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

Our approach began by exploring a variety of models, including a mixture of generative and discriminative methods as well as some algorithmic approaches.

The idea was then to combine best performing models (based on the accuracy), and calculate an average of the probabilities obtained from each method. The classifier would then output based on the value of this blended probability being greater than 0.5.

One of the biggest challenges to overcome in this project was the size of the data set. The full known data for training, validation and test consisted of 430 rows, which meant that the models needed to be finely tuned to obtain the best results.

Preliminaries: 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 252 entries in the training set, 84 in the validation and 84 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

In order to optimise our models, some feature selection was used. When this was applied to the model, it is outlined in the model description below.

The advantages of feature selection in this case is that we can use it to reduce overfitting. Due to the small data size the model is more sensitive to noise in the data. By using feature selection, we reduce the amount of misleading data points. The following methods of feature selection were considered.

- **select kBest features**
  - This method performs chi-squared tests on the data, which helps to determine the strength of each variable's relationship with the output. This was implemented by using the SelectKBest function from the sklearn.feature_selection library.
- **Principal Component Analysis**
  - This is a method of unsupervised machine learning, which helps to summarise the large number of variables in the data by selecting a smaller number of variables as representative samples.

Since the impact of these varied on a case-by-case basis, the models where feature selection is applied is detailed in the respective sections.

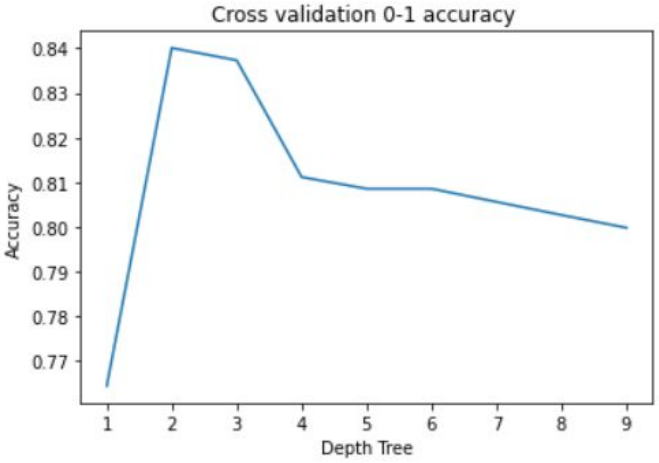
Models - Binary Classifiers:

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 that it belongs to, based on the majority class in that region in the training set.

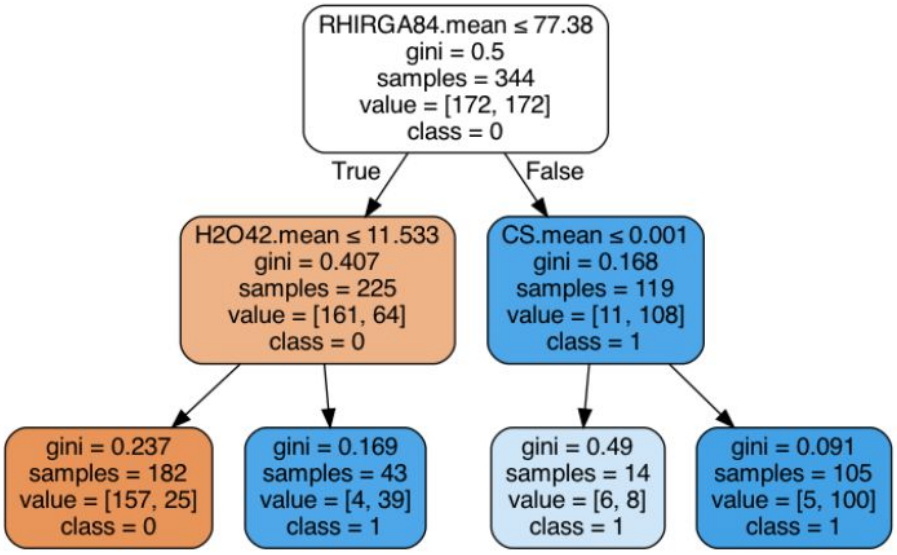
Advantages and Disadvantages: Using a decision tree for the classification task was good, as these handle well qualitative predictors, such as the event class. One drawback of using this method is that decision trees are very sensitive to variation in the training data. This was of particular concern,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

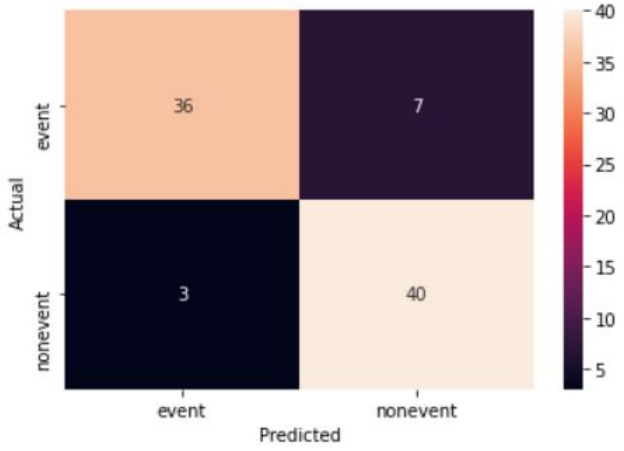
Tuning For this model the data was not normalised. The parameter for tuning was the depth of the tree. For each value for the depth, the model accuracy for decision trees was measured by the classification error rate, calculated using cross-validation over the training and validation sets combined, and 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 (which gave a cross-validation accuracy of 84%). This value was used to make the predictions on the test set.

Results: The figure below shows the final decision tree output, and below that, the heatmap of results. Overall, this gave an 88% accuracy rate on the test set.



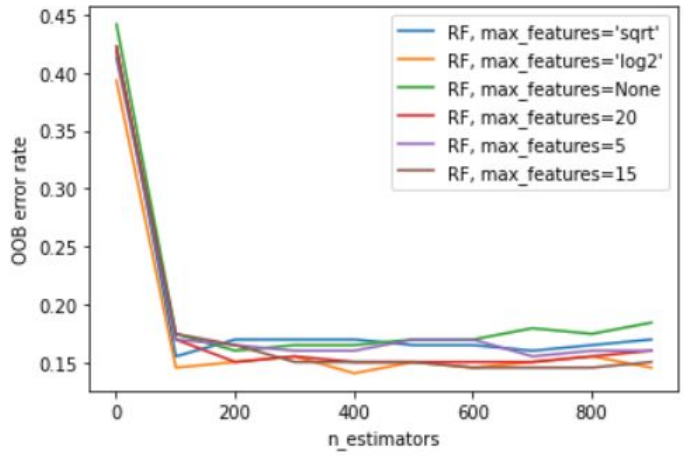


Random Forests:

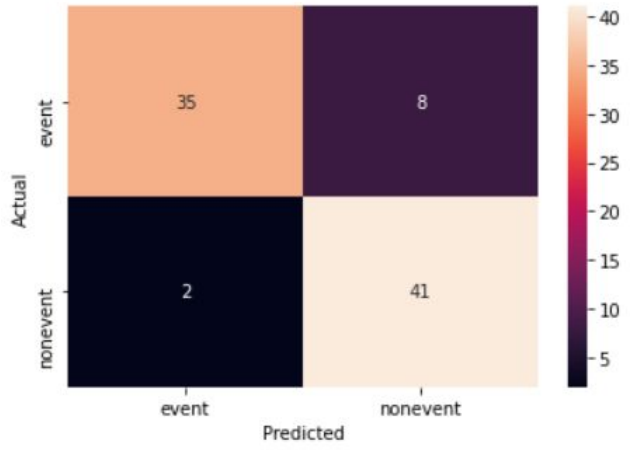
Description: The Random Forest classifier is an extension of decision trees that work by constructing multiple decision trees on the training set and outputting the modal class of each one. The method uses bootstrap aggregating (or bagging) to help avoid overfitting to the training set. The error was measured using the out-of-bag estimate.

Advantages and Disadvantages: Random forest classifiers will usually outperform a single decision tree, and help to overcome its issue of overfitting. However, they may still be susceptible to this issue, particularly due to the small size of the training data.

Tuning: The 2 parameters for tuning the model are the number of trees in the RF and the number of variables that are chosen randomly at each node. The error on the validation set for tuning was calculated using the out-of-bag error. The results of the analysis on the training set are displayed below. The final model chosen for use on the test set used the number of features chosen at each step was log2, and a total of 410 trees.



Results: The final tuned model gave an accuracy of 88% on the test set.



Extreme Gradient Boosting:

Description: Extreme Gradient Boosting is a supervised learning method that uses a decision tree as its base. It is an ensemble method that constructs several decision trees iteratively.

Advantages and Disadvantages: The XGB method is advantageous to use as it is efficient and helps to reduce overfitting of data. However, XGB may not perform well on a small training set (such as in this case).

Tuning: To tune the model, the 5 needed parameters were found through a random grid search. The parameters are:

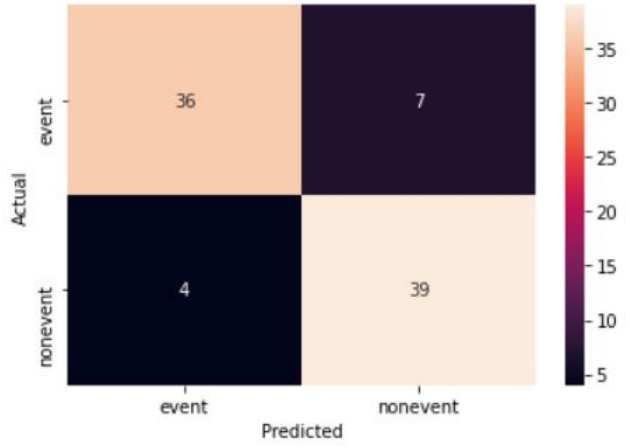
- colsample_bytree: the subsample ratio of columns when constructing each tree
- gamma: Minimum loss reduction required to make a further partition on a leaf node of the tree
- learning_rate: Step size shrinkage used in update to prevents overfitting
- max_depth: Maximum depth of a tree
- n_estimators: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

The results of the parameters found from the random grid search on the validation set are shown below:

```
{'colsample_bytree': 0.9915346248162882, 'gamma': 0.4812236474710556, 'learning_rate': 0.10553468874760924, 'max_depth': 3, 'n_estimators': 212, 'subsample': 0.6592347719813599}
```

The parameter values found gave a 5-fold CV accuracy of 90.4%.

Results: With the above parameter values, the final accuracy on the test set was 87%. Below is shown a heatmap of predicted vs actual classes.

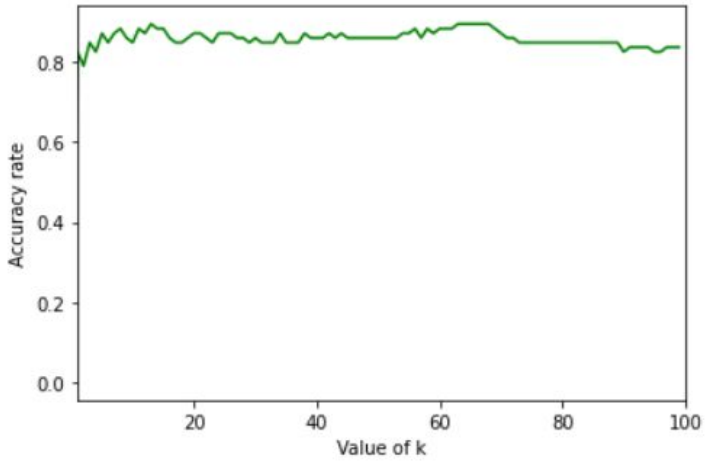


KNN

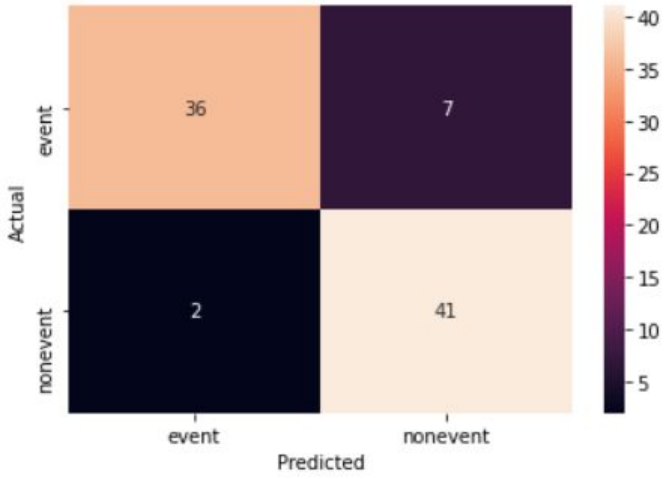
Description: This model computes a distance between each new observation and the training set data and classifies based on the classes of its nearest neighbours in the training set. The algorithm uses the normalised data to ensure consistency between feature distances.

Advantages and Disadvantages: One challenge of using k-NN on the data was the high dimensionality of the values. Each data point had 100 components. To overcome this, we used feature selection, such as the select k-best approach and neighbourhood component analysis. Another drawback of this method was the fact that there were relatively few observations in the training set that were classified as events, in particular where class2 = II, which meant that it was unlikely to obtain such classified outcomes.

Tuning: The main tuning parameter is the number of neighbours, k, considered by the classifier. If this is too low, the model will be too sensitive to any noise in the training data, while too high a value of k will in turn cause underfitting. The model was fitted to the normalised training data and the classification accuracy was calculated on the validation set, for each value of k. The graph below shows the validation accuracy rate as the value of k changes. The highest accuracy was obtained at k=13.



Results: The results for the binary classifier gave an 89% accuracy on the test set, and the validation accuracy with k=13 was 89%. The heatmap of predicted vs actual categories is shown below.

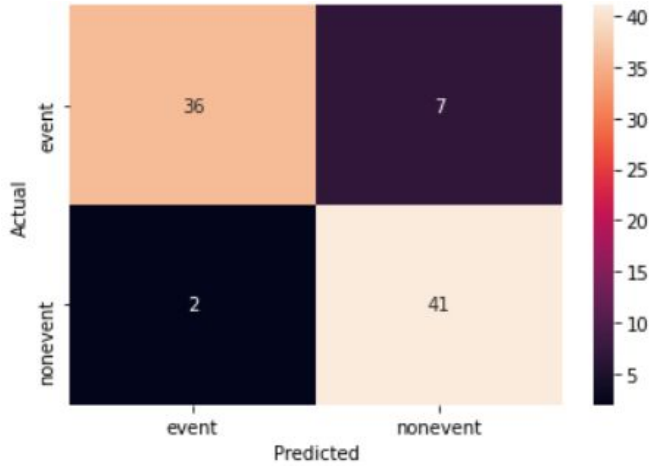


Logistic Regression

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 and Disadvantages: 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 could be an issue on the small data set size, and can cause convergence issues in the model.

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 best accuracy on the validation set was 87%.
The final accuracy on the test set of the logistic regression model was 89.5%. The heatmap of predictions on the test set is shown below.

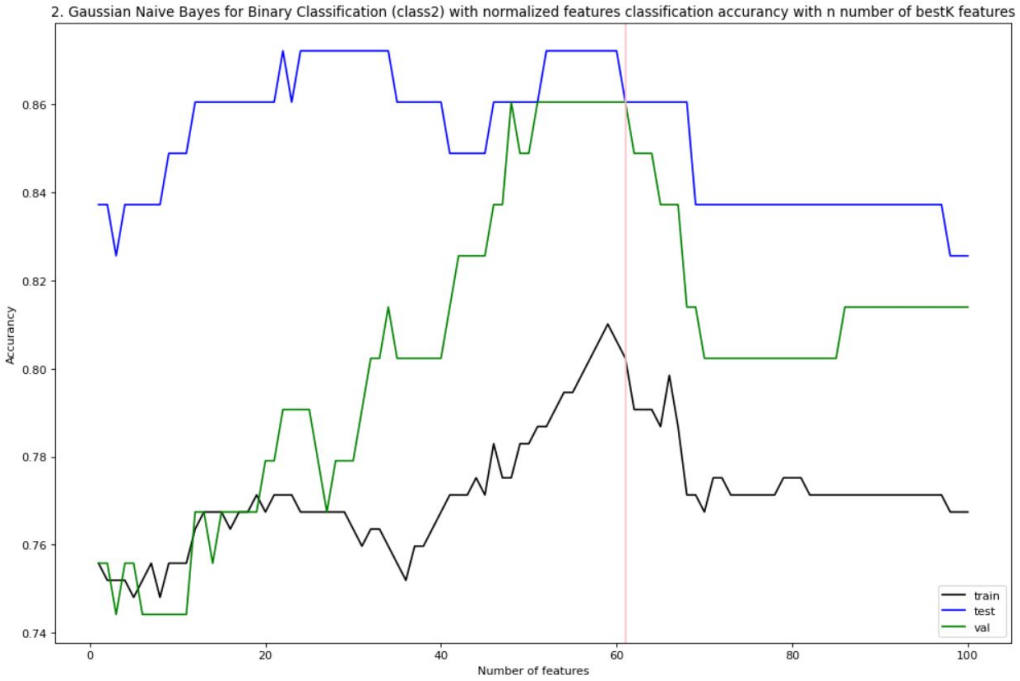


Naive Bayes

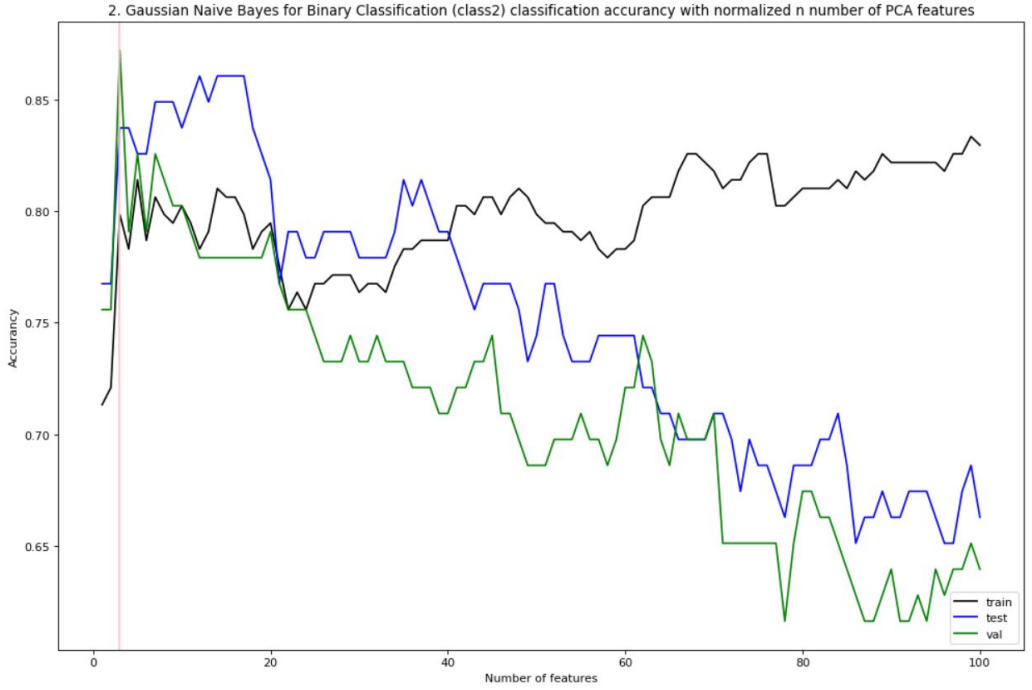
Description: The Naive Bayes classifier is a generative classifier that uses Bayes formula to calculate the posterior probability of an observation belonging to each class. The class is assigned if the probability is greater than 0.5.

Advantages and Disadvantages: Naive Bayes is a well performing model, however, the assumption of Naive Bayes (that all variables are independent) may not necessarily hold for the data. In particular, from our preliminary look at the data, we saw that some variables were correlated and therefore may not have been independent.

Tuning: Data was normalised for this model. To try to overcome limitations of Naive Bayes, select k best feature selection was used. The graph below shows the model accuracy on the training, validation and test sets as the number of features varied. The value used for the final model was 60 features, reducing the total number of dimensions by 40.

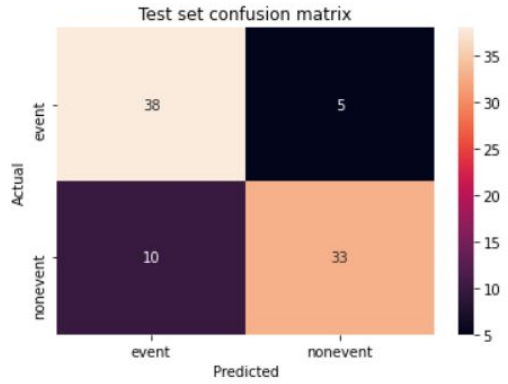


The same tuning was repeated with features selected using Principal Component Analysis (PCA). The graph of accuracy rates on the training, validation and test set is shown below.



The optimal number of components for the binary classifier using the normalised data was taken to be 3.

Results The highest accuracy obtained on the test set was 81%, using normalised data and features selected by PCA. The confusion matrix output in this case is shown below.



Support Vector Machine

Description: In brief, this method creates a boundary between the two different classes of data, and then assigns new points based on which side they belong to. This is an algorithmic model which does not have a probabilistic interpretation.

Advantages and Disadvantages: In general, the SVM classifier performs well. However, the method is not always applicable, as it requires some distinct boundary between the classes.

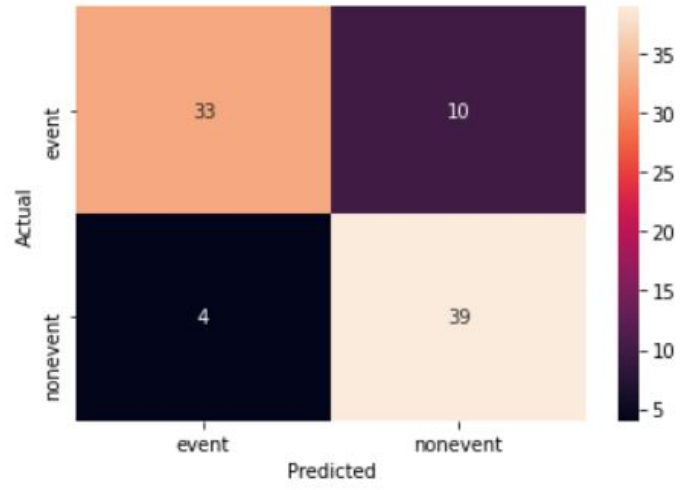
Tuning: The 4 parameters needed for tuning were found through a random grid search, with the error estimated through cross validation:

- C: Regularization parameter. (l2-regularization)
- gamma: Kernel coefficient for ‘rbf’, ‘poly’ and ‘sigmoid’.
- kernel: Specifies the kernel type to be used in the algorithm

The optimal values for these parameters found are shown below. These gave a 5-fold CV accuracy of 89.8%.

```
{'C': 11.091998785656282, 'gamma': 0.0008642674399024057, 'kernel': 'linear'}
```

Results: The final accuracy obtained on the test set for the binary classifier was 83.7%. The confusion matrix produced is shown below.



Final Accuracies for Binary Classifiers

	Decision Tree	Random Forest	XGB	k- Nearest Neighbour	Logistic Regression	Naive Bayes (Best K)	Naive Bayes (PCA)	SVM
Training	88%	100%	100%	85%	87%	81%	84%	98%
Validation	84%	87%	90%	78%	87%	85%	87%	90%
Test	88%	88%	87%	80%	89%	87%	93%	83%

Binary Classifier Blended model

The final blend of models chosen was XGB, Naive Bayes and Logistic Regression. This was chosen as a combination of an algorithmic, generative and discriminative method. The final choice of models was based on not only those which had a high accuracy, but also consistent accuracy over the validation and test sets. All the models chosen also output probabilities, which meant that they were easily comparable.

In combining the methods, firstly classification probability for classifying as 'event' was calculated for each selected model, using the optimally tuned parameters described above. Next, we computed the mean of these probabilities. If the mean probability calculated was greater than 0.5, the data point would be classified as 'event'.

The final accuracies for the binary classifier obtained were as below:

Accuracy on train set: 0.9612403100775194

Accuracy on validation set: 0.9651162790697675

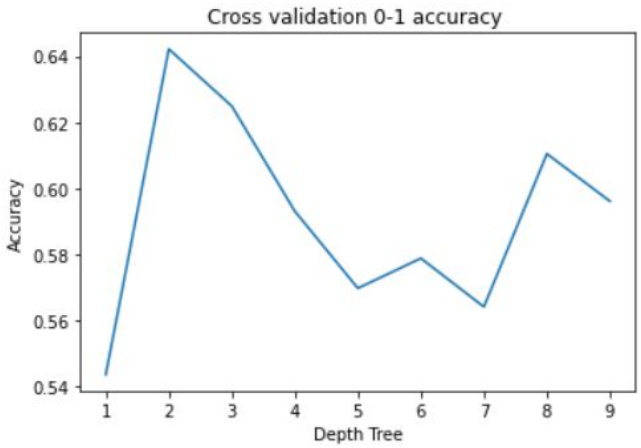
Accuracy on test set: 0.9186046511627907

Models - Multiclass Classifiers

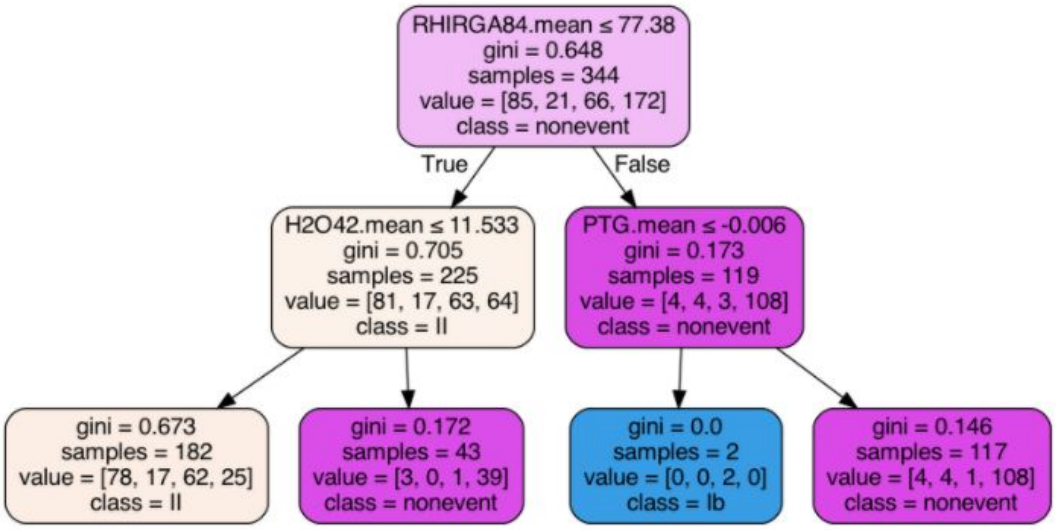
Where appropriate, the above models were adapted to multiclass classifiers and re-tuned.

Decision Tree:

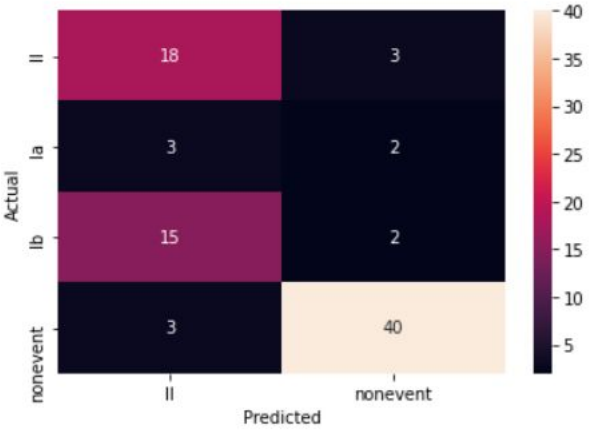
A similar procedure to the binary classifier was used for the multiclass predictor. Once again, the parameters were tuned using CV to minimise the 01 loss. The range of depths and their respective accuracies are shown in the graph below. As in the binary classifier, a depth of 2 was chosen to maximise the accuracy on the test set.



The decision tree produced is shown below.

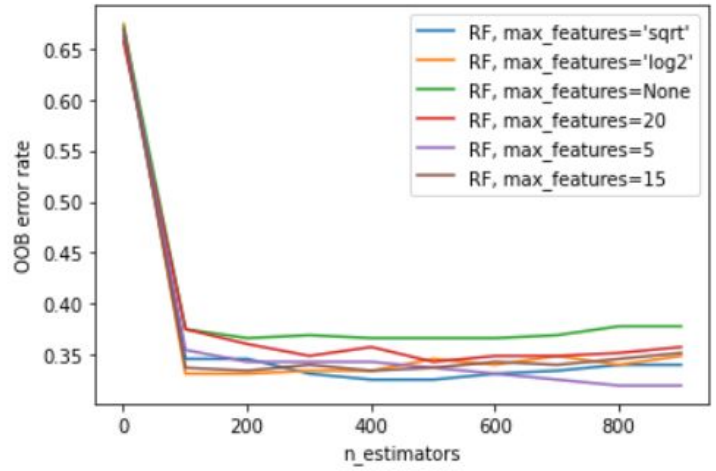


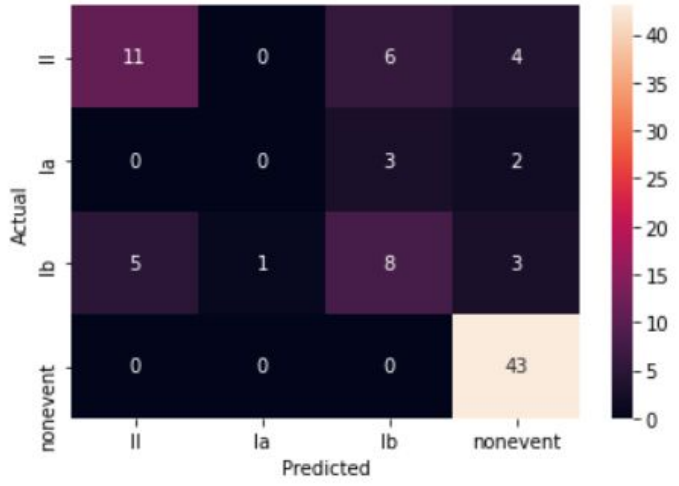
Overall, the model gave an accuracy rate of 60% on the test set. The final heatmap of predictions is shown below, which shows that a 2 level decision tree classifies all observations into **nonevent** or into **II**. By looking at the tree plot, we can see that at most a 2 level tree can classify into 3 classes. The class of Ib is unlikely to be classified, because very few observations in training got into that leaf node, and this did not happen on our results in the test set.



Random Forests

Tuning: As with the model used for the binary classifier, the model was tuned using an OOB estimate on the validation set. The OOB error rates are shown below. The final parameters chosen for the multiclass model were 15 max features and 410 estimators. This gave a final value of 66% OOB accuracy on the validation set, and 72% accuracy on the test set.

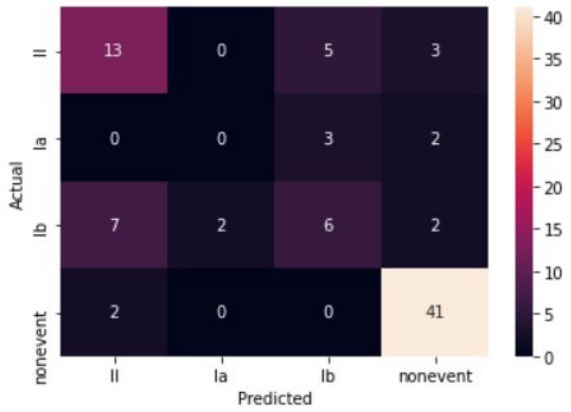




Extreme Gradient Boosting:

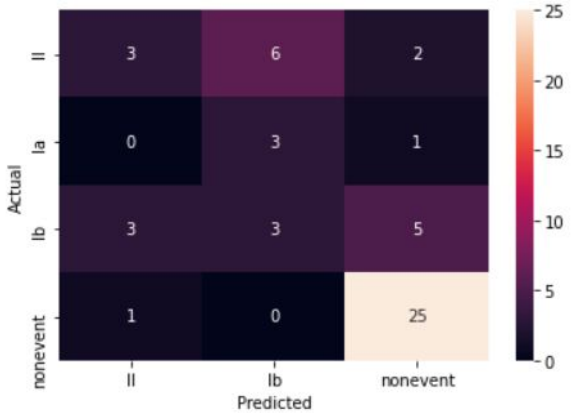
The parameters used for the multiclass model are detailed below. These gave a 5-fold CV accuracy of 69.8% and the same accuracy on the test set.

{'colsample_bytree': 0.7604881960143233, 'gamma': 0.08182797143285225, 'learning_rate': 0.07927973937929789, 'max_depth': 2, 'n_estimators': 177, 'subsample': 0.8092261699076477}



KNN

Parameter tuning: The parameter to tune is the number of nearest neighbours. Chosen in the same way as the binary classifier, k was set to equal to 9.

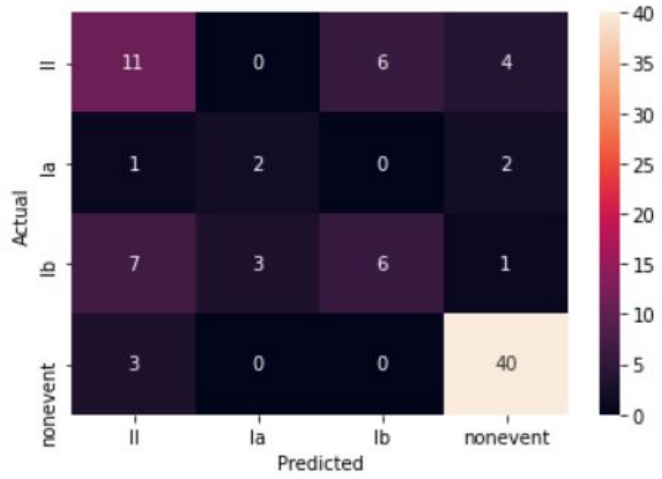


Naive Bayes

SVM

The parameters for the multiclass model were once again found using the random grid search and accuracy was measured using 5-fold CV. The final parameters gave a 5-fold CV accuracy of 69.1% for the SVM multiclass classifier.

```
{'C': 109.53031576544694, 'gamma': 0.0005494254346819604, 'kernel': 'rbf'}
```



Multiclass Classifier Conclusions

	Decision Tree	Random Forest	XGB	k- Nearest Neighbour	Naive Bayes (with Best K)	Naive Bayes (with PCA)	SVM
Training	66%	100%	100%	66%	62%	69%	83%
Validation	64%	66%	70%	58%	64%	62%	69%
Test	67%	72%	70%	58%	62%	65%	68%

Blended Model Outcome and Results:

Similarly to the binary classifier, the final step of the model was to combine the best performing classifiers and calculate a blended probability, which then would classify the test observation into the class which had probability greater than 0.5.

The models chosen for the blend were SVM, XGB and Naive Bayes. These were chosen due to high accuracy over the test set.

The final accuracies of the blended model are given by:

Accuracy on train set: 0.937984496124031

Accuracy on validation set: 0.9534883720930233

Accuracy on test set: 0.686046511627907