MIE1628: Big Data Science
Predicting Reddit Posts Popularity

By-
Saket Thavanani

TABLE OF CONTENTS

SECTION 1: INTRODUCTION

The social media and news are great sources for news these days. But to decide whether a news or a post is real or fake is one of the key challenges of the 21st century. One such social media platform is Reddit. Reddit is a popular discussion and social media rating website. People can individually submit their posts and can get a certain score based upon users upvotes and downvotes, which further decides the rank of the post in the reddit website. The popularity of a post can depend upon a wide range of factors such as text, title, time of posting and many others. In this project, we aim to predict the future popularity of reddit posts using a wide variety of features.

Note: All the models and coding were performed on the Databricks Platform using Pyspark.

I will be roughly dividing the project into six sections-

- Data Cleaning
- EDA
- Outlier Detection
- Feature Engineering
- Feature Selection
- Modeling and Tuning

My contribution in this project was for the following sections-

- ✓ EDA
- ✓ Outlier Detection
- ✓ Feature Engineering
- ✓ Modeling and Tuning

SECTION2: DATASET

The training dataset includes 12525 rows and 56 columns. The plot shown below shows the distribution of different data types across 56 columns, we can see that string data type contributes the highest followed by boolean and integer columns.

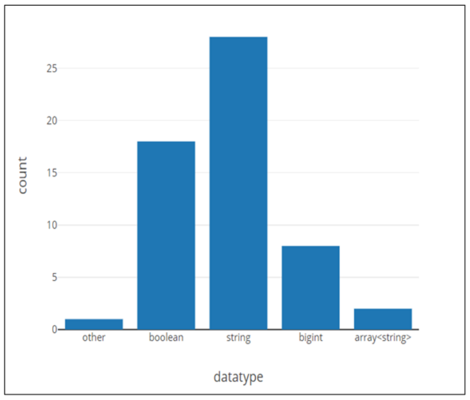


Figure 1: Distribution of Data Types in Datasets

The target variable in our case is the score column present in the form of discrete score, it can be considered as a continuous target variable when building a machine learning model. We can see that the majority of the scores are 0 as per the plots shown below. Toward the higher end scores we just have single single scores.

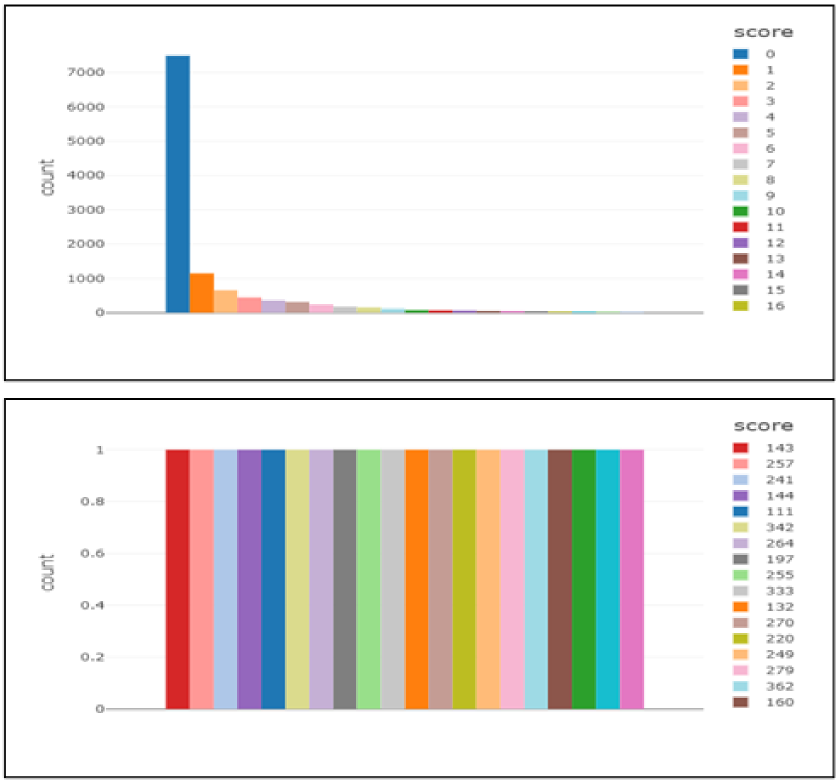


Figure 2: Distributions of Scores

SECTION 3: OUTLIER DETECTION

We can see from the score distribution that scores are heavily skewed towards one side forming a right skewed distribution. In order to reduce the skewness and identify outliers more precisely we decided to do the log transformation. Once we have the log transformations, we can see that the plot on the right is much closer to normal distribution as compared to plot on the left**. Post this, I implemented** Interquartile Range (IQR) method to eliminate outliers from the dataset with respect to the post popularity (i.e. scores).

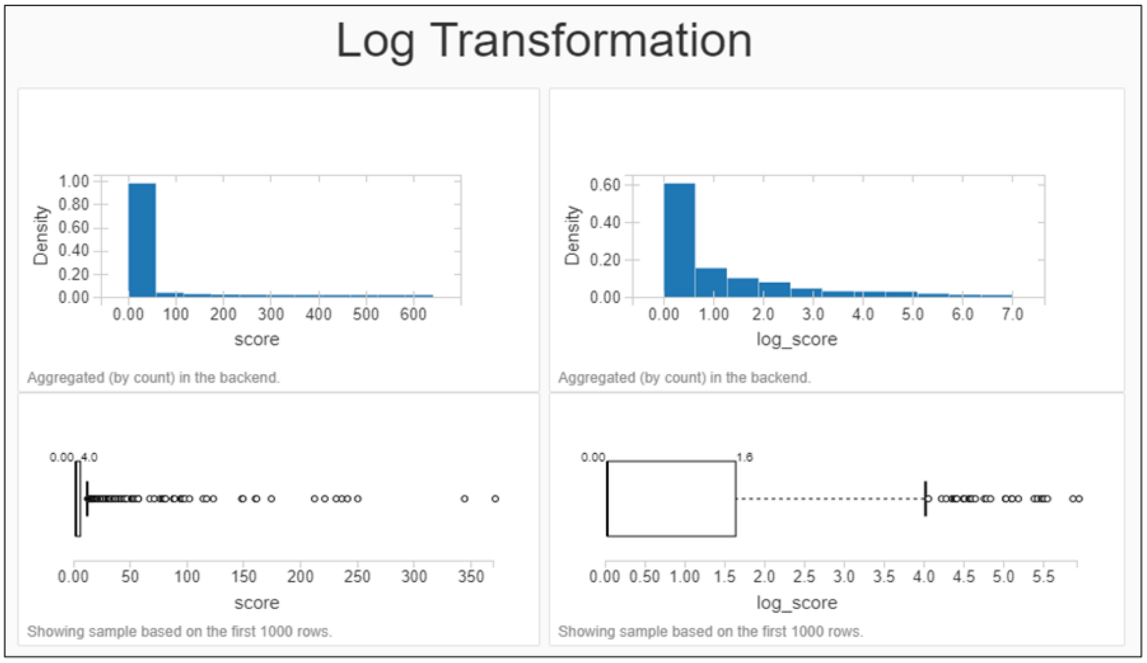


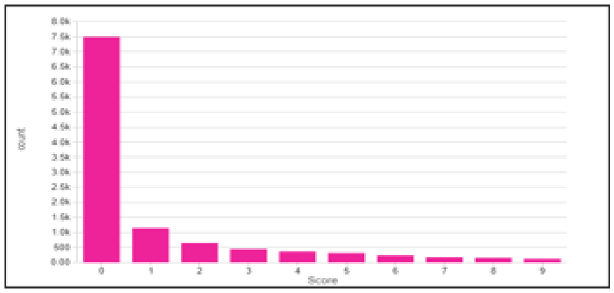
Figure 3: Distribution before and after Log Transformation in the form of histogram and box plot

Post the IQR, I noticed that all the scores above 7 were being excluded from the dataset (1600) leaving behind only low range scores below 7. This methodology was discarded because removing all the medium to high range scores from the dataset can make it difficult for the machine learning model to learn the dependencies for high range scores. So , we decided to go with the z score method which gave us reasonable results. Z-score methodology eliminated all the scores beyond +-3 standard deviation eliminating 250 samples from the dataset.

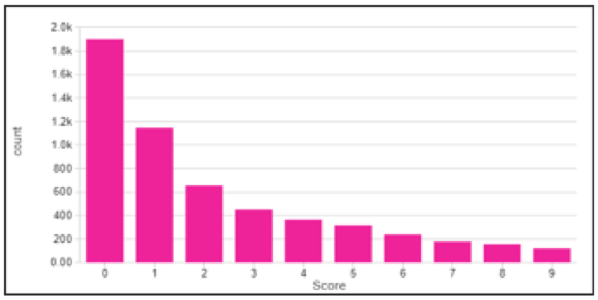
In order to consider a data point as an outlier in multivariate dimension, I decided to consider outliers with respect to two dimensions: number of comments and score, and if a data point is outlier for both the univariate dimensions then we can confidently say that the data point is highly probable to be an outlier for full dataset as well. This final consideration removed approximately 200 data points, leaving behind 12,330 data points.

SECTION 4: UNDER SAMPLING

Since the majority of the scores present in the dataset were 0 accounting to almost 75% of the dataset. In order to make our model learn all the scores appropriately we decided to adopt the under-sampling strategy. We did implement oversampling technique because it leads to a biased model which is prone to overfitting and also takes high computation time. We achieved 75% reduction in majority class after sampling and the approximately 5600 samples were removed. The figure below shows the distribution of the score before and after under sampling.



Initial Score Ratio



Score Ratio After Under sampling

Figure 4: Distribution of score before and after under sampling

SECTION5: FEATURE ENGINEERING

Apart from the existing features in the dataset, Our group made a tremendous effort to generate new innovative features from the existing ones. The new features included-

- Average length and Sentiment of the Web Scraped comments.
- 6 Window Features
- Date, Time and Day of Post
- BERT Embeddings
- Length of the tile

In the feature engineering section, **I contributed** by extracting **Window Features** and **BERT Embeddings** for Tittle Column. **I also contributed** in generating all the **Statistical plots** required for Feature engineering section using the Databricks dashboard and display functionality.

WEB SCRAPING

The column containing the URLs was used to access the website corresponding to that particular post. Further, the PRAW wrapper was used to scrape the comments from that website. The sentiment and the average length of the extracted comments were used as new innovative features.

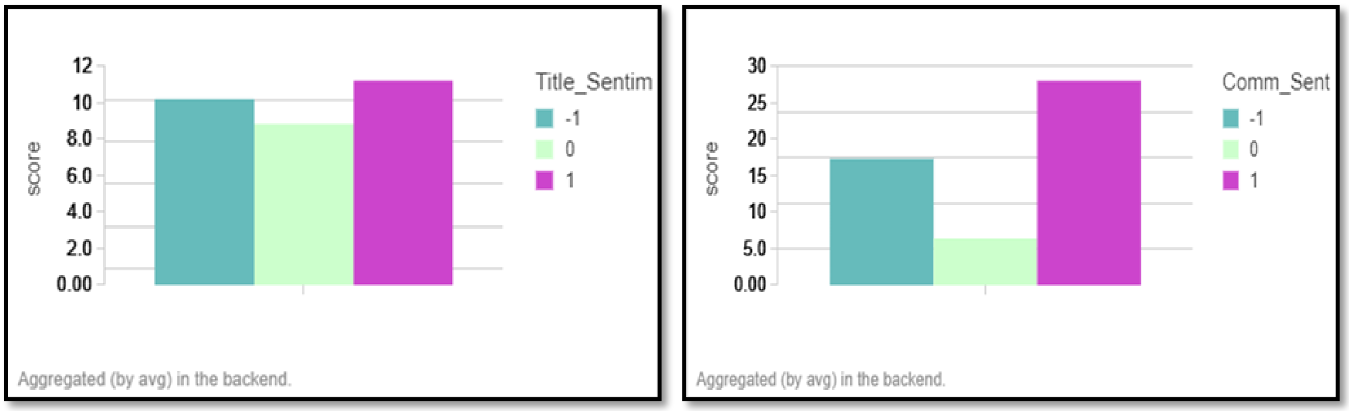


Figure 4: Average Score vs Sentiment of title and extracted comments

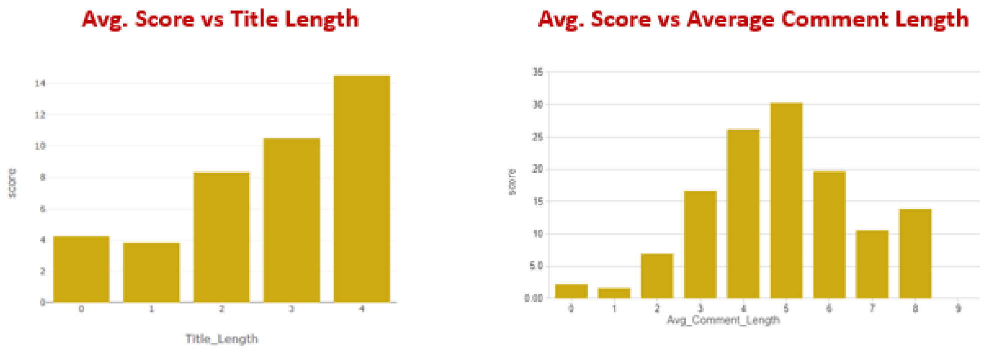


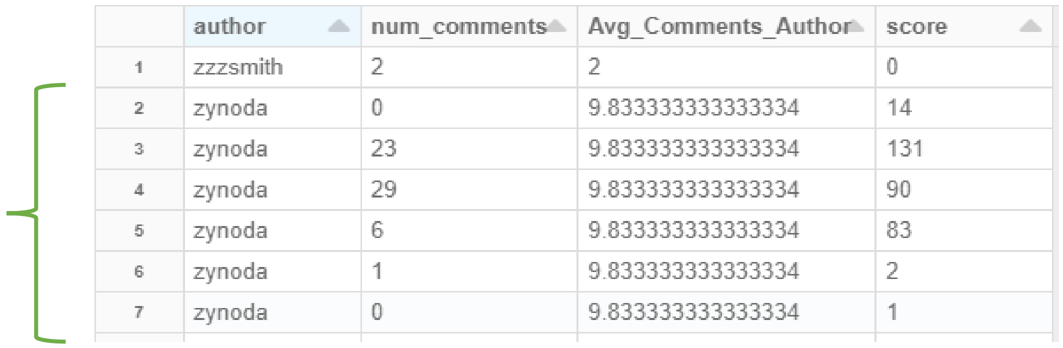
Figure 5: Average Score vs length of titles and average comment length

WINDOW FEATURES

I applied window over two categories author of the post and subreddit id. For each category I used three numerical columns to compute the average value for each author or subreddit id. This generated 6 new innovative features. The three numerical columns used were- number of comments, No-follow Boolean and Is Cross-Postable Boolean. The average for number of comments and respective Booleans were computed for each author and subreddit id.

The figure below explains the window function methodology visually. We can see on an average that if an author has higher average number of comments then most of his posts tend to be popular having high scores.

For instance, if we look at author Zynoda who has average number of comments as 9.8334 which is pretty high, then most of his posts tend to have higher scores as well.



	author	num_comments	Avg_Comments_Author	score
1	zzzsmith	2	2	0
2	zynoda	0	9.833333333333334	14
3	zynoda	23	9.833333333333334	131
4	zynoda	29	9.833333333333334	90
5	zynoda	6	9.833333333333334	83
6	zynoda	1	9.833333333333334	2
7	zynoda	0	9.833333333333334	1

Figure 6: Schematic of Window operation for 1 author

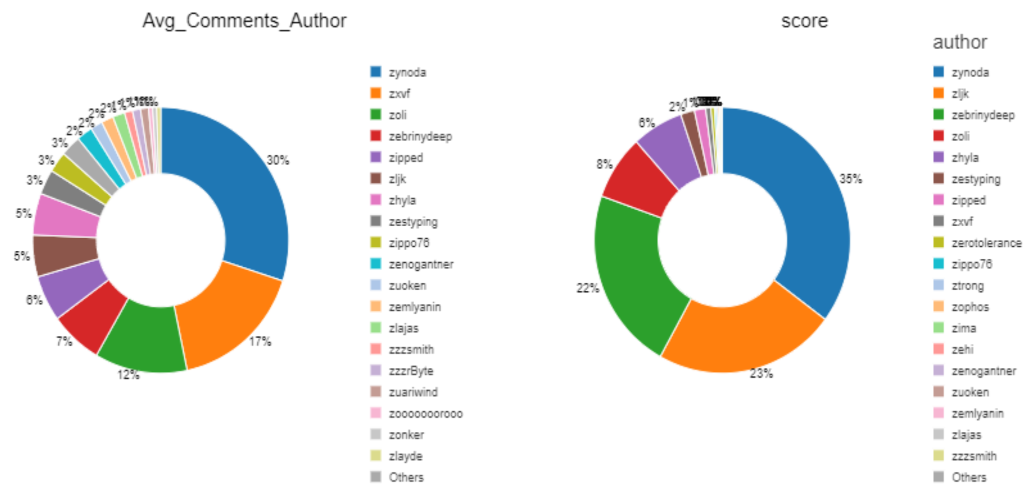


Figure 7: Distribution of Average comments per author and average score per author in form of pie plot

BERT EMBEDDINGS

Along with the Word Embeddings for the text data derived by my group. **I decided to extract the BERT Embeddings**. The Spark- NLP (John Snow Labs) package was used to extract the BERT Embedding of title column. BERT stands for **Bidirectional Encoder Representations from Transformers**. The BERT is pretrained on entire **Wikipedia(2500 million words) and Book Corpus (800 words)** in a bidirectional manner. This training in a deeply bidirectional training enables BERT to understand the text and learn the information from left to right and right to left and this makes possible for it to understand the **semantic and syntactic** meaning of the textual data [1].

For the project, **BERT base** was used to extract embeddings. It consists of **12 layers (transformer blocks), 12 attention heads, and 110 million parameters.**

Methodology Used

- Initially the embedding vector of dimension (768,1) was extracted for each word present in the title column
- Further, to get a single vector for each row the average of all vectors in a text was taken to generate a document vector corresponding to each row.
- Finally, this vector was converted to a dense vector format which can used by Spark-ML.

UTC HOUR

From the below figure, we can conclude that if a post is posted during the late night or evening it tends to have higher score on an average as compared to the posts posted early mornings or afternoon.

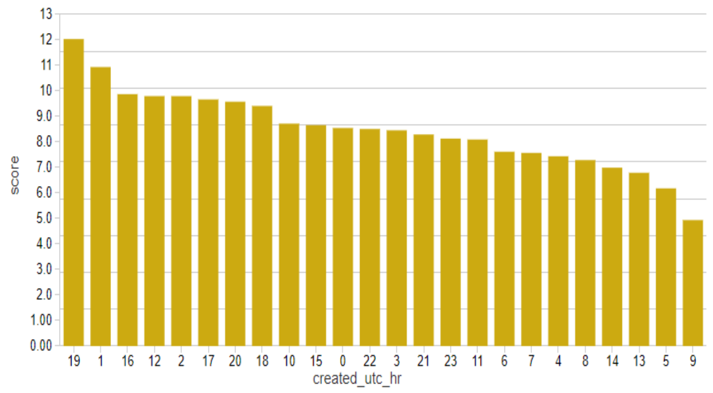


Figure 7: Average score vs time of reddit post in 24:00 hr format

SECTION 6: MODELING AND TUNING

Three models were implemented to train the dataset. The first model was Random Forest trained using all the features and Word Embeddings. The second model was Gradient Boosting Using all features and Word Embeddings. The third model was Random forest which only used the BERT Embeddings extracted for the title column. All the features obtained from previous steps passed through a pipeline **and chi-square selector** was used to **select 45 top features** and then these were merged with the textual embeddings and fed as an input to the machine learning models

I implemented the third model Random Forest which uses only the BERT Embeddings for the title column. Due to the compatibility issues of Spark-ML and Spark NLP, I was not able to merge all the features extracted using Spark ML with the BERT Embeddings obtained using Spark NLP into a single Vector. Hence, third model was only implemented using BERT embeddings. Since the vector size of BERT embeddings was 768 for each row, I was not able to test the performance of BERT Model on hold out test set due to cluster limitations in term of computational time and processing power. Hence, the BERT model was trained and tuned by splitting the initial training dataset into 70:30 ratio.

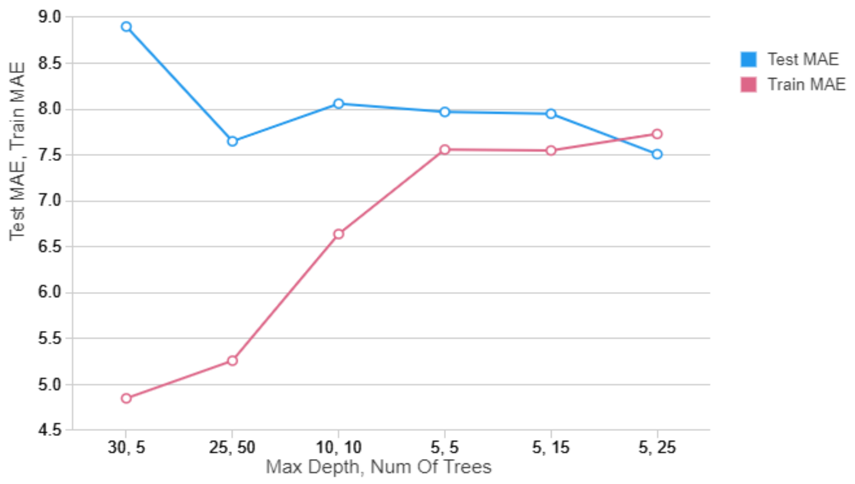


Figure 8: Hyperparameter Tuning for Random Forest Model with BERT Embedding

The training was done manually for 6 different combinations of number of trees and max depth for random forest model. The other two models were trained and tuned by other people in my group. We also tuned other two models for different combinations of hyperparameters using manual search. The figures given below show the hyperparameter tuning results for the all three models.

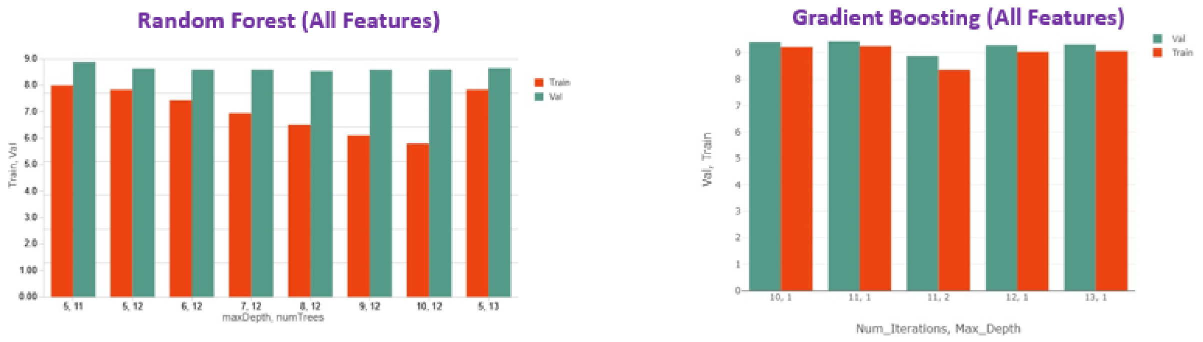
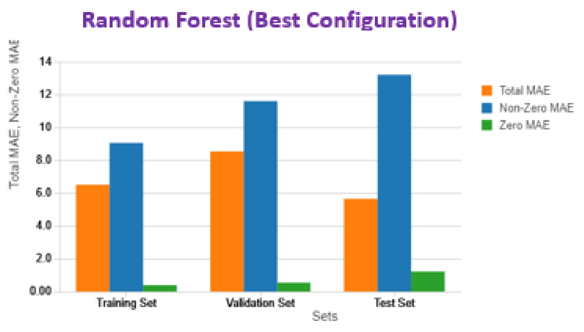


Figure 9: Hyperparameter Tuning for Random Forest and Gradient Boosting (All Features)



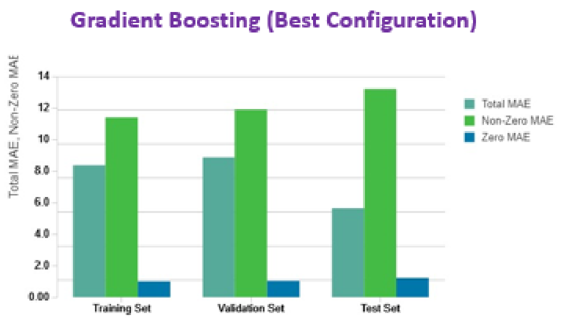


Figure 10: Best Tuned configuration for RF and GB (All Features)

Post the hyperparameter tuning, we observed that best hyperparameter for Gradient Boosting are Number of iterations=11 and maximum depth =3. The best hyperparameters for random forest model with all features were number of trees =12 and maximum depth =8. The Figure 10 shows the performance of both the models with the best hyperparameters on training validation and test set.

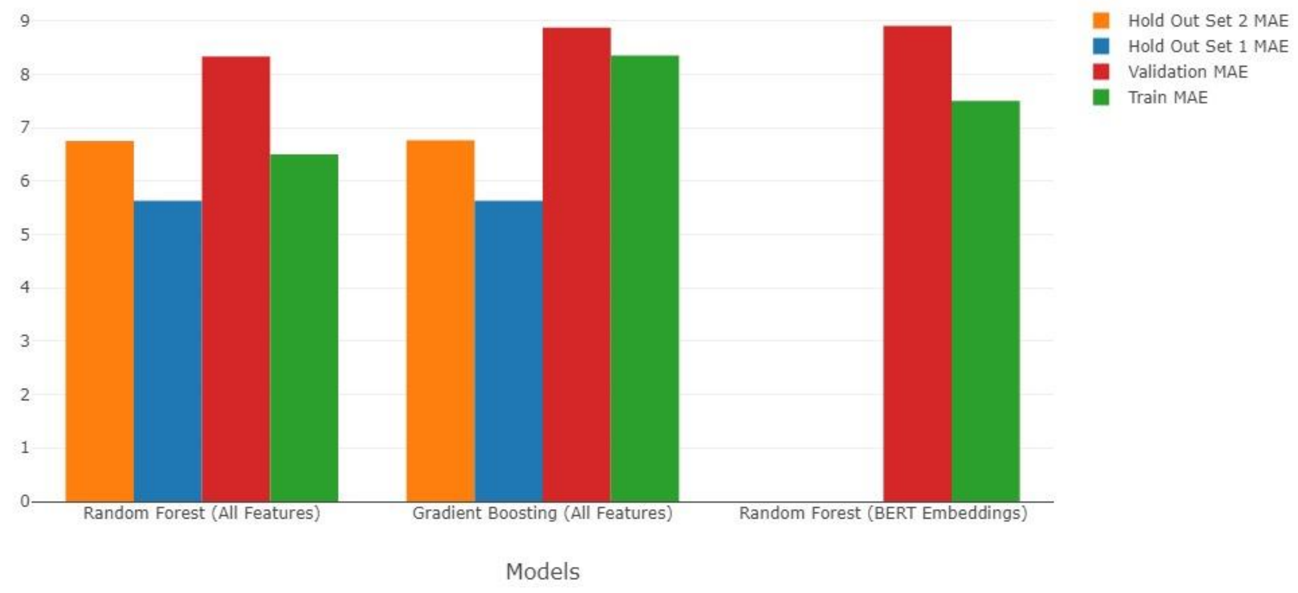


Figure 11: Performance on Train, Validation, Test Set and Hold Out Set

SECTION 7: CONCLUSION

The best performing model was the Gradient Boosting model with all the features and word embeddings. The best performance was observed for number of iterations =11 and maximum depth= 3. We might expect to get better results if we increase the maximum depth to 5, but due to the cluster limitations we could not try that. The best performing model gave a **Mean absolute error of 5.50 on hold out set 1 and 6.27 on hold out set 2.**

SECTION 8: FUTURE WORK

- Our first aim would be to merge the BERT Embeddings with the existing innovative features. This can improve the result of the models significantly.
- We could also use some auto-regressive models like **Xlnet Embeddings** to calculate the sentiment of the text in order to **capture the negations** into account.
- We would also like to extract few other features from textual data using **POS tagging, Named Entity Recognition and dependency parsing** to understand the text data grammatically and to see the effect of entities present in the text.

SECTION 9: REFERENCES

1- Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

SECTION 10: APPENDIX

Contributions

At a high level, the following should be noted: (i) each member of Team 1 completed the tasks that were assigned to them in the meetings and (ii) each member contributed in each step as the approach was discussed and agreed upon as a Team. However, the Lead represents the primary personals in charge of the step and as such contributed heavily towards that step. Please see the contribution breakdown below:

Steps	Contribution
Research	All, Lead: Faraz
Exploratory Data Analysis / Data Cleaning / Visualizations	All, Lead: Faraz
Data Techniques (Skewness / Outlier Detection / Undersampling)	All, Lead: Fahad and Saket
Feature Selection	All, Lead: Nileena and Obaid
Feature Engineering	All, Lead: Fahad, Nileena and Saket
Modelling / Tuning / Testing	All, Lead: Fahad and Saket
Code Merging / Pipeline	Fahad and Obaid