Evaluation of supervised learning models for credit card fraud prediction

Pau Casé Barrera
paucase6@gmail.com

Table of Contents

Dataset Information

Source

The dataset contains transactions made by credit cards in September 2013 by European cardholders and was retrieved from Kaggle (ULB, 2017)..
No exact collection means is explained by the publisher, but it is said that it is real and therefore it may be a European bank who provided the data. The original source is not published because of confidentiality issues and is also the same reason the variable names are hidden.

The publisher institution is known and the complete explanation and the organizations involved point towards a real-world dataset.

Description

This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) accounts for 0.172% of all transactions.

It contains over 25 unidentified features, and two identified features: the time since the first transaction and the amount of the transaction. The transactions were made in September 2013, in two consecutive days.

There are 284,807 transactions (rows) and 31 columns: 30 features and one target variable, the label.

Original Usage

The dataset was collected and analyzed during a research collaboration of Worldline and the Machine Learning Group (http://mlg.ulb.ac.be) of ULB (Université Libre de Bruxelles) on big data mining and fraud detection.

Objective of the Project

The goal is to predict the fraudulent transactions by looking at the possible features of the dataset.

Data attributes and target variable

The variables are unnamed for reasons of privacy and confidentiality. All the variables are a result of a PCA transformation to further protect the data's origin, except for time and amount.

Column Name	**Data Type**	**Example Value**
Class (fraud or no fraud)	Integer	0
Time	Float	0
V1	Float	-1.35981
V2	Float	-0.07278
V3	Float	2.536347
V4	Float	1.378155
V5	Float	-0.33832
V6	Float	0.462388
V7	Float	0.239599
V8	Float	0.098698
V9	Float	0.363787

V10	Float	0.090794
V11	Float	-0.5516
V12	Float	-0.6178
V13	Float	-0.99139
V14	Float	-0.31117
V15	Float	1.468177
V16	Float	-0.4704
V17	Float	0.207971
V18	Float	0.025791
V19	Float	0.403993
V20	Float	0.251412
V21	Float	-0.01831
V22	Float	0.277838
V23	Float	-0.11047
V24	Float	0.066928
V25	Float	0.128539
V26	Float	-0.18911
V27	Float	0.133558
V28	Float	-0.02105
Amount	Float	149.62

Exploratory Analysis

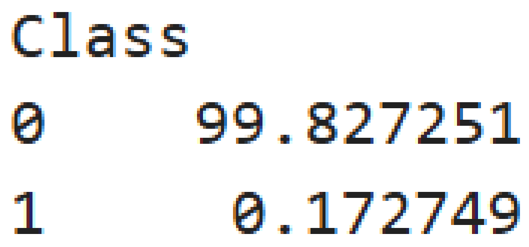
Missing Values and Outliers

There are no values missing from any column. Detecting outliers in this dataset was very tricky because of the lack of knowledge of the meaning of each variable and given that the dataset has undergone a PCA transformation, it was especially tricky to identify.

Therefore, all data points were kept, since it was preferred to work with some outliers rather than eliminating some valuable columns, especially considering the few rows labeled as fraud.

Distribution of the target value

The distribution of the target value is very unbalanced, only 0.17% of the rows are fraud. This is understandable when the data domain is taken into account: there are far more non-fraudulent transactions than fraudulent ones. The distribution is shown in the following picture:



```
Class
0     99.827251
1      0.172749
```

FIGURE 1:DISTRIBUTION OF THE TRANSACTIONS (FRAUD=1 VS NON FRAUD=0)

Correlations and important features

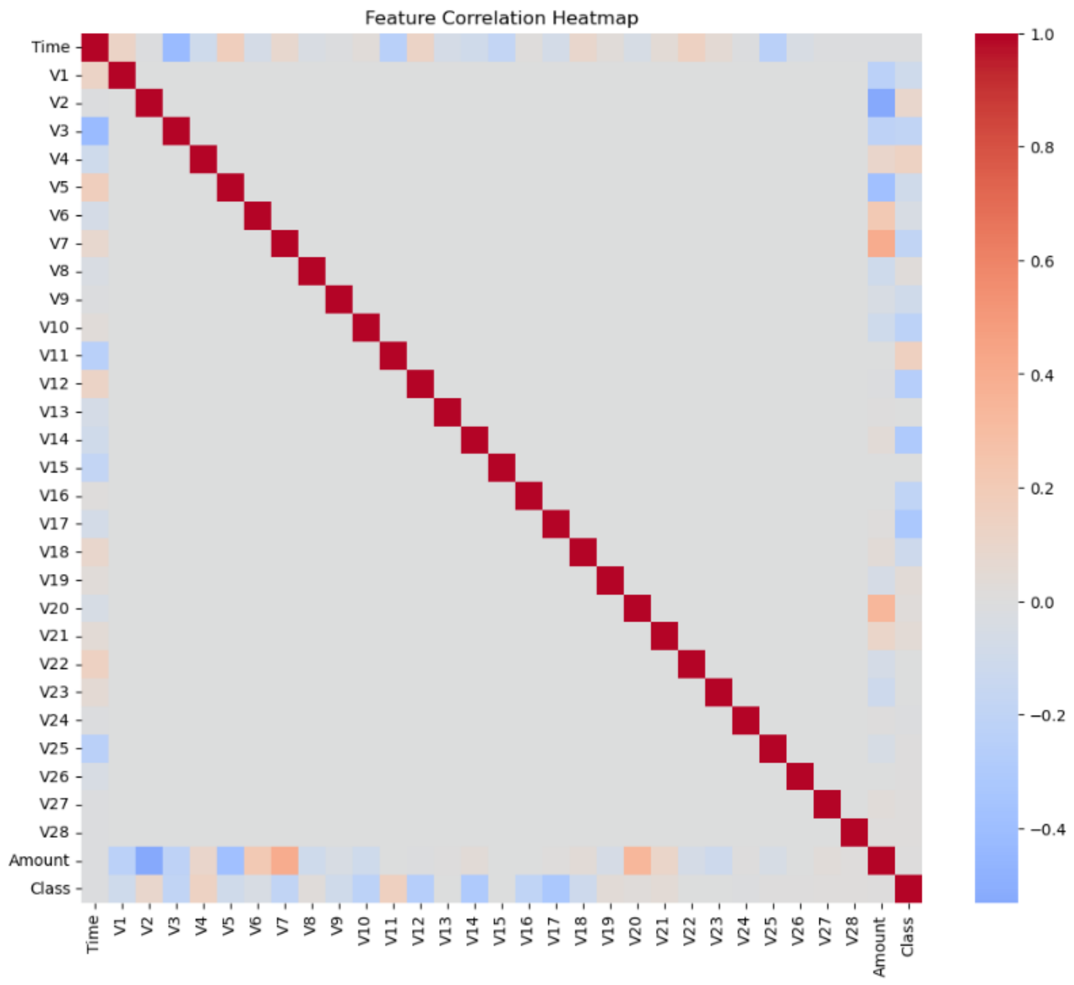


FIGURE 2: CORRELATION MATRIX OF ALL THE VARIABLES IN THE DATASET

There is absolutely no correlation among the unnamed variables. This is a result of the PCA transformation applied to protect the privacy of the data.

Therefore, thanks to this there are no leakage features for sure, although we can see some components are related to the class in a slightly direct or indirect relation.

Additionally, time and amount have virtually no correlation with the label.

The PCA transformation generated components instead of keeping the features, and no specific component seems to be clearly more important.

However, we can still classify variables by importance.

From the correlation matrix it seems that components 6, 8, 13, 15, and 19 through 28 are not very correlated with the target variable.

This happens for the amount and time variables too, as could be expected. It could be interesting to turn these into bins to see if there is some range of amounts that fraudsters aim for, and the same could be interesting for time to see if they operate in specific time ranges.

By contrast, the other components seem to have a higher correlation, though none exceeding a value of around 0.4.

Final comments on the data

Despite the uncommon situation of not knowing what features one is working with, the dataset was chosen because this topic has gained relevance in the last years, with fraud being detected very commonly. Adding to this difficulty, the number of fraud transactions is very low, so this dataset is a challenge that can provide many learnings.

Classification Methodology

Preprocessing steps

No preprocessing steps were done on the already processed data. A test was conducted which transformed the continuous time and price variables into discrete variables, but minor differences were found in the results.

Classification methods used) along with hyperparameters that were adjusted

The three methods used were:

- Random Forest
- Decision Tree
- K-Neighbor Classifier

Random Forest

The random forest model was quite slow to process given the large number of rows in the dataset. The adjustments of the hyperparameters had to be done on a very small portion of the data: 10%, around 15000 samples, keeping the ratio of positives of the whole data. To test, cross-validation tests were done, with them also keeping the proportion of fraudulent transactions.

The first test was done to evaluate the accuracy of the cross validation runs, testing different maximum depths for the trees.

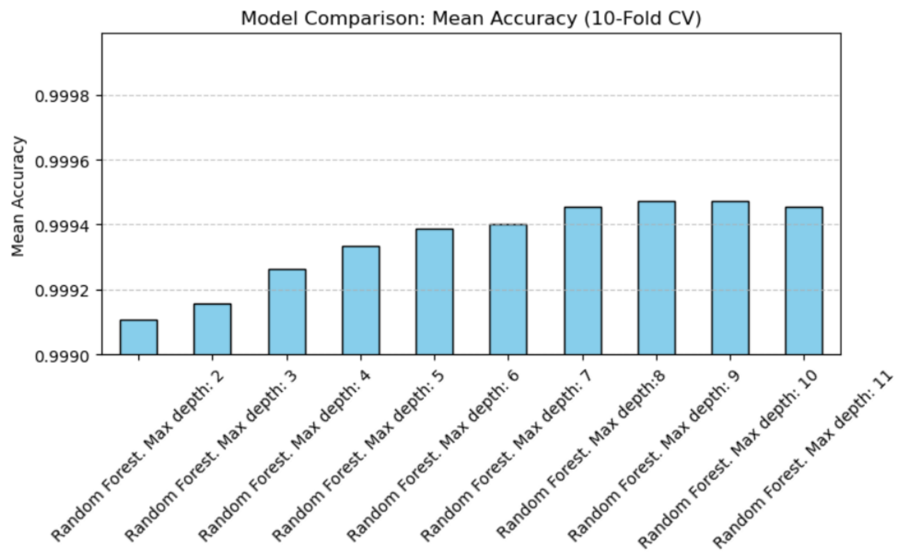


FIGURE 3: ACCURACY COMPARISON AMONG RANDOM FOREST MODELS WITH DIFFERENT DEPTHS

It looks like accuracy increases the deeper the trees are built, but it also shows that after a maximum depth of around 8, the model might be overfitting, so the final model used this maximum depth.

Now, the average_precision_score scikit-learn function was used to calculate a value like the area under the precision-recall curve. This metric was used to check if the model was performing well: This time, maximum depths from 3 to 9 were checked with two different fixed number of estimators: 100 and 50 (unlabeled in graph) were tested.

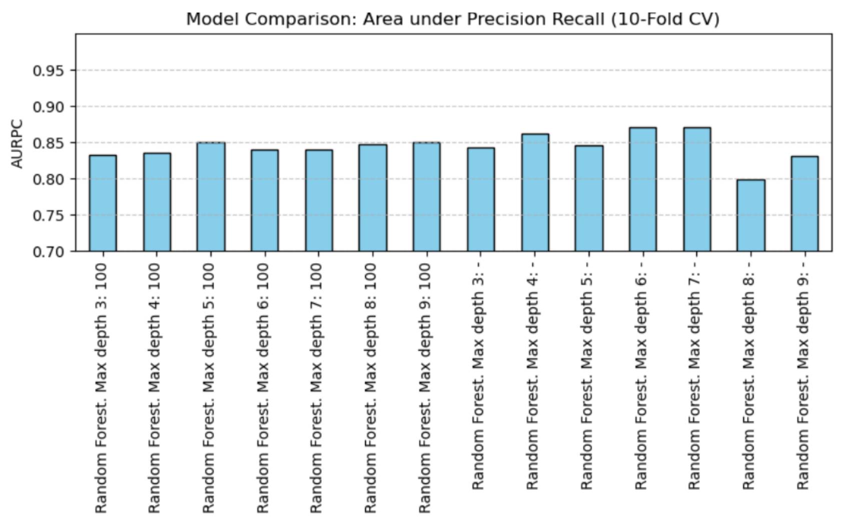


FIGURE 4: AUPRC COMPARISON AMONG RANDOM FOREST MODELS WITH DIFFERENT DEPTHS

No clear relationship is seen between the depth and this metric, so the two graphs lead to thinking that a maximum depth of 7 may be the best option.

Further tests were done with a constant max depth of 7, now varying the number of estimators (trees), and the results showed that any number over 50 got similar results. This may be because of the very low number of fraud cases, and not enough test cases to test off given the dataset was reduced to be able to train on a regular computer.

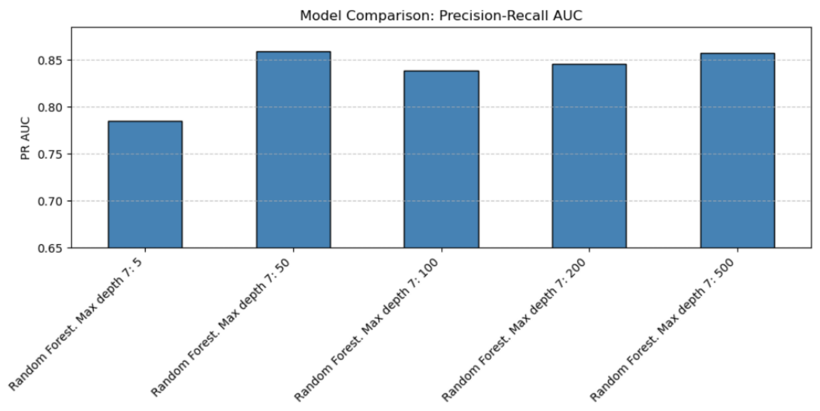


FIGURE 5: ACCURACY COMPARISON AMONG RANDOM FOREST MODELS WITH DIFFERENT NUMBER OF TREES

This resulted in the graph shown above, and clearly the number of trees affected the results, but not in a linear manner, although a slight improvement seems to happen the more trees are used.

After these tests, the Random Forest Classifier was adjusted at a max depth of 7 with 50 trees, which allowed for good results and a reasonable time to train and test.

Decision Tree

The Decision Tree was also tested with multiple depths to check the performance of the model, using the same metrics as in the random forest tests: area under precision recall curve and accuracy.

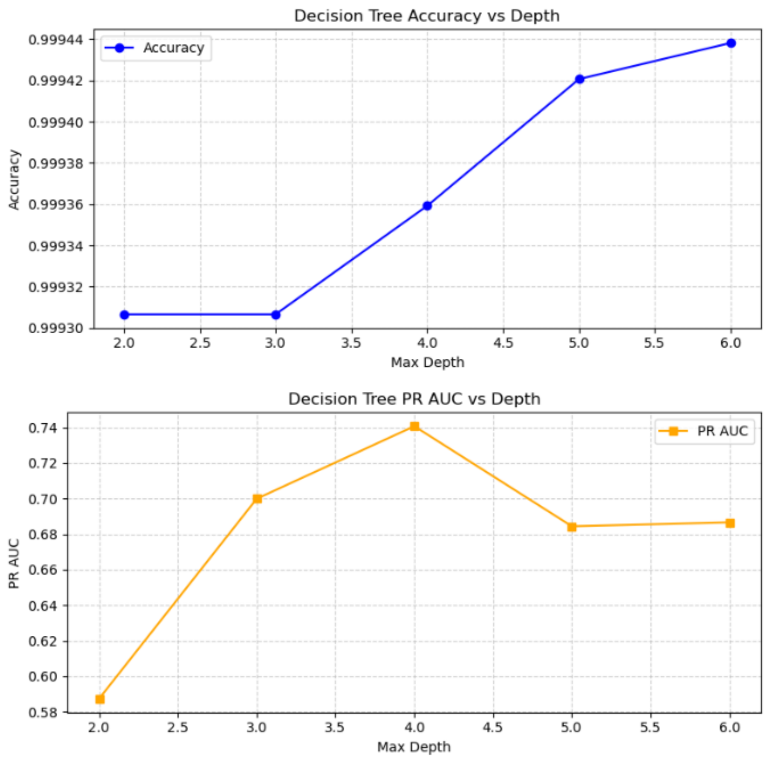


FIGURE 6: ACCURACY AND AUPRC COMPARISON AMONG DECISION TREE MODELS WITH DIFFERENT DEPTHS

The lower values seem to not represent the model as well, but from a depth of 3, the results look quite good. These tests were run on a split of 40% of the dataset to lower runtime.

Once observing the results, the depth 4 tree was chosen for the final model. And it was later tested on the full dataset for the results. This was done because it looks like it could be the best value, although it might have been coincidental because of how the splits were formed for the cross-validation.

K-Neighbor Classifier

To find the best hyperparameters for the K-Neighbor Classifier, more tests were conducted, which monitored improvements on the model's train set using cross validation on a 60% of the dataset, since this classifier was faster than the other two. First, tests were done on the number of K, fixing the distance metric to Euclidean, and with uniform weights. Accuracy stayed mostly constant, and the area under the precision-recall curve had a similar effect.

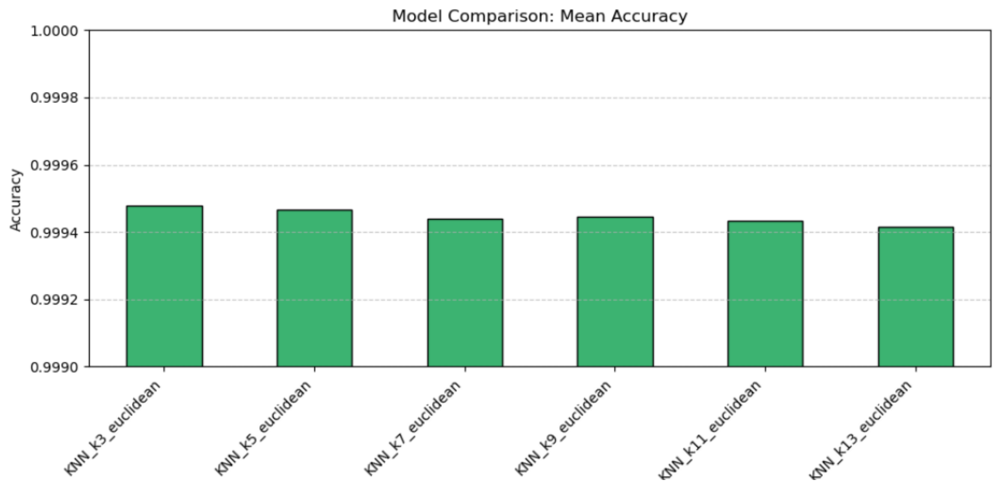


FIGURE 7: ACCURACY COMPARISON AMONG K-NEIGHBOR CLASSIFIER MODELS WITH DIFFERENT K VALUES.

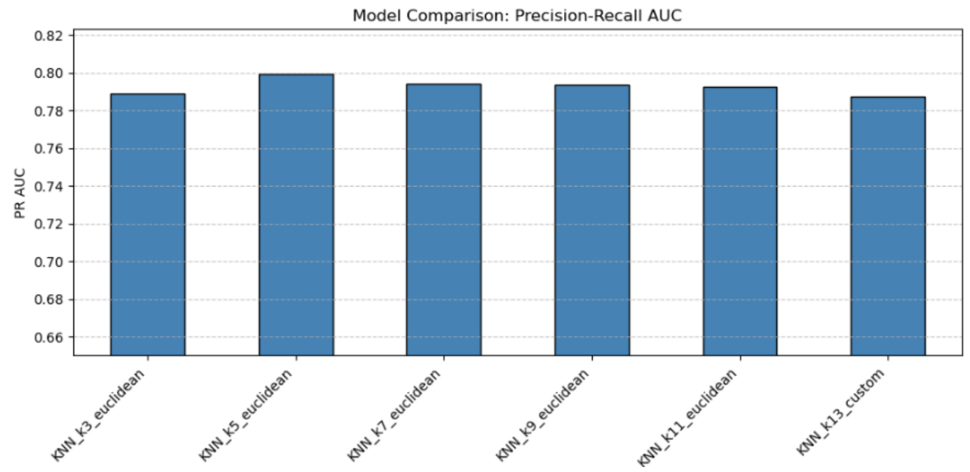


FIGURE 8: AUPRC COMPARISON AMONG K-NEIGHBOR CLASSIFIER MODELS WITH DIFFERENT K VALUES.

No clear relationship was found no matter what k value was used, so further tests were conducted to evaluate different weight metrics. The following chart shows a similar test, now using weights which are inverse to their distance of the data points, as opposed to the previous tests which used uniform weights.

The results below seem to be better in the precision recall area under the curve. Therefore, this weight setting will be kept at ‘distance’ for the final model.

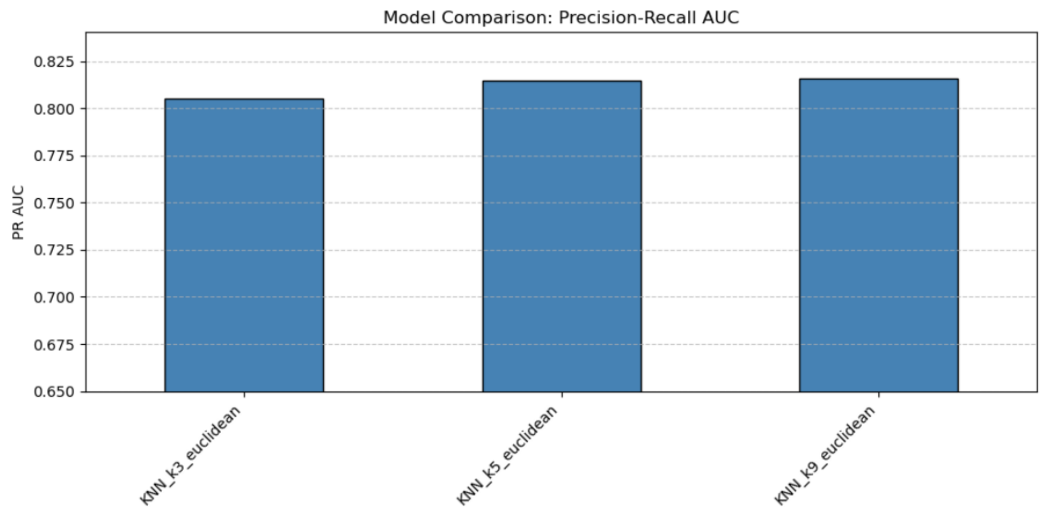


FIGURE 9: AUPRC COMPARISON AMONG K-NEIGHBOR CLASSIFIER MODELS WITH DIFFERENT K VALUES. USING DISTANCE-RELATED WEIGHTS

The final K-Neighbor Classifier model will have a k of 5, with a Euclidean metric and a weight setting of distance in the standard scikit-learn implementation.

Classification evaluation

To evaluate the models, the big imbalance between 0s and 1s in the target class must be considered. In these cases, it is a good idea to plot a graph with precision recall values which are built by trying different thresholds on the classification step. The model generates a probability for each possible classification, and the threshold will determine whether it is classified as one or the other. The area under the curve from the plotted graph is an interesting measure for these cases. (Leevy, 2023)

Additionally, a confusion matrix was used to understand what values were achieved with the ideal threshold value, and accuracy was also checked, although the values were not so relevant, because of the very large imbalance in the predicted feature.

Results

Method	**Hyperparameter settings**	**Accuracy (using 10-fold c.v.)**
Random Forest	Max_depth=7, num_estimators=50	0.999161
Decision Tree	Max_depth=6	0.999105
Decision Tree	Max_depth=6, threshold=0.003	0.998710
K-Neighbor Classifier	n_neighbors=5, metric=euclidean, weights=distance	0.998502

The table does not include previous tests as this is explained in the methodology section. Only parameters that were found to be best were run on the full dataset because of time constraints.

Random Forest

After conducting the tests on 10% of the data to allow for multiple runs to be done, it was likely that when running on the full dataset, the Random Forest would underperform compared to the previous tests. The 10-fold cross-validation demonstrated exactly this.

```
Accuracy: 0.999161
PR AUC:   0.628041
```

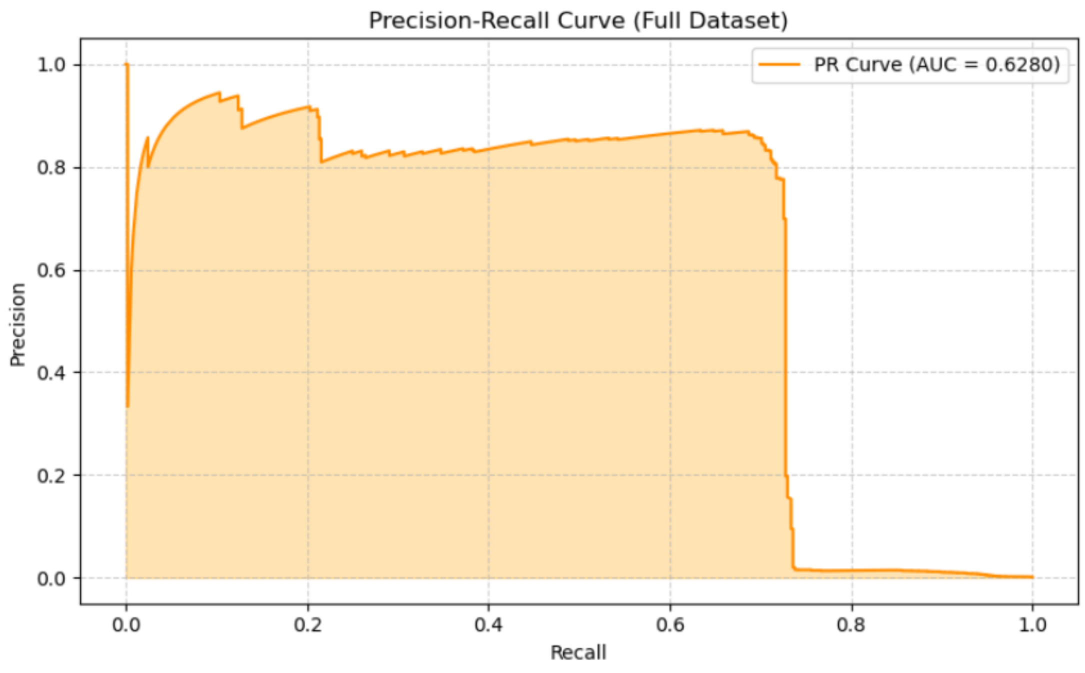


FIGURE 10: PRAUC AND ACCURACY VALUES AND P-R CURVE PLOT ON THE FINAL RANDOM FOREST MODEL

It is possible that this decrease in the area under the curve has happened because the parameters chosen were perfected for a dataset of a smaller dimension and may have failed to work as well for this larger dataset even if the distributions were similar. The parameters set also allowed us to have reasonable runtimes.

Despite the worse performance, the model could still predict a good number of fraud transactions correctly, as can be seen in the confusion matrix.

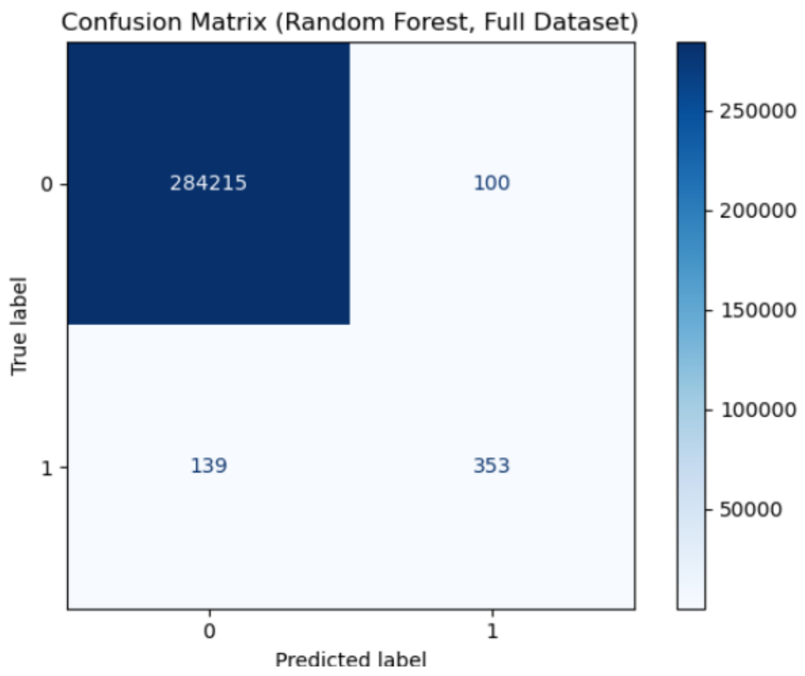


FIGURE 11: CONFUSION MATRIX OF THE FINAL RANDOM FOREST MODEL

No other less restrictive tests were run because a modification of the restrictive parameters resulted in more trees and of larger depth. This led to a far larger runtime was too high to test on the full dataset.

Decision Tree

The Decision Tree with a maximum depth of 4 gave better results overall, allowing us to get a larger value for the area under the Precision-Recall curve than any other model tested. The results of this were reflected in the confusion matrix too.

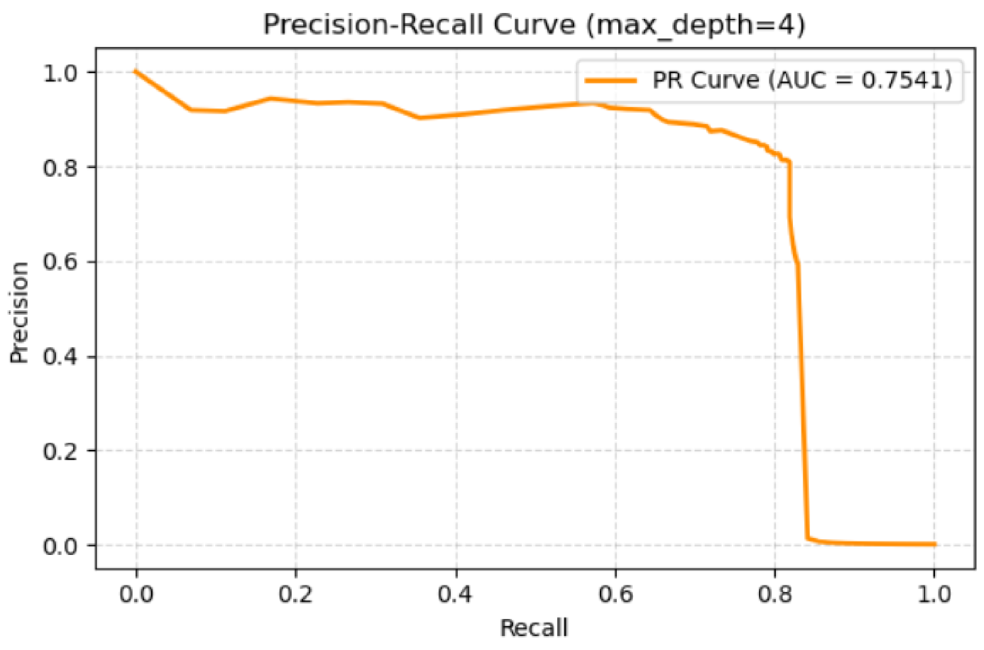


FIGURE 12: P-R CURVE PLOT ON THE FINAL DECISION TREE MODEL

This model was the best performant of the three. Therefore, a final tweak was applied by changing the threshold values to classify as one class or the other. The goal was to increase the number of fraudulent cases found, even if that came at the cost of non-fraudulent cases labeled as fraud. The

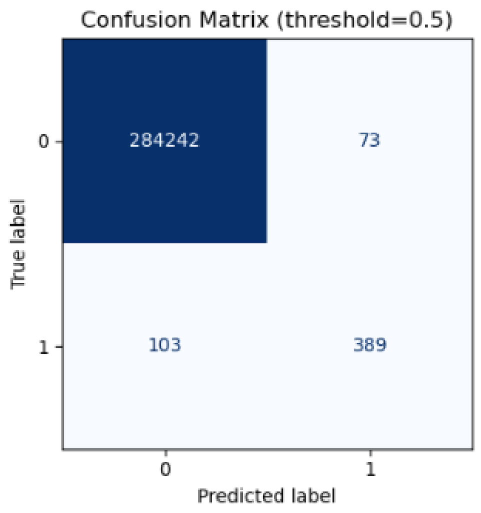


FIGURE 14: CONFUSION MATRIX ON FINAL DECISION TREE MODEL. AUTOMATIC THRESHOLD

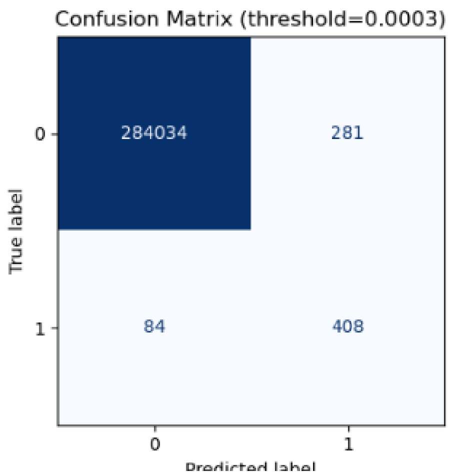


FIGURE 13: CONFUSION MATRIX ON THE FINAL DECISION TREE MODEL - MODIFIED THRESHOLD

two following confusion matrices represent the two options the model would provide to a bank, depending on their preferences.

From these two confusion matrices we can see that a small improvement in fraud case detection comes at a high cost. About 10 times as many non-fraudulent cases are labeled as fraud for each extra correct one with the change in the threshold.

K-Neighbor Classifier

Using the K-Neighbors Classifier, the area under the curve got very bad results, and therefore no threshold was found that gave a good recall-precision balance.

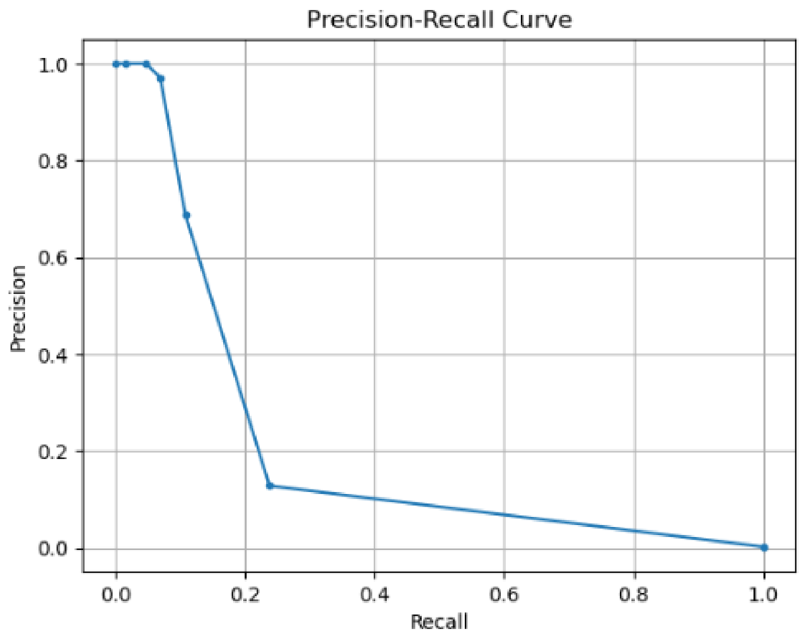


FIGURE 15: P-R CURVE PLOT ON THE FINAL K-NEIGHBOR CLASSIFIER MODEL

By looking at the confusion matrix, we can see that it could barely find any fraudulent cases, although the number of non-fraudulent cases that were labeled as fraud by the model were very low. This is probably the opposite of what a bank looking for fraud would want.

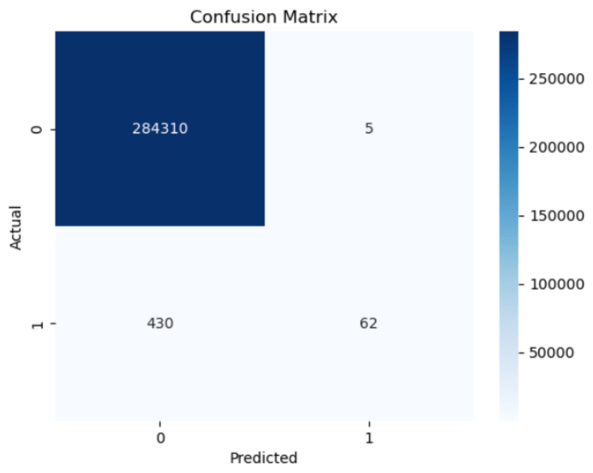


FIGURE 16: CONFUSION MATRIX ON THE FINAL K-NEIGHBOR CLASSIFIER MODEL

The reason why the K-Neighbors Classifier did poorly on the whole dataset cross validation is unclear, since the cross-validation steps had a similar distribution of each class and the 60% it was first tested on gave far better results with the same hyperparameters and same fraud case distribution. When using the default parameters on the full dataset, the results were even worse, around a 0.11 Precision-Recall AUC.

Conclusions

Having a lot of data is beneficial and is important especially when the target feature is very imbalanced. In this dataset, though, it was proven that while it can be beneficial, it also needs more computing power.

The Random Forest did not perform well on the full dataset possibly because of the restrictive parameters that had to be set to be able to get a runtime that was feasible. The decision tree with a depth of 4 performed, possibly because of its simplicity and good generalization. The K-neighbor classifier did not work well compared to the other two models, which was surprising given the good results it provided with 60% of the data.

Bibliography

Leevy, J. H. (2023). Threshold optimization and random undersampling for imbalanced credit card data. *Journal of Big Data*.

ULB, M. L. (2017). *https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud*. Retrieved from Kaggle.