coreML'23

Twitter Project

Aniket Sanyal (7023518), Yamini Supriya (7010131), Shams Ibne Noor (701589)

Summer 2023

1 Introduction

The purpose of this project was to perform sentiment analysis on a given dataset consisting of tweets from Twitter. We use different machine learning and deep learning methods in order to fulfil two primary objectives. Namely, using regression in order to compute a sentiment score of a tweet, and classification to determine the category of sentiment(i.e. positive, negative or neutral) of a tweet. This report illustrates the pre processing steps carried out, the models used and an overview of the results of the model.

2 Data Analysis and Preprocessing

2.1 Regression Task

Our regression task aimed to predict 'score_compound' values for each tweet, which range from -1 to 1 and signify sentiment polarization. Values near 1 imply positive sentiment, while those near -1 suggest negativity. The histogram (Figure 1) illustrates this distribution, revealing that many tweets are 'neutral' (around 0). Interestingly, 'positive' tweets outnumber 'negative' ones in our dataset.

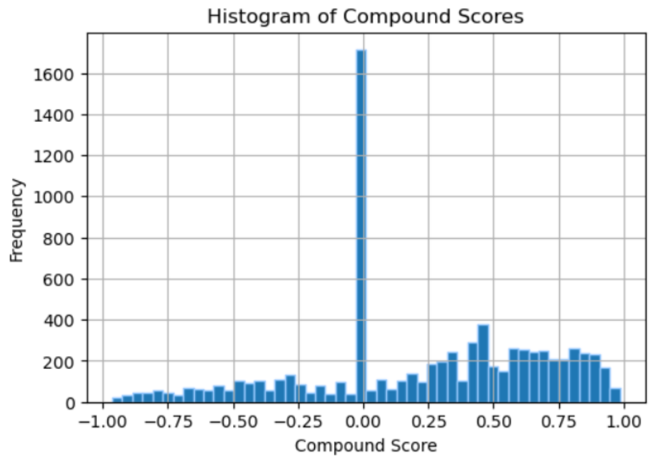
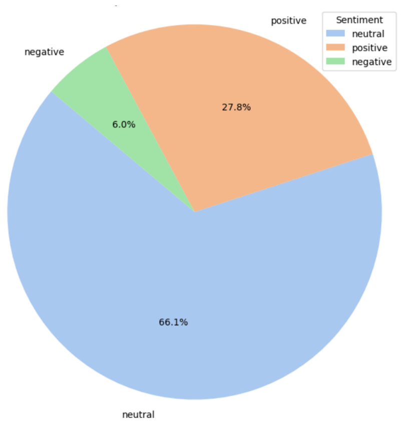


Figure 1: 'score_compound' distribution in tweets_train data

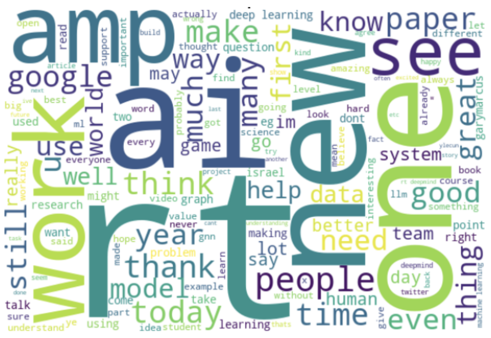
In the regression task, one of our targetted parameters was the 'text' column of our tweets_train dataset. In this column, we had our tweets in raw text format which required some preprocessing before putting through our regression models. To conduct a sentiment analysis we need the contents in the text column in a format where the noise is reduced and the output is tokenized. For processing the text column we implemented a text_preprocessing() function. Here in this function, we decapitalized all the text, used the 're' library to remove non-alphabetic characters, used nltk library to remove all the stopwords such as "and", "the", "is" etc., and also tokenize tweets into individual words. Also, we used the Porter Stemmer, which reduces words to their base or root forms. Apart from the 'text' column we also took several other features into account for our regression task, among them the 'possibly_sensitive' that had 'true' or 'false' as its value which we changed to 1 and 0.

2.2 Classification Task

Effective data preprocessing is essential to ensure the quality and relevance of the data used for training a machine learning model. Here are the steps we took to clean and preprocess the text data. These steps include converting all text in the 'text' column to lowercase to ensure consistent representation and avoid treating differently cased words as separate entities. Punctuation marks and special characters were removed from the text, significantly reducing noise and simplifying the data. URLs and web addresses within the text were eliminated using regular expressions to remove potentially irrelevant information. Additionally, common stopwords like 'and,' 'the,' and 'is' were removed to focus on more meaningful words and reduce data dimensionality. To further streamline the text, stemming and lemmatization techniques were applied, reducing words to their base or root forms, thus minimizing variations. These preprocessing steps collectively enhance the quality and suitability of the text data for subsequent sentiment classification model training.
A pie chart and word cloud were constructed from the training set to capture import information such as the sentiment distribution and important words in the tweets.



(a) Proportion of sentiments



(b) Wordcloud of all tweets

Figure 2: Information after preprocessing the tweets

3 ML Modelling

3.1 Regression Task

Initially we went with a basic Linear Regression model where we took the tweet text and scores in to account. We preprocessed the 'text' according to above mentioned methods and later used 'HashingVectorizer' to reduce the dimensionality of the text and vectorize them. We splited our training dataset in 80-20 split to get our train and validation data. We used LinearRegression from sklearn.linear_model library to conduct this regression.

Recognizing the complexity of the sentiment score, we recognized the necessity of considering additional parameters. We later went Multiple Regression method where we could use some other features of our training data.We hypothesized that factors like 'retweet_count,' 'like_count,' and 'possibly_sensitive' could significantly influence sentiment scores. We implemented a multiple regression model where we took these three features along side the preprocessed texts as our independent variables.We utilized the LinearRegression library, using its capability to handle multiple parameters in a systematic manner.

Later on, we focused on techniques where we could address the issue of overfitting and colinieարity into account as well. As we are considering multiple features for our regression task it is possible that there might be correction between the features. From a realword perspective a tweet's likes count can be correlated to it's number of replies. Considering those issues, we decided to go with Ridge and Lasso regression methods. Both of the regression methods corresponds to L2 L1 regularization that is useful for prevent overfitting. We created two separate pipelines for Ridge and Lasso regression and trained our data through those pipeline. For the Ridge Regression we used Ridge(alpha=1.0) and for Lasso Regression we used Lasso(alpha=0). For both of the regression models we used TF-IDF vectorizer to vectorize our corresponding parameters. We added categorical data such as tweets type using OneHotEncoder and also added some other features which we didn't use in the earlier methods.

3.2 Classification Task

We initially employed logistic regression as a baseline model to predict the sentiment labels of the tweets. Logistic regression is a simple yet powerful linear classification algorithm widely used for binary and multi-class classification tasks. For our implementation, we utilized the LogisticRegression class from the sklearn.linear model module, with a maximum of 1000 iterations to ensure convergence.

Later we explored an alternative approach to address the class imbalance in multi-class classification tasks. Class imbalance can impact model performance, so we employed an upsampling technique on the training data. This aimed to enhance minority class representation ('positive' and 'negative'), creating a more balanced distribution and potentially improving generalization on unseen data. To do this, we used the Random Forest Classifier [1], an ensemble learning algorithm. This technique builds multiple decision trees and combines their predictions for accurate and robust classifications.

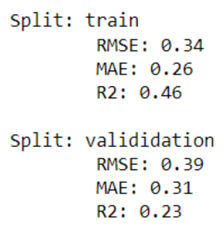
We also examined the use of a Bidirectional Long Short-Term Memory (LSTM) neural network [2], designed for sequence data like text. Created with the Keras library in Python, this model captures context from past and future tokens, making it great for text classification. Addressing class imbalance, we assigned class weights during training. This boosts the importance of minority classes ('negative' and 'positive'), helping the model learn their patterns effectively.

We employed scikit-learn's KNeighborsClassifier and GridSearchCV for hyperparameter optimization. KNeighborsClassifier is a k-nearest neighbors classifier, making predictions based on nearby data points. GridSearchCV conducts an exhaustive search over specified hyperparameters to find optimal values, enhancing the classifier's performance. The code fits the model using training data and target labels. The purpose is to determine the ideal number of neighbors for the classifier, enhancing its predictive accuracy.

Finally we employed the Gradient Boosting classifier [3], an ensemble technique that combines outputs from multiple weak learners, usually decision trees, to build a strong predictive model. Gradient Boosting reduces errors iteratively by fitting new models to previous models' errors, guided by gradient descent optimization. This process aims to minimize a loss function by adjusting weak learners' parameters. The algorithm was used for sentiment analysis on Twitter data, converting text to numerical features using TF-IDF. To address class imbalance, we employed RandomOverSampler, ensuring unbiased predictions across sentiment classes. Training on resampled data showed promising results, especially in improving recall for the challenging negative sentiment class.

4 Model Selection

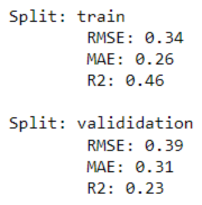
4.1 Regression Task



```
Split: train
        RMSE: 0.34
        MAE: 0.26
        R2: 0.46

Split: valididation
        RMSE: 0.39
        MAE: 0.31
        R2: 0.23
```

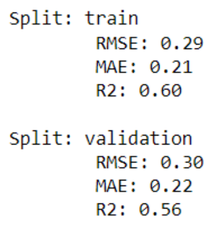
(a) Linear Regression



```
Split: train
        RMSE: 0.34
        MAE: 0.26
        R2: 0.46

Split: valididation
        RMSE: 0.39
        MAE: 0.31
        R2: 0.23
```

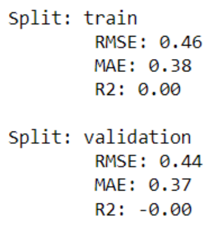
(b) Multiple Regression



```
Split: train
        RMSE: 0.29
        MAE: 0.21
        R2: 0.60

Split: validation
        RMSE: 0.30
        MAE: 0.22
        R2: 0.56
```

(c) Ridge Regression



```
Split: train
        RMSE: 0.46
        MAE: 0.38
        R2: 0.00

Split: validation
        RMSE: 0.44
        MAE: 0.37
        R2: -0.00
```

(d) Lasso Regression

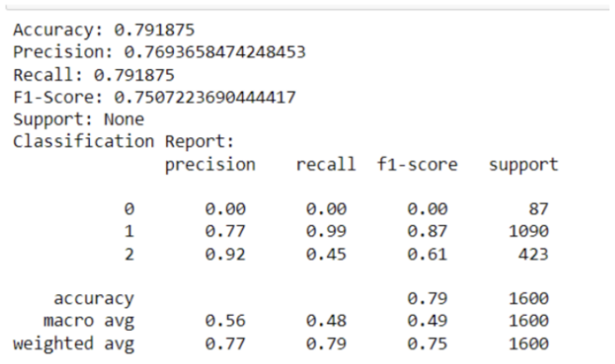
Figure 3: Performance metrics for regressions models

In the regression task for selecting our model, we compared several aspects such as RMSE, MAE, and R-squared value into accounts. We aimed on selecting a model that has lower RMSE (Root Mean Squared Error) and MAE (Mean Absolute Error) values as these are the indicator for better predictive accuracy. We observed the R-squared value of each model We also looked into the predicted scores that we got from our tweet_test2 dataset just to have a general idea of how our models are working.

In figure 3 (a) and (b) we can see that the Linear Regression model and the Multiple Regression model have the same performance metrics. One of the reasons for this outcome might be colinearity between a couple of parameters. Overfitting might also be the issue here. However, if we look into the RMSE and MAE values from Ridge and Lasso Regression we can see that there are some differences between them. The ridge regression (c) has the lowest RMSE and MAE value for both the training and validation split. Also, the R-squared value of Ridge regression is higher than other models. This indicates that the Ridge Regression model aligns with the actual variation in the test data. Considering the performance metrics for all four regression models it seems that the Ridge regression model is the best one of the bunch.

4.2 Classification Task

The goal of model selection is not only to ensure a decent accuracy, but to ensure good performance of the model across all the three possible sentiments. Hence the important metrics to differentiate between model performances is recall and F1 score, which are especially handy when there is an inherent class imbalance in the dataset.

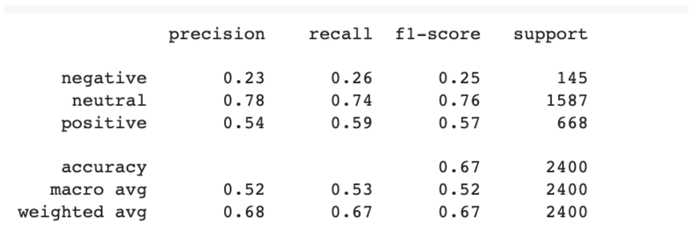


```
Accuracy: 0.791875
Precision: 0.7693658474248453
Recall: 0.791875
F1-Score: 0.7507223690444417
Support: None
Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        87
           1       0.77      0.99      0.87      1090
           2       0.92      0.45      0.61       423

    accuracy                           0.79      1600
   macro avg       0.56      0.48      0.49      1600
weighted avg       0.77      0.79      0.75      1600
```

(a) Results of Logistic Regression



```
              precision    recall  f1-score   support

    negative       0.23      0.26      0.25       145
     neutral       0.78      0.74      0.76      1587
    positive       0.54      0.59      0.57       668

    accuracy                           0.67      2400
   macro avg       0.52      0.53      0.52      2400
weighted avg       0.68      0.67      0.67      2400
```

(b) Results of Random Forest Classifier

Figure 4: Results of 2 models

The model demonstrated promising performance in terms of precision and recall, with an F1-Score of 0.75. However, further evaluation on the test set revealed an F1-Score of only 53.44 percent, indicating a need for improvement. To address this, we explored more complex models and fine-tuned hyperparameters to enhance the model's performance on the validation set, aiming for improved generalization to unseen data.

This model greatly improves the recall and f1 score for tweets with a negative sentiment, which were all incorrectly classified by Logistic Regression.

Upon evaluation of the Bidirectional LSTM model on the validation set, we observed an F1-score of 0.7660399419163408. This suggests promising performance in capturing sentiment patterns in the text data. However, applying the same model to the test set yielded an F1-score of 0.6319887889984773.

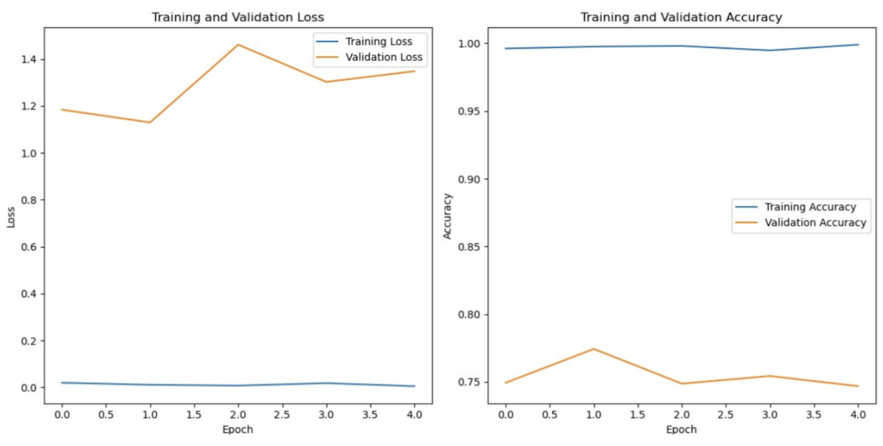
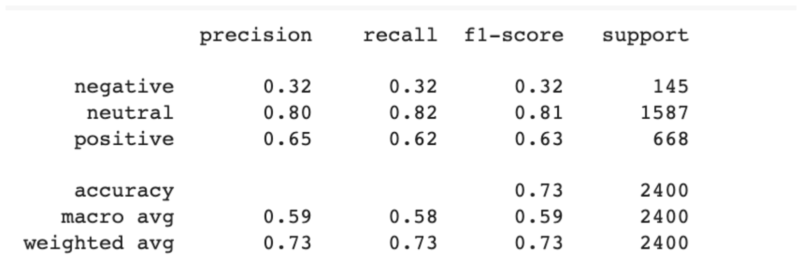


Figure 5: Training vs Test Loss for BLSTM model

By closely examining loss curves and accuracy trends, we find that it suggests potential overfitting, where the model memorizes the training data without generalizing well. To address this, we considered techniques like early stopping or model simplification to improve generalization and achieve better validation performance.

The gradient boosting classifier yields the following results



```
              precision    recall  f1-score   support

    negative       0.32      0.32      0.32       145
     neutral       0.80      0.82      0.81      1587
    positive       0.65      0.62      0.63       668

    accuracy                           0.73      2400
   macro avg       0.59      0.58      0.59      2400
weighted avg       0.73      0.73      0.73      2400
```

Figure 6: Results for XG Boost Classifier

Hence we see that it achieves the best recall and F1 score amongst all the models that we have implemented.

5 Empirical Results

5.1 Regression Task

The model that we selected for our regression task is the Ridge Regression model, this model had the lowest RMSE for training 0.29 for validation 0.30. This low RMSE value indicates that the average residual value is lower than the other model. This means that the model predictions of score_compound are close to the actual data. The MAE value (training 0.21 validation 0.22) also gives us a similar outcome. Also, the higher R-Squared value for the Ridge Regression model suggests that our prediction is well aligned with our actual data.

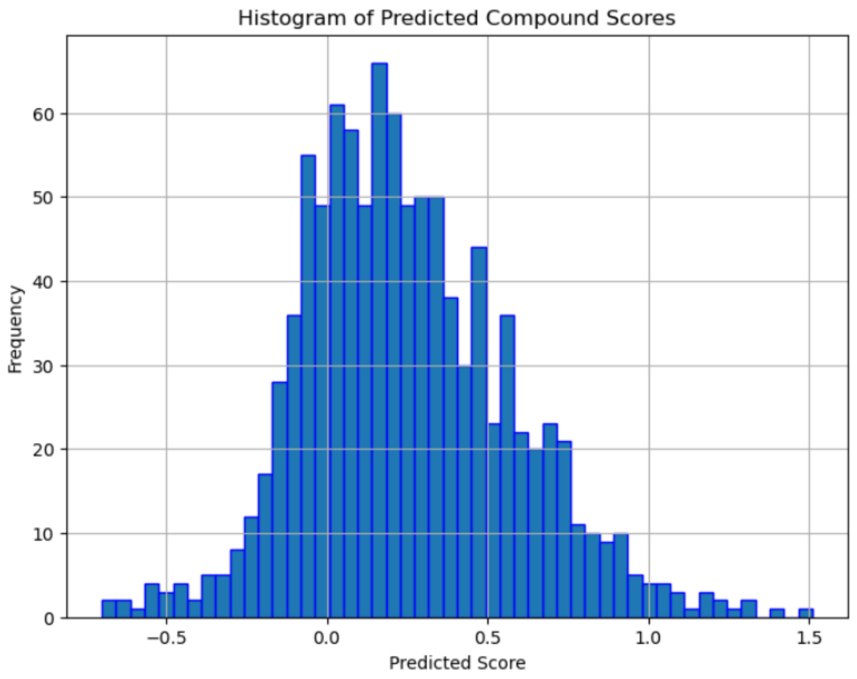
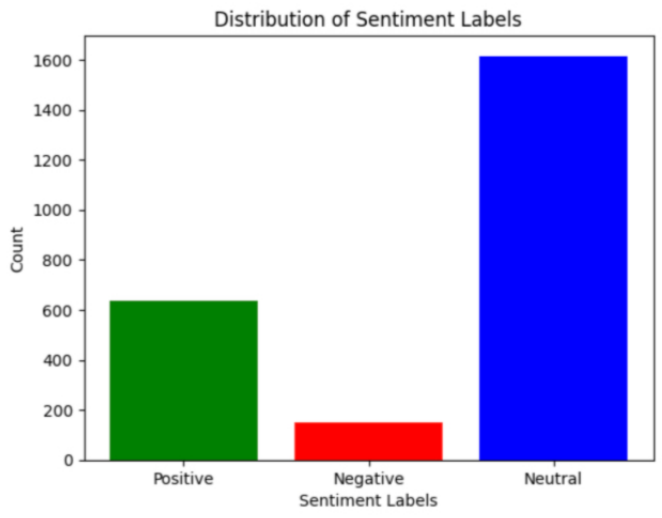
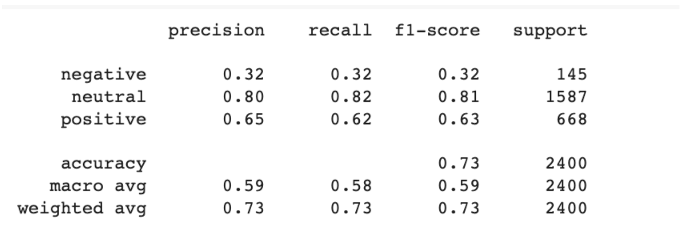


Figure 7: Distribution of predicted_score in tweet_test2 dataset

5.2 Classification Task

The final model selected for classification is the XG Boost Classifier, which is an advanced gradient boosting classifier, coupled with a RandomOverSampler to address the class imbalance. We find the following class distribution in the test split





```
              precision    recall  f1-score   support

    negative       0.32      0.32      0.32       145
     neutral       0.80      0.82      0.81      1587
    positive       0.65      0.62      0.63       668

    accuracy                           0.73      2400
   macro avg       0.59      0.58      0.59      2400
weighted avg       0.73      0.73      0.73      2400
```

(a) Sentiment Distribution of Tweets for Validation set

(b) Validation Set Results

Figure 8: Empirical Results for classification

The distribution of the predicted sentiments on the test split follows closely on what we expect i.e. similar to the training set : most of the tweets are neutral, some are positive, while a few are negative. This also explains the follows F1 scores -

The F1 score, which is the most important metric for imbalanced dataset analysis, is the highest for the neutral tweets and least for the negative tweets. This model also boasts an accuracy of 71%.

References

[1] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[2] U.B. Mahadevaswamy and P. Swathi. Sentiment analysis using bidirectional lstm network. *Procedia Computer Science*, 218:45–56, 2023. International Conference on Machine Learning and Data Engineering.

[3] Tianqi Chen and Carlos Guestrin. XGBoost. aug 2016.