# Predicting Risky Credit Card Applicants

## 1. Introduction

Credit risk detection is one of the hot applications for machine learning and deep learning. This project used customer profile information along with credit records of customers to build machine learning models to detect risky credit card applicants. The project has 5 stages: data cleansing, data exploration and visualization, imbalance data problem discussion, machine learning model building and finally deep learning network implementation on risky cases predictions. This project also gave some insights on how to tune models to solve imbalanced dataset problems.

## 2. Data cleansing

- application_record.csv: It is about personal information on customers, the card users. It includes their ID, age, salary, occupation type and other 14 of column names.

1) DAYS_EMPLOYED: Greater than 300,000 days of employed days do not make sense so we acknowledged these sorts of values as outliers, and these values have been dropped. Plus, values on this column did not look intuitive to understand at a glance. Thus, we converted values like -1134 into employed years like 3. DAYS_BIRTH column was also processed as well.
2) OCCUPATION_TYPE: There were null values in this column. Rows with this null value were 2 out of 9 on this dataset, these null values were dropped
3) FLAG_MOBIL: There is only one kind of value in this column, which is that people have mobile phones. It is meaningless because it cannot help anything in machine learning for classification.

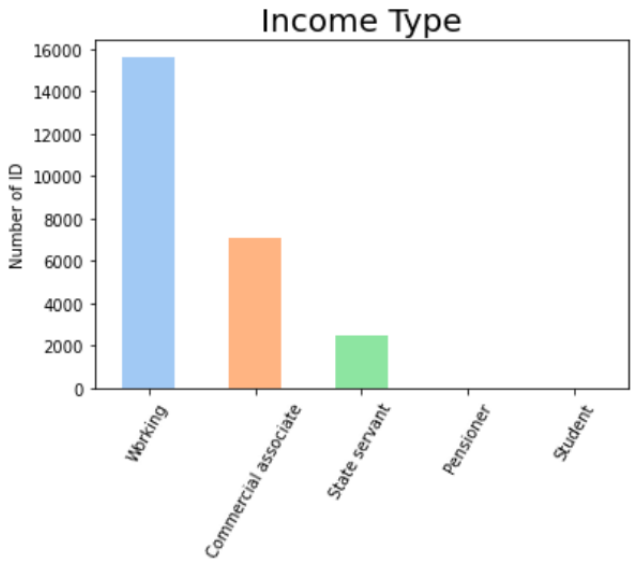
- credit_record.csv:credit_record is about customers' debt payment status, which includes their ID and debt status for each month.
    1) STATUS: There were credibility status values like X and C. They did not look intuitive to understand at a glance. We divided them into 'risky' or 'safe' if the default is over 2 months and converted them into 1 and 0 for machine learning models.

- Dataset merging: we merged two of the datasets into one dataset by using the merge function through Id to make it fit for machine learning. We created the final dataset in this process.

## 3. Exploratory Data Analysis

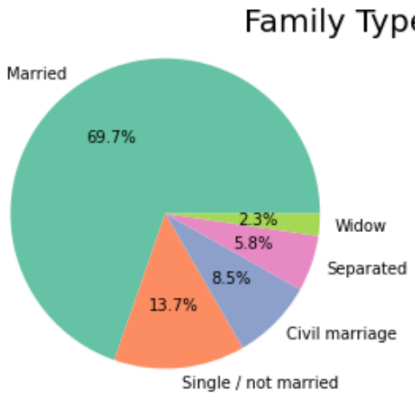
From the data set, which was handled and managed in the second part, the data cleansing part, exploratory data analysis became easier. Therefore, we could easily find the following: in the data set, there are more females than males, there are more people with realty than those who do not have realty, and there are more people without kids than with kids.

By plotting the par charts and pie charts, findings become clearer.



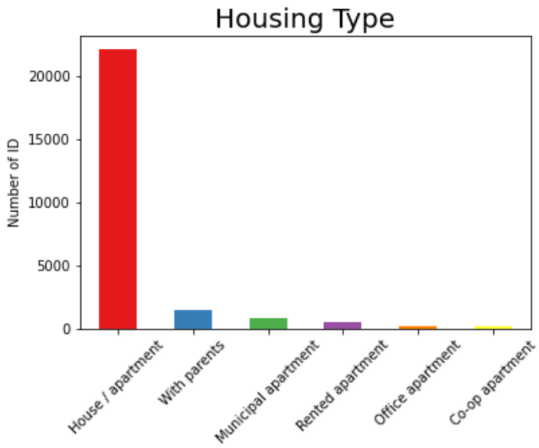

Fig[1] Income type

With the income types, the bar chart shows that over half was from working and 23% was from commercial.



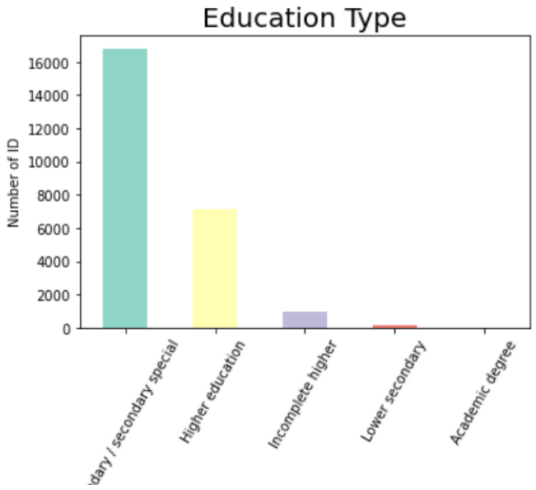

Fig[2]Family type

For family status, marriage counted the most, almost 70%, and single came second about 13%.



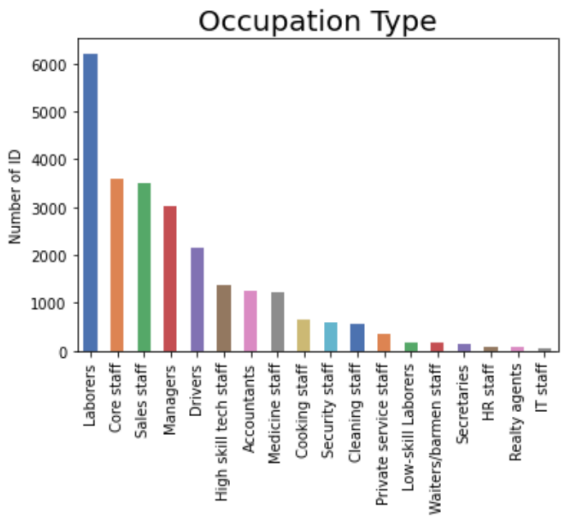

Fig[3]. Housing type

For housing type, apartment/ house was the most common type.




Fig[4]. Education Type

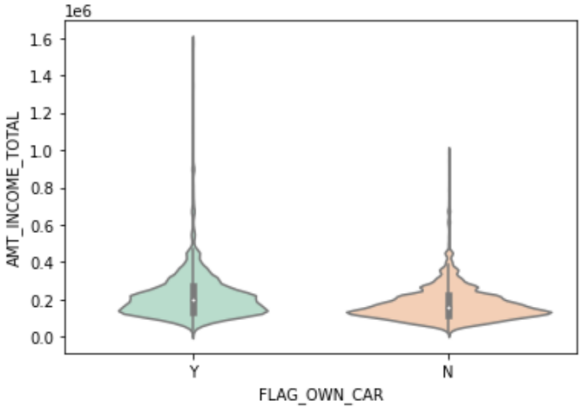
Most people got secondary, or secondary special education.




Fig[5]. Occupation type

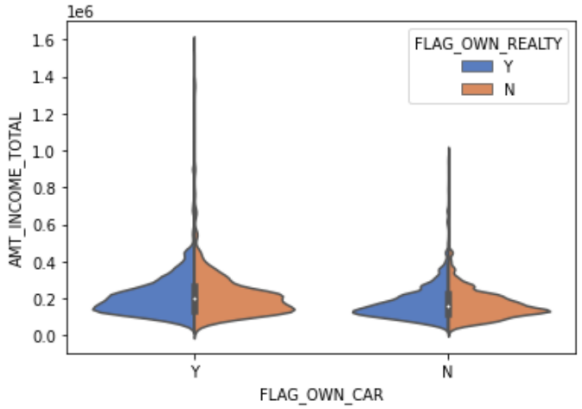
The occupation type was diverse, with laborers counted the most.

To figure out a relationship between the variables, we drew some violin plots between the variables.




Fig[6]. Violin plot between the car owners and total amount of income

From the violin plot between the car owners and the total amount of income, we could find out that the car owners have higher income, and even the highest income came from the car owners. This implies that car owners tend to have higher incomes than those who do not own a car.




Fig[7]. Violin plot between the car owners and total amount of income + realty

And when realty was included in the analysis, we could conclude that higher-income people tend to have cars.

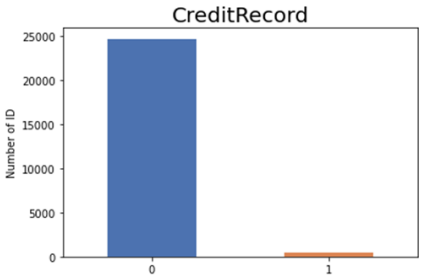
## 4. Imbalanced Data

**Concept:**

Imbalanced Data is data in which the target class has an uneven distribution of observations. The degree of imbalance depends on the proportion of the minority class. Imbalanced data could be classified as mild, moderate, or extreme. When the proportion of minority class is 20~40% it is mild, at 1~20% it is moderate and when it is less than 1% it is extremely imbalanced.

**Our Case:**

The target class of our data set is the column named ‘label’ which indicates whether the applicant’s credit is risky or safe. 0 indicates safe applicants and 1 indicates risky applicants. When we count the number of each applicant, we have 24712 safe applicants and just 422 risky applicants. This column is moderately imbalanced (close to extreme) as the proportion of majority and minority classes are 98.321% to 1.679%.




```
apl_v4['label'].value_counts()

0    24712
1      422
Name: label, dtype: int64
```

Fig[8].Credit card record

**The problem of Imbalanced Data**

The problem of imbalanced data arises when a classification model is trying to classify and predict the minority class. Usually, the predictive machine learning algorithms aim to score higher accuracy. Because of the small proportion of the minority class, however, the models try to score higher on the prediction of the majority class and the prediction of the minority class will be less considered. This goes against the goal of the projects that are trying to predict the minority class.

**Our case:**

| | Predicted: Risky | Predicted: Safe |
|---|---|---|
| Actual: Risky | True Positive | False Negative |
| Actual: Safe | False Positive | True Negative |

Fig[9].Confusion matrix

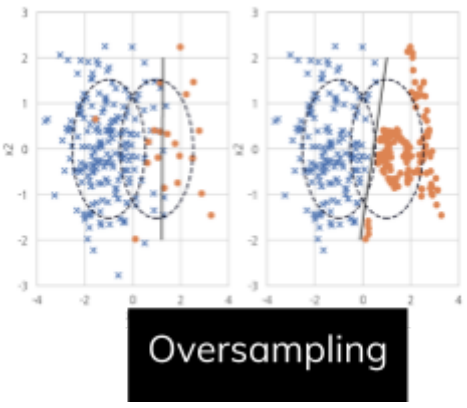
The confusion matrix above shows the possible results of our model. The goal is to predict the actual risky applicants which means that the most important value is True Positive. When we run a classification model, the overall accuracy would be high but that does not guarantee a good score of True Positive. Instead, the True Positive is likely to be very low as most of the data used to train the model would be data of Safe applicants.

**Handling imbalanced data problems:**

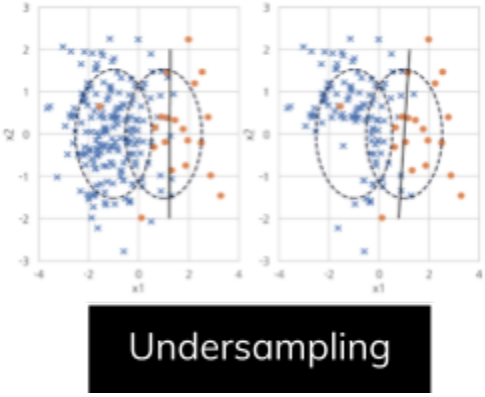
A few methods are used to handle imbalanced data problems. Changing the weight of each class is one solution. In this case, we should adjust and increase the weight of the minority class so that the minority class would have more significance.

Oversampling which is using techniques to increase minority class can be used.

Undersampling uses techniques such as Random sampling, Tomek's Links, Condensed Nearest Neighbor, and One-Sided Selection to decrease the majority class.




Fig[10]. Oversampling




Fig[11]. Undersampling

## 5. Machine learning Models

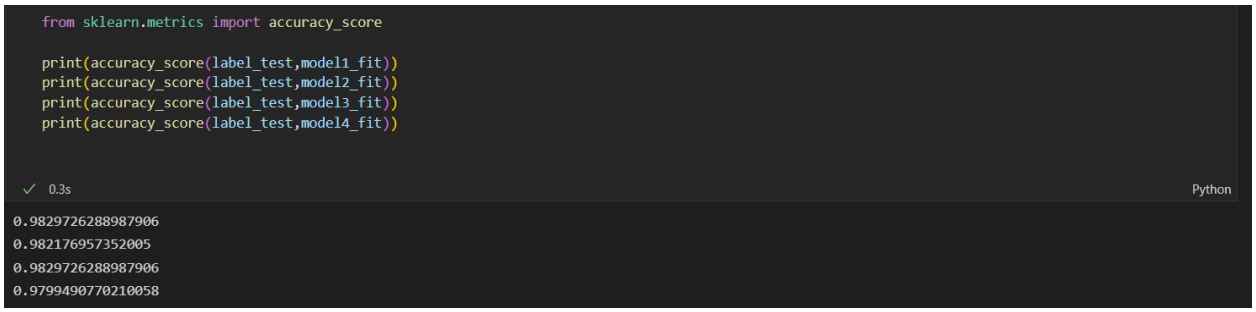
### Data encoding:

The dataset data is composed of two types: categorical data and numerical data. To train the machine learning model, the data needs to be converted into numerical data, in this project the job was done with a one-hot encoder for categorical data and a standard scalar for numerical data respectively, the dataset is taken as input for machine learning models.

The machine learning models used in this project are Logistic regression, random forest, SVM, and HistGradientBoostClassifier, these 4 classifiers were combined with the data transformation function by using the make_pipeline function.

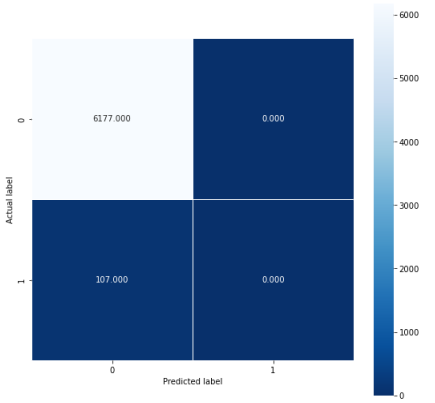
The original data is being separated into training data and testing data in a proportion of 0.75 and 0.25 respectively. After the separation, there are 18850 training data points and 6284 testing data points.

Without much tuning in machine learning parameters, basically, every model has achieved a high accuracy result in the testing dataset, which is a staggering 0.98!

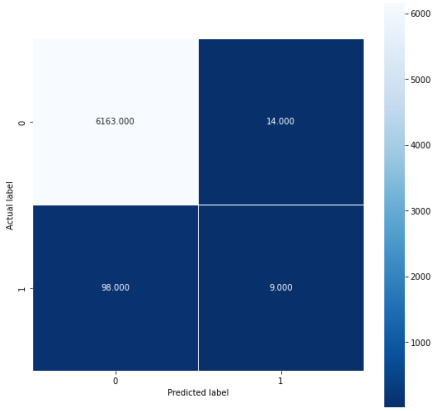



Fig[12] Accuracy results

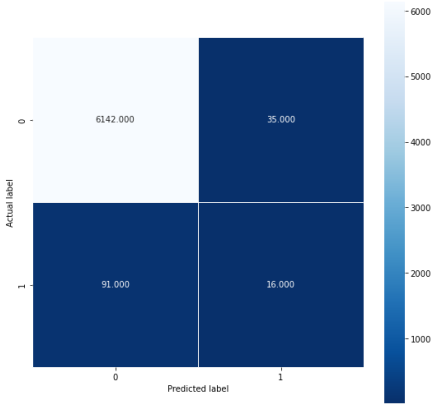
But accuracy is not the only metric in determining this model, accuracy the most important metric is its ability to predict risky cases, this ability is better viewed from the confusion matrix:



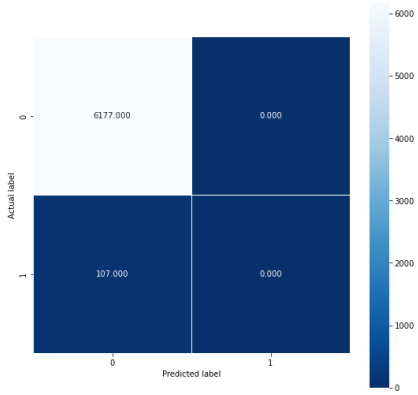

Fig[14].HistGradientBoostingClassifier




Fig[13] Logistic regression



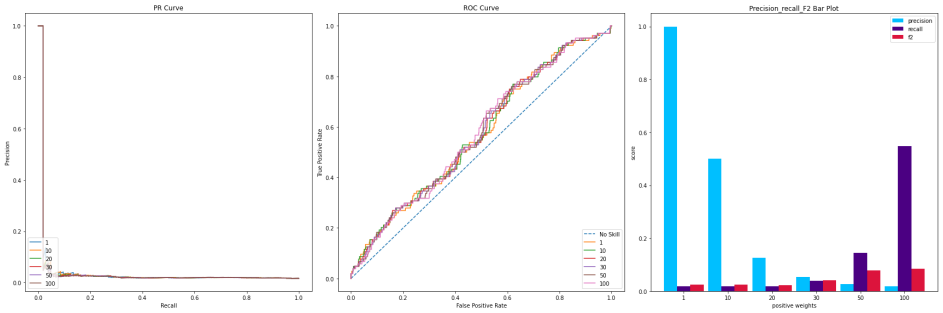

Fig[15] SVM



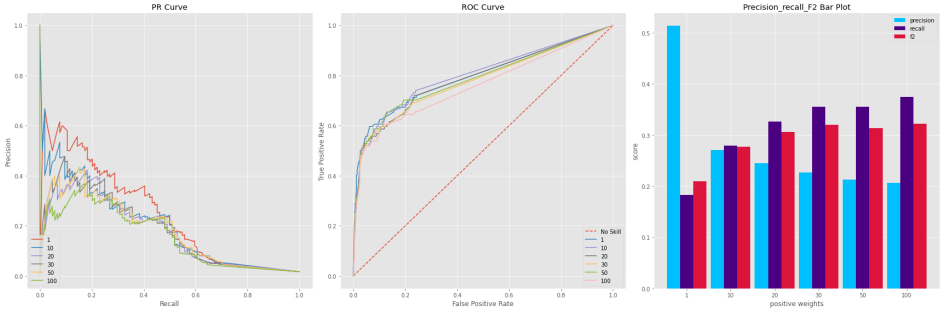

Fig[16] RandomForestClassifier

Risky cases are identified as positive cases 1 in confusion matrices, which is the lower right corner, the number in the corner indicates the model's ability to predict positive cases is low: out of 107 total positive cases in test dataset, only a small fraction of cases were identified, the best model random forest was just able to identify 16 cases, therefore model tuning is needed.

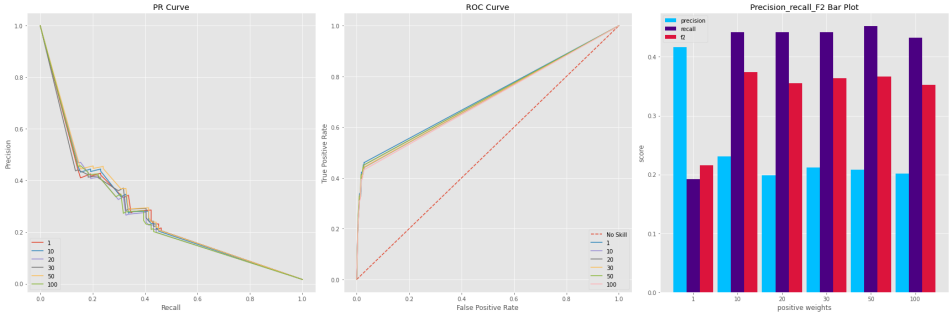
To tune the model and make the model more sensitive to risky cases, class weight was introduced in modeling. Since class_weight is only available for logistic regression, random forest and Decision tree, these 3 models were used to test the effect of weight class on the results of the machine learning models. To illustrate the effect, PR plot, ROC curve and precision_recall_F2 bar plot were used here.




Fig[17]Logistic regression




Fig[18]Random Forest




Fig[19]Decision tree

In these plots, the first and second plot in each row, there are multiple lines, which correspond to different class weights, the separation of lines is insignificant, meaning that the weight loss changes don't have a significant influence on the predictability of the machine learning models.

And for the bar plot for random forest and decision tree, precision is around 0.2 when recall rate goes near 0.4, the results concluded that these machine learning models are not optimal for solving the risky customer detection problem.

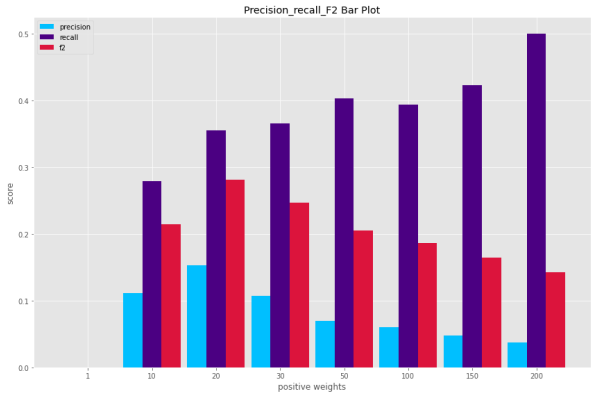
## 6. Deep Learning model implementation

To further address the issue faced by machine learning models, deep learning models were used in the hope to solve the problem faced by machine learning models.

The setup for the basic deep learning model is a 3-layer binary deep neural network, with 20 nodes in each hidden layer. The activation function used in hidden layers is the Relu function and in the output layer, the activation function used is the sigmoid function. The loss function used was binary cross-entropy and stochastic gradient descent was used for the model's optimizer.

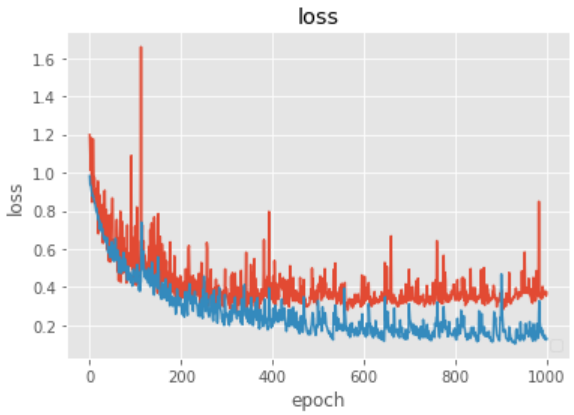
To address the imbalanced data problem, class weight was used again, weight = 1,10,20,30,50,100,

150,200 were used and precision, recall, and F2 bar plots shown down below:



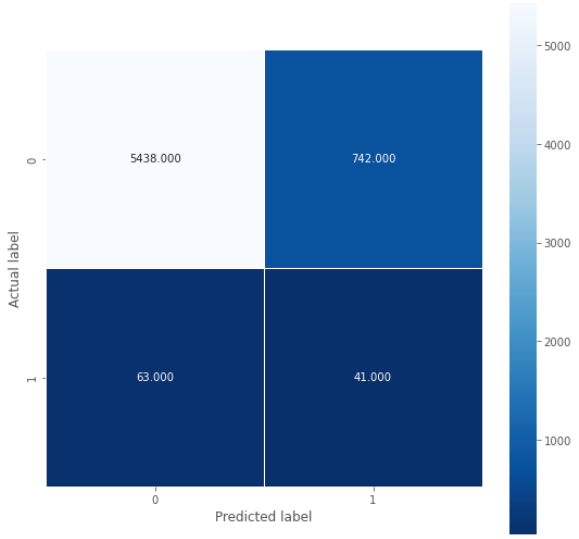

Fig[20] precision_recall_F2 for deep learning

But here the problem is the precision score is not high enough and the recall rate is low. What's more, the loss function is not stable

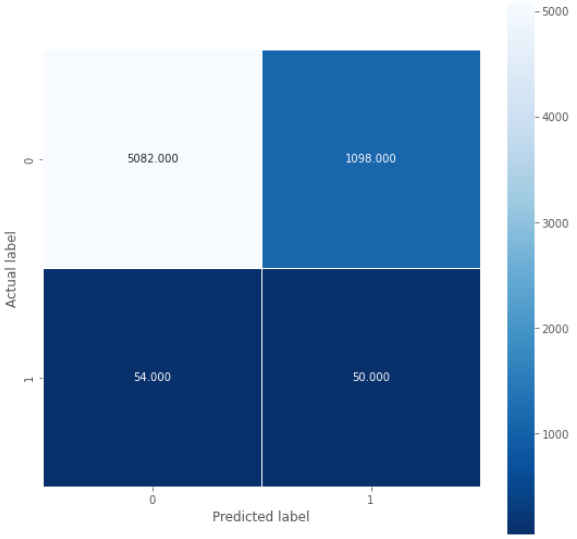



Fig[21] Loss drop curve

To make model training more stable, batch size = 5 is used, and the optimal results is achieved with class weight = 30 and 35




Fig[22] Confusion matrix with class weight = 30




Fig[23] Confusion matrix with class weight = 35

With the aforementioned configuration for the deep learning model, these two models have achieved a good result of identifying almost 50% of 104 total positive cases, with less than 12% false positive rate; this is acceptable, considering the number of risky cases in the total population is extremely small.

## 7. Conclusion

In conclusion, this project serves as an experiment on working on imbalanced dataset problems, the problem can be generated in many other cases, such as heart attack detection, and fraudulent information detection. What's more this project also gave some clues on how to tune deep learning hyperparameters, especially for imbalanced data for a better deep learning model.

## References

1.https://developers.google.com/machine-learning/data-prep/construct/sampling-splitting/imbalanced-data?hl=en

2.https://www.analyticsvidhya.com/blog/2021/06/5-techniques-to-handle-imbalanced-data-for-a-classification-problem/

3.https://machinelearningmastery.com/undersampling-algorithms-for-imbalanced-classification/

4.http://glemaitre.github.io/imbalanced-learn/generated/imblearn.under_sampling.OneSidedSelection.html

5.https://machinelearningmastery.com/bagging-and-random-forest-for-imbalanced-classification/