Infosys Springboard Internship Program

Project Documentation title

Text Classification on Emotion Dataset

By Triveni Sunkara

Mentor - Sudheer Kumar

Table of Contents

Problem Statement and Business Use Case

Problem Statement

- **Challenge:** Manually Analyzing large Volumes of text data to understand customer emotions is impractical and time-consuming
- **Need:** Develop an automated system for accurate emotion detection from textual data.

Business Use Case

Customer Support and Feedback Analysis

- **Scenario:** A large e-commerce company receives thousands of customer feedback messages and support tickets daily.
- **Solution:** Implement an automated emotion detection system.
- **Impact:**
  - **Improved Customer Service:** Prioritize and address critical issues based on negative emotions.
  - **Enhanced Customer Experience:** Recognize and reinforce successful aspects of the service from positive feedback.
  - **Data-Driven Decision Making:** Use aggregated emotion data for strategic planning.

Dataset and Data Collection

Dataset Overview

- **Source:** Received via email.
- **Content:** Text data labelled with six emotions – sadness, joy, love, anger, fear, surprise.
- **Size:**16,000 records

Data Visualization

- Tools Used: WordCloud and Matplotlib.
- Purpose: To understand word frequency distribution across different emotions.

Data Preprocessing

Data preprocessing is a crucial step in the data analysis and machine learning pipeline as it can significantly impact the performance and reliability of downstream analyses and models. Properly preprocessing data can help improve model accuracy, reduce overfitting, and ensure that the resulting insights are accurate and reliable. Here are the key steps involved in text data preprocessing:

1. **Lower Case**
    - Converting all text to lowercase ensures uniformity and consistency. It prevents the model from treating words like "Happy" and "happy" as different entities.
2. **Remove Links**
    - URLs and links within the text are often irrelevant to the analysis and can introduce noise. Removing these ensures that the data focuses on meaningful content.
3. **Remove Next Lines**
    - Removing newline characters (\n) helps in maintaining the continuity of the text. This step is crucial when texts are merged or processed line by line.
4. **Words Containing Numbers**
    - Words that contain numbers (e.g., "win10") are often not useful in the context of natural language processing. Removing such words can reduce noise and improve model clarity.
5. **Extra Spaces**
    - Multiple spaces between words can cause inconsistency in text representation. Removing extra spaces ensures a clean and consistent format.
6. **Special Characters**
    - Special characters (e.g., @, #, $, %, etc.) are usually not necessary for understanding the text’s emotional content. Removing them helps in focusing on the actual words.
7. **Removal of Stop Words**
    - Common words that do not carry significant meaning (e.g., "is," "and," "the") are removed. This step helps in reducing the dimensionality of the data and focusing on important words that contribute to the text's meaning.

8. Stemming

- Stemming reduces words to their root form (e.g., "running" to "run") by removing suffixes. It helps in standardizing words to their base form, though it may sometimes produce non-dictionary words.

9. Lemmatization

- Lemmatization is similar to stemming but ensures that the reduced form is a valid word (e.g., "running" to "run"). It considers the context and part of speech, making it more accurate than stemming.

Importance of Data Preprocessing

Proper data preprocessing is essential for several reasons:

- **Improves Model Accuracy:** By removing noise and irrelevant information, preprocessing helps in enhancing the model's performance.
- **Reduces Overfitting:** Clean and consistent data helps in building models that generalize well to new data.
- **Ensures Reliable Insights:** Accurate preprocessing ensures that the insights drawn from the data are valid and reliable.

In summary, data preprocessing is a foundational step that can significantly influence the success of data analysis and machine learning projects. It ensures that the data fed into models is clean, consistent, and meaningful, leading to better performance and more accurate results.

Modeling Approach

Data Division

- **Training, Testing and Validation Sets:**

Split the dataset into training, testing and Validation sets for both balanced and imbalanced datasets.

Handling Imbalanced Data

1. Custom Oversampling

Custom oversampling is a technique used to address class imbalances in a dataset. Class imbalance occurs when some classes have significantly fewer samples than others,

which can lead to biased and inaccurate models. Custom oversampling aims to balance the dataset by increasing the number of samples in the minority classes. Here are the key aspects of custom oversampling:

Random Selection and Duplication:

- Custom oversampling involves randomly selecting samples from the minority class and duplicating them to increase their representation in the dataset. This can be done repeatedly until the minority class reaches a desired level of representation.
- For example, if you have a dataset with 1,000 samples in the majority class and only 100 in the minority class, you might duplicate the minority class samples until you have 1,000 samples in each class.

Advantages:

- **Improves Model Performance:** By balancing the classes, custom oversampling helps the model learn equally from all classes, improving its ability to generalize and make accurate predictions on unseen data.
- **Easy to Implement:** This technique is straightforward and does not require complex algorithms or computations.

Disadvantages:

- **Overfitting Risk:** By duplicating the same samples, there is a risk that the model may overfit to the minority class, learning specific patterns that do not generalize well to new data.
- **Increased Dataset Size:** Duplicating samples increases the size of the dataset, which can lead to higher computational costs and longer training times.

SMOTE (Synthetic Minority Oversampling Technique):

SMOTE, or Synthetic Minority Oversampling Technique, is a widely-used method to address class imbalance in datasets, especially in the context of classification problems. Unlike traditional oversampling methods that duplicate existing samples, SMOTE generates new, synthetic samples by interpolating between existing minority class samples. This helps in creating a more balanced dataset without simply repeating the same data points.

Advantages:

- **Avoids Overfitting:** By generating new, synthetic samples rather than duplicating existing ones, SMOTE reduces the risk of overfitting that is often associated with traditional oversampling methods.
- **Improved Model Performance:** Balancing the dataset helps machine learning models learn better from all classes, leading to improved accuracy and generalization.

Disadvantages:

- **Risk of Noise Introduction:** If the minority class contains noisy data, SMOTE may generate synthetic samples that amplify this noise, potentially degrading model performance.
- **Computational Complexity:** Finding nearest neighbors and generating synthetic samples can be computationally intensive, especially for large datasets.

Text Representation

TF-IDF (Term Frequency-Inverse Document Frequency):

TF-IDF, or Term Frequency-Inverse Document Frequency, is a widely-used statistical measure in natural language processing and information retrieval. It evaluates the importance of a word in a document relative to a collection of documents (corpus). This technique is essential for text representation and helps in transforming text data into numerical form for machine learning algorithms.

Models Used and Hyperparameter Tuning

Models Used

- SVM (Support Vector Machine)
- Random Forest
- Naïve Bayes
- XG Boost

SVM

Support Vector Machine (SVM) is a powerful and versatile supervised machine learning algorithm used primarily for classification tasks, but it can also be adapted for regression.

SVMs are renowned for their effectiveness in high-dimensional spaces and their ability to handle both linear and non-linear classification problems.

Random Forest

Random Forest is a popular ensemble learning method used for both classification and regression tasks. It operates by constructing multiple decision trees during training and outputs the mode (classification) or average prediction (regression) of the individual trees. Random Forests offer several advantages, making them widely used in various machine learning applications.

Naïve Bayes

Naive Bayes is a simple yet effective probabilistic classifier based on Bayes' theorem with the "naive" assumption of independence between features. Despite its simplicity, Naive Bayes classifiers are widely used in text classification, spam filtering, and other applications where the assumption of feature independence holds reasonably well.

XG Boost

XGBoost, short for Extreme Gradient Boosting, is a powerful and efficient implementation of the gradient boosting algorithm. It is known for its high performance, scalability, and accuracy in a wide range of machine learning tasks, particularly in structured/tabular data problems such as regression, classification, and ranking.

Hyperparameter Tuning

Hyperparameter tuning is the process of finding the optimal set of hyperparameters for a machine learning model to achieve the best performance on a given dataset. Hyperparameters are configuration settings that are external to the model and cannot be directly estimated from the data. They control aspects of the learning process, such as the model's complexity, regularization, and optimization strategy. Unlike model parameters, which are learned during training, hyperparameters must be set before training begins and are typically selected based on trial and error, heuristics, or optimization algorithms.

GridSearchCV: Grid search is a hyperparameter tuning technique used to find the optimal set of hyperparameters for a machine learning model by exhaustively searching through a predefined grid of hyperparameter values. It is a systematic and straightforward approach that

evaluates all possible combinations of hyperparameters based on a specified range or list of values. Grid search is commonly used in conjunction with cross-validation to identify the hyperparameter configuration that maximizes the model's performance metric on a validation set.

Modeling Results

Performance Metrics

Accuracy, precision, recall, and F1-score are commonly used metrics to evaluate the performance of classification models. Each metric provides valuable insights into different aspects of a model's predictive capabilities and is calculated based on the predictions made by the model compared to the ground truth labels.

Accuracy: Accuracy measures the proportion of correctly classified instances among all instances in the dataset.

Precision: Precision measures the proportion of true positive predictions among all positive predictions made by the model.

Recall: Recall measures the proportion of true positive predictions among all actual positive instances in the dataset.

F1-score: F1-score is the harmonic mean of precision and recall. It provides a balance between precision and recall, especially in imbalanced datasets.

Confusion Matrix

A confusion matrix is a tabular representation of the performance of a classification model that summarizes predictions made by the model compared to the actual ground truth labels. It provides valuable insights into the model's behavior and helps evaluate its performance across different classes.

Structure of a Confusion Matrix: A confusion matrix typically consists of four cells, representing different combinations of predicted and actual class labels: -

True Positives (TP): Instances that were correctly classified as positive by the model.

False Positives (FP): Instances that were incorrectly classified as positive by the model (predicted positive, but actually negative).

True Negatives (TN): Instances that were correctly classified as negative by the model.

False Negatives (FN): Instances that were incorrectly classified as negative by the model (predicted negative, but actually positive).

Visual Analysis: Evaluated the performance of classifiers using confusion matrices to visualize accuracy and error rates across different emotion classes.

Here is the table summarizing the training and testing accuracy for each of the models used in Text Classification on Emotion Dataset project:

Sl.no	Models	Training Accuracy	Testing Accuracy
1	SVM	0.92	0.90
2	Random Forest	0.80	0.77
3	Naïve Bayes	0.77	0.76
4	XG Boost	0.95	0.91

Analysis:

- **XG Boost:** appears to be the best-performing model in terms of both training and testing accuracy.
- **SVM** also performed well, though slightly lower than XG Boost on the testing set.
- **Random Forest and Naïve Bayes** showed lower performance compared to SVM and XG Boost, with Random Forest performing better than Naive Bayes.

Recommendations and Resolutions

Recommendations

- **XG Boost** should be considered the preferred model for deployment given its superior performance.
- **SVM** is also a viable option if further tuning and optimization are desired.
- Additional techniques such as hyperparameter tuning and ensemble methods could be explored to potentially improve the performance of Random Forest and Naïve Bayes.

General Resolutions for All Models:

- **Cross-Validation:** Use k-fold cross-validation for a more robust estimation of model performance and to reduce the risk of overfitting.
- **Ensemble Methods:** Consider combining the models using ensemble techniques like stacking or voting to leverage the strengths of each model.
- **Regular Monitoring:** Continuously monitor the model performance post-deployment to ensure it remains effective with new data and retrain as necessary.
- **Interpretability:** Ensure that the models, especially complex ones like XGBoost, are interpretable. Use tools like SHAP (SHapley Additive exPlanations) for feature importance and impact analysis.

Thank You