Final Report

Introduction

In the Preliminary Results, the development and evaluation of initial machine learning models were conducted, following the steps outlined in the Analytic Plan. This Final Report builds on those findings by refining the predictive models and implementing advanced modeling techniques to help the ABC Corporation identify customers at high risk of attrition. Model performance is compared to select the most effective approach, and the most important predictors of customer attrition are highlighted. Based on these insights, actionable recommendations are provided to help ABC Corporation retain at-risk customers. Finally, next steps - including areas for improvement and strategies for ongoing model maintenance, are outlined to ensure long term model effectiveness and reliability.

1 Summary of Analytic Plan and Preliminary Results

The Analytic Plan outlined the initial steps for identifying customers at high risk of attrition for ABC Corporation. Guided by the business objective of retaining valuable customers and reducing acquisition costs, the plan included comprehensive data exploration, proposed feature engineering, and initial variable selection.

In the exploratory analysis, both demographic and behavioral variables were examined to uncover attrition patterns. Key predictors that emerged included customer age, education level, transaction activity, credit usage, and periods of inactivity. To refine the feature set, correlation analysis was used, and logarithmic transformations of skewed variables were proposed as part of the data preparation process.

Two supervised learning algorithms - Logistic Regression and k-Nearest Neighbors (kNN) - were selected for initial modeling, given their strengths in binary classification problems. To address class imbalance, undersampling techniques were applied to the training data. Model performance was evaluated using key metrics, including accuracy, precision, recall, F1 score, and AUC. Both models performed well: kNN achieved the highest recall and F1 score, while Logistic Regression offered greater interpretability for business stakeholders.

Based on these findings, subsequent work in the Final Report focused on refining the Logistic Regression and kNN models by introducing new features and conducting more extensive exploratory data analysis to assess variable interactions and potential multicollinearity. In addition, ensemble methods - Random Forest and Gradient Boosting - were implemented to enhance predictive performance. All models were compared, and business-focused recommendations were developed to guide ABC Corporation in reducing customer attrition.

The following sections detail data preparation, model refinement and implementation, comparisons between models, and business-oriented recommendations based on the best model.

2 Data Preparation

As outlined in the previous milestones (Analytic Plan and Preliminary Results), the data underwent several preprocessing steps including encoding categorical predictors, applying logarithmic transformations to address skewness and non-linearity, and verification for missing values. For the final report, additional features were engineered to enhance the predictive models and improve their ability to identify customers at risk of attrition.

2.1 Feature Engineering

2.1.1 Average Transaction Size

Average Transaction Size is calculated as the ratio of Total Amount and Total Transaction Count. This variable helps identify spending patterns, such as customers who make infrequent but large purchases compared to those who are making frequent, small purchases. Recognizing these patterns can provide valuable insights into customer behavior and may help flag profiles associated with higher attrition risk. The code used to calculate this variable is included in Code Listing 1 in Appendix A.

2.1.2 Engagement Score

Engagement Score combines several activity predictors - total transaction count, total transaction amount, change in transaction amount between Q1 and Q4, and change in transaction count between Q1 and Q4 - in relation to the customer's credit limit. The activity predictors were first normalized and summed to capture overall account engagement. This sum was then divided by the credit limit to measure how much of the available credit the customer is using, reflecting engagement with their credit line. Finally, the resulting score was normalized to ensure consistent scaling and improve the performance of the logistic regression and kNN models. The code used to calculate this variable can be found in Code Listing 2 in Appendix A.

2.1.3 Transaction Count to Relationship Count

The Transaction Count to Relationship Count is calculated by dividing the total number of transactions by the number of customer relationships (products held). This predictor can help indicate the depth of a customer's relationship with the ABC Corporation, reflecting how actively they engage relative to the breadth of the relationship. The code used to calculate this variable is in Code Listing 3 in Appendix A.

2.2 Impact of Engineered Features

Although Transaction Count to Relationship Count was designed to reflect the depth of customer relationships, it did not have a meaningful impact on the performance of the predictive models. In contrast, Average Transaction Size and Engagement Score consistently demonstrated strong predictive power for identifying customer attrition. Including these features allowed the models to achieve high performance with fewer overall predictors, improving interpretability - particularly for the logistic regression model. Additionally, both predictors were ranked among the most important variables in the Random Forest and Gradient Boosting Models. The specific impact of these two predictors is discussed in more detail in later sections.

2.3 Encoding Indicators

To support modeling flexibility, the categorical variables Gender, Marital_Status, and Income_Category were encoded as factors, building on the ordinal encodings for Education_Level and Card_Category developed in the Preliminary Results. Because Gradient Boosting Models require numerical predictors, these encodings enabled the GBMs to incorporate all variables, including those that are categorical. The encoding logic for these variables is provided in Table 2, Table 3, and Table 4 in Appendix B. The corresponding code used for these custom encoding schemes can be found in Code Listing 4 in Appendix A.

2.4 Customers_All Data Frame

Customers_All is a consolidated data frame that includes the original variables, encoded factors, logarithmically transformed variables, and engineered features. This structure allows for each subsetting of columns depending on the requirements of each modeling technique. A complete list of all variables included in this data frame can be found in Table 1 in Appendix B.

3 Model Refinement

The logistic regression and k-Nearest Neighbors models from the Preliminary Results were re-run with the inclusion of the newly engineered features - Average Transaction Size, Engagement Score, and Transaction Count to Relationship Count. Additional exploratory data analysis was conducted to determine the predicate power of variables, explore potential interactions, and assess multicollinearity between predictors. This refinement process helped reduce overfitting and increase interpretability by ensuring only the most meaningful predictors were included in the models.

3.1 Logistic Regression Model

3.1.1 Variable Selection

The customers_all data set was subset to create customers_lasso, which included 23 numerical predictors, with dummy variables included for categorical predictors. The code for defining this data set and creating the balanced training set through undersampling of the majority class, can be found in Code Listing 5 in Appendix A. Predictors were refined using the Lasso method with the binomial family, which is well-suited for binary classification and variable selection. Using 10-fold cross-validation and an optimal lambda value of 0.0130, it was determined appropriate to remove the following variables from the model: Customer_Age, Dependent_Count Income_Category, Months_on_book, Total_Amt_Chng_Q4_Q1, Total_Trans_Amt, Avg_Utilization_Ratio, Education_Level_dummy, Card_Category_dummy, Credit_Limit_log, and Trans_Ct_to_Relationship_Ct. The resulting coefficients are provided in Table 5 in Appendix B.

Next, the customers_logistic dataset was created (Code Listing 6 in Appendix A), excluding variables with coefficients shrunk to zero by Lasso. The code in Code Listing 6 also details the creation of a balanced training set for the logistic regression model.

After completing the expanded exploratory analysis (Section 3.1.2), the Lasso method was re-run, this time including the following interaction terms:

- Total_Revolving_Bal * Avg_Utilization_Ratio_log
- Total_Relationship_Ct * Engagement_Score
- Total_Amt_Chng_Q4_Q1 * Engagement_Score
- Total_Amt_Chng_Q4_Q1 * Engagement_Score * Gender

The code for this step is provided in Code Listing 7 in Appendix A. Using 10-fold cross-validation and an optimal lambda value of 0.0099, the retained interaction terms were Total_Revolving_Bal * Avg_Utilization_Ratio_log, Total_Amt_Chng_Q4_Q1 * Engagement_Score, and Gender * Engagement_Score. The resulting coefficients are provided in Table 6 in Appendix B.

3.1.2 Expanded Exploratory Data Analysis (EDA)

To assess multicollinearity among predictors, a heatmap of correlation coefficients was created (Figure 1 in Appendix C). Correlations above 0.7 or below -0.7 were considered significant and, therefore, indicative of multicollinearity. Based on this threshold, the only notable case of multicollinearity was between Avg_Utilization_Ratio_log and Total_Revolving_Bal, with a correlation of 0.85. All other predictor pairs had correlations with magnitudes no greater than 0.55.

To evaluate the predictive strength of individual variables, a facet plot of box plots, shown in Figure 2 in Appendix C, was created using the training_set_logistic (defined in Code Listing 6 in Appendix A). These parallel box plots compared the distributions of numeric predictors between Existing Customers versus Attrited Customers. Strong predictors were identified by substantial differences in medians or minimal overlap between the middle 50% of values (interquartile ranges) across groups. Based on this analysis, the following predictors emerged as particularly powerful indicators of attrition:

- Avg_Utilization_Ratio_log
- Contacts_Count_12_mon
- Engagement_Score
- Months_Inactive_12_mon
- Total_Trans_Ct

Additionally, Total_Ct_Chng_Q4_Q1, Total_Relationship_Count, and Total_Revolving_Bal showed moderate but noteworthy predictive value.

To explore potential interactions, scatterplots were created for pairs of quantitative variables, colored by attrition value, and faceted by gender, yielding the following insights:

- Figure 3 in Appendix C shows a noticeably different slope between attrition groups when comparing Total Revolving Balance and Average Utilization Ratio, though the faceted plot (Figure 4 in Appendix C) suggests little difference between genders.

- Figure 5 in Appendix C reveals distinct patterns across attrition groups when comparing Total Relationship Count and Engagement Score, with existing customers having greater Engagement Scores and higher Total Relationship Counts overall. However, Figure 6 in Appendix C, indicates that there is little difference between genders for this relationship.

- Figure 9 in Appendix C shows minimal difference between attrition groups in the relationship between Avg_Open_To_Buy and Avg_Utilization_Score. When faceted by gender, Figure 10 in Appendix C shows almost no gender-related differences.

By focusing on predictors and interactions with meaningful results, noise, multicollinearity, and overfitting can be reduced when creating models in the final report. This further exploration will also improve the models' interpretability and generalizability.

3.1.3 Logistic Regression Model Comparison

Three Logarithmic Regression models were built:

- Logistic Regression Model 1: Full predictor set (customers_logistic_set; shown in Code Listing 6 in Appendix A)
- Logistic Regression Model 2: Powerful predictors plus significant interaction terms (shown in Code Listing 8 in Appendix A)
- Logistic Regression Model 3: Powerful predictors without interaction terms (Code Listing 9 in Appendix A)

In Model 1, all predictors except Marital_Status were statistically significant, as shown in the coefficient summary (Table 7 in Appendix B).

The test set confusion matrix was:

		True Customer status		
		Existing	Attrited	Total
Predicted Customer status	Existing	1490	49	1539
	Attrited	217	270	487
	Total	1707	319	2026

From this, the accuracy was calculated as 86.87%, indicating that the model correctly classified approximately 87% of the cases. To further evaluate the performance, the following metrics were computed:

- Precision = 0.5544: Of the customers predicted to attrite, 55.44% actually did.
- Recall (Sensitivity) = 0.8464: The model correctly identified 84.64% of customers who actually attrited.
- F1 Score = 0.6700: A balance between precision and recall, indicating moderate model performance, particularly in capturing attrited customers.

This model outperformed the preliminary models on all metrics.

Model 2 included key predictors and interaction terms; the only variable that was not statistically significant was GenderM, as shown in the coefficient summary (Table 8 in Appendix B).

When applied to the withheld test data, the model produced the following confusion matrix:

		True Customer status		
		Existing	Attrited	Total
Predicted Customer status	Existing	1420	52	1472
	Attrited	287	267	554
	Total	1707	319	2026

From this, the accuracy was calculated as 83.27%, indicating that the model correctly classified approximately 83% of the cases. To further evaluate the performance, the following metrics were computed:

- Precision = 0.4819: Of the customers predicted to attrite, 48.19% actually did.
- Recall (Sensitivity) = 0.8370: The model correctly identified 83.70% of customers who actually attrited.
- F1 Score = 0.6117: A balance between precision and recall, indicating moderate model performance, particularly in capturing attrited customers.

While this model had slightly better recall, it underperformed on other metrics and was less accurate than the null model, which predicts the majority class with 84.25% accuracy.

Model 3 used key predictors without interaction terms, and all were statistically significant, as shown in Table 9 in Appendix B. When applied to the withheld test data, the model the confusion matrix produced was:

		True Customer status		
		Existing	Attrited	Total
Predicted Customer status	Existing	1402	58	1460
	Attrited	305	261	566
	Total	1707	319	2026

From this, the accuracy was calculated as 82.08%, indicating that the model correctly classified approximately 82% of the cases. To further evaluate the performance, the following metrics were computed:

- Precision = 0.4611: Of the customers predicted to attrite, 46.11% actually did.
- Recall (Sensitivity) = 0.8182: The model correctly identified 81.82% of customers who actually attrited.
- F1 Score = 0.5898: A balance between precision and recall, indicating moderate model performance, particularly in capturing attrited customers.

This model had the lowest performance across all metrics.

A summary of the key performance metrics for all new and preliminary models is provided in Table 10 in Appendix B. Overall, Model 1 in the Final Report emerged as the best-performing logistic regression model, outperforming the others in accuracy, precision, recall, and F1 score. Importantly, it balances strong predictive performance with interpretability and generalizability, making it the preferred choice for deployment and the logistic regression model that will be compared to other models developed in the Final Report.

3.2 k-Nearest Neighbors Model

Because k-Nearest Neighbors requires numeric predictors, the customers_kNN dataset was created, including 23 numerical predictors with dummy variables representing categorical predictors. A training set and test set were prepared, ensuring balanced class distributions and properly scales predictors, as required for kNN. The balanced training set contained 2,616 observations, helping to reduce the risk of overfitting and mitigate the effects of the curse of dimensionality, given that the number of observations *n* was substantially larger than the number of predictors *p*. The definitions of these datasets are provided in Code listing 10 in Appendix A.

A for loop was run to identify the optimal value of k. Based on the results summarized in Table 11 in Appendix B, k = 5 was selected, as it optimized accuracy (0.8840) and F1 Score (0.9282). The final kNN model was built using k = 5 on the training data set then applied to the test data set (Code listing 10 in Appendix A), resulting in the confusion matrix:

		True Customer status		
		Existing	Attrited	Total
Predicted Customer status	Existing	1519	47	1566
	Attrited	188	272	460
	Total	1707	319	2026

From this, the accuracy was calculated as 88.4%, indicating that the model correctly classified approximately 88.4% of the cases. Additional performance metrics were as follows:

- Precision = 0.5913: Of the customers predicted to attrite, 59.13% actually did.
- Recall (Sensitivity) = 0.8527: The model correctly identified 85.27% of customers who actually attrited.
- F1 Score = 0.6983: A moderate balance between precision and recall.

4 Advanced Model Methods

To leverage the advantages of ensemble methods based on decision trees, Random Forest and Gradient Boosting models were developed. These advanced, and at times computationally intensive, methods were included with the goal of improving predictive performance beyond what was achieved with the previously built Logistic Regression and kNN models, particularly due to their ability to capture complex interactions and reduce overfitting.

The development, hyperparameter tuning, and evaluation of the Random Forest and Gradient Boosting models are outlined in the following sections.

4.1 Random Forest Model

The development of the datasets used to train and test the Random Forest models is shown in Code Listing 11 in Appendix A. Since Random Forest models do not require dummy variables or logarithmic transformations, a subset of the customers_all dataset, customers_rf, was created including only the original, unchanged predictors. As an extra precaution to minimize model bias, the data was randomly split 70/30 into a training set and test set. To ensure equal representation of the Attrition classes, undersampling was implemented to balance the training set.

4.1.1 Bagging Model

Random Forest Model 1 was developed using bagging, a special case where m, the number of predictors considered at each split, equals p, the total number of predictors in the data set. The model was built with 500 trees, using 22 predictors at each split ($m = p$). The out-of-bag (OOB) error estimate 6.27%, meaning that 93.73% of OOB observations were correctly classified.

To assess the model's performance using a true holdout test set, the following confusion matrix was produced from the test data set:

		True Customer status		
		Existing	Attrited	Total
Predicted Customer status	Existing	2356	29	2385
	Attrited	205	449	654
	Total	2561	478	3039

The code used to develop and analyze the Random Forest model with bagging can be found in Code Listing 12 in Appendix A.

Based on the test data, the Random Forest model with bagging achieved an accuracy of 92.30%. Additional performance metrics examined were:

- Precision = 0.6865: Of the customers predicted to attrite, 68.65% actually did.

- Recall (Sensitivity) = 0.9393: The model correctly identified 93.93% of customers who actually attrited.
- F1 Score = 0.7933: A strong balance between precision and recall.
- AUC = 0.9817: This high area under the ROC curve indicates excellent discriminative ability between existing and attrited customers. The ROC curve is shown in Figure 11 in Appendix C.

Beyond overall performance, variable importance was assessed using Mean Decrease Accuracy (MDA) and Mean Decrease Gini, as shown in Figure 12 and Figure 13 respectively in Appendix C. The predictor Total_Trans_Ct ranked highest on both metrics, providing robust evidence that the model relies heavily on total transaction count for accurate attrition prediction and that this variable consistently improves decision tree splits. Other variables that were ranked high on both metrics included Avg_Transaction_size, Total_Revolving_Bal, Engagement_Score, and Total_Trans_Amt.

4.1.2 Hyperparameter Tuning

Hyperparameter tuning was performed on the values of ntree (the number of trees in the forest), mtry (the number of predictors at each split), and nodesize (the minimum size of terminal nodes) to improve upon Random Forest Model 1.

First, to ensure the model had enough trees to stabilize performance, the OOB error was examined for different values of ntree while using the default values for mtry ($mtry \approx \sqrt{p} = \sqrt{22} \approx 5$) and nodesize (=1). Graphical representations of the test error curves were analyzed between 0 and 1000 (Figure 14 in Appendix C), 0 and 200 (Figure 15 in Appendix C), and 0 and 50 trees (Figure 16 in Appendix C) to observe when the curve flattened. Figure 16 clearly shows that the test error begins to stabilize between 25 and 35 trees. To pinpoint the optimal value, the point where the error reduction first dropped by less than 0.0005 was identified at 26 trees, based on calculations shown in Code Listing 13 in Appendix A. This result is highlighted in Figure 17 in Appendix C with a vertical blue line.

Next, to determine the optimal number of predictors at each split, mtry, a grid search was performed at values around $\sqrt{p}$ (where p = 22). The initial grid search tested values between 2 and 10, as shown in Code Listing 14 in Appendix A. Results from this grid search (Table 12 in Appendix B) indicated that mtry = 8 achieved the highest accuracy. A refined grid search using integer values between 6 and 10 (to hone in around 8) confirmed that mtry = 8 remained the best value (Table 13 in Appendix B).

Finally, with the ntree and mtry optimized, nodesize was tuned by running a for loop to identify the value that minimized OOB error (Code Listing 15 in Appendix A). Based on the results shown in Table 14 in Appendix B and Figure 18 in Appendix C, the optimal node size was confirmed as 1.

4.1.3 Best Random Forest Model

Based on the tuning process, the optimized hyperparameters for the Random Forest model were ntree = 26, mtry = 8, and nodesize = 1. Using these values, Random Forest Model 2 was built (Code Listing 16 in Appendix A). The model consisted of 26 trees, with 8 predictors randomly selected at each split. The out-of-bag (OOB) error estimate was 6.88%, meaning that 93.12% of OOB observations were correctly classified.

To assess the model's performance using a true holdout test set, the following confusion matrix was generated:

		True Customer status		
		Existing	Attrited	Total
Predicted Customer status	Existing	2361	24	2385
	Attrited	200	454	654
	Total	2561	478	3039

Based on the test data, the Random Forest Model 2 achieved an accuracy of 92.63%. Additional performance metrics explored included:

- Precision = 0.6942: Of the customers predicted to attrite, 69.42% actually did.
- Recall (Sensitivity) = 0.9498: The model correctly identified 94.98% of customers who actually attrited.
- F1 Score = 0.8021: A strong balance between precision and recall.
- AUC = 0.9837: This high area under the ROC curve indicates excellent discriminative ability between existing and attrited customers. The ROC curve is shown in Figure 19 in Appendix C.

Variable importance was also assessed using Mean Decrease Accuracy (MDA) and Mean Decrease Gini, as shown in Figure 20 and Figure 21 respectively in Appendix C. Once again, the predictor Total_Trans_Ct ranked highest on both metrics, confirming its critical role in predicting attrition and its consistent contribution to decision tree splits. Other variables that were ranked high on both metrics were Total_Trans_Amt, Avg_Transaction_size, and Engagement_Score.

As shown in Table 15 in Appendix B, Random Forest Model 2 slightly outperformed Random Forest Model 1 across all performance metrics. Given its improved performance, combined with its simplicity and enhanced interpretability, Random Forest Model 2 was selected as the preferred model. It will serve as the benchmark for comparison to other selected predictive models in Section 5.

4.2 Gradient Boosting Model

In contrast with the Random Forest modeling, Gradient Boosting modeling requires numeric predictors and benefits using log-transformed variables rather than raw data values. The development of the data set used to train the Gradient Boosting models (GBM) is shown in Code Listing 17 in Appendix A. This code also defines the training data set for the GBMs, balances the classes using undersampling, and creates the test data set for evaluating GBM performance on unseen data.

4.2.1 Baseline Gradient Boosting Model

Gradient Boosting Model 1 was first built using the default parameter values: number of trees (n.trees = 100), shrinkage parameter ($\lambda = 0.001$), and interaction depth (d = 1), as shown in Code Listing 18 in Appendix A. The summary results showing the relative influence of each predictor can be seen in Table 16 in Appendix B and Figure 22 in Appendix C. The relative influence results showed that Total_Trans_Ct again ranked highest, reinforcing its importance in predicting customer attrition.

Before assessing the model performance, the optimal number of trees was reviewed by examining model improvement across iterations, confirming that n.trees = 100 was appropriate (Code Listing 18, Figure 23 in Appendix C).

Gradient Boosting Model 1's performance was assessed based on the results of the confusion matrix (Code Listing 18 in Appendix A):

		True Customer status		
		Existing	Attrited	Total
Predicted Customer status	Existing	2210	36	2246
	Attrited	351	442	793
	Total	2561	478	3039

Based on the test data, the Gradient Boosting Model 1 achieved an accuracy of 87.27%. Additional performance metrics explored included:

- Precision = 0.5574: Of the customers predicted to attrite, 55.74% actually did.
- Recall (Sensitivity) = 0.9247: The model correctly identified 92.47% of customers who actually attrited.
- F1 Score = 0.6955: A moderate balance between precision and recall.

While this baseline GBM performed well enough, the next section focuses on optimizing its performance through hyperparameter tuning.

4.2.2 Hyperparameter Tuning

A grid search was conducted using for loop (Code Listing 19 in Appendix A) to identify the optimal values for the number of trees (n.trees), shrinkage value (shrinkage, λ), and interaction depth value (d). Based on results shown in Table 17 in Appendix B, the optimal hyperparameters were identified at the minimum cross-validation error (cv_error = 0.1595). The best-performing combination was n.trees = 3798, shrinkage = 0.010, and interaction depth = 7. These values were used to develop the final, optimized Gradient Boosting Model 2.

4.2.3 Best Gradient Boosting Model

Code Listing 20 in Appendix A shows how Gradient Boosting Model 2 was built using the best-performing combination of hyperparameters. The summary results showing the relative influence of each predictor can be seen in Table 18 in Appendix B and Figure 24 in Appendix C. The relative influence results showed that Total_Trans_Ct once again ranked highest, reinforcing its importance in predicting customer attrition.

Gradient Boosting Model 2's performance was assessed based on the results of the confusion matrix (Code Listing 20 in Appendix A):

		True Customer status		
		Existing	Attrited	Total
Predicted Customer status	Existing	2429	21	2450
	Attrited	132	457	589
	Total	2561	478	3039

Based on the test data, the Gradient Boosting Model 2 achieved an accuracy of 87.27%. Additional performance metrics explored included:

- Precision = 0.7759: Of the customers predicted to attrite, 77.59% actually did.
- Recall (Sensitivity) = 0.9561: The model correctly identified 95.61% of customers who actually attrited.
- F1 Score = 0.8566: A strong balance between precision and recall.
- AUC = 0.9908: This high area under the ROC curve indicates excellent discriminative ability between existing and attrited customers. The ROC curve is shown in Figure 25 in Appendix C.

While this baseline GBM performed well enough, the next section focuses on optimizing its performance through hyperparameter tuning.

5 Model Comparison

Table 19 in Appendix B compares the key performance metrics - accuracy, precision, recall, F1 score, and AUC - across the Null, Logistic Regression, k-Nearest Neighbors, Random Forest, and Gradient Boosting models. While the Random Forest model achieved the highest accuracy (92.63%), the Gradient Boosting model outperformed all the other models on precision, recall, F1 score, and AUC, indicating stronger overall predictive performance in identifying customers at risk of attrition.

The Gradient Boosting model grows trees sequentially, with each new tree learning from the errors of the previous ones. This gradual learning process typically leads to robust performance, particularly in complex datasets. Although gradient boosting models are more complex than other models (i.e., logistic regression models), techniques such as analyzing variable importance and partial dependence plots help enhance their interpretability by highlighting the most influential predictors.

6 Business Recommendations

6.1 Final Predictive Model

Having outperformed the other models built in this assessment in nearly all of the performance metrics, the Gradient Boosting model is recommended as the primary tool to predict customer attrition at ABC Corporation. This model was built on 4,740 iterations with all 19 predictors at non-zero influence. Based on the relative influence statistics (shown in Table 18 in Appendix B), the most critical predictor is Total_Trans_Ct (total transaction count). Other highly influential predictors included Total_Trans_Amt, Total_Revolving_Bal, Avg_Transaction_size, and Engagement_Score.

6.2 Business Implications of Critical Features

The model's findings highlight that customer behavior, particularly around transaction activity, is the strongest indicator of attrition risk. Customers with fewer transactions, reduced spending per transaction, or lower engagement with their credit lines are at higher risk for attrition. Notably, demographic predictors such as dependent count, education level, and customer age had minimal influence, suggesting that attrition is driven more by account usage patterns than by static customer demographics.

ABC Corporation should consider that transaction count, amount, and engagement can provide early warning signals of disengagement. Customers who show significant decline in transaction count or spending may be considering leaving ABC Corporation, signaling an opportunity for a timely intervention to retain the customers.

6.3 Next Steps

6.3.1 Actionable Strategies

ABC Corporation can operationalize the Gradient Boosting model as part of a targeted retention campaign. By proactively identifying high-risk customers - particularly those showing declining transaction activity - the corporation can provide personalized incentives such as bonus points, fee waivers, or exclusive offers to encourage customers re-engagement.

Personalized outreach strategies have been shown to improve retention and increase the lifetime value of customers. If customer engagement does not improve through automated incentive campaigns, ABC Corporation can escalate intervention via direct, personalized communication via emails or app notifications, with messaging tailored to the specific patterns and risk factors that triggered concern.

In parallel, ABC Corporation should develop a transaction monitoring dashboard to track customer activity trends - such as the number and amount of transactions and quarterly changes in transaction volume. This dashboard can enable marketing and service teams to act in real time as significant shifts in customer engagement are detected. Ownership of the dashboard should reside with the Data Science and IT teams.

6.3.2 Model Deployment

ABC Corporation can deploy the Gradient Boosting model in batch mode, scoring customer records at regular intervals to flag at-risk customers. Given that transactional data updates monthly, a quarterly schedule would be an optimal balance between responsiveness and operational efficiency, while allowing for periodic model retraining. Batch processing is preferable over real-time scoring due to the computational cost of the model and the fact that attrition risk signals emerge over time.

To ensure successful deployment, ABC Corporation will need to establish a data pipeline that regularly updates customer transactions and engagement metrics. The Data Science team should oversee the model governance, monitoring KPIs such as accuracy, recall, precision, F1 Score, and AUC. Regular retraining of the model (at least semi-annually) will be needed to maintain long-term model effectiveness.

Additionally, the marketing and customer service teams will need to be provided with training on how to interpret the model's outputs and apply recommended interventions effectively. This will ensure alignment between predictive insights and on-the-ground customer engagement efforts.

6.3.3 Limitations and Future Improvements

While the current model has a strong predictive performance, it could be improved to further increase predictive accuracy and business impact. Future work could involve adopting more advanced ensemble methods, such as XGBoost with SHAP-based interpretability.

A noteworthy limitation is that, because the model relies heavily on customer activity data, there is a risk of bias if certain customer segments have sparse histories. For example, new customers would not have as extensive transaction history. To mitigate this issue, ABC Corporation may consider incorporating qualitative feedback or external data sources to ensure balanced risk assessment and minimal bias. Additional data sources may also provide richer insight into customer retention and attrition. ABC Corporation may consider incorporating qualitative data around customer satisfaction collected via survey.

Finally, to ensure sustained impact, model performance should be continuously monitored, validated, and retrained using fresh data as customer behaviors, preferences, and economic conditions evolve over time.

7 Conclusion

The Final Report, along with the previously presented Analytic Plan and Preliminary Results, intended to help ABC Corporation proactively identify and retain customers at risk of attrition using a data-driven, machine learning approach. After rigorous data preparation, feature engineering, exploratory data analysis, and building and refining models, the Gradient Boosting model emerged as the most effective predictive model, as it outperformed the logistic regression, k-nearest neighbor, and random forest models in key performance metrics including precision, recall, F1 score, and AUC.

The model's findings emphasized the high influence of customers' transactional behavior, with total transaction count consistently emerging as the most influential predictor. Other features, such as total transaction amount, average transaction amount, and engagement score, were also consistently among the most influential predictors. These results enable ABC Corporation to develop targeted retention strategies focused on transactional behavior patterns.

To capitalize on the results of this report, ABC Corporation has been encouraged to integrate the Gradient Boosting modeling into a quarterly batch-processing workflow, develop a dashboard to monitor transaction-based attrition predictors, and enhance engagement through targeted retention campaigns and personalized outreach. Continued success will require monitoring, validating, and retraining the model as well as introducing new data sources to ensure evolution of the model regardless of changes in customer behaviors and business needs.

8 Appendix A: Code Samples

Code Listing 1: Average Transaction Size

```
Avg_Transaction_size <-
     customers1$Total_Trans_Amt/customers2$Total_Trans_Ct
```

Code Listing 2: Engagement Score

```
Avg_Transaction_size <-
      customers1$Total_Trans_Amt/customers2$Total_Trans_Ct

Engagement_Score <- scale((scale(customers1$Total_Trans_Amt) +
     scale(customers1$Total_Trans_Ct) +
     scale(customers1$Total_Amt_Chng_Q4_Q1) +
     scale(customers1$Total_Ct_Chng_Q4_Q1) +
     scale(customers1$Total_Relationship_Count))/customers1$Credit_Limit)
```

Code Listing 3: Transaction Count to Relationship Count

```
Trans_Ct_to_Relationship_Ct <-
     customers1$Total_Trans_Ct/customers1$Total_Relationship_Count
```

Code Listing 4: Custom Encoding Schemes

```
customers_all$Gender_dummy <- as.numeric(factor(customers_all$Gender,
                                               levels = c("M", "F")))

customers_all$Marital_Status_dummy <-
      as.numeric(factor(customers_all$Marital_Status,
         levels = c("Single", "Married", "Divorced", "Unknown")))

customers_all$Income_Category_dummy <-
     as.numeric(factor(customers_all$Income_Category,
         levels = c("Less than $40K", "$40K - $60K", "$60K - $80K", "$80K
         - $120K", "$120K +", "Unknown")))
```

Code Listing 5: Lasso with Newly Engineered Features

```
set.seed(123)
```

```
customers_lasso <-
     subset(customers_all, select = -c(CLIENTNUM, Attrition_Flag,
     Education_Level, Card_Category, Credit_Limit, Avg_Open_To_Buy))

balanced_x <- model.matrix(Attrition_Indicator ~ ., data =
balanced_training_set)

balanced_y <- balanced_training_set$Attrition_Indicator

balanced_cv.lasso <- cv.glmnet(balanced_x, balanced_y, alpha = 1, family =
"binomial", nfolds = 10)

balanced_bestlam <- balanced_cv.lasso$lambda.min
balanced_bestlam

## [1] 0.001848238

balanced_bestlam2 <- balanced_cv.lasso$lambda.1se
balanced_bestlam2

## [1] 0.01303895

out <- glmnet(balanced_x, balanced_y, alpha = 1)
lasso.coef <- predict(out, type = "coefficients", s = balanced_bestlam2)
```

Code Listing 6: Logistic Regression Model 1

```
set.seed(123)

customers_logistic <-
     subset(customers_lasso, select = -c(Customer_Age, Dependent_Count,
     Income_Category, Months_on_book, Total_Amt_Chng_Q4_Q1,
     Total_Trans_Amt, Avg_Utilization_Ratio, Education_Level_dummy,
     Card_Category_dummy, Credit_Limit_log, Trans_Ct_to_Relationship_Ct))

training_set_logistic <- customers_logistic[train, ]
test_set_logistic <- customers_logistic[-train,]

attrited_logistic <- training_set_logistic %>%
     filter(Attrition_Indicator == 1)
existing_logistic <- training_set_logistic %>%
     filter(Attrition_Indicator == 0)

existing_undersampled_logistic <- existing_logistic %>%
sample_n(nrow(attrited))
balanced_training_set_logistic <- bind_rows(attrited_logistic,
existing_undersampled_logistic) %>% sample_frac(1)
```

```
glm.fits <- glm(Attrition_Indicator ~ ., data =
balanced_training_set_logistic, family = binomial)

summary(glm.fits)

glm.probs <- predict(glm.fits, test_set_logistic, type = "response")


glm.pred <- rep(0, nrow(test_set_logistic))
glm.pred[glm.probs > .5] <- 1

table(glm.pred, test_set_logistic$Attrition_Indicator)
```

Code Listing 7: Lasso with Newly Engineered Features

```
set.seed(123)

balanced_training_set_logistic2 <-
     subset(balanced_training_set, select = -c(Customer_Age,
     Marital_Status, Income_Category, Months_on_book, Total_Trans_Amt,
     Avg_Utilization_Ratio, Education_Level_dummy, Card_Category_dummy,
     Credit_Limit_log, Trans_Ct_to_Relationship_Ct))

balanced_training_set_lasso2 <- balanced_x_lasso2 <-

balanced_x_lasso2 <- model.matrix(Attrition_Indicator ~ .
     + Total_Revolving_Bal * Avg_Utilization_Ratio_log
     + Total_Relationship_Count * Engagement_Score
     + Total_Amt_Chng_Q4_Q1 * Engagement_Score
     + Total_Amt_Chng_Q4_Q1 * Engagement_Score * Gender,
     data = balanced_training_set_lasso2)[, -1]

balanced_y_lasso2 <- balanced_training_set$Attrition_Indicator


balanced_cv.lasso2 <- cv.glmnet(balanced_x_lasso2, balanced_y_lasso2,
      alpha = 1, family = "binomial", nfolds = 10)
balanced_bestlam_lasso2 <- balanced_cv.lasso2$lambda.min
balanced_bestlam_lasso2

## [1] 0.0003815152

balanced_bestlam2_lasso2 <- balanced_cv.lasso2$lambda.1se
balanced_bestlam2_lasso2

## [1] 0.009900413
```

```r
out2 <- glmnet(balanced_x_lasso2, balanced_y_lasso2, alpha = 1)
lasso.coef <- predict(out2, type = "coefficients", s =
balanced_bestlam2_lasso2)
```

Code Listing 8: Logistic Regression Model 2

```r
set.seed(123)

glm.fits2 <- glm(Attrition_Indicator ~
     Avg_Utilization_Ratio_log + Contacts_Count_12_mon + Engagement_Score +
     Months_Inactive_12_mon + Total_Relationship_Count + Total_Trans_Ct +
     Total_Revolving_Bal * Avg_Utilization_Ratio_log +
     Total_Amt_Chng_Q4_Q1 * Engagement_Score + Gender * Engagement_Score,
     data = balanced_training_set_logistic2, family = binomial)

summary(glm.fits2)

glm.probs2 <- predict(glm.fits2, test_set_logistic2, type = "response")


glm.pred2 <- rep(0, nrow(test_set_logistic2))

glm.pred2[glm.probs2 > .5] <- 1

table(glm.pred2, test_set_logistic$Attrition_Indicator)
```

Code Listing 9: Logistic Regression Model 3

```r
set.seed(123)

glm.fits3 <- glm(Attrition_Indicator ~
     Avg_Utilization_Ratio_log + Contacts_Count_12_mon + Engagement_Score +
     Months_Inactive_12_mon + Total_Relationship_Count + Total_Trans_Ct,
     data = balanced_training_set_logistic, family = binomial)

summary(glm.fits3)

glm.probs3 <- predict(glm.fits3, test_set_logistic, type = "response")

glm.pred3 <- rep(0, nrow(test_set_logistic))

glm.pred3[glm.probs3 > .5] <- 1

table(glm.pred3, test_set_logistic$Attrition_Indicator)
```

Code Listing 10: k-Nearest Neighbors Model

```r
set.seed(123)

# select only numerical predictors
customers_kNN <- subset(customers_all,
     select = -c(CLIENTNUM, Attrition_Flag, Gender, Education_Level,
     Marital_Status, Income_Category, Card_Category))

# balance data
training_set_kNN <- customers_kNN[train, ]

attrited_kNN <- training_set_kNN %>% filter(Attrition_Indicator == 1)
existing_kNN <- training_set_kNN %>% filter(Attrition_Indicator == 0)

existing_kNN_undersampled <- existing_kNN %>% sample_n(nrow(attrited_kNN))
balanced_training_set_kNN <-
     bind_rows(attrited_kNN, existing_kNN_undersampled) %>% sample_frac(1)


training_set_kNN <- balanced_training_set_kNN
test_set_kNN <- customers_kNN[-train, ]

training_set_kNN <- na.omit(training_set_kNN)
test_set_kNN <- na.omit(test_set_kNN)

#before scaling, split into predictors and outcome
x_train <- training_set_kNN[, setdiff(names(training_set_kNN),
     "Attrition_Indicator")]

x_test <- test_set_kNN[, setdiff(names(test_set_kNN),
     "Attrition_Indicator")]



#scale predictors using training set
preProc <- preProcess(x_train, method = c("center", "scale"))
training_set_kNN_scaled <- predict(preProc, training_set_kNN)
test_set_kNN_scaled <- predict(preProc, test_set_kNN)


# extract target variables
y_train <- training_set_kNN$Attrition_Indicator
y_test <- test_set_kNN$Attrition_Indicator

y_test <- factor(y_test, levels = c(0, 1))
y_train <- factor(y_train, levels = c(0, 1))
```

```
# determine optimal value of k
results <- data.frame(k = integer(), Accuracy = numeric(), F1 = numeric(),
     AUC = numeric())


for (k in seq(1, 19, 2)) {
  knn.pred <-
    knn(train = training_set_kNN_scaled, test = test_set_kNN_scaled,
    cl = y_train, k = k)
  cm <- confusionMatrix(knn.pred, y_test)

  # F1 Score (for binary classification)
  precision <- cm$byClass["Precision"]
  recall <- cm$byClass["Recall"]
  f1 <- 2 * (precision * recall) / (precision + recall)

  # AUC
  knn.prob <- attr(knn(train = training_set_kNN_scaled,
                       test = test_set_kNN_scaled,
                       cl = y_train,
                       k = k,
                       prob = TRUE), "prob")
  knn_prob_adj <- ifelse(knn.pred == levels(y_train)[2], knn.prob,
  1 - knn.prob)
  auc_val <- auc(roc(response = y_test, predictor = knn_prob_adj,
  levels = rev(levels(y_test))))

  # Save results
  results <- rbind(results, data.frame(k = k,
  Accuracy = cm$overall["Accuracy"], F1 = f1, AUC = auc_val))
}

# run kNN model with optimal k=7
knn_pred <- knn(training_set_kNN_scaled, test_set_kNN_scaled, y_train, k=5)


# confusion matrix
confusionMatrix(knn_pred, y_test)
```

Code Listing 11: Random Forest Data Sets

```
set.seed(123)

# data
customers_rf <- subset(customers_all,
     select = -c(CLIENTNUM, Attrition_Flag, Education_Level_dummy,
     Card_Category_dummy, Credit_Limit_log, Avg_Open_To_Buy_log,
     Avg_Utilization_Ratio_log))
```

```
customers_rf$Attrition_Indicator <-
     as.factor(customers_rf$Attrition_Indicator)

# training set - 70/30 split
train_rf <- sample(1:nrow(customers_rf), 0.7*nrow(customers_rf))
training_set_rf <- customers_rf[train_rf, ]
test_set_rf <- customers_rf[-train_rf, ]

# balance training set
attrited_rf <- training_set_rf %>% filter(Attrition_Indicator == 1)
existing_rf <- training_set_rf %>% filter(Attrition_Indicator == 0)

existing_undersampled_rf <- existing_rf
     %>% sample_n(nrow(attrited_rf))

balanced_training_set_rf <- bind_rows(
      attrited_rf, existing_undersampled_rf)
      %>% sample_frac(1)
```

Code Listing 12: Random Forest Model 1 (with Bagging)

```
set.seed(123)

bag_customers <- randomForest(Attrition_Indicator ~ .,
     data = balanced_training_set_rf,
     mtry = ncol(balanced_training_set_rf) - 1, importance = TRUE)

# Confusion Matrix
rf_pred <- predict(bag_customers, newdata = test_set_rf)

confusionMatrix(rf_pred, test_set_rf$Attrition_Indicator)

# Variable Importance
importance(bag_customers)

varImpPlot(bag_customers)

# AUC
rf_probs <- predict(bag_customers, newdata = test_set_rf, type = "prob")

rf_auc <- roc(response = test_set_rf$Attrition_Indicator,
      predictor = rf_probs[, 1])

auc(rf_auc)

# plot ROC curve
roc_obj <- roc(response = test_set_rf$Attrition_Indicator,
               predictor = rf_probs[, 1],
               levels = c(0, 1))

plot(roc_obj, main = "ROC Curve for Random Forest Model with Bagging",
     col = "coral", lwd = 2)
```

Code Listing 13: Tuning Ntree - Random Forest Model

```
set.seed(123)

# look at OOB error across trees to determine best ntrees value
rf_model <- randomForest(Attrition_Indicator ~ ., data = training_set_rf,
     mtry = 5, ntree = 1000)

plot(rf_model)


rf_model <- randomForest(Attrition_Indicator ~ ., data = training_set_rf,
     mtry = 5, ntree = 200)

plot(rf_model)


rf_model <- randomForest(Attrition_Indicator ~ ., data = training_set_rf,
     mtry = 5, ntree = 50)

plot(rf_model)


#quantifying where the random forest error curve flattens
oob_error <- rf_model$err.rate[, "OOB"]
error_diff <- diff(oob_error)
which(abs(error_diff) <0.0005)[1]

## [1] 10

# use rolling window to avoid noise
smooth_diff <- rollmean(abs(error_diff), 5, fill = NA)

optimal_ntree <- which(smooth_diff < 0.0005) [1]

plot(oob_error, type = "l", main = "OOB Error by Tree", ylab = "OOB Error",
xlab = "Number of Trees")
abline(v = optimal_ntree, col = "turquoise", lty = 2)
```

Code Listing 14: Tuning mtry - Random Forest Model

```r
set.seed(123)

# use 5-fold cross validation to find the best mtry in grid search
mtry_grid <- expand.grid(mtry = c(2, 4, 6, 8, 10))

cv_control <- trainControl(method = "cv", number = 5)

rf_cv <- train(Attrition_Indicator ~ .,
               data = balanced_training_set_rf,
               method = "rf",
               metric = "Accuracy",
               tuneGrid = mtry_grid,
               trControl = cv_control,
               ntree = 26)

rf_cv

# refined search
mtry_grid2 <- expand.grid(mtry = c(6, 7, 8, 9, 10))

rf_cv2 <-train(Attrition_Indicator ~ .,
               data = balanced_training_set_rf,
               method = "rf",
               metric = "Accuracy",
               tuneGrid = mtry_grid2,
               trControl = cv_control,
               ntree = 500)
rf_cv2
```

Code Listing 15: Tuning nodesize- Random Forest Model

```r
set.seed(123)

# optimize nodesize
nodesize_values <- c(1, 3, 5, 10, 20)

#write a loop to test each value
results <- data.frame(nodesize = nodesize_values, OOB_Error = NA)

for (i in seq_along(nodesize_values)) {
  rf_model <- randomForest(Attrition_Indicator ~ .,
                           data = training_set_rf,
                           nodesize = nodesize_values[i],
                           ntree = 26,
                           mtry = 8)
  oob_error <- rf_model$err.rate[26, "OOB"]
  results$OOB_Error[i] <- oob_error
}

plot(results$nodesize, results$OOB_Error, type = "b", pch = 19,
     xlab = "Nodesize", ylab = "OOB Error",
     main = "OOB Error vs Nodesize")
```

Code Listing16: Random Forest Model 2 (using optimized hyperparameters)

```
set.seed(123)

# Random Forest Model with nodesize = 1
rf_model_best1 <- randomForest(Attrition_Indicator ~ .,
     data = balanced_training_set_rf, mtry = 8, ntree = 26, nodesize = 1,
     importance = TRUE)


rf_model_best1

# Confusion Matrix
rf_pred_1 <- predict(rf_model_best1, newdata = test_set_rf)
confusionMatrix(rf_pred_1, test_set_rf$Attrition_Indicator)

# Variable Importance
importance(rf_model_best1)

varImpPlot(rf_model_best1)

# AUC
rf_probs1 <- predict(rf_model_best1, newdata = test_set_rf, type = "prob")

rf_auc1 <- roc(response = test_set_rf$Attrition_Indicator,
      predictor = rf_probs1[, 1])

auc(rf_auc1)

# plot ROC curve
roc_obj1 <- roc(response = test_set_rf$Attrition_Indicator,
                predictor = rf_probs1[, 1],
                levels = c(0, 1))



plot(roc_obj1, main = "ROC Curve for Random Forest Model (nodesize = 1)",
     col = "maroon", lwd = 2)
```

Code Listing 17: Gradient Boosting Data Sets

```
set.seed(123)

# define data
customers_boost <- subset(customers_all,
     select = -c(CLIENTNUM, Attrition_Flag, Education_Level, Card_Category,
     Credit_Limit, Avg_Open_To_Buy, Avg_Utilization_Ratio, Gender,
     Marital_Status, Income_Category))

# training set - 70/30 split
train_boost <- sample(1:nrow(customers_boost), 0.7*nrow(customers_boost))

training_set_boost <- customers_boost[train_boost, ]

test_set_boost <- customers_boost[-train_boost, ]


# balance training set
attrited_boost <- training_set_boost %>% filter(Attrition_Indicator == 1)

existing_boost <- training_set_boost %>% filter(Attrition_Indicator == 0)

existing_undersampled_boost <- existing_boost %>%

sample_n(nrow(attrited_boost))

balanced_training_set_boost <- bind_rows(attrited_boost,

existing_undersampled_boost) %>% sample_frac(1)
```

Code Listing 18: Gradient Boosting Model 1 (using default parameter values)

```
set.seed(123)

#train gbm model
boost_model <- gbm(Attrition_Indicator ~.,
     data = balanced_training_set_boost, distribution = "bernoulli",
     cv.folds = 5)

summary(boost_model)



# optimize number of trees
trees_best <- gbm.perf(boost_model, method = "cv")

#confusion matrix
gbm_probs <- predict(boost_model, newdata = test_set_boost,
     n.trees = trees_best, type = "response")

gbm_pred <- ifelse(gbm_probs > 0.5, 1, 0)
gbm_pred <- factor(gbm_pred)

test_set_boost$Attrition_Indicator <-

factor(test_set_boost$Attrition_Indicator)

confusionMatrix(gbm_pred, test_set_boost$Attrition_Indicator)
```

Code Listing 19: Tuning Hyperparameter - Gradient Boosting Model

```
set.seed(123)

# define grid of parameters
shrinkage_values <- c(0.001, 0.005, 0.01)
depth_values <- c(3, 5, 7)
n.trees.max <- 2000

# create data frame to store results
boost_grid_results <- data.frame(
  shrinkage = numeric(),
  interaction.depth = integer(),
  best_iter = integer(),
  cv_error = numeric()
)

# Grid search Loop
for(shrink in shrinkage_values) {
  for (depth in depth_values) {

    cat("Fitting model with shrinkage = ", shrink, "and depth = ",
    depth, "...\n")

    # fit gradient boosting model with CV
    gbm_model <- gbm(
      Attrition_Indicator ~.,
      data = training_set_boost,
      distribution = "bernoulli",
      n.trees = n.trees.max,
      shrinkage = shrink,
      interaction.depth = depth,
      n.minobsinnode = 10,
      bag.fraction = 0.5,
      cv.folds = 5,
      verbose = FALSE
    )

    # get optimal number of trees based on CV error
    best_iter <- gbm.perf(gbm_model, method = "cv", plot.it = FALSE)
    cv_err <- min(gbm_model$cv.error)

    # store the results
    boost_grid_results <- rbind(boost_grid_results, data.frame(
      shrinkage = shrink,
      interaction.depth = depth,
      best_iter = best_iter,
      cv_error = cv_err
    ))
  }
}

best_model <- boost_grid_results %>% arrange(cv_error) %>% slice(1)
```

Code Listing 20: Gradient Boosting Model 2 (with tuned hyperparameters)

```
set.seed(123)

#train gbm model
boost_model_best <- gbm(Attrition_Indicator ~.,
                        data = balanced_training_set_boost,
                        distribution = "bernoulli",
                        n.trees = best_model$best_iter,
                        shrinkage = best_model$shrinkage,
                        interaction.depth = best_model$interaction.depth,
                        n.minobsinnode = 10,
                        bag.fraction = 0.5,
                        verbose = FALSE)

summary(boost_model_best)

gbm_probs_best <- predict(boost_model_best, newdata = test_set_boost,
     type = "response")


gbm_pred_best <- ifelse(gbm_probs_best > 0.5, 1, 0)

gbm_pred_best <- factor(gbm_pred_best)
test_set_boost$Attrition_Indicator <-
factor(test_set_boost$Attrition_Indicator)

confusionMatrix(gbm_pred_best, test_set_boost$Attrition_Indicator)
```

```
# AUC
boost_roc <- roc(
  response = test_set_rf$Attrition_Indicator,
  predictor = gbm_probs_best
)


boost_auc <- auc(boost_roc)
print(paste("AUC:", round(boost_auc, 4)))


# plot ROC curve

plot(boost_roc, main = "ROC Curve for Gradient Boosting Model",
     col = "plum", lwd = 2)
```

9 Appendix B: Tables

Table 1: Complete list of variables included in Customers_All data frame

Variable Name	**Variable Type**	**Description**	**Possible Values**
CLIENTNUM	quantitative, integer	unique customer identifier	positive integer values
Attrition_Flag	qualitative, factor	indicates whether or not the customer has attrited	"Existing customer" "Attrited Customer"
Attrition_Indicator	quantitative, factor	indicates whether or not the customer has attrited	0 = Existing customer 1 = Attrited Customer
Customer_Age	quantitative, integer, demographic	customer's age in years	integer values between 26 and 73
Gender	qualitative, factor, demographic	customer's gender	“M” = Male “F” = Female
Gender_dummy	quantitative,	customer's gender	1 = Male

	factor, demographic		2 = Female
Variable Name	**Variable Type**	**Description**	**Possible Values**
Dependent_count	quantitative, integer, demographic variable	number of dependents the customer has	integer values between 0 and 5
Education_level	qualitative, factor, demographic	customer's education level	"Uneducated", "High School", "College", "Graduate", "Post-Graduate", "Doctorate", "Unknown"
Education_Level_dummy	quantitative, factor, demographic	customer's education level	0 = Unknown 1 = Uneducated 2 = High School 3 = College 4 = Graduate 5 = Post-Graduate 6 = Doctorate
Marital_Status	qualitative, factor, demographic	customer's marital status	"Single", "Married", "Divorced", "Unknown"
Marital_Status_dummy	quantitative, factor, demographic	customer's marital status	1 = Single 2 = Married 3 = Divorced 4 = Unknown
Income_Category	qualitative, factor, demographic	annual income range of customer	"Less than $40K", "$40K-$60K", "$60K-$80K", "80K-$120K", "$120K +", “Unknown”
Income_Category_dummy	quantitative, factor	annual income range of customer	1 = Less than $40K 2 = $40K-$60K 3 = $60K-$80K 4 = 80K-$120K

			5 = $120K + 6 = Unknown
Variable Name	**Variable Type**	**Description**	**Possible Values**
Card_Category	qualitative, factor	type of card the customer has	"Blue", "Silver", "Gold", "Platinum"
Card_Category	quantitative, factor	type of card the customer has	0 = Blue 1 = Silver 2 = Gold 3 = Platinum
Months_on_book	quantitative, integer	period of customer's relationship with ABC in months	integer values between 13 and 56
Total_Relationship_Count	quantitative, integer	number of products held by the customer	integer values between 1 and 6
Months_Inactive_12_mon	quantitative, integer	number of months customer has been inactive in the last 12 months	integer values between 0 and 6
Contacts_Count_12_mon	quantitative, integer	number of times the customer has been in contact with ABC over the last 12 months	integer values between 0 and 6
Credit_Limit	quantitative	credit limit on the credit card	values between $1438 and $34,516
Credit_Limit_log	quantitative	logarithmic transformation of credit limit on the credit card	values between 7.2712 and 10.4492
Total_Revolving_Balance	quantitative, integer	revolving balance on the credit card	integer values between $0 and $2,517
Avg_Open_To_Buy	quantitative	average open to buy credit line from the last 12 months	values between $3 to $34,516
Avg_Open_To_Buy_log	quantitative	logarithmic transformation of average open to buy credit line from the last 12 months	values between 1.0986 and 10.4492
Total_Amt_Chng_Q4_Q1	quantitative	change in transaction amount (Q4 over Q1)	values between 0.000 and 3.397
Total_Trans_Amt	quantitative, integer	total transaction amount from the last 12 months	integer values between $510 and $18,484

Total_Trans_Ct	quantitative, integer	total transaction count from the last 12 months	integer values between 10 and 139
Variable Name	**Variable Type**	**Description**	**Possible Values**
Total_Ct_Chng_Q4_Q1	quantitative	change in number of transactions (Q4 over Q1)	values between 0.000 and 3.714
Avg_Utilization_Ratio	quantitative	average card utilization ratio	values between 0.000 and 0.999
Avg_Utilization_Ratio_log	quantitative	logarithmic transformation of average card utilization ratio	values between -6.2146 and 0.0010
Avg_Transaction_size	quantitative	average transaction size	values between 19.1379 and 190.1932
Engagement_Score	quantitative	combination of activity predictors in relation to credit limit	values between -6.9765 and 8.3039
Trans_Ct_to_Relationship_Ct	quantitative	ratio between transaction count and relationship count	values between 2.5 and 138

Table 2: Encoding Logic - Gender

VARIABLE: 'Gender_dummy'	
Numeric Indicators	***Category***
1	'Male'
2	'Female

Table 3: Encoding Logic - Marital Status

VARIABLE: 'Marital_Status_dummy'	
Numeric Indicators	***Category***
1	'Single'
2	'Married'
3	'Divorced'

4	‘Unknown’

Table 4: Encoding Logic - Income Category

VARIABLE: ‘Income_Category_dummy’	
Numeric Indicators	***Category***
1	‘Less than $40K’
2	‘$40K - $60K’
3	‘$60K - $80K’
4	‘$80K - $120K’
5	‘$120K +’
6	‘Unknown’

Table 5: Lasso Coefficients

Predictor	***Coefficient***
Customer_Age	.
GenderM	-3.3047e-02
Dependent_Count	.
Marital_StatusMarried	-2.6807e-02
Marital_StatusSingle	7.9924e-03
Marital_StatusUnknown	.
Income_Category$40K - $60K	.
Income_Category$60K - $80K	.
Income_Category$80K - $120K	.

Income_CategoryLess than $40K	.
Predictor	***Coefficient***
Income_CategoryUnknown	.
Months_on_book	.
Total_Relationship_Count	-2.4460e-02
Months_Inactive_12_mon	4.1697e-02
Contacts_Count_12_mon	5.3661e-02
Total_Revolving_Bal	-1.6562e-06
Total_Amt_Chng_Q4_Q1	.
Total_Trans_Amt	.
Total_Trans_Ct	-1.0058e-02
Total_Ct_Chng_Q4_Q1	-2.7690e-01
Avg_Utilization_Ratio	.
Education_Level_dummy	.
Card_Category_dummy	.
Credit_Limit_log	.
Avg_Open_To_Buy_log	-5.7194e-03
Avg_Utilization_Ratio_log	-4.1697e-02
Avg_Transaction_size	5.0452e-03
Engagement_Score	-5.2886e-02
Trans_Ct_to_Relationship_Ct	.

Table 6: Lasso Coefficients - With Interaction Terms

Predictor	***Coefficient***
GenderM	-2.3691e-02
Dependent_Count	5.1244e-03
Total_Relationship_Count	-2.5952e-02
Months_Inactive_12_mon	4.6106e-02
Contacts_Count_12_mon	5.6901e-02
Total_Revolving_Bal	.
Total_Amt_Chng_Q4_Q1	-1.8686e-02
Total_Trans_Ct	-1.0408e-02
Total_Ct_Chng_Q4_Q1	-2.7789e-01
Avg_Open_To_Buy_log	.
Avg_Utilization_Ratio_log	-3.4421e-02
Avg_Transaction_size	5.5962e-03
Engagement_Score	-1.5494e-02
Total_Revolving_Bal:Avg_Utilization_Ratio_log	3.1135e-05
Total_Relationship_Count: Engagement_Score	.
Total_Amt_Chng_Q4_Q1:Engagement_Score	-6.3306e-02
GenderM:Total_Amt_Chng_Q4_Q1	.
GenderM:Engagement_Score	1.4909e-02
GenderM:Total_Amt_Chng_Q4_Q1:Engagement_Score	.

Table 7: Logistic Regression 1 Coefficient Summary

Coefficients:

	Estimate	Std. Error	z value	Pr(>|z|)	
(Intercept)	0.7889	0.8253	0.985	0.3391	
GenderM	-0.4180	0.435	-2.912	0.0036	**
Marital_StatusMarried	-0.4304	0.2573	-1.673	0.0944	.
Marital_StatusSingle	0.0625	0.2597	0.241	0.8098	
Marital_StatusUnknown	-0.1052325	0.3290	-0.320	0.7491	
Total_Relationship_Count	-0.2164	0.0465	-4.650	3.31e-06	***
Months_Inactive_12_mon	0.4489	0.06324	7.097	1.27e-12	***
Contacts_Count_12_mon	0.5231	0.0597	8.757	< 2e-16	***
Total_Revolving_Bal	0.0002	0.0001	1.713	0.0868	.
Total_Trans_Ct	-0.0836	0.0048	-17.556	< 2e-16	***
Total_Ct_Chng_Q4_Q1	-2.1784	0.3147	-6.922	4.47e-12	***
Avg_Open_To_Buy_log	-0.1617	0.0676	-2.392	0.0168	*
Avg_Utilization_Ratio_log	-0.4963	0.0583	-8.513	< 2e-16	***
Avg_Transaction_Size	0.0516	0.0033	15.846	< 2e-16	***
Engagement_Score	-0.7568	0.1225	-6.180	6.42e-10	***

Signif. Codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 '' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 3626.5 on 2615 degrees of freedom
Residual deviance: 1653.2 on 2601 degrees of freedom
AIC: 1683.2

Coefficients:

Number of Fisher Scoring iterations: 6

Table 8: Logistic Regression 2 Coefficient Summary

Coefficients:

	Estimate	Std. Error	z value	Pr(>|z|)	
(Intercept)	-1.313	5.702e-01	-2.303	0.02128	*
Avg_Utilization_Ratio_log	-5.150e-01	4.912e-02	-10.484	< 2e-16	***
Contacts_Count_12_mon	4.710e-01	5.349e-02	8.805	< 2e-16	***
Engagement_Score	-6.411e-01	1.963e-01	-3.266	0.00109	**
Months_Inactive_12_mon	4.380e-01	5.710e-02	7.670	1.72e-14	***
Total_Relationship_Count	-2.668e-01	4.031e-02	-6.619	3.62e-11	***
Total_Trans_Ct	-4.807e-02	3.653e-03	-13.159	< 2e-16	***
Total_Revolving_Bal	5.806e-04	1.416e-04	4.100	4.13e-05	***
Total_Amt_Chng_Q4_Q1	5.702e-01	2.899e-01	1.967	0.04918	*
GenderM	-5.7663e-02	1.270e-01	0.454	0.64993	
Avg_Utilization_Ratio_log:Total_Revolving_Bal	1.743e-04	5.112e-05	3.410	0.00065	***
Engagement_Score:Total_Amt_Chng_Q4_Q1	-7.713e-01	2.374e-01	-3.250	0.00116	**
Engagement_Score:GenderM	6.888e-01	1.503e-01	4.582	4.60e-06	***

Signif. Codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 3626.5 on 2615 degrees of freedom

Residual deviance: 2014.1 on 2603 degrees of freedom

```
Coefficients:

AIC: 2040.1

Number of Fisher Scoring iterations: 6
```

Table 9: Logistic Regression 3 Coefficient Summary

```
Coefficients:
                              Estimate   Std. Error   z value     Pr(>|z|)
(Intercept)                     0.1400       0.3504     0.399        0.797
Avg_Utilization_Ratio_log      -0.3654       0.0242   -15.112     < 2e-16  ***
Contacts_Count_12_mon           0.4673       0.0524     8.926     < 2e-16  ***
Engagement_Score               -0.8468       0.0860    -9.850     < 2e-16  ***
Months_Inactive_12_mon          0.4319       0.0560     7.715     1.21e-14 ***
Total_Relationship_Count       -0.2738       0.0387    -7.070     1.54e-12 ***
Total_Trans_Ct                 -0.0475       0.0035   -13.654     < 2e-16  ***
---------------------------------------------------------------------------
Signif. Codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 '' 1
---------------------------------------------------------------------------
(Dispersion parameter for binomial family taken to be 1)

Null deviance: 3626.5 on 2615 degrees of freedom
Residual deviance: 2073.7 on 2609 degrees of freedom
AIC: 2087.7

Number of Fisher Scoring iterations: 6
```

Table 10: Performance Comparison between Logistic Regression Models

Model	Accuracy	Precision	Recall	F1 Score
Null	0.8425	n/a	0	0

Preliminary Results - Logistic Regression 1	0.8386	0.4926	0.8307	0.6184
Preliminary Results - Logistic Regression 2	0.8401	0.4953	0.8307	0.6206
Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**
Preliminary Results - Logistic Regression 3	0.8391	0.4953	0.8339	0.6200
Final Report - Logistic Regression 1	0.8687	0.5544	0.8464	0.6700
Final Report - Logistic Regression 2	0.8327	0.4819	0.8370	0.6117
Final Report - Logistic Regression 3	0.8208	0.4611	0.8182	0.5898

Table 11: Performance of kNN Model for odd values of k between 1 and 19

k	**Accuracy**	**F1 Score**	**AUC**
1	0.8549	0.9090	0.8438
3	0.8776	0.9241	0.9199
5	0.8840	0.9282	0.9317
7	0.8811	0.9262	0.9399
9	0.8815	0.9265	0.9432
11	0.8800	0.9254	0.9449
13	0.8810	0.9261	0.9452
15	0.8786	0.9246	0.9453
17	0.8825	0.9271	0.9457
19	0.8810	0.9262	0.9462

Table 12: Grid Search for mtry = 2, 4, 6, 8, 10

mtry	**Accuracy**	**Kappa**
2	0.9138	0.8277
4	0.9308	0.8616
6	0.9347	0.8695
8	0.9378	0.8755
10	0.9352	0.8703

Table 13: Grid Search for mtry = 6, 7, 8, 9, 10

mtry	**Accuracy**	**Kappa**
6	0.9321	0.8643
7	0.9387	0.8773
8	0.9404	0.8808
9	0.9365	0.8729
10	0.9343	0.8686

Table 14: OOB Error based on nodesize = 1, 3, 5, 10, 20

nodesize	**OOB_Error**
1	0.0040
3	0.0457
5	0.0408

10	0.0421
20	0.0425

Table 15: Performance Comparison between Random Forest Models

Model	Accuracy	Precision	Recall	F1 Score	AUC
Random Forest Model 1 (with bagging)	0.9230	0.6865	0.9393	0.7933	0.9817
Random Forest Model 2 (using tuned hyperparameters)	0.9263	0.6942	0.9498	0.8021	0.9837

Table 16: Gradient Boosting Model 1 (default parameters) - relative influence statistics

Variable	Relative Influence
Total_Trans_Ct	42.8060
Total_Revolving_Bal	21.1771
Avg_Transaction_size	11.1955
Engagement_Score	8.0700
Total_Ct_Chng_Q4_Q1	5.2617
Months_Inactive_12_mon	3.8472
Total_Trans_Amt	2.6263
Total_Relationship_Count	2.0190
Contacts_Count_12_mon	1.8022
Total_Amt_Chng_Q4_Q1	0.8914
Trans_Ct_to_Relationship_Ct	0.3037
All other variables have relative influence = 0	

Table 17: Performance Comparison between Random Forest Models

Shrinkage	Interaction Depth	Best Iteration (n.trees)	cv_error
0.001	1	5000	0.5135
0.001	3	5000	0.3269
0.001	5	5000	0.2670
0.001	7	5000	0.2378
0.005	1	5000	0.3093
0.005	3	5000	0.1836
0.005	5	4996	0.1682
0.005	7	4998	0.1598
0.010	1	5000	0.2431
0.010	3	4908	0.1670
0.010	5	4740	0.1588
0.010	7	3798	0.1595

Table 18: Gradient Boosting Model 2 (with tuned hyperparameters) - relative influence statistics

Variable	Relative Influence
Total_Trans_Ct	34.6310
Total_Trans_Amt	13.1763

Total_Revolving_Bal	12.0302
Avg_Transaction_size	11.9250
Engagement_Score	6.3050
Variable	**Relative Influence**
Total_Ct_Chng_Q4_Q1	4.0947
Total_Relationship_Count	3.7785
Total_Amt_Chng_Q4_Q1	3.5661
Months_Inactive_12_mon	2.5631
Trans_Ct_to_Relationship_Ct	1.5990
Customer_Age	1.5966
Avg_Open_To_Buy_log	1.0054
Contacts_Count_12_mon	0.9747
Credit_Limit_log	0.9007
Avg_Utilization_Ratio_log	0.7937
Months_on_book	0.7521
Education_Level_dummy	0.1463
Dependent_count	0.1196
Card_Category_dummy	0.0420

Table 19: Performance Comparison between Models

Model	Accuracy	Precision	Recall	F1 Score	AUC
Null	0.8425	n/a	0	0	-
Logistic Regression	0.8687	0.5544	0.8464	0.6700	0.9105

kNN	0.8840	0.5913	0.8527	0.6983	0.9462
Random Forest	0.9263	0.6942	0.9498	0.8021	0.9837
Gradient Boosting	0.8727	0.7759	0.9561	0.8566	0.9908

10 Appendix C: Figures

Figure 1 *shows the heat map for correlation coefficients.*

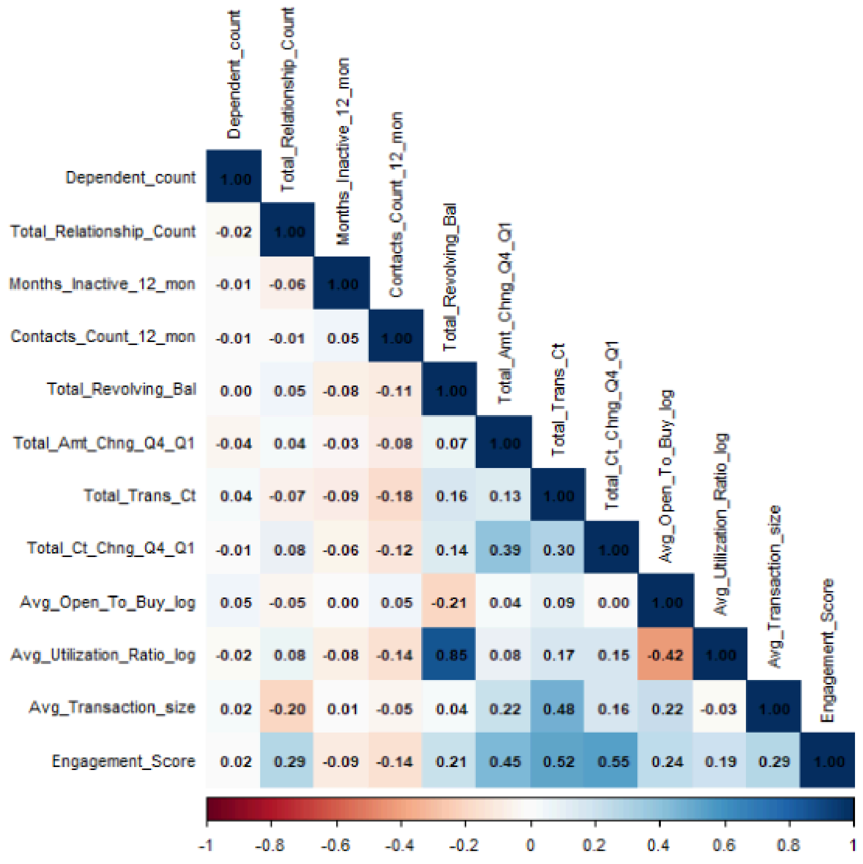


Figure 2 *shows the facet grid of box plots of quantitative predictors versus the value of Attrition_Indicator.*

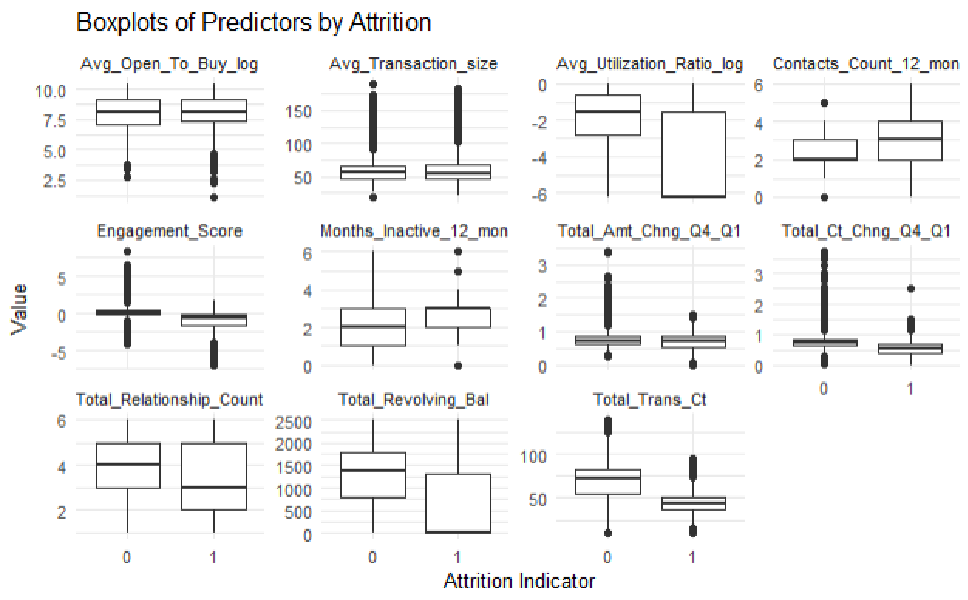


***Figure 3** shows the relationship between Total Revolving Balance and Average Utilization Ratio, colored by Attrition.*

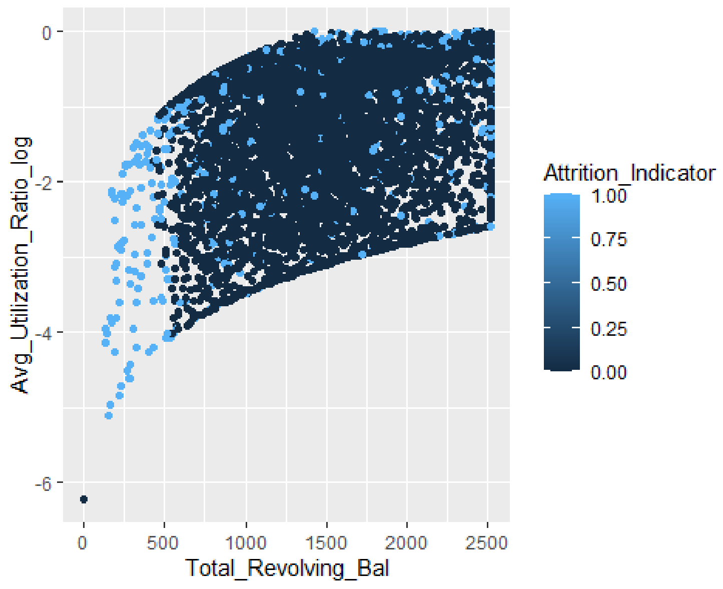


***Figure 4** shows the relationship between Total Revolving Balance and Average Utilization Ratio, colored by Attrition and faceted by Gender.*

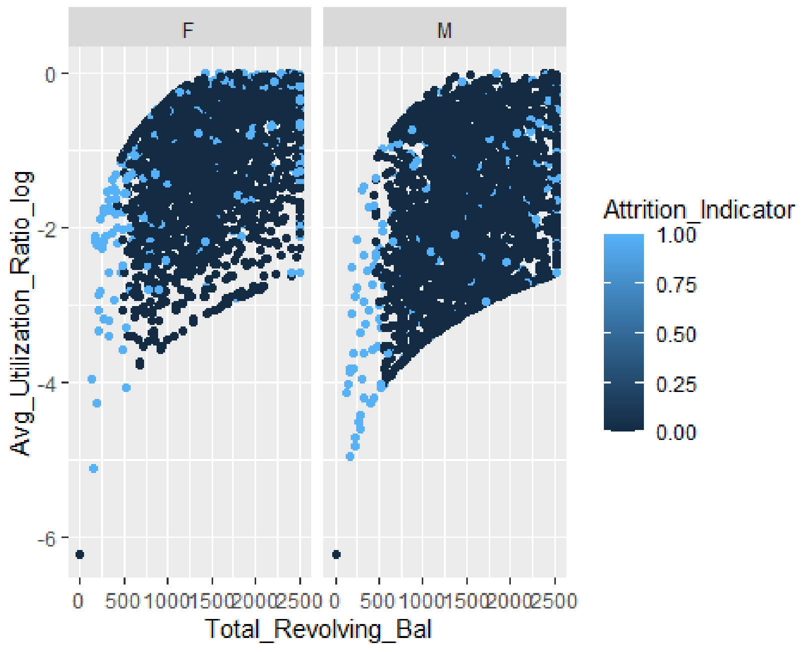


***Figure 5** shows the relationship between Total Relationship Count and Engagement Score, colored by Attrition.*

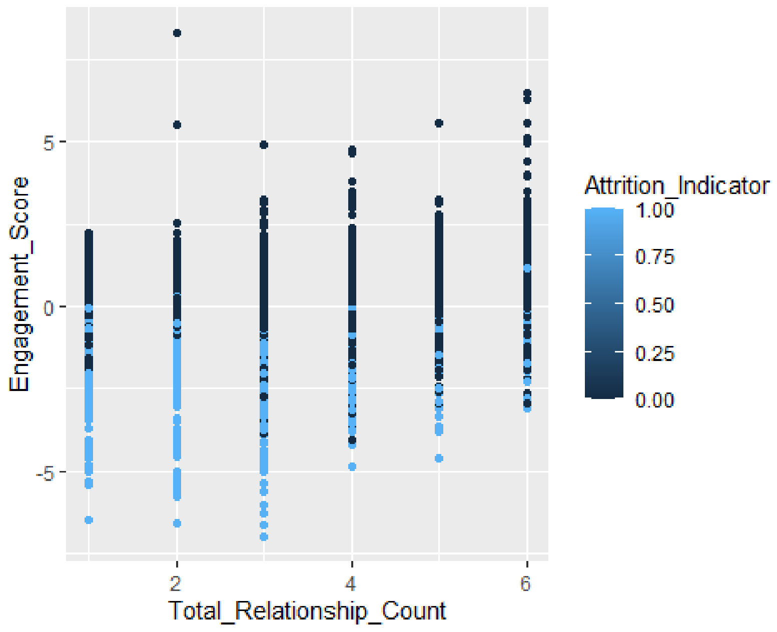


Figure 6 *shows the relationship between Total Relationship Count and Engagement Score, colored by Attrition and faceted by Gender.*

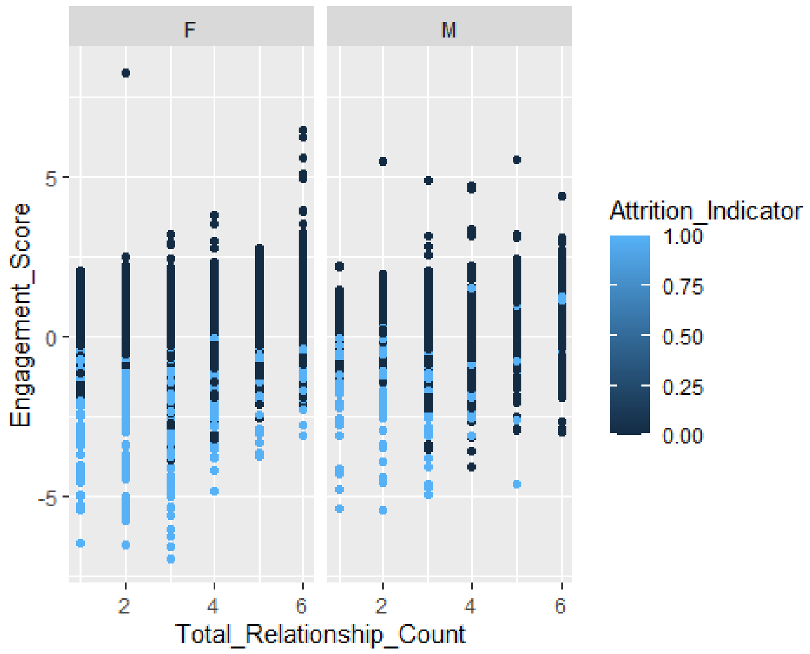


Figure 7 *shows the relationship between Total Amount Change and Engagement Score, colored by Attrition.*

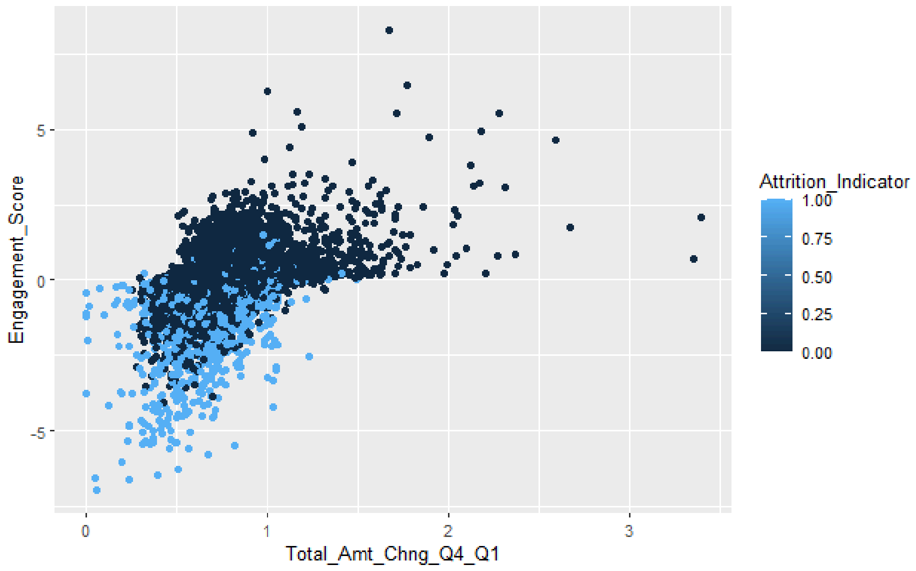


Figure 8 *shows the relationship between Total Amount Change and Engagement Score, colored by Attrition and faceted by Gender.*

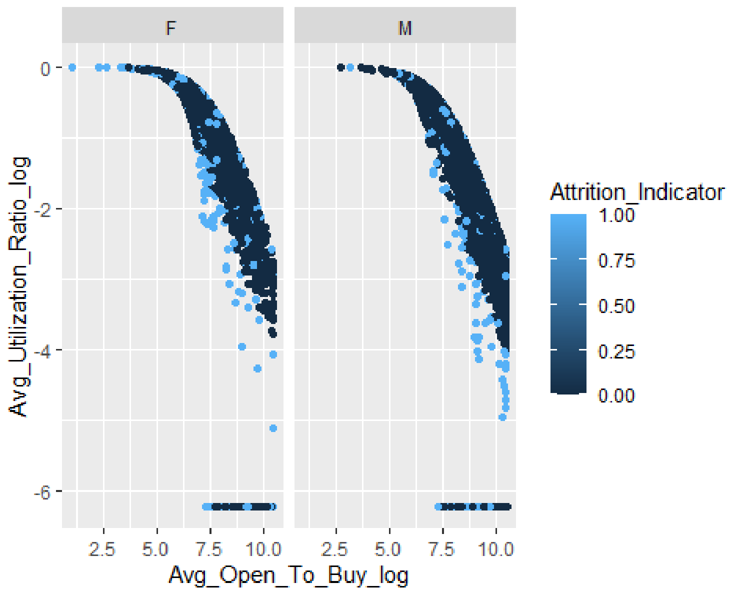


Figure 9 *shows the relationship between Average Open To Buy and Average Utilization Ratio colored by Attrition.*

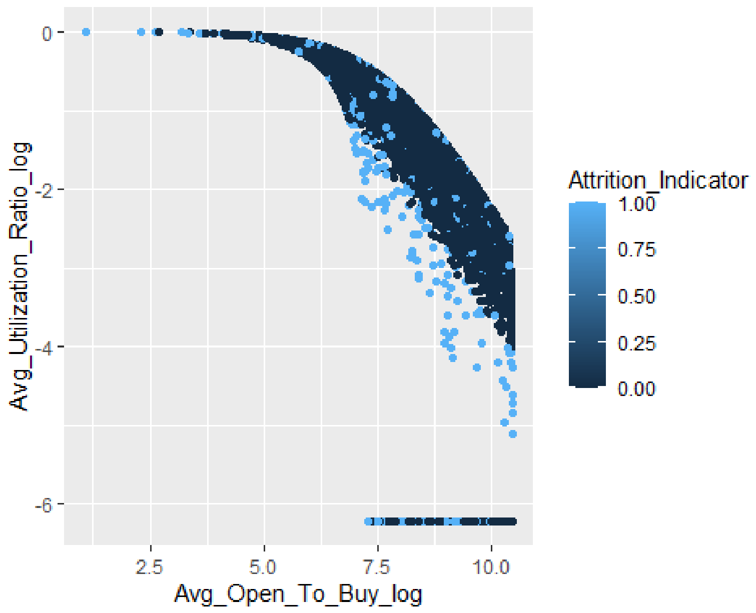


Figure 10 *shows the relationship between Average Open To Buy and Average Utilization Ratio colored by Attrition faceted by Gender.*

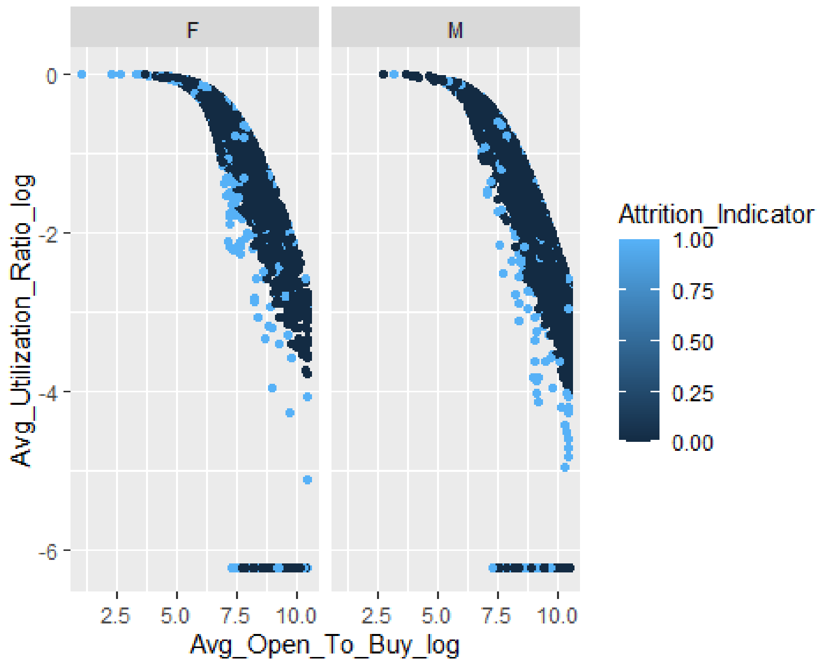


Figure 11 *shows the graph of the ROC Curve for Random Forest Model 1 (with bagging).*

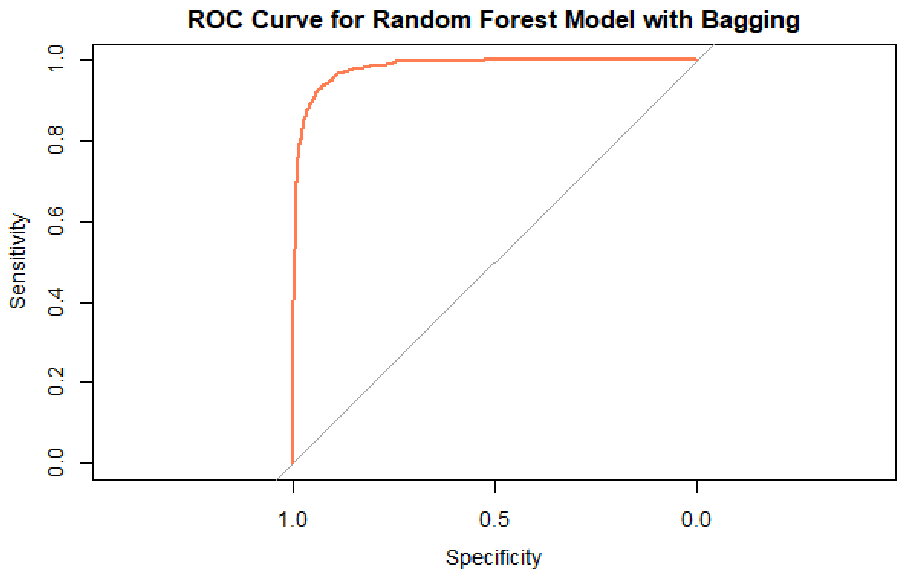


***Figure 12** shows variable importance plot Random Forest Model 1 (with bagging) based on the Mean Decrease Accuracy (MDA).*

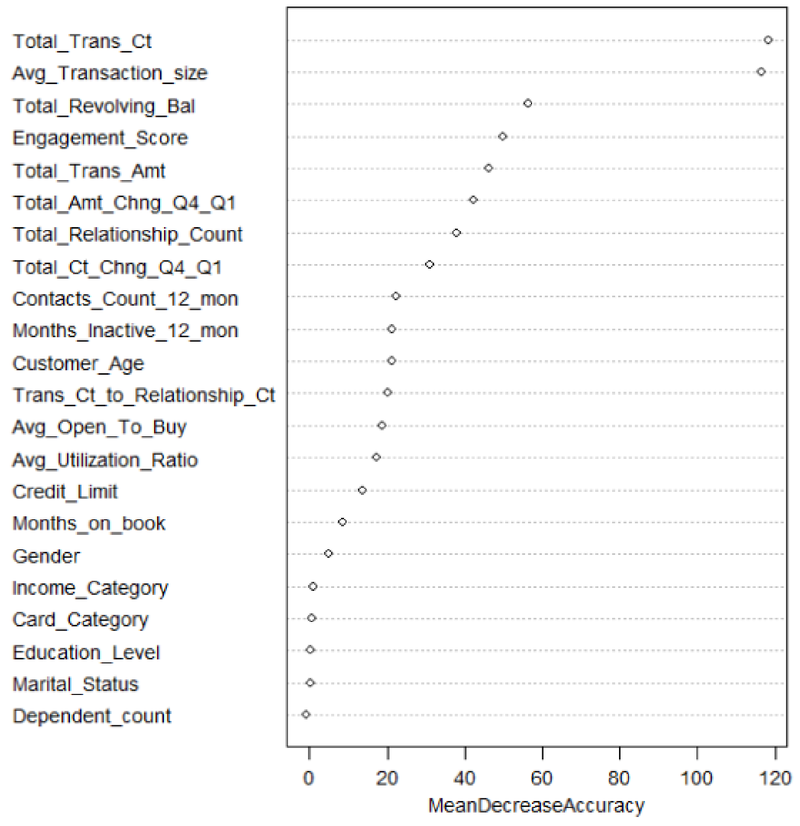


***Figure 13** shows variable importance plot Random Forest Model 1 (with bagging) based on the Mean Decrease Gini.*

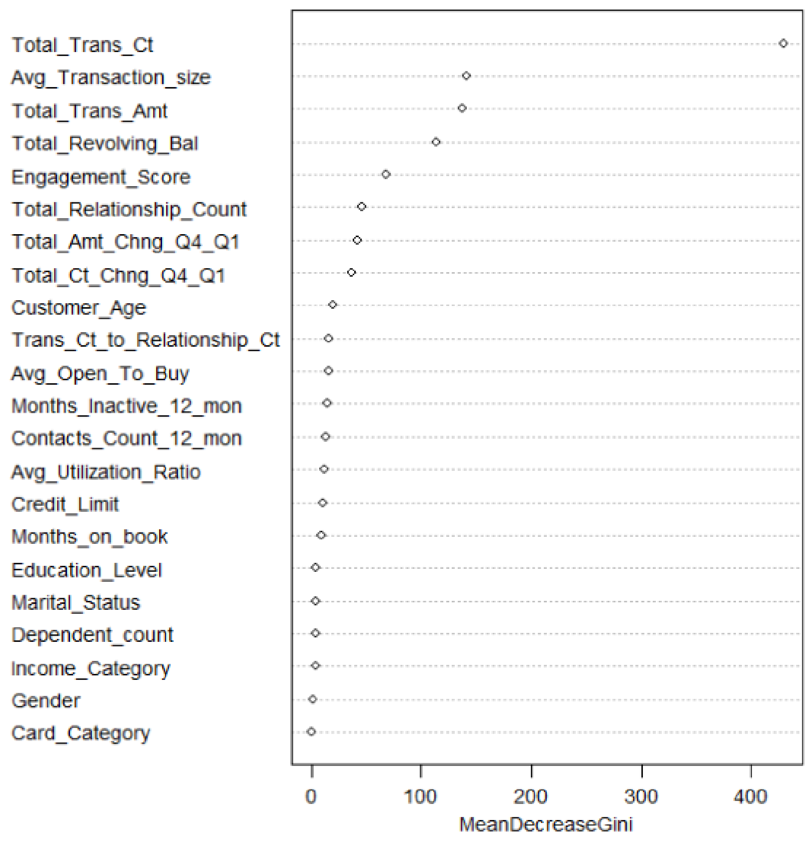


Figure 14 *shows the test error curve for the Random Forest model between 0 and 1,000 trees using the default values for mtry and nodesize.*

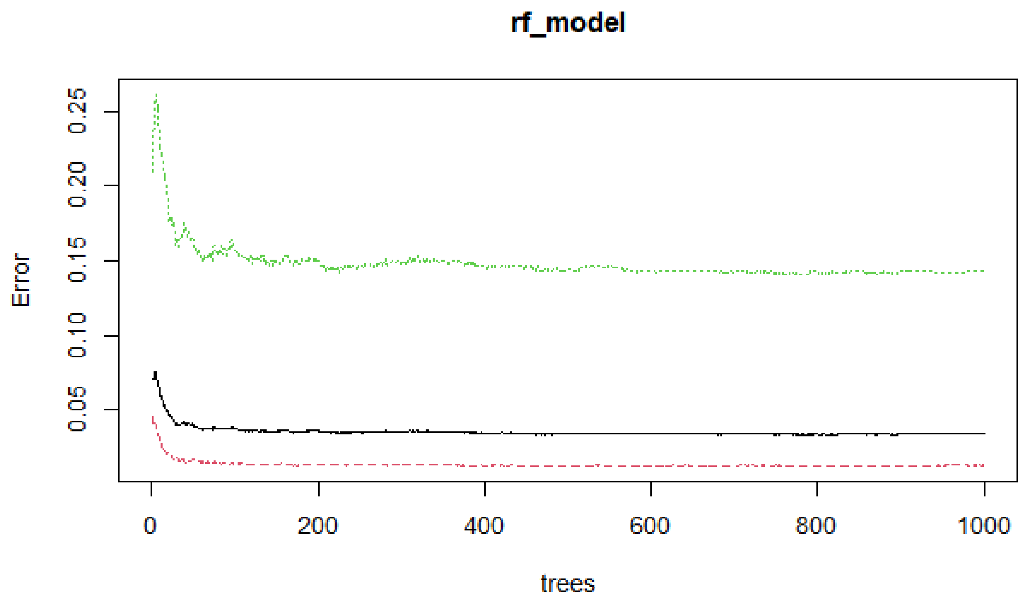


Figure 15 *shows the test error curve for the Random Forest model between 0 and 200 trees using the default values for mtry and nodesize.*

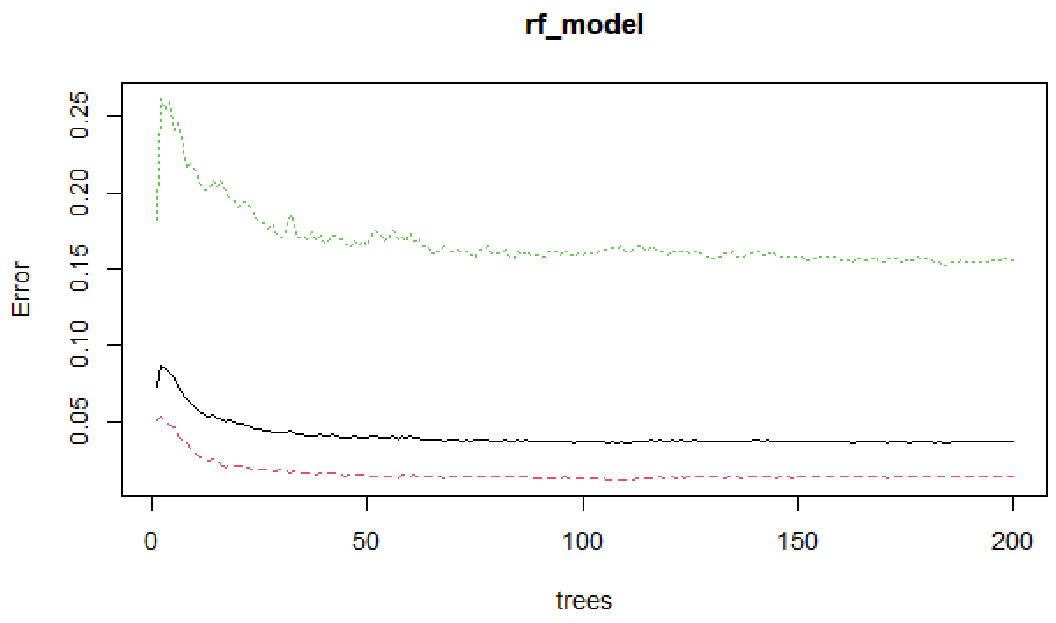


Figure 16 *shows the test error curve for the Random Forest model between 0 and 50 trees using the default values for mtry and nodesize.*

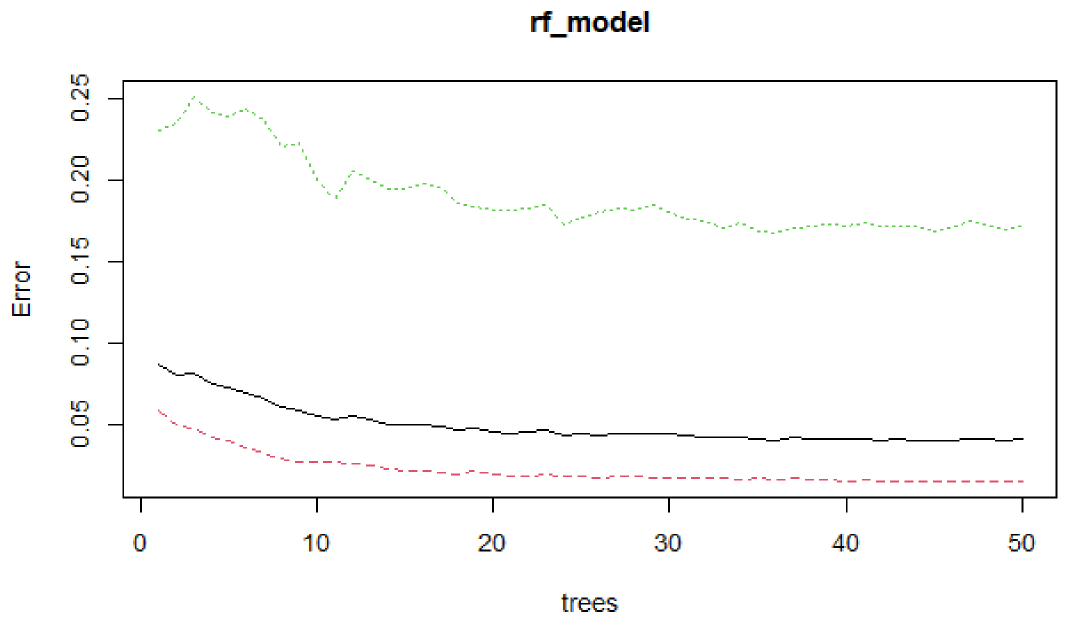


Figure 17 *shows the test error curve for the Random Forest model between 0 and 50 trees using the default values for mtry and nodesize, with the first occurrence where the OOB Error decreases by less than 0.0005 by the dashed, vertical, blue line.*

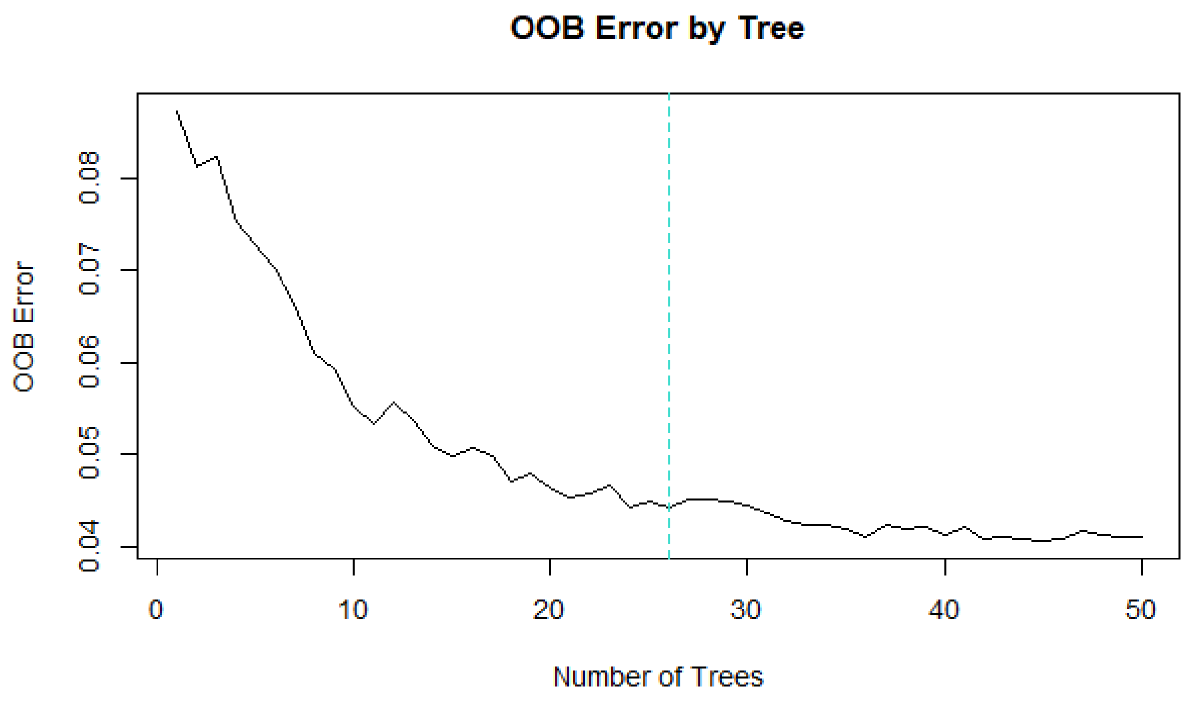


***Figure 18** shows the OOB test error curve for the Random Forest model versus nodesize.*

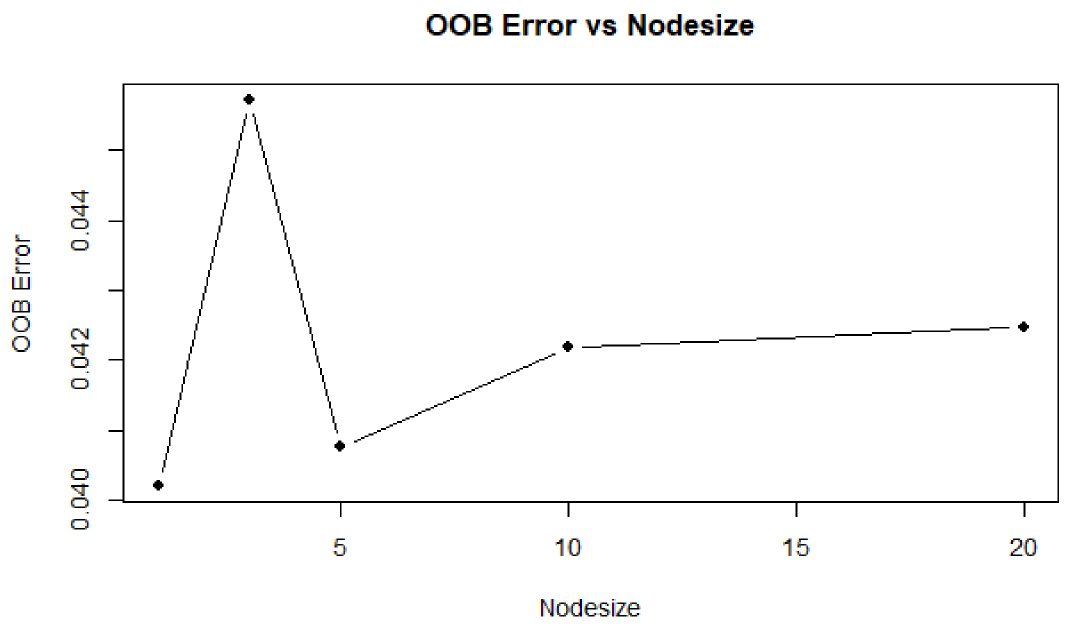


***Figure 19** shows the graph of the ROC Curve for Random Forest Model 2 (with tuned hyperparameters) .*

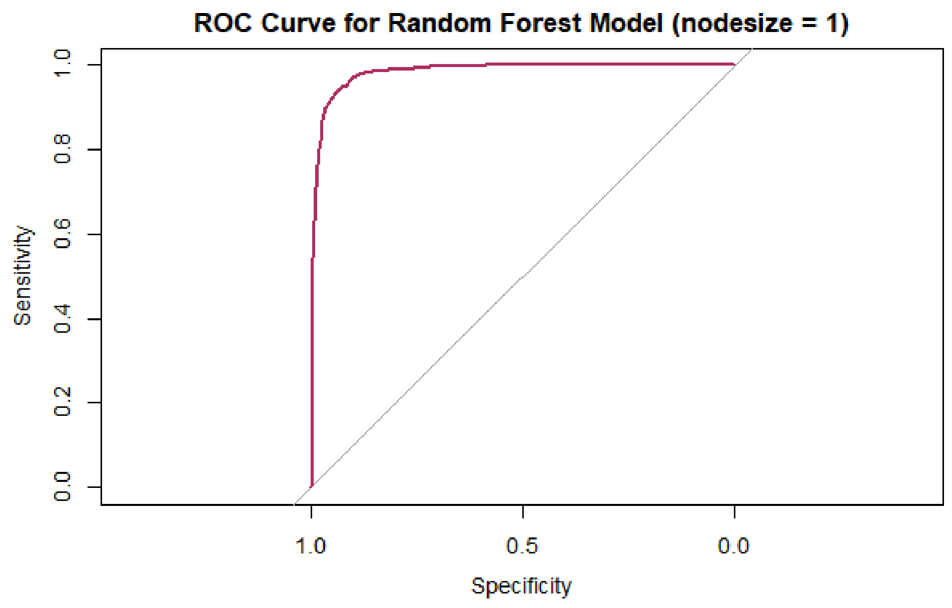


Figure 20 *shows variable importance plot Random Forest Model 2 (with tuned hyperparameter) based on the Mean Decrease Accuracy (MDA).*

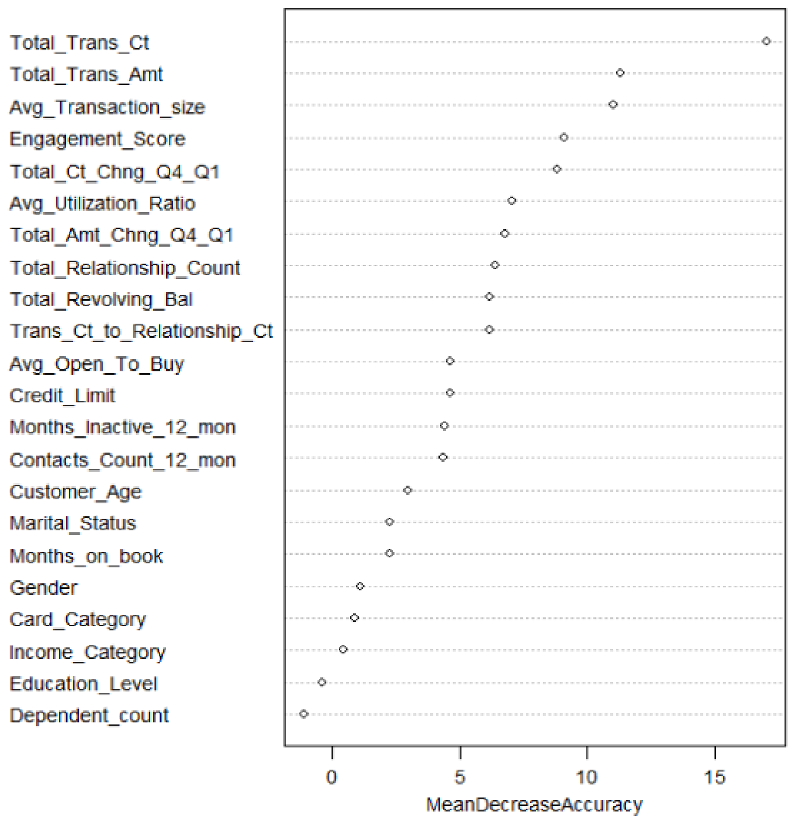


Figure 21 *shows variable importance plot Random Forest Model 2 (with tuned hyperparameter) based on the Mean Decrease Gini.*

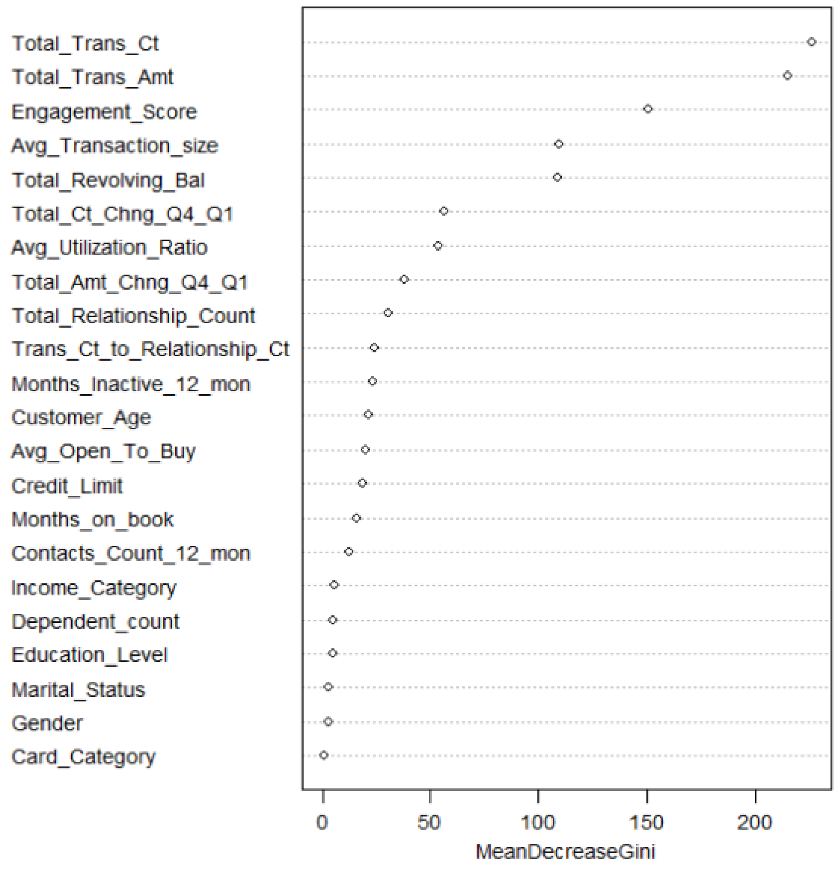


Figure 22 *shows the relative influence plot for Gradient Boosting Model 1 (default parameter values).*

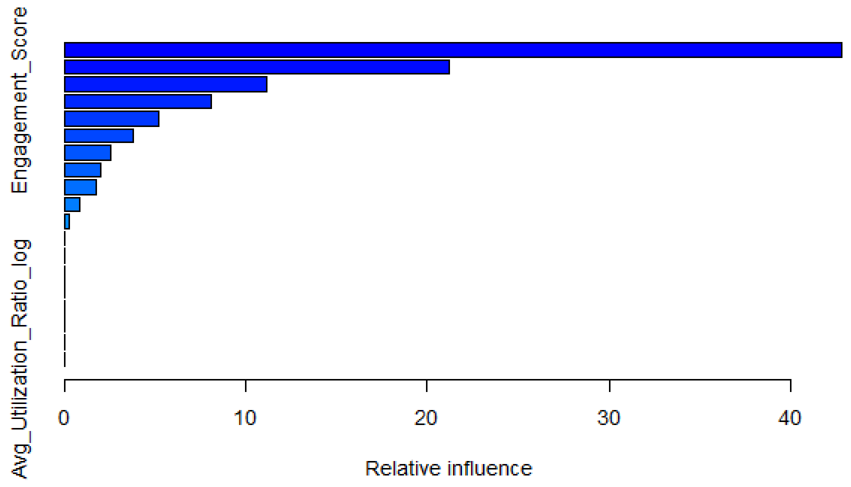


Figure 23 *shows the Bernoulli deviance for iterations of Gradient Boosting Model 1 with n.trees between 0 and 100.*

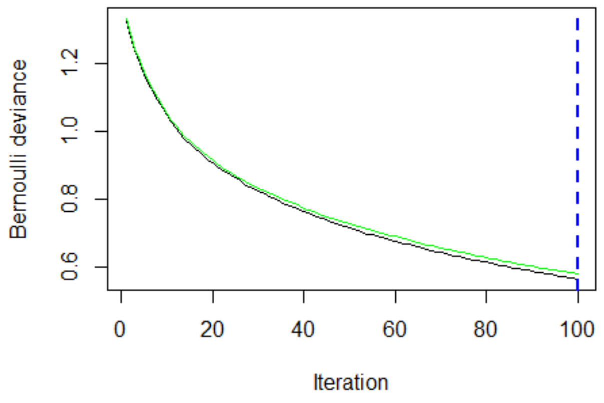


***Figure 24** shows the relative influence plot for Gradient Boosting Model 2 (with tuned hyperparameters).*

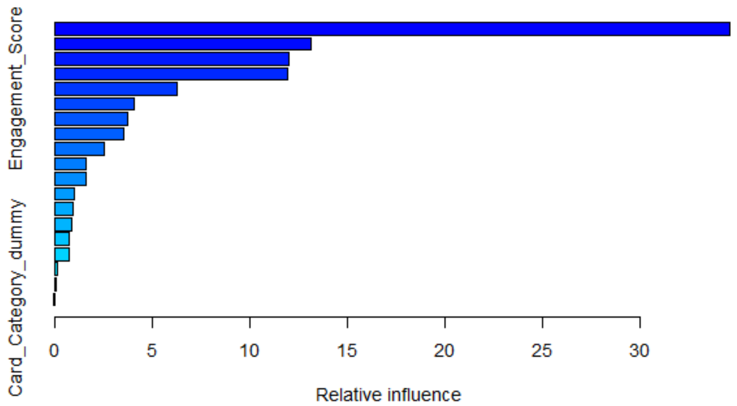
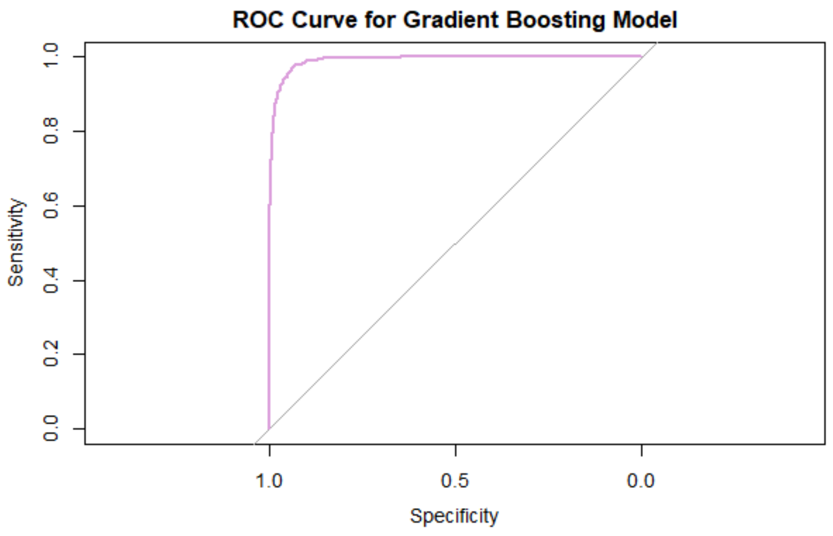


***Figure 25** shows the graph of the ROC Curve for Gradient Boosting Model 2 (with tuned hyperparameters) .*



11 References

James, G., Witten, D., Hastie, T., Tibshirani, R. (2021). *An Introduction to Statistical Learning with Applications in R.* Springer.