# Loan Default Prediction

Youssef Ashraf Kandil
Computer Engineering
The American University in Cairo
youssefkandil@aucegypt.edu

Abdullah Mohamed Kassem
Computer Engineering
The American University in Cairo
Abdullahkassem@aucegypt.edu

## Analysis criteria

In this project we are required to predict the loan defaults based on some features, which means that we have a classification machine learning problem composed of two classes 0 and 1 (where 1 label indicates a default). By analyzing the data we realized that the defaulters are the minority class in the dataset, implying that we need to consider the recall rather than the accuracy and precision, when measuring the models' performances.
In other words, in the banking field, the bank needs to minimize the percentage of fallaciously accepted loans. Which means that we need to consider minimizing the false negative predictions of our chosen models.

## Further data preprocessing:

In this phase we started analyzing the dataset once again before working on the training of the models. And by doing so we realized a major problem that was hidden in the previous phases. First the data was highly imbalanced, it has a minority class of about 25% of the data. Some of the features were highly populated with null values, namely [upfront charges, LTV, and rate of interest] , and at the same time, those rows containing the null values all belong to defaulters.

## Data analysis and preprocessing steps we took in this phase:

- We first found that while handling the NULL values we deleted some rows that have NULL values, these rows turned to mostly belong to class 1 which was already the minority class.
- So to fix that we deleted the columns with the top3 NULL values. That greatly reduced the number of NULLs in our data.
- Because some models cannot handle NULL values we needed to remove every NULL value from our dataframe.
- Still there were some values. We handled those by replacing the Nulls with a random value or the mean, depending on the feature.
- Then we ran all the classification models that we covered in the course.
- At the time we thought that the results we got were not acceptable, because the minority class 1, which represents the loan defaulters, had a best recall value of maximum 0.6. So we attempted multiple ways to fix that.
- After searching the internet we thought that the poor recall may have been due to the imbalance between the labels. Class 0 represented around 75% of the data while Class 1 represented the remaining 25%.
- We searched for ways to handle that imbalance and came across SMOTE (Synthetic Minority Oversampling Technique) and Near Miss.
- The model metrics we got were excellent however, we realized that we performed SMOTE on all the data (train set & test set). After thinking it through we realized that is not the right thing to do, as we used synthetic data within the test set to be predicted, and that only the training data should undergo the SMOTE process and the test data should be left as it is.
- So we only performed SMOTE on the train set and reran the models. The results were once again disappointing.
- So we again looked for another method to enhance our results. This time our method was to first use NearMiss. Which is similar to SMOTE in the sense that it balances the dataset, however it does the opposite. While SMOTE increases the minority class size, NearMiss reduces the majority class size.
- After performing NearMiss we performed PCA (Principal Component Analysis) for dimension reduction to better improve our results.

- Again the results were not much better than our original results and were far worse in some models.
- After consulting our instructor, We were advised to go with our original results, so we compared the performance of the models and for the top 3 performing models we used cross validation to tune the parameters to get the best results from this model. Then we found our best performing model.
- In an attempt to better improve our results we decided to use knn for data imputation or other imputation methods that are more complex than the methods we used (mean, zero, random).
- We attempted to use sklearn's sklearn.impute.KNNImputer but we could not get it to work properly in time.
- So we decided that for the next phase we will start by 2 main things to better improve our chosen model.
    - First is using complex data imputation especially for the income feature as it is important to our prediction and has a relatively high NULL percentage of 6%.
    - Then we will modify the loss function to improve our results.

## Cross validation parameters reference

***Knn;***

These parameters were given to the sklearn function RandomizedSearchCV. Which performs fit and score on the model and using cross validation it tunes parameters from the parameters given to it and it finds the best parameters and returns them. Then these optimum parameters are available for us.

```
parameters_knn={'n_neighbors': [3, 13, 23, 33]}
```

***Decision tree;***

```
parameters_tree={'max_depth': [10, 20, 50, 100, None],
        'max_features': ['auto', 'sqrt'],
        'min_samples_leaf': [1, 2, 4],
        'min_samples_split': [2, 5, 10],
        }
```

***Random forest;***

```
parameters={'bootstrap': [True, False],
        'max_depth': [10, 20, 50, 100, None],
        'max_features': ['auto', 'sqrt'],
        'min_samples_leaf': [1, 2, 4],
        'min_samples_split': [2, 5, 10],
        'n_estimators': [50, 100]}
```

***Neural network;***

```
parameters_two={
'hidden_layer_sizes': [ (2,2) , (2,8,5) , (8,8,8)],
'max_iter': [200, 500],
'activation': ['logistic', 'tanh'],
'learning_rate': ['constant', 'adaptive']
        }
```

## Highest performing models:

1. Random forests
2. Decision trees
3. Neural networks

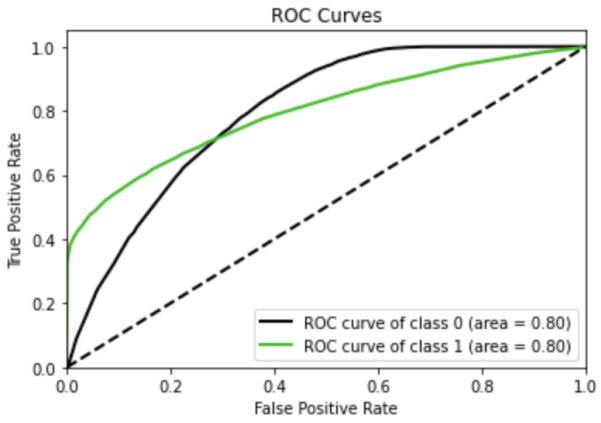
## K Nearest Neighbor

In this model we chose a random value of k = 30, and calculated the precision and recall, realizing that the model performed expectedly badly. Even Though it had a bearable average recall, it had a very bad recall value for label (1) class, which is the minority one.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.84 | 1.00 | 0.91 | 22319 |
| 1 | 0.99 | 0.41 | 0.58 | 7415 |
| accuracy | | | 0.85 | 29734 |
| macro avg | 0.91 | 0.70 | 0.74 | 29734 |
| weighted avg | 0.87 | 0.85 | 0.83 | 29734 |

So to be sure that the model is not suitable for our problem we performed cross validation using the following parameters for K [3, 13, 23, 33], and we yielded the following results.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.83 | 0.99 | 0.90 | 22319 |
| 1 | 0.90 | 0.41 | 0.56 | 7415 |
| accuracy | | | 0.84 | 29734 |
| macro avg | 0.87 | 0.70 | 0.73 | 29734 |
| weighted avg | 0.85 | 0.84 | 0.82 | 29734 |




Since the results of the best knn parameters from the cross validation were not much different to the results we got from the random K=30 and since the ROC curve did not show acceptable values we decided that KNN is not the right option and moved on.

Thus by analyzing the results of the cross validation, we came up with these conclusions:

- The KNN model will be excluded from consideration
- Recall values are going to be priorities based on the minority class value, then average value, then majority class value. With preference to the first.

## Logistic regression

Because our project is a classification problem that has 2 classes 0 and 1, we thought that logistic regression would yield good results, however it didn't. Because our first results were so bad, we decided that using cross validation to find better parameters would not be worth it.

Notice that this model had very high recall value for the majority class, however we could not choose it as the minority class has bad value. And we want to minimize the false negatives as mentioned before.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.78 | 0.97 | 0.86 | 22319 |
| 1 | 0.66 | 0.17 | 0.28 | 7415 |
| accuracy | | | 0.77 | 29734 |
| macro avg | 0.72 | 0.57 | 0.57 | 29734 |
| weighted avg | 0.75 | 0.77 | 0.72 | 29734 |

## Decision tree

Based on the literature review on Loan Default prediction ML algorithms we performed in the first phase, the decision tree models had high performance so we had high hopes when trying it. Also the decision tree works in a way similar

We first ran the decision tree model using the default sklearn parameters:

```
class sklearn.tree.DecisionTreeClassifier(*, criterion='gini', splitter='best', max_depth=None, min_samples_split=2,
min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, class_weight=None, ccp_alpha=0.0)
```

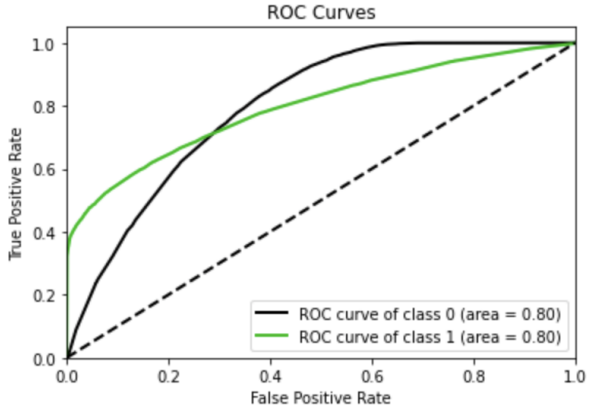
and here are the results:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.87 | 0.85 | 0.86 | 22319 |
| 1 | 0.57 | 0.61 | 0.59 | 7415 |
| accuracy | | | 0.79 | 29734 |
| macro avg | 0.72 | 0.73 | 0.73 | 29734 |
| weighted avg | 0.79 | 0.79 | 0.79 | 29734 |

These results had the highest recall we have seen yet, which matched our expectations.

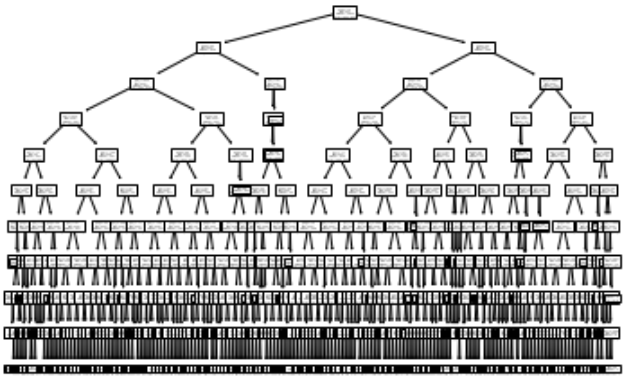
Then we tried using cross validation to improve the results. The cross validation parameters were suited to reduce the overfitting that is highly associated with decision trees. For instance, we assigned different depth thresholds as a way of (artificially) pruning the tree. Here are the results of the best performing model parameters from cross validation:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.83 | 0.99 | 0.90 | 22319 |
| 1 | 0.90 | 0.41 | 0.56 | 7415 |
| accuracy | | | 0.84 | 29734 |
| macro avg | 0.87 | 0.70 | 0.73 | 29734 |
| weighted avg | 0.85 | 0.84 | 0.82 | 29734 |




We can see when comparing the results of the default parameters and the cross validation best parameters that the average precision, recall and f1-score have all improved. However the recall in the class 1 has dropped significantly, and the minority class recall is the value we care about the most, for reasons stated above.

The results we got from the default parameters are the best results we have seen yet. And can expect that the random forest model would probably perform much better as it is less prone to overfitting and show higher results in general cases.



Here is a visual representation of the decision tree that has been built

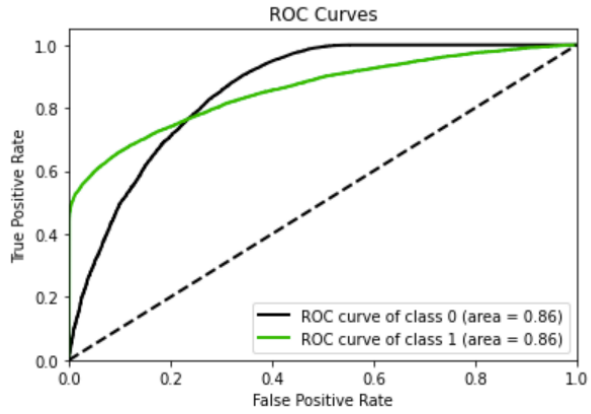
## Random forest

Random forest was the best performing model for many of the papers we read, which should be no surprise as decision trees performed well and the random forest is a collection of multiple decision trees.

We started by using cross validation to find the optimum parameters for the model, where we also applied the mentioned . And here are the results:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.86 | 0.99 | 0.92 | 22319 |
| 1 | 0.94 | 0.52 | 0.67 | 7415 |
| accuracy | | | 0.87 | 29734 |
| macro avg | 0.90 | 0.75 | 0.80 | 29734 |
| weighted avg | 0.88 | 0.87 | 0.86 | 29734 |

```
array([[22070,   249],
       [ 3551,  3864]])
```




The ROC curve had a greater AUC than the one in the decision tree. Although all the metrics were higher in the random forest because the decision tree had a recall of 0.61 we are led to believe that the decision tree is better than the random forest.

## Naive bayes

We tried the Naive Bayes model and it produced terrible results as shown below:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.76 | 0.99 | 0.86 | 22319 |
| 1 | 0.73 | 0.08 | 0.14 | 7415 |
| accuracy | | | 0.76 | 29734 |
| macro avg | 0.75 | 0.53 | 0.50 | 29734 |
| weighted avg | 0.75 | 0.76 | 0.68 | 29734 |

This is not very surprising because as we interpret, semantically viewing, there is no high probabilistic dependency between the features in the dataset. Consequently, the model was not able to train well over the given data. So we decided to disregard this model and did not consider bayesian networks, instead we focused our efforts on more promising models.

## Neural networks

In this model, we first ran a sample test case to understand the dynamics of it, and to have a vision of how effective it can be. So we created a neural network of exactly two hidden layers, each having 8 perceptrons. And chose normal parameters with a logistic activation function. The results were so promising compared to other models as it had 0.5 recall on the minor class with only two hidden layers.

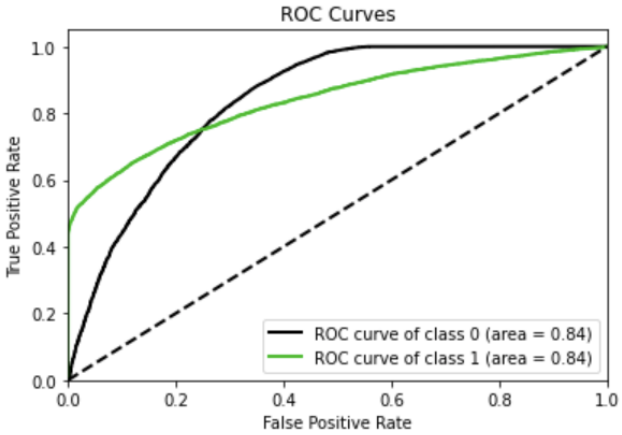
Parameters

*MLPClassifier_model-two = MLPClassifier ( hidden_layer_sizes = (8,8) , activation = 'logistic' , solver='adam' , max_iter = 500 , learning_rate = 'adaptive' )*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.85 | 0.99 | 0.92 | 22319 |
| 1 | 0.95 | 0.49 | 0.64 | 7415 |
| accuracy | | | 0.87 | 29734 |
| macro avg | 0.90 | 0.74 | 0.78 | 29734 |
| weighted avg | 0.88 | 0.87 | 0.85 | 29734 |

Thus we decided to run a cross validation training to find the best set of parameters over the training set, with these parameters:

parameters_two={
'hidden_layer_sizes': [ (2,2) , (2,8,5) , (8,8,8)],
'max_iter': [200, 500],
'activation': ['logistic', 'tanh'],
'learning_rate': ['constant', 'adaptive']
}




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.85 | 0.99 | 0.92 | 22319 |
| 1 | 0.95 | 0.49 | 0.64 | 7415 |
| accuracy | | | 0.87 | 29734 |
| macro avg | 0.90 | 0.74 | 0.78 | 29734 |
| weighted avg | 0.88 | 0.87 | 0.85 | 29734 |

We can see that there is no significant difference between the results of the best parameter set and the originally tested one, which means that having more than two layers did probably overfit the data, or resulted in losing the semantic meaning of the data.

Also by looking at the ROC curve, and comparing it to the other models, we see that it has better values than knn and decision tree but similar to random forest.

## Conclusion

Based on this comprehensive study, we decided to choose the Random forest as our model for next phases, and work on enhancing the loss function and construct more rigorous validation tests to find the optimal parameters. Even though the decision tree model had the best results in terms of the minority class recall value, we decided to consider the Random

Forest as it is based on the prior model and we can consider setting the parameters of the Random forest to closely match the results of the decision tree model.