Report Assigment 1 APML24/25@UU

Rens van Moorsel, Ruben de Groot, Ewoud ter Wee, Lucas van der Jagt
Group 17

November 27, 2024

Abstract

This paper is about experimenting with 3 algorithms: Decision Tree, Random Forest and Isolation Forest in the context of outlier detection. After training, these algorithms will be compared with different quality scores: accuracy, precision, recall and F1 score. The paper contains some data visualization and experimenting with the hyper parameters of the 3 different models. At the end of the paper it is concluded that Decision Tree and Random Forest are equally good and are working a lot better with outliers then the Isolation Forest algorithm, but all models are too simple for outlier detection.

1 Introduction

The data used in this assignment is data from a marketing campaign of a Portuguese bank, within this data there are labels indicating if a data point is an outlier. The task is to use this data to train outlier detection algorithms in a supervised and an unsupervised way. Outliers are data points that differ from the rest of the dataset. Outlier detection is very useful for pre-processing datasets, as outliers will decrease the machine learning model performance, as outliers disrupt the patterns in the data. There are a lot of different methods to find outliers in a dataset. This experiment implements three of these methods and compares the F1 scores between them. The three methods used in this experiment are: Decision Three, Random Forest and Isolation Forest.
First, the data is analyzed by visualizing the data, so the dataset can be understood properly.

2 Data

Before building any predictive model it is crucial to thoroughly understand the data. A clear understanding of the data ensures that the underlying patterns, relationships, and characteristics are identified. Analyzing and exploring the data, helps by a better understanding of the descriptions of the data, the ratio of inliers to outliers, the correlation between the features and the class, and the correlation among the features themselves. Only then the data can be used for training a model.

2.1 Descriptions of the data

The data used in the experiment is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed. The data set was preprocessed and the name of the file was "dataBank-additional-full_normalised.csv".

The data set has 40188 rows (samples) with each 63 features. The dataset includes both numerical and categorical data, with the categorical features one-hot encoded. So the data types are int64 or float64. The categorial features are for example: Job categories, Marital status, Education levels, etc..

Feature	Count	Mean	Std. Dev.	Min	25%	Median	75%	Max
Age	40188	0.2841	0.1279	0.0000	0.1852	0.2593	0.3704	1.0000
Job = Housemaid	40188	0.0256	0.1580	0.0000	0.0000	0.0000	0.0000	1.0000
Job = Services	40188	0.0961	0.2947	0.0000	0.0000	0.0000	0.0000	1.0000
Job = Admin	40188	0.2523	0.4343	0.0000	0.0000	0.0000	1.0000	1.0000
Job = Blue-collar	40188	0.2259	0.4182	0.0000	0.0000	0.0000	0.0000	1.0000
Job = Technician	40188	0.1641	0.3704	0.0000	0.0000	0.0000	0.0000	1.0000
Job = Retired	40188	0.0410	0.1983	0.0000	0.0000	0.0000	0.0000	1.0000
Job = Management	40188	0.0712	0.2571	0.0000	0.0000	0.0000	0.0000	1.0000
Job = Unemployed	40188	0.0246	0.1548	0.0000	0.0000	0.0000	0.0000	1.0000
Job = Self-employed	40188	0.0347	0.1830	0.0000	0.0000	0.0000	0.0000	1.0000
...								
Previous	40188	0.0241	0.0696	0.0000	0.0000	0.0000	0.0000	1.0000
Poutcome = Nonexistent	40188	0.8659	0.3408	0.0000	1.0000	1.0000	1.0000	1.0000
Poutcome = Failure	40188	0.1025	0.3034	0.0000	0.0000	0.0000	0.0000	1.0000
Poutcome = Success	40188	0.0316	0.1749	0.0000	0.0000	0.0000	0.0000	1.0000
Emp. var. rate	40188	0.7286	0.3256	0.0000	0.3333	0.9375	1.0000	1.0000
Cons. price idx	40188	0.5367	0.2248	0.0000	0.3406	0.6033	0.6988	1.0000
Cons. conf. idx	40188	0.4303	0.1926	0.0000	0.3389	0.3766	0.6025	1.0000
Euribor 3M	40188	0.6813	0.3913	0.0000	0.1610	0.9574	0.9810	1.0000
Nr. employed	40188	0.7724	0.2706	0.0000	0.5123	0.8597	1.0000	1.0000
Class	40188	0.1030	0.3040	0.0000	0.0000	0.0000	0.0000	1.0000

Table 1: Statistical description of the dataset.

In table 1 above (a part of) the normalized data with some metrics are given like the mean, standard deviation and the median. A couple of things can be concluded from this like admin and blue-collar have a relatively high mean so it can be concluded that is a frequent job. Previous has a very low mean (0.0241) and standard deviation (0.0696), suggesting it is typically zero but occasionally non-zero. And Euribor 3M and emp.var.rate have wide ranges and meaningful variation (standard deviations of 0.3913 and 0.3256), which might make them more impactful in capturing patterns related to the class.

2.2 Correlation of the features

To decide which features are useful to predict if a client is subscribed to the product, it is important to visualize the data. A good way of checking the correlation between variables is a correlation matrix. This matrix shows which variables are related or inversely correlated to each other.

The features with the highest positive and negative correlations with the class provide the most insight into whether an instance is classified as an outlier or an inlier. From the figures below, several features stand out. The following features exhibit high correlation: Duration with 0.39, Poutcome=success with 0.31, and Previous with 0.22. Conversely, certain features have a high negative correlation, including Pdays with -0.32, Nr. employed with -0.34, Euribor 3M with -0.30, and Emp.var.rate with -0.29. These features alone already offer a strong indication of whether an instance is an outlier or not.

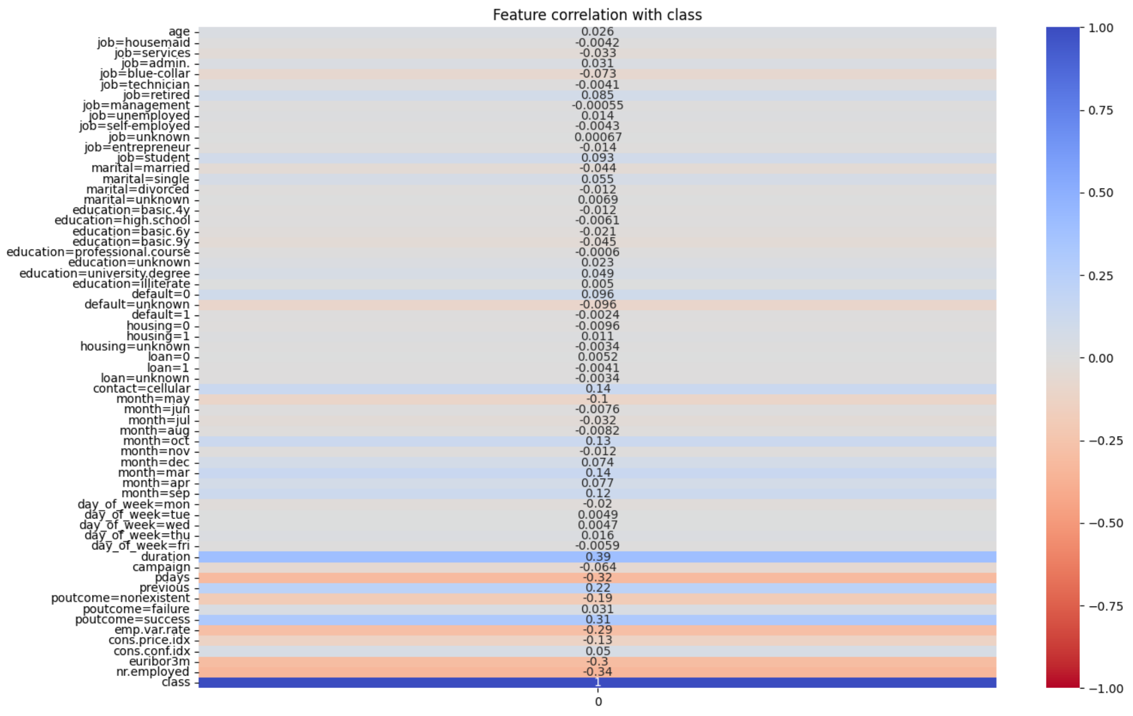

Figure 1: Feature correlation with class

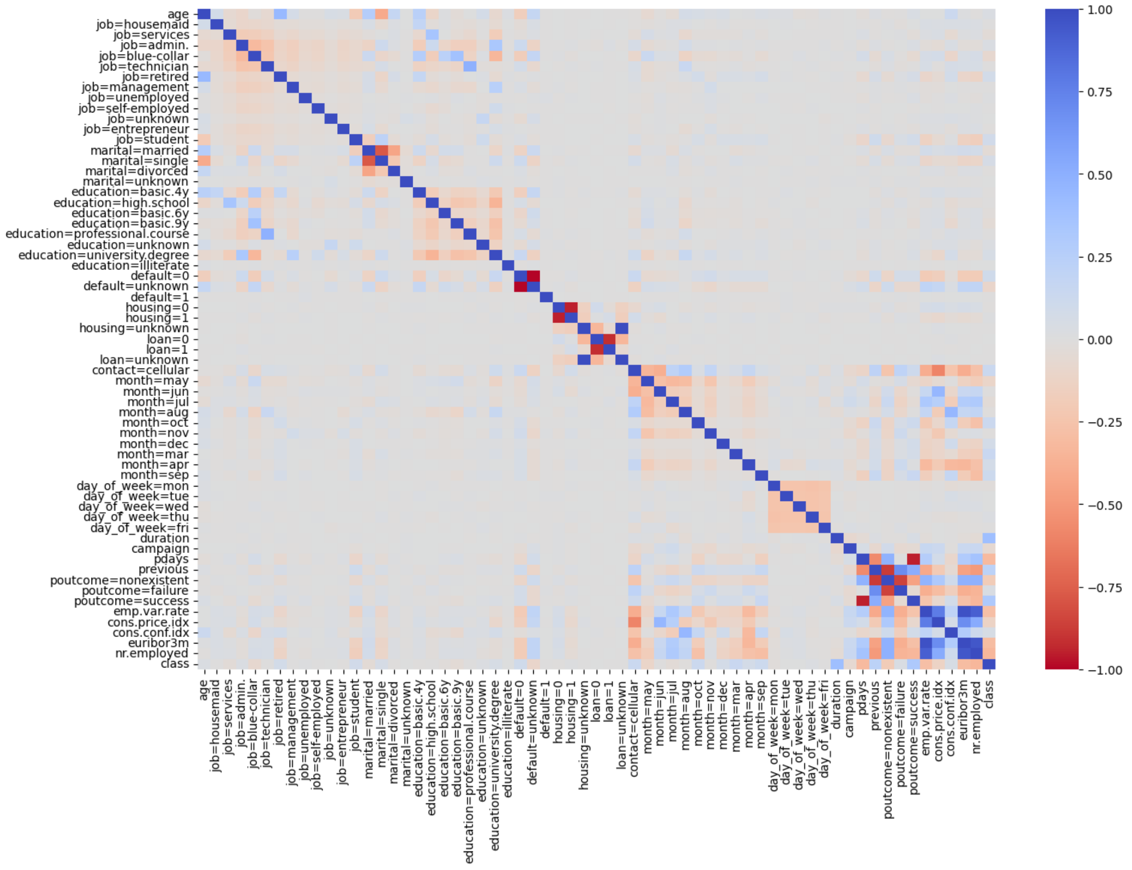

Figure 2: Correlation matrix between features

It is also interesting to examine the correlation between these features themselves. When looking at the correlation between all the features that have a high correlation with the class, no notable relationship are observed. Duration, Poutcome=success, and previous seem to have little influence on each other.

However, the situation becomes more interesting when considering the correlation between features that have a high correlation with the class and features that have a high negative correlation with the class. For example, Pdays and Poutcome=success have a correlation of -0.95. These features not only have a strong influence on the class but even a stronger influence on each other. This is also evident for Poutcome=nonexistent and previous, which exhibit a strong negative correlation.

While the features with high correlation to the class did not seem to influence each other much, it can be observed that this is not the case for the features with a negative correlation to the class. For instance, Nr unemployed, Emp.var.rate and Euribor 3M all have a very strong correlation with each other and are negatively correlated with the class.

2.3 Distribution of outliers

Moreover, it is important to examine the ratio of inliers to outliers in the dataset. This is especially helpful for the model selection and quality measures later on. The objective is to employ an algorithm capable of assigning a label of "1" to an instance, indicating that the instance is an outlier, or a label of "0," signifying that the instance is an inlier. The graph below visualizes the distribution of inliers and outliers within the dataset. It reveals that there are approximately 36,000 inlier instances, while the remaining 4,000 are outliers. Thus, the proportion of outliers in the dataset is approximately 1/9. From this, it can be concluded that the classes are imbalanced, as the inliers and outliers are unequally distributed.

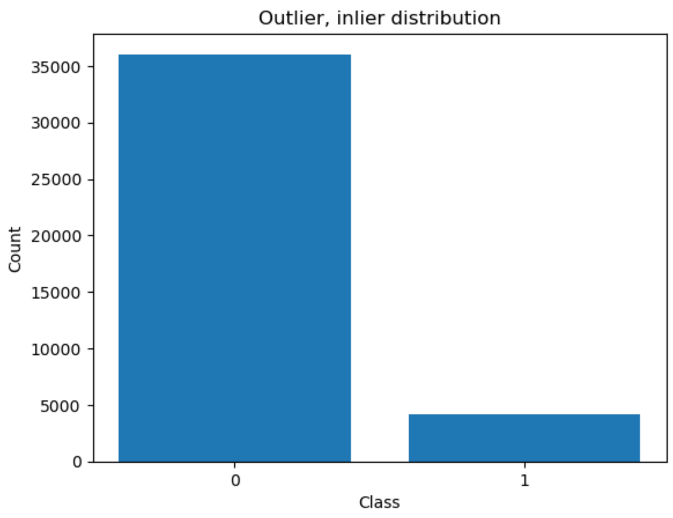

Figure 3: Distribution between outliers and inliers

3 Methods and Experimental Setups

3.1 Experimental Setup / Model Selection

For the experiment, two algorithms will be trained with 80 percent of the data. The other 20 percent will be used to measure the quality of the trained algorithm. These percentages are usual in machine learning [1] [3] as 20 percent is enough to check if the model is not overfitted [1] [3] and 80 percent is still more then enough to train the algorithm [2] [3]. These two algorithms will be checked with the use of 5-fold cross validation. Furthermore one other algorithm will be trained with 100 percent of the data. The validation of this algorithm will be done with the complete dataset. The algorithms used are:

- Decision Tree: A decision tree contains decision nodes and leaf nodes. It works by splitting the data at a node to reduce the impurity as much as possible until a leaf node is reached. In a perfect split situation, a leaf node will only contain 1 class, in reality, sometimes it is needed to adjust some hyper-parameters like maximum depth of the tree or the minimum amount of samples in a leaf node. This can then result in higher impurity than desirable in a leaf node, but this can be a good thing as this can increase generalization performance and reduce over-fitting.

  - Cons:
    Decision trees are not amazing for the domain. A decision tree will try to reduce impurity as much as possible at each node. However the dataset is highly imbalanced at about 90 percent inliers and 10 percent outliers. This means that the inliers are a lot more heavily weighed than the outliers and reducing the impurity will mainly focus on correctly classifying the inliers.

    Another thing to note is that outlier could or could not be clustered together. If outliers are not clustered together the decision tree will not lose much impurity by splitting on those singular datapoints and will probably ignore it for other splits.

  - Pros:
    Decision trees are not all bad however, in outlier detection the data is usually not linearly separable, which decision trees can handle well, decision trees are also very simple and easy to explain. Irrelevant features will not be accounted for in a split and collective outliers far away from the data can be easily split for reducing impurity.

- Random Forest: To motivate the choice for Random Forest, it might be good to first point out a flaw of decision trees: their hypothesis space is really large. A slight adjustments in a few datapoints can result in a hugely different decision tree.

  Random forests are in essence a lot of decision trees and they counteracts this problem by first bootstrapping the data, meaning the data is randomly sampled with replacement. This is done n times to create n new datasets, for each dataset the features are bagged, meaning x features are randomly taken where x =< features-count. This will reduce the diversity of the trees. The name forest comes from the idea that the prediction is done by letting the datapoints go through all the trees of the forest and taking a majority vote for each tree.

  Bootstrapping makes sure each tree contains the same data, making it less sensitive to changes in the training data and bagging makes sure the trees stay diverse. Bagging may results in "good" and "bad" trees, but because it is randomly sampled, it can be assumed that these cancel out. This majority vote will reduce overfitting.

- Isolation Forest: The previous described methods were supervised, so both the features and the label are used for the training. In Isolation forest this is not the case. This method is unsupervised. So for the training only the features are used. Without the labels. Here the method rest on the following idea. And that is that an outlier is easier to isolate on average, when randomly selecting one of the features, and randomly select a value to split. The isolation trees are grown until each point is isolated or it converges according to another hyper parameter. In this way a Forest of isolation trees is build. After this the assumption that the average path length to isolate an outlier is shorter than the path of an inlier can be used. For this it is possible to set a threshold for the path length to classify outliers.

3.2 Evaluation Metrics

After the training, the quality of the algorithms will be measured. To measure the quality, the following methods will be used:

- Accuracy Score: Accuracy measures how often a predication is correct, albeit positive or negative. In the context of outlier detection, this quality measure is not great. If 90 percent of data are inliers and the model will indentify all the inliers correctly and all the outliers incorrectly, the accuracy score will still measure 90 percent.
- Precision Score: The Precision Score measures how many of the detected outliers are really an outlier. It does not measure how many outliers are missed.
- Recall Score: The Recall Score tells something about how many outliers are detected as outlier. However, it does not look at inliers that are detected as outlier.
- F1 Score: The F1 Score is a balance between Precision Score and Recall Score. They are counted equally. The F1 Score gives a good view about how many outliers are detected, as well as how many inliers are wrongly detected as outlier.

In the domain, the goal is to detect as many outliers as possible, while being as precise as possible. Neither recall or precision is much more important as the other, so the F1 score would be a great measurement of quality, while the accuracy is the worst one.

3.3 Implementation Details

During the experiment, various Python libraries are needed. For importing and managing data Pandas is used. The graphs are plotted with use of the matplotlib library, and the heatmaps are plotted with seaborn. To train and test the models and split the data, the SkLearn library is used. Numpy is used for calculations.

The three models all need parameters. To make sure the models are trained optimally, these parameters must be selected carefully. Each model has the following parameters:

- Decision Tree:
  - max_depth_range: This is a list of max depth values that are used to train the model. To find the optimal model, multiple max depth values are tested. The max depth tells the algorithm how many levels the tree can contain. In this experiment there is chosen for the following range: 2-8. This range is chosen because this balances simple and complex trees. This way the trees do not become too complex that they capture noise in the data.
  - min_samples_leaf_range: The min samples leaf range is a list of min samples leaf values that are used to train the model. To find the optimal model, multiple values are tested. The min samples leaf parameter specifies the minimum number of samples required to form a leaf node. In this experiment there is chosen for the following range: [2, 12, 22, 32, 42, 52, 62, 72, 82, 92]. This range is chosen to prevent having to much impurity in a singular node, but because of the size of the dataset, some higher values were also needed. During experimentation a peak was found in the range so the range was kept like this. Because of the broad range, it can be assumed that the maximum was not too local.
- Random Forest:
  - n_estimators: This parameter is how many trees are build in the algorithm. There is chosen for the values: [10, 50, 100, 200] because more trees will usually result in better performance, however more trees also result in more computation power and there was only access to consumer laptop hardware. So because of time constraint, the forest was kept at only 200 trees.
  - max_features: This is how many features each node in a tree maximally looks at for the decision. There is chosen for the sqrt of n, log2 of n and n where n is the amount of features because these were the options the documentation.
- Isolation Forest:

– n_estimators: This parameter is how many trees are build in the algorithm. There is chosen for the values [10, 50, 100, 200] because more trees is usually better, but it was important to not have to many trees, as this results in a too high computation cost.
– max_samples: This is a list of max sample values used to train the model. The max sample parameter specifies the number of samples used for building each isolation tree. The following range is chosen: [100, 250, 500, 750, 1000, 2000, 4000]. This range is chosen because a peak was found in this range and the range was quite broad.

4 Results and Discussion

Note: As a lot of these models use some randomness within their algorithm, these graphs are susceptible to some change.

After training the models and measuring the qualities, the best performing models resulted in the following quality measures:

Model	Validation Accuracy	Test Accuracy	Validation Recall	Test Recall	Validation Precision	Test Precision	Validation F1	Test F1
Decision Tree	0.92	0.92	0.52	0.52	0.64	0.63	0.58	0.57
Random Forest	0.92	0.92	0.52	0.53	0.66	0.61	0.56	0.57
Isolation Forest	X	0.85	X	0.38	X	0.31	X	0.35

Figure 4: Quality measures of the algorithms

As can be seen, the accuracy is usually very high, but as stated before, this is not very relevant. More interesting is the F1-score, where can be seen that decision tree and random forest both perform very similar, but not great, with a score of 0.57. Isolation forest performs even worse at an F1-score of 0.35, although this is not a very fair comparison as Isolation Forest in an unsupervised method compared to the other 2. We can also see that precision is slightly better than recall in both Decision Tree and Random Forest, in Isolation Forest however, Recall is slightly better than precision, this can be the case because it's harder to be precise without labels.

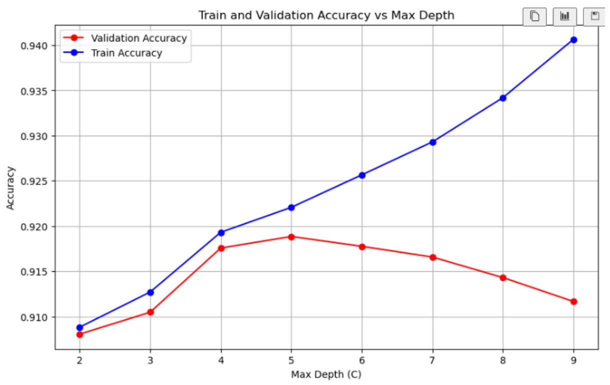

Figure 5: Train and Validation Accuracy vs Max Depth for Decision Tree

In this graph a classic example of overfitting in a decision tree can be seen. The max depth increases, which increases the complexity of the decision tree, this will result in better training accuracy, but a peak can be seen at 5 for the validation accuracy, if the complexity of the tree gets increased further, the model starts to be overfitted to the training data. And the validation accuracy goes down.

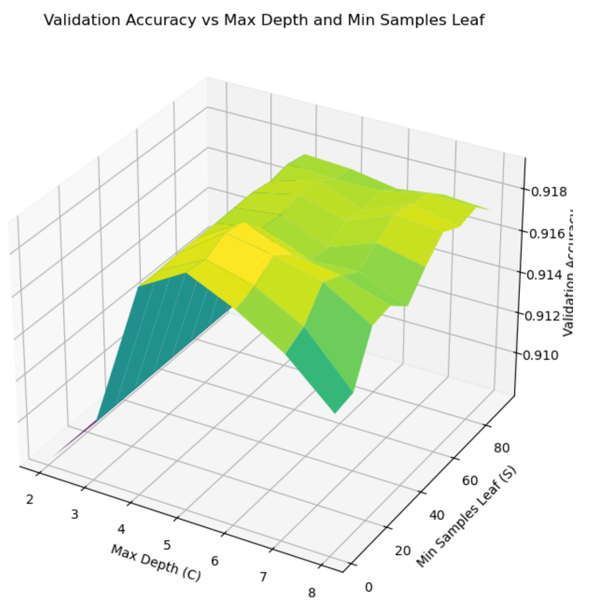

Figure 6: Validation Accuracy vs Max Depth and Min Samples Leaf for Decision Tree

This graph shows us the accuracy score of decision trees. Almost all scores are above 90 percent due to the highly imbalanced nature of the dataset. There are relevant peaks and the giant decreasing slope when decreasing max-depth too much. But this does not paint a great picture of the quality of the model in the domain. As the model was not above 90% accurate in outlier detection.

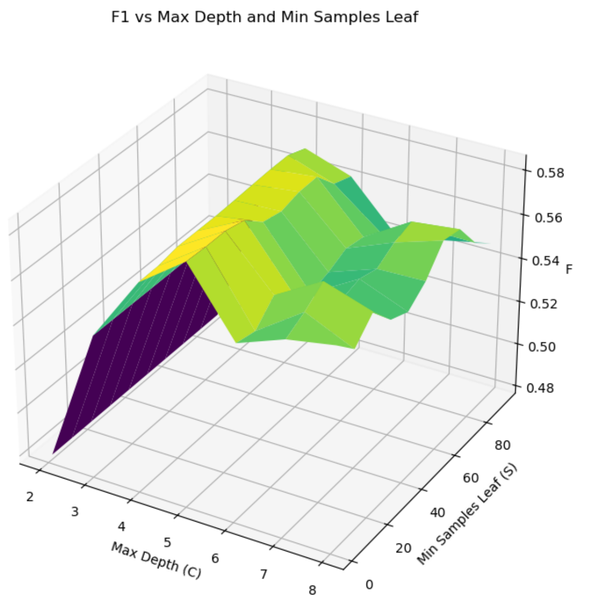

Figure 7: F1 vs Max Depth and Min Samples Leaf for Decision Tree

In this graph, the F1 score can be seen with various combinations of the Max Depth and Min Samples Leaf parameters with the Decision Tree algorithm. As can be seen by comparing the graph with the graph in figure 6, Again we see that when the Max Depth of the tree is too low, the tree is too simple to capture the patterns we want. We want at least a depth of around 5 to start capturing some patterns. The minimum samples in a leaf doesn't have a great influence on our score, so that hyper paramaters isn't so important, as long as you do not make it too small with an incredibly deep tree. Deeper than our scope that is. Cause that will lead to overfitting.

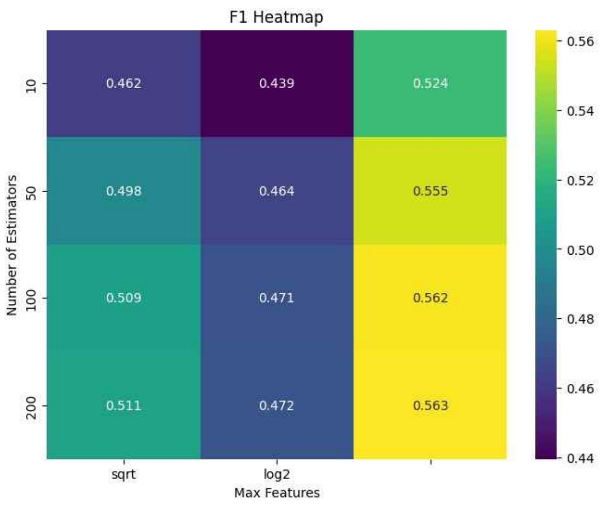

Figure 8: F1 heatmap for Random Forest

For random forest, in the heatmap can be seen that the number of estimators has a positive influence on the F1 score, this is to be expected as more trees results in better generalization in this algorithm.
Interestingly, feature bagging has a negative impact in the domain. The best performing max-features is the one where there is no feature bagging at all. The reason behind this could be because the dataset is very complex, causing feature bagging to miss intricate patterns.

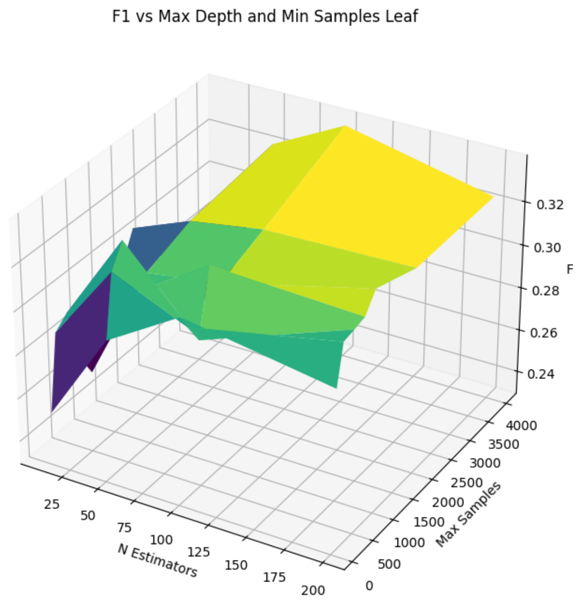

Figure 9: F1 vs Max Depth and Min Samplas Leaf for Isolation Forest

The graph for isolation forest looks very disorganized. The reason behind this can be that isolation forest is very random and unsupervised. These does seem to be some correlation between more samples per tree and better performance, but overall it scores very badly anyways.
In practical situations however, you usually don't have labels for which data points are outliers and which are not.

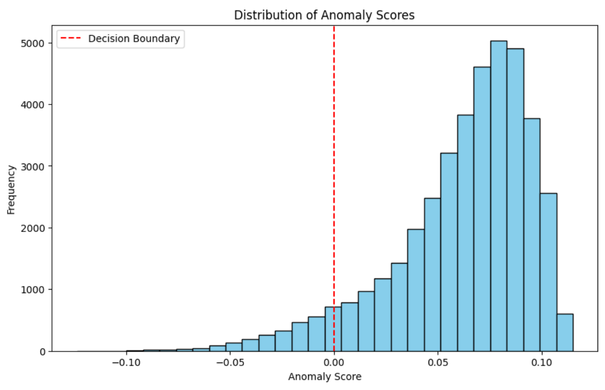

Figure 10: Distribution of Anomaly Scores

In the graph, the anomaly scores of isolation forest can be seen. The lower the anomaly score, to more abnormal a datapoint is. The decision boundary is set at 0, meaning the all negative values are classified as outliers and all positive values are classified as inliers. The distribution shows that isolation forest does not really classify any outliers with a very low anomaly score. From this, it can be concluded that isolation forest does not see any outliers that are very far away from the rest of the data in this dataset, which makes classification rather hard, especially unsupervised. The precision and recall for isolation forest are not great however, so that means this graph unforunately tells us less as well about the actual outliers.

5 Conclusions

Based on the experimental results, the Decision Tree and Random Forest models resulted in similar F1 scores, both achieving a score of 0.57. While consistent this is not a particularly strong score given the data set. The Isolation Forest, being an unsupervised method, underperformed significantly with an F1 score of 0.35, which is expected due to its unsupervised nature compared to the other models, but in practical situations you do not have labels for outliers, so the comparison is not completely fair. In conclusion all the models are too simple for outlier detection and the supervised models obviously perform better. Random forest does not gain an advantage over decision trees.
For future adaptations of outlier detection, a different model would be the way to go, however if you are set on using these 3 models, there are still many other hyper parameters to explore.

References

[1] Nikolaj Buhl. Train-test-validation split in 2024, 2023.

[2] Sakshi Raheja. Train-test-validation split in 2024, 2024.

[3] A. Aylin Tokuç. Splitting a dataset into train and test sets, 2024.