IME 672A: DATA MINING AND KNOWLEDGE DISCOVERY



PROJECT REPORT

ON

CREDIT CARD APPROVAL PREDICTION

**Submitted to: Dr. Faiz Hamid**

**Submitted by:**

**GROUP 12**

Vipul Kohli 160795

Abhiyanshu Gautam 170038

Amit Badoni 170091

Anshika Verma 170131

Harsh Verma 170290

**Problem Description**

Title: Credit Card Approval Prediction

Context

Credit score cards are a common risk control method in the financial industry. It uses personal information and data submitted by credit card applicants to predict the probability of future defaults and credit card borrowings. The bank is able to decide whether to issue a credit card to the applicant. Credit scores can objectively quantify the magnitude of risk.

Task: Binary Classification of applicants as ‘good’ or ‘bad’

Build a binary classification machine learning model to predict if an applicant is 'good' or 'bad' client, different from other tasks, the definition of 'good' or 'bad' is not given. The model should do ‘good’ and ‘bad’ label construction as well. Based on this predicted label, the applications will be approved or rejected. Also, unbalance data problem is a big problem in this task.

**Data Understanding**

There are 2 datasets:

1. **Application Record** – Includes generic information about a customer recorded while filing for a credit card application. It has **18 attributes** (**6 binary, 5 numeric and 7 categorical**)
2. **Credit Record** – Describes how a customer is performing after his/her application was approved. It has **3 attributes**. ‘Months\_Balance’ represents a month whose performance is described. The month of the extracted data is the starting point, backwards, 0 is the current month, -1 is the previous month, and so on. ‘Status’ has 9 values, which are as follows: ‘0’: 1-29 days past due, ‘1’: 30-59 days past due, ‘2’: 60-89 days overdue, ‘3’: 90-119 days overdue, ‘4’: 120-149 days overdue, ‘5’: Overdue or bad debts, write-offs for more than 150 days, ‘C’: paid off that month and ‘X’: No loan for the month.

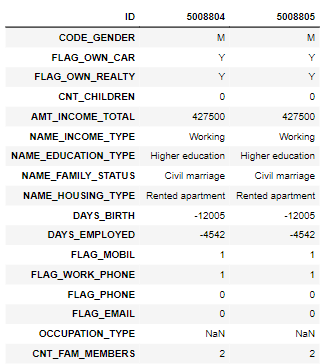
Application Record: Credit Record:

|  |  |
| --- | --- |
| Feature | Explanation |
| ID | Client Number |
| MONTHS\_BALANCE | Record Number |
| STATUS | Loan/Credit Status |

|  |  |  |
| --- | --- | --- |
| Feature Name | Explanation | Attribute type |
| ID | Client Identification Number | Nominal |
| CODE\_GENDER | Gender - Male/Female | Binary |
| FLAG\_OWN\_CAR | Car owner or not | Binary |
| FLAG\_OWN\_REALTY | Property owner or not | Binary |
| CNT\_CHILDREN | Number of children | Numerical |
| AMT\_INCOME\_TOTAL | Annual Income | Numerical |
| NAME\_INCOME\_TYPE | Income Category | Nominal |
| NAME\_EDUCATION\_TYPE | Education Level | Nominal |
| NAME\_FAMILY\_STATUS | Marital Status | Ordinal |
| NAME\_HOUSING\_TYPE | Way of living | Ordinal |
| DAYS\_BIRTH | Birthday | Numerical |
| DAYS\_EMPLOYED | Employment start date | Numerical |
| FLAG\_MOBIL | Owns a mobile phone | Binary |
| FLAG\_WORK\_PHONE | Has work phone | Binary |
| FLAG\_PHONE | Has a phone | Binary |
| FLAG\_EMAIL | Has an email | Binary |
| OCCUPATION\_TYPE | Occupation | Nominal |
| CNT\_FAMILY\_MEMBERS | Family Size | Numerical |

**Data Pre-processing**

In application record, although there are different customer IDs, there is equal information over some distinct ID values as shown in the table below. But we can't simply drop them because for two distinct IDs with the same information in Application Record; they can have different information in credit record. So, we have to take the intersection of both data frames and only take valid customer IDs present in both data frames. After taking the intersection of both data frames, our data size in Application Record reduced from 438,557 to 36,457.



To remove duplicates, we define a new attribute “cust\_id”, the sum of all numerical attribute values of a particular ID. The same value of “cust\_id” points to duplicate data.

We will then assign a trait 'b' or 'g' to every data point in the credit record data frame. This assigning is done using EDA-Vintage-Analysis Method**1**. If a customer has a payment for more than 60 days, we assign the 'b' trait to the 'g' trait. Sor for status value {2, 3, 4, 5} we assign ‘b’ trait and for {0, 1, C, X} we assign ‘g’ trait to that id.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | MONTHS\_BALANCE | STATUS | trait |
| 5001712 | -8 | C | g |
| 5149838 | -21 | 4 | b |

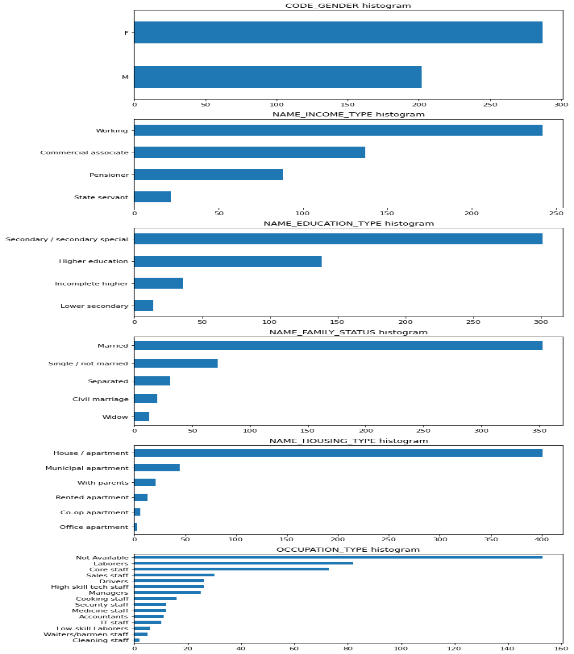
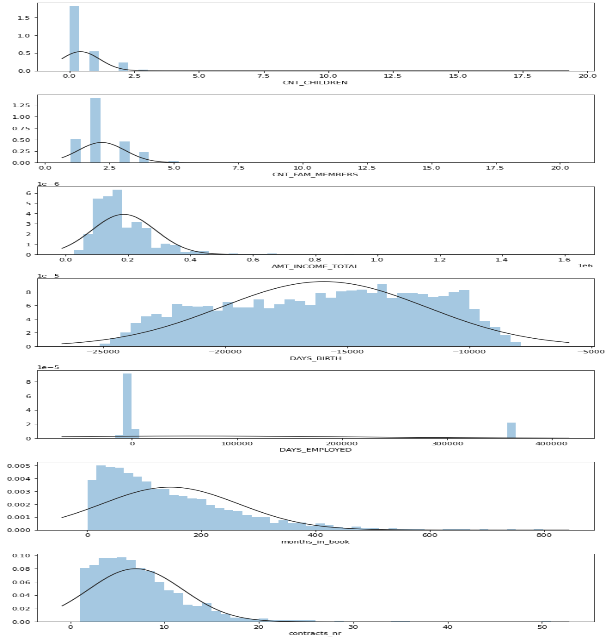
Now for identifying a customer id as a good or bad customer, we calculate the percentage of 'b' and 'g' trait present in that id; if there are at least 5% 'b' traits present in that id, then we can say that that customer belongs to the class of bad customer. Even after setting such low threshold of 5%, the number of bad customers were only 1.6% indicating huge class imbalance.

Now we check for missing values in both data frames, and we find that there are no missing values in any of our data frames except the attribute "OCCUPATION\_TYPE" in the credit record data frame. So, we assign every missing value in that attribute to "Not Available" and make our data set complete. After this, we merge both of our datasets in a single data set, which makes our final data set size (36457, 20)

**Data Visualisation**

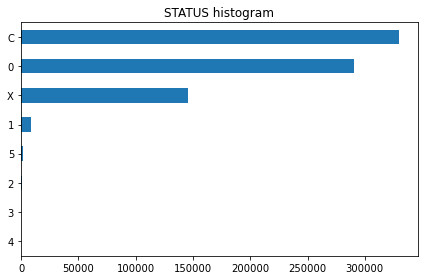
1. **Application Record** –

We started by plotting the curves to know in what way our data is distributed by plotting histogram and box plots and if there is any data that is redundant. While plotting the curves, we found that the mobile attribute i.e whether the customer has mobile phone or not is redundant because every customer has mobile phone so we removed that feature from our data set.

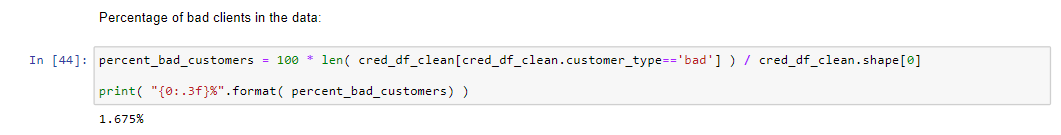


Apart from this, we saw that there is a sudden spike in employment days. We explored further to realise that this attribute was same for all people whose income type is pension. We dealt with this problem in normalisation.

1. **Credit Record** –

Plot of credit history for all the customers was drawn and thoroughly visualised, from which we get knowledge that the most of the bad customers does not return the loan money till 5 months if once the Due date is crossed. 

After the plotting work we went on to look an example of a bad customer so that we can get any useful insight which we have been missing while plotting and then we calculated the percentage of bad customer in the training data set

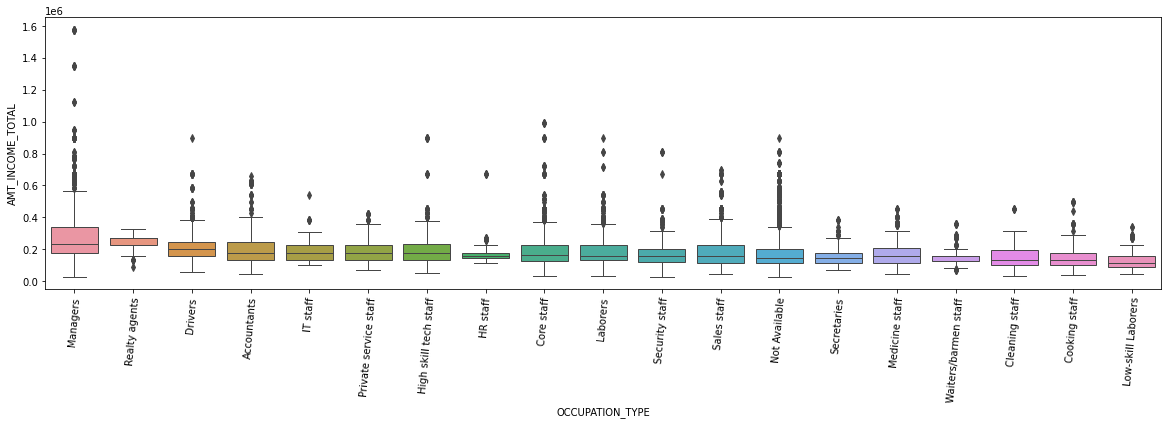


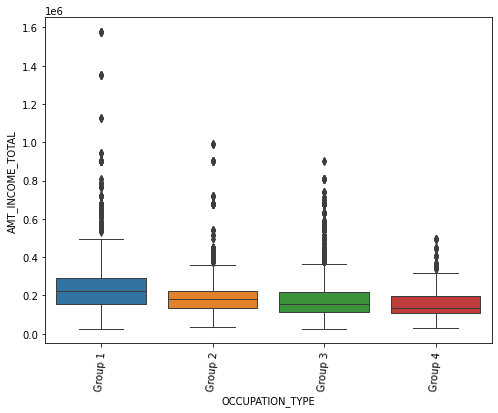
**Data Reduction**

After doing the visualisation, we checked correlations among the attributes to reduce the redundancy among the attributes. We used Kendall Tau test**2** for calculating the correlation because our data does not form a normal gaussian plot. Then a Heat plot was made using which we removed similar attributes.



Then we further shortened the attributes so that most of the attributes are of same type, after that we have done the clustering and the similar customers are grouped into 4 groups –

Initially 

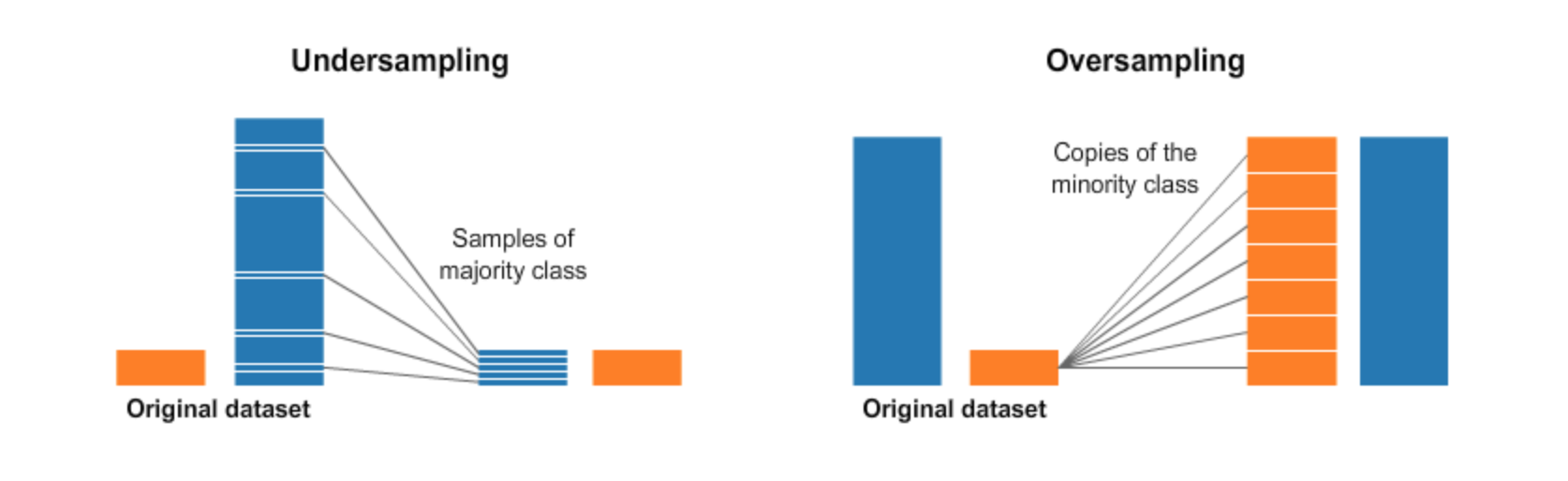
After shortening, cleaning and clustering - 

**Data Transformation and Oversampling**

The numerical attributes – “Months in book”, “Days employed”, “Income”, “Age”, “Count of family members” are skewed, have different range and contain a lot of outliers. To overcome this, the median and interquartile range were used to standardize these features. This is known as **robust scaling.** For the attribute “Days employed”, we replaced the huge erroneous value of 365423 by 1 and normalized the remaining values by decimal scaling. This is done so that both tree based and regression (or related algorithms) can perform well.

|  |  |
| --- | --- |
| *Before Transformation* |  |
| *After Transformation* |  |

Our data-set has only 1% bad customers which makes it highly imbalanced. Dealing with this imbalance is perhaps the most important and challenging aspect of this project. We tried three ways to overcome the problem – Undersampling, Oversampling, mixed over and under sampling.



Best results were obtained upon using Oversampling using SMOTE algorithm. SMOTE or **Synthetic Minority Oversampling Technique** works by utilizing a **k-nearest neighbour**algorithm to create synthetic data. SMOTE first starts by choosing random data from the minority class, then k-nearest neighbours from the data are set. Synthetic data would then made between the random data and the randomly selected k-nearest neighbour**2**.

We first split the data into training and test set. Then we applied the over-sampling. It is important to note that if we first do oversampling and then do the splitting, then the results obtained will be **optimistically** **biased**. It is because in that case, the test data will be having impurity of synthetic samples created by SMOTE**3**.

|  |  |  |
| --- | --- | --- |
| Training data before Oversampling | Good Customers: 25177 | Bad Customers: 342 |
| Training data after Oversampling | Good Customers: 25177 | Bad Customers: 25177 |

**Modelling and Evaluation**

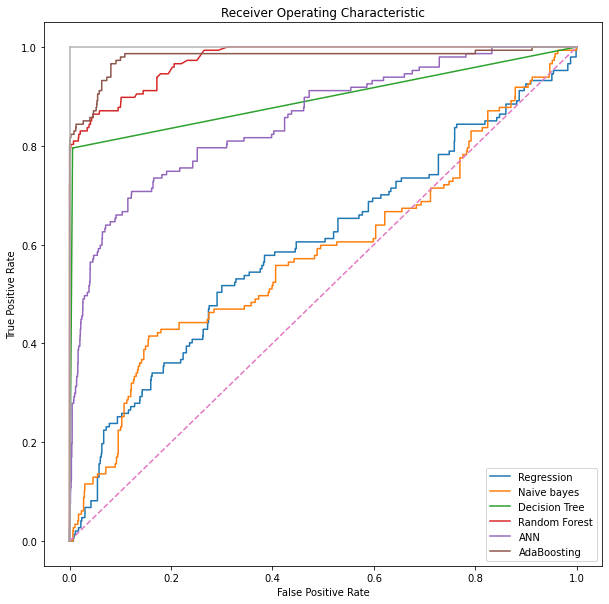
* Accuracy is not the correct performance metric since our test dataset is highly imbalanced (77:1). Even if a model predicts all the customers as good customers, it will still have an accuracy of **.**
* Models were judged based on their precision and recall. Precision tells us how many customers predicted as bad were actually bad. Recall tells us what percentage of bad customers were correctly classified.
* Usually, there is a trade-off between precision and recall. Recall would be given more importance by banks having plethora of customer applications. This is because they can afford to deny credit card to some good customers while minimizing the approval of bad customer applications. Similarly, for banks having shortage of customers, precision will be more important.
* F1 score, which is the harmonic mean of precision and recall is chosen as the overall performance metric.

|  |  |  |  |
| --- | --- | --- | --- |
| **SNo** | **Model** | **Parameters** | **Justification for parameters** |
| 1 | Naïve Bayes Classifier | No parameter required | NA |
| 2 | Logistic Regression | Maximum iterations: 1000 | * Iterations were increased from default value of 100 because of large dataset |
| 3 | Decision tree | Quality of split: Gini  Maximum depth: 27  Minimum sample split: 2 | * Not much change in F1 score was observed with different Splitting criteria. However, best results were obtained with default splitting criteria: Gini index * Maximum depth and Minimum sample split were obtained by running a for loop keeping other parameters as constant and maximizing F1 score |
| 4 | Random forest | Quality of split: Gini  Maximum depth: 27  Min sample split: 2  Min samples at leaf: 1  Number of decision trees: 250 | * Since random forest is a collection of decision trees, first three parameters were kept the same as decision tree. * Minimum samples at leaf node by minimizing F1 score for different values by running a for loop * Number of decision trees were chosen by manual tuning |
| 5 | Artificial Neural Network | Hidden layers: 3  Nodes in hidden layer: 15  Activation function for hidden layer: Relu function  Activation function for output layer: Sigmoid function | * No substantial increase in F1 score was seen upon increasing the number of hidden layers beyond 3. * Activation function were chosen using heuristics. No substantial improvement was seen upon changing them. * Nodes in hidden layer: As a heuristic they were taken as 2/3 of sum of nodes in output layer and input layer**4** |
| 6 | Support Vector Machine | Kernel: RBF  C value: 0.7 | * Best F1 score was obtained using RBF kernel * Value of C was obtained by manual tuning |
| 7 | AdaBoost | Number of decision trees: 200 | * Number of decision trees were chosen by manual tuning |

|  |  |  |
| --- | --- | --- |
| **Support Data** | Good Customers | Bad Customers |
| Train | 25177 | 25177 |
| Test | 10791 | 147 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Accuracy** | | **Precision** | | **Recall** | | **F1 Score** | |
| SNo | Model | Train | Test | Train | Test | Train | Test | Train | Test |
| 1 | Naïve Bayes Classifier | 53 | 11 | 52 | 1 | 97 | 92 | 68 | 3 |
| 2 | Logistic Regression | 71 | 79 | 75 | 2 | 64 | 36 | 69 | 4 |
| 3 | Support Vector Machine | 90 | 88 | 90 | 7 | 91 | 58 | 90 | 12 |
| 4 | Artificial Neural Network | 94.8 | 95.7 | 93 | 18 | 93 | 59 | 95 | 27 |
| 5 | Decision tree | 99.8 | 98.6 | 100 | 49 | 100 | 80 | 100 | 61 |
| 6 | Random forest | 99.9 | 99.4 | 100 | 80 | 100 | 80 | 100 | 80 |
| 7 | AdaBoost | 100 | 99.6 | 100 | 86 | 100 | 82 | 100 | 84 |

**Results and Interpretations**



1. The worst performing models were Naïve Bayes and Logistic Regression. These models are practically useless as their ROC curves lie close to Y=X line. The poor performance of naïve bayes is due to its class conditional independence as the customer attributes are not independent. For example, if a person has an age of 42 then the probability of his “Income Type” attribute being “Student” is 0. Similarly, lot of other dependencies are there. Logistic Regression performs poorly simply because the dependent and independent variables are not linearly separable.
2. SVM and ANN perform poorly because of heavy class imbalance. The artificial samples generated using SMOTE are not good enough training data for these models. The solution is to either collect more data or use deep neural networks**5**.
3. Tree based models proved to be excellent for this classification. AdaBoost gives the best performance. It performs slightly better than Random Forest as it gives weights to the individual decision trees based on their accuracy unlike random forest which considers all the decision trees as equal. However, since the decision tree was complex, the splits and nodes were not intuitive and hence could not be interpreted.

**References**

1. [EDA-Vintage-Analysis](https://www.listendata.com/2019/09/credit-risk-vintage-analysis.html)
2. [Kendall-Tau](https://en.wikipedia.org/wiki/Kendall_rank_correlation_coefficient#:~:text=In%20statistics%2C%20the%20Kendall%20rank,association%20between%20two%20measured%20quantities.&text=can%20be%20formulated%20as%20special%20cases%20of%20a%20more%20general%20correlation%20coefficient.)
3. [SMOTE](https://arxiv.org/pdf/1106.1813.pdf)
4. [Order of Split and Oversampling](https://stats.stackexchange.com/questions/60180/testing-classification-on-oversampled-imbalance-data)
5. [Nodes in ANN hidden layer](https://www.researchgate.net/post/How-to-decide-the-number-of-hidden-layers-and-nodes-in-a-hidden-layer#:~:text=The%20number%20of%20hidden%20neurons,size%20of%20the%20input%20layer.)
6. [SVM under imbalanced data](http://scholar.google.co.in/scholar_url?url=https://link.springer.com/content/pdf/10.1007/978-3-540-30115-8_7.pdf&hl=en&sa=X&ei=rZDeX4WUA9GrywTtr5GwDA&scisig=AAGBfm3Z0CQmztMM-ZUNFNtEbz-GZkZoxw&nossl=1&oi=scholarr)

THANK YOU ☺