System Requirements Specification Index

For

NLP/ML Usecase

Wine variety analysis with NLP and ML L2

1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

Problem Statement	: **Wine variety analysis with NLP and ML**
Description	: Use relevant methods operations to perform specified activities which are given in the instructions.

Use Case Scenario: Wine Reviews Classification with NLP Insights
In this use case, you're tasked with building a machine learning pipeline that analyze and classifies wine reviews based on several features, including the wine's title, description, points, and price. The project involves preprocessing textual data (wine reviews), extracting meaningful insights using NLP, and training a machine learning model. Additionally, the project includes performing various NLP-based analyses to understand key patterns in the reviews.

What you will gain from this code (Knowledge analysis)
Natural Language Processing (NLP)

- Tokenization: Learn how to tokenize sentences into words using nltk.word_tokenize().
- Stop word removal: Use NLTK's stop words to remove common words that do not contribute meaningfully to the text.
- TF-IDF Vectorization: Understand how to convert text data into numerical form using TfidfVectorizer() for machine learning models.
- Parts of Speech Tagging: Learn how to use NLTK to identify adjectives in text and perform basic POS tagging.

Machine Learning Model Training

- Label encoding: Use LabelEncoder() to convert categorical labels (wine varieties) into numerical form for classification.
- Random Forest Classifier: Learn how to build and train a machine learning model using RandomForestClassifier() from sklearn.
- Feature augmentation: Understand how to augment TF-IDF text vectors with additional numerical features (e.g., points, price) using hstack() from scipy.

Project folder structure.

:-Wineanalysis

Winenlptemplate.py

Test500.csv

Train500likes.csv

Wine_model.pkl(make sure this is generated)

Test folder.

Key Objectives:

1. **Load and Preprocess Data**
2. **NLP Feature Extraction**: Tokenize and remove stopwords, then vectorize text using TF-IDF.
3. **Model Training**: Use a Random Forest Classifier to predict wine varieties
4. **NLP Insights**: Gain insights from the textual data, such as most frequent words, longest reviews, and most common adjectives.

5. **Model Deployment**: Save the trained model for future use.

Train Data (train500likes.csv)

- **Purpose**: The training data is used to build the machine learning model. The model learns patterns from this dataset, including how features such as wine reviews, points, and price are related to the target variable (in this case, the wine variety).
- **Typical Structure**:
    - **review_title**: The title of the wine review.
    - **review_description**: The detailed description of the wine review.
    - **points**: The score or points given to the wine.
    - **price**: The price of the wine.
    - **variety**: The target variable or label that the model is trying to predict (e.g., the variety of wine).
    - **country**: The country of origin of the wine.
    - **Likes** :based on the usage
- **Usage**: In the code, the training data is loaded, preprocessed (combining review_title and review_description), vectorized (using TF-IDF), and used to train a **Random Forest** model

Test Data (test500.csv)

- **Purpose**: The test data is used to evaluate the performance of the model after it has been trained on the training data. The test data serves as an unseen dataset that allows you to assess how well the model generalizes to new data.
- **Typical Structure**: The test data will have the same structure as the training data but without the target variable (or if the target variable is present, it's used for evaluation purposes, not training).
    - **review_title**: The title of the wine review.
    - **review_description**: The detailed description of the wine review.
    - **points**: The score or points given to the wine.
    - **price**: The price of the wine.
    - **variety**: The wine variety for evaluation purposes.
    - **country**: The country of origin of the wine.
- **Usage**: The test data is preprocessed similarly to the training data (combining review_title and review_description) and used to generate insights such as frequent words, the most common adjectives, average points, etc. The model could also be applied to the test data to make predictions, though this isn't explicitly shown in the provided code
- Note you will be given Train.csv data and Test.csv data you must you the data to get the result in the console and run the testcase

<u>**Task to be Performed**</u>

1. Create the predication model using Randomforest
2. Based on the provided code, how would you extract the most frequently used word from the test data reviews?
3. What is the longest review from the test data?
4. Calculate the average length of reviews in the test data. How does it compare to the average length in the training data?

5. Which adjective appears most frequently in the reviews from the test data?
6. What is the shortest review in the test data?
7. What is the most frequently occurring review title in the test data?
8. Which country appears most frequently in the test data?
9. Which is the Most liked wine variety (based on average points)?

Execution Steps to Follow:

1. All actions like build, compile, running application,running test cases will be through Command Terminal.

2. To open the command terminal the test takers, need to go to Application menu (Three horizontal lines at left top) -> Terminal -> New Terminal

3. This editor Auto Saves the code

4. If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use CTRL+Shift+B-command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.

5. These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.

6. To setup environment:

You can run the application without importing any packages

7. To launch application:

Commands that are used to install the library

- pip install pandas
- pip install scikit-learn
- pip install nltk
- pip install scipy
- pip install numpy
- nltk.download('punkt')
- nltk.download('stopwords')
- pip install requests

8. To run Test cases:

```
Python -m unittest
```

9. Before Final Submission also, you need to use CTRL+Shift+B-command compulsorily

on code IDE. This will push or save the updated contents in the internal git/repository for code

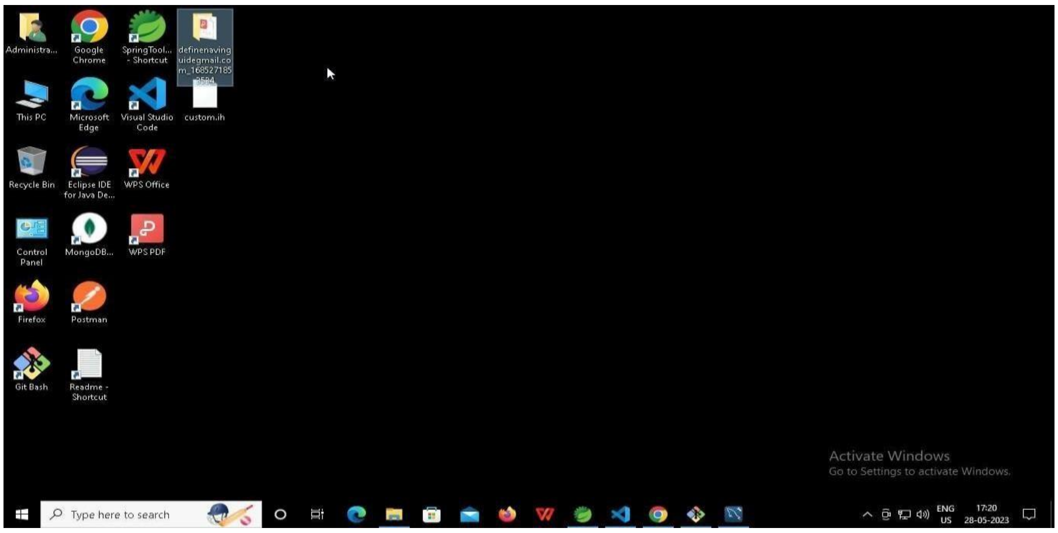
To run the application use the code

Python winenlptemplate.py   (python script name.py)

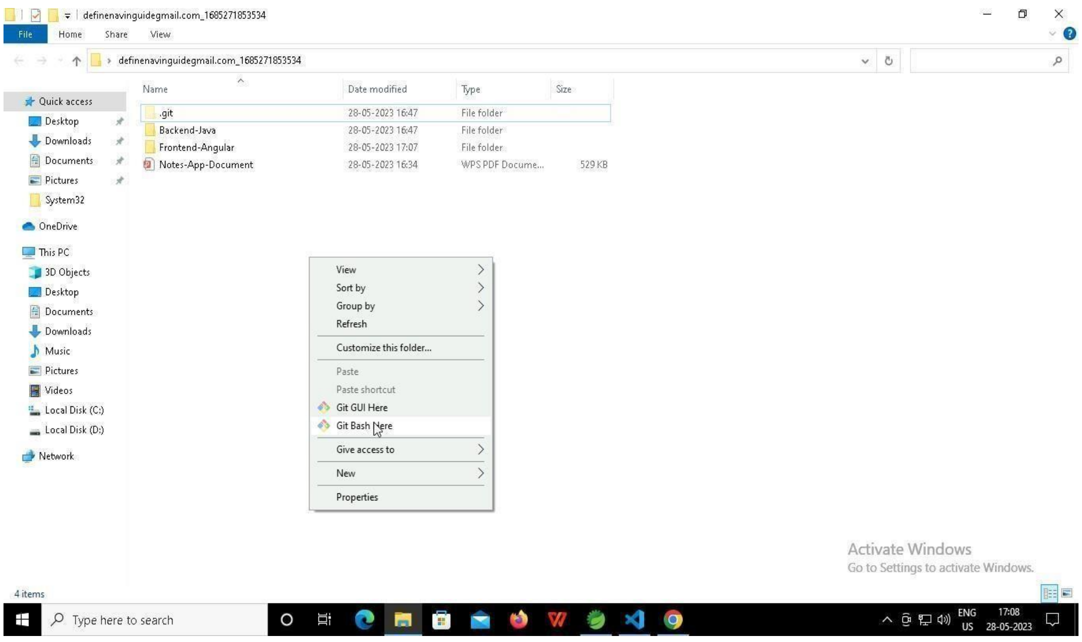
-----x-----

You can run test cases as many numbers of times and at any stage of Development, to check how many test cases are passed/failed and accordingly refactor your code.

1. **Make sure before final submission you commit all changes to git**. For that open the project folder available on desktop



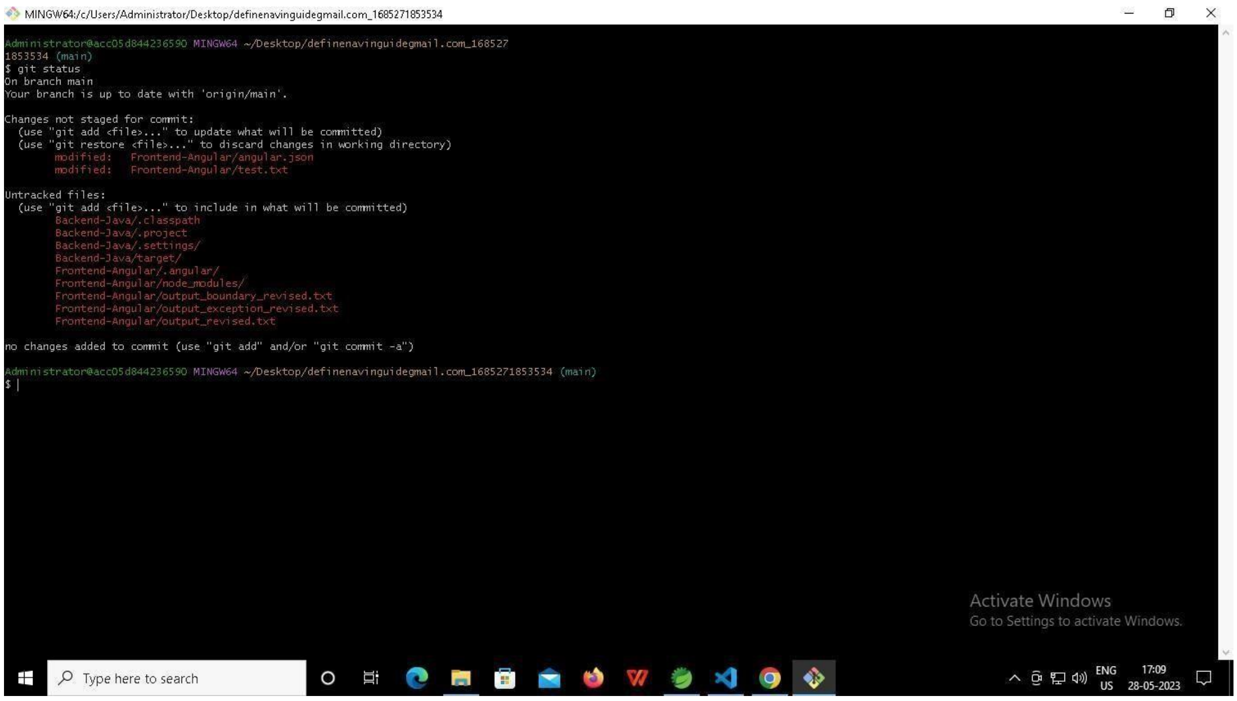
a. Right click in folder and open Git Bash



Name	Date modified	Type	Size
.git	28-05-2023 16:47	File folder	
Backend-Java	28-05-2023 16:47	File folder	
Frontend-Angular	28-05-2023 17:07	File folder	
Notes-App-Document	28-05-2023 16:34	WPS PDF Docume...	529 KB

b. In Git bash terminal, run following commands

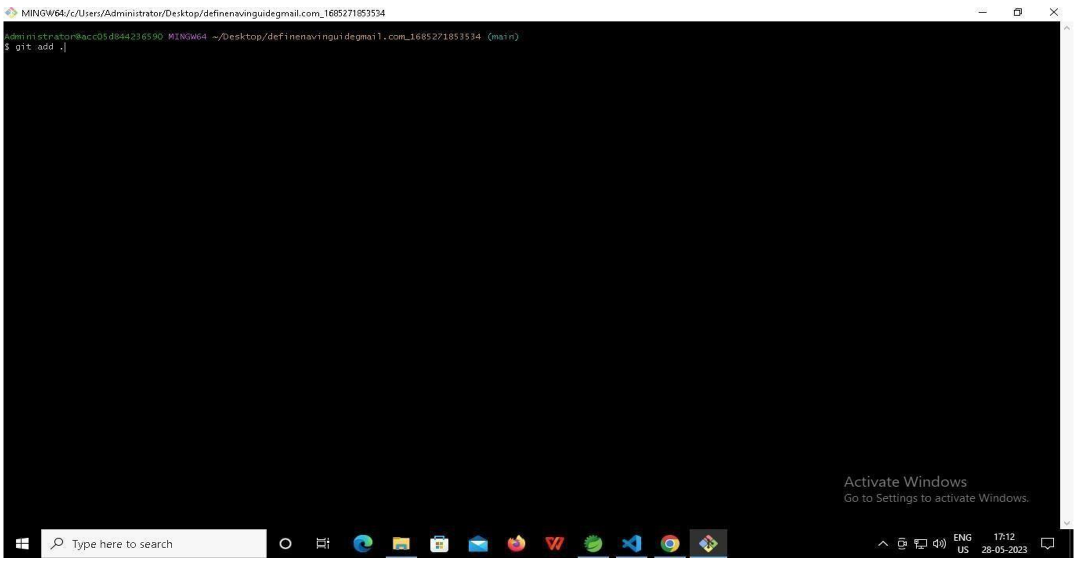
c. git status



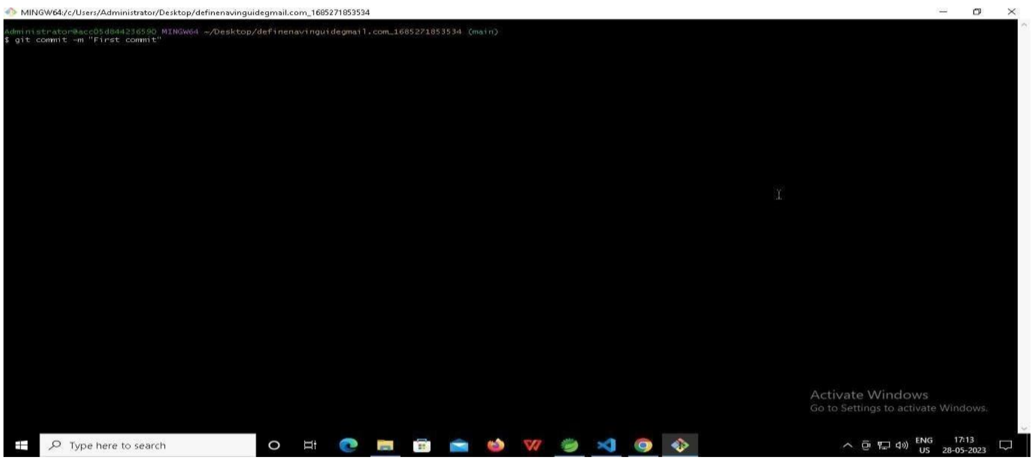
```
Administrator@acc05d844236590 MINGW64 ~/Desktop/definenavinguidegmail.com_168527
1853534 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Frontend-Angular/angular.json
        modified:   Frontend-Angular/test.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Backend-Java/.classpath
        Backend-Java/.project
        Backend-Java/.settings/
        Backend-Java/target/
        Frontend-Angular/.angular/
        Frontend-Angular/node_modules/
        Frontend-Angular/output_boundary_revised.txt
        Frontend-Angular/output_exception_revised.txt
        Frontend-Angular/output_revised.txt

no changes added to commit (use "git add" and/or "git commit -a")

Administrator@acc05d844236590 MINGW64 ~/Desktop/definenavinguidegmail.com_1685271853534 (main)
$ |
```

d. git add .

e. git commit -m “First commit”
   (You can provide any message every time you commit)

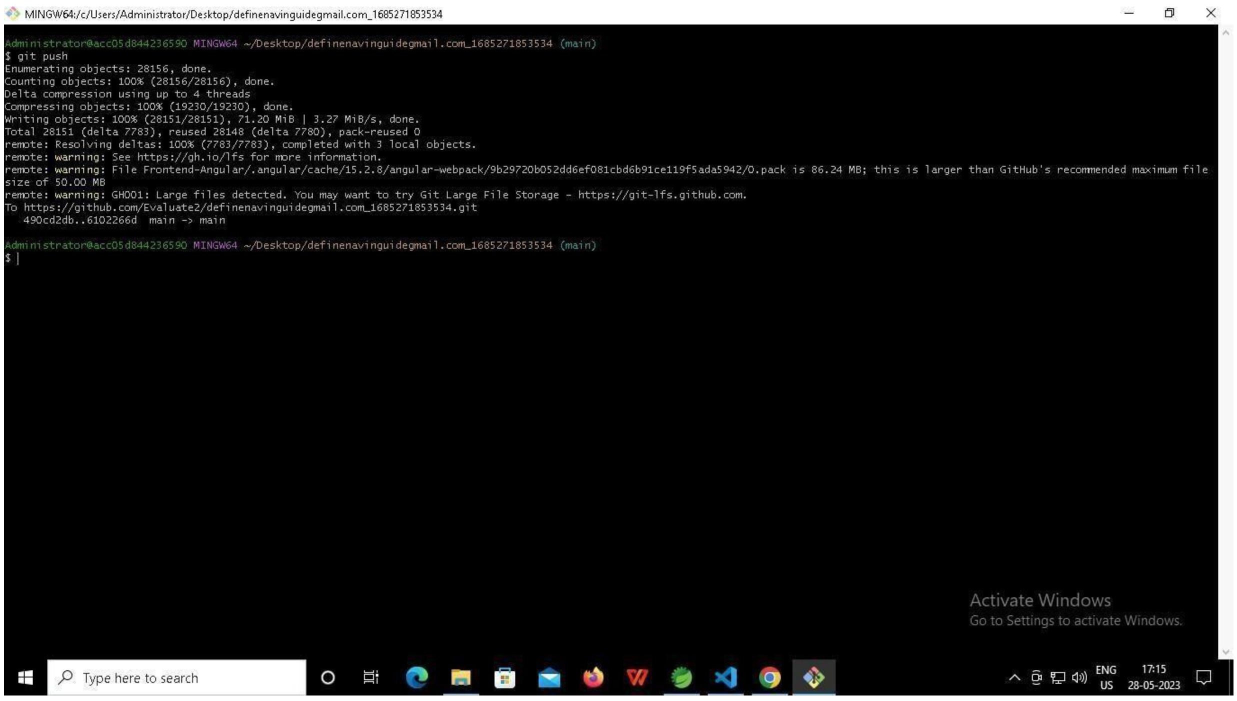


```
Administrator@acc05d844236590 MINGW64 ~/Desktop/definenavinguidegmail.com_1685271853534 (main)
$ git add .
```



```
Administrator@acc05d844236590 MINGW64 ~/Desktop/definenavinguidegmail.com_1685271853534 (main)
$ git commit -m "First commit"
```

f. git push



MINGW64:/c/Users/Administrator/Desktop/definenavinguidegmail.com_1685271853534

```
Administrator@acc05d844236590 MINGW64 ~/Desktop/definenavinguidegmail.com_1685271853534 (main)
$ git push
Enumerating objects: 28156, done.
Counting objects: 100% (28156/28156), done.
Delta compression using up to 4 threads
Compressing objects: 100% (19230/19230), done.
Writing objects: 100% (28151/28151), 71.20 MiB | 3.27 MiB/s, done.
Total 28151 (delta 7783), reused 28148 (delta 7780), pack-reused 0
remote: Resolving deltas: 100% (7783/7783), completed with 3 local objects.
remote: warning: See https://gh.io/lfs for more information.
remote: warning: File Frontend-Angular/.angular/cache/15.2.8/angular-webpack/9b29720b052dd6ef081cbd6b91ce119f5ada5942/0.pack is 86.24 MB; this is larger than GitHub's recommended maximum file
size of 50.00 MB
remote: warning: GH001: Large files detected. You may want to try Git Large File Storage - https://git-lfs.github.com.
To https://github.com/Evaluate2/definenavinguidegmail.com_1685271853534.git
   490cd2db..6102266d  main -> main

Administrator@acc05d844236590 MINGW64 ~/Desktop/definenavinguidegmail.com_1685271853534 (main)
$ |
```

-----x-----