IMDB Sentiment Analysis

KIRAN BELGAL, LEIGH ANNE LEMOINE, TYLER NGUYEN

IMDB is an online platform for films, TV series, podcasts, videogames, and more, where users are free to express their opinions. In this study, we attempt to predict whether a given review has a positive or negative sentiment through analyzing its contents. We preprocessed 50,000 IMDB reviews and then performed binary classification on the numerically vectorized reviews using the Logistic Regression, KNN, LDA, QDA, and Random Forest machine learning models.

1. INTRODUCTION

Thinking of watching a movie tonight? Consider looking for a review on IMDB, an online database of information about TV series, movies, video games, and more. As a leading review site in the film industry since 1990, IMDB has amassed millions of ratings and reviews from users all across the world. The website allows for users to input a rating (with 1 star being the lowest and 10 stars the highest) along with a written review of the content. In our project, we will conduct a sentiment analysis of an IMDB dataset consisting of 50,000 entries of reviews on the IMDB site. Our goal is to use statistical methodology to predict the overall binary sentiment of each review using only the words and phrases in each data entry.

2. PREPROCESSING

In order to accurately predict the sentiment of the IMDB reviews, we first had to preprocess our dataset. Upon loading in our dataset of IMDB reviews and sentiments, we conducted some elementary, exploratory data analysis to understand the size and balance of our dataset. Then, we converted our textual data into a form that is suitable for statistical analysis using TF-IDF. Finally, we utilized Principal Component Analysis (PCA) to reduce the dimension of our data, improving computational cost and efficiency. Each one of the steps we took will be broken down in detail in the following sections.

2.1. Data Cleaning

Before conducting any sophisticated analysis, we first cleaned the data to prepare for the later stages of experimentation and analysis. The general process we took to clean the data included:

1. Removing html line breaks.
2. Converting all characters to lowercase.
3. Removing punctuation, special characters, numbers.
4. Removing stop words using tokenization.

By taking these steps to clean the data, we avoid muddling the analysis with neutral, meaningless words such as "a" or "the" in order to focus on keywords that might better predict sentiment. Now that the data is clean, the exploratory data analysis and visualization can begin.

2.2. Descriptive Statistics

In order to get a better grasp of the dataset, we constructed plots to serve as visual representations of the data.

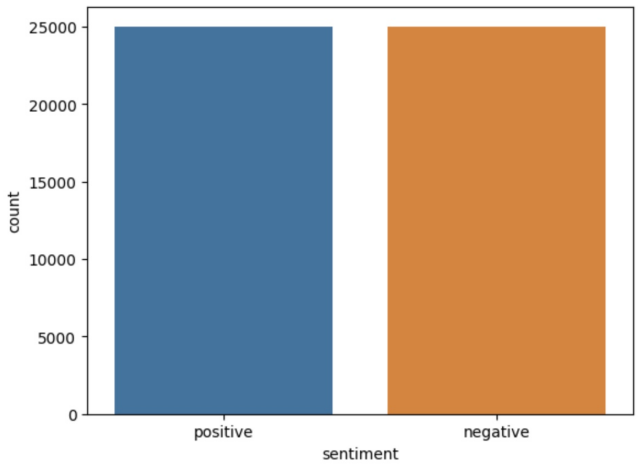

Fig. 1. A bar chart quantifying the amount of positive and negative reviews.

Figure 1, shown above, displays the sentiment makeup of the dataset. In this particular IMDB dataset, we have a perfect balance of positive and negative reviews. This means we do not have to perform any techniques such as SMOTE to balance the dataset on our own.

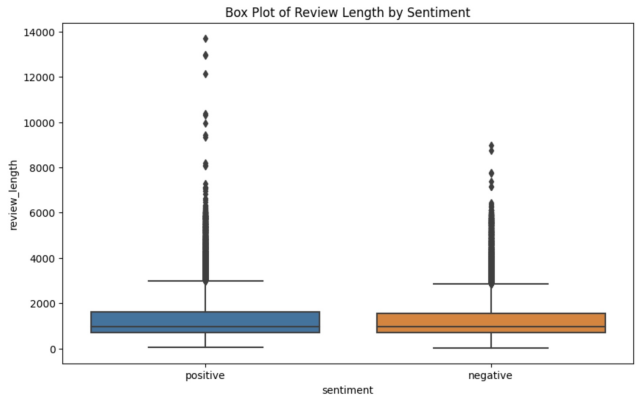

Fig. 2. Side-by-side boxplots displaying the length of positive and negative reviews. Evidently, the negative and positive reviews have similar distributions of lengths. However, the positive review distribution contains more extreme outliers on the higher end of the length range.

Fig. 3. The most common words in negative reviews.

Fig. 4. The most common words in positive reviews.

Comparing the two figures above, we can see that there is not much difference between the two word clouds (aside from a few words such as "bad" versus "good"), illustrating the need for a more in-depth look at and analysis of the reviews.

2.3. TF-IDF

Because our review data is textual, we had to transform our data from text to numbers. During data cleaning, we had already removed line breaks, punctuation, special characters, numbers, and stop words. However, to convert our textual review data into a format suitable for analysis, we used TF-IDF, or Term Frequency-Inverse Document Frequency. Here is a breakdown of the method:

1. Term Frequency (TF): TF measures the frequency of each term within a review, calculated as the ratio of term occurrences to the total number of terms in the review.
2. Inverse Document Frequency (IDF): IDF quantifies the importance of a term across the entire set of reviews, computed as the logarithm of the ratio of the total reviews to those that contain the term.
3. TF-IDF Score: The TF-IDF score for a term in a document is the product of its TF and IDF scores, creating a numerical representation of the textual data. This results in a high-dimensional vector space, which will be discussed further in the following section.

2.4. Principal Component Analysis (PCA)

After performing TF-IDF on our reviews, the resulting vectorized data had extremely high dimensionality at 162,401 total features. Since high-dimensional data is highly correlated, computationally expensive to work with, and can hinder the performance of some models, we performed Principal Component Analysis, or PCA, to reduce the dimensionality of our data. By doing so, we reduce the dimensionality of our data while retaining the most important information and minimizing the loss of information. PCA accomplishes this by transforming the original features into a new set of uncorrelated variables called principal components. These components are ordered by the amount of variance they explain in the data, allowing us to keep the most significant information while discarding less critical aspects.

In our case, PCA helps us overcome the above challenges associated with the high dimensionality of the TF-IDF vectorized data. However, we first had to determine how many components to use. We did so using a Scree plot to find the optimal number of components.

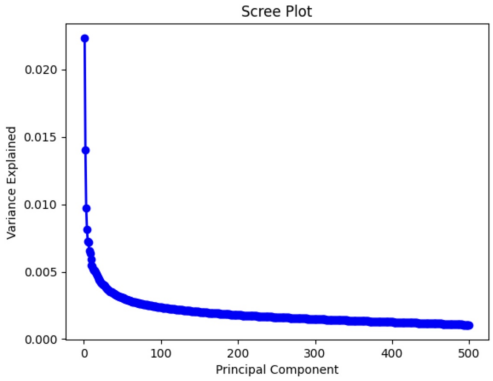

Fig. 5. Explained variance for different amounts of PCA components.

Scree plots display the number of principal components on the x-axis and the corresponding explained variance on the y-axis. Based on the curve above, we saw that there was not a significant change in variance after 50 components, so we finally performed PCA on our vectorized dataset with 50 components.

3. EXPERIMENT

In our pursuit of identifying the optimal model for prediction, we employed four supervised learning algorithms.

3.1. Logistic Regression

Logistic regression analysis is a statistical technique to evaluate the relationship between various predictor variables (either

categorical or continuous) and an outcome which is binary (dichotomous). A common issue with modelling is over fitting in which a model closely follows the training data set rather than following the underlying pattern. Due to this, over fitted models fail to translate to testing data sets and have no real practicality. To lessen the odds of over-fitting, we will apply L2 regularization to avoid extreme coefficients.

In tuning our model, we focused on a single hyper-parameter, namely the value of C, controlling the weight assigned to the regularization penalty. Higher C values lead to a closer fit to the training data, while lower values prioritize the penalty term during coefficient computation. Employing Cross Validation, we determined that C=5 produced the optimal choice. The test set produced the following results:

Table 1. Logistic Regression Model - Performance

	Precision	Recall	F1 Score	Support
negative	0.85708	0.82485	0.84066	7525
positive	0.83011	0.86154	0.84553	7475
accuracy			0.84313	15000
macro average	0.84360	0.84319	0.84310	15000
weighted average	0.84364	0.84313	0.84309	15000

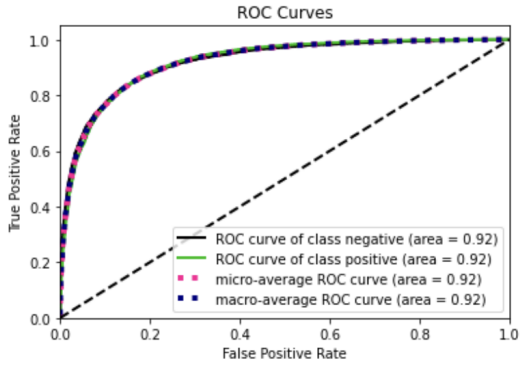

Fig. 6. An ROC curve of our Logistic Regression model with optimal C.

The results show that the model performs equally strong on negative reviews and positive reviews with both having f1-scores of around 84 percent. In addition, the ROC curve is quite good as the area under the curve (AUC) is around 0.92. An ROC curve plots the True Positive Rate vs. False Poistive Rate at different classification thresholds. Theoretically, the upper left point where the TPR and FPR both equal 1 indicates the golden standard. Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives. Typically, an AUC value of 0.9 - 0.1 is excellent, 0.8-0.9 is very good, 0.7-0.8 is good, 0.6-0.7 is satisfactory, and 0.5-0.6 is unsatisfactory.

3.2. K-Nearest Neighbor

K-Nearest Neighbors is simple approach to supervised machine learning that creates a non-linear boundary by measuring zones in which one response category dominates another. It is a non-parametric model and there are no assumptions being made about the underlying distribution of the data. Furthermore, it generally performs well on large data sets which is advantageous for our research since we have 50,000 samples in our data set.

The K in K-Nearest Neighbors is an non-even hyperparameter that determines how many of the neighbors will be considered when making a zone for each category. A small K can lead to over sensitivity to noise, while a large K may lead to an over-biased model. It is essential to choose the optimal K that strikes the balance between variance and minimizing bias.

We find the optimal K using 5-Fold Cross Validation:

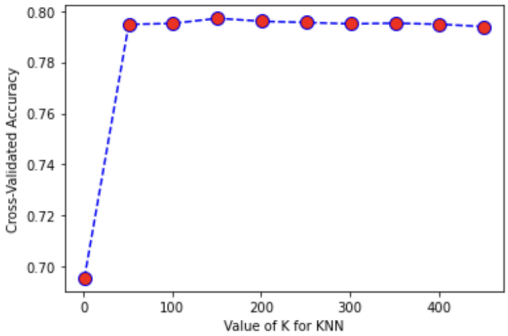

Fig. 7. Chart labelling each odd value of K and its corresponding CV accuracy from 1 - 501

The chart illustrates that small value's of K perform much worse than values greater than 50. Interestingly, the drop off as K becomes abnormally large seems much more shallow than the drop off as K becomes increasingly small. Nonetheless, the optimal value of K based on the chart is 151 as this maximizes our cross validation accuracy of 79.72 percent on the training set. Utilizing K = 151 we can produce our results on the testing set:

Table 2. KNN Model - Performance (Optimal K)

	Precision	Recall	F1 Score	Support
negative	0.81702	0.75894	0.78691	7525
positive	0.77353	0.82890	0.80026	7475
accuracy			0.79380	15000
macro average	0.79528	0.79392	0.79358	15000
weighted average	0.79535	0.79380	0.79356	15000

The results indicate that KNN is stronger at predicting positive reviews than negative reviews. The overall accuracy of 79.380 percent is quite good, but there is a significant drop off between the accuracy of KNN and the accuracy of Logistic Regression found previously. Due to this, and the extreme computation cost of running a high dimensional KNN model, other models may be preferred in this instance. The results also indicate minimal overfitting in our model due to tuning, as the testing accuracy was around 79.38 percent, a statistic very similar to the 79.72 percent found on the training set.

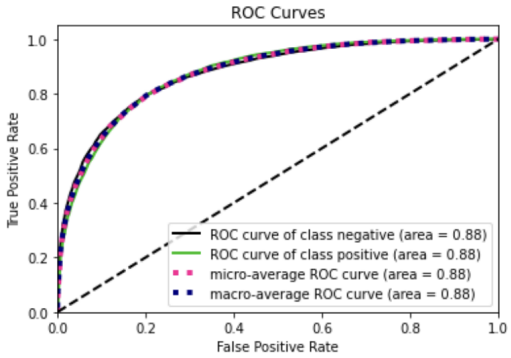

Fig. 8. An ROC curve showing the performance of our KNN model with optimal K

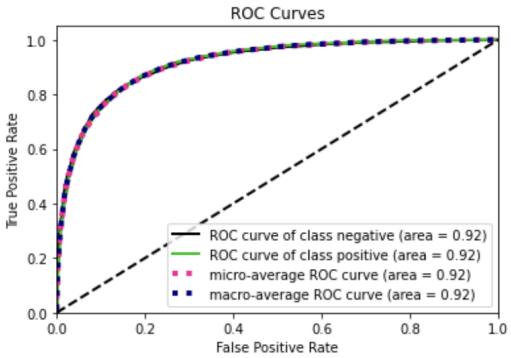

Fig. 9. The ROC curve of our LDA model.

3.3. Linear and Quadratic Discriminant Analysis

Linear Discriminant Analysis and Quadratic Discriminant Analysis differ from the previous models in two major ways.

First, LDA and QDA are generative models, meaning that they capture the joint probability of their response variable and its predictors. This differs from Logistic Regression and KNN because those are discriminate models that capture the conditional probability of response given the predictors. While generative models are more flexible than discriminate models, they are more sensitive to outlier.

The second key difference is that, LDA and QDA make the assumption that the data comes from multivariate normal distribution. Similarly to KNN, LDA and QDA are both non-parametric models which is a main cause for their aforementioned flexibility. LDA is a special case of QDA because its boundary is strictly linear while QDA often has a quadratic boundary and differing covariance matrices for both classes.

After fitting our models and using five fold cross validation, we find that LDA has a training accuracy of 83.79 percent while QDA has a training accuracy of 78.18 percent.

We ran the models on our test set:

Table 3. LDA Model - Performance

	Precision	Recall	F1 Score	Support
negative	0.86254	0.80385	0.83216	7525
positive	0.81520	0.87104	0.84219	7475
accuracy			0.83733	15000
macro average	0.83887	0.83745	0.83718	15000
weighted average	0.83895	0.83733	0.83716	15000

The results show a clear performance difference as LDA's testing accuracy of 83.733 percent vastly outperformed QDA's testing accuracy of 78.167 percent. Since both LDA and Logistic Regression have linear decision boundaries, it is of no surprise that with Logistic Regression performing well, LDA followed suit. Additionally, our testing results were very similar to the training results, demonstrating little overfitting. Both ROC curves also show that our models were quite good, with LDA covering 92 percent of the area and QDA covering 85 percent.

Table 4. QDA Model - Performance

	Precision	Recall	F1 Score	Support
negative	0.80125	0.75110	0.77536	7525
positive	0.76428	0.81244	0.78763	7475
accuracy			0.78167	15000
macro average	0.78277	0.78177	0.78149	15000
weighted average	0.78283	0.78167	0.78147	15000

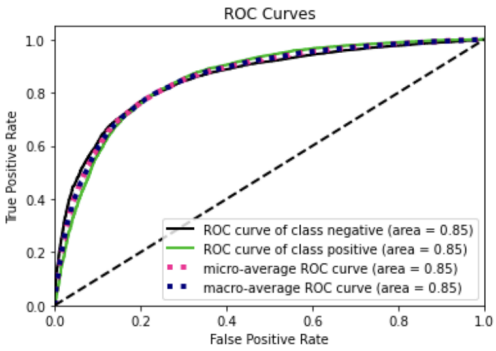

Fig. 10. The ROC curve of our QDA model.

3.4. Random Forests

Random Forests are the last model we used to analyze the IMDB data. Random forests are built from decision 'trees' hence the name. These decision trees are built from nodes that split the data two ways into leaves repeatedly until samples are appropriately classified. Each split depends on the value of only one variable and the quality of that split is measured by its impurity. The impurity is a measure of how 'pure' each leaf of the split is in the sense of having only one class within the leaf.

So each decision tree is made up of nodes with each split leading to leaves below it until the data is sorted enough. This 'enough' is measured using a stop-splitting rule which tells us when the improvement from adding a new leaf isn't worth the corresponding increase in bias and processing time.

A random forest is composed of n_{tree} decision trees. These decision trees are created by first drawing n_{tree} bootstrap samples. Then we build a tree by creating a subset of the p variables and then using the variable with the best result to split the node.

We repeat this until halted by our stop-splitting rule.

Once we have grown n_{tree} number of trees we classify our data by inputting each sample into every decision tree and then taking the majority vote of the trees as the sample's final classification. Since each decision tree in the forest was trained using a different set of predictors the resulting forest will have high predictive power.

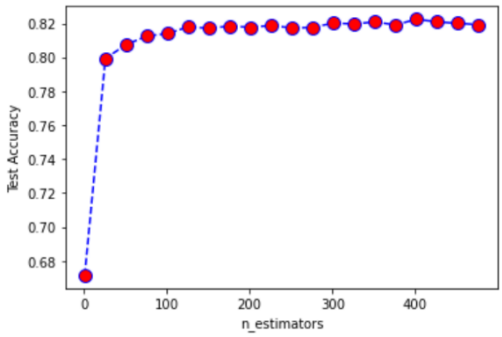

Fig. 11. A graph showing the increase in accuracy corresponding to a different number of trees (represented by $n_{estimators}$).

The question of how large n_{tree} should be is primarily determined by available computational power. As you can see from the graph above, at first the performance of the model is greatly improved by each additional tree but the addition rapidly stagnates. This means that there is not much difference between 100 trees and 1000 trees. For our analysis, we chose to use 100 trees. The model was trained with the 50 principal components from our PCA as predictors and then the sentiment of the review (positive or negative) as the response variable.

Random forests come with several hyperparameters (tree depth, number of trees, and feature selection method). It is vital to tune these parameters to achieve optimal values. There are several methods for testing different combinations of these parameters including grid search or random search, but generally, the goal of tuning is to improve model accuracy and its ability to model new data.

We used cross-validation via grid search to find other best hyperparameters and found that in addition to $n\_estimators = 100$, the $max\_features = "auto"$, $max\_depth = 100$, $minsamples\_split = 12$, $minsamples\_leaf = 4$, $bootstrap = True$ were the best hyperparameters given by cross-validation. The classification report of this final model and its ROC curve are displayed below

Table 5. Random Forest - Performance (Best Hyperparameters)

	Precision	Recall	F1 Score	Support
negative	0.82395	0.80545	0.81460	7525
positive	0.80848	0.82676	0.81751	7475
accuracy			0.81607	15000
macro average	0.81622	0.81610	0.81606	15000
weighted average	0.81624	0.81607	0.81605	15000

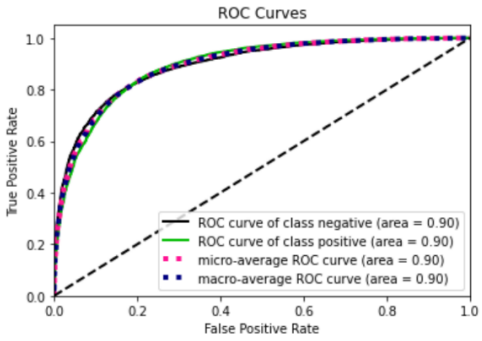

Fig. 12. The ROC curve of our random forest model with best hyperparameters.

We can see that the testing accuracy of 81.607 percent and ROC curve, which covers about 90% of the area, are quite good. F1-scores are both fairly high and there is little variation in score across positive and negative sentiments. The remaining error is likely due to difficulties classifying or possibly having suboptimal hyperparameters.

4. CONCLUSION AND SUMMARY

In the following sections, we will give a recap of our methods and analysis, along with some remarks on our results.

4.1. Results and Analysis

The first, perhaps most important, step of comparing model efficacy is looking at overall accuracy percentages. The table below displays accuracy for each model we used.

Table 6. Model Accuracies

Model	Accuracy (%)
Logistic Regression	84.31
KNN	79.38
LDA	83.73
QDA	78.17
Random Forest	81.61

Evidently, logistic regression had the highest accuracy, followed closely by LDA. QDA performed the worst. Intuitively, this makes sense since LDA and logistic regression are (computationally) the most similar models out of the methods we used, and their similar accuracy reinforces their high strength in predicting sentiment for our dataset. Additionally, our data is not thought to be normally distributed, so it makes sense that logistic regression outperformed LDA.

At the bottom of the accuracy list are KNN and QDA at a respective 79% and 78%. Once again, it makes sense that KNN and QDA may have similarly low accuracy scores due to their tendency to overfit. In other words, KNN and QDA may have been too flexible with the training data, capturing noise and specific patterns that don't generalize well to new, unseen data. The lower accuracy scores for KNN and QDA suggest that their decision boundaries might be overly tailored to the training dataset's oddities.

It's worth noting that KNN and QDA could possibly benefit from methods such as regularization, which can reduce variance and help overfitting at the cost of introducing some bias into the model fitting process. However, due to the strong performance of logistic regression and LDA, we decided it was not necessary to attempt to address potential overfitting problems with QDA and KNN.

The random forest method ended up being middle of the road as far as accuracy goes at 81.61%. While it did demonstrate some effectiveness due to its superior accuracy rate over QDA and KNN, its computational cost was considerably higher than logistic regression, making it a worse, less efficient choice for a final model.

To go further in quantifying the performance of our various models, we also generated recall, precision, and F1 scores for each model for the positive and negative classifications.

Table 7. Model Comparison - "Negative" Classification

Model	Precision	Recall	F1 Score
Logistic	0.8571	0.8249	0.8407
KNN	0.7735	0.8289	0.8003
LDA	0.8625	0.8039	0.8322
QDA	0.8013	0.7511	0.7754
Random Forest	0.8240	0.8055	0.8146

The table above displays the results for the negative side of precision, recall and F1 scores. Unsurprisingly, logistic regression and LDA come in as the best performers overall, and especially for the precision column. Interestingly enough, LDA has a slightly higher precision score than logistic regression, meaning that when it predicts negative class, it is slightly more accurate than logistic regression. However, the difference in this case is so small that it is negligible.

Once again, QDA is the worst performing method, being the only method with recall and F1 scores dipping well below the 0.8 level. This means QDA frequently has false positives, and the model struggles to balance precision and recall for the negative class.

Table 8. Model Comparison - "Positive" Classification

Model	Precision	Recall	F1 Score
Logistic	0.8301	0.8615	0.8455
KNN	0.8170	0.7589	0.7869
LDA	0.8152	0.8710	0.8422
QDA	0.7643	0.8124	0.7876
Random Forest	0.8085	0.8268	0.8175

The positive table above tells the same tale; logistic regression and LDA are high performers, while QDA and KNN sit at the bottom and random forest takes the middle ground. Once again, this suggests that LDA and logistic regression correctly predict positive sentiments the strong majority of the time, while QDA and KNN are a level below.

Overall, the positive and negative tables simply reinforce the accuracy statistics for each model. The optimal prediction method seems to be logistic regression, closely followed by LDA, while KNN and QDA are generally methods to avoid if aiming for a high accuracy. While random forest had a decent performance, its computational cost makes it a less desirable technique to use for predicting sentiment.

4.2. Final Comments and Reflection

Across the board, our models were very effective in predicting the sentiment of IMDB reviews. Their accuracies were significantly above 50% (which would be the approximate percentage corresponding to randomly guessing "positive" or "negative"). Although our models were robust, there are a couple concerns that might have affected our results negatively.

Correlated reviews: The sentiments for a singular movie (or film series) likely has some correlation with each other. Because the reviews are publicly available, any potential, future reviewer may be biased by the reviews currently on the site. For instance, a movie with extremely negative reviews may cause future reviewers to have a more negative sentiment than they otherwise would have. Although this introduces some correlation between data points and we did not take it into account, the effect is likely minimal and our results are not threatened by it.

Additional models: We tested a total of five methods (KNN, QDA, LDA, logistic regression, and random forest) but there are certainly more available techniques out there for prediction. With these models alone, we managed to achieve a high accuracy rating and did not feel that it is necessary to use even more models, but it is possible that there is a better technique out there.

Model complexity: The models we used are relatively standard and void of any highly sophisticated tools. There are of course many more options that may aid prediction accuracy or improve the performance of a specific model. As mentioned before, QDA's low performance could have been partially due to overfitting and might have benefitted from a technique like regularization. However, for our needs, logistic regression (where we did utilize regularization) and LDA were more than sufficient and we did not feel the need to introduce further methodology.

5. REFERENCES

1. *Elbow Method in Supervised Learning: Optimal k-value*. Moussa Doumbia. https://medium.com/@moussadoumbia_90919/elbow-method-in-supervised-learning-optimal-k-value-99d425f229e7

2. *Logistic Regression and Regularization: Avoiding Overfitting and Improving Generalization*. Rith Pansanga.c https://medium.com/@rithpansanga/logistic-regression-and-regularization-avoiding-overfitting-and-improving-generalization-e9afdcddd09d

3. *A Look at Precision, Recall, and F1 Score*. Towards Data Science. https://towardsdatascience.com/a-look-at-precision-recall-and-f1-score-36b5fd0dd3ec

4. *Beginner's Guide to Random Forest Hyperparameter Tuning*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/03/beginners-guide-random-forest-hyperparameter-tuning/