Course: 02582 Computational Data Analysis, 2020
Case 1
Anders Vestergaard s154993
Ghassen Lassoued s196609

Contents

1 Introduction

We are given a data matrix, $\mathbf{X}$, with dimension $n = 100 \times p = 100$ hence we have 100 features and 100 observations. $\mathbf{X}$ consist of "continuous" features and categorical features, we therefor split up these two groups in a matrix $\mathbf{X}_r$ and $\mathbf{X}_c$ respectively.The type of "continuous" features can be interval or ratio since we do not know the real meaning of the 0 value in these attributes. The dimensions of $\mathbf{X}_r$ are $n \times 95$ and the dimension of $\mathbf{X}_c$ are $n \times 5$. In the categorical data $\mathbf{X}_c$ there are a total of 66 NaN elements. The categorical attributes are nominal.
The response is given by the $n \times 1$ vector, consisting of "continuous" elements. We denote the response $\mathbf{y}$. The task is to build a predictive model of $\mathbf{y}$ based on $\mathbf{X}$. In Fig. 1.1 is seen a scatter plot matrix of the first 10 features in $\mathbf{X}_r$.

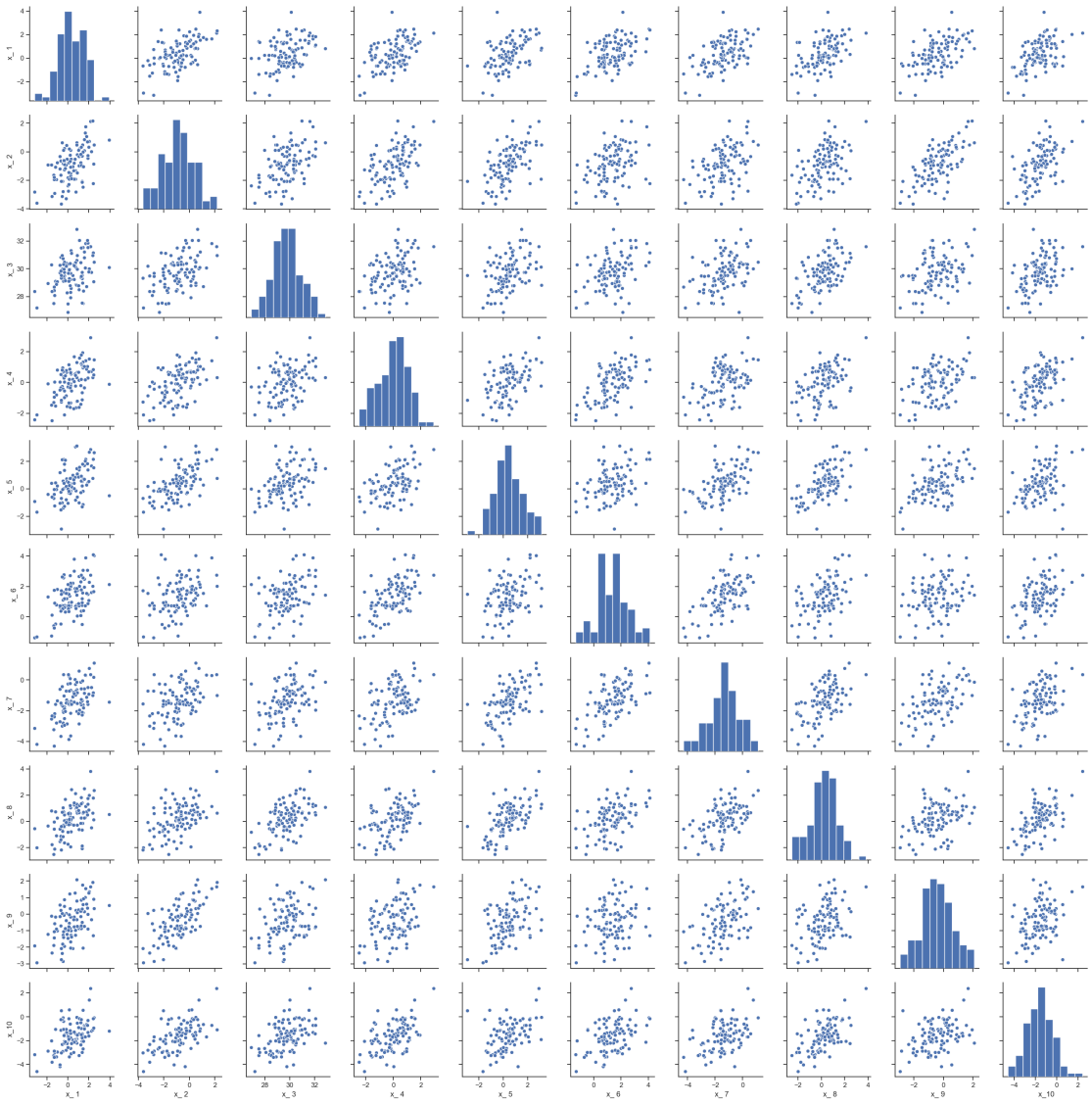

Figure 1.1: Scatter plot matrix of the first 10 column in $\mathbf{X}_r$

Even tough Fig. 1.1 does not give the full picture, it appears that the features are correlated. We further acknowledge this correlation, with a heat-map of $corr(\mathbf{X}_r)$ in Fig. 1.2.

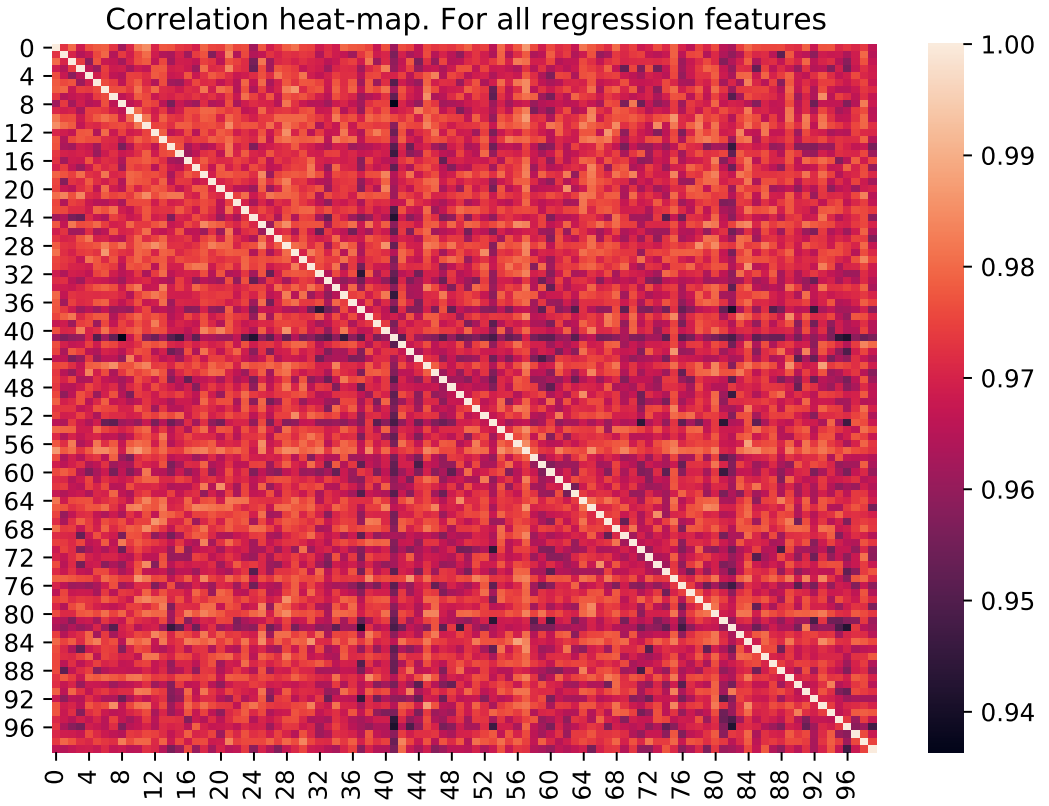

Figure 1.2: Heat-map of correlation matrix from $\mathbf{X_r}$

From the colour-bar in Fig. 1.2 it is seen, that there is no correlation below 0.93. From Lecture 3 we talked about "3 Blessings of Dimensionality", where one of them is:

- Several features will be correlated and we can average over them

And as we concluded, there certainly is a correlation between features.

Model Selection And Method 2

From ESL, the main goal of model selection is described as:

- Model Selection: Estimating the performance of different models in order to choose the best one

In this chapter we:

- Choose which overall modelling methods we want to use.
- Choose how we handled missing values
- How we handled categorical features ($\mathbf{X}_c$)

In general terms we choose the statistical model:

$$Y = f(X) + \epsilon \tag{2.1}$$

where we assume that $E[\epsilon] = 0$ and $Var[\epsilon] = \sigma_\epsilon^2$ We want to find an approximation to $f(X)$, this approximation/model is denoted $\hat{f}(X)$. The model often depends on a hyper-parameter, denoted λ, we therefore write $\hat{f}(X) = \hat{f}(X, \lambda)$ one of the main events in Model Selection is to choose the optimal λ, which we denote $\hat{\lambda}$, hence our prediction will be:

$$\hat{Y} = \hat{f}(X, \hat{\lambda}) \tag{2.2}$$

2.1 Missing Values

There are 66 NaN present in the categorical features distributed on 49 rows, thus removing the observations with NaN elements, would remove 49% of our data (we tried it, but it did not improve RMSE). We investigate the categorical features: Let $S = [G, H, I, J, K]$ be the state space of our categorical features C_1, C_2, C_3, C_4, C_5. We count occurrences of each state in each categorical feature and display them in the below table:

	G	H	I	J	K	NaN
C_1	17	14	19	16	19	15
C_2	14	13	16	27	15	15
C_3	14	23	16	21	13	13
C_4	18	19	18	19	16	10
C_5	14	22	14	17	20	13

Table 2.1: Occurrences of states in each category. i.e 'G' was observed 17 times out of 100 observations in feature C_1 etc.

From a purely visible inspection of Table 2.1 one could suspect that the distribution of the states is uniform across C_1, C_2, C_3, C_4, C_5 i.e $Pr(C_i = s) = 0.2 \; \forall s \in S$, $i = [1, 2, 4, 5]$. Due to this it might not be good to replace NaN values with the most frequent state in each category. The following methods to handle missing values has been chosen:

- Include 'NaN' as a new state, i.e we use the state space $S' = S \cup \{NaN\}$
- Replace 'NaN' values with random state from S.

KNN impute was tried as-well, but did not seem to work batter than random impute.

2.2 Factor Handling

When doing regression, all variables have to be numerical. For that reason, categorical features are transformed into dummy variables using one hot encoding. This method is sufficient when the categorical attributes are not ordinal. In this case, it was assumed that the attributes are only nominal. The transformation made is clarified in the following image :

```
In [6]: before_1hot
Out[6]:
0       K
1       K
2       K
3       J
4       K
5       H
6       K
7       G
8       J
9       J
10      H
11      H
12      H
13      H
14      G
15      K
16      H
17      H
18      J
19      G
20      K
21      G
22      G
23      K
24      I
Name:  C_ 5, dtype: object
```

Figure 2.1: C_5 column before one hot encoding

```
In [14]: after_1hot.iloc[:25,:]
Out[14]:
     C_ 5_ G   C_ 5_ H   C_ 5_ I   C_ 5_ J   C_ 5_ K
0           0         0         0         0         1
1           0         0         0         0         1
2           0         0         0         0         1
3           0         0         0         1         0
4           0         0         0         0         1
5           0         1         0         0         0
6           0         0         0         0         1
7           1         0         0         0         0
8           0         0         0         1         0
9           0         0         0         1         0
10          0         1         0         0         0
11          0         1         0         0         0
12          0         1         0         0         0
13          0         1         0         0         0
14          1         0         0         0         0
15          0         0         0         0         1
16          0         1         0         0         0
17          0         1         0         0         0
18          0         0         0         1         0
19          1         0         0         0         0
20          0         0         0         0         1
21          1         0         0         0         0
22          1         0         0         0         0
23          0         0         0         0         1
24          0         0         1         0         0
```

Figure 2.2: C_5 column after one hot encoding

It can be seen that before one hot encoding, there are 5 different values in C_ 5 which are K,J,H,G,I. This column was transformed to 5 columns as indicated in figure 2.2.

2.3 Model Selection

Suppose we have a model, with hyper parameter λ. We wish to split $(\mathbf{y}, \mathbf{X})$ into [$\mathcal{T}$ =Train, V = Validation, T= Test] sets, and then validate λ on V according to some criteria. But since our data does not have the condition $n >> p$ ($n = p = 100$), some of the sets in $(\mathcal{T}, V, T)$ will contain few observations, which reduces our confidence. Instead we find our optimal hyper-parameter $\hat{\lambda}$, using K-fold Cross-Validation (CV). Divide $(\mathbf{y}, \mathbf{X})$ into two sets [$\mathcal{T}$=Train, T = Test], where $\mathcal{T}$ consist of N observations, then the test set, T, have $t = n - N$ observations. We perform K-fold CV on $\mathcal{T}$ and choose $\hat{\lambda}$ such that: (From the

notation in ESL)

$$\hat{\lambda} = \text{argmin}_\lambda\{CV(\hat{f}, \lambda)\} = \text{argmin}_\lambda\Big\{\frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}^{-\kappa(i)}(x_i, \lambda))\Big\} \tag{2.3}$$

where, $L(y, \hat{f}(X, \lambda))$ is our loss function, which we want to minimise. The model chosen will then be $\hat{f}(X, \hat{\lambda})$. Since the case competition consist of minimising RMSE, we choose the squared error loss function:

$$L(Y, \hat{f}(X, \lambda)) = (Y - \hat{f}(X, \lambda))^2 \tag{2.4}$$

Often one use the one standard error rule since K-fold CV tend to over-fit, however it did not appear to be a benefit in this case. The Bias-Variance decomposition for L, given in Eq. (2.4), is:

$$\begin{aligned} Err(x) =& E[L(Y, \hat{f}(x, \lambda))^2 \mid X = x] && (2.5)\\ &= \sigma_\epsilon^2 + \Big(E[\hat{f}(x, \lambda)] - f(x)\Big)^2 + E[(E[\hat{f}(x, \lambda)] - f(x))^2] && (2.6)\\ &= \sigma_\epsilon^2 + Bias^2(\hat{f}(x, \lambda)) + Var[\hat{f}(x, \lambda)] && (2.7) \end{aligned}$$

Because the output $\mathbf{y}$ is "continuous", we only examine regression methods. As mentioned earlier, we have a relatively high dimension space, and hence, it is natural to investigate regularisation methods, such as Ridge and the Lasso-Ridge hybrid ElasticNet. We investigate four regression methods:

- Elastic-Net
  - Reduce features that are superfluous. We use grid-search, and as recommended from the slides: we choose a fine grid of λ and fewer α, which is the parameters that decides the balance between Lasso and Ridge penalty.
- AdaBoostRegressor build with decision trees
  - Decision trees can handle mixed data good, and has low bias, (high variance)
  - Boosting decrease the variance
- Support Vector Machine Regression (SVR)
  - We wanted to try it

NaN approach / Model	Ridge	ElasticNet	AdaBoostRegression	SVR
NaN included as new category	5.516	4.93	4.78	5.28
NaN replaced by most frequent state	5.41	4.91	4.99	5.36
NaN replaced by random state	5.312	4.91	4.68	5.2

Table 2.2: Model Selection score, from Eq. (2.3)

Hence we choose AdaBoostRegression with NaN replaced by random state, which gave a 4.68 CV score with hyper parameters: learning-rate = 0.1 and numbers of trees = 50

Model Assessment 3

From ESL, the main goal of model assessment is described as:

- Model assessment: Having chosen a final model, estimating its prediction error on new data

We do this by a bootstrap method:

- for i=1:B
  - set random state to i
  - split data into [$\mathcal{T}$, T]
  - train model on $\mathcal{T} = [\mathbf{X}_{train}, \mathbf{y}_{train}]$ using parameters found in Model Selection
  - Calculate RMSE[i] on $T = [\mathbf{X}_{test}, \mathbf{y}_{test}]$
- Take average $\hat{RMSE} = \frac{1}{B}\sum_{i=1}^{B} RMSE[i]$

We perform the above operation with B=1000 on AdaBoostRegression with learning rate 0.1 and 50 decision trees. We then get the following estimated RMSE:

$$\hat{RMSE} = 5.28 \tag{3.1}$$

We show a plot of our prediction of $\mathbf{y}_{test}$ in Fig. 3.1

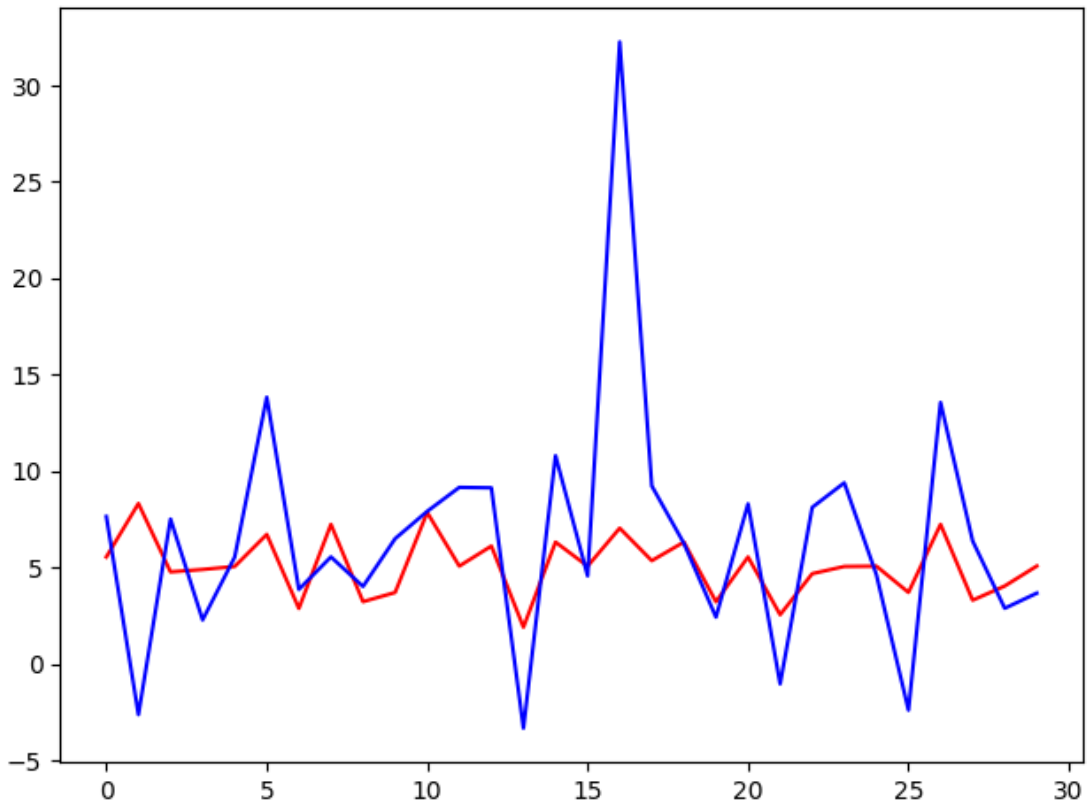

Figure 3.1: $\mathbf{y}_{test}$ is given in blue and the AdaBoostRegression with learning rate 0.1 and 50 decision is the read line. it appears that our model has low bias but high variance.