Term Project

Introduction and Aims

The aim of the project was to predict, given atmospheric data, whether a New Particle Formation (NPF) event would occur on a particular day. Explained briefly, this is an atmospheric event that can lead to cloud formation.

In the project, the data used were the daily means and standard deviations of measurements collected at the Hyytiälä forestry research station. On each day either: no event took place (which was classified as "nonevent" in the raw data set); or there was an event, which was subdivided into three event types: "Ia", "Ib" or "II".

The goal of the project was to predict, given unseen atmospheric data as described above, firstly, whether an event took place (a binary classifier), and secondly, the type of event that occurred (multiclass classifier).

Outline of our approach

explore a mixture of models and choose the best one

blended method (combining two or more models with high accuracy, and predicting based on the average of each model's probabilities)

We tried a range of models and decided to use a combined approach to predict on the final dataset. The idea was to combine a generative and discriminative method, as well as an algorithmic approach, to find the optimal performer.

One of the biggest challenges to overcome in this project was the size of the data set. The full known data set consisted of under 300 rows, which meant that the models needed to be finely tuned to obtain the best accuracies, and there was some danger of overfitting the models.

Preliminaries:

Data Analysis

- **correllation**
- **PCA**

Data Preparation

Data Cleaning

To initially prepare the data for modelling, the following steps were made:

- removed the ID and 'partlybad' columns
- set the new index as the 'date'
- added the 'class2' column, which was set as non-event where 'class4' =nonEvent, or event, where 'class4' = Ia, Ib or II. This column would be used for the binary classifier.

Train/Validation/Test Data Set Splitting

Initially, the data was split with a stratified sample on the "class2" column only, that is, between event/non-event. This we found to give high differences in the error on the validation and test sets, which meant that we then resampled the train, test and validation to take a stratified sample between the "class4" variable.

The data was split 60:20:20 into the training, validation and test sets. This gave 154 entries in the training set, 52 in the validation and 52 in the test set. As well as these, we also included a train/validation data set (80% of all data) for cross validation.

Breakdown of each “class2” category in the training, validation and test sets for each model:

	Train	Validation	Test
nonEvent	0.493506	0.500000	0.50000
Ia	0.071429	0.076923	0.076923
Ib	0.214286	0.211538	0.211538
II	0.2207	0.211538	0.211538

Normalisation

To improve accuracy of our results, we normalised the data between 0 and 1. The data was normalised before splitting into the separate training, validation and test sets.

Some models did not use the normalised data, for example decision trees. Whenever the standard normalisation is not used, this is detailed in the model information.

Feature Selection

Approaches used

- select best K
- PCA/NCA

Models - Binary Classifiers:

Decision Tree:

Description

To grow a classification decision tree, we use recursive binary splitting. The tree will assign the new observations based on the region it belongs to, based on the majority class in that region in the training set.

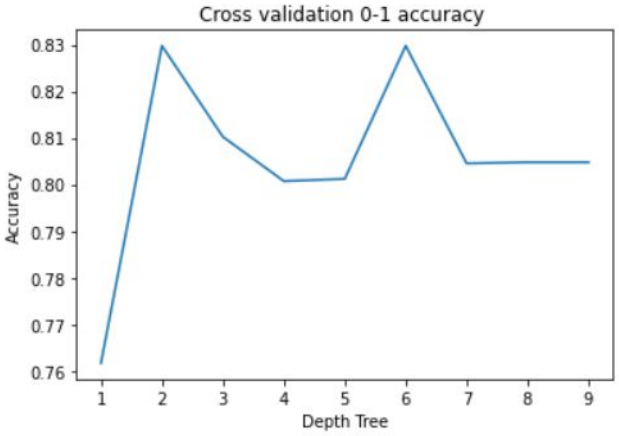
Advantages

Using a decision tree for the classification task was good, as these handle well qualitive predictors, such as the event class. One drawback of using this method is that decision trees are very sensitive to variation in the training data. This was of particular concern, since the training set size was very small

Tuning the model

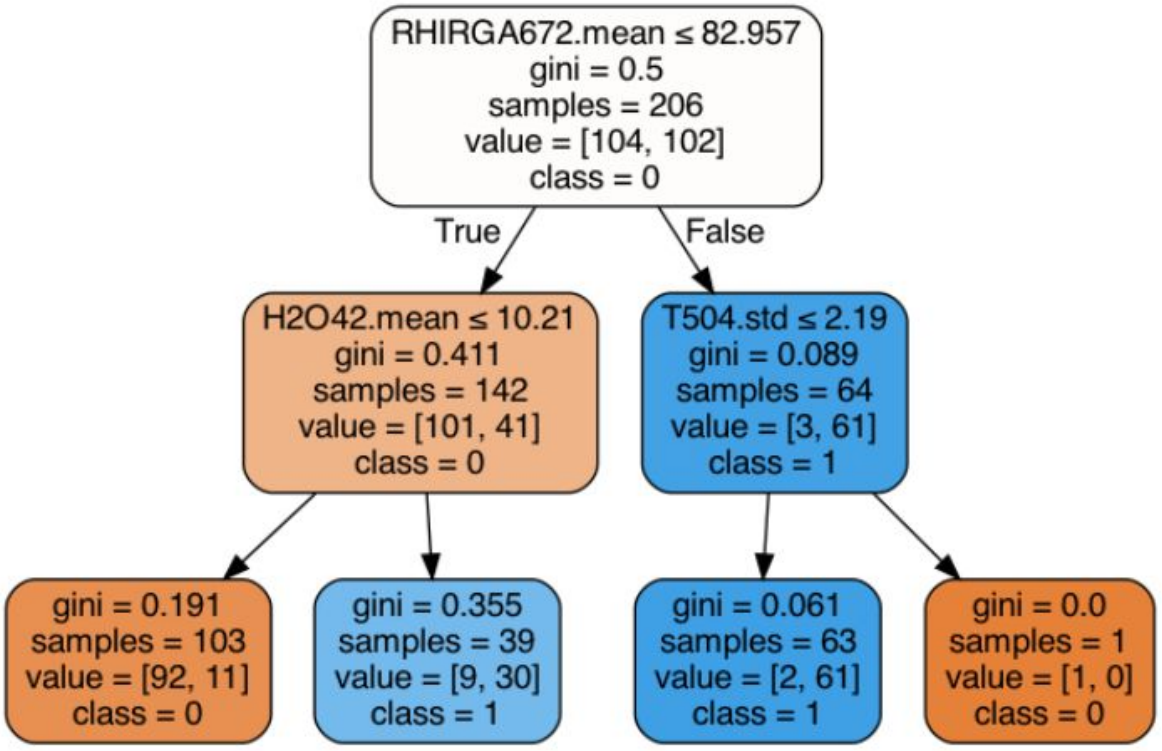
For this model the data was not normalised.

The parameter for tuning was the depth of the tree. For each value for the depth, the model accuracy for decision trees was measured by the classification error rate, calculated using cross-validation over the training and validation sets combined, and the depth was chosen that minimised the 0-1 loss.



The above graph shows the change in the CV 0-1 accuracy as the depth of the tree varied. The depth which maximised the accuracy was 2.

Final model output:
The figure below shows the final decision tree. This gave an 81% accuracy rate on the test set.



HEATMAP for decision tree?

Random Forests:
Description
Advantages

Parameter tuning: The 2 parameters are the number of trees in the RF and the number of variables chosen randomly at each node

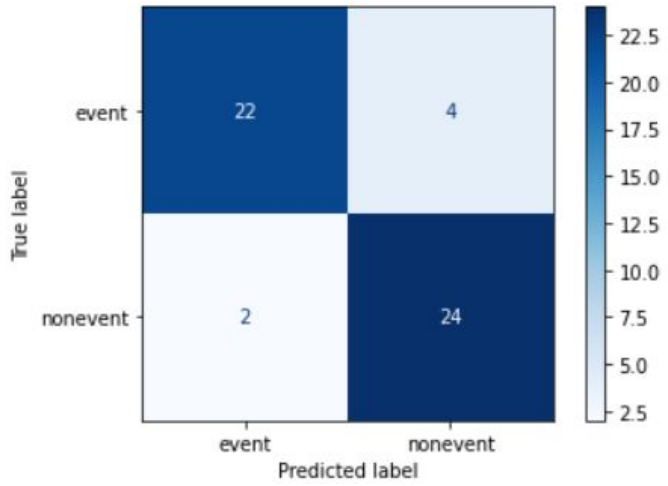
Measure accuracy over test set and AUC

Extreme Gradient Boosting:

Description

Tuning and Results The 5 parameters found through random grid search and then error estimated through cross validation

- colsample_bytree: the subsample ratio of columns when constructing each tree
- gamma: Minimum loss reduction required to make a further partition on a leaf node of the tree
- learning_rate: Step size shrinkage used in update to prevents overfitting
- max_depth: Maximum depth of a tree
- n_estimators: Size of sample of trees generated
- subsample: Subsample ratio of the training



KNN

Description: The model estimates the distance between each new observation and the training set and assigns the class based on the classes of the nearest neighbours in the test set.

Advantages

Tuning the model

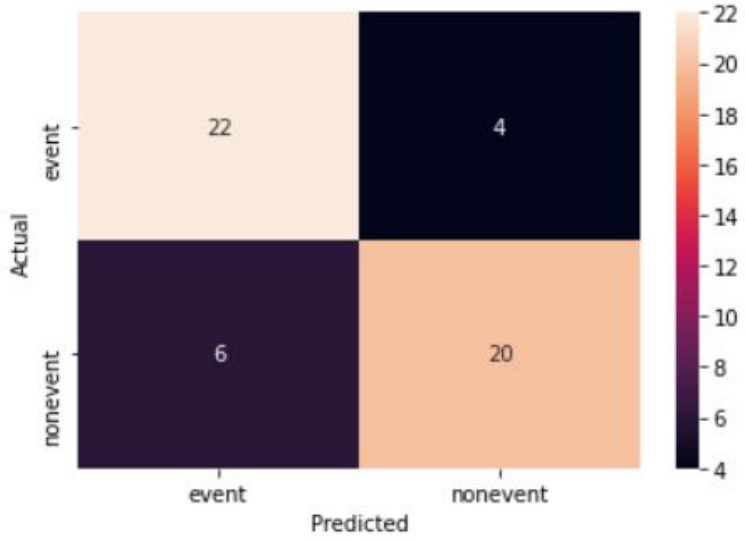
The parameter to tune is the number of nearest neighbours
NCA - Neighbourhood component analysis
feature selection using chi distribution

results

KNN returns a prediction, rather than a probability, so then to adapt that to our blended approach we needed to estimate the probabilities, using a majority vote method. This can be less accurate method than direct computation of probabilities.
The training data has a very high number of dimensions, which can sometimes not work very well with the k-NN algorithm.

tuning: For optimal performance of the knn, we tried to reduce the number of dimensions considered when calculating the distances between the training set. For this, the “Select k best” method (described in the data preparation section) and Neighbourhood Component Analysis was used.

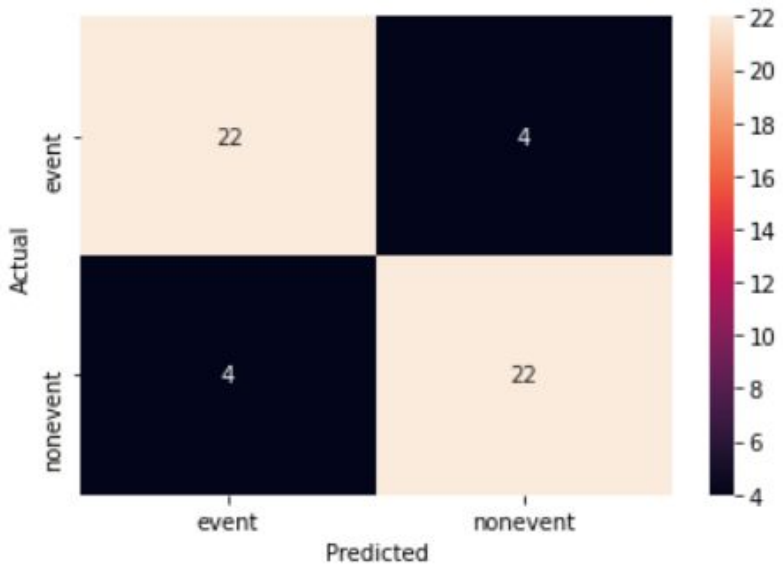
Selecting the k best features results was tried on the validation set, finding the error for the validation sets.



Logistic Regression

Description: Logistic Regression is a generative method that aims to find a logistic link function to the observed data and returns a probability that each belongs to a particular class.
Advantages: Logistic regression returns a probability that each observation belongs to a particular class. One limitation of using this method however, is that if the variables in the data are well-separated (ie that some are good predictors while others are not), which can frequently happen on a small size data set, and can cause convergence issues in the parameter estimates.
Tuning and Results:
To ensure that all coefficients shared the same scale, it was important that data used in the logistic regression model was normalised. Other normalisations were tried (for example with standard deviation 0.5), however the best and most consistent result was obtained on the data set normalised with mean 0 and standard deviation 1.

The final accuracy on the test set of the logistic regression model was 85%. The heatmap of predictions on the test set is shown below.



Naive Bayes

Support Vector Machine

brief description: this method is algorithmic and does not have a probabilistic interpretation.

parameter tuning

results

Final Accuracies for Binary Classifiers

	Decision Tree	Random Forest	XGB	k- Nearest Neighbour	Logistic Regression	Naive Bayes	SVM
Training	89%	100%	100%	90%	86%	80%	98%
Validation	83%	86%	89%	87%	85%	77%	88%
Test	81%	83%	88%	85%	85%	81%	87%

Binary Classifier Blended model

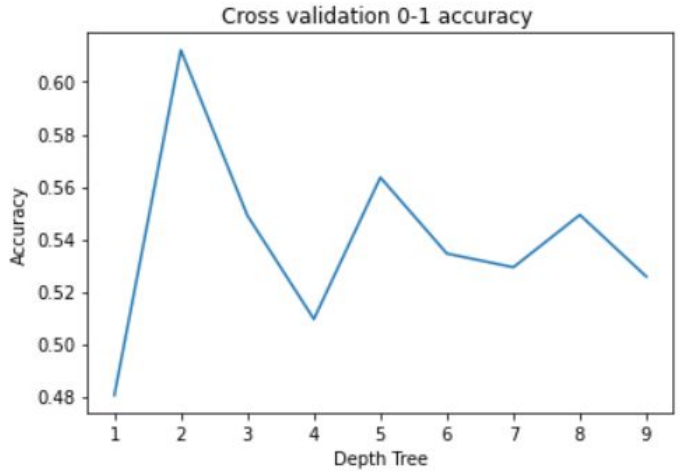
Final predictions and results

- which methods we chose and why
- combine top methods
- report final metrics of
    - accuracy
    - perplexity

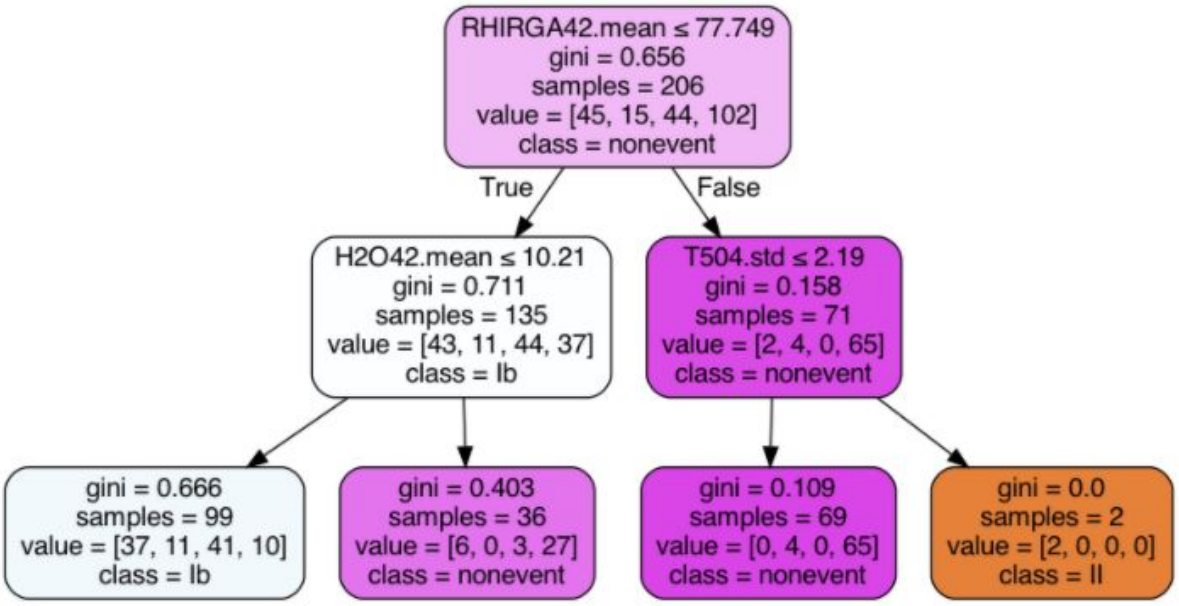
Models - Multiclass Classifiers

Decision Tree:

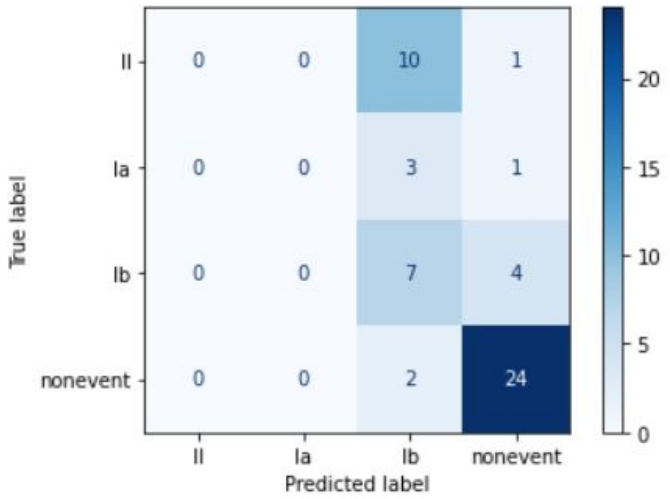
A similar procedure to the binary classifier was used for the multiclass predictor. Once again, the parameters were tuned using CV to minimise the 01 loss. The range of depths and their respective accuracies are shown in the graph below.



The decision tree produced is shown below. The model gave a



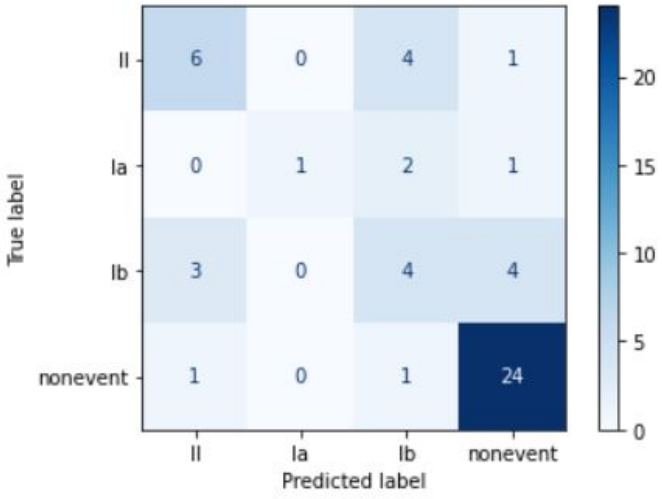
The final heatmap of predictions is shown below. The method gave an accuracy rate of 60% on the test set.



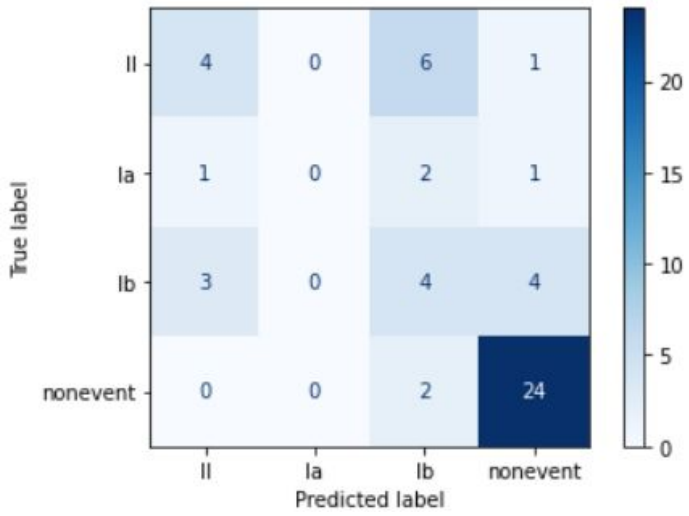
The confusion matrix shows that a 2 level decision tree classifies all observations into **nonevent** or into **Ib**. By looking at the tree plot, we can see that at most a 2 level tree can classify into 3 classes. The class of II is unlikely to be classified, because very few observations in training got into that leaf node.

Random Forests

Parameter tuning: The 2 parameters are the number of trees in the RF and the number of variables chosen randomly at each node



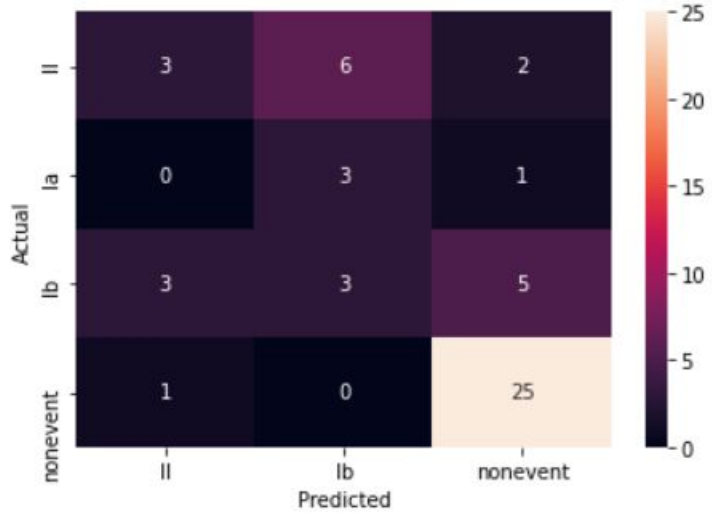
Extreme Gradient Boosting:



KNN

Parameter tuning: The parameter to tune is the number of nearest neighbours

NCA - Neighbourhood component analysis



Naive Bayes

SVM

Multiclass Classifier Conclusions

	Decision Tree	Random Forest	XGB	k- Nearest Neighbour	Naive Bayes	SVM
Training	66%	100%	100%	66%	60%	86%
Validation	61%	61%	67%	58%	52%	67%
Test	60%	67%	62%	58%	52%	67%

Final predictions and results

- which methods we chose and why
- combine top methods
- report final metrics of
    - accuracy
    - perplexity

Conclusion