An Investigation of Bank Churn

Daniel G. Barstow

Department of Computer Science and Information Systems

Elmhurst University

Dr. Jim Kulich

12 December 2023

Table of Contents

Abstract

An analysis of a bank churn dataset was conducted using PyCaret in Google Colab. First, an exploratory data analysis was conducted to aid in determining feature engineering steps. An initial set of machine learning models were created to establish a baseline for performance. Then feature engineering techniques were used to augment the dataset and new models were created to assess the change in performance. 5 models in total were created using combinations of feature engineering techniques: 4 of which being light gradient boosting machines, and the other being a quadratic discriminant analysis. The best performing model was determined to be the QDA achieving a recall for class 1 of 0.985. A cost benefit analysis was conducted to determine the benefit of using the model over a dummy classifier that always predicts churn. It was found that the QDA was indeed more cost-effective than a dummy.

Background

Churn is a metric that refers to a loss of customers who use a service—like a bank. Because it is more costly for a bank to acquire new customers than retaining their old customers, they have a vested interest in minimizing churn as much as possible. The goal of this project is to predict churn accurately using machine learning methods. The dataset was acquired from Kaggle and has 11 relevant features: credit_score, country, gender, age, tenure, balance, products_number, credit_card, active_member, estimated_salary, and churn.

Exploratory Data Analysis

Fig. 1 shows the proportion of churned customers to non-churned customers. The dataset is rather imbalanced as only 20.4% churned. Fig. 2 shows the distributions of the numerical features. Credit score appears to be normally distributed with a slight left skew. Age shows a right skew, indicating a larger number of younger customers. Tenure is roughly uniformly distributed with dips at 0 and 10 years. Balance has a large peak at zero balance with the rest of the distribution being somewhat normal. Estimated Salary is uniformly distributed. Fig. 3 shows histograms of the categorical features. Country indicates that the majority of customers are from France, with Spain and Germany having a similar number of customers. Gender shows there are slightly more males than females. The number of products shows that most customers only use 1 or 2 financial products, with few using 3 or 4. Credit card shows that a majority of customers have a credit card. Active member shows that roughly half of customers are active members.

Fig. 4 shows boxplots of each of the features vs. churn. The credit score distribution appears similar for both churned and non-churned customers. There are some outliers for the churned group exhibiting the lowest credit scores. The age distributions are noticeably different. Churned customers tend to be older than ones who don't. The presence of more outliers on the not-churned side means that there are more customers with ages that fall outside the typical range within the group that has not churned. The tenure distributions are similar with the churned portion exhibiting a slightly larger variance. The balance distributions are slightly

different, with churned customers having a slightly larger median balance. The estimated salary distributions show no clear difference between those who churn and those who do not. Churn varies by country: France has the lowest, followed by Spain, then Germany. Churn for females is slightly higher than for males. There is a clear increase in churn as the number of products increases—every customer with 4 products churned (although the sample is small relative to the size of the dataset). Owning a credit card does not make a significant difference in churn rates. Active members have a lower churn rate than inactive members.

Machine Learning (Without Feature Engineering)

Machine learning techniques were used before feature engineering to establish a baseline. The highest performing model was a gradient boosted classifier with high accuracy but low recall and is shown in fig. 5. The confusion matrix shown in fig. 6 shows that the model had 284 true positives, 2302 true negatives, 87 false positives, and 327 false negatives. The lift curve in fig. 7 shows high values at the start, meaning the mode is effective at identifying the most likely positive cases. It also indicates that the model provides better predictions than random guessing. The gain chart in fig. 8 shows that a significant percentage of the positive class (60%) is captured using a small percentage of the sample (20%). Both of these charts show diminishing returns as the sample size increases, and that the model is better at predicting no churn than churn. So, while the model is good at identifying potential positive cases, it tends to miss a substantial portion of actual positives which is why we see a low recall value. This might be acceptable or even preferable given that the cost of having a false positive is much less than the cost of having a false negative.

The decision boundary in fig. 9 is rather complex. There's a mix of points in the two regions, indicating that the model has some degree of misclassification. It's not clear how the data should be separated just from looking at the plot. Fig. 10 is a feature importance plot for the gradient boosting classifier. We see that age, products_number, active_member, balance, and country_germany are the top 5 most important features for this model, while features like credit_card, country_france, country_spain, and tenure do not contribute at all to the model.

Machine Learning (With Feature Engineering)

For each of the following feature engineering steps, models were trained independently on the data at each step.

K-Means Clustering

The first feature engineering step was clustering. The categorical features were encoded and the numerical features were scaled using the StandardScaler. The elbow method was used to determine the optimal number of clusters and is shown in fig. 11. Looking at the graphs, it would seem the best clusterings use between 2 and 4 clusters. 3 sets of models were created, with 2, 3, and 4 clusters respectively. Adding these clusters identities back into the dataset as

features, performance overall went down slightly while recall increased slightly. This slight increase in recall is likely due to data leakage, and so one should be skeptical of the results.

Grouping Classes Together

The second feature engineering step was grouping records that had products_number = 3 or 4 together as one due to the small sample of people with 4 products. Doing this slightly decreased overall performance, while slightly increasing recall.

Binning Ages

The third feature engineering step was to bin the ages. The bins were 18-29, 30-39, 40-49, 50-59, 60-69, 70-79, 80-89, and 90+. Doing this led to slightly decreased performance overall while slightly increasing recall. There was likely a better binning scheme.

Box-Cox Transform on Age and Balance

A box-cox transform was done on the age and balance features due to their distribution. This also slightly decreased overall performance while slightly increasing recall.

Leaving Out Tenure and Estimated Salary

The tenure and estimated salary features were left out due to their low feature importance in the original model. Doing this also slightly decreased overall performance while slightly increasing recall.

Principal Component Analysis

Principal components for the data were found using Scikit-learn's PCA library. With 2 components, performance didn't improve at all. With 3 components, recall increased slightly but slightly reduced performance overall. With 4 components, overall performance slightly decreased and the increase to recall was less than with 3 components.

Synthetic Minority Oversampling Technique (SMOTE)

Using SMOTE also gave a small improvement to recall at the cost of reduced overall performance.

Best Model Comparison

At this point, different combinations of feature engineering steps were done to create slightly different datasets that new models were trained on.

Model 1

- Added a new column `balance_to_salary_ratio`
- Ignore tenure and estimated salary features
- SMOTE

- PCA

Model 1 had a higher recall than the model without feature engineering (0.4810 to 0.5315), while overall performance and AUC went down slightly (shown in fig. 12). We see a significant decrease in the number of false negatives at the cost of some false positives in fig. 13. While the decision boundary in fig. 14 is less complex than the model before feature engineering, there is still a lot of misclassification, as the points are not well separated. The feature importance plot in fig. 15 gives a good deal of importance to all the principal components.

Model 2

- Added a new column `balance_to_salary_ratio`
- Ignore tenure and estimated salary features
- SMOTE

Model 2 had higher recall than the model without feature engineering (0.4810 to 0.5126), while overall performance and AUC went down slightly (shown in fig. 16). The number of false negatives in the confusion matrix in fig. 17 went down at the cost of some false positives (but not as much as model 1). The decision boundary in fig. 18 is somewhat similar to the first one, but the blue region is much smaller. The feature importance plot in fig. 19 shows that credit_score, age, and balance_to_salary_ratio were the most informative features, while credit_card, country_spain, and country_france were the least informative features.

Model 3

- Ignore tenure and estimated salary features
- SMOTE
- PCA

Model 3 had a good deal of recall more than the model without feature engineering (0.4810 to 0.5932), while the overall performance and AUC went down slightly (shown in fig. 20). The number of false negatives in the confusion matrix in fig. 21 decreased at the cost of many false positives. The decision boundary in fig. 22 shows mostly green regions, with the blue regions being pushed off to the sides. The feature importance plot in fig. 23 shows that most of the principal components are relatively important.

Model 4

- Ignore tenure and estimated salary features
- SMOTE

Model 4 had higher recall than the model without feature engineering (0.4810 to 0.5455), while overall performance and AUC went down slightly (shown in fig. 24). The number of false negatives in the confusion matrix in fig. 25 decreased at the cost of some false positives. The decision boundary in fig. 26 is similar to the model without feature engineering but with more green rather than blue. The feature importance plot in fig. 27 shows that balance, credit_score,

age, and products_number are the most important features, while credit_card, country_France, and country_Spain are the least important features.

Model 5

- Quadratic Discriminant Analysis
- Synthetic Minority

The prior models were deemed not good enough due to their low recall. Recall is arguably the most important metric in this problem because the cost of false negative is very high. Since the Quadratic discriminant analysis had the highest recall in the very first run in fig. 5, this was the model improved on for the rest of the project.

The QDA created had the highest recall of any model so far (0.4810 to 0.9227). However, this came at a significant cost of accuracy (0.8657 to 0.2586). The true positive rate for this model was 0.686 and it had the best confusion matrix with respect to true positives so far and is shown in fig. 28. Then an Undersampling technique called Tomek Links was used to undersample the majority class. This led to the confusion matrix shown in fig. 29. This model had a true positive rate of 0.985, the highest achieved so far.

Discussion

While models 1-4 performed better than the first try, they weren't good enough to justify their use. The QDA model was explored and seemed to be the most promising given its high recall. Recall is important in this problem because the cost of acquiring new customers is significantly higher than the cost of retaining customers. The cost of committing a false negative error is likely somewhere between the average and median account balance ($76,485.89, $97,198.54). The cost of committing a false positive error is unknown, but it should certainly be less than the average or median account balance. The question is, at what point does the cost of missing out on some churn become larger than the cost of assuming everyone will churn?

If we assume that the cost of a false positive is half of the lower estimate ($38,242.95), and the cost of a false positive on average is $86,842.22, then the total costs are $89m and $91m for the model and the dummy, respectively. This means that our model is slightly more cost effective than the dummy classifier. But since we don't know the cost of a false positive there will be a point at which using the dummy is more cost effective than using a model. To determine this, we take the difference of the total costs divided by the number of false positives in our model. Doing this, we find that the threshold cost per false positive is $616.26. This means that if we determine that the cost of making a false positive error is above this number, then we should opt for the dummy classifier because we won't gain any benefit from using a model. This conclusion is based on an assumption that the cost of a false positive is $38,242.95. In practice, the cost of a false positive might be significantly lower than our estimate. If it were instead $10,000, then the threshold cost per false positive is $28,857.69,

meaning that our model performs much better than a dummy classifier if the cost of making a false positive is lower.

This all means that the QDA is a good enough model to use for this problem if the cost of making a false positive is low enough.

Relevant Figures

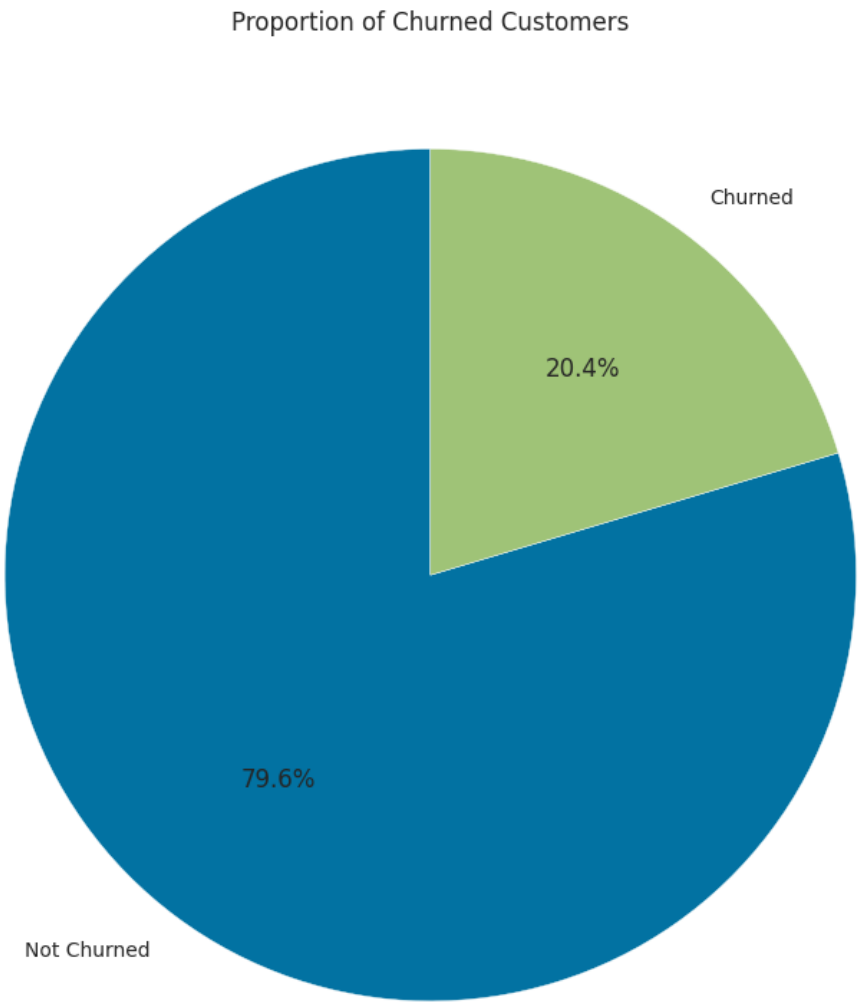

Figure 1 – A pie chart of the proportion of churned customers.

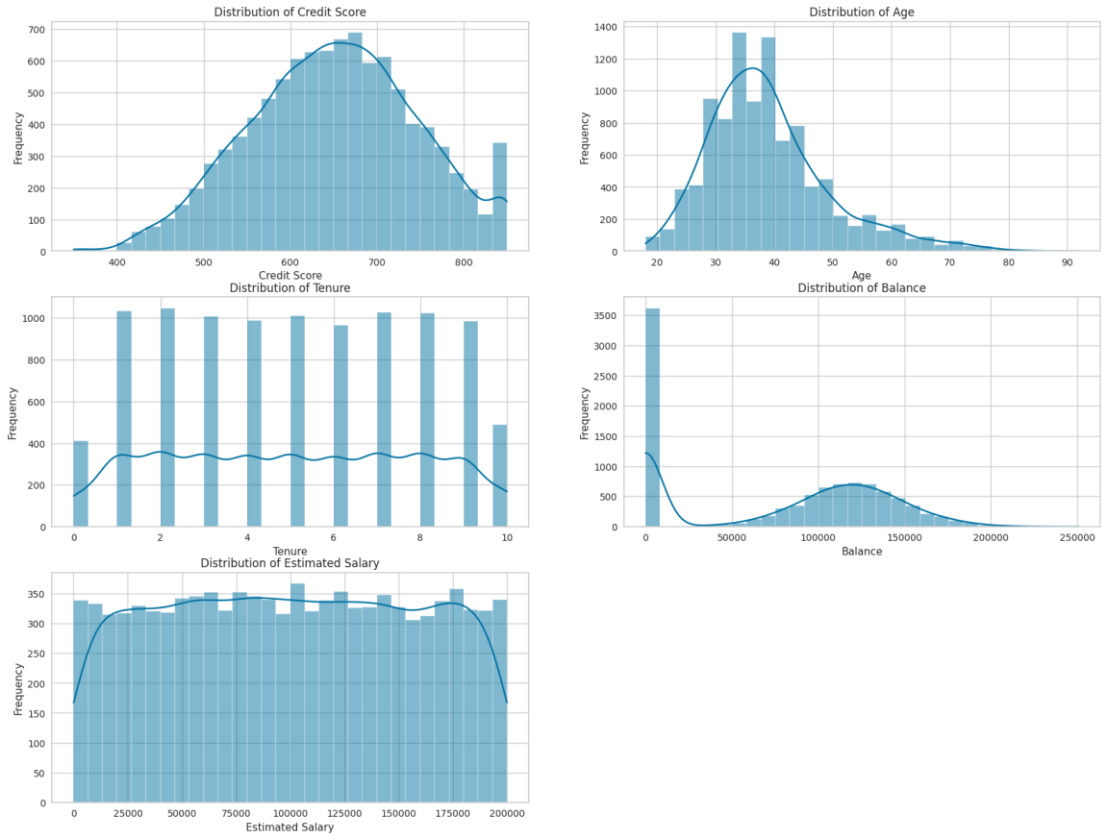

Figure 2 – Distribution plots of the numerical features.

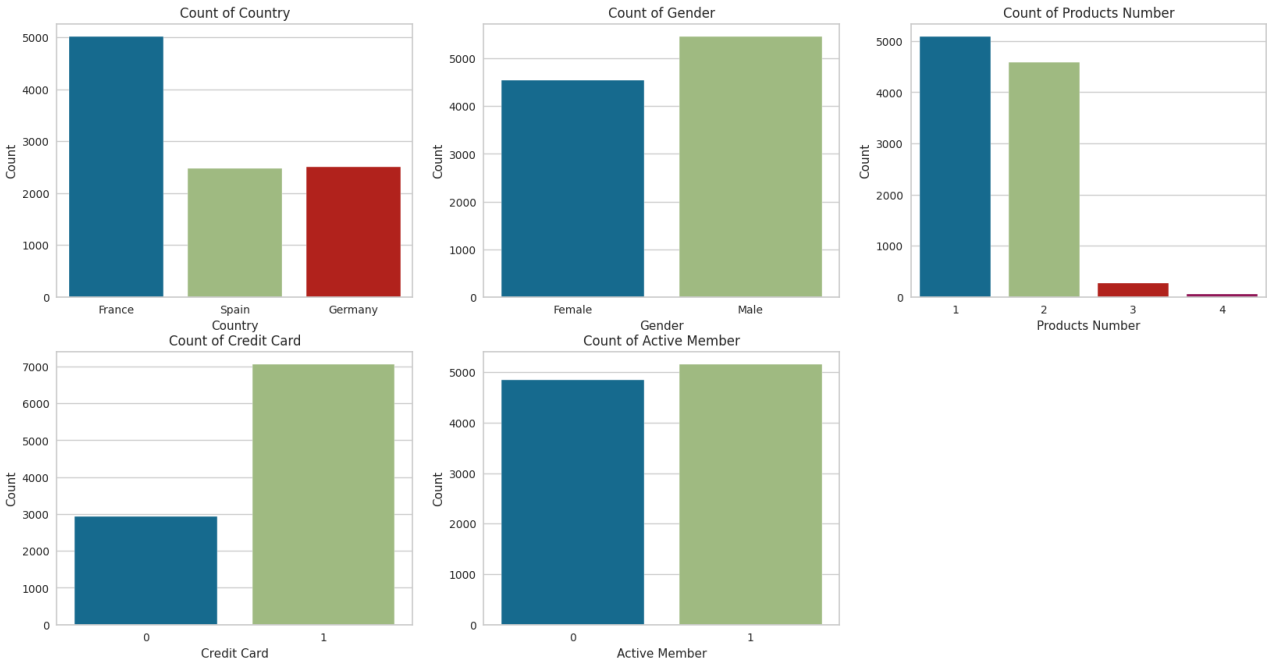

Figure 3 – Barplots of the categorical features.

Figure 4 – Boxplots of the features' relationship to Churn.

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
gbc	Gradient Boosting Classifier	0.8657	0.8691	0.4810	0.7750	0.5928	0.5178	0.5393
lightgbm	Light Gradient Boosting Machine	0.8614	0.8591	0.5098	0.7295	0.5993	0.5188	0.5314
rf	Random Forest Classifier	0.8599	0.8556	0.4649	0.7531	0.5735	0.4955	0.5168
ada	Ada Boost Classifier	0.8551	0.8515	0.4817	0.7143	0.5745	0.4914	0.5056
et	Extra Trees Classifier	0.8533	0.8461	0.4466	0.7296	0.5528	0.4713	0.4922
xgboost	Extreme Gradient Boosting	0.8509	0.8433	0.5000	0.6824	0.5768	0.4890	0.4979
lda	Linear Discriminant Analysis	0.8069	0.7649	0.2307	0.5619	0.3264	0.2365	0.2686
ridge	Ridge Classifier	0.8054	0.0000	0.1213	0.6205	0.2021	0.1449	0.2094
dummy	Dummy Classifier	0.7963	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
dt	Decision Tree Classifier	0.7900	0.6839	0.5048	0.4858	0.4946	0.3622	0.3627
lr	Logistic Regression	0.7894	0.6698	0.0575	0.3837	0.0995	0.0498	0.0793
nb	Naive Bayes	0.7851	0.7478	0.0610	0.3511	0.1034	0.0453	0.0689
knn	K Neighbors Classifier	0.7594	0.5384	0.0806	0.2399	0.1200	0.0185	0.0230
svm	SVM - Linear Kernel	0.6923	0.0000	0.1694	0.1474	0.1106	-0.0043	0.0004
qda	Quadratic Discriminant Analysis	0.3134	0.5040	0.8213	0.2074	0.3122	0.0026	0.0102

Figure 5 – Machine learning performance before feature engineering.

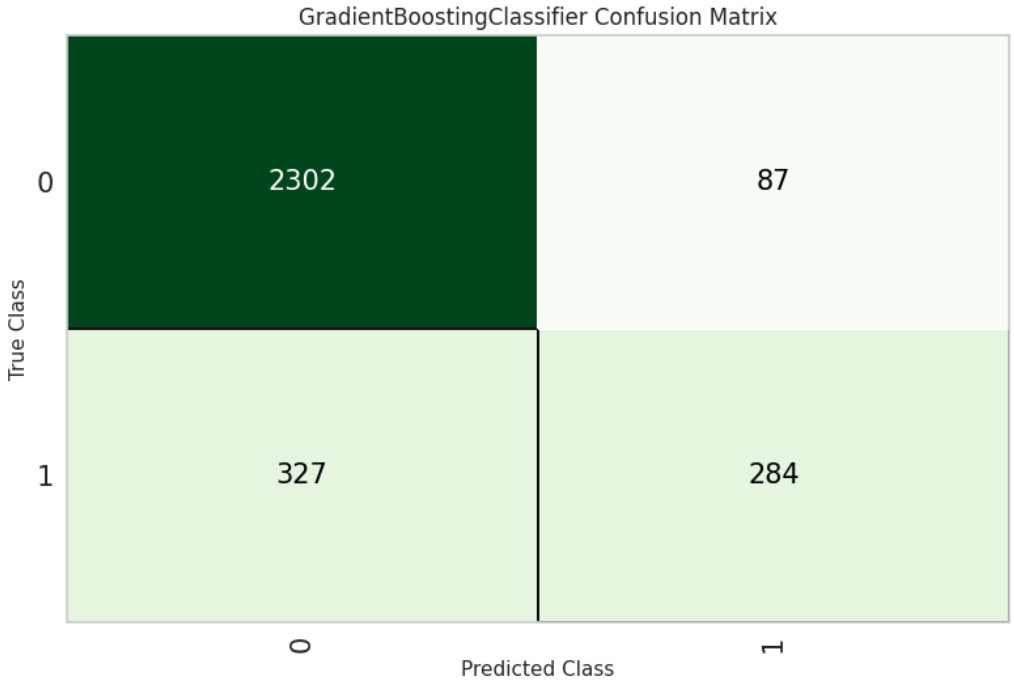

Figure 6 – Confusion Matrix for the best model created before feature engineering.

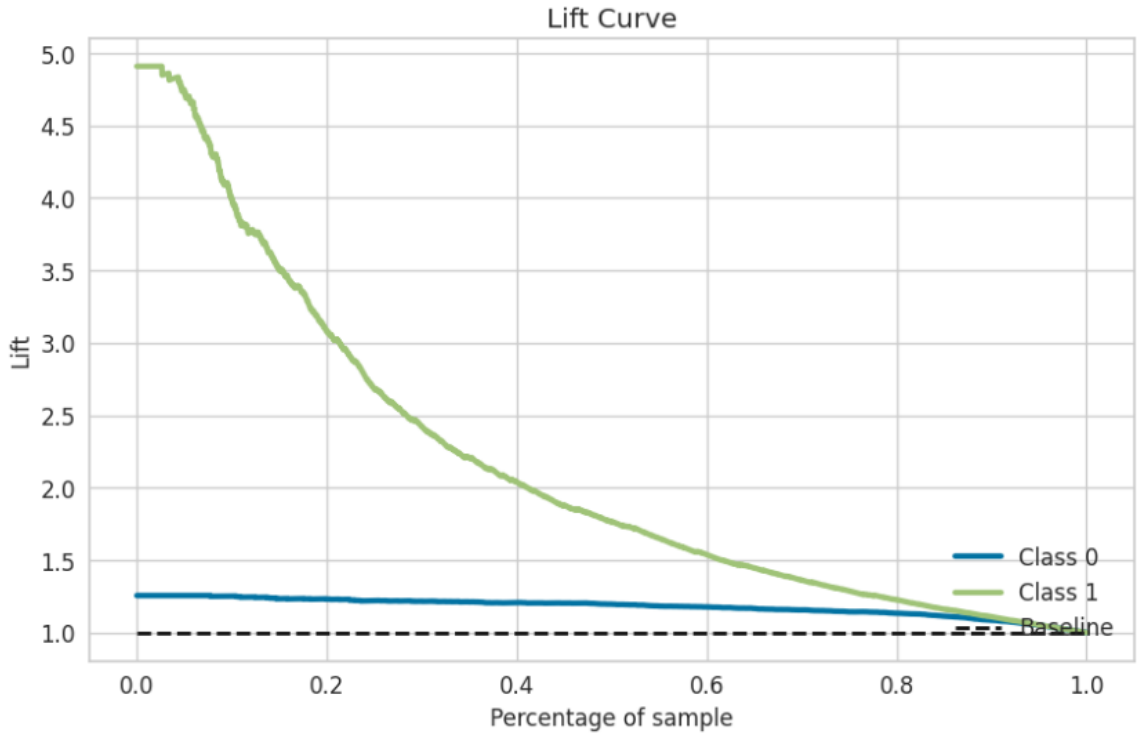

Figure 7 – The lift curve for the best model created before feature engineering.

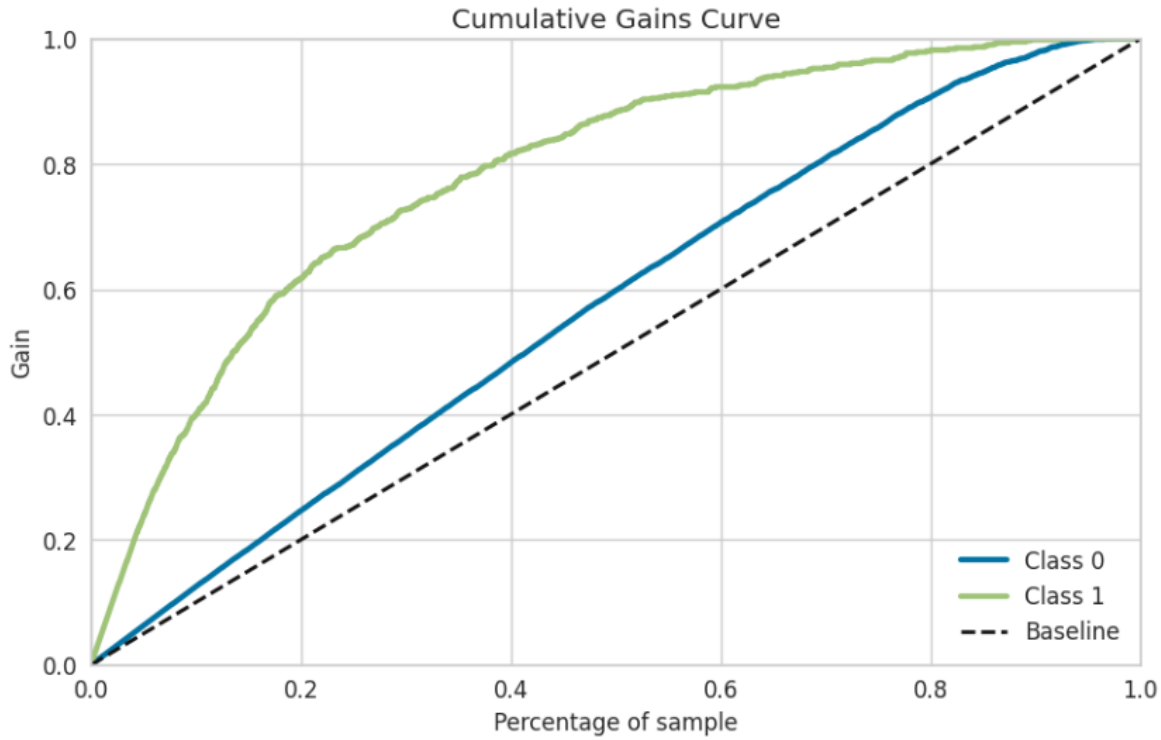

Figure 8 – The cumulative gains curve for the best model created before feature engineering.

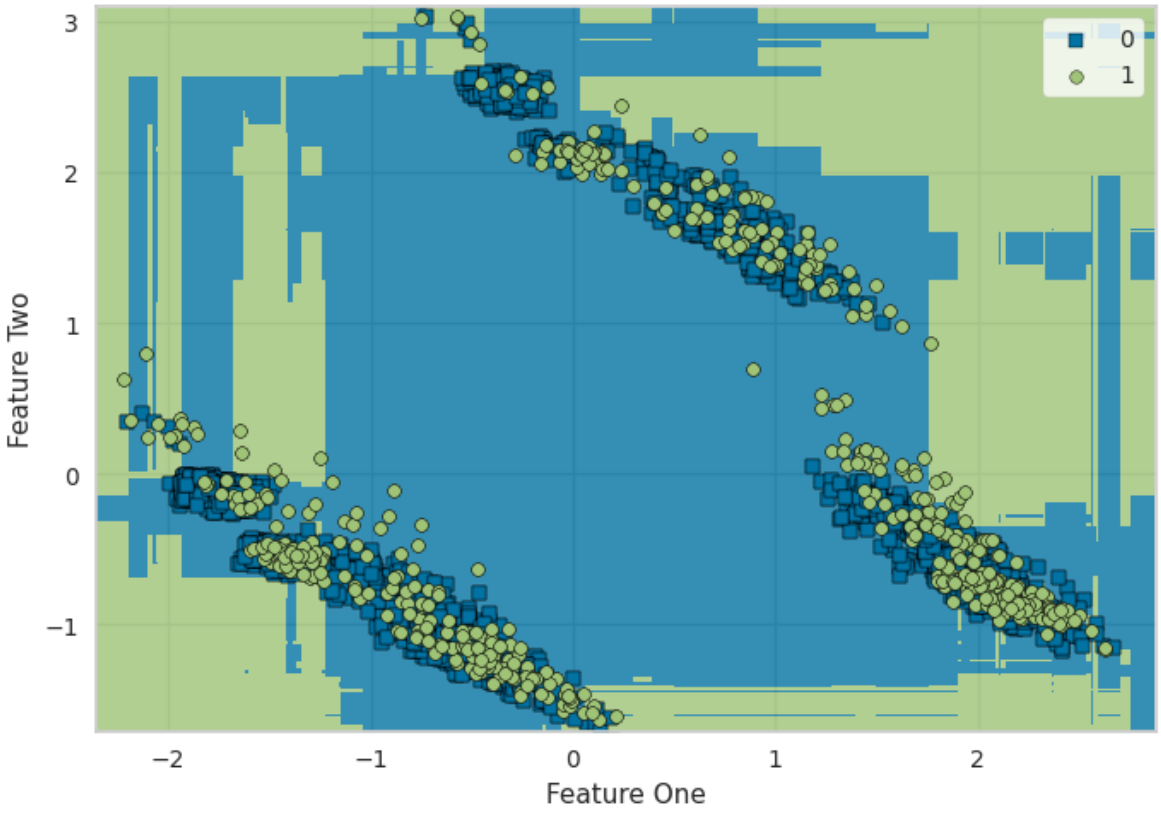

Figure 9 – The decision boundary for the best model created before feature engineering.

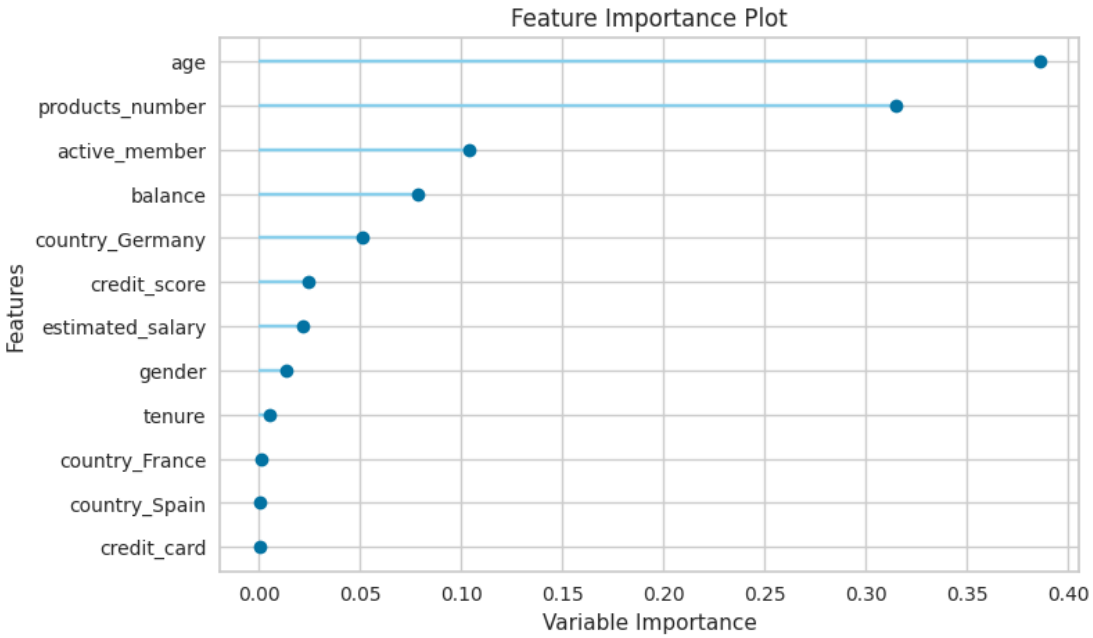

Figure 10 – The feature importance plot for the best model before feature engineering.

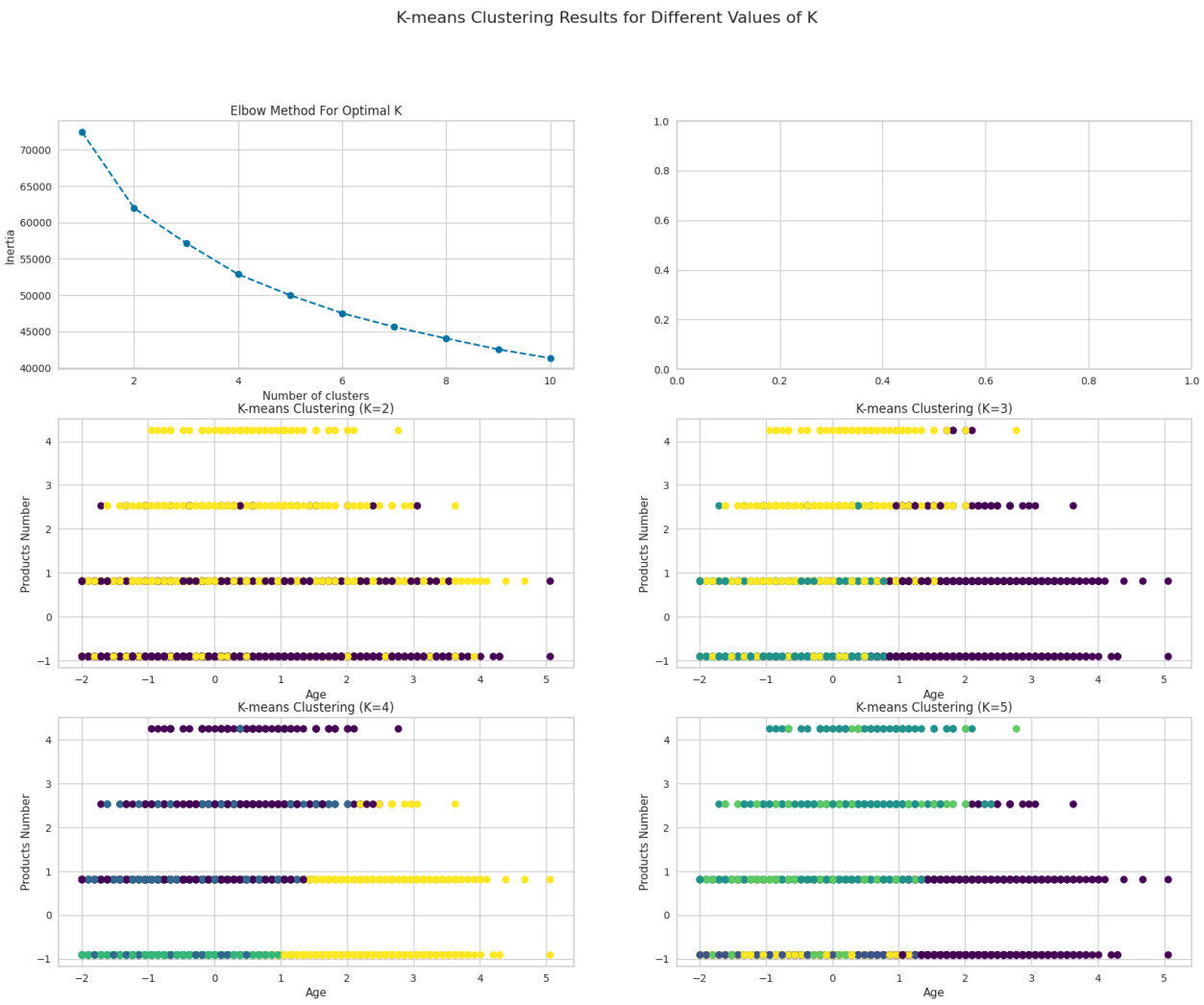

Figure 11 – K-means clustering plots using the Elbow method.

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
lightgbm	Light Gradient Boosting Machine	0.8446	0.8422	0.5315	0.6434	0.5816	0.4873	0.4910
gbc	Gradient Boosting Classifier	0.8397	0.8453	0.6066	0.6065	0.6061	0.5055	0.5058
xgboost	Extreme Gradient Boosting	0.8336	0.8164	0.5069	0.6094	0.5528	0.4518	0.4550
rf	Random Forest Classifier	0.8329	0.8143	0.5203	0.6031	0.5581	0.4559	0.4580
ada	Ada Boost Classifier	0.8251	0.8346	0.6262	0.5643	0.5930	0.4821	0.4837
et	Extra Trees Classifier	0.8199	0.7988	0.4936	0.5659	0.5265	0.4161	0.4180
nb	Naive Bayes	0.8069	0.7973	0.1065	0.6699	0.1741	0.1298	0.2033
dummy	Dummy Classifier	0.7963	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
dt	Decision Tree Classifier	0.7811	0.6810	0.5125	0.4673	0.4883	0.3497	0.3506
qda	Quadratic Discriminant Analysis	0.7570	0.7468	0.2641	0.6780	0.2893	0.1996	0.2661
ridge	Ridge Classifier	0.7116	0.0000	0.6746	0.3824	0.4880	0.3081	0.3322
lr	Logistic Regression	0.7106	0.7621	0.6774	0.3817	0.4882	0.3078	0.3325
lda	Linear Discriminant Analysis	0.7104	0.7568	0.6585	0.3790	0.4810	0.3000	0.3220
knn	K Neighbors Classifier	0.6593	0.6621	0.5596	0.3122	0.4007	0.1886	0.2045
svm	SVM - Linear Kernel	0.6330	0.0000	0.5023	0.3043	0.3568	0.1405	0.1538

Figure 12 – The set of performances for Model 1.

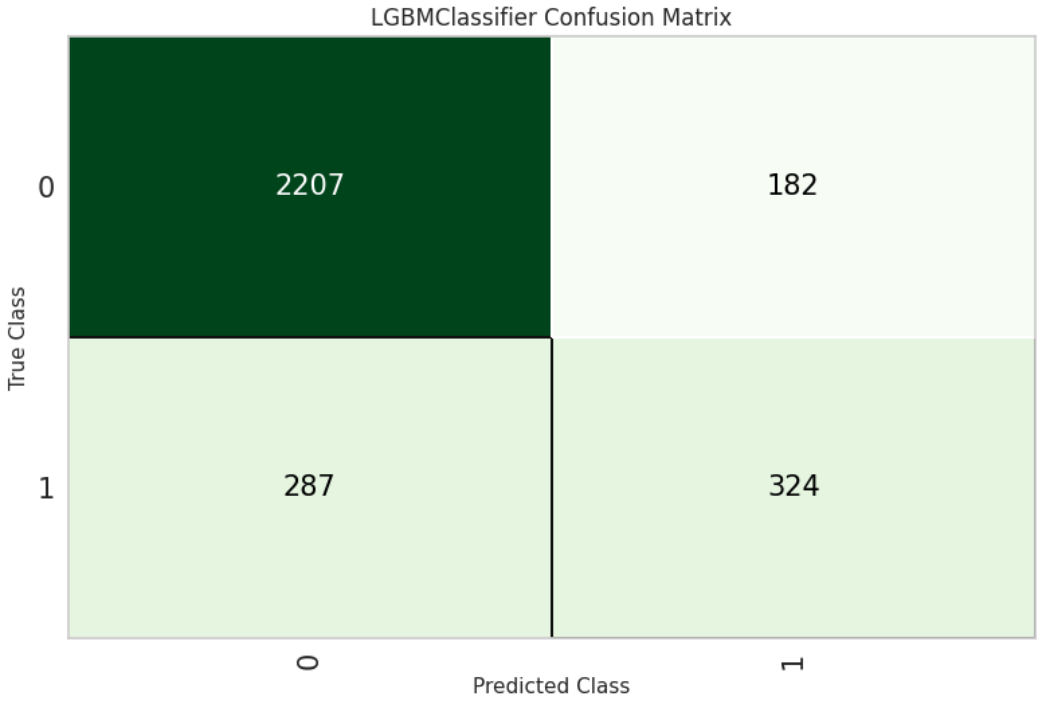

Figure 13 – The confusion matrix for Model 1.

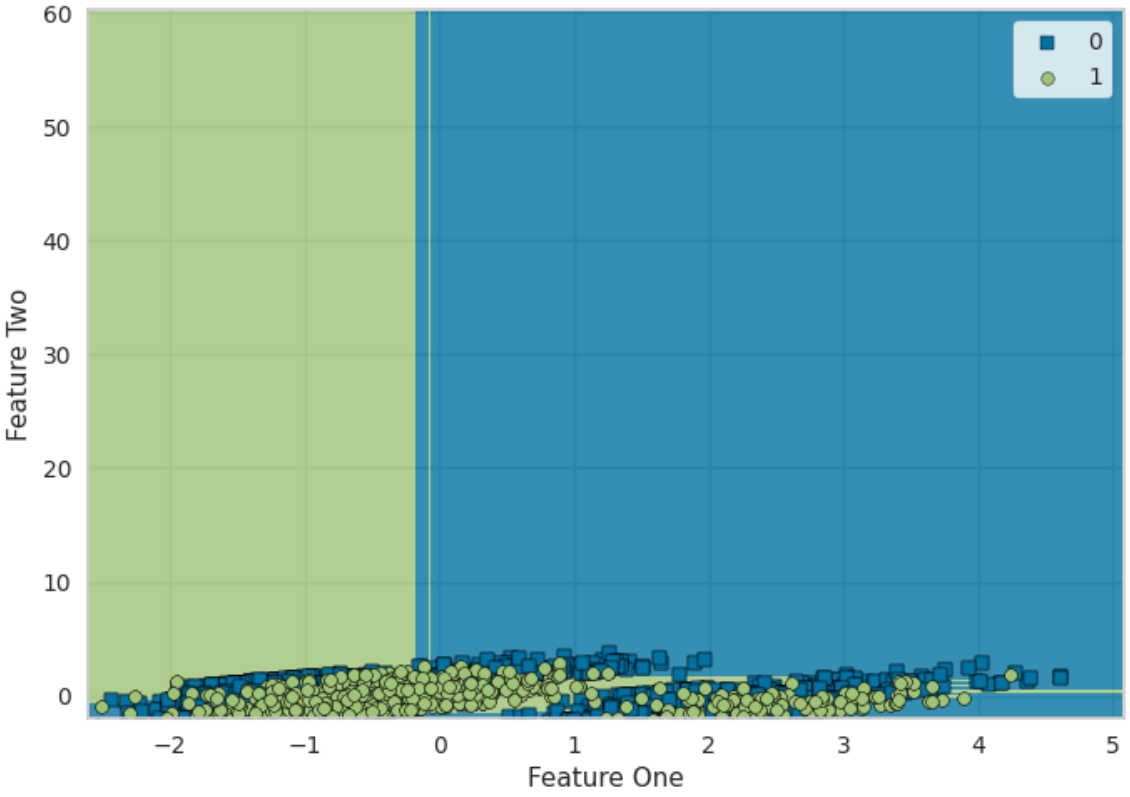

Figure 14 – The decision boundary for Model 1.

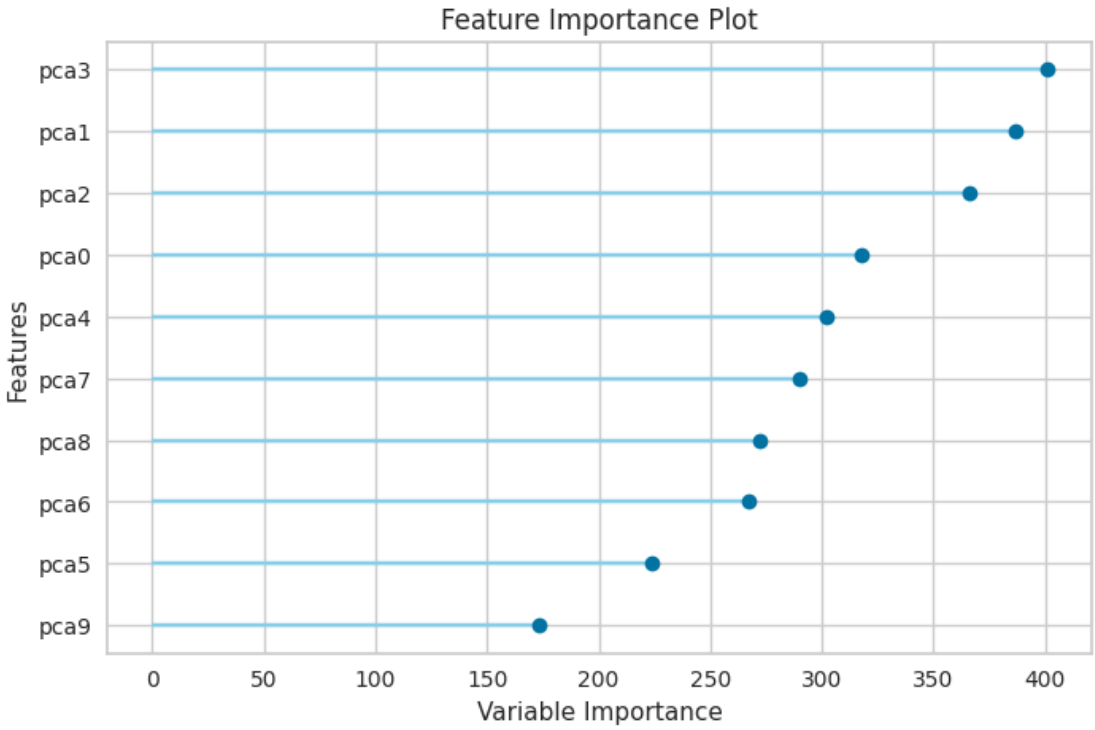

Figure 15 – The feature importance plot for Model 1.

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
lightgbm	Light Gradient Boosting Machine	0.8564	0.8557	0.5126	0.7013	0.5918	0.5074	0.5168
gbc	Gradient Boosting Classifier	0.8563	0.8632	0.5203	0.6973	0.5953	0.5103	0.5187
rf	Random Forest Classifier	0.8504	0.8299	0.5013	0.6800	0.5763	0.4882	0.4970
ada	Ada Boost Classifier	0.8480	0.8490	0.5427	0.6535	0.5920	0.4998	0.5035
xgboost	Extreme Gradient Boosting	0.8456	0.8347	0.5076	0.6561	0.5722	0.4799	0.4859
et	Extra Trees Classifier	0.8359	0.8155	0.4796	0.6272	0.5426	0.4450	0.4514
nb	Naive Bayes	0.8147	0.7847	0.1858	0.6998	0.2822	0.2168	0.2856
dummy	Dummy Classifier	0.7963	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
dt	Decision Tree Classifier	0.7867	0.6870	0.5189	0.4784	0.4975	0.3625	0.3632
ridge	Ridge Classifier	0.7116	0.0000	0.6746	0.3824	0.4880	0.3081	0.3322
lda	Linear Discriminant Analysis	0.7116	0.7620	0.6746	0.3824	0.4880	0.3081	0.3322
lr	Logistic Regression	0.7109	0.7621	0.6774	0.3820	0.4884	0.3082	0.3328
svm	SVM - Linear Kernel	0.7106	0.0000	0.4628	0.4318	0.3785	0.2307	0.2482
knn	K Neighbors Classifier	0.6586	0.6624	0.5596	0.3116	0.4002	0.1877	0.2036
qda	Quadratic Discriminant Analysis	0.6077	0.6180	0.4999	0.2986	0.3039	0.1031	0.1277

Figure 16 – The set of performances for Model 2.

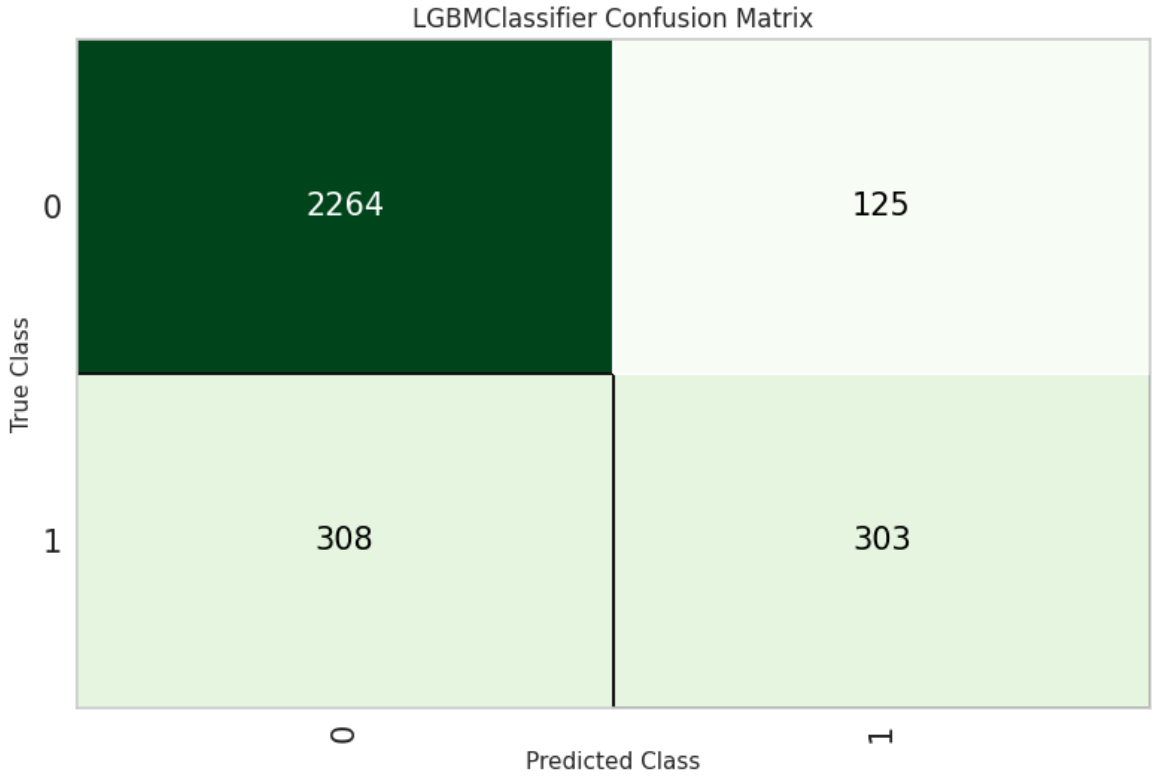

Figure 17 – The confusion matrix for Model 2.

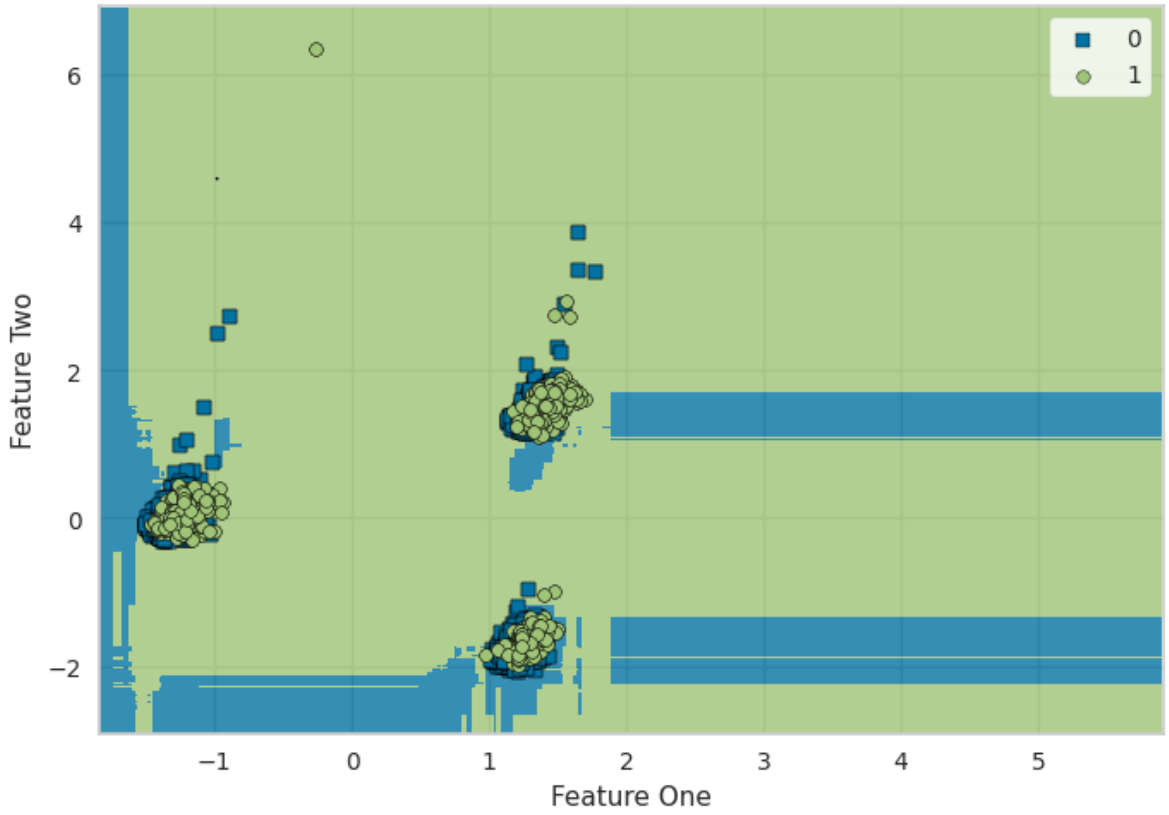

Figure 18 – The decision boundary for Model 2.

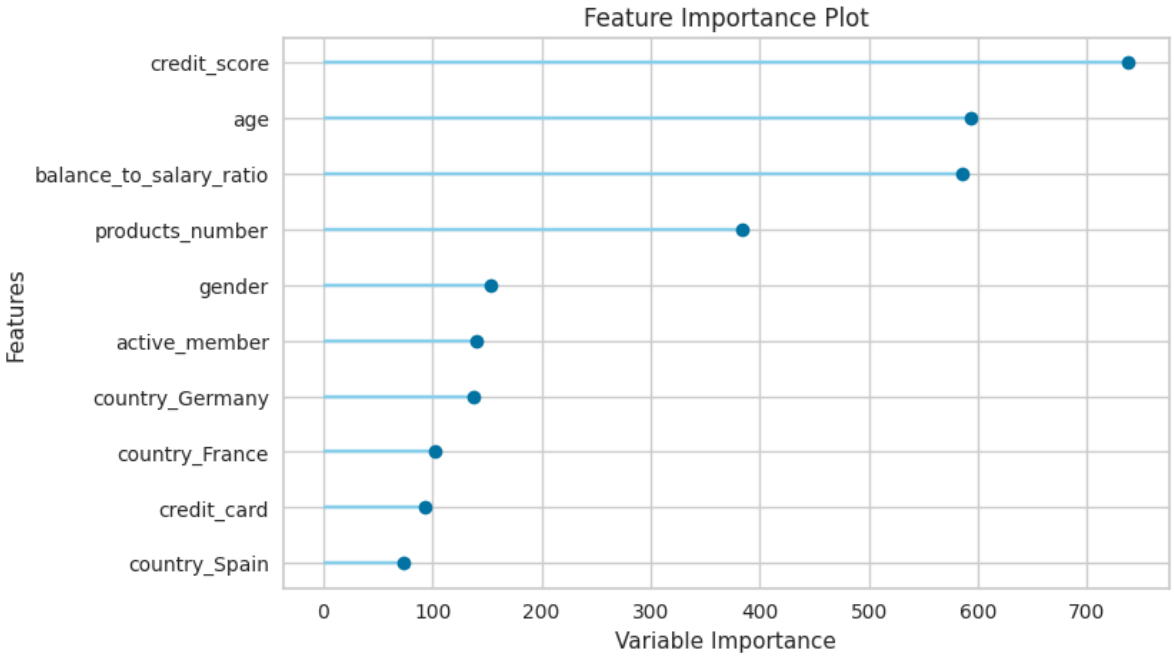

Figure 19 – The feature importance plot for Model 2.

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
lightgbm	Light Gradient Boosting Machine	0.8489	0.8557	0.5932	0.6402	0.6150	0.5213	0.5223
xgboost	Extreme Gradient Boosting	0.8377	0.8366	0.5589	0.6118	0.5836	0.4832	0.4842
rf	Random Forest Classifier	0.8340	0.8339	0.5602	0.5994	0.5788	0.4756	0.4763
et	Extra Trees Classifier	0.8314	0.8215	0.5357	0.5970	0.5636	0.4597	0.4614
gbc	Gradient Boosting Classifier	0.8267	0.8601	0.6662	0.5642	0.6103	0.5000	0.5033
ada	Ada Boost Classifier	0.8083	0.8410	0.6668	0.5230	0.5859	0.4636	0.4696
dummy	Dummy Classifier	0.7963	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
dt	Decision Tree Classifier	0.7766	0.6918	0.5491	0.4599	0.5002	0.3578	0.3604
qda	Quadratic Discriminant Analysis	0.7676	0.7991	0.6528	0.4515	0.5333	0.3854	0.3975
lr	Logistic Regression	0.7167	0.7525	0.6795	0.3885	0.4941	0.3172	0.3413
ridge	Ridge Classifier	0.7130	0.0000	0.6718	0.3838	0.4882	0.3091	0.3326
lda	Linear Discriminant Analysis	0.7129	0.7662	0.6718	0.3836	0.4881	0.3088	0.3324
nb	Naive Bayes	0.6890	0.7559	0.7160	0.3657	0.4840	0.2935	0.3275
knn	K Neighbors Classifier	0.6019	0.5819	0.4712	0.2486	0.3253	0.0800	0.0882
svm	SVM - Linear Kernel	0.5037	0.0000	0.6122	0.2264	0.3300	0.0500	0.0719

Figure 20 – The set of performances for Model 3.

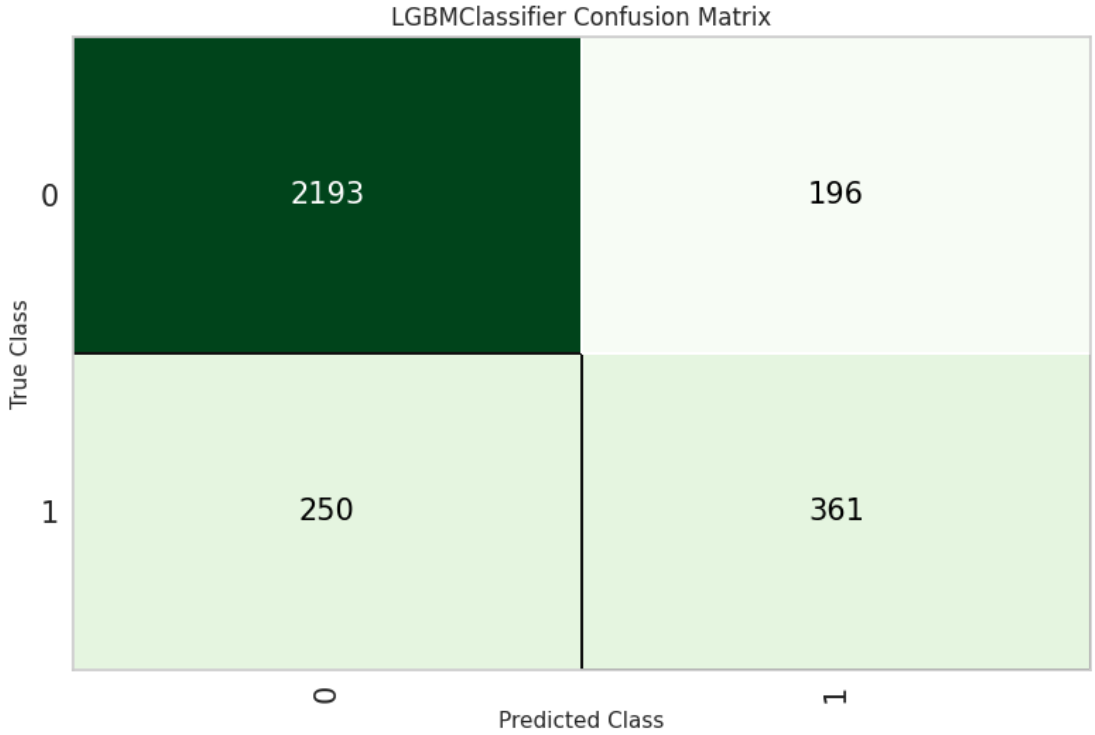

Figure 21 – The confusion matrix for Model 3.

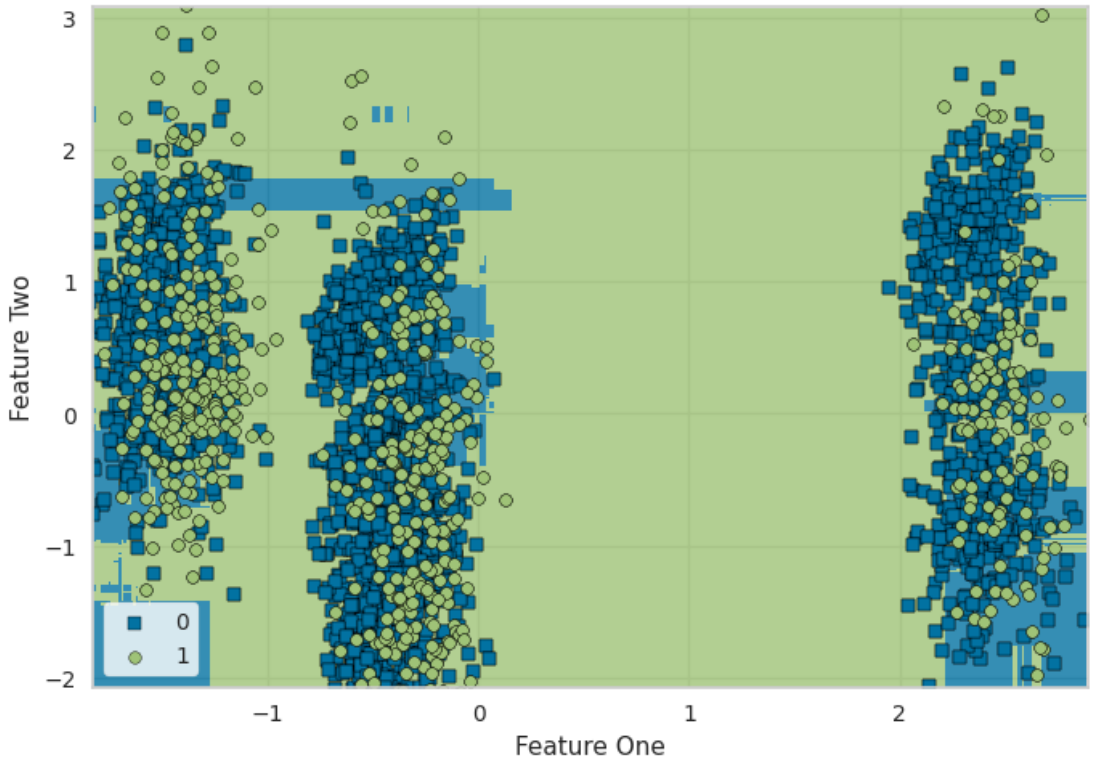

Figure 22 – The decision boundary for Model 3.

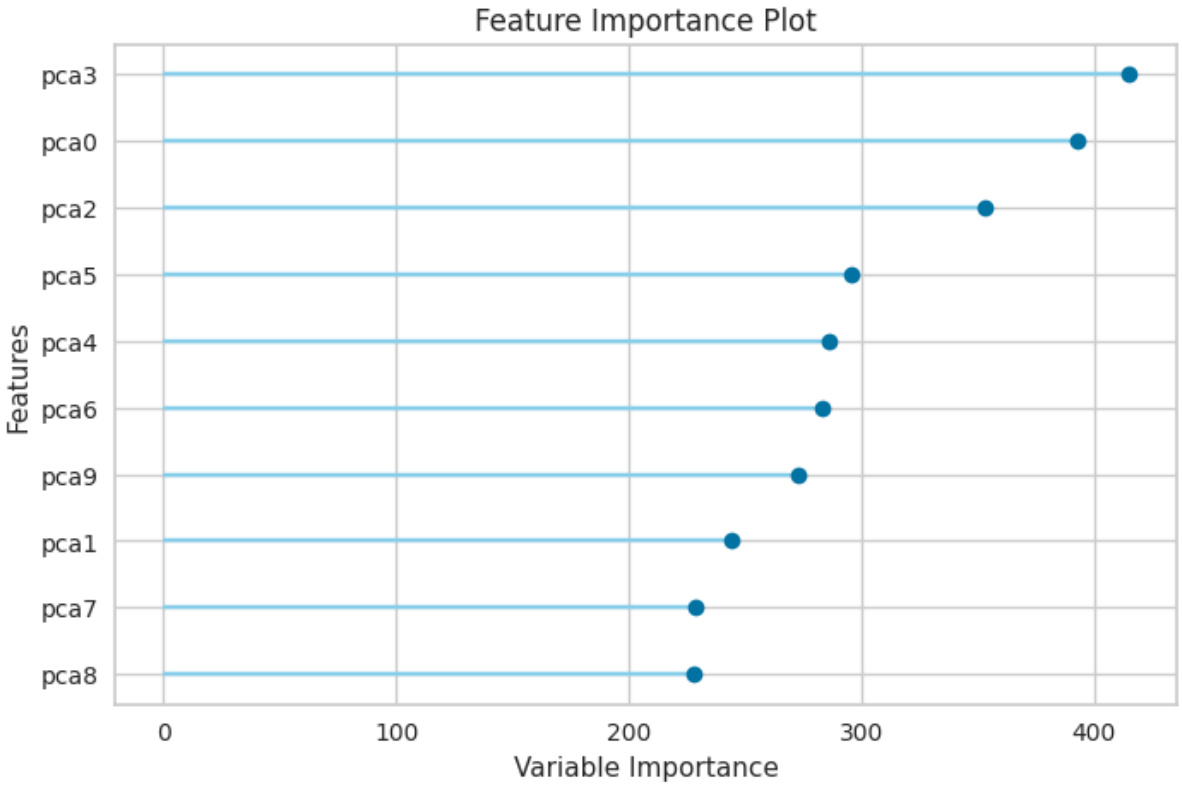

Figure 23 – The feature importance plot for Model 3.

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
lightgbm	Light Gradient Boosting Machine	0.8599	0.8619	0.5455	0.7002	0.6129	0.5290	0.5354
gbc	Gradient Boosting Classifier	0.8587	0.8671	0.5406	0.6978	0.6087	0.5243	0.5309
xgboost	Extreme Gradient Boosting	0.8544	0.8483	0.5350	0.6817	0.5993	0.5120	0.5177
rf	Random Forest Classifier	0.8516	0.8380	0.5160	0.6783	0.5847	0.4967	0.5043
ada	Ada Boost Classifier	0.8470	0.8500	0.5560	0.6451	0.5962	0.5027	0.5053
et	Extra Trees Classifier	0.8450	0.8234	0.5118	0.6534	0.5725	0.4798	0.4860
dummy	Dummy Classifier	0.7963	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
dt	Decision Tree Classifier	0.7937	0.6950	0.5287	0.4942	0.5102	0.3799	0.3806
ridge	Ridge Classifier	0.7130	0.0000	0.6718	0.3838	0.4882	0.3091	0.3326
lda	Linear Discriminant Analysis	0.7129	0.7662	0.6718	0.3836	0.4881	0.3088	0.3324
nb	Naive Bayes	0.6879	0.7556	0.7153	0.3645	0.4828	0.2916	0.3258
lr	Logistic Regression	0.6689	0.7201	0.6753	0.3428	0.4544	0.2521	0.2818
knn	K Neighbors Classifier	0.6013	0.5818	0.4712	0.2482	0.3250	0.0794	0.0876
svm	SVM - Linear Kernel	0.4986	0.0000	0.6827	0.2408	0.3236	0.0718	0.1298
qda	Quadratic Discriminant Analysis	0.4929	0.4955	0.4958	0.2227	0.2331	0.0007	-0.0154

Figure 24 – The set of performances for Model 4.

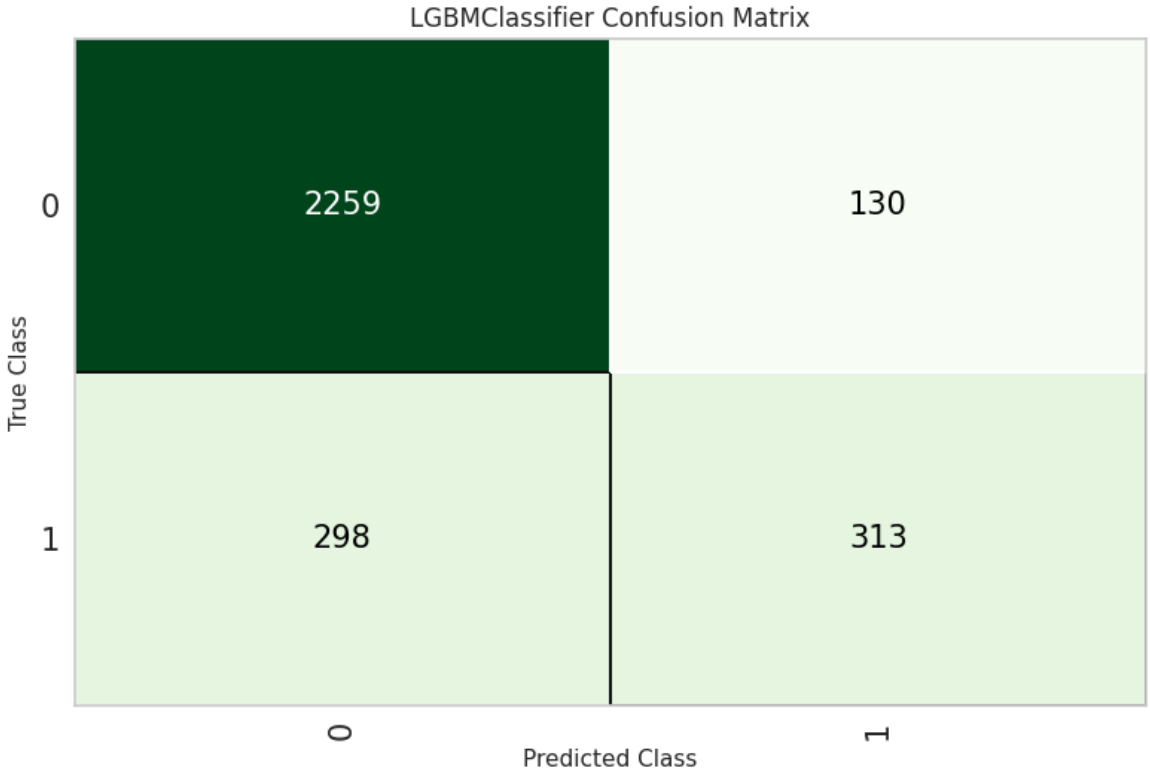

Figure 25 – The confusion matrix for Model 4.

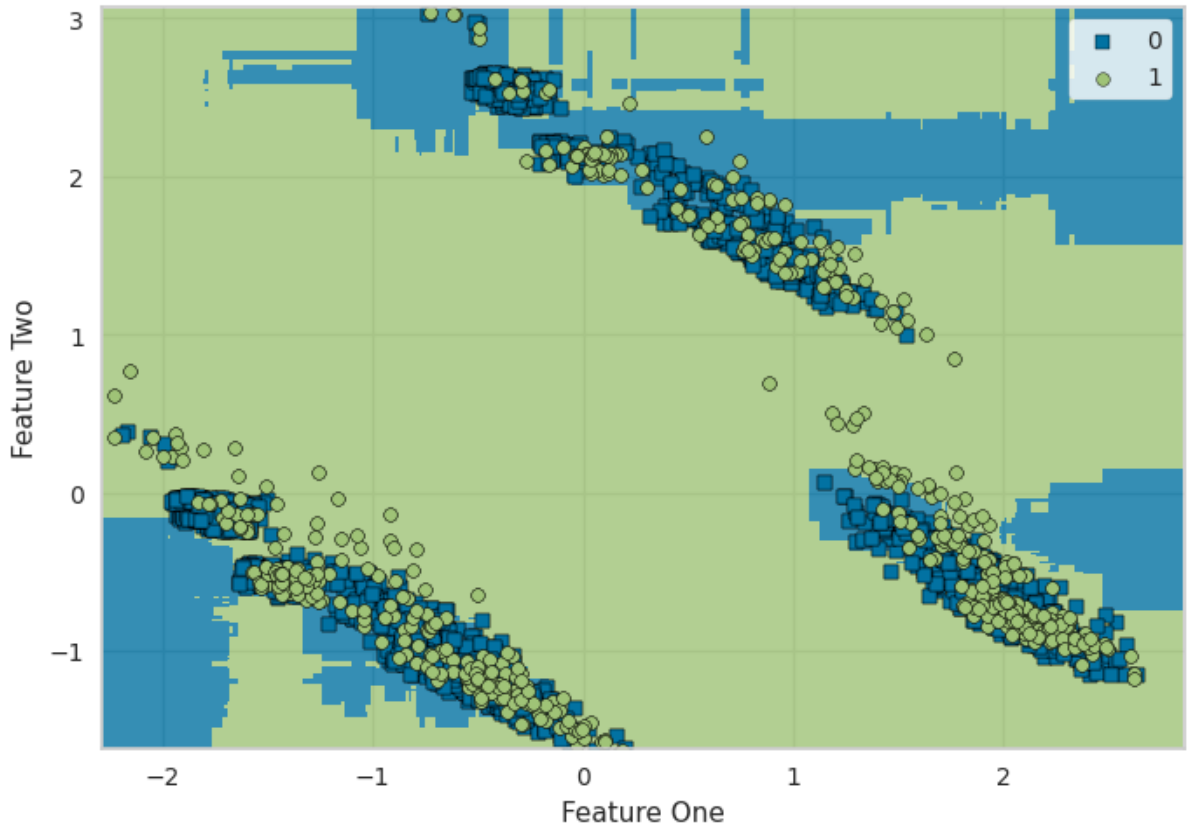

Figure 26 – The decision boundary for Model 4.

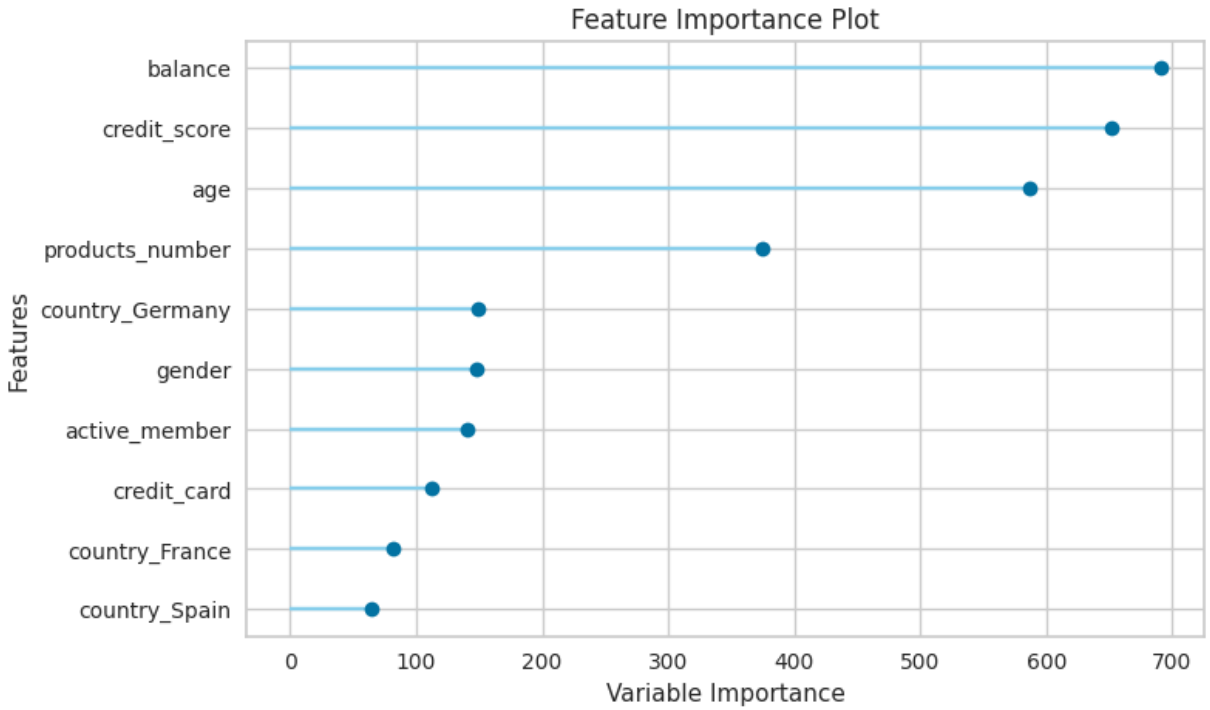

Figure 27 – The feature importance plot for Model 4.

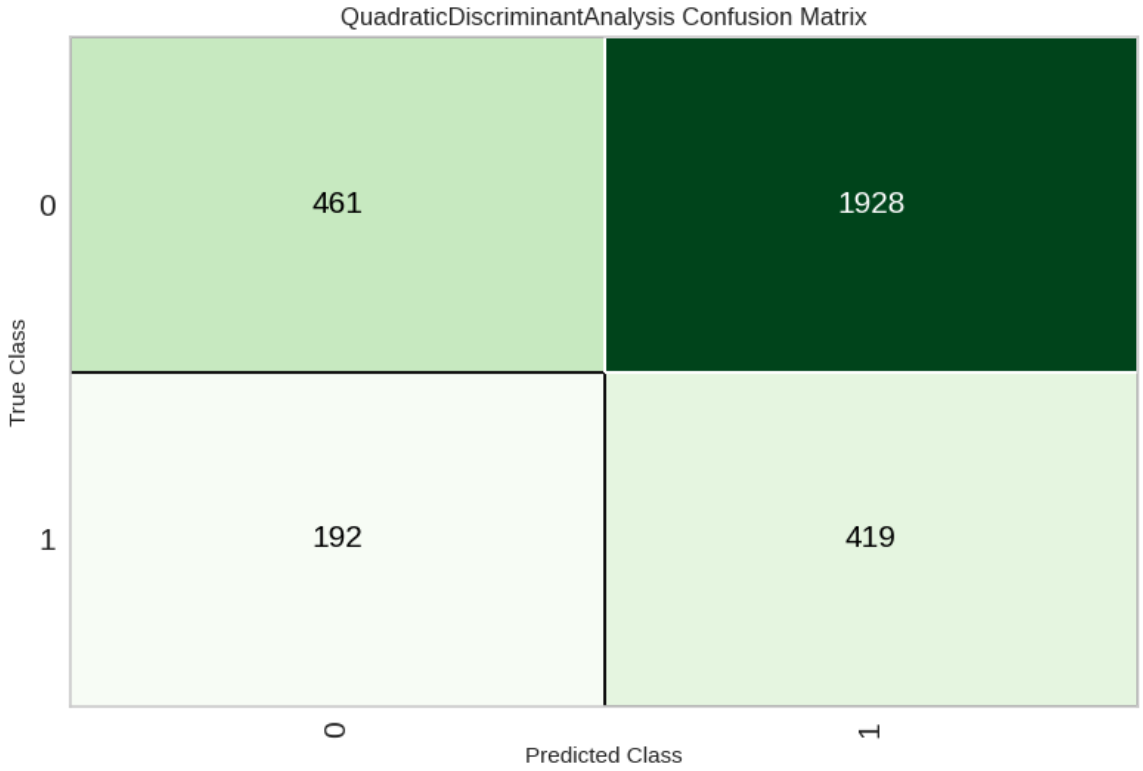

Figure 28 – The confusion matrix for the first QDA model created.

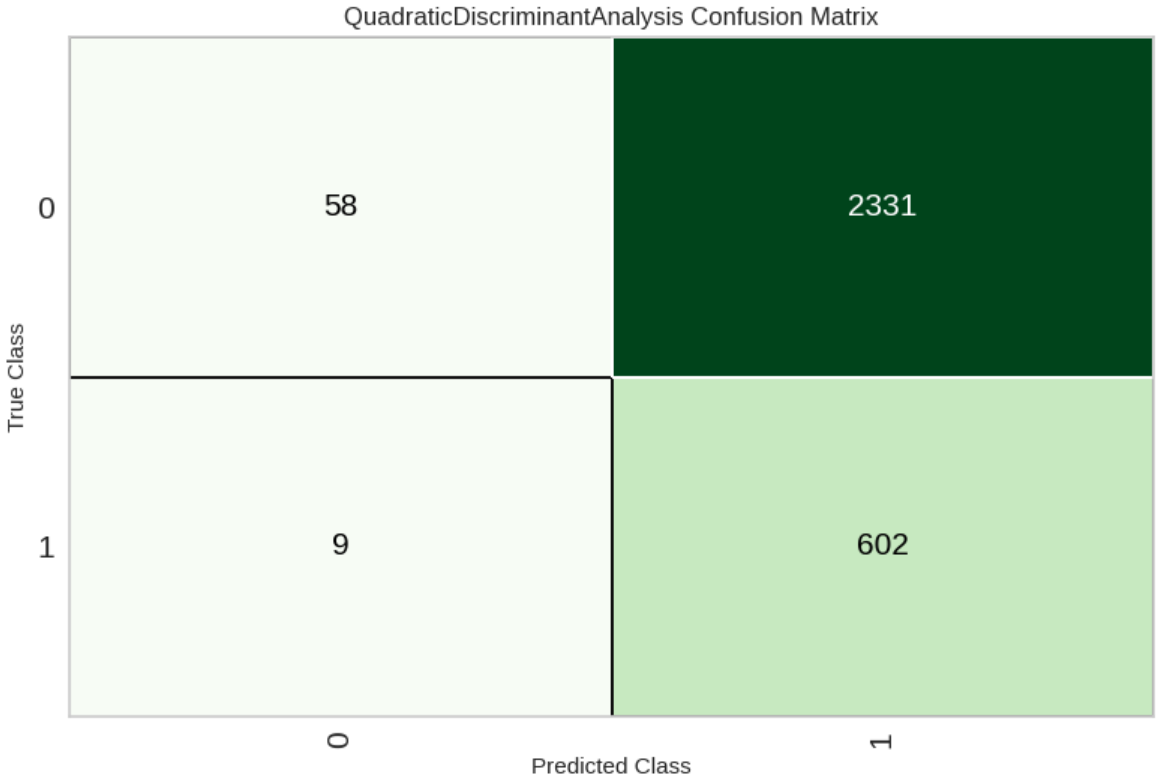

Figure 29 – The confusion matrix for the second QDA model.

References

Colab Project Link
https://colab.research.google.com/drive/1p68HiOaIdi0DcdntDf3RgioQXKZCIE36?usp=sharing

Kaggle Bank Churn dataset
https://www.kaggle.com/datasets/gauravtopre/bank-customer-churn-dataset

Pycaret documentation
https://pycaret.gitbook.io/docs/

Scikit-Learn documentation
https://scikit-learn.org/stable/modules/classes.html