Term Project Report - New Particle Formation Event Classifier

Heli Huhtilainen

December 17, 2020

Abstract

This project report is for the Helsinki University course Introduction to Machine Learning. The report describes the development of an event classifier for a new particle formation data set. The main goal was to find a binary classifier that could separate event days from nonevent day, predict how probable these outcomes are and additionally get a multi class prediction for three types of event classes and nonevent. The project consists of data analysis, model and feature selection, and model hyper parameter tuning. The performance of the solution was tested in a challenge with a separate data set where the target values were hidden.

After initial data analysis and exploration decision tree and random forest were compared, and random forest was chosen and further toned. Predictions for the challenge were made with two separate random forest models. The overall challenge ranking for the solution was eleventh, and this was also the ranking of the binary classifier. Best performing part was the multi class classifier that produced fourth best performance. The worst performing part of the solution was of the perplexity of the binary classification probabilities. After the challenge some improvements for the perplexity and model accuracy were found from model calibration.

1 Introduction and initial approach

The this term project the goal is to make a machine learning model and that could predict whether on a certain day there is a new particle formation event, and if it is, to which class the event belongs to.

The project started with some exploratory data analysis. I did most of that as was instructed in the course exercises. As my in my one person team there was limited time to get the project done in time, I decided to pick two algorithms and research them further. I also aimed to find a model that could answer all the challenge questions. In the development I used Python with several libraries: pandas [tea20] and numpy [Har+20] for data wrangling and analysis, matplotlib [Hun07] and seaborn [Wt20] for plots, sklearn [Ped+11] and scipy [JOP+20] for machine learning models.

2 Exploratory data analysis

2.1 Looking at the data

The same train data set was given to all groups doing the project. The project data set has been gathered from Hyytiälä forestry research station in years 2000 – 2009, and it consists of 430 rows and 102 features. One of these features is the date and 100 features are measurements like temperature, radiation or relative humidity. Of these measurements 50 are daily means, and 50 are daily standard deviations of the measurements taken that day.

The data set is quite clean and ready for model development. In the target values there are equal amounts of 215 nonevents and 215 events, and the events are further divided into three classes. The Dal Mass report describes the classes and how they are divided [Dal+05]. Class II events (amount 106) are not as clear as Class I, and the most clear and strong events are classified as Class Ia (amount 26).

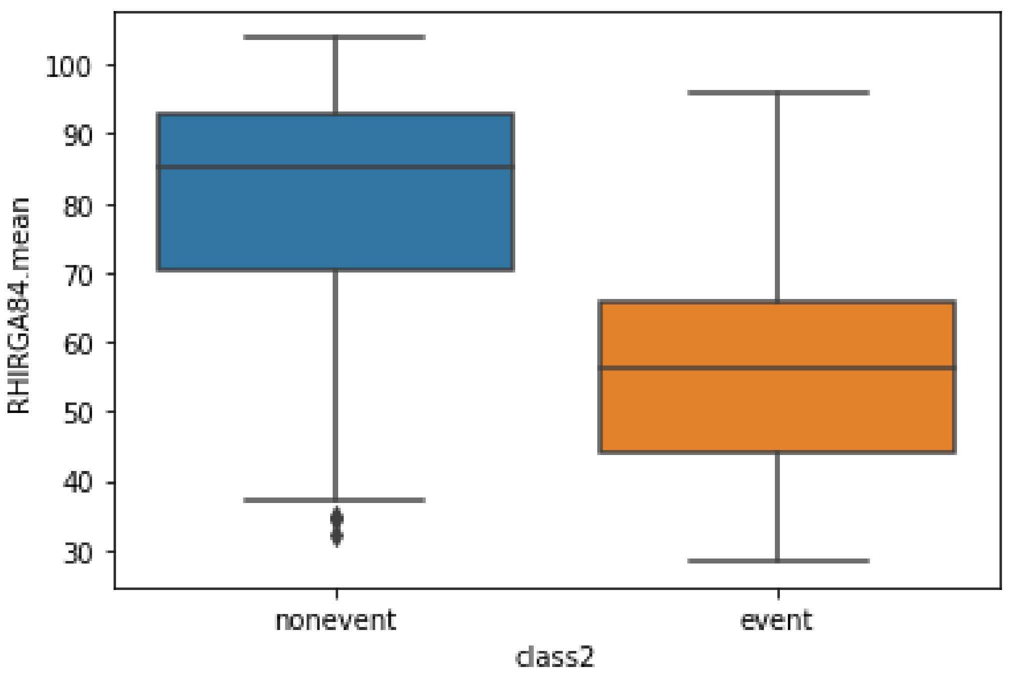


Figure 1: A box plot pairing relative humidity and binary classes shows quite a clear separation.

The remaining Class I events are classified as Class Ib (amount 83). Knowing this, it is probable that some of the events are harder to separate from each other than others, as the quality of the data differs, as do the amounts of the different classes. Some of this behaviour can be seen in the figures 1 and 2, where the classes are compared against relative humidity. In two classes scenario the classes are well separated, as in the four class scenario there is some overlapping in the three different event classes.

There are some strong positive and negative correlation of certain features, as figure 3 shows. This could be explained partly by the fact that some of the features measure same quality but on different heights, and partly that certain phenomena might occur or not occur at the same time.

2.2 Unsupervised learning methods

Next I examined the data using K-means clustering. All of the training data rows are used, and 50 mean features are chosen for these models. The data is normalized to better separate the effect of different features. The figure 4 shows the effect of loss reducing when the amount of clusters is increased. First the reduction is fast, and grows soon smaller. It seems that the two fist clusters are most pronounced. When K-means model is restricted to four models and the result is compared to the actual classes, the model does not work too well as a classifier. The confusion matrix in figure 5 shows the result of this comparison, and as the diagonal shows the correct results we can see that more than half of the rows are misclassified.

When trying hierarchical clustering with average and Ward linkage functions, the Ward linkage seemed to produce quite balanced clusters as can be seen in figure 7, but when examining them further they did not classify the actual classes that well. The classification result was quite similar to the k-means clustering as we can see in the figure 6.

Next I tried dimensionality reduction with principal component analysis (later PCA). The features are again normalized to better see how they affect in relation to each other. When PCA is reduced to two components, the nonevents seem to separate quite well from the other classes in the figure 8.

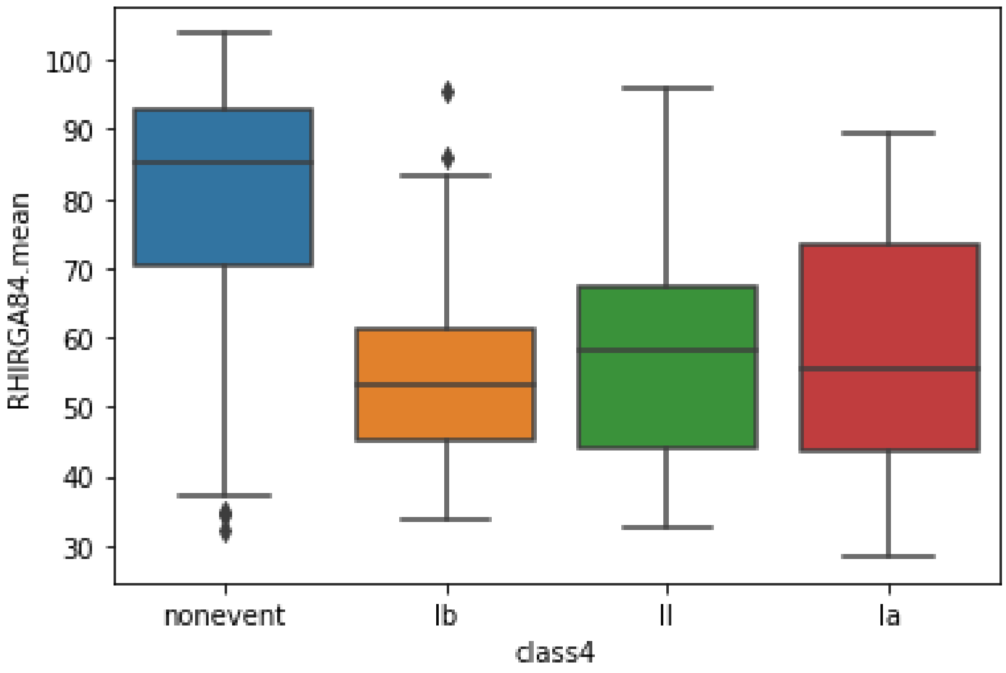


Figure 2: A box plot pairing relative humidity and four classes shows some overlapping of three event classes.

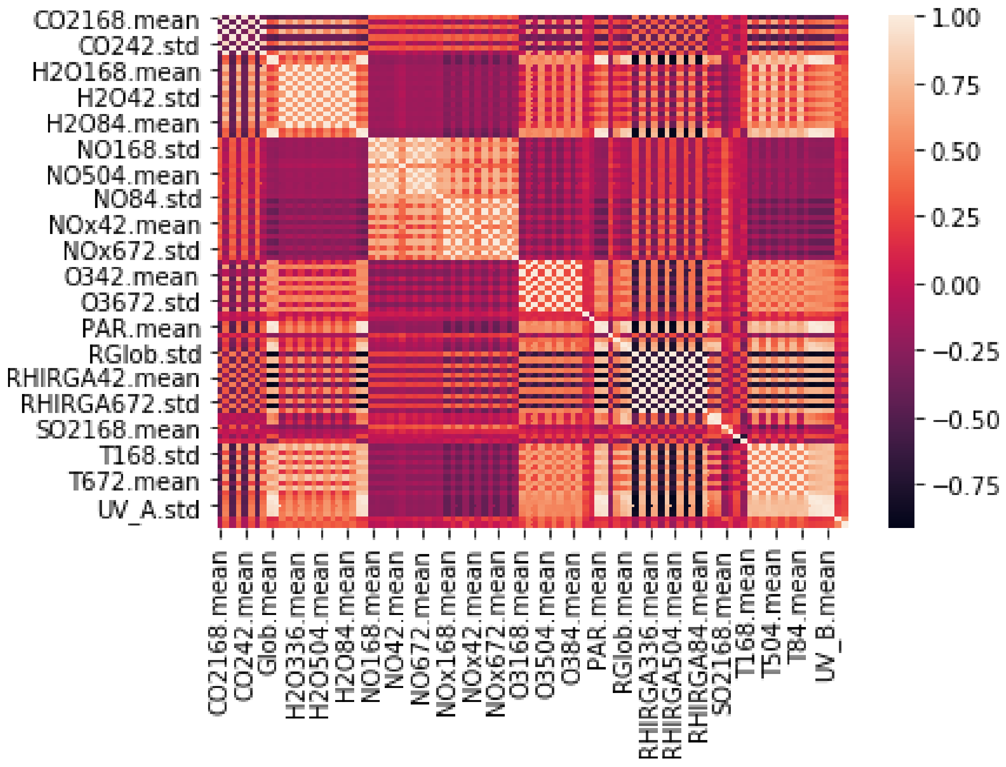


Figure 3: A heat map of all feature correlation shows that some of the features are strongly correlated.

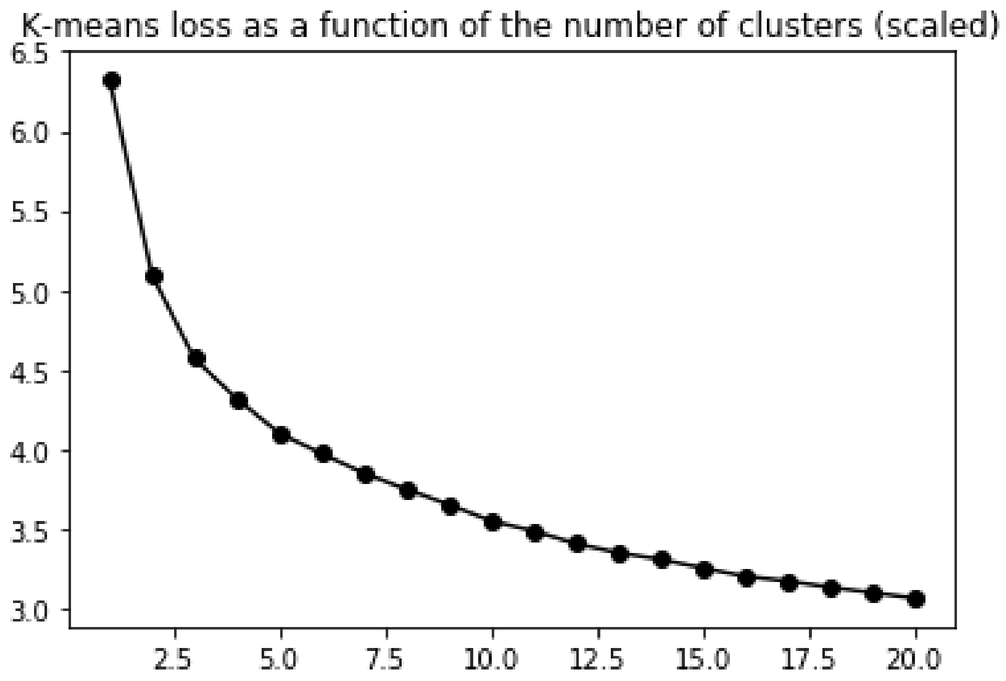


Figure 4: Loss of k-means model is reduced when amount of clusters is increased.

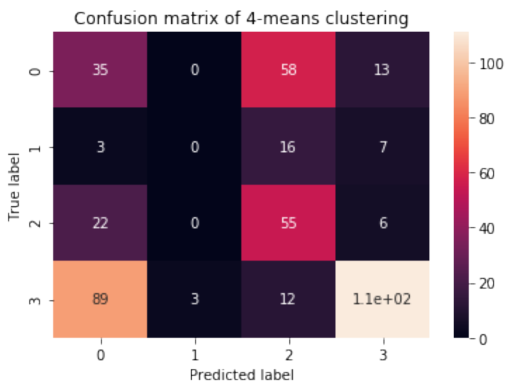


Figure 5: K-means with four clusters does not perform well as a classifier.

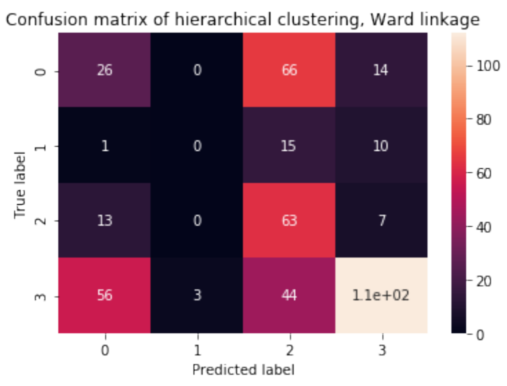


Figure 6: Hierarchical clustering results with Ward linkage function are quite similar to k-means.

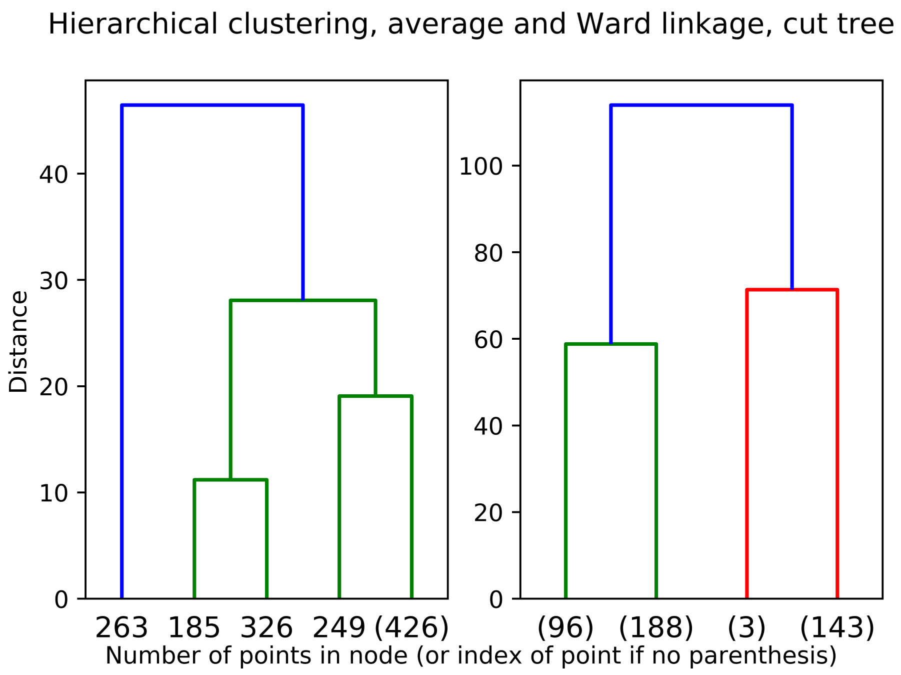


Figure 7: Hierarchical clustering produces very different clusters when the linkage is changed.

When examining the amount of variance explained, the first ten components seem to explain most of the variance as plotted in figure 9.

3 Machine learning approaches

After doing some exploratory analysis, and getting to know some different machine learning methods during the course, tree models seemed like they might be a good choice for this project. I thought that tree methods might work well with this type of data, that does not seem linear and has quite a lot of different features. There is some natural feature selection in the algorithms themselves, and that might be beneficial here. Also I had written my Bachelor's thesis about decision trees and random forests, and was eager to explore them with some actual problems.

3.1 Decision tree description

In decision tree the classification model is done by dividing the training set observations so that the split is as pure as possible. The purity is measured in this case by Gini index [Jam13, chapter 8.1], which is a measure of total variance across the classes of the observations. The good thing about decision trees is that their function is very easy to understand. The down side is that their accuracy is rarely the best possible, and the results can vary a lot if there are changes in the data.

3.2 Random forest description

Random forest is a method that uses an ensemble of decision trees that are produced by using different subsets of the training data [Jam13, chapter 8.2]. By making several trees the effect of variations in separate trees can be made more stable by averaging their result. This makes the model perform better than a single decision tree. The downside in random forest is that it is not as easy to interpret, as there are usually a lot of trees so it is not so easy to see how the result is produced.

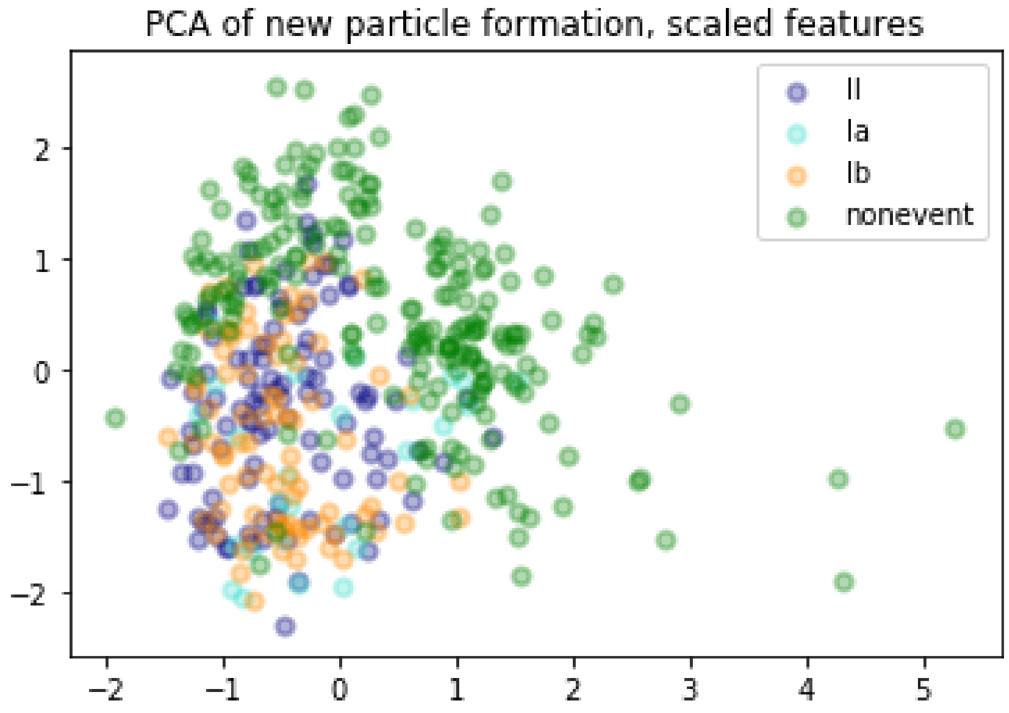


Figure 8: Nonevent observations can be seen separated when features are embedded to two dimensions.

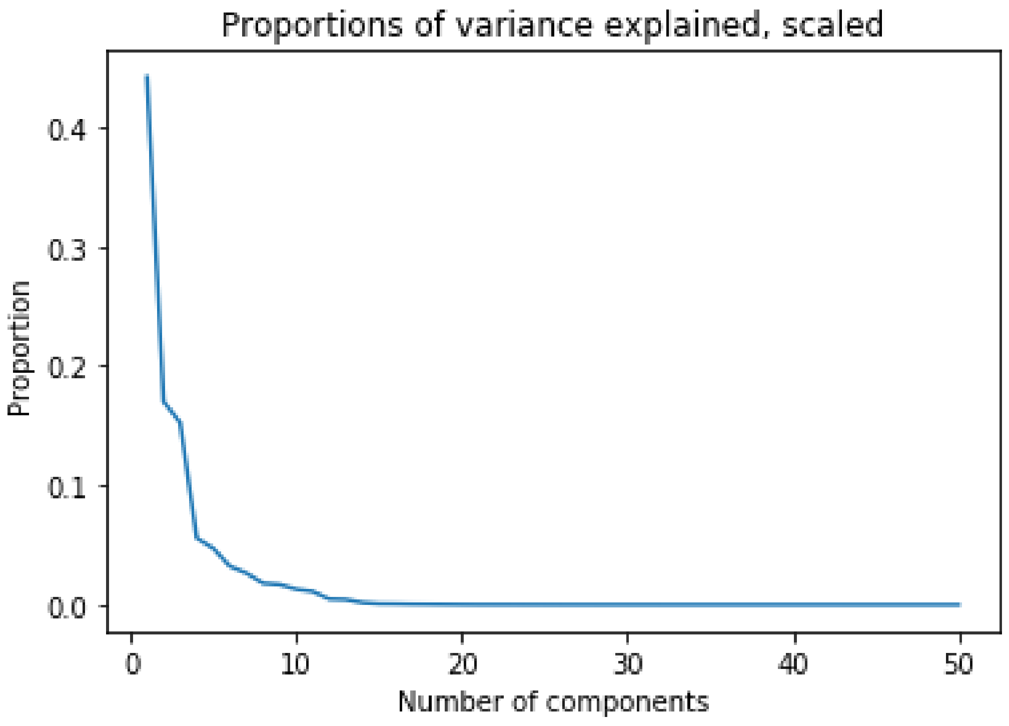


Figure 9: Most of the variance is explained with the first ten principal components.

Listing 1: Decision Tree Classifier

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None,
                       max_leaf_nodes=None, min_impurity_decrease=0.0,
                       min_impurity_split=None, min_samples_leaf=1,
                       min_samples_split=2, min_weight_fraction_leaf=0.0,
                       presort='deprecated', random_state=0,
                       splitter='best')
```

Listing 2: Random Forest Classifier

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None,
                       max_features='auto', max_leaf_nodes=None,
                       max_samples=None, min_impurity_decrease=0.0,
                       min_impurity_split=None, min_samples_leaf=1,
                       min_samples_split=2, min_weight_fraction_leaf=0.0,
                       n_estimators=100, n_jobs=None, oob_score=False,
                       random_state=0, verbose=0, warm_start=False)
```

3.3 Trying out the models

I took the multi classification accuracy as my model selection criteria, as it seemed the most interesting and challenging problem. The initial models were made again with 50 % division to train and validation data, and using 50 mean feature columns. I used Python and scikit-learn library implementations [sci20b] and [sci20c] to make the models. The default model parameters were used, expect for the random seed that was set to 0 to be able to reproduce the results. The listing 1 shows the initial model parameters for decision tree, and listing 2 for random forest.

The resulting decision tree is plotted in the figure 10. The details are quite small, but by zooming in the plot gives some insights. The decision tree root has all 215 training samples, and the resulting tree is a whole 10 depth tree that is built until the Gini error is 0. The first split is made by RHINGA336 <= 76.904 (relative humidity) [Wik20]. At root the classification is nonevent, which seems logical as it is the biggest class. The colors represent different classes, and the darker the color is, the more pure the classification is.

The first five trees of the random forest in figure can be found in the figure 11. The trees seem to vary quite a bit, but I guess that is one of the benefits of this model: to get a lot of different results, and then calculate averages. In the root nodes there are 135-143 samples, which is resulting from the default setting that uses bootstrapping and only a subset of the samples. In all the trees the initial split is done with different attributes: different RHINGAs and one by NOx84 (nitrogen monoxide + nitrogen dioxide concentration in ppb-units) [Wik20]. The depth of the trees seem to vary between 10 - 15. This maybe could be worth investigating, so that the trees are not overfitted by too complex structure and depth. Overall these trees still look quite good in my opinion.

I also did binary prediction models of both decision tree and random forest. The only difference in these was that the three event classes were combined into one class.

The initial results for these single models were surprisingly good. Multi class accuracy for decision tree accuracy 0.65 and random forest 0.71. For binary classification the accuracies were 0.80 for decision

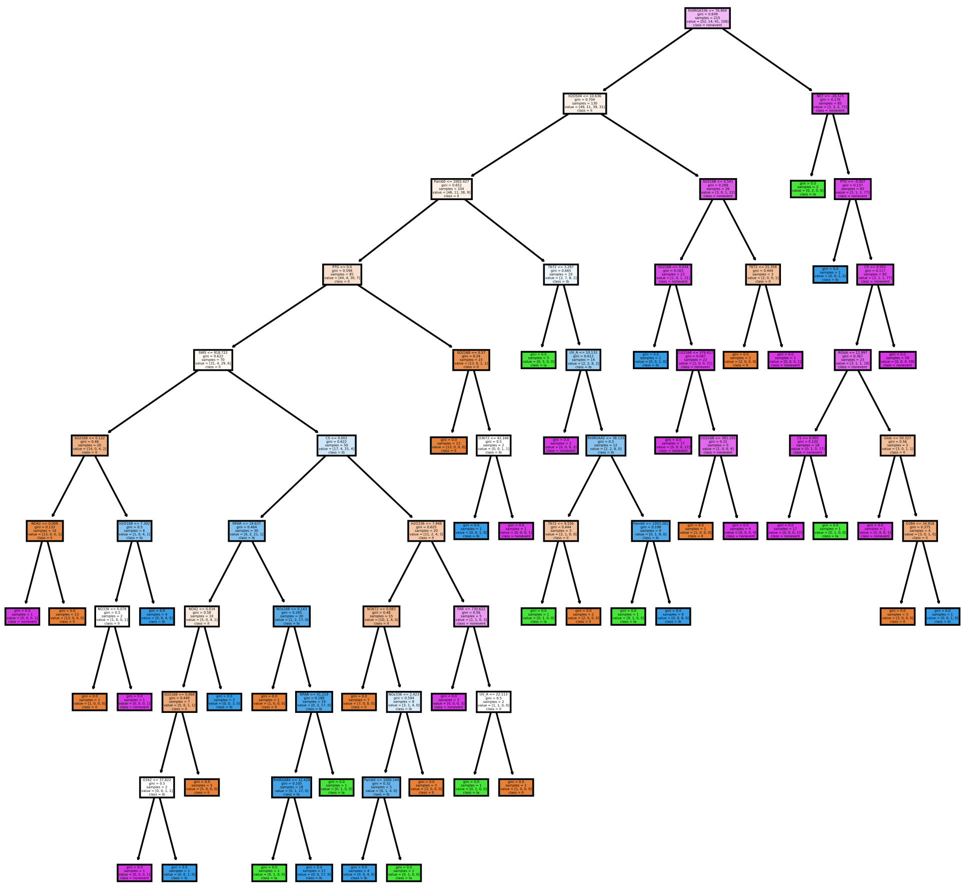


Figure 10: Decision tree classifier with four npf classes: green Ia, blue Ib, orange II and pink nonevent.

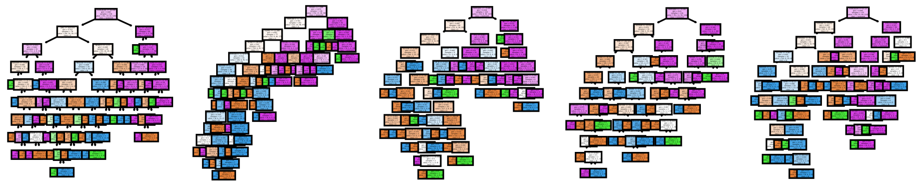


Figure 11: First five trees of random forest show some variance in the model. Classes: green Ia, blue Ib, orange II and pink nonevent.

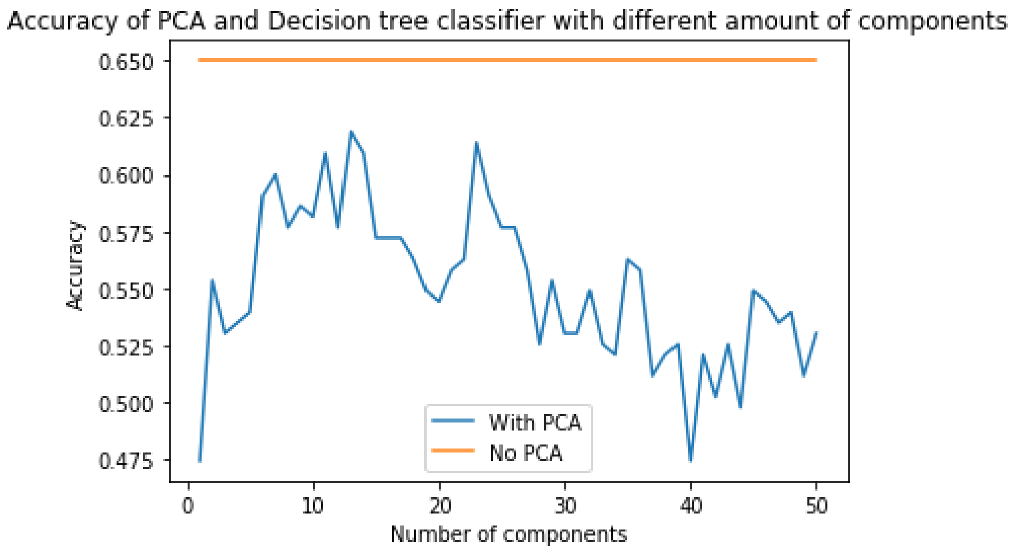


Figure 12: PCA did not improve decision tree performance in multi class prediction.

tree, and 0.87 for random forest.

4 Feature selection

4.1 Combining PCA to the models

Next I tried if feature selection would improve the results of multi class accuracy. First I tested combining PCA to the models. I did this in the four class classification scenario, again using 50 mean features as the base, with 50 % train validation split. The PCA was done with different amount of principal components. The figure 12 shows that decision tree suffered quite a bit by this combination, and it also did not improve random forest either, as is shown in the figure 13. These results were done without cross validation, so at least the no PCA values were a bit overly optimistic as they are taken from the initial model try-outs that were done previously. The trend seemed quite clear anyway, and I decided to try something else next.

4.2 Comparing different feature subsets

To test different feature subsets I used 10-fold cross validation. I used 50 % (215 rows) to make the train and validation set, and remaining 50 % to test the resulting models. The results were averages of the accuracies of ten models made with the cross validation data sets.

I tried three different approaches to feature selection. As a base solution I decided to concentrate using only the mean features. Additionally I tested adding a month feature, but this solution could not be utilized in the challenge prediction, as in the test set the date column had been hidden. As a third solution I tried aggregating the correlating mean values resulting in 19 features. Finally I tried doing a feature importance analysis with random forest, and using the top 20 features. The figure 14 shows that the results stayed approximately the same, expect for the top 20 features, that overfitted the random forest model and performed very poorly on the test set.

I also tested binary random forest with cross validation using the 50 mean values and then adding the month. The results were 0.88 with the cross validation set, and 0.87 with test set. With the month variable the results were 0.89 with the cross validation set and 0.86 with the test set.

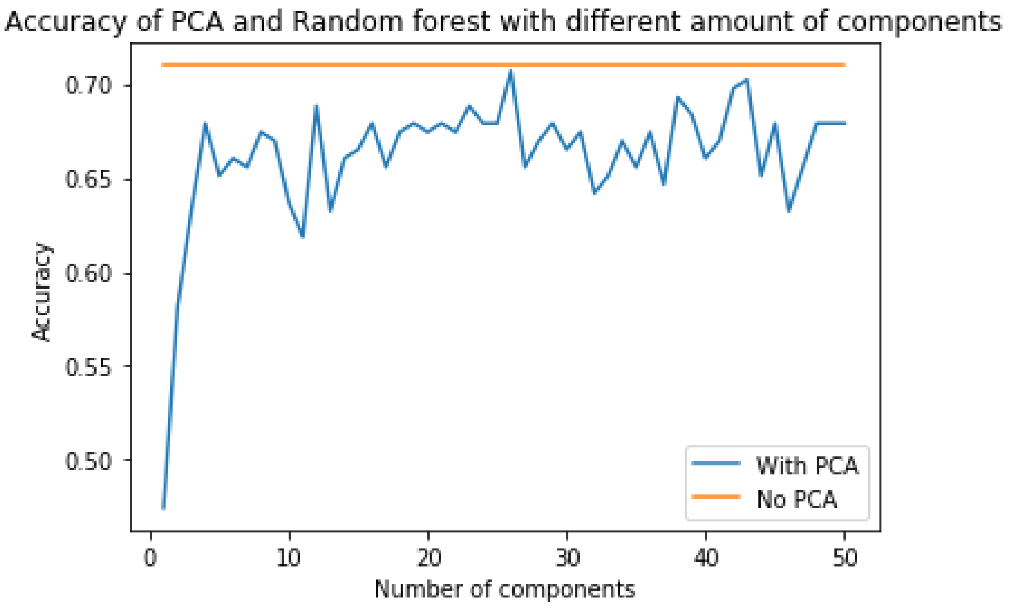
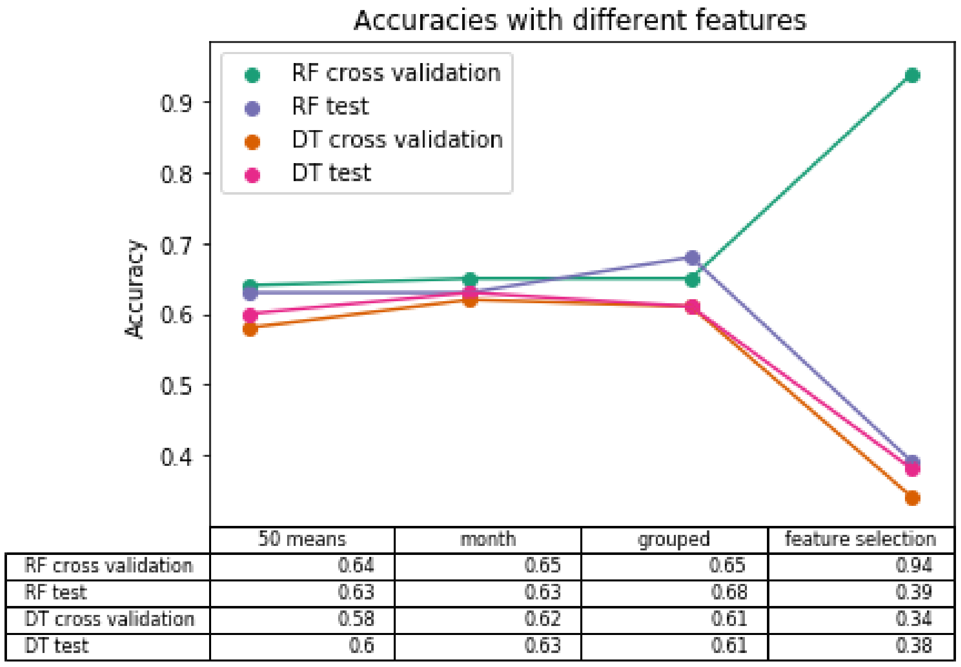


Figure 13: PCA did not improve random forest performance in multi class prediction.



	50 means	month	grouped	feature selection
RF cross validation	0.64	0.65	0.65	0.94
RF test	0.63	0.63	0.68	0.39
DT cross validation	0.58	0.62	0.61	0.34
DT test	0.6	0.63	0.61	0.38

Figure 14: Different features selection produced quite similar results, except for reducing features to top 20 that performs poorly.

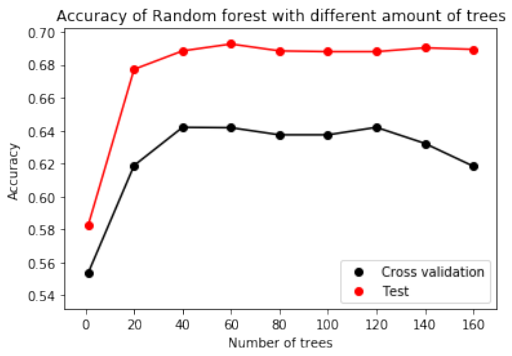


Figure 15: Tree amount and accuracy with random seed 0.

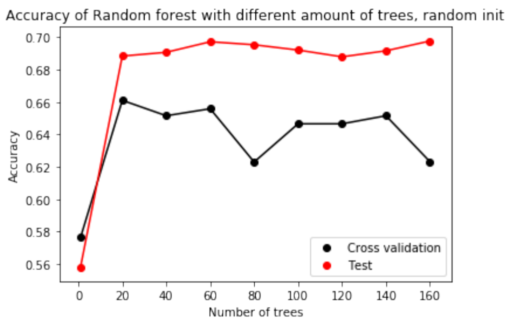


Figure 16: Tree amount and accuracy with random initialization.

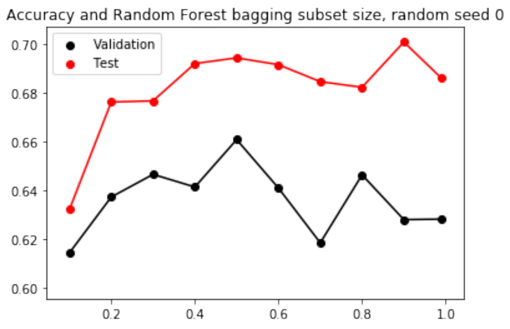


Figure 17: Bagging subset size and accuracy with random seed 0.

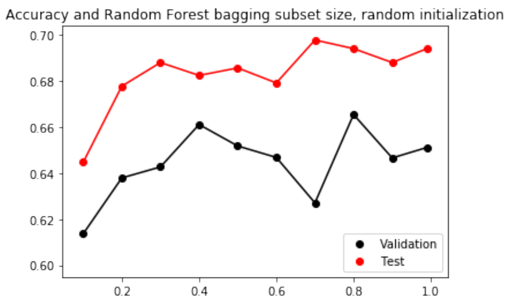


Figure 18: Bagging subset size and accuracy with random initialization.

5 Parameter selection

5.1 Adjusting random forest tree amount

At this point I decided to concentrate on testing the random forest, as it was performing better than decision tree.

I tested adjusting the model parameter that defines how many trees a random forest has. I did this with the similar cross validation setting as the feature selection tests. It seemed that 60–120 trees produce quite same accuracy as is shown in the figures 15 and 16, and 60 trees seemed to work best. Quite surprisingly the test accuracies were constantly better than the cross validation accuracies, and there seemed to be more variation in cross validation results. This might be due to the small sizes of the cross validation test sets.

Next I tried how adjusting the bootstrap sample size might affect the results using the same cross validation method as previously. The sample sizes varied from 18 in figure 19 to 124. The figure 20 shows a tree with sample size 75. Smaller subsets produced worse accuracy as is shown in the figures 17 and 18, so there didn't seem to be reason to restrict the size of bootstrapped samples further from the default.

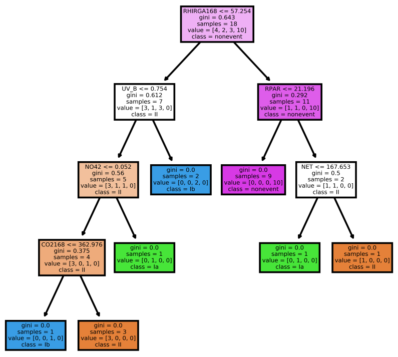


Figure 19: Random forest tree with 10 % bagging subset.

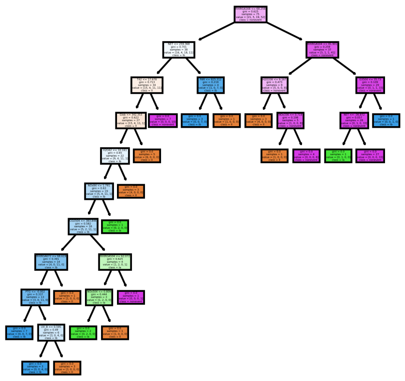


Figure 20: Random forest tree with 50 % bagging subset.

6 Results

6.1 Models for the challenge

For the challenge I decided to make two random forest models, one for the multi class prediction and the other for the binary classification. From the features I used 50 mean values, as the grouping did not significantly improve the results. In the hyper parameters I decided to stick to the default model parameters where the tree amount is 100, as making changes to these did not seem to improve the model accuracy. I predicted my binary accuracy to be all data cross validation test score 0.87 - 0.01, as the test result might give a bit optimistic result. This score I added manually to the file after adding the other results. To get the binary classification probabilities, I used function that predicts class probabilities for the rows. The probabilities are computed as the mean predicted class probabilities of the trees where the class probability of a single tree is the fraction of samples of the same class in a leaf [sci20c].

The overall challenge ranking for the solution was eleventh, and this was also the ranking of the binary classifier. The challenge results are in the table 1. Best performing part was the multi class classifier that produced fourth best performance. The worst performing part of the solution was of the perplexity of the binary classification probabilities.

6.2 Analysis of the results and improvements

After the challenge I wondered how the model could be improved. The most obvious improvement would be to fix the perplexity issue. The problem with my original solution was that the simple way of calculating the probabilities resulted in some of the probabilities to be 0 or 1, and this resulted in an overflowing perplexity.

I found a solution for the perplexity problem from scikit-learn probability calibrator [sci20a] that uses sigmoid function to reduce the confidence of overly confident predictions. The listing 3 shows the model parameters for this model. I also made modifications to my challenge models by paying closer attention to the results I had had from testing different solutions. I took 60 trees for the random forest model, and grouped the correlating variables resulting in 19 features for the train set. I also improved my accuracy prediction to 0.87 as 0.875 was the average of the test that I had done for the binary

State	Binary accuracy	Accuracy of accuracy	Perplexity	Multi accuracy
Challenge	0.8673575	0.0073575	Inf	0.6818653
Improved	0.870466	0.000466	1.539664	0.683938

Table 1: Results of challenge models and improved models.

Listing 3: Calibrated Binary Random Forest Classifier

```
CalibratedClassifierCV(
        base_estimator=RandomForestClassifier(
        bootstrap=True, ccp_alpha=0.0,
        class_weight=None, criterion='gini',
        max_depth=None, max_features='auto',
        max_leaf_nodes=None, max_samples=None,
        min_impurity_decrease=0.0,
        min_impurity_split=None,
        min_samples_leaf=1, min_samples_split=2,
        min_weight_fraction_leaf=0.0,
        n_estimators=60, n_jobs=None,
        oob_score=False, random_state=0,
        verbose=0, warm_start=False),
        cv='prefit', method='sigmoid')
```

accuracy. I did two new models with these changes, and tested it against the challenge test set. All of the results were improved by these changes, as is shown in the second row of table 1.

6.3 Insights learned

In the beginning of the project I could have taken a more broad approach in finding the most suitable model for this project, and that might have resulted in me choosing some other model to develop further. I am however satisfied with the results that I finally managed to get by using this method. I chose to examine the multi class accuracy most carefully, and that was also the part of the solution that got the best ranking. This approach was also a good way of getting to know more about the details and functions that were available in the library implementations.

The most important insight I have gained from this project was to take the time really to look at my results and do the changes that even the smaller differences in the results suggest. I also think that doing cross validation several times with more models and comparing the results might have provided more insight to the model. In my current solution the results were varying quite a bit because of the randomness of the model. Also splitting the training data in different ratios and doing stratifying might have reduced the uncertainty of the model accuracy. One important reminder was that it is good to look at the test data right in the beginning. Now I realised a bit late that the dates were missing from the test data, after I already had spent some time in researching the month feature that I could not utilize in the end.

After the project I think it would have been beneficial to have a team to discuss the problem together and divide the workload. Working alone was however an easier choice mostly because of the ongoing pandemic, and allowed me to approach the problem in my own schedule. As a whole I am quite happy with the end result, I learned a more structured way of approaching a machine learning problem, got some routine doing plots of the results and got a lot of ideas how to do things better in the future.

References

[Dal+05] Miikka Dal Maso, Markku Kulmala, Ilona Riipinen, Robert Wagner, Tareq Hussein, Pasi P. Aalto, and Kari E. J. Lehtinen. "Formation and growth of fresh atmospheric aerosols: eight years of aerosol size distribution data from SMEAR II, Hyytiälä, Finland." In: *Boreal Environment Research* 10 (2005), pages 323–336. ISSN: 1239-6095. `http://www.borenv.net/BER/archive/pdfs/ber10/ber10-323.pdf`.

[Har+20] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pages 357–362. `https://doi.org/10.1038/s41586-020-2649-2`. `https://doi.org/10.1038/s41586-020-2649-2`.

[Hun07] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pages 90–95. `https://doi.org/10.1109/MCSE.2007.55`.

[Jam13] Gareth James. *An introduction to statistical learning : with applications in R*. Springer texts in statistics. New York: Springer, 2013.

[JOP+20] Eric Jones, Travis Oliphant, Pearu Peterson, et al. *SciPy: Open source scientific tools for Python*. 2001–2020. `http://www.scipy.org/`.

[Ped+11] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pages 2825–2830.

[sci20a] scikit-learn. *Calibrated Classifier CV*. Dec. 17, 2020. `https://scikit-learn.org/stable/modules/generated/sklearn.calibration.CalibratedClassifierCV.html#sklearn.calibration.CalibratedClassifierCV`.

[sci20b] scikit-learn. *Decision Tree Classifier*. Dec. 16, 2020. `https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html`.

[sci20c] scikit-learn. *Random Forest Classifier*. Dec. 16, 2020. `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`.

[tea20] The pandas development team. *pandas-dev/pandas: Pandas*. Version 1.0.1. Feb. 2020. `https://pandas.pydata.org/pandas-docs/version/1.0.1/`.

[Wik20] Smear Wikispace. *Hyytiälä Database*. Dec. 16, 2020. `https://wiki.helsinki.fi/pages/viewpage.action?pageId=243959901`.

[Wt20] Michael Waskom and the seaborn development team. *mwaskom/seaborn*. Version latest. Sept. 2020. `https://doi.org/10.5281/zenodo.592845`. `https://doi.org/10.5281/zenodo.592845`.