Documentation Report on

"Text Classification and Sentimental Analysis on Social Media Content"

Computer Science and Engineering

Walchand Institute of Technology

(An Autonomous Institute)

By Lohade Sanskruti

Under Guidance of

Mr. Sudheer Kumar

CERTIFICATE

This is to certify that the Project entitled

“Text Classification and Sentimental Analysis on Social Media Content”

Is

Submitted By

Lohade Sanskruti

(Mr. Sudheer Kumar Y)
Project Guide

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

WALCHAND INSTITUTE OF TECHNOLOGY

SOLAPUR

(2023-2024)

INDEX

Abstract

During my internship at Infosys Springboard, I worked on two key projects: Sentiment Analysis using a lexicon-based approach and Text Classification using a supervised dataset. Under the guidance of Mr. Sudheer Kumar, I engaged in daily sessions over Microsoft Teams for eight weeks, where I received extensive training and mentorship.

For the Sentiment Analysis project, I utilized the AFINN and VADER lexicon-based methods to evaluate the sentiment of textual reviews. The process involved comprehensive text preprocessing, including converting text to lowercase, removing URLs and hyperlinks, stemming, lemmatization, and eliminating stopwords. Through Exploratory Data Analysis (EDA), I gained insights into the dataset, which facilitated effective sentiment calculation.

In the Text Classification project, I employed a supervised Emotions.csv dataset. Following similar preprocessing steps, I vectorized the text data and assessed class imbalance within the dataset. By applying various machine learning models, I evaluated performance metrics to determine the most effective model for accurate text classification.

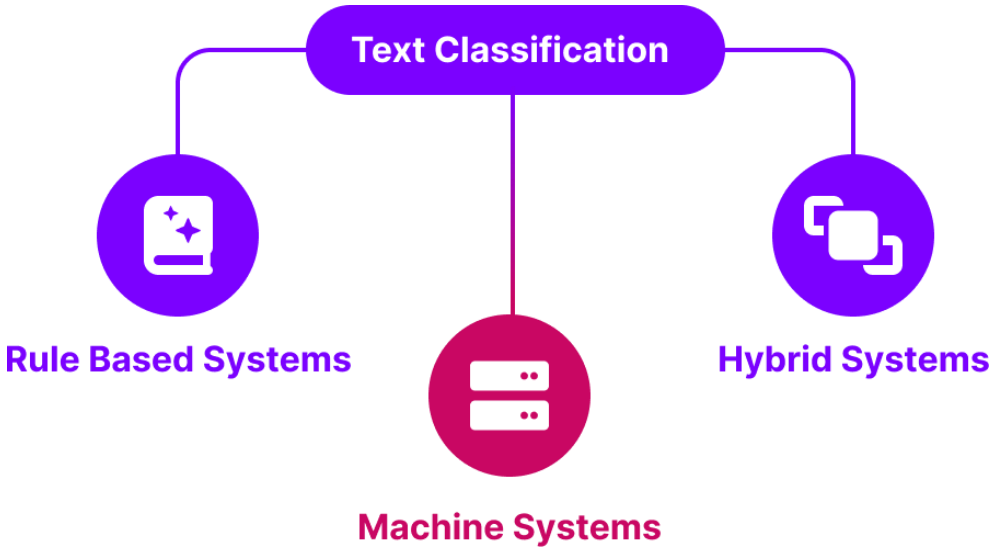
Both projects showcased my ability to preprocess and analyze text data, apply lexicon-based and machine learning techniques, and interpret results to draw meaningful conclusions. This experience significantly enhanced my skills in natural language processing and machine learning, preparing me for future challenges in these domains.

Introduction

Text classification, also known as text categorization, is a fundamental task in natural language processing (NLP) that involves assigning predefined categories or labels to textual data. Classification is a two-step process, learning step and prediction step, in machine learning, in the learning step, the model is developed based on given training data and in the prediction step, the model is used to predict the response for given data.

There are many approaches to automatic text classification, but they all fall under three types of systems:

- Rule-based systems
- Machine learning-based systems
- Hybrid systems



Multi Class Imbalance

Multi-class imbalance is a condition in a dataset where the number of instances across different classes is not evenly distributed. This means that some classes (categories) have significantly more instances than others.
Handling class imbalance is crucial for building robust and accurate machine learning models, especially when the minority class is of particular interest. There are several strategies to address class imbalance:

1) Data-Level Techniques:

1. Resampling Techniques:

(a) Oversampling: Increase the number of instances in the minority class.

- Random Oversampling: Randomly duplicate examples from the minority class.
- SMOTE (Synthetic Minority Over-sampling Technique): Generate synthetic examples for the minority class by interpolating between existing examples.
- ADASYN (Adaptive Synthetic Sampling): A variation of SMOTE that focuses more on difficult-to-learn examples.

(b) Undersampling: Decrease the number of instances in the majority class.

- Random Undersampling: Randomly remove examples from the majority class.
- Tomek Links: Remove examples that are very close to each other but belong to different classes.
- Cluster Centroids: Replace clusters of majority class examples with their centroids.

2) Algorithm-Level Techniques:

- Class Weights: Assign higher weights to the minority class during training. Many algorithms support this, such as logistic regression, decision trees, and neural networks.

3) Ensemble Methods:

- Balanced Random Forest: An ensemble method that balances each bootstrap sample.
- EasyEnsemble and BalanceCascade: Ensemble methods that focus on creating balanced datasets for each classifier.

4) Evaluation Metrics:

- Precision, Recall, F1-Score: Useful for evaluating models on imbalanced datasets.
- ROC-AUC (Receiver Operating Characteristic - Area Under the Curve): Good for binary classification.
- PR-AUC (Precision-Recall AUC): More informative for imbalanced datasets as it focuses on the performance of the positive (minority) class.
- Confusion Matrix: Provides a detailed breakdown of model performance.

Approach to Text Classification

After preprocessing your text data for text classification, the next steps involve several key processes to build and evaluate a text classification model. Here's a step-by-step guide:

Steps for Text Classification After Data Preprocessing:

1. Convert Text Data to Numerical Format:

   Text data needs to be converted into numerical format because machine learning models require numerical input.

   Common techniques include:

   - Bag of Words (BoW): Converts text into fixed-length vectors based on word counts.
   - Term Frequency-Inverse Document Frequency (TF-IDF): Converts text into vectors based on word importance.

2. Split the Data:

   - Train-Test Split: Divide your dataset into training and testing sets. A common split is 80% for training and 20% for testing.
   - Validation Set: Sometimes, you might also split off a validation set from the training data to fine-tune model parameters.

3. Choose a Model:

   Traditional Machine Learning Models:

   - Logistic Regression
   - Support Vector Machines (SVM)
   - Naive Bayes
   - Decision Trees and Random Forests

4. Train the Model:

    - Fit the model to your training data.
    - This step involves feeding the numerical representations of your text data into the chosen algorithm and allowing it to learn patterns.

5. Evaluate the Model:

    - Accuracy, Precision, Recall, F1 Score: Use these metrics to evaluate model performance.
    - Confusion Matrix: Helps in understanding the types of errors your model is making.

The right approach to text classification is methodical and iterative, involving careful data preparation, informed model selection, and continuous performance evaluation. By meticulously following these steps, one can develop a highly accurate and efficient text classification system. This process not only ensures the model's relevance to the specific problem at hand but also its adaptability to evolving data and requirements, thereby securing its long-term utility and effectiveness.

Traditional Machine Learning Models

Naïve Bayes Classifier:

Naive Bayes classifiers are probabilistic models based on **Bayes' theorem**, assuming independence between features. The Naive Bayes family of statistical algorithms are some of the most used algorithms in text classification and text analysis. One of the members of that family is Multinomial Naive Bayes (MNB) with a huge advantage. Naive Bayes is based on Bayes' Theorem, which helps us compute the conditional probabilities of the occurrence of two events, based on the probabilities of the occurrence of each individual event. So, we're calculating the probability of each tag for a given text, and then outputting the tag with the highest probability.

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

The probability of A, if B is true, is equal to the probability of B, if A is true, times the probability of A being true, divided by the probability of B being true.

This means that any vector that represents a text will have to contain information about the probabilities of the appearance of certain words within the texts of a given category, so that the algorithm can compute the likelihood of that text belonging to the category.

Support Vector Machines (SVM):

SVMs find the hyperplane that best separates data points of different classes with the maximum margin. They can be used for linear and non-linear classification through kernel tricks.

Logistic Regression:

It models the probability that a given input belongs to a particular class using a logistic function (sigmoid function).

Random Forests:

Random forests are an ensemble method that combines multiple decision trees to improve performance and reduce overfitting. Each tree is trained on a random subset of the data and features.

Gradient Boosting Classifier (XGBOOST):

GBMs build an ensemble of decision trees sequentially, where each tree tries to correct the errors of the previous trees.

XGBoost is an optimized implementation of gradient boosting that includes regularization, parallel processing, and other enhancements to improve speed and performance. XGBoost is currently the dominant algorithm for building accurate models on conventional data (also called tabular or structured data).

Performance Metrics

In a data science text classification project, several performance metrics can be used to evaluate the effectiveness of your model. These metrics help you understand how well your model is performing and where it might need improvement. Here's an overview of common performance metrics for text classification:

1. Accuracy

Accuracy measures the proportion of correctly classified instances out of the total instances. It's a straightforward metric but can be misleading if the classes are imbalanced.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

2. Precision, Recall, and F1 Score

These metrics are particularly useful for imbalanced datasets.

Precision: The proportion of true positive predictions out of all positive predictions. It indicates the accuracy of the positive predictions.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Recall: The proportion of true positive predictions out of all actual positives. It indicates the model's ability to identify all relevant instances.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

F1 Score: The harmonic mean of precision and recall. It balances the two metrics and is useful when you need a single performance score.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

3. Confusion Matrix

A confusion matrix provides a detailed breakdown of predictions into true positives, true negatives, false positives, and false negatives. It helps visualize the performance of the classification model.

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

4. ROC Curve and AUC

- ROC Curve: A graphical representation of the true positive rate (recall) vs. the false positive rate at various threshold settings.
- AUC (Area Under the Curve): A single scalar value representing the area under the ROC curve. It ranges from 0 to 1, with higher values indicating better performance.

Choosing the right metric depends on the specifics of your text classification task, including the class distribution and the business objectives. It's often helpful to look at multiple metrics to get a comprehensive understanding of model performance.

Implementation

Naïve Bayes Classifier:

Classification report for Testing data:

	precision	recall	f1-score	support
0	0.66	0.91	0.77	1365
1	0.60	0.96	0.74	1610
2	1.00	0.06	0.11	399
3	0.96	0.26	0.41	654
4	0.91	0.21	0.34	587
5	0.00	0.00	0.00	185
accuracy			0.65	4800
macro avg	0.69	0.40	0.39	4800
weighted avg	0.71	0.65	0.57	4800

Support Vector Machines (SVM):

Classification report for Testing data:

	precision	recall	f1-score	support
0	0.88	0.90	0.89	1365
1	0.83	0.91	0.87	1610
2	0.79	0.57	0.66	399
3	0.86	0.81	0.83	654
4	0.83	0.82	0.82	587
5	0.77	0.61	0.68	185
accuracy			0.84	4800
macro avg	0.83	0.77	0.79	4800
weighted avg	0.84	0.84	0.84	4800

Logistic Regression:

Classification report for Testing data:

	precision	recall	f1-score	support
0	0.83	0.92	0.87	1365
1	0.75	0.94	0.84	1610
2	0.84	0.41	0.55	399
3	0.89	0.71	0.79	654
4	0.87	0.68	0.77	587
5	0.85	0.38	0.52	185
accuracy			0.81	4800
macro avg	0.84	0.68	0.72	4800
weighted avg	0.82	0.81	0.80	4800

Random Forest Classifier:

Classification report for Testing data:

	precision	recall	f1-score	support
0	0.90	0.86	0.88	1365
1	0.85	0.90	0.87	1610
2	0.76	0.65	0.70	399
3	0.84	0.87	0.85	654
4	0.82	0.83	0.83	587
5	0.75	0.72	0.73	185
accuracy			0.85	4800
macro avg	0.82	0.80	0.81	4800
weighted avg	0.85	0.85	0.85	4800

XGBOOST Algorithm:

Classification report for Testing data:

	precision	recall	f1-score	support
0	0.93	0.85	0.89	1365
1	0.82	0.90	0.86	1610
2	0.76	0.71	0.73	399
3	0.85	0.83	0.84	654
4	0.84	0.82	0.83	587
5	0.72	0.76	0.74	185
accuracy			0.85	4800
macro avg	0.82	0.81	0.82	4800
weighted avg	0.85	0.85	0.85	4800

In [83]:

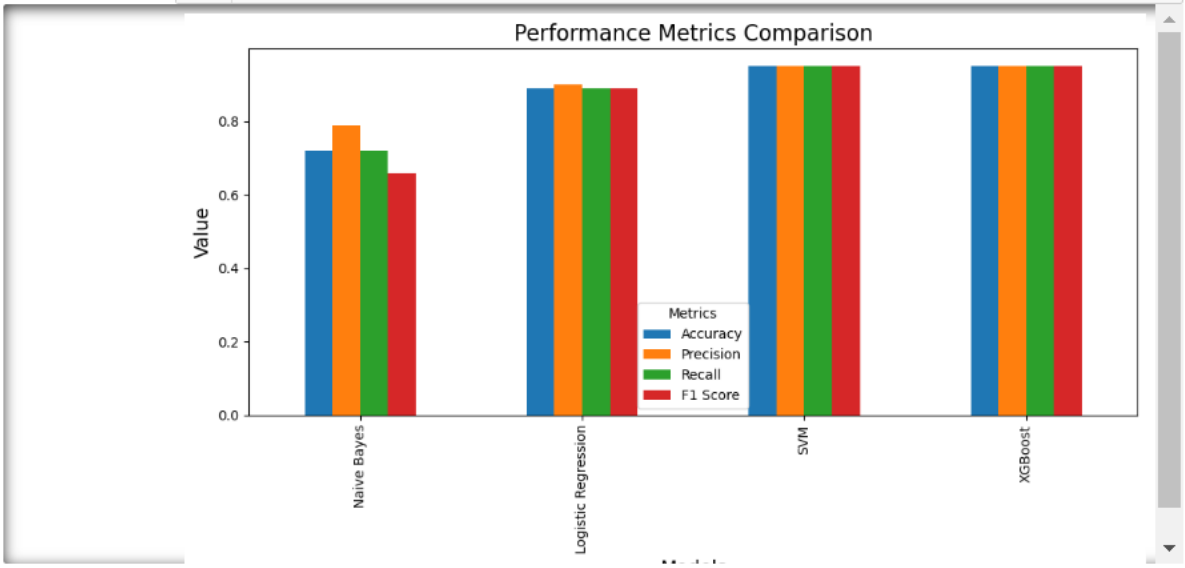
```
results = {}
models = ['Naive Bayes','Logistic Regression','SVM','XGBoost']
accuracy = [0.72, 0.89, 0.95, 0.95]
precision = [0.79, 0.90, 0.95, 0.95]
recall = [0.72, 0.89, 0.95,0.95]
f1_score = [0.66, 0.89, 0.95,0.95]

for model, acc, prec, rec, f1 in zip(models, accuracy, precision, recall,
    results[model] = {
        'Accuracy': acc,
        'Precision': prec,
        'Recall': rec,
        'F1 Score': f1
    }
results_df = pd.DataFrame(results).T
results_df
```

Out[83]:

	Accuracy	Precision	Recall	F1 Score
Naive Bayes	0.72	0.79	0.72	0.66
Logistic Regression	0.89	0.90	0.89	0.89
SVM	0.95	0.95	0.95	0.95
XGBoost	0.95	0.95	0.95	0.95

In [89]:

```
results_df.plot(kind='bar', figsize=(10, 6))
plt.title('Performance Metrics Comparison', fontsize = 16)
plt.ylabel('Value', fontsize = 14)
plt.xlabel('Models', fontsize = 14)
plt.legend(title='Metrics')
plt.tight_layout()
plt.show()
```



Sentimental Analysis

Lexicon-based sentiment analysis is a technique used in natural language processing to detect the sentiment of a piece of text. Lexicons like AFINN, SentiWordNet, or VADER are commonly used for sentiment analysis.

1. VADER (Valence Aware Dictionary and Sentiment Reasoner) - Bag of words approach

This approach takes all the words in our sentence and it has a value of either positive, negative or neutral for each of those words and it combines up and tells how the sentiment of the statement is . This approach does not account for relationships between words which in human speech is very important but at least is a good start

In [143]:
```
def categorize_sentiment(compound):
    if compound > 0.05:
        return 'Positive'
    elif compound < -0.05:
        return 'Negative'
    else:
        return 'Neutral'
merge['Sentiment'] = merge['compound'].apply(categorize_sentiment)
merge.head()
```

Out[143]:

ntent	repliedAt	appVersion	sortOrder	appId	neg	neu	pos	compound	Sentiment
m will ppy to it for you...	23-07-2020 16:32	4.16.6.2	most_relevant	com.anydo	0.250	0.695	0.055	-0.8555	Negative
re not of any s with ized...	10-12-2020 09:38	NaN	most_relevant	com.anydo	0.052	0.887	0.060	0.1027	Positive
hear hat! It ds like ht h...	11-07-2021 11:16	5.11.1.2	most_relevant	com.anydo	0.145	0.813	0.042	-0.6369	Negative

2. AFINN : (Affective Norms for English Words)

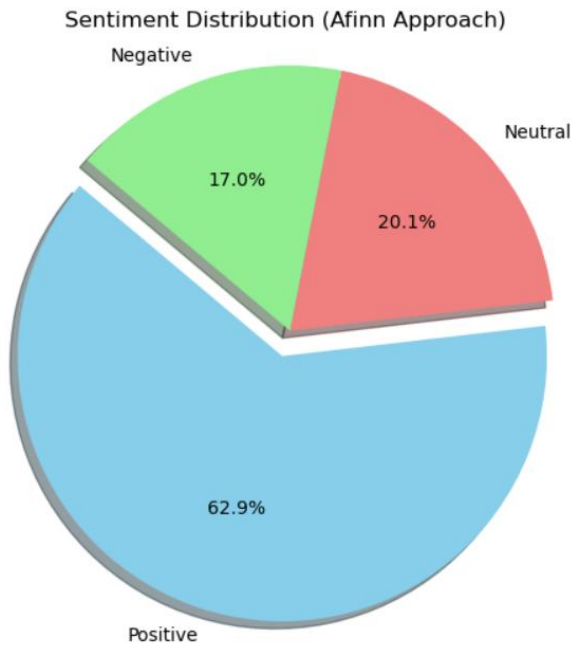
AFINN was manually created by collecting words from various sources and assigning sentiment scores to them based on their emotional impact. The sentiment scores range from -5 (most negative) to +5 (most positive), with 0 indicating neutrality. It has been widely used in sentiment analysis tasks due to its simplicity and effectiveness.

In [189]:

```
from afinn import Afinn
afinn = Afinn()
def calculate_sentiment_score(text):
    return afinn.score(text)
def find_sentiment(score):
    if score > 0:
        return 'Positive'
    elif score < 0:
        return 'Negative'
    else:
        return 'Neutral'
df['Sentiment_Score'] = df['content'].apply(calculate_sentiment_score)
df['Sentiment'] = df['Sentiment_Score'].apply(find_sentiment)
afinn_df = df[['content','Sentiment_Score','Sentiment']]
afinn_df.tail()
```

Out[189]:

	content	Sentiment_Score	Sentiment
16782	excel app	0.0	Neutral
16783	love easi use make life organ love way put pho...	12.0	Positive
16784	love could make plan check app almost everyday...	6.0	Positive
16785	exactli need	0.0	Neutral
16786	good	3.0	Positive



Applications

1) Spam Detection: Automatically classifying emails or messages as spam or not spam.

2) Sentiment Analysis: Determining the sentiment expressed in text, commonly used for analysing customer reviews and social media sentiment.

3) Topic Classification: Categorizing news articles or documents into topics like sports, politics, technology, etc.

4) Language Detection: Identifying the language in which a piece of text is written.

5) Intent Recognition: Understanding the intent behind user queries in applications like chatbots and virtual assistants.

Conclusion/Summary

In conclusion, the tasks we worked under Infosys Springboard Internship effectively applied lexicon-based models for sentiment analysis and performed Text classification. For text classification, the study addressed the challenges of multi-class imbalanced data within a supervised dataset. Various models were implemented and evaluated using performance metrics, with classification reports generated for each model. Among all the models tested, Support Vector Machine (SVM) and XGBoost demonstrated superior performance.

References

- https://monkeylearn.com/text-classification/#:~:text=Tutorial-,What%20is%20Text%20Classification%3F,and%20all%20over%20the%20web
- https://www.geeksforgeeks.org/what-is-the-right-approach-for-text-classification-problems/
- https://youtu.be/E3MAVkitm28?si=v5jeRDWKGnCR8dmK
- https://youtu.be/r7qS9mKVLdw?si=oqpIg-i_6UQ5wYQx
- https://youtu.be/ZKkEQSpRgCs?si=yIVmK1WgeRCxgZpF