Fintech – Homework 2

1. Classification

i) DNN for binary classification

In this part of the Homework, I create a DNN binary classification algorithm in order to achieve a credit card fraud detection. I use the library TensorFlow to create my model and Sklearn to get the optimal parameters. First, I import and slip into two parts the data. Then, I implement several functions related to the NN.

- ***Creation_of_network****(nb_hidlayer, nb_hidunits, learning_rate)* create a NN of the good size using the loss function "sparse_categorical_crossentropy" and the "Adam" optimizer.
- ***Training_model****(nb_hidlayer, nb_hidunits, learning_rate, epochs, batch_size)* trains our model previously created with ***Creation_of_network*** with the epoch and batch wanted.
- ***Affiche****(history)* print the loss and accuracy graphs corresponding to the training of our model.
- ***Optiparameters****()* find the optimal parameters among these in the grid search. (1st approach)
- *Optiparameters2()* find the optimal parameters among these in the grid search. (2cd approach)

In order to find the optimal Hyperparameters, I employed two different Gridsearch approaches:

- I achieve a Gridsearch on the: *learning_rate, batch_size, nb_hidunits* and *nb_hidlayer*. I use 3 values of each parameter in the parameter grid and fix the epoch number to 50, to reduce the computation time. Then, I repeat the Gridsearch for different parameter values.
  But the results are not very good because the training lose curve is lower than the validation one, from the first epochs. We can then conclude that our model doesn't generalise very well.

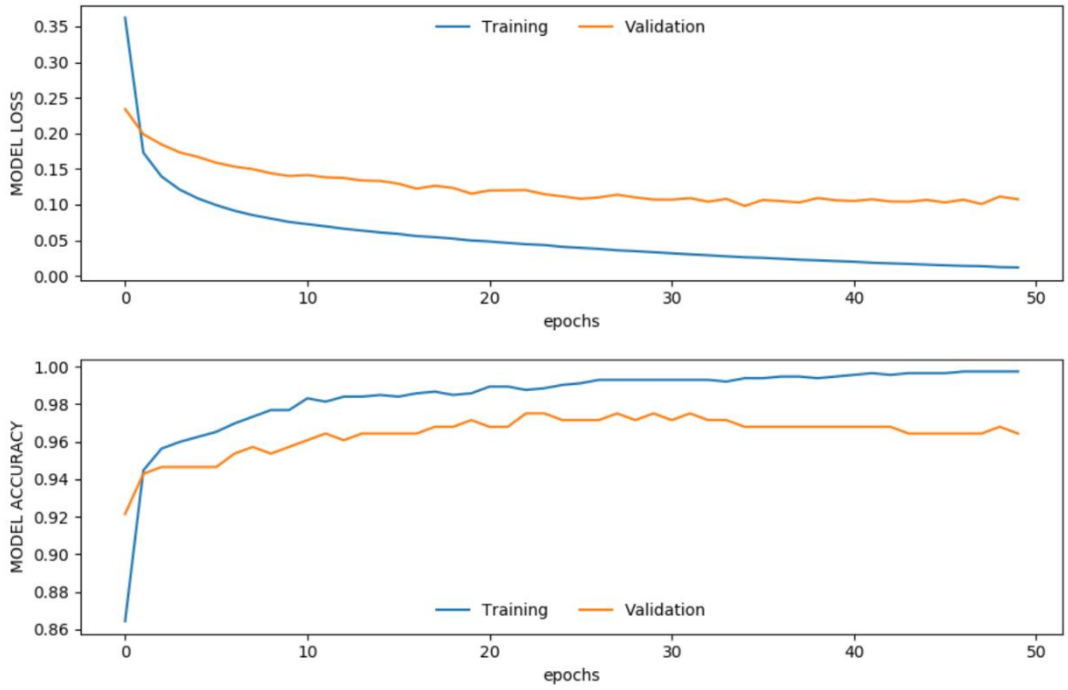


Figure 1: {'batch_size': 3, 'epochs': 50, 'learning_rate': 0.0005, 'nb_hidlayer': 1, 'nb_hidunits': 20} (score =0.978)

- I assume that the NN has 2 layers and achieve a Gridsearch on the: number on unit in *(Layer_1, Layer_2), batch_size*.I choose a very little learning rate for more precision (*learning_rate = 0,0001),*. I get better curves and a best score where we can easily differentiate when the overfitting starts.

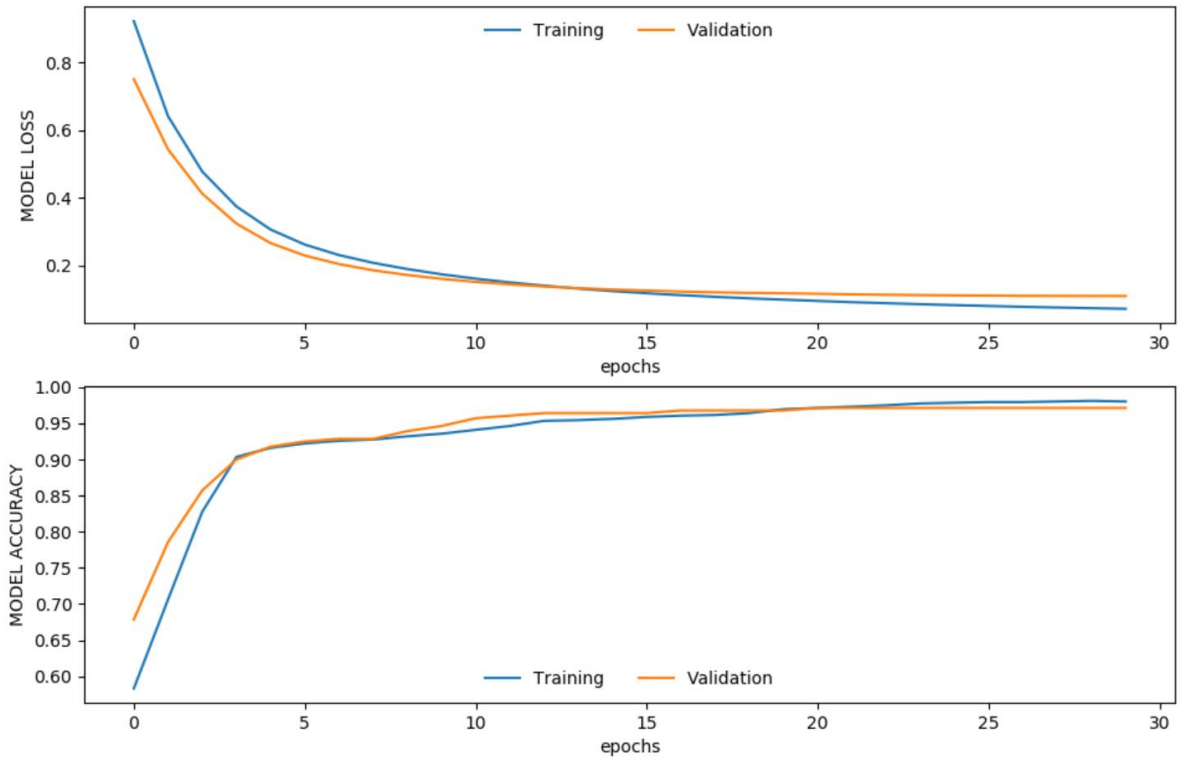
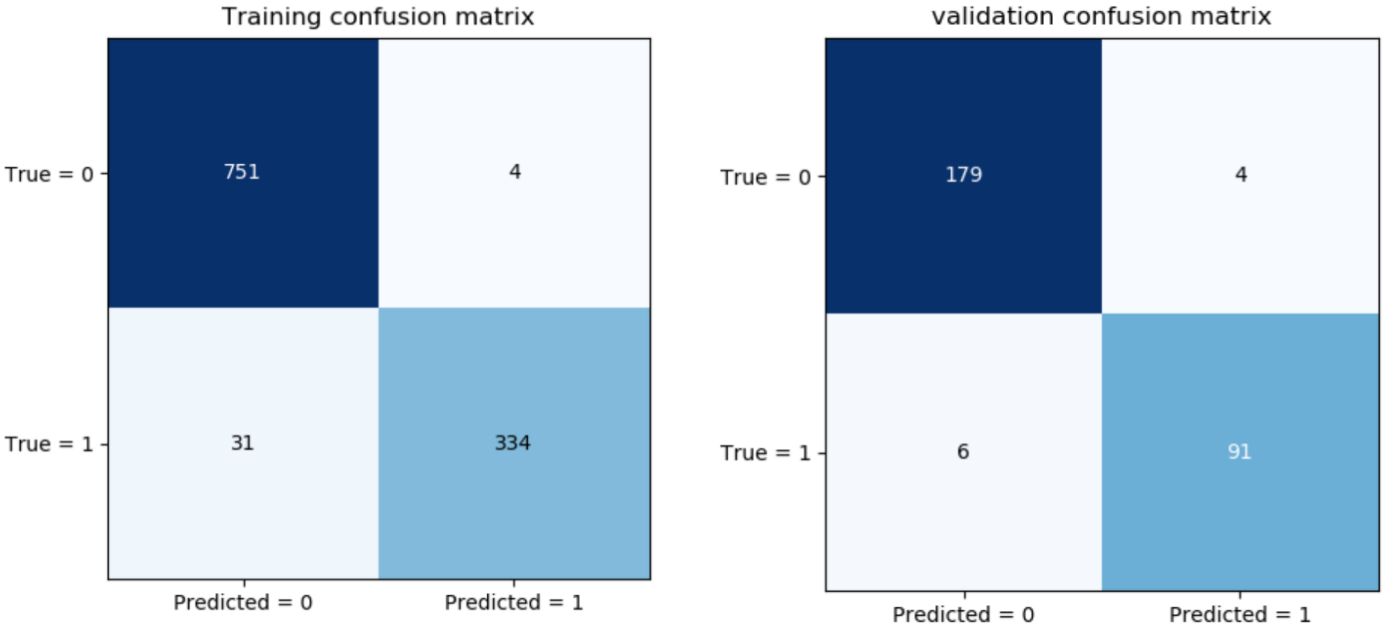


Figure 2: ***{'batch_size': 3, 'epochs': 30, 'learning_rate': 0.0001, 'nLayer_1': 15, 'nLayer_2': 15} (score = 0.982)***

- Final parameters***: batch size = 3, learning rate = 0.0001, nb_Layer = 2, nb_units = 15***

ii) Confusion Matrix



iii) Precision, recall, F1-score

In this part, I use the sklearn functions to calculate the metrics of both the training set and the validation test.

```
Training set Precision =  0.976
Training set Recall =  0.925
Training set f1 score =  0.95
Average of the 3 metrics =  0.95

Validation set Precision =  1.0
Validation set Recall =  0.913
Validation set f1 score =  0.955
Average of the 3 metrics =  0.956
```

iv) Difference between decision tree and random forest

These two methods are similar as they both use the concept of "trees". But they are different because on the one hand decision tree is built on an entire dataset. But on the other hand, a random forest randomly selects some rows and features in the data to build multiple decision trees from. After many trees are built using this method, we consider the result of each tree and take the solution with the higher result number.

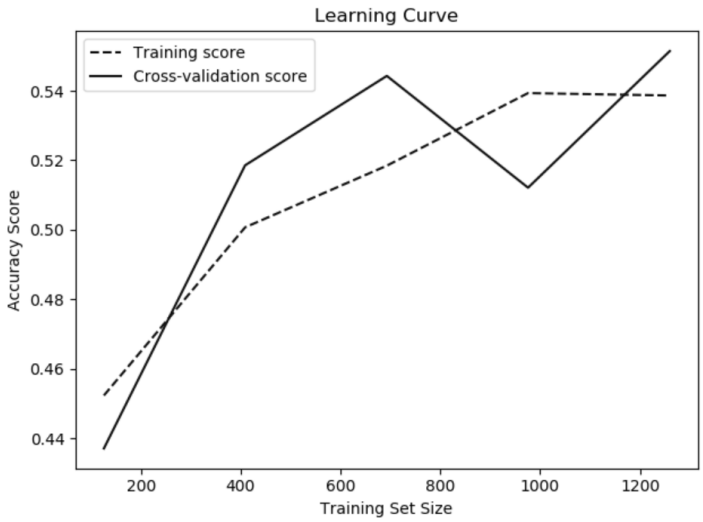
v) Accuracy, precision, recall and F1-score with decision tree and random forest

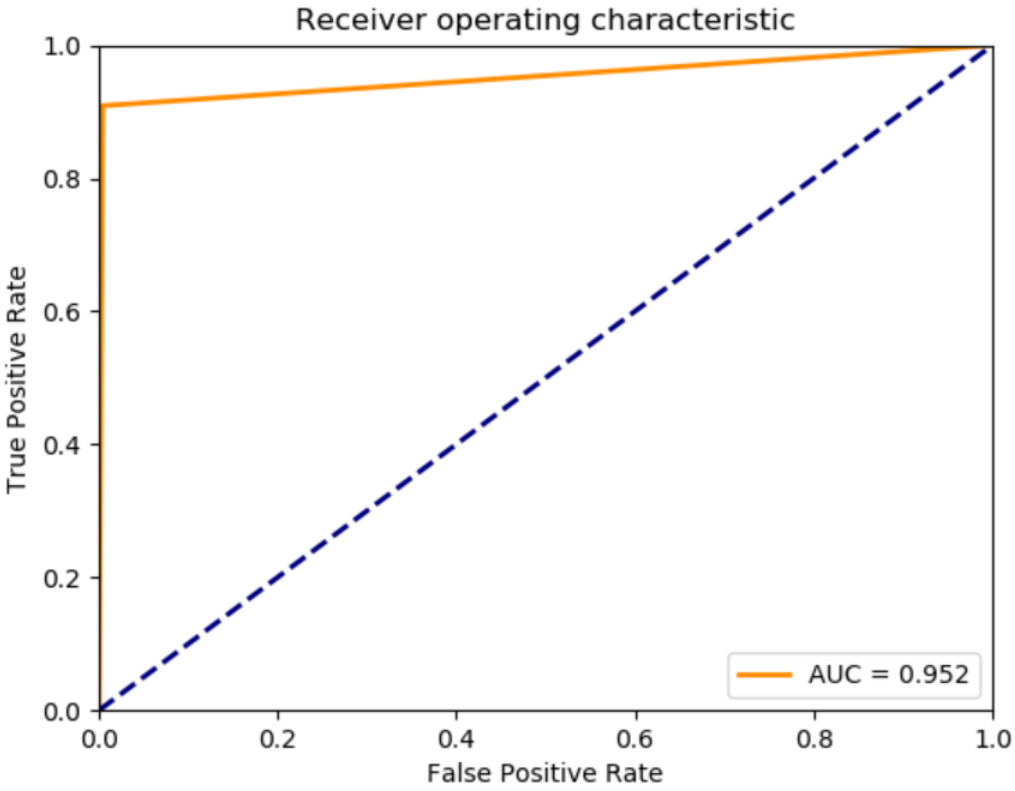
In this part, I use sklearn to get the metrics. For the random forest, I use a value of *n_estimators*=900. I get the following results with the decision tree better than the random forest.

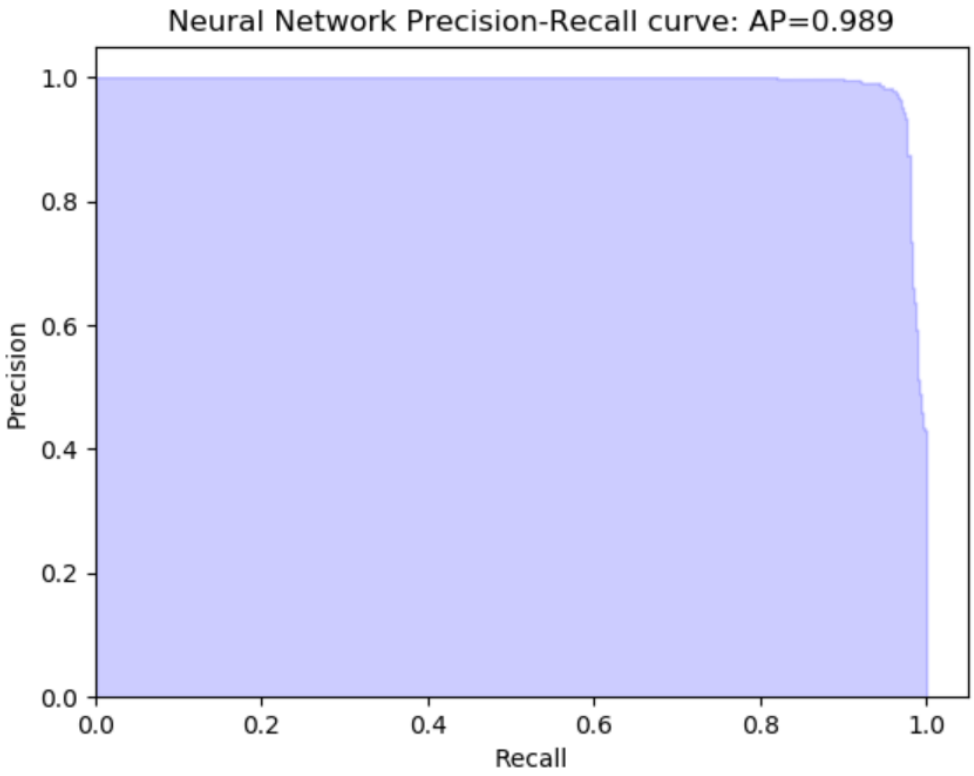
```
DecisionTree Acurracy =  0.996
DecisionTree Precision =  1.0
DecisionTree  Recall =  0.989
DecisionTree  f1 score =  1.326

RandomForest Acurracy =  0.989
RandomForest Precision =  1.0
RandomForest  Recall =  0.966
RandomForest  f1 score =  1.313
```

vi) Learning curve, ROC curve and precision curve

In this part, I use the sklearn functions to compute the elements on the Neural Network the we have created. This is my results:




Receiver operating characteristic
True Positive Rate
False Positive Rate
AUC = 0.952
0.0
0.2
0.4
0.6
0.8
1.0


Neural Network Precision-Recall curve: AP=0.989
Precision
Recall
0.0
0.2
0.4
0.6
0.8
1.0