Project 1: Santander Customer Satisfaction

Jesus Casas, Victoria Miteva, Luis Olmos, Abdul Hasib Safi, and Luis Cedeno

Department of Mathematics

California State University Northridge

Math 445

Dr. Adriano Zambom

May 5, 2025

1. Problem Overview

For this project, our group was tasked to solve the challenge posed by the Santander Customer Satisfaction Kaggle competition — to predict customer dissatisfaction using an anonymized dataset provided by a Spanish bank. The dataset contains 76,020 total observations and 371 numerical features, with the target variable TARGET indicating whether a customer is dissatisfied (1) or satisfied (0) (**Appendix 1**). Due to the anonymized nature of the feature names, the analysis focuses on uncovering statistical relationships and data-driven patterns rather than relying on domain-specific knowledge. The central task is to apply the methods learned in class throughout the semester to build and evaluate machine learning models capable of accurately identifying dissatisfied customers despite severe class imbalance and the absence of contextual feature descriptions.

2. Exploratory Data Analysis & Data Preprocessing

Before training our models, we conducted a thorough exploration of the data and applied key preprocessing steps to improve data quality, reduce redundancy, and address the class imbalance in the target variable. Our focus was on identifying structural issues like imbalance, duplicated or constant features, and multicollinearity — which can be found in our summary of the Key Findings from EDA (**Appendix 2**).

2.1 Initial Data Understanding

The dataset includes 371 anonymized numeric features. As shown in the structure screenshot (**Appendix 3**), many of the columns are operational or financial indicators with values often concentrated around zero.

The response variable TARGET is highly imbalanced, with dissatisfied customers (TARGET = 1) representing less than 5% of the total observations. This imbalance is

visualized in the bar plot (**Appendix 4**). As a result, we focused on ROC-AUC and F1 Score as more informative evaluation metrics than simple accuracy. We also applied downsampling for some models to help prevent overfitting to the majority class.

No missing values were found in the dataset, so no imputation was required.

2.2 Feature Quality Check

To improve data quality and reduce redundancy: We first identified 62 duplicated features, which were removed. This brought the feature count down from 371 to 309 (**Appendix 5**). Next, we detected 34 constant columns — features that contain the same value for every observation and thus offer no predictive power. However, most of these overlapped with the previously removed duplicates. Only 1 additional constant column was removed at this step (**Appendix 6**). Constant columns are often the result of operational flags or empty financial transaction fields. Removing them prevents noise and reduces computational overhead.

2.3 Correlation Analysis

We computed the correlation matrix of all numeric features and found that 134 features had strong pairwise correlation ($r > 0.9$) (**Appendix 7**). Such high correlation introduces redundancy and can negatively affect model performance, particularly in linear models or distance-based algorithms like KNN.

These correlated features were removed to reduce multicollinearity. The visual impact is evident in the side-by-side correlation heatmap (**Appendix 8**).

2.4 Feature Importance

We ran preliminary feature importance analysis using Random Forest. While this was not used for explicit feature selection, it served as a useful validation step. The top

features identified included var15, var38, and several num_var indicators (**Appendix 9**) — confirming that meaningful signals remained in the dataset post-cleaning. We chose not to perform manual feature selection for two reasons:

1. Tree-based models like Random Forest and CART inherently handle feature selection by splitting only on relevant features.
2. Premature manual selection could have excluded useful interactions or penalized minority-class signals in such an imbalanced setting.

3. Modeling Approach

Based on the findings from our exploratory data analysis, we selected four supervised learning models to evaluate on the dataset: CART (Classification and Regression Trees), K-Nearest Neighbors (KNN), Random Forest (RF), and Extreme Gradient Boosting (XGBoost). These models were selected to balance simplicity and performance, with consistent sampling strategies used to address class imbalance where needed.

3.1 CART

To establish a baseline, we trained a CART model on the downsampled dataset (2,400 samples per class; see **Appendix 10**) and evaluated it on a held-out validation set. The model achieved a Validation AUC of 0.809 and Accuracy of 0.7474 (**Appendix 11**) and a Kaggle ROC of 0.79671 (**Appendix 13**), showing that downsampling improved minority class sensitivity without sacrificing overall performance. The resulting tree structure (**Appendix 12**) also offered interpretability and implicit feature selection.

3.2 Random Forest

We trained a Random Forest classifier (200 trees, mtry=106), which yielded a low OOB error (4.24%, **Appendix 14**), mainly due to accurate prediction of the majority

class. However, validation results indicated poor identification of dissatisfied customers (F1: 0.0626, Precision: 0.2286, Recall: 0.0362, **Appendix 14**). While the ROC curve (**Appendix 15**) suggested moderate ranking ability, the low F1 confirmed underperformance in hard classification due to class imbalance. The error rate plateaued before 200 trees (**Appendix 16**), consistent with diminishing returns from increasing tree count. These findings suggest that while Random Forest ranks predictions adequately, class weighting or resampling might improve hard classification. Despite weak threshold-based validation performance, the Kaggle submission achieved a ROC AUC of 0.79886, confirming good ranking ability (**Appendix 17**).

3.3 KNN

We implemented a KNN classifier after normalizing all features. Initial experiments showed that model performance was sensitive to the number of neighbors (k), so we tested values from 170 to 200. As shown in **Appendix 18**, the ROC score gradually improved with larger k, and we ultimately selected k = 201 for the final model.

Before applying downsampling, KNN produced balanced validation metrics with an F1 Score, Precision, and Recall all equal to 0.6208 (**Appendix 19**) and a Kaggle-submitted ROC score of 0.80794 (**Appendix 20, 21**). After applying downsampling to improve F1 metrics, F1 Score increased to 0.6437, Precision to 0.5861, and Recall to 0.7139 (**Appendix 22**). However, this increase in F1 came at the cost of reduced ranking performance on the test set: while the original model achieved a Kaggle ROC AUC of 0.80794 (**Appendix 21**), the downsampled version dropped significantly to 0.49993 (**Appendix 23**). This highlights that tuning for threshold-based metrics like F1 can hurt the model's ability to rank predictions globally, as measured by AUC.

3.4 XGBoost

XGBoost was the strongest-performing model overall. To account for the severe class imbalance in the dataset, we used the scale_pos_weight parameter, set based on the class ratio in the training data. The model was trained with max_depth = 6, a learning rate (eta) of 0.05, and early stopping after 20 rounds.

On the validation set, XGBoost achieved an impressive AUC on Colab of 0.8477, as shown in **Appendix 24**. However, the initial F1 score at the default 0.5 threshold was extremely low (0.0065). We then performed threshold tuning to maximize the F1 score, identifying a best threshold of 0.77, which improved the F1 to 0.2962 (**Appendix 25**).

XGBoost also delivered the highest Kaggle ROC AUC score among all models: 0.83535 (**Appendix 26**), demonstrating strong generalization performance despite the imbalanced data.

4. Results & Conclusions

Among the four models tested, XGBoost achieved the highest overall performance, with a validation AUC of 0.8477 and a Kaggle ROC AUC of 0.83535. While its default F1 score was low, threshold tuning significantly improved classification performance. KNN, particularly after downsampling, also performed well, achieving a Kaggle ROC AUC of 0.80794 and strong balance between precision and recall. Random Forest demonstrated good ranking ability but underperformed on minority class detection without resampling. CART, though simple, benefited noticeably from downsampling and served as a strong, interpretable baseline.

Handling class imbalance for certain models was critical to achieving useful results. XGBoost proved to be the most effective overall, both in terms of validation and leaderboard performance.

Appendix

- **Appendix 1.** [Santander Customer Satisfaction - Kaggle]

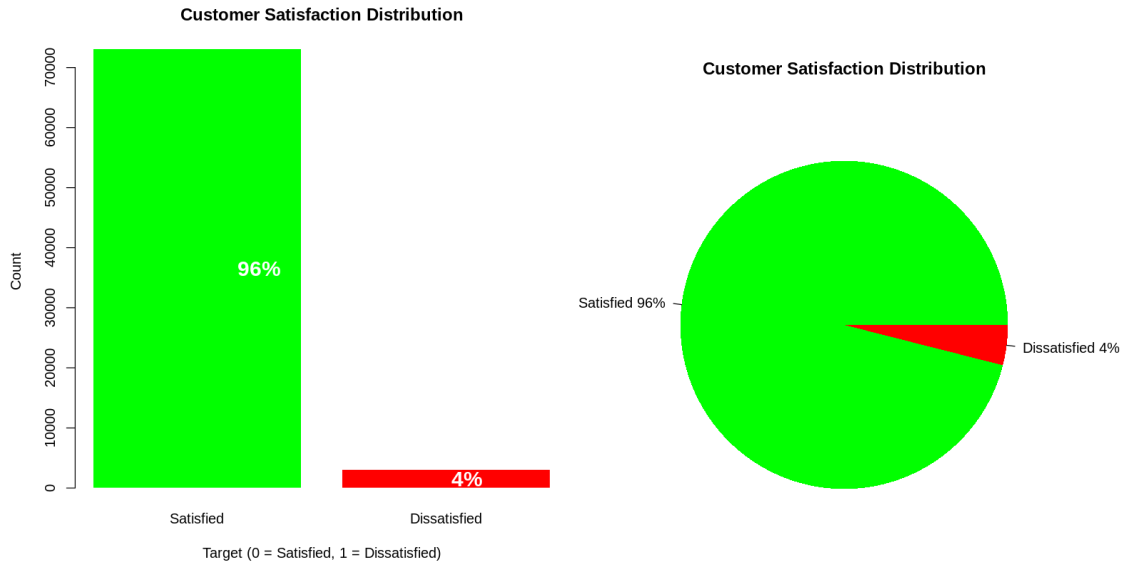
- **Appendix 2.** Key Findings from EDA

```
Key Findings from EDA:
1. Class Imbalance: 24.27 to 1
2. Duplicate Columns: 62
3. Constant Columns: 34
4. Missing Values: 0 columns have missing values
```

- **Appendix 3.** Structure of Data

```
76020 obs. of  371 variables:
$ ID                              : int  1 3 4 8 10 13 14 18 20 23 ...
$ var3                            : int  2 2 2 2 2 2 2 2 2 ...
$ var15                           : int  23 34 23 37 39 23 27 26 45 25
$ imp_ent_var16_ult1              : num  0 0 0 0 0 0 0 0 0 ...
$ imp_op_var39_comer_ult1         : num  0 0 0 195 0 0 0 0 0 0 ...
$ imp_op_var39_comer_ult3         : num  0 0 0 195 0 0 0 0 0 0 ...
$ imp_op_var40_comer_ult1         : num  0 0 0 0 0 0 0 0 0 0 ...
$ imp_op_var40_comer_ult3         : num  0 0 0 0 0 0 0 0 0 0 ...
$ imp_op_var40_efect_ult1         : num  0 0 0 0 0 0 0 0 0 0 ...
$ imp_op_var40_efect_ult3         : num  0 0 0 0 0 0 0 0 0 0 ...
$ imp_op_var40_ult1               : num  0 0 0 0 0 0 0 0 0 0 ...
```

- **Appendix 4**. Distribution Bar Plot and Pie Chart of TARGET Variable



❖ **Appendix 5**. List of Duplicate Features

```
Number of duplicate columns: 62
Duplicate columns identified:
 [1] "ind_var2"                      "ind_var13_medio"
 [3] "ind_var18"                     "ind_var26"
 [5] "ind_var25"                     "ind_var27_0"
 [7] "ind_var28_0"                   "ind_var28"
 [9] "ind_var27"                     "ind_var29_0"
[11] "ind_var29"                     "ind_var32"
[13] "ind_var34"                     "ind_var37"
[15] "ind_var41"                     "ind_var39"
[17] "ind_var46_0"                   "ind_var46"
[19] "num_var13_medio"               "num_var18"
[21] "num_var26"                     "num_var25"
[23] "num_var27_0"                   "num_var28_0"
[25] "num_var28"                     "num_var27"
[27] "num_var29_0"                   "num_var29"
[29] "num_var32"                     "num_var34"
[31] "num_var37"                     "num_var41"
[33] "num_var39"                     "num_var46_0"
[35] "num_var46"                     "saldo_var28"
[37] "saldo_var27"                   "saldo_var29"
[39] "saldo_var41"                   "saldo_var46"
[41] "delta_num_reemb_var13_1y3"     "delta_num_reemb_var17_1y3"
[43] "delta_num_reemb_var33_1y3"     "delta_num_trasp_var17_in_1y3"
[45] "delta_num_trasp_var17_out_1y3" "delta_num_trasp_var33_in_1y3"
[47] "delta_num_trasp_var33_out_1y3" "imp_amort_var18_hace3"
[49] "imp_amort_var34_hace3"         "imp_reemb_var13_hace3"
[51] "imp_reemb_var33_hace3"         "imp_trasp_var17_out_hace3"
[53] "imp_trasp_var33_out_hace3"     "num_var2_0_ult1"
[55] "num_var2_ult1"                 "num_reemb_var13_hace3"
[57] "num_reemb_var33_hace3"         "num_trasp_var17_out_hace3"
[59] "num_trasp_var33_out_hace3"     "saldo_var2_ult1"
[61] "saldo_medio_var13_medio_hace3" "saldo_medio_var13_medio_ult1"
```

```
Original dimensions: 76020 rows x 371 columns
Dimensions after removing duplicates: 76020 rows x 309 columns
```

❖ **Appendix 6**. List of Constant Features

```
Number of constant columns: 34
Constant columns identified:
 [1] "ind_var2_0"                    "ind_var2"
 [3] "ind_var27_0"                   "ind_var28_0"
 [5] "ind_var28"                     "ind_var27"
 [7] "ind_var41"                     "ind_var46_0"
 [9] "ind_var46"                     "num_var27_0"
[11] "num_var28_0"                   "num_var28"
[13] "num_var27"                     "num_var41"
[15] "num_var46_0"                   "num_var46"
[17] "saldo_var28"                   "saldo_var27"
[19] "saldo_var41"                   "saldo_var46"
[21] "imp_amort_var18_hace3"         "imp_amort_var34_hace3"
[23] "imp_reemb_var13_hace3"         "imp_reemb_var33_hace3"
[25] "imp_trasp_var17_out_hace3"     "imp_trasp_var33_out_hace3"
[27] "num_var2_0_ult1"               "num_var2_ult1"
[29] "num_reemb_var13_hace3"         "num_reemb_var33_hace3"
[31] "num_trasp_var17_out_hace3"     "num_trasp_var33_out_hace3"
[33] "saldo_var2_ult1"               "saldo_medio_var13_medio_hace3"
```

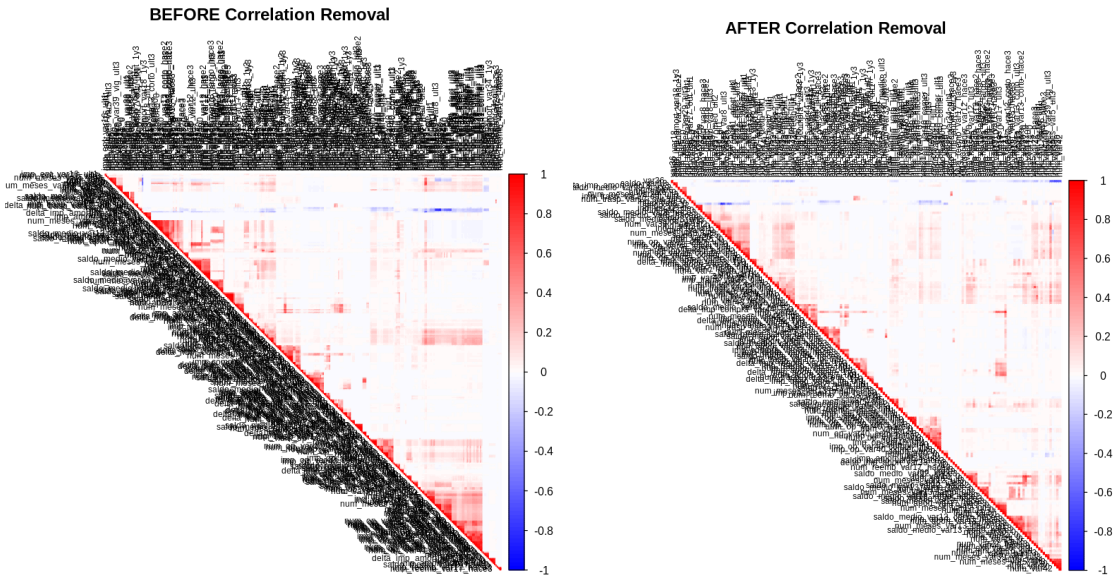
```
TRAIN dimensions no dupes: 76020 rows x 309 columns
TRAIN Dimensions after removing constants: 76020 rows x 308 columns
```

❖ **Appendix 7**. List of Highly Correlated Features

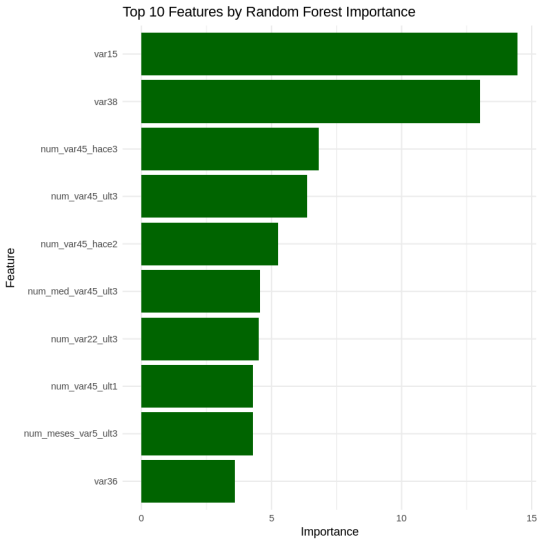
```
Number of highly correlated features (r > 0.9): 134
```

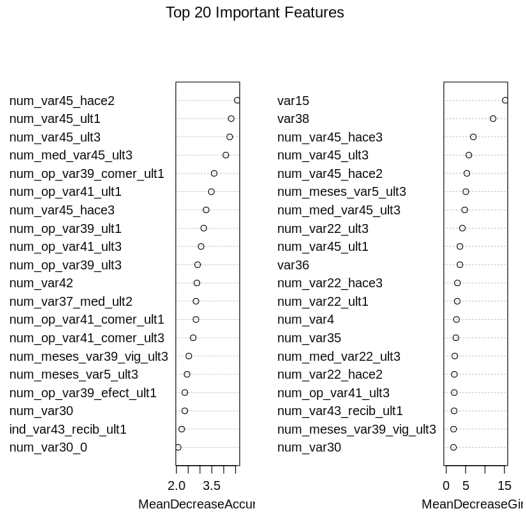
[1]	"imp_op_var39_efect_ult1"	"imp_op_var39_efect_ult3"
[3]	"imp_op_var39_ult1"	"ind_var8"
[5]	"ind_var24"	"ind_var26_0"
[7]	"ind_var26_cte"	"ind_var32_0"
[9]	"ind_var33"	"num_var1_0"
[11]	"num_var1"	"num_var5"
[13]	"num_var8"	"num_var12"
[15]	"num_var13_0"	"num_var13_corto_0"
[17]	"num_var13_largo"	"num_var13"
[19]	"num_var14"	"num_var24"
[21]	"num_op_var41_ult3"	"num_op_var39_hace2"
[23]	"num_op_var39_hace3"	"num_op_var39_ult1"
[25]	"num_op_var39_ult3"	"num_var31_0"
[27]	"num_var33_0"	"num_var33"
[29]	"num_var35"	"num_var37_0"
[31]	"num_var41_0"	"num_var44_0"
[33]	"saldo_var6"	"saldo_var31"
[35]	"saldo_var42"	"delta_imp_amort_var34_1y3"
[37]	"delta_num_aport_var33_1y3"	"delta_num_compra_var44_1y3"
[39]	"delta_num_venta_var44_1y3"	"imp_amort_var18_ult1"
[41]	"ind_var10cte_ult1"	"ind_var9_cte_ult1"
[43]	"ind_var9_ult1"	"num_op_var39_comer_ult3"
[45]	"num_op_var41_comer_ult3"	"num_op_var41_efect_ult3"
[47]	"num_op_var39_efect_ult1"	"num_op_var39_efect_ult3"
[49]	"num_trasp_var17_in_hace3"	"num_trasp_var33_in_ult1"
[51]	"num_venta_var44_hace3"	"num_var45_ult3"
[53]	"saldo_medio_var8_ult1"	"saldo_medio_var12_ult1"
[55]	"saldo_medio_var12_ult3"	"saldo_medio_var13_corto_ult1"
[57]	"saldo_medio_var13_corto_ult3"	"saldo_medio_var13_largo_ult1"
[59]	"saldo_medio_var13_largo_ult3"	"saldo_medio_var13_medio_hace2"
[61]	"saldo_medio_var13_medio_ult3"	"saldo_medio_var17_hace2"
[63]	"saldo_medio_var17_ult1"	"saldo_medio_var17_ult3"
[65]	"saldo_medio_var29_ult1"	"saldo_medio_var29_ult3"
[67]	"saldo_medio_var33_ult3"	"saldo_medio_var44_ult1"
[69]	"saldo_medio_var44_ult3"	"imp_op_var39_comer_ult1"
[71]	"imp_op_var39_comer_ult3"	"ind_var13_0"
[73]	"ind_var13_corto_0"	"ind_var13_largo_0"
[75]	"ind_var12"	"ind_var25_cte"
[77]	"ind_var31_0"	"ind_var37_cte"
[79]	"ind_var1_0"	"ind_var1"
[81]	"ind_var39_0"	"ind_var44_0"
[83]	"ind_var5_0"	"ind_var6_0"
[85]	"ind_var6"	"ind_var8_0"
[87]	"ind_var13_corto"	"ind_var13_largo"
[89]	"ind_var13_medio_0"	"ind_var18_0"
[91]	"ind_var20_0"	"ind_var20"
[93]	"ind_var24_0"	"num_var26_0"
[95]	"num_op_var40_ult1"	"ind_var34_0"
[97]	"ind_var40_0"	"ind_var40"
[99]	"num_var39_0"	"num_var12_0"
[101]	"ind_var44"	"saldo_var1"
[103]	"saldo_var12"	"saldo_var26"
[105]	"num_var34_0"	"num_var18_0"
[107]	"delta_imp_aport_var13_1y3"	"delta_imp_aport_var17_1y3"
[109]	"saldo_var34"	"saldo_var17"
[111]	"delta_imp_reemb_var33_1y3"	"delta_imp_trasp_var33_out_1y3
[113]	"imp_compra_var44_ult1"	"ind_var7_emit_ult1"
[115]	"num_var22_ult3"	"ind_var5"
[117]	"num_var13_corto"	"num_var13_medio_0"
[119]	"ind_var17"	"num_op_var41_ult1"
[121]	"num_op_var40_comer_ult1"	"num_op_var39_comer_ult1"
[123]	"delta_imp_reemb_var13_1y3"	"imp_reemb_var17_hace3"
[125]	"imp_reemb_var33_ult1"	"delta_imp_trasp_var17_in_1y3
[127]	"delta_imp_trasp_var17_out_1y3"	"imp_trasp_var33_in_hace3"
[129]	"imp_trasp_var33_out_ult1"	"num_med_var45_ult3"
[131]	"saldo_var24"	"saldo_var13_corto"
[133]	"imp_aport_var17_hace3"	"saldo_var33"

❖ **Appendix 8**. Correlation Heatmap Before & After Removal



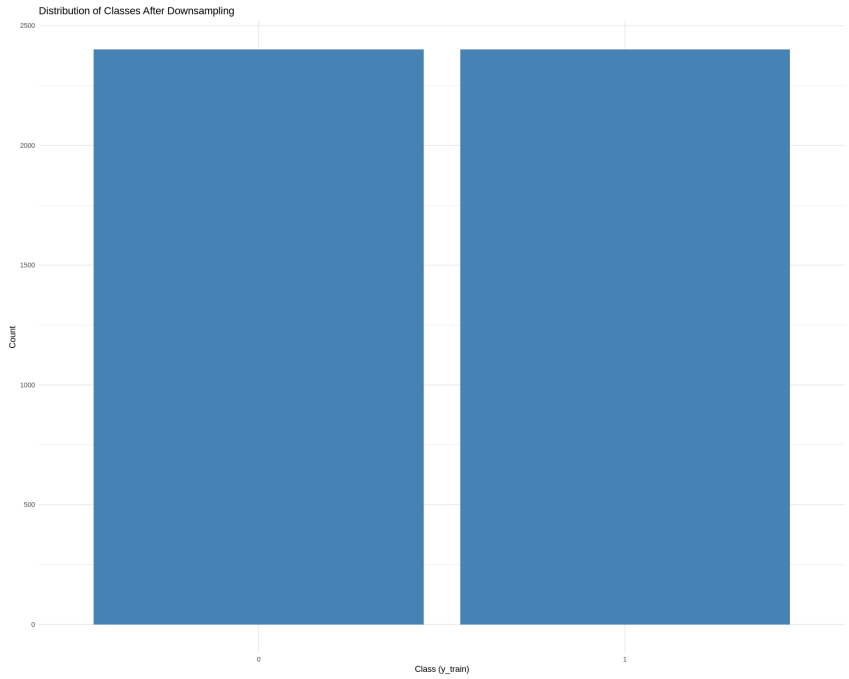
❖ **Appendix 9**. Bar Chart and Dot Plot Visualizing Feature Importance (Top 20)





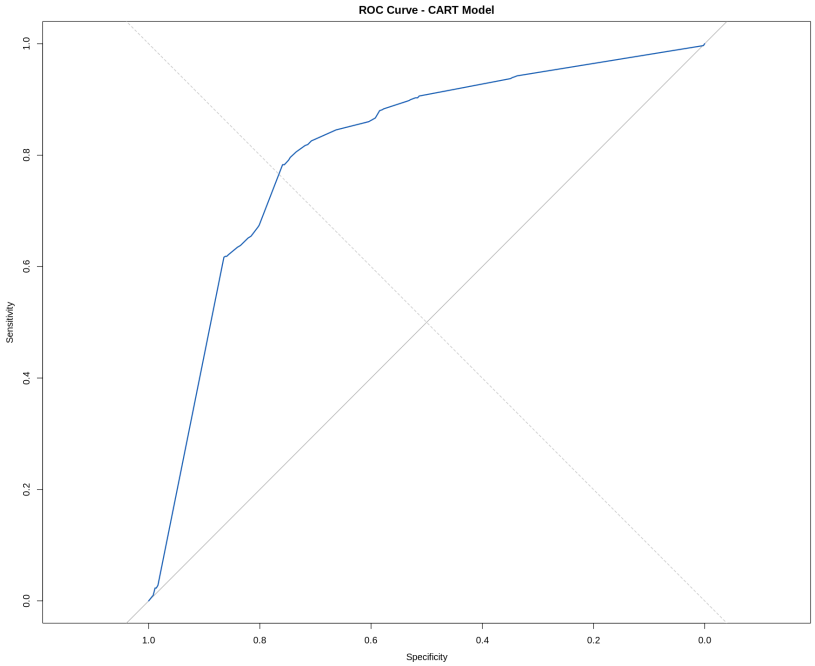

```
[1] "Top 20 important variables:"
                            0           1 MeanDecreaseAccuracy
cluster_dist_7         38.90762 -32.3206844             37.59595
var38                  35.60608  -0.4269644             36.50347
cluster_dist_1         34.05278 -30.3772409             33.22062
cluster_dist_6         34.08767 -37.0362577             32.85168
cluster_dist_12        33.00041 -32.5305544             31.97705
cluster_dist_3         32.66843 -32.4397306             31.73565
cluster_dist_11        31.65748 -26.9844062             30.91629
cluster_dist_5         31.45467 -25.6169273             30.58462
cluster_dist_4         29.50040 -16.9392195             29.00712
cluster_dist_2         29.40515 -32.1681165             28.51962
cluster_dist_9         29.32932 -33.5846981             28.39499
cluster_dist_8         28.76511 -34.0725189             27.81039
var15                  26.61646  -6.8634566             26.90287
saldo_var30            25.93223  -9.7519557             26.13823
cluster_dist_10        24.97424 -16.8706364             24.67420
num_var45_hace2        23.29260 -14.9188143             22.66916
num_var22_hace3        21.46048 -12.2119876             20.65383
num_var45_hace3        21.46482 -21.7732182             20.54274
num_var45_ult1         19.94517 -12.4880691             19.60757
saldo_medio_var5_hace3 19.07968  -4.8095957             19.38556
                       MeanDecreaseGini
cluster_dist_7                294.30236
var38                         486.85882
```

❖ **Appendix 10**. CART Model TARGET Distribution After Downsampling

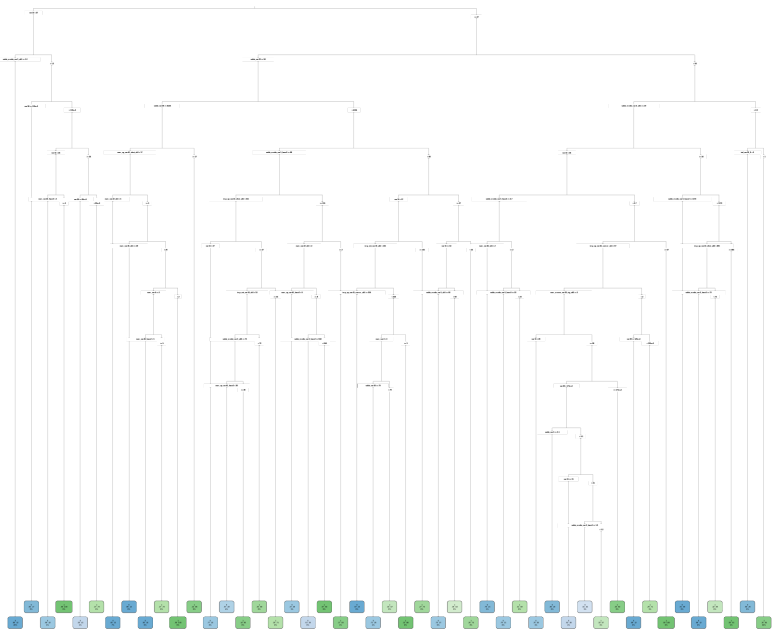


❖ **Appendix 11**. CART Model Evaluation: AUC-ROC

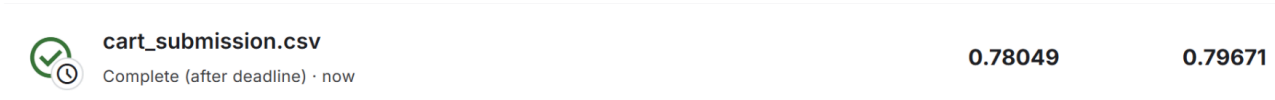
```
CART Model Validation AUC: 0.809
CART Model Validation Accuracy: 0.7474
```



❖ **Appendix 12**. Final CART Tree Structure



❖ **Appendix 13**. CART Kaggle Submission ROC Score



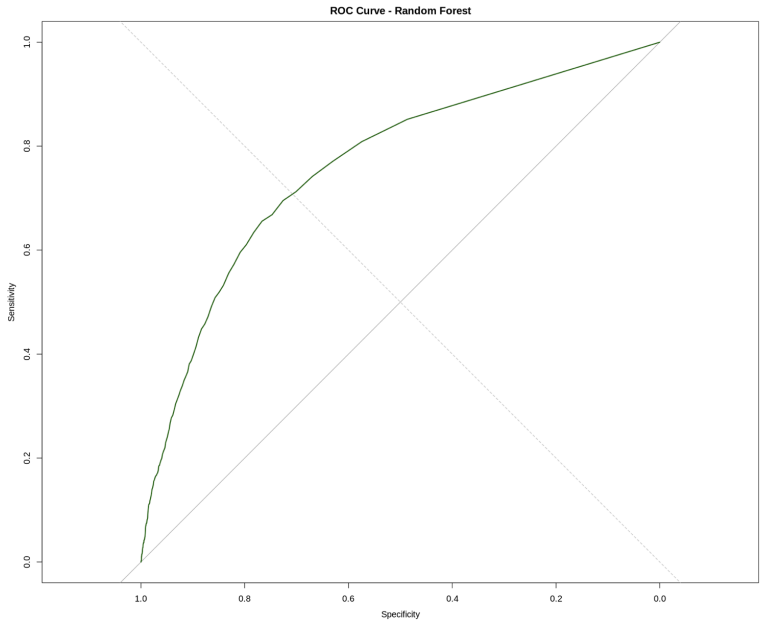
❖ **Appendix 14**. Random Forest Summary: ntrees, OOB Error, F1 Score

```
ntree      OOB      1      2
   10:   5.35%  1.71% 93.01%
   20:   4.75%  0.99% 95.20%
   30:   4.56%  0.76% 95.76%
   40:   4.51%  0.70% 96.14%
   50:   4.41%  0.60% 96.19%
   60:   4.39%  0.57% 96.28%
   70:   4.37%  0.54% 96.47%
   80:   4.37%  0.53% 96.71%
   90:   4.37%  0.52% 96.75%
  100:   4.36%  0.50% 96.99%
  110:   4.36%  0.50% 97.22%
  120:   4.34%  0.48% 97.27%
  130:   4.33%  0.47% 97.22%
  140:   4.32%  0.46% 97.13%
  150:   4.33%  0.46% 97.32%
  160:   4.33%  0.46% 97.36%
  170:   4.32%  0.45% 97.27%
  180:   4.30%  0.44% 97.18%
  190:   4.31%  0.45% 97.13%
  200:   4.31%  0.44% 97.46%
RF Validation F1 Score: 0.0626
RF Validation Precision: 0.2286
RF Validation Recall: 0.0362
          Reference
Prediction     0     1
         0 21815   851
         1   108    32
```

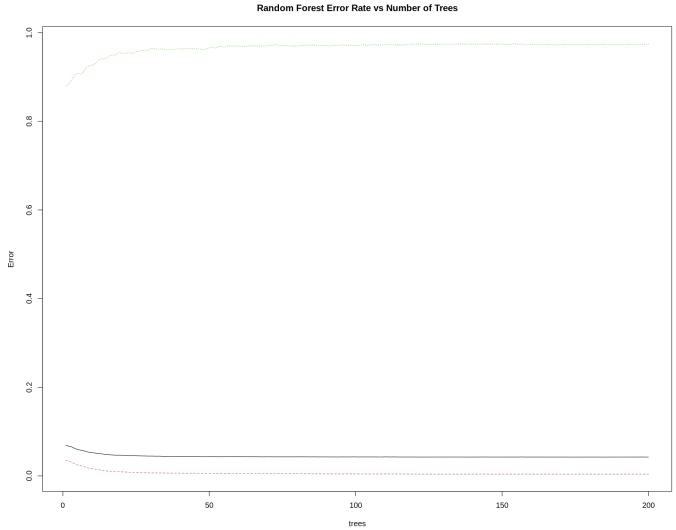
```
                Type of random forest: classification
                      Number of trees: 200
No. of variables tried at each split: 106

        OOB estimate of  error rate: 4.24%
Confusion matrix:
      0   1 class.error
0 72717 295 0.004040432
1  2931  77 0.974401596
```

❖ **Appendix 15**. Random Forest ROC Curve

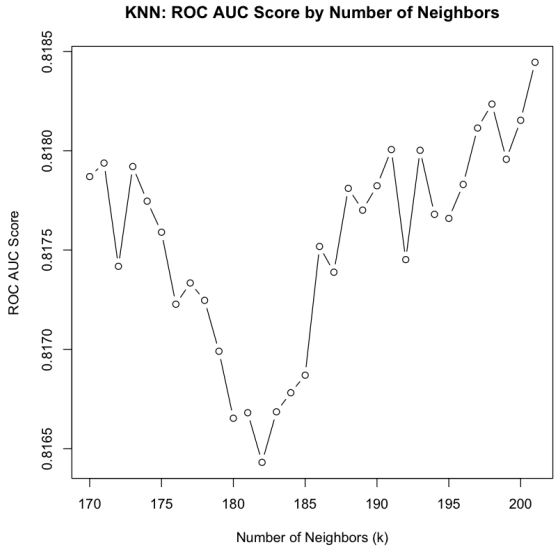


❖ **Appendix 16**. Random Forest: Error Rate vs Number of Trees



❖ **Appendix 17**. Random Forest Kaggle Submission ROC Score

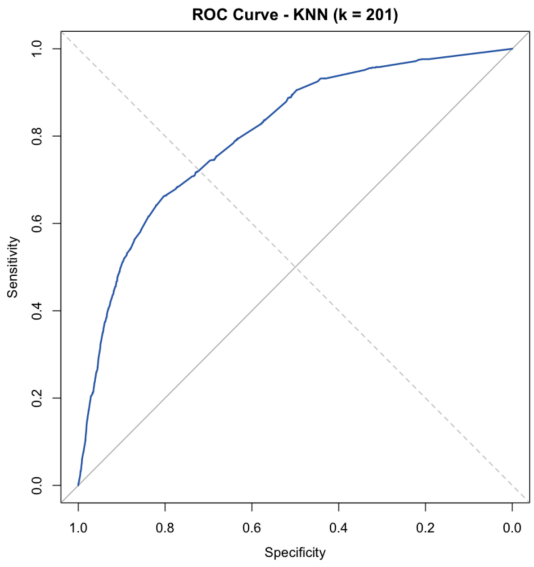
❖ **Appendix 18**. KNN Optimal Number of Neighbors (k)



❖ **Appendix 19**. KNN F1 Score Before Downsampling

```
Validation Accuracy: 0.6208
Validation F1 Score: 0.6208
Validation Precision: 0.6208
Validation Recall: 0.6208
          Reference
Prediction   0   1
         0 447 273
         1 273 447
```

❖ **Appendix 20**. KNN ROC Curve Before Downsampling



❖ **Appendix 21**. KNN Kaggle ROC Score Before Downsampling

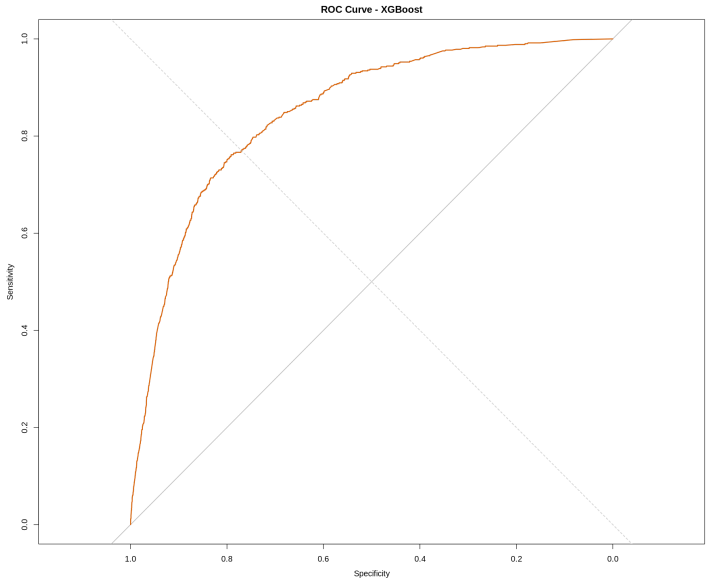
❖ **Appendix 22**. KNN F1 Score After Downsampling

```
Validation F1 Score: 0.6437
Validation Precision: 0.5861
Validation Recall: 0.7139
          Reference
Prediction   0   1
         0 357 206
         1 363 514
```

❖ **Appendix 23**. KNN Kaggle Submission ROC Score After Downsampling

❖ **Appendix 24**. XGBoost Colab ROC AUC

```
Validation AUC: 0.8476792
```



- **Appendix 25**. XGBoost F1 Score Before & After Threshold Tuning

  Before:
  ```
  Validation F1 Score: 0.0065
  ```

  After:
  ```
  Best Threshold: 0.77
  Best F1 Score: 0.2962
  ```

- **Appendix 26**. XGBoost Kaggle Submission ROC Score

- **Appendix 27**. Code: Colab Notebook

  Project 1 Code

- **Appendix 28**. Presentation Slides

  Project1_SatantanderSlides