HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Final Project

Introduction to Machine Learning

Group 93
Ruilin Wang
Johanna Männistö
Zhixu Gu

January 21, 2023

1 Introduction

The models described in this paper are attempting to predict the type of new particle formation (NPF) event given a set of features (observations and measurements). The data provided included data from 464 different days, where 100 different measurements or observations were recorded. Using these features, we attempt to create a model which predicts when an NPF event or non-event occurs given the provided data. This is our binary classification task. After completing this binary classification task we then expanded our search to see about creating a model to predict four different NPF events: Ia, Ib, II, or non-event. In this multi-class classification task, the NPF events Ia, Ib, and II correspond to the class "event" in the binary classification task, while non-event is consistent across both tasks.

We explored various different data processing tasks, implemented several machine learning classifiers, and at times combined classifiers to create an accurate and general classifier for this NPF event prediction task.

In Section 2, we will discuss what we did for data preprocessing, including feature selection and data cleaning. In Section 3, we will move to our exploration on different models, including Logistic Regression, Random Forest, Naive Bayes and KNN, with their corresponding results. Finally, in Section 4 we provide a short conclusion that Random Forest performs best and further insights on this task is discussed.

2 Data Analysis

2.1 Data Exploration

Our first step in tackling the challenge of creating a model which predicts event types given data for a day was to perform preliminary data exploration.

The training data set had a total of 464 data samples and 100 different features. The breakdown of each of these features can be seen in Table 1. With a binary task between "event" and "non-event", there is a equal split between the data however when examining the finer classification the data becomes skewed heavily towards the "non-event" classification and against class "Ia".

	Non-Event	Ia	Ib	II
Count	232	34	85	113
Percentage	50%	7.3%	18.3%	24.4%

Table 1: Category Counts in NPF data set

A correlation matrix for all the variables was generated can be seen in Figure 23 included in the appendix. The multiple variables make it difficult to see clearly but patterns do arise from the matrix. As expected, measurements of the same variable at different heights (ex. CO2168.mean or CO2236.mean) have strong correlations.

There is a significant amount of correlation across variables in this data set, evident in the correlation matrix. However, given that our goal is predictions, issues with multicollinearity shouldn't have a large impact.

2.2 Feature Selection

As found in the Data Exploration section, a large number of features were included in the training data set. We performed a number of different methods to investigate which features to remove in order to reduce the high-dimensionality of the data set. More detailed discussion of feature selection techniques are included for each model.

2.2.1 Feature Contribution

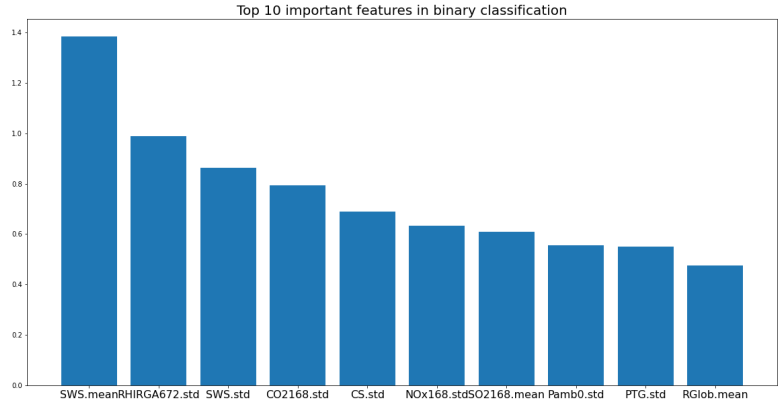

Figure 1: Feature importance

We can evaluate the feature contribution in prediction by checking their weights in the linear model. Feature importance can be calculated by e^w. However, we, here just want to rank the contribution, so the importance is ranked by coefficients. Logistic regression model has an attribute: coeff_. The larger score, the more important variable is. Here we select top 10 important features in doing binary prediction.

2.2.2 SelectKBest

We performed feature selection using SelectKBest from sklearn on a model by model basis. Given a value k, the number of features one wants to select for a model, SelectKBest chooses k features with the best scores. For our purposes we used ANOVA F-value to calculate the scores for each of the features. When applied, model accuracy on the train, cross-validation, and test data sets were calculated as the feature number increased using SelectKBest in order to determine which features to include in a model. Section 3.4.4 illustrates the process taken for Naive Bayes.

2.2.3 PCA

Another technique to reduce high dimensionality is PCA, or Principal Component Analysis. PCA is a unsupervised learning technique that transforms the data set to a smaller set of variables that captures most of the information contained in the larger set. Essentially finding a low-dimensional representation (the principal components) of a high-dimensional data set.

After normalizing the data, we plot and see how much of the data's variance is captured as the number of principle components increases. As seen in Figure 2, we hit 90% cumulative explained variance threshold at 12 principal components. Other techniques recommend choosing the "elbow" of the graph to determine the best number of principal components. Following this method the best number of components would be 4, however this explains less than 80% of the variance in the data set. Given this, we decided to have additional principal components to explain more of the data set.

2.2.4 Variance

In order to reduce feature dimensions, we also examined the variance of all the features. If any features have a zero or very small variance within the training data sets, these features are behaving similarly to

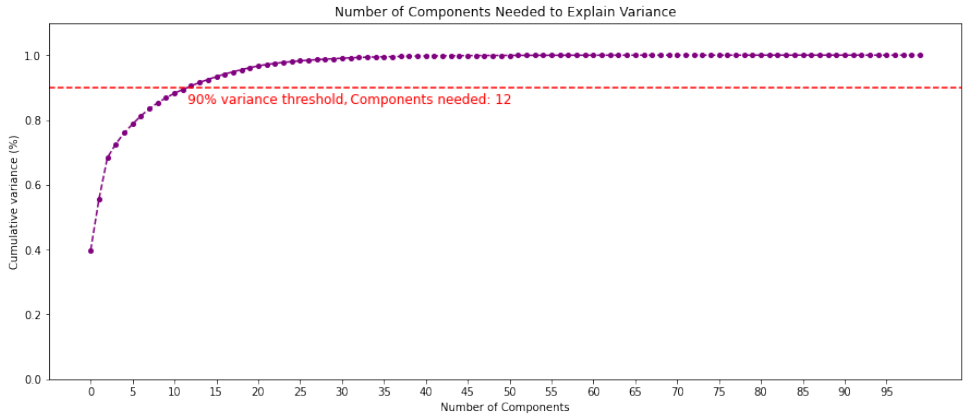

Figure 2: Cumulative Explained Variance as Principal Components Increase

constants. Removing these low variance features is one method to reduce the dimensionality of the data set.

After examining the variances of each feature in the data set, one feature "partlybad" was found to have a variance of 0, while 7 other features were found to have variances of 0.01 or less. These features included:

- partlybad
- NO336.std
- NO504.std
- NO672.std
- PTG.mean
- CS.mean
- CS.std

Models were also trained on data that excluded the aforementioned features. Model performance was compared across the approaches presented in Section 2.2 and the best performing model was selected. Under each model's section, information about which data was used for the final model is included.

2.3 Data Cleaning

To prepare the data for modeling we performed the following steps:

- column "date" was set as the index for the training data
- columns titled "ID" and "partlybad" were removed from the data set
- column "class2" was added in order to classify binary events: nonEvent was equivalent to "class4"'s nonEvent and Event encompassed "class4"'s categories of Ia, Ib, and II.
- data was standardized using StandardScaler from sklearn

2.4 Train-Validate-Test Split

The Train/Validate/Test split was 8:1:1, with a final corresponding sample size of: 371/46/47. The split maintained the proportion of each group in the data set as a whole using stratify.

Table 2 shows the distribution of data across the train, validate, and test splits. Overall, the data is evenly split between the two classes: Non-Event and Event.

Table 3 shows the train-validate-test split data distribution for the multi-classification task. Here, as we also see in the data set as a whole, there is a significant difference between amount of data for each event

Data Set	Non-Event	Event
Train	185	186
Validate	23	24
Test	23	23

Table 2: Category Counts for Binary Classification Data Sets

class. Most notably class Ia may suffer from a small data set and steps may need to be taken in order to ensure predictions are made for that class.

Data Set	Non-Event	Ia	Ib	II
Train	186	27	68	90
Validate	23	3	9	12
Test	23	4	8	11

Table 3: Category Counts for Multi-Class Classification Data Sets

3 Classifier Models

In this section we outline which classifier models we trained in order to complete the binary and multi-class classification tasks. For each model, descriptions of the model along with its advantages and disadvantages are provided. Additionally, the model parameters and results for both the binary and multi-class tasks are discussed and shown.

3.1 Logistic Regression

3.1.1 Model Description

Logistic regression model predicts labels based on $P(Y|\mathbf{x})$.

$$P(Y = 1|X) = \frac{1}{1 + \exp -\beta X}$$

It is a very common model used for binary classification because it's simple and easy to understand and interpret. We used the logistic regression model with a L1 regularization term to first do the binary classification (event or nonevent label).

3.1.2 Advantages

Advantages of this model include, the Logistic model with Lasso(L1) regularization can effectively control coefficients and only keep useful features. We also can rank the importance of features by checking the values of coefficients. The linear model is also easier to interpret results

3.1.3 Disadvantages

Linear model means the decision boundary is linear, so the model might not fit very well on the complex data. When we run the model, we ran into the issue where the model coefficients still do not converge even if we set a higher max iteration number.

3.1.4 Parameters and Model Tuning

We used penalty=l1, C=10, max_iter=1000, solver='saga' as our parameters for this model.

3.1.5 Results

Since we only use the logistic model to do the binary classification (as the first part of our two-stage model), we here only display the results of classifying event/nonevent labels. The event and nonevent labels are balanced in the provided dataset so we don't need to add class weights. The results on binary classification seem good. There are two different labels, 0 is non-event and 1 is event. The accuracy is 87% on validation set and 90% on test set. We also performed cross-validation to ensure to make sure the model is reliable. The average accuracy of 5-fold cross-validation is 87%.

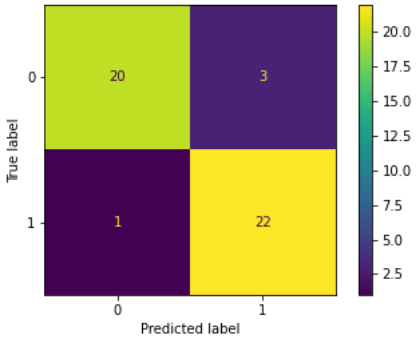

Figure 3: Logistic regression results on validation set

3.2 Random Forest

3.2.1 Model Description

Random Forest is an ensemble method machine learning algorithm which can be used for classification and regression. It uses multiple decision trees to make predictions. The basic idea behind Random Forest is to train a large number of decision trees on random subsets of the data, and then average their predictions to obtain more accurate and stable estimates.

Random Forest is a powerful and widely-used machine learning algorithm that is well-suited for a variety of tasks, especially when working with large datasets. It is generally considered to be one of the most accurate and stable ensemble methods, which makes it a popular choice for many machine learning practitioners.

3.2.2 Advantages

One of the key advantages of Random Forest is its use for both regression and classification tasks, and its ability to handle a large number of features. It is also less prone to overfitting than other algorithms, because it averages the predictions of many different decision trees, which helps to reduce the variance in the predictions.

Another advantage of Random Forest is that it is easy to parallelize, which means that it can be trained on very large datasets using multiple computers. This makes it a popular choice for working with big data.

3.2.3 Disadvantages

Some potential disadvantages of Random Forest include the fact that it can be time-consuming to train, especially on large datasets, and that the model is difficult to interpret, because it is a black box model. This means that it can be hard to understand why a particular prediction was made, which can be a disadvantage in certain applications.

3.2.4 Parameters and Model Tuning

Here, we applied the random state of 42.
In this part, we will discuss two hyperparameters in the training of random forest: n estimator and weights. To determine the best hyperparamter of n estimators, we calculate the accuracy and perplexity accroding to n estimators. Accrdoing to the results in valid set, it seems that the accuracy remains almost the same when n estimator is higher than 20, however, when choosing this hyperparameter of 63, we can have a minimum value 1.24 in perplexity.

When doing multiple classifications, it is very likely that the random forest would be affected by the imbalanced dataset, as it favors in majority of votings. In this way, we applied weighted random forest, to make a higher weight for the rare class. It is also important to look into the confusion matrix for the validation set each time to see how to leverage the mis-labeling predictions. Finally, we chose the weights as "nonevent":1, "Ia": 20, "II": 3.5, "Ib": 2 in our model at last.

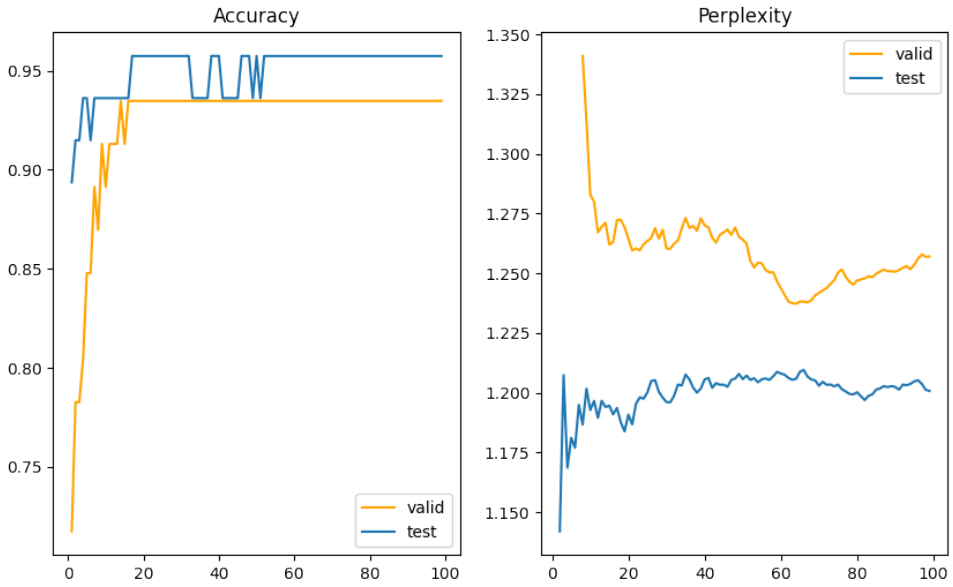

Figure 4: Accuracy and perplexity results on n estimators on binary class

3.2.5 Classification Results

Table 4 summarizes the results of the Random Forest model on the binary and multi-classification tasks. First, let's see the confusion matrix results of binary classification. Then, let's see the confusion matrix results of multiple classification.

It seems that Random Forest classifier works really well in the test set, which is clearly higher than Logistic and other classifier mentioned in this section.

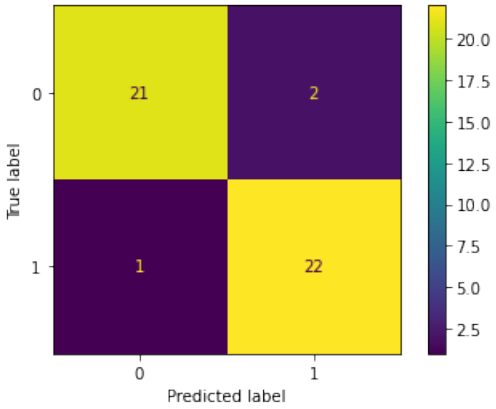

Figure 5: Test Result of Random Forest in Binary Classification

Data	Validation Accuracy	Test Accuracy
Binary Classification	0.935	0.957
Multiple Classification	0.761	0.723

Table 4: Results on Random Forest Classification Tasks

3.3 Logistic Regression + Random Forest

3.3.1 Motivation/Descriptions

We've described two models above, and we thought of creating a two-step model. That is first to predict if the data is an event or not and then given the binary class we apply a random forest model to further decide which type of event label it is. So in this case the first probability is $P(event = 1|X)$ and the second probability is actually conditional probability: $P(class = II,\ Ia\ or\ Ib|event = 1)$.

The generalization of ensemble models is the reason why we choose Random Forest as the second model. We've already found that the label distribution in event is not very balanced so we need to prevent the model from always predicting the most frequent labels.

3.3.2 Advantages

The main benefit of employing the two-step model is that it successfully avoids producing two opposite results on binary and multi-class classification. Sometimes, the probability of non-event could be less than 0.5 and is still classified as non-event. If we make prediction first on binary and then do the multiclass classification, we will get consistent predictions.
Based on our experiments results, the two-step model performs well, even better than linear models.

3.3.3 Disadvantages

The two-step method also brings more complexity in calculating the accuracy. We have to first compute the number of correctly predicted labels and separate them as non-event and event. After that, we calculate the precision in event labels and finally sum up all correct labels.
Another issue is that if the accuracy in binary classification task is low, it will significantly affect the performance of the latter model. This highlights the importance of good preprocessing - where we made an effort to have good feature selection with scaled data in order to have a good performance on the first step.

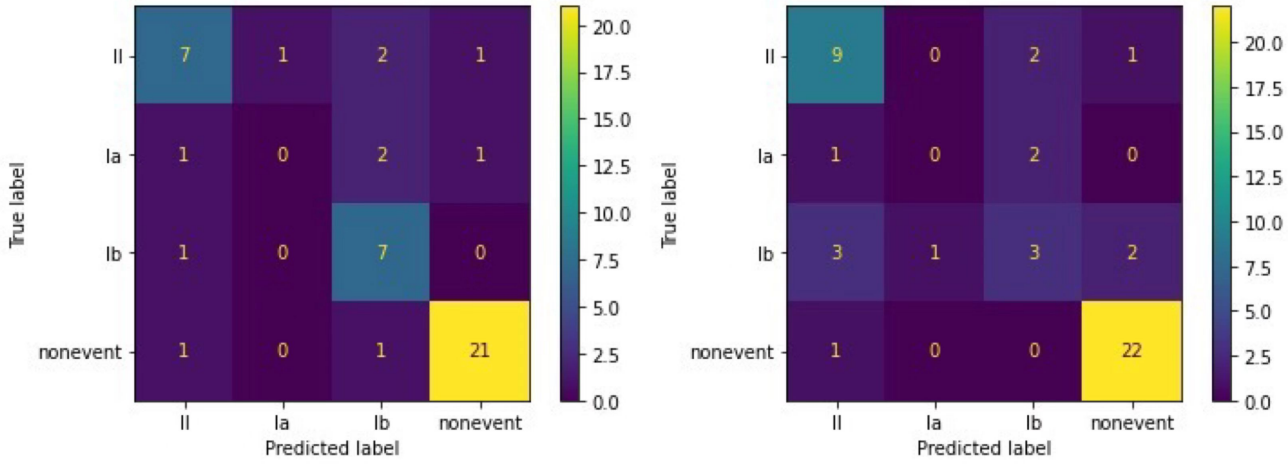

Figure 6: Valid, Test Result of Random Forest in Multiple Classification. left side is the valid set

3.3.4 Results

Figure 7 showcases the structure of this combination model.

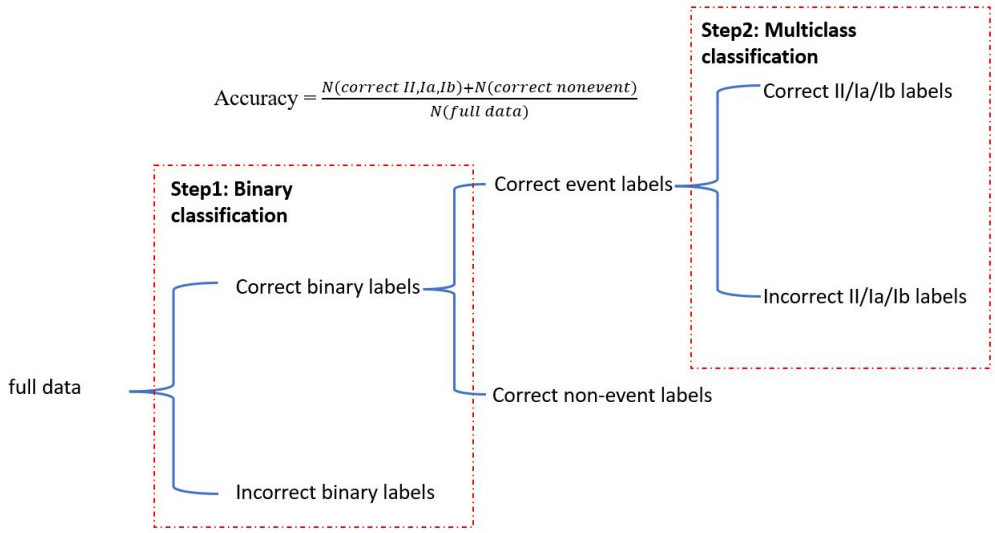

Figure 7: Two-step model workflow and accuracy calculation

Table 5: Two steps details

Total test data	First step	Second step
47	23 (correct event labels)	11 correct event-specific labels 12 incorrect event-specific labels
	21 (correct nonevent labels)	
	3 (wrong labels)	

The accuracy of Logistic Regression+Random Forest is about 68.1% on test set as calculated in Table 5. It seems very good comparing with some models like Naive Bayes (see Section 3.4). After cross-validation, we found the average accuracy is 2 percent below score of multi-class random forest. We ultimately decided to use Random Forest do the 4-class prediction directly, but this approach had good results.

3.4 Naive Bayes

The Naive Bayes classifier uses Bayes's theorem to classify data. Notably, this algorithm has two primary assumptions:

1. Each feature is independent
2. Each feature has equal importance

It is based on the Bayes Theorem: $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$ which allows us to find the probability of A given B. In our classification tasks this results in finding class A given a set of features B.

3.4.1 Advantages

There are numerous advantages to using Naive Bayes as a classifier. It's a simple implementation which provides interpretable results. Particularly, given its assumption that each feature is independent, it does not fall under the "Curse of Dimensionality" that other machine learning methods do as feature number increases - this means we do not necessarily need to be very concerned about feature reduction for this model.

With its assumptions, Naive Bayes introduces some amount of bias but reduces variance (bias-variance trade-off) which results in a classifier that can perform well in multiple scenarios.

Additionally, Naive Bayes works well with small data sets, given that our data set is only 464 points large if the entire set is used, or 371 points large if using the training split, this feature is especially helpful.

3.4.2 Disadvantages

The simplicity and assumptions of Naive Bayes also has drawbacks. Namely, if the classes that the model is classifying are imbalanced, this can result in skewed probabilities and reduce its accuracy. Our data set for the binary classification task is equal between the "non-event" and "event" classes, however this balanced data does not hold when we have our multi-class classification set. There, our "event" class which used to be one half of the set, is split into 3 unequal parts.

Ultimately, not all of our features are independent within our data - however Naive Bayes is known to still perform well when this assumption does not hold.

3.4.3 Data Selection

We compared the results of Naive Bayes models trained on the optimal SelectKBest and PCA features to determine which features selection method should be applied for optimized performance.

To determine the optimal number of features when using SelectKBest, we compared the training, validation, and test accuracy as the number of features increased. Figure 2 shows how the train, validate, and test sets all perform at different numbers of included features when using SelectKBest. At 55 features, the Naive Bayes model reached 87% accuracy on the test set extracted from the training data available and had an average accuracy of 81% on the cross-validation sets.

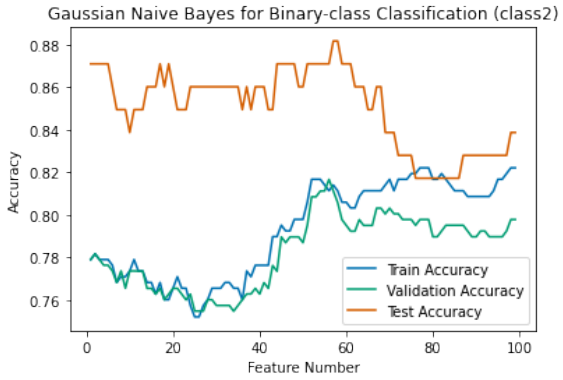

Figure 8: class2 Train, Validate, Test Set Accuracy by Feature Number

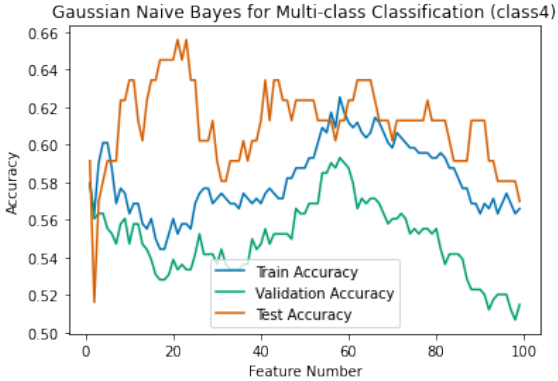

Figure 9: class4 Train, Validate, Test Set Accuracy by Feature Number

Searching for the number of features with SelectKBest from sklearn which provided the best cross-validation score for the multi-class classification task with Naive Bayes resulted in selecting a total of 57 features. This leaves us with an average cross-validation accuracy of 58% and test accuracy of 60%.

In sections 3.4.5 and 3.4.6 we compare the performance of models trained with these selected features versus the 12 principal components.

The final models of the Naive Bayes models had the following number of features or components:

	SelectKBest Features	**PCA**
Binary	55	12
Multi	57	12

Table 6: Feature Number by Classification Task

3.4.4 Parameters and Model Tuning

After determining the optimal data to use for the models, we then used GridSearchCV to determine the best hyperparameters for the model. Naive Bayes has two parameters that can be set: priors and var_smoothing, different var_smoothing values were explored using GridSearchCV to optimize model performance.

The optimized models of the Naive Bayes models had the following parameters:

	SelectKBest	PCA
Binary var_smoothing	0.03511191	0.0432876
Multi var_smoothing	0.12328467	0.3511191

Table 7: Best parameters by model task and data set

3.4.5 Binary Classification Results

In the table below the null accuracy, the accuracy that is achieved by predicting the most common class, is listed as the first row in the table. Before data normalization and feature selection, the Gaussian Naive Bayes Classifier on the binary classification task achieved an accuracy of 0.80 on the cross-validation set. Small improvements can be seen after using KBest features or PCA for feature selection of the model, but no great improvements are seen. Overall the model performs significantly better than the dummy classifier.

The model using PCA has a better accuracy on the cross-validation set and slightly less accurate results on the test set compared to the model using KBest features. This indicates that it may be better at generalizing on unseen data points compared to the KBest features model. This may be in part due to Naive Bayes' set up - we know from our data exploration that there are many correlated features. 12 principal components may reduce the extra features more so than KBest, resulting in a Naive Bayes that doesn't have as much performance degredation.

Data	Training Accuracy	Validation Accuracy	Test Accuracy
Null Accuracy	0.5	-	0.5
All Features	0.82	0.80	0.85
KBest Features	0.81	0.81	0.87
PCA Features	0.84	0.83	0.85

Table 8: Naive Bayes Binary Classification Results

The confusion matrices show that both models are very similar in accuracy, although on the test set the KBest model performs better. The model using KBest features predicted 54 events and 39 nonevents while the PCA model predicted 48 events and 45 nonevents - a closer to 50-50 split. The KBest model seemed to be more skewed towards an event bias.

The final binary-class model chose was using 12 principal components and a var_smoothing parameter of 0.0432876.

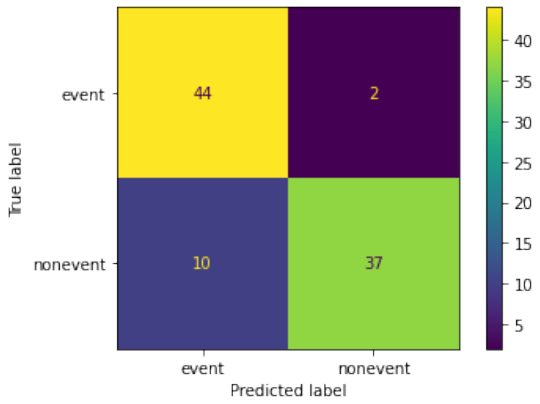

Figure 10: Confusion Matrix for class2 Binary Classification with KFoldBest

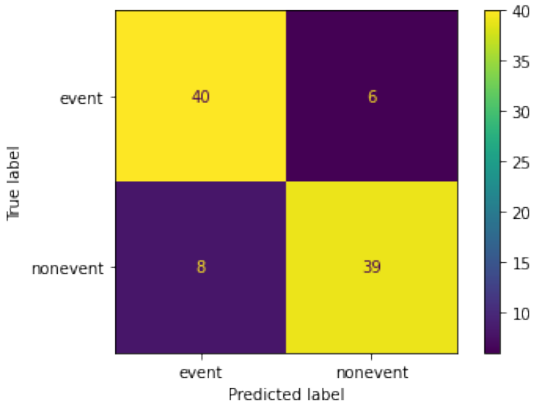

Figure 11: Confusion Matrix for class2 Binary Classification with PCA

3.4.6 Multi-class Classification Results

Before data normalization and feature selection, the Gaussian Naive Bayes Classifier on the multi-class classification task achieved an accuracy of 0.5.

In the table below the null accuracy, the accuracy that is achieved by predicting the most common class, is listed as the first row in the table. We can see that before performing any feature selection or data processing, the performance of the classifier is only minutely better than that of a dummy classifier. Overall, the model using PCA performed best on the cross-validation accuracy, indicating that it has better capacity to generalize.

Data	Training Accuracy	Validation Accuracy	Test Accuracy
Null Accuracy	0.49	-	0.50
All Features	0.55	0.51	0.57
KBest Features	0.60	0.58	0.62
PCA Features	0.67	0.64	0.59

Table 9: Naive Bayes Multi-Class Classification Results

The confusion matrices show the slight improvement of the PCA model versus the KBest model. However, the model using KBest features does seemed to have a better ability at classifying event types II and Ib than the model using PCA (this can be in part due to its preference for the event class which was evident in binary classification confusion matries). Both models are unable to correctly classify event type Ia, as predicted earlier given the small sample size of this event class.

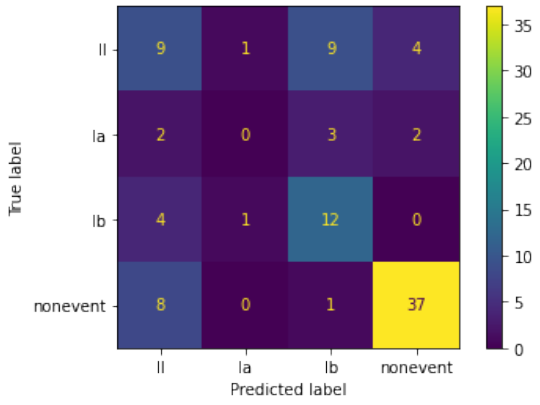

Figure 12: Confusion Matrix for class4 NB Classification using KFoldBest

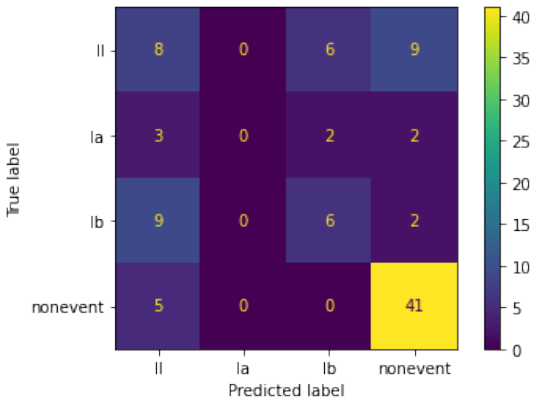

Figure 13: Confusion Matrix for class4 Binary Classification with PCA

The final multi-class model chosen used 12 principal components and a var_smoothing parameter of 0.3511191.

3.5 KNN

K-NN, or K-Nearest Neighbors, is an algorithm that searches for K points closest to a given point and estimates the probability of the point being part of a certain class given the identified nearest "neighbors".

3.5.1 Advantages

KNN is another simple model to implement, it simply considers the data it has and uses that data to make predictions on a new point. While this characteristic allows KNN to be changeable and flexible

enough to receive new data, in our case we do not have any new data to give our model and thus are unable to exploit this feature. Unlike with Naive Bayes or linear regression models, there aren't any assumptions about the data set.

3.5.2 Disadvantages

The core component to KNN is its calculation of distance from a point. This means that when there is a high level of dimensionality, or a large number of points to consider, this can be a time consuming and costly. Additionally, the classifications of points to a class depend on having neighbors, with our skewed data it is likely that we will run into issues classifying points to event "Ia" given its low frequency in the training set.

The performance of KNN is also highly dependent on the chosen value of k. With a value of k that is too low, over-fitting occurs, however if the value of k is too high, the model is under-fit.

3.5.3 Parameters and Model Tuning

The value of K is the most important parameter with the KNN algorithm. To determine the best value of K, the accuracy of the model was considered as the value of k changed. The figures below show how validation accuracy changes as the value of k increases. Using this information, it was determined that the optimal value of k was 9 for the binary classifier and was 29 for the multi-class classifier. The same procedure was performed to test the performance of models using the 12 PCA components.

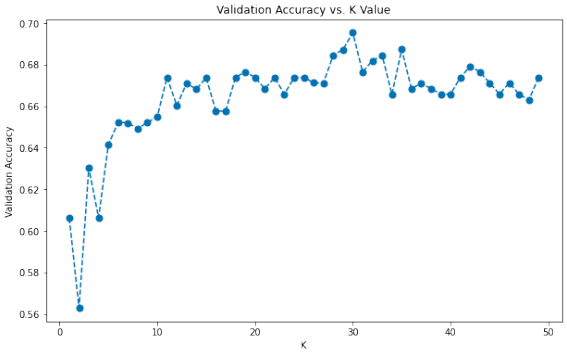

Figure 14: Validation Accuracy as K changes for Multi-Class Classification

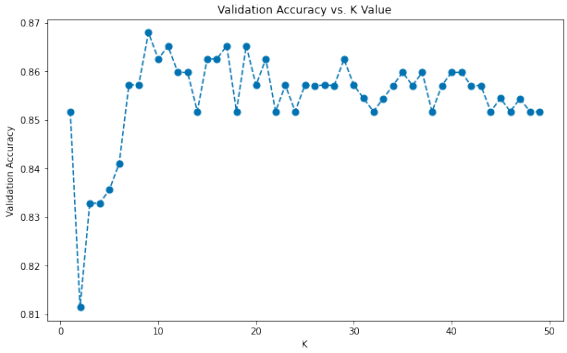

Figure 15: Validation Accuracy as K Increases for Binary Classification

Plotting how the validation accuracy changes as K increased showed that K = 8 would be the value of K to maximize the model's accuracy, however this value of K is an even number and thus could run into issues of how to break ties, after further exploration it was found that setting the value of K to 9 would not significantly change the resulting validation accuracy or accuracy on test or train data and would have the benefit of being able to easily break ties if encountered.

After determining the optimal K value, GridSearchCV was performed to find the optimal parameters for each model. Those parameters are summarized below. For the most part the optimal parameters were determined by the k value while other parameters remained unchanged. We see, as one would expect, that the more classes we are classifying a point into, the more neighbors we must consider.

		SelectKBest	PCA	High Variance (>0.01)
Binary	*K*	9	13	15
	weights	distance	distance	distance
	algorithm	tree_ball	ball_tree	ball_tree
Multi	*K*	29	23	27
	weights	distance	uniform	distance
	algorithm	ball_tree	ball_tree	ball_tree

Table 10: Best Parameters by KNN Model

3.5.4 Binary Classification Results

For the KNN models, we considered 3 different features selection options:

1. PCA
2. KBest
3. Features with less than 0.01 variances removed

In all instances of KNN's optimized models,we see that we can achieve 100% accuracy on the training set which raised suspicions of over-fitting. Beyond this, each of the models had very similar results. Ultimately, the model that used 55 KBest features had the best performance on the cross-validation and test set and so it was determined to be the best KNN model to use for this task.

Data	Training Accuracy	Validation Accuracy	Test Accuracy
Null Accuracy	0.49	-	0.50
All Features	0.77	0.79	0.88
KBest Features	1.0	0.87	0.90
PCA Features	1.0	0.86	0.88
High Variance Features	1.0	0.85	0.89

Table 11: KNN Binary Classification Results

The confusion matrices below show the classification results for the 3 different models. We can see that the KBestFold model has the fewest errors, although the other two models are close behind in performance.

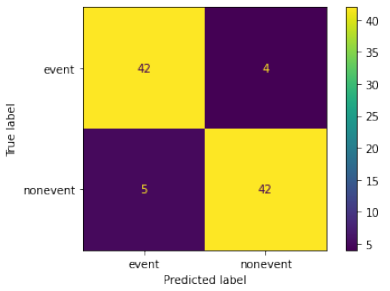

Figure 16: Confusion Matrix for Binary Class KNN Classifier with KBestFold Features

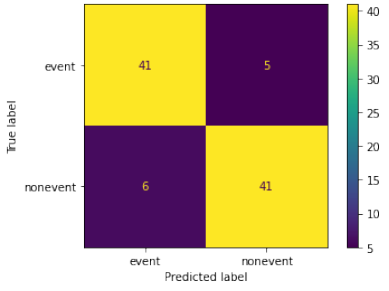

Figure 17: Confusion Matrix for Binary Class KNN Classifier with 12 Principal Components

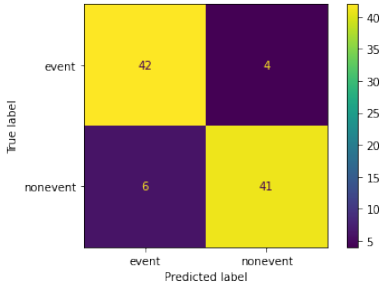

Figure 18: Confusion Matrix for Binary Class KNN Classifier with High Variance Features

The final KNN Binary classification model had the following parameters:

Parameters of final KNN model	
Data	55 Features
n_neighbors	9
weight	distance
algorithm	ball_tree

Table 12: Optimal Binary KNN Model Parameters

3.5.5 Multi-Class Classification Results

As in Section 3.5.4, for the KNN models, we considered the same 3 features selection options.

The KBest features model had the highest validation accuracy between all three considered models. It appears that optimizing the model resulted in extremely high training accuracy performance on the KBest and High Variance models, but did not results in similar performance on the model using PCA.

Data	Training Accuracy	Validation Accuracy	Test Accuracy
Null Accuracy	0.49	-	0.50
KBest Features	1.0	0.69	0.70
PCA Features	0.71	0.67	0.71
High Variance Features	1.0	0.67	0.66

Table 13: KNN Multi-Class Classification Results

In the confusion matrices for the models, their similar performances is evident. Each model seems to predict a similar number of the non-event class (accurately and inaccurately) and none are able to predict occurrences of class Ia. The models' weights parameter used "distance" in all instances but one (see Table 8). With distance as the weight, closer neighbors to a point have more influence than more distant neighbors - essential if we need to consider between 23 and 29 neighbors while there are only 34 instances of class Ia in the entire data set. However this didn't seem to provide much support indicating we either need additional data or a different model would be optimal for this task. Figure 19 shows the KBest classification results where we see it is slightly better at predicting class II and Ib than the other models.

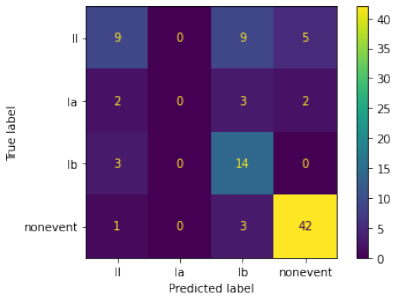

Figure 19: Confusion Matrix for Multi-Class KNN with KBestFold Features

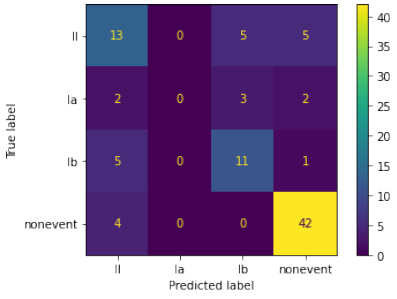

Figure 20: Confusion Matrix for Multi-Class KNN with 12 Principal Components

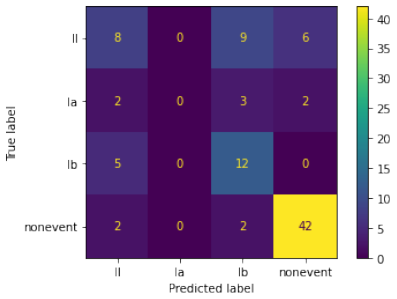

Figure 21: Confusion Matrix for Multi-Class KNN with High Variance Features

The final KNN multi-class classification model had the following parameters:

Parameters of final KNN model	
Data	57 Features
n_neighbors	29
weights	distance
algorithm	ball_tree

Table 14: Optimal KNN Multi-Class Parameters

4 Conclusion

After testing four different machine learning models with a range of supervised approaches, we found that our Random Forest model performed the best for the binary and multi-class classification tasks, with an accuracy of 93.5% and 95.7% validation and test dataset in the binary classification and accuracy of 76.1% and 72.3% in the multiclass classification task. When tested on a previous unseen test set of 965 data points in the challenge task, our model has a slightly lower score of 86.2% (binary score) and 70.1%(multiclass score).

Task	Validation Accuracy	Test Accuracy(part of training)	True Test
Binary Classification	0.935	0.957	0.862
Multiple Classification	0.761	0.723	0.701

We also examined the importance of preprocessing methods in our method in the hidden dataset. It seems that without feature selection(choosing all 100 features), we can only achieved 86.0%(-0.02%) in binary and 68.6%(-0.15%) in multi-classification accuracy. Without data normalization, we got 86.2%(-0.00%) in binary and 69.1%(-0.10%) in multi-classification accuracy. In this sense, it appears that multi-classification benefits from feature selection and normalization. It seems that binary classification result is not affected by the normalization in Random Forest, as one would expect given the nature of Random Forest. The gap between development and test data accuracy is caused by the limited amount of training

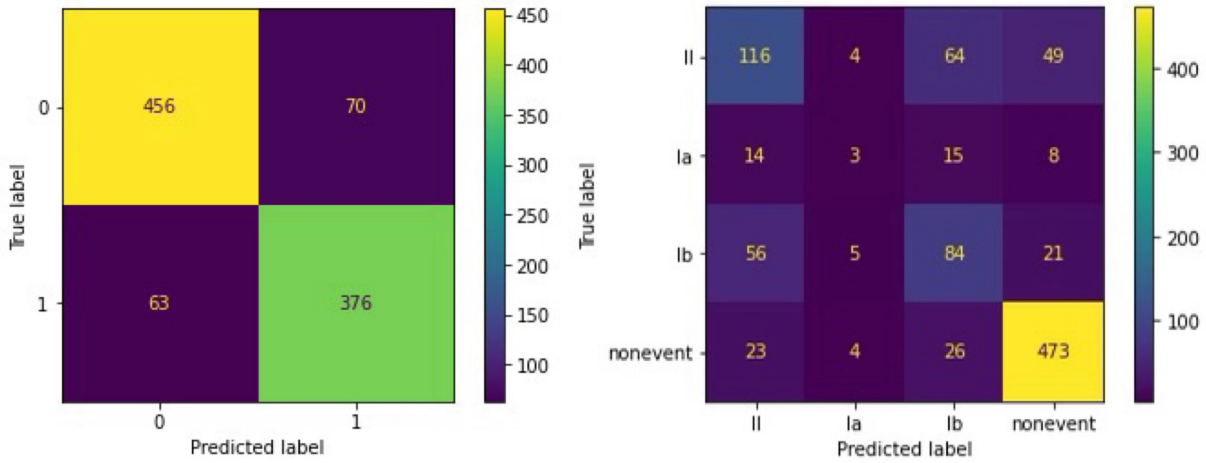

Figure 22: Confusion Matrix Result of Random Forest in Binary and Multi Classification on hidden test

data. We only have 464 pieces of data for training, and the predictions are conducted on a dataset size of 965. As we learned from the course, the loss of training data is always smaller than test data. Therefore, it's possible (and even likely) that the model performs worse on a new dataset.

Comparing those different models, we notice the advantages of Random Forest, such as tree structure and ensemble learning. Random Forest outperforms linear models, Naive Bayes and KNN. According to our results, Random Forest also has a good multi-class generalization.

Many of the classifiers experienced difficulty predicting the smallest event class (Ia) in the multi-class classification task due to the small set of data available for this class. This highlights the importance of

having well-rounded data, a fact that is easier said than done. Frequently, there is an absence of good data. So, model selection and parameters must be chosen to optimize a model's ability to generalize given the data available. This is a common problem to have in the real world and it was beneficial to experience it in a project setting. The Random Forest model likely achieved better results partly due to it's weighted classes in the multi-class setting, encouraging it to predict class Ia even when it might have ignored the class without. Not all models tested were manipulated in this way and their accuracy suffered as a result. In the future, performing oversampling is a data augmentation technique we should consider for models where we cannot manipulate the weights at all or enough to see predictions of the smaller class.

Across the models, different feature selections were attempted, including PCA, KBest, and removing low variance features. However, there was not one tried and true method across each model that resulted in the best results. This demonstrates the importance of using multiple different approaches to find the best result.

5 Self-Evaluation

5.1 Project Evaluation

Project Self-Evaluation: 4

For this class project, as a group we utilised a variety of methods presented in the course, while also challenging ourselves to think outside of the box and be creative looking for solutions to this classification task. For our data processing, we utilized unsupervised methods (PCA), along with other data processing approaches to determine which features to use for the models. Deeper data exploration could have been undertaken though to have a better understanding of the data set as a whole and have improved intuitions on the relationships between each variable and to help motivate our choices of machine learning models.

Within our report we took care to communicate relevant results as well as interesting findings or experiences. We looked to use different types of supervised machine learning methods which also challenged us to use different data wrangling approaches to optimize performance. Ultimately our models performance were aligned with many of our classmates. We could have tried a few other different approaches such as SVM or other model stacking techniques like other classmates, however we restricted ourselves to the 5 outlined in this report.

As a result of these successes and a few areas of improvement, we self-evaluate our project to have a score of 4.

5.2 Group Evaluation

Group Self-Evaluation: 5

Our group met regularly and kept one another up to data with frequent project related communication. Everyone was on the same page and fulfilled each of their commitments to the project, ensuring that each individual shared the project workload equally. Discussions on model results and model choices also furthered group members understanding of machine learning approaches discussed in class.

Overall we had a positive group experience that made this project an enjoyable learning opportunity and deepened our understanding of the topics covered in this course. As a result of this we self-evaluate our group work to have a score of 5.

6 Appendix

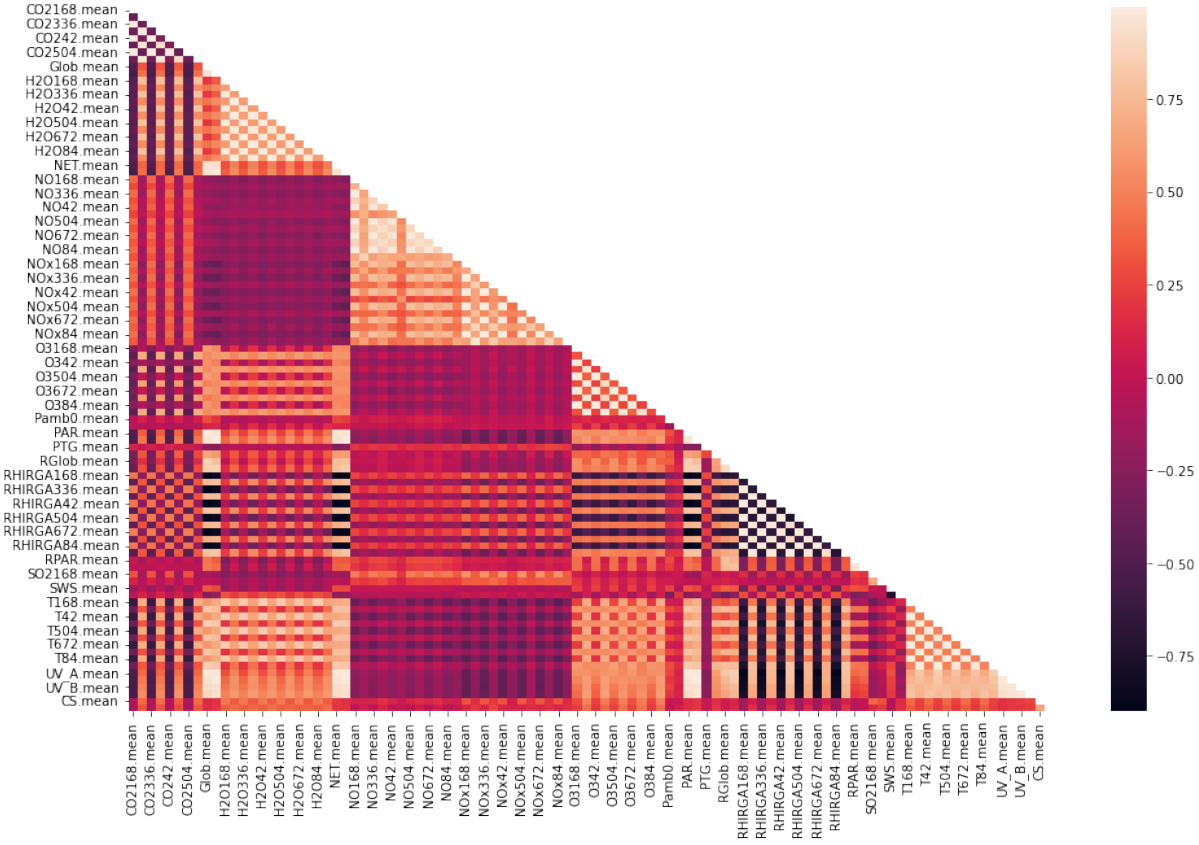

Figure 23: Correlation Matrix of Features