**Resume Selection by Naive Bayes Classifier**

**Introduction**

In today's competitive job market, efficiently screening resumes is crucial for identifying qualified candidates. In this report, we explore the use of Naive Bayes Classifier for automating the resume selection process. By leveraging natural language processing (NLP) techniques and machine learning algorithms, we aim to build a model capable of accurately classifying resumes as relevant or irrelevant based on their content.

**STEP 1: IMPORTING LIBRARIES**

**Installing NLTK, Gensim, and Wordcloud**

The initial step involves installing and importing necessary libraries for data manipulation, visualization, and text preprocessing. NLTK (Natural Language Toolkit) and Gensim are essential for text processing tasks, while Wordcloud is utilized for visualizing word frequencies in the resume dataset.

**STEP 2: LOADING THE DATASET**

**Loading the Dataset**

The dataset containing resumes and their corresponding labels (relevant or irrelevant) is loaded into the environment for further analysis and model building.

**STEP 3: PERFORMING EXPLORATORY DATA ANALYSIS**

**Exploratory Data Analysis**

Exploratory data analysis (EDA) is conducted to gain insights into the structure and characteristics of the resume dataset. This includes examining the distribution of resume labels, visualizing word frequencies, and identifying potential patterns or trends.

**STEP 4: PERFORMING DATA CLEANING**

**Preprocessing Text Data and Removing Unnecessary Words from Dataset**

**Removing Carriage Returns:**

The code resume\_df['resume\_text'] = resume\_df['resume\_text'].str.replace('\r', '') removes carriage returns (\r) from the 'resume\_text' column of the DataFrame resume\_df. Carriage returns are non-printable characters often found in text data that can interfere with text processing tasks.

**Importing and Extending Stopwords List:**

The code from nltk.corpus import stopwords imports the stopwords corpus from the NLTK library.

The stopwords.words('english') function retrieves a list of English stopwords from the NLTK corpus.

Additional custom stopwords such as 'from', 'subject', 'edu', 're', 'use', 'email', and 'com' are appended to the existing stopwords list. These custom stopwords are common in resumes but may not provide useful information for classification purposes.

**Defining Preprocessing Function:**

The preprocess(text) function is defined to perform text preprocessing tasks on resume text data.

Inside the function, an empty list result is initialized to store preprocessed tokens.

The gensim.utils.simple\_preprocess(text) function tokenizes the input text using Gensim's simple preprocessing method.

**For each token in the tokenized text:**

Tokens are filtered out if they are present in Gensim's predefined stopwords list (gensim.parsing.preprocessing.STOPWORDS).

Tokens are filtered based on length (> 2 characters) to remove very short words.

Tokens are filtered out if they are present in the custom stop words list (stop\_words).

The filtered tokens are then joined together into a single string using ' '.join(result) and returned as the preprocessed text.

This preprocessing function ensures that the resume text data is cleaned and stripped of unnecessary words and characters, making it more suitable for subsequent analysis and model training.

**STEP 5: VISUALIZING CLEANED DATASETS**

**PLOTTING THE WORDCLOUD**

**1) FOR CLASS 1:**

A WordCloud is generated to visualize the most frequent words present in resumes classified as Class 1 (relevant). The following steps are performed:

Setting Figure Size: The size of the figure for displaying the WordCloud is set to (20, 20) to ensure clarity and readability.

Generating WordCloud: A WordCloud object (wc\_class1) is created using the WordCloud library. Parameters such as max\_words, width, height, and stopwords are specified to customize the WordCloud.

Displaying WordCloud: The WordCloud is displayed using the plt.imshow() function, which visualizes the frequency of words based on their size and placement within the WordCloud.

**2) FOR CLASS 0:**

Similarly, a WordCloud is generated for resumes classified as Class 0 (irrelevant). The steps for generating and displaying the WordCloud are identical to those for Class 1.

**STEP 6: PREPARING THE DATA BY APPLYING COUNT VECTORIZATION**

**CONVERTING SENTENCES INTO TOKENIZED FORMS AND THEN CONVERTING TO NUMERICAL VALUES FOR MODEL TRAINING**

**Importing CountVectorizer:** The CountVectorizer class from the sklearn.feature\_extraction.text module is imported. CountVectorizer is used to convert text data into numerical vectors based on the frequency of words.

**Instantiating CountVectorizer:** An instance of CountVectorizer (vectorizer) is created.

**Tokenizing and Converting to Numerical Values:** The fit\_transform() method of CountVectorizer is applied to the 'cleaned' column of the resume dataset (resume\_df['cleaned']). This method tokenizes the sentences and converts them into numerical vectors. The result is stored in the variable count\_vectorized.

**Printing Feature Names:** The get\_feature\_names() method of CountVectorizer retrieves the feature names (tokens) extracted from the text data.

**Printing Processed Data:** The processed data in numerical form is printed using the toarray() method, which converts the sparse matrix representation into a dense array.

**STEP 7: TRAINING A NAIVE BAYES CLASSIFIER**

**Splitting the Data into Training and Testing Sets**

The dataset is split into features (X) and the target variable (y), where X represents the numerical vectors obtained from CountVectorizer, and y represents the 'class' column indicating the relevance of resumes. The data is further divided into training and testing sets using the train\_test\_split function from sklearn.model\_selection.

**Training a Multinomial Naive Bayes Classifier**

A Multinomial Naive Bayes classifier is chosen for training the model due to its effectiveness in handling text classification tasks. The classifier is instantiated with a specified alpha value and trained on the training data (X\_train and y\_train) using the fit() method.

**STEP 8: ASSESSING THE TRAINED MODEL**

**PLOTTING CONFUSION MATRIX**

**1) FOR TRAINING DATA**

The confusion matrix is plotted to visualize the performance of the trained Naive Bayes classifier on the training data. The following steps are performed:

Prediction on Training Data: Predictions (y\_pred\_train) are made on the training data (X\_train) using the trained classifier (Bayes\_clf).

Confusion Matrix Calculation: The confusion matrix is calculated using the confusion\_matrix function from sklearn.metrics with the actual labels (y\_train) and predicted labels (y\_pred\_train).

Heatmap Visualization: The confusion matrix is visualized as a heatmap using the sns.heatmap function from the Seaborn library. Annotations are added to indicate the number of true positive, true negative, false positive, and false negative predictions.

**2) FOR TEST DATA**

Similarly, the confusion matrix is plotted to evaluate the performance of the trained Naive Bayes classifier on the test data. Predictions (y\_pred\_test) are made on the test data (X\_test), and the confusion matrix is calculated and visualized using the same steps as for the training data. This provides insights into the classifier's ability to generalize to unseen data.

**Assessment Metrics of the Trained Model**

After evaluating the trained Naive Bayes Classifier on both training and test datasets, the following assessment metrics are obtained:

**Accuracy:**

Accuracy of the model on the train dataset: 0.95

Accuracy of the model on the test dataset: 0.84

Overall accuracy of the model: 0.84

**Precision:**

Precision of the model: 1.0

**Recall:**

Recall of the model: 0.84

**F1 Score:**

F1 score of the model: 0.91

**Conclusion:**

The Naive Bayes Classifier demonstrates high accuracy and precision in classifying resumes, with an accuracy of 84% on the test dataset. The precision of 1.0 indicates that the classifier accurately identifies relevant resumes without classifying irrelevant resumes as relevant. Overall, the classifier performs well in resume selection, showcasing its effectiveness in automating the hiring process.

These assessment metrics provide valuable insights into the performance of the trained Naive Bayes Classifier and its ability to make accurate predictions on unseen resume data.