IML-project, final report

Fanni Franssila , Outi Savolainen, Sini Suihkonen

December, 2021

Data analysis

We opened the datafile in excel in order to obtain our first idea about how the values look. We discussed the importance of the columns with standard deviation. We set the values of the date column as row names and removed the columns partlybad, id and date. We also discarded the standard deviation columns. After these changes, we produced a correlation matrix in order to see how the different mean values are correlated.

Based on the correlation matrix and discussion about how the measured values in different heights might affect the decision about the class, we decided to choose only features from specific heights, if height was given. We tested first the given data with multiple initial modifications: only mean values, and low (42 meters) and high (504 meters) mean values. There was no big difference in the accuracy received from these different tries and we decided to choose the combination of low and high values.

Machine learning approaches

From the very beginning, we decided to go for a multiclass classifier, and so we started the project by searching for the best multiclass classifier for the data. We tested three different classifiers, which were k-NN, SVM and random forest classifiers. Each of the classifiers was trained and tested with the modified data sets. For the accuracy tests, we used a training set of 92 data points and a test set of 366 data points. The best accuracy received was with the random Forest classifier using the mean values data set: ~0.64%. The other classifiers reached between 60-63% accuracies for our test set. Based on these results, we chose to use the random forest classifier for our project.

Out of the available random forest implementations, we tried caret and randomForest R-libraries both for the training and the predicting tasks. In the end, we decided that the caret package, which in itself utilizes the randomForest package, allowed for more customization of the training process. For that reason, we eventually ended up choosing the caret package.

One of the pros of the random forest classifier is its ability to generally perform rather well in classification tasks, as it is a rather mature model. It also gave us the highest accuracies on our training and validation sets out of the classifier models we considered. In addition, the random forest classifier can perform multiclass classification, which we wanted for our approach.

On the other hand, the random forest classifier can be a bit of a black box algorithm. There was not that much tuning that we could do ourselves to try and optimize the model for this particular data. However, we were able to make up for that with some training control parameters and PCA.

The code below forms the primary components:

```r
npf.pcA2 <- prcomp(scale(npf[,vars]))
npf.pcA2.withtest <- prcomp(scale(rbind(npf[,vars],npf_test[,vars])))
```

Feature selection

We used repeated 10-fold cross-validation in order to automatically adjust the hyperparameters used to train our model. Cross-validation was repeated ten times.

After principal component analysis was introduced and after doing some experiments with it, we found out its benefits. We tried this approach to our selected features. This however, gave us worse results than we obtained in the exercise related with this data. We decided to pick all the mean variables and let the PCA functionality do the feature selection for us. The data is scaled and centered before primary component analysis.

For the final classification done on the test set, the primary component analysis was performed on all of the data available which meant the combined training and test sets. We decided to use the first 14 primary components for our model. This choice was based on our experiments with different numbers of PCs and the fact that using 14 components provided the best performance.

Results and insights

We noticed that we were able to improve the performance of the model slightly by using PCA and conducting the classification by using the chosen primary components.

Instead of using high levels of automation in the training process, we also noticed that we were able to get better results by tuning and defining some parameters and phases, such as PCA, separately. This also allowed us to have more control over the classification.

Below is the code we used to take the primary components of the combined training and test sets, as well as the methods we used to train our classifier on the training data, and to predict the labels of the test data points.

```r
train.pc <- data.frame(npf.pcA2.withtest$x[1:458,1:14])
train.pc$class4 <- npf$class4
test.pc <- data.frame(npf.pcA2.withtest$x[459:(458+nrow(npf_test)),1:14])
ctrl <- trainControl(method = "repeatedcv",
                     number = 10,
                     repeats = 10,
                     classProbs = TRUE)
rFClass4 <- train(factor(class4) ~ .,
                    method="rf",
                    data=train.pc,
                    trControl=ctrl)
pred_test4 <- predict(rFClass4, newdata = test.pc)
probs_test4 <- predict(rFClass4, newdata = test.pc, type = "prob")
length(which(pred_test4=="nonevent"))
```

```
## [1] 587
```

```r
length(which(pred_test4=="Ia"))
```

```
## [1] 18
```

```r
length(which(pred_test4=="Ib"))
```

```
## [1] 167
```

```r
length(which(pred_test4=="II"))
```

```
## [1] 193
```

In the assignment instructions, we were informed that the proportion of the nonevent days is slightly bigger than the proportion of the event days. This seems to match the proportions obtained on the test data. Also, the proportion of the event days seems to correlate to our observations on the training data.

Binary classification

We speculate the final accuracy to be 0.78. This is based on the model's performances on different train/validation splits. Most of the splits with training set sizes of 100 to 270 data points resulted in binary classification accuracies of 0.77 to 0.83. Since the actual test data set is bigger than our validation set and since there is the possibility of over fitting, we figured that the actual classification accuracy would probably be a little lower. However, we also considered the fact that even if the actual test set is bigger, so is the training set, when it does not need to be split. Combining these facts, we ended up predicting the accuracy to be approximately 0.78, which is only slightly worse than the accuracies we got on our validation sets.

Below is the code we used to experiment with different training and validation set splits and with different numbers of principal components.

```r
set.seed(42)
# Calculates the accuracy to each class and the total accuracy
npf.pcA2 <- prcomp(npf[,vars], center=T, scale=T)
idx <- sample.int(nrow(npf),229)
training_set <- npf[ idx,]
validation_set <- npf[-idx,]
train.pc <- data.frame(npf.pcA2$x[idx,1:14])
train.pc$class4 <- npf[idx,]$class4
validate.pc <- data.frame(npf.pcA2$x[-idx,1:14])
validate.pc$class4 <- npf[-idx,]$class4
ctrl <- trainControl(method = "repeatedcv",
                     number = 10,
                     repeats = 10,
                     classProbs = TRUE)
rFClass4 <- train(factor(class4) ~ .,
                    method="rf",
                    data=train.pc,
                    trControl=ctrl)
probs4 <- predict(rFClass4, newdata = validate.pc, type = "prob")
pred <- predict(rFClass4, newdata = validate.pc)
confusionMatrix(pred, factor(validate.pc$class4))
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction  Ia  Ib  II nonevent
##   Ia         2   0   0        0
##   Ib         5  21  13        2
##   II         3  11  27        3
##   nonevent   5   7  22      108
##
## Overall Statistics
##
##                Accuracy : 0.69
##                  95% CI : (0.6257, 0.7492)
##     No Information Rate : 0.4934
##     P-Value [Acc > NIR] : 1.362e-09
##
##                   Kappa : 0.4925
##
##  Mcnemar's Test P-Value : 3.322e-05
##
## Statistics by Class:
##
##                      Class: Ia Class: Ib Class: II Class: nonevent
## Sensitivity           0.133333    0.5385    0.4355          0.9558
## Specificity           1.000000    0.8947    0.8982          0.7069
## Pos Pred Value        1.000000    0.5122    0.6136          0.7606
## Neg Pred Value        0.942731    0.9043    0.8108          0.9425
## Prevalence            0.065502    0.1703    0.2707          0.4934
## Detection Rate        0.008734    0.0917    0.1179          0.4716
## Detection Prevalence  0.008734    0.1790    0.1921          0.6201
## Balanced Accuracy     0.566667    0.7166    0.6668          0.8313
```

This model obtains the multiclass accuracy of 0.69 which corresponds to binary classification accuracy of 0.83 when all the event classes are considered to be one combined class. Here, a training set of size 229 is used.