**Exercise Sheet 1.**

**Data pre-processing. Regression**

**The stars indicate the difficulty of the exercises;**

**[\* easy, \*\* medium, \*\*\* challenging]**

1. **Understanding the structure of the data**

a\*. Describe types of summaries for continuous vs categorical data

What information is presented e.g. min, max, median for continuous variables?

What are the limits of this information.

e.g. can we infer the presence of outliers?

Can we draw any conclusion on the type of distribution?

For categorical data, what properties should it have to be informative?

e.g. do constant (near-constant) features carry any information?

b\*. How do we identify outliers for continuous data? What about categorical features?

Discuss visualisation approaches combined with standardisation (Z transforms).

Do extreme outliers influence the identification of outliers, using a Z transformation (e.g. through μ, σ).

For categorical features, how do we link the frequency of entries to the “usefulness” of a feature?

c\*\*. How do we deal with missing values?

Focussing on features (predictors), missing values can be inferred either by single-value imputation (e.g. median, minimum, maximum) or by model-imputation.

Whenever we impute values, we acknowledge that the reliability of the process is tightly linked to the quality of the data. Therefore, it is essential to check and compare the distributions prior and after the imputation.

Generate 3 features, introduce some missing values, and attempt a value-based and model-based imputation.

*# Generate a dataset with some relationship between the features.*

*fea\_1 <- rnorm(200, mean=50, sd=10)*

*fea\_2 <- feature\_1 \* 2 + rnorm(200, mean=0, sd=5)*

*fea\_3 <- cut(fea\_1, breaks=c(-Inf, 45, 55, Inf), labels=c("lo", "me", "hi"))*

*data <- data.frame(fea\_1, fea\_2, fea\_3)*

*# Introduce missing values in feature 1.*

*data$fea\_1[sample(1:200, 40)] <- NA*

***Python:***

import numpy as np

import pandas as pd

fea\_1 = np.random.normal(50, 10, 200)

fea\_2 = fea\_1 \* 2 + np.random.normal(0, 5, 200)

# Map fea\_1 values to "lo", "me", or "hi"

bins = [-np.inf, 45, 55, np.inf]

labels = ["lo", "me", "hi"]

fea\_3 = pd.cut(fea\_1, bins=bins, labels=labels)

data = pd.DataFrame({'fea\_1': fea\_1, 'fea\_2': fea\_2, 'fea\_3': fea\_3})

# Introduce missing values in feature 1.

missing\_indices = np.random.choice(200, 40, replace=False)

data.loc[missing\_indices, 'fea\_1'] = np.nan

1. **\*\* Adjusting data without tampering with signal**

We simulated a dataset with 3 features (*PreProcessing\_ex2.csv*), 2 continuous, one categorical; outliers were introduced to illustrate their effect on standardization and scaling. Summarise the information available in the data and assess the impact of the outliers.

1. **\*\*\* Cross-validation**

Write code (R, Python) to assess the usefulness of cross-validation.

The inputs comprise a vector of values {1,2, 3, …, 100}, a function *f* and CV option

CV options include: random splitting, LOOCV, k-fold CV

The function *f* can be linear, quadratic, polynomial, logarithmic, exponential.

Step 1: generate the input and output

x = [1, 2, 3, …, 100]

y = [3, 5, …, 201]

## the y values were generated using the function below

## try other examples as well e.g. modify the coefficients

## to each function value (my.y.\*) add noise (modelled as a normal distribution N(0,0.1))

my.x = seq(1,100)

my.y.linear = 2 \* my.x + 1

my.y.quadratic = my.x \* my.x + 5 \* my.x + 9

my.y.polynomial= my.x \* my.x \* my.x + 5 \* my.x \* my.x - 3 \* my.x + 11

my.y.log = log(my.x + 1)

my.y.exp = exp(my.x/10)

***Python:***

my\_x = np.arange(1, 101)

my\_y\_linear = 2 \* my\_x + 1

my\_y\_quadratic = my\_x\*\*2 + 5 \* my\_x + 9

my\_y\_polynomial = my\_x\*\*3 + 5 \* my\_x\*\*2 - 3 \* my\_x + 11

my\_y\_log = np.log(my\_x + 1)

my\_y\_exp = np.exp(my\_x / 10)

Step 2: guess the function

for the example presented above, test first a linear function y = f(x) = ax+b

from [x = 1, y = 3] and [x = 2, y = 5] you have 2 equations with 2 unknowns.

By solving the system of equations, you infer a = 2 and b = 1

For the quadratic function fit

For the cubic function fit

Step 3: calculate the errors for the whole domain/ codomain

Step 4: perform the Step 1 – 3 on a cross-validation setting.

*# Remember the random splitting of the data, for assessing the inference.*

*trainIndex <- createDataPartition(data$y, p = .8, list = FALSE)*

*train\_data <- data[trainIndex, ]*

*test\_data <- data[-trainIndex, ]*

***Python:***

from sklearn.model\_selection import train\_test\_split

train\_data, test\_data = train\_test\_split(data, test\_size=0.2, stratify=data['y'], random\_state=42)

Step 5: assess the MSEs for the various polynomial models.

Generate a plot similar to the ones presented in the lecture, where on the x-axis we present the degree of the polynomial and on the y-axis the MSE.

For the exponential and logarithmic functions, you can use the Taylor approximations.

N(0,σ)

δ=1.35

A dataset comprises the following predictors (pred1, pred2, pred3, pred4) and the output variable, y (*regression\_input.csv*). This following exercises will assist with a thorough understanding of regression tasks.

y: 0 10 20 30 40 50 60 70 80 90 100

pred1: 0 1 6 15 18 24 26 33 37 42 49

pred2: 1 -3 5 15 22 27 28 35 30 38 49

pred3: 0.00 1.58 1.82 1.97 2.09 2.18 2.26 2.33 2.40 2.45 2.51

pred4: -5 48 194 443 798 1246 1799 2446 3196 4048 4995

**Explore the data**

[i\*] generate the scatter plots for all predictors and comment on their relationship with the output variable

**Simple (linear) regression. Assessing the effect of single predictors.**

[ii\*] calculate the coefficients and R2 values for the simple linear regressions for all predictors. Comment the coefficients and R2 values. Propose a first step for the forward selection.

[iii\*\*] on the simple linear regressions, test a non-linear option. Justify your choice of non-linear function.

Generate the evaluation plots for all models.

**Other loss functions**

[iv\*\*] The model initially created focuses on the MSE.

Optimise the model using MAE and the Huber Loss.

**Bias/ Variance decomposition**

[v\*\*\*] For one of the models, calculate the squared bias, the variance and irreducible error.

**Model selection. Forward/ Backward greedy approach**

[vi\*] comment the following multiple linear model

Call:

lm(formula = y ~ pred1 + pred2 + pred3 + pred4)

Residuals:

Min 1Q Median 3Q Max

-4.0443 -0.6245 0.3889 0.8521 3.8189

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -0.237360 2.631506 -0.090 0.9311

pred1 1.613740 0.575488 2.804 0.0310 \*

pred2 -0.134739 0.318304 -0.423 0.6868

pred3 5.372330 2.335277 2.301 0.0611 .

pred4 0.003363 0.002738 1.228 0.2653

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.621 on 6 degrees of freedom

Multiple R-squared: 0.9963, Adjusted R-squared: 0.9938

F-statistic: 398.7 on 4 and 6 DF, p-value: 2.101e-07

[vii\*] investigate full forward selection

[viii\*] investigate full backward selection; comment on convergence

[ix\*] assess interactions between terms

**Categorical features**

[x\*] consider an additional predictor

pred5: 0 0 0 1 1 0 1 1 0 1 1

Answer (i) and (ii) wrt pred5; generate also the full linear model.

[xi\*] consider an additional predictor; what differs (in terms of properties) from pred5

Pred6: 2 0 0 0 2 1 3 1 3 1 1

Answer (i) and (ii) wrt pred6; generate also the full linear model. Try the forward and backward selection. Look for interactions. Generate the best possible model for this data.

Optional (additional) questions

**Logistic regression \*\*\***

Use pred5 as output variable, and build a model using predictors 1-4.

**Non-parametric regression \*\*\***

Build a non-parametric model with all predictors (1-6). Compare it to previous models.

**Regularisation \*\*\***

Assess the effect of ridge, lasso and elastic-net regularisation on the parametric models.