ST 4052
Final Project

CREDIT CARD OFFER ACCEPTANCE TRENDS IN BANKING

Group 04

S14853 - PRAMUDI RAJAMANTHRI
S15030 - VIDURA CHATHURANGA
S15091 - CHALANI WIJAMUNIGE

Abstract

This study aims to construct a comprehensive descriptive analysis and fit predictive models with high accuracies for a dataset obtained from the platform 'Kaggle' which focuses on identifying the reasons behind customers accepting or rejecting a credit card offers offered by banks. The dataset comprises of records of 18,000 customers with 18 variables containing both categorical and numerical. At the completion of this analysis, the insights gained about credit card offer acceptance trends and possibilities of future acceptance are summarized in this report alongside an accurate predictive model.

Content

List of Figures

List of Tables

1. Introduction

In the everchanging landscape of financial transactions, credit cards emerge as one of the leading methods of payment, shaping the dynamics of modern commerce. To remain competitive, banks strategically offer a variety of credit card promotions, aiming to expand their customer base. However, the pivotal challenge faced by financial institutions lies in understanding the types of offers most likely to be embraced by customers. This data analysis project delves into the comprehensive credit card offer acceptance information of a bank, encompassing records from 18,000 customers and 18 distinct features. By unraveling the complex fabric of customer behavior and identifying the subtle reasons behind offer acceptance or rejection, this endeavor seeks to provide actionable insights into credit card offer acceptance trends. The ultimate objective is to leverage machine learning techniques to predict the likelihood of a customer accepting a credit card offer, offering the financial industry a predictive tool for refining their promotional strategies.

2. Description of the problem

The problem at hand revolves around the challenge faced by banks in identifying credit card offer acceptance trends. With credit cards being a predominant payment method, banks aim to expand their customer base by strategically offering various promotions. The goal of this data analysis project is to uncover the factors influencing customer decisions and employ machine learning techniques to predict whether a customer will accept a credit card offer, providing valuable insights for refining promotional strategies.

3. Objectives of the study

1. Investigate and understand the diverse factors influencing customer behavior in credit card offer acceptance within the dataset of 18,000 customers and 18 distinct features.

2. Uncover actionable insights that reveal the subtle reasons behind customers accepting or rejecting credit card offers, contributing to a comprehensive understanding of offer acceptance trends.

3. Develop and implement machine learning techniques to predict the likelihood of a customer accepting a credit card offer, providing financial institutions with a valuable predictive tool to enhance and refine their promotional strategies.

4. Description of the data set

"Credit Card Offer Acceptance Trends in Banking" dataset contains credit card offer acceptance information of a bank. There are records of 18,000 customers and 18 features (2 unique valued, 8-Qualitative and 8-Quantitative) with the target variable indicating whether an offer is accepted or not.

Source: Credit Card Offer Acceptance Trends in Banking (kaggle.com)

No	Variable	Description	Type
01	Index	Index	Unique
02	Customer Number	Unique identifier for each customer.	Unique
03	Offer Accepted	Whether the customer accepted the offer. (Yes, No)	Response
04	Reward	Type of reward offered. (Air Miles, Cash Back, Points)	Qualitative
05	Mailer Type	Type of mailer used to send the offer. (Letter, Postcard)	Qualitative
06	Income Level	The customer's income level. (High, Medium, Low)	Qualitative
07	Bank Accounts Open	Number of bank accounts the customer has opened.	Quantitative
08	Overdraft Protection	Whether the customer has overdraft protection on their account. (Yes, No)	Qualitative
09	Credit Rating	How well the customer's payment record reflects their ability and willingness to repay the debt under terms accepted by creditors. (High, Medium, Low)	Qualitative
10	Credit Cards Held	Number of credit cards the customer hold.	Quantitative
11	Homes Owned	Number of homes the customer has.	Quantitative
12	Household Size	Size of household that the customer belongs is kept track of. (1,2,3,4,5,6,7,8,9)	Qualitative
13	Own Your Home	Whether the customer owns their home. (Yes, No)	Qualitative
14	Average Balance	Average balance across all accounts.	Quantitative
15	Q1 Balance	The customer's balance in 1st quarter.	Quantitative
16	Q2 Balance	The customer's balance in 2nd quarter.	Quantitative
17	Q3 Balance	The customer's balance in 3rd quarter.	Quantitative
18	Q4 Balance	The customer's balance in 4th quarter.	Quantitative

Table 4.1

5. Data preprocessing

- Index Variables was dropped from the data set as it didn't provide any meaningful contribution to the study.
- There were no duplicate records were identified in the dataset.
- 24 missing records were detected in each of the Q1 Balance, Q2 Balance, Q3 Balance, and Q4 Balance variables.
- Those missing values were imputed using the mean values of each variable from the training set.
- Initially, the Household Size variable had nine distinct values.
- The Household Size variable was recoded: Categories 1, 2, 3, 4 were combined into a new category named "Small," and categories 5, 6, 7, 8, 9 were combined into a new category named "Big."
- Finally, the dataset was randomly split into training and test datasets which contains 80% and 20% of the entries from original dataset.

6. Important results of the Descriptive Analysis

The pie chart (Figure 6.1) illustrates a notable imbalance in class distribution within the training set, with the representation of the "Yes" category significantly smaller compared to the prevalence of the "No" category. The dataset exhibits a pronounced imbalance, prompting the need for careful consideration and appropriate handling of this skewed distribution during further analysis.

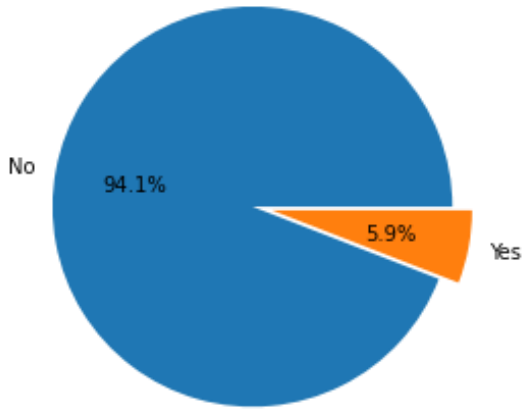


Figure 6.1

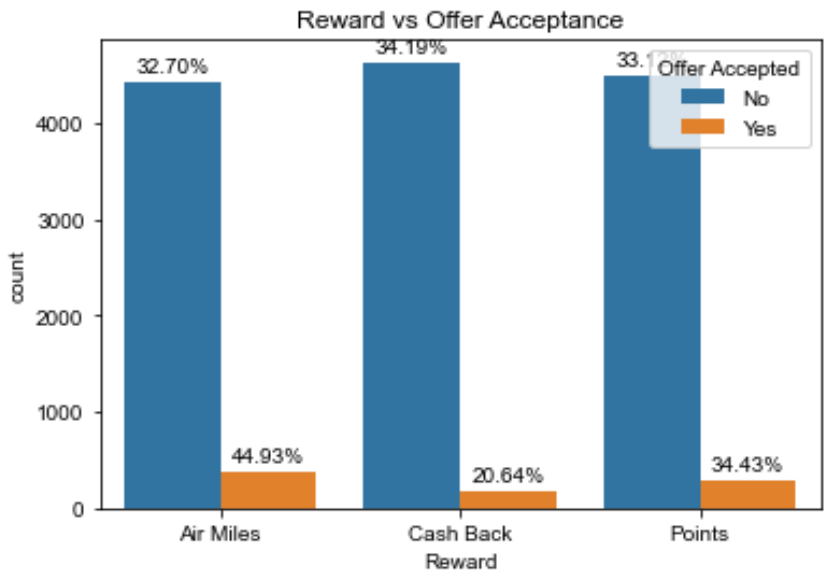


Figure 6.2

The Figure 6.2 plot is used to analyze the relationship between the type of reward received and the response variable - Offer Acceptance. Upon observation, it is apparent that the distribution of not accepting the offer is relatively consistent across all categories of the 'Reward' variable. However, in the case of the 'Accepting' category, there is a slight variation among the three reward categories. This suggests that rewards may play a significant role in influencing credit card offer acceptance.

These findings align with insights from the American Bankers Association, as mentioned in their articles. According to them, the analysis, coupled with other studies, indicates that card rewards are accessible, valuable, and well-understood by consumers across all income levels. Credit card offers, as revealed by this analysis, indeed exert a noteworthy influence on the acceptance or rejection of credit card offers. *(https://www.aba.com/news-research/analysis-guides/the-benefits-of-credit-card-rewards)*

This visualization (Figure 6.3) illustrates the relationship between the response variable 'Offer Accepted' and the 'Number of Bank Accounts Opened' variable. However, the overall distribution of acceptance and rejection of credit card offers appears similar across the three categories of the 'Number of Bank Accounts Opened' variable. This suggests that the number of bank accounts alone may not be a decisive factor in determining credit card offer acceptance, as the distribution remains consistent across different account counts.

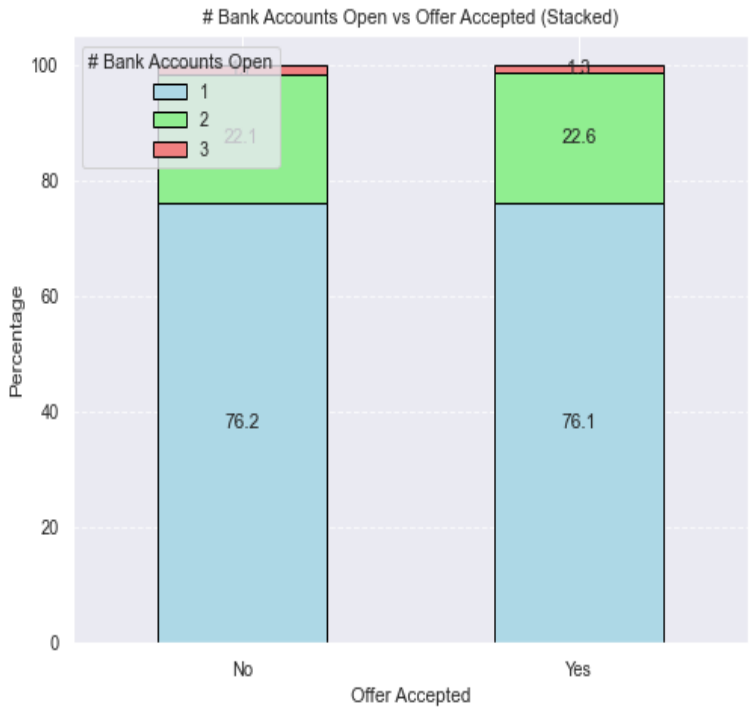


Figure 6.3

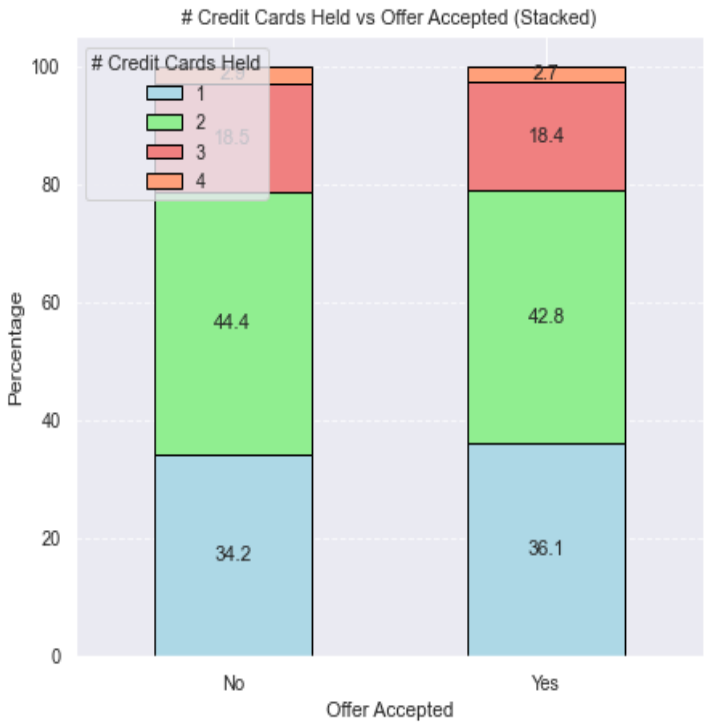


Figure 6.4

The presented visualization (Figure 6.4) showcases the correlation between the response variable 'Offer Accepted' and the 'Number of Credit cards held' variable. Notably, the distribution patterns for both acceptance and rejection of credit card offers seem consistent across the four categories of the 'Number of Credit cards held' variable. This implies that the singular consideration of the number of credit cards held may not be a determining factor in credit card offer acceptance, as the distribution remains uniform across varying counts of credit cards held.

The depicted plot (Figure 6.5) visualizes the relationship between the variables "No of Houses Owned" and "Offer Accepted." In this portrayal, the distribution among the categories of the "No of Homes Owned" variable appears relatively consistent for both categories of the "Offer Accepted" variable. This observation suggests that the number of homes owned may not exert a significant influence on the acceptance of credit cards, as the distribution remains comparable across different counts of homes owned.

Figure 6.5

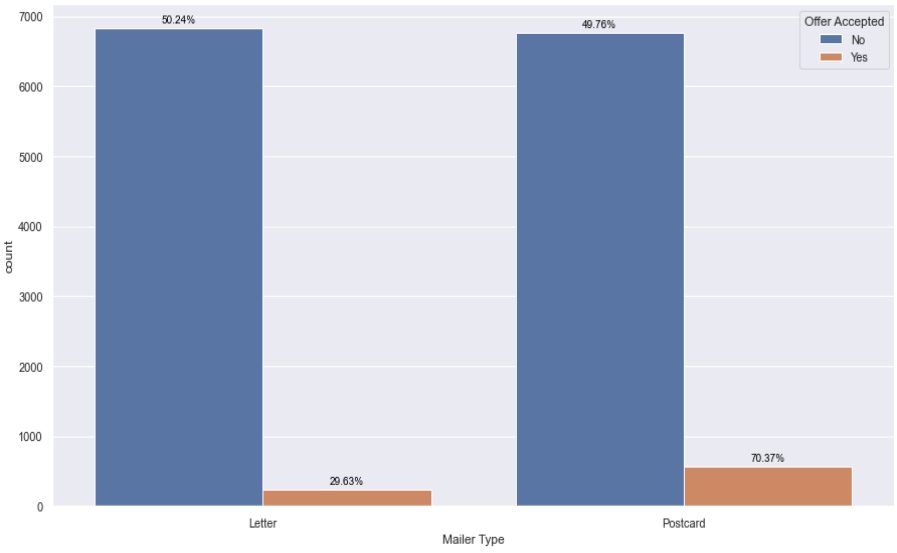


Figure 6.6

"In our examination of the Credit Rating variable's impact on credit card acceptance, a notable and substantial behavioral difference becomes evident between accepted and rejected incidents. This pronounced contrast strongly indicates a significant association between these variables, emphasizing the pivotal role of Credit Rating in determining credit card acceptance. The discernible impact implies that a deeper analysis and careful consideration of Credit Rating are essential for gaining comprehensive insights into the dynamics of customer responses to credit card offers.

Regarding the Mailer type (Figure 6.6), while the distribution among the "Not accepted" category remains similar, there is a noticeable distinction between the "Accepted" category concerning the two mailer types, "Letter" and "Postcard." This discrepancy suggests a meaningful impact of the Mailer type on the acceptance of credit cards by customers. The observed significant difference in distribution implies that the method of communication, whether through letters or postcards, plays a substantial role in influencing customer decisions regarding credit card offers.

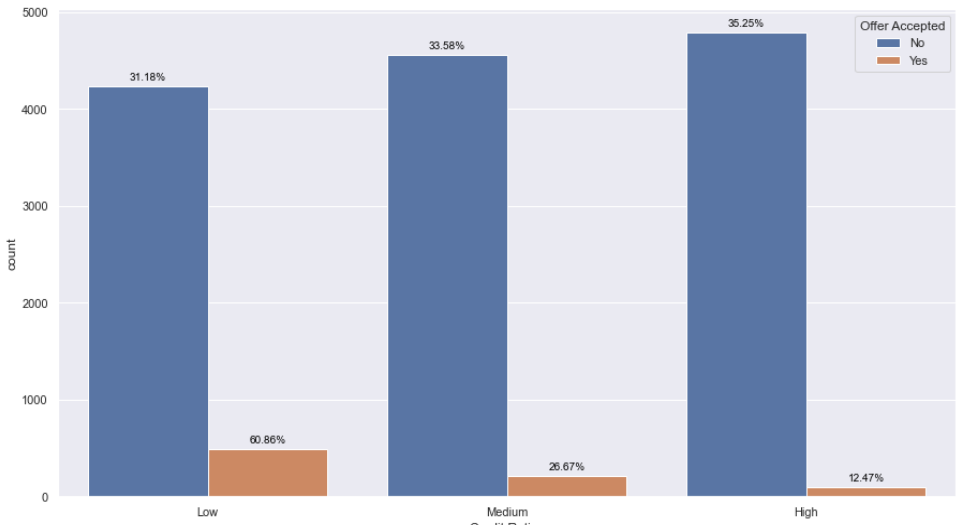


Figure 6.7

The importance of credit scores is reiterated in an article by Investopedia (https://www.investopedia.com/articles/pf/07/credit_card_rating.asp#:~:text=Your%20credit%20score%20is%20an,you%20are%20able%20to%20borrow.), where it is highlighted that your credit score plays a crucial role in your ability to obtain a credit card or borrow money. The ideas presented in this reference align with and validate the results obtained in our analysis."

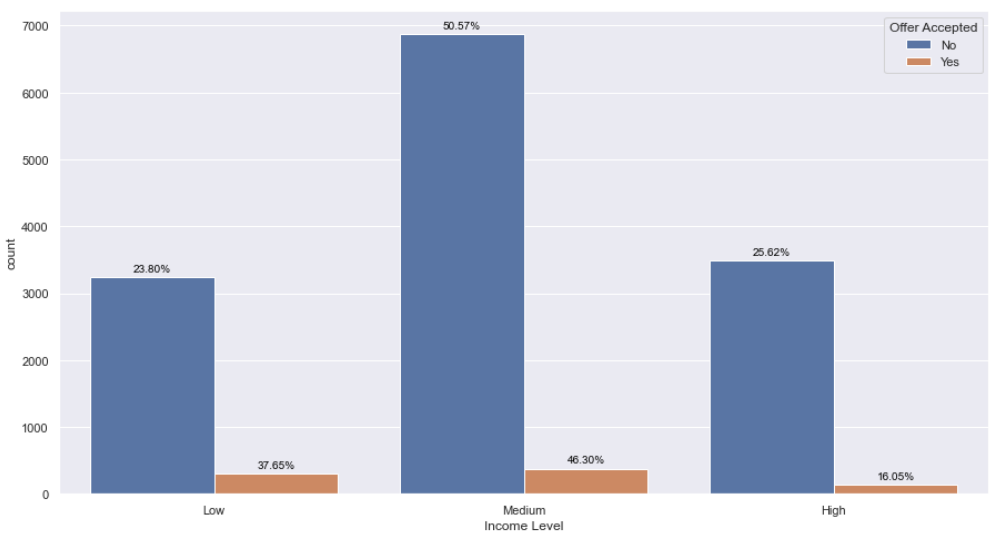


Figure 6.8

This analysis underscores the pivotal role of income level as a decisive factor influencing credit card acceptance. The distributions and proportions of acceptance and rejection manifest notable variations across the distinct categories of the 'Income Level' variable as shown in fugure 6.8.

This observation aligns cohesively with insights discussed in articles from CNBC (https://www.cnbc.com/select/how-does-salary-and-income-impact-your-credit-score/) and IndusInd Bank (https://www.indusind.com/iblogs/categories/manage-your-finance/the-crucial-role-of-income-in-credit-card-approval/)

In the latter, it is emphasized that credit card approval transcends merely having a good credit score, with income being a paramount factor considered by banks. These references fortify our findings, highlighting the intrinsic connection between income and credit card approval decisions. They underscore the imperative nature of factoring in diverse income levels in the assessment of credit card acceptance trends.

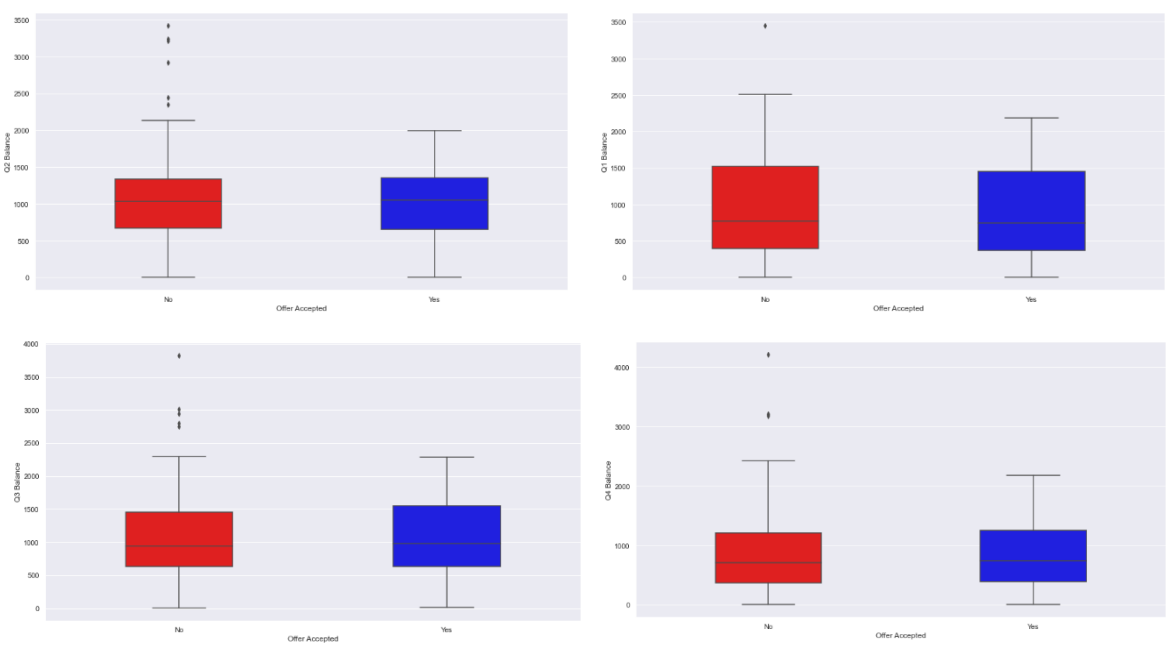


Figure 6.9

Next, we need to assess the relationship between quantitative variables and the response variable, Offer Acceptance. Upon analyzing the impact of customer balances in each quarter of the year, we have uncovered significant insights into their behavior (Figure 6.9).

In each quarter, the distribution of balances for the "Offer Not Accepted" category appears considerably wider compared to the "Accepted" category. Moreover, numerous outliers are observed specifically within the "Not Accepted" category. The distributions of balances in each quarter exhibit noticeable differences between the two acceptance categories. These findings suggest that there is a substantial impact of balances in Q1, Q2, Q3, and Q4 on the acceptance of credit cards by customers.

Turning our attention to the numerical variable of Average Balance, insights derived from the boxplots unveil notable patterns. The distribution of Average Balance associated with the "Offer Not Accepted" category appears broader compared to the distribution for the "Offer Accepted" category. Moreover, a higher prevalence of outliers is evident in the "Not Accepted" category. These distinct distribution patterns strongly indicate a substantial impact of Average Balance on offer acceptance. The observed differences in distribution strongly suggest that Average Balance plays a pivotal role in determining the likelihood of offer acceptance. Coincidentally, The Nest discusses a related notion in their report titled "Do Credit Cards Check Your Bank Balance Before Issuing You a Card?" (source: https://budgeting.thenest.com/credit-cards-check-bank-balance-before-issuing-card-24697.html), aligning closely with our findings and emphasizing the significance of Average Balance in credit card acceptance dynamics.

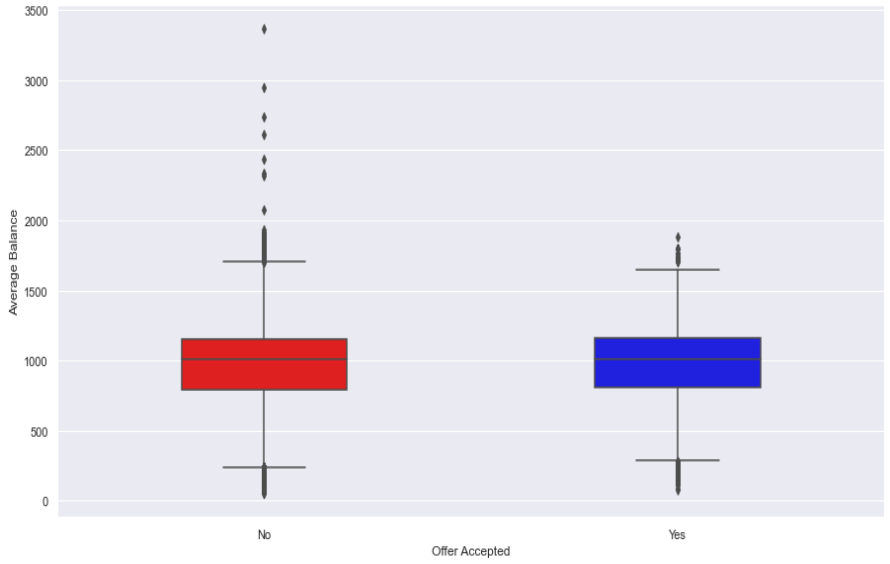


Figure 6.10

Multiple Correspondence Analysis for Categorical Variables

In the Multiple Correspondence Analysis conducted for the categorical variables, it was observed that the first 2 components explain 20.41% of the entire variance. Considering the association of each variable with the response variables, the information uncovered on the surface analysis are further clarified as the variables Reward, Mailer Type, Income Level, Credit Rating shows a high association with the response variable. But those are scattered around the category 'Offer_Accepted_No', which suggests that those are more related towards that category while 'Offer_Accepted_Yes' category shows distant relationships. The reason behind this can be the imbalanced nature of the dataset, as when there are plenty of datapoints, that category captures most of the variance.

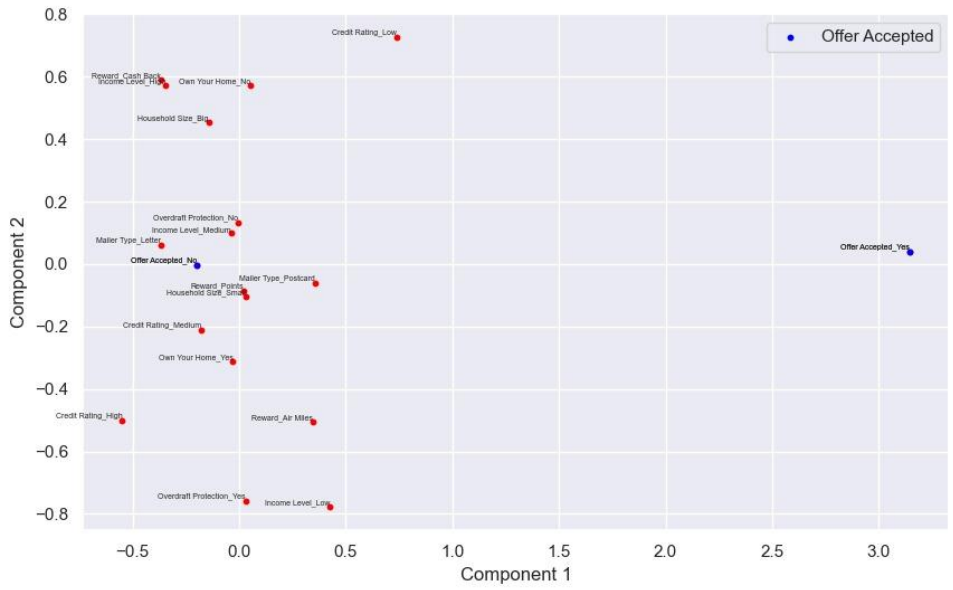


Figure 6.11

Partial Least Square – Discriminant Analysis

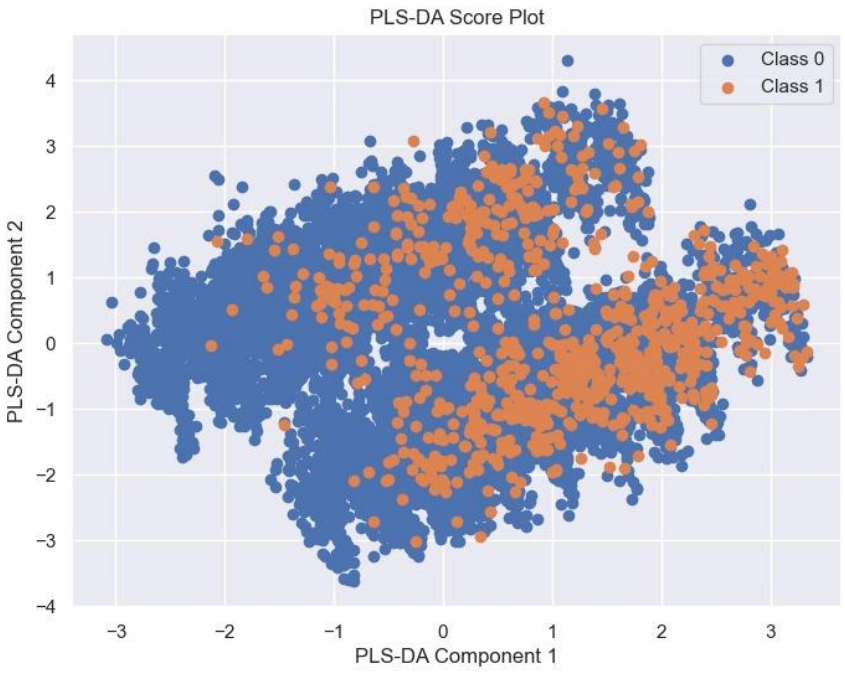


Figure 6.12

The variance explained by the first 2 components of PLS – DA is approximately added up to 18.13% which is considerably low. Therefore, the findings of this analysis cannot be clearly interpreted and according to the score plot where there are no distinct clusters to be observed. But a certain pattern is quite visible in the scattering of scores in the plot which can be addressed as a highlighting fact.

Spearman Correlation of Continuous Variables.

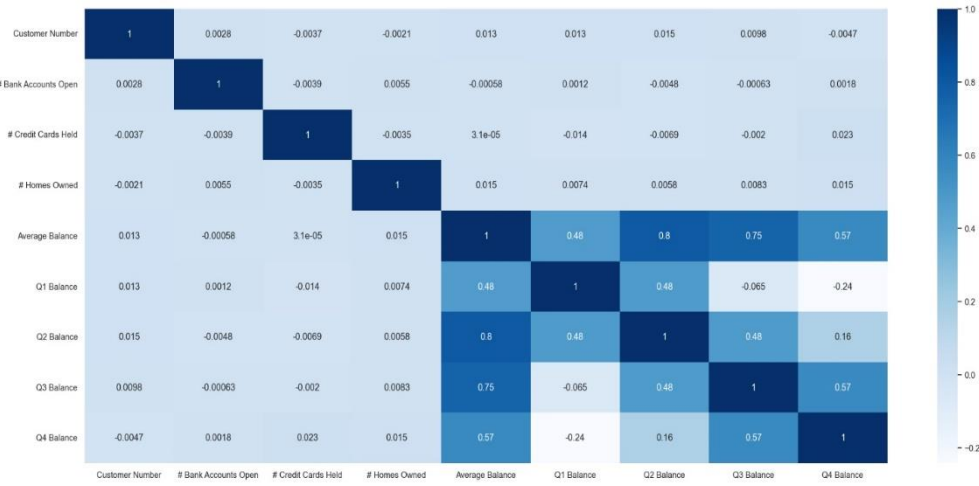


Figure 6.13

Considering the correlations between continuous variables of the dataset, only the Average Balance and Quarterly Balances shows some correlation and the highest correlation is shown between Average Balance and Q2 Balance which is 0.8. Therefore, a significant multicollinearity is not observed. Due to this, its possible to start model fitting, keeping logistic regression as a baseline.

7. Important results of the Advanced Analysis

In the advanced analysis phase, the main objective is to delve deep into the classifying the acceptance of credit card offers based on customer details. It is aimed to evaluate performance of a range of predictive models in their capacity to accurately classify instances based on the 'Offer Accepted' variable. Since, no significant multicollinearity was observed in the descriptive study, models will be chosen keeping logistic regression as a baseline. To initiate the model fitting, Logistic Regression, Logistic Ridge, Logistic Lasso, K-Nearest Neighbor, Gaussian Naïve Bays, Support Vector Machine, Random Forest, and XGBoost models were fitted and evaluated.

	Without Applying Balancing Techniques															
	Training Set								Testing Set							
	Logistic Regression	Logistic Ridge	Logistic Lasso	KNN	Gaussian Naïve Bayes	SVM	Random Forest	XGBoost	Logistic Regression	Logistic Ridge	Logistic Lasso	KNN	Gaussian Naïve Bayes	SVM	Random Forest	XGBoost
Accuracy	0.94	0.94	0.94	0.94	0.94	0.94	1.0	0.97	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
Precision	0.47	0.47	0.47	0.75	0.47	0.47	1.0	0.98	0.48	0.48	0.48	0.54	0.48	0.48	0.48	0.58
Recall	0.5	0.5	0.5	0.52	0.5	0.5	1.0	0.72	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
F1 score	0.48	0.48	0.48	0.52	0.48	0.48	1.0	0.79	0.49	0.49	0.49	0.5	0.49	0.49	0.49	0.5

Table 7.1

According to the results given in the table, almost every model has not been successful in predicting the target class which is the acceptance of the offer as the evaluation metrics precision, recall and F1 score show values ranging from 45% to 60%. The reason behind these low metric values can be again the extreme imbalanced nature of the dataset as the number of offer accepted cases is significantly low compared to offer rejected cases.

To overcome this, 2 balancing techniques which are Synthetic Minority Oversampling Technique (SMOTE) and Random Under Sampling Technique, were introduced in order to improve precision and recall which represents the accuracy in identifying the positive cases (accepted cases) as that's where the main focus of this study is relied on.

	With SMOTE															
	Training Set								Testing Set							
	Logistic Regression	Logistic Ridge	Logistic Lasso	KNN	Gaussian Naïve Bayes	SVM	Random Forest	XGBoost	Logistic Regression	Logistic Ridge	Logistic Lasso	KNN	Gaussian Naïve Bayes	SVM	Random Forest	XGBoost
Accuracy	0.83	0.83	0.83	0.99	0.83	0.81	1.0	0.93	0.83	0.83	0.83	0.83	0.83	0.83	0.88	0.83
Precision	0.55	0.55	0.55	0.99	0.55	0.55	1.0	0.71	0.55	0.55	0.55	0.55	0.55	0.55	0.8	0.65
Recall	0.63	0.63	0.63	0.89	0.61	0.64	1.0	0.82	0.66	0.66	0.66	0.66	0.66	0.66	0.81	0.66
F1 score	0.56	0.56	0.56	0.94	0.55	0.55	1.0	0.75	0.56	0.56	0.56	0.56	0.56	0.56	0.8	0.66

Table 7.2

	With Random Under Sampling															
	Training Set								Testing Set							
	Logistic Regression	Logistic Ridge	Logistic Lasso	KNN	Gaussian Naïve Bayes	SVM	Random Forest	XGBoost	Logistic Regression	Logistic Ridge	Logistic Lasso	KNN	Gaussian Naïve Bayes	SVM	Random Forest	XGBoost
Accuracy	0.83	0.83	0.83	0.78	0.81	0.83	0.97	0.89	0.87	0.87	0.83	0.77	0.82	0.83	0.9	0.76
Precision	0.56	0.56	0.55	0.54	0.55	0.56	0.83	0.66	0.56	0.56	0.55	0.52	0.55	0.55	0.56	0.63
Recall	0.65	0.65	0.65	0.89	0.63	0.65	0.89	0.86	0.63	0.63	0.66	0.59	0.64	0.66	0.58	0.62
F1 score	0.57	0.57	0.56	0.94	0.55	0.57	0.86	0.71	0.57	0.57	0.56	0.51	0.55	0.56	0.56	0.61

Table 7.3

After balancing the dataset using both the balancing techniques, the performances of all models have improved significantly. By carefully examining, we identified that Random Forest and XGBoost in both scenarios perform better in term of all the evaluation metrics and also, they show considerably less overfitting. Therefore, to further improve their performance, hyper parameter tuning was applied.

	With SMOTE			
	Random Forest		XGBoost	
	Training Set	Test Set	Training Set	Test Set
Accuracy	1.0	0.89	0.97	0.87
Precision	1.0	0.85	0.94	0.74
Recall	1.0	0.81	0.95	0.79
F1 score	1.0	0.83	0.88	0.75

Table 7.4

Best Parameters (Random Forest) : {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}
Best Parameters (XGBoost) : {'learning_rate': 0.2, 'max_depth': 7, 'min_child_weight': 1, 'n_estimators': 200, 'subsample': 0.8}

	With Random Under Sampling			
	Random Forest		XGBoost	
	Training Set	Test Set	Training Set	Test Set
Accuracy	0.77	0.73	0.81	0.75
Precision	0.77	0.64	0.75	0.54
Recall	0.81	0.5	0.81	0.61
F1 score	0.72	0.57	0.82	0.55

Table 7.5

Best Parameters (Random Forest) : {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 200}
Best Parameters (XGBoost) : {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight': 3, 'n_estimators': 100, 'subsample': 0.8}

As observed in these 2 tables, after applying the tuned parameters, the improvement in precision and recall is quite evident, By comparing the model performance under 2 balancing techniques, it is visible that the models fitted to the dataset balanced using SMOTE performs better than the models fitted to the under sampled dataset. It also should be noted that there is a possible information loss when using the random under sampling technique compared to the oversampling technique. Therefore, the study will be carried further on the dataset balanced with SMOTE.

Considering the performance of Random Forest and XGBoost with SMOTE, as both models show better performance, both were combined to form an ensemble model using the voting classifier using soft voting.

	Voting Classifier	
	Training Set	Testing Set
Accuracy	0.99	0.92
F1-Score	0.96	0.92
Precision	0.99	0.88
Recall	0.97	0.9

Table 7.6

As observed in the table, the voting classifier shows a significant improvement in performance as the training accuracies are almost 100% in terms of all evaluation metrics.

Now to further reduce the complexity of the models as there are 17 predictor variables, the dimensionality of the dataset was reduced. According to the variable importance plot obtained from Random Forest, all the initial models were refitted considering the most important 15, 13 and 10 variables sequentially.

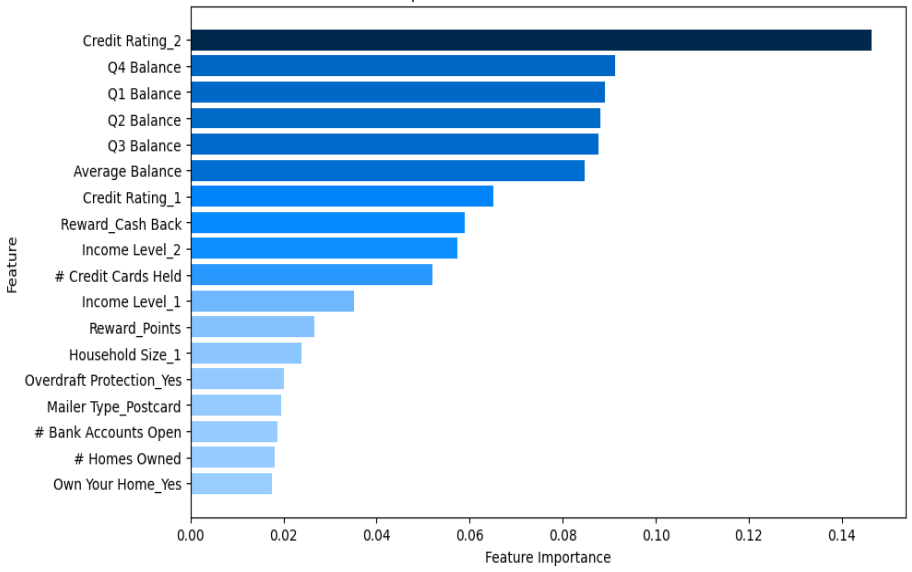


Figure 7.1

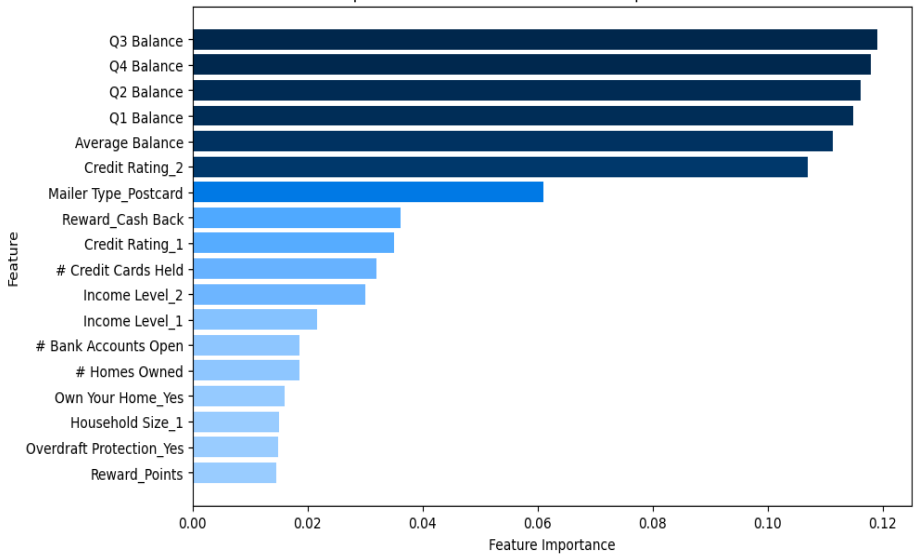


Figure 7.2

According to the performances of the reduced models, the model with top 10 most important features which are 'Q1 Balance', 'Q2 Balance', 'Q3 Balance', 'Q4 Balance', 'Average Balance', 'Credit Rating_2', 'Mailer Type_Postcard', 'Reward_Cash Back', 'Credit Rating_1', '# Credit Cards Held', gave better and justifiable results in term of all the evaluation metrics as given in the following table. Here the models are fitted on the dataset balanced using SMOTE.

	Training Set								Testing Set							
	Logistic Regression	Logistic Ridge	Logistic Lasso	KNN	Gaussian Naïve Bayes	SVM	Random Forest	XGBoost	Logistic Regression	Logistic Ridge	Logistic Lasso	KNN	Gaussian Naïve Bayes	SVM	Random Forest	XGBoost
Accuracy	0.86	0.86	0.86	0.99	0.95	0.83	1.0	0.93	0.95	0.95	0.95	0.83	0.86	0.95	0.87	0.84
Precision	0.56	0.56	0.56	1.0	0.75	0.55	1.0	0.73	0.57	0.57	0.55	0.83	0.56	0.48	0.83	0.72
Recall	0.61	0.61	0.61	0.83	0.71	0.64	1.0	0.77	0.51	0.51	0.5	0.82	0.61	0.4	0.81	0.66
F1 score	0.57	0.57	0.57	0.86	0.72	0.55	1.0	0.75	0.5	0.5	0.5	0.82	0.57	0.49	0.82	0.72

Table 7.7

According to the accuracies shown in the table, it is identified that models K-Nearest Neighbor, Random Forest and XGBoost show the best performance out of all the fitted models under the reduced variable scenario. Further to improve the performance and to ensure minimum overfitting of Random Forest and XGBoost, those 2 models were hyper parameter tuned.

	Random Forest		XGBoost	
	Training Set	Test Set	Training Set	Test Set
Accuracy	1.0	0.85	0.93	0.86
Precision	1.0	0.83	0.75	0.84
Recall	1.0	0.87	0.94	0.71
F1 score	1.0	0.83	0.79	0.75

Table 7.8

Since all the 3 models, which are the Reduced KNN model, Reduced and hyper parameter tuned Random Forest and XGBoost models still shows high accuracies, to collectively capture their intelligence, an ensemble model was fitted using the voting classifier with ‘soft’ voting. The evaluation metrics of the voting classifier is as follows.

	Voting Classifier	
	Training Set	Testing Set
Accuracy	0.97	0.92
F1-Score	0.92	0.83
Precision	0.85	0.82
Recall	0.97	0.83

Table 7.9

It is evident and clearly observable that the ensemble model capturing the KNN, hyper parameter tuned Random Forest and XGBoost models, which is the voting classifier build upon the dataset balanced using SMOTE with reduced dimensionality outperforms all the models which were fitted so far in terms of simplicity, high accuracies, and less overfitting.

Partial Dependence plots

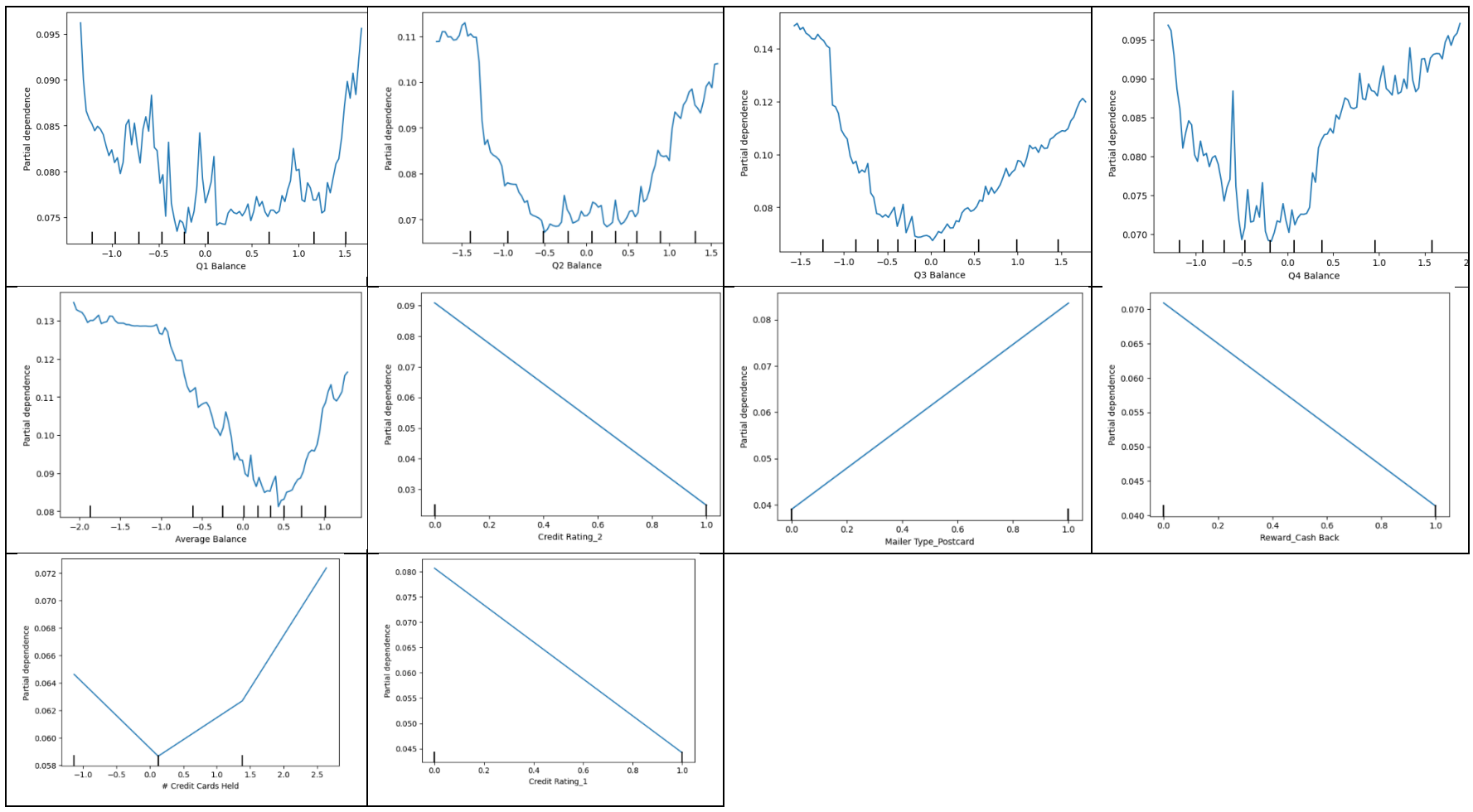


Figure 7.3

Observing the above partial dependence plots considering the selected variables utilized when fitting the ensemble model based on soft voting which we selected as the best model in this prediction scenario depict that variables namely 'Q1 Balance', 'Q2 Balance', 'Q3 Balance', 'Q4 Balance', 'Average Balance' affects on the acceptance of the credit card offers at higher values while lower values causes non-acceptance of the offers. Moreover, the acceptance of the credit card offer is highly probable when the number of credit cards are at a high value which is quite justifiable as customers who maintain several credit cards due to their high financial potential.

8. Issues Encountered and Proposed Solutions

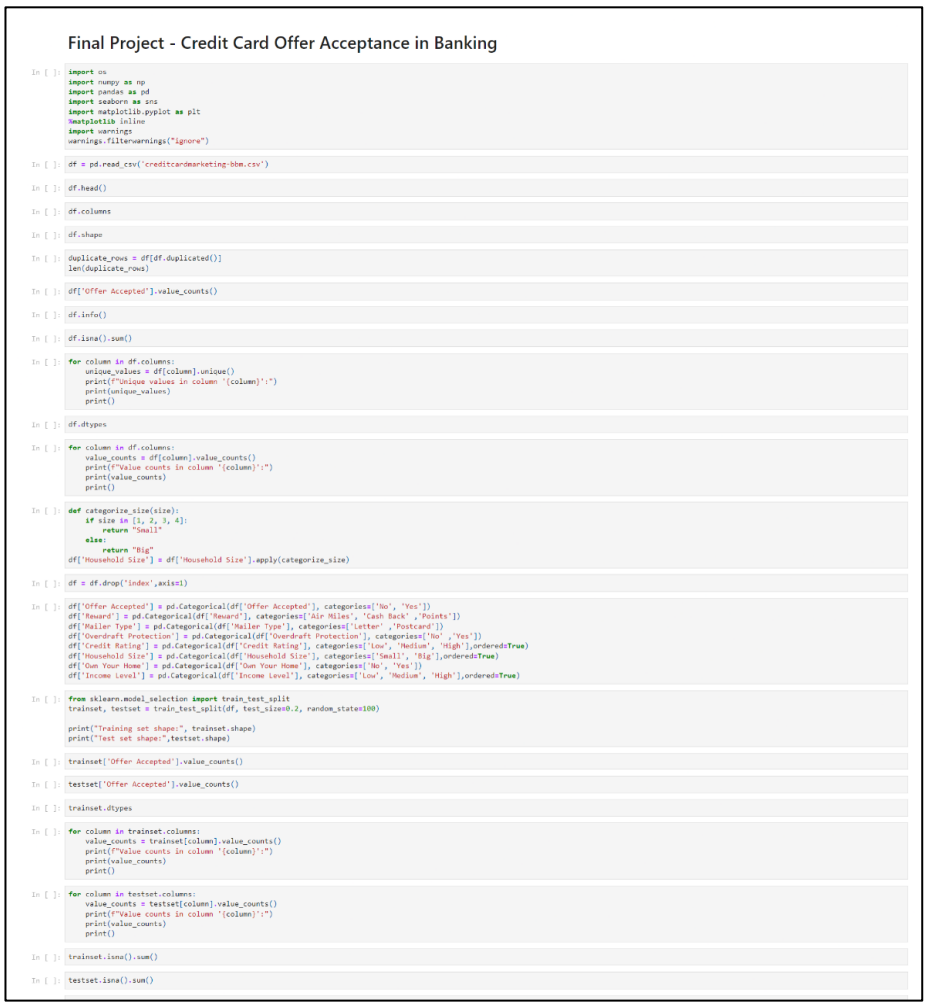
- The 'Household Size' categorical variable had 9 levels in it which is a considerably large number of categories to have and may further increase dimensionality in model fitting. To reduce the number of levels, the 9 levels were divided in to 2 categories capturing level 1,2,3 and 4 to one category and the rest of the levels to another category labeled as 'small' and 'big' respectively.
- The data distribution over the 2 categories of the response variable was highly imbalanced and this caused the majority class to highly influence the evaluation metrics of the fitted models where the precision and F1 score turned out to be significantly low when compared to accuracy due to the lower number of accepted offer (true positive) instances. To overcome this, the dataset was balance using Synthetic Minority Over Sampling Technique (SMOTE) and Random under sampling technique.
- In the initially fitted models to the original training dataset, none of the models predicted 'Offer_Accepted_Yes', in other words the true positive cases accurately. This disrupts the entire study as the main focus of this study is to predict offers that are getting accepted. Dataset balancing techniques were applied to balance the dataset to overcome this issue as well.
- The initial dataset had 17 predictor variables and after applying one-hot encoding on categorical variables, the dimensionality further increased making the models more complex, while inhibiting its performance. To reduce dimensionality, a feature importance plot was obtained and the most important 10 features were extracted as it gave the optimum performance and models were refitted.
- Some of the fitted models showed high performance in terms of all evaluation metrics, so selecting a single model which outperforms rest of the models was difficult, therefore to capture the performance of all the high performing models, an ensemble model was fitted using a voting classifier and that outperformed rest of the models.

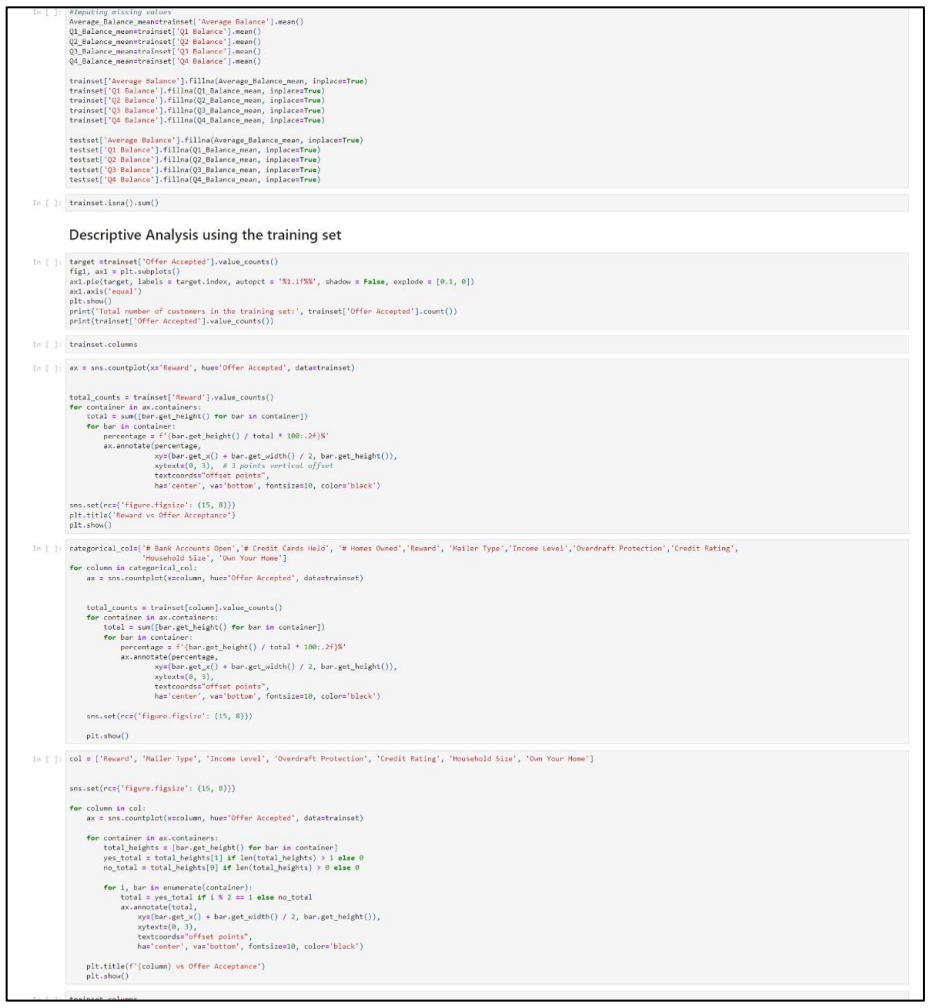
9. Discussion and Conclusion

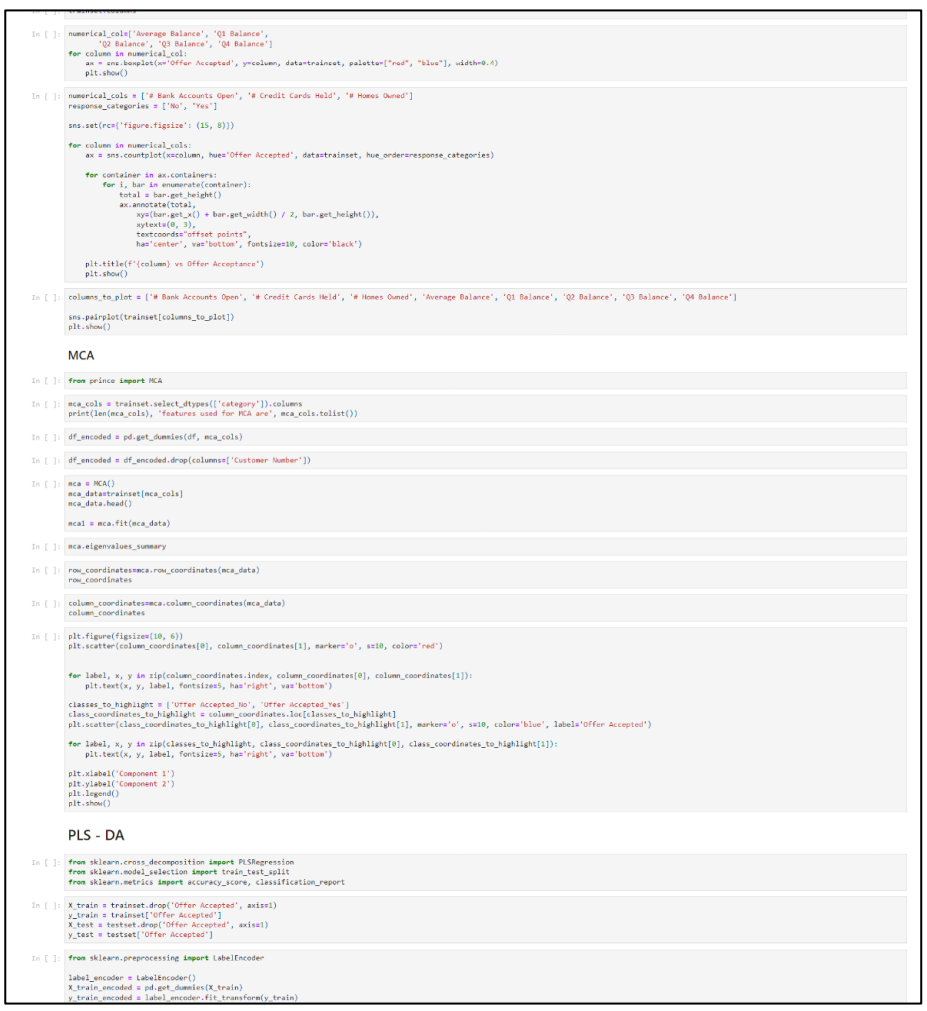
- At the surface analysis it was discovered that features such as Quarterly Balances, Average Balance, Credit Rating, Reward, Mailer Type, Credit Cards Held and Income Level are the most significant features which determine the acceptance of a credit card offer and that was further proved in variable importance plot of Random Forest model.
- Due to the imbalanced nature of the dataset, the models fitted on the dataset which was balanced using the oversampling technique SMOTE and Random Under Sampling Technique performed better.

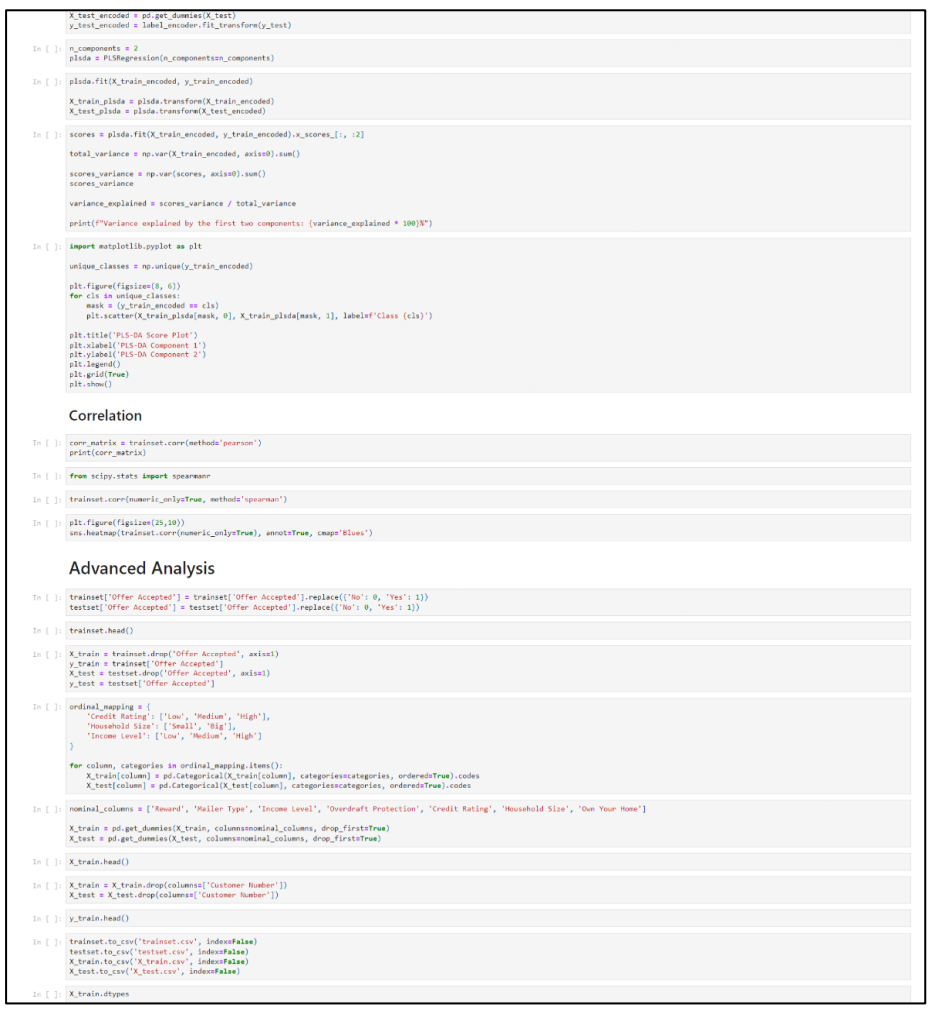
- Due to the possible information loss in Random Under Sampling Technique, the accuracies of models fitted on data balanced using SMOTE showed higher values and better performance.
- Out of all the initial models, Random Forest and XGBoost models performed the best and their performance was further improved while reducing overfitting by applying hyper parameter tuning.
- The best performing models in the initial stage was combined together in an ensemble model using the Voting Classifier and that showed the best accuracies so far.
- The performance of models was further improved by reducing the dimensionality where the dataset with 10 features with the highest feature importance were extracted and models were refitted.
- Out of all models fitted after dimensionality reduction, KNN, hyper parameter tuned Random Forest and XGBoost models showed the highest accuracies and they collectively formed an ensemble model which is the Voting Classifier and that outperformed all the predictive models fitted so far showing the best accuracies in term of precision, F1 score and recall.
- Therefore, Voting Classifier fitted on the reduced dataset with balanced data (applying SMOTE) was identified as the best model with the most accurate predictions and less information loss.

10. Appendix of the Code









Model fitting

```
from sklearn.preprocessing import StandardScaler

numerical_columns = ['# Bank Accounts Open', '# Credit Cards Held', '# Homes Owned',
                     'Average Balance', 'Q1 Balance', 'Q2 Balance', 'Q3 Balance', 'Q4 Balance']


scaler = StandardScaler()
X_train_scaled = X_train.copy()
X_train_scaled[numerical_columns] = scaler.fit_transform(X_train_scaled[numerical_columns])

X_test_scaled = X_test.copy()
X_test_scaled[numerical_columns] = scaler.transform(X_test_scaled[numerical_columns])
```

```
from sklearn.linear_model import LogisticRegression, LogisticRegressionCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier
import lightgbm as lgb
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score, classification_report
import joblib

random_state = 42
classifiers = [
    ('Logistic Regression', LogisticRegression(max_iter=1000)),
    ('Logistic Ridge', LogisticRegression(penalty='l2', max_iter=1000)),
    ('Logistic Lasso', LogisticRegression(penalty='l1',solver='liblinear')),
    ('KNN', KNeighborsClassifier()),
    ('Gaussian Naive Bayes', GaussianNB()),
    ('SVM Linear', SVC(kernel='linear', probability=True, random_state=42)),
    ('Random Forest', RandomForestClassifier(random_state=42)),
    ('XGBoost', XGBClassifier(random_state=42))
]


results = {}
classification_reports = {}
trained_models = {}

for name, clf in classifiers:

    clf.fit(X_train_scaled, y_train)

    trained_models[name] = clf

    y_pred = clf.predict(X_train_scaled)
    classification_report_text = classification_report(y_train, y_pred)
    accuracy = accuracy_score(y_train, y_pred)
    f1 = f1_score(y_train, y_pred)
    precision = precision_score(y_train, y_pred)
    recall = recall_score(y_train, y_pred)

    classification_reports[name] = classification_report_text
    results[name] = {'Accuracy': accuracy, 'F1-Score': f1, 'Precision': precision, 'Recall': recall}

import pandas as pd
results_df = pd.DataFrame(results)
print(results_df)

for name, report in classification_reports.items():
    print(f"Classification Report for {name}:\n{report}")

for name, model in trained_models.items():
    filename = f"{name}_model.joblib"
    joblib.dump(model, filename)
```

```
results = {}
classification_reports = {}

for name, clf in trained_models.items():
    y_pred = clf.predict(X_test_scaled)
    classification_report_text = classification_report(y_test, y_pred)
    accuracy = accuracy_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    classification_reports[name] = classification_report_text
    results[name] = {'Accuracy': accuracy, 'F1-Score': f1, 'Precision': precision, 'Recall': recall}

results_df = pd.DataFrame(results)
print(results_df)

for name, report in classification_reports.items():
    print(f"Classification Report for {name}:\n{report}")
```

```
from sklearn.metrics import confusion_matrix
classes = ['Not accepted', 'Accepted']

confusion_matrices = {}

for name, clf in trained_models.items():
    y_pred = clf.predict(X_test_scaled)
    cm = confusion_matrix(y_test, y_pred)
    confusion_matrices[name] = cm
```

```
for name, cm in confusion_matrices.items():
    plt.figure(figsize=(8, 6))
    plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
    plt.title(f"Confusion Matrix - {name}")
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)
    plt.xlabel('Predicted Label')
    plt.ylabel('True Label')

    thresh = cm.max() / 2.
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
            plt.text(j, i, format(cm[i, j], 'd'),
                     horizontalalignment="center",
                     color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.show()
```

Applying balancing techniques

```
import numpy as np
from sklearn.metrics import classification_report, precision_recall_curve, auc
from imblearn.over_sampling import SMOTE, ADASYN
from imblearn.under_sampling import RandomUnderSampler
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier
import lightgbm as lgb
import pandas as pd
import joblib

resampling_methods = {
    'SMOTE': SMOTE(random_state=42),
    'RandomUnderSampler': RandomUnderSampler(random_state=42)
}

classifiers = [
    ('Logistic Regression', LogisticRegression()),
    ('Logistic Ridge', LogisticRegression(penalty='l2', max_iter=1000)),
    ('Logistic Lasso', LogisticRegression(penalty='l1', solver='liblinear')),
    ('KNN', KNeighborsClassifier()),
    ('Gaussian Naive Bayes', GaussianNB()),
    ('Random Forest', RandomForestClassifier(random_state=42)),
    ('XGBoost', XGBClassifier(random_state=42)),
    ('SVM Linear', SVC(kernel='linear', probability=True, random_state=42))
]

results = {}
modified_trained_models = {}
classification_reports = {}

for resampling_name, resampling_method in resampling_methods.items():
    print(f"Applying {resampling_name}...")

    X_resampled, y_resampled = resampling_method.fit_resample(X_train_scaled, y_train)

    for name, clf in classifiers:
        clf.fit(X_resampled, y_resampled)
        modified_trained_models[(resampling_name, name)] = clf

        y_probs = clf.predict_proba(X_train_scaled)[:, 1]

        precision, recall, thresholds = precision_recall_curve(y_train, y_probs)

        pr_auc = auc(recall, precision)

        f1_scores = 2 * (precision * recall) / (precision + recall + 1e-10)
        optimal_threshold = thresholds[np.argmax(f1_scores)]
        print(f"Optimal Threshold for {name}: {optimal_threshold}")

        y_pred_optimal = (y_probs >= optimal_threshold).astype(int)

        report = classification_report(y_train, y_pred_optimal)
        classification_reports[(resampling_name, name)] = report

for (resampling_name, model_name), model in modified_trained_models.items():
    filename = f"{resampling_name.lower()}_{model_name}_model.joblib"
    joblib.dump(model, filename)

for (resampling_name, model_name), report in classification_reports.items():
    print(f"\nClassification Report for {resampling_name} - {model_name}:\n{report}")
```

```
import joblib
test_classification_reports = {}

for (resampling_name, model_name), clf in modified_trained_models.items():


    model_filename = f"{resampling_name.lower()}_{model_name}_model.joblib"
    trained_model = joblib.load(model_filename)

    y_probs_test = trained_model.predict_proba(X_test_scaled)[:, 1]
    y_pred_optimal_test = (y_probs_test >= optimal_threshold).astype(int)


    test_report = classification_report(y_test, y_pred_optimal_test)
    test_classification_reports[(resampling_name, model_name)] = test_report


for (resampling_name, model_name), test_report in test_classification_reports.items():
    print(f"\nClassification Report for {resampling_name} - {model_name} (on test set):\n{test_report}")
```

```
rf_model_smote = joblib.load('smote_Random Forest_model.joblib')
rf_model_us = joblib.load('randomundersampler_Random Forest_model.joblib')
feature_names = X_train.columns

feature_importances_smote = rf_model_smote.feature_importances_
importance_df_smote = pd.DataFrame({'Feature': feature_names, 'Importance': feature_importances_smote})
importance_df_smote = importance_df_smote.sort_values(by='Importance', ascending=False)

plt.figure(figsize=(10, 6))
plt.barh(importance_df_smote['Feature'], importance_df_smote['Importance'])
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.title('Variable Importance - SMOTE Random Forest')
plt.show()

feature_importances_us = rf_model_us.feature_importances_
importance_df_us = pd.DataFrame({'Feature': feature_names, 'Importance': feature_importances_us})
importance_df_us = importance_df_us.sort_values(by='Importance', ascending=False)

plt.figure(figsize=(10, 6))
plt.barh(importance_df_us['Feature'], importance_df_us['Importance'])
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.title('Variable Importance - Under Sampled Random Forest')
plt.show()
```

Hyper parameter tuning

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import StratifiedKFold
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from sklearn.metrics import classification_report

param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

smote = SMOTE(random_state=42)
X_resampled_smote, y_resampled_smote = smote.fit_resample(X_train_scaled, y_train)

rf_classifier_smote = RandomForestClassifier(random_state=42)
grid_search_smote = GridSearchCV(rf_classifier_smote, param_grid, cv=StratifiedKFold(n_splits=5), scoring='f1', n_jobs=-1)
grid_search_smote.fit(X_resampled_smote, y_resampled_smote)

best_model_smote = grid_search_smote.best_estimator_

print("Best parameters for SMOTE model:", grid_search_smote.best_params_)

y_pred_smote = best_model_smote.predict(X_test_scaled)

report_smote = classification_report(y_test, y_pred_smote)
print("Classification Report for the tuned SMOTE model:")
print(report_smote)


undersampler = RandomUnderSampler(random_state=42)
X_resampled_undersampler, y_resampled_undersampler = undersampler.fit_resample(X_train_scaled, y_train)

rf_classifier_undersampler = RandomForestClassifier(random_state=42)
grid_search_undersampler = GridSearchCV(rf_classifier_undersampler, param_grid, cv=StratifiedKFold(n_splits=5), scoring='f1', n_jobs=-1)
grid_search_undersampler.fit(X_resampled_undersampler, y_resampled_undersampler)

best_model_undersampler = grid_search_undersampler.best_estimator_


print("Best parameters for Random Undersampler model:", grid_search_undersampler.best_params_)

y_pred_undersampler = best_model_undersampler.predict(X_test_scaled)

report_undersampler = classification_report(y_test, y_pred_undersampler)
print("Classification Report for the tuned Random Undersampler model:")
print(report_undersampler)
```

```
y_pred_smote = best_model_smote.predict(X_train_scaled)

report_smote = classification_report(y_train, y_pred_smote)
print("Classification Report for the tuned SMOTE model:")
print(report_smote)

y_pred_undersampler = best_model_undersampler.predict(X_train_scaled)
report_undersampler = classification_report(y_train, y_pred_undersampler)
```

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import StratifiedKFold
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from sklearn.metrics import classification_report

param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

smote = SMOTE(random_state=42)
X_resampled_smote, y_resampled_smote = smote.fit_resample(X_train_scaled_selected, y_train)

rf_classifier_smote = RandomForestClassifier(random_state=42)
grid_search_smote = GridSearchCV(rf_classifier_smote, param_grid, cv=StratifiedKFold(n_splits=5), scoring='f1', n_jobs=-1)
grid_search_smote.fit(X_resampled_smote, y_resampled_smote)

best_reduced_model_smote = grid_search_smote.best_estimator_

print("Best parameters for SMOTE model:", grid_search_smote.best_params_)

y_pred_smote = best_reduced_model_smote.predict(X_test_scaled_selected)

report_smote = classification_report(y_test, y_pred_smote)
print("Classification Report for the tuned SMOTE model:")
print(report_smote)


param_grid_xgb = {
    'n_estimators': [50, 100, 200],
    'max_depth': [3, 5, 7],
    'learning_rate': [0.01, 0.1, 0.2],
    'subsample': [0.8, 1.0],
    'min_child_weight': [1, 3, 5],
}

# SMOTE
smote = SMOTE(random_state=42)
X_resampled_smote, y_resampled_smote = smote.fit_resample(X_train_scaled_selected, y_train)

xgb_classifier_smote = XGBClassifier(random_state=42)
grid_search_smote_xgb = GridSearchCV(xgb_classifier_smote, param_grid_xgb, cv=StratifiedKFold(n_splits=5), scoring='f1', n_jobs=-1)
grid_search_smote_xgb.fit(X_resampled_smote, y_resampled_smote)

best_reduced_model_smote_xgb = grid_search_smote_xgb.best_estimator_
```

```
plt.yticks(range(X_train_scaled.shape[1]), X_train.columns[sorted_idx_undersampler])
plt.xlabel('Feature Importance')
plt.ylabel('Feature')
plt.title('Variable Importance Plot for Random Undersampler-Random Forest model')
plt.show()


feature_importances_smote = best_model_smote_xgb.feature_importances_
sorted_idx_smote = feature_importances_smote.argsort()

colors_smote = generate_color_gradient(feature_importances_smote)

plt.figure(figsize=(10, 6))
plt.barh(range(X_train_scaled.shape[1]), feature_importances_smote[sorted_idx_smote], color=colors_smote[sorted_idx_smote])
plt.yticks(range(X_train_scaled.shape[1]), X_train.columns[sorted_idx_smote])
plt.xlabel('Feature Importance')
plt.ylabel('Feature')
plt.title('Variable Importance Plot for SMOTE-XGBoost model')
plt.show()


feature_importances_undersampler = best_model_undersampler_xgb.feature_importances_
sorted_idx_undersampler = feature_importances_undersampler.argsort()

colors_undersampler = generate_color_gradient(feature_importances_undersampler)

plt.figure(figsize=(10, 6))
plt.barh(range(X_train_scaled.shape[1]), feature_importances_undersampler[sorted_idx_undersampler], color=colors_undersampler[sorted_idx_undersampler])
plt.yticks(range(X_train_scaled.shape[1]), X_train.columns[sorted_idx_undersampler])
plt.xlabel('Feature Importance')
plt.ylabel('Feature')
plt.title('Variable Importance Plot for Random Undersampler-XGBoost model')
plt.show()
```

Voting Classifier

```
from sklearn.ensemble import VotingClassifier

xgb_smote_classifier = best_model_smote_xgb
rf_smote_classifier = best_model_smote

ensemble_classifier = VotingClassifier(
    estimators=[
        ('xgb_smote', xgb_smote_classifier),
        ('rf_smote', rf_smote_classifier)
    ],
    voting='soft'
)

smote = SMOTE(random_state=42)
X_resampled_smote, y_resampled_smote = smote.fit_resample(X_train_scaled, y_train)
ensemble_classifier.fit(X_resampled_smote, y_resampled_smote)


y_pred_train = ensemble_classifier.predict(X_train_scaled)


report_ensemble = classification_report(y_train, y_pred_train)
print("Classification Report for the Ensemble Classifier:")
print(report_ensemble)


y_pred_test = ensemble_classifier.predict(X_test_scaled)

report_ensemble = classification_report(y_test, y_pred_test)
print("Classification Report for the Ensemble Classifier:")
print(report_ensemble)
```

Reduced Models

```
X_train.columns
```

```
selected_features = [
    'Q1 Balance', 'Q2 Balance', 'Q3 Balance','Q4 Balance',
    'Average Balance', 'Credit Rating_2', 'Mailer Type_Postcard',
    'Reward_Cash Back','Credit Rating_1',  '# Credit Cards Held']

X_train_scaled_selected = X_train_scaled[selected_features]
X_test_scaled_selected = X_test_scaled[selected_features]
```

```
from sklearn.linear_model import LogisticRegression, LogisticRegressionCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier
import lightgbm as lgb
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score, classification_report
import joblib

random_state = 42
classifiers = [
    ('reduced_Logistic Regression', LogisticRegression(max_iter=1000)),
    ('reduced_Logistic Ridge', LogisticRegression(penalty='l2', max_iter=1000)),
    ('reduced_Logistic Lasso', LogisticRegression(penalty='l1',solver='liblinear')),
    ('reduced_KNN', KNeighborsClassifier()),
    ('reduced_Gaussian Naive Bayes', GaussianNB()),
    ('reduced_SVM Linear', SVC(kernel='linear', probability=True, random_state=42)),
    ('reduced_Random Forest', RandomForestClassifier(random_state=42)),
    ('reduced_XGBoost', XGBClassifier(random_state=42))
]


results = {}
classification_reports = {}
reduced_trained_models = {}

for name, clf in classifiers:

    clf.fit(X_train_scaled_selected, y_train)

    reduced_trained_models[name] = clf

    y_pred = clf.predict(X_train_scaled_selected)
    classification_report_text = classification_report(y_train, y_pred)
    accuracy = accuracy_score(y_train, y_pred)
    f1 = f1_score(y_train, y_pred)
    precision = precision_score(y_train, y_pred)
    recall = recall_score(y_train, y_pred)

    classification_reports[name] = classification_report_text
    results[name] = {'Accuracy': accuracy, 'F1-Score': f1, 'Precision': precision, 'Recall': recall}

import pandas as pd
results_df = pd.DataFrame(results)
print(results_df)

for name, report in classification_reports.items():
    print(f"Classification Report for {name}:\n{report}")

for name, model in reduced_trained_models.items():
    filename = f"{name}_model.joblib"
    joblib.dump(model, filename)
```

```
results = {}
classification_reports = {}

for name, clf in reduced_trained_models.items():
    y_pred = clf.predict(X_test_scaled_selected)
    classification_report_text = classification_report(y_test, y_pred)
    accuracy = accuracy_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    classification_reports[name] = classification_report_text
    results[name] = {'Accuracy': accuracy, 'F1-Score': f1, 'Precision': precision, 'Recall': recall}

results_df = pd.DataFrame(results)
print(results_df)

for name, report in classification_reports.items():
    print(f"Classification Report for {name}:\n{report}")
```

```
import numpy as np
from sklearn.metrics import classification_report, precision_recall_curve, auc
from imblearn.over_sampling import SMOTE, ADASYN
from imblearn.under_sampling import RandomUnderSampler
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier
import lightgbm as lgb
import pandas as pd
import joblib

resampling_methods = {
    'SMOTE': SMOTE(random_state=42),
    'RandomUnderSampler': RandomUnderSampler(random_state=42)
}

classifiers = [
    ('reduced_Logistic Regression', LogisticRegression(max_iter=1000)),
    ('reduced_Logistic Ridge', LogisticRegression(penalty='l2', max_iter=1000)),
    ('reduced_Logistic Lasso', LogisticRegression(penalty='l1',solver='liblinear')),
    ('reduced_KNN', KNeighborsClassifier()),
    ('reduced_Gaussian Naive Bayes', GaussianNB()),
    ('reduced_SVM Linear', SVC(kernel='linear', probability=True, random_state=42)),
    ('reduced_Random Forest', RandomForestClassifier(random_state=42)),
    ('reduced_XGBoost', XGBClassifier(random_state=42))
]

results = {}
modified_reduced_trained_models = {}
classification_reports = {}

for resampling_name, resampling_method in resampling_methods.items():
    print(f"Applying {resampling_name}...")

    X_resampled, y_resampled = resampling_method.fit_resample(X_train_scaled_selected, y_train)

    for name, clf in classifiers:
```

```
print("Classification Report for the tuned Random Undersampler model:")
print(report_undersampler)
```

```
from xgboost import XGBClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from sklearn.model_selection import StratifiedKFold

param_grid_xgb = {
    'n_estimators': [50, 100, 200],
    'max_depth': [3, 5, 7],
    'learning_rate': [0.01, 0.1, 0.2],
    'subsample': [0.8, 1.0],
    'min_child_weight': [1, 3, 5],
}

smote = SMOTE(random_state=42)
X_resampled_smote, y_resampled_smote = smote.fit_resample(X_train_scaled, y_train)

xgb_classifier_smote = XGBClassifier(random_state=42)
grid_search_smote_xgb = GridSearchCV(xgb_classifier_smote, param_grid_xgb, cv=StratifiedKFold(n_splits=5), scoring='f1', n_jobs=-1)
grid_search_smote_xgb.fit(X_resampled_smote, y_resampled_smote)

best_model_smote_xgb = grid_search_smote_xgb.best_estimator_

print("Best parameters for SMOTE XGBoost model:", grid_search_smote_xgb.best_params_)

y_pred_smote_xgb = best_model_smote_xgb.predict(X_test_scaled)

report_smote_xgb = classification_report(y_test, y_pred_smote_xgb)
print("Classification Report for the tuned SMOTE XGBoost model:")
print(report_smote_xgb)

undersampler = RandomUnderSampler(random_state=42)
X_resampled_undersampler, y_resampled_undersampler = undersampler.fit_resample(X_train_scaled, y_train)

xgb_classifier_undersampler = XGBClassifier(random_state=42)
grid_search_undersampler_xgb = GridSearchCV(xgb_classifier_undersampler, param_grid_xgb, cv=StratifiedKFold(n_splits=5), scoring='f1', n_jobs=-1)
grid_search_undersampler_xgb.fit(X_resampled_undersampler, y_resampled_undersampler)

best_model_undersampler_xgb = grid_search_undersampler_xgb.best_estimator_

print("Best parameters for Random Undersampler XGBoost model:", grid_search_undersampler_xgb.best_params_)

y_pred_undersampler_xgb = best_model_undersampler_xgb.predict(X_test_scaled)

report_undersampler_xgb = classification_report(y_test, y_pred_undersampler_xgb)
print("Classification Report for the tuned Random Undersampler XGBoost model:")
print(report_undersampler_xgb)
```

```
y_pred_smote = best_model_smote_xgb.predict(X_train_scaled)

report_smote = classification_report(y_train, y_pred_smote)
print("Classification Report for the tuned SMOTE model:")
print(report_smote)


y_pred_undersampler = best_model_undersampler_xgb.predict(X_train_scaled)

report_undersampler = classification_report(y_train, y_pred_undersampler)
print("Classification Report for the tuned Random Undersampler model:")
print(report_undersampler)
```

```
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.colors import LinearSegmentedColormap

def generate_color_gradient(importances):
    norm_importances = (importances - min(importances)) / (max(importances) - min(importances))

    # Define a custom colormap starting from light blue and degrading towards dark blue
    cmap = LinearSegmentedColormap.from_list('custom_gradient', ['#99CCFF', '#66B2FF', '#0089FF', '#0071D7', '#0058A9', '#00407A', '#00274C'])

    colors = cmap(norm_importances)
    return colors

feature_importances_smote = best_model_smote.feature_importances_
sorted_idx_smote = feature_importances_smote.argsort()

colors_smote = generate_color_gradient(feature_importances_smote)

plt.figure(figsize=(10, 6))
plt.barh(range(X_train_scaled.shape[1]), feature_importances_smote[sorted_idx_smote], color=colors_smote[sorted_idx_smote])
plt.yticks(range(X_train_scaled.shape[1]), X_train.columns[sorted_idx_smote])
plt.xlabel('Feature Importance')
plt.ylabel('Feature')
plt.title('Variable Importance Plot for SMOTE-Random Forest model')
plt.show()


feature_importances_undersampler = best_model_undersampler.feature_importances_
sorted_idx_undersampler = feature_importances_undersampler.argsort()

colors_undersampler = generate_color_gradient(feature_importances_undersampler)

plt.figure(figsize=(10, 6))
plt.barh(range(X_train_scaled.shape[1]), feature_importances_undersampler[sorted_idx_undersampler], color=colors_undersampler[sorted_idx_undersampler])
```

```
print("Best parameters for SMOTE XGBoost model:", grid_search_smote_xgb.best_params_)

y_pred_smote_xgb = best_reduced_model_smote_xgb.predict(X_test_scaled_selected)

report_smote_xgb = classification_report(y_test, y_pred_smote_xgb)
print("Classification Report for the tuned SMOTE XGBoost model:")
print(report_smote_xgb)
```

```
y_pred_smote = best_reduced_model_smote.predict(X_train_scaled_selected)

report_smote = classification_report(y_train, y_pred_smote)
print("Classification Report for the tuned SMOTE model:")
print(report_smote)


y_pred_smote_xgb = best_reduced_model_smote_xgb.predict(X_train_scaled_selected)

report_smote_xgb = classification_report(y_train, y_pred_smote_xgb)
print("Classification Report for the tuned SMOTE XGBoost model:")
print(report_smote_xgb)
```

Voting Classifier

```
from sklearn.ensemble import VotingClassifier
from sklearn.metrics import classification_report
import joblib

# Load the pre-trained models
best_reduced_model_knn = joblib.load("smote_reduced_KNN_model.joblib")
xgb_reduced_smote = best_reduced_model_smote_xgb
rf_reduced_smote= best_reduced_model_smote

voting_classifier = VotingClassifier(
    estimators=[
        ('xgb_smote', xgb_reduced_smote),
        ('rf_smote', rf_reduced_smote),
        ('KNN', best_reduced_model_knn)
    ],
    voting='soft'
)

voting_classifier.fit(X_resampled_smote, y_resampled_smote)

y_pred_ensemble = voting_classifier.predict(X_train_scaled_selected)

report_ensemble = classification_report(y_train, y_pred_ensemble)
print("Classification Report for the Ensemble Model:")
print(report_ensemble)
y_pred_ensemble = voting_classifier.predict(X_test_scaled_selected)

report_ensemble = classification_report(y_test, y_pred_ensemble)
print("Classification Report for the Ensemble Model:")
print(report_ensemble)
```

Partial Dependence plots

```
from sklearn.inspection import PartialDependenceDisplay

import matplotlib.pyplot as plt
features_to_plot = [    'Q1 Balance', 'Q2 Balance', 'Q3 Balance','Q4 Balance',
    'Average Balance', 'Credit Rating_2', 'Mailer Type_Postcard',
    'Reward_Cash Back','Credit Rating_1',  '# Credit Cards Held']

disp1 = PartialDependenceDisplay.from_estimator(best_reduced_model_smote, X_train_scaled_selected,
                                                ['Q1 Balance'])
plt.show()
disp1 = PartialDependenceDisplay.from_estimator(best_reduced_model_smote, X_train_scaled_selected,
                                                ['Q2 Balance'])
plt.show()
disp1 = PartialDependenceDisplay.from_estimator(best_reduced_model_smote, X_train_scaled_selected,
                                                ['Q3 Balance'])
plt.show()
disp1 = PartialDependenceDisplay.from_estimator(best_reduced_model_smote, X_train_scaled_selected,
                                                ['Q4 Balance'])
plt.show()
disp1 = PartialDependenceDisplay.from_estimator(best_reduced_model_smote, X_train_scaled_selected,
                                                ['Average Balance'])
plt.show()
disp1 = PartialDependenceDisplay.from_estimator(best_reduced_model_smote, X_train_scaled_selected,
                                                ['Credit Rating_2'])
plt.show()
disp1 = PartialDependenceDisplay.from_estimator(best_reduced_model_smote, X_train_scaled_selected,
                                                ['Mailer Type_Postcard'])
plt.show()
disp1 = PartialDependenceDisplay.from_estimator(best_reduced_model_smote, X_train_scaled_selected,
                                                ['Reward_Cash Back'])
plt.show()
disp1 = PartialDependenceDisplay.from_estimator(best_reduced_model_smote, X_train_scaled_selected,
                                                ['# Credit Cards Held'])
plt.show()
disp1 = PartialDependenceDisplay.from_estimator(best_reduced_model_smote, X_train_scaled_selected,
                                                ['Credit Rating_1'])
plt.show()
```