INSTITUTE FOR ADVANCED COMPUTING AND SOFTWARE DEVELOPMENT AKURDI, PUNE

DOCUMENTATION ON

"CREDIT RISK MODELLING"

e-DBDA May-2021

Submitted by:

Group No. : 07

Pavan Khairnar (1323)

Mayur Urane (1355)

Mr. Prashant Karhale
Centre Coordinator

Mr.Akshay Tilekar
Project Guide

TABLE OF CONTENTS

1. Introduction

What is Credit Risk? In simple words, it is the risk of borrower not repaying loan, credit card or any other type of loan. Sometimes customers pay some instalments of loan but don't repay the full amount which includes principal amount plus interest. For example, you took a personal loan of USD 100,000 for 10 years at 9% interest rate. You paid a few initial instalments of loan to the bank but stopped paying afterwards. Remaining unpaid instalments are worth USD 30,000. It's a loss to the bank.

It's not restricted to retail customers but includes small, medium and big corporate houses. In news, you might have heard of Kingfisher Company became non-performing asset (NPA) which means the company had not been able to pay dues. High NPAs lead to huge financial losses to the bank which turns to reduction of interest rate on the deposit into banks. Serious honest borrowers with good credit history (credit score) would have to suffer. Hence it is essential that banks have sufficient capital to protect depositors from risks.

What is Credit Risk Modelling? Credit risk modelling refers to data driven risk models which calculates the chances of a borrower defaults on loan (or credit card). If a borrower fails to repay loan, how much amount he/she owes at the time of default and how much lender would lose from the outstanding amount. In other words, we need to build probability of default, loss given default and exposure at default models as per advanced IRB approach under Basel norms.

1.1. Statement of the problem

Do you remember or aware of 2008 recession? In US, mortgage home loan were given to low creditworthy customers (individuals with poor credit score). Poor credit score indicates that one is highly likely to default on loan which means they are risky customers for bank. To compensate risk, banks used to charge higher interest rate than the normal standard rate. Banks funded these loans by selling them to investors on the secondary market. The process of selling them to investors is a legal financial method which is called Collateralized debt obligations (CDO). In 2004-2007, these CDOs were considered as low-risky financial instrument (highly rated).

As these home loan borrowers had high chance to default, many of the them started defaulting on their loans and banks started seizing (foreclose) their property. The real estate bubble burst and a sharp decline in home prices. Many financial institutions globally invested in these funds resulted to a recession. Banks, investors and re-insurers faced huge financial losses and bankruptcy of many financial and non-financial firms. Even non-financial firms were impacted badly

because of either their investment in these funds or impacted because of a very low demand and purchasing activities in the economy. In simple words, people had a very little or no money to spend which leads to many organizations halted their production. It further leads to huge job losses. US Government bailed out many big corporate houses during recession. You may have understood now why credit risk is so important. The whole economy can be in danger if current and future credit losses are not identified or estimated properly.

1.2 Project Goals and Scope

We will go through detailed steps to develop a data-driven credit risk model in Python to predict the probabilities of default (PD) and assign credit scores to existing or potential borrowers. We will determine credit scores using a highly interpretable, easy to understand and implement scorecard that makes calculating the credit score a breeze

1.3 Novelty/Benefits:

Reduce time to credit decisions

Financial institutions spend a significant amount of money and time in physically verifying applicant details. AI can be leveraged to extract meaningful insights from unstructured alternate data sources such as text and images, and to verify the authenticity of the information provided by applicants without the necessity for physical investigation. This helps significantly reduce loan or corporate credit processing time.

Improve customer experience with intelligent product selection

Today’s digitally savvy customers prefer personalized products that are relevant and customized to their needs. Intelligent analysis of smartphone metadata and transactional data help zero in on the most pertinent customer information to offer a pre-selection of suitable credit products. This, in turn, helps enhance customer experience.

Check creditworthiness through smart applications

Once the customer zeroes in on the credit products, smart credit scoring apps using AI-based algorithm can help analyze customer behavior in real-time. This helps extract and contextualize relevant information to verify the customer’s credit worthiness and calculating maximum credit limit. AI can also be used to enhance decisions for structured financing through reliable estimates of future cash flow and ability to pay back debt.

Meet regulatory requirements

It is crucial for banks to meet regulatory requirement of leveraged transactions which requires them to ensure due diligence for granting loans or refinancing existing transactions. With high quality data input such as smartphone metadata, AI applications help reduce data bias and create a transparent approach to enable credit scoring.

For banks, using AI and machine learning to enrich the credit risk management process not only brings in greater efficiency, but also enhance fraud detection mechanisms and reduce time to market. AI's power to adopt new data sources and analyze with more granularity is the way forward to ensure accurate credit scores, improve credit risk detection and engage better with customers.

2. Product Overview and Summary

2.1 Purpose

The main purpose of credit risk analysis is to quantify the level of credit risk that the borrower presents to the lender. It involves assigning measurable numbers to the estimated probability of default of the borrower.
Credit risk analysis is a form of analysis performed by a credit analyst on potential borrowers to determine their ability to meet debt obligations. The main goal of credit analysis is to determine the creditworthiness of potential borrowers and their ability to honor their debt obligations.

If the borrower presents an acceptable level of default risk, the analyst can recommend the approval of the credit application at the agreed terms. The outcome of the credit risk analysis determines the risk rating that the borrower will be assigned and their ability to access credit.

2.2 Scope

Credit risk is defined as the risk of loss resulting from the failure by a borrower to repay the principal and interest owed to the leader. The lender uses the interest payments from the loan to compensate for the risk of potential losses. When the borrower defaults on his/her obligations, it causes an interruption in the cash flows of the lender.

Performing credit risk analysis helps the lender determine the borrower's ability to meet debt obligations in order to cushion itself from loss of cash flows and reduce the severity of losses. Borrowers who present a high level of credit risk are charged a high interest rate on the loan to compensate the lender for the high risk of default.

When calculating the credit risk of a particular borrower, lenders consider various factors commonly referred to as the "5 Cs of Credit." The factors include the borrower's capacity to repay credit, character, capital, conditions, and collateral. The lender uses the factors to evaluate the characteristics of the borrower and conditions of the loan to estimate the probability of default and the subsequent risk of financial loss.

The 5 Cs of Credit incorporate both qualitative and quantitative financial measures, and the lender may analyze different documents, such as the borrower's income

statement, balance sheet, credit reports, and other documents that reveal the financial situation of the borrower.

The Overarching Purpose of Credit Risk Analysis
Credit analysts may use various financial analysis techniques, such as ratio analysis and trend analysis to obtain measurable numbers that quantify the credit loss. The techniques measure the risk of credit loss due to changes in the creditworthiness of borrowers.

When measuring the credit loss, we consider both losses from counterparty default, as well as deteriorating credit risk rating

Drivers that Quantify Credit Risk
The following are the key parameters that show a higher correlative relationship with credit risk:

1. Probability of default
Probability of default is defined as the probability that the borrower will not be able to make scheduled principal and interest payments over a specified period, usually one year. The default probability depends on both the borrower's characteristics and the economic environment.

For individuals, the default probability is determined by the FICO score, and lenders use the score to decide whether or not to extend credit. For business entities, the default probability is implied by the credit rating.

Usually, credit rating agencies are required to assign a credit rating to entities that issue debt instruments, such as bonds. Borrowers with a high default probability are charged a higher interest rate to compensate the lender for bearing the higher default risk.

2. Loss given default
Loss given default is defined as the amount of money that a lender stands to lose when a borrower defaults on the debt obligations. While there is no accepted method for quantifying the loss given default per loan, most lenders calculate loss given default as a percentage of total exposure to loss in the entire loan portfolio.

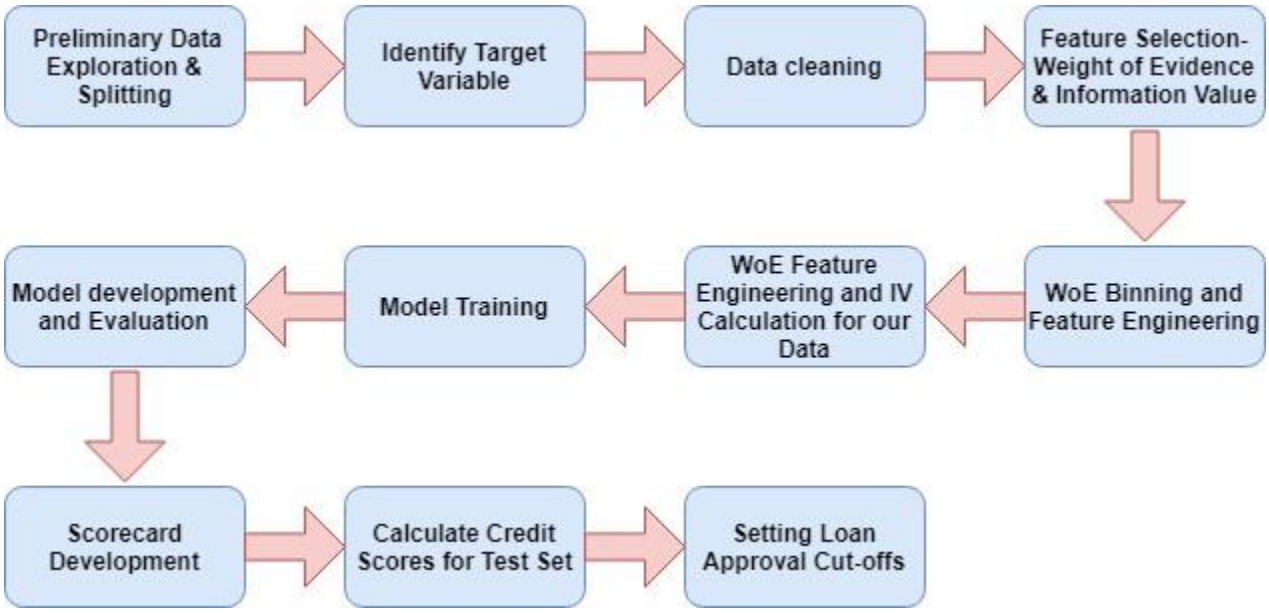
For example, if ABC Bank lends $1,000 to Borrower A and $10,000 to Borrower B, the bank stands to lose more money in the event that Borrower B defaults on repayments.

3. Exposure at default
Exposure at default measures the amount of loss that a lender is exposed to at any particular point, due to loan defaults. Financial institutions often use their internal risk management models to estimate the level of exposure at default.

Initially, the exposure is calculated per loan, and banks use the figure to determine the overall default risk for the entire loan portfolio. As borrowers make loan repayments, the value of exposure at default reduces gradually.

2.3 Summary
Credit risk analysis is a form of analysis performed by a credit analyst to determine a borrower's ability to meet their debt obligations.
The purpose of credit analysis is to determine the creditworthiness of borrowers by quantifying the risk of loss that the lender is exposed to.
The three factors that lenders use to quantify credit risk include the probability of default, loss given default, and exposure at default.

3. Project Flow Diagram



4. Data Analysis and Interpretation

4.1 Preliminary Data Exploration & Splitting

We will use a dataset made available on Kaggle that relates to consumer loans issued by the Lending Club, a US P2P lender. The raw data includes information on over 450,000 consumer loans issued between 2007 and 2014 with almost 75 features, including the current loan status and various attributes related to both borrowers and their payment behavior. Refer to the data dictionary for further details on each column.

The concepts and overall methodology, as explained here, are also applicable to a corporate loan portfolio.

Initial data exploration reveals the following:

18 features with more than 80% of missing values. Given the high proportion of missing values, any technique to impute them will most likely result in inaccurate results

Certain static features not related to credit risk, e.g., id, member_id, url, title

Other forward-looking features that are expected to be populated only once the borrower has defaulted, e.g., recoveries, collection_recovery_fee. Since our objective here is to predict the future probability of default, having such features in our model will be counterintuitive, as these will not be observed until the default event has occurred

We will drop all the above features.

Identify Target Variable

Based on the data exploration, our target variable appears to be loan_status. A quick look at its unique values and their proportion thereof confirms the same.

```
1  # explore the unique values in loan_status column
2  loan_data['loan_status'].value_counts(normalize = True)
```

```
Current                                             0.480878
Fully Paid                                          0.396193
Charged Off                                         0.091092
Late (31-120 days)                                  0.014798
In Grace Period                                     0.006747
Does not meet the credit policy. Status:Fully Paid  0.004263
Late (16-30 days)                                   0.002612
Default                                             0.001784
Does not meet the credit policy. Status:Charged Off 0.001632
Name: loan_status, dtype: float64
```

Image 1: Distribution of defaults

Based on domain knowledge, we will classify loans with the following loan_status values as being in default (or 0):

Charged Off

Default
Late (31–120 days)
Does not meet the credit policy. Status:Charged Off
All the other values will be classified as good (or 1).
Data Split
Let us now split our data into the following sets: training (80%) and test (20%). We will perform Repeated Stratified k Fold testing on the training test to preliminary evaluate our model while the test set will remain untouched till final model evaluation. This approach follows the best model evaluation practice.
Image 1 above shows us that our data, as expected, is heavily skewed towards good loans. Accordingly, in addition to random shuffled sampling, we will also stratify the train/test split so that the distribution of good and bad loans in the test set is the same as that in the pre-split data. This is achieved through the train_test_split function's stratify parameter.
Splitting our data before any data cleaning or missing value imputation prevents any data leakage from the test set to the training set and results in more accurate model evaluation. A code snippet for the work performed so far follows:

4.2 Data Cleaning

Next comes some necessary data cleaning tasks as follows:
Remove text from the emp_length column (e.g., years) and convert it to numeric
For all columns with dates: convert them to Python's datetime format, create a new column as a difference between model development date and the respective date feature and then drop the original feature
Remove text from the term column and convert it to numeric
We will define helper functions for each of the above tasks and apply them to the training dataset. Having these helper functions will assist us with performing these same tasks again on the test dataset without repeating our code.

4.3 Feature Selection

Next up, we will perform feature selection to identify the most suitable features for our binary classification problem using the Chi-squared test for categorical features and ANOVA F-statistic for numerical features.
The p-values, in ascending order, from our Chi-squared test on the categorical features are as below:

	Feature	p-value
0	grade	0.000000
1	home_ownership	0.000000
2	verification_status	0.000000
3	purpose	0.000000
4	addr_state	0.000000
5	initial_list_status	0.000000
6	pymnt_plan	0.000923
7	application_type	1.000000

Image 2: p-values from Chi-squared test

For the sake of simplicity, we will only retain the top four features and drop the rest.

The ANOVA F-statistic for 34 numeric features shows a wide range of F values, from 23,513 to 0.39. We will keep the top 20 features and potentially come back to select more in case our model evaluation results are not reasonable enough.

Next, we will calculate the pair-wise correlations of the selected top 20 numerical features to detect any potentially multicollinear variables. A heat-map of these pair-wise correlations identifies two features (out_prncp_inv and total_pymnt_inv) as highly correlated. Therefore, we will drop them also for our model.

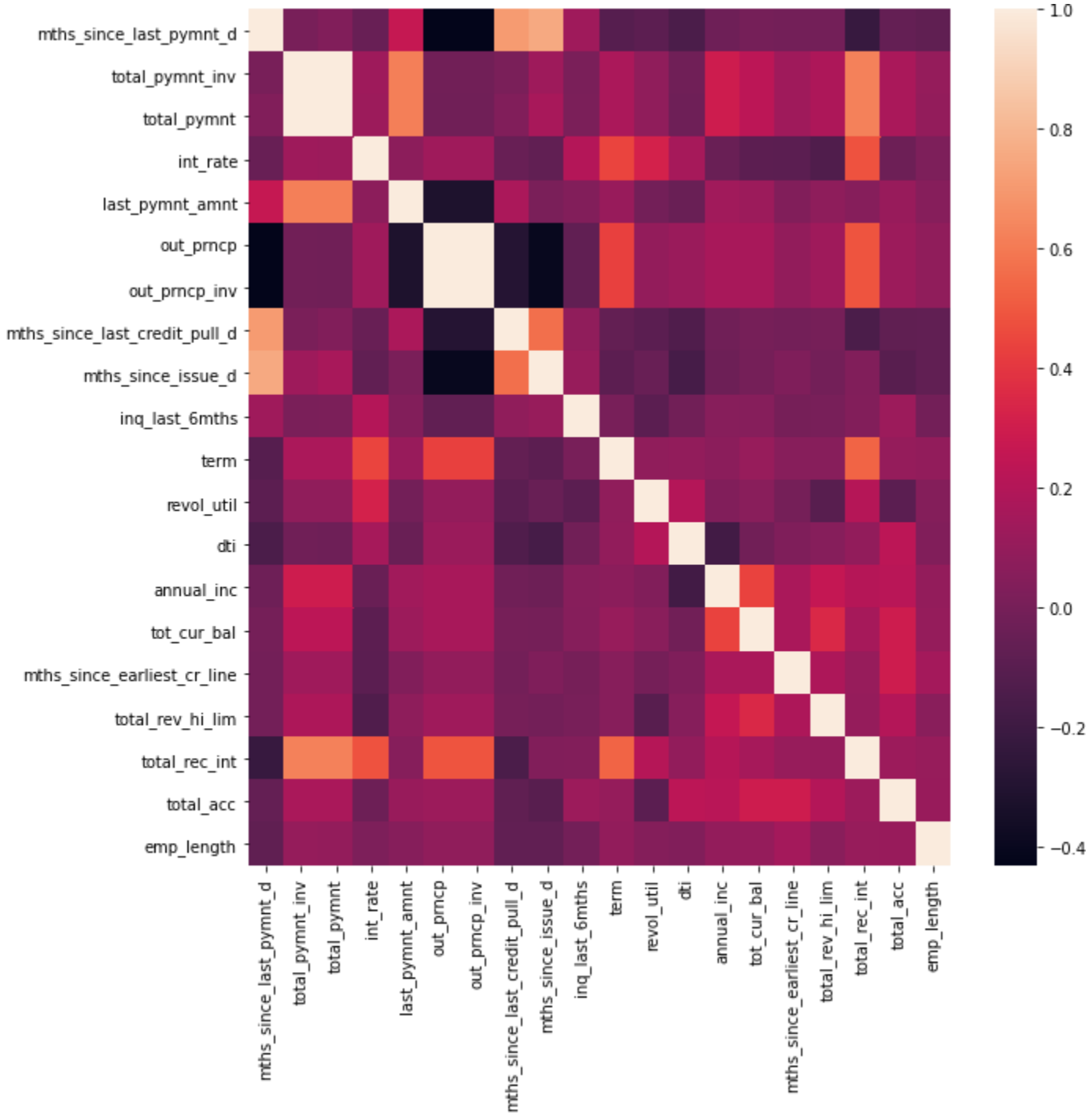


Image 3: Pair-wise correlations

Next, we will simply save all the features to be dropped in a list and define a function to drop them. The code for these feature selection techniques follows:

One-Hot Encoding and Update Test Dataset

Next, we will create dummy variables of the four final categorical variables and update the test dataset through all the functions applied so far to the training dataset.

Note a couple of points regarding the way we create dummy variables:

We will use a particular naming convention for all variables: original variable name, colon, category name

Generally speaking, in order to avoid multicollinearity, one of the dummy variables is dropped through the drop_first parameter of pd.get_dummies. However, we will not do so at this stage as we require all the dummy variables to calculate the Weight of Evidence (WoE) and Information Values (IV) of our categories — more on this later. We will drop one dummy variable for each category later on . We will also not create the dummy variables directly in our training data, as doingso would drop the categorical variable, which we require for WoE calculations. Therefore, we will create a new dataframe of dummy variables and thenconcatenate it to the original training/test dataframe.
Next up, we will update the test dataset by passing it through all the functions defined so far. Pay special attention to reindexing the updated test dataset after creating dummy variables. Let me explain this by a practical example.
Consider a categorical feature called grade with the following unique values in the pre-split data: A, B, C, and D. Suppose that the proportion of D is very low, and due to the random nature of train/test split, none of the observations with D in the grade category is selected in the test set. Therefore, grade's dummy variables in the training data will be grade:A, grade:B, grade:C, and grade:D, but grade:D will not be created as a dummy variable in the test set. We will be unable to apply a fitted model on the test set to make predictions, given the absence of a feature expected to be present by the model. Therefore, we reindex the test set to ensure that it has the same columns as the training data, with any missing columns being added with 0 values. A 0 value is pretty intuitive since that category will never be observed in any of the test samples.

4.4 WoE Binning and Feature Engineering

Creating new categorical features for all numerical and categorical variables based on WoE is one of the most critical steps before developing a credit risk model, and also quite time-consuming.
But what is WoE and IV?
Weight of Evidence (WoE) and Information Value (IV) are used for feature engineering and selection and are extensively used in the credit scoring domain.
WoE is a measure of the predictive power of an independent variable in relation to the target variable. It measures the extent a specific feature can differentiate between target classes, in our case: good and bad customers.
IV assists with ranking our features based on their relative importance.
According to Baesens et al.[1] and Siddiqi[2], WOE and IV analyses enable one to:
Consider each variable's independent contribution to the outcome
Detect linear and non-linear relationships
Rank variables in terms of its univariate predictive strength
Visualize the correlations between the variables and the binary outcome
Seamlessly compare the strength of continuous and categorical variables without creating dummy variables
Seamlessly handle missing values without imputation. (Note that we have not imputed any missing values so far, this is the reason why. Missing values will be

assigned a separate category during the WoE feature engineering step)
Assess the predictive power of missing values
Weight of Evidence (WoE)
The formula to calculate WoE is as follow:

$$WoE = \ln\left(\frac{\%\ of\ good\ customers}{\%\ of\ bad\ customers}\right)$$

A positive WoE means that the proportion of good customers is more than that of bad customers and vice versa for a negative WoE value.
Steps for WoE feature engineering
Calculate WoE for each unique value (bin) of a categorical variable, e.g., for each of grad:A, grad:B, grad:C, etc.
Bin a continuous variable into discrete bins based on its distribution and number of unique observations, maybe using pd.cut (called fine classing)
Calculate WoE for each derived bin of the continuous variable
Once WoE has been calculated for each bin of both categorical and numerical features, combine bins as per the following rules (called coarse classing)
Rules related to combining WoE bins
Each bin should have at least 5% of the observations
Each bin should be non-zero for both good and bad loans
The WOE should be distinct for each category. Similar groups should be aggregated or binned together. It is because the bins with similar WoE have almost the same proportion of good or bad loans, implying the same predictive power
The WOE should be monotonic, i.e., either growing or decreasing with the bins
Missing values are binned separately
The above rules are generally accepted and well documented in academic literature[3].
Why discretize numerical features
Discretization, or binning, of numerical features, is generally not recommended for machine learning algorithms as it often results in loss of data. However, our end objective here is to create a scorecard based on the credit scoring model eventually. A scorecard is utilized by classifying a new untrained observation (e.g., that from the test dataset) as per the scorecard criteria.
Consider that we don't bin continuous variables, then we will have only one category for income with a corresponding coefficient/weight, and all future potential borrowers would be given the same score in this category, irrespective of their income. If, however, we discretize the income category into discrete classes (each with different WoE) resulting in multiple categories, then the potential new borrowers would be classified into one of the income categories according to their income and would be scored accordingly.
WoE binning of continuous variables is an established industry practice that has been in place since FICO first developed a commercial scorecard in the 1960s, and there is substantial literature out there to support it. Some of the other rationales to

discretize continuous features from the literature are:
A scorecard is usually legally required to be easily interpretable by a layperson (a requirement imposed by the Basel Accord, almost all central banks, and various lending entities) given the high monetary and non-monetary misclassification costs. This is easily achieved by a scorecard that does not has any continuous variables, with all of them being discretized. Reasons for low or high scores can be easily understood and explained to third parties. All of this makes it easier for scorecards to get 'buy-in' from end-users compared to more complex models
Another legal requirement for scorecards is that they should be able to separate low and high-risk observations[4]. WoE binning takes care of that as WoE is based on this very concept
Monotonicity. It is expected from the binning algorithm to divide an input dataset on bins in such a way that if you walk from one bin to another in the same direction, there is a monotonic change of credit risk indicator, i.e., no sudden jumps in the credit score if your income changes. This arises from the underlying assumption that a predictor variable can separate higher risks from lower risks in case of the global non-monotonous relationship7
An underlying assumption of the logistic regression model is that all features have a linear relationship with the log-odds (logit) of the target variable. Is there a difference between someone with an income of $38,000 and someone with $39,000? Most likely not, but treating income as a continuous variable makes this assumption. By categorizing based on WoE, we can let our model decide if there is a statistical difference; if there isn't, they can be combined in the same category
Missing and outlier values can be categorized separately or binned together with the largest or smallest bin — therefore, no assumptions need to be made to impute missing values or handle outliers

4.5 Information Value (IV)

IV is calculated as follows:

$$IV = \sum (\%\ of\ good\ customers - \%\ of\ bad\ customers) \times WoE$$

According to Siddiqi[2], by convention, the values of IV in credit scoring is interpreted as follows:

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

Note that IV is only useful as a feature selection and importance technique when using a binary logistic regression model.

WoE Feature Engineering and IV Calculation for our Data

Enough with the theory, let's now calculate WoE and IV for our training data and perform the required feature engineering. We will define three functions as follows, each one to:

calculate and display WoE and IV values for categorical variables

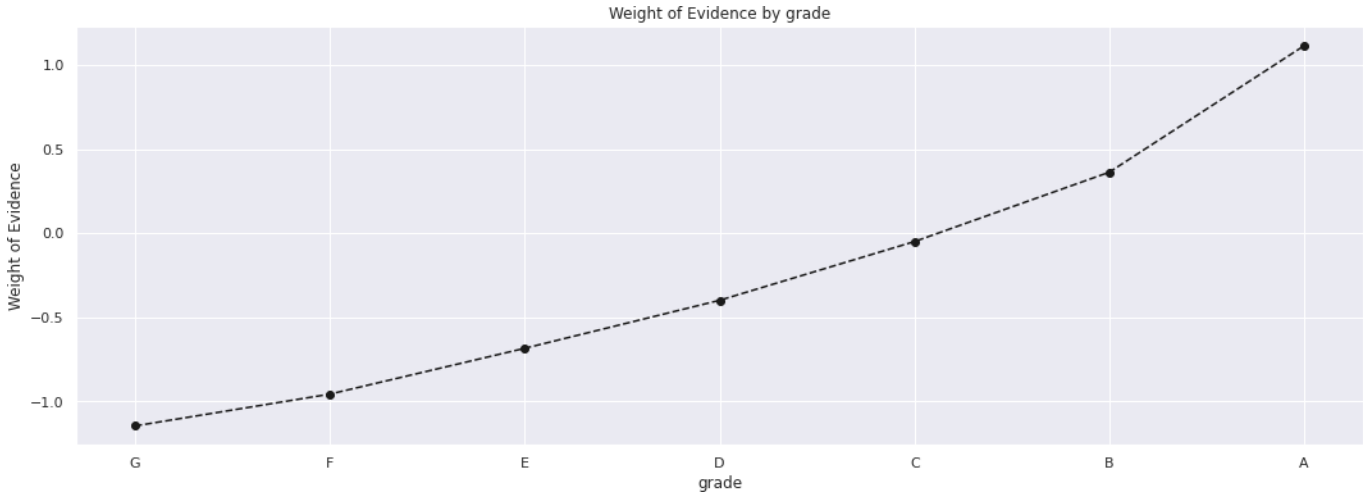
calculate and display WoE and IV values for numerical variables

plot the WoE values against the bins to help us in visualizing WoE and combining similar WoE bins

Sample output of these two functions when applied to a categorical feature, grade, is shown below:

	grade	n_obs	prop_good	prop_n_obs	n_good	n_bad	prop_n_good	prop_n_bad	WoE	diff_prop_good	diff_WoE	IV
0	G	2623	0.721693	0.007032	1893.0	730.0	0.005697	0.017904	-1.144981	NaN	NaN	0.292145
1	F	10606	0.758061	0.028432	8040.0	2566.0	0.024198	0.062932	-0.955774	0.036369	0.189207	0.292145
2	E	28590	0.804477	0.076643	23000.0	5590.0	0.069224	0.137097	-0.683340	0.046416	0.272434	0.292145
3	D	61713	0.845527	0.165438	52180.0	9533.0	0.157049	0.233801	-0.397915	0.041050	0.285425	0.292145
4	C	100342	0.885870	0.268993	88890.0	11452.0	0.267536	0.280865	-0.048620	0.040343	0.349295	0.292145
5	B	109344	0.921422	0.293125	100752.0	8592.0	0.303238	0.210723	0.363975	0.035552	0.412595	0.292145
6	A	59810	0.961361	0.160336	57499.0	2311.0	0.173057	0.056678	1.116232	0.039939	0.752257	0.292145

WoE and IV values



WoE plot

Once we have calculated and visualized WoE and IV values, next comes the most tedious task to select which bins to combine and whether to drop any feature given its IV. The shortlisted features that we are left with until this point will be treated in one of the following ways:

There is no need to combine WoE bins or create a separate missing category given the discrete and monotonic WoE and absence of any missing
values: grade, verification_status, term

Combine WoE bins with very low observations with the neighboring
bin: home_ownership, purpose

Combine WoE bins with similar WoE values together, potentially with a separate missing
category: int_rate, annual_inc, dti, inq_last_6mths, revol_util, out_prncp, total_py mnt, total_rec_int, total_rev_hi_lim, mths_since_earliest_cr_line, mths_since_issu e_d, mths_since_last_credit_pull_d

Ignore features with a low or very high IV
value: emp_length, total_acc, last_pymnt_amnt, tot_cur_bal, mths_since_last_pym nt_d_factor

Note that for certain numerical features with outliers, we will calculate and plot WoE after excluding them that will be assigned to a separate category of their own.

Once we have explored our features and identified the categories to be created, we will define a custom ‘transformer’ class using sci-kit
learn’s BaseEstimator and TransformerMixin classes. Like other sci-kit learn’s ML models, this class can be fit on a dataset to transform it as per our requirements. Another significant advantage of this class is that it can be used as part of a sci-kit learn’s Pipeline to evaluate our training data using Repeated Stratified k-Fold Cross-Validation. Using a Pipeline in this structured way will allow us to perform cross-validation without any potential data leakage between the training and test folds.we have been using all the dummy variables so far, so we will also drop one dummy variable for each category using our custom class to avoid multicollinearity.

The code for our three functions and the transformer class related to WoE and IV follows:

4.6 Model Training

Finally, we come to the stage where some actual machine learning is involved. We will fit a logistic regression model on our training set and evaluate it using RepeatedStratifiedKFold. Note that we have defined the class_weight parameter of the LogisticRegression class to be balanced. This will force the logistic regression model to learn the model coefficients using cost-sensitive learning, i.e., penalize false negatives more than false positives during model training. Cost-sensitive learning is useful for imbalanced datasets, which is usually the case in credit scoring. Refer to my previous article for further details on imbalanced classification problems.

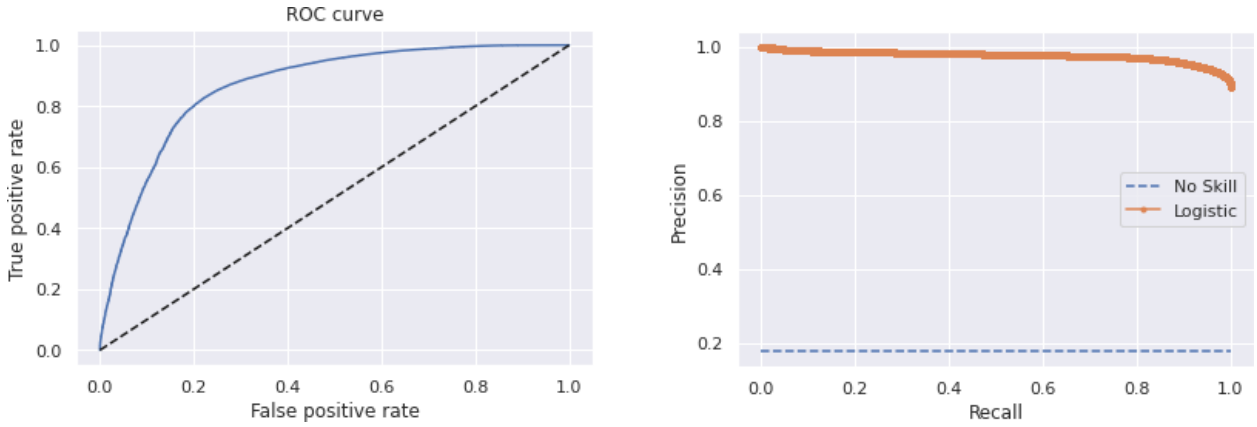
Our evaluation metric will be Area under the Receiver Operating Characteristic Curve (AUROC), a widely used and accepted metric for credit scoring. RepeatedStratifiedKFold will split the data while preserving the class imbalance and perform k-fold validation multiple times.

After performing k-folds validation on our training set and being satisfied with AUROC, we will fit the pipeline on the entire training set and create a summary table with feature names and the coefficients returned from the model.

Prediction Time

It all comes down to this: apply our trained logistic regression model to predict the probability of default on the test set, which has not been used so far (other than for the generic data cleaning and feature selection tasks). We will save the predicted probabilities of default in a separate dataframe together with the actual classes.

Next, we will draw a ROC curve, PR curve, and calculate AUROC and Gini. Our AUROC on test set comes out to 0.866 with a Gini of 0.732, both being considered as quite acceptable evaluation scores. Our ROC and PR curves will be something like this:



Code for predictions and model evaluation on the test set is:

4.7 Scorecard Development

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. Remember the summary table created during the model training phase? We will append all the reference categories that we left out from our model to it, with a coefficient value of 0, together with another column for the original feature name (e.g., grade to represent grade:A, grade:B, etc.).

We will then determine the minimum and maximum scores that our scorecard should spit out. As a starting point, we will use the same range of scores used by FICO: from 300 to 850.

The coefficients returned by the logistic regression model for each feature category are then scaled to our range of credit scores through simple arithmetic. An additional step here is to update the model intercept's credit score through further scaling that will then be used as the starting point of each scoring calculation.

At this stage, our scorecard will look like this (the Score-Preliminary column is a simple rounding of the calculated scores):

Feature name	Coefficients	Original feature name	Score - Calculation	Score - Preliminary
Intercept	2.948892	Intercept	598.515609	599.0
grade:A	0.980200	grade	24.463996	24.0
grade:B	0.792926	grade	19.789993	20.0
grade:C	0.609305	grade	15.207145	15.0
grade:D	0.487386	grade	12.164269	12.0
grade:E	0.331929	grade	8.284354	8.0
grade:F	0.190615	grade	4.757402	5.0
home_ownership:OWN	-0.048374	home_ownership	-1.207327	-1.0
home_ownership:OTHER_NONE_RENT	-0.104195	home_ownership	-2.600515	-3.0
verification_status:Source Verified	-0.281427	verification_status	-7.023912	-7.0
verification_status:Verified	-0.459417	verification_status	-11.466206	-11.0
purpose:debt_consolidation	-0.315579	purpose	-7.876281	-8.0
purpose:credit_card	-0.222064	purpose	-5.542316	-6.0
purpose:educ__ren_en__sm_b__mov	-0.481318	purpose	-12.012820	-12.0
purpose:vacation__house__wedding__med__oth	0.003981	purpose	0.099355	0.0
term:36	-0.094122	term	-2.349118	-2.0
int_rate:<7.071	0.955408	int_rate	23.845255	24.0
int_rate:7.071-10.374	0.285426	int_rate	7.123724	7.0
int_rate:10.374-13.676	0.056631	int_rate	1.413405	1.0
int_rate:13.676-15.74	0.039281	int_rate	0.980393	1.0

Depending on your circumstances, you may have to manually adjust the Score for a random category to ensure that the minimum and maximum possible scores for any given situation remains 300 and 850. Some trial and error will be involved here.

4.8 Calculate Credit Scores for Test Set

Once we have our final scorecard, we are ready to calculate credit scores for all the observations in our test set. Remember, our training and test sets are a simple

collection of dummy variables with 1s and 0s representing whether an observation belongs to a specific dummy variable. For example, in the image below, observation 395346 had a C grade, owns its own home, and its verification status was Source Verified.

	Intercept	grade:A	grade:B	grade:C	grade:D	grade:E	grade:F	home_ownership:OWN	home_ownership:OTHER_NONE_RENT	verification_status:Source Verified	verification_status:Verified
395346	1	0	0	1	0	0	0	1	0	1	0
376583	1	1	0	0	0	0	0	0	1	0	0
297790	1	0	0	1	0	0	0	0	1	1	0
47347	1	0	1	0	0	0	0	0	1	0	1
446772	1	0	0	0	1	0	0	0	0	1	0

Accordingly, after making certain adjustments to our test set, the credit scores are calculated as a simple matrix dot multiplication between the test set and the final score for each category. Consider the above observations together with the following final scores for the intercept and grade categories from our scorecard:

Feature name	Final Score
Intercept	598
grade:A	24
grade:B	20
grade:C	15
grade:D	12
grade:E	8
grade:F	5

Intuitively, observation 395346 will start with the intercept score of 598 and receive 15 additional points for being in the grade:C category. Similarly, observation 3766583 will be assigned a score of 598 plus 24 for being in the grade:A category. We will automate these calculations across all feature categories using matrix dot multiplication. The final credit score is then a simple sum of individual scores of each feature category applicable for an observation.

Setting Loan Approval Cut-offs

So how do we determine which loans should we approve and reject? What is the ideal credit score cut-off point, i.e., potential borrowers with a credit score higher than this cut-off point will be accepted and those less than it will be rejected? This cut-off point should also strike a fine balance between the expected loan approval and rejection rates.

To find this cut-off, we need to go back to the probability thresholds from the ROC curve. Remember that a ROC curve plots FPR and TPR for all probability thresholds between 0 and 1. Since we aim to minimize FPR while maximizing TPR, the top left corner probability threshold of the curve is what we are looking for. This ideal threshold is calculated using the Youden's J statistic that is a simple difference between TPR and FPR.

The ideal probability threshold in our case comes out to be 0.187. All observations with a predicted probability higher than this should be classified as in Default and

vice versa. At first, this ideal threshold appears to be counterintuitive compared to a more intuitive probability threshold of 0.5. But remember that we used the class_weight parameter when fitting the logistic regression model that would have penalized false negatives more than false positives.
We then calculate the scaled score at this threshold point. As shown in the code example below, we can also calculate the credit scores and expected approval and rejection rates at each threshold from the ROC curve. This can help the business to further manually tweak the score cut-off based on their requirements.

4. CONCLUSION

Credit risk modelling in python can help banks and other financial institutions reduce risk and prevent society from experiencing financial crises. Thus, in this way we have created a model to predict probability of person defaulting a loan.