1. The chosen model and rationale behind the selection.

Answer: I apply 3 Machine Learnig algorithm like Decision Tree Classifier, Random Forest

Classifier & Multinomialnb, Here based on the Accuracy, Precision, Recall And F1 Score

i select MultinomialNB, It's Having 99% accuracy where As other two model overfit the training data having 100% Accuracy,

Also works well in text data primarily due to its simplicity, efficiency, and the unique characteristics of text data.

Rationale for Choosing MultinomialNB:

1. Text Classification Problem: MultinomialNB is particularly well-suited for text classification tasks, where the input data consists of text documents or features with discrete counts, such as word frequencies.

2. Multiple Classes: If the problem involves classifying data into multiple classes (more than two), MultinomialNB can handle this scenario efficiently. It's an extension of the standard Naive Bayes classifier designed to work with multinomially distributed data.

3. Feature Independence Assumption: The Naive Bayes algorithm assumes feature independence, which may not hold true for all datasets. However, in text classification tasks, this assumption often works reasonably well due to the nature of language.

4. Efficient Training: MultinomialNB's training process is computationally efficient, making it suitable for large datasets or situations where training speed matters.

5. Natural Frequency Representation: In text classification, features are often represented as word frequencies or counts. MultinomialNB works well with such representations.

6. Sparse Data Handling: In text data, the feature space can be high-dimensional, and many features might have zero counts in each document. MultinomialNB is capable of handling such sparse data efficiently.

7. Baseline Model: MultinomialNB can serve as a solid baseline model for text classification tasks. It's relatively simple and can provide a decent starting point for more complex models.

2. Any preprocessing or feature extraction methods employed.

Answer:

1. Tokenization:

Tokenization involves splitting text into individual words or tokens. This is the first step in processing text data and is essential for further analysis.

2. Lowercasing:

Converting all text to lowercase helps in treating words with different cases as the same, reducing vocabulary size and simplifying comparisons.

3. Stop Word Removal:

Stop words like "and," "the," "is," etc., are common words that don't carry much information. Removing them can reduce noise and improve computational efficiency.

4. Stemming and Lemmatization:

Stemming and lemmatization reduce words to their base or root form. Stemming is a more aggressive approach, while lemmatization considers the context and morphological analysis to bring words to their base form.

5. Removing Punctuation and Special Characters:

Punctuation marks and special characters often don't contribute much to the meaning of text and can be removed.

6. Handling Contractions:

Expand contractions (e.g., "can't" to "cannot") to ensure consistent vocabulary representation.

7. TF-IDF (Term Frequency-Inverse Document Frequency):

TF-IDF is a method that combines term frequency (word frequency in a document) and inverse document frequency (inverse proportion of the number of documents containing the word) to weigh the importance of words in a corpus.

3. Instructions on how to run the script and expected outputs.

Answer:

1. Install Python 3.8.5 or Higher .

2. Clone the github Repo.

3. Open terminal into working directory and run -> pip install requirements.txt

4. After installation process in done now let's run the script.py file.

5. In Command Line / Terminal Type -> python script.py --input resume_dir --output output_dir

Note: Make sure inside resume_dir folder having all the testing resume file in pdf format.

I attach one drive link here where you find one video and see how it's works and how to run the script.

Link: https://drive.google.com/file/d/10PCEnmS_NZaYgVHl3lPuV8SPHkDhV2OI/view?usp=sharing