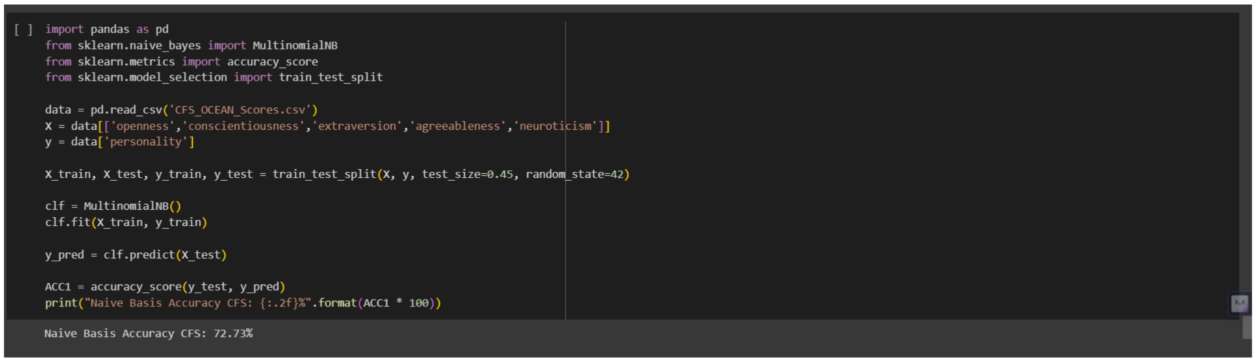


ChatGPT
Ml_projectp1.ipynb - Colaborat
Untitled document - Google Do
ML project proposal - Google D
colab.research.google.com/drive/1WVxcW48p810C6QWkSTAzwjjhj4XHB1hj?usp=sharing#scrollTo=TgOr0uRhjr8r
Ml_projectp1.ipynb
File Edit View Insert Runtime Tools Help Cannot save changes
Share
+ Code + Text
Copy to Drive
Connect
Decision Tree
Search
ENG IN
02:15 19-11-2023

```
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('CS_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%
```

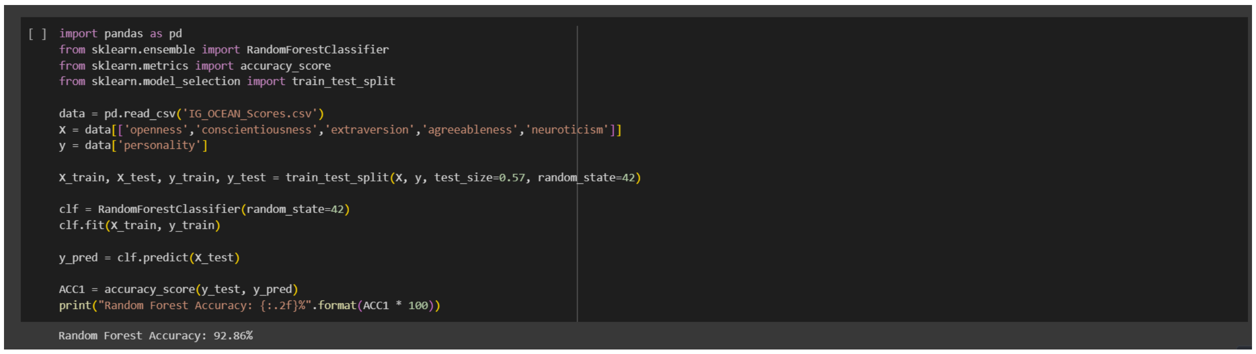
NAIVE BAYSE - CFS



```
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%
```

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	-	-	-	83.33
RANDOM FOREST	92.86	-	-	-
NAIVE BAYES	-	72.73	-	-
DECISION TREE	-	-	81.82	-

Results Interpretation:

- SVM with PCC: Achieved 83.33% accuracy, indicating that SVM performed reasonably well with features selected based on Pearson Correlation Coefficient.
- Random Forest with CFS: Achieved the highest accuracy of 92.86%, suggesting that Random Forest is very effective when combined with Correlation-based Feature Selection.

- Naive Bayes with Chi-Square: Had an accuracy of 72.73%, showing moderate effectiveness of Naive Bayes with Chi-Square feature selection.
- Decision Tree with Chi-Square: Scored 81.82% accuracy, indicating a good performance, especially considering the simplicity of Decision Trees.

These results suggest that the choice of feature selection technique and machine learning model significantly impacts the performance of a predictive model. The high accuracy of Random Forest with CFS might indicate that this combination is particularly effective for the type of data and problem being addressed.

Project Management

- Work Completed

  As a team, we have accomplished 70% of the project, with a strong focus on implementing machine learning algorithms to the feature selection methods. The rest 30% implementation part involves implementing the other machine learning algorithms on the feature selection algorithms. Everyone contributed equally to the project.

  - Tagore Hari Prasad Chintamaneni
    - Implemented the CHI-Squared test feature selection technique and created a Decision Tree model based on the features selected by the Chi-Squared test.

  - Ajay Eedara
    - Implemented the Information Gain technique and created a Random Forest model based on the features selected by the Information Gain.

  - Jaideep Tripurani
    - Implemented the CFS based subset evaluator technique and created a Naive bayes model based on the features selected by the CFS based subset evaluator.

  - Pavan Kumar Manam
    - Implemented the Pearson Correlation coefficient technique and created a Support vector machine model based on the features selected by the Pearson Correlation coefficient.

- Works to be Completed.
    - Tagore Hari Prasad Chintamaneni
        - Needs to implement the machine learning algorithms developed by the remaining team members on his feature selection algorithm (CHI-squared test)
    - Ajay Eedara
        - Needs to implement the machine learning algorithms developed by the remaining team members on his feature selection algorithm (Information Gain)
    - Jaideep Tripurani
        - Needs to implement the machine learning algorithms developed by the remaining team members on his feature selection algorithm (CFS based subset evaluator).
    - Pavan Kumar Manam
        - Needs to implement the machine learning algorithms developed by the remaining team members on his feature selection algorithm (Pearson Correlation Coefficient)

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