# Credit Risk Analysis Report

## Summary

The purpose of this analysis is to predict if a loan will get repaid or will be defaulted. This is analysed using a logistic regression (LR) model. Since LR is a classification model, its outcome is binary; and hence it is a good model to use for such kind of analysis. It fits the input data into a sigmoid curve, resulting in the scores to be either a class 0 (loan will be repaid) or class 1 (loan will be defaulted). Our model must predict the two classes correctly in order to be considered a good model.

The model gets a high accuracy score of 99%. But on closer analysis, it performs poorly in predicting healthy and risky loans. The model could predict if a loan was healthy but could not predict if it was risky with the same rigour. The model predicted a loan to be risky, it was correct only 87% of the time, i.e. of all the loans that the model predicted were risky, only 87% were actually risky.

And out of all the loans that were actually risky, the model predicted only 89% of them to be risky. The model was not able to identify 11% of risky loans at all.

The model fared well in accuracy because the data we fed it is imbalanced. Based on the support section of the classification report, we can see that healthy loans were 18759 whereas risky loans in the dataset were only 625. So, it was able to predict the healthy loans but failed to predict the 1 (high risk loans) in all cases.

Based on these scores, I would not recommend the use of this model to the company.

## Context

The current analysis is that of a financial dataset with loan status. The task includes analysing the data and predicting if a loan in the future is a high-risk one or healthy one. The approach to predicting this includes analysing various aspects (called features) of previous loans and deriving any relationship between these features and the loan status. If a relationship exists between the features and the status of the loan, we can predict the loan outcome of future loans based on the features data of that loan.

In this particular analysis, the dataset consists of more than 77,500 loans and their outcomes, along with following 7 aspects (features) for each loan: loan size, the interest rate, income of borrower, the total debt, number of bank accounts, any derogatory marks and the total debt.

## Model Used

The prediction is done by using a logistic regression. A logistic regression is characterised by having a binary outcome i.e. the probability of an outcome is mostly 1 or 0, but very rarely 0.5. This is due to the sigmoid shape of the logistic regression. The model fits the data into this sigmoid curve, making the outcome land in either the lower half of the curve or the upper half but very rarely in the central part which is a steep line.

### Step1: Preparing the Data

Before we can input the data into the model, we must prepare it for analysis. This includes, importing the .csv into python, reading it and then using pandas to convert it to a pandas dataframe. Next we need to ensure the data is numerical. All categorical data is first encoded into numerical values. In this analysis, the data was already in numerical format, as gauged by the dtypes function. Thus, we did not have to encode the data in this particular case.

### Step 2: Splitting the Data

Next, we need to split the dataframe into features and labels. Features are the various aspects of loans as mentioned earlier, while the labels are the outcomes.

### Step 3: Testing and Training sets

Once the data is split into features and labels, it has to be further split into testing and training sets. Logistics regression is a supervised machine learning (ML) method. For all supervised ML methods, the algorithm has to be trained with data whose outcome is already known to us. Typically, 75% of the data is used for train the model, while 25% is used to test the model, to check if the model indeed works.

To split the data into testing and training sets, scikit-learn module of python has a train\_test\_split function which makes it convenient to split the data. The interesting feature about this function is that we can specify the way the data is split by using the stratify parameter. So, if the dataset has 40% loan outcome as ‘healthy loan’ and 60% as high risk, then the training and testing data sets are also divided accordingly.

### Step 4 & 5: Initialising & fitting

Once the data is split into testing and training sets, the model is initialised. A random state is defined, which helps to generate the same set of random numbers every time the model is run. This helps in replicating the method in the future. Then the training data is fitted into the model by calling the fit function.

## Model performance

Once the model is trained, it is tested using the testing data. This is done by calling the predict function. An easy way to check the outcome is to create a dataframe with two series: one with the predictions and the second series containing the actual testing outcome, i.e. the 25% of the data (test data). Thus, we can compare the predictions vs actual results visually. But there are more accurate ways to gauge a model’s performance.

Our predictions can create 4 possible outcomes:

* True Positive (TP) - we predict positive and it’s actually positive i.e. predicted a healthy loan and it is indeed a healthy loan (predicted 0, actual 0)
* True Negative (TN) - we predict negative and it’s actually negative i.e. predicted a risky loan and it is indeed a risky loan (predicted 1, actual 0)
* False Positive (FP) - we predict positive and it’s actually negative i.e. predicted a healthy loan but it is actually a risky loan (predicted 0, actual 1)
* False Negative (FN) - we predict negative and it’s actually positive: predicted a risky loan when actually it is healthy loan class (predicted 1, actual 0).

Three indicators are calculated using the above to gauge the model performance:

### Accuracy Score

Besides eyeballing the data, a mathematical way to check how well the model has performed is to check the ratio of correctly predicted values to total number of values. This is the accuracy score and can be checked simply by calling the accuracy\_score function from scikit-learn.

**In this case, the accuracy score is 99.24% which is a good score for a financial use case.**

This means that our model can predict if a loan will be repaid or defaulted in more than 99 cases out of 100, which is a robust score for a loan prediction.

### Precision, Recall and Confusion Matrix

Accuracy scores are not good indicators to show the minority data. So, if the dataset contains very few loan defaulters but many healthy loans, then the risky loan data is not well represented. It causes an imbalance in the dataset and can cause the model to make faulty predictions about the minority data. These faulty predictions are not well captured in the accuracy score.

It is important to know that if a model predicts a positive score, it is indeed a true positive. **This is done by precision score, which is simply the ratio of true positives to all positives, whether true or false.**

**Whereas, a recall is the ratio of all true positives to all actual positives only.**

A confusion matrix is generated using the confusion\_matrix function. In this case, the model has generated 18,679 True Positives, 558 True Negatives, 80 False Negatives and 67 False Positives. Thus, the model has not accurately predicted all outcomes.

**Classification Report**

To neatly summarise the accuracy, precision and recall, a classification report is generated. It displays the results in an easy-to-understand format.

**Precision values:** In this case, our we got a precision score of 1.00 for predicting healthy loans, which means when the model predicted a loan to be healthy, it was indeed healthy.

However, when the model predicted a loan to be risky, it was correct only 87% of the time, i.e. of all the loans that the model predicted were risky, only 87% were actually risky.

**Recall value** **of 0.89:** Out of all the loans that were actually risky, the model predicted only 89% of them to be risky. The model was not able to identify 11% of risky loans at all.

For financial data, this is not a good enough model fit.

The Support section of the classification report shows that the data we fed the model was indeed imbalanced. There were 18,759 cases of healthy loans but only 625 cases on risky loans. Thus, our accuracy score was high but our precision and recall scores were comparatively low.

Based on these scores, I would not recommend the use of this model to the company.