Stock Valuation Ratios and their Role in Investment Decisions, Gurugram

ISBN 978-1-943295-24-1

Harsh Nipesh Thakkar
ICFAI Business School
(harshthakk989@gmail.com)

Peacock Solar is a household solar installation company based in Gurugram, Haryana. It provides hassle free installation of solar power. In an era of rising demand for renewable energy, solar power is seen as a future of energy. The markets are becoming more competitive as better technologies increase the efficiency and lower the cost of solar power. In India, solar power is in its nascent stage of development and being price sensitive markets, cost remains the bottom line of competition. The present study is an attempt to showcase the strategy adopted by Peacock solar to enhance its sales by making solar available on finance." The objective of this research paper is come up with a model that anticipates the probability associated with default for homeowner who avails solar on finance. The next objective is to develop a scorecard that represents this probability of default in form of credit score for enhanced understanding and decision making. By making solar available on finance, the company aims to overcome its price related hindrances. The methodology used for development of credit risk model is Logistic Regression as it is one of the best techniques for predicting a binary outcome (will default or will not default). This is followed by a technique for scorecard development. It can then be concluded that credit risk can be reduced to a considerable extent if correct analytical methodologies are put in place which will bring down the default rates on credit.

Keywords: Credit Risk Modelling, Creditworthiness and Solar Power

1. Introduction

1.1 Company Profile

Peacock Solar specialises in solar services. It provides hassle free installation of solar power. Since, its inception in 2017, it has successfully completed 40+ projects and has thus saved 24 metric tonnes of carbon dioxide emission. In terms of power, it has till date installed project of 200KW. Presently operating in Kota (Rajasthan), the company is all set to mark its presence in pan India.
With the advent of 21st century, the energy demands have skyrocketed all over the world. Lack of technological development in the energy sector has put pressure of non-renewable sources to match up the demand for energy. This has caused rise in the level of carbon dioxide and other poisonous gases to increase at an alarming rate.
As solar remains a tough market to compete in Indian environment, Peacock solar has maintained its edge by deploying cutting edge technology and financial innovation to increase its reach and reduce the cost of solar installation. Peacock solar strives for cleaner environment by providing solar as an energy alternative. With technological advancement in last decade, cost of solar panels has reduced significantly.

1.2 Introduction to Credit Risk Modelling

Credit risk is the chance of a borrower defaulting on a debt by failing to make the required payments. Risk is an inherent part of the lending paradigm for financial institutions and other lenders. Pinpointing the amount of risk that comes with each loan is a difficult task. Credit risk modelling has multiple aspects to it, not only we need to calculate the probability for measuring the chances for default, we also need to assess what is the extent to which the company will suffer a loss in case of default.
Historical data of consumers is collected to study the behaviour of consumers based on selected parameters from the data. Then data is refined and worked to bring out a model that will help in anticipating the associated probabilities of default. This helps companies to create a cover that will enable them to prepare for such uncertainties.

1.3 How peacock solar aims to use Credit Risk Modelling

Peacock solar operates under two business model:

- CAPEX
- OPEX

Under CAPEX (Capital Expenditure) business model, the homeowners pay upfront for solar power installation at their place. In this way entire burden of the cost is borne by the homeowners.

Under OPEX (Operational Expenditure) business model, the company on its own expenditure installs the plant at homeowner's premise and the homeowners are expected to pay for their monthly usage unit wise. Here arises the risk of default on the part of homeowner. If homeowner defaults then entire cost is borne by the company. So, to reduce this risk the company want to develop a credit risk model to assess the probability of default of the homeowner before the installation is made.

2. Literature Review

Credit risk can be assessed using accounting measures (Altman, 1968 and Ohlson, 1980). They made accounting variables as a basis for assessing the risk for corporate bankruptcy. Applying statistics to accounting variable help to get a score. They developed a Z score that would show the chances for business to enter bankruptcy. The higher the score greater will be the chances of company facing bankruptcy. The limitation of this study was that is predictability is based on financial statements which are prepared on the principal of going concern and thus it is assumed that the firm will not default.

The concept of value at risk is a better measure for assessing the credit risk (Jorion, 2006). This approach is used by J.P. Morgan and chase and is based on the idea that the model used to manage credit risk has to be applicable to all types of financial instruments subjected to substantial credit risk and valuation methods have to correspond with actual market prices. Credit Metrics is therefore used for valuation of bond prices. Credit Metrics offers a different approach to credit risk measuring than structural models. It uses completely different variables than structural approaches. Credit Metrics offers so called empirical Value at Risk approach to measuring credit risk, which should reflect current market prices, and addresses the question "how much funding will be lost" in the worst case. As it more of a hybrid model and contains structural model characteristic. Then the stock price consideration will again act as a drawback because market prices do not reflect true market sentiments and does not have all information contained in them.

"Risk neutrals", which is a way in which we compute no risk probabilities of upcoming cash flows and the calculating their present value at T-bill rate (Jarrow and Turnbull,1992). They divided the problem of credit risk modelling in two part, firstly assessing the chances of default and the assessing the loss that can be suffered due to default.

2.1 Consolidated Research Gaps

- Analytical approach to assessing credit risk remains a research gaps in all of the articles that were reviewed. A proper analytical tool or methodology could be a better way for assessing the credit risk.
- Representation of credit risk in form of probability associated with default is another research gap identified. There should be a lucid way to represent probability of default that would enhance the interpretability for results.

2.2 Research Objective

Based on research gaps explored above in the literature review and based on the recommendations of the company, Peacock Solar, the following are the objectives of research report

- To develop a model for estimating probability of default associated to an applicant homeowner who wants solar on finance.
- To develop a scorecard to effectively represent probability of default for enhanced understanding.

3. Research Methodology

As research objectives have been formulated above based on the gaps identified in the literature review. This part of report describes the research methodology in a logical sequence which can be adopted to achieve these objectives.

3.1 Methodology for Objective

3.1.1 Sample Size

The dataset that is used for model development is an American dataset that has 466284 observations.

3.1.2 Modelling Technique

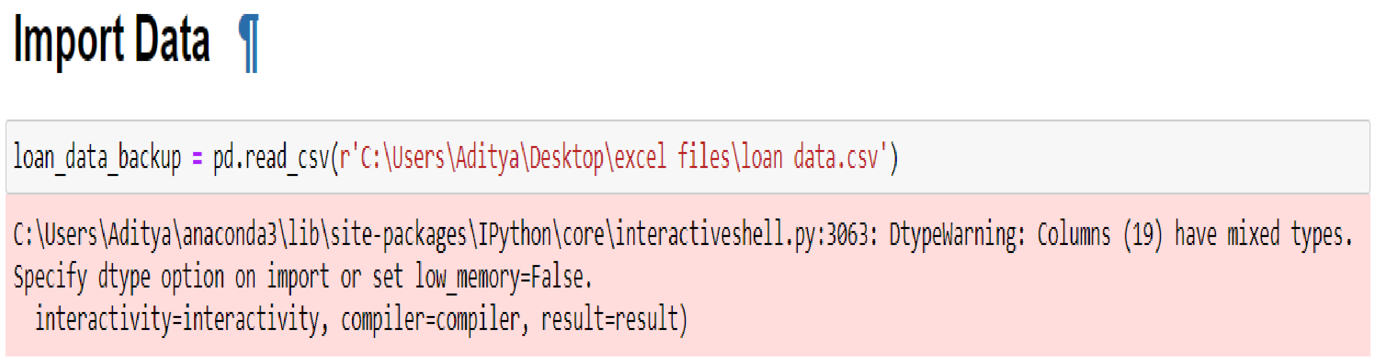
Logistic regression has been used to develop the model for anticipating the probability associated with default.

3.1.3 Modelling Steps

3.1.3.1 General Pre-Processing of Data

As this model development was done on python, the first step was to get each and every variable into a correct datatype as supported in python.

Identifying data types of each variable

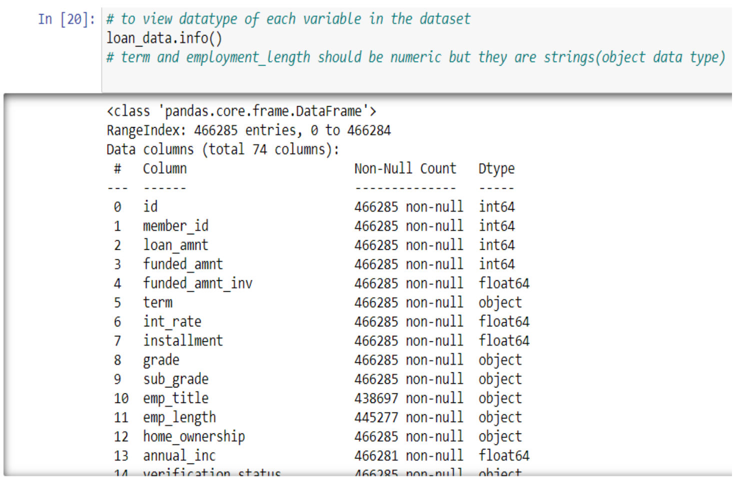


Import Data ¶

```
loan_data_backup = pd.read_csv(r'C:\Users\Aditya\Desktop\excel files\loan data.csv')
```

```
C:\Users\Aditya\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3063: DtypeWarning: Columns (19) have mixed types.
Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

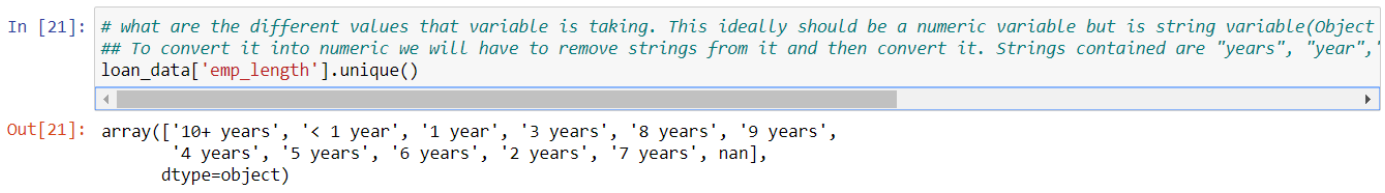
Figure 1



```
In [20]: # to view datatype of each variable in the dataset
         loan_data.info()
         # term and employment_length should be numeric but they are strings(object data type)

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 466285 entries, 0 to 466284
Data columns (total 74 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   id                           466285 non-null  int64
 1   member_id                    466285 non-null  int64
 2   loan_amnt                    466285 non-null  int64
 3   funded_amnt                  466285 non-null  int64
 4   funded_amnt_inv              466285 non-null  float64
 5   term                         466285 non-null  object
 6   int_rate                     466285 non-null  float64
 7   installment                  466285 non-null  float64
 8   grade                        466285 non-null  object
 9   sub_grade                    466285 non-null  object
 10  emp_title                    438697 non-null  object
 11  emp_length                   445277 non-null  object
 12  home_ownership               466285 non-null  object
 13  annual_inc                   466281 non-null  float64
 14  verification_status          466285 non-null  object
```

Pre-Processing Continuous Variables

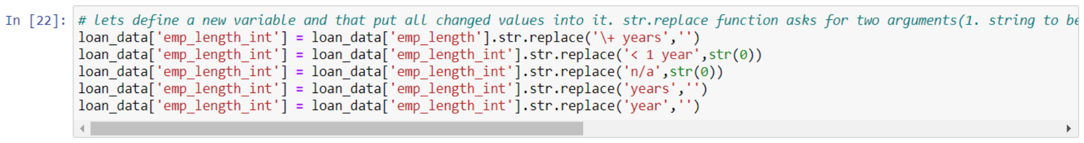
Identifying all different values that a continuous variable is taking using unique () python function.



```
In [21]: # what are the different values that variable is taking. This ideally should be a numeric variable but is string variable(Object
         ## To convert it into numeric we will have to remove strings from it and then convert it. Strings contained are "years", "year",'
         loan_data['emp_length'].unique()

Out[21]: array(['10+ years', '< 1 year', '1 year', '3 years', '8 years', '9 years',
                '4 years', '5 years', '6 years', '2 years', '7 years', nan],
               dtype=object)
```

Figure 2

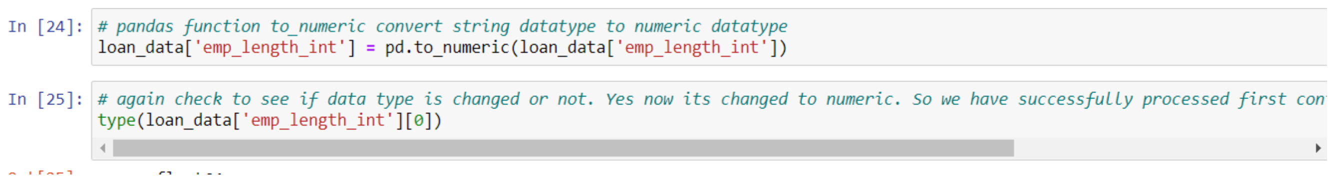
Removing strings from the continuous variable to convert them to numeric datatype.



```
In [22]: # lets define a new variable and that put all changed values into it. str.replace function asks for two arguments(1. string to be
         loan_data['emp_length_int'] = loan_data['emp_length'].str.replace('\+ years','')
         loan_data['emp_length_int'] = loan_data['emp_length_int'].str.replace('< 1 year',str(0))
         loan_data['emp_length_int'] = loan_data['emp_length_int'].str.replace('n/a',str(0))
         loan_data['emp_length_int'] = loan_data['emp_length_int'].str.replace('years','')
         loan_data['emp_length_int'] = loan_data['emp_length_int'].str.replace('year','')
```

Figure 3

Now finally converting to numeric datatype.



```
In [24]: # pandas function to_numeric convert string datatype to numeric datatype
         loan_data['emp_length_int'] = pd.to_numeric(loan_data['emp_length_int'])

In [25]: # again check to see if data type is changed or not. Yes now its changed to numeric. So we have successfully processed first con
         type(loan_data['emp_length_int'][0])

Out[25]:         float64
```

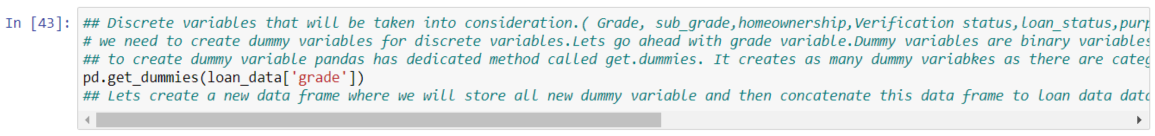
Figure 4

This example above shows how a single continuous variable in the dataset was converted to its correct form for operations in python. In a similar way all other continuous variables were also processed to make them ready for further analysis in python.

Pre-processing of Discrete Variables

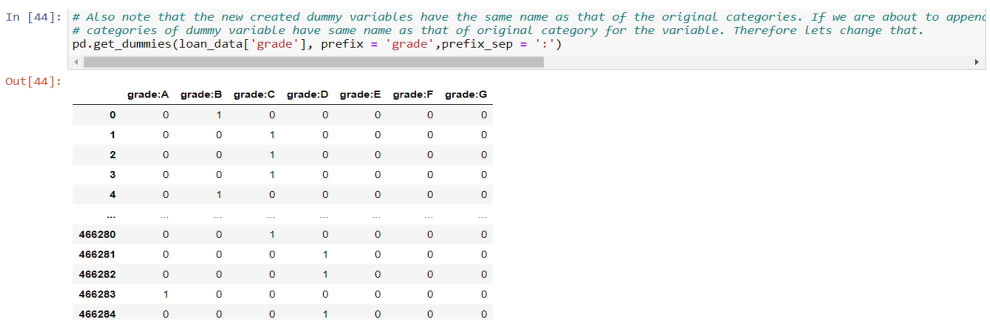
An example of one of the discrete variables 'Grades' is taken to show how all discrete variables in the data set were processed to their correct datatype in python.

Firstly, dummy variables were created for each of the categories of discrete or categorical variable.



```
In [43]: ## Discrete variables that will be taken into consideration.( Grade, sub_grade,homeownership,Verification status,loan_status,purp
         # we need to create dummy variables for discrete variables.Lets go ahead with grade variable.Dummy variables are binary variables
         ## to create dummy variable pandas has dedicated method called get.dummies. It creates as many dummy variabkes as there are categ
         pd.get_dummies(loan_data['grade'])
         ## Lets create a new data frame where we will store all new dummy variable and then concatenate this data frame to loan data data
```

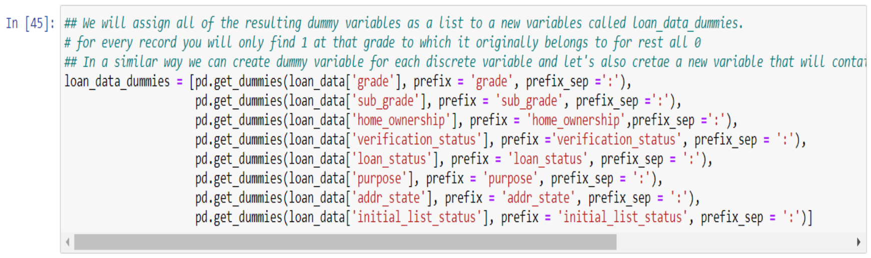
Figure 5



```
In [44]: # Also note that the new created dummy variables have the same name as that of the original categories. If we are about to appenc
         # categories of dummy variable have same name as that of original category for the variable. Therefore lets change that.
         pd.get_dummies(loan_data['grade'], prefix = 'grade',prefix_sep = ':')

Out[44]:
         grade:A  grade:B  grade:C  grade:D  grade:E  grade:F  grade:G
      0        0        1        0        0        0        0        0
      1        0        0        1        0        0        0        0
      2        0        0        1        0        0        0        0
      3        0        0        1        0        0        0        0
      4        0        1        0        0        0        0        0
    ...      ...      ...      ...      ...      ...      ...      ...
 466280        0        0        1        0        0        0        0
 466281        0        0        0        1        0        0        0
 466282        0        0        0        1        0        0        0
 466283        1        0        0        0        0        0        0
 466284        0        0        0        1        0        0        0
```

Figure 6

In this way every observation in the dataset will have value of 1 at the grade category to which it belongs and rest all will be zero. In a similar way dummy variable were created for all discrete variables for better representations.

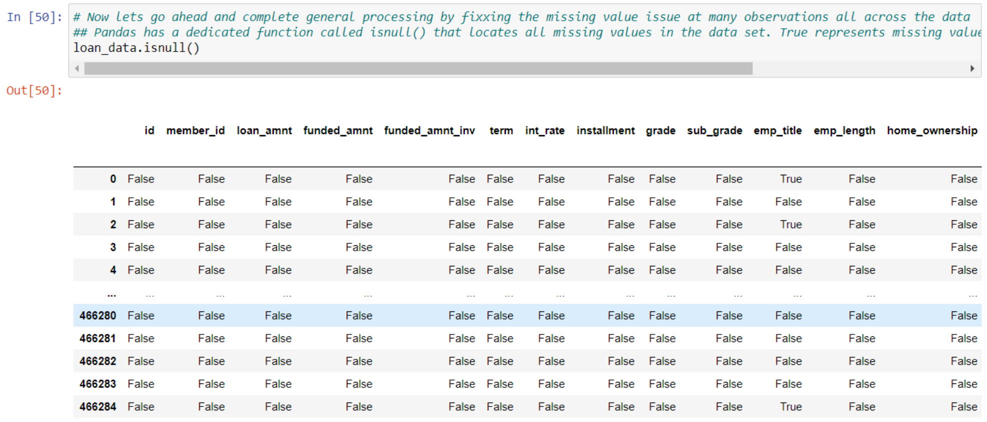


```
In [45]: ## We will assign all of the resulting dummy variables as a list to a new variables called loan_data_dummies.
         # for every record you will only find 1 at that grade to which it originally belongs to for rest all 0
         ## In a similar way we can create dummy variable for each discrete variable and let's also cretae a new variable that will contai
         loan_data_dummies = [pd.get_dummies(loan_data['grade'], prefix = 'grade', prefix_sep =':'),
                              pd.get_dummies(loan_data['sub_grade'], prefix = 'sub_grade', prefix_sep =':'),
                              pd.get_dummies(loan_data['home_ownership'], prefix = 'home_ownership',prefix_sep =':'),
                              pd.get_dummies(loan_data['verification_status'], prefix ='verification_status', prefix_sep = ':'),
                              pd.get_dummies(loan_data['loan_status'], prefix = 'loan_status', prefix_sep = ':'),
                              pd.get_dummies(loan_data['purpose'], prefix = 'purpose', prefix_sep = ':'),
                              pd.get_dummies(loan_data['addr_state'], prefix = 'addr_state', prefix_sep = ':'),
                              pd.get_dummies(loan_data['initial_list_status'], prefix = 'initial_list_status', prefix_sep = ':')]
```

Figure 7

Pre-Processing for Missing Values

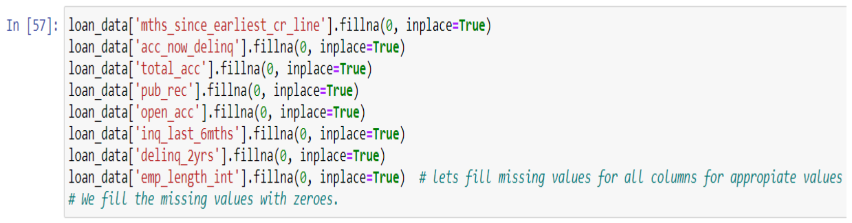
Identify missing values in each column using is null function of python.



```
In [50]: # Now lets go ahead and complete general processing by fixxing the missing value issue at many observations all across the data
         ## Pandas has a dedicated function called isnull() that locates all missing values in the data set. True represents missing value
         loan_data.isnull()

Out[50]:
            id  member_id  loan_amnt  funded_amnt  funded_amnt_inv   term  int_rate  installment  grade  sub_grade  emp_title  emp_length  home_ownership

      0  False      False      False        False            False  False     False        False  False      False       True       False           False
      1  False      False      False        False            False  False     False        False  False      False      False       False           False
      2  False      False      False        False            False  False     False        False  False      False       True       False           False
      3  False      False      False        False            False  False     False        False  False      False      False       False           False
      4  False      False      False        False            False  False     False        False  False      False      False       False           False
    ...    ...        ...        ...          ...              ...    ...       ...          ...    ...        ...        ...         ...             ...
 466280  False      False      False        False            False  False     False        False  False      False      False       False           False
 466281  False      False      False        False            False  False     False        False  False      False      False       False           False
 466282  False      False      False        False            False  False     False        False  False      False      False       False           False
 466283  False      False      False        False            False  False     False        False  False      False      False       False           False
 466284  False      False      False        False            False  False     False        False  False      False       True       False           False
```

Figure 8

Replacing missing values of interest variables with zero.



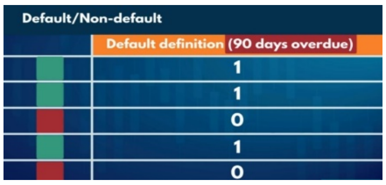
```
In [57]: loan_data['mths_since_earliest_cr_line'].fillna(0, inplace=True)
         loan_data['acc_now_delinq'].fillna(0, inplace=True)
         loan_data['total_acc'].fillna(0, inplace=True)
         loan_data['pub_rec'].fillna(0, inplace=True)
         loan_data['open_acc'].fillna(0, inplace=True)
         loan_data['inq_last_6mths'].fillna(0, inplace=True)
         loan_data['delinq_2yrs'].fillna(0, inplace=True)
         loan_data['emp_length_int'].fillna(0, inplace=True)  # lets fill missing values for all columns for appropiate values
         # We fill the missing values with zeroes.
```

Figure 9

3.1.3.2 PD Model: Data Preparation

The first step is setting up of dependent variable or the what is to be predicted. The variable of interest in the dataset is ‘loan status’, as it clearly sets out the status of an individual on his loan obligation. Therefore, to be able to distinguish between good and bad loans, it will need to define what default is. More specifically, it needs a default definition. This definition comprises rules stating when a borrower is considered to have defaulted on a loan.

A common definition is that a borrower has defaulted if they are more than 90 days past due on the loan. But this is not the only definition to be used though. The default definition results in the loan being classified as a not defaulted or good or defaulted or bad. Usually, a new variable in the data set of a Boolean or binary type is created. Zero will stand for defaulted loan and 1 for good loan.



Default/Non-default	
	Default definition (90 days overdue)
🟩	1
🟩	1
🟥	0
🟩	1
🟥	0

Figure 10

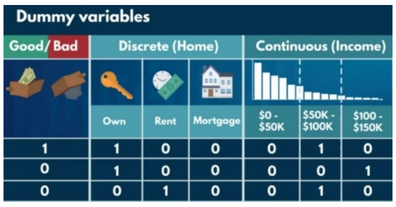
The established statistical methodology to model probability of default is a logistic regression where the dependent variable is precisely whether a customer defaulted or not. The logistic regression estimates relationship between two things, the logarithm of odds of an outcome of interest or dependent variable and a linear combination of predictors. In our case the outcome of interest is the non-default or default event.

$$\ln\left(\frac{\text{Non-defaults}}{\text{Defaults}}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_m X_m$$

Figure 11

When it comes to PD model, interpretability is of prime importance as required by regulator for Basel 2. The model should be very easy to understand and apply. Even people who have never heard of statistical analysis should be able to work with it. This is why it is an established practice for all independent variables in PD model to be dummy variables that is binary categorical variables or indicator variable.

In other words, independent variables will be included in the model as dummy variable. Dummy variables were already created earlier. But, while it has many discrete variables such as external rating etc, it also has many continuous variables. The convention is to convert continuous variable to dummy variable. Once that is achieved the PD model will be a logistic regression model with the binary indicator for good or bad or non-default or default.



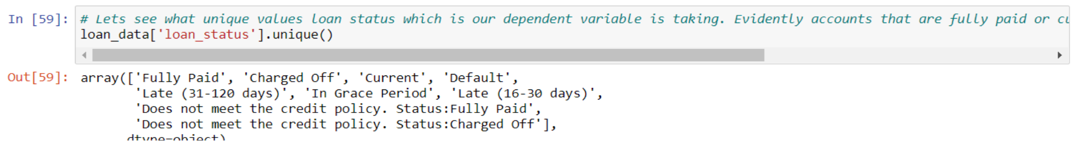
Dummy variables

Good/ Bad	Discrete (Home)			Continuous (Income)		
	Own	Rent	Mortgage	\$0 - \$50K	\$50K - \$100K	\$100 - \$150K
1	1	0	0	0	1	0
0	1	0	0	0	0	1
0	0	1	0	0	1	0

Figure 12

Processing Dependent Variable

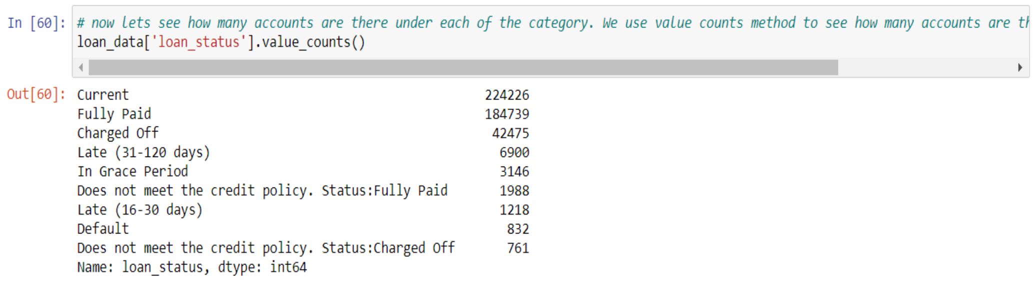
Identifying all unique values that dependent variable is taking.



```
In [59]: # Lets see what unique values loan status which is our dependent variable is taking. Evidently accounts that are fully paid or cu
         loan_data['loan_status'].unique()

Out[59]: array(['Fully Paid', 'Charged Off', 'Current', 'Default',
                'Late (31-120 days)', 'In Grace Period', 'Late (16-30 days)',
                'Does not meet the credit policy. Status:Fully Paid',
                'Does not meet the credit policy. Status:Charged Off'],
               dtype=object)
```

Figure 13

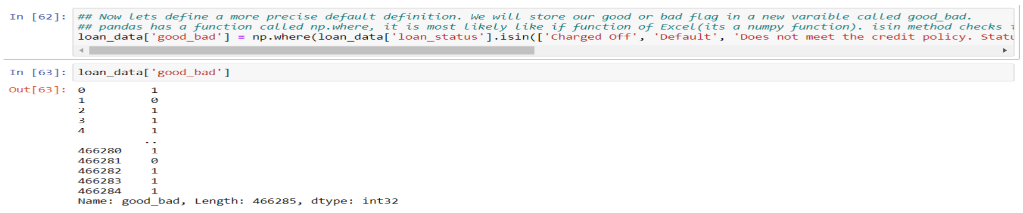
Using value count to see how many accounts belongs to each of categories of dependent variables.



```
In [60]: # now lets see how many accounts are there under each of the category. We use value counts method to see how many accounts are th
         loan_data['loan_status'].value_counts()

Out[60]: Current                                                224226
         Fully Paid                                             184739
         Charged Off                                             42475
         Late (31-120 days)                                       6900
         In Grace Period                                          3146
         Does not meet the credit policy. Status:Fully Paid       1988
         Late (16-30 days)                                        1218
         Default                                                   832
         Does not meet the credit policy. Status:Charged Off       761
         Name: loan_status, dtype: int64
```

Figure 14

The loan status of charged off, default, does not meet credit policy status: charged off, late (31-120 days) are all to be counted under defaulted loan rest all in not defaulted. Going ahead with a code that would segregate all loans under default column as 0 and all other as 1. It will then be stored in a new variable called good bad variable that will act as dependent variable.



```
In [62]: ## Now lets define a more precise default definition. We will store our good or bad flag in a new varaible called good_bad.
         ## pandas has a function called np.where, it is most likely like if function of Excel(its a numpy function). isin method checks i
         loan_data['good_bad'] = np.where(loan_data['loan_status'].isin(['Charged Off', 'Default', 'Does not meet the credit policy. Statu

In [63]: loan_data['good_bad']
Out[63]: 0         1
         1         0
         2         1
         3         1
         4         1
                  ..
         466280    1
         466281    0
         466282    1
         466283    1
         466284    1
         Name: good_bad, Length: 466285, dtype: int32
```

Figure 15

Processing Independent continuous variable to categorical dummy variables.

Continuous variables are required to be converted to categorical dummy variable so going ahead with PD model. This can be achieved using technique of Fine classing, weight of Evidence, Coarse classing, Information value.

Below is an example to explain this work.

Example: If we select a variable to range from zero to 100 such as the debt-to-income ratio, it could split it into 50 categories with 2% width each. 0-2, 2-4, 4-6 and so on. These initial categories rarely matter as it will later bundle them up nonetheless.

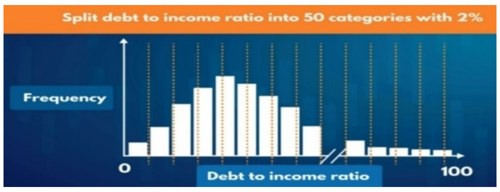


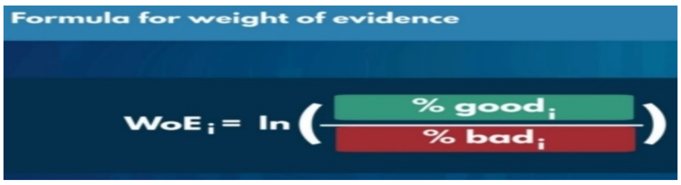
Figure 16

So conceptually with fine classing both discreet and continuous variable can be represented in the form of categories. But the question that still remains is how to actually run these arbitrary categories into good usable dummies. Since it will be having categories in both cases i.e. discrete and continuous, the approach of creating dummies for continuous variable will remain same.

It starts by getting some rough initial assessment of the ability of each category in continuous variable to predict the dependent variable. Using the technique called Weight of Evidence we can differentiate in a better way between good and bad loans. More specifically, WoE shows the extent to which each of the different categories of an independent variable explains the dependent variable. It is calculated using the formula.

The formula of the weight of evidence is the natural logarithm of the ratio of the proportion of observations of the first type of outcome of the dependent variable that fall in each of the category of the explanatory variable and the proportion of the observation of the second type of outcome of the dependent variable that fall into the each of the categories of the explanatory variable.

So, this formula in our case will be like this. The two type of outcome are not defaulted and defaulted or bad. So, the weight of evidence would be the natural logarithm of the ration of the proportion of goods from the total number of good loans that fall into the category to the proportions of bads to total number of bads that fall into a category.



$$WoE_i = \ln\left(\frac{\%\,good_i}{\%\,bad_i}\right)$$

Figure 17

The course classing is a process of merging initial categories based on similar WoE and creating broader categories. It has many continuous variables that will have initially 5 or 6 or may even have 50 categories initially. But, based on weight of evidence it will combine them into bigger categories. Usually, it has preferred those categories that have similar weight of evidence to be bundled up together. In this way it lowers the number of dummies and improve our PD model.

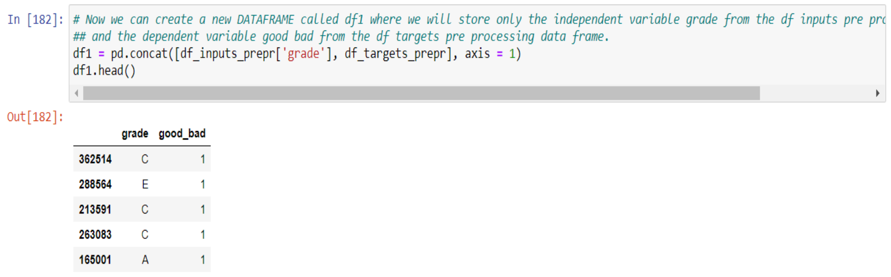
For information value understanding, let us assume that an original independent variable has been split into categories. It may have been categorical originally or categories might have been determined through fine classing. Suppose there are k categories of this variable, then it can calculate weight of evidence for each of the k categories. From there it can weight these weights of evidence of each category. It can weight each by the difference of the proportion of goods from the total number of good that fell into the respective category and the proportion of bads from the total number of bads that fall into the respective category. Then it simply sums them to reach a weighted average of the weight of evidence of the k categories. The result is called Information value of the original explanatory variable with respect to the outcome variable.

Information value shows the extent to which original explanatory variable explains the outcome variable. Therefore, information value can be used for preselection of variable.

$$\sum_{i=1}^{k}\left[\left(\%\,\text{good} - \%\,\text{bad}\right) \times \ln\left(\frac{\%\,\text{good}}{\%\,\text{bad}}\right)\right]$$

Figure 18

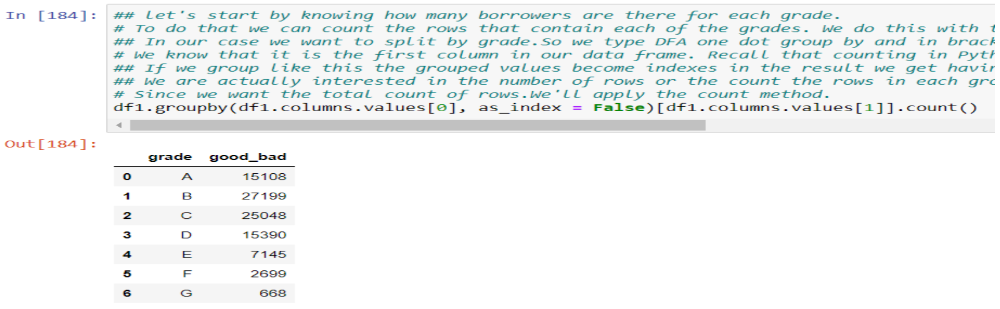
The value of information value ranges between zero and one. The father away the value is from zero, the better the independent variable is explaining the dependent variable.
Calculating weight of evidences for discrete variables
Weight of evidences calculated for each of the discrete variables. Below is an example showing how this was carried out for 'Grade' variable.
First all those accounts that are identified that belonged to each of the grade.



```
In [182]: # Now we can create a new DATAFRAME called df1 where we will store only the independent variable grade from the df inputs pre pro
          ## and the dependent variable good bad from the df targets pre processing data frame.
          df1 = pd.concat([df_inputs_prepr['grade'], df_targets_prepr], axis = 1)
          df1.head()

Out[182]:
               grade  good_bad
        362514     C         1
        288564     E         1
        213591     C         1
        263083     C         1
        165001     A         1
```

Figure 19

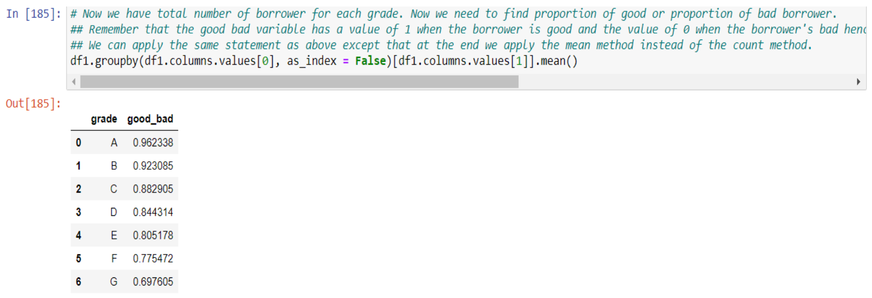
The values of the grade variable are letters from A to G. They represent external grades, with A showing the highest credit worthiness and G showing the lowest. In order to find the weight of evidence of grade, it must first find the proportion of good and bad borrowers by grade. Thus, it first calculates number of borrowers in each grade. To do that it need to count the rows that contain each of the grades.



```
In [184]: ## Let's start by knowing how many borrowers are there for each grade.
          # To do that we can count the rows that contain each of the grades. We do this with t
          ## In our case we want to split by grade.So we type DFA one dot group by and in brack
          # We know that it is the first column in our data frame. Recall that counting in Pyth
          ## If we group like this the grouped values become indexes in the result we get havir
          ## We are actually interested in the number of rows or the count the rows in each gro
          # Since we want the total count of rows.We'll apply the count method.
          df1.groupby(df1.columns.values[0], as_index = False)[df1.columns.values[1]].count()

Out[184]:
             grade  good_bad
          0      A     15108
          1      B     27199
          2      C     25048
          3      D     15390
          4      E      7145
          5      F      2699
          6      G       668
```

Figure 20

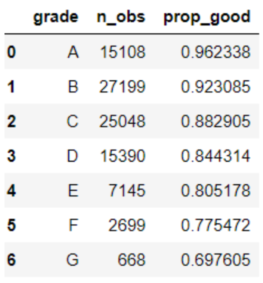
Another piece of information that will be need is the proportion of good and bad borrowers are within each group. This can be summarised either by the proportion of good borrowers or by the proportion of bad borrowers. It does not matter which one. Since the proportion of good borrowers equal to 1 minus proportion of bad borrowers. Let's calculate proportion of good borrowers here. The good bad variable has a value equal to 1 when the borrower is good and 0 when borrower is bad. Hence, it would get proportion of good borrower simply by calculating the average of good or bad.



```
In [185]: # Now we have total number of borrower for each grade. Now we need to find proportion of good or proportion of bad borrower.
          ## Remember that the good bad variable has a value of 1 when the borrower is good and the value of 0 when the borrower's bad henc
          ## We can apply the same statement as above except that at the end we apply the mean method instead of the count method.
          df1.groupby(df1.columns.values[0], as_index = False)[df1.columns.values[1]].mean()

Out[185]:
             grade  good_bad
          0      A  0.962338
          1      B  0.923085
          2      C  0.882905
          3      D  0.844314
          4      E  0.805178
          5      F  0.775472
          6      G  0.697605
```

Figure 21

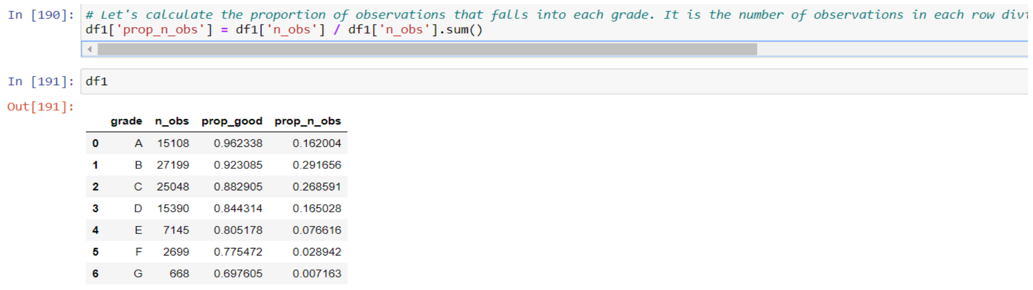
Now the two previous tables are concatenated.



	grade	n_obs	prop_good
0	A	15108	0.962338
1	B	27199	0.923085
2	C	25048	0.882905
3	D	15390	0.844314
4	E	7145	0.805178
5	F	2699	0.775472
6	G	668	0.697605

Figure 22

Now calculate proportion of observation that falls into each grade. It is the number of observations in each row divided by the sum of the number of observations in each in each row.

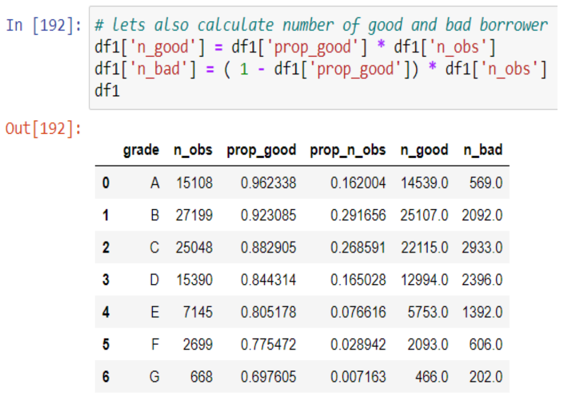


```
In [190]: # Let's calculate the proportion of observations that falls into each grade. It is the number of observations in each row divi
          df1['prop_n_obs'] = df1['n_obs'] / df1['n_obs'].sum()

In [191]: df1

Out[191]:
```

	grade	n_obs	prop_good	prop_n_obs
0	A	15108	0.962338	0.162004
1	B	27199	0.923085	0.291656
2	C	25048	0.882905	0.268591
3	D	15390	0.844314	0.165028
4	E	7145	0.805178	0.076616
5	F	2699	0.775472	0.028942
6	G	668	0.697605	0.007163

Figure 23

Now calculate number of good and bad borrower by grade. It will store the number of good borrowers in a variable called good.

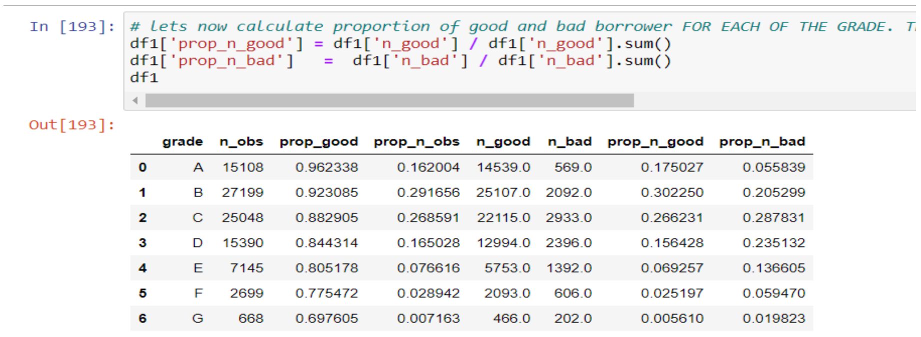


```
In [192]: # lets also calculate number of good and bad borrower
          df1['n_good'] = df1['prop_good'] * df1['n_obs']
          df1['n_bad'] = ( 1 - df1['prop_good']) * df1['n_obs']
          df1

Out[192]:
```

	grade	n_obs	prop_good	prop_n_obs	n_good	n_bad
0	A	15108	0.962338	0.162004	14539.0	569.0
1	B	27199	0.923085	0.291656	25107.0	2092.0
2	C	25048	0.882905	0.268591	22115.0	2933.0
3	D	15390	0.844314	0.165028	12994.0	2396.0
4	E	7145	0.805178	0.076616	5753.0	1392.0
5	F	2699	0.775472	0.028942	2093.0	606.0
6	G	668	0.697605	0.007163	466.0	202.0

Figure 24

Now calculate the proportion of good borrowers and bad borrowers for each grade.

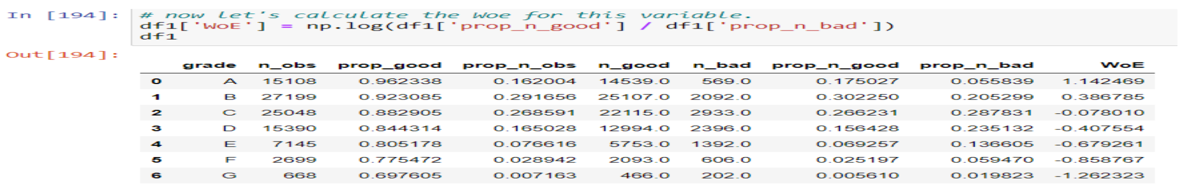


```
In [193]: # lets now calculate proportion of good and bad borrower FOR EACH OF THE GRADE. T
          df1['prop_n_good'] = df1['n_good'] / df1['n_good'].sum()
          df1['prop_n_bad']   =  df1['n_bad'] / df1['n_bad'].sum()
          df1

Out[193]:
```

	grade	n_obs	prop_good	prop_n_obs	n_good	n_bad	prop_n_good	prop_n_bad
0	A	15108	0.962338	0.162004	14539.0	569.0	0.175027	0.055839
1	B	27199	0.923085	0.291656	25107.0	2092.0	0.302250	0.205299
2	C	25048	0.882905	0.268591	22115.0	2933.0	0.266231	0.287831
3	D	15390	0.844314	0.165028	12994.0	2396.0	0.156428	0.235132
4	E	7145	0.805178	0.076616	5753.0	1392.0	0.069257	0.136605
5	F	2699	0.775472	0.028942	2093.0	606.0	0.025197	0.059470
6	G	668	0.697605	0.007163	466.0	202.0	0.005610	0.019823

Figure 25

In this way there is everything it need to calculate weight of evidence for the categories of grade variable.



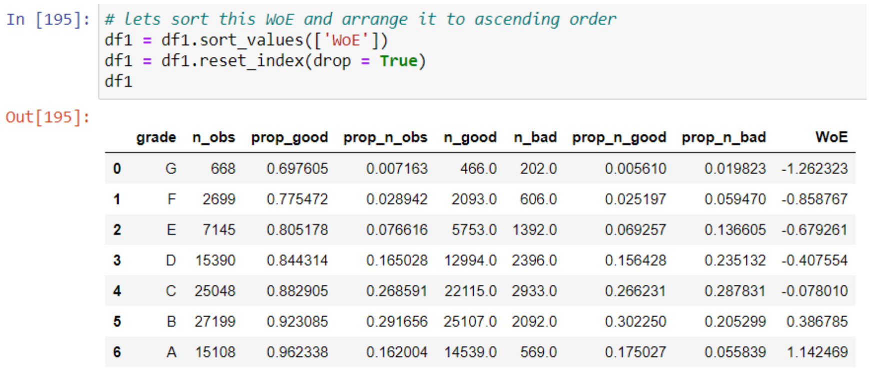
```
In [194]: # now let's calculate the WoE for this variable.
          df1['WoE'] = np.log(df1['prop_n_good'] / df1['prop_n_bad'])
          df1
```

Out[194]:

	grade	n_obs	prop_good	prop_n_obs	n_good	n_bad	prop_n_good	prop_n_bad	WoE
0	A	15108	0.962338	0.162004	14539.0	569.0	0.175027	0.055839	1.142469
1	B	27199	0.923085	0.291656	25107.0	2092.0	0.302250	0.205299	0.386785
2	C	25048	0.882905	0.268591	22115.0	2933.0	0.266231	0.287831	-0.078010
3	D	15390	0.844314	0.165028	12994.0	2396.0	0.156428	0.235132	-0.407554
4	E	7145	0.805178	0.076616	5753.0	1392.0	0.069257	0.136605	-0.679261
5	F	2699	0.775472	0.028942	2093.0	606.0	0.025197	0.059470	-0.858767
6	G	668	0.697605	0.007163	466.0	202.0	0.005610	0.019823	-1.262323

Figure 26

Arranging it in ascending order.



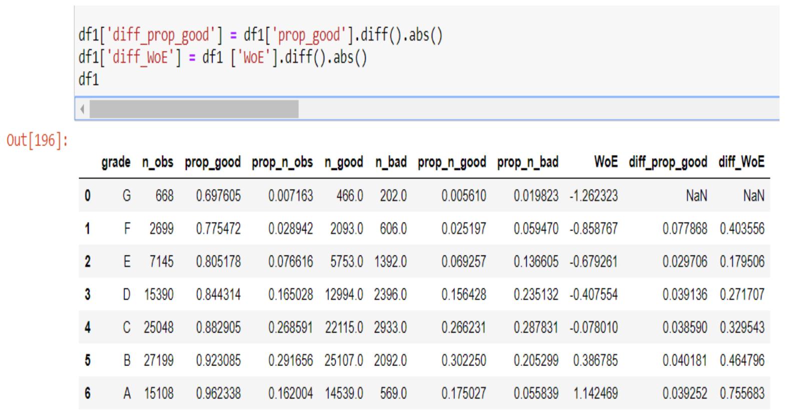
```
In [195]: # lets sort this WoE and arrange it to ascending order
          df1 = df1.sort_values(['WoE'])
          df1 = df1.reset_index(drop = True)
          df1
```

Out[195]:

	grade	n_obs	prop_good	prop_n_obs	n_good	n_bad	prop_n_good	prop_n_bad	WoE
0	G	668	0.697605	0.007163	466.0	202.0	0.005610	0.019823	-1.262323
1	F	2699	0.775472	0.028942	2093.0	606.0	0.025197	0.059470	-0.858767
2	E	7145	0.805178	0.076616	5753.0	1392.0	0.069257	0.136605	-0.679261
3	D	15390	0.844314	0.165028	12994.0	2396.0	0.156428	0.235132	-0.407554
4	C	25048	0.882905	0.268591	22115.0	2933.0	0.266231	0.287831	-0.078010
5	B	27199	0.923085	0.291656	25107.0	2092.0	0.302250	0.205299	0.386785
6	A	15108	0.962338	0.162004	14539.0	569.0	0.175027	0.055839	1.142469

Figure 27

Also calculate differences in the proportion of good loans between two subsequent categories and the difference of weight of evidence between two subsequent categories.



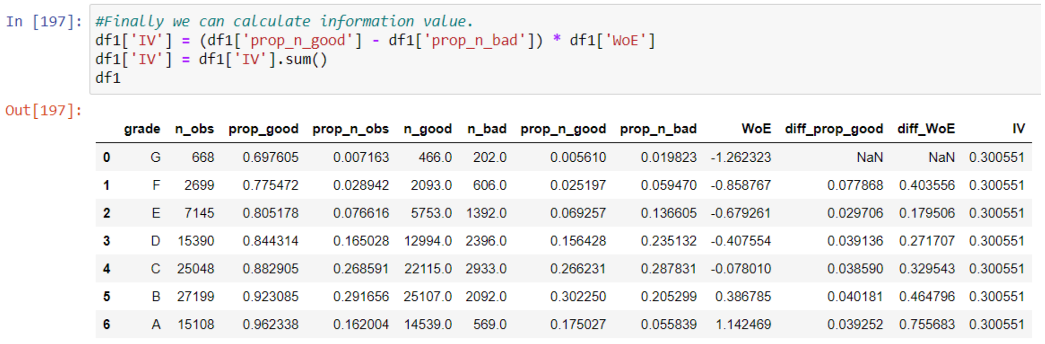
```
df1['diff_prop_good'] = df1['prop_good'].diff().abs()
df1['diff_WoE'] = df1 ['WoE'].diff().abs()
df1
```

Out[196]:

	grade	n_obs	prop_good	prop_n_obs	n_good	n_bad	prop_n_good	prop_n_bad	WoE	diff_prop_good	diff_WoE
0	G	668	0.697605	0.007163	466.0	202.0	0.005610	0.019823	-1.262323	NaN	NaN
1	F	2699	0.775472	0.028942	2093.0	606.0	0.025197	0.059470	-0.858767	0.077868	0.403556
2	E	7145	0.805178	0.076616	5753.0	1392.0	0.069257	0.136605	-0.679261	0.029706	0.179506
3	D	15390	0.844314	0.165028	12994.0	2396.0	0.156428	0.235132	-0.407554	0.039136	0.271707
4	C	25048	0.882905	0.268591	22115.0	2933.0	0.266231	0.287831	-0.078010	0.038590	0.329543
5	B	27199	0.923085	0.291656	25107.0	2092.0	0.302250	0.205299	0.386785	0.040181	0.464796
6	A	15108	0.962338	0.162004	14539.0	569.0	0.175027	0.055839	1.142469	0.039252	0.755683

Figure 28

Lastly calculating information value for this variable.



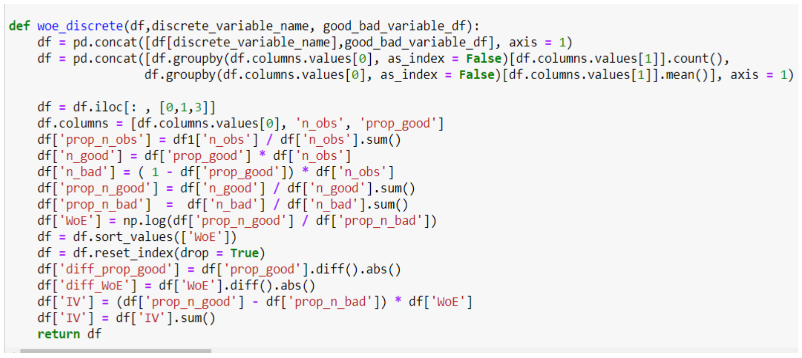
```
In [197]: #Finally we can calculate information value.
          df1['IV'] = (df1['prop_n_good'] - df1['prop_n_bad']) * df1['WoE']
          df1['IV'] = df1['IV'].sum()
          df1
```

Out[197]:

	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	668	0.697605	0.007163	466.0	202.0	0.005610	0.019823	-1.262323	NaN	NaN	0.300551
1	F	2699	0.775472	0.028942	2093.0	606.0	0.025197	0.059470	-0.858767	0.077868	0.403556	0.300551
2	E	7145	0.805178	0.076616	5753.0	1392.0	0.069257	0.136605	-0.679261	0.029706	0.179506	0.300551
3	D	15390	0.844314	0.165028	12994.0	2396.0	0.156428	0.235132	-0.407554	0.039136	0.271707	0.300551
4	C	25048	0.882905	0.268591	22115.0	2933.0	0.266231	0.287831	-0.078010	0.038590	0.329543	0.300551
5	B	27199	0.923085	0.291656	25107.0	2092.0	0.302250	0.205299	0.386785	0.040181	0.464796	0.300551
6	A	15108	0.962338	0.162004	14539.0	569.0	0.175027	0.055839	1.142469	0.039252	0.755683	0.300551

Figure 29

This same task is required to be done for all discrete variable therefore instead of repeating it, here is a code that will automate this process for all discrete variable.

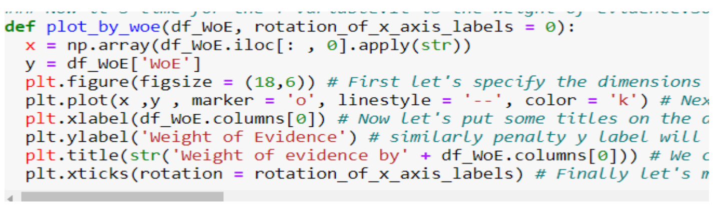


```python
def woe_discrete(df,discrete_variable_name, good_bad_variable_df):
    df = pd.concat([df[discrete_variable_name],good_bad_variable_df], axis = 1)
    df = pd.concat([df.groupby(df.columns.values[0], as_index = False)[df.columns.values[1]].count(),
                    df.groupby(df.columns.values[0], as_index = False)[df.columns.values[1]].mean()], axis = 1)

    df = df.iloc[: , [0,1,3]]
    df.columns = [df.columns.values[0], 'n_obs', 'prop_good']
    df['prop_n_obs'] = df1['n_obs'] / df['n_obs'].sum()
    df['n_good'] = df['prop_good'] * df['n_obs']
    df['n_bad'] = ( 1 - df['prop_good']) * df['n_obs']
    df['prop_n_good'] = df['n_good'] / df['n_good'].sum()
    df['prop_n_bad']  =  df['n_bad'] / df['n_bad'].sum()
    df['WoE'] = np.log(df['prop_n_good'] / df['prop_n_bad'])
    df = df.sort_values(['WoE'])
    df = df.reset_index(drop = True)
    df['diff_prop_good'] = df['prop_good'].diff().abs()
    df['diff_WoE'] = df['WoE'].diff().abs()
    df['IV'] = (df['prop_n_good'] - df['prop_n_bad']) * df['WoE']
    df['IV'] = df['IV'].sum()
    return df
```

Figure 30

Once the process is automated for obtaining WoE, it can now go ahead with the process of coarse classing. For this a plot of WoE for all the categories of original independent variable and then see which of the categories can be combined together. Those categories which have similar WoE mean that they differentiate between good and bad borrower equally (dependable variable) so they can then be combined together to form a new category. This reduces number of dummy variables in the model.

So, python library matplotlib was imported and created a function under variable name df_temp that would every time automatically plot WoE of any variable.



```python
def plot_by_woe(df_WoE, rotation_of_x_axis_labels = 0):
  x = np.array(df_WoE.iloc[: , 0].apply(str))
  y = df_WoE['WoE']
  plt.figure(figsize = (18,6)) # First let's specify the dimensions
  plt.plot(x ,y , marker = 'o', linestyle = '--', color = 'k') # Nex
  plt.xlabel(df_WoE.columns[0]) # Now let's put some titles on the c
  plt.ylabel('Weight of Evidence') # similarly penalty y label will
  plt.title(str('Weight of evidence by' + df_WoE.columns[0])) # We c
  plt.xticks(rotation = rotation_of_x_axis_labels) # Finally let's m
```

Figure 31

So, let us now go ahead and plot WoE for grade variable.

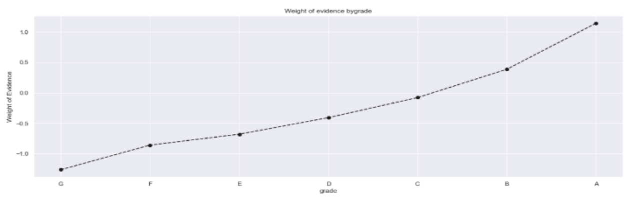
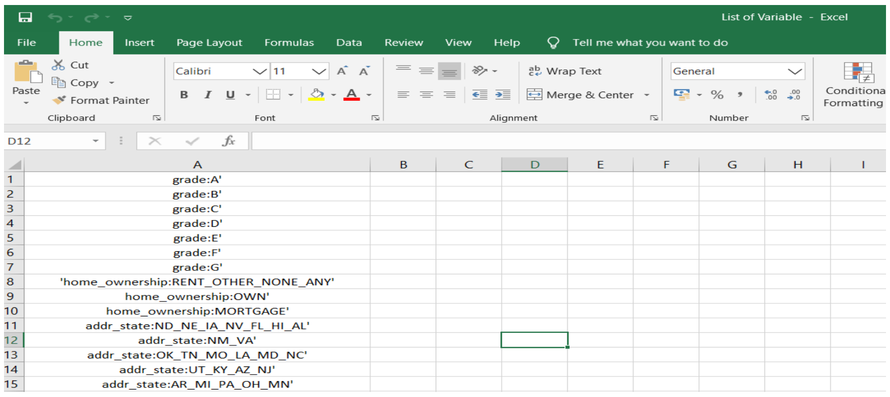


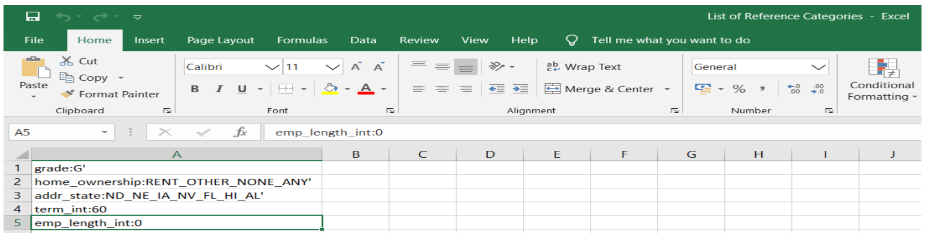
Figure 32

From the plot, it can be noted that none of the grade category seem to have similar weight of evidence, and this is correct. That is because grades are issues by external credit rating agencies and they make sure that each grade by them carry different weightage. The greater the weight, the greater is the weight. Thus, it would be wise to keep all the original seven categories of grade variable for PD model. All the final dummy categories were stored in an excel file “List of dummies”. When these categories will be used for regression, it should keep one category out as ‘Reference Category’. It is the category against which the impact of all others will be assessed. So, it has established that it will keep that category as reference category which will have worst credit risk i.e., the category with lowest Weight of evidence.



	A	B	C	D	E	F	G	H	I
1	grade:A'								
2	grade:B'								
3	grade:C'								
4	grade:D'								
5	grade:E'								
6	grade:F'								
7	grade:G'								
8	'home_ownership:RENT_OTHER_NONE_ANY'								
9	home_ownership:OWN'								
10	home_ownership:MORTGAGE'								
11	addr_state:ND_NE_IA_NV_FL_HI_AL'								
12	addr_state:NM_VA'								
13	addr_state:OK_TN_MO_LA_MD_NC'								
14	addr_state:UT_KY_AZ_NJ'								
15	addr_state:AR_MI_PA_OH_MN'								

Figure 33



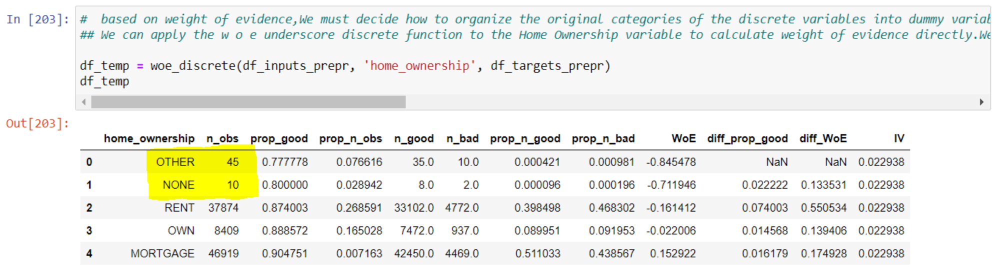
	A
1	grade:G'
2	home_ownership:RENT_OTHER_NONE_ANY'
3	addr_state:ND_NE_IA_NV_FL_HI_AL'
4	term_int:60
5	emp_length_int:0

Figure 34

Coarse Classing

Below is an example showing coarse classing is done for all the discreet and continuous variables to obtain final categories for regression.

First, weight of evidence was obtained for variable 'home ownership'.



```
In [203]: #  based on weight of evidence,We must decide how to organize the original categories of the discrete variables into dummy variat
          ## We can apply the w o e underscore discrete function to the Home Ownership variable to calculate weight of evidence directly.We

          df_temp = woe_discrete(df_inputs_prepr, 'home_ownership', df_targets_prepr)
          df_temp
```

Out[203]:

	home_ownership	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	OTHER	45	0.777778	0.076616	35.0	10.0	0.000421	0.000981	-0.845478	NaN	NaN	0.022938
1	NONE	10	0.800000	0.028942	8.0	2.0	0.000096	0.000196	-0.711946	0.022222	0.133531	0.022938
2	RENT	37874	0.874003	0.268591	33102.0	4772.0	0.398498	0.468302	-0.161412	0.074003	0.550534	0.022938
3	OWN	8409	0.888572	0.165028	7472.0	937.0	0.089951	0.091953	-0.022006	0.014568	0.139406	0.022938
4	MORTGAGE	46919	0.904751	0.007163	42450.0	4469.0	0.511033	0.438567	0.152922	0.016179	0.174928	0.022938

Figure 35

Next a plot of WoE for all its categories and see if any of them can be merged.

```
In [204]: plot_by_woe(df_temp)
```

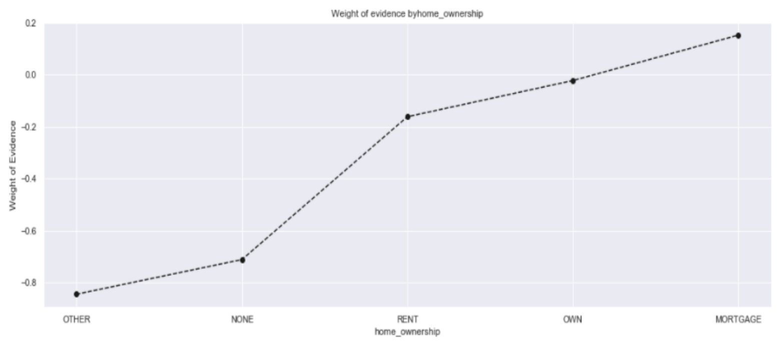
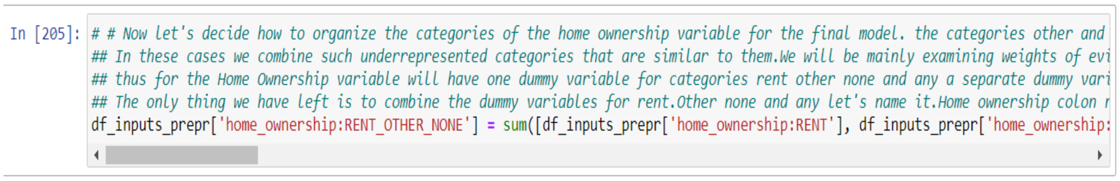


Figure 36

Deciding which categories to merge together.

- Clearly the categories "OTHERS" and "NONE" are associated with the highest probability of default.
- It is still worth looking into their proportion in the table above the graph for the total number of observations.
- From n_obs it can be seen that very few loans are associated with these categories, so it doesn't really make sense to have dummy variable for these categories. So, "other"," none" can be combined with the next riskiest category "Rent".



```
In [205]: # # Now let's decide how to organize the categories of the home ownership variable for the final model. the categories other and
          ## In these cases we combine such underrepresented categories that are similar to them.We will be mainly examining weights of evi
          ## thus for the Home Ownership variable will have one dummy variable for categories rent other none and any a separate dummy vari
          ## The only thing we have left is to combine the dummy variables for rent.Other none and any let's name it.Home ownership colon r
          df_inputs_prepr['home_ownership:RENT_OTHER_NONE'] = sum([df_inputs_prepr['home_ownership:RENT'], df_inputs_prepr['home_ownership:
```

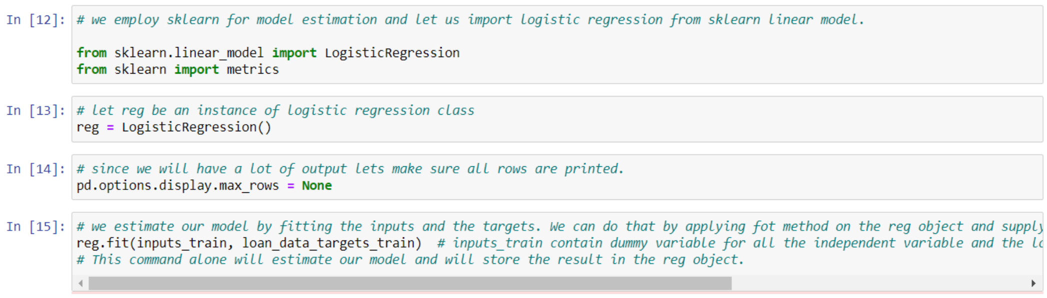
Figure 37

So, now this independent variable 'home_ownership' has 3 categories only for PD model. First is 'rent_other_none', 'own", 'mortgage'. The reference category is 'rent_other_none'.

Similar pre-processing is done for all the independent variable (discrete or continuous). All the resulting final categories are collected in a separate excel file. These will later be used as independent variable for logistic regression.

3.1.3.3 PD Model Estimation

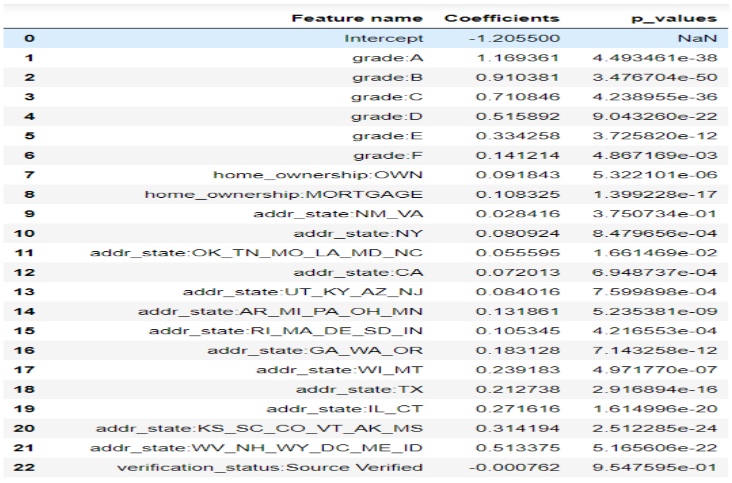
Importing logistic regression model from sklearn (library for python) and applying appropriate functions to get coefficients and p-values for independent variable.



```
In [12]: # we employ sklearn for model estimation and let us import logistic regression from sklearn linear model.

         from sklearn.linear_model import LogisticRegression
         from sklearn import metrics

In [13]: # let reg be an instance of logistic regression class
         reg = LogisticRegression()

In [14]: # since we will have a lot of output lets make sure all rows are printed.
         pd.options.display.max_rows = None

In [15]: # we estimate our model by fitting the inputs and the targets. We can do that by applying fot method on the reg object and supply
         reg.fit(inputs_train, loan_data_targets_train)  # inputs_train contain dummy variable for all the independent variable and the lo
         # This command alone will estimate our model and will store the result in the reg object.
```

Figure 38

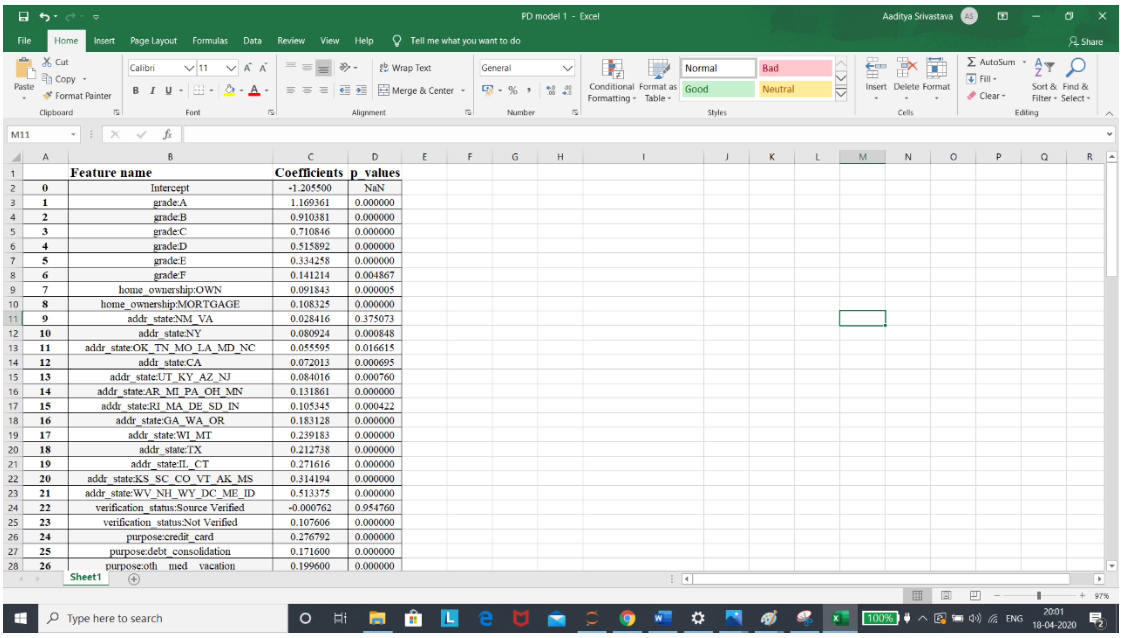
The result for above command is (this only shows 22 coefficients, however actually there were 95 in total coefficients) shown mellow.



	Feature name	Coefficients	p_values
0	Intercept	-1.205500	NaN
1	grade:A	1.169361	4.493461e-38
2	grade:B	0.910381	3.476704e-50
3	grade:C	0.710846	4.238955e-36
4	grade:D	0.515892	9.043260e-22
5	grade:E	0.334258	3.725820e-12
6	grade:F	0.141214	4.867169e-03
7	home_ownership:OWN	0.091843	5.322101e-06
8	home_ownership:MORTGAGE	0.108325	1.399228e-17
9	addr_state:NM_VA	0.028416	3.750734e-01
10	addr_state:NY	0.080924	8.479656e-04
11	addr_state:OK_TN_MO_LA_MD_NC	0.055595	1.661469e-02
12	addr_state:CA	0.072013	6.948737e-04
13	addr_state:UT_KY_AZ_NJ	0.084016	7.599898e-04
14	addr_state:AR_MI_PA_OH_MN	0.131861	5.235381e-09
15	addr_state:RI_MA_DE_SD_IN	0.105345	4.216553e-04
16	addr_state:GA_WA_OR	0.183128	7.143258e-12
17	addr_state:WI_MT	0.239183	4.971770e-07
18	addr_state:TX	0.212738	2.916894e-16
19	addr_state:IL_CT	0.271616	1.614996e-20
20	addr_state:KS_SC_CO_VT_AK_MS	0.314194	2.512285e-24
21	addr_state:WV_NH_WY_DC_ME_ID	0.513375	5.165606e-22
22	verification_status:Source Verified	-0.000762	9.547595e-01

Figure 39

The next task is to identify which of these coefficients are statistically significant (alpha = 0.5). This was done separately in a excel sheet.



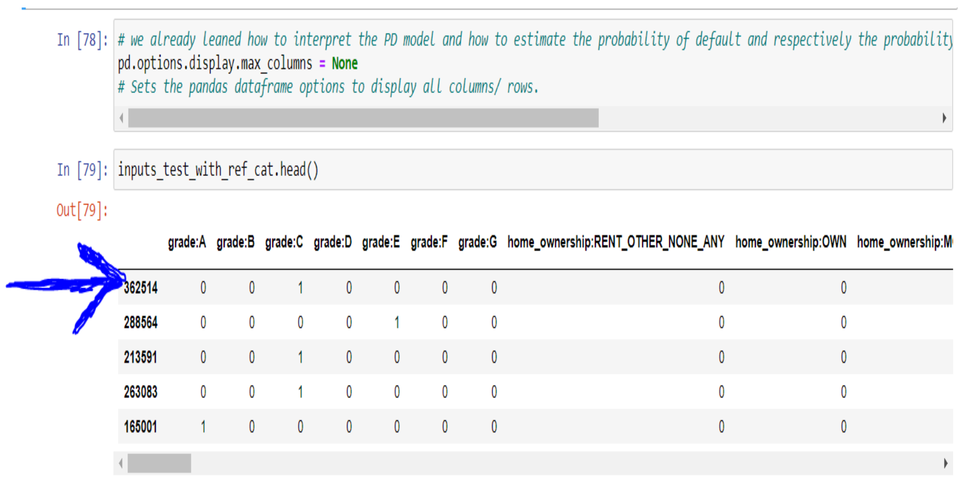
	A	B	C	D
1		Feature name	Coefficients	p_values
2	0	Intercept	-1.205500	NaN
3	1	grade:A	1.169361	0.000000
4	2	grade:B	0.910381	0.000000
5	3	grade:C	0.710846	0.000000
6	4	grade:D	0.515892	0.000000
7	5	grade:E	0.334258	0.000000
8	6	grade:F	0.141214	0.004867
9	7	home_ownership:OWN	0.091843	0.000005
10	8	home_ownership:MORTGAGE	0.108325	0.000000
11	9	addr_state:NM_VA	0.028416	0.375073
12	10	addr_state:NY	0.080924	0.000848
13	11	addr_state:OK_TN_MO_LA_MD_NC	0.055595	0.016615
14	12	addr_state:CA	0.072013	0.000695
15	13	addr_state:UT_KY_AZ_NJ	0.084016	0.000760
16	14	addr_state:AR_MI_PA_OH_MN	0.131861	0.000000
17	15	addr_state:RI_MA_DE_SD_IN	0.105345	0.000422
18	16	addr_state:GA_WA_OR	0.183128	0.000000
19	17	addr_state:WI_MT	0.239183	0.000000
20	18	addr_state:TX	0.212738	0.000000
21	19	addr_state:IL_CT	0.271616	0.000000
22	20	addr_state:KS_SC_CO_VT_AK_MS	0.314194	0.000000
23	21	addr_state:WV_NH_WY_DC_ME_ID	0.513375	0.000000
24	22	verification_status:Source Verified	-0.000762	0.954760
25	23	verification_status:Not Verified	0.107606	0.000000
26	24	purpose:credit_card	0.276792	0.000000
27	25	purpose:debt_consolidation	0.171600	0.000000
28	26	purpose:oth_med_vacation	0.199600	0.000000

Figure 40

The criteria followed to select the significant variable was that if any category of original independent proved to be significant then that entire variable was taken along with all of its categories. Even if other categories would be insignificant. So, in this way there are all significant variables along with their coefficients.

3.3.3.4. Calculating Probability of Default

First, multiply the values of a borrower on the independent dummy variables by the respective model coefficients. So, when an exponent is raised on this result, it equals the odds for being good versus bad. From there it can easily estimate probability of being good. Now each observation from all the dummy categories of the original independent variable can only have a value of 1 for one of the dummy variables, the rest are always zero. So, calculating the power on which the exponents should be raised to obtain the odds boil down to the following to summing the regression coefficients for all dummy variables to which an observation belongs. Below is how it's done using a practical example in python. Taking the first observation from this dataset. Its index is 362514.



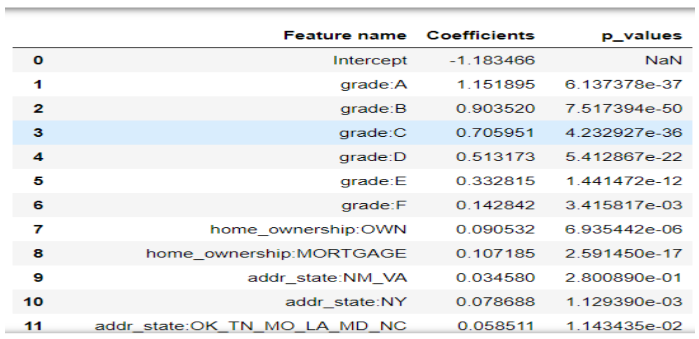
```
In [78]: # we already leaned how to interpret the PD model and how to estimate the probability of default and respectively the probability
         pd.options.display.max_columns = None
         # Sets the pandas dataframe options to display all columns/ rows.
```

```
In [79]: inputs_test_with_ref_cat.head()
```

Out[79]:

	grade:A	grade:B	grade:C	grade:D	grade:E	grade:F	grade:G	home_ownership:RENT_OTHER_NONE_ANY	home_ownership:OWN	home_ownership:M
362514	0	0	1	0	0	0	0	0	0	
288564	0	0	0	0	1	0	0	0	0	
213591	0	0	1	0	0	0	0	0	0	
263083	0	0	1	0	0	0	0	0	0	
165001	1	0	0	0	0	0	0	0	0	

Figure 41



	Feature name	Coefficients	p_values
0	Intercept	-1.183466	NaN
1	grade:A	1.151895	6.137378e-37
2	grade:B	0.903520	7.517394e-50
3	grade:C	0.705951	4.232927e-36
4	grade:D	0.513173	5.412867e-22
5	grade:E	0.332815	1.441472e-12
6	grade:F	0.142842	3.415817e-03
7	home_ownership:OWN	0.090532	6.935442e-06
8	home_ownership:MORTGAGE	0.107185	2.591450e-17
9	addr_state:NM_VA	0.034580	2.800890e-01
10	addr_state:NY	0.078688	1.129390e-03
11	addr_state:OK_TN_MO_LA_MD_NC	0.058511	1.143435e-02

First, taking the intercept = -1.183466; then the external grade of this observation is 'C' as for this observation it is taken dummy = 1 at grade C. Adding its respective coefficient to intercept. And similarly, coefficients of all dummies will be added that have observation of 1. So, the final summation that it gets = -1.183466 + 0.705951+ 0.107185+ 0.074778+0.001095+ 0.259357+ 0.054842+ 0.076765+ 0.095389 = 0.191896.
This value of 0.191896 are log odds. Ln((1-PD)/PD) = 0.191896.
Getting rid of the logarithm by raising the exponents to a power.
(1 – PD) / PD = exp (0.191896) = 1.211544

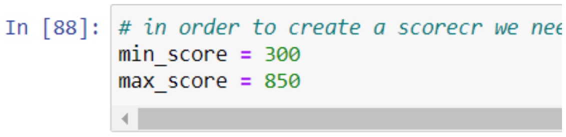
Therefore, the estimated probability of being good borrower is equal to 1.211544 / (1+1.211544) = .547827.
Therefore, the probability that this person will not default is 54.78 %.

3.2 Methodology for Objective 2

As mentioned previously, that PD model must be easy to understand and interpret. Even people having no understanding of statistical analysis should be able to understand it. Thus, keeping that in mind it has converted the PD model into a scorecard that can easily be understood by anyone.

A scorecard tool produces individual credit worthiness assessment that directly corresponds to a specific probability of default. As these credit worthiness assessments are named after the scorecard. They are thus called Credit scores. Thus, it will create a scorecard based on our PD model. Summary table contains all the coefficients of the PD model arranged as a data frame

Assigning minimum and maximum values for scorecard is required. In this project it has taken minimum score = 300 and maximum score = 850. It then went ahead to store these values in two different variables.

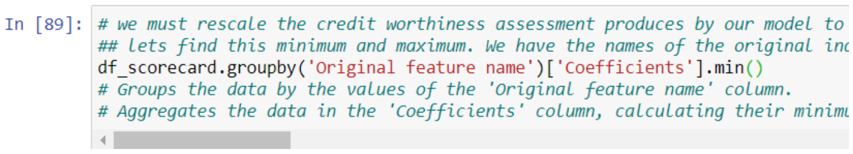


```
In [88]: # in order to create a scorecr we nee
         min_score = 300
         max_score = 850
```

Figure 42

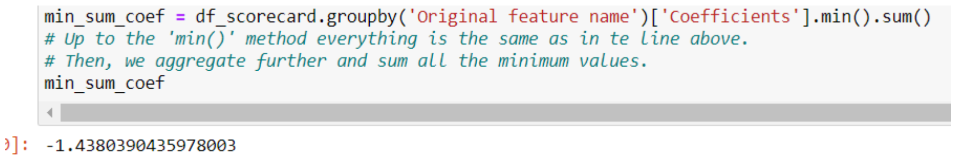
Next is to rescale these credit worthiness assessments is terms of probabilities to the credit score range. To achieve that, apart from the range of scorecard it also needs highest and lowest of the credit worthiness that PD model can estimate.

Theoretically, the lowest credit worthiness calculation it can get from the PD model would be in the case if borrower fall into all those categories of original independent variable with the lowest model coefficients. Similarly, the maximum credit worthiness assessment it can get from the PD model would be in the case where a borrower falls into a category of original independent variables with highest model coefficients. So, let us first find this minimum and maximum.



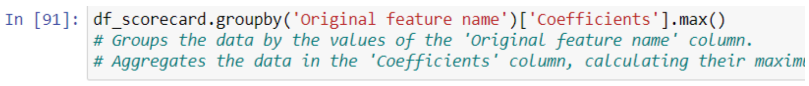
```
In [89]: # we must rescale the credit worthiness assessment produces by our model to
         ## lets find this minimum and maximum. We have the names of the original inc
         df_scorecard.groupby('Original feature name')['Coefficients'].min()
         # Groups the data by the values of the 'Original feature name' column.
         # Aggregates the data in the 'Coefficients' column, calculating their minimu
```

Figure 43



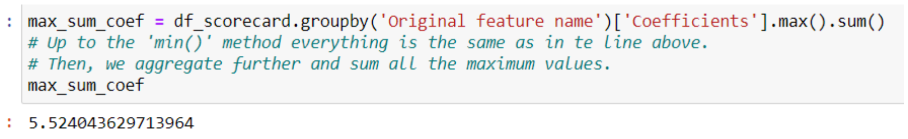
```
    min_sum_coef = df_scorecard.groupby('Original feature name')['Coefficients'].min().sum()
    # Up to the 'min()' method everything is the same as in te line above.
    # Then, we aggregate further and sum all the minimum values.
    min_sum_coef

]:  -1.4380390435978003
```

Figure 44



```
In [91]: df_scorecard.groupby('Original feature name')['Coefficients'].max()
         # Groups the data by the values of the 'Original feature name' column.
         # Aggregates the data in the 'Coefficients' column, calculating their maxim
```

Figure 45



```
: max_sum_coef = df_scorecard.groupby('Original feature name')['Coefficients'].max().sum()
  # Up to the 'min()' method everything is the same as in te line above.
  # Then, we aggregate further and sum all the maximum values.
  max_sum_coef

: 5.524043629713964
```

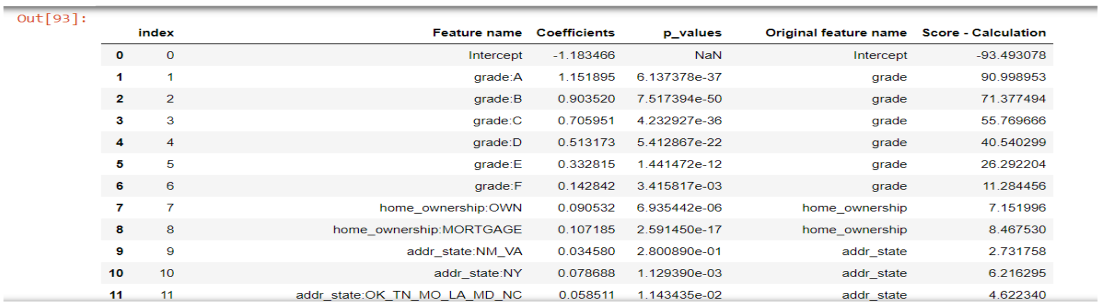
Figure 46

Now the question is how do it rescale coefficients to score? For converting each dummy variable coefficient to a score, it has to multiply each coefficient by ratio of the difference between the maximum score and minimum desired score to the difference between the maximum sum of coefficients and the minimum sum of coefficients.

$$variable\_score = variable\_coef\,\frac{(max\_score - min\_score)}{(max\_sum\_coef - min\_sum\_coef)}$$

Figure 47

Let’s do this for all regression coefficients and store the result in score-calculation variable.



Out[93]:

	index	Feature name	Coefficients	p_values	Original feature name	Score - Calculation
0	0	Intercept	-1.183466	NaN	Intercept	-93.493078
1	1	grade:A	1.151895	6.137378e-37	grade	90.998953
2	2	grade:B	0.903520	7.517394e-50	grade	71.377494
3	3	grade:C	0.705951	4.232927e-36	grade	55.769666
4	4	grade:D	0.513173	5.412867e-22	grade	40.540299
5	5	grade:E	0.332815	1.441472e-12	grade	26.292204
6	6	grade:F	0.142842	3.415817e-03	grade	11.284456
7	7	home_ownership:OWN	0.090532	6.935442e-06	home_ownership	7.151996
8	8	home_ownership:MORTGAGE	0.107185	2.591450e-17	home_ownership	8.467530
9	9	addr_state:NM_VA	0.034580	2.800890e-01	addr_state	2.731758
10	10	addr_state:NY	0.078688	1.129390e-03	addr_state	6.216295
11	11	addr_state:OK_TN_MO_LA_MD_NC	0.058511	1.143435e-02	addr_state	4.622340

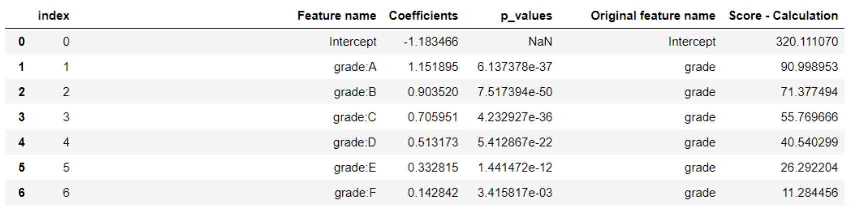
Figure 48

Intercept seems to have an unusual value. This is because the intercept is not a dummy variable, it is an integral part for calculation of credit worthiness assessment. In fact, the score reflecting the intercept is very close to the lowest score an observation would get in worst credit rating wise.

So, it wishes to replace the current score of intercepts with the minimum desired score which is 300.

$$intercept\_score = \frac{(intercept\_coef - \min\_score)}{(\max\_sum\_coef - \min\_sum\_coef)}(\max\_score - \min\_score) + \min\_score$$

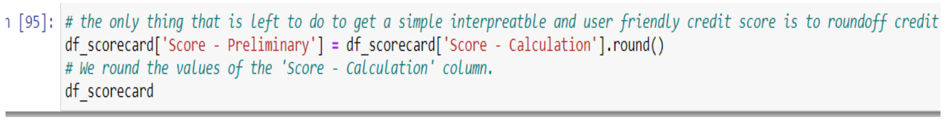
Figure 49



	index	Feature name	Coefficients	p_values	Original feature name	Score - Calculation
0	0	Intercept	-1.183466	NaN	Intercept	320.111070
1	1	grade:A	1.151895	6.137378e-37	grade	90.998953
2	2	grade:B	0.903520	7.517394e-50	grade	71.377494
3	3	grade:C	0.705951	4.232927e-36	grade	55.769666
4	4	grade:D	0.513173	5.412867e-22	grade	40.540299
5	5	grade:E	0.332815	1.441472e-12	grade	26.292204
6	6	grade:F	0.142842	3.415817e-03	grade	11.284456

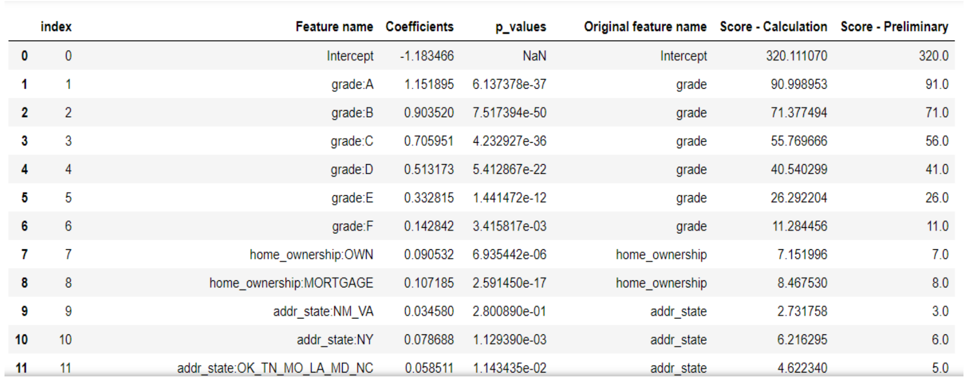
Figure 50

So, the only thing left is to round off the scores to the nearest figure.



```
[95]: # the only thing that is left to do to get a simple interpreatble and user friendly credit score is to roundoff credit
      df_scorecard['Score - Preliminary'] = df_scorecard['Score - Calculation'].round()
      # We round the values of the 'Score - Calculation' column.
      df_scorecard
```

Figure 51



	index	Feature name	Coefficients	p_values	Original feature name	Score - Calculation	Score - Preliminary
0	0	Intercept	-1.183466	NaN	Intercept	320.111070	320.0
1	1	grade:A	1.151895	6.137378e-37	grade	90.998953	91.0
2	2	grade:B	0.903520	7.517394e-50	grade	71.377494	71.0
3	3	grade:C	0.705951	4.232927e-36	grade	55.769666	56.0
4	4	grade:D	0.513173	5.412867e-22	grade	40.540299	41.0
5	5	grade:E	0.332815	1.441472e-12	grade	26.292204	26.0
6	6	grade:F	0.142842	3.415817e-03	grade	11.284456	11.0
7	7	home_ownership:OWN	0.090532	6.935442e-06	home_ownership	7.151996	7.0
8	8	home_ownership:MORTGAGE	0.107185	2.591450e-17	home_ownership	8.467530	8.0
9	9	addr_state:NM_VA	0.034580	2.800890e-01	addr_state	2.731758	3.0
10	10	addr_state:NY	0.078688	1.129390e-03	addr_state	6.216295	6.0
11	11	addr_state:OK_TN_MO_LA_MD_NC	0.058511	1.143435e-02	addr_state	4.622340	5.0

Figure 52

4. Findings

Objective 1

- An analytical approach using logistic regression model that helps in estimating the probability of default associated with any borrower.

Objective 2

- Successfully converted the probability of default model for first objective into a scorecard. This would enable easier interpretability of the probability of default model.

5. Conclusion

- Models based on analytical techniques such as logistic regression prove to be a viable alternative all the theories discussed in literature review. It provides more precise and realistic approach for estimating probability of default. Building a scorecard has indeed provided with better representation of this probability.
- This model will now enable Peacock solar to thoroughly examine the applicant homeowner who want solar on finance and then take a decision if he should be given solar on credit based on his credit score.

6. Limitation

The data used for creating the model was a banking industry data, there was no historical data available for solar industry. Therefore, banking industry data was used a proxy for solar industry data.

The dataset that has been used is American dataset. Ideally as this project is for Indian company for business in India, Indian data set should have been used. But, as no previous data was available for India, it used American dataset.

7. Managerial Implications

Probability of default model will help managers to make better decisions before deciding to give out credit. This will enhance decision making and will certainly bring down number of defaults, thereby helping them to enhance the profitability of the company A wider analysis of the relevant factors helps boost confidence in managers while they make efficient decisions for the company. A scorecard helps them to increase the interpretability of the model thereby contributing to easy decision making.

8. References

1. Fischer Black and Myron Scholes, 1973 *The Pricing of Options and Corporate Liabilities* [online] Chicago, Chicago journals
2. Available from:
3. https://www.cs.princeton.edu/courses/archive/fall09/cos323/papers/black_scholes73.pdf [Accessed 15-4-2020]
4. Jorion (2006), *Good and Bad Credit Contagion: Evidence from Credit Default Swaps* [Online] Edition- 2006 California
5. Available from:
6. http://business.umsl.edu/files/pdfs/Zhang_Publication/JFE_CreditContagion.pdf [Accessed 18-4-2020]
7. Jarrow and Turnbull (1992), *Structural versus reduced form models: a new information-based perspective* [online], Journal of investment management.
8. Available from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.139.3546&rep=rep1&type=pdf [Accessed 17-4-2020]
9. Joy Begley, Jin Ming, Susan Watts., 1997 *Bankruptcy Classification Errors in the 1980s: An Empirical Analysis of Altman's and Ohlson's Models* [online] Bloomington
10. Available from: https://www.researchgate.net/profile/Joy_Begley/publication/249853617_Bankruptcy_Classification_Errors_in_the_1980s_An_Empirical_Analysis_of_Altman_and_Ohlson's_Models/links/56e36ba908ae68afa10caad5.pdf [Accessed 14-4-2020]