STATS 202: Data Mining and Analysis
Final Project Write-Up
By Jeremy Bischoff

Introduction

The goal of this project is to develop and evaluate predictive models to determine the relevance of URLs for search engine queries, a critical task in improving the effectiveness of search engines. Search engines typically rely on hundreds of signals to assess the relevance of a document and present the most suitable results to users. In this project, I was provided with a training dataset consisting of 80,046 observations, each containing 10 different signals that could potentially be used to predict whether a URL is relevant to a specific query. Additionally, a test dataset with 30,001 observations was provided for making predictions of this relevance. The task is to analyze these signals, build and tune machine learning models, and generate predictions on the test set, with submissions evaluated on accuracy via Kaggle.

Throughout the project, I engaged in several key steps of the data mining process: exploring and preprocessing the data, identifying and handling outliers, engineering features, and selecting and tuning various machine learning models. I experimented with multiple popular classification algorithms, including logistic regression, Linear Discriminant Analysis (LDA), a neural network MLP classifier, and a gradient boosting classifier. After thorough evaluation of these models through a 5-fold cross-validation technique, I identified the gradient boosting classifier to be the best-performing model of the four, with the highest average accuracy across all the folds. This process involved thoroughly exploring the data to perform feature engineering, optimizing the model's effectiveness by adding or removing specific attributes to address issues such as multicollinearity, and fine-tuning hyperparameters to balance the bias-variance tradeoff, ensuring that the model was neither too simple nor overly complex. Additionally, I applied techniques such as data standardization and k-fold cross-validation, all with the goal of building a robust and accurate model for predicting URL relevance in the test set.

Initial Look at the Data

I began by examining the structure and composition of both the training and test datasets. The training dataset consists of 80,046 observations with 14 attributes, while the test dataset consists of 30,001 observations and 13 attributes. The missing attribute in the test dataset is the ‘relevance’ column, which we aim to predict using our model. Each dataset contains various signals that may influence the relevance of a URL for a given search query. The attributes include identifiers such as query_id, url_id, and id (a combined version of the query and URL IDs), along with features like query_length, is_homepage, and several numerical signals (sig1 through sig8). The training dataset uniquely contains the relevance column, our target variable,

indicating whether a URL is relevant (1) or not (0) for the associated query. Below is a look at a snippet of the training data.

Training Data Head:

	query_id	url_id	query_length	is_homepage	sig1	sig2	sig3	sig4	sig5	sig6	sig7	sig8	relevance	id
0	4631	28624	2	1	0.09	0.15	1288	352	376	13	0.46	0.35	0	4631.28624
1	4631	28625	2	1	0.20	0.35	4662	337	666	28	0.43	0.27	1	4631.28625
2	4631	28626	2	1	0.36	0.49	1121	385	270	15	0.34	0.20	1	4631.28626
3	4631	28627	2	1	0.21	0.45	2925	478	640	14	0.44	0.33	1	4631.28627
4	4631	28628	2	1	0.25	0.42	1328	429	412	27	0.40	0.57	1	4631.28628

Positives, Negatives, and Missing Values

To begin my analysis, I performed an initial exploration of the dataset to assess the distribution of relevant and non-relevant URLs, as well as to check for any missing values. The dataset consists of 80,046 observations, with 45,059 classified as non-relevant (negatives), making up approximately 56.3%, and 34,987 classified as relevant (positives), accounting for about 43.7%. This distribution suggests a relatively balanced dataset, though with a slight skew towards non-relevant URLs, which is an important consideration for model training as it could influence the performance of classification algorithms. Additionally, I checked for missing values across both the training and test datasets. Fortunately, no missing values were detected in either dataset, ensuring that data imputation or handling strategies were unnecessary. This clean dataset provided a solid foundation for the next steps in the data mining process and allowed me to move forward and delve deeper into the data.

Box Plot Analysis

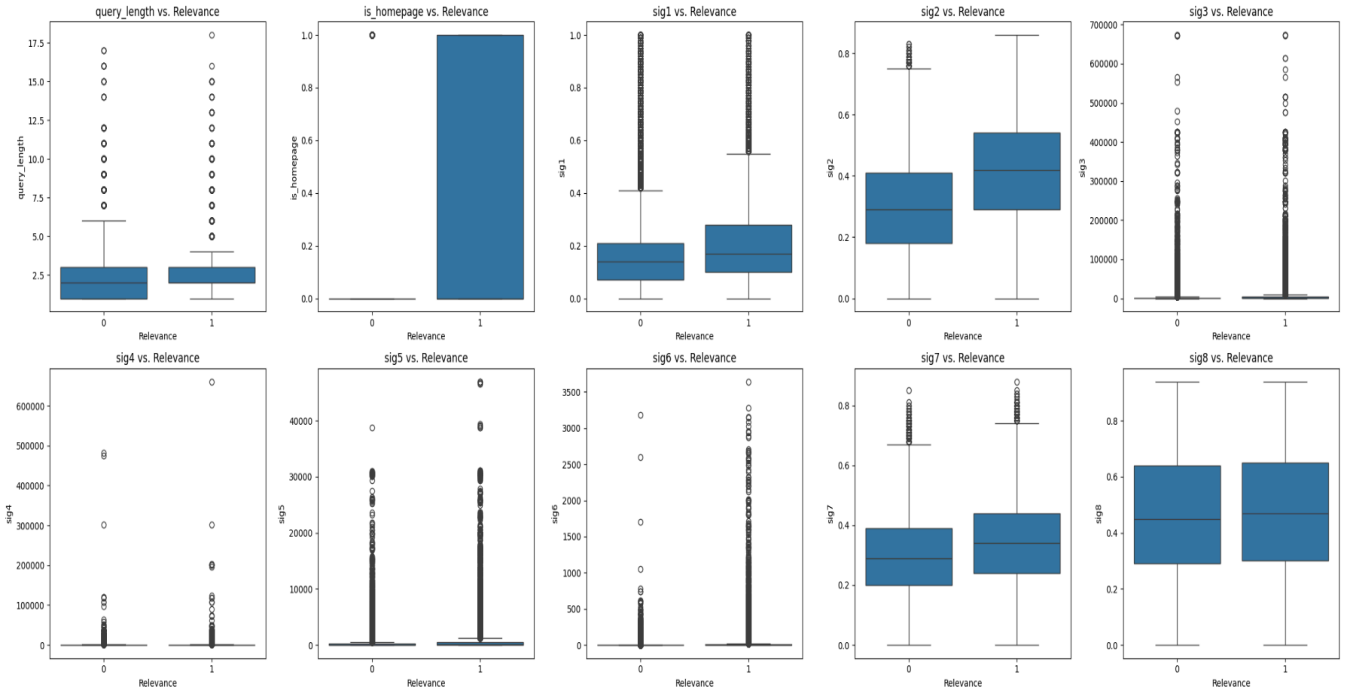
To gain deeper insights into the relationships between individual features and the target variable, relevance, I utilized box plots as a key tool to further explore the data. Box plots are particularly useful for visualizing the distribution of data and identifying differences between groups, where in this case between relevant and non-relevant URLs. By examining the median, quartiles, and potential outliers within these plots, I hoped to reveal patterns that might suggest whether certain features are strong indicators of relevance. Specifically, I was looking to see if the distributions of the features differed significantly between the two relevance classes, as this could indicate stronger predictive power. Additionally, the box plots allowed me to identify features with substantial overlap or outliers, which could affect the model's performance and might require further preprocessing or transformation.

The results of this analysis provided some valuable insights into the data, but mainly paved the way for further exploration into the attributes and their relationships. The 'query_length' feature showed a slight difference between relevant and non-relevant URLs, with

a higher median query length associated with relevance. However, the significant overlap between the distributions suggested that while ‘query_length’ could indicate relevance, it is not a strong predictor alone. On the other hand, the ‘is_homepage’ feature was revealed to be a highly discriminative attribute, with the majority of relevant URLs being identified as homepages. The lack of variability of this binary feature highlights its potential importance in the predictive model, which could serve as a key factor in predicting relevance.

Several signals in the data, including ‘sig1’, ‘sig3’, ‘sig4’, ‘sig5’, and ‘sig6’, showed significant overlap and a broad spread of values between the relevance classes, indicating limited predictive and differentiating power. These features showed similar distributions for relevant and non-relevant URLs, suggesting that they might not contribute much individually to the model's performance. To address this, I later explored applying transformations to reduce the impact of outliers and explored using interactions with other features to uncover potential predictive relationships. In contrast to these weaker features, ‘sig2’, ‘sig7’, and ‘sig8’ demonstrated more powerful differences between the relevance classes, making them potentially stronger features to be included in any model. Below are the feature vs. relevance box plots:

Feature vs. Relevance Box Plots:



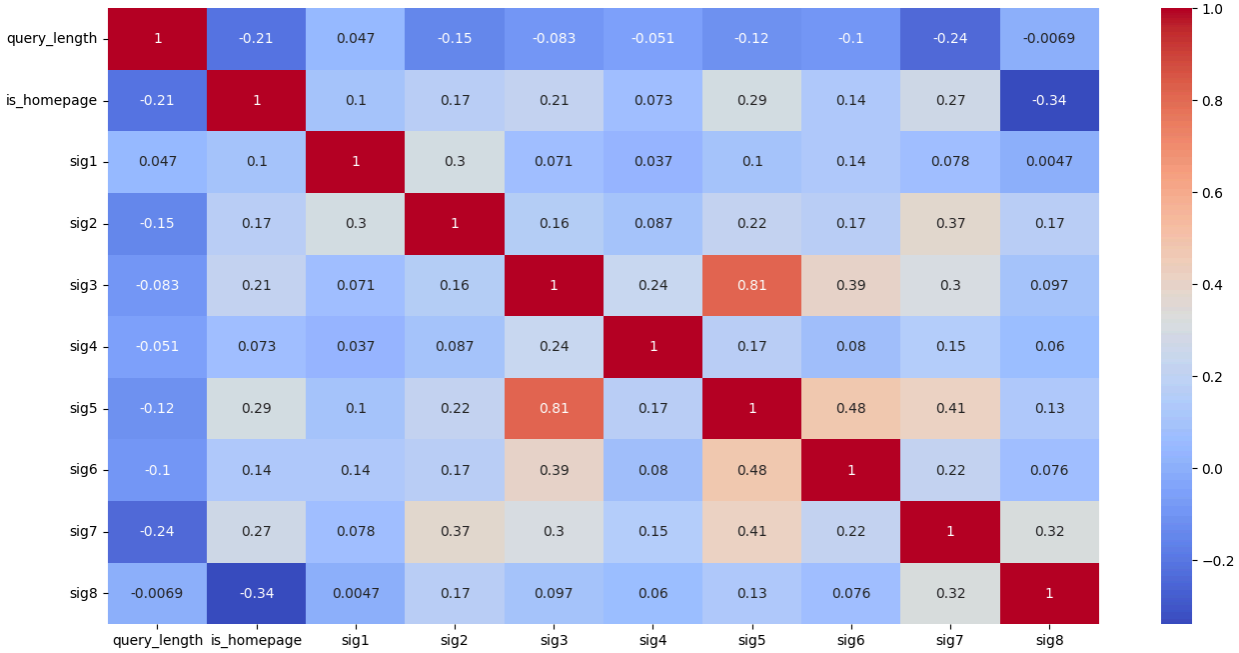
Heatmap Correlation Matrix

To further refine the feature selection and engineering process, I utilized a heatmap correlation matrix as an analytical tool. A heatmap correlation matrix visually represents the

pairwise correlations between features in the dataset, with color gradients indicating the strength and direction of these correlations. This tool is particularly useful for identifying multicollinearity, where two or more features are highly correlated with each other, which can lead to redundancy in the model and potential overfitting. By taking a look at the correlations, I hoped to reveal relationships that might necessitate the removal of certain features or the introduction of interaction terms to capture more complex, non-linear patterns. Specifically, I was looking for strong correlations that could signal redundancy and moderate correlations that might indicate the potential for creating interaction terms to enhance model performance.

The heatmap analysis revealed several important correlations that guided my feature selection decisions. Notably, ‘sig3’ showed a high correlation of 0.81 with ‘sig5’, indicating potential redundancy between these two features. To avoid issues with multicollinearity, I ultimately decided to exclude ‘sig3’ from the model after this analysis, as it likely provided redundant information already captured by ‘sig5’. Additionally, the heatmap showed several feature pairs with moderate correlations. Interactions between ‘is_homepage’ and other signals such as ‘sig5’, ‘sig7’, and ‘sig8’ were considered due to their correlations of 0.29, 0.27, and -0.34, respectively. Similarly, interactions between ‘sig5’ and ‘sig6’ (0.48), ‘sig5’ and ‘sig7’ (0.41), and ‘sig7’ and ‘sig8’ (0.32) were all identified as moderate correlations, potentially warranting the inclusion of interaction terms for these variables. After the heatmap analysis, I decided to include interaction terms for all of the moderate correlations into the dataset, adding an additional six terms to the data, but dropping the feature ‘sig3’. Below is the heatmap correlation matrix:

Heatmap Correlation Matrix:

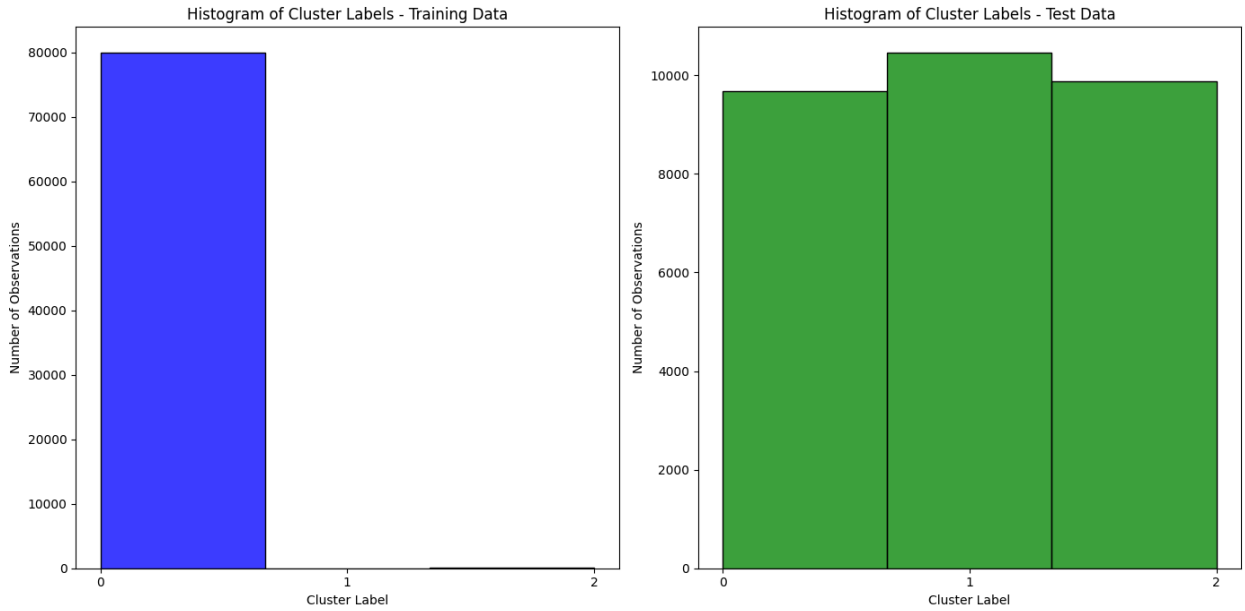


K-Means Clustering Feature

In aims of enhancing the feature set and improving the model's predictive performance, I decided to use K-Means clustering as a means to add a cluster label feature into the data. K-Means clustering is a widely used unsupervised learning technique that groups data points into clusters based on their similarity, potentially revealing underlying patterns or structures within the data that are not immediately apparent. By assigning a cluster label to each data point, we can introduce a new feature that captures these patterns, which might help differentiate between relevant and non-relevant URLs. I chose to use K=3, meaning the algorithm was set to divide the data into three clusters. This choice was guided by the expectation that a moderate number of clusters would strike a balance between capturing meaningful groupings without overcomplicating the model. Then, the inclusion of these cluster labels as a feature could potentially enhance the model's ability to detect patterns in the data that are relevant to predicting URL relevance.

The results of the K-Means clustering analysis revealed a significant imbalance in the clustering of the training data. One dominant cluster contained 79,932 observations, while the remaining two clusters were sparsely populated with only 22 and 92 observations, respectively. This stark imbalance suggests that the K-Means algorithm primarily identified a single, large group of similar data points, with the other clusters capturing outliers or less frequent patterns. In contrast, the test data showed a more balanced distribution across the three clusters, with each cluster containing approximately a third of the observations. Based on these results, I decided to incorporate the cluster label as an additional feature in the model. The primary reason for this decision was the balanced distribution in the test data, which suggests that the cluster label could serve as a valuable differentiating feature. Through capturing distinct patterns in the test data, this feature could enhance the model's predictive performance, contributing to more accurate relevance predictions. Below are my K-Means clustering results in histograms:

K-Means Clustering Histogram of Labels:



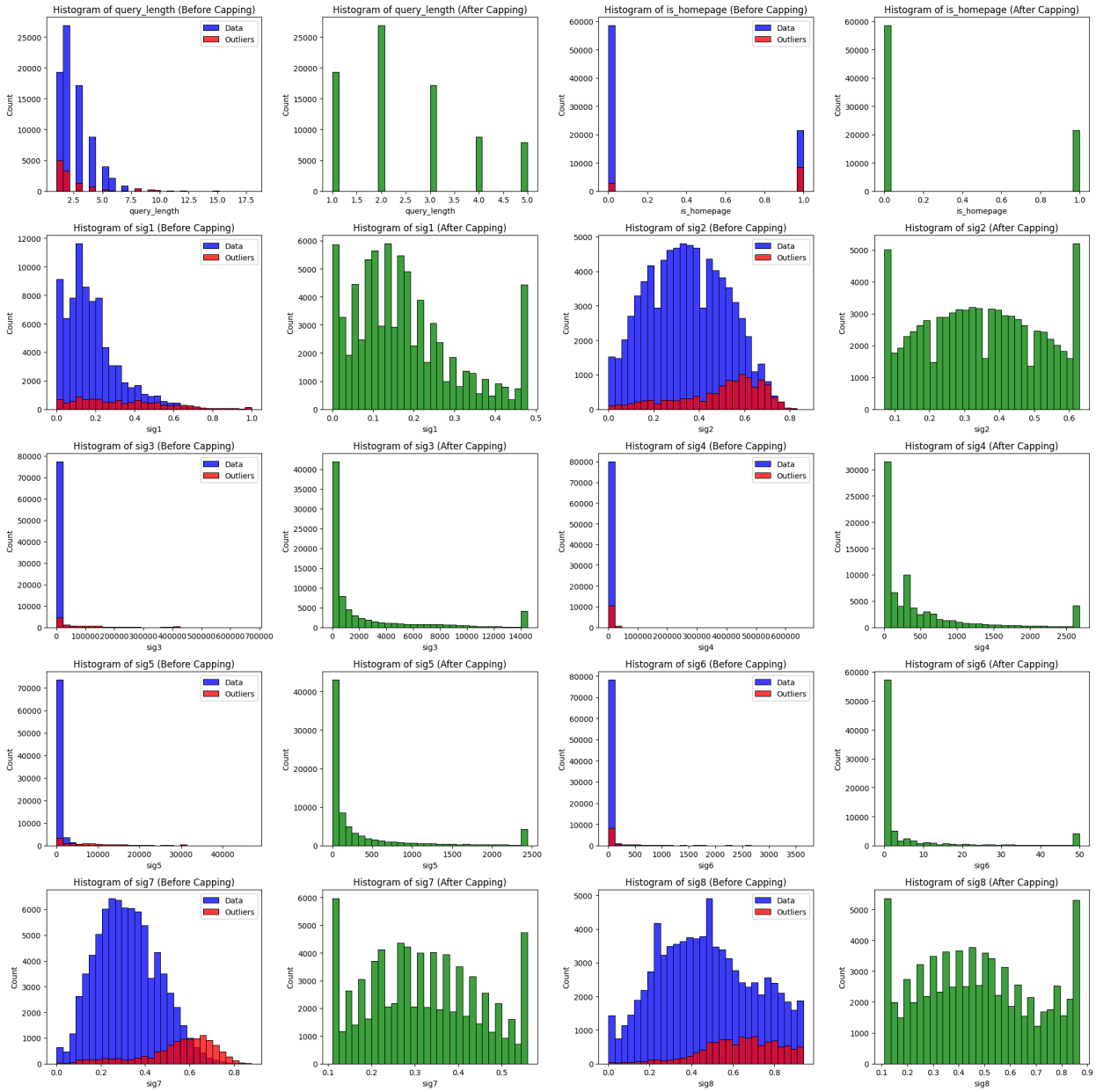
Outliers and Transformations

To address potential issues with outliers and their influence on predictive models, I first focused on identifying and addressing outliers within the dataset. Outliers can have a significant impact on classification models, often inflating errors and distorting relationships between variables, which can lead to poor model performance and reduced generalizability. Therefore, it is crucial to detect these extreme values in the analysis process. To address this, I decided to use Z-score analysis, a statistical method that quantifies how many standard deviations a data point is from the mean of the dataset, to identify outliers. By calculating the Z-scores for each feature, I was able to pinpoint data points that significantly deviated from the mean, using a threshold of $Z > 3$ used to flag potential outliers. Using this approach, I revealed the detection of 11,482 outliers in the training data and 4,554 outliers in the test data.

Recognizing the need to mitigate the influence of these outliers without discarding valuable data, I opted for a capping transformation. This technique involves replacing values above the 95th percentile with the 95th percentile value, and values below the 5th percentile with the 5th percentile value. Capping preserves the overall structure and distribution of the data while minimizing the impact of extreme values. To assess the effectiveness of this transformation, I created side-by-side histograms comparing the distributions of each feature before and after capping. The visualizations clearly demonstrated how the capping successfully reduced the presence of outliers, leading to more uniform distributions. For example, features like ‘sig5’, which initially had a highly skewed distribution with numerous outliers, showed a more normalized distribution after capping. This adjustment not only reduced outlier-driven bias but also maintained the overall variability in the dataset, which is crucial for accurately capturing

relationships between the features and relevance in the data. Below are the histograms of features before and after capping outliers:

Histograms of Features Before vs. After Capping Outliers:



Fitting Predictive Models

In developing a reliable model to predict URL relevance, I explored several classification algorithms, including logistic regression, Linear Discriminant Analysis (LDA), a neural network MLP classifier, and a gradient boosting classifier. Each of these models operates differently, providing various strengths and weaknesses in handling the data. Logistic regression is a linear

model that predicts the probability of a binary outcome by modeling the relationship between the features and the target variable using a logistic function. It is effective when the relationship between the features and the target is approximately linear. LDA, on the other hand, assumes that the data from each class comes from a Gaussian distribution and attempts to find a linear combination of features that best separates the classes. This model works well when the assumptions about the distribution hold true. The neural network MLP (Multilayer Perceptron) classifier is a more complex model that consists of multiple layers of interconnected nodes (neurons), capable of capturing non-linear relationships between features. It is particularly powerful when there are intricate patterns in the data but can be prone to overfitting. Finally, the gradient boosting classifier is an ensemble learning method that builds a strong model by sequentially adding weaker models (usually decision trees) and correcting the errors of the previous models. It is best in capturing complex patterns and interactions between features but requires careful tuning to prevent overfitting.

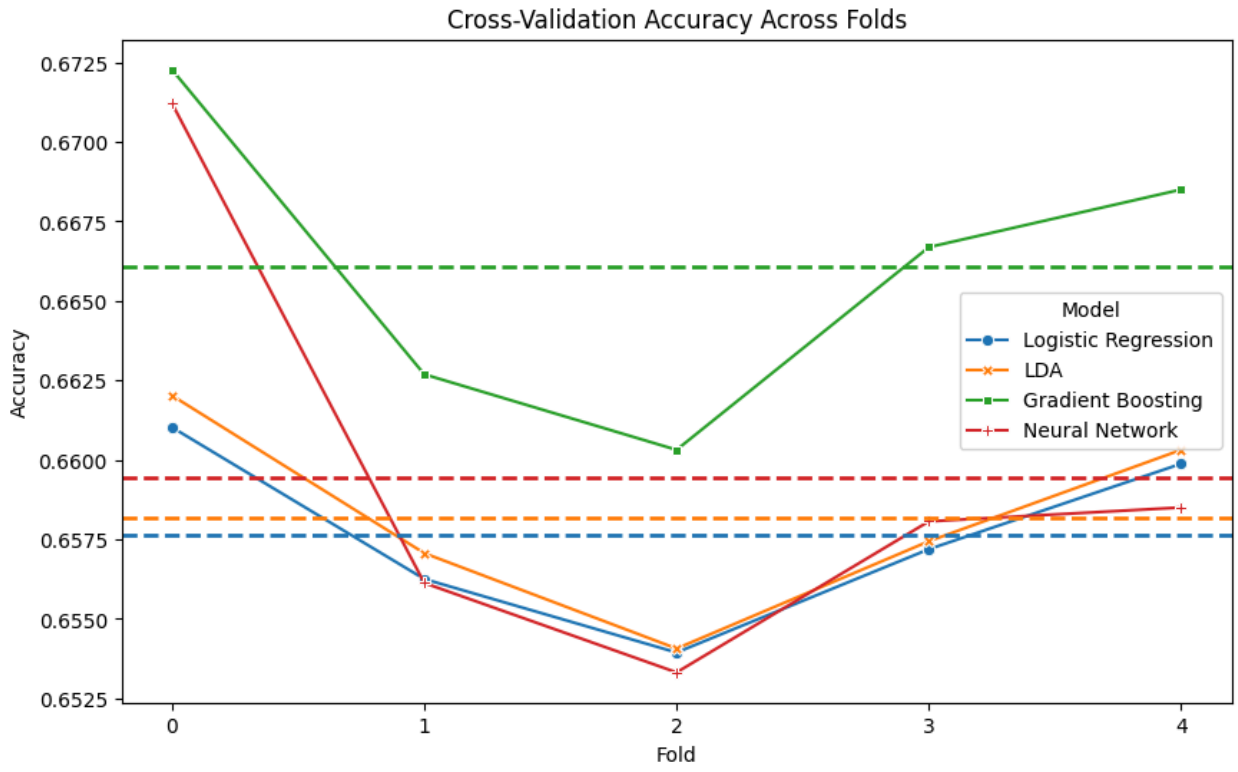
To ensure the validity of the model selection process, I used 5-fold cross-validation instead of a simple holdout validation set or Leave-One-Out Cross-Validation (LOOCV). In 5-fold cross-validation, the dataset is split into five subsets. The model is then trained and validated five times, each time using a different subset as the validation set and the remaining four subsets as the training set. This method provides a more reliable estimate of model performance by ensuring that every observation is used for both training and validation, thereby reducing the risk of overfitting to a particular subset of data. Compared to a holdout validation set, which might provide a biased estimate depending on how the data is split, 5-fold cross-validation offers a better balance between bias and variance. Although LOOCV could be an alternative, it would be too computationally expensive for this dataset. The 5-fold cross-validation strikes a good balance between computational efficiency and the quality of the performance estimate, making it well-suited for this predictive task.

Among the models I evaluated, the gradient boosting classifier consistently delivered the best performance in terms of accuracy. To build this model, I carefully selected hyperparameters to balance bias and variance. I set the number of estimators to 200, which means that the model consists of 200 sequentially added decision trees. This choice makes it so the model has enough weak learners to form a strong predictive ensemble without overfitting. The learning rate, set at 0.1, controls the contribution of each tree to the final model. A lower learning rate helps the model make gradual adjustments, reducing the risk of overfitting by preventing large swings in predictions. Additionally, I chose a maximum tree depth of 3, ensuring that the trees remain shallow. Shallow trees are less likely to capture noise in the training data, which further controls overfitting. The "sqrt" option for max_features was employed to limit the number of features considered for each split, which introduces randomness and increases the generalization ability of the model by reducing the correlation between trees.

The gradient boosting model's performance was evaluated using 5-fold cross-validation, resulting in a consistent average cross-validation score of 0.667. This consistent performance across different folds suggests that the model generalizes well to unseen data, effectively

balancing the bias-variance tradeoff. With this model, I was able to obtain an accuracy of 0.66744 on the public test set in the Kaggle submission. Below is a line plot of the cross-validation accuracy of each of the models:

Cross-Validation Accuracy of Models:



Appendix

Github repo: https://github.com/JeremyBischoff/STATS202FinalProject.git

Colab link: https://colab.research.google.com/drive/1mDkltobphO66z-rhnchyT3-ZnM78OQAI#scrollTo=cqu3wTR7yOiz