HW 02 - Statistical Learning

G21: Davide Mascolo - Antonella Cruoglio - Giuliana Iovino - Mario Napoli

26 maggio 2022

Part A

We pick randomly *m = 10* observations from the training set which will be used for the second part of the homework. As we can se, there are *1561 observations* and *7042 features*. In order to try to reproduce the 2020-winning solution, we essentially worked on a two steps procedure, with a nonlinear dimensionality reduction followed by a SVM regression.

```
1561 7042
```

As a first approach we tried to use all the features in the dataset on which we have applied the KPCA technique. Despite this we have obtained a bad performance so we decided to go on with another approach.

We have tried to find a subset of relevant features that are able to explain the target variable. So, since we don't have the domain knowledge, we tried to apply the shrinkage methods to do features selection. In particular, we have applied Penalized Regression like *Lasso* and *Elastic Net*, followed by a SVM, but also in this case we didn't get a good performance.

Then, in order to identify good predictors, we tried to use different subgroups of features. We have used the statistical properties of frequency spectrum, then the group of statistics extracted from the STFT and as a last attempt the statistical properties of the signal.

At this point, we have not obtained a satisfying result so we decided to make a feature engineering operation on the *Mel-frequency cepstral coefficients*. We tried to reduce the huge number of features summarizing the information for each mel frequency for each of the 171 temporal instants on which is registered. We have used different statistics like **Mean**, **Variance**, **Standard Deviation** and **Range**. Then, we note that the mean was the variable most capable to explain the tempo variable. Reference

Here, we can see the features engineering of the mel variables. From this point we consider only this group of variables and genre variable for our analysis.

```r
## Transform Data (train)
dt  <- train
idx <- 1
for (i in 1:40){
    dt[,paste("mel", i, sep = "_")] = rowMeans(dt[, idx:(idx+170)], na.rm = TRUE)
    idx <- idx + 170
     }
```

We have used the 10 Fold cross validation repeated 20 times to do Grid Search in order to find the best kernel and best parameters of SVM to use. The kernel that performed best is the Radial with a value of C equal to 10 and a value of sigma equal to 0.01. So, we retrained the method with these parameters values through a 10 Fold cross validation repeated 100 times.

```
Support Vector Machines with Radial Basis Function Kernel

1551 samples
  41 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 100 times)
Summary of sample sizes: 1396, 1396, 1395, 1396, 1396, 1397, ...
Resampling results:

  RMSE      Rsquared   MAE
  11.79131  0.8295167  6.400534

Tuning parameter 'sigma' was held constant at a value of 0.01
Tuning
 parameter 'C' was held constant at a value of 10
```

These are the results obtained, we have a much better performance with respect to the previous tests. We did the same engineering operation for mel features also on the test set.

```r
## Transform Data (test)
dt2 <- x_test[, -c("id", "genre")]
idx <- 1
for (i in 1:40){
    dt2[,paste("mel", i, sep = "_")] = rowMeans(dt2[, idx:(idx+170)], na.rm = TRUE)
    idx <- idx + 170
     }
```

These are the first ten predictions on the test set.

id	target
12	137.2069
15	154.8102
19	118.0743
21	169.1176
23	167.1315
25	165.8139
28	141.4310
29	151.3909
47	140.0090
57	169.7052

Part B

Split Conformal Prediction for Regression

Point 1

Starting from the best model used in the Part A, we implement the *Split conformal Prediction for Regression* algorithm. We have created a function to train the previous model with the best couple values of parameters. Also in this case, we transformed the data with the same operations used in the previous cases.

```r
## Train Control
## KFCV
tr <- trainControl(method = "cv",
                   number = 10)

## Best Parameters
C        <- 10
sigma    <- 0.01
Grid_rad <- expand.grid(C = C, sigma = sigma)

## SVM
svm_reg <- function(data){

  ## Our best function implementation of SVM
  return(train(tempo ~ ., data = data,
                method = "svmRadial",
                tuneGrid = Grid_rad,
                trControl = tr))
}
```

We implemented the function to make the Split Conformal Prediction.

```r
## Predicting with Confidence
conformal_split <- function(data, alpha, reg_mod, y,
                            x_new){

  ## INPUT:
  ## data:    dataset
  ## alpha:   miscoverage level alpha (0,1)
  ## reg_mod: regression algorithm
  ## y:       response variable vector
  ## x_new:   new data points

  ## OUTPUT:
  ## list of predictions band over x (lower and upper)

  n <- nrow(data)

  ## Randomly split D_n into two equal sized subsets
  idx <- sample(1:n, as.integer(n/2))
  D_1 <- data[idx, ]
  D_2 <- data[-idx, ]

  y_2 <- y[-idx]

  ## Train on D_1
  model <- reg_mod(D_1) ## regression train function

  ## Predict and evaluate residuals on D_2
  predictions <- predict(model, newdata =
                           D_2[,-c("tempo")])
  res         <- abs(y_2 - predictions)

  ## d = the k-th smallest value of {Ri}i where
  ## k = d(n/2 + 1)(1 - alpha)
  o <- order(res) ## ordered indexes
  k <- ceiling(((n/2)+1) * (1 - alpha))
  d <- res[o][k]

  lo       <- rep(NA, nrow(x_new))
  up       <- rep(NA, nrow(x_new))
  ## predictions on new data
  pred_new <- predict(model, newdata = x_new)

  for (i in 1:nrow(x_new)) {
    lo[i] <- pred_new[i] - d
    up[i] <- pred_new[i] + d

  }
  return(list(lower = lo, upper = up))
}
```

We apply the algorithm of Conformal Prediction to 10 observations that we set aside before from the training set.

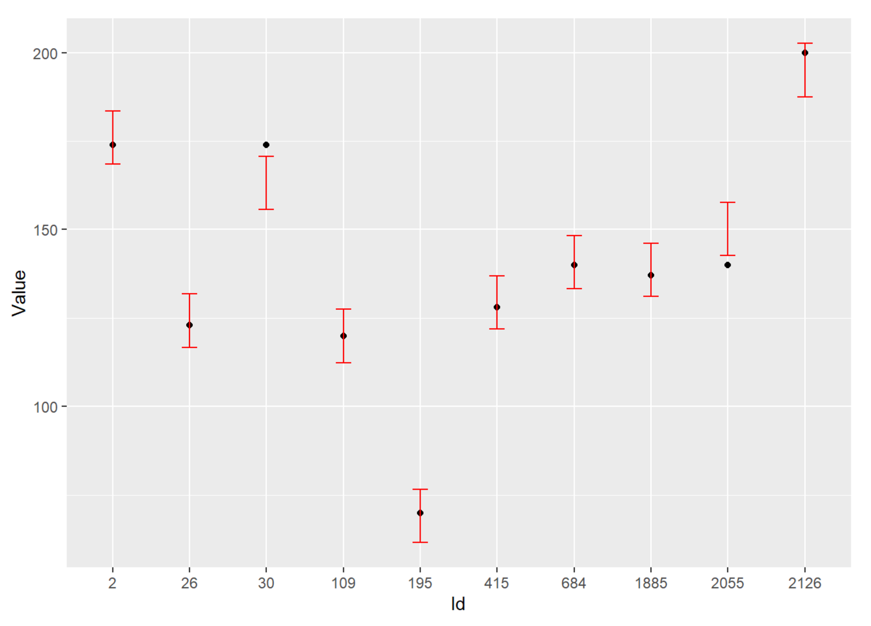
```r
## Seed
set.seed(1234)

## Check (m = 10 observations)
cp_10 <- as.data.frame(conformal_split(mel_train,
                                       alpha = 0.3,
                              reg_mod = svm_reg,
                              y = mel_train$tempo,
                              x_new = mel_train_10[
                                ,-c("tempo")]),
                       col.names = c("Lower", "Upper"))

## Adding target and id variables
cp_10 <- cbind.data.frame(Id = df_10$id,
                          Value = mel_train_10$tempo,
                          cp_10)
```

Id	Value	Lower	Upper
26	123	116.70786	131.76856
1885	137	131.13353	146.19423
2126	200	187.54085	202.60154
195	70	61.53818	76.59888
415	128	121.86577	136.92647
684	140	133.24425	148.30494
2	174	168.44645	183.50715
30	174	155.68094	170.74164
109	120	112.36652	127.42722
2055	140	142.63578	157.69647

We can see that not all the intervals cover the actual response. We know that $\alpha \in (0, 1)$ is the miscoverage level, that is the proportion of the time that the interval doesn't contains the true value of interest. Of course, we note that if we decrease the value of α we have that more actual response values falling in the intervals, but the interval is larger. In this case we set $\alpha = 0.3$.



Point 2

We pick randomly 100 observations from the test set and buil their predictive sets.

```r
## Transform Data
dt4 <- df_100_test[, -c("id", "genre")]
idx <- 1
for (i in 1:40){
    dt4[,paste("mel", i, sep="_")] = rowMeans(dt4[,
    idx:(idx+170)], na.rm=TRUE)
    idx <- idx + 170
     }
```

We apply the algorithm of Conformal Prediction to 100 observations that we picked randomly from the test set. We see only the first 20 predictions and we cannot do the same plot of the previous case because in the test set we don't have the target variable.

```r
## Check (m = 100 observations)
cp_100 <- as.data.frame(conformal_split(mel_train,
                                        alpha = 0.3,
                reg_mod = svm_reg,
                y = mel_train$tempo,
                x_new = mel_test_100),
                col.names = c("Lower", "Upper"))
```

Lower	Upper
128.59063	144.10342
63.17862	78.69141
146.03047	161.54326
146.48342	161.99621
117.62370	133.13648
133.83264	149.34543
133.67853	149.19131
161.94543	177.45821
158.45421	173.96699
83.29662	98.80941
146.42509	161.93788
164.35185	179.86464
99.31508	114.82786
127.92085	143.43364
131.65828	147.17107
162.71229	178.22507
119.45510	134.96789
123.59117	139.10396
162.63294	178.14572
154.52066	170.03345