**Importing Libraries**

A screenshot of a computer

Description automatically generated

A machine learning project starts with importing libraries and modules. These libraries are essential for data processing, visualisation, model building, and evaluation.

* **Core Libraries**: Text processing, numeric computations, file handling, and data manipulation all employ core libraries like os, pandas, numpy, and re.
* **Text Preprocessing**: Natural Language Toolkit features like stopwords and WordNetLemmatizer remove unnecessary words and simplify text.
* **Visualization**: Matplotlib and seaborn create advanced visualisations like graphs, charts, and word clouds to understand data patterns and linkages.
* **Data Splitting and Encoding** Train\_test\_split divides data into training and testing sets, while LabelEncoder numericalizes category labels.
* **Text Vectorization:** Text-based machine learning requires TfidfVectorizer to turn text into numerical features using Term Frequency-Inverse Document Frequency.
* **Machine Learning Models**: Scikit-learn provides classifiers like RandomForestClassifier, SVC, and LogisticRegression, while Keras delivers deep learning with Sequential, Dense, and Dropout layers.
* **Evaluation Metrics**:Using metrics like classification\_report and accuracy\_score, model performance can be assessed.
* **Warnings and Utilities**: Warnings and utilities: deactivate unnecessary signals and store trained models for pickle deployment. It includes everything for a machine learning workflow.

**Data collection - (Jay Dilipbhai Yadav)**

A screenshot of a computer

Description automatically generated

This code segment shows how to load a dataset for text-based cyber threat identification using machine learning. The dataset is downloaded from Kaggle using kagglehub. This function checks the downloaded.zip file. The zipfile module extracts the contents to a useable directory.

The dataset file path is dynamically constructed using Python's os.path methods for system compatibility. Using pd.read\_csv, a sophisticated structured data method, the dataset is read into a Pandas DataFrame. The top few rows of the dataset are displayed using data.head() to indicate structure.

Text (material to analyse) and label (category(s) like "malware," "attack-pattern," or "TIME") are columns in the output dataset. Entity, relation, and Comment columns provide entry metadata. The missing values (id, label) in some rows must be addressed during data preprocessing. This prepares data for cleaning, feature engineering, and machine learning.

A screenshot of a computer

Description automatically generated

This code demonstrates a custom balanced data sampling function, ensuring that each dataset class has a maximum number of samples. The dataset (data), label column (label\_column), and required number of samples per class are required by the function sample\_by\_class.

The function iterates across each label column class class. It splits rows by class and randomly selects up to the specified number of samples (or fewer if the class has fewer rows). A new DataFrame is created by joining these sampled rows, ensuring proportionality across all classes.

The dataset is applied with the label column set to 'label' and the number of samples per class set to 2500. The function is defined. The balanced dataset Sampled\_data has up to 2500 samples for all classes. Using sampled\_data, display the first five balanced dataset rows.

This approach works well with imbalanced datasets when some classes dominate. Balanced datasets prevent over-represented classes from biassing machine learning class models, improving performance and fairness.

**Data Cleaning-(Jay Dilipbhai Yadav)**

A screenshot of a computer

Description automatically generated

A dataset with missing values is shown in this sample. Eliminating null rows improves data quality and consistency.

The function data.isna().sum() checks each dataset column for missing values. For dataset completeness, this function counts column nulls. Zero missing values in all columns indicate no null values in the dataset.

The code deletes null rows using data.dropna(inplace=True) to demonstrate data cleansing. The inplace=True option allows processing without dataset copying. Missing values are examined again using dataset operation.isna().sum(). A clean dataset for processing has no missing values in any output columns.

Handling missing values in data preprocessing is a standard step, ensuring that machine learning models receive high-quality, consistent input data, improving performance and dependability.

A screenshot of a computer

Description automatically generated

In this code sample, meaningful columns and text preprocessing prepare the dataset for text-based machine learning.

The initial step only extracts the text and label columns from the dataset. The text column contains data, while the label column contains categories or classes. That makes the dataset appropriate for text analysis and categorisation. The dataset's first rows are displayed using data.head() for structure.

Next, a custom function preprocess\_text is defined to clean and standardize the text data. It performs the following steps:

1. **Lowercase Conversion**: Converts all characters to lowercase for uniformity, removing case sensitivity.
2. **Removal of Special Characters and Numbers**: Uses a regular expression (re.sub) to strip non-alphabetic characters and numbers, leaving only text.
3. **Stopword Removal**: Eliminates common words (e.g., "the," "and") that do not add meaningful information for classification.
4. **Lemmatization**: Reduces words to their base or root form using WordNetLemmatizer (e.g., "running" becomes "run").

Meaningful columns and text preprocessing prepare this dataset for text-based machine learning.

Only the text and label columns are extracted from the dataset in the first step. The data column contains data, while the class label contains classes.

A screenshot of a computer program

Description automatically generated

Prepare machine learning datasets by preprocessing raw text data. When the preprocess\_text function is applied to the text column of the dataset, a new process column called clean\_text is produced. This column contains original text process data.

To clarify dataset structure, the new clean\_text column follows text. The right index is used to enter the clean\_text column using data.columns.get\_loc('text'). While ensuring that processed data is freely accessible next to raw data for comparison and verification, the original order of the columns is maintained.

The result is a dataset with three columns:

1. **text**: The original unprocessed text data.
2. **clean\_text**: The preprocessed and cleaned version of the text data (e.g., lowercased, lemmatized, and devoid of special characters and stopwords).
3. **label**: The categorical label indicating the classification for each row.

Displaying the first few rows using data.head() shows the transformation's success. This cleaned dataset is now optimized for feature extraction and subsequent machine learning tasks.