GROUP-2

Final Capstone Report

Project Title: Risk Analysis in Banking

Mentor: Jatinder Bedi

Team Members:
Nivedita Madhekar,
Aitha Vardhan,
Dharma Sasta,
Teja Naresh,
Samiuddin Mohammed

INTRODUCTION

Summary of the problem statement, data, and findings

Problem Statement: Analyzing the Risk Analytics in Banking dataset, to predict if a customer who has applied for a loan will default: if an individual has a late payment X days more than at least one of the first Y installments of the loan in our sample.
All other cases when the payment is paid on time. based on different attributes provided in the dataset.

Aim: To build the best model to classify whether the client is a defaulter or not.

Outcome: The loan-providing companies find it hard to give loans to the people due to their insufficient or non-existent credit history. This project helps the client to understand the driving factors(or driver variables) behind loan default, i.e., the variables that are strong indicators of default. The client can utilize this knowledge for its portfolio and risk assessment.

Data: We have two tables, Application Data and Previous Application Data. We have dropped columns with more than 40% null values and dropped the rest of the null values as there are approximately 350000 rows and 122 columns and approximately 1600000 rows and 37 columns, which is rather large for Personal computers, further on while applying complex models this can have a lot of impact in terms of running speed of the algorithms, and hence all the null values have been dropped.

Findings: After having a basic look at our data, we can see that the data is not normally distributed. We can also see a lot of redundant columns and there are around 25 columns with more than 40% of NULL values. There are no duplicates in either of the tables. We also see that SK_ID_CURR is the common column in both tables and can be used to join these two tables. We also observe that there is a high class imbalance in the TARGET variable with an imbalance of 91%-9%.

Implications:

- Determine which borrowers are most likely to default: This enables banks to take precautions against defaults, such as requesting more collateral from borrowers or raising interest rates.
- Identify the causes of default: Banks can learn more about the causes of default by examining the data. Using this data, programs to assist borrowers in avoiding default can be created.
- Create lending guidelines: The information may be used by banks to create lending guidelines that are more likely to reduce their risk of default. For instance, they could decide to lend to clients who have a solid credit history or high incomes.

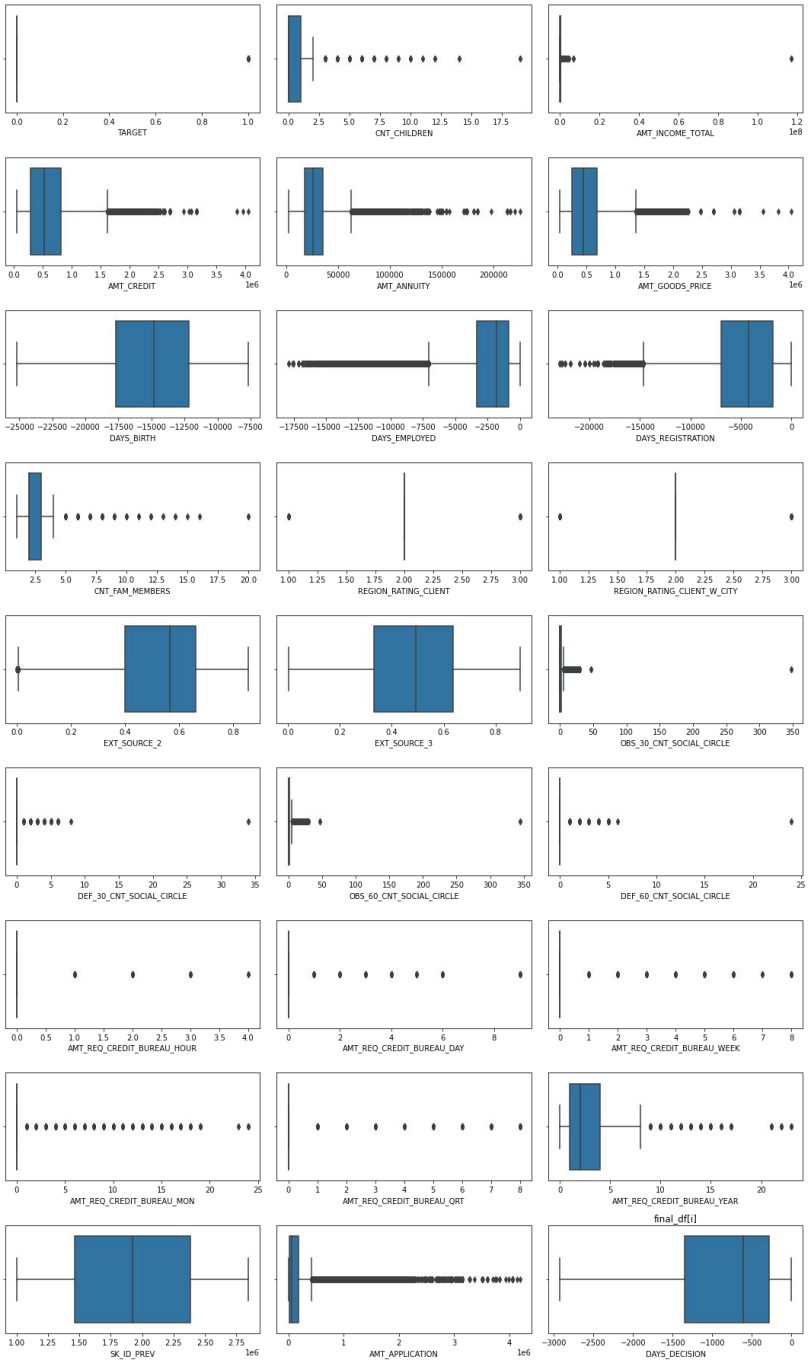
Overview of the final process

We started with the application data. Here we had 307511 rows and 122 columns, and after dropping all the columns with more than 43% null values, we were left with 73 columns. On further dropping of null values, we were left with 167732 rows. Coming to the previous application data, we started with 1670214 rows × 37 columns.

After dropping columns with more than 40% null values and then dropping the rest of the null values, we were left with 26 columns. After this, we merged the two tables on SK_ID_CURR, and the final data looked like 788066 rows and 49 columns since we dropped redundant columns like SK_ID_CURR, Gender, and dropped FLAG_OWN_CAR and kept the age of the car and those who didn't have a car, the car age was made to be 0. We kept the FLAG_OWN_REALITY column and dropped all the normalized Data of their apartment or house. We also dropped other columns such as information about the client if they gave the right phone number, or if their permanent address matches the contact address, and such other columns. We have further dropped the FLAG_DOCUMENT columns because they just have information about whether the client has submitted a document or not.

After dropping redundant columns, we had our final data frame and we checked for the distribution of the numerical columns and for outliers using box plots. For the categorical columns, we did a bivariate analysis with respect to the target variable in the form of a bar plot. Here we noticed a high class imbalance in the target variable of 91%-9% of non-defaulters to defaulters. For the numerical columns, we performed a distribution analysis and detected outliers with the help of box plot. We did not want to introduce bias in our model or create fake data, and wanted to maintain the authentic data that we had, we have not treated the outliers and left them as it is. We have taken that action because we did not want our model to miss any of the actual patterns and learn fake patterns. And also because there are a lot of outliers, we did not want to lose the authenticity.

After the plots, we separated the columns into numerical (except the TARGET) and categorical and performed label encoding on the categorical columns as we did not wish to increase the number of categorical columns. After encoding, we performed power transformation on all the numerical columns and created a final_df with the encoded categorical columns and transformed numerical columns, and then finally, we added the TARGET variable to this, and we had our final dataset.



Box plot of numeric variables

After the train test split, we built a Logit Model as our base model as it is a simple linear model and it will help us decide which machine learning techniques will help us enhance our final model. We have built Gaussian NB as the next model and we have observed a better performance. The next model we have built is Random Forest, followed by Decision Tree and we have worked on improving this model to get the best results. We have also compared all the best models with and without an over-sampling technique called SMOTE.

STATISTICAL TESTS TO VALIDATE THE OBSERVATION FOR TARGET VARIABLE [DEFAULT/ NON-DEFAULT]

We have made a few observations from graphs and performed a normality assumption test by using the Jarque-Bera test, which found that the data is not normally distributed. Considered to perform non-parametric tests.

```
Jarque-Bera test for TARGET p-value 0.0
Jarque-Bera test for CNT_CHILDREN p-value 0.0
Jarque-Bera test for AMT_INCOME_TOTAL p-value 0.0
Jarque-Bera test for AMT_CREDIT p-value 0.0
Jarque-Bera test for AMT_ANNUITY p-value 0.0
Jarque-Bera test for AMT_GOODS_PRICE p-value 0.0
Jarque-Bera test for DAYS_BIRTH p-value 0.0
Jarque-Bera test for DAYS_EMPLOYED p-value 0.0
Jarque-Bera test for DAYS_REGISTRATION p-value 0.0
Jarque-Bera test for CNT_FAM_MEMBERS p-value 0.0
Jarque-Bera test for REGION_RATING_CLIENT p-value 0.0
Jarque-Bera test for REGION_RATING_CLIENT_W_CITY p-value 0.0
Jarque-Bera test for EXT_SOURCE_2 p-value 0.0
Jarque-Bera test for EXT_SOURCE_3 p-value 0.0
Jarque-Bera test for OBS_30_CNT_SOCIAL_CIRCLE p-value 0.0
Jarque-Bera test for DEF_30_CNT_SOCIAL_CIRCLE p-value 0.0
Jarque-Bera test for OBS_60_CNT_SOCIAL_CIRCLE p-value 0.0
Jarque-Bera test for DEF_60_CNT_SOCIAL_CIRCLE p-value 0.0
Jarque-Bera test for AMT_REQ_CREDIT_BUREAU_HOUR p-value 0.0
Jarque-Bera test for AMT_REQ_CREDIT_BUREAU_DAY p-value 0.0
Jarque-Bera test for AMT_REQ_CREDIT_BUREAU_WEEK p-value 0.0
Jarque-Bera test for AMT_REQ_CREDIT_BUREAU_MON p-value 0.0
Jarque-Bera test for AMT_REQ_CREDIT_BUREAU_QRT p-value 0.0
Jarque-Bera test for AMT_REQ_CREDIT_BUREAU_YEAR p-value 0.0
Jarque-Bera test for SK_ID_PREV p-value 0.0
Jarque-Bera test for AMT_APPLICATION p-value 0.0
Jarque-Bera test for DAYS_DECISION p-value 0.0
```

Chi-square test of Independence:

This test is used to test whether the categorical feature is independent or not with respect to the Target Categorical Variable(Non default/default).

H0: The variables are independent.

H1: The variables are not independent (i.e. variables are dependent).

```
Chi-Square test for NAME_CONTRACT_TYPE_x p-value: 2.0442196153514 89e-249
Chi-Square test for FLAG_OWN_REALTY p-value: 0.002344510083239603
Chi-Square test for NAME_TYPE_SUITE p-value: 7.161168673830531e-15
Chi-Square test for NAME_INCOME_TYPE p-value: 0.0
Chi-Square test for NAME_EDUCATION_TYPE p-value: 0.0
Chi-Square test for NAME_FAMILY_STATUS p-value: 1.8726251656866347e-208
Chi-Square test for NAME_HOUSING_TYPE p-value: 5.5070643564552594e-198
Chi-Square test for OCCUPATION_TYPE p-value: 0.0
Chi-Square test for ORGANIZATION_TYPE p-value: 0.0
Chi-Square test for NAME_CONTRACT_TYPE_y p-value: 0.0
Chi-Square test for FLAG_LAST_APPL_PER_CONTRACT p-value: 3.8146702189260165e-06
Chi-Square test for NAME_CASH_LOAN_PURPOSE p-value: 1.768102911322972e-259
Chi-Square test for NAME_CONTRACT_STATUS p-value: 0.0
Chi-Square test for NAME_PAYMENT_TYPE p-value: 1.9781110155181423e-161
Chi-Square test for CODE_REJECT_REASON p-value: 0.0
Chi-Square test for NAME_CLIENT_TYPE p-value: 6.55623969986925e-44
Chi-Square test for NAME_GOODS_CATEGORY p-value: 0.0
Chi-Square test for NAME_PORTFOLIO p-value: 2.857234220347244e-309
Chi-Square test for NAME_PRODUCT_TYPE p-value: 0.0
Chi-Square test for CHANNEL_TYPE p-value: 0.0
Chi-Square test for NAME_SELLER_INDUSTRY p-value: 0.0
Chi-Square test for NAME_YIELD_GROUP p-value: 0.0
```

We cab observe that for category variables, the p-value is greater than 0.05. Thus, we fail to reject (i.e. accept) the null hypothesis and conclude that these variables are independent with respect to the target variable

For the remaining variables, the p-value is less than 0.05, so we reject the null hypothesis and conclude that these variables are dependent with respect to the target variable.

Mann-Whitney U test:

This test is used to test whether the numeric features are independent or not with respect to the Target Categorical variable (non-default/default). Instead of comparing means, this test focuses on ranking observations within each group.

H0: There is no relationship /independent

H1: There is a relationship /dependent

```
Mann-Whitney U test for NAME_CONTRACT_TYPE_x p-value: 1.6018433821550464e-249
Mann-Whitney U test for FLAG_OWN_REALTY p-value: 0.002311367169247295
Mann-Whitney U test for NAME_TYPE_SUITE p-value: 3.681920288915631e-15
Mann-Whitney U test for NAME_INCOME_TYPE p-value: 7.486748881642618e-246
Mann-Whitney U test for NAME_EDUCATION_TYPE p-value: 0.0
Mann-Whitney U test for NAME_FAMILY_STATUS p-value: 1.8715111554500819e-06
Mann-Whitney U test for NAME_HOUSING_TYPE p-value: 5.2732180434978487e-126
Mann-Whitney U test for OCCUPATION_TYPE p-value: 4.990556183660432e-64
Mann-Whitney U test for ORGANIZATION_TYPE p-value: 0.4510695555014763
Mann-Whitney U test for NAME_CONTRACT_TYPE_y p-value: 3.528296424109862e-29
Mann-Whitney U test for FLAG_LAST_APPL_PER_CONTRACT p-value: 3.3673524265143135e-06
Mann-Whitney U test for NAME_CASH_LOAN_PURPOSE p-value: 0.06823532417229376
Mann-Whitney U test for NAME_CONTRACT_STATUS p-value: 0.0
Mann-Whitney U test for NAME_PAYMENT_TYPE p-value: 2.2022757144807232e-162
Mann-Whitney U test for CODE_REJECT_REASON p-value: 0.0
Mann-Whitney U test for NAME_CLIENT_TYPE p-value: 2.1412537562116624e-05
Mann-Whitney U test for NAME_GOODS_CATEGORY p-value: 2.0279186738254067e-275
Mann-Whitney U test for NAME_PORTFOLIO p-value: 0.06006419152845837
Mann-Whitney U test for NAME_PRODUCT_TYPE p-value: 2.3363961842779077e-28
Mann-Whitney U test for CHANNEL_TYPE p-value: 2.5425960599645366e-49
Mann-Whitney U test for NAME_SELLER_INDUSTRY p-value: 1.5348112348031346e-146
Mann-Whitney U test for NAME_YIELD_GROUP p-value: 1.202790938607495e-219
Mann-Whitney U test for CNT_CHILDREN p-value: 3.7960160081004016e-18
```

```
Mann-Whitney U test for AMT_INCOME_TOTAL p-value: 1.0
Mann-Whitney U test for AMT_CREDIT p-value: 7.997338765251749e-106
Mann-Whitney U test for AMT_ANNUITY p-value: 3.0087374796366382e-06
Mann-Whitney U test for AMT_GOODS_PRICE p-value: 1.789770724293738e-269
Mann-Whitney U test for DAYS_BIRTH p-value: 0.0
Mann-Whitney U test for DAYS_EMPLOYED p-value: 0.0
Mann-Whitney U test for DAYS_REGISTRATION p-value: 1.6507546598411615e-224
Mann-Whitney U test for CNT_FAM_MEMBERS p-value: 0.0005621300655982066
Mann-Whitney U test for REGION_RATING_CLIENT p-value: 0.0
Mann-Whitney U test for REGION_RATING_CLIENT_W_CITY p-value: 0.0
Mann-Whitney U test for EXT_SOURCE_2 p-value: 0.0
Mann-Whitney U test for EXT_SOURCE_3 p-value: 0.0
Mann-Whitney U test for OBS_30_CNT_SOCIAL_CIRCLE p-value: 6.189717818996287e-44
Mann-Whitney U test for DEF_30_CNT_SOCIAL_CIRCLE p-value: 6.580533337422404e-171
Mann-Whitney U test for OBS_60_CNT_SOCIAL_CIRCLE p-value: 2.7618292306621e-43
Mann-Whitney U test for DEF_60_CNT_SOCIAL_CIRCLE p-value: 3.166260150805973e-118
Mann-Whitney U test for AMT_REQ_CREDIT_BUREAU_HOUR p-value: 0.004589049012633305
Mann-Whitney U test for AMT_REQ_CREDIT_BUREAU_DAY p-value: 4.363123260201685e-15
Mann-Whitney U test for AMT_REQ_CREDIT_BUREAU_WEEK p-value: 0.0005319130561491676
Mann-Whitney U test for AMT_REQ_CREDIT_BUREAU_MON p-value: 3.967774465313669e-21
Mann-Whitney U test for AMT_REQ_CREDIT_BUREAU_QRT p-value: 8.306105735708053e-09
Mann-Whitney U test for AMT_REQ_CREDIT_BUREAU_YEAR p-value: 4.7165064983804084e-104
Mann-Whitney U test for SK_ID_PREV p-value: 0.04611997071455976
Mann-Whitney U test for AMT_APPLICATION p-value: 7.476362456572387e-37
Mann-Whitney U test for DAYS_DECISION p-value: 0.0
Mann-Whitney U test for TARGET p-value: 0.0
```

We can observe that for numeric variables of the p-value is greater than 0.05. Thus we fail to reject (i.e. accept) the null hypothesis and conclude that the variable is related with respect to the target variable

For the remaining variables, the p-value is less than 0.05, so we reject the null hypothesis and conclude that these variables are dependent with respect to the target variable.

After conducting the stats tests for numerical and categorical columns with respect to the target variable, we found these to be the important features-

```
'const', 'NAME_CONTRACT_TYPE_x', 'FLAG_OWN_REALTY', 'NAME_TYPE_SUITE',
'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS',
'NAME_HOUSING_TYPE', 'OCCUPATION_TYPE', 'ORGANIZATION_TYPE',
'NAME_CONTRACT_TYPE_y', 'NAME_CASH_LOAN_PURPOSE', 'CODE_REJECT_REASON',
'NAME_CLIENT_TYPE', 'NAME_GOODS_CATEGORY', 'NAME_PORTFOLIO',
'CHANNEL_TYPE', 'NAME_SELLER_INDUSTRY', 'NAME_YIELD_GROUP',
'CNT_CHILDREN', 'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE',
'DAYS_BIRTH', 'DAYS_EMPLOYED', 'DAYS_REGISTRATION', 'CNT_FAM_MEMBERS',
'REGION_RATING_CLIENT', 'REGION_RATING_CLIENT_W_CITY', 'EXT_SOURCE_2',
'EXT_SOURCE_3', 'DEF_30_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE',
'AMT_REQ_CREDIT_BUREAU_DAY', 'AMT_REQ_CREDIT_BUREAU_MON',
'AMT_REQ_CREDIT_BUREAU_QRT', 'DAYS_DECISION'],
```

Step-by-step walkthrough of the solution

Once we had our final_df, we performed Train-Test-Split on the data and split it to 70-30 and used the stratify option so as to maintain the relative class distribution in both training and testing sets, which in turn can lead to more accurate evaluations of the model's performances on unseen data.
We used a logistic regression model as our base model because by starting with logistic regression, we established a baseline level of performance as, if logistic regression achieved satisfactory results the problem might not have required the complexity of advanced algorithms. But since it didn't and the model f1-score was around 1.2% and the other parameters also performed badly, we realized it was a bad model due to high class imbalance and it was unable to predict the non-defaulters with high accuracy. Another reason it might not have performed well is due to the presence of outliers and non-linear data.

	Model	Accuracy	Recall	Precision	F1 Score	Cohen-Kappa
0	Logistic Regression	0.907571	0.006035	0.545455	0.011937	0.009932

The next model we ran was Gaussian Naïve Bayes, as it is a simple and lightweight algorithm, computationally efficient, and has low memory requirements, which make it suitable for quick model-building and initial model testing. Due to its assumptions that features are conditionally independent given the class label, it was unable to perform satisfactorily with respect to classifying the data. However, it performed well when compared to the Logistic Regression Model.

	Model	Accuracy	Recall	Precision	F1 Score	Cohen-Kappa
0	Logistic Regression	0.907571	0.006035	0.545455	0.011937	0.009932
1	GaussianNB	0.844874	0.235348	0.205124	0.219199	0.133531

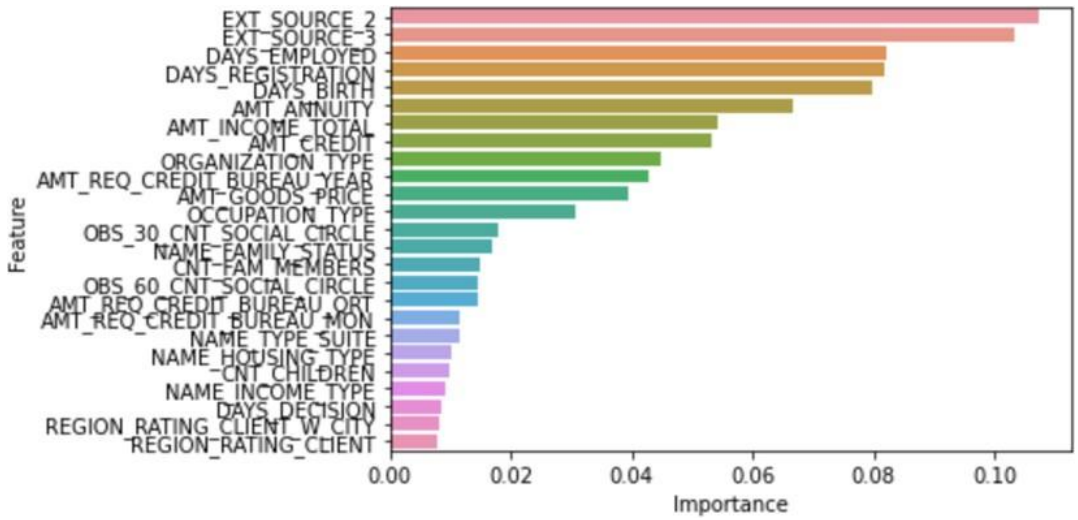
The next model we built was the Random Forest Classifier Model. We used this ensemble machine learning algorithm as it is capable of dealing with both linear and nonlinear interactions between features and the target variable. It can capture intricate relationships that simpler linear models may overlook. This model performed well, with an f1 score of 77.3% and a reliability of 75.5%. But this model is in an overfitting condition, and hence the testing data is not performing well.

	Model	Accuracy	Recall	Precision	F1 Score	Cohen-Kappa
0	Logistic Regression	0.907571	0.006035	0.545455	0.011937	0.009932
1	GaussianNB	0.844874	0.235348	0.205124	0.219199	0.133531
2	RandomForest	0.965777	0.630246	0.999782	0.773126	0.755643

The next model we decided to run was the Decision Tree Classifier without hypertuning the model. We can see that the F1 Score is 90.5% and the accuracy is 98%. This model was also in overfitting condition, and the testing data did not perform well.

	Model	Accuracy	Recall	Precision	F1 Score	Cohen-Kappa
0	Logistic Regression	0.907571	0.006035	0.545455	0.011937	0.009932
1	GaussianNB	0.844874	0.235348	0.205124	0.219199	0.133531
2	RandomForest	0.965777	0.630246	0.999782	0.773126	0.755643
3	Decision Tree	0.982451	0.910259	0.901073	0.905642	0.895968

We ran a feature importance of the Decision Tree and plotted it in a bar plot. It shows how features contribute to predictions within the specific decision tree model.



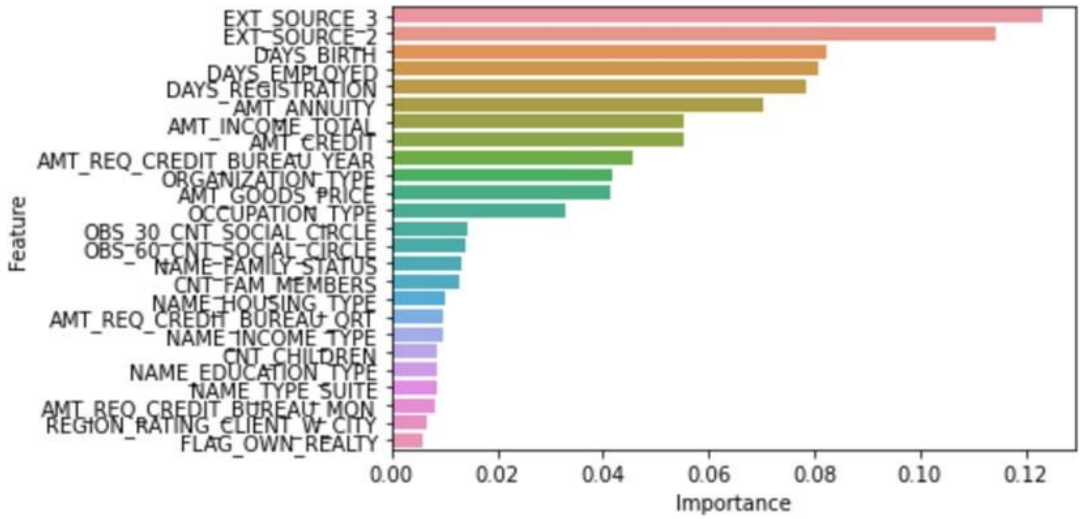
Next, we ran a GridSearch CV, which is a technique that automates the process of searching through a predefined set of hyperparameter values to find the combination that yields the best performance for a given model. These were the results after running the GridSearchCV Model for the decision tree classifier from the previous model.

```
Fitting 5 folds for each of 108 candidates, totalling 540 fits
Best Parameters: {'class_weight': None, 'criterion': 'entropy', 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_spli
t': 2}
Best Model: DecisionTreeClassifier(criterion='entropy')
```

Next, we ran a Decision Tree model with the hypertuning parameters, and we clearly saw an increase in the F1-Score and the Cohen-Kappa Score from the Decision tree model to the hypertuned model. We saw an increase in all the model evaluation parameters.

	Model	Accuracy	Recall	Precision	F1 Score	Cohen-Kappa
0	Logistic Regression	0.907571	0.006035	0.545455	0.011937	0.009932
1	GaussianNB	0.844874	0.235348	0.205124	0.219199	0.133531
2	RandomForest	0.965777	0.630246	0.999782	0.773126	0.755643
3	Decision Tree	0.982451	0.910259	0.901073	0.905642	0.895968
4	DT GridSearchCV	0.985361	0.923014	0.919106	0.921056	0.912988

After running the feature importance and plotting it, we see that almost all features are the same, and the plot is shown below.



After finding the best features with the help of the Logit Model and Feature Importance from 2 Decision Tree Models, one with hyper tuning and one without, we selected 36 columns that are -

'NAME_CONTRACT_TYPE_x', 'FLAG_OWN_REALTY', 'NAME_TYPE_SUITE', 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE', 'OCCUPATION_TYPE', 'ORGANIZATION_TYPE', 'NAME_CONTRACT_TYPE_y', 'NAME_CASH_LOAN_PURPOSE', 'CODE_REJECT_REASON', 'NAME_CLIENT_TYPE', 'NAME_GOODS_CATEGORY', 'NAME_PORTFOLIO', 'CHANNEL_TYPE', 'NAME_SELLER_INDUSTRY', 'NAME_YIELD_GROUP', 'CNT_CHILDREN', 'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE', 'DAYS_BIRTH', 'DAYS_EMPLOYED', 'DAYS_REGISTRATION', 'CNT_FAM_MEMBERS', 'REGION_RATING_CLIENT', 'REGION_RATING_CLIENT_W_CITY', 'EXT_SOURCE_2', 'EXT_SOURCE_3', 'DEF_30_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE', 'AMT_REQ_CREDIT_BUREAU_DAY', 'AMT_REQ_CREDIT_BUREAU_MON', 'AMT_REQ_CREDIT_BUREAU_QRT', 'DAYS_DECISION'.

After finding out the important features, we performed a train-test-split on it again with just these 36 features and built the next models with our new training and testing data.
We built a Decision Tree with important features and used the best parameters from the GridSearchCV Model. And here we see that our model evaluators remain the same.

	Model	Accuracy	Recall	Precision	F1 Score	Cohen-Kappa
0	Logistic Regression	0.907571	0.006035	0.545455	0.011937	0.009932
1	GaussianNB	0.844874	0.235348	0.205124	0.219199	0.133531
2	RandomForest	0.965777	0.630246	0.999782	0.773126	0.755643
3	Decision Tree	0.982451	0.910259	0.901073	0.905642	0.895968
4	DT GridSearchCV	0.985361	0.923014	0.919106	0.921056	0.912988
5	DT-ImportantFeature	0.985314	0.921551	0.919869	0.920709	0.912617

We built an AdaBoost with decision trees, as it is a powerful ensemble technique that can be applied to classification problems. It's capable of achieving high accuracy and is particularly effective when the base models complement each other's strengths and weaknesses. Here, we see a slight increase in all the model evaluators.

	Model	Accuracy	Recall	Precision	F1 Score	Cohen-Kappa
0	Logistic Regression	0.907571	0.006035	0.545455	0.011937	0.009932
1	GaussianNB	0.844874	0.235348	0.205124	0.219199	0.133531
2	RandomForest	0.965777	0.630246	0.999782	0.773126	0.755643
3	Decision Tree	0.982451	0.910259	0.901073	0.905642	0.895968
4	DT GridSearchCV	0.985361	0.923014	0.919106	0.921056	0.912988
5	DT-ImportantFeature	0.985314	0.921551	0.919869	0.920709	0.912617
6	AdaBoost-DT ImpF	0.985395	0.922191	0.920130	0.921159	0.913111

Another approach to handling the class imbalance was to use SMOTE Analysis. SMOTE is an over-sampling technique used to generate synthetic samples for the minority class in imbalanced datasets to balance class distribution and improve model performance. We increased the minority class from 9% to 27%, changing the class imbalance from 91%-9% to 73%-27%. And we have built models with this new synthetic data.

We have built a Random Forest Ensemble Model, a Decision Tree Model, AdaBoost with Random Forest, and a Decision Tree, all with Important Features that were selected. We have chosen these specific models to build after SMOTE Analysis as they have performed well before.

	Model	Accuracy	Recall	Precision	F1 Score	Cohen-Kappa
0	Logistic Regression	0.907571	0.006035	0.545455	0.011937	0.009932
1	GaussianNB	0.844874	0.235348	0.205124	0.219199	0.133531
2	RandomForest	0.965777	0.630246	0.999782	0.773126	0.755643
3	Decision Tree	0.982451	0.910259	0.901073	0.905642	0.895968
4	DT GridSearchCV	0.985361	0.923014	0.919106	0.921056	0.912988
5	DT-ImportantFeature	0.985314	0.921551	0.919869	0.920709	0.912617
6	AdaBoost-DT ImpF	0.985395	0.922191	0.920130	0.921159	0.913111
7	RF-SMOTE	0.969213	0.894385	0.992893	0.941068	0.920309
8	DT-SMOTE	0.939911	0.871893	0.905943	0.888592	0.847474
9	AdaBoost-DT-SMOTE	0.940077	0.872656	0.905867	0.888952	0.847939
10	AdaBoost-RF-SMOTE	0.969402	0.895247	0.992710	0.941463	0.920826

Here we see that the Random Forest Model has performed the best with Adaptive Boosting as well, while the AdaBoost RF with SMOTE has the best F1-Score and Cohen-Kappa Score. So

we shall select this model to be our final model as it gives the most confidence to our problem, i.e., defaulters.

Model evaluation

Describe the final model (or ensemble) in detail. What was the objective, what parameters were prominent, and how did you evaluate the success of your model(s)? A convincing explanation of the robustness of your solution will go a long way to supporting your answer.

We have created a separate data frame to store all of our different model metrics so that it is easier for comparison, and from there, we have picked the AdaBoost Random Forest after SMOTE Analysis as our final model. These are the parameters we have considered to evaluate the success of our final model's performance:

Accuracy: Accuracy provides an overall measure of a model's correctness, but it can be misleading in imbalanced datasets where one class dominates.
Recall: Recall measures the model's ability to identify all relevant positive instances, helping assess how well it avoids false negatives.
Precision: Precision indicates the model's accuracy when it predicts positives, helping assess how well it avoids false positives.
F1 Score: The F1 Score is the harmonic mean of precision and recall, and is particularly useful when dealing with imbalanced datasets. And since we have a high class imbalance, this is the most important metric for us to decide our model's performance.
Cohen's Kappa: The Cohen Kappa Score is a statistical measure of inter-rater reliability for categorical variables. It's especially useful when the class distribution is imbalanced.

Logistic Regression and GaussianNB have lower accuracy and generally weaker performance across all metrics compared to other models, whereas Decision Tree, RandomForest, and various AdaBoost-DT models show strong performance across metrics. While DT GridSearchCV, DT-ImportantFeature, and AdaBoost-DT ImpF demonstrate consistently high performance, models with SMOTE oversampling (RF-SMOTE, DT-SMOTE, and AdaBoost-DT-SMOTE) maintain good performance, though sometimes slightly lower than their non-SMOTE counterparts. We observe that Decision Tree-based models and variants, as well as RandomForest and AdaBoost-DT, are among the top-performing models based on the metrics. The use of SMOTE seems to enhance the models' ability to handle imbalanced data but may introduce some trade-offs in specific cases.

	Model	Accuracy	Recall	Precision	F1 Score	Cohen-Kappa
0	Logistic Regression	0.907571	0.006035	0.545455	0.011937	0.009932
1	GaussianNB	0.844874	0.235348	0.205124	0.219199	0.133531
2	RandomForest	0.965777	0.630246	0.999782	0.773126	0.755643
3	Decision Tree	0.982451	0.910259	0.901073	0.905642	0.895968
4	DT GridSearchCV	0.985361	0.923014	0.919106	0.921056	0.912988
5	DT-ImportantFeature	0.985314	0.921551	0.919869	0.920709	0.912617
6	AdaBoost-DT ImpF	0.985395	0.922191	0.920130	0.921159	0.913111
7	RF-SMOTE	0.969213	0.894385	0.992893	0.941068	0.920309
8	DT-SMOTE	0.939911	0.871893	0.905943	0.888592	0.847474
9	AdaBoost-DT-SMOTE	0.940077	0.872656	0.905867	0.888952	0.847939
10	AdaBoost-RF-SMOTE	0.969402	0.895247	0.992710	0.941463	0.920826

We have chosen the final model as an ensemble learning model because that combines the strengths of Adaptive Boosting and Decision Trees while incorporating SMOTE oversampling. This ensemble approach helps us to leverage the individual benefits of these techniques to achieve a higher level of precision, recall, and reliability in classifying whether a client will default or not.
Adaboost is an ensemble learning technique in which it aggregates multiple weak learners of the Decision Tree into a single strong learner. We also performed SMOTE Analysis to rectify the class imbalance to a certain extent.

The metrics of our final model, AdaBoost Random Forest with SMOTE Analysis, are-

Model	AdaBoost-RF-SMOTE
Accuracy	0.969402
Recall	0.895247
Precision	0.99271
F1 Score	0.941463
Cohen-Kappa	0.920826

- 97% of the dataset is being accurately classified by our model. In datasets with high imbalances, accuracy alone cannot be adequate. Yet, our model's ability to maintain a high accuracy level indicates that it can efficiently handle both classes.
- The recall of 89.52% indicates that our model accurately identifies over 90% of the individuals whom the bank should accept for loans. The high recall indicates that our model is successful in identifying people who are actually suitable.
- The precision value is 99.27, where 99 of the time it is accurate in predicting defaulters out of 100. This is crucial because incorrectly classifying people as defaulters may have negative implications.

- The F1 score of 94.15% is a balanced measure that combines precision and recall, and our model has an excellent balance between finding positive scenarios and limiting incorrect predictions. This is a strong sign that your model is performing well across different classification tasks.
- With a Cohen's Kappa score of 0.9208, your model's predictions and actual results have a high degree of agreement. This substantial agreement is a reliable and consistent indicator of the success of your model.

The combination of high accuracy, recall, precision, F1 score, and Cohen's Kappa indicates that our model is not just performing well but also providing meaningful and accurate predictions. Above is a comprehensive review of our final model's performance, and by examining these metrics, we have provided a clear picture of our model's strengths and its ability to handle the given classification problem effectively and reliably.

Comparison to benchmark

The benchmark we originally developed using the Logistic Regression model is surpassed by our final solution, which makes use of the AdaBoost-RF-SMOTE model. Several important performance indicators show this improvement. Let's look more closely at the comparison and the causes of the improvement that was seen.

Comparison with the Benchmark: Logistic Regression vs. AdaBoost-RF-SMOTE
Our benchmark, the Logistic Regression model, provided the following performance metrics:

Model	Logistic Regression
Accuracy	0.907571
Recall	0.006035
Precision	0.545455
F1 Score	0.011937
Cohen-Kappa	0.009932

In contrast, the AdaBoost-RF-SMOTE model yielded the following results:

Model	AdaBoost-RF-SMOTE
Accuracy	0.969402
Recall	0.895247
Precision	0.99271
F1 Score	0.941463
Cohen-Kappa	0.920826

Improvement in the Benchmark

The AdaBoost-RF-SMOTE model showcases significant improvements across all metrics compared to the Logistic Regression benchmark. This improvement can be attributed to several factors:

- **Ensemble Learning with AdaBoost:**
  AdaBoost enhances the model's performance by combining multiple weaker models (base learners) into a robust ensemble. This collaborative approach allows for better generalization and increased predictive accuracy.

- **Base Model: Random Forest:**
  Utilizing Random Forest as the base classifier within the AdaBoost framework further enhances the model. Random Forest is known for its ability to capture complex relationships in the data, leading to improved predictive capabilities.

- **Addressing Class Imbalance with SMOTE:**
  The integration of the Synthetic Minority Over-sampling Technique (SMOTE) mitigates the class imbalance issue, enhancing the model's ability to predict the minority class. SMOTE generates synthetic instances to balance the class distribution, thereby preventing the model from being biased toward the majority class.

We can say that the AdaBoost Random Forest Model is a better model than the Logistic Regression Model due to these factors:

- **Higher Recall and Precision:**
  The AdaBoost-RF-SMOTE model demonstrates a remarkable increase in both recall and precision. This signifies the model's capacity to effectively identify positive cases (recall) while maintaining a low rate of false positives (precision).

- **Balanced F1 Score and Cohen's Kappa:**

  The F1 Score, which harmonizes precision and recall, showcases a balanced performance for the AdaBoost-RF-SMOTE model. Similarly, Cohen's Kappa coefficient indicates substantial agreement between predicted and actual classes beyond random chance.

- **Reduction of False Negatives and False Positives:**

Base Model:

Logistic regression is a simple linear model that calculates the likelihood that a given input belongs to a specific class. It can't naturally deal with imbalanced classes.

```
1 confusion_matrix(ytest, ypred_LR)
```

```
array([[214436,    110],
       [ 21742,    132]], dtype=int64)
```

```
1 confusion_matrix(ytrain, ytrain_LR)
```

```
array([[500398,    210],
       [ 50758,    280]], dtype=int64)
```

Final Model:

It is an ensemble technique for learning called Adaboost (Adaptive Boosting) that combines the predictions of several weak learners to generate a stronger classifier. It focuses on cases where the former classifier's mistakes may indirectly aid in lowering the number of false positives and false negatives. It focuses on the predictions that were misclassified by the weak classifiers, which can indirectly assist in lowering the number of false positives and false negatives.

```
1 confusion_matrix(YTest, ypred_abclsm)
```

```
array([[213784,    534],
       [  8509,  72720]], dtype=int64)
```

```
1 confusion_matrix(YTrain, ytrain_abclsm)
```

```
array([[500836,      0],
       [     0, 188771]], dtype=int64)
```

- **ROC-AUC Score and ROC Curve:**

Base Model-

An ROC AUC score of 0.50 is considered to be at the level of chance. The model is not able to distinguish between positive and negative classes. This indicates that the model's ability to predict the positive class is on par with guessing at random.

We can draw the following conclusions based on the ROC AUC Score:

- The model is not well-trained
- The model is not using the features in an effective way.

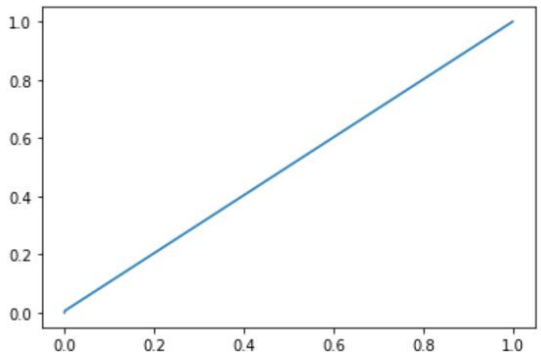
It is crucial to remember that a model is not absolutely worthless if it has a ROC AUC score of 0.50. The model will still have a chance to predict correctly, but it will also make a lot of incorrect predictions.

```
1 roc_auc_score(ytest, ypred_LR)
```

```
0.5027609255095263
```

```
1 fpr, tpr, thresh = roc_curve(ytest, ypred_LR)
2 plt.plot(fpr,tpr)
```

```
[<matplotlib.lines.Line2D at 0x1c1b928ace0>]
```



ROC Curve for Logistic Regression Model

Final Model-

An ROC AUC of 0.94 is a good score. It means that the model is able to distinguish between positive and negative classes very well. There are a few possible inferences that can be made out of this:

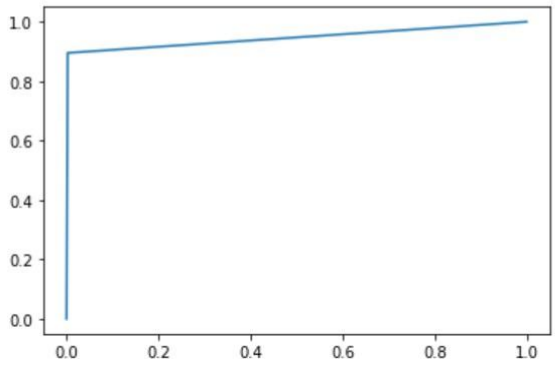
- The data is well-suited for the model. The model is able to learn the patterns in the data that are necessary to distinguish between positive and negative classes.
- The model is complex enough. The model has enough parameters to learn the patterns in the data.
- The model is not overfitting the data. The model is not learning the patterns in the training data too well and is able to generalize to new data.

```
1 roc_auc_score(YTest, ypred_abclsm)
```

```
0.9463775734390073
```

```
1 fpr, tpr, thresh = roc_curve(YTest, ypred_abclsm)
2 plt.plot(fpr,tpr)
```

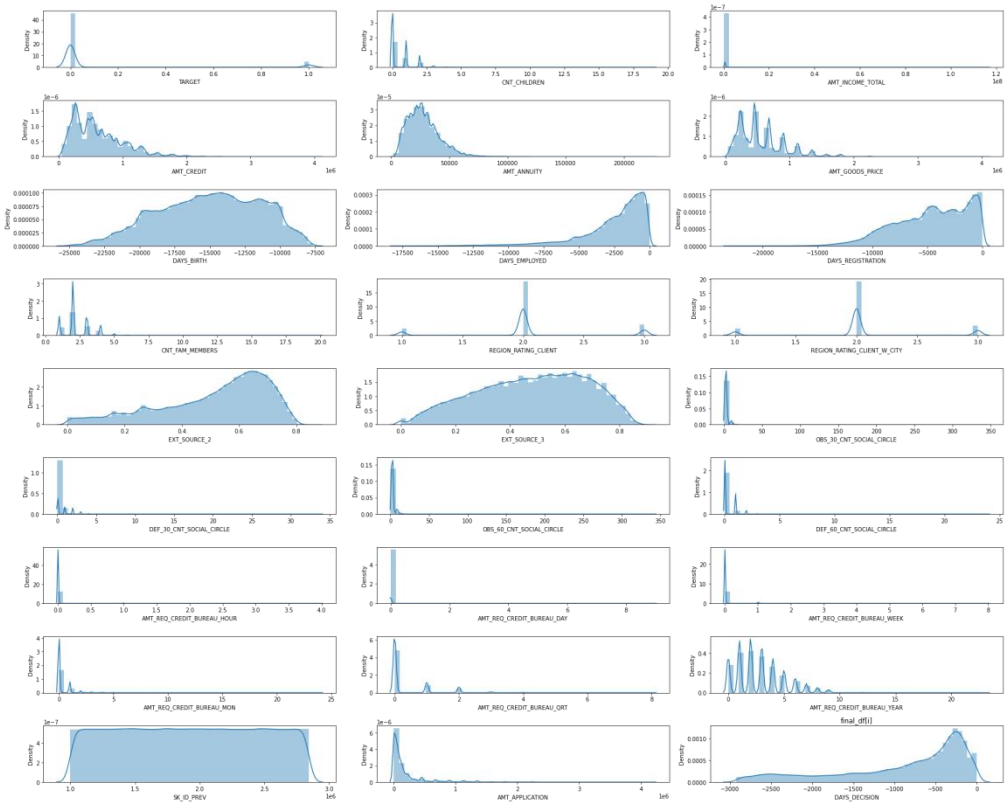
```
[<matplotlib.lines.Line2D at 0x1c1b91f8fa0>]
```



ROC-AUC Curve for AdaBoost Random Forest with SMOTE

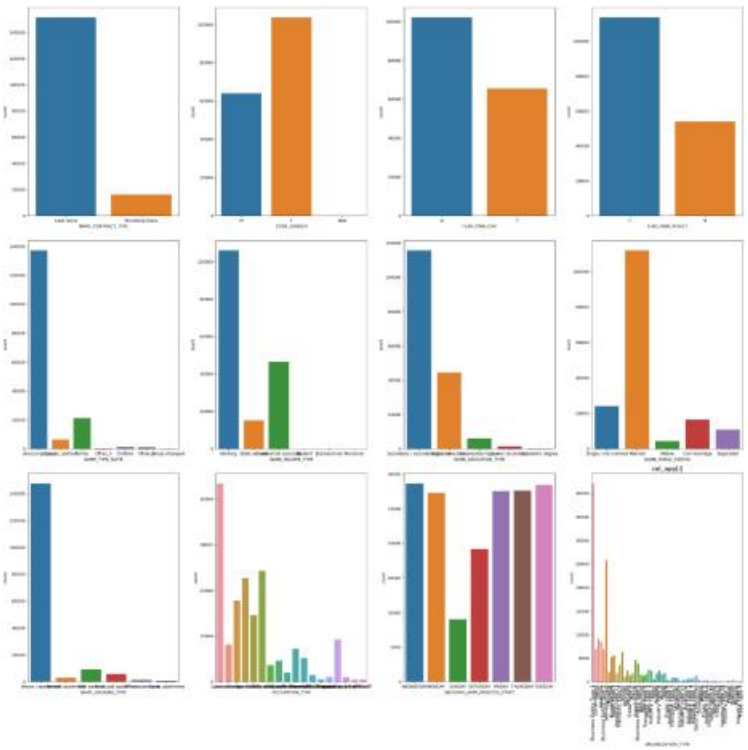
In conclusion, our final solution, the AdaBoost-RF-SMOTE model, outperforms the benchmark Logistic Regression model in terms of accuracy, recall, precision, F1 Score, and Cohen's Kappa. This improvement is attributed to the ensemble nature of AdaBoost, the power of Random Forest as the base model, and the class-balancing effect of SMOTE. The results underscore the significance of employing advanced techniques to enhance model performance and accuracy in classification tasks, showcasing a considerable advancement over the baseline benchmark.

Visualizations



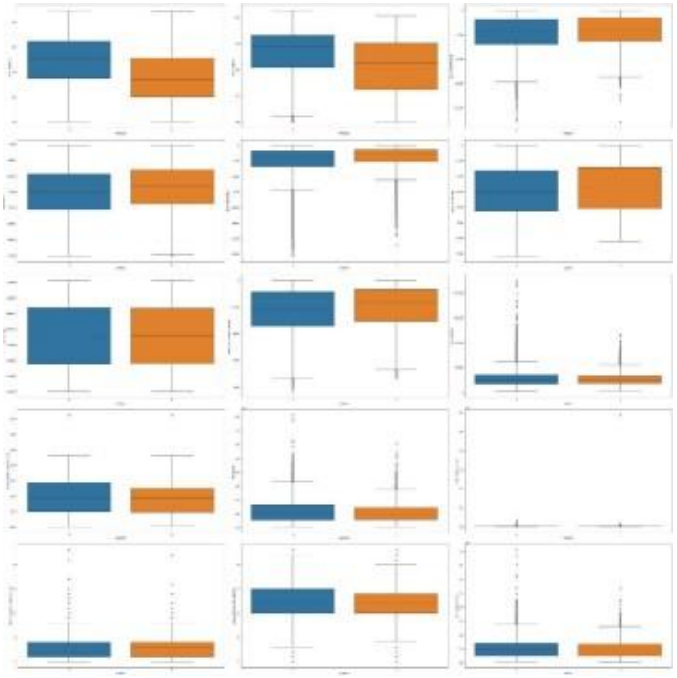
`Distplot of numerical variables`

The distribution plot gave us a rough understanding of the data. It is visible that most variables are not normally distributed, there are some extreme values present in the data and some are discrete numerical variables.



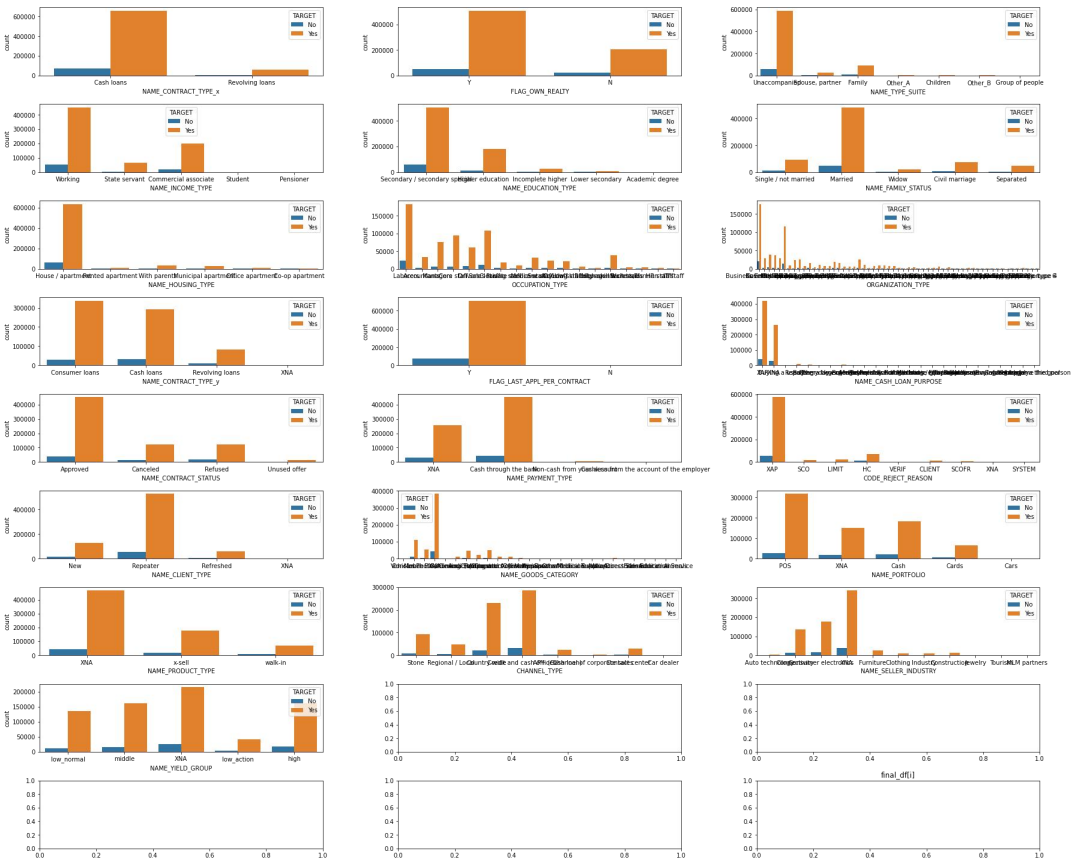
Categorical Variables-Bar Plot

From the contract types, it's visible that most of the loans are cash loans and the minority are revolving loans. We can also observe that there are a high number of loans from females and married people The other high number of people who took loans have a house/apartment.



Numerical variables vs. Target

It can be observed that the data for both defaulter and non-defaulter contain the same range and means for most of the numerical columns. These variables have been obtained after running the feature importance from the Decision tree model.



Barplot of categorical variables, with hue being the TARGET

Here we observe that most classes are non-defaulters and very few are defaulters. That is we can observe a high class imbalance and this can be fixed by over-sampling techniques, i.e., SMOTE Analysis.

Implications

The confusion matrix for the final model for the testing data-

```
[213784,    534]
[  8509,  72720]
```

The model correctly predicted 213,784 borrowers who did not default and 534 borrowers who did default. The model also incorrectly predicted 8,509 borrowers who did not default and correctly predicted 72,720 borrowers who did default.

TN: 213874	FP: 534
FN: 8509	TP: 72720

The majority class (borrowers who did not default) continues to outnumber the minority class (borrowers who did default), although the gap has narrowed after SMOTE Analysis. In other words, even if a borrower is truly in default danger, the model is less likely to forecast that the borrower won't default.
The FN is still being penalized with greater severity than the FP because of the fact that a FN indicates that the bank may have lost money by lending money to a borrower who is likely to default. An FP indicates that the bank has turned down a loan from a potential customer who is actually a good risk, which might cost the bank future revenue.

Our solution has a profound effect in the banking sector as it has an accuracy of 96% and can correctly classify a given case if the client will be a defaulter or not. The approach can help banks decide who to lend money to more effectively. This can assist them in lowering the risk associated with making loans to borrowers who are liable for defaulting. The bank should choose a model with a higher recall if giving money to borrowers who are more prone to default is of more concern to it. However, the bank should employ a model with a lower recall if it is more concerned with turning down loans from creditworthy clients.

Here are some ways that our approach can have a significant impact on the banking industry:

- **Improved risk management:** By using the model to identify borrowers who are at default risk, banks may make improvements to their risk management procedures.
- **Cost savings:** The model can assist banks in cutting back on expenses related to money lending, such as the cost of loan origination and the cost of loan defaults.
- **Increased earnings:** By providing loans to borrowers who are more likely to repay them, the model can assist banks in maximizing their profits.

- **Improved customer service:** By identifying borrowers who are having trouble repaying their loans and offering them assistance, the model can help banks offer better customer service.

Once we get approval on our model, we can look into deploying it in the real world. Here are a few typical strategies for implementing a machine learning model:
On-premises: The model may be installed on-site, which implies that the company's servers are used to host it. If the company has the funds to maintain its own servers and security infrastructure, this can be a suitable alternative.
Cloud-based: The model is hosted on the servers of a cloud provider when it is installed in the cloud. If the company lacks the capacity to manage its own servers or if it wants to avoid the up-front expenses of on-premises deployment, this may be a smart choice.

Now let us look into how we can deploy our model in the cloud-based premise

- Choose the location where the model will be used. On-premises solutions and cloud platforms like AWS, Azure, or Google Cloud are typical choices.
- Consider the anticipated use and make sure the deployment infrastructure can grow to handle the model's predictions without sacrificing speed.
- To enable other systems to provide data to the model for predictions as well as obtain the results, an API (Application Programming Interface) must be developed. Both the response format and the input data format should be specified using this API.
- Once deployed, we must ensure that our data maintains its integrity and security. We must implement security measures to safeguard sensitive client information and stop unauthorized access to the model. This might involve data anonymization, encryption, and authentication.

Limitations

No matter how precise or advanced a model is, it always has its limitations. The following list of limitations that applies to the remedies offered by our final model, according to us:

Imbalanced Data:
Despite the application of the Synthetic Minority Over-sampling Technique (SMOTE), the confusion matrix still reveals a class imbalance in the final model's predictions. There are much more borrowers who did not default than borrowers who did. The model may find it challenging to develop an effective defaulter prediction skill due to this class imbalance. This might affect the model's ability to predict the minority class accurately.
False Negatives:
As a result of the model's high false negative rate, it is more likely to forecast that a borrower won't default when they actually will. Due to this, the bank may wind up lending money to borrowers who are in danger of defaulting, which might cause the bank to suffer financial losses.

Unseen Data:
The model's performance is evaluated based on the current dataset. If the underlying data distribution changes in the future, the model might require retraining and reevaluation to maintain accuracy and precision.
Bias:
Models may unintentionally replicate biases found in training data, producing unfair or biased results. Fairness and integrity must be guaranteed, which is a difficult task.

The following actions may be taken to improve the solution:
Using regularization: This method penalizes the model for being overly complicated. This could assist in avoiding overfitting.
Reducing False Negatives: If we can reduce the false negatives, the bank will not loan out to a client who can default but our model predicts that the client will not be a defaulter.
Explainable AI: AI that can be explained methods may be applied to make the model easier to understand. As a result, the model may be used more successfully and with greater confidence.
Combine the model with other elements: The model shouldn't be relied upon alone to decide whether or not to extend credit to a borrower. It should be considered alongside other elements, such as the borrower's income and credit history, before making a loan decision.
Model monitoring: The model has to be kept under constant observation to make sure it is operating as planned.

Closing Reflections

-

Through the course of this project, several key insights and lessons have been gained:
Importance of Data Preprocessing:
The significance of thorough data preprocessing cannot be overstated. Addressing missing values, handling outliers, and ensuring data quality lay the foundation for accurate model training and interpretation.

Model Selection and Evaluation:
The process of selecting the appropriate model and evaluating its performance is crucial. The need to consider both accuracy and interpretability when choosing models became evident, as complex models might require additional attention to explainability.

Class Imbalance Challenge:
Class imbalance can greatly impact model performance. The realization that advanced techniques like SMOTE can help mitigate this challenge was an important takeaway.

Model Explainability:
The importance of not just building accurate models but also being able to explain them effectively was highlighted. We have also learned how to explain the evaluation of our models and the different metrics we must use.

Future Improvements

Reflecting on the project, a few aspects could be approached differently in future endeavors:

Feature Engineering Emphasis:
Devote more time to feature engineering, exploring additional features that could potentially enhance model discrimination and prediction accuracy. Use Feature Extraction techniques to gain more insight into the data.

Interpretable Models for Insights:
Place greater emphasis on employing interpretable models earlier in the process to gain insights into feature importance and relationships. Also look into conducting better univariate and multivariate analysis and make more insightful inferences from the visualization of data.

Robustness to Data Variability:
Develop strategies to test model robustness against varying data distributions over time, ensuring the models remain reliable in dynamic environments.

Continuous Model Monitoring:
Establish a systematic process for monitoring model performance post-deployment, enabling timely updates and maintenance to align with changing requirements.

By embracing these insights and adjustments, future projects can be approached with an enhanced understanding of potential challenges and opportunities, leading to more robust and impactful outcomes.

References:

- https://www.reply.com/avantage-reply/en/publications/white-paper/Shared%20Documents/Avantage_Reply_DataRobot_AI_Models_in_Banking.pdf
- https://www.kaggle.com/datasets/sabarostami/risk-analytics-in-banking?select=columns_descript
- https://www.analyticsvidhya.com/blog/
- https://towardsdatascience.com/machine-learning-predicting-bank-loan-defaults-d48bffb9aee2