**Logo

Description automatically generated with medium confidence**

**MH6804 Python for Data Analysis**

**GROUP 5**

**Tang Shiwei**

**Paris Siripatana**

**Jonathan Then Tian Meng**

**Liu Zhiyang**

**Ku Chi-Chen**

[**Introduction**](#_heading=h.w7sw76eruh6d) **3**

**Data Pre-Processing 4**

**Training & Testing Procedure** **5**

[**Machine Learning Methods**](#_heading=h.rgrw69v0bq0b) **5**

* [Logistic Regression](#_heading=h.ask3cvwr0hp1) 5
* [Random Forests](#_heading=h.z6wyc9euk2ik) 6
* [Naive Bayes](#_heading=h.iyygm3dnngp3) 8
* [S](#_heading=h.1y2r1dab9w4h)upport Vector Machines 9
* [Voting for ML Methods](#_heading=h.wcapew18y6i5) 10

[**Deep Learning Methods**](#_heading=h.hqrrqu28w08v) **10**

* [MLP Classifier from SKlearn](#_heading=h.nu1n529fgvbo) 11
* [MLP Classifier from Tensorflow](#_heading=h.wa1z9cezm6za) 12
  + [Tensorflow tuned using](#_heading=h.hi6q4oncc9wb) Cross Entropy 12
  + [Tensorflow tuned using Focal Loss](#_heading=h.kujvykbibbio) 13

[**Summary**](#_heading=h.6c0tje5u8n4e) **15**

[**Future Improvements**](#_heading=h.ecg7pcbaupfd) **16**

**References 16**

# 

# **Introduction**

The data is related to direct marketing campaigns of a Portuguese banking institution from May 2008 to November 2010 created by: Paulo Cortez (Univ. Minho) and Sérgio Moro (ISCTE-IUL) in 2012. The marketing campaigns were based on phone calls and more than one contact to the same client was required. The classification goal is to predict if the client will subscribe to a term deposit (variable y). The dataset contains 45211 instances with 16 features.

| **Client Profile Attributes** |
| --- |
| 1- age (numeric)  2 - job : type of job (categorical: admin., unknown, unemployed, management, housemaid, entrepreneur, student, blue-collar, self-employed, retired, technician, services)  3 - marital status : marital status (categorical: married, divorced, single)  4 - education level (categorical: unknown, secondary, primary, tertiary)  5 - default: has credit in default? (binary: yes, no)  6 - balance: average yearly balance, in euros (numeric)  7 - housing: has a housing loan? (binary: yes, no)  8 - loan: has a personal loan? (binary: yes, no) |
| **Marketing Campaign Attributes** |
| 9 - communication type (categorical: unknown, telephone, cellular)  10 - day: last contact day of the month (numeric)  11 - month: last contact month of year (categorical: Jan, Feb, Mar, …, Dec)  12 - duration: last contact duration, in seconds (numeric) |
| **Other Attributes** |
| 13 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)  14 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric, -1 means client was not previously contacted)  15 - previous: number of contacts performed before this campaign and for this client (numeric)  16 - poutcome: outcome of the previous marketing campaign (categorical: unknown, other, failure, success) |
| **Output Variable (Target)** |
| 17 - y: has the client subscribed a term deposit? (binary: yes, no) |

**Data Pre-Processing**

|  |  |
| --- | --- |

Figure 1: Exploratory Data Analysis

The data set target variable or "y" has two values, which are labeled as "yes" and "no". However, the ratio of those values is significantly imbalanced, the ‘yes’ label has only 13.15%, whereas the ‘no’ label has 86.85%. Therefore, this data set requires a resampling method like SMOTE during training.

Some of the categorical features in the data set have unknown labels. The *poutcome* feature contains up to 80% of the unknown data, and the *contract* feature holds around 30% of it. Consequently, the loss of significant observations if applying the dropping method with the data set. Therefore, treating it as one categorical is primary for this project.

Both ordinal and one hot encoding were applied for the data set. The *education* feature was labeled as 0, 1, 2, and 3 for unknown, primary, secondary, and tertiary, respectively. Moreover, grouping months into the season before creating dummy features will make the data more compact with similar prediction power without grouping. The categorical features that exclude *education* were encoded by pd.get\_dummies (with drop first) to prevent duplicate features. Moreover, unknown variables were labeled as "AA" because the drop first parameter will drop alphabetically. The Balance feature contains some outliers, which could affect algorithms. Using Z-score to eliminate outliers that have Z-score more than 3 or less than -3 taken into account.

**Training & Testing Procedure**

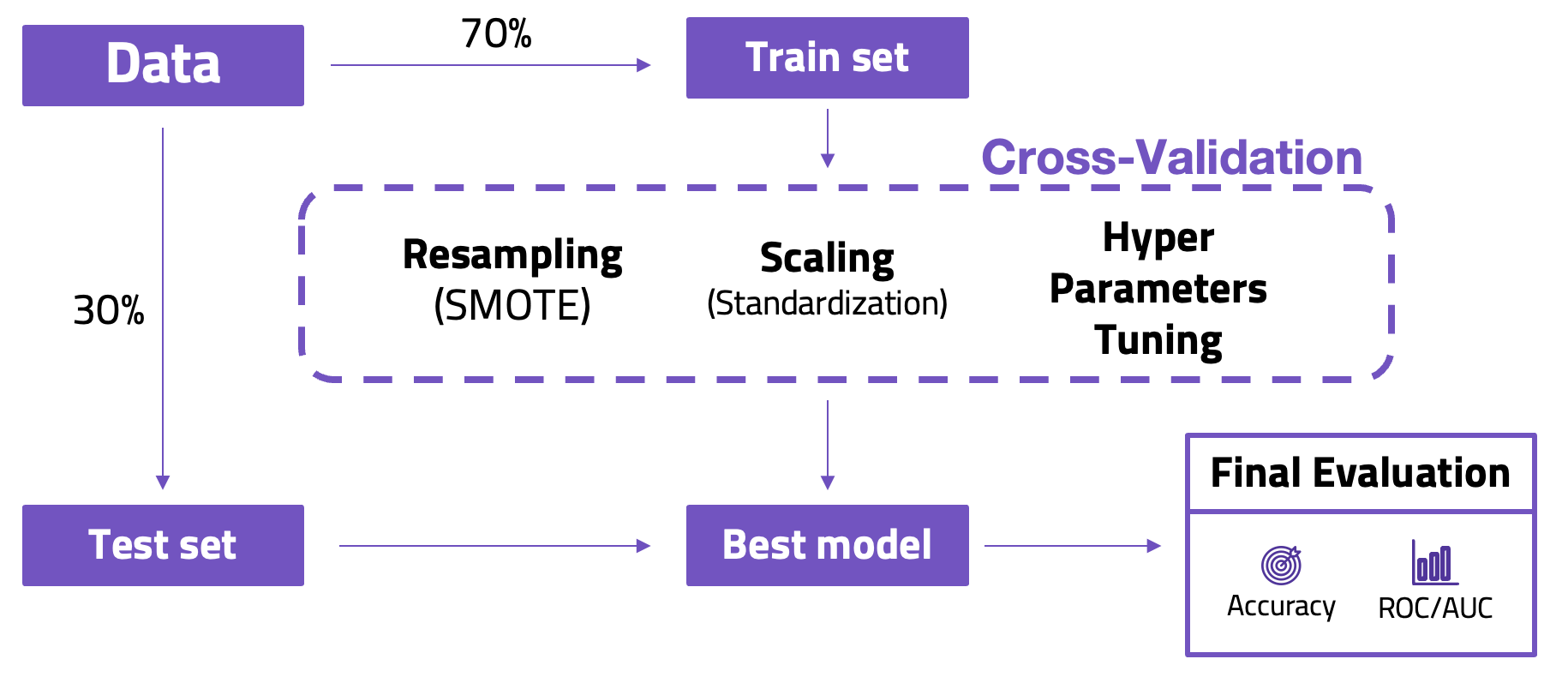


Figure 2: Training and Testing Procedure Model

The data was separated into a 7:3 ratio for training and testing after the preprocessing process. The training set was used in the cross-validation process to find the best model for each algorithm. Due to an imbalanced data set, using SMOTE to resample pseudo-train data is a must. Moreover, scaling data like standardization is required for some machine learning algorithms and faster computation. After resampling and scaling data, the tuning methods will be applied for achieving the optimal hyperparameters to predict the pseudo-test. The best model from the cross-validation process will be used to predict the test set from the first splitting before evaluation with accuracy and ROC/AUC metrics.

# 

## **Machine Learning Methods**

### Logistic Regression

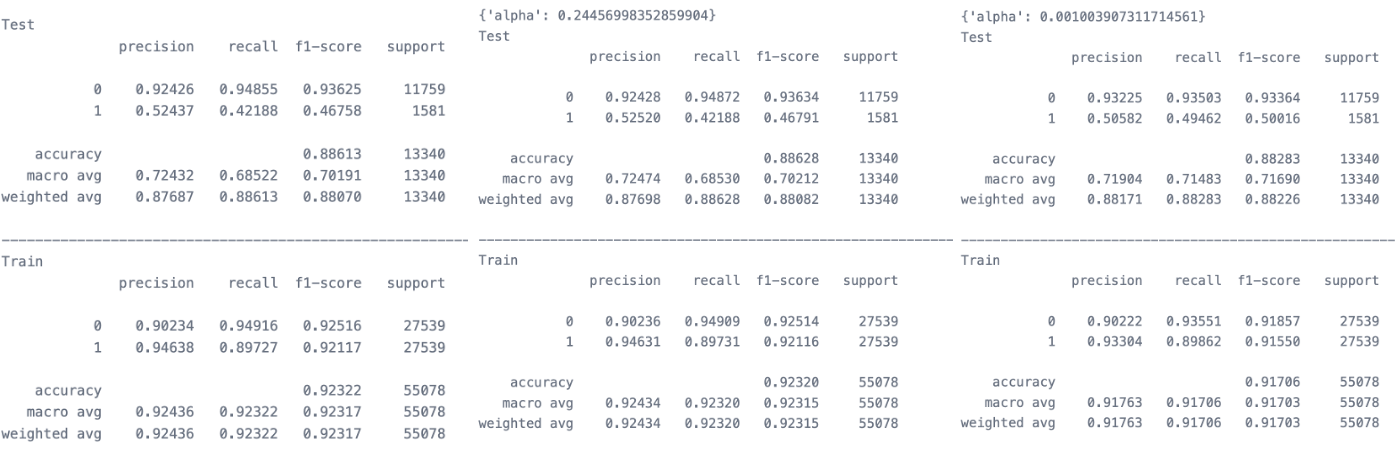


Figure 3: Logistic, Ridge, Lasso Regression Classification Reports

Logistic Regression is one of the simplest models for classification problems that generate the probability, which has a threshold of 0.5. The prediction with a probability above the threshold will classify into a different class from the below one. In this algorithm in sci-kit learn, we choose 3 parameters for tuning.

* *penalty*: Specify the norm of the penalty (l1, l2, and none)
* *C*: Inverse of regularization strength; must be a positive float. Smaller values specify stronger regularization.
* *solver*: Algorithm to use in the optimization problem. (we choose "saga" due to standardization and large data set)

The basic logistic regression (penalty is none) doesn't have any parameters to tune, but it provides a good result. For regularization methods like Ridge and Lasso after tuning, they deliver similar results to the initial model. but, other metrics were improved. ForArea under the ROC Curve, all models provide almost the same result. However, Lasso has the best AUC of 0.8652 with the following parameters;

| * *penalty*: “l1” * *C*: 0.001003907311714561 * *solver*: “saga” | Figure 4: Comparison of Reg Model’s AUC |
| --- | --- |

### Random Forests

Random Forests is a very efficient statistical learning method. It is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The *RandomForestClassifier* built in *sklearn.ensembl*e has 2 most important parameters:

* *n\_estimators:* determine the number of decision trees that make up our random forest.
* *max\_features:* defines the number of features that each decision tree takes into consideration at each split. We used the default value as the square root of the number of features.

After tuning the parameters we found that the n\_estimators=100 to 500 will give the best similar performance and due to the time consuming we used parameters with *n\_estimators*=100, *max\_features*=‘sqrt’, *criterion* =‘gini’ as the final model.

Results and the confusion matrix are given below. Our model predicted 764 true positive answers and 835 false negative answers. The accuracy of this model is nearly 0.8984. The Area Under the ROC curve is 0.9138. We are also focused on the False Negative Rate which is 0.5222. This should be improved since that’s what we predicted the client would not buy the term deposit but in fact they will. Decreasing the FNR helps the banking institution give more effective marketing campaigns.

| ACC | AUC | TPR | TNR | FPR | FNR | PPV |
| --- | --- | --- | --- | --- | --- | --- |
| 0.8984 | 0.9138 | 0.4778 | 0.9557 | 0.0443 | 0.5222 | 0.5950 |

|  |  |
| --- | --- |

Figure 5: Random Forest Confusion Matrix and ROC AUC

For feature selection, the Gini importance is given below:

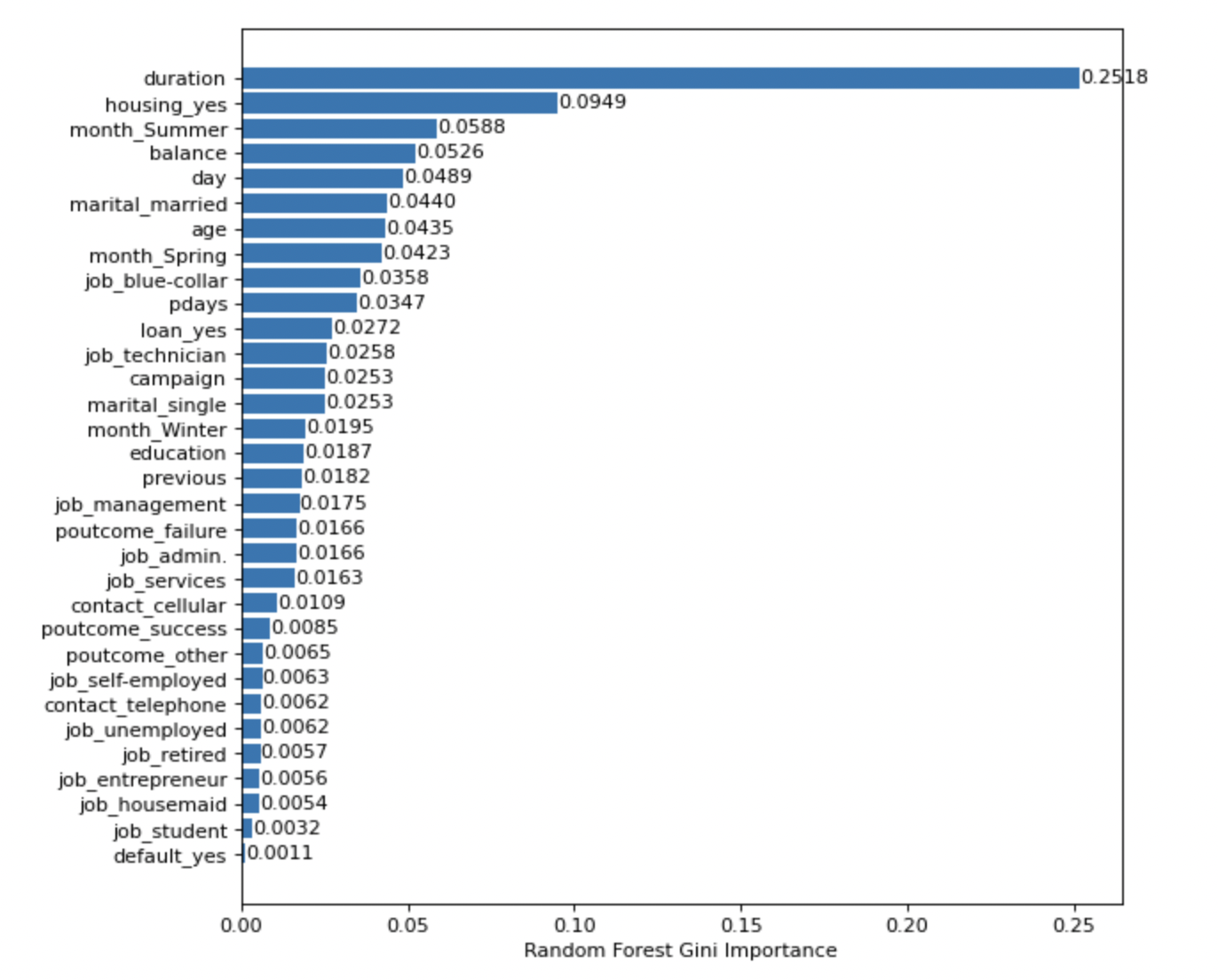


Figure 6: Gini Importance of Random Forest

We noticed that among the 16 features (32 dummy features), the last contact *duration*, having *housing* loan and yearly *balance* are the most important variable.​ A client portrait such as married with a blue-collar job will have more possibility to subscribe the product.​

Though the contacting duration is the most important, the marketing result is more relevant to customers' profile, instead of the marketing campaigns.

### Naive Bayes

Naive Bayes Classifier is a classification technique based on Bayes’ Theorem with an assumption of independence among predictors. It is easy to implement, efficient and robust to isolated noise points. From the results in Figure 7, we get an accuracy of 0.86 and an area under the ROC curve of 0.83.

|  |  |
| --- | --- |

Figure 7: Evaluation Metrics of Naive Bayes Algorithm

### 

### Support Vector Machines

A support vector machine (SVM) is a supervised machine learning model that uses classification algorithms for two-group classification problems. We used a smaller dataset with 10% of the original data, totalling 4521 records that was provided in the Github link to test computationally demanding machine learning algorithms. To find the best model, we first had to determine the type of kernel that we were going to use, namely, linear, polynomial, radial basis function or sigmoid. After that, we also need to state the tuning parameter C which defines the weight of how much samples inside the margin contribute to the overall error. Finally, we fitted the data with the model and did a 5-fold cross validation using the GridSearchCV function. The best model chosen was a linear kernel with a tuning parameter C of 0.1. The accuracy is 0.9 and the AUC 0.81.

|  |  |
| --- | --- |

Figure 8: Evaluation Metrics of Support Vector Machines

**Voting for ML Methods**

A voting ensemble (or a “majority voting ensemble“) is an ensemble machine learning model that combines the predictions from multiple other models. It is a technique that may be used to improve model performance, ideally achieving better performance than any single model used in the ensemble.

However, there are some limitations for the voting method. The voting method is not suitable for deep learning to some extent, since the model from deep learning is usually computationally heavy, which makes cross-validation very hard. Therefore, Multiple layer perceptron (MLP) will be a separate part.

The following figures explains how the voting method work and the accompanying results.

| Figure 9: Overview of how Voting Method works | Figure 10: Results of the Voting Method |
| --- | --- |

Test Accuracies are almost the same but F1 scores and recalls vary.

In the voting method, with the accuracy almost the same as other methods, the voting method (GBDT+XG Boost) is the best.

# **Deep Learning Methods**

We also implemented deep learning methods. It is worth mentioning that we use only the feed forward neural network, or the Multilayer Perceptron (MLP), since we don’t have the time series data or multi dimension data that need to alter the way to train the model. So the CNN and LSTM models are not introduced in our methods.

### MLP Classifier from SKlearn

First the easiest way for us to check the performance of MLP is to simply use the SKlearn model, we first balance the data using SMOTE, build the model and tune the hyperparameters, then we have the parameter results:

|  |  |
| --- | --- |

Figure 11: Tuned hyperparameters for MLP

We can find that the accuracy is already satisfying with almost ninety percent, and AUC is 0.7 which is not too bad, however the F1 score and the recall is a disaster, by using the SKlearn model, we cannot achieve our goal in that this model cannot identify the positive samples effectively.

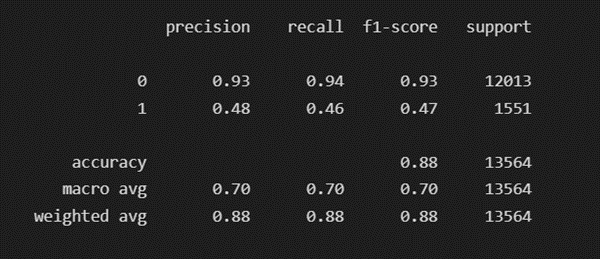


Figure 12: Classification Report of MLP

We summarize three reasons for this, the first is The MLP model can be tuned further, the second is The test set is highly imbalanced with 12013 negative labels and 1551 positive labels. The SMOTE provides duplicated and useless information for the classifier to learn.

For the first reason, we tuned our model further, so we introduced the tensorflow in our deep learning models.

### MLP Classifier from Tensorflow

#### Tensorflow tuned using Cross Entropy

We tuned the tensorflow model based on the SMOTE balanced dataset, and the tuning result is as follows:

|  |  |
| --- | --- |

Figure 13: Tensorflow Model Tuned Hyperparameters

Using these hyperparameters to fit the dataset, we can find some progress in the performance. As shown in the comparison, all the scores have risen, especially the AUC rose from 0.7 to 0.88, and we have a better F1 score and recall.

|  | |
| --- | --- |
|  |  |

Figure 14: Results for Tensorflow Model tuned using Cross Entropy

The improvement is mainly due to the altering of the loss function, the optimizer and the drop out in the hidden layer.Yet this improved result is still not satisfying because we want to identify the positive more precisely, we should look back to see the original issue that caused this poor performance of recall.

So instead of using SMOTE to do the data set rebalance, We focus on the model itself. We seek to find another loss measure for the deficient dataset so that it can do better on the minor sample. And we introduced focal loss to further improve the deep learning model.

#### Tensorflow tuned using Focal Loss

To introduce the Focal Loss, we put the original cross entropy loss as comparison.

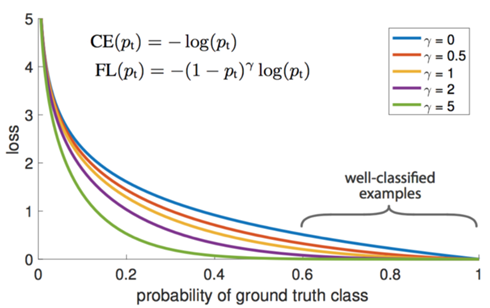
|  |
| --- |
|
|

In the cross entropy, the loss for each sample is even weighted. It measures the error in classification or the distance between the predicted model and ideal model. However, this loss function performs badly when encountering the imbalance data since the weight of the majority and minority samples are the same and training can hardly reflect the loss reduction regarding the minority.

To reduce the bias from the imbalanced dataset, the focal loss introduced two more params, which is alpha and gamma.

|  |  |
| --- | --- |

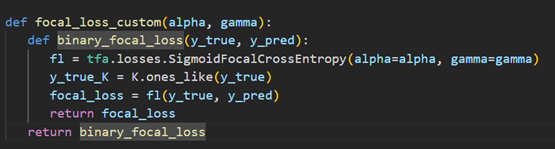
Here the α is the weight parameter for sharing the whole loss between the positive and negative samples.the γ is the focusing parameter, and γ>=0,and the (1-Pt)^γ is the modulating factor, we can find that when γ =0, α=0, the Focal Loss will reduce to the cross entropy loss.

  
Figure 15: Different Gamma Values in Focal Loss

The main idea of the focal loss is to reduce the weight of the easy classified weight and focus the model training on the hard samples. The α controls the weight of the positive and negative and the γ controls the weight of the easy classified and hard classified.

When γ is certain, the same easy example (pt = 0.9) has a loss that is 100+ times smaller than the standard cross-entropy loss, but for the hard example (pt < 0.5), the loss is at most 4 times smaller. In this way the weight of hard examples is relatively much higher. This loss increases the importance of those misclassifications or the minor samples thus improving the performance regarding minor samples.

By using the tensor flow addon package and our self-defined function as you can see in Figure 16, we implemented the MLP using focal loss and tuned the parameters.



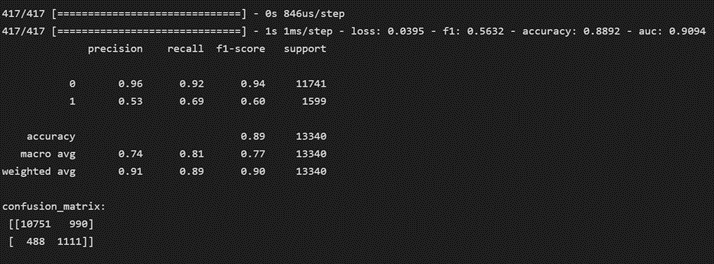


Figure 16: Self-Defined Function for Focal Loss & Final Results

Finally, compared with the previous results using the SMOTE, our enhanced MLP model does better in F-1 score, recall of the positive, and also we maintain the accuracy. The model is actually more robust in that we don’t need to generate the data to balance anymore, and we don’t need to alter the threshold to reduce the false negative rate, all the issues can be solved using focal loss, and it actually even does better.

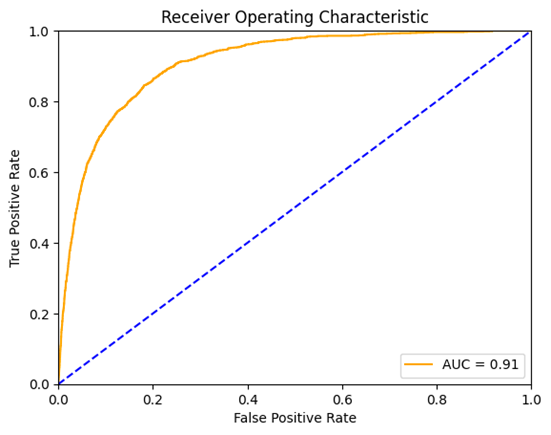


Figure 17: Tensorflow tuned with Focal Loss AUC Result

# 

# **Summary**

We summarize all the work we did and discuss the further improvements, so first we introduced the machine learning methods training on the SMOTE dataset, and all the tuned model actually performed quite good, including the voting methods.

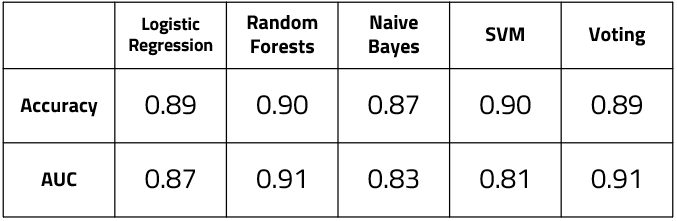


Figure 18: Results of Evaluation Metrics for Machine Learning Models

And then we introduced the deep learning methods, which includes the MLP classifier from SKlearn and two other models from tensorflow, by altering the loss function to focal loss, we not only maintain the accuracy but also enhanced all the other metrics, as you can see the recall is much better now.

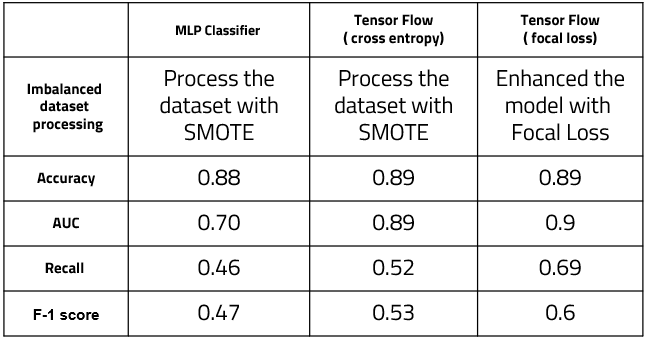


Figure 19: Results of Evaluation Metrics for Deep Learning Models

# 

# **Future Improvements**

Despite the progress we have, there are still some aspects that deserve future exploration. Firstly, regarding the conventional machine learning methods, since the random forest is one of the best models, we will try to stack different base estimators into ensembled methods like the random forest to further improve the performance.

Secondly, we can further tune the focal loss further to fit the deep learning model in that this procedure can be quite time-consuming if the parameter grid is larger. What’s more, finding other loss measures for the multi- layer perceptron model to enhance the overall performance better.

Last but not least, computing power permitted, we will try to conduct voting on deep learning models.

# **References**

T. -Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, "Focal Loss for Dense Object Detection," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 42, no. 2, pp. 318-327, 1 Feb. 2020, doi: 10.1109/TPAMI.2018.2858826.

z-o-e (2013). Bank data analysis. Github.  
<https://github.com/z-o-e/bank_data_analysis>