MIT | Academy of Engineering

(An Autonomous Institute Affiliated to Savitribai Phule Pune University)

GOOGLE PLAY STORE SENTIMENT ANALYSIS USING NLP

TY B.TECH

MACHINE LEARNING

EXAM REPORT

SUBMITTED BY

Vruksheeka Deshmukh	**202101070058**
Swastik Gaikwad	**202101070062**
Sanika Thakare	**202101070120**

SCHOOL OF COMPUTER ENGINEERING AND TECHNOLOGY

MIT ACADEMY OF ENGINEERING, ALANDI (D), PUNE-412105

MAHARASHTRA (INDIA)

MAY, 2024

PROBLEM STATEMENT AND DATASET

PROBLEM STATEMENT

" Google Play Store Sentiment Analysis Using NLP ”

The project aims to conduct sentiment analysis on reviews gathered from the Google Play Store. By leveraging Natural Language Processing (NLP) techniques, the system will automatically classify these reviews into positive or negative sentiments. The primary objective is to provide valuable insights into customer experiences with various applications, helping app developers and businesses make informed decisions. Additionally, the project seeks to demonstrate the scalability of sentiment analysis techniques in handling large textual datasets and their applicability in real-world scenarios.

DATASET

The dataset comprises over 12,000 app store reviews, each characterized by various attributes. These attributes include essential details such as review ID, user name, and content of the review. Additionally, the dataset contains information about the review score, thumbs-up count, review creation version, and the time of the review. Furthermore, some reviews have corresponding replies, as indicated by the presence of columns like 'replyContent' and 'repliedAt'. The dataset offers a comprehensive view of user feedback on various applications available on the platform

OBJECTIVES

- Efficiently classify app store reviews into positive and negative sentiments using NLP techniques.
- Extract valuable insights into customer sentiments towards various applications.
- Ensure robust sentiment analysis with diverse classification models and tools.
- Demonstrate scalability in handling large textual datasets with over 12,000 reviews.

ABSTRACT

This sentiment analysis project leveraged Natural Language Processing (NLP) methods to categorize app store reviews into positive or negative sentiments. With a dataset comprising over 12,000 reviews, each rated on a scale from 1 to 5, the initial preprocessing phase involved removing irrelevant columns, addressing missing values, and segregating reviews based on scores into positive (4 and 5) and negative (1 to 3) sentiments. Textual data underwent rigorous preprocessing, including punctuation and stopwords removal, to ensure cleanliness.

In the feature extraction stage, CountVectorizer was employed to convert text data into numerical representations, while the 'score' column was binarized to facilitate sentiment classification. Multiple classification models, including Naive Bayes, SVM, Random Forest, and Logistic Regression, were trained and evaluated for their efficacy in sentiment classification. Additionally, NLTK's VADER Sentiment Analyzer provided valuable insights into customer sentiments, thereby assisting businesses in informed decision-making processes. This comprehensive approach underscores the project's endeavor to accurately discern and analyze user sentiments in the app store domain.

EXPLANATION OF THE NLP FUNCTION USED

The Natural Language Processing (NLP) features used in this sentiment analysis project include:

1. Text preprocessing: Removing punctuation and stopwords from the text data to clean it and reduce noise.

2. Tokenization: Splitting the text into individual words or tokens to analyze them separately.

3. Stemming: Reducing words to their root form to normalize variations (e.g., converting "running" to "run").

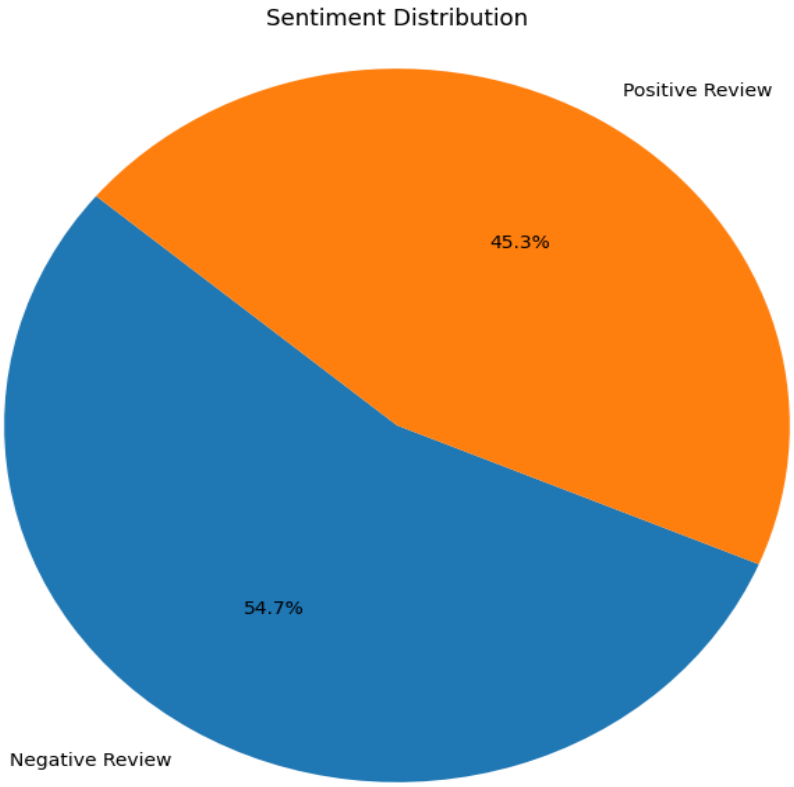
4. TF-IDF (Term Frequency-Inverse Document Frequency): Representing the text data numerically by assigning weights to words based on their frequency in the document and across the corpus.

5. CountVectorizer: Converting text data into numerical form by counting the occurrences of each word in the text.

6. TfidfVectorizer: Converting text data into numerical form using the TF-IDF weighting scheme.

7. VADER (Valence Aware Dictionary and sEntiment Reasoner) Sentiment Intensity Analyzer: Analyzing the sentiment of text data by assigning polarity scores to individual words and computing an overall sentiment score for the text.

Data Visualization



The pie chart visualizes the sentiment distribution of app store reviews after sentiment analysis. In the chart, 53.7% of the reviews are classified as negative sentiment, while 43.3% of the reviews are classified as positive sentiment. This distribution provides insights into the overall sentiment of the app store reviews, indicating that a significant portion of the reviews expresses negative sentiments, while a slightly smaller portion expresses positive sentiments.

PROBLEM STATEMENT IMPLEMENTATION

▪ DATASET SPLITTING

The process of splitting the data into training and testing sets is essential for effectively evaluating the performance of machine learning models. By partitioning the dataset into two distinct subsets, one for training and the other for testing, we ensure that the model's performance is assessed on unseen data. This helps in estimating how well the model will generalize to new, unseen instances in real-world scenarios. Setting aside a portion of the data for testing allows us to validate the model's predictions and assess its accuracy and robustness. Additionally, specifying a random seed ensures reproducibility, enabling consistent results across different runs of the code.

▪ MODELS USED

1. Logistic Regression:

- Logistic Regression is a linear classification algorithm used for binary classification tasks.

- It models the probability of a binary outcome using the logistic function.

- Despite its simplicity, Logistic Regression is effective for tasks where the relationship between features and target is linear.

2. Support Vector Machine (SVM):

- SVM is a powerful classification algorithm known for its ability to handle high-dimensional data and complex decision boundaries.

- It works by finding the hyperplane that best separates the classes in the feature space.

- SVM can handle both linear and non-linear classification tasks through the use of different kernel functions.

3. Random Forest (RF):

- Random Forest is an ensemble learning method that constructs multiple decision trees during training.

- It combines the predictions of multiple decision trees to improve accuracy and reduce overfitting.

- Random Forest is robust to noise and outliers and is suitable for handling large datasets with high dimensionality.

4. Naive Bayes (NB):

- Naive Bayes is a probabilistic classification algorithm based on Bayes' theorem and the assumption of independence between features.

- Despite its simplicity and the "naive" assumption, Naive Bayes often performs well in practice, especially for text classification tasks.

- It's efficient, requires a small amount of training data, and is particularly suitable for tasks with a large number of features.

- **GridSearchCV**

GridSearchCV (Grid Search Cross-Validation) is a technique used in machine learning for hyperparameter optimization, specifically in the context of models trained using cross-validation.

In machine learning, hyperparameters are parameters that are not learned during the training process but are set prior to training and affect the learning process. Examples of hyperparameters include the learning rate in a neural network or the regularization parameter in a support vector machine.

GridSearchCV works by exhaustively searching through a specified subset of hyperparameters for a given model. It evaluates each combination of hyperparameters using cross-validation, which involves splitting the training dataset into multiple subsets (folds), training the model on a subset of folds, and validating it on the remaining fold. This process is repeated for each combination of hyperparameters.

GridSearchCV is used for two primary reasons:

1. Hyperparameter Tuning:
   It helps in finding the optimal combination of hyperparameters for a given model and dataset. By systematically searching through a grid of hyperparameter values, it can identify the combination that results in the best performance (e.g., highest accuracy, lowest error) on a validation set.

2. Preventing Overfitting:
   Using cross-validation helps prevent overfitting by providing an estimate of how the model will perform on unseen data. By evaluating the model's performance on multiple validation sets, GridSearchCV provides a more robust estimate of how well the model will generalize to new data compared to using a single validation set.

PIPELINE IN ML :

Pipelining in machine learning, specifically using scikit-learn's Pipeline module. Pipelining allows you to chain multiple steps together, where the output of one step serves as the input to the next step.

Example:

```
pipeline1 = Pipeline([
    ('bow', CountVectorizer()),  # strings to token integer counts
    ('tfidf', TfidfTransformer()),  # integer counts to weighted TF-IDF scores
    ('svm', SVC(kernel='linear')),
])
```

1. The 'bow' step represents the CountVectorizer, which converts strings (text) into token integer counts.

2. The 'tfidf' step represents the TfidfTransformer, which converts the integer counts obtained from CountVectorizer into weighted TF-IDF scores (Term Frequency-Inverse Document Frequency).

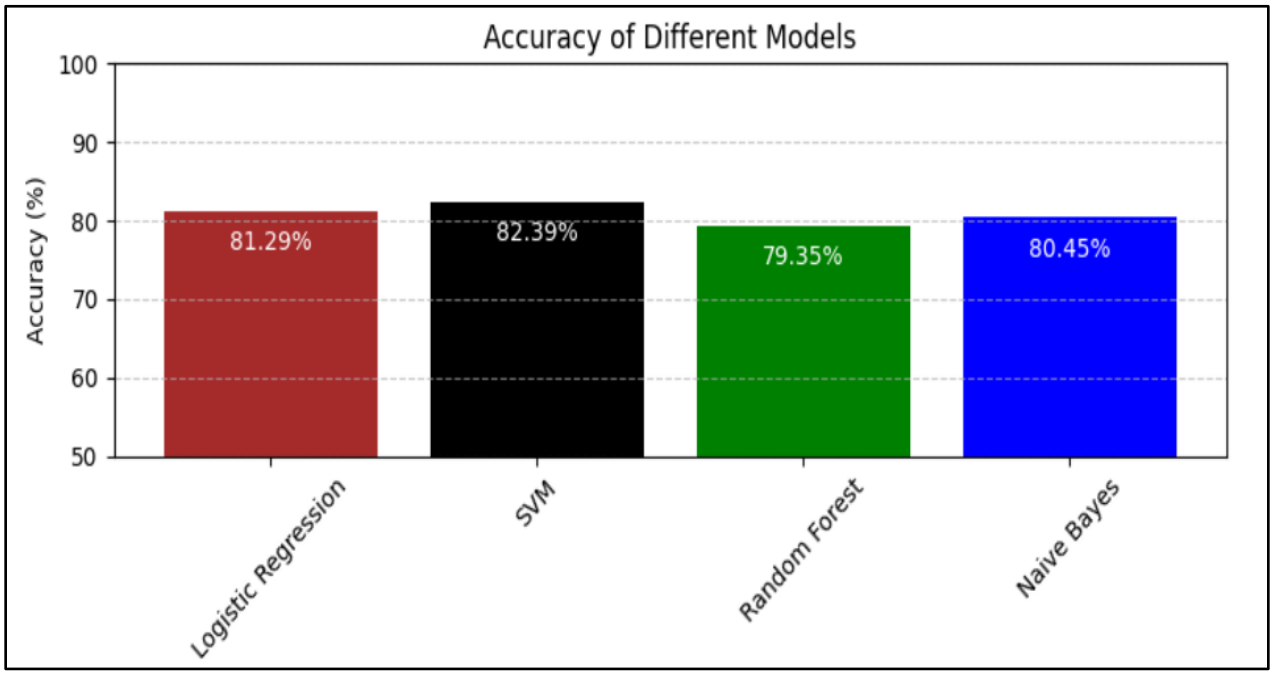
3. The 'svm' step represents the Support Vector Machine (SVM) classifier, specifically using a linear kernel for classification.

By encapsulating these preprocessing steps and the classifier into a single pipeline, it becomes easier to manage and apply the entire workflow consistently. This pipeline can then be used to fit the model on training data and make predictions on new data seamlessly.

RESULTS

1. ACCURACY

The table presents the accuracy scores of various classification models, with the highest accuracy achieved by SVM after hyperparameter tuning using GridSearchCV. This indicates that SVM, when optimized through GridSearchCV, outperformed other models in accurately classifying the data into the correct categories.



2. TIME COMPLEXITIES

The table displays the time complexities of training and testing for various classification models. Training time refers to the duration taken to train the model on the training data, while testing time indicates the time required to make predictions on the testing data.

Model	Training Time (s)	Testing Time (s)
Logistic Regression	13.433	0.144
Support Vector Machine	162.436	2.622
Random Forest	10.087	0.221
Naive Bayes	0.2	0.074

The two models with the minimum time complexity are Naive Bayes and Logistic Regression. Naive Bayes achieves this due to its simple probabilistic model, which requires minimal computation during both training and prediction. Similarly, Logistic Regression's linear nature and convex optimization lead to efficient training and inference times, making it suitable for large-scale datasets and real-time applications.

CONCLUSION

The sentiment analysis project effectively employed NLP techniques to categorize app store reviews as positive or negative, focusing on scores 4 and 5 for positive sentiments and scores 1 to 3 for negative ones. Among the models employed, the SVM model showcased the highest accuracy of 82.38%, owing to its capability to discern intricate data relationships with a linear kernel. Furthermore, Naive Bayes and Logistic Regression exhibited minimal time complexity, rendering them suitable for real-time sentiment analysis applications. Overall, the project showcased scalability in handling substantial textual datasets and offered valuable insights into customer sentiments across various applications on the Google Play Store.