Credit Score Scaling and Data Drift

The problem aims to show the maximum score and the minimum score along with the Score card after we enter the PDO, LOC and Want Odds. After adjusting the values, one can set the desired score range and get the corresponding score card and download it. Now when a new individual comes, with the help of this score card we can generate his credit score. If this score lying in the range is more than a cutoff the person gets the loan. Also view the PSI table to check whether there is a data drift.

Credit score:

A credit score is a numeric representation of an individual's creditworthiness or the ability to repay the debt on time. It is the result of a statistical model, based on information about the borrower. Credit scores usually lies in a range which may vary on the company. Higher the score the individual gets, the chances of getting the loan increases. They are also used to determine the interest rate and the credit limit one receives.

About the dataset

- Customer: Customer ID
- checking_balance: Status of existing checking account
- months_loan_duration: Duration in month
- credit_history: Credit history
- purpose: Purpose of loan
- amount: Credit amount
- savings_balance: Savings accounts/bonds
- employment_length: Present employment since
- installment_rate: installation rate in percentage of disposable income
- personal_status: Personal status and sex
- other_debtors: Other debtors
- residence_history: Present residence since
- property: Property
- age: Age in years
- instalment_plan: Other insatllment plans
- housing: Housing
- existing_credits: Number of existing credits in this bank
- job: Job
- dependents: Number of people being liable to provide maintenance for
- telephone: Telephone
- foreign worker: Foreign worker
- default: Whether default or not

The dataset had 22 features including the loan status and various attributes related to both borrowers and their payment behavior like checking_balance, credit_history, amount, employment_length, property, age etc.

Initial data exploration revealed the following

- There were 9 features that had continuous values and 13 features which had discrete values.
- Our target variable was the column named default.
- There were no missing values.

Feature engineering

There is huge amount of data. However not all the features available to us are useful. Features are selected based on the predictive strength of the feature. We can quantify the predictive power of a feature using the concept of Information Value. It measures the strength/ ability of the independent variables to separate two categorical (dependent) variables.

Logistic regression requires all features to be numerical. So, the categorical variables should be transformed using one hot encoding or other techniques. WOE is another such method. Under each feature for every category, we find the WOE.

This concept is used in the preprocessing step. We use this to calculate the score.

We calculate WOE for each unique value of a categorical variable and transform a continuous variable into discrete bins to calculate the WOE values for those bins.

Benefits of using WOE transformation are

1. It can handle missing values as these can be binned together.
2. Outliers can be handled. The value that feeds into the model fitting is the WOE transformed value and not the raw value.
3. It also handles categorical value so no need for dummy variables.
4. It helps to build a strict linear relationship with log-odds used in logistic regression.

Dummy example to explain WOE and IV

Feature – X (values)	Number of events	Number of non-events	Percentage events	Percentage non-events	WOE	IV
A	90	2400	90/490 = 0.184	2400/9510 = 0.25	Ln(0.184/0.25)=-0.3065	0.02
B	130	1300	130/490 = 0.265	1300/9510 = 0.14	Ln(0.265/0.137)=0.659	0.175
C	80	3500	80/490 = 0.16	3500/9510 =0.37	Ln(0.16/0.37)=0.838	0.176
D	100	1210	100/490 = 0.2	1210/9510 =0.18	Ln(0.2/0.18) = 0.105	0.002
E	90	1100	90/490 = 0.184	1100/9510 = 0.12	Ln(0.184/0.12)=0.427	0.026
Sum	490	9510				0.399

The weight of evidence tells the predictive power of a single feature concerning its independent feature. If any of the categories/bins of a feature has a large proportion of events compared to the proportion of non-events, we will get a high value of Woe which in turn says that that class of the feature separates the events from non-events.

For example, consider category C of the feature X in the above example, the proportion of events (0.16) is very small compared to the proportion of non-events (0.37). This implies that if the value of the feature X is C, it is more likely that the target value will be 0 (non-event). The Woe value only tells us how confident we are that the feature will help us predict the probability of an event correctly.

Therefore

WoE: $$[\ ln\left(\frac{\text{Distr Good}}{\text{Distr Bad}}\right)\ ] \times 100.$$

IV: $$\sum_{i=1}^{n} (\text{Distr Good}_i - \text{Distr Bad}_i) * ln\left(\frac{\text{Distr Good}_i}{\text{Distr Bad}_i}\right)$$

Feature Selection using Information Value

It measures the predictive power of independent variables which is useful in feature selection.

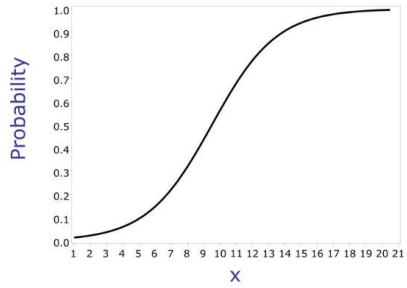
Information Value	Variable Predictiveness
Less than 0.02	Not useful for prediction
0.02 to 0.1	Weak predictive Power
0.1 to 0.3	Medium predictive Power
0.3 to 0.5	Strong predictive Power
>0.5	Suspicious Predictive Power

Model Fitting & Interpreting Results

Now we fit a logistic regression model using our newly transformed WOE of the training dataset. When scaling the model into a scorecard, we will need both the Logistic Regression coefficients from model fitting as well as the transformed Woe values.

Logistic regression

Linear regression is not a good classication algorithm, that is where logistic regression comes in. Here there are two possible outputs and the independent variables are non collinear. The output is a probability and we consider a threshold to conclude the result. Unlike linear regression it fits a S shaped cuve on the dataset.



Thus the logistic regression model looks like this

$f_{\vec{w},b}(x) = g(\vec{w}.\vec{x} + b) = \frac{1}{1+e^{-(\vec{w}.\vec{x}+b)}}$

Here we don't use the squared error cost function because otherwise we end up in local minimas if we use that. Instead we use the special type of cost function that is derived using maximum likehood and is called the logistic loss function.

$J(\vec{w},b) = -\frac{1}{m}\sum[y^i \log\left(f_{\vec{w},b}(\vec{x}^i)\right) + (1-y^i)\log\left(1-f_{\vec{w},b}(\vec{x}^i)\right)]$

We minimize the cost function using gradient descent and hence we get the coefficient and the intercept.

Coming back to our problem

For each independent variable Xi, its corresponding score is :

Score_i=(βi*WOE_i+ α/n)*Factor+Offset/n

Where,

βi – logistic regression coefficient for the variable Xi.

α – logistic regression intercept

WoE – weight of evidence value for the varieble Xi

N – number of indeendent variable Xi in the model

Factor, offset – known as scalling parameter where

- Factor= PDO /ln (2)
- Offset=Target Score - (factor*ln (Target odds)), Offset is the middle point of the range. Suppose my range is 300-900 then offset is 600

The final total score is the sum of all scores based on the input values of the independent variables.

Total Score = ∑ Score_i

PDO (Points to double the odds)

How many points does the score change if the odds double i.e., from 100:1 to 200:1 to 400:1 and so on.

20 pdo means odds will double with each 20 points.

LOC (line of credit)

A line of credit refers to the maximum amount of credit that a lender or financial institution is willing to extend to a borrower.

oddTODO

Good:bad ratio. Note that in credit risk monitoring odds means good:bad (because we want higher scores to be given to customers having good risk and bad scores to be given to customer having bad risk).

Want Odds

A particular odds ratio of Good and Bad e.g., 10:1 or 50:1

Our work was fully done using R program.

Libraries used

- R shiny
  Shiny is an R library that enables building interactive web applications that can execute R code on the backend.
- DT
  The library (DT) statement loads the DT package which provides functionalities for creating interactive data tables in the shiny application.
- Data.table
  R package provides an enhanced version of data. frame that allows us to do blazing fast data manipulations.

This is how the dataset looked like

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Customer	checking_	months_l	credit_his	purpose	amount	savings_b	employm	installmer	personal_	other_det	residence	property	age	installmer	housing
2	1001	< 0 DM	6	critical	radio/tv	1169	unknown	> 7 yrs	4	single mal	none	4	real estate	67	none	own
3	1002	1 - 200 DM	48	repaid	radio/tv	5951	< 100 DM	1 - 4 yrs	-	-	none	2	real estate	22	none	own
4	1003	unknown	12	critical	education	2096	< 100 DM	4 - 7 yrs	2	single mal	none	-	real estate	49	none	own
5	1004	< 0 DM	42	repaid	furniture	7882	< 100 DM	4 - 7 yrs	-	-	guarantor	4	building s	45	none	for free
6	1005	< 0 DM	24	delayed	car (new)	4870	< 100 DM	1 - 4 yrs	3	single mal	none	4	unknown/	53	none	for free
7	1006	unknown	36	repaid	education	9055	unknown	1 - 4 yrs	2	single mal	none	4	unknown/	35	none	for free
8	1007	unknown	24	repaid	furniture	2835	501 - 1000	> 7 yrs	3	single mal	none	4	building s	53	none	own
9	1008	1 - 200 DM	36	repaid	car (used)	6948	< 100 DM	1 - 4 yrs	2	single mal	none	2	other	35	none	rent
10	1009	unknown	12	repaid	radio/tv	3059	> 1000 DM	4 - 7 yrs	2	divorced r	none	4	real estate	61	none	own
11	1010	1 - 200 DM	30	critical	car (new)	5234	< 100 DM	unemploy	4	married m	none	2	other	28	none	own
12	1011	1 - 200 DM	12	repaid	car (new)	1295	< 100 DM	0 - 1 yrs	3	female	none	1	other	25	none	rent
13	1012	< 0 DM	48	repaid	business	4308	< 100 DM	0 - 1 yrs	3	female	none	4	building s	24	none	rent
14	1013	1 - 200 DM	12	repaid	radio/tv	1567	< 100 DM	1 - 4 yrs	1	female	none	1	other	22	none	own
15	1014	< 0 DM	24	critical	car (new)	1199	< 100 DM	> 7 yrs	4	single mal	none	4	other	60	none	own
16	1015	< 0 DM	15	repaid	car (new)	1403	< 100 DM	1 - 4 yrs	2	female	none	4	other	28	none	rent
17	1016	< 0 DM	24	repaid	radio/tv	1282	101 - 500 D	1 - 4 yrs	4	female	none	2	other	32	none	own
18	1017	unknown	24	critical	radio/tv	2424	unknown	> 7 yrs	4	single mal	none	4	building s	53	none	own
19	1018	< 0 DM	30	fully repai	business	8072	unknown	0 - 1 yrs	2	single mal	none	3	other	25	bank	own

Out target or the dependent variable was ‘default’ which had values Good and Bad. This refers whether the person has returned the loan or not.

To get the descriptive statistics about the data we created a function get_summary_table then those data were stored in a table called info.

We ended up with something like this

	variable	N_Distinct	Type	Missing
1	checking_balance	4	Categorical	0
2	months_loan_duration	33	Numeric	0
3	credit_history	5	Categorical	0
4	purpose	10	Categorical	0
5	amount	921	Numeric	0
6	savings_balance	5	Categorical	0
7	employment_length	5	Categorical	0
8	installment_rate	5	Categorical	0
9	personal_status	5	Categorical	0
10	other_debtors	3	Categorical	0
11	residence_history	5	Categorical	0
12	property	4	Categorical	0

Next, we created a function called get_cuts which helped us to bin up the continuous variables and the discrete variables. Then using those we created a list called get_cuts_list which helped us to store all the bins for each variable.

Name	Type	Value
stats	list [20]	List of length 20
checking_balance	list [2]	List of length 2
months_loan_duration	list [3]	List of length 3
credit_history	list [2]	List of length 2
purpose	list [2]	List of length 2
amount	list [3]	List of length 3
savings_balance	list [2]	List of length 2
employment_length	list [2]	List of length 2
installment_rate	list [2]	List of length 2
personal_status	list [2]	List of length 2

Where the bins for each variable looks like

Name	Type	Value
stats	list [2]	List of length 2
credit_history	list [2]	List of length 2
Type	character [1]	'Categorical'
cut_at	character [5]	'critical' 'repaid' 'delayed' 'fully repaid' 'fully repaid this bank'
amount	list [3]	List of length 3

We now create a function called woe_infoval which helped us to calculate the woe's along with that another function get_data_cut which helped us to convert out initial table into one containing only bins.

	Customer	checking_balance	months_loan_duration	credit_history	purpose	amount	savings_balance	employment_length
1	1001	< 0 DM	(-Inf,9]	critical	radio/tv	(932,1.26e+03]	unknown	> 7 yrs
2	1002	1 - 200 DM	(36, Inf]	repaid	radio/tv	(4.72e+03,7.18e+03]	< 100 DM	1 - 4 yrs
3	1003	unknown	(9,12]	critical	education	(1.91e+03,2.32e+03]	< 100 DM	4 - 7 yrs
4	1004	< 0 DM	(36, Inf]	repaid	furniture	(7.18e+03, Inf]	< 100 DM	4 - 7 yrs
5	1005	< 0 DM	(18,24]	delayed	car (new)	(4.72e+03,7.18e+03]	< 100 DM	1 - 4 yrs
6	1006	unknown	(30,36]	repaid	education	(7.18e+03, Inf]	unknown	1 - 4 yrs
7	1007	unknown	(18,24]	repaid	furniture	(2.32e+03,2.85e+03]	501 - 1000 DM	> 7 yrs
8	1008	1 - 200 DM	(30,36]	repaid	car (used)	(4.72e+03,7.18e+03]	< 100 DM	1 - 4 yrs
9	1009	unknown	(9,12]	repaid	radio/tv	(2.85e+03,3.59e+03]	> 1000 DM	4 - 7 yrs
10	1010	1 - 200 DM	(24,30]	critical	car (new)	(4.72e+03,7.18e+03]	< 100 DM	unemployed
11	1011	1 - 200 DM	(9,12]	repaid	car (new)	(1.26e+03,1.48e+03]	< 100 DM	0 - 1 yrs

Hence, we get the data table df_woe which contains all the woe values

	foreign_worker	Customer	telephone	dependents	job	existing_credits	housing	installment_plan
1	-0.03580569	1001	0.09593931	-0.003937013	0.02122096	0.11236297	0.1924198	0.1197
2	-0.03580569	1002	-0.07103577	-0.003937013	0.02122096	-0.07622031	0.1924198	0.1197
3	-0.03580569	1003	0.76214005	0.009174376	0.25131443	-0.07622031	0.1924198	0.1197
4	-0.03580569	1004	0.76214005	0.009174376	0.02122096	-0.07622031	-0.4761218	0.1197
5	-0.03580569	1005	-0.07103577	0.009174376	0.02122096	0.11236297	-0.4761218	0.1197
6	-0.03580569	1006	0.09593931	0.009174376	0.08479654	-0.07622031	-0.4761218	0.1197
7	-0.03580569	1007	-0.07103577	-0.003937013	0.02122096	-0.07622031	0.1924198	0.1197
8	-0.03580569	1008	0.09593931	-0.003937013	-0.20902729	-0.07622031	-0.4069860	0.1197
9	-0.03580569	1009	-0.07103577	-0.003937013	0.08479654	-0.07622031	0.1924198	0.1197
10	-0.03580569	1010	-0.07103577	-0.003937013	-0.20902729	0.11236297	0.1924198	0.1197
11	-0.03580569	1011	-0.07103577	-0.003937013	0.02122096	-0.07622031	-0.4069860	0.1197
12	-0.03580569	1012	-0.07103577	-0.003937013	0.02122096	-0.07622031	-0.4069860	0.1197

Then I created a list called rv1 that contained DFCUT, DFWOE and WOE_IV for future use that contained the bins we created for each feature, the WOE values for each category and the iv values.

rv1	list [3]	List of length 3
DFCUT	list [1000 x 22] (S3: data.table, da	A data.table with 1000 rows and 22 columns
DFWOE	list [1000 x 23] (S3: data.table, da	A data.table with 1000 rows and 23 columns
WOE_IV	list [20]	List of length 20

Finally, we will fit a logistic regression model using our newly transformed WoE of the training dataset. When scaling the model into a scorecard, we will need both the Logistic Regression coefficients from model fitting as well as the transformed WoE values. We use the glm function (general linear model function) to train the model with the data from the WOE table. The family is binomial which indicates that it is a logistic regression problem. We store the coefficients of the features from the model_summary in a variable coef.

	Variable	Estimate
1	(Intercept)	0.9233768
2	checking_balance	0.8342239
3	months_loan_duration	0.7169071
4	credit_history	0.7439617
5	purpose	0.9792829
6	amount	0.8488013
7	savings_balance	0.8062378
8	employment_length	0.7263110
9	installment_rate	1.7796682
10	personal_status	1.0307780
11	other_debtors	1.1423833
12	residence_history	4.2121514
13	property	0.4303380

The final piece of our puzzle is creating a simple, easy-to-use, and implement credit risk scorecard that can be used by any layperson to calculate an individual's credit score given certain required information about him and his credit history. For each attribute, its Weight of Evidence (WoE) and the regression coefficient of its

characteristic now could be multiplied to give the score points of the attribute. Then scaling is done with the PDO, LOC and want odds ratio.

We choose to scale the points such that a total score of 600 points corresponds to good/bad odds of 50 to 1 and an increase of the score of 20 points corresponds to a doubling of the good/bad odds.

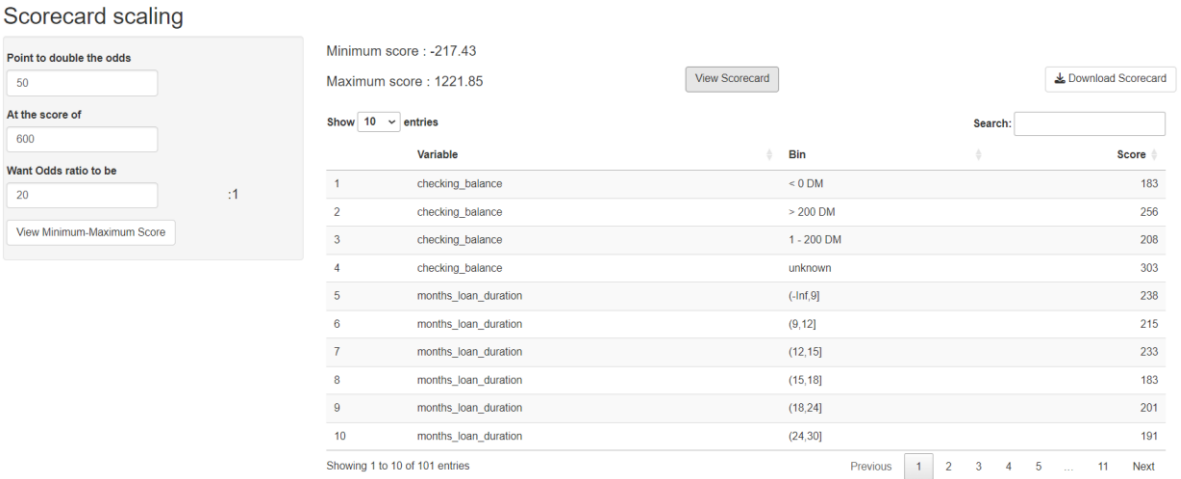
Then we created a function called scard to store the correspond score for each category for all of the variables.

	Variable	Bin	Score
1	checking_balance	< 0 DM	280.88439
2	checking_balance	> 200 DM	308.29587
3	checking_balance	1 - 200 DM	290.37667
4	checking_balance	unknown	326.27767
5	months_loan_duration	(-Inf,9]	271.95222
6	months_loan_duration	(9,12]	263.25563
7	months_loan_duration	(12,15]	269.96044
8	months_loan_duration	(15,18]	250.85032
9	months_loan_duration	(18,24]	257.88012
10	months_loan_duration	(24,30]	254.17961
11	months_loan_duration	(30,36]	245.34953

The score is just the summation from the score card for each person.

Min_sum is the sum of the minimum scores in each feature and maxsum is the sum of the maximum scores in each feature for the observation, this helps us to visualize the score range that we want to set.

So, the scorecard is generated. Now I built an app using R shiny named the Scorecard scaling. Here I change the inputs to get a desired minimum and maximum score. Once set, this scorecard will be the final one with the help of which we will create the new person's score which will be in our desired range.



<u>Ui and server</u>

A simple shiny app is a directory containing two R scripts/files, one is ui. R , which controls the layout and appearance of our app, the other is server. R , which contains the instructions that our computer needs to build our app.

In ui, we have the siderbar panel and the mainpanel. In the siderbar, for specifying numerical values in 'Points to double the odds','at the score of' and 'want odds ratio to be' I used numericInput function. I used action button for 'View Minimum-Maximum Score'. In the mainpanel I used htmlOutput to view the maximum score and the minimum score calculated in the server. Also, I gave an action button to view the score card using DTOutput, a download button to download the desired scorecard.

In the server using previous functions I calculated the minimum and the maximum score and used observeEvent to display the calculated value only after the action tab was used. Again, another observeEvent was added to show the modified score card only after the button ViewScore card was used. I used the downloadhandler to specify what to download and in which format.

PSI (Population stability Index)

In machine learning, data drift refers to a change in the distribution of input data that can impact the performance of a trained model. It occurs when the statistical properties of the incoming data change over time, leading to a discrepancy between the data the model was trained on and the data it encounters during inference.

Data drift can occur due to various reasons, such as changes in user behavior, changes in the underlying population, shifts in data sources or changes in the data collection process. When data drift happens, the model may become less accurate or even produce incorrect predictions.

Detecting data drift is crucial for maintaining model performance.

Thus, it measures to what extent the population that was used to construct the rating system is similar to the population that is currently being observed. The lower, the better stability of the model.

PSI = Sum [(A-D)*Log(A/D)]

Where,

A= Population distribution, actual sample (observed data, at time t)

D= Population distribution, development or training sample (when developed the model (t0))

General convention used in the industry:

PSI < 0.1 means no significant

0.10<=PSI<0.25 means moderate shift

PSI >= 0.25 means significant shift

Created the PSI function that could calculate the PSI for the scores as well as for individual variables. Used the previously built functions and finally using the rshiny we got this.

Stability Check

Upload train data

Browse... Test.csv

Upload complete

Upload out of sample data

Browse... Test.csv

Upload complete

Give the id column name

Customer

Give the target column name

default

Check stability

Variables

Show

Select variable

checking_balance

Character Stability:

checking_balance	train	oot	train_perc	oot_perc	deviation	PSI
< 0 DM	53	53	31.36	31.36	0.00	0.00
> 200 DM	7	7	4.14	4.14	0.00	0.00
1 - 200 DM	34	34	20.12	20.12	0.00	0.00
unknown	75	75	44.38	44.38	0.00	0.00

Stability Check

Upload train data

Browse... Test.csv

Upload complete

Upload out of sample data

Browse... Test.csv

Upload complete

Give the id column name

Customer

Give the target column name

default

Check stability

Scores

Show

Population Stability:

score_bin	train	oot	train_perc	oot_perc	deviation	PSI
(-Inf,966]	17	17	10.06	10.06	0.00	0.00
(966,995]	17	17	10.06	10.06	0.00	0.00
(995,1.01e+03]	17	17	10.06	10.06	0.00	0.00
(1.01e+03,1.05e+03]	17	17	10.06	10.06	0.00	0.00
(1.05e+03,1.07e+03]	17	17	10.06	10.06	0.00	0.00
(1.07e+03,1.09e+03]	17	17	10.06	10.06	0.00	0.00
(1.09e+03,1.11e+03]	16	16	9.47	9.47	0.00	0.00
(1.11e+03,1.13e+03]	17	17	10.06	10.06	0.00	0.00
(1.13e+03,1.17e+03]	17	17	10.06	10.06	0.00	0.00
(1.17e+03, Inf]	17	17	10.06	10.06	0.00	0.00

In the siderbarPanel I used inputid to upload the train and the test data also to mention the id column and the target column based on our problem statement, selectInput to select between score and variable and an action button to display the PSI table. In the server the function that we created is called and is used to calculate the PSI and is based on what we selected on the check stability select input box.