PREDICTIVE MODELING FOR BANK CUSTOMER

Classifying a current or potential customer as a good or a bad risk is an example of **classification** problem. Compared to regression problem, the label is a categorical variable in the classification problem.

In visualization for classification problem, as in regression, **colinear** features should be identified so that they can be eliminated or otherwise dealt with. However, we are also looking for features that help **separate the label categories.** Separation is achieved when there are distinctive feature values for each label category. A good separation results in low classification error rate.

About the Data:

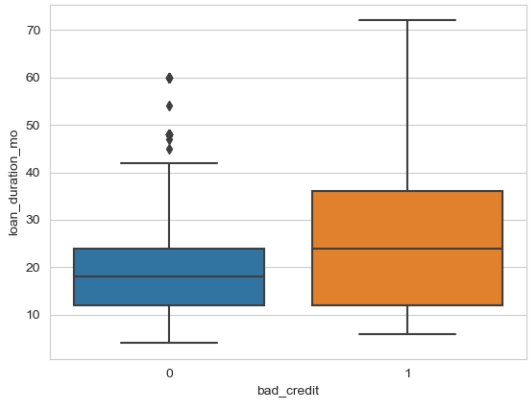
The data sample is from a German bank on its customers. The code loads the dataset and assigns human-readable names to the columns. Of the 21 columns excluding 'customer_id', the 20 are the features representing the information (numeric and categorical) a bank might have on its customers. The label column (bad_credit) is in binary. The categorical features are also coded in unreadable way, thus, will be recoded to make sense of the information.

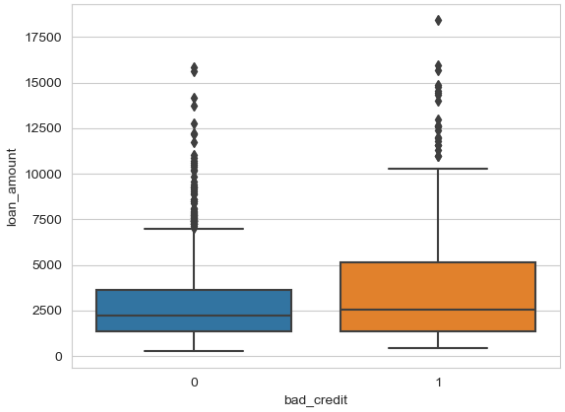
The data has only 30% of the cases being bad credit showing clear data imbalance. It is enough to bias the training of any model.

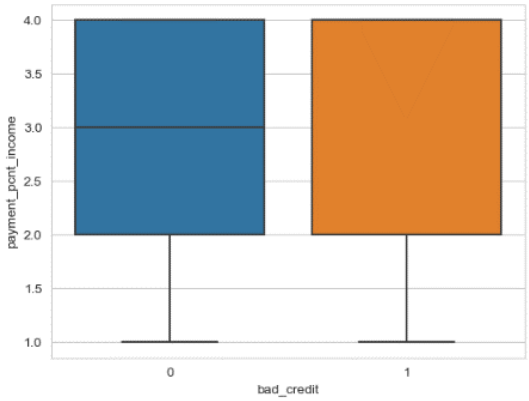
Visualizing the class by Numeric Features:

Using the boxplot, we get the following plots for the numeric features.

Box Plots are useful since it **helps us focus on the overlap of the quartiles of the distribution** to check for feature importance. In this case, we see enough differences in the quartiles for the feature to be useful in separation of the label case.

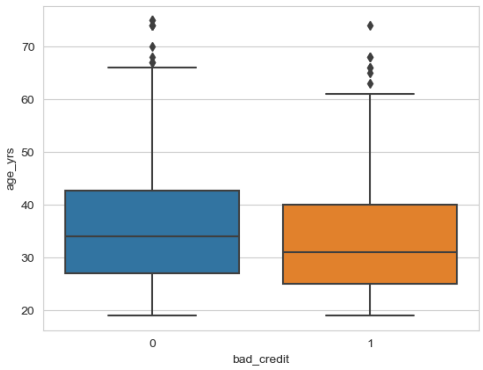


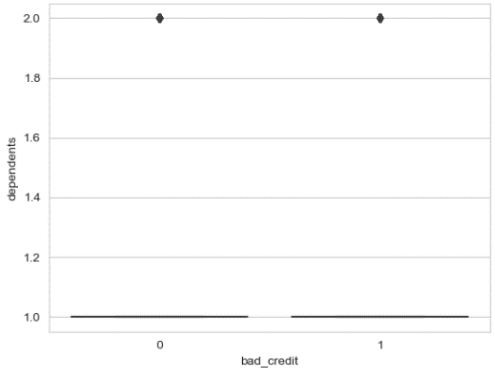


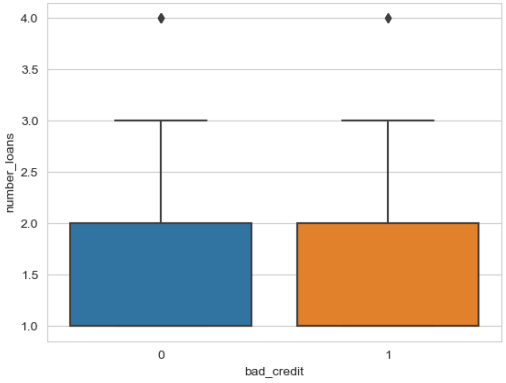


1. For loan_duration_mo, loan_amount, and payment as percent of income, there is a clear separation between good and bad credit customers. As expected, bad credit customers have longer duration on loans, take bigger loans and have a bigger portion of their income as payment for the loan.

   The Violin plot for the first two show the difference for the more extreme values. These features may be less important than as indicated by box plot (plots in the code).





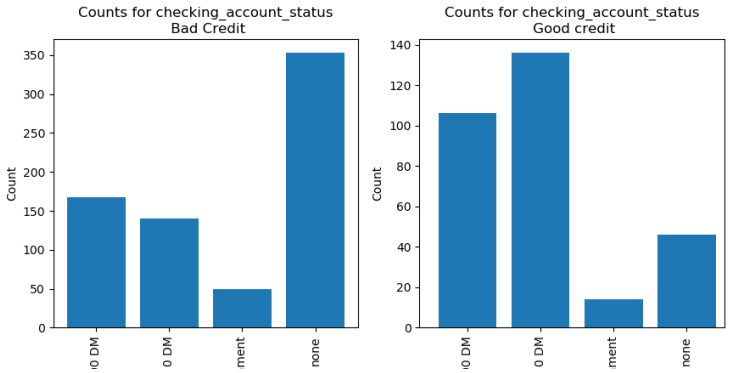


2. On the other hand, the age of the customer, number of loans taken and number of dependents seem to have less predictive importance on having a bad credit. The latter two cases seem to result from the median value being zero as there are not enough non-zero cases, thus making these features less useful.

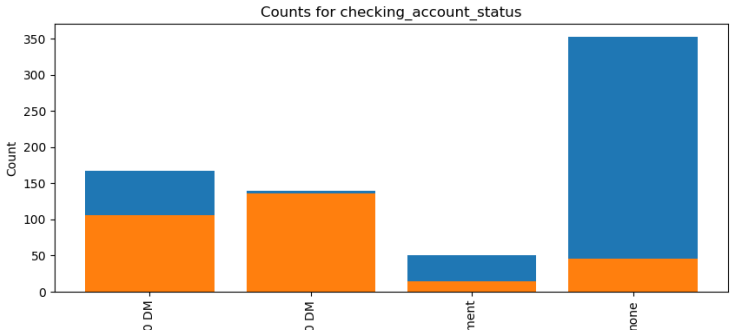
Visualizing the class separation by the categorical features:

Ideally, a categorical feature will have very different counts of the categories for each of the label values. So, a bar plot is a good tool to use here.

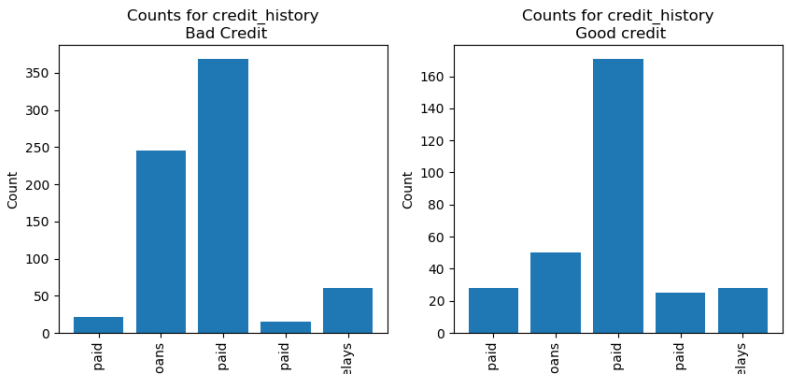
1. Features such as checking_account_status and credit_history show significant difference in distribution between the label categories. Other include Property and likely Purpose



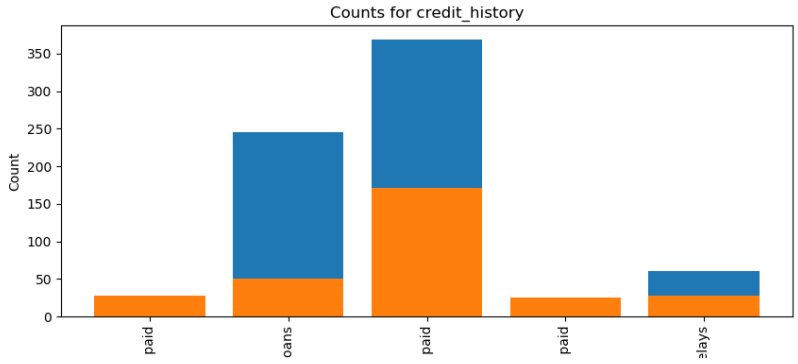
For easier visualization, another bar plot for check_account_status is:



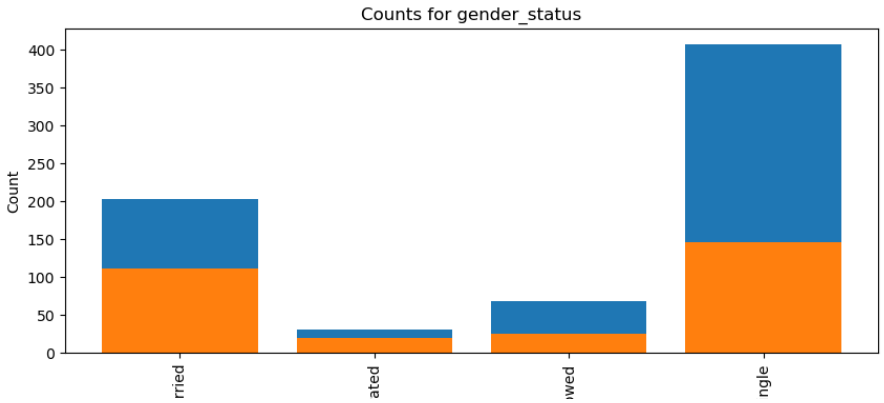
The label categories are superimposed on each other and present more detail. Blue bar referring to bad credit customers.

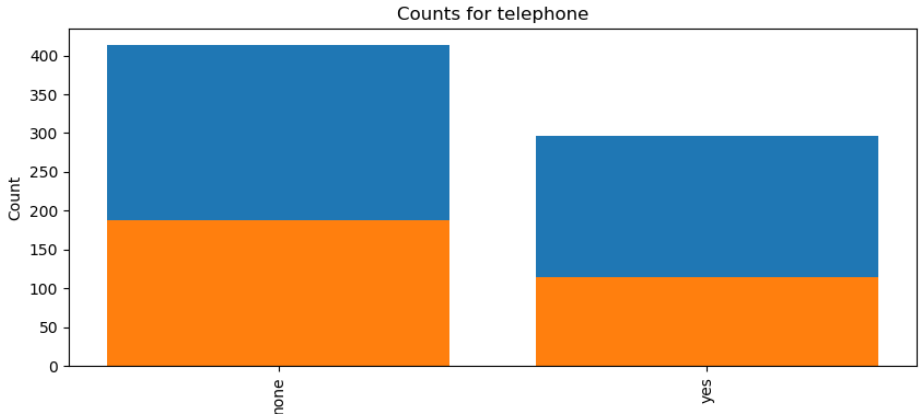


Similarly,

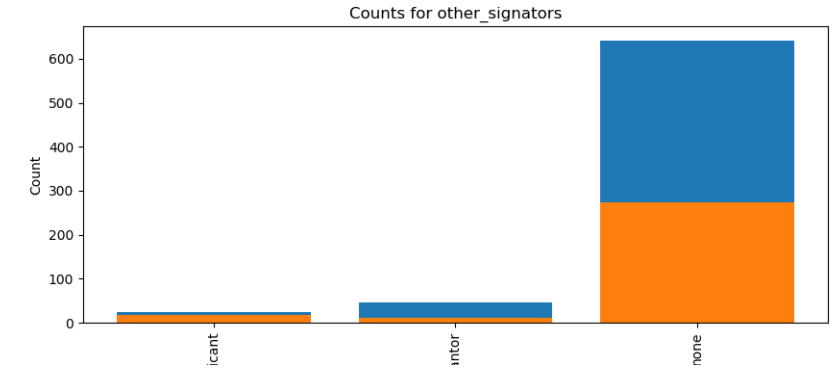


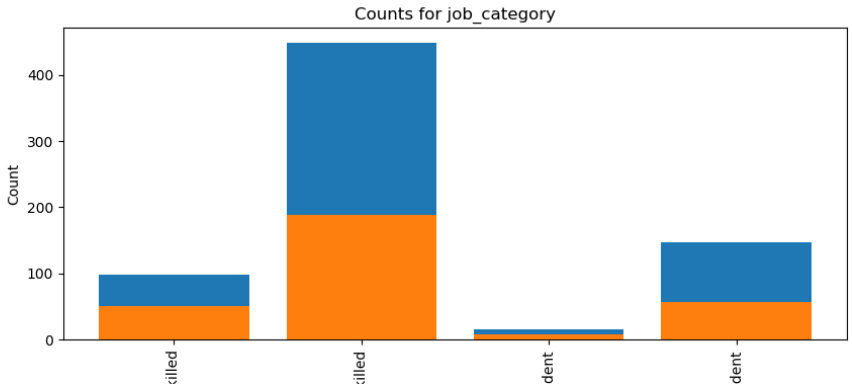
2. Features such as gender_status and telephone show small difference and are unlikely to be significant.

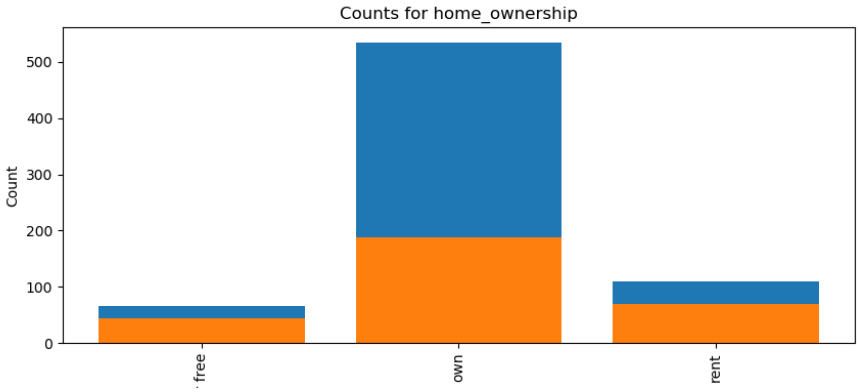


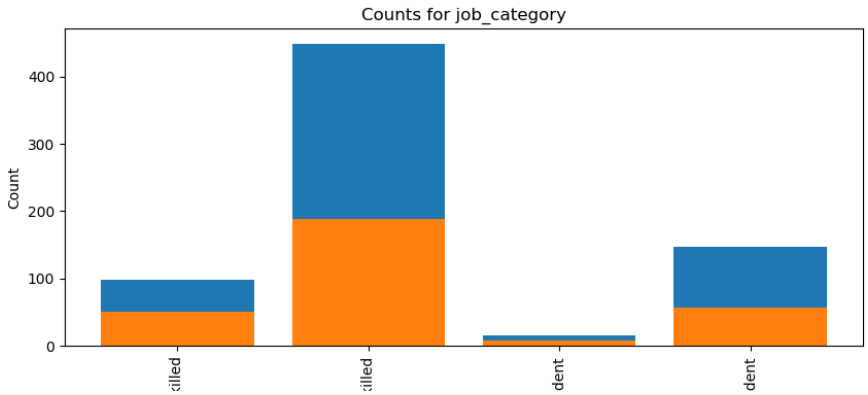


3. Features like other_signators, foreign_worker home_ownership, and job_category have a dominant category with very few cases of other categories and will likely be very little power to separate the cases. Other features include Savings_balance, other_oustanding_credit.









Thus, only a few of these categorical features will be useful in separating the cases.

Data Preparation:

Some of the important steps are:

- Recode character strings to eliminate characters that will not be processed correctly.
- Find and treat missing values.
- Set correct data type of each column.
- Transform categorical features to create categories with more cases and coding likely to be useful in predicting the label.
- Apply transformations to numeric features and the label to improve the distribution properties.
- Locate and treat duplicate cases.

Removing Duplicate rows:

It can seriously bias the machine learning models as they add undue weights to the models. Having columns with values guaranteed to be unique can be used to detect and remove duplicates, here, it is 'customer_id'.

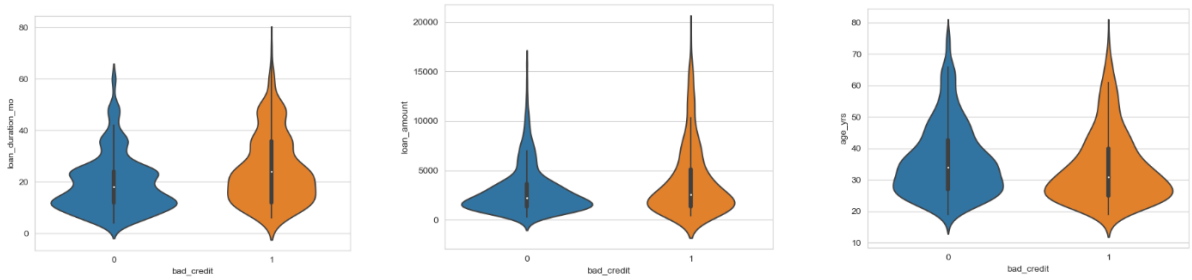
In considering which case to remove, if the duplicates have different dates of creation, newest date often selected, or else the choice is often arbitrary.

In the German credit dataset, there were 12 duplicate cases found. Here, the first instance will be kept and the prepped data saved for later use.

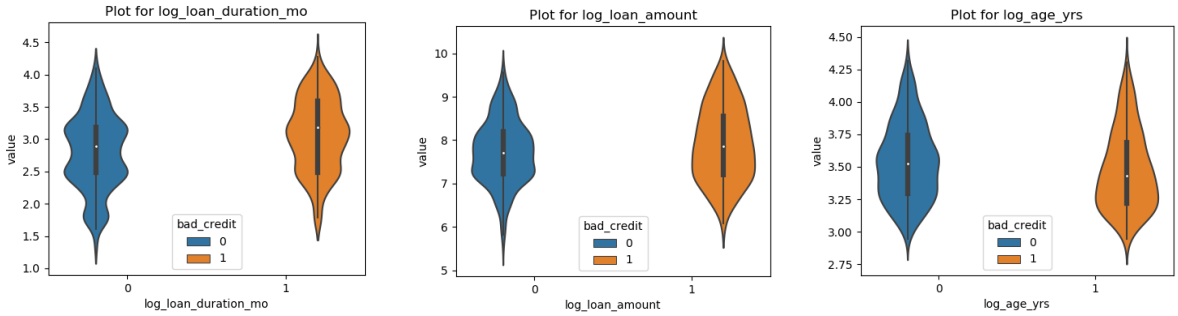
Feature Engineering:

This is to determine if any improvement in predictive power can be expected. As seen before, several numeric features had a ***strong left skew***. So, a log transformation might help in case like this, and use Padas 'applymap' method.

The features are: loan_duration_mo, loan_amount, and age_yrs



Though the log tranformation makes distribution more symmetric, it does not show any improvement in the label cases as seen below. So, these features will not be used further.



Note: Recalling the visualization of the categorical features, there are a few categories with few cases. However, it is not clear how these can be reasonably combined. It may just be the case that some of these categorical features are not terribly predictive.

Applying Classification:

The data is prepared to create numpy arrays to use scikit-learn model. This is a three step process:

a. Encode the categorical string variables as integers.
b. Transform the integer coded variables to dummy variables.
c. Append each dummy coded categorical variable to the model matrix.

Then, the numeric features are concatenated to the numpy array before splitting the data into 'Training' and 'Test' dataset.

In this process:

a. An index vector is Bernoulli sampled using the train_test_split function from the model_selection package of scikit-learn.
b. The first column of the resulting index array contains the indices of the samples for the training cases.
c. The second column of the resulting index array contains the indices of the samples for the test cases.

Constructing the Logistic Regression Model:

Using the LogisticRegression method from the scikit-learn linear_model package, the probabilities for score was obtained. The two columns obtained are for score of 0 and 1 respectively, and in most cases probability of a score of 0 is higher than 1.

Recall that the log likelihood for the two-class logistic regression are computed by applying sigmoid or logistic transformation to the output of the linear model. The threshold between two likelihoods is set to '0.5'.

Evaluation of Score and the classification model:

Confusion Matrix

	Score positive	Score negative
Actual positive	182	30
Actual negative	39	49

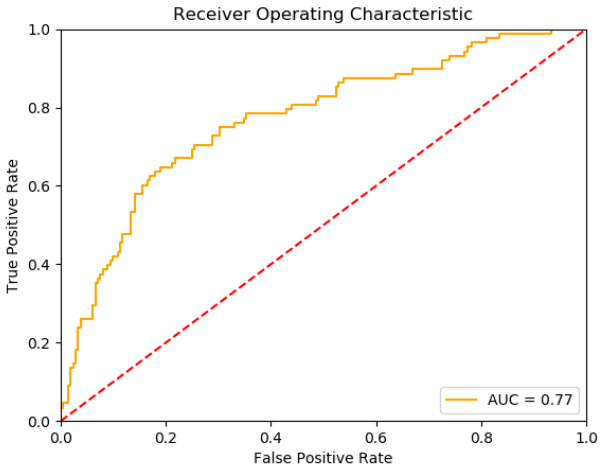
Accuracy 0.77

	Positive	Negative
Num case	212	88
Precision	0.82	0.62
Recall	0.86	0.56
F1	0.84	0.59

Results:

a. The confusion matrix shows: i) mostly positive cases being correctly classified, 182 vs. 30
   ii) many negative cases scored incorrectly with only 49 correct and 39 incorrect.
b. Overall accuracy is 0.77, but this metric is very misleading. As the negative cases were poorly classified. In fact, it is the bad credit customers that the banks care most about. So, Accuracy should be taken with a healthy skepticism.
c. The class imbalance is confirmed. Of the 300 test cases, 212 are positive and 88 are negative.
d. Precision, Recall and F1 indicate the correct classification of the positive cases but not the negative cases.

Calculating **ROC** curve and **AUC**:



The AUC obtained is 0.77, but given the class imbalance of the two cases of the label column, additional step of computing a **weighted model** is needed for more reliability. It is important to note that ***"a falsely classified bad credit risk customer as a good one cost the bank FIVE times more than classifying a good one as a bad."*** Thus, what we seek is better prediction of bad credit risk customers.

Assigning weight to the classes addresses the class imbalance. In this case, weights of 0.3 and 0.7 are chosen for good and bad credit respectively. Comparing the performance with earlier case:

For threshold = 0.5				For threshold = 0.5		
Confusion Matrix (**Unweighted**)				Confusion Matrix (**Weighted**)		
	Score positive	Score negative			Score positive	Score negative
Actual positive	182	30		Actual positive	141	71
Actual negative	39	49		Actual negative	21	67
Accuracy	0.77			Accuracy	0.69	
	Positive	Negative			Positive	Negative
Num case	212	88		Num case	212	88
Precision	0.82	0.62		Precision	0.87	0.49
Recall	0.86	0.56		Recall	0.67	0.76
F1	0.84	0.59		F1	0.75	0.59

Results:

1. Although, the classification for the positive cases has dropped, as mentioned earlier, the bank is most concerned about the bad credit customer. As observed, the classification of the bad credit customers has significantly improved with the weighted model.

2. Recall is the metric for the positive case, here 'bad credit' is our positive case. It has significantly improved as well. F1 is constant while Precision and Accuracy have dropped.

A Better Threshold:

The score is determined by setting the threshold along the ***sigmoidal or logistic function***. It is possible to favor either positive or negative cases by changing the threshold along the curve. The model is tested for thresholds (0.45, 0.4, 0.35, 0.3, 0.25).

For 0.4 and 0.35, the model improves on its classification of bad credit customer without severely misclassifying good customers as well. The final choice is at the bank's discretion.

For threshold = 0.4				For threshold = 0.35		
Confusion Matrix				Confusion Matrix		
	Score positive	Score negative			Score positive	Score negative
Actual positive	117	95		Actual positive	106	106
Actual negative	18	70		Actual negative	15	73
Accuracy	0.62			Accuracy	0.6	
	Positive	Negative			Positive	Negative
Num case	212	88		Num case	212	88
Precision	0.87	0.42		Precision	0.88	0.41
Recall	0.55	0.8		Recall	0.5	0.83
F1	0.67	0.55		F1	0.64	0.55

Reference:

[1] https://courses.edx.org/courses/course-v1:Microsoft+DAT276x+2T2018/course/

NOTE:

I prepared this note/report while I took DAT276x course for R and Python to better understand the lectures. I made some minor modification for better clarification and my preference while preparing this summary for “Predictive Model for Bank Customers.”