SCORECARD DEVELOPMENT

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

By

Ishika (181328)

Under the supervision of

Dr. Ravindara Bhatt

to

Department of Computer Science & Engineering and Information Technology
Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

CERTIFICATE

Candidate's Declaration

I hereby declare that the work presented in this report entitled **"SCORECARD DEVELOPMENT"** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from January 2022 to May 2022 under the supervision of **Dr. Ravindara Bhatt** (Associate Professor, Department of CSE).
The matter embodied in the report has not been submitted for the award of any other degree or diploma.

[illegible]

Ishika, 181328

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

[illegible]

Dr. Ravindara Bhatt
Associate Professor
Department of CSE
Dated: 26-05-2022

[illegible]

Aman Bhardwaj
Sr. Analytic Consultant
GDCA
Dated: 26-05-2022

ACKNOWLEDGEMENT

I hereby declare that this project has been done by me under the supervision of Dr. Ravindara Bhatt (Associate Professor) in the department of Computer Science Engineering and Information Technology, Jaypee University of Information Technology. I also declare that neither this project nor any part of this project has been submitted elsewhere for award of any degree or diploma.

Supervised by:
Dr. Ravindara Bhatt
Associate Professor
Department of Computer Science & Engineering and Information Technology
Jaypee University of Information Technology

Submitted by:
Ishika (181328)

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

With the continuous improvement in the banking domain many humans now are making use of bank loans but the bank has its confined resources which it can grant to specific people only, thus finding out to which person the loan may be granted such that it is a secure alternative for the bank is a complicated procedure. One of the quality indicators of the loan is loan status. It would not display the entirety at once, however it acts as a primary step of the loan lending process. So in this project we attempt to lessen this risk factor in the back of selecting the safe person in order to reduce efforts of banks and save its assets.

Recovery of loans is a very crucial factor contributing in the financial statements of a bank. It is very hard to forecast the probability of loan repayment by the customer. In past years, loan approval prediction system is a domain that many researchers have worked on. Machine Learning strategies are very helpful in forecasting results for large amounts of data. In this project, we build a scorecard by carrying out various number of steps and tasks, that helps in scoring each loan applicant based on certain characteristics. That score further helps to decide a bank whether that applicant would be safe to grant a loan or not.

CHAPTER- 1

INTRODUCTION

1.1 Introduction

Handing out the loans is the core commercial enterprise of almost each bank. The primary portion of the bank's belongings come straight from the profits gained from the loans granted by the banks. The top goal in the banking environment is to invest their assets in reliable hands where it is. These days almost all banks/financial institutions approve loans after a regress manner of verification and validation but still there is no confirmation whether the selected applicant is the deserving right applicant out of all applicants. With the use of this system, we can forecast if that particular applicant is right or not and the complete process of validation of attributes is led by a machine learning algorithm. It will aid in finding the loyalty of clients and help the bank in minimizing the credit risk and maximize the volume of credits, which is the main objective of any bank to function.

What is Credit Risk?
Credit risk is the possibility of a loss resulting from a borrower's failure to repay a loan or meet contractual obligations. Generally, it refers to the risk that a lending institution may not receive the owed amount and interest, which can result in an interruption of cash flows and increased costs for collection.
Credit risks are estimated based on the applicant's overall ability to repay a loan as per its original conditions and terms. Credit risk is a crucial factor that needs to be taken care of and minimized to the maximum extent possible in order for a bank to function.
Although it's impossible to know exactly who will default, proper assessment and managing credit risk can decrease the impact of a loss.

Two most important points to determine in the banking sector are: "1) Is the borrower risky and how much? 2) Should the borrower be lent loan provided the risk? The reaction to the first query governs the applicant's rate of interest. Interest rate, along with else matters (for instance, money's time value), validates the risk potential associated with the borrower, i.e. more the

interest rate, more the risk associated with the borrower. We will then speak whether the applicant is appropriate to grant the loan depending on the rate of interest. Lenders (investors) grant loans to creditors in exchange for the guarantee of interest-providing reimbursement. Which means, the lender only gives a return (interest) if the borrower returns the loan amount. However, whether he/she repays the debt or not, the lender is the one losing money. Banks grant credit to customers in return for the promise of repayment. Some would default on their debts, not able to repay them due to various reasons. The bank keeps insurance to reduce the chance of fail in the scenario of a defaulter. The amount insured can cover the complete debt sum or just partly. Some of the banks use manual steps to find out whether a borrower is right for a loan depending on results or not. Procedures that were carried manually were maximum times efficient, but it was incapable when there was a huge no. of loan applicants. In those cases, decision making process would take a long span. Thus, as a result, this scorecard system can be used to evaluate the likelihood of an applicant being 'bad' in an efficient manner. This model draws out and introduces the vital characteristics of a borrower that have an effect on the borrower's loan status. As a final result, it outputs the probability of the borrower being safe.
The drawback of this version is that it emphasizes one-of-a-kind weights to everything however in real existence sometimes loans can be accepted on the premise of a single weighted thing only, which isn't possible through this system.

1.2 Problem Statement

This Scorecard Development Model will output a score that will represent the probability of the applicant (borrower) repaying on time if they are granted a loan or a credit card.
We'll begin by performing exploratory data analysis , then data transformation, followed by characteristics generation and selection and then building the model on those characteristics, and in the end we'll be testing the model built and interpreting the results generated by the scorecard.
This project has taken the data from previous records of the customers to whom loans were approved or declined depending on a set of attributes. So, the model is trained on previous records to obtain accurate results. The main purpose of this project is to predict the safety of loans and decrease the credit risk.
The results generated by this model will help to make a bank manager's work easier and faster.

1.3 Objectives

Scorecards are a very useful tool for bank employees and for the applicants also. The goal of this project is to find a quick, fast and easy path to select the fine and safe applicants. The bank can gain special precedence from this. The objective of this project is to develop an originations scorecard. To build a scorecard, we need to identify the relevant characteristics providing maximum lift to the model from the provided list of attributes. Later, those will be the characteristics on which an application would be scored. This could be carried out by predicting if the loan can be assigned to that applicant on the basis of various parameters like credit score, income, age, marital status, gender, education etc. So, the scorecard helps in identifying the applications that can be passed or granted with the least amount of risk possible. A time restrict can be set for the applicant to see whether his/her loan can be passed or not. This project is exclusively for dealing of the managing authority of a Bank / finance company, the complete process of forecasting is executed in a private manner and no stakeholders would be able to change the processing. Results obtained against a specific loan Id might be sent to different departments of bank such that they can take the required step on application. This aids all other departments to do other procedures and formality.

1.4 Methodology

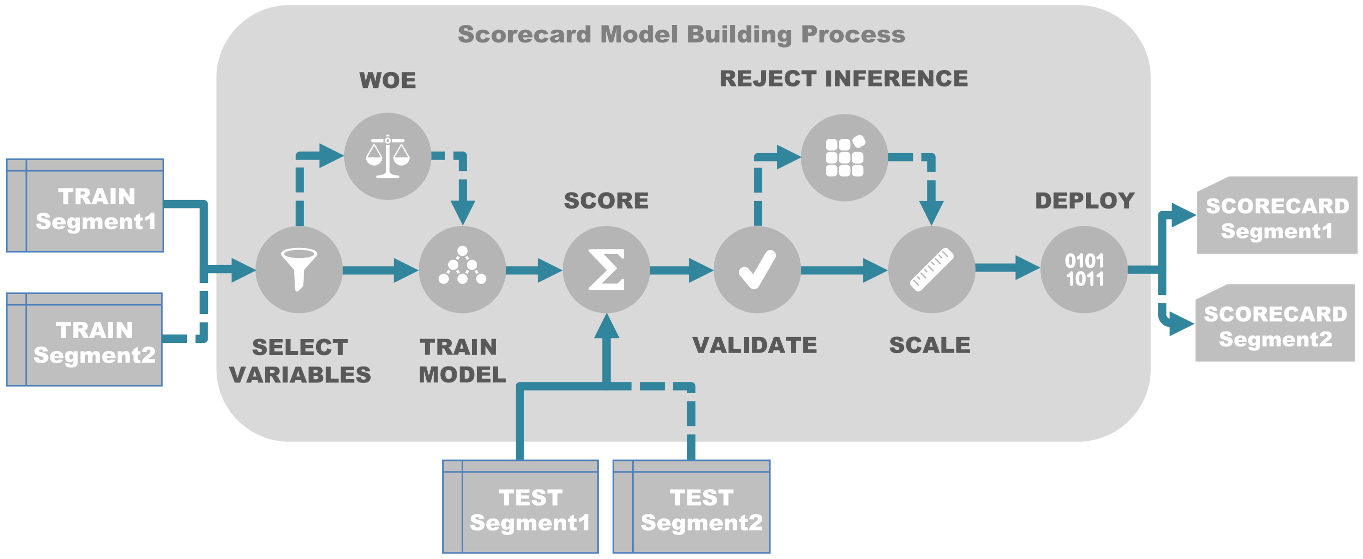


Fig. 1 Methodology Used

1.5 Organization

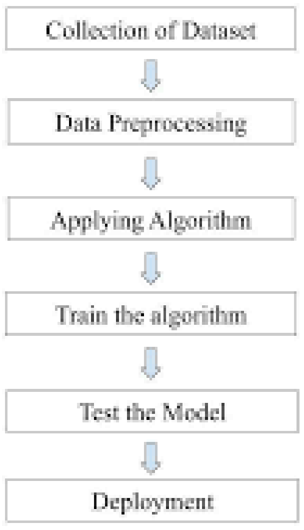


Fig. 2 Organization

CHAPTER-2

LITERATURE REVIEW

Sr. No.	Ref	Author	Description
1	1	Sergey Krylov	The author proposes theoretical and methodological aspects of the applied strategic analysis (ASA) as a new instrument of the balanced scorecard (BSC) comprehensive study of the organization economic activity.
2	2	Alejandro Correa, Catherine Nieto, Andrés González, Darwin Amezquita	In this paper the use of cluster analysis as a part of a predictive algorithm is proposed. This methodology is applied by first determining to which cluster a prospect client belongs, and then calculate a specific credit risk scorecard for each cluster.
3	3	Rory P. Bunker, M. Asif Naeem, Wenjun Zhang	In this paper, it is investigated the extent to which features derived from bank statements provided by loan applicants, and which are not declared on an application form, can enhance a credit scoring model.

Table 1. Literature survey

CHAPTER-3

SYSTEM DEVELOPMENT

3.1 Dataset Used

The dataset contains information of 65,000 applications containing 66 column fields.

The data types of the fields are:

Continuous – 51

Categorical – 15

There were four roles that were assigned to the variables to be used in the model development:

- Target
- Weight
- Partition
- Predictor

Following are some of the variables that were present in the dataset:

Loan to Value

No. of mortgages

Credit Bureau Score

No. of delinquencies

Open Date

Closed Date

Occupation

Credit Limit Assigned

Loan Term

3.2 Flow Chart

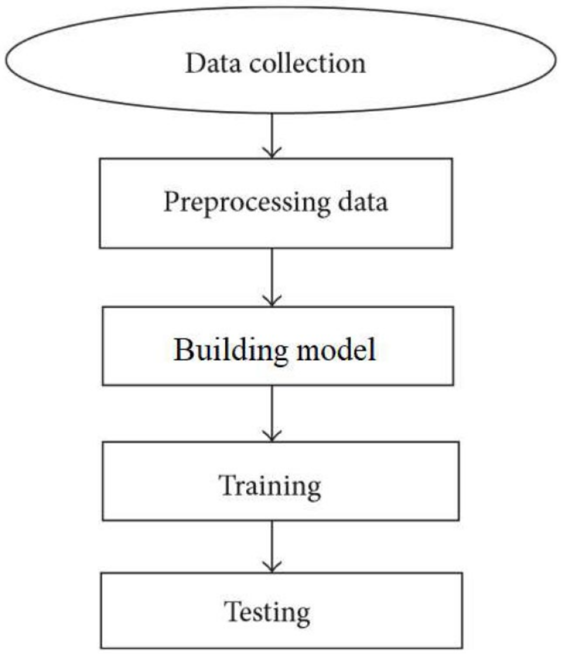


Fig. 3 Flow Chart

3.3 Data Preprocessing

Data preprocessing refers to the process of changing raw data into a format that can be understood. Data in the actual world is dirty and is full of inconsistencies, noise, incomplete information, and missing values. Data Preprocessing includes the steps we need to follow to transform or encode data so that it could be easily understood and processed by the machine. It is also a crucial stage in data mining as raw data cannot be worked with. The quality of the data needs to be seen before using machine learning or data mining algorithms.

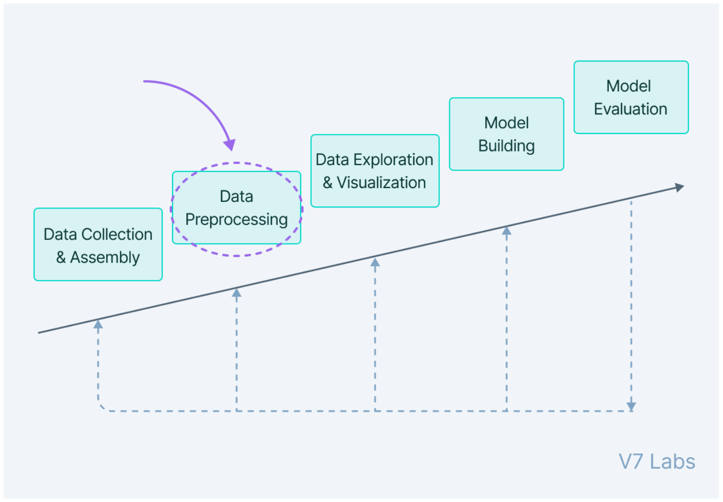


Fig.4 Depicting the stage of data preprocessing

Data Preprocessing has the following major tasks:

1. Data integration
2. Data cleaning
3. Data reduction
4. Data transformation

Data Integration
Data Cleaning
Data Preprocessing
Data Transformation
Data Reduction or Dimension Reduction

Fig.5 Major Tasks in Data Preprocessing

1. Data Cleaning :Data Cleaning is typically performed as part of data preprocessing to clean the raw records by dealing with missing values, smoothing the noisy data, removing inconsistencies, and removing outliers.
2. Data Integration : Data Integration is among the data preprocessing steps that are used to merge the records found in more than one sources into a single larger data store like a data warehouse. It is needed specifically whilst we are working on solving a real-world problem like finding the existence of nodules from the images of CT scan. The only one alternative is to combine the images from many medical nodes to form one larger database.
3. Data Transformation : Once data has been cleaned, we need to consolidate the quality data into alternate forms by changing the values, structure, or format of data using the data transformation strategies such as Generalization, Normalization and Feature Selection.
4. Data Reduction : The size of the dataset in a data warehouse can be way too huge to be dealt by data analysis and data mining algorithms. One viable answer is to achieve a reduced representation of the dataset that is much less in size but gives the same quality of analytical outputs. Dimensionality reduction techniques are used to perform feature extraction. The dimensionality of a dataset refers to the attributes or individual features of the data. This approach's objective is to lessen the number of redundant features we consider in machine learning algorithms. Dimensionality reduction can be done using techniques like Principal Component Analysis(PCA) etc.

3.4 Algorithm Used

Following machine learning classification models go in the backend while building and training the model on the characteristics chosen in order to generate a scorecard as an output. The quick characteristics of each model is explained below.

3.4.1 Logistic Regression

Logistic regression is a useful supervised ML algorithm which is used to solve binary classification cases (when target variable is categorical). In this we model the probability of a discrete result given an input variable.One way to imagine about logistic regression is that it is a linear regression however for classification problems. Logistic regression makes use of a logistic function defined

below which models an output variable that is binary. The main differentiation between linear regression and logistic regression is that logistic regression's value range is always between 0 and 1. Additionally, in contrast to linear regression, logistic regression does not need to have a linear relationship between input and output variables. This happens due to the application of a nonlinear log transformation to the odds ratio.
In the case of Logistic regression, we do not fit a regression line, but we fit an "S" shaped logistic function, which forecasts two greatest values “(0 or 1)”. The logistic function curve shows the probability of something, for example, despite of whether the cells are destructive or not, a mouse is said to be corpulent or not depending on its weight, and so on. It is an important algorithm because it finds out probabilities and segregate the use of various varieties of data and very easily finds the most effective attributes that are used in classification.

$$\text{Logistic function} = 1 / 1 + e^{-x}$$

, where x is the input variable

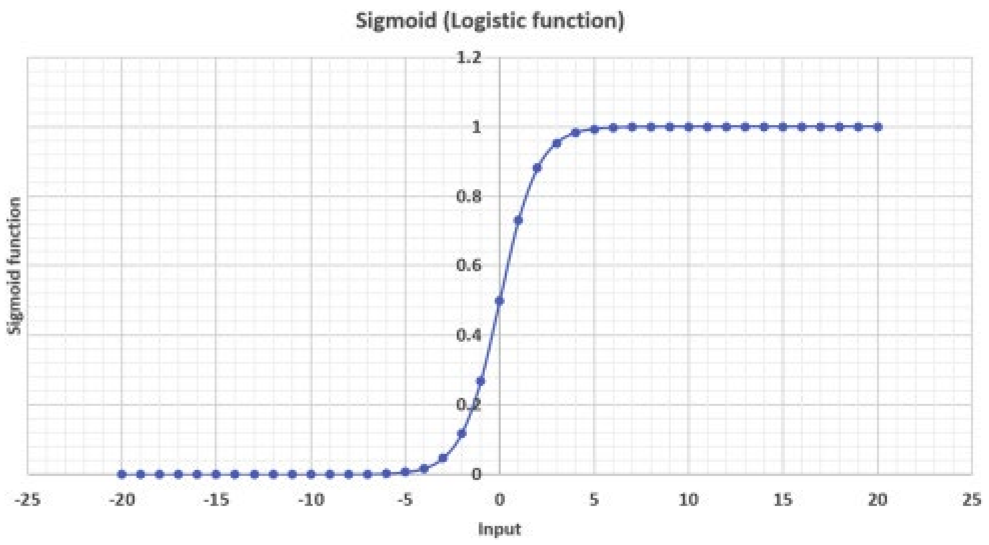


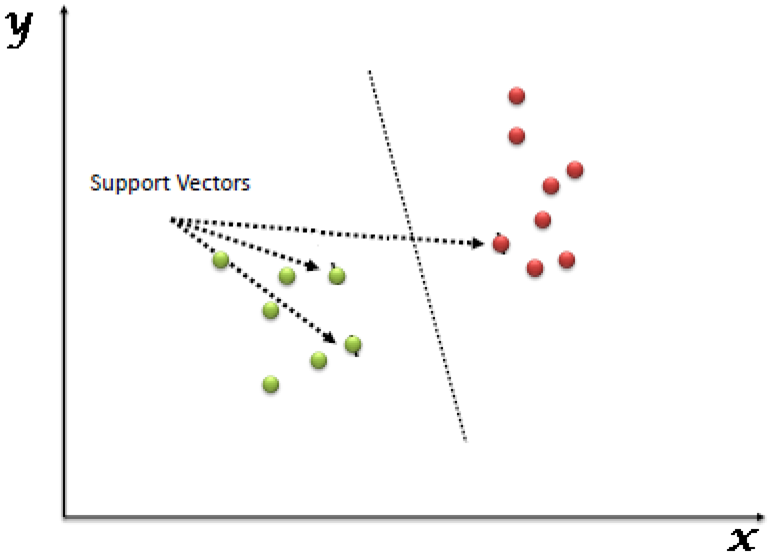
Fig. 6 Logistic Regression applied for a range of -20 to 20

3.4.2 Support Vector Machine

Support Vector Machine (SVM) is used to tackle with both classification or regression problems and it comes in the category of supervised machine learning algorithm. Although, in most cases it is used to deal with classification problems. A support vector machine is a statistical learning method that works on the foundation of minimization of structure risk.In the SVM algorithm, every data item is first plotted as a single point in n-dimensional space (where n refers to the no. of features present) where the result of every attribute is the value of a specific coordinate. After that, we carry out the classification by looking for the hyper-plane which differentiates between the two different class pretty good.

Support Vectors simply refers to the coordinates of individual observation. SVM classifier is a partition that divides the 2 classes by a line or hyper-plane in best possible way.

In Python, scikit-learn is used to implement SVM.



3.4.3 Decision Tree

Both classification as well as regression problems can be tackled by the use of Decision trees. A decision tree can be called a map of the potential results of a series of related options. It provides an individual or organization the opportunity to weigh potential actions against one another on the basis of their costs, probabilities, and advantages. They can come in use either to make a informal discussion or to lay out an algo that forecasts the best mathematical option.

A decision tree generally begins with a unary node, which further divides into potential outcomes. Each one of those outcomes results in more additioned nodes, which section off into other likelihoods. This provides it a shape like a tree.

Decision trees are popular for the following reasons:

- They are easy to learn
- They can be handy with or without the availability of hard data, and any data needs minimum preparation

- Existing trees could accumulate new options
- They pick out the best of several options
- Due to their ability to join with other decision-making tools

3.4.4 K Nearest Neighbour (KNN)

K Nearest Neighbor algorithm comes in the Supervised Learning category of Machine Learning and is applied in classification problems (in most cases) and regression. The algorithm can come in handy to tackle both classification and regression queries. The symbol K denotes the number of nearest neighbors to a new not known variable which has to be classified or predicted. KNN is very easy to deploy and is used widely as an initial step in any setup of machine learning. It is frequently used as a benchmark for more difficult classifiers like Support Vector Machines (SVM) and Artificial Neural Networks (ANN). It has proved to be a flexible algorithm that can also be used for dealing with missing values and to resample the data. As K Nearest Neighbor indicates it takes into account K Nearest Neighbors (data points) in order to forecast the continuous value and class for data points that are new. The algorithm's learning is Instance-based learning, Lazy learning and non-Parametric.

How to choose the value for K?

The value of K is a crucial parameter in the KNN algorithm. Some points to keep in mind for choosing the Value of K are:

1. Using error curves: In the figure below, error curves are shown for various K values for both training and test dataset.

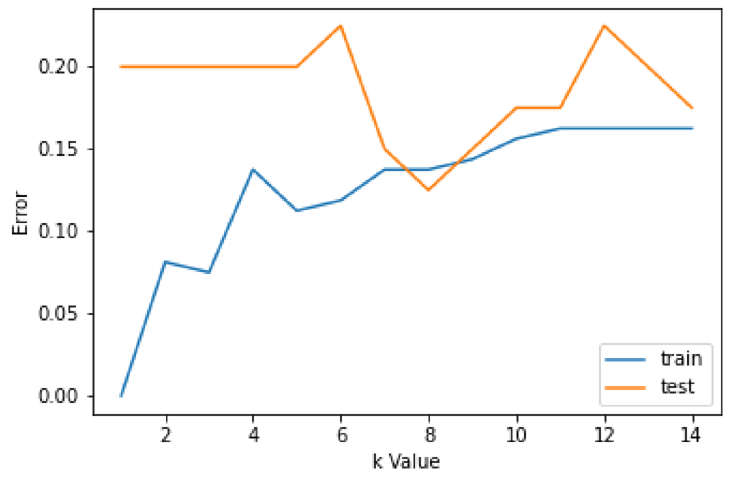


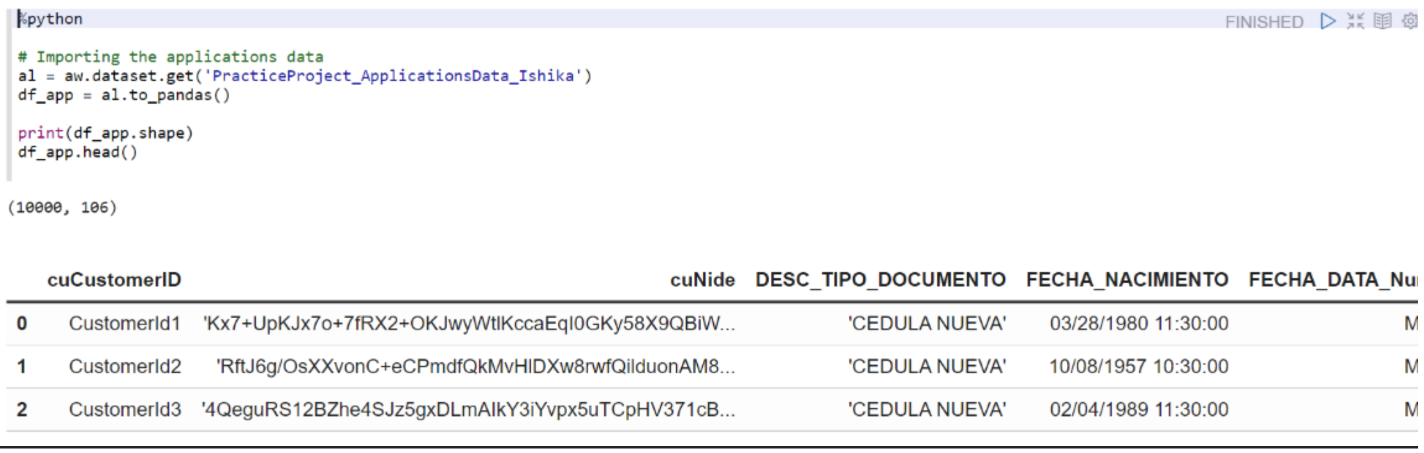
Fig. 7 Using error curve to choose a value of K

2. Domain knowledge also plays a deciding role in choosing the value of K.

3. The value of K should be odd in case of binary classification.

3.5 Model Development

3.5.1 Reading the dataset



```
%python

# Importing the applications data
al = aw.dataset.get('PracticeProject_ApplicationsData_Ishika')
df_app = al.to_pandas()

print(df_app.shape)
df_app.head()
```

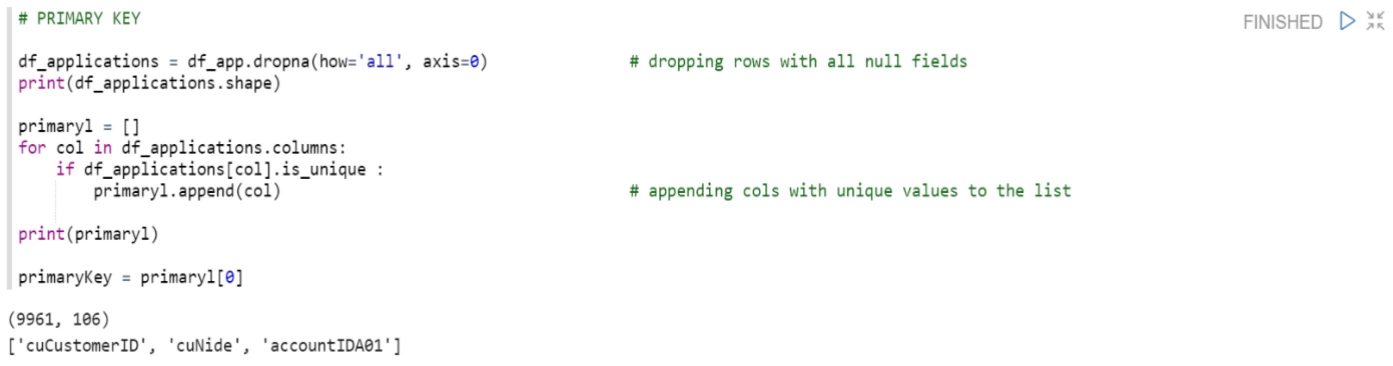
```
(10000, 106)
```

	cuCustomerID	cuNide	DESC_TIPO_DOCUMENTO	FECHA_NACIMIENTO	FECHA_DATA_Nu
0	CustomerId1	'Kx7+UpKJx7o+7fRX2+OKJwyWtlKccaEqI0GKy58X9QBiW...	'CEDULA NUEVA'	03/28/1980 11:30:00	M
1	CustomerId2	'RftJ6g/OsXXvonC+eCPmdfQkMvHIDXw8rwfQilduonAM8...	'CEDULA NUEVA'	10/08/1957 10:30:00	M
2	CustomerId3	'4QeguRS12BZhe4SJz5gxDLmAIkY3iYvpx5uTCpHV371cB...	'CEDULA NUEVA'	02/04/1989 11:30:00	M

Fig. 8 Importing and Reading the data

3.5.2 Data Preprocessing

Identifying and assigning the primary key to the dataset.

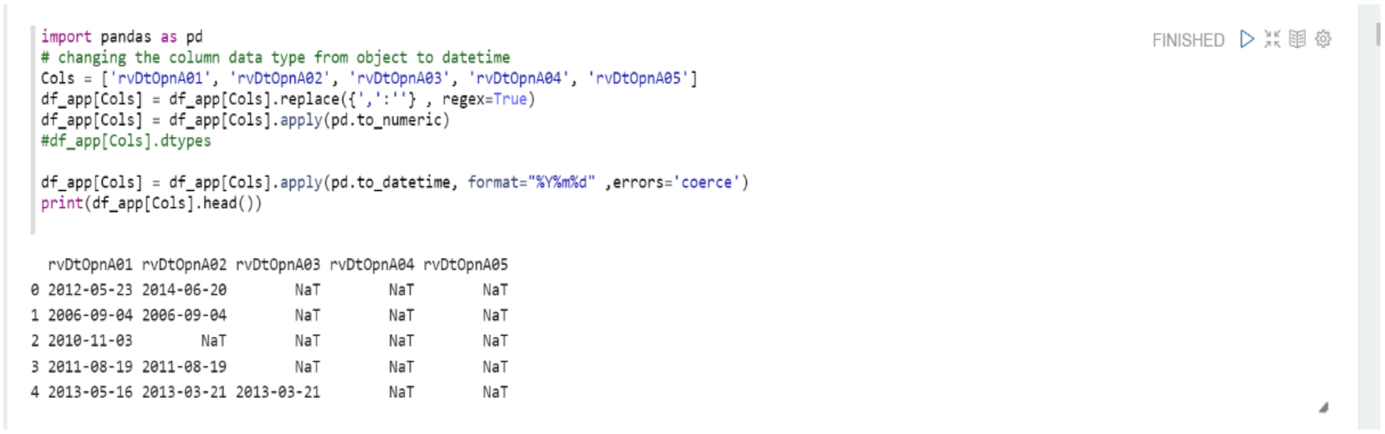


```
# PRIMARY KEY

df_applications = df_app.dropna(how='all', axis=0)              # dropping rows with all null fields
print(df_applications.shape)

primaryl = []
for col in df_applications.columns:
    if df_applications[col].is_unique :
        primaryl.append(col)                                     # appending cols with unique values to the list

print(primaryl)

primaryKey = primaryl[0]
```

```
(9961, 106)
['cuCustomerID', 'cuNide', 'accountIDA01']
```

Fig. 9 Assigning the Primary key to the dataset

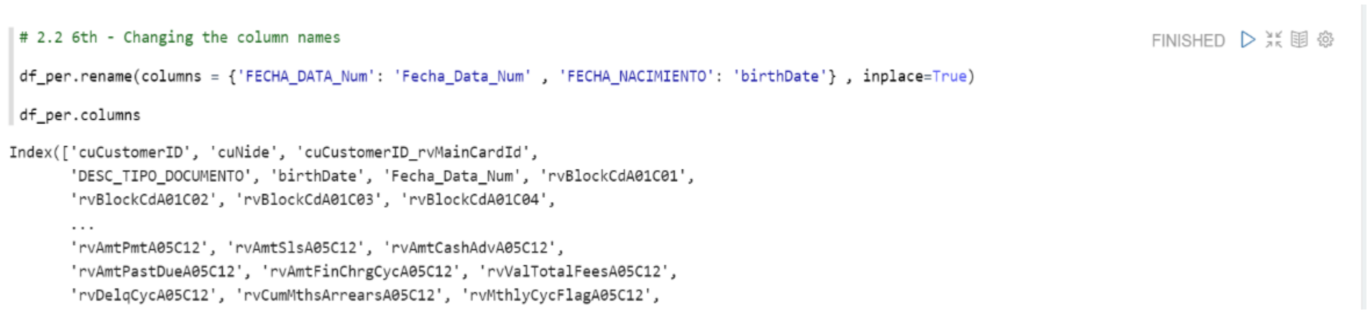
Converting the data types of the variables. For instance, here date columns had object data type which have been converted to datetime.



```
import pandas as pd
# changing the column data type from object to datetime
Cols = ['rvDtOpnA01', 'rvDtOpnA02', 'rvDtOpnA03', 'rvDtOpnA04', 'rvDtOpnA05']
df_app[Cols] = df_app[Cols].replace({',':''} , regex=True)
df_app[Cols] = df_app[Cols].apply(pd.to_numeric)
#df_app[Cols].dtypes

df_app[Cols] = df_app[Cols].apply(pd.to_datetime, format="%Y%m%d" ,errors='coerce')
print(df_app[Cols].head())
```

```
  rvDtOpnA01 rvDtOpnA02 rvDtOpnA03 rvDtOpnA04 rvDtOpnA05
0 2012-05-23 2014-06-20        NaT        NaT        NaT
1 2006-09-04 2006-09-04        NaT        NaT        NaT
2 2010-11-03        NaT        NaT        NaT        NaT
3 2011-08-19 2011-08-19        NaT        NaT        NaT
4 2013-05-16 2013-03-21 2013-03-21        NaT        NaT
```

Fig. 10 Converting data types of variables

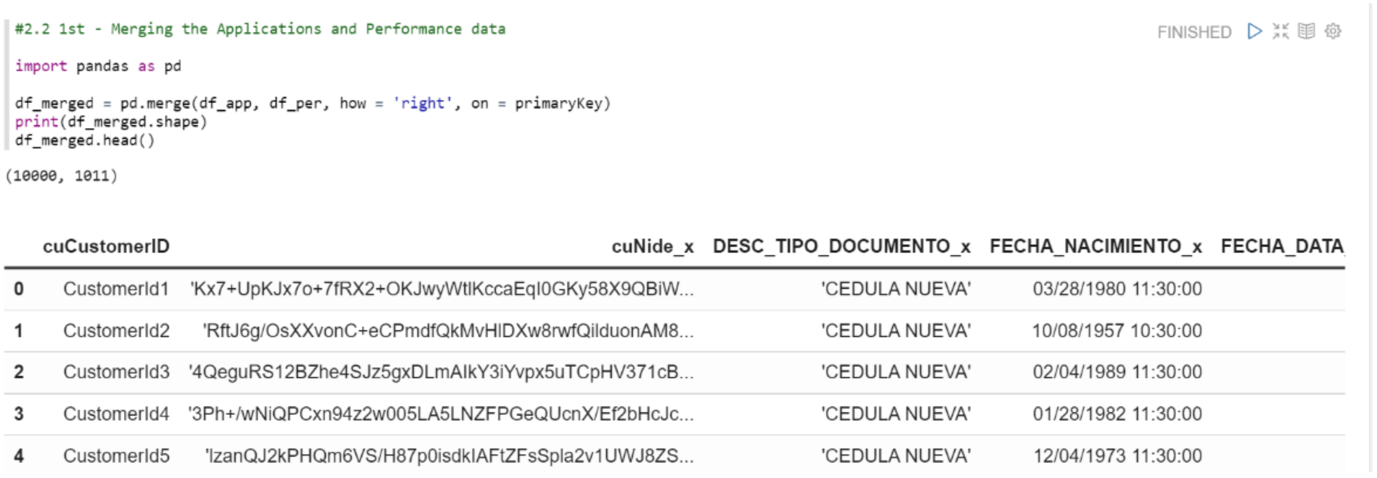


```
# 2.2 6th - Changing the column names

df_per.rename(columns = {'FECHA_DATA_Num': 'Fecha_Data_Num' , 'FECHA_NACIMIENTO': 'birthDate'} , inplace=True)

df_per.columns
```

```
Index(['cuCustomerID', 'cuNide', 'cuCustomerID_rvMainCardId',
       'DESC_TIPO_DOCUMENTO', 'birthDate', 'Fecha_Data_Num', 'rvBlockCdA01C01',
       'rvBlockCdA01C02', 'rvBlockCdA01C03', 'rvBlockCdA01C04',
       ...
       'rvAmtPmtA05C12', 'rvAmtSlsA05C12', 'rvAmtCashAdvA05C12',
       'rvAmtPastDueA05C12', 'rvAmtFinChrgCycA05C12', 'rvValTotalFeesA05C12',
       'rvDelqCycA05C12', 'rvCumMthsArrearsA05C12', 'rvMthlyCycFlagA05C12',
```

Fig. 11 Changing the column names

Applications and Performance datasets are merged on the primary key in order to conduct further validations and to prep the data for model building.



```
#2.2 1st - Merging the Applications and Performance data

import pandas as pd

df_merged = pd.merge(df_app, df_per, how = 'right', on = primaryKey)
print(df_merged.shape)
df_merged.head()
```

```
(10000, 1011)
```

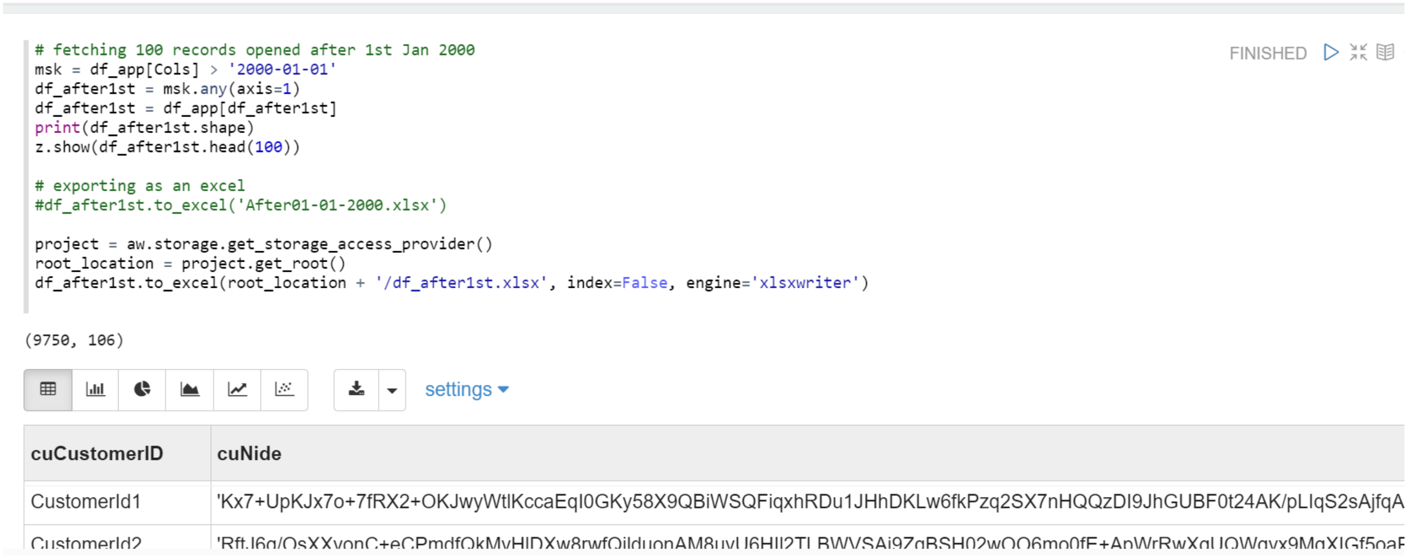
	cuCustomerID	cuNide_x	DESC_TIPO_DOCUMENTO_x	FECHA_NACIMIENTO_x	FECHA_DATA
0	CustomerId1	'Kx7+UpKJx7o+7fRX2+OKJwyWtlKccaEqI0GKy58X9QBiW...	'CEDULA NUEVA'	03/28/1980 11:30:00	
1	CustomerId2	'RftJ6g/OsXXvonC+eCPmdfQkMvHIDXw8rwfQilduonAM8...	'CEDULA NUEVA'	10/08/1957 10:30:00	
2	CustomerId3	'4QeguRS12BZhe4SJz5gxDLmAIkY3iYvpx5uTCpHV371cB...	'CEDULA NUEVA'	02/04/1989 11:30:00	
3	CustomerId4	'3Ph+/wNiQPCxn94z2w005LA5LNZFPGeQUcnX/Ef2bHcJc...	'CEDULA NUEVA'	01/28/1982 11:30:00	
4	CustomerId5	'IzanQJ2kPHQm6VS/H87p0isdkIAFtZFsSpla2v1UWJ8ZS...	'CEDULA NUEVA'	12/04/1973 11:30:00	

Fig. 12 Merging the two datasets

3.5.3 Data Investigation and Validation

Data investigation takes place where data is observed for accuracy and certain checks and validations take place in order to ensure the integrity of the data.

First hundred records of accounts opened after 1st Jan are fetched and saved and exported to an excel for further validations.



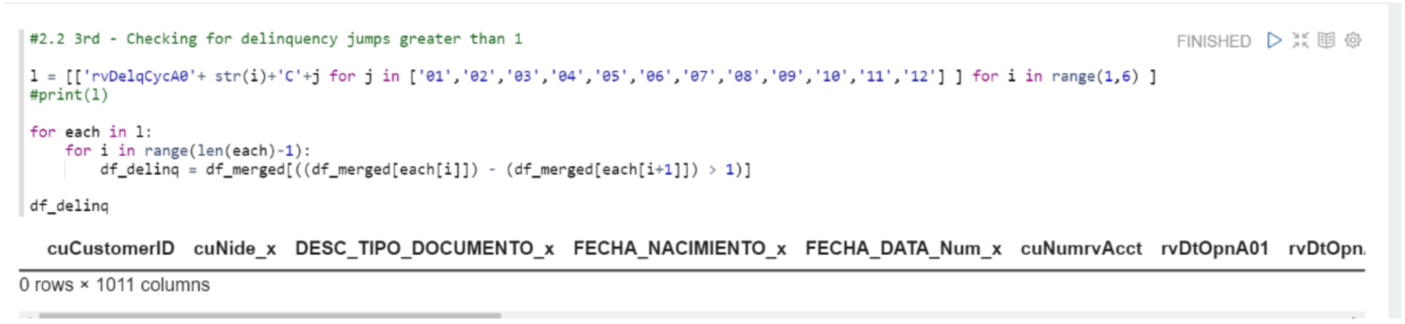
```
# fetching 100 records opened after 1st Jan 2000
msk = df_app[Cols] > '2000-01-01'
df_after1st = msk.any(axis=1)
df_after1st = df_app[df_after1st]
print(df_after1st.shape)
z.show(df_after1st.head(100))

# exporting as an excel
#df_after1st.to_excel('After01-01-2000.xlsx')

project = aw.storage.get_storage_access_provider()
root_location = project.get_root()
df_after1st.to_excel(root_location + '/df_after1st.xlsx', index=False, engine='xlsxwriter')
```

```
(9750, 106)
```

cuCustomerID	cuNide
CustomerId1	'Kx7+UpKJx7o+7fRX2+OKJwyWtlKccaEqI0GKy58X9QBiWSQFiqxhRDu1JHhDKLw6fkPzq2SX7nHQQzDI9JhGUBF0t24AK/pLIqS2sAjfqA
CustomerId2	'RftJ6g/OsXXvonC+eCPmdfQkMvHIDXw8rwfQilduonAM8uvU6HII2TLBWVSAi9ZqBSH02wQO6mo0fE+ApWrRwXqUQWqvx9MqXIGf5oaF

Fig. 13 Fetching specific records and exporting to excel

Certain checks and validations are applied on the merged data to check for data integrity.

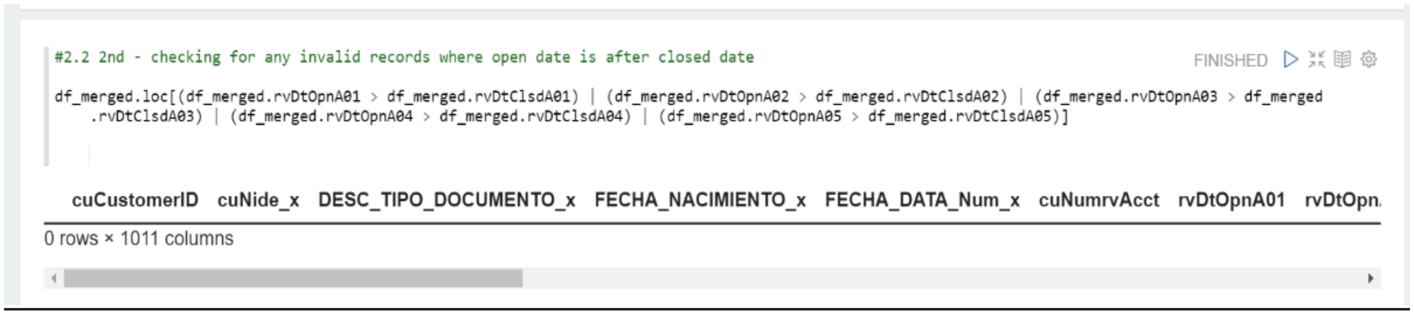


```
#2.2 3rd - Checking for delinquency jumps greater than 1

l = [['rvDelqCycA0'+ str(i)+'C'+j for j in ['01','02','03','04','05','06','07','08','09','10','11','12'] ] for i in range(1,6) ]
#print(l)

for each in l:
    for i in range(len(each)-1):
        df_delinq = df_merged[((df_merged[each[i]]) - (df_merged[each[i+1]]) > 1)]

df_delinq
```

```
   cuCustomerID  cuNide_x  DESC_TIPO_DOCUMENTO_x  FECHA_NACIMIENTO_x  FECHA_DATA_Num_x  cuNumrvAcct  rvDtOpnA01  rvDtOpn
0 rows × 1011 columns
```

Fig. 14 Delinquency jump check on data

Checking if closed date is populated before open date for any record.



```
#2.2 2nd - checking for any invalid records where open date is after closed date

df_merged.loc[(df_merged.rvDtOpnA01 > df_merged.rvDtClsdA01) | (df_merged.rvDtOpnA02 > df_merged.rvDtClsdA02) | (df_merged.rvDtOpnA03 > df_merged
    .rvDtClsdA03) | (df_merged.rvDtOpnA04 > df_merged.rvDtClsdA04) | (df_merged.rvDtOpnA05 > df_merged.rvDtClsdA05)]
```

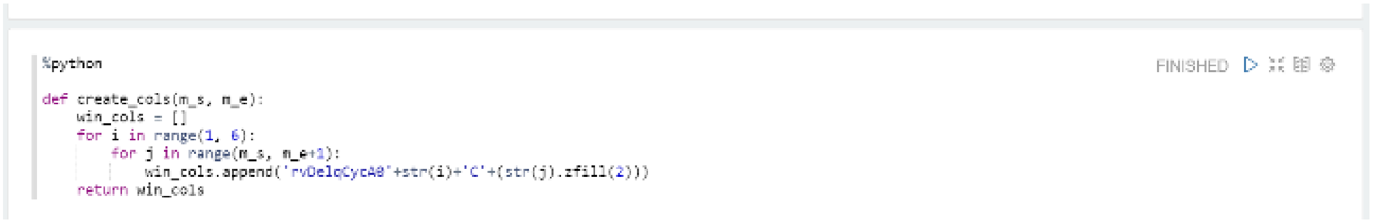
```
   cuCustomerID  cuNide_x  DESC_TIPO_DOCUMENTO_x  FECHA_NACIMIENTO_x  FECHA_DATA_Num_x  cuNumrvAcct  rvDtOpnA01  rvDtOpn
0 rows × 1011 columns
```

Fig. 15 Date check on data

3.5.4 Roll Rate Analysis

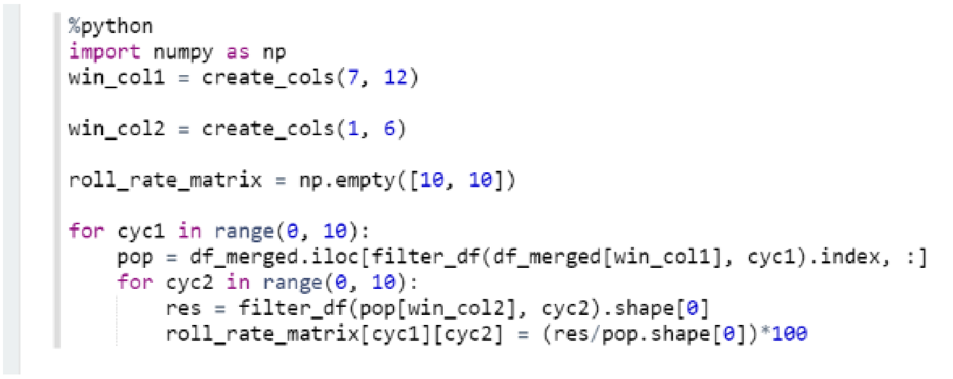
Roll Rate Analysis has a popular application in the credit risk forecasting domain. It is used to generate a definition for the bad customers (or defaulters). Roll Rate Analysis provides us the answer to the question “Who will be categorized as a bad customer” along with a quantitative reasoning.

It can also be defined as the ratio of applicants who will better, remain same, or worsen off with time. It helps us visualize the percentage of accounts that will move from one delinquency bucket to another over an interval. An account can roll either forward in delinquency or backwards. For ex, a forward roll is when an account moves from 30 days past due to 60 days past due. An account at 30 days past due coming out of delinquency by paying the amount is a backward roll.



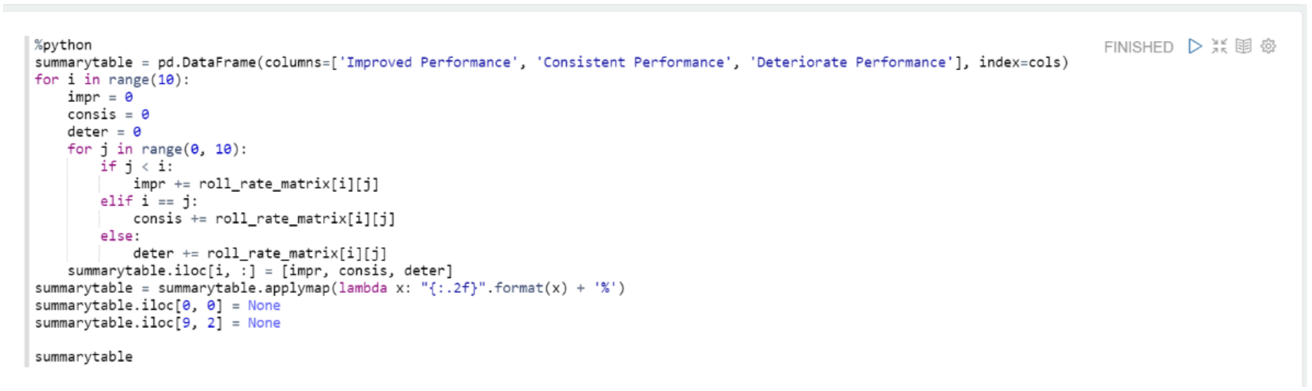
```python
%python

def create_cols(m_s, m_e):
    win_cols = []
    for i in range(1, 6):
        for j in range(m_s, m_e+1):
            win_cols.append('rvDelqCycA0'+str(i)+'C'+(str(j).zfill(2)))
    return win_cols
```

Fig. 16 Roll-Rate Analysis



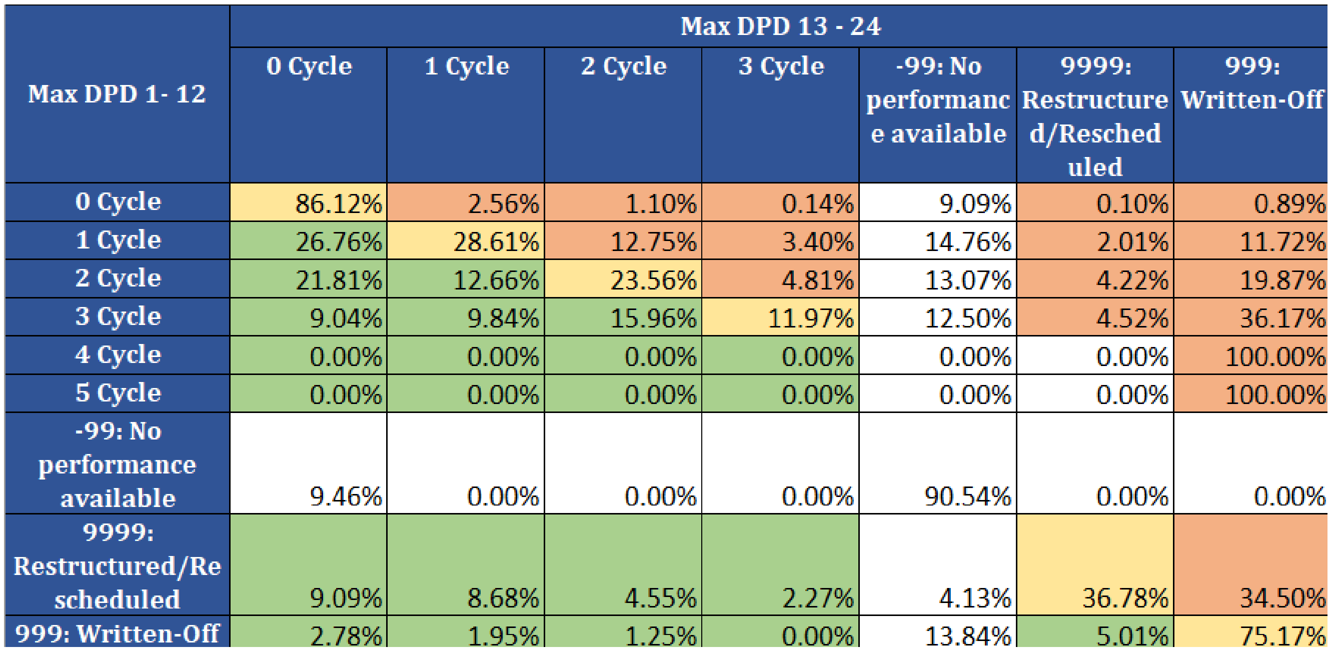
```python
%python
import numpy as np
win_col1 = create_cols(7, 12)

win_col2 = create_cols(1, 6)

roll_rate_matrix = np.empty([10, 10])

for cyc1 in range(0, 10):
    pop = df_merged.iloc[filter_df(df_merged[win_col1], cyc1).index, :]
    for cyc2 in range(0, 10):
        res = filter_df(pop[win_col2], cyc2).shape[0]
        roll_rate_matrix[cyc1][cyc2] = (res/pop.shape[0])*100
```

Fig. 17 Roll-Rate Analysis



```python
%python
summarytable = pd.DataFrame(columns=['Improved Performance', 'Consistent Performance', 'Deteriorate Performance'], index=cols)
for i in range(10):
    impr = 0
    consis = 0
    deter = 0
    for j in range(0, 10):
        if j < i:
            impr += roll_rate_matrix[i][j]
        elif i == j:
            consis += roll_rate_matrix[i][j]
        else:
            deter += roll_rate_matrix[i][j]
    summarytable.iloc[i, :] = [impr, consis, deter]
summarytable = summarytable.applymap(lambda x: "{:.2f}".format(x) + '%')
summarytable.iloc[0, 0] = None
summarytable.iloc[9, 2] = None

summarytable
```

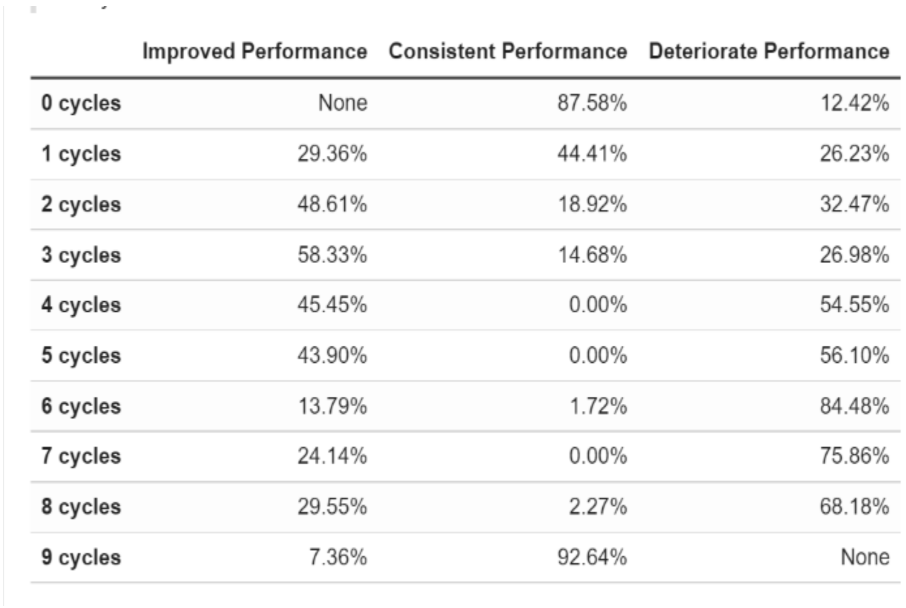
Fig. 18 Roll-Rate Analysis



Max DPD 1- 12	Max DPD 13 - 24						
	0 Cycle	1 Cycle	2 Cycle	3 Cycle	-99: No performance available	9999: Restructured/Rescheduled	999: Written-Off
0 Cycle	86.12%	2.56%	1.10%	0.14%	9.09%	0.10%	0.89%
1 Cycle	26.76%	28.61%	12.75%	3.40%	14.76%	2.01%	11.72%
2 Cycle	21.81%	12.66%	23.56%	4.81%	13.07%	4.22%	19.87%
3 Cycle	9.04%	9.84%	15.96%	11.97%	12.50%	4.52%	36.17%
4 Cycle	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	100.00%
5 Cycle	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	100.00%
-99: No performance available	9.46%	0.00%	0.00%	0.00%	90.54%	0.00%	0.00%
9999: Restructured/Rescheduled	9.09%	8.68%	4.55%	2.27%	4.13%	36.78%	34.50%
999: Written-Off	2.78%	1.95%	1.25%	0.00%	13.84%	5.01%	75.17%

Fig. 19 Roll Rate Matrix

It can be observed from the below roll rate analysis table that once an account hits 4 Cycle, there is a significant increase and shift in the percentage of deteriorated performance. Hence, from the above roll rate analysis conducted, bad definition of accounts hitting 4 cycle or more is chosen.



	Improved Performance	Consistent Performance	Deteriorate Performance
0 cycles	None	87.58%	12.42%
1 cycles	29.36%	44.41%	26.23%
2 cycles	48.61%	18.92%	32.47%
3 cycles	58.33%	14.68%	26.98%
4 cycles	45.45%	0.00%	54.55%
5 cycles	43.90%	0.00%	56.10%
6 cycles	13.79%	1.72%	84.48%
7 cycles	24.14%	0.00%	75.86%
8 cycles	29.55%	2.27%	68.18%
9 cycles	7.36%	92.64%	None

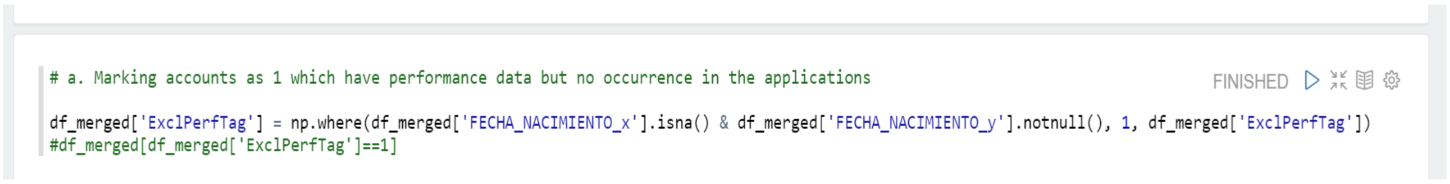
Fig. 20 Performance matrix

3.5.5 Performance Definition and Summary Counts

At the beginning of a scorecard development project, a performance definition must be chosen. A performance definition is a rule which classifies accounts into the categories of exclusions, rejects, good, bad or non-taken ups, based on account performance during the performance window. These definitions act like a sieve on the account records and classify each record into one of the performance categories. Each record begins at the top of the list and is tested against each condition in the identification logic column sequentially. When a record satisfies one of the conditions, it receives that performance category identification and is no longer tested.
Accounts are classified under the following performance categories:

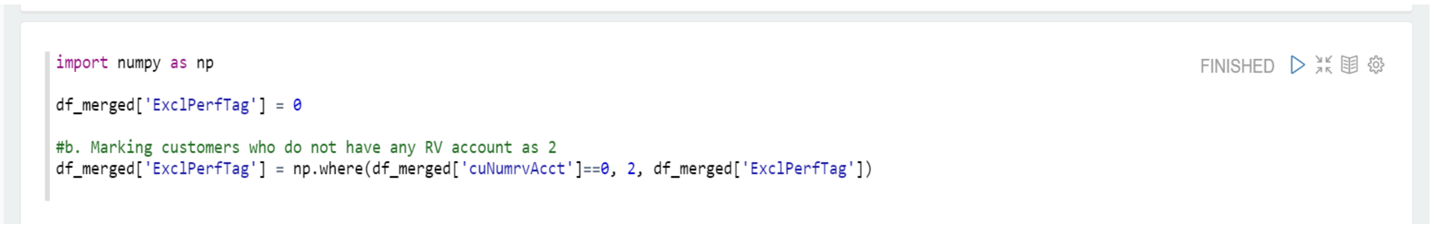
- Exclusions
- Rejects
- Bads
- Indeterminates
- Insufficient
- Goods

Exclusions – Exclusions refer to a set of account types which are removed from consideration or are treated in a special manner during scorecard development. These accounts are not included in the development.
Rejects –These are applicants that were rejected by the current screening process. The performance of these accounts is inferred during inference.
Bads –These are accounts that a bank would decline if it was known how they would perform. They represent the accounts with the highest risk and have reached a bad status in the performance window, starting from the observation point.
Indeterminates –These are accounts that could be either good or bad and are hard to evaluate in terms of risk.
Insufficient –These are accounts that do not have delinquency but cannot be classified as “good” because they have not yet demonstrated enough experience to be considered a good risk.
Goods –These are accounts you would like to have again. They represent the lowest risk.



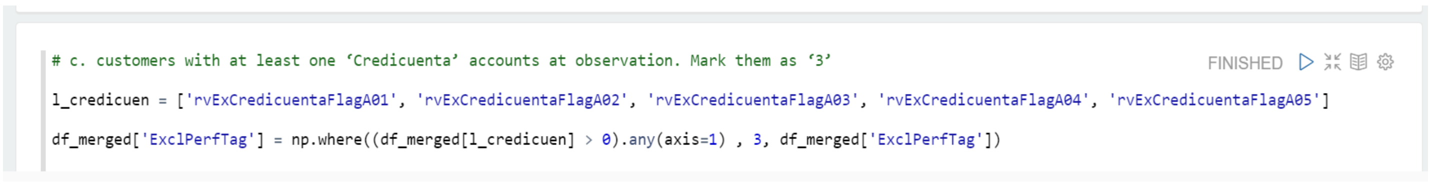
```
# a. Marking accounts as 1 which have performance data but no occurrence in the applications

df_merged['ExclPerfTag'] = np.where(df_merged['FECHA_NACIMIENTO_x'].isna() & df_merged['FECHA_NACIMIENTO_y'].notnull(), 1, df_merged['ExclPerfTag'])
#df_merged[df_merged['ExclPerfTag']==1]
```

Fig. 21 Marking accounts as 1 acc to performance definition



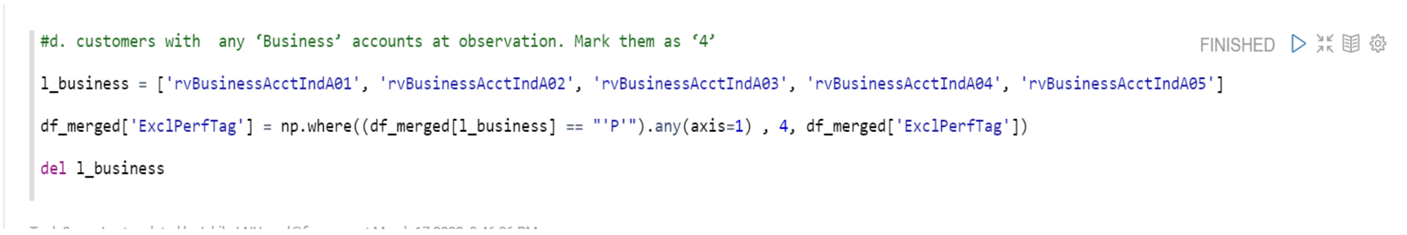
```
import numpy as np

df_merged['ExclPerfTag'] = 0

#b. Marking customers who do not have any RV account as 2
df_merged['ExclPerfTag'] = np.where(df_merged['cuNumrvAcct']==0, 2, df_merged['ExclPerfTag'])
```

Fig. 22 Marking accounts as 2 acc to performance definition



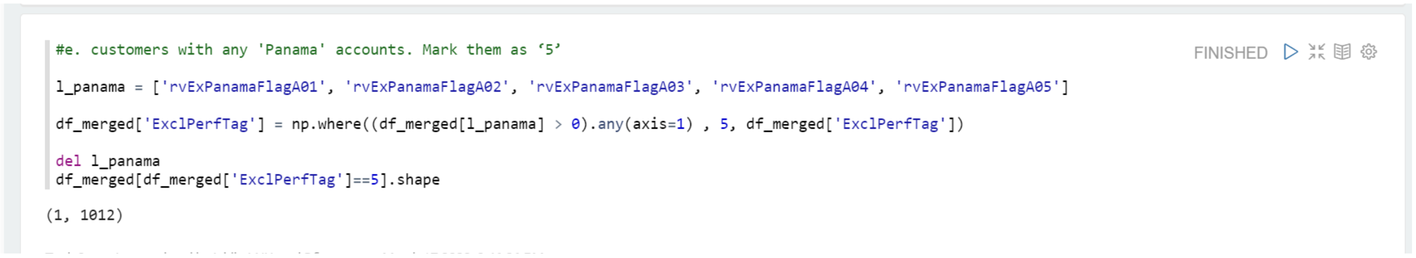
```
# c. customers with at least one ‘Credicuenta’ accounts at observation. Mark them as ‘3’

l_credicuen = ['rvExCredicuentaFlagA01', 'rvExCredicuentaFlagA02', 'rvExCredicuentaFlagA03', 'rvExCredicuentaFlagA04', 'rvExCredicuentaFlagA05']

df_merged['ExclPerfTag'] = np.where((df_merged[l_credicuen] > 0).any(axis=1) , 3, df_merged['ExclPerfTag'])
```

Fig. 23 Marking accounts as 3 acc to performance definition



```
#d. customers with  any ‘Business’ accounts at observation. Mark them as ‘4’

l_business = ['rvBusinessAcctIndA01', 'rvBusinessAcctIndA02', 'rvBusinessAcctIndA03', 'rvBusinessAcctIndA04', 'rvBusinessAcctIndA05']

df_merged['ExclPerfTag'] = np.where((df_merged[l_business] == "'P'").any(axis=1) , 4, df_merged['ExclPerfTag'])

del l_business
```

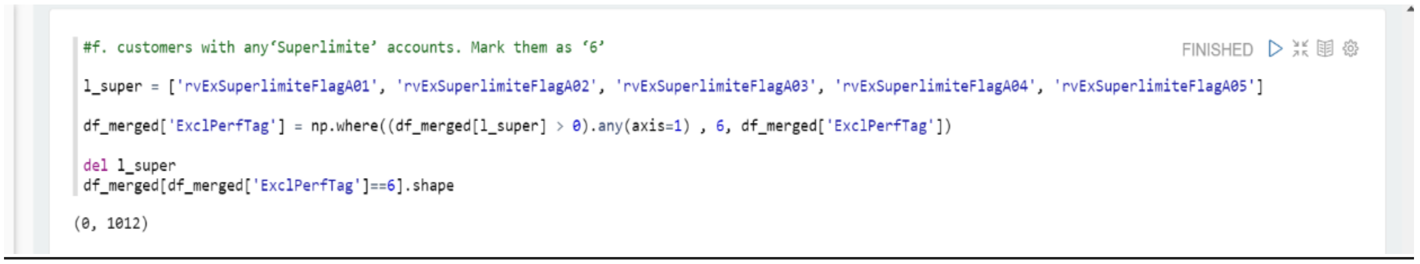
Fig. 24 Marking accounts as 4 acc to performance definition



```
#e. customers with any 'Panama' accounts. Mark them as ‘5’

l_panama = ['rvExPanamaFlagA01', 'rvExPanamaFlagA02', 'rvExPanamaFlagA03', 'rvExPanamaFlagA04', 'rvExPanamaFlagA05']

df_merged['ExclPerfTag'] = np.where((df_merged[l_panama] > 0).any(axis=1) , 5, df_merged['ExclPerfTag'])

del l_panama
df_merged[df_merged['ExclPerfTag']==5].shape
```

```
(1, 1012)
```

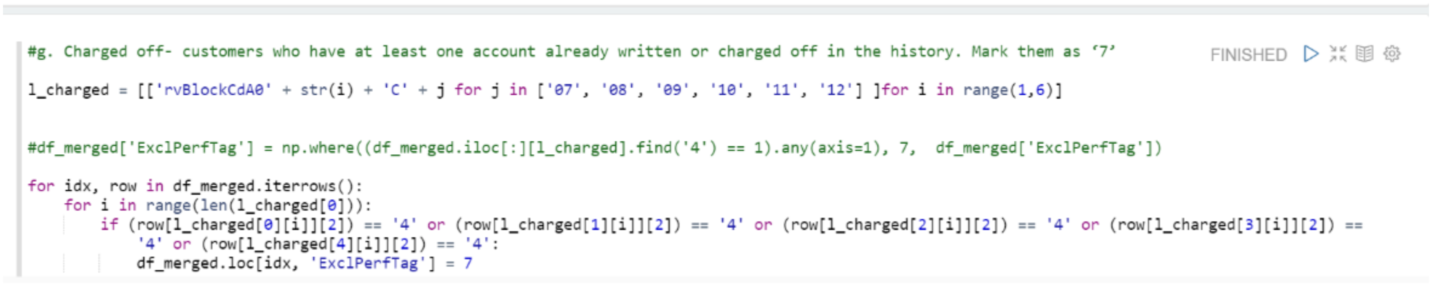
Fig. 25 Marking accounts as 5 acc to performance definition



```
#f. customers with any‘Superlimite’ accounts. Mark them as ‘6’

l_super = ['rvExSuperlimiteFlagA01', 'rvExSuperlimiteFlagA02', 'rvExSuperlimiteFlagA03', 'rvExSuperlimiteFlagA04', 'rvExSuperlimiteFlagA05']

df_merged['ExclPerfTag'] = np.where((df_merged[l_super] > 0).any(axis=1) , 6, df_merged['ExclPerfTag'])

del l_super
df_merged[df_merged['ExclPerfTag']==6].shape
```

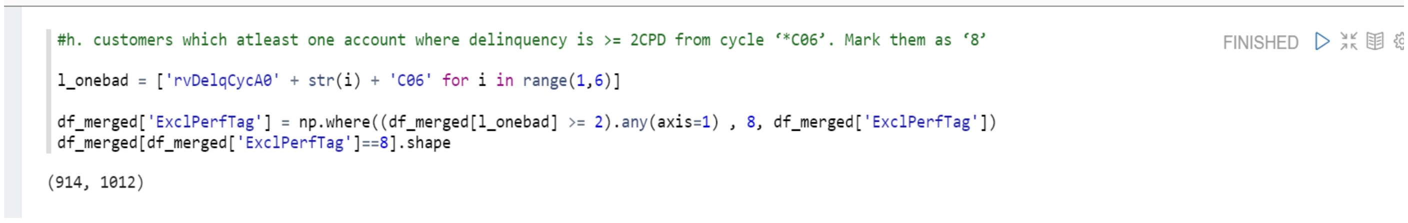
```
(0, 1012)
```

Fig. 26 Marking accounts as 6 acc to performance definition



```
#g. Charged off- customers who have at least one account already written or charged off in the history. Mark them as ‘7’

l_charged = [['rvBlockCdA0' + str(i) + 'C' + j for j in ['07', '08', '09', '10', '11', '12'] ]for i in range(1,6)]

#df_merged['ExclPerfTag'] = np.where((df_merged.iloc[:][l_charged].find('4') == 1).any(axis=1), 7,  df_merged['ExclPerfTag'])

for idx, row in df_merged.iterrows():
    for i in range(len(l_charged[0])):
        if (row[l_charged[0][i]][2]) == '4' or (row[l_charged[1][i]][2]) == '4' or (row[l_charged[2][i]][2]) == '4' or (row[l_charged[3][i]][2]) ==
        '4' or (row[l_charged[4][i]][2]) == '4':
            df_merged.loc[idx, 'ExclPerfTag'] = 7
```

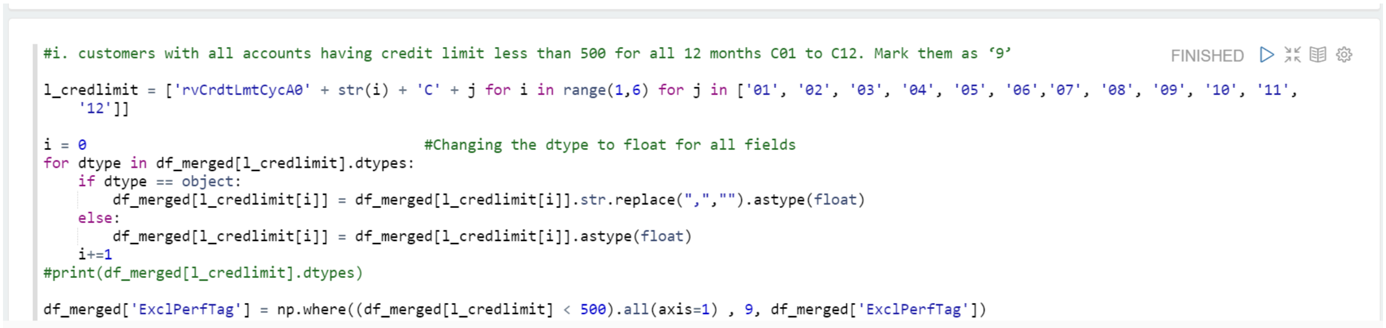
Fig. 27 Marking accounts as 7 acc to performance definition



```
#h. customers which atleast one account where delinquency is >= 2CPD from cycle ‘*C06’. Mark them as ‘8’

l_onebad = ['rvDelqCycA0' + str(i) + 'C06' for i in range(1,6)]

df_merged['ExclPerfTag'] = np.where((df_merged[l_onebad] >= 2).any(axis=1) , 8, df_merged['ExclPerfTag'])
df_merged[df_merged['ExclPerfTag']==8].shape
```

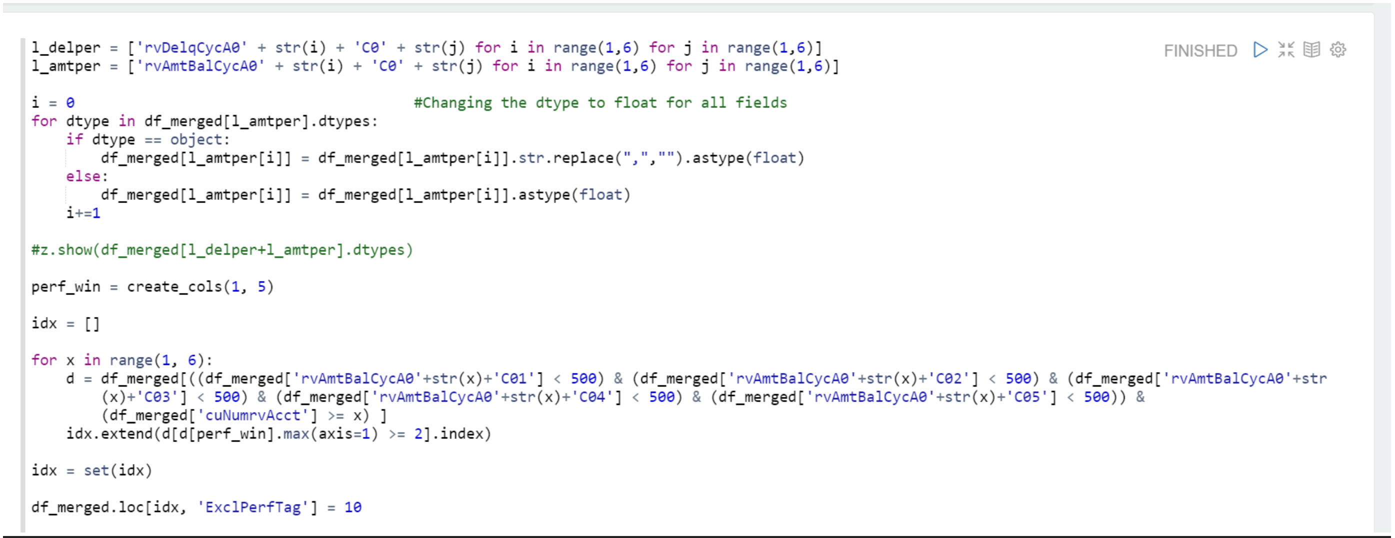
```
(914, 1012)
```

Fig. 28 Marking accounts as 8 acc to performance definition



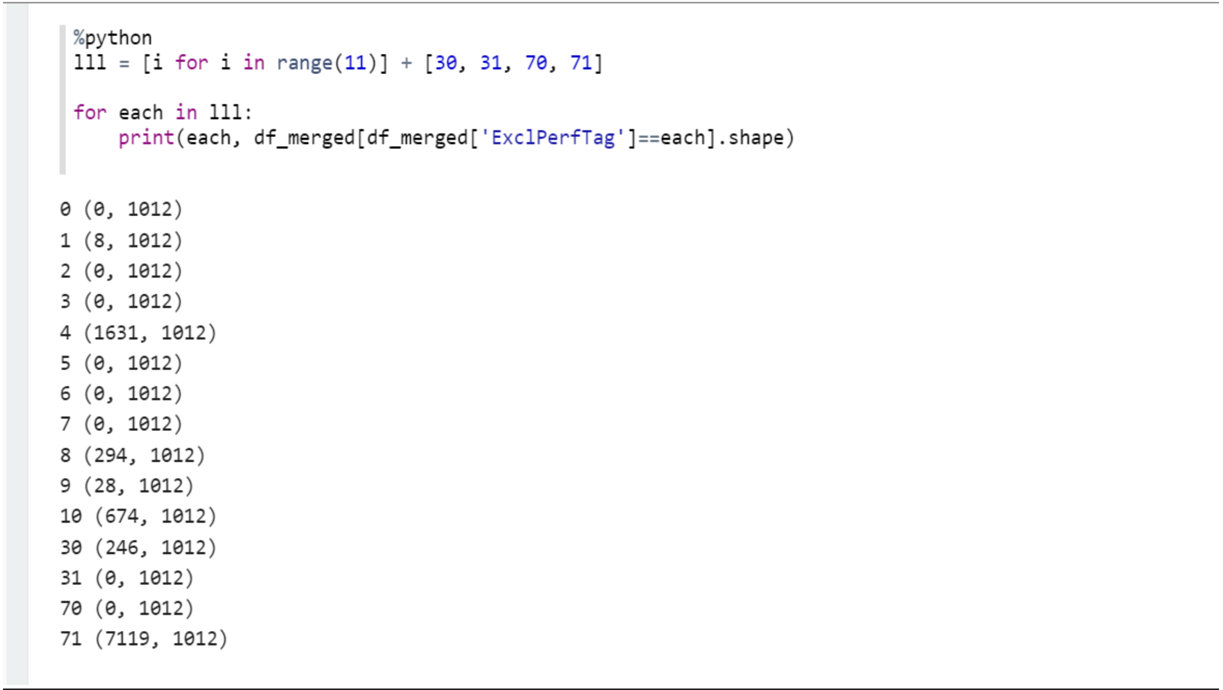
```python
#i. customers with all accounts having credit limit less than 500 for all 12 months C01 to C12. Mark them as ‘9’

l_credlimit = ['rvCrdtLmtCycA0' + str(i) + 'C' + j for i in range(1,6) for j in ['01', '02', '03', '04', '05', '06','07', '08', '09', '10', '11',
    '12']]

i = 0                                       #Changing the dtype to float for all fields
for dtype in df_merged[l_credlimit].dtypes:
    if dtype == object:
        df_merged[l_credlimit[i]] = df_merged[l_credlimit[i]].str.replace(",","").astype(float)
    else:
        df_merged[l_credlimit[i]] = df_merged[l_credlimit[i]].astype(float)
    i+=1
#print(df_merged[l_credlimit].dtypes)

df_merged['ExclPerfTag'] = np.where((df_merged[l_credlimit] < 500).all(axis=1) , 9, df_merged['ExclPerfTag'])
```

Fig. 29 Marking accounts as 9 acc to performance definition



```python
l_delper = ['rvDelqCycA0' + str(i) + 'C0' + str(j) for i in range(1,6) for j in range(1,6)]
l_amtper = ['rvAmtBalCycA0' + str(i) + 'C0' + str(j) for i in range(1,6) for j in range(1,6)]

i = 0                                       #Changing the dtype to float for all fields
for dtype in df_merged[l_amtper].dtypes:
    if dtype == object:
        df_merged[l_amtper[i]] = df_merged[l_amtper[i]].str.replace(",","").astype(float)
    else:
        df_merged[l_amtper[i]] = df_merged[l_amtper[i]].astype(float)
    i+=1

#z.show(df_merged[l_delper+l_amtper].dtypes)

perf_win = create_cols(1, 5)

idx = []

for x in range(1, 6):
    d = df_merged[((df_merged['rvAmtBalCycA0'+str(x)+'C01'] < 500) & (df_merged['rvAmtBalCycA0'+str(x)+'C02'] < 500) & (df_merged['rvAmtBalCycA0'+str
        (x)+'C03'] < 500) & (df_merged['rvAmtBalCycA0'+str(x)+'C04'] < 500) & (df_merged['rvAmtBalCycA0'+str(x)+'C05'] < 500)) &
        (df_merged['cuNumrvAcct'] >= x) ]
    idx.extend(d[d[perf_win].max(axis=1) >= 2].index)

idx = set(idx)

df_merged.loc[idx, 'ExclPerfTag'] = 10
```

Fig. 30 Marking accounts as 10 acc to performance definition



```
%python
lll = [i for i in range(11)] + [30, 31, 70, 71]

for each in lll:
    print(each, df_merged[df_merged['ExclPerfTag']==each].shape)
```

```
0 (0, 1012)
1 (8, 1012)
2 (0, 1012)
3 (0, 1012)
4 (1631, 1012)
5 (0, 1012)
6 (0, 1012)
7 (0, 1012)
8 (294, 1012)
9 (28, 1012)
10 (674, 1012)
30 (246, 1012)
31 (0, 1012)
70 (0, 1012)
71 (7119, 1012)
```

Fig. 31 Summary Counts

3.5.4 Scorecard Development

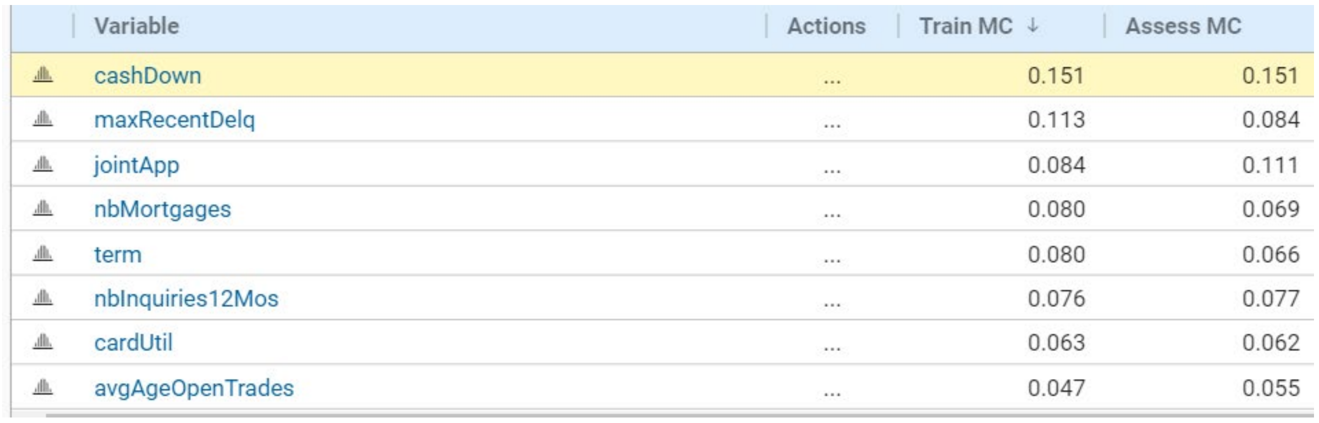
When developing the models, it is important to leverage a modelling algorithm that not only derives high predictive power but is also able to provide intuitive visibility into the relationship of a predictive variable with bad rate over time. Multiple modelling algorithms are leveraged for solving binary choice or dichotomous outcome problems. An algorithm is used in the background that is capable of capturing the non-linear relationship between predictive variables and the outcome. The entire value range of a predictive variable is partitioned into mutually exclusive and collectively exhaustive set of “bins”. The score is obtained as the sum of the score weight for each bin of a predictive variable. A unique feature of the algorithm is the capability to constrain the score formulas to exhibit desirable patterns or trends that adhere to intuitive understanding of the variable’s relationship with risk. The algorithm leverages divergence as the objective function to obtain the score weight for each “bin”. Divergence is a measure of separation between the score distribution of the two types of outcomes (in this case bads and goods); a higher separation indicates higher predictive power and in turn higher divergence.

In order to maximize divergence, it is necessary to utilize a search algorithm to obtain a set of optimal score weights. The methodology leverages quadratic and non-linear programming for finding the optimal solution that would yield maximum divergence. Another key component of the algorithm involves calculation of the Marginal Contribution (MC) of each predictive variable. Marginal contribution estimates a variable's unique contribution to the total predictive power of the model.

Following are the steps carried out in the sequence for scorecard development:

- Initial Shortlisting of Characteristics
- Final Selection of Characteristics
- Treatment of Multi Collinearity
- Scaling of Scores

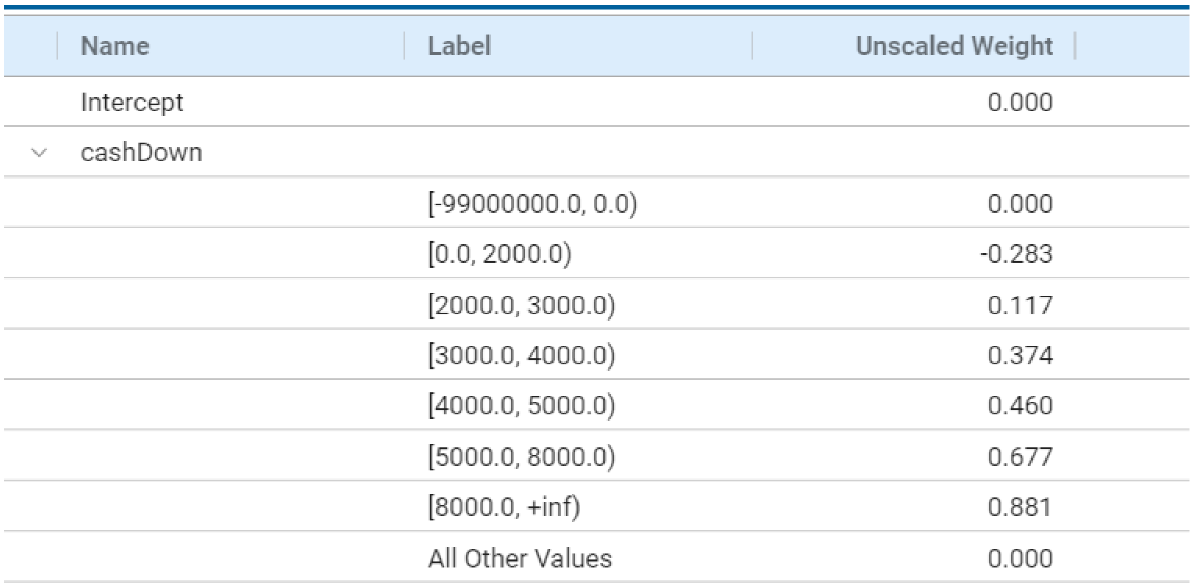
Following image shows the predictive characteristics used to build the scorecard.



Variable	Actions	Train MC ↓	Assess MC
cashDown	...	0.151	0.151
maxRecentDelq	...	0.113	0.084
jointApp	...	0.084	0.111
nbMortgages	...	0.080	0.069
term	...	0.080	0.066
nbInquiries12Mos	...	0.076	0.077
cardUtil	...	0.063	0.062
avgAgeOpenTrades	...	0.047	0.055

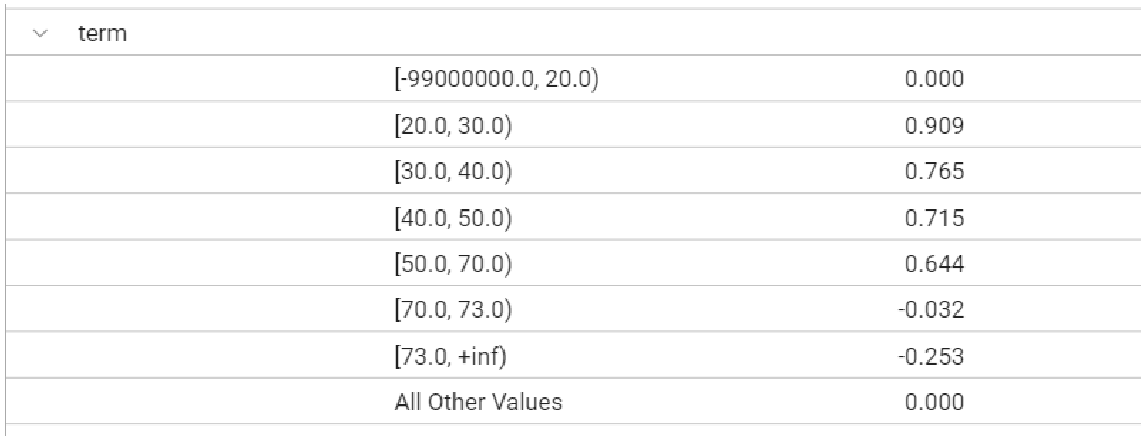
Fig. 32 Characteristics Used

Below images show the weights generated for different characteristics.



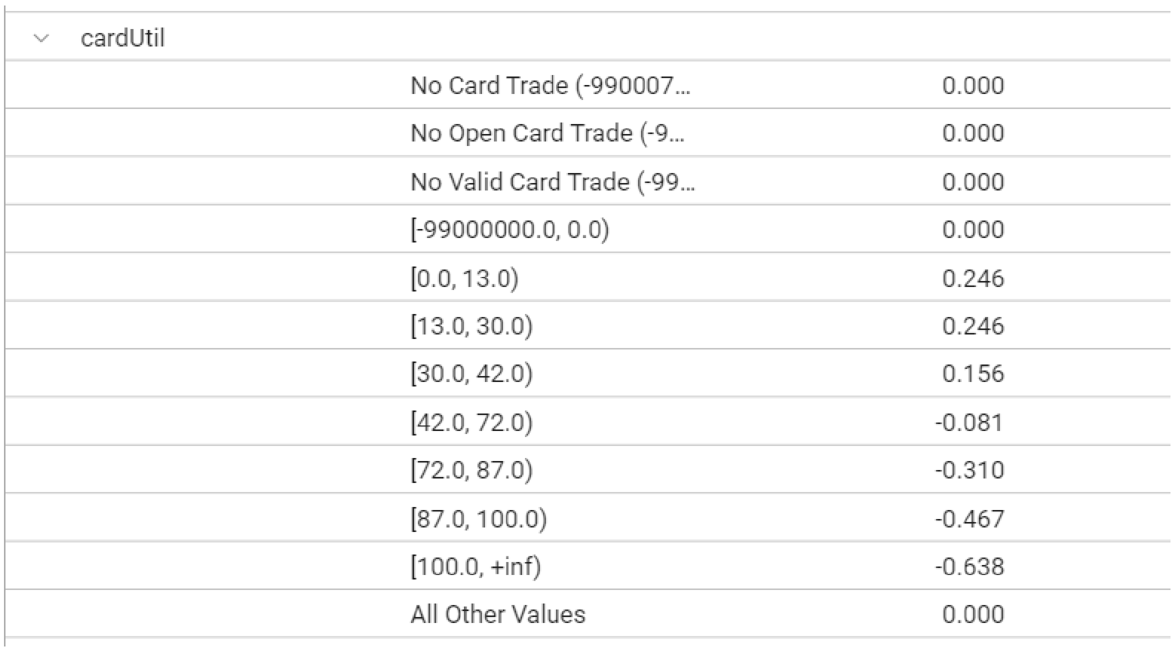
Name	Label	Unscaled Weight
Intercept		0.000
cashDown		
	[-99000000.0, 0.0)	0.000
	[0.0, 2000.0)	-0.283
	[2000.0, 3000.0)	0.117
	[3000.0, 4000.0)	0.374
	[4000.0, 5000.0)	0.460
	[5000.0, 8000.0)	0.677
	[8000.0, +inf)	0.881
	All Other Values	0.000

Fig. 33 Scores for characteristic 1



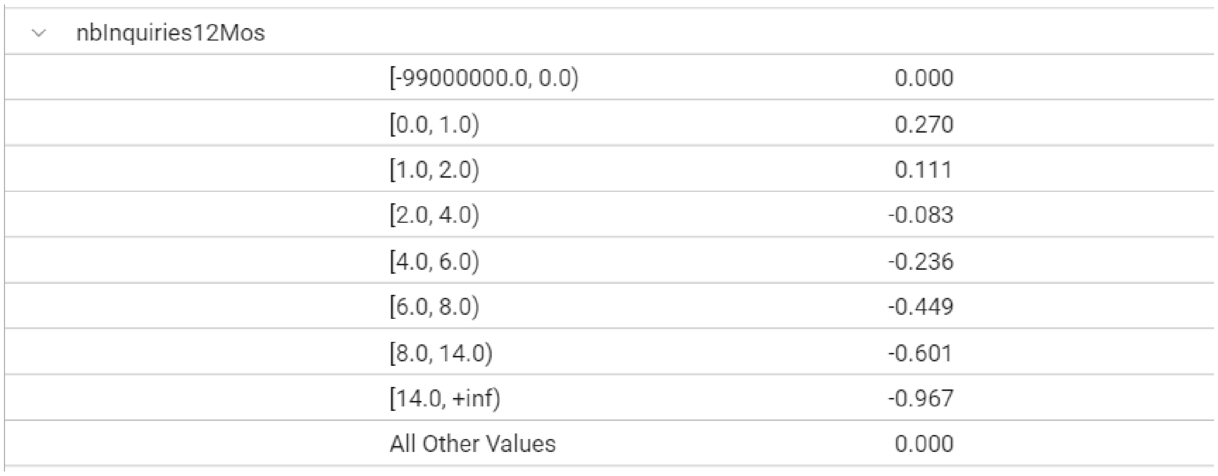
term		
	[-99000000.0, 20.0)	0.000
	[20.0, 30.0)	0.909
	[30.0, 40.0)	0.765
	[40.0, 50.0)	0.715
	[50.0, 70.0)	0.644
	[70.0, 73.0)	-0.032
	[73.0, +inf)	-0.253
	All Other Values	0.000

Fig. 34 Scores for characteristic 2



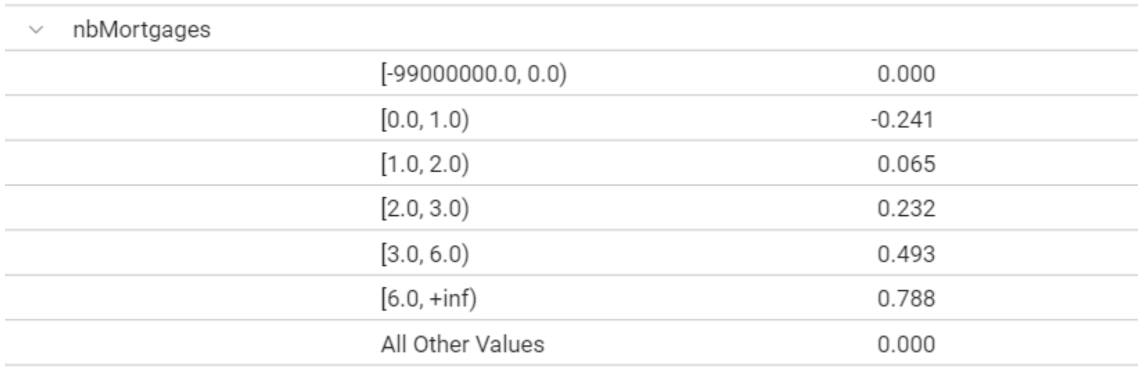
cardUtil	
No Card Trade (-990007…	0.000
No Open Card Trade (-9…	0.000
No Valid Card Trade (-99…	0.000
[-99000000.0, 0.0)	0.000
[0.0, 13.0)	0.246
[13.0, 30.0)	0.246
[30.0, 42.0)	0.156
[42.0, 72.0)	-0.081
[72.0, 87.0)	-0.310
[87.0, 100.0)	-0.467
[100.0, +inf)	-0.638
All Other Values	0.000

Fig. 35 Scores for characteristic 3



nbInquiries12Mos	
[-99000000.0, 0.0)	0.000
[0.0, 1.0)	0.270
[1.0, 2.0)	0.111
[2.0, 4.0)	-0.083
[4.0, 6.0)	-0.236
[6.0, 8.0)	-0.449
[8.0, 14.0)	-0.601
[14.0, +inf)	-0.967
All Other Values	0.000

Fig. 36 Scores for characteristic 4



⌄ nbMortgages		
	[-99000000.0, 0.0)	0.000
	[0.0, 1.0)	-0.241
	[1.0, 2.0)	0.065
	[2.0, 3.0)	0.232
	[3.0, 6.0)	0.493
	[6.0, +inf)	0.788
	All Other Values	0.000

Fig. 37 Scores for characteristic 5

CHAPTER-4

PERFORMANCE ANALYSIS

4.1 Results Achieved

The model's performance was evaluated using statistics like the Divergence, K-S, Gini, and trade-off curve. A description of the model statistics is given in below sub-sections

Divergence

There are two elements to identify how well the scorecard rank orders risk.
The difference between the mean score of the goods and the mean score of the bads.
The area of overlap between the goods and the bads.
Divergence summarizes these two elements. It is a measure of separation between the goods and bads distribution as shown below.

$$\text{Divergence} = \frac{(\mu_{Good} - \mu_{Bad})^2}{\frac{1}{2}\left(\sigma^2_{Good} + \sigma^2_{Bad}\right)}$$

Where μ_{Good} = mean score of goods,
μ_{Bad} = mean score of bads
σ_{Good} = standard deviation of goods
σ_{Bad} = standard deviation of bads

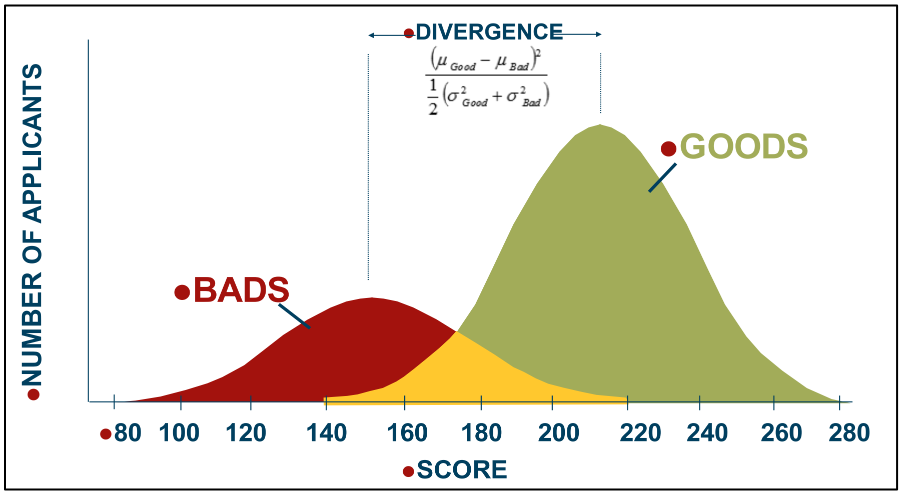


Fig. 38 Divergence

The higher the separation, i.e. the lesser the overlap between the goods and bads distribution, the better the scorecard.

K-S Statistic

K-S is a non-parametric measure. It determines whether two underlying distributions differ by comparing their cumulative distributions. It is defined as the maximum vertical distance between the two cumulative distributions.

$$K\text{-}S = \max_s \left| F_G(s) - F_B(s) \right|$$

Where $F_G(s)$ and $F_B(s)$ are the cumulative score distribution functions of goods and bads respectively.

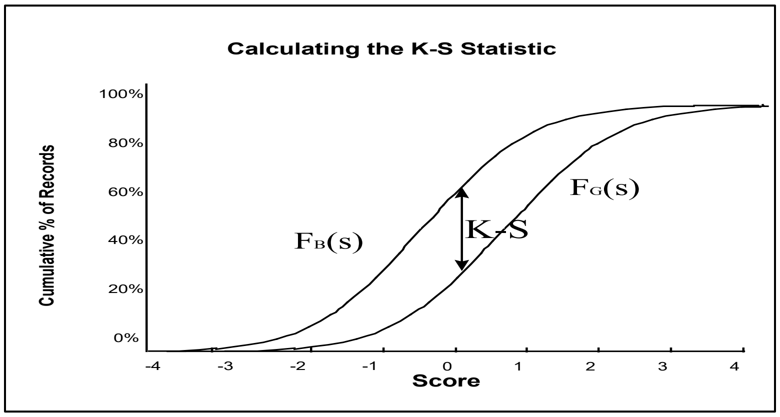


Fig. 39 K-S Statistic

Gini Coefficient

When we plot cumulative score distribution from the goods against the cumulative score distribution from the bads, we get what is known as the Trade-off curve. For each score distribution, the plotted trade-off curve shows how many Bad accounts are captured by the score at the low end of the distribution for a given number of good records. In general, the farther the score trade-off curve bows to the left and up, the better the scoring model. The diagonal 45° line is the “line of random selection” and is the equivalent of a scoring system that does not work at all. The Gini Coefficient is defined as the ratio of area between the Trade-Off curve and the diagonal line to the area above the diagonal line.

Gini Coefficient = (Trade-Off Curve Area – 0.5)/0.5

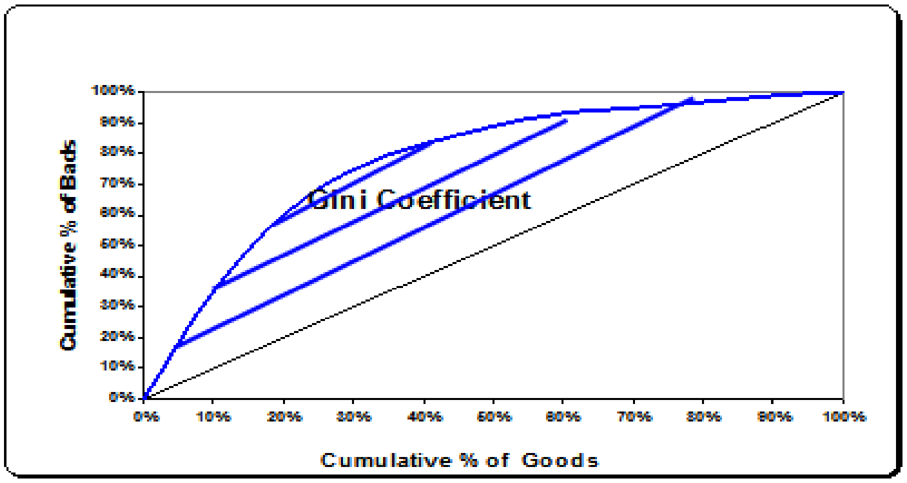


Fig. 40 Gini Coefficient

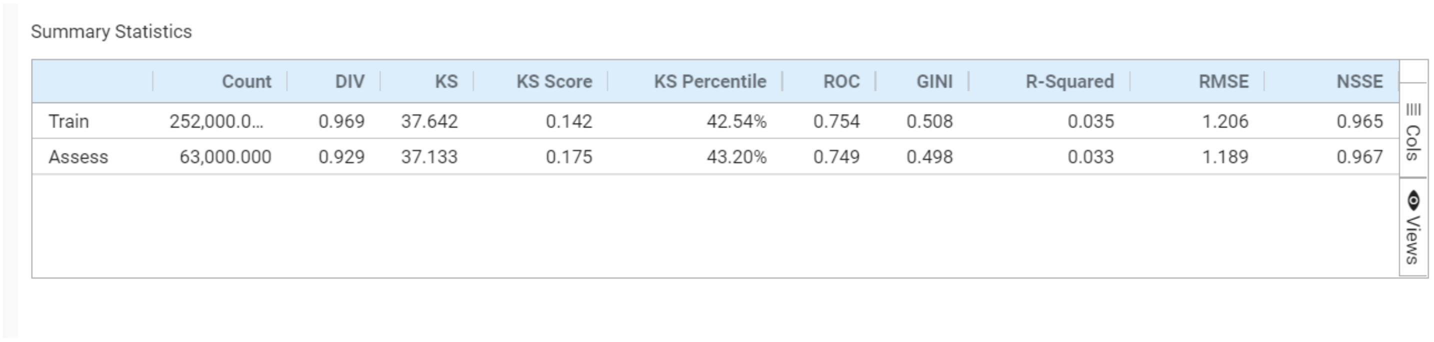
ROC Area (ROC)

Similar to calculating the Gini Coefficient, the ROC is determined from the cumulative score distribution from the goods against the cumulative score distribution from the bads, known as the Trade-off curve. However, the ROC is the unscaled area under the Trade-Off Curve, instead of the value scaled by subtracting and dividing by 0.5

ROC = Trade-Off Curve Area = 0.5*(Gini Coefficient) + 0.5

4.2 Comparing the Results

The table below shows and compares the accuracy metrics of the built scorecard model.



Summary Statistics

	Count	DIV	KS	KS Score	KS Percentile	ROC	GINI	R-Squared	RMSE	NSSE
Train	252,000.0...	0.969	37.642	0.142	42.54%	0.754	0.508	0.035	1.206	0.965
Assess	63,000.000	0.929	37.133	0.175	43.20%	0.749	0.498	0.033	1.189	0.967

Fig. 41 Summary Statistics

Images below show the plots of the K-S and ROC for the model.

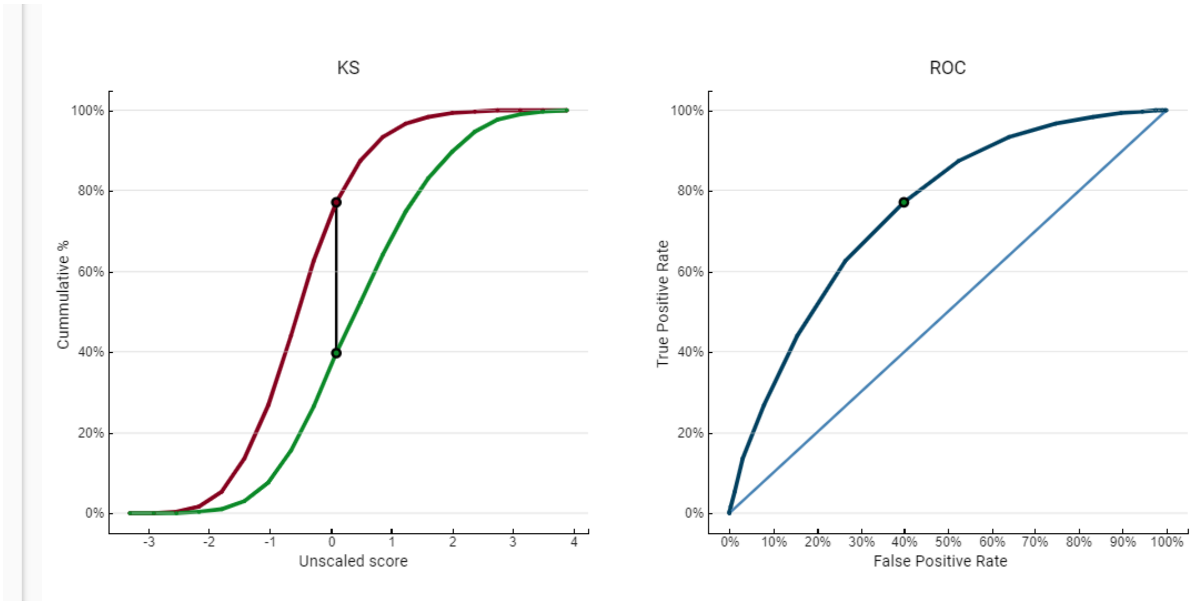


Fig. 42 KS and ROC plots

CHAPTER-5

CONCLUSIONS

5.1 Conclusion

Viewing from a correct examination of positive points and constraints on the component, it may be thoroughly derived that the result is a greatly efficient component. This application system is implementing correctly and satisfying all of the banker's requirements. This component may also be effortlessly integrated with many other systems.

This scorecard provides the bank with an opportunity to manage and approve or reject their applications coming in with minimal risk possible.

The predictive modelling technique used to generate the scorecard which ranks orders applicants on the basis of their riskiness is proved to be an efficient and accurate approach. It resulted in a divergence of 0.969 for the train dataset and a divergence of 0.929 for the test dataset along with a Gini of 0.508 for the train dataset.

5.2 Future Scope

There can be a shift in the population in the near future, when the population coming in no longer matches the population on which the model is built. In that case, the model will need to be modified or a new model might be needed. Along with the changing economy and changing policies of the bank, the model will need to be maintained and modified accordingly.

For the future scope, different scorecards for different clusters of population could also be built in order to assess the risk better.

REFERENCES

1. Sergey Krylov. Balanced Scorecard Concept Development: Analysis
https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2834249

2. Alejandro Correa, Catherine Nieto, Andrés González, Darwin Amezquita. Constructing a credit risk scorecard using Predictive Clusters
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.421.3236&rep=rep1&type=pdf

3. Rory P. Bunker, M. Asif Naeem, Wenjun Zhang. Improving a Credit Scoring Model by Incorporating Bank Statement Derived Features
https://arxiv.org/ftp/arxiv/papers/1611/1611.00252.pdf