NLP Data Analysis and Model Building

README File

Devansh Hukmani, Dhairyav Shah, Keerthi Bhuvan Koduri, Manoj Naidu Yandrapu, Naina Gupta, Pariyani Mohammed Yusuf, Pengxiang Zhang, Priya Murali, Sudhamshu Vidyananda, Sumit Kamboj, Swathi Nadimpilli

File Reader:

This Python script allows users to read various types of files including CSV, Excel (XLSX), JSON, plain text, and files hosted at a URL. It utilizes the Pandas library for data manipulation and retrieval.

Requirements:

Python 3.x

Pandas library

Requests library (for reading files from URLs)

Installation:

1. Make sure you have Python installed. If not, you can download it from [python.org](https://www.python.org/downloads/).

2. Install the required libraries by running:

```
pip install pandas requests
```

Usage:

1. Run the script.

2. Enter the file path or URL when prompted.

3. Enter the file type (CSV, XLSX, URL, JSON, Text) when prompted.

The script will then read the specified file and display the first few rows of the DataFrame.

Supported File Types:

- CSV (.csv)

- Excel (.xlsx)

- JSON (.json)

- Text (.txt)

- Files hosted at a URL

Example:

Suppose you have a CSV file named `data.csv` and you want to read it. You can simply run the script, provide the path to `data.csv` when prompted for the file path or URL, and specify `CSV` as the file type.

```
Enter the file path or URL: data.csv

Enter the file type (CSV, XLSX, URL, JSON, Text): CSV
```

Disclaimer:

This script assumes that the input files are correctly formatted and accessible. Please ensure that the file paths or URLs provided are valid and that the files are accessible.

DataFrame Information:

The script displays information about the DataFrame after reading the file. This includes details such as the DataFrame's structure, the number of rows and columns, and the data types of each column. Understanding this information is crucial for comprehending the composition and characteristics of the dataset.

To access DataFrame information, the script utilizes the `info()` method from the Pandas library.

DataFrame Descriptive Statistics:

In addition to DataFrame information, the script provides summary statistics to offer insights into the data's distribution and characteristics. Summary statistics include count, mean, standard deviation, minimum, maximum, and quartile values for numerical columns. These statistics help users understand the central tendency, dispersion, and range of values within the dataset.

The script calculates descriptive statistics using the `describe()` method from the Pandas library.

Exploratory Data Analysis (EDA):

Exploratory Data Analysis (EDA) is typically done manually rather than automated for several reasons:

1. Deeper Understanding: Manual exploration allows for a deeper understanding of the data's nuances and patterns.

2. Tailored Analysis: Analysts can focus on specific aspects of the data that are relevant to their research objectives.

3. Flexibility: Manual exploration provides flexibility in choosing analysis techniques based on the dataset's characteristics.

4. Quality Control: Human judgment is essential for identifying data quality issues and outliers that may be overlooked by automated tools.

5. Domain Expertise: Analysts can leverage their domain knowledge to ask relevant questions and interpret findings in context.

6. Communication and Interpretation: Manual exploration facilitates better communication and interpretation of results to stakeholders.

NLP Preprocessing Tool

This Python script facilitates natural language processing (NLP) preprocessing tasks on text columns within a Pandas DataFrame. It employs various NLP techniques such as tokenization, stopword removal, punctuation removal, part-of-speech (POS) tagging, and lemmatization to prepare textual data for further analysis.

Requirements:

Python 3.x

NLTK library

TextBlob library

ipywidgets library

Jupyter Notebook environment/ Google Colab (for interactive usage)

Installation:

1. Ensure you have Python installed. If not, download it from [python.org] (https://www.python.org/downloads/).

2. Install the required libraries by running:

```
pip install nltk textblob ipywidgets
```

3. Additionally, you may need to install NLTK corpora and resources. You can do this by running the following Python script once:

```
nltk.download('punkt')

nltk.download('stopwords')

nltk.download('wordnet')

nltk.download('averaged_perceptron_tagger')
```

` ` `

Usage:

1. Import the necessary libraries.

2. Ensure your textual data is stored in a Pandas DataFrame.

3. Call the `preprocess_nlp_columns(df)` function, where `df` is your DataFrame.

4. Select the text column(s) you want to preprocess using the provided checkboxes.

5. Click the "Process" button to initiate the NLP preprocessing.

6. The processed text will be stored in new columns with names prefixed by "tokenized_" and suffixed by "_lemmatized_pos".

Functionality:

Tokenization: Splits text into individual words or tokens.

Stopword Removal: Removes common stopwords from the text.

Punctuation Removal: Eliminates punctuation and special symbols from the text.

POS Tagging: Assigns part-of-speech tags to each token.

Lemmatization: Reduces words to their base or dictionary form based on their POS tags.

Disclaimer:

This script assumes the input DataFrame contains textual data in string format. It also relies on NLTK and TextBlob libraries for NLP functionalities. Ensure your environment meets the requirements and the input data is appropriately formatted before usage.

Sentiment Analysis Tool:

This Python script enables sentiment analysis on text columns within a Pandas DataFrame using the TextBlob library. Sentiment analysis helps determine the emotional tone conveyed by textual data, categorizing it as positive, neutral, or negative.

Requirements:

Python 3.x

TextBlob library

ipywidgets library

Jupyter Notebook environment/ Google Colab (for interactive usage)

Installation:

1. Ensure you have Python installed. If not, download it from [python.org](https://www.python.org/downloads/).

2. Install the required libraries by running:

```
pip install textblob ipywidgets
```

Usage:

1. Import the necessary libraries.

2. Ensure your textual data is stored in a Pandas DataFrame.

3. Call the `sentiment_analysis(df)` function, where `df` is your DataFrame.

4. Select the text column you want to analyze sentiment for using the provided dropdown menu.

5. Click the "Apply Sentiment Analysis" button to initiate the sentiment analysis.

6. A new column named `sentiment_label` will be added to the DataFrame, indicating the sentiment labels (Positive, Neutral, or Negative) for each text entry.

Functionality:

Sentiment Analysis: Utilizes TextBlob library to analyze the sentiment of text data. It categorizes text into positive, neutral, or negative sentiments based on the polarity of the text.

Disclaimer:

This script assumes the input DataFrame contains textual data in string format. It relies on the TextBlob library for sentiment analysis. Ensure your environment meets the requirements and the input data is appropriately formatted before usage.

Feature Extraction Tool:

This Python script facilitates feature extraction from textual and numerical columns within a Pandas DataFrame. It preprocesses the text data using tokenization and padding techniques and optionally incorporates numerical features for modeling tasks such as classification or regression.

Requirements:

Python 3.x

TensorFlow library (for text preprocessing)

Scikit-learn library

ipywidgets library

Jupyter Notebook environment/ Google Colab (for interactive usage)

Installation:

1. Ensure you have Python installed. If not, download it from [python.org](https://www.python.org/downloads/).

2. Install the required libraries by running:

```
pip install tensorflow scikit-learn ipywidgets
```

Usage:

1. Import the necessary libraries.

2. Ensure your dataset is stored in a Pandas DataFrame with textual and numerical columns.

3. Call the `feature_extraction(df, text_colum, numerical_features, target_column)` function, providing the DataFrame, text column, numerical features, and target column.

4. Use the provided dropdown menus to select the text column and target column.

5. Optionally, select the numerical features you want to include by checking the corresponding checkboxes.

6. Click the "Extract Features" button to trigger the feature extraction process.

7. Extracted features are stored in `X` (feature matrix) and `y` (target variable).

Functionality:

Text Data Processing: Tokenizes and pads text data to prepare it for modeling tasks.

Numerical Feature Integration: Optionally includes numerical features along with text features for modeling.

Target Variable Encoding: Encodes the target variable using label encoding.

Disclaimer:

This script assumes the input DataFrame contains both textual and numerical data. It relies on TensorFlow and Scikit-learn libraries for feature extraction and preprocessing tasks. Ensure your environment meets the requirements and the input data is appropriately formatted before usage.

Interactive Model Building and Evaluation Tool:

This Python script facilitates the construction, training, and evaluation of various machine learning and deep learning models for classification tasks using textual and numerical features from a Pandas DataFrame. It supports a wide range of models including Support Vector Machines (SVM), Logistic Regression, Decision Trees, Random Forests, XGBoost, Naive Bayes, K-Nearest Neighbors (KNN), Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN).

Requirements:

Python 3.x

Scikit-learn library

TensorFlow library (for deep learning models)

XGBoost library (for XGBoost model)

Matplotlib library (for visualization)

ipywidgets library

Jupyter Notebook environment/ Google Colab (for interactive usage)

Installation:

1. Ensure you have Python installed. If not, download it from [python.org](https://www.python.org/downloads/).

2. Install the required libraries by running:

   ```
   pip install scikit-learn tensorflow xgboost matplotlib ipywidgets
   ```

Usage:

1. Import the necessary libraries.

2. Ensure your dataset is stored in a Pandas DataFrame with both textual and numerical features along with the target variable.

3. Use the dropdown menu to select the desired model type.

4. Click on the model type to trigger the model building, training, and evaluation process.

5. The script will print the accuracy and classification report of the selected model.

Supported Models:

Convolutional Neural Networks (CNN)

Recurrent Neural Networks (RNN)

Support Vector Machines (SVM)

Logistic Regression

Decision Trees

Random Forests

XGBoost

Naive Bayes

K-Nearest Neighbors (KNN)

Functionality:

Model Building: Constructs machine learning or deep learning models based on the selected type.

Model Training: Trains the constructed model using the provided dataset.

Model Evaluation: Evaluates the trained model's performance using accuracy and classification reports.

Disclaimer:

This script assumes the input DataFrame contains both textual and numerical data, along with a target variable for classification tasks. It relies on various machine learning and deep learning libraries for model construction and evaluation. Ensure your environment meets the requirements and the input data is appropriately formatted before usage.

Automated Model Selection and Evaluation Tool:

This Python script automates the process of building, training, and evaluating multiple machine learning and deep learning models for classification tasks using textual and numerical features from a Pandas DataFrame. It supports a diverse range of models to suit different datasets and requirements.

Requirements:

Python 3.x

Scikit-learn library

TensorFlow library (for deep learning models)

XGBoost library (for XGBoost model)

ipywidgets library (for interactive model selection)

Jupyter Notebook environment/ Google Colab (for interactive usage)

Installation:

1. Ensure you have Python installed. If not, download it from [python.org](https://www.python.org/downloads/).

2. Install the required libraries by running:

```
pip install scikit-learn tensorflow xgboost ipywidgets
```

Usage:

1. Import the necessary libraries.

2. Ensure your dataset is stored in a Pandas DataFrame with both textual and numerical features along with the target variable.

3. Execute the script.

4. The script automatically builds, trains, and evaluates multiple models including CNNs, RNNs, SVMs, Logistic Regression, Decision Trees, Random Forests, XGBoost, and Naive Bayes.

5. It selects the best-performing model based on accuracy and prints its details along with the classification report.

Supported Models:

Convolutional Neural Networks (CNN)

Recurrent Neural Networks (RNN)

Support Vector Machines (SVM)

Logistic Regression

Decision Trees

Random Forests

XGBoost

Naive Bayes

Functionality:

Automated Model Selection: Automatically builds, trains, and evaluates various models.

Best Model Identification: Identifies the best-performing model based on accuracy.

Detailed Evaluation: Provides accuracy scores and classification reports for all models.

Disclaimer:

This script assumes the input DataFrame contains both textual and numerical data, along with a target variable for classification tasks. It relies on various machine learning and deep learning libraries for model construction and evaluation. Ensure your environment meets the requirements and the input data is appropriately formatted before usage.