**Fake Review Classification and Topic Modeling**

**GEETHAPRIYA MOHAN**

**(MDTM30)**

**PROJECT OBJECTIVE**

To build an efficient system for detecting fake reviews in e-commerce platforms using Natural Language Processing (NLP) techniques. Additionally, the project involves extracting meaningful topics from the reviews to gain insights into customer opinions and sentiment.

**DATA LOADING AND DATA PREPROCESSING**

Data Loading and Preprocessing are foundational steps in any machine learning or data analysis pipeline. These steps ensure that raw data is transformed into a clean, structured, and meaningful format suitable for analysis or training models.

**Data Loading**

Definition: The process of importing raw data into a programming environment (e.g., Python, other tools) from various data sources for analysis and modeling.

Data Sources:

* Files (e.g., CSV, JSON, Excel, or text files).

**Tools for Loading Data**:

* Libraries: pandas, seaborn, numpy
* Specialized tools: TensorFlow’s Dataset API for deep learning tasks, sklearn

A screenshot of a computer

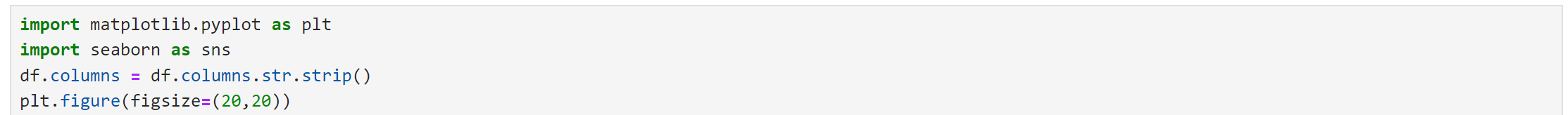
Description automatically generated

**Data balancing:**

Data balancing refers to the process of addressing the issue of class imbalance in a dataset. Class imbalance occurs when the number of instances in one class significantly outweighs the number of instances in other classes. This is common in classification problems where the target variable has an uneven distribution, such as fraud detection.

A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated

**Data Preprocessing**

Definition: The process of cleaning, transforming, and preparing raw data to improve its quality and structure, ensuring compatibility with machine learning models.

**Key Steps in Preprocessing:**

1. Handling Missing Values:
   * Imputation: Replace missing values with mean, median, mode, or interpolated values.
   * Dropping: Remove rows or columns with too many missing values.
2. Data Cleaning:
   * Removing duplicates.
   * Fixing or removing outliers.
   * Standardizing inconsistent data formats (e.g., date formats).
3. Feature Scaling:
   * Normalize or standardize numerical values to improve model performance.
   * Methods include:
     + Min-Max Scaling.
     + Z-score Standardization.
4. Encoding Categorical Variables:
   * Convert non-numerical data into numerical form using techniques like:
     + One-hot encoding.
     + Label encoding.
5. Feature Engineering:
   * Create new features that may enhance the model's understanding of the data.
   * Examples: Creating date-based features (e.g., day, month), interaction terms, or aggregations.
6. Data Splitting:
   * Split data into training, validation, and test sets to evaluate model performance.
7. Handling Class Imbalance (if applicable):
   * Oversampling minority classes (e.g., SMOTE).
   * Undersampling majority classes.

A screenshot of a chat

Description automatically generated

A screenshot of a computer

Description automatically generated

**Importance of Data Loading and Preprocessing**

* Improves Data Quality: Helps remove noise and inconsistencies.
* Ensures Model Compatibility: Data must match the input format expected by machine learning algorithms.
* Enhances Performance: Properly preprocessed data improves model accuracy, training efficiency, and generalization.
* Cleaned data set



A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**BAG OF WORDS**

Bag of Words (BoW**)** is a fundamental text representation technique used in natural language processing (NLP) and machine learning. It represents text data in a numerical format, which can be used as input to machine learning models.

In the Bag of Words model:

1. A text document is represented as a "bag" (unordered collection) of its words.
2. The model disregards grammar, word order, and sentence structure.
3. It focuses solely on the **occurrence (or frequency)** of words in the text.

**Key Features of Bag of Words**

* **Vocabulary**:
  + A unique set of words from the entire corpus (all the documents being analyzed).
* **Representation**:
  + Each document is transformed into a vector based on the vocabulary.
  + The vector's dimensions correspond to the vocabulary's size.
  + Each entry in the vector represents the frequency (or presence) of a word in the document.

**How Bag of Words Works**

**Step 1: Tokenization**

Split the text into individual words or tokens. For example:

* Input: "I like coding. I like data science."
* Tokens: ["I", "like", "coding", "I", "like", "data", "science"]

**Step 2: Build Vocabulary**

Create a list of unique words from all the documents. For example:

* Vocabulary: ["I", "like", "coding", "data", "science"]

**Step 3: Vectorization**

Represent each document as a vector of word counts (or presence/absence). For example:

* Document: "I like coding"
  + Vector: [1, 1, 1, 0, 0] (Counts of "I", "like", "coding", "data", "science")
* Document: "I like data science"
  + Vector: [1, 1, 0, 1, 1]

**Variants of Bag of Words**

1. **Binary BoW**:
   * Only considers whether a word is present or absent (1 or 0) in a document.
2. **Term Frequency (TF)**:
   * Uses the raw count of word occurrences in a document.
3. **TF-IDF (Term Frequency-Inverse Document Frequency)**:
   * Adjusts word counts by considering their importance in the entire corpus, giving less weight to common words.

A screenshot of a chat

Description automatically generated

A screenshot of a table

Description automatically generated

**Advantages of Bag of Words**

1. Simplicity: Easy to implement and understand.
2. Effective for Small Datasets: Works well for small-scale tasks.
3. Foundation for Advanced Methods: Basis for more complex text representation techniques like TF-IDF and embeddings.

**Disadvantages of Bag of Words**

1. Ignores Word Order: Loses contextual and syntactic information.
2. Sparse Representation: Large vocabulary can lead to high-dimensional, sparse vectors.
3. Lack of Semantics: Fails to capture the meaning or relationships between words.

A screenshot of a computer

Description automatically generated

**TF-IDF**

TF-IDF (Term Frequency-Inverse Document Frequency**)** is a numerical statistic used to represent how important a word is to a document in a collection or corpus. It is widely used in text mining, information retrieval, and natural language processing (NLP) to transform text data into a numerical format that machine learning models can process.

**Formula for TF-IDF**

TF-IDF(t,d)=TF(t,d)×IDF(t)

Where:

* t: A specific term (word).
* d: A specific document.
* TF(t,d)\text{TF}(t, d)TF(t,d): Term Frequency of term ttt in document ddd.
* IDF(t)\text{IDF}(t)IDF(t): Inverse Document Frequency of term ttt across all documents.

**Components of TF-IDF**

**1. Term Frequency (TF)**

Measures how often a term ttt appears in a document ddd, relative to the total number of words in the document**.**

TF(t,d)= Number of occurrences of t in d​/Total number of words in d.

**2. Inverse Document Frequency (IDF)**

Measures how unique or rare a term ttt is across a collection of documents.

IDF(t)=log(N/1+DF(t))

**Purpose of TF-IDF**

* The TF component highlights frequently occurring words in a document, while the IDF component down weights common words (e.g., "the", "and") that are less informative. The combination ensures that terms unique to a document receive a higher weight.

A screenshot of a computer

Description automatically generated

A white rectangular object with a black border

Description automatically generated

**MACHINE LEARNING**

**Random forest**

A Random Forest is a popular ensemble learning method used for classification, regression, and other tasks. It operates by constructing a collection of decision trees at training time and outputs either the mode (most common value) of the classes for classification tasks or the average prediction for regression tasks.

**Key Characteristics of Random Forest:**

1. Ensemble of Decision Trees: Random Forest combines multiple decision trees to improve performance and reduce the risk of overfitting.
2. Random Sampling:
   * Bootstrap Sampling: Each tree is trained on a randomly selected subset (with replacement) of the training data, known as a "bootstrap sample."
   * Feature Selection: For each split in a tree, only a random subset of features is considered for splitting, which helps in reducing correlation between trees.
3. Voting/Averaging:
   * In classification, the Random Forest predicts the class with the majority vote from all individual trees.
   * In regression, it averages the predictions of the individual trees.
4. Parallelizable: The trees are constructed independently, making it highly scalable and suitable for parallel computation.
5. Robust to Overfitting: By averaging the predictions of multiple trees, Random Forest reduces variance, making it less likely to overfit compared to a single decision tree.

**Advantages:**

* Handles large datasets with higher dimensionality.
* Robust to missing data and outliers.
* Reduces overfitting compared to individual trees.
* Provides feature importance rankings.

A screenshot of a computer

Description automatically generated

A graph of blue and orange bars

Description automatically generated

A close-up of a computer screen

Description automatically generated

**DEEP LEARNING**

**Long Short-Term Memory (LSTM)**:

LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN) architecture designed to model and learn from sequential data. It is widely used in tasks where context or memory of previous inputs is important, such as time-series analysis, speech recognition, and natural language processing (NLP).

**Key Characteristics of LSTM:**

1. Handles Long-Term Dependencies:
   * Unlike traditional RNNs, LSTMs are designed to retain information over long sequences, addressing the issue of vanishing or exploding gradients in RNNs.
2. Special Memory Mechanism:
   * LSTM cells have a "memory cell" that stores information, and three main gates (input, forget, and output gates) regulate the flow of information.

**Core Components of an LSTM Cell:**

1. Forget Gate:
   * Decides which information to discard from the cell state.
   * Output: A value between 0 (forget completely) and 1 (retain completely).
2. Input Gate:
   * Determines what new information to add to the cell state.
   * Involves a sigmoid function (to decide which inputs to update) and a tanh function (to regulate the scale of updates).
3. Cell State:
   * The memory that runs through the network, modified by the input and forget gates.
4. Output Gate:
   * Determines what part of the cell state to output as the hidden state for the next time step.

**How LSTMs Work:**

At each time step:

1. The current input and the hidden state from the previous step are processed.
2. The forget gate adjusts the memory by filtering unnecessary information.
3. The input gate updates the cell state with new information.
4. The output gate decides the current hidden state and sends it to the next time step.

A screenshot of a computer

Description automatically generated

A close-up of a white rectangular object

Description automatically generated

A close up of a number

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A graph with blue squares

Description automatically generated

**PRETRAINED MODELLING**

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated



A screenshot of a computer

Description automatically generated

**Comparison tables of Fake review classification**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-score** | **Accuracy** |
| **logistic Regression**  CG  OG | 0.89  0.85 | 0.84  0.90 | 0.87  0.88 | 0.87 |
| **Randomforest**  CG  OG | 0.83  0.89 | 0.90  0.82 | 0.86  0.85 | 0.86 |
| **Lstm**  CG  OG | 0.50  0.00 | 1.00  0.00 | 0.66  0.00 | 0.50 |
| **Premodeling**  CG  OG | 0.00  0.00 | 0.00  0.00 | 0.00  0.00 | 1.00 |

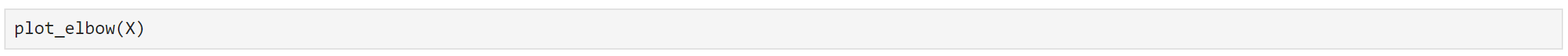
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Aspect** | **Logistic Regression** | **Random Forest** | **LSTM** (Long Short-Term Memory) | **Pretrained Modeling** (e.g., BERT, GPT) |
| Type of model | Linear Model | Ensemble Model (Decision Trees) | Recurrent Neural Network (RNN) | Transfer Learning (often based on Transformers) |
| Data type | Structured/tabular data | Structured/tabular data | Sequential/time-series data | Text, images, audio, and multimodal data |
| Input Shape | Fixed-length feature vectors | Fixed-length feature vectors | Sequential data (varying lengths allowed) | Preprocessed tokens (e.g., text embeddings, images, etc.) |
| Strengths | Simple and interpretable. | Handles non-linear data well. | Captures long-term dependencies in sequences. | Leverages pre-trained knowledge, reducing data needs. |
| Weaknesses | Cannot model non-linear relationships effectively | Computationally expensive for many trees. | Computationally expensive; requires large datasets | High computational resources and storage requirements. |
| Key Hyperparameters | Regularization (L1/L2) coefficient. | Number of trees, depth of trees, splitting criteria. | Number of layers, units, learning rate, dropout. | Pretrained model type (e.g., BERT, GPT), fine-tuning steps. |
| Interpretability | High (easy to explain coefficients). | Medium (via feature importance). | Low (difficult to interpret neuron weights). | Low (complex deep networks, though visualizations help). |
| Scalability | Highly scalable; works with large datasets. | Scales well with parallelism (trees). | Poor for very large datasets without distributed systems. | Requires distributed systems for large-scale tasks. |
| Training Time | Very fast. | Medium (slower with more trees). | Slow (long training time for large datasets). | Very slow without pretrained models; fast fine-tuning. |
| Example Libraries | scikit-learn, statsmodels | scikit-learn, XGBoost | TensorFlow, PyTorch | Transformers (Hugging Face), OpenAI |

**K MEANS CLUSTERING**

K-Means clustering is a **centroid-based unsupervised machine learning algorithm** used to group data points into a predefined number of clusters (k) based on their similarity. It aims to minimize the variance within each cluster while maximizing the variance between clusters. Each cluster is defined by its centroid, which is the mean of all the data points assigned to that cluster.

A screenshot of a computer

Description automatically generated



A graph with a line

Description automatically generated

**How K-Means Clustering Works:**

The process involves the following steps:

1. **Initialization**:
   * Choose the number of clusters (kkk) to create.
   * Initialize kkk centroids randomly (or by using methods like K-Means++ for better initial placement).
2. **Assignment Step**:
   * Assign each data point to the cluster whose centroid is closest, typically using the **Euclidean distance**.
3. **Update Step**:
   * For each cluster, calculate the mean of all the points assigned to it, and update the centroid of the cluster to this mean.
4. **Iterative Optimization**:
   * Repeat the **assignment** and **update** steps until one of the following occurs:
     + The centroids no longer change significantly.
     + A maximum number of iterations is reached.
     + The total within-cluster variance (inertia) stops decreasing significantly.
5. **Output**:
   * Final clusters and their centroids.

A white rectangular object with a black border

Description automatically generated

A close-up of a computer screen

Description automatically generated

A white rectangular object with a black border

Description automatically generated

A diagram of a number of blue squares

Description automatically generated

A white rectangular object with a white border

Description automatically generated

A screenshot of a graph

Description automatically generated

**TOPIC MODELING**

Topic Modeling is an unsupervised machine learning technique used to automatically identify hidden themes or topics in a collection of documents. It groups words in the text into clusters based on their likelihood of co-occurrence and assigns each document a mixture of these topics.

Topic modeling is widely used in natural language processing (NLP) to analyze and summarize large datasets of text.

A white rectangular object with a black border

Description automatically generated with medium confidence

A screenshot of a computer code

Description automatically generated

A white background with black text

Description automatically generated

**Key Points:**

1. **Unsupervised Learning**:
   * It does not require labeled data. Instead, it discovers patterns or structures (topics) based on the text's content.
2. **Probabilistic Models**:
   * Algorithms like Latent Dirichlet Allocation (LDA) and Non-negative Matrix Factorization (NMF) are commonly used.
   * These models assume that each document is a mixture of topics and each topic is a mixture of words.
3. **Output**:
   * The result is a set of topics, where each topic is represented by a list of words.
   * Each document gets a probability distribution over the discovered topics.

**Applications:**

* **Document Clustering**: Grouping similar documents together.
* **Text Summarization**: Extracting the main themes from large collections of text.
* **Search Optimization**: Enhancing information retrieval systems.
* **Sentiment Analysis**: Understanding themes behind opinions in reviews or feedback.

**REFERENCES:**

* **Source**:<https://drive.google.com/file/d/1-5pMPrdiLM0MubZNRhH-tGA9-jvkEPHw/view?usp=sharing>
* <https://scikit-learn.org/stable/>
* <https://huggingface.co/models?p=2&sort=trending>