Text Mining and Sentiment Analysis Final Project

Sarcasm Prediction on Reddit

"By Applying Supervised Text Classification Algorithms"

Ashkan Samavatian (Matriculation number: 965235)

1) *Introduction*

1-1) Project Overview

The current project is focused on analyzing the "Sarcasm on Reddit" dataset, which is available on Kaggle, to predict the probability of a parent comment on Reddit receiving a sarcastic response based on the parent comment's text and the corresponding subreddit. Considering the "parent_comment" and "subreddit" columns as the features and the "label" column as the target in a random subset of the "train-balanced-sarcasm.csv" file and implementing the codes on the "Google Colab" service, the methodology contains applying four different supervised text classification algorithms with Cross-Validation (respectively The Logistic Regression, K-Nearest Neighbors, Decision Tree, and Random Forest algorithms) and evaluating their performance using confusion matrix, classification report and ROC curve and comparing their results.

In the end, another approach containing the implementation of stratified k-fold Cross-Validation on all mentioned supervised machine learning algorithms was applied to find the best model performance on the subset, and then the corresponding model which was the Logistic Regression was used on the whole "train-balanced-sarcasm.csv" dataset to predict the probability of getting sarcastic comments by the parent comments.

1-2) Literature (Background) Discussion

Reddit is a social media site in which users communicate by commenting on submissions, which are titled posts consisting of embedded media, external links, and/or text, that are posted on topic-specific forums known as subreddits. Users comment on submissions and on other comments, resulting in a tree-like conversation structure such that each comment has a parent comment. Thus, understanding the sentiments or tone, such as sarcasm, behind the interactions in this social network is crucial.

Sarcasm is a form of sentiment whereby people express implicit information, usually the opposite of the message content to hurt someone emotionally or criticize something humorously. Sarcasm identification in textual data is one of the hardest challenges in natural language processing (NLP) since it occurs infrequently and is difficult for even humans to discern.

2) Research Question and Methodology

2-1) Research Question and the Goal of the Project

The main question in this project which should be addressed is giving only the Parent Comments and the Reddit Categories (Subreddits) to a model to predict the probability of a Parent Comment receiving a sarcastic comment.

2-2) Methodology

Since the "Sarcasm on Reddit" dataset is a labeled dataset, I decided to apply different supervised text classification algorithms to address the research question. I have performed four algorithms with the Cross-Validation technique (respectively The Logistic Regression, K-Nearest Neighbors, Decision Tree, and Random Forest algorithms) and then I have evaluated their performance using the Confusion Matrix, Classification Report, and ROC Curve.

Since obtaining high performance of the models had not been emphasized in the scope of this project, I did not go into the tuning concept and its methods like "GridSearchCV" or "RandomizedSearchCV". However, I tried to compare the performance of the models in two different paths:

1) **First Path:** Performing all the processes of model training with cross-validation, prediction, and evaluation for all four algorithms on a random subset to have an insight into the potential effectiveness of each model and then comparing the results to select the best one among them.

2) **Second Path:** Instantiating and training the models on a random subset and applying a cross-validation process to find the best model before performing the prediction step and then choosing the best model to do the training and prediction procedures on the whole dataset.

To perform this project, I employed the following steps:

2-2-1) Data Acquisition, Handling, and Exploration

I have used my "Kaggle API" ("Kaggle username" and "Kaggle key") to download the dataset directly from Kaggle to the Google Colab service. The dataset's identifier is "danofer/sarcasm", and it was downloaded in a compressed format. Then I unzipped only the "train-balanced-sarcasm.csv" file for the upcoming process.

After unzipping the "train-balanced-sarcasm.csv" file and storing it in the "reddit_df" variable, I chose a random subset consisting 10% of the original dataset, and I stored it in the "reddit_sample" variable to use in the "first path". I also used the "random_state" parameter to ensure that the same random subset selection would be reproducible across multiple runs of the code.

Then I performed basic exploratory data analysis to understand the subset's structure, the diversity within the features, and the most important property in this step, which was the balance between the target classes, as it ensures that the models are not biased towards any class.

2-2-2) Text Preprocessing

Text data in the "parent_comment" column required some text preprocessing to convert it into a more suitable format for the machine learning algorithms. Thus, I decided to define a function that executes text lowercasing, punctuation removal, word tokenization, stemming, and stopword removal and then I applied the function to the "parent_comment" column in the subset (for the first path), and in the whole dataset (for the second path) to have more prepared data for my project.

2-2-3) Feature Engineering and Target Defining

In this step, I had to convert all the columns that were needed to be in my features from categorical data to numeric ones. According to the research question, I had to use the "parent_comment" and "subreddit" columns as the features and both were in the type of object in my subset.

Based on the nature of the "parent_comment" column and my technical limitations I had the following considerations to choose the suitable technique for the purpose of transformation of the categorical data into numeric data:

1) Since I had data with a large vocabulary, it was crucial to reduce dimensionality by focusing on the vocabulary that was more informative.
2) According to my limitations in the Google Colab service, I had to choose a technique with the ability of sparse representation as most elements in text vectors became zero after the transformation and I needed to use my working memory in a more efficient way.

Hence, I decided to use the TF-IDF (Term Frequency – Inverse Document Frequency) technique that naturally reduced dimensionality by focusing on more important terms (words) in each parent comment and presented a sparse matrix that stored only non-zero elements. The TF-IDF formula is presented as below:

$$tfidf(t,d,D) = tf(t,d).idf(t,D)$$

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

$$idf(t,D) = log\frac{N}{|\{d \in D : t \in d\}|}$$

Where:

$tf(t,d)$ is term frequency. (The relative frequency of term t within document d)

$f_{t,d}$ is the raw count of a term in a document. (The number of times that term t occurs in document d.)

$\sum_{t' \in d} f_{t',d}$ is the total number of terms in document d. (Counting each occurrence of the same term separately)

$idf(t, D)$ is a measure of how much information the word provides. (If it is common or rare across all documents.)

N is the total number of documents in the corpus.

$|\{d \in D : t \in d\}|$ is the number of documents where the term t appears.

Similar to the "parent_comment" column, I had also some considerations for the "subreddit" column to choose the suitable technique for the purpose of transformation of the categorical data into numeric data:

1) Since I had 5651 unique subreddits in my subset and 14878 unique subreddits in the whole dataset, I had to choose a technique that reduces dimensionality without introducing any noises or biases.
2) Like the technical limitations with the "parent_comment" column, I had to choose a technique with the ability of sparse representation as most of the values in the matrix became zero after the transformation.
3) Since in the "parent_comment", the output of the TF-IDF vectorizer was also a sparse matrix, I had to look for a more compatible and more efficient technique to combine these two features together in the final feature matrix.

Thus, one-hot encoding was used for the "subreddit" column to convert its categorical values into a binary matrix format because of its suitability for my project and efficiency of representing the information in a compatible form with the other feature which was "parent_comment".

Then "X_text" and "subreddit_one_hot_sparse" matrices were concatenated horizontally to produce a single combined feature matrix, "X", to be used for the training and testing procedures, and finally, the "label" column which was numeric data was considered as the target, "y", in the project.

2-2-4) Data Splitting

In this step, the data in the subset was split into train and test sets with 80% of the data used for the training and 20% for the testing. The "random_state" parameter was used to ensure that the same random split of data into train and test sets is reproducible across multiple runs of the code.

N.B: For the "Second Path" all the processes of "Feature Engineering", "Target Defining", and "Data Splitting" were performed exactly in the same way with a slight modification in the notations.

2-2-5) Model Training

As I mentioned in the initial lines of this section, I tried to perform this project in two different paths:

1) **First Path:** Performing all the processes of model training with cross-validation, prediction, and evaluation for all four algorithms on a random subset to have an insight into the potential effectiveness of each model and then comparing the results to select the best one among them.
2) **Second Path:** Instantiating and training the models on a random subset and applying a cross-validation process to find the best model before performing the prediction step and then choosing the best model to do the training and prediction procedures on the whole dataset.

For the "First Path", four different supervised text classification algorithms were employed on the subset as below:

1) Logistic Regression / 2) K-Nearest Neighbors (KNN) / 3) Decision Tree / 4) Random Forest

A 4-fold cross-validation technique was applied separately to all the algorithms to evaluate the performance of each model on the training data. In other words, by applying cross-validation with cv=4, the training data (X_train) was divided into 4 folds (subsets), and all models were trained 4 times, each time using 3 of these folds for training and the remaining one for the validation. The results were the arrays of 4 accuracy values (one for each fold) for each model and the average of these values in each array provided a more robust estimate of the performance of the corresponding model on unseen data.

After the training process of each model with cross-validation, all the models were fitted to the "X_train" and the "y_train" of the subset ("reddit_sample") to finalize the training procedure. In the upcoming lines, a brief explanation of each algorithm is presented:

2-2-5-1) Logistic Regression Algorithm

Logistic Regression is a statistical model used for binary classifications. It estimates the probability that a given input belongs to a certain class. The logistic regression model uses the logistic function to squeeze the output of a linear equation between 0 and 1. The logistic function is defined as:

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \ldots + \beta_n X_n)}}$$

Where:

$P(Y = 1|X)$ is the probability that the target variable Y is 1 given the values of the feature variables X.

$\beta_0, \beta_1, \ldots, \beta_n$ are the parameters of the model.

$X_1, \ldots, X_n$ are the features variables.

The goal is to find the parameters β that maximize the likelihood of observing the sample data.

In this project, I used LogisticRegression (max_iter=1000) since I did not want to encounter the convergence warning because of the size of my data.

2-2-5-2) K-Nearest Neighbors (KNN)

KNN is a type of instance-based learning algorithm that can be used for classification problems. It classifies a data point based on the majority class among its K nearest neighbors. Given a new observation X, the KNN algorithm searches the training set for the K training examples that are closest to X and returns the most common output value among them. The distance between points can be measured using various metrics such as Euclidean distance or Cosine Similarity Distance.

In this project, I used KNeighborsClassifier(n_neighbors=3, algorithm='brute', metric='cosine') since these modifications on the main algorithm were helpful in working with a great number of text data that was processed by the TF-IDF technique.

2-2-5-3) Decision Tree

A Decision Tree is a flowchart-like tree structure where each internal node represents a feature, the branch represents a decision rule, and each leaf node represents the outcome. A Decision Tree algorithm parses the training data hierarchically by imposing conditions over attributes so that documents belonging to each class are predominantly placed in a single node. These split conditions are then used to assign test instances with unknown labels to leaf nodes. The majority class of the leaf node is used to predict the label of the test instance.

In this project, I used DecisionTreeClassifier(max_depth=10, random_state=101) to prevent the tree from growing very deep, since I had data with a big size and also, I wanted to set a seed for the model random processes to ensure its reproducibility.

2-2-5-4) Random Forest

Combinations of multiple Decision Trees can be used to create the Random Forest algorithm, which is one of the best-performing classifiers among supervised text classification algorithms. In other words, Random Forest takes the concept of a single Decision Tree, adds layers of randomness to create a whole forest of diverse trees, and then leverages the combined wisdom of this forest to make more accurate and robust predictions.

In this project, I used RandomForestClassifier(n_estimators=10, random_state=101, n_jobs=-1) since I wanted to make whole the process faster with respect to the amount of data and also I wanted to be sure about the process reproducibility of the Random Forest model.

For the "Second Path", those four models were instantiated and trained on the extracted train set from our subset "reddit_sample". Since I wanted to apply the cross-validation evaluation on four different types of models, I decided to use the stratified cross-validation method in this part to ensure that each fold is a good representative of the overall train set. Thus, A Stratified 4-fold Cross-Validation was employed on them to evaluate their mean accuracy scores. This rigorous evaluation provided a clear comparison of their performances.

After this evaluation process, the best model, which was the Logistic Regression model, was chosen for the training and prediction procedures on the whole dataset. (reddit_df)

As mentioned before, all the processes of “Text Preprocessing”, “Feature Engineering”, “Target Defining”, and “Data Splitting” were performed exactly like for the first path with a slight modification in the notations where they were needed.

3) Experimental Results

3-1) Dataset Overview

In this project, the “train-balanced-sarcasm.csv” file from the “Sarcasm on Reddit” dataset was used to perform both training and testing procedures. This dataset is available on the Kaggle website and contains 1010826 rows of comments which were written on their corresponding parent comments on the related subreddits by some authors (Redditors) and were labeled as sarcastic (1) or non-sarcastic (0). This dataset is a balanced dataset with exactly 505413 sarcastic labels and 505413 non-sarcastic labels, and it contains the date and time of comment creation and the scores that were given to the comments by the other Redditors.

3-2) Evaluation Metrics

After performing the prediction phase and specifically predicting the probability of receiving a sarcastic comment from the parent comments, the models’ performances were evaluated based on Accuracy, Confusion Matrix, Precision, Recall, F1-Score, and AUC-ROC analysis in both two paths of the project. In the upcoming lines, a brief explanation of each metric is presented:

3-2-1) Accuracy

Accuracy is simply the percentage of correct predictions out of all predictions. In this project, this metric is shown in the form of the number of correct predictions and the ratio of the correct predictions for all the models.

3-2-2) Confusion Matrix

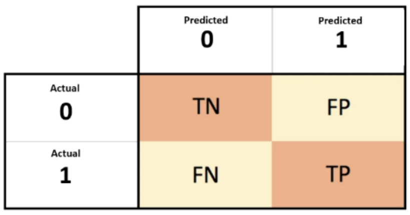
Confusion Matrix is a table that is used to describe the performance of a classification model on a set of data for which the true values are known. Confusion Matrix has four different parts as below:

True Negative (TN): Actual is negative, and the prediction is also negative.

True Positive (TP): Actual is positive, and the prediction is also positive.

False Positive (FP): Actual is negative, but the prediction is positive. (Type 1 error)

False Negative (FN): Actual is positive, but the prediction is negative. (Type 2 error)



	Predicted 0	Predicted 1
Actual 0	TN	FP
Actual 1	FN	TP

3-2-3) Precision, Recall, F1-Score (Classification Report)

In the classification Report, we have three different metrics that are explained below:

Precision is the ratio of correctly predicted positive observations to the total predicted positives. $\text{Precision} = \frac{TP}{TP + FP}$

Recall is the ratio of correctly predicted positive observations to all observations in the actual class. $\text{Recall} = \frac{TP}{TP + FN}$

F1-Score is the weighted average of precision and recall. $\text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$

In fact, the f1-score tries to find the balance between precision and recall. We then have the metrics' Macro Average and Weighted Average, which show each metric's average respectively without and considering each class's support. Thus, these two items present an insight into the data balance.

3-2-4) AUC-ROC (Area Under Curve – Receiver Operating Characteristic) Analysis

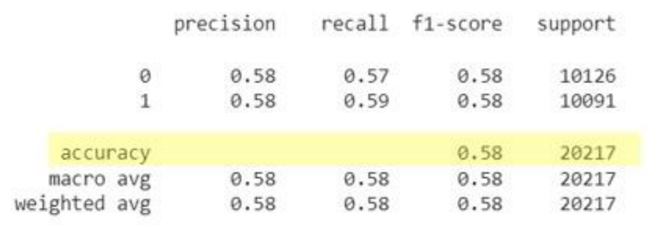
AUC-ROC or the Area Under the ROC Curve provides a measure of how well the model can correctly classify instances with the positive label (label=1) while minimizing false positives. The AUC-ROC is a threshold-independent measure. It is a single scalar value that summarizes the performance of the binary classifiers. A value of 1 represents a perfect classifier, while a value of 0.5 represents a random classifier.

To evaluate models' performances in prediction of the probability of receiving sarcastic comments, the AUC-ROC can be a more informative measure than accuracy.

3-3) Experimental Findings

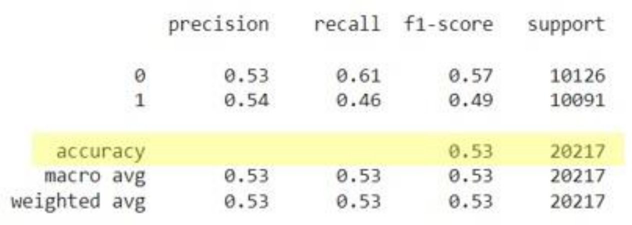
Based on the discussed metrics, the below results were achieved after the evaluation processes:

First Path:



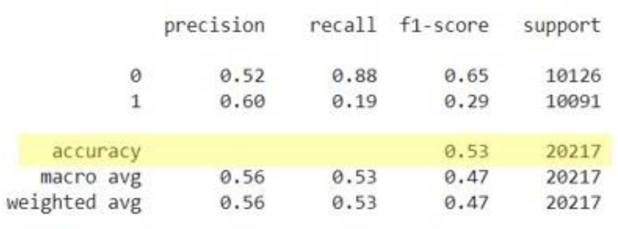
```
              precision    recall  f1-score   support

           0       0.58      0.57      0.58     10126
           1       0.58      0.59      0.58     10091

    accuracy                           0.58     20217
   macro avg       0.58      0.58      0.58     20217
weighted avg       0.58      0.58      0.58     20217
```

Accuracy and the Classification Report of the Logistic Regression Model



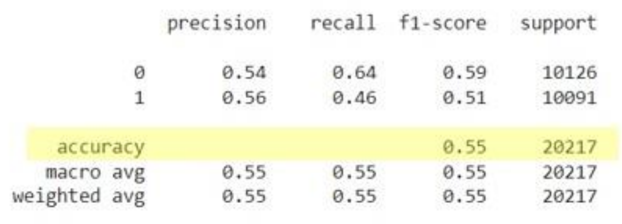
```
              precision    recall  f1-score   support

           0       0.53      0.61      0.57     10126
           1       0.54      0.46      0.49     10091

    accuracy                           0.53     20217
   macro avg       0.53      0.53      0.53     20217
weighted avg       0.53      0.53      0.53     20217
```

Accuracy and the Classification Report of the KNN Model



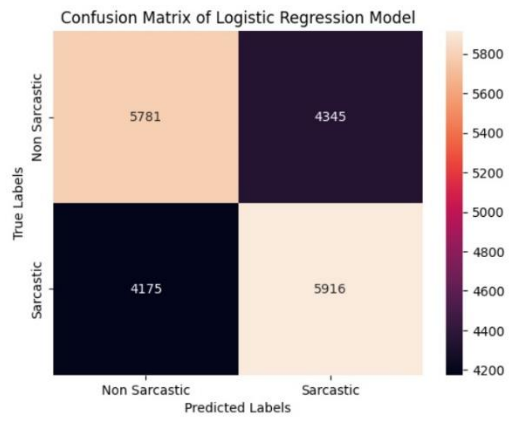
```
              precision    recall  f1-score   support

           0       0.52      0.88      0.65     10126
           1       0.60      0.19      0.29     10091

    accuracy                           0.53     20217
   macro avg       0.56      0.53      0.47     20217
weighted avg       0.56      0.53      0.47     20217
```

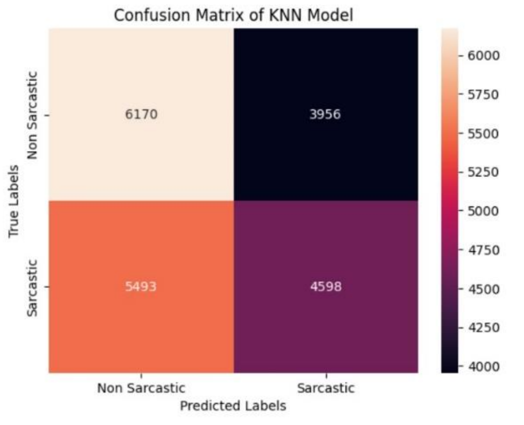
Accuracy and the Classification Report of the Decision Tree Model

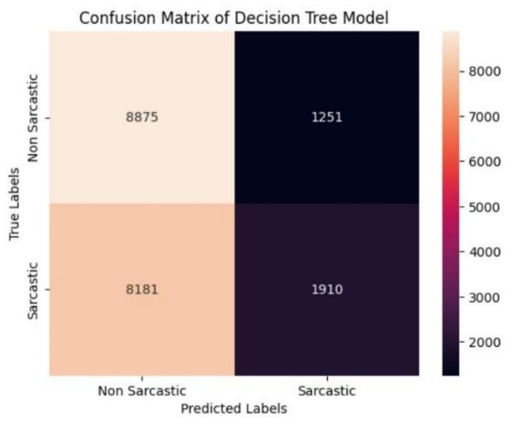


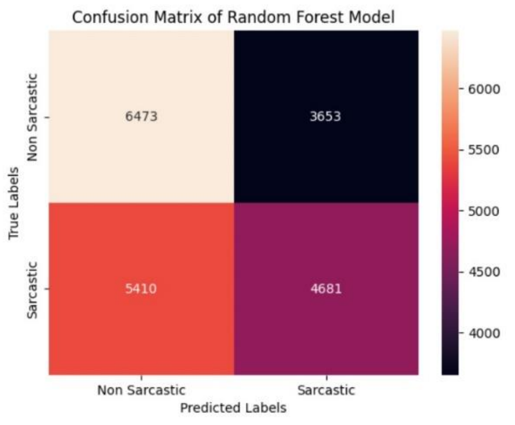
```
              precision    recall  f1-score   support

           0       0.54      0.64      0.59     10126
           1       0.56      0.46      0.51     10091

    accuracy                           0.55     20217
   macro avg       0.55      0.55      0.55     20217
weighted avg       0.55      0.55      0.55     20217
```

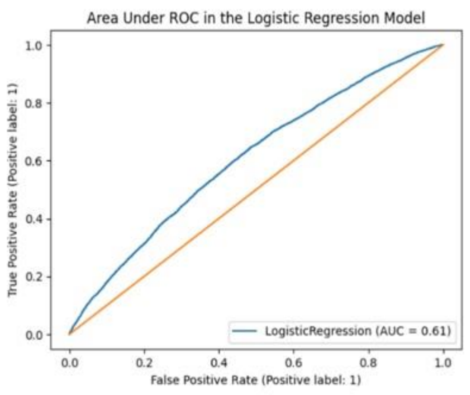
Accuracy and the Classification Report of the Random Forest Model

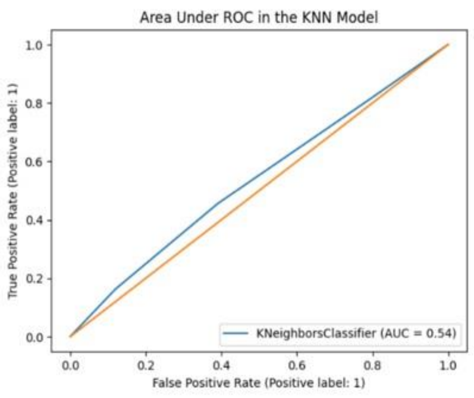


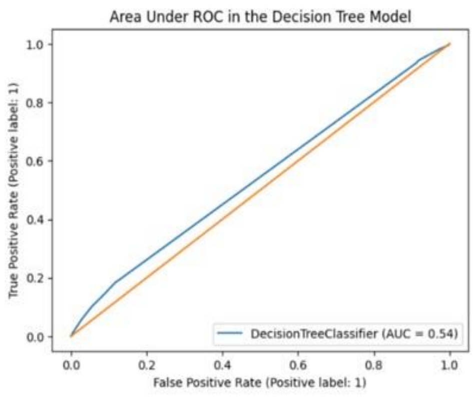


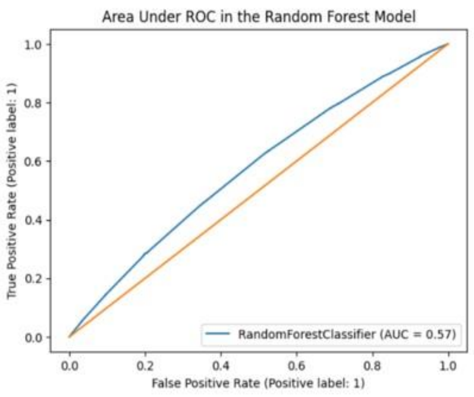


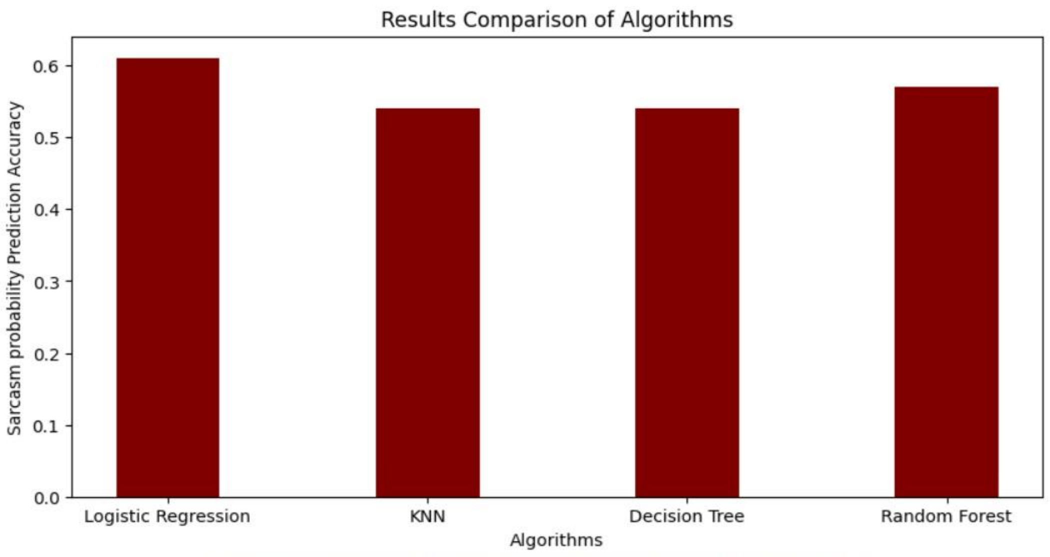








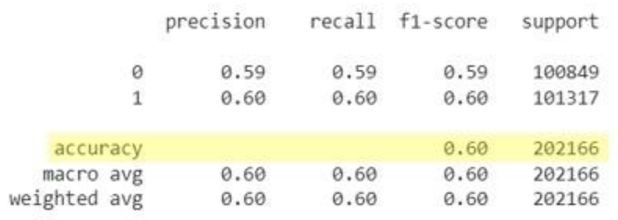




Performance accuracy of the models in sarcasm probability prediction

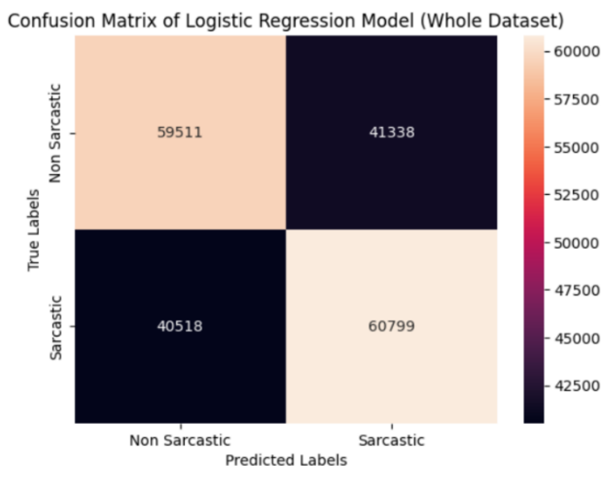
According to the presented results, the models demonstrated varying levels of success in predicting the labels, and the logistic regression model showed relatively higher accuracy in predicting the probability of receiving sarcastic comments. The comparison of the models based on the ROC curve which illustrates the accuracy of predicting sarcasm, was visualized in the bar plot above to provide a clear insight into each model's performance.

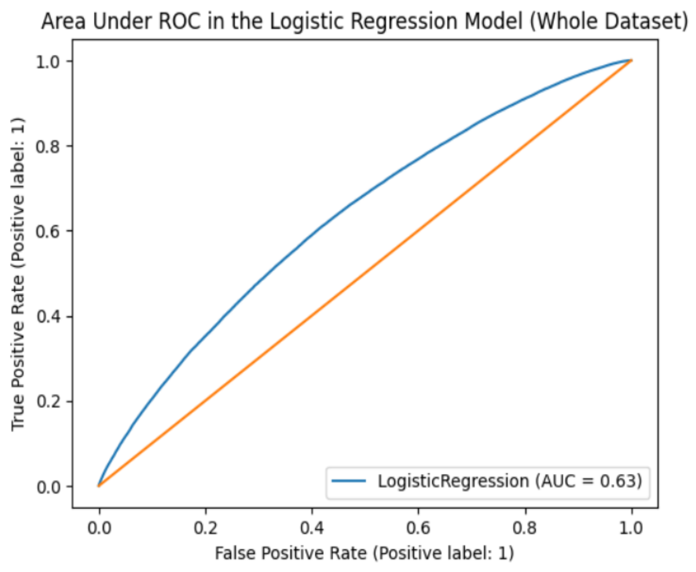
Second Path: In this part of the project, after training the Logistic Regression model (which was the best-evaluated model in the Stratified 4-fold Cross-Validation process with the cv-score=0.57) on a train set extracted from the whole dataset (reddit_df) and performing the prediction procedure, the evaluation metrics showed that the final accuracy of the sarcasm probability prediction had a slight improvement in compare to the results which were achieved from the first path. (Based on the AUC-ROC analysis, the accuracy of the logistic regression model in sarcasm probability prediction performed on the subset was 0.61 but the accuracy of the model performed on the whole dataset was 0.63.)



```
              precision    recall  f1-score   support

           0       0.59      0.59      0.59    100849
           1       0.60      0.60      0.60    101317

    accuracy                           0.60    202166
   macro avg       0.60      0.60      0.60    202166
weighted avg       0.60      0.60      0.60    202166
```

Accuracy and the Classification Report of the Logistic Regression Model (Whole Dataset)





4) Concluding Remarks

4-1) Discussion on the Results

In this project, four different supervised machine-learning algorithms were successfully employed to detect sarcasm in Reddit comments based on the parent comments and subreddits. As it is interpretable from the visualized results, the performance accuracies of the models are not desirable, and these models are not perfect predictors. However, a small improvement in accuracy was achieved in the Logistic Regression model in the result of applying the model to the whole dataset in the second path. It is also understood that the balanced data helped the models to proceed unbiased in terms of precision and recall. Finally, it is illustrated that with the chosen parameters for the models, based on my limitations in the project, the Logistic Regression and the Random Forest models had better performances, while the KNN and the Decision Tree models' results were so close to a totally random classifier result.

4-2) Future Work

According to the results, there is an opportunity for the models' performance improvement. Future work could be refining the models further by hyperparameters tuning or experimenting with more advanced models like neural networks. It is also possible to explore other feature engineering techniques, more sophisticated text preprocessing methods, or incorporate additional features to enhance the predictive performance of the models.

References

1) https://www.kaggle.com/datasets/ danofer/sarcasm

2) Mikhail Khodak, Nikunj Saunshi, Kiran Vodrahalli. A Large Self-Annotated Corpus for Sarcasm

3) Christopher Ifeanyi Eke, Azah Anir Norman, Liyana Shuib, Henry Friday. Sarcasm identification in textual data: systematic review, research challenges, and open directions

4) Aditya Joshi, Pushpak Bhattacharyya, Mark J. Carman. Automatic Sarcasm Detection: A Survey