Introduction to Machine Learning
Fall 2020

Predicting new particle formation events with Machine Learning

Julia Sanders, Bernardo Williams and Mikko Saukkoriipi

December 11, 2020

University of Helsinki
Faculty of Science

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Degree programme
Faculty of Science	Introduction to Machine Learning Fall 2020

Tekijä — Författare — Author

Julia Sanders, Bernardo Williams and Mikko Saukkoriipi

Työn nimi — Arbetets titel — Title

Predicting new particle formation events with Machine Learning

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidantal — Number of pages
Report	December 11, 2020	19

Tiivistelmä — Referat — Abstract

Several machine learning classification models are used over the dataset npf_train.csv divided in the purpose of predicting a binary label a the purpose of predicting a multi-class label. The objective is to extend the model and predictions to unseen data, and also to give an estimate of the accuracy the model will have on the unseen data.

For the fit of the models we used two data reduction techniques, PCA [3] and best feature selection and two normalization methods, min-max and standardize normalization. We tried fitted algorithmic, generative and discriminative methods using either validation or cross validation for both of the problems and found which ones performed the best in terms of accuracy over an unbiased test set. At last we found that taking the average prediction of the best algorithmic, discriminative and generative methods gives estimates with less variance and higher accuracy.

Avainsanat — Nyckelord — Keywords

Machine Learning, Naive Bayes, KNN, Decision Tree, XGB, SVM

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

1. Introduction

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).

2. Preliminaries

2.1 Initial Analysis

Before beginning the modelling, we investigated the level of correlation between the variables, by producing a correlation matrix. As the figure below shows, several variables were highly correlated. The data given contained means and standard deviations of atmospheric variables, so the high correlation could be caused by means and standard deviations of the same variable.

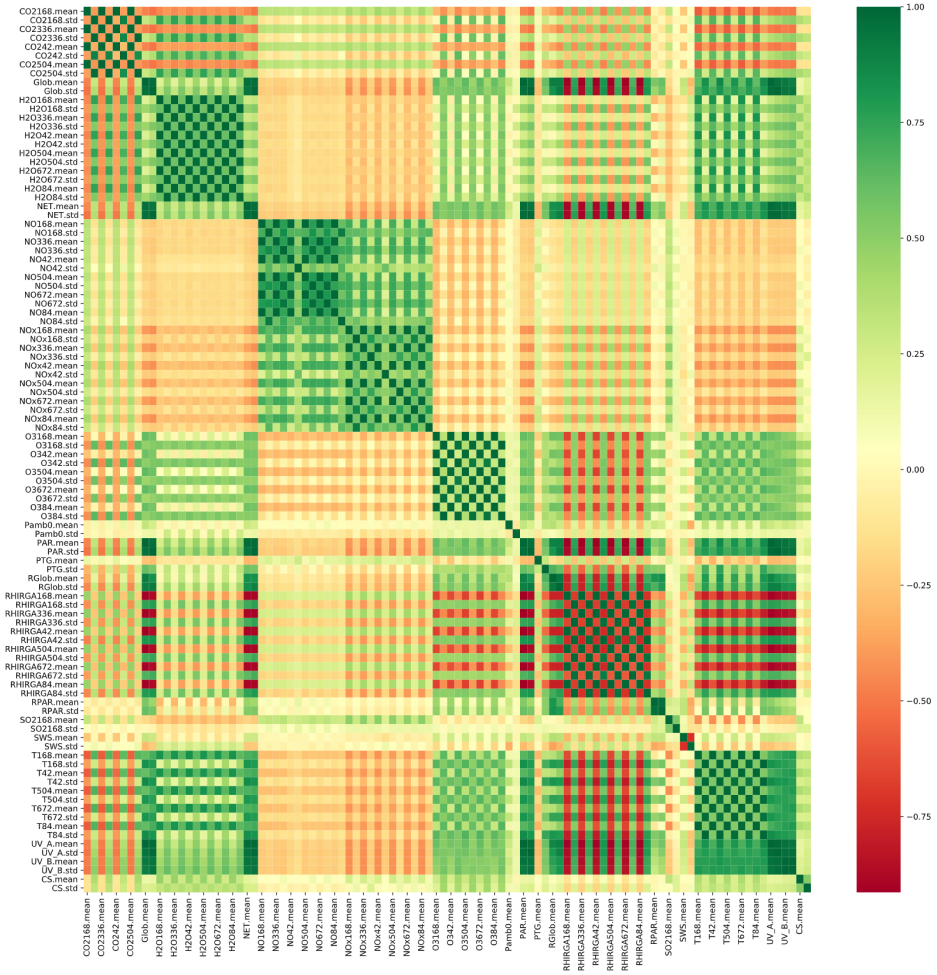

Figure 2.1: Correlation Matrix

2.2 Data Cleaning

To initially prepare the data for modelling [1], 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.

2.3 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 split			
	Train	Validation	Test
nonEvent	49.3%	50.0%	50.0%
Ia	7.1%	7.7%	7.7%
Ib	21.4%	21.2%	21.2%
II	22.1%	21.2%	21.2%

2.4 Normalization

To improve accuracy of our results and to avoid to avoid numerical problems, we normalised the data between 0 and 1. The data was normalised before splitting into the separate training, validation and test sets. We also normalized the data by scaling the data with the standard deviation and taking out the mean.

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.

2.5 Feature selection and dimension reduction

The data set itself had a very high dimensional, and by our initial analysis, we found that many variables were correlated with each other. To counter the effects of this in our models, we used methods of feature selection [4].

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.

2.5.1 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 [2].

2.5.2 Principal Component Analysis

This is a method of unsupervised machine learning, which helps to summarise a large number of variables in the data by selecting a smaller number of variables as representative samples. The representative samples chosen are those which explain most of the variance in the data set.

3. Binary Classifiers

3.1 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.

Figure 3.1 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: Figure 3.3 shows the final decision tree output, and below that, figure

3.2 the heatmap of results. Overall, this gave an 88 % accuracy rate on the test set.

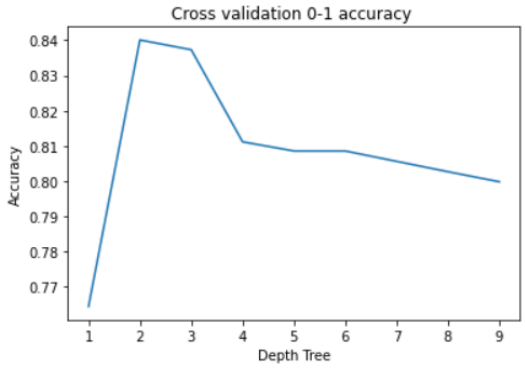

Figure 3.1: Binary classification decision Tree Cross validation accuracy

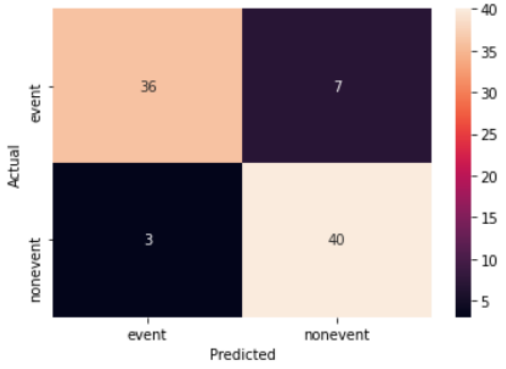

Figure 3.2: Binary classification decision tree confusion matrix

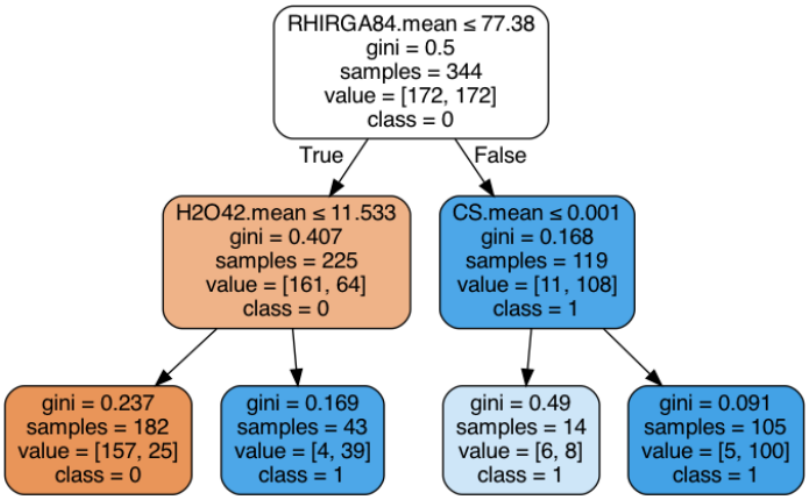

Figure 3.3: Decision tree output

3.2 Random Forest

Description: The Random Forest classifier is an extension of decision trees that works 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 and on each node of each tree chooses randomly a subset of variable to reduce the variance of the model. 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.

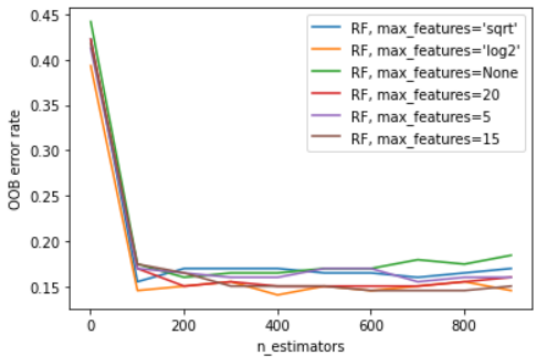

Figure 3.4: Random forest error rate

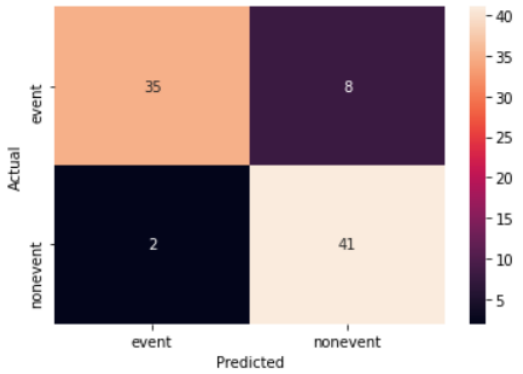

Figure 3.5: Random forest confusion matrix

3.3 Extreme Gradient Bosting

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

Results: With the parameter in figure 3.6, the final accuracy on the test set was 87%. Figure 3.7 shows a confusion matrix of predicted vs actual classes. The parameter values found gave a 5-fold CV accuracy of 90.4 %.

The results of the parameters found from the random grid search on the validation set	
colsample_bytree	0.9915346248162882
gamma	0.4812236474710556
learning_rate	0.10553468874760924
max_depth	3
n_estimators	212
subsample	0.6592347719813599

Figure 3.6: XGB parameters

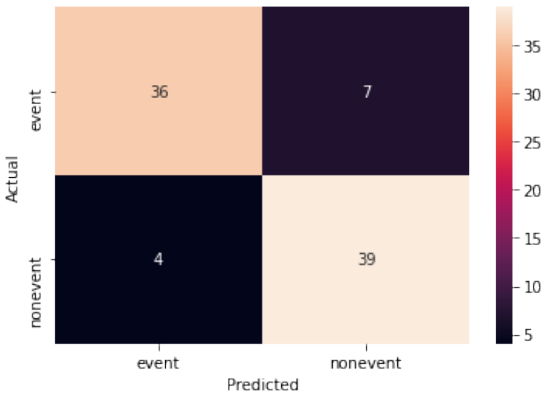

Figure 3.7: XGB confusion matrix

3.4 K-nearest neighbors

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

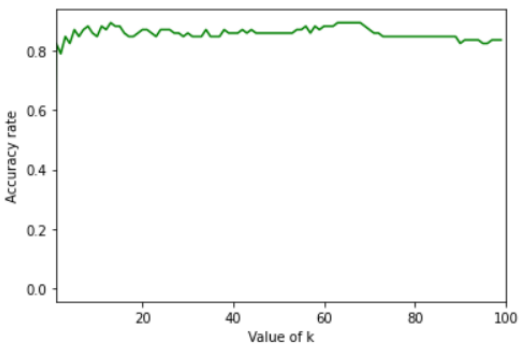

Figure 3.8: KNN accuracy by value of k

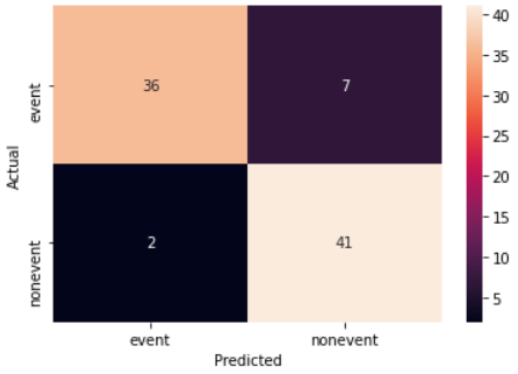

Figure 3.9: KNN confusion matrix

set, and the validation accuracy with k=13 was 89%. The heatmap of predicted vs actual categories is shown below.

3.5 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 confusion matrix of predictions on the test set is shown in figure 3.10.

3.6 Naive Bayes

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

Advantages and Disadvantages: Naive Bayes is a well-performing model, however,

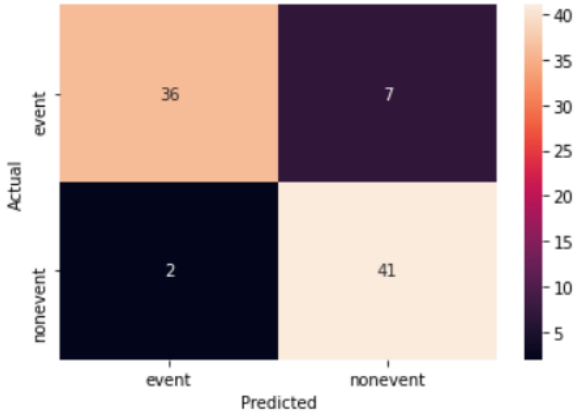

Figure 3.10: Logistic regression confusion matrix

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: The Naive Bayes classifier was tested with normalized and not normalized parameters using best k feature selection and PCA dimension reduction. Surprisingly in binary classification, select best k features worked better with the normalized data, but PCA results were better without normalization. The graph 3.11 show the model accuracy on the training, validation and test sets as the number of k features and graph 3.13 as the number of PCA components.

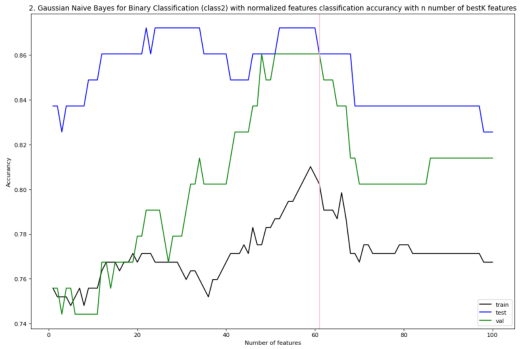

Figure 3.11: Naive Bayes Binary accuracy by number of bestK features

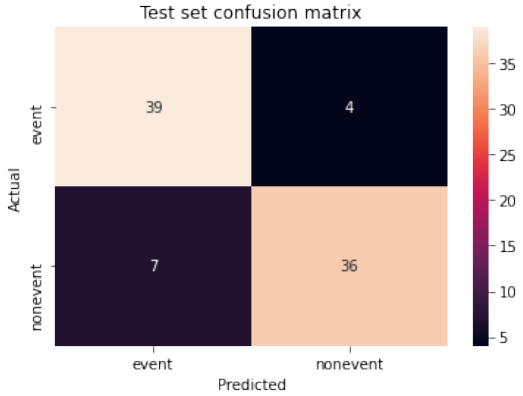

Figure 3.12: Naive Bayes Binary with kBest confusion matrix

With select best k features, the best result was found with k=60. This increased the model accuracy from 84 % to 87 %. The same tuning process was repeated with features selected using Principal Component Analysis (PCA). The optimal number of PCA components for Naive Bayes binary classification was found to be 12. This improved model prediction accuracy from 84 % to 93 %.

Results The highest accuracy obtained on the test set was 93%, using not nor-

malized data and 12-PCA components.

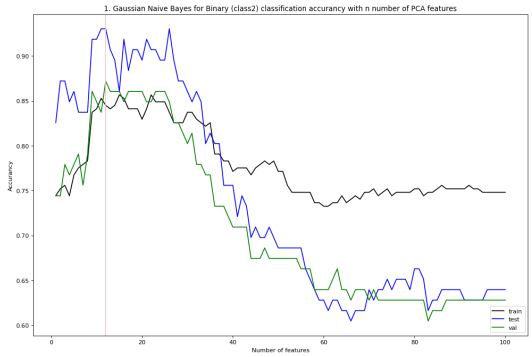

Figure 3.13: Naive Bayes Binary accuracy by number of PCA components

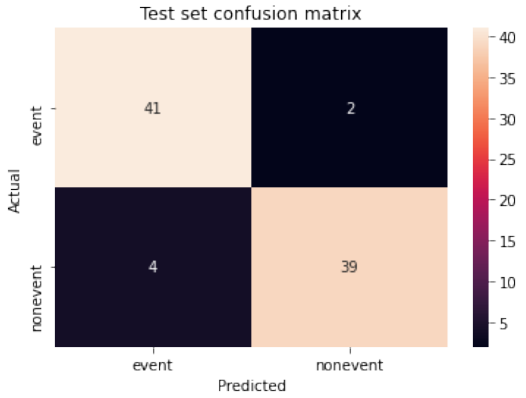

Figure 3.14: Naive Bayes Binary with PCA confusion matrix

3.7 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 3 parameters needed for tuning were found through a random grid search, with the error estimated through cross validation. It is important to note that although not deatiled here, we tried PCA and best feature selection over SVM, but it did not show in any improvement over the accuracy of the test set for the best values od K chosen through cross-validation. That's why we decided to keep all variables.

- 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 in the table 3.15. These gave a 5-fold CV accuracy of 89.8%.

Results: The final accuracy obtained on the test set for the binary classifier was 83.7%. The confusion matrix produced is in figure 3.16.

The optimal parameters for Support Vector Machine	
C	11.0919987857
Gamma	0.00086426744
Kernel	linear

Figure 3.15: Optimal parameters for SVM

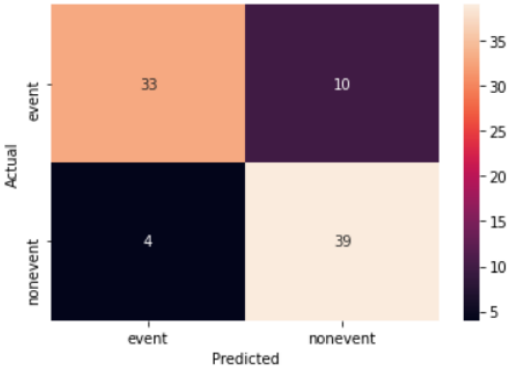

Figure 3.16: SVM confusion matrix

3.8 Summary of accuracies for Binary Classifiers

In the table 3.17 are all the models and their accuracies we tested for binary classification. The models we chose for our blended model are bolded.

Summary of binary models accuracies			
	Training	Validation	Test
Naive Bayes	**84%**	**87%**	**93%**
Logistic Regression	**87%**	**87%**	**89%**
Random Forest	100%	87%	88%
Decision Tree	88%	84%	88%
XGB	**100%**	**90%**	**87%**
SVM	98%	90%	83%
KNN	85%	78%	80%

Figure 3.17: Summary of tested binary models sorted by test accuracy.

3.9 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 methods. 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".

Accuracies of Binary the blended model	
Training	96.12 %
Validation	96.51 %
Test	91.86 %

Figure 3.18: Accuracies of the binary blended model

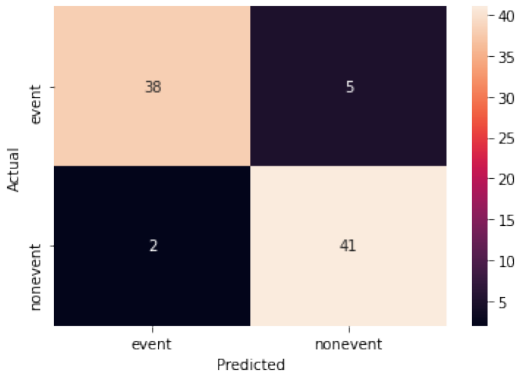

Figure 3.19: Binary blended model confusion matrix

4. MultiClass Classifiers

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

4.1 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 0-1 loss. The range of depths and their respective accuracies are shown in the figure 4.1. As in the binary classifier, a depth of 2 was chosen to maximise the accuracy on the test set.

Overall, the model gave an accuracy rate of 60% on the test set. The final confusion matrix of predictions is shown in figure 4.2, which shows that a 2 level decision tree classifies all observations into nonevent or into lI. 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.

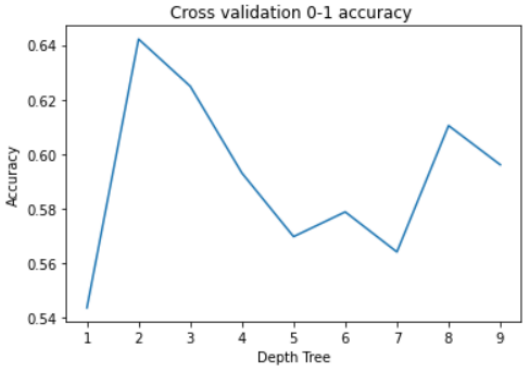

Figure 4.1: Decision tree MultiClass Accuracy

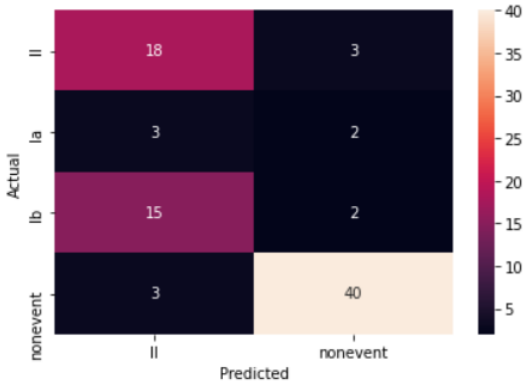

Figure 4.2: Decision tree MultiClass confusion matrix

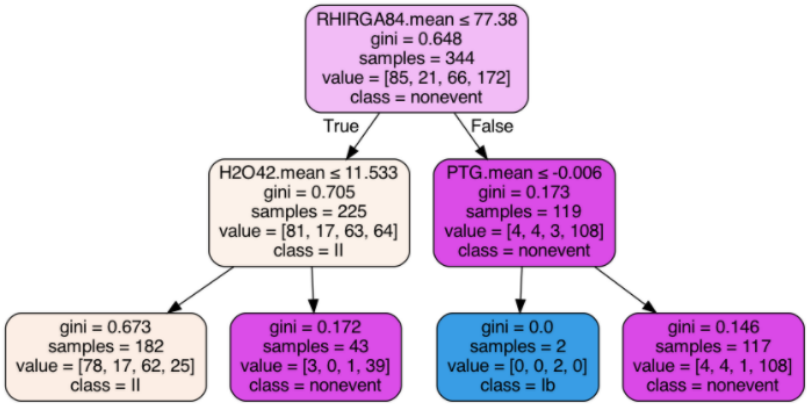

Figure 4.3: Decision tree MultiClass model

4.2 Random Forest

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 in figure 4.4. 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.

4.3 Extreme Gradient Boosting

Tuning: Extreme gradient boosting hyper-parameters were chosen through random grid search through cross validation. The parameters chosen 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.

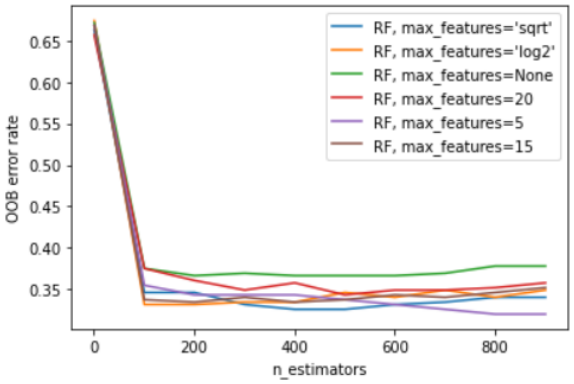

Figure 4.4: Random Forest MultiClass error rate

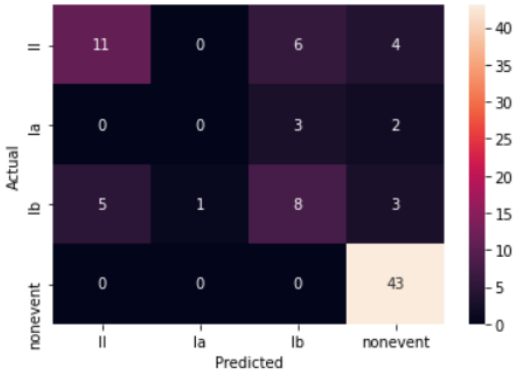

Figure 4.5: Random forest MultiClass confusion matrix

The optimal parameters for XGB	
Colsample bytree	0.7604881960
Gamma	0.08182797143
Learning rate	0.07927973938
Max depth	2
N estimators	177
Subsample	0.80922616991

Figure 4.6: Optimal parameters for Extreme Gradient Boosting

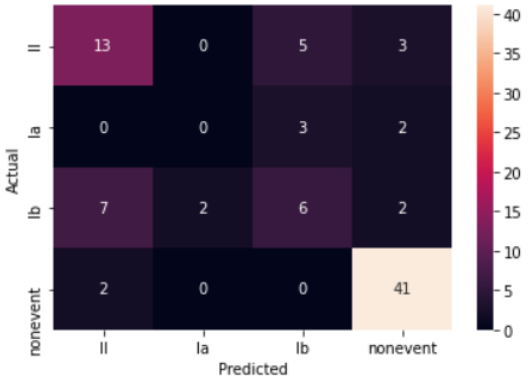

Figure 4.7: Extreme Gradient Boosting MultiClass confusion matrix

4.4 K-nearest neighbors

Parameter tuning: The parameter to tune is the number of nearest neighbours. This parameter was chosen in the same way as the binary classifier and the value of k used on the test set was equal to 9. Confusion matrix is shown in the figure 4.8.

4.5 Naive Bayes

As with the binomial (class2) prediction, the Naive Bayes model was tested with and without normalized parameters and with two different dimension reduction methods.

Without normalizing columns and before using feature selection or dimension reduction, the Naive Bayes prediction accuracy was 53%. After implementing optimized best K feature selections [4.9], the accuracy increased to 64%. After this, we implemented PCA feature reduction, and accuracy increased further to 69% [4.11]. This was a great example of the feature selection when we increased accuracy from 53% to 69%.

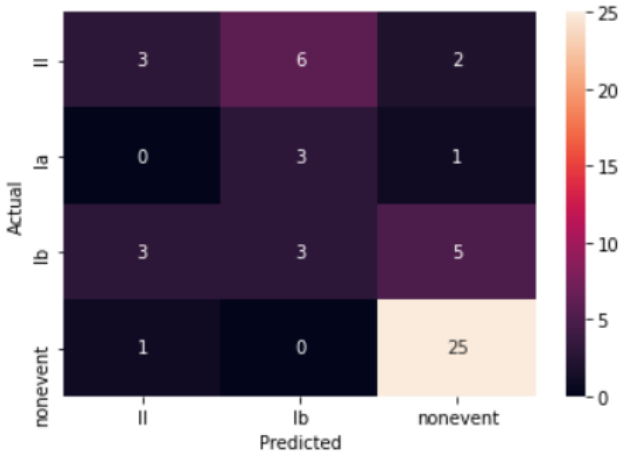

Figure 4.8: KNN MultiClass confusion matrix

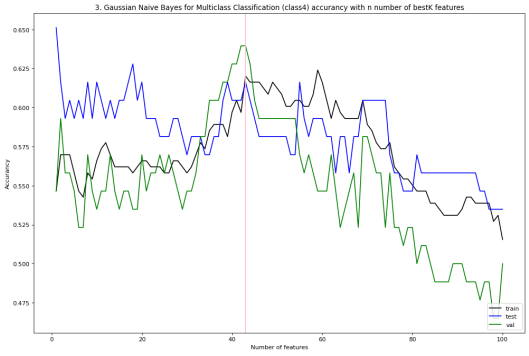

Figure 4.9: Naive Bayes Multiclass accuracy by number of best K features

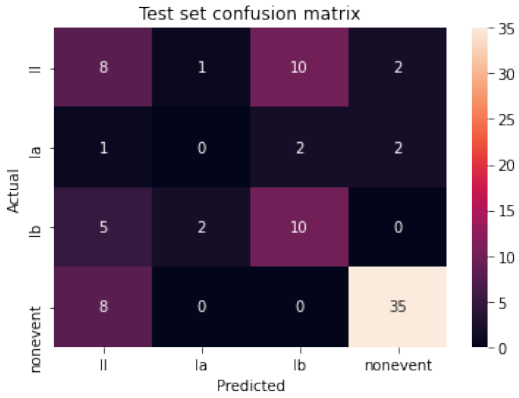

Figure 4.10: Naive Bayes Multiclass with kBest confusion matrix

4.6 Support Vector Machine

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.

The optimal parameters for Support Vector Machine	
C	109.53031576544694
gamma	0.0005494254346819604
Kernel	rbf

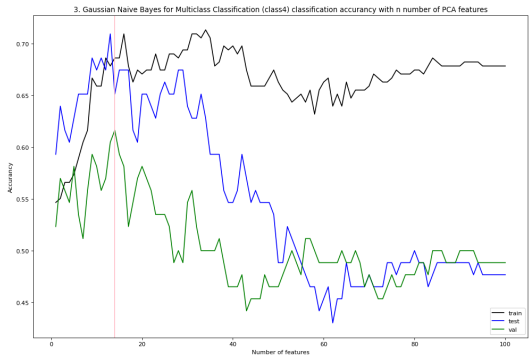

Figure 4.11: Naive Bayes Multiclass accuracy by number of PCA components

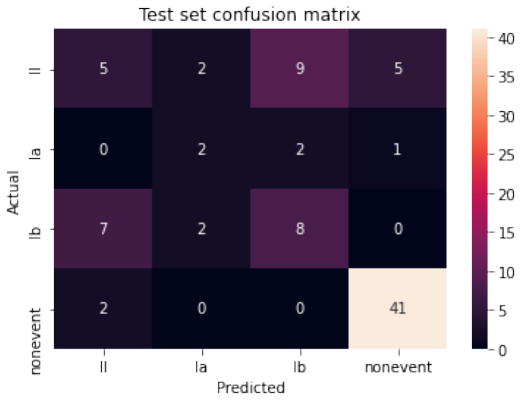

Figure 4.12: Naive Bayes Multiclass with PCA confusion matrix

4.7 Summary of accuracies for Multiclass Classifiers

Summary of multiclass models accuracies			
	Training	Validation	Test
Random Forest	100%	66%	72%
XGB	**100%**	**70%**	**70%**
SVM	**83%**	**69%**	**68%**
Decision Tree	66%	64%	67%
Naive Bayes	**69%**	**62%**	**65%**
Log Reg	72%	54%	65%
KNN	66%	58%	58%

Figure 4.13: Summary of multiclass models accuracies sorted by test accuracy. Bolded models used are used in blended model.

4.8 Multiclass Classifier Blended model

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 which has the highest probability. 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 can be found in table 4.14 and confusion matrix in figure 4.15.

Accuracies of the Multiclass blended model	
Training	94.96 %
Validation	97.67 %
Test	69.77 %

Figure 4.14: Accuracies of Multiclass blended model

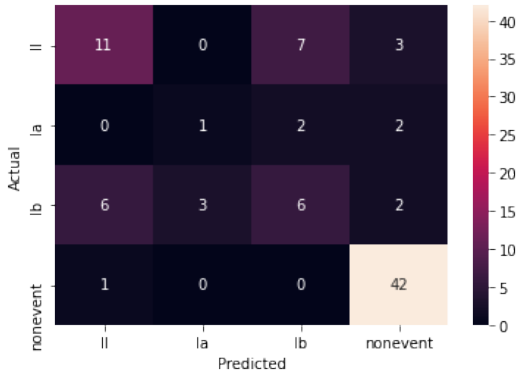

Figure 4.15: Confusion Matrix Multi-class Blend

5. Conclusions

This has been an exciting project, and it was interesting to implement our newly learned skills into the real problem. Besides implementing the skills we had learned during this course, we also spent a significant amount of time studying Machine Learning and feature tuning to receive the best possible model. Since there was a challenge between the project group's models, we decided to take one step further and combine the best models for both binary and multiclass predictions.

When we started this project, our first approach was to find the models that can give the best accuracy for binary and multiclass prediction. We expected that the best models for these two tasks might be different, and that's why we separately tested binary and multiclass prediction accuracy for each model. Our hypothesis was revealed to be partly correct when while the margins between the best models were relatively low [ref bin table and ref multi table].

We decided to choose XGB, Logistic regression and Naive Bayes with PCA for the binary blended model. In the binary prediction task, the models had accuracies of 87%, 89%, and 93%. After combining these three models, the final accuracy was ended to be 92%. In the multiclass blended model, We decided to choose SVM, XGB, and Naive Bayes with PCA. The model accuracies were 68%, 70%, and 65%. The blended model received an accuracy of 68.60%. It is important to note that blending the models reduces the variance of the accuracies over train, val and test set. Although

the accuracy of NB has higher accuracy on test, it has smaller accuracy on train and validation (84% and 87% respectively). The blended model is more consistent on training, validation and test accuracies.

The blended model accuracies seem to be close to the accuracies of its parts. It seems like all the models have problems with the same classifications task. Still blending improves the accuracy by a small margin in binary classification task, and in both cases gives more consistent accuracies over training, validation and test.

5.1 Evaluation of our Model's Performance

Comparing with the published results, our model obtained a final accuracy of 87% on the unseen data set. In attempt to explain this good performance, we reviewed the likeliest explanations for errors, and where our model was able to overcome common error causes. In particular, we looked at the trade-off between variance and bias.

The combination of multiple methods would help to reduce both the bias and the variance. Intuitively, by averaging different models, we have a better chance of nearing the true distribution, thus reducing the bias type error. In addition, using the mean of probabilities causes the overall variance error of predictions to be lowered, by 'softening' the impact of each observation on the overall method. For example, if a particular observation highly affected one model, this would be balanced by the averaging method.

The perplexity of our model was very low, at 1.3422 on the unseen data set. Intuitively, his is likely due to the low variance of the predictions. Perplexity assigns large weights to incorrectly classified observations. Therefore, the blended model which reduces variance of the predictions should, in general, also have a lower variance on the unseen data

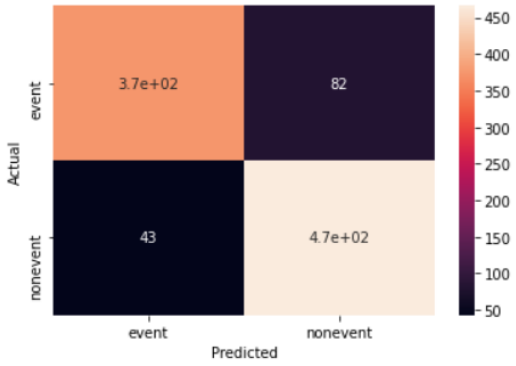

Figure 5.1: Confusion Matrix for binary predictions on the unseen data set

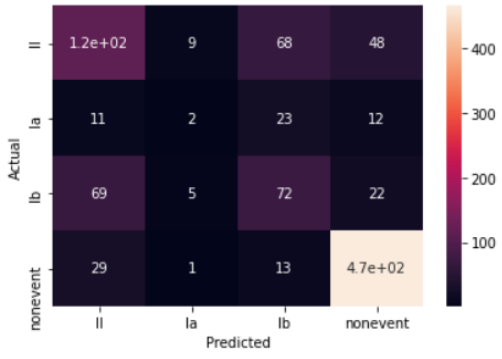

Figure 5.2: Confusion Matrix for multi-class predictions on the unseen data set

References

[1] Projects GitHub repository: https://github.com/williwilliams3/TermProjectIML.

[2] scikit-learn, select best k.

[3] G. James, D. Witten, T. Hastie, and R. Tibshira. *An Introduction to Statistical Learning*. Springler Texts in Statistics, 2017.

[4] A. Kulakov. Towards data science: Feature selection techniques in machine learning with python, 2016.