# Bank Customer Churn Prediction

December 24, 2024

### 0.1 A Data Analysis Project by Garrett Power

The dataset I will be using for this project was retreived from Kaggle at this link: https://www.kaggle.com/datasets/gauravtopre/bank-customer-churn-dataset/data.

In this project my goal is to predict whether or not a customer (or client) will leave the "ABC Multinational" bank, or simply to predict churn. The dataset consists of 10000 records with both categorical and numerical variables. These are the variables included in the dataset:
1. **customer_id**, an identification variable for each client.
2. **credit_score**, the credit score of that client.
3. **country**, the country of that client.
4. **gender**, that client's gender.
5. **age**, that client's age.
6. **tenure**, the amount of years that client has been with the bank.
7. **balance**, the account balance of that client.
8. **products_number**, how many bank products that client has.
9. **credit_card**, how many bank credit cards that client has.
10. **active_member**, whether or not that client is an active customer.
11. **estimated_salary**, that clients salary estimated.
12. **churn**, used as the target and marked by a 1 for a client having left the bank at some point or a 0 if they have not.

| | customer_id | credit_score | country | gender | age | tenure | balance | \ |
|---|---|---|---|---|---|---|---|---|
| 0 | 15634602 | 619 | France | Female | 42 | 2 | 0.00 | |
| 1 | 15647311 | 608 | Spain | Female | 41 | 1 | 83807.86 | |
| 2 | 15619304 | 502 | France | Female | 42 | 8 | 159660.80 | |
| 3 | 15701354 | 699 | France | Female | 39 | 1 | 0.00 | |
| 4 | 15737888 | 850 | Spain | Female | 43 | 2 | 125510.82 | |

| | products_number | credit_card | active_member | estimated_salary | churn |
|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 101348.88 | 1 |
| 1 | 1 | 0 | 1 | 112542.58 | 0 |
| 2 | 3 | 1 | 0 | 113931.57 | 1 |
| 3 | 2 | 0 | 0 | 93826.63 | 0 |
| 4 | 1 | 1 | 1 | 79084.10 | 0 |

## 0.2 Data Cleaning & Wrangling

This data had a lot going on and there were many ways to go about cleaning/wrangling it. I chose to start out by checking for nulls, dropping customer_id, encoding categorical variables like gender and country, and then separating the country column into another dataframe.

Null values:

```
False
```

| | credit_score | gender | age | tenure | balance | products_number | credit_card | \ |
|---|---|---|---|---|---|---|---|---|
| 0 | 619 | 0 | 42 | 2 | 0.00 | 1 | 1 | |
| 1 | 608 | 0 | 41 | 1 | 83807.86 | 1 | 0 | |
| 2 | 502 | 0 | 42 | 8 | 159660.80 | 3 | 1 | |
| 3 | 699 | 0 | 39 | 1 | 0.00 | 2 | 0 | |
| 4 | 850 | 0 | 43 | 2 | 125510.82 | 1 | 1 | |

| | active_member | estimated_salary | churn | country_Germany | country_Spain |
|---|---|---|---|---|---|
| 0 | 1 | 101348.88 | 1 | False | False |
| 1 | 1 | 112542.58 | 0 | False | True |
| 2 | 0 | 113931.57 | 1 | False | False |
| 3 | 0 | 93826.63 | 0 | False | False |
| 4 | 1 | 79084.10 | 0 | False | True |

*Remarks:* You can immediately see how the encoding went. Gender is now a binary 0 for female, or 1 for male. Each country was now split into a separate true/false boolean column, other than France which was marked by both Germany and Spain being labelled false. As I mentioned, I removed these country columns and placed them in a separate dataframe. There were many zero balance values in this dataset, which I will investigate further and consider missing data.

## 0.3 Exploratory Analysis and additional considerations

I chose to start with getting the summary statistics of the dataset, viewing the country distribution, and scaling the data for later use. Then I began plotting things such as histograms, density pairplots, and investigating the zero balance distribution.

| | credit_score | gender | age | tenure | balance | \ |
|---|---|---|---|---|---|---|
| count | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | |
| mean | 650.528800 | 0.545700 | 38.921800 | 5.012800 | 76485.889288 | |
| std | 96.653299 | 0.497932 | 10.487806 | 2.892174 | 62397.405202 | |
| min | 350.000000 | 0.000000 | 18.000000 | 0.000000 | 0.000000 | |
| 25% | 584.000000 | 0.000000 | 32.000000 | 3.000000 | 0.000000 | |
| 50% | 652.000000 | 1.000000 | 37.000000 | 5.000000 | 97198.540000 | |
| 75% | 718.000000 | 1.000000 | 44.000000 | 7.000000 | 127644.240000 | |
| max | 850.000000 | 1.000000 | 92.000000 | 10.000000 | 250898.090000 | |

| | products_number | credit_card | active_member | estimated_salary | \ |
|---|---|---|---|---|---|
| count | 10000.000000 | 10000.00000 | 10000.000000 | 10000.000000 | |
| mean | 1.530200 | 0.70550 | 0.515100 | 100090.239881 | |

```
std           0.581654      0.45584       0.499797      57510.492818
min           1.000000      0.00000       0.000000         11.580000
25%           1.000000      0.00000       0.000000      51002.110000
50%           1.000000      1.00000       1.000000     100193.915000
75%           2.000000      1.00000       1.000000     149388.247500
max           4.000000      1.00000       1.000000     199992.480000
```

```
              churn
count  10000.000000
mean       0.203700
std        0.402769
min        0.000000
25%        0.000000
50%        0.000000
75%        0.000000
max        1.000000
```

*Remarks:* The data looks to be relatively normal with 10000 records in each column, but the main thing that stuck out to me was the difference in ranges. All of the variables had rather different ranges other than the binary variables. I will continue feature engineering later in an attempt to improve model performance with things such as scaling, resampling, and imputation.

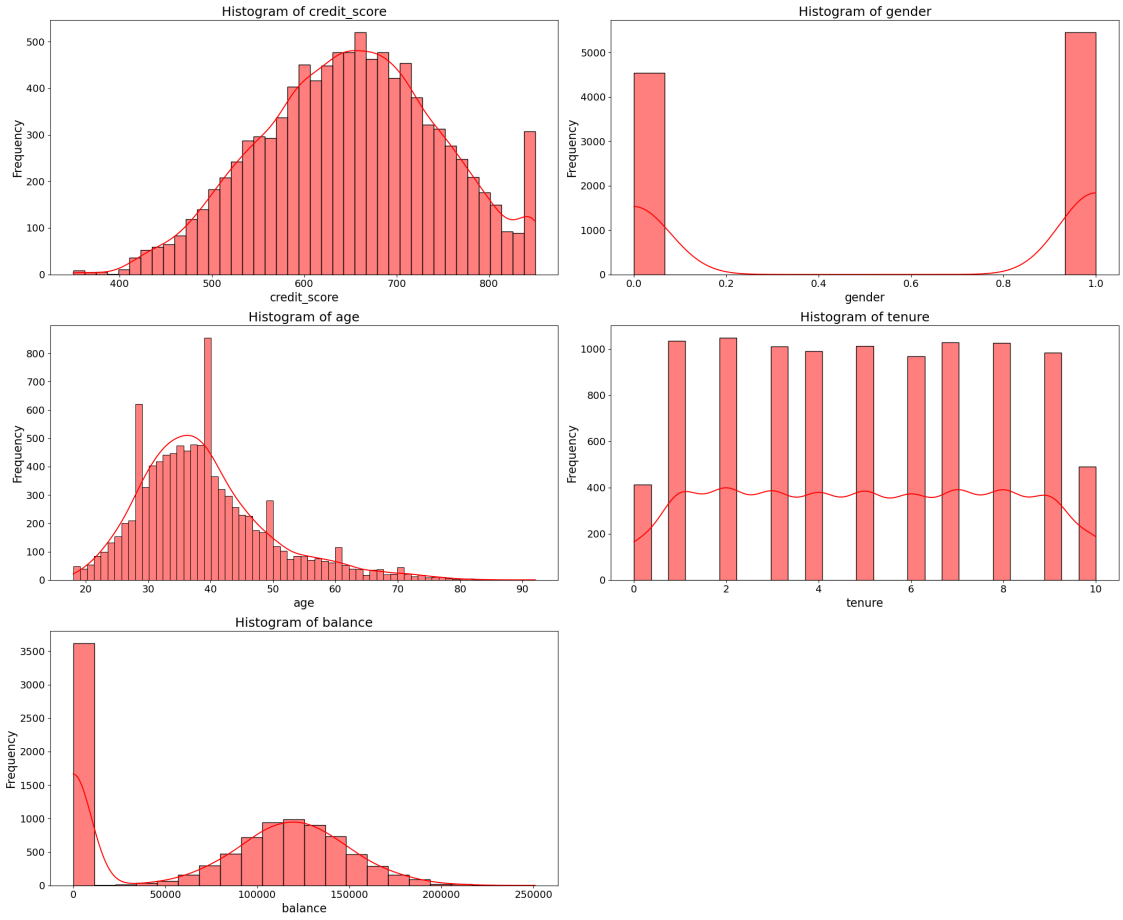
```
   Country  Count
0   France   5014
1  Germany   2509
2    Spain   2477
```

*Remarks:* Noticeably, France is contained in half of all records compared to the other two countries amounting to a quarter each. I summed the count of each country (because they were boolean variables) and turned them into numerical columns, also adding a specific France column to the dataframe with just countries.

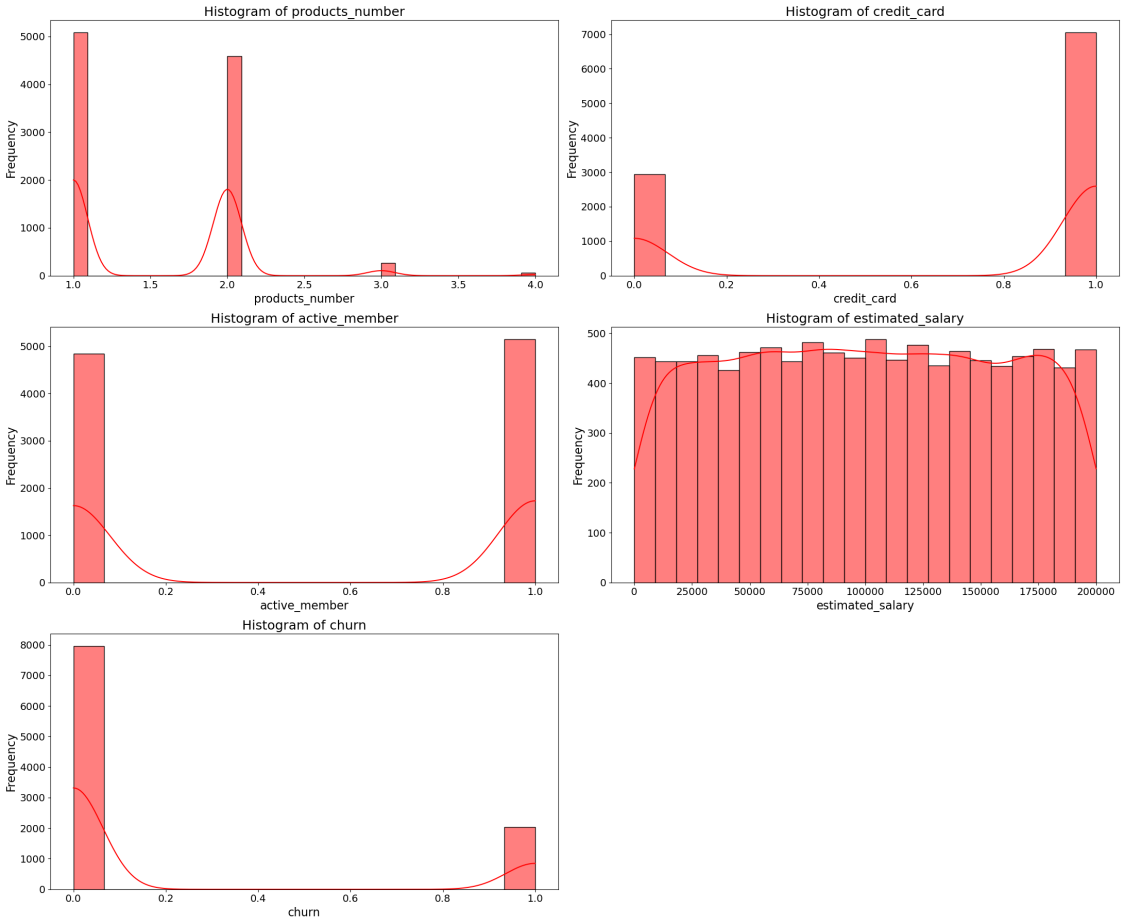
```
   credit_score  gender       age  tenure   balance  products_number  \
0     -0.326221       0  0.293517       2 -1.225848                1
1     -0.440036       0  0.198164       1  0.117350                1
2     -1.536794       0  0.293517       8  1.333053                3
3      0.501521       0  0.007457       1 -1.225848                2
4      2.063884       0  0.388871       2  0.785728                1

   credit_card  active_member  estimated_salary  churn
0            1              1          0.021886      1
1            0              1          0.216534      0
2            1              0          0.240687      1
3            0              0         -0.108918      0
4            1              1         -0.365276      0
```

*Remarks:* I used scikit-learn's StandardScaler tool on all variables that weren't binary or within a short range (0-3, or 0-10) after having tested different models on unscaled data and coming up shorter with prediction metrics/results.

### 0.3.1 Variable Distributions

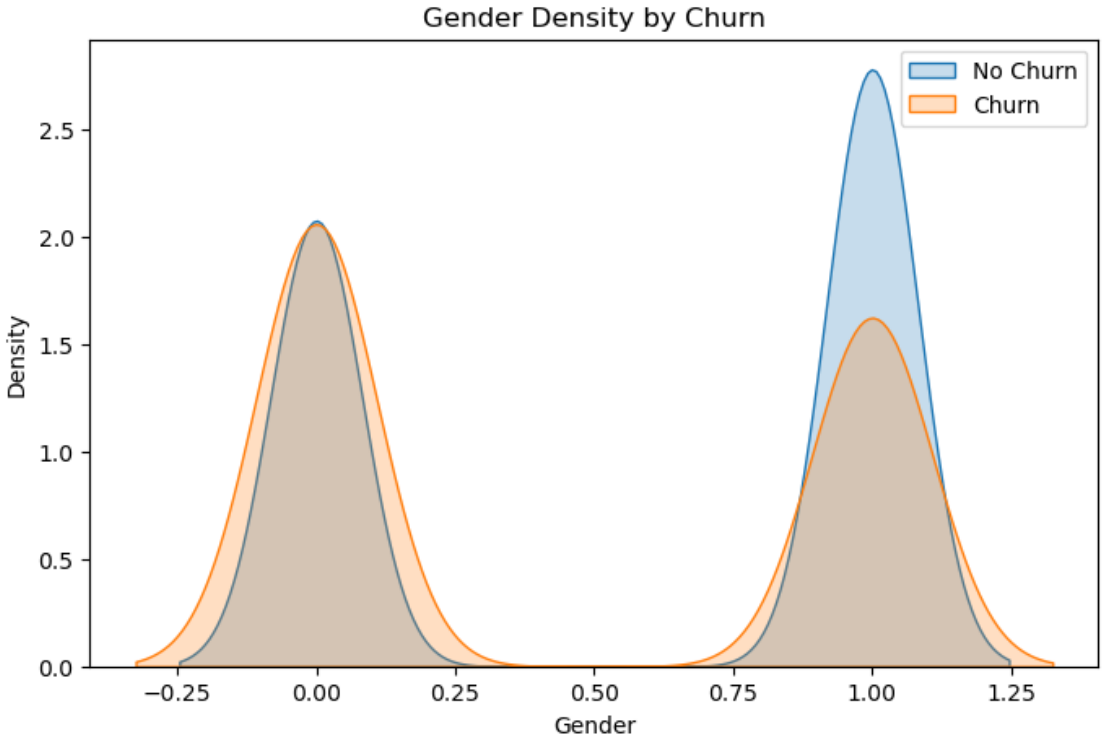



*Remarks:* The histograms for this dataset are actually rather interesting.In the first 5 variables we have credit_score, gender, age, tenure, and balance. Credit_score is mostly normal in distribution, although slightly skewed to the right with a number of outliers at the high score range (around 850-900). Gender is fairly even in distribution. Age is skewed left, which is expected for a given customer population and due to the older customers increasing the overall range. Age does have a number of outliers in around the year 30 and 40 value. Tenure is mostly uniform from 1 year to 9 years, with much less customers having 0 or 10 years with the bank. Balance is quite normal in distribution without the very large number of zero balance records that are showcased, which we will try to address later.

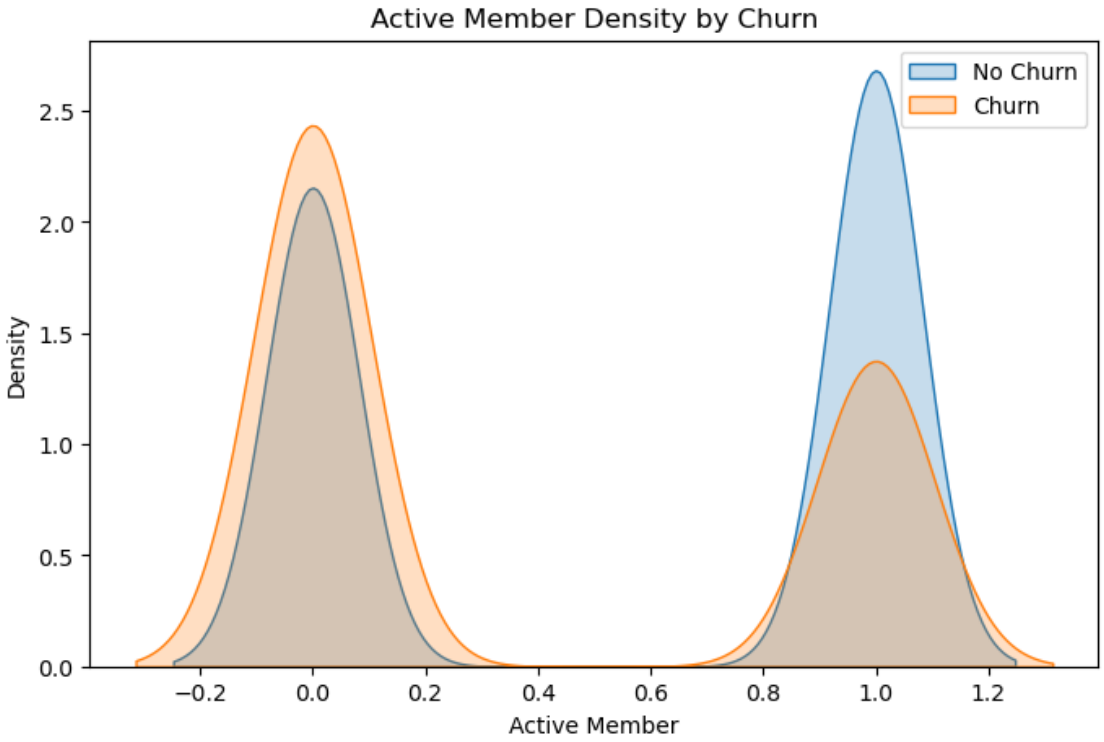



*Remarks:* In this second set of histograms we look at product_number, credit_card, active_member, estimated_salary, and our churn variable. Products_number is heavily skewed left with easily around >95% of all records indicating 1 or 2 products with the bank compared to 3 or 4. Most customers have 1 credit card, with only around a quarter not having one at all. Active_members with the bank is fairly even with inactive (0) and active (1). Estimated_salary has a generally uniform distribution from 0-200,000 with around 450 records in each salary interval/range. Churn is our y variable we want to predict, and it is heavily imbalanced. Skewed to the left (0 = never having left the bank), most customers have not left before while around 1/5 of customers have left at least once. This imbalance will be the main focus of the modeling efforts.

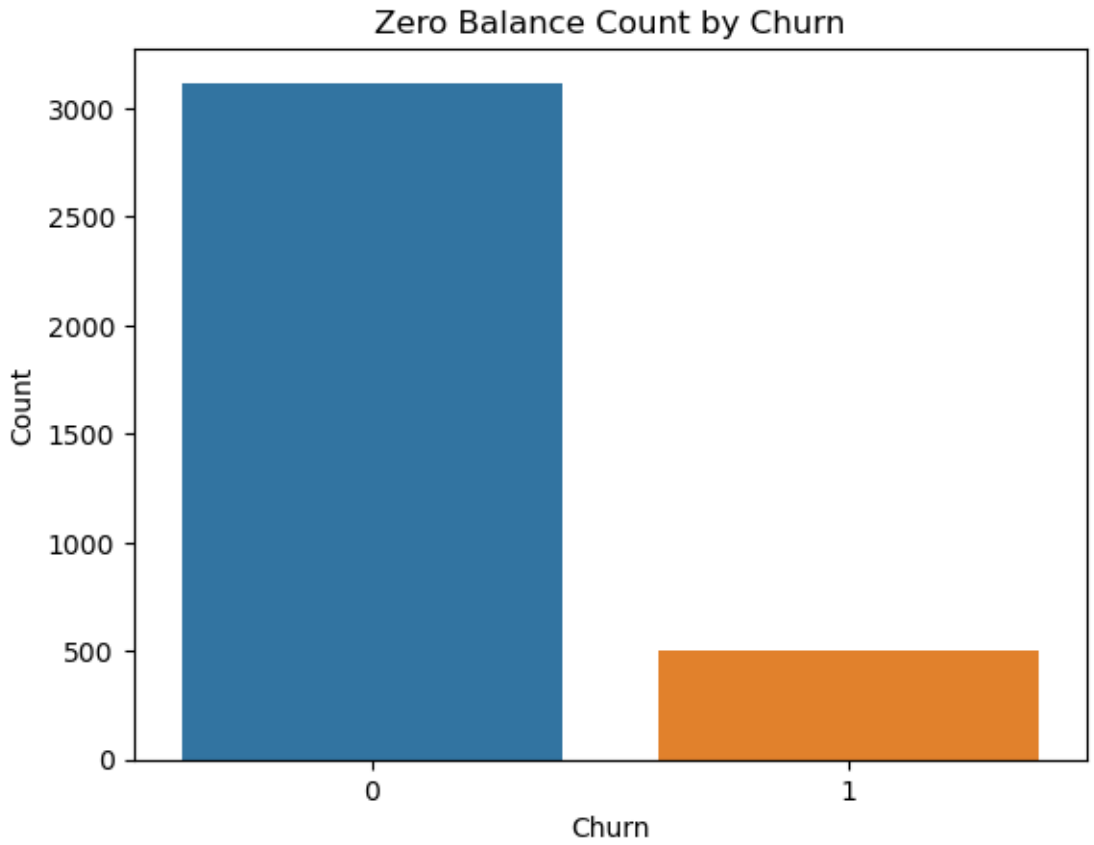
### 0.3.2 Density by Churn




*Remarks:* I wanted to see how churn was distributed across gender and found that men seemed to leave the bank (or churn) less often than women along with having a slightly higher density, which could be seen in the gender histogram.




*Remarks:* I also wanted to view active_members by churn after seeing the distribution as relatively even. As you can see active members (1) are also less likely to have left.




*Remarks:* Besides the imbalance within the binary churn classes (0 or 1), I wanted to see how my other main issue, the zero balances, would be distributed across churn. As you can see, around 85% of all zero balance records are also paired with a 0 churn value and have never left the bank before.

### 0.4 Train & Test Split

In my train and test split I wanted to create multiple sets of transformed data to test models with. I chose to first split with the dataframe that had no countries. Then, I used SimpleImputer to take care of the zero balances for my balance feature, using the mean to better represent the given data. I then used StandardScaler on the imputed X_train and X_test sets with select columns (age, credit_score, balance, estimated_salary) because of how different the ranges were across the dataset. I dont need to scale for the models I chose to use, but when I tested unscaled data they performed worse than the scaled counterparts. Lastly, I used SMOTE to oversample (normal training and imputed sets) the minority churn class (1) to try and improve overall prediction capability for the models. You can see the resampled distribution below.

```
Original churn distribution: churn
0    6356
1    1644
Name: count, dtype: int64
```

```
Resampled churn distribution: churn
0    6356
1    6356
Name: count, dtype: int64
```

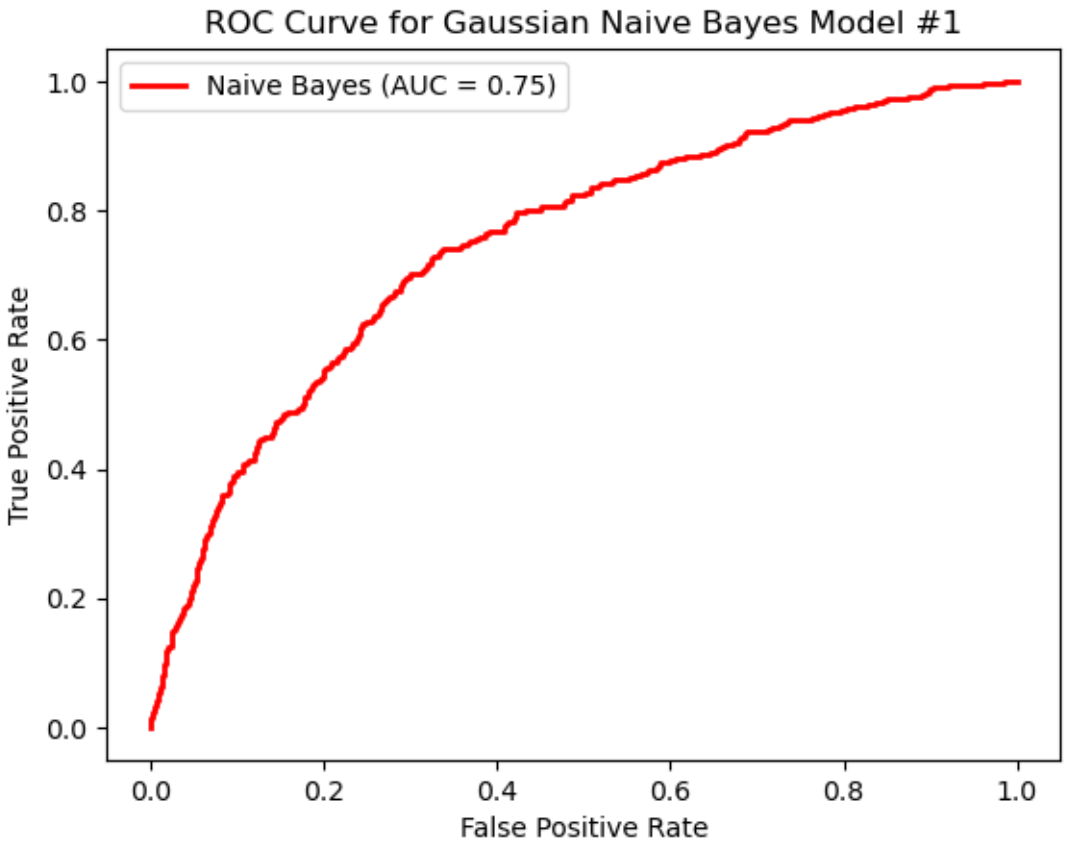
*Remarks:* As you can see, the imbalanced 80% y training set for churn was resampled and evenly distributed for both binary classes.

## 0.5 Model 1: Naive Bayes

My first model is of the type Gaussian Naive Bayes. In my first attempt, I use resampled data without imputation in the first test, and GridSearch to adjust my parameter. In my second attempt, I use imputed and scaled train/test sets. The printed results include the chosen parameter value, a model confusion matrix, a classification report, cross-validation ROC AUC scores, and a ROC AUC plotted curve with the corresponding score.

### 0.5.1 Model #1

```
Best Gaussian Model Parameter: {'var_smoothing': 1e-09}

Model Confusion Matrix:
 [[1082  525]
 [ 107  286]]

Model Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.67      0.77      1607
           1       0.35      0.73      0.48       393

    accuracy                           0.68      2000
   macro avg       0.63      0.70      0.62      2000
weighted avg       0.80      0.68      0.72      2000

Model ROC AUC Score: 0.7505917970203515
```

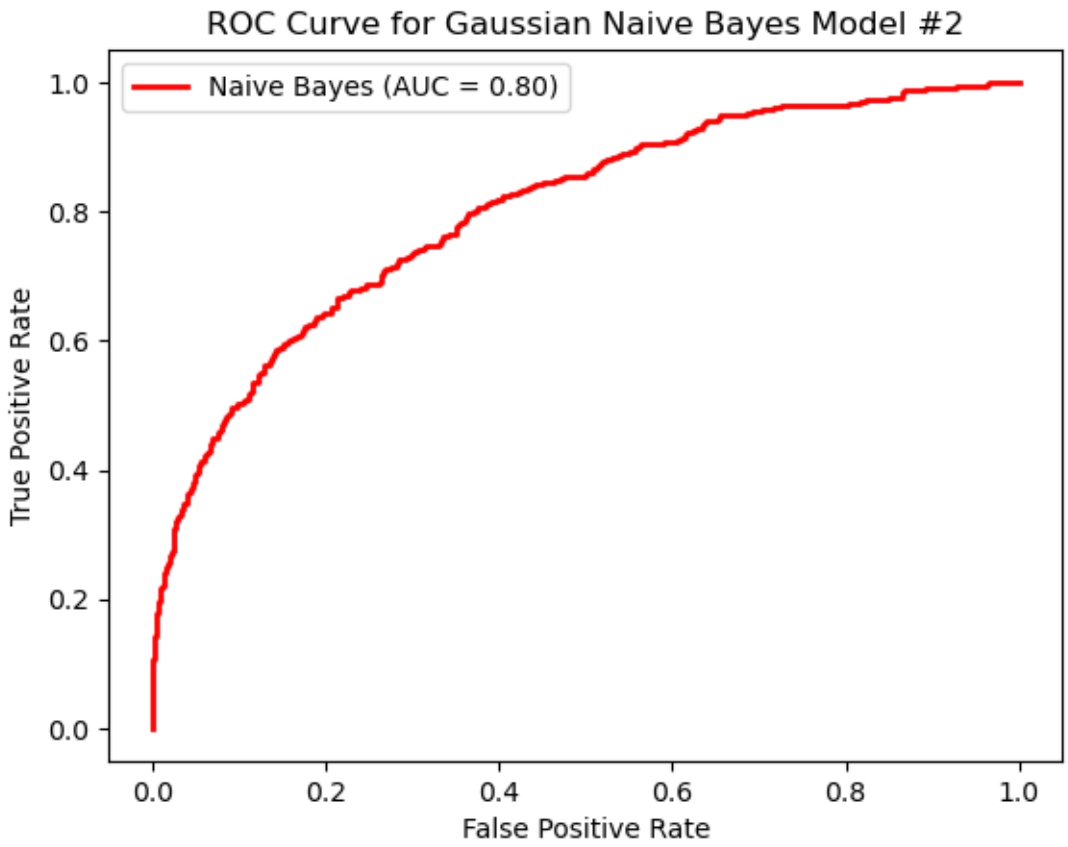



```
Cross-Validation ROC AUC Scores: [0.74009471 0.78614497 0.80743772 0.77271408
0.79302184]

Mean of Cross-Validation ROC AUC Scores: 0.779882663541543
```

*Remarks:* For my first Naive Bayes model, the results are not stellar. The f1 score is okay, 0.77, for predicting a customer who has never left (churn=0), but doing quite poorly for customers that have (churn=1) with a score of 0.48. The precision is also worse, with a score of 0.35. The accuracy is on the lower side, 0.68, and the ROC curve suggests the model is struggling to discriminate between positive and negative instances.

### 0.5.2 Model #2

```
Best Gaussian Model Parameter: {'var_smoothing': 1e-06}

Model Confusion Matrix:
 [[1201  406]
 [ 123  270]]

Model Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.75      0.82      1607
           1       0.40      0.69      0.51       393

    accuracy                           0.74      2000
   macro avg       0.65      0.72      0.66      2000
weighted avg       0.81      0.74      0.76      2000

Model ROC AUC Score: 0.8016533898291667
```




```
Cross-Validation ROC AUC Scores: [0.7846079  0.82826564 0.84724357 0.83382556
```

```
0.85134028]

Mean of Cross-Validation ROC AUC Scores: 0.8290565904586857
```

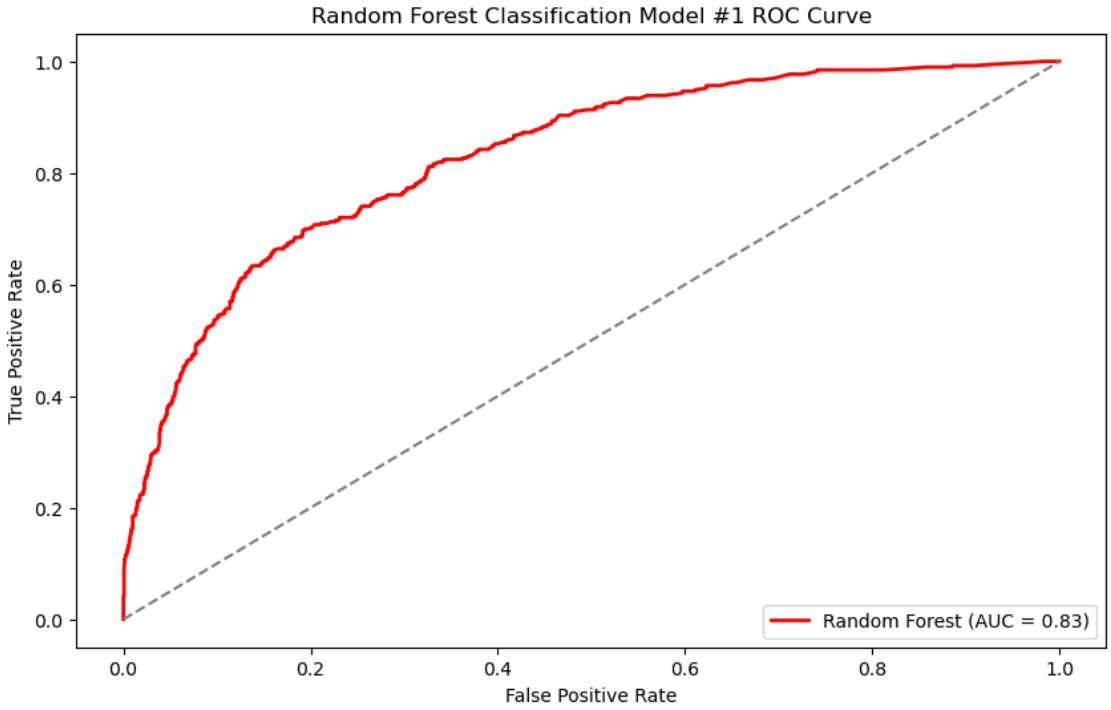
*Remarks:* For my second Naive Bayes model, there results are only slightly better. The ROC Curve and AUC score might look nice, but the model is really struggling to identify and predict customers who may leave. The imbalanced data should be better addressed with resampling, but I may have missed an opportunity to designate more valuable features and normalize them in a more comprehensive way.

## 0.6 Model 2: RandomForestClassifier

In my second model, I wanted to use a model type that is known for handling data well regardless of scaling, imbalance, binary classification, and nonlinear interactions. In my first attempt I use resampled data and GridSearch in the same manner as my Naive Bayes model, with the printed results showing the best chosen parameters, a classification report, a plotted ROC curve, and cross-validated $R^2$ and F1 scores. My second attempt did the same, but with imputed and scaled train/test sets. After those two, I decided to look at specifying the threshold value to see try and acquire better precision and F1 scores. I couldn't obtain scores that I would deem superior or noteworthy, but I decided to run a third model with an increased range in parameter values and also includes a precision-recall curve.

### 0.6.1 Model #1

```
Fitting 5 folds for each of 32 candidates, totalling 160 fits
Best Model Parameters:
 {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 1,
'min_samples_split': 2, 'n_estimators': 200}

Random Forest Model Classification Report:
               precision    recall  f1-score   support

           0       0.91      0.85      0.88      1607
           1       0.51      0.64      0.57       393

    accuracy                           0.81      2000
   macro avg       0.71      0.75      0.72      2000
weighted avg       0.83      0.81      0.81      2000
```

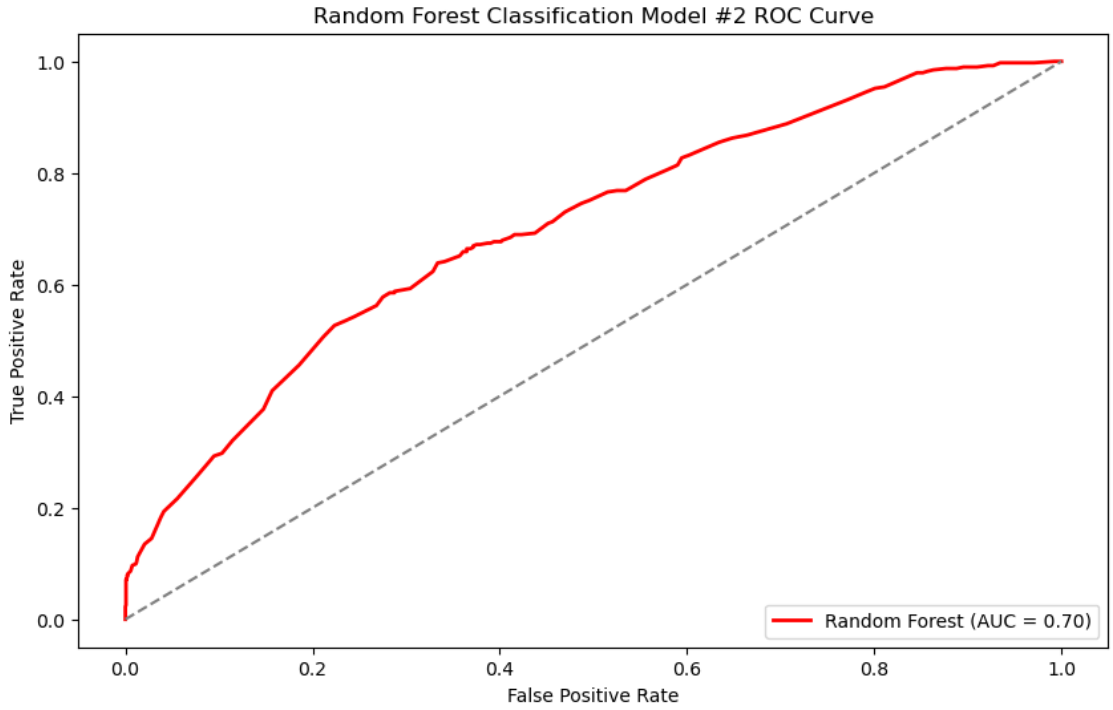



```
Cross-validated R² score for Random Forest Classification Model: 0.383
Cross-validated F1 score for Random Forest Classification Model: 0.843
```

*Remarks:* In my first random forest classifier model, the evaluation metrics are starting to look better. I have more parameters I can tune with this kind of classification model (which is useful for binary targets). Despite the modest increase in overall scores, the R2 score shows that the model is not fit well.

### 0.6.2 Model #2

```
Fitting 5 folds for each of 32 candidates, totalling 160 fits
Best Model Parameters:
 {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 1,
'min_samples_split': 2, 'n_estimators': 200}

Random Forest Model Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.88      0.89      1607
           1       0.56      0.63      0.59       393

    accuracy                           0.83      2000
   macro avg       0.73      0.75      0.74      2000
weighted avg       0.84      0.83      0.83      2000
```




```
Cross-validated R² score for Random Forest Classification Model: 0.521
Cross-validated F1 score for Random Forest Classification Model: 0.880
```

*Remarks:* In my second random forest classifier model, the classification metrics have modestly increased overall again. I used the imputed and resampled set with this one, which resulted in a higher R2 score and lower AUC score. The higher R2 is nothing significant, but it is a step foward toward the model having decent fit.

### 0.6.3 Threshold Test

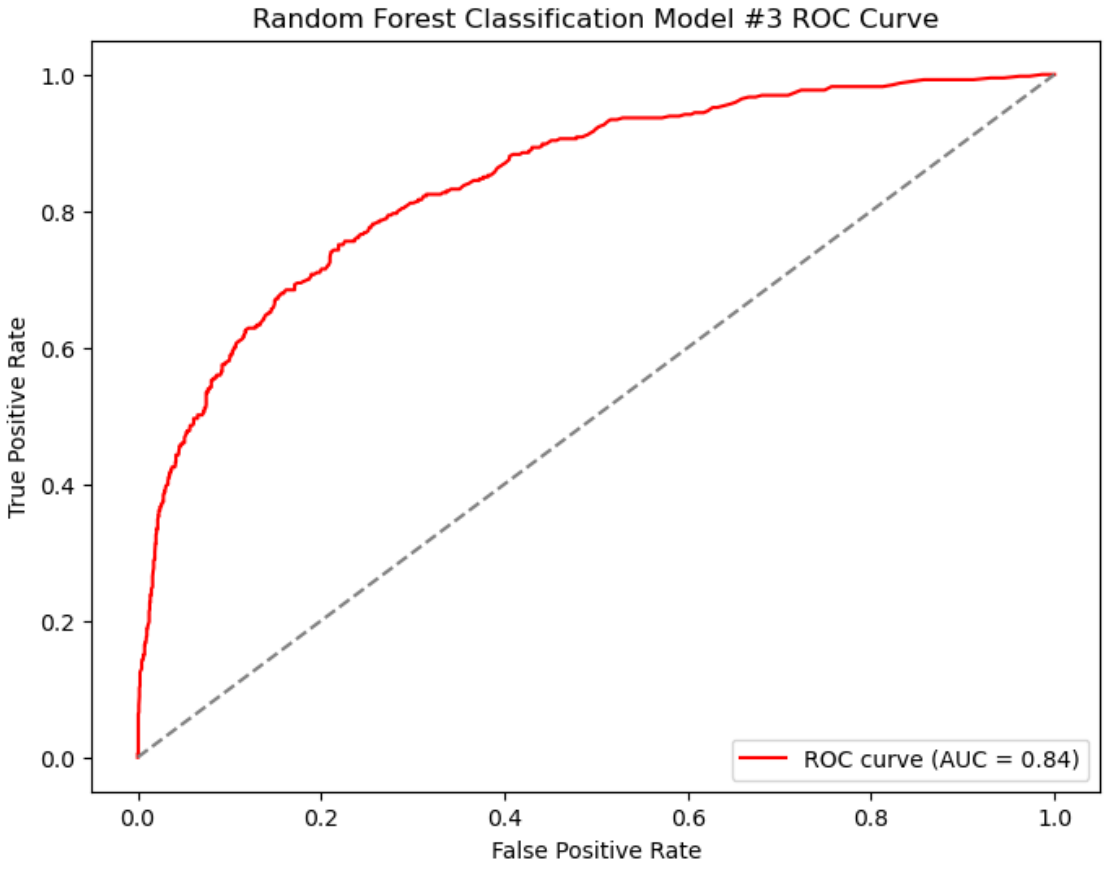
```
Threshold = 0.3: Precision = 0.40488431876606684, Recall = 0.8015267175572519,
F1 Score = 0.53800170794193

Threshold = 0.4: Precision = 0.46476510067114096, Recall = 0.7048346055979644,
F1 Score = 0.5601617795753286

Threshold = 0.5: Precision = 0.5585585585585585, Recall = 0.6310432569974554, F1
Score = 0.5925925925925926

Threshold = 0.6: Precision = 0.6375, Recall = 0.5190839694656488, F1 Score =
0.5722300140252454
```

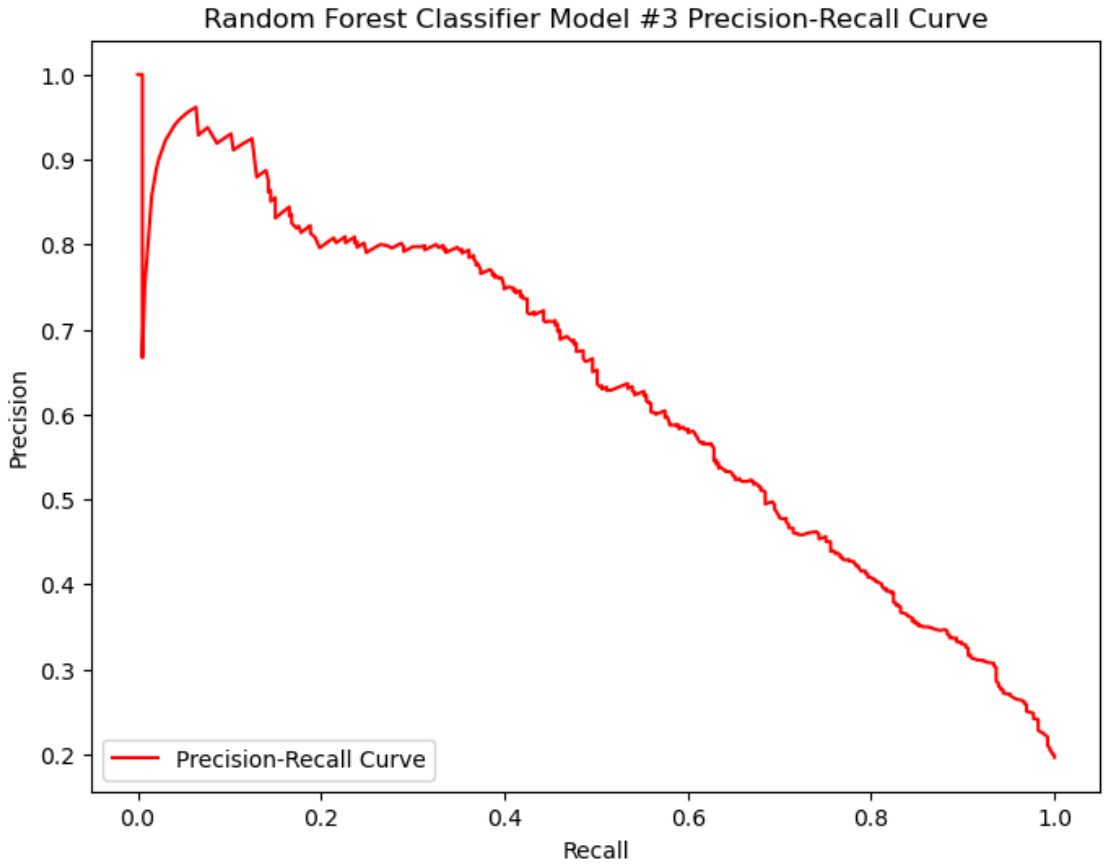
*Remarks:* After my last 2 random forest models I wanted to look at more options in how I can improve the model. I tried to find a better threshold where the model could accurately identify and predict both classes of churn (0 & 1), but there was no significant gain in any positive direction. The

scores appeared low and no trade-off seemed valuable. Perhaps I could have tested this differently or used another method.

### 0.6.4 Model #3

```
Fitting 5 folds for each of 36 candidates, totalling 180 fits
Best Model Parameters:
 {'max_depth': 25, 'min_samples_leaf': 1, 'min_samples_split': 2,
'n_estimators': 400}

Random Forest Model Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.88      0.89      1607
           1       0.56      0.63      0.59       393

    accuracy                           0.83      2000
   macro avg       0.73      0.75      0.74      2000
weighted avg       0.84      0.83      0.83      2000

Cross-validated F1 score for Random Forest Classification Model:
0.8834505052547451
Cross-validated R² score for Random Forest Classification Model:
0.5340028403327247
```


Random Forest Classification Model #3 ROC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (AUC = 0.84)

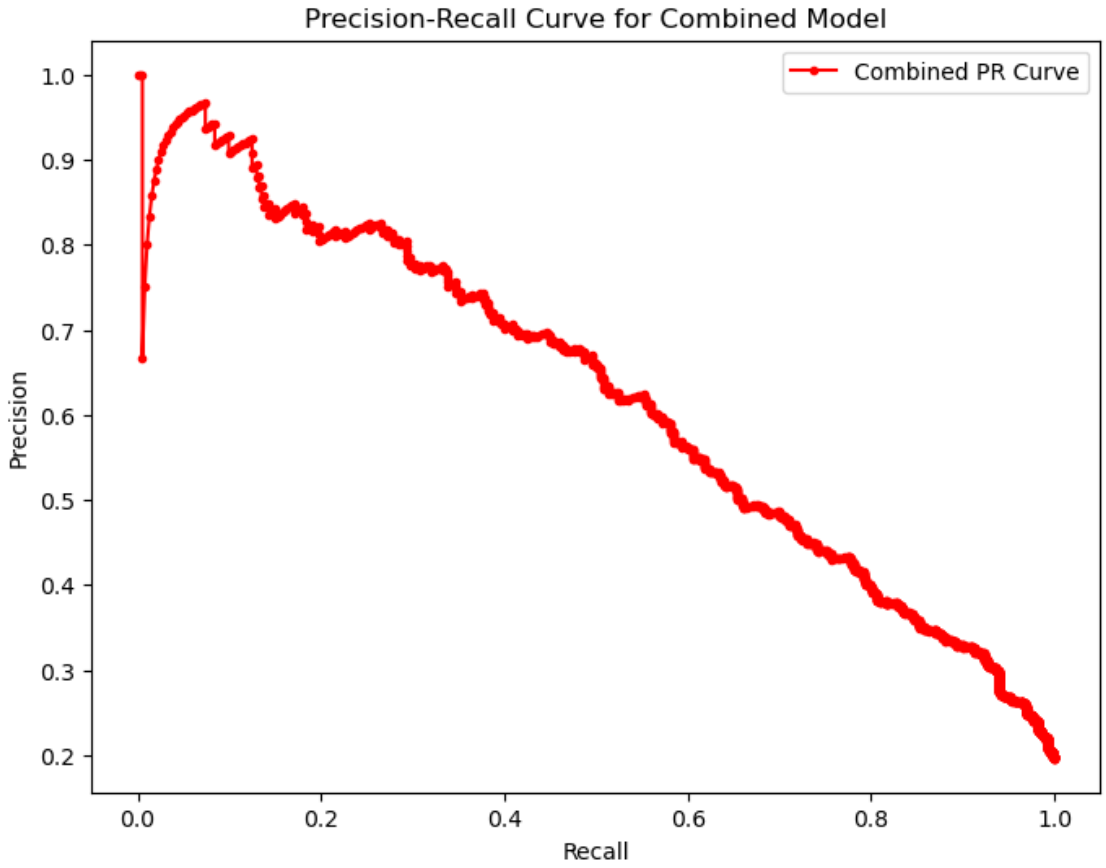



*Remarks:* As a last ditch effort with an individual random forest model attempt, I increased the parameter selection and tried to get more of an understanding on why my model is performing the way it is with additional metrics such as a precision-recall curve. As you can see, the precision-recall of my random forest model has a lackluster performance and would ideally be curved as straight as possible from Precision 1.0 to Recall 1.0 (parallel with each axis maximum).

## 0.7 Model 3: Combined XGBoost & RandomForestClassifier

Wanting to explore at least one more option, I did some research on XGBoost and found an ensemble method of prediction modeling using VotingClassifier. I also wanted to try at least one more method of improving the imbalanced prediction results and found the Repeated Stratified KFold, which employs cross-validation and is known to improve estimated ML model performance. I set both models up with parameters determined by GridSearch, combined them, and conducted similar tests that I have done thus far. The results include a classification report, precision-recall curve, and the combined ROC AUC score.

```
Fitting 15 folds for each of 36 candidates, totalling 540 fits
Fitting 15 folds for each of 243 candidates, totalling 3645 fits
Combined Model Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.88      0.89      1607
           1       0.55      0.61      0.57       393

    accuracy                           0.82      2000
   macro avg       0.72      0.74      0.73      2000
weighted avg       0.83      0.82      0.83      2000
```



```
ROC AUC Score for Combined Model: 0.841
```

*Remarks:* My last model, as a combination effort of XGBoost and random forest, performed ever so slightly worse than my final individual random forest model. Hopefully I can learn more about why my models struggled with the imbalanced binary target "churn", and how I could have improved prediction accuracy and classification with my given features/variables.

## 0.8 Conclusion & Final Remarks

It appears that my RandomForestClassifier model performed the best alone, and my Naive Bayes model was least effective. After completing this project I am left with more questions and an interest in exploring new avenues of analysis. I clearly have much more to learn, but I enjoyed working with this dataset from Kaggle and trying to figure out the best method of dealing with these kinds of features and the binary target value. If I had more time or found more time, I would have liked to explore other things such as alternate prediction methods and using the additional country variables. I think they could have potentially been a great feature if tuned well. I could have tried more scaling and normalization techniques to better fit my models, used other methods such as z score to better identify outliers. I would have also liked trying to trim features down and engineering those selected more effectively.