Introduction to Machine Learning
Fall 2020

Predicting New Particle Formation Events with Machine Learning

Julia Sanders, Bernardo Williams and Mikko Saukkoriipi

December 27, 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 27, 2020	21

Tiivistelmä — Referat — Abstract

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

For fitting the models we used two dimension reduction techniques, PCA[3] and best feature selection and two normalization methods, min-max and standardizing normalization. We tried fitting algorithmic, generative and discriminative methods using either validation or cross validation to measure accuracy for both the binary and multiclass classifiers and found which ones performed the best in terms of accuracy over an unbiased test set. Lastly, we found that taking the average prediction of the best algorithmic, discriminative and generative methods gave estimates with higher accuracy and more consistent accuracy over train, validation and test.

Avainsanat — Nyckelord — Keywords

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

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 separate 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 starting modelling, we investigated the level of correlation between the variables. To do this, we produced a correlation matrix (see Appendix A.1). This figure showed us that several variables were highly correlated, which emphasised the importance of good parameter tuning and feature selection when creating the models.

2.2 Data Cleaning

To initially prepare the data for modelling [1], the following steps were made:

- removed the "ID" and "partlybad" columns
- 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 Normalisation

To improve accuracy of our results and avoid numerical problems, we normalised the data used in the models. This was carried out before splitting, to ensure all data was normalised in the same way. We tried two approaches for normalisation: the min-max scaler, which scales all data between 0 and 1; and the standard scaler, which fits data to have a mean of 0 and standard deviation of 1. Both normalisation methods were implemented using the sklearn_preprocessing library.

Some models did not use the normalised data, for example decision trees. Whenever normalisation had some impact on the model, this is included in the model discussion.

2.4 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. We found that this gave high differences in the error on the validation and test sets. Therefore, we 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. The breakdown of each "class4" category in the training, validation and test sets used is shown in the table below.

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%

2.5 Feature selection and dimension reduction

The data set itself was 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 a chi-squared test 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: Decision trees are grown by recursive binary splitting, which separates the training set into regions. The tree assigns new observations to the majority class in the training set of the region they belong to.

Advantages and Disadvantages: Using a decision tree for the classification task was good, as these can handle 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, as 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 calculated using 5-fold cross-validation over the training and validation sets combined. Figure 3.1 shows the depth parameter tuning, and the change in the 5-fold CV 0-1 accuracy as the value of the depth 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. Figure A.1 shows the confusion matrix of results over the test set. Overall, the final decision tree model gave an accuracy of 88 % on the test set.

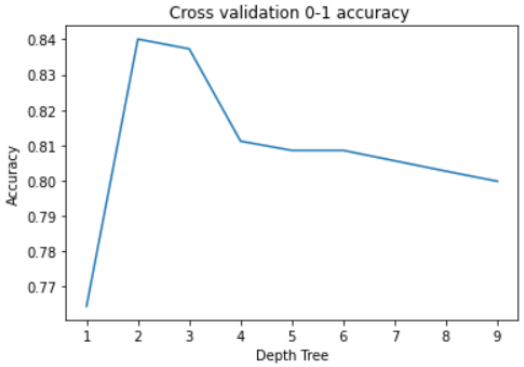

Figure 3.1: Binary classification decision Tree Cross validation accuracy

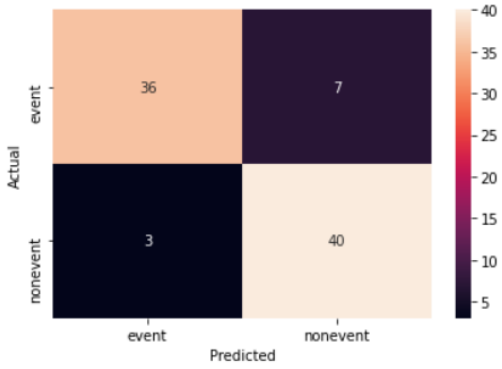

Figure 3.2: Binary classification decision tree confusion matrix

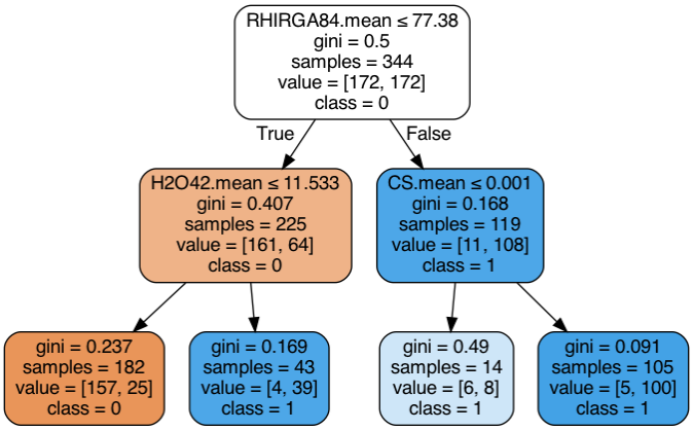

Figure 3.3: Decision tree output

3.2 Random Forest

Description: The Random Forest classifier is an extension of the decision tree 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. At each node, the tree randomly chooses a subset of the variables to reduce the variance of the model.

Advantages and Disadvantages: Random forest classifiers will usually outperform

a single decision tree, and help to overcome overfitting. However, they may still be affected by it, particularly due to the small size of the training data set.

Tuning: The 2 parameters for tuning the model are the number of trees in the RF and the number of variables chosen at each node. The error was calculated using the out-of-bag error estimate. The results of the analysis on the training set are shown in 3.4. 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, and 3.5 shows the confusion matrix of predictions over the test set.

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

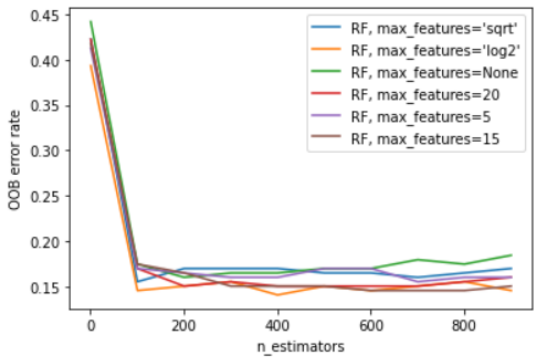

Figure 3.4: Random forest error rate

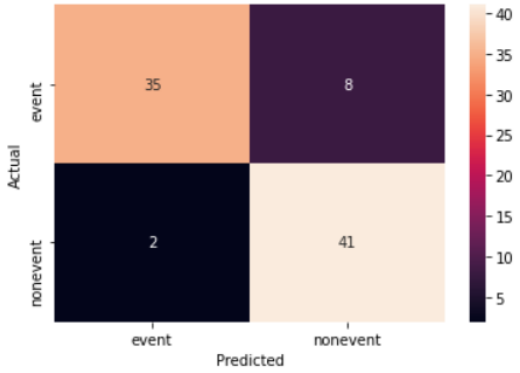

Figure 3.5: Random forest confusion matrix

3.3 Extreme Gradient Boosting

Description: Extreme Gradient Boosting (XGB) is an algorithmic supervised learning method with a decision tree as its base. It constructs multiple decision trees in an iterative way, by considering the residuals left after each tree.

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

Tuning: XGB has 5 parameters for tuning. These values were found through a random grid search, and the accuracies were measured using 5-fold cross-validation. The parameters to find are:

- colsample_bytree: Subsample fraction of variables for each constructed tree. By using different subsamples, we can help to reduce the overall variance of the models.
- gamma: This is the minimum value of the gain needed to make a split at each node. This parameter helps to prevent overfitting by preventing unnecessary splits.

- learning rate: This parameter scales the contribution of each decision tree. Again, this helps to prevent overfitting.
- maximal depth: The maximum depth of each tree
- n estimators: The size of the sample of trees generated.
- subsample: The ratio of the subsample of observations with the training set.

Results: With the parameters as shown 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

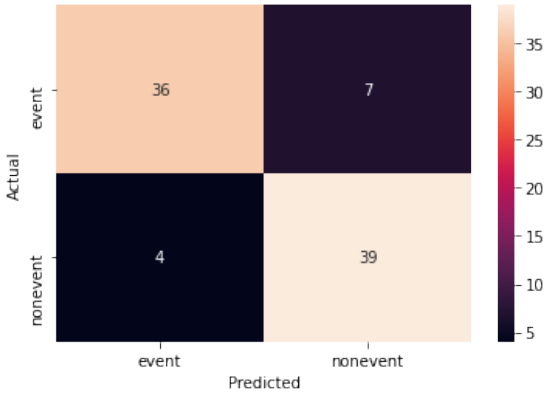

Figure 3.7: XGB confusion matrix

3.4 K-nearest neighbors

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

Advantages and Disadvantages: One challenge of using k-NN on the data was the high dimensional nature of the data. Each data point had 100 components. To overcome this, we used feature selection, such as the select k-best approach and principle 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. A too high 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 in 3.8 shows the validation accuracy rate as the value of k changes. The highest accuracy was obtained at k=13.

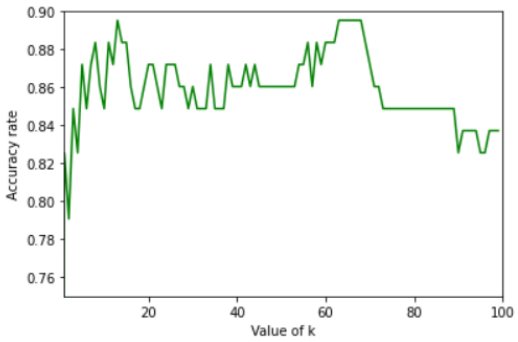

Figure 3.8: KNN accuracy by value of k

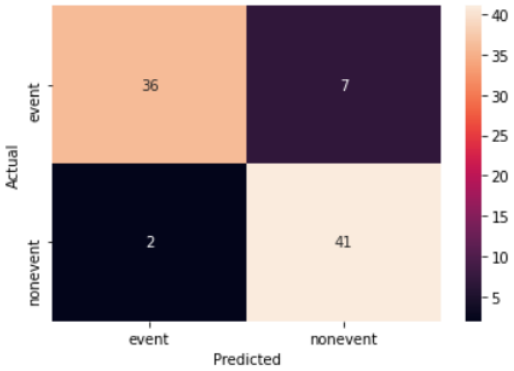

Figure 3.9: KNN confusion matrix

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 confusion matrix of predicted vs actual categories is shown in 3.9.

3.5 Logistic regression

Description: Logistic Regression is a discriminative method that aims to fit 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, can cause problems if the variables in the data are well-separated (i.e. that some are good predictors while others are not). This 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 (betas) shared the same scale, it was important that data used in the logistic regression model was normalised. The normalisation used was the standard scaler, which scales the data to a mean of 0 and standard deviation of 1 as this gave the best and most consistent results. 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.

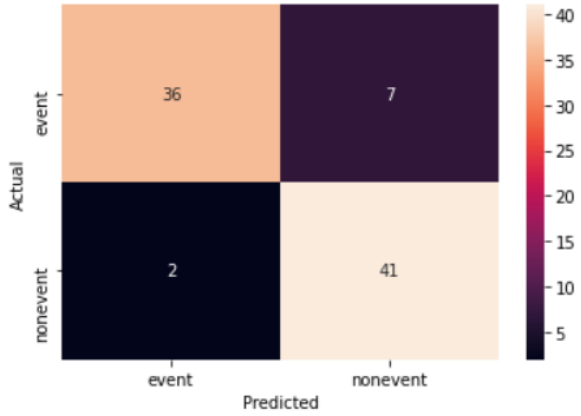

Figure 3.10: Logistic regression confusion matrix

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 that which has the highest probability.

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. From our preliminary look at the data, many variables were highly correlated, and this could indicate that the Naive Bayes assumption could be invalid.

Tuning: The Naive Bayes classifier was tested with both normalised and unnormalised features, and feature selection using both bestK and PCA to reduce the number of dimensions in the data. Surprisingly, in the binary classification, select best k features performed better with normalised data, while results using PCA were better without normalisation. The graph in 3.11 shows the model accuracy on the training, validation and test sets as the number of features chosen by select bestK features varied. The number of features used in the final model was 60. The graph in 3.13 shows the corresponding results as the number of PCA components varied. The optimal number of components was 12, which was used in the final model.

Results: The highest accuracy obtained on the test set was 93%, using unnormalised data and 12 PCA components. The confusion matrix of predictions using this method on the test set is shown in 3.14. 3.13 shows the confusion matrix of predictions obtained using the bestK feature selection with 60 features. Without feature selection, the Naive Bayes approach gave an accuracy of 84% on the test set. Both methods of feature selection improved the accuracy of Naive Bayes: using bestK features gave an accuracy of 87%; whereas using PCA gave an accuracy of 93%.

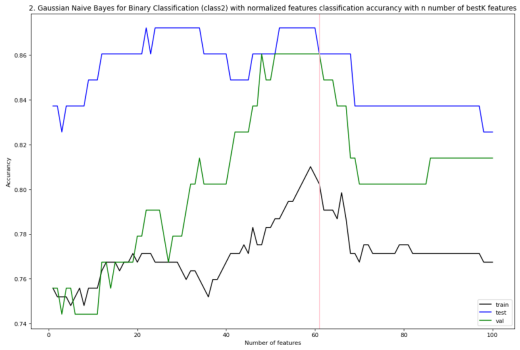

Figure 3.11: Naive Bayes Binary accuracy by number of bestK features

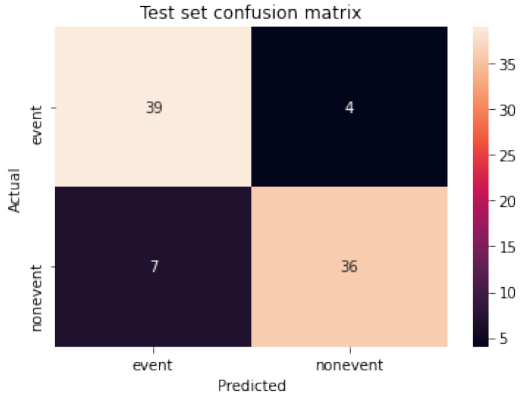

Figure 3.12: Naive Bayes Binary with kBest confusion matrix

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

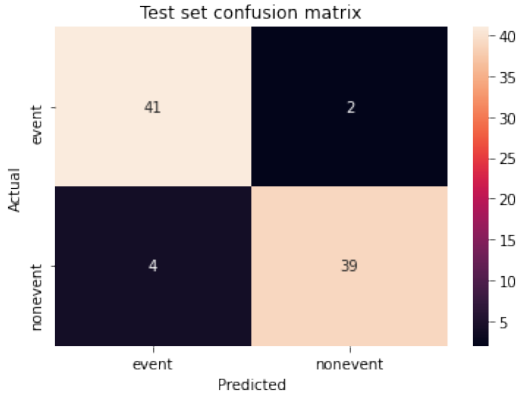

Figure 3.14: Naive Bayes Binary with PCA confusion matrix

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. The error was estimated through 5-fold cross validation and all variables were used. It is important to note that, although not detailed here, PCA and best feature selection were tried over SVM, but did not show any improvement in accuracy on the test set. The parameters needed for SVM are listed below. These were found by random grid search on the validation set.

- C: Regularization parameter. (l2-regularization)
- gamma: Kernel coefficient for “rb”,“pol” 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% over the validation and training set.

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

Figure 3.15: Optimal parameters for SVM

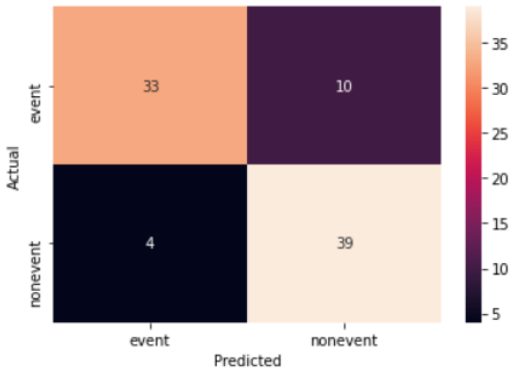

Figure 3.16: SVM confusion matrix

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

3.8 Binary Classifier Blended Model

To produce the final binary classifier, we chose to blend the best performing models. The idea was that this would temper the influence of any one particular model. Table 3.17 shows a summary of all models tried and their accuracies on the training, validation and test sets. The models chosen are shown in bold.

The models chosen for the final blend were XGB, Naive Bayes and Logistic Regression. This was chosen as a combination of algorithmic, generative and discriminative methods. The choice was based on not only the models which had a high accuracy, but also those which had a consistent accuracy over the validation and test sets. Another consideration for selection was that the model output a classification probability, as this would help in the blending process.

In combining the methods, firstly all models were re-trained on the entire training data (combining the training, validation and test sets as described above) using their respective parameters, normalisations and feature selections. The classification probability for “event” was then calculated separately in each selected model. Next,

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.

we computed the mean of these three probabilities: if the mean probability calculated was greater than 0.5, the data point would then be classified as "event".

The table in figure 4.19 shows the accuracies on the training, validation and test sets of the final blended model. Note that the combined accuracy on the test set is higher than all individual models, with the exception of the Naive Bayes. Figure 3.19 shows the confusion matrix of the blended model's predictions on the test set. The accuracies on train, validation and test are more consistent than the individual models, the accuracy on train and validation are close, and the test accuracy is right below.

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

Figure 3.18: Accuracies of the binary blended model

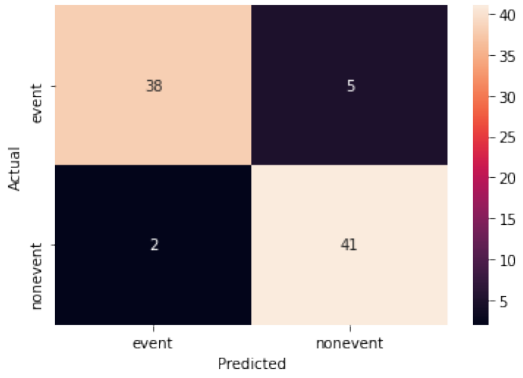

Figure 3.19: Binary blended model confusion matrix

4. MultiClass Classifiers

4.1 Decision Tree

A similar procedure to the binary classifier was used for the decision tree multiclass predictor. Once again, the depth of the tree was selected using 5-fold Cross Validation 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 classified all observations into either nonevent or into II. Looking at the tree plot, we can see that the tree with depth 2 can at most classify into 3 classes, as there are no regions that have a majority class of Ia. The class of Ib is also unlikely to be predicted, because there are very few observations in that majority region in the training set.

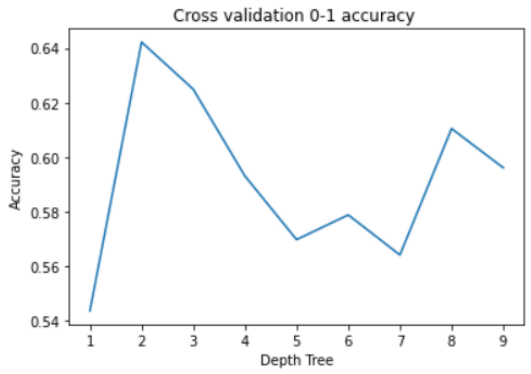

Figure 4.1: Decision tree MultiClass Accuracy

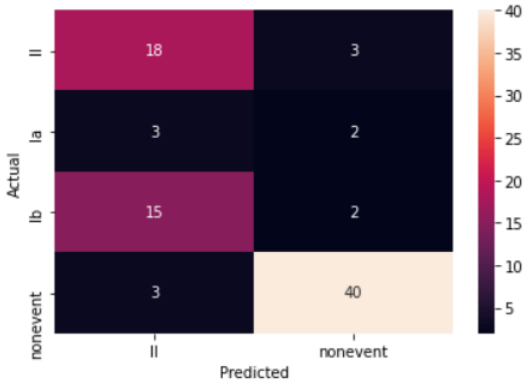

Figure 4.2: Decision tree MultiClass confusion matrix

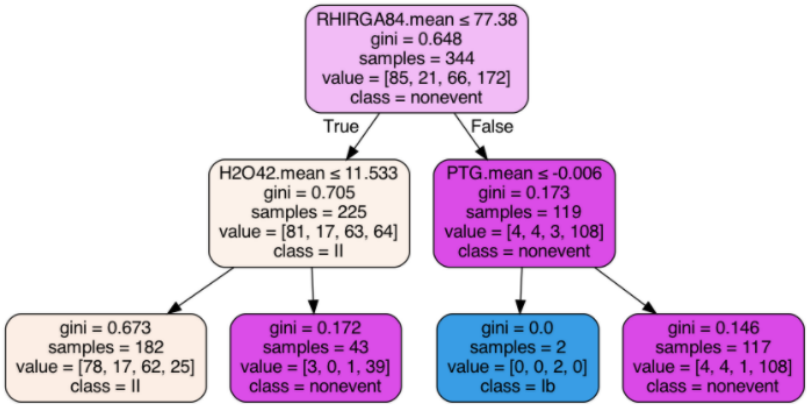

Figure 4.3: Decision tree MultiClass model

4.2 Random Forest

As with the model used for the binary classifier, the model was tuned using the out-of-bag error 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.

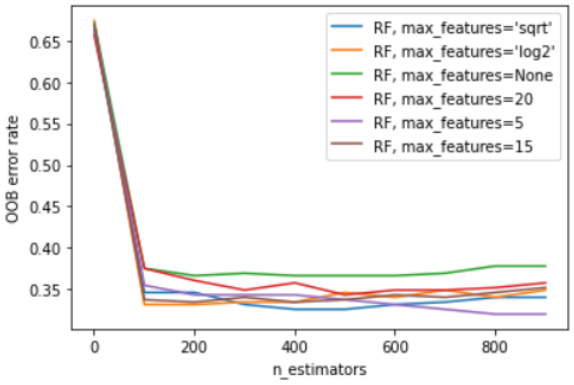

Figure 4.4: Random Forest MultiClass error rate

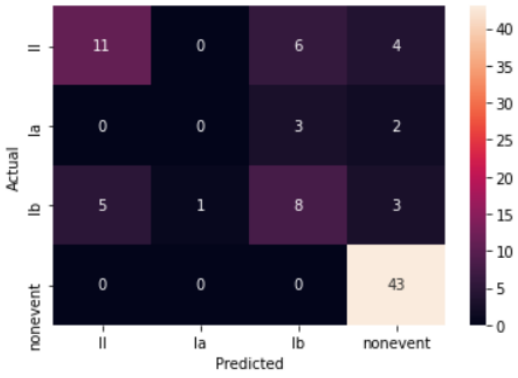

Figure 4.5: Random forest MultiClass confusion matrix

4.3 Extreme Gradient Boosting

Extreme gradient boosting hyper-parameters were chosen through random grid search with accuracy measured by 5-fold cross validation. The parameters chosen for the multiclass model are detailed in table 4.6. These parameters gave a 5-fold CV accuracy of 69.8% and the same accuracy on the test set.

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

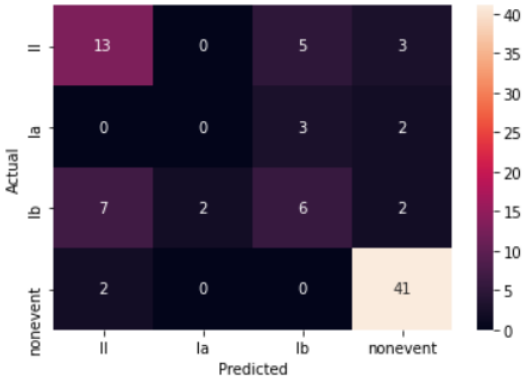

Figure 4.7: Extreme Gradient Boosting MultiClass confusion matrix

4.4 K-nearest neighbors

The parameter to tune was 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. In addition, bestK features was applied to reduce the number of dimensions, but this was not used in the final model, as the results did not show a clear improvement when not all features were used. The effect of feature selection on the model are shown in figure 4.8.

The final accuracy on the test set of this model was 58%. The confusion matrix on the test set is shown in the figure 4.9.

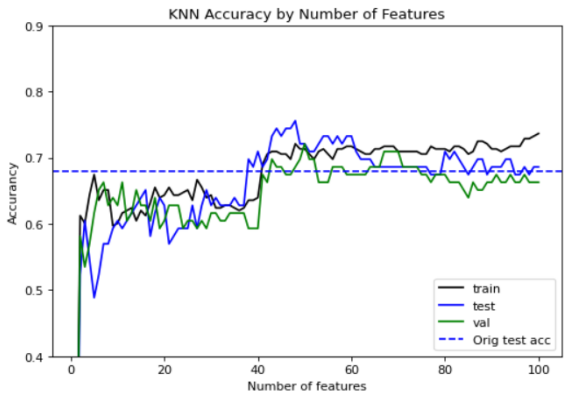

Figure 4.8: KNN MultiClass accuracy by number of best K features

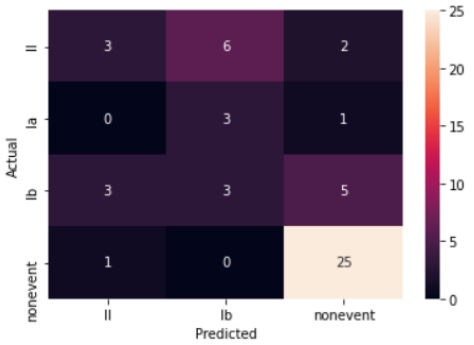

Figure 4.9: KNN MultiClass confusion matrix

4.5 Logistic Regression

The multiclass logistic regression model was implemented similarly to the binary classifier. Once again, normalisation was implemented to ensure coefficients of a shared scale.

One observation of the results obtained by the multiclass logistic regression model is the high difference in the accuracy on the validation set (52%) compared to the accuracy on the test set (65%). In a reliable model, we would expect these errors to take a similar value. As well as this, the logistic regression model output convergence warnings for the multiclass setting. Convergence issues in the logistic regression model can happen when variables are not well separated. From our preliminary analysis of the data, we saw that many variables were correlated, suggesting an explanation for the convergence issues and inconsistent accuracy results. The confusion matrix of predictions for the test set is shown in figure 4.10.

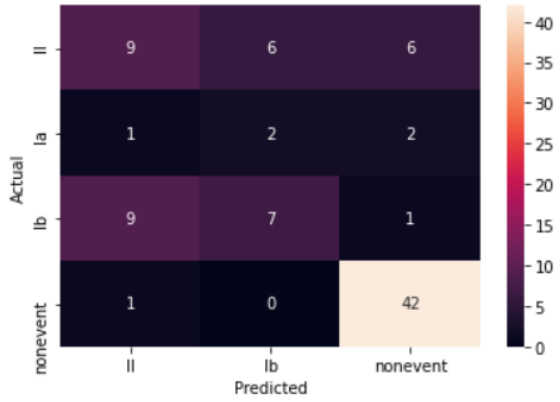

Figure 4.10: Logistic Regression MultiClass confusion matrix

4.6 Naive Bayes

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

Without normalising features and before dimension reduction, the Naive Bayes prediction accuracy was 53%. After implementing the optimised best K feature selection, as shown in figure 4.11, the accuracy improved to 64%. With PCA feature reduction, the accuracy further increased to 69% [4.13].

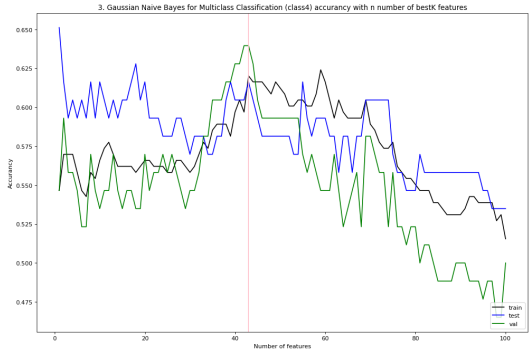

Figure 4.11: Naive Bayes Multiclass accuracy by number of best K features

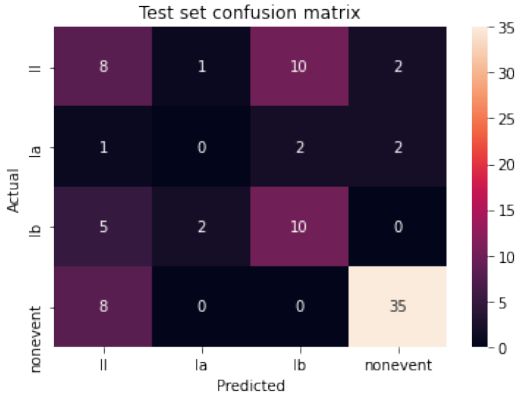

Figure 4.12: Naive Bayes Multiclass with kBest confusion matrix

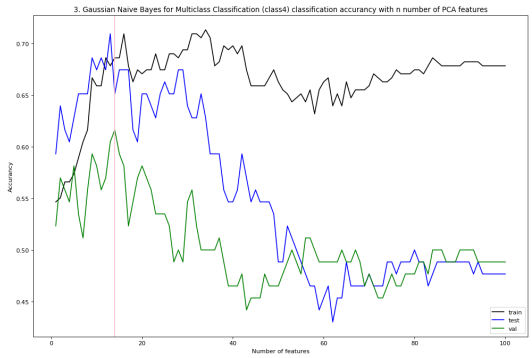

Figure 4.13: Naive Bayes Multiclass accuracy by number of PCA components

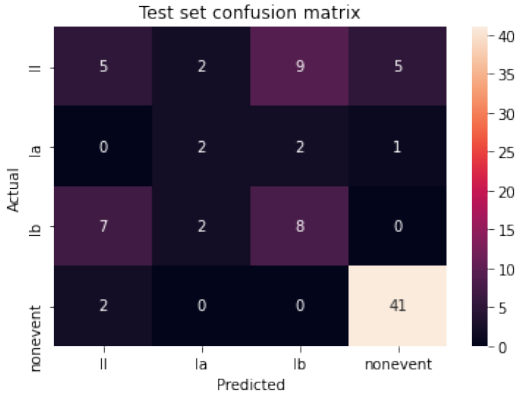

Figure 4.14: Naive Bayes Multiclass with PCA confusion matrix

4.7 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 cross validation. The parameters used in the final model are shown in table 4.15, and gave a 5-fold CV accuracy of 69.1%, and an accuracy of 68% on the test set. The confusion matrix of predictions on the test set is shown in figure 4.16.

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 to the class with highest probability. Table 4.17 shows the full list

Parameters for SVM Multiclass	
C	109.530315
gamma	0.000549425
Kernel	rbf

Figure 4.15: Parameters for SVM multiclass

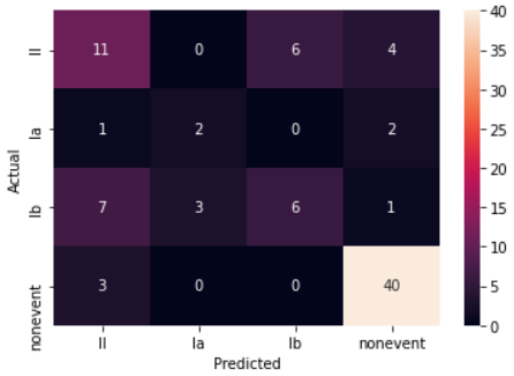

Figure 4.16: SVM Multiclass confusion matrix

of multiclass classification models and their accuracies on the training, validation and test sets. The models chosen for the blended model are shown in bold.

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%**
Logistic Regression	72%	54%	65%
KNN	66%	58%	58%

Figure 4.17: Summary of multiclass models accuracies sorted by test accuracy. Bolded models used are used in blended model.

The models chosen were SVM, XGB and Naive Bayes. These were chosen due to high accuracy over the test set and consistency of results. Unlike in the binary model, logistic regression was not selected. This was due to the large difference in accuracy over the validation and test sets.

The final accuracies of the blended model can be found in table 4.18 and confusion matrix in figure 4.19.

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

Figure 4.18: Accuracies of Multiclass blended model

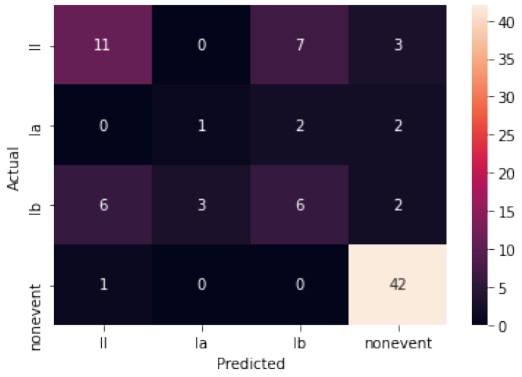

Figure 4.19: Confusion Matrix Multi-class Blend

5. Conclusion

This has been an exciting project, and it was interesting to implement our newly learned skills in a real-life problem. To implement the best possible model, we spent a significant time studying machine learning and feature selection, then implemented the skills we had learnt during the course. Since there was a challenge between the project group's models, we decided to take one step further, by combining the best models to create a blended model 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, which is why we separately tested binary and multiclass prediction accuracy for each model. Our hypothesis was revealed to be partly correct while the margins between the best models were relatively low, tables 4.17 and 4.17.

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 individual accuracies of 87%, 89%, and 93%. After combining these three models, the final accuracy was 92%. In the multiclass blended model, we chose SVM, XGB, and Naive Bayes with PCA. The individual model accuracies were 68%, 70%, and 65%. The blended model had an accuracy of 69.77%. It is important to note that blending the models helped to reduce the variance of the accuracies over the training, validation

and test sets. Although the accuracy of NB was higher on the test set, it had lower accuracy on the train and validation sets (84% and 87% respectively). The blended model performed more consistently on the training, validation and test accuracies.

The blended model's accuracy was close to those of its component models. Each model had problems with many of the same classification tasks.

Compared with the published results, our model obtained a final accuracy of 87% on the unseen data set. Intuitively, by averaging different models, we have a better chance of nearing the true distribution, thus potentially reducing the bias error. In addition, using the mean of probabilities may have caused the overall variance error of the predictions to be lowered, by 'softening' the impact of each observation on the overall method.

The perplexity of our blended model was very low, at 1.3422 on the unseen data set. As perplexity assigns large weights to incorrectly classified observations, the lower variance of probabilities in the blended model may have helped to reduce the perplexity. The small perplexity could also be attributed to the good performance of our binary model, as we had the highest AUC, and if we only looked at the binary classification, our model would have gotten 89.1% accuracy, which would have placed it at the top.

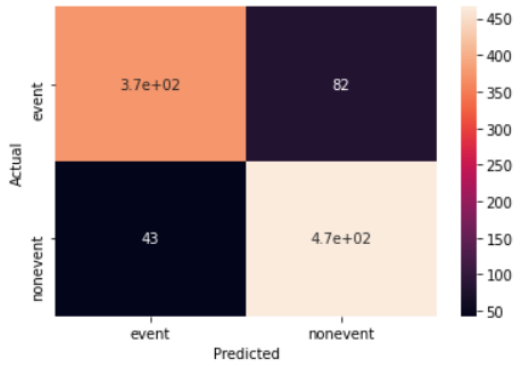

Figure 5.1: Confusion Matrix for binary predictions on the unseen data set

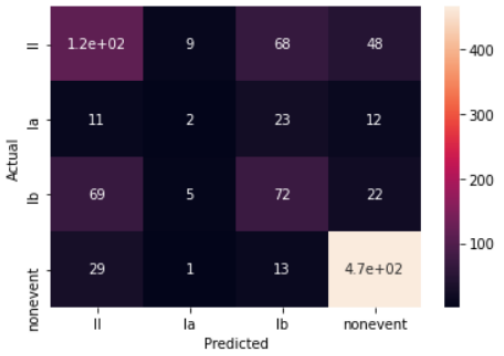

Figure 5.2: Confusion Matrix for multi-class predictions on the unseen data set

A. Appendix

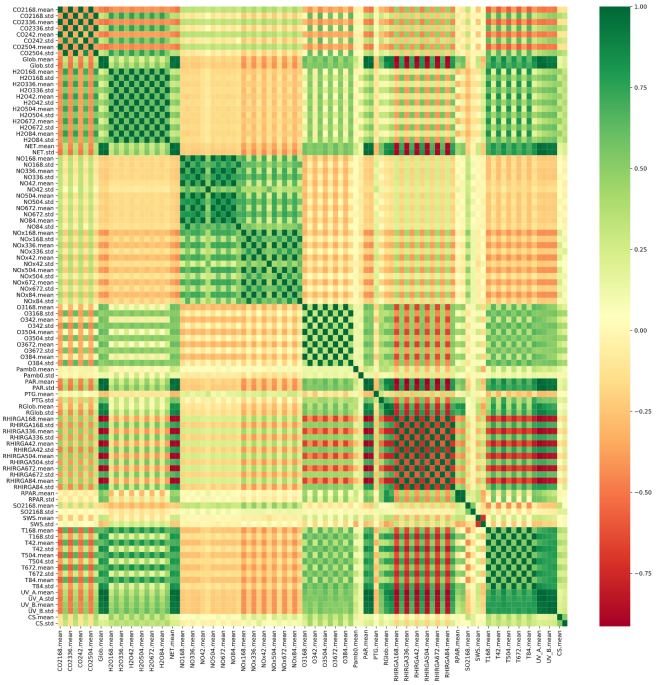

Figure A.1: Correlation Matrix

A.1 Group Work Review:

- Did the group have constructive and focused discussions?
  The group discussions focussed on our overall strategy and the direction of the project. In the group, we were able to share our ideas as well as correct each other's misconceptions on the course material. Meetings were focussed and weekly objectives were agreed on at the end of the meeting. Any questions that arose throughout the week we discussed via chat group on Telegram.

- Did the work proceed as planned towards the commonly agreed objective?
  Our group meetings took place once a week. In the meetings, we agreed on objectives for the week. For example, we agreed to each try out individual models, and we would then discuss results in the following meeting, and decide on improvements.

- Did all group members take responsibility and do their fair share of the work?
  All group members completed the agreed on work throughout the week. Every member participated actively in his work and in the work of the others, a collaborative learning and working environment was formed.

- Did the group work offer benefits for learning?
  The group work was beneficial to the learning, as we were able to support each other's understanding of the course, as well as suggest improvements to each other's work. We were able to reinforce the topics learned in the course and complement them with new topics and models which we leaned on our own. The atmosphere within the group was always friendly.

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.