Introduction to Machine Learning Project Report

Iira Häkkinen and Tuomo Artama

17 December, 2022

Grade suggestion: 4

1. Introduction

In this report we present an R implementation of a multi-class classifier to predict if different types of new particle formation (NPF) events occur on a given day. An NPF event happens, when smaller particles start to form larger particles that affect cloud formation (Kerminen et al. 2018). There are three types of NPF events: Ia, Ib and II (Kerminen et al. 2018). There is also a possibility that an NPF event does not happen on a given day.

The objective of this report is to present a classifier to solve the aforesaid task and analyze the results critically. We chose as our classifier a multi-class classifier that uses logistic regression with RIDGE regularization. The classifier was trained on a dataset containing various atmospheric parameters (such as temperature, CO2 measurements, etc.) on multiple days throughout the years 2000-2011.

The contents of the report are as follows. First, we discuss the methods we used to choose the classification model, explain the concept of linear regression briefly and go through what kind of feature selection was done. Secondly, we analyze the results of our classifier. Lastly, we conclude the analysis, discuss the results and provide ideas what could be done to improve the classification results.

2. Methods Used

Firstly, it is important to note that since the beginning of this project, we were aware of how limited time we have available for this project. This time limitation had a large effect on our choices throughout the project. This time limitation was kept in mind, so that we could perform as well as we can with the time that we had. Especially, it is the main reason behind our model selection. We chose a simple model, so that we would spend less time debugging the model, and more time on analyzing the results and writing a good report.

When choosing the classifier, we recollected what kind of machine learning classifiers we are aware of, and especially how well we know their implementations in R. We chose the linear regression model as our classifier, but we also talked over the Naive Bayes classification as well as the k-nearest neighbour classification. We were not sure, if the Naive Bayes assumption is valid with this dataset. We also concluded that we do not know anything about implementing the kNN-classifier in R. We ended up choosing linear regression, because we were familiar with it's implementation in R and it is straightforward to use.

2.1 Linear regression

Linear regression is a widely used approach for supervised learning (James et al. 2013). It predicts a quantitative response Y on the predictors $X = X_1, X_2...X_n$, where $n \in \mathbb{Z}^+$ is the number of predictors. The underlying assumption is that there is approximately linear relationship between the predicted variable Y and the predictor variables X (James et al. 2013). More formally, linear regression can be defined as

$$Y = \alpha + \beta X + \epsilon \tag{1}$$

where α is the intercept term and $\beta = \beta_1, \beta_2, ..., \beta_n$ are the slope terms corresponding to each predictor $X_1, X_2, ..., X_n$. The ϵ term is a mean-zero random error term (James et al. 2013), which inclues the error of the model assumptions. In reality, the true relationship between Y and X might not be linear or there might be more affecting covariates than $X = X_1, X_2, ..., X_n$ (James et al. 2013).

The terms α and $\beta = \beta_1, \beta_2, ..., \beta_n$ are called model coefficients (James et al. 2013). They are, in practice, unknown and are estimated by linear regression. The coefficients assign weights for the predictors X and after the coefficients have been estimated, we can get a prediction for the value of Y. In our project, the predictors $X = X_1, X_2, ..., X_n$ are the various atmospheric parameters from the dataset (such as CO2168.mean, CO2168.std, etc.).

In this project, we use linear regression to predict the probabilities of NPF event Ia, Ib or II occurring or an NPF even not occurring at all. To be more exact, we calculate with linear regression the probabilities Y that *class4* is either Ia, Ib, II or *nonevent*. The class is then chosen according to which class has the highest probability of occurring. This way we get to use linear regression in a quantitative problem, even though linear regression is generally a tool for predicting qualitative problems (James et al. 2013).

Linear regression might not be the the optimal classifier for this task, but provides better values than just guessing the class of a given day.

2.2 Data Analysis

Our data analysis was mainly done by physically examining the data through Excel. Our understanding of the data, as well as the chosen linear regression model, relied also on our previous knowledge of it throughout the exercise sets of the Introduction to Machine Learning course (Puolamäki 2022).

The most evident feature selection was done after eyeing the data through Excel. We excluded the columns *date*, *id* and *partlybad*, because they give no actual information to our regression model. The columns *date* and *id* are self-explanatory, and the column *partlybad* was removed because it is always FALSE, and therefore offers no notable information.

After that, tried analyzing the data more thoroughly with R. We examined the means of both the mean values as well as the standard deviation values of the atmospheric parameters and plotted them. This did not, however, give us any apparent information of the behaviour of the data set. We also examined the correlation between some of the atmospheric parameters, as was done in the exercise set one (Puolamäki 2022). We saw that there is a linear correlation between the mean values of the same measured entity, for example the mean carbon dioxide measurements from different heights This example is displayed in figure 1 for CO2 measurements at heights $1.68m$, $3.36m$, $0.42m$ and $5.04m$).

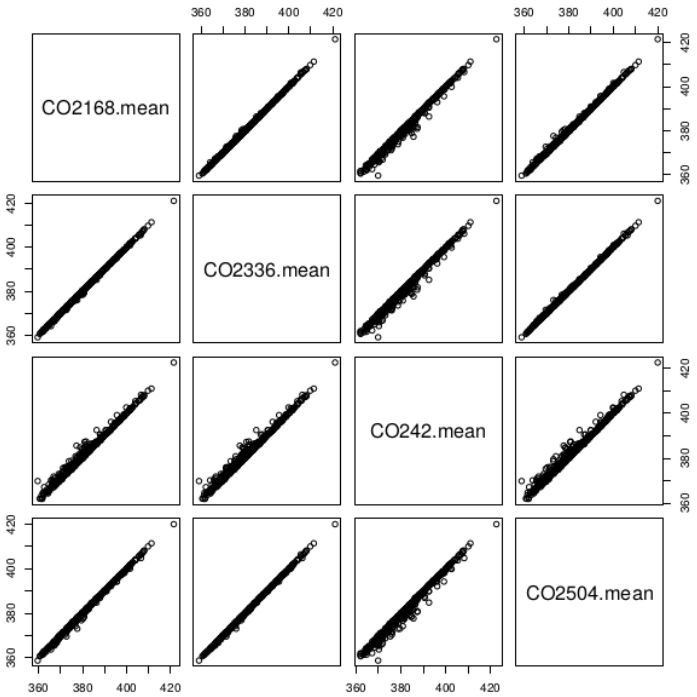

Figure 1: Colinearity of the measurements of the variable CO2 at different heights.

This suggests that the measurements for the same parameter (such as CO2 in this example) form colinear groups in the data. This means, that the measurements inside one colinear group give out no additional information for a classification model and, thus, only one measurement from a colinear group can be chosen as a covariate for the model without losing any data (James et al. 2013). The linear regression implementation that we chose, can do this automatically and, therefore, we did not have to do any data manipulation conserning the colinear groups.

2.3 Implementation

Our multi-class classifier is implemented in R using logistic regression. The classifier is implemented using the *glmnet* regressor provided by the library *glmnet*. The implementation of the model can be seen in the code block below (for full implementation, see the appendix of this report). We chose to use both RIDGE and LASSO regularization in a 50/50 relationship with our logistic regression model. This can be done with *glmnet* by setting the *alpha* parameter value to 0.5. The parameter *alpha* controls the choice of LASSO or RIDGE (RIDGE being 0 and LASSO 1) and *lambda* controls the volume of the regularization to the coefficients (β in equation 1) of the model. We chose regularization over no regularization, because by testing both, we noticed that the accuracy was better with regularization.

```
model = glmnet(class4 ~ ., npf_train, family = "multinomial", alpha = 0.5, lambda = 0.012)
```

The choice of both RIDGE and LASSO as the regularization method was done by simply trying different values by hand for both RIDGE and LASSO regularization and these two together. By manually changing

the value of the *alpha* parameter, we noticed that having a 50/50 ratio of LASSO and RIDGE gave us better validation and cross-validation accuracies, than having only LASSO or only RIDGE. We did not try different combinations of *alpha* and *labda* exhaustively, so it is possible that the model can be improved in this respect.

The choice of the *lambda* value in the *glmnet* function was done by examining how the validation accuracy and cross-validation accuracy behave as a function of lambda. We calculated these accuracies iteratively, during each iteration we trained 20 different models with different *lambda* values. All models had the same *alpha* value, 0.5. First, we had an educated guess on what scale the *lambda* value should have. We first set a somewhat broad interval of $]0, 1]$ for *lambda* and examined how the accuracies behaved (see figure 2). It is evident, that the best accuracies are achieved with a lambda value that is close to 0.

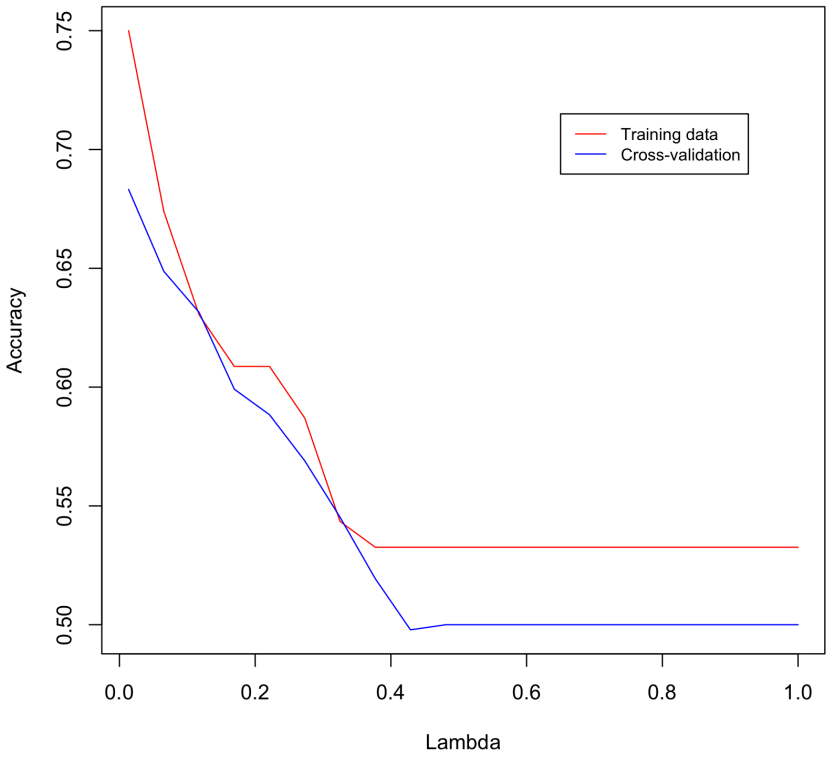

Figure 2: Validation accuracy and 5-fold cross-validation accuracy as a function of lambda in an interval from 0 to 1.

We then examined iteratively two smaller intervals of 20 lambda values, first the interval of $[0.005, 0.040]$ and then the interval of $[0.004, 0.018]$. Our experiments showed, that with *lambda* value of approximately 0.012, both the validation accuracy as well as the 5-fold cross-validation accuracy peaked. The value $\lambda = 0.012$ was then chosen as our final value for the regression model. Figure 3 illustrates the behaviour if the two accuracy values as a function of lambda on the second interval of $[0.005, 0.040]$.

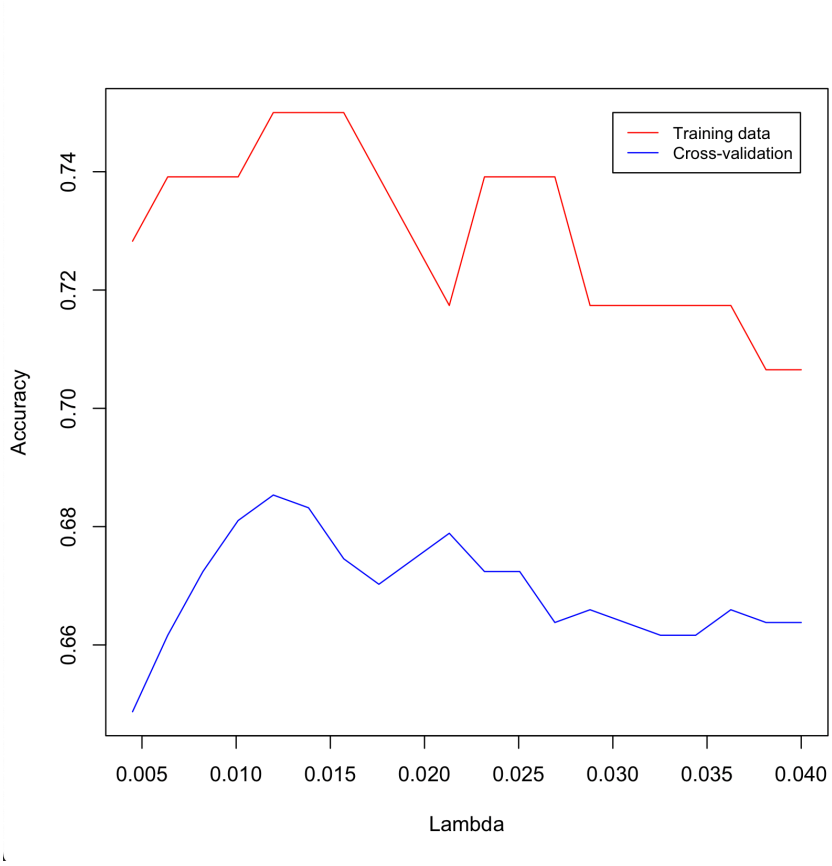

Figure 3: Validation accuracy and 5-fold cross-validation accuracy as a function of lambda in an interval from 0.005 to 0.040.

An even better combination of the model parameters *alpha* and *lambda* could have been acquired with a more automated method. Especially, our method for choosing *alpha* is not as convincing as our method of choosing *lambda* for the given *alpha*. The main reason we did not loop through many different combinations of these model parameters was that it would have been time-consuming. Thus, our choice of *alpha* may be a fault in our model, since we do not have solid evidence backing this choice.

3. Results

The results show that our classifier appears to be working at least better than a 50/50-classifier.

- cross-validation and cross-validation error
- accuracy and perplexity
- how the regularization (LASSO, RIDGE) affects the resulting accuracy and perplexity
- other types of models and how they perform in respect to each other (**if we have time**)
  - choose the best out of these as our classifier

4. Discussion

Overall, this linear regression model implemented with *glmnet* in R gives fairly good results for the classification task of predicting if different types of new particle formation events occur on a given day. There are

still possible ways to refine the implementation so that the accuracy and cross-validation accuracy would be even better.

Our analysis of the results gives an overview of the goodness of our model, but more estimates could have been calculated. For example, we did not calculate the perplexity of the model on the validation data. Additionally, the individual accuracies of the predicted classes Ia, Ib, II and nonevent could have also been calculated and analyzed.

We would have liked to do comparison on different classification models, but from pretty far on saw that this might not be possible due to the limited time window for completing this project. By implementing a few classifiers more, we could have compared the accuracies of these classifiers and chosen the best one out of these. The pros and cons of different classifiers could also have been discussed.

6. Grading

When considering all aspects of this project, we think we did rather well. We recognize, that our time limitations have affected the scale of this project. We did not have time to analyze our model and the results as widely as may have been possible, but the analysis that we made was done thoroughly. So, we think that all that is expalained on this report, has been done rather well. Our classifier may not be the most sophisticated, but it works sufficiently well (and the goodness of the classifier also does not affect the grading). We have provided answers for all the required questions from the term project description in this report.

This report shows that we have a good understanding on the topics of the course. We understand the perks as well as the limitations of our chosen classification method, and we have analyzed the results critically. The parts we did not have time for have also been mentioned to show that our understanding of the task at hand does not limit only to the contents of this report. The report has been written in a clear manner and it follows the scientific style of writing.

Our suggestion for our grade is 4. We stuck to our schedule really well and were fully aware of our limitations, so that we could output a report that is constructed clearly and contains proper analysis of our choises. Because we can think of many things that could have yet been done, we are not sure if this project if worthy of the grade 5. When looking at the grading instructions for the grade 3, we think that we have in some parts performed better than the descriptions there.

References

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An Introduction to Statistical Learning*. Vol. 112. Springer.

Kerminen, Veli-Matti, Xuemeng Chen, Ville Vakkari, Tuukka Petäjä, Markku Kulmala, and Federico Bianchi. 2018. "Atmospheric New Particle Formation and Growth: Review of Field Observations." *Environmental Research Letters* 13 (10): 103003.

Puolamäki, Kai. 2022. "Solutions for Exercise Set 1."

Appendix

The R implementation of our multi-class classifier is provided here.

```
options(error = function() traceback(3))

library(glmnet)
library(glmnetUtils)

npf <- read.csv("npf_train.csv")
npf_test <- read.csv("npf_test_hidden.csv")

# Remove columns "id", "date" and "partlybad" and set dates as rownames
rownames(npf) <- npf[, "date"]
npf <- npf[, -c((1:2), 4)]


# Some helpful functions
accuracy <- function(real, pred) {
    sum(as.numeric(real == pred)) / length(pred)
}

# Create a linear model
model <- glmnet(class4 ~ ., npf, family = "multinomial", alpha = 0, lambda = 0.01)

coefficients(model)

# Create variable class2 where the value is "event" if there was an NPF event
# and "nonevent" otherwise.
npf$class2 <- factor("event", levels = c("nonevent", "event"))
npf$class2[npf$class4 == "nonevent"] <- "nonevent"

# Predict class4 on training data
predicted <- predict(model, newdata = npf, type = "response")[,,1]

predicted_class4 <- c()
predicted_class2 <- c()
predicted_probs <- c()
for (i in 1:length(predicted[, 1])) {
    pred_class <- colnames(predicted)[which.max(predicted[i, ])]
    predicted_class4 <- c(predicted_class4, pred_class)
    event_prob <- 1 - predicted[i, 4]
    predicted_probs <- c(predicted_probs, event_prob)
    predicted_class2 <- c(predicted_class2, c("nonevent", "event")[1 + (event_prob > 0.5)])
}

acc <- accuracy(npf$class4, predicted_class4)

cat("Multiclass accuracy: ")
cat(acc)
cat("\n")


binary_acc <- accuracy(npf$class2, predicted_class2)
```

```r
cat("Binary accuracy: ")
cat(binary_acc)
cat("\n")

# Predict class4 on test data
predicted <- predict(model, newdata = npf_test, type = "response")[,,1]

predicted_class4 <- c()
predicted_class2 <- c()
predicted_probs <- c()
for (i in 1:length(predicted[, 1])) {
    pred_class <- colnames(predicted)[which.max(predicted[i, ])]
    predicted_class4 <- c(predicted_class4, pred_class)
    event_prob <- 1 - predicted[i, 4]
    predicted_probs <- c(predicted_probs, event_prob)
    predicted_class2 <- c(predicted_class2, c("nonevent", "event")[1 + (event_prob > 0.5)])
}


write.csv(data.frame(class4=predicted_class4, p=predicted_probs), "answers.csv", row.names=F, quote=F)
```