SY19

November 4, 2016

Part I

Abreviations

Abstract

Part II

Introduction

Part III

Ex 1

Part IV

Ex 2 - Breast Cancer Recurring Time

0.1 Introduction

This part aims to build the best model to predict the recurring time of breast cancer based on about 30 features computed from a breast mass. This regression problem will take advantage of a given dataset describing about 200 patient cases.

0.2 Dataset Description

We first take a look at the original dataset to get first hints on how each feature contributes to the recurring time.

The dataset comprises 194 patient cases, each of which is described through 32 features and the cancer recurring time `Time`.

0.2.1 Time

0.2.2 Features Description

Feature Correlation

Based on the definition of the parameters, we already know that many features are correlated :

- The mean of each parameters is bigger than the "worst" value;
- The radius, the perimeter and the area are linked together;
- The compactness can be computed with the perimeter and the area thanks to the given formula : $Compactness = \frac{perimeter^2}{area - 1}$
- *MORE FEATURE CORRELATION*

0.2.3 Data Relevance

We should first check that every point is relevant to our study, in other words, that there is no abnormal point in the dataset. Cook's Distance is an interesting measure to verify this important criteria, it can be computed after a simple Linear Regression.

TALK ABOUT COOK'S DISTANCE

According to this graph, no point is located beyond the critical Cook's boundary. This means that we can potentially use each and every patient case of our dataset to build our regression model.

0.2.4 Relation between "feature" and "time"

GRAPH : each Feature vs. Time

QQ-PLOT : Non Linearities

Heteroscedasticity

0.3 Measures to Compare Models

Before building any model, we have to properly define the measures we will later use to compare them.

0.3.1 Some Measures

TALK ABOUT MSE, R2 AND OTHER STUFF

0.3.2 Data Split

These measures should not be applied on a set whose data was also used to train the model. Indeed, this would include a biais that might distort our conclusions. To cope with this problem, we have to split the dataset into two disjointed sets :

- Training Set : About 75% of the dataset dedicated to the building the model;
- Test Set : The remaining 25% only used at the end to provide some kind of objective measure of the model performance.

Once it is done, we can finally dive in the model building.

0.4 K-nearest neighbors (KNN)

0.4.1 Idea

We start our analysis with a very simple model called the KNN. Given an positive integer k and a test observation x0. The KNN model first identifies the k closest points to x0 from the training data. Then estimates

The KNN model in R is done by calling the function reg of the package knn. As we will see in the following sections, For most prediction algorithms, we first have to build the prediction model on the training data, and then use the model to test our predictions. However, the KNN function does both in a single step. In order to find the best k we set a maximum number of neighbors to be considered (in our model it is 20), then we calculate the MSE for each k which is the mean of the squared difference between the real value of Time and the predicted one. All the steps are detailed in the code below.

```
library(FNN)
library(tikzDevice)
k_max = 120;
MSE = rep(0,k_max)

for( k in 1:k_max)
```

```
{
reg = knn.reg(train=cancer.train.x, test=cancer.test.x,
    y=cancer.train.y, k=k)
MSE[k] = mean((cancer.test.y - reg$pred)^2)
}

best_k_test = which.min(MSE)
best_k_mse = MSE[best_k_test]
sprintf("Best knn1 = %d and the best MSE1 = %f", best_k_test, best_k_mse)

tikz('Figures/knn.tex',width=5, height=5)

plot(1:k_max, MSE, xlab='k', ylab='MSE', main='MSE against k neighbours')
points(x = best_k_test, y = best_k_mse, col = "red", pch = 16)
abline(h = best_k_mse, col='red')
abline(v = best_k_test, col='red')
dev.off()
```

The graph below shows the MSE plotted against the values of k in a range from 1 to 20. We can notice that a minimum is reached between 10 and 20. We use the function which.min that returns the index of the minimum value.

The minimum MSE which yields to the best k is colored in red. Its coordinates correspond to (**12,967.386966**). The best k is therefore . Now that we have the k that minimizes the MSE we call KNN algorithm with this best k and plot the predicted values against the real values. The figure above shows the result. The red line is the function y=x; so further are the points from this line the further are the predicted values ($\hat{y}$) from the real one (y).

```
best_reg_test = knn.reg(train= cancer.train.x, test = cancer.test.x,
    y=cancer.train.y, k = best_k_test)
tikz('Figures/knn_predicted_test.tex',width=5, height=5)
plot(cancer.test.y, best_reg_test$pred, xlab='y', ylab='y-hat',
    main='y-hat (Predicted) against y')
abline(0,1, col='red')
dev.off()
```

When notice that the predicted $\hat{y}$ diverge a lot the real values y. We actually expected those results since the MSE=967.386966 which is quite high.

0.4.2 Best k: CROSS VALIDATION

In our previous reasoning was quite optimistic because we tried finding the best k with minimizing the MSE in the test data. Therefore the model is very specific to our test data which yield to a high bias. The solution here is to find the best k among the train data and then use the best k in the test data.
To find the best k number of neighbors we use the method of cross validation on the train data. There are two methods in cross validation: **cross valida-**

tion leave one out and **K-fold cross validation**. As we do not have that much predictors we can afford the computation of cross validation leave one out.

```
library("kknn")
model_kknn = train.kknn(Time ~., data= cancer.train, kmax = 30, ks =
    NULL, distance = 2, kernel = "optimal")
best_k_train = model_kknn$best.parameters$k
```

```
library("kknn")
```

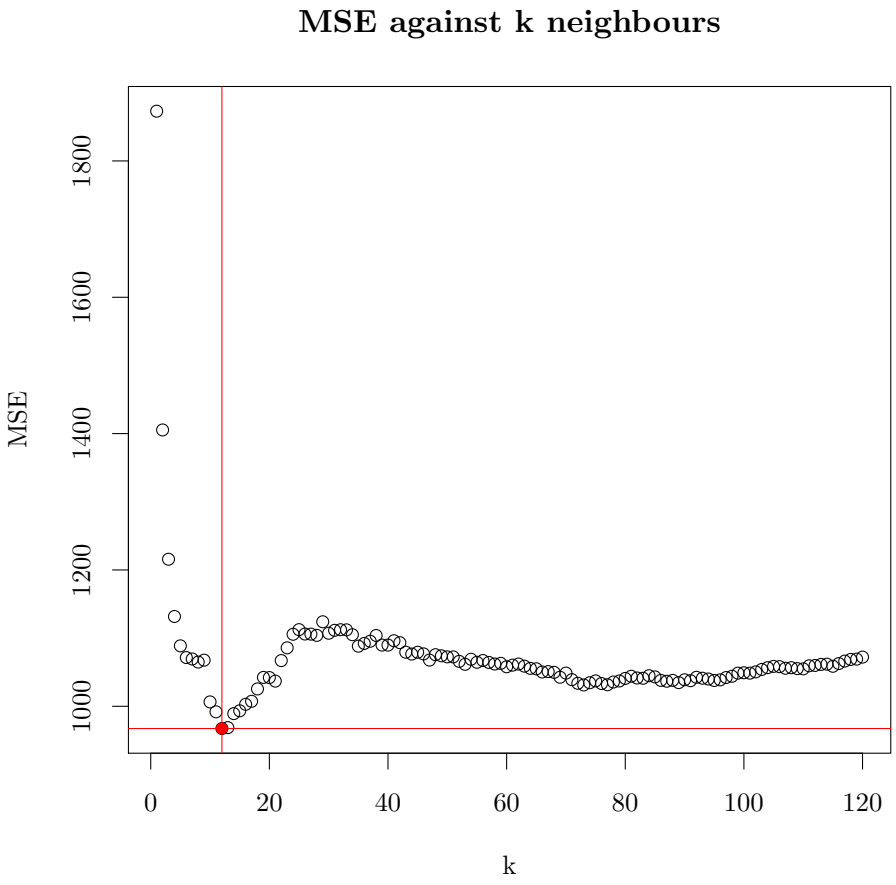

Figure 1: MSE against k neighbours

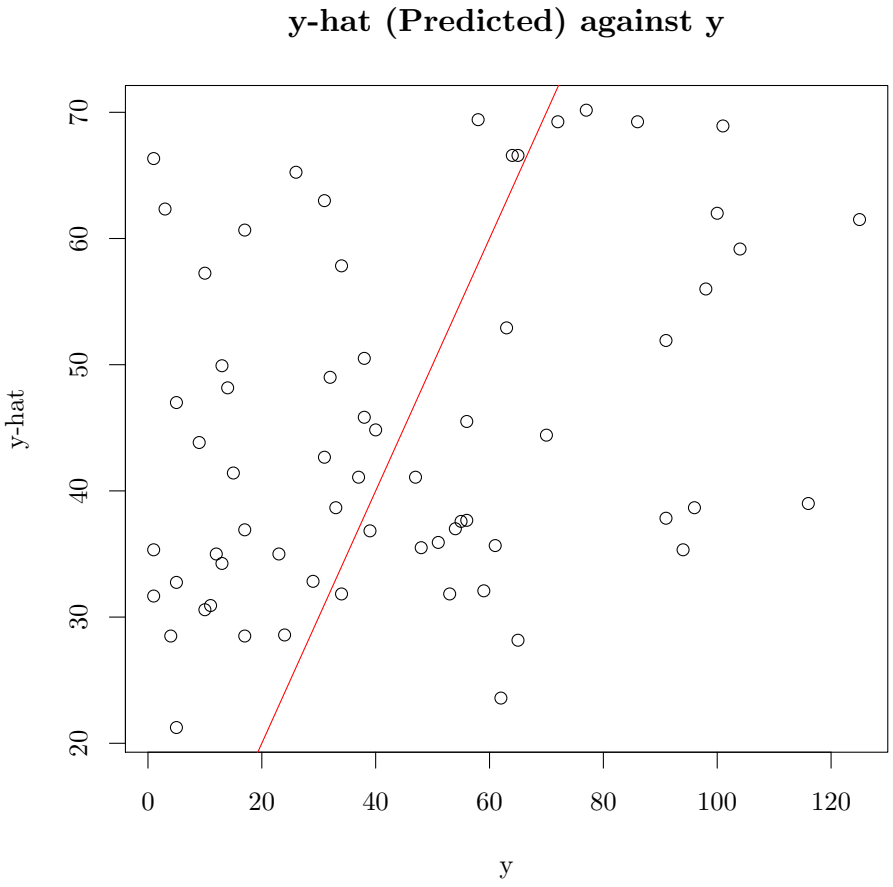

Figure 2: $\hat{y}$ against y

```
model_kknn = train.kknn(Time ~., data= cancer.train, kmax = 30, ks =
    NULL, distance = 2, kernel = "optimal")
best_k_train = model_kknn$best.parameters$k
```

On the test data with the best training k form the LOOCV model

```
best_reg_train = knn.reg(train= cancer.train.x, test = cancer.test.x,
    y=cancer.train.y, k = best_k_train)
tikz('/Users/slam/Desktop/Git/SY19-TP1/Figures/knn_predicted_LOOCV.tex',width=5,
    height=5)
plot(cancer.test.y, best_reg_train$pred, xlab='y', ylab='prediction')
abline(0,1, col='red')
```

```
dev.off()
```

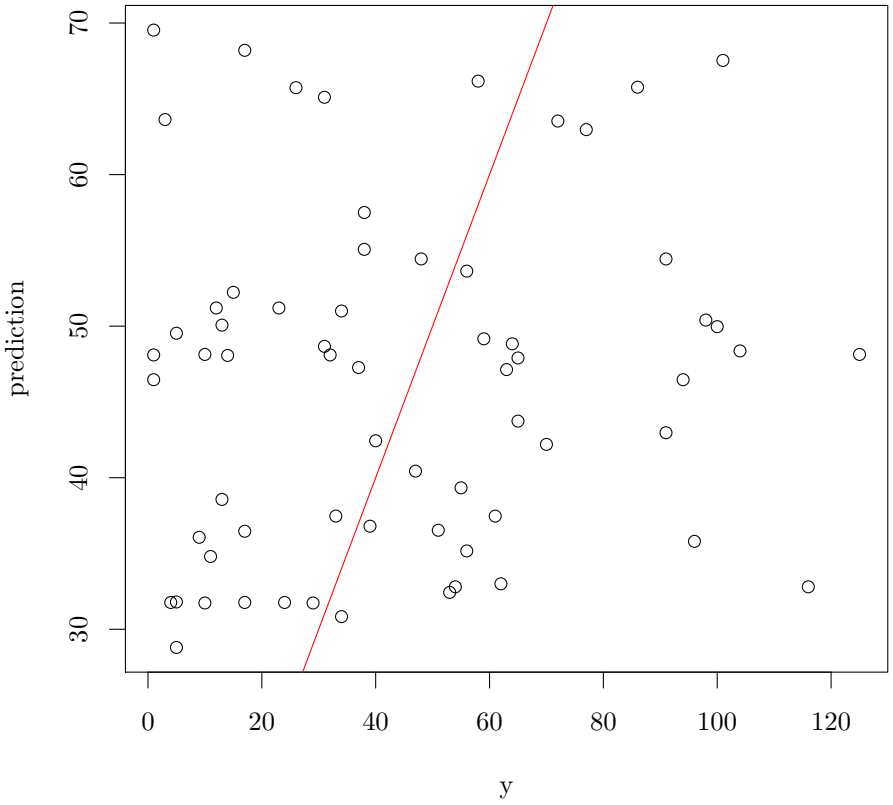

Figure 3: $\hat{y}$ against y

The prediction is not better but the model is not biased

0.5 Simple Linear Regression

0.5.1 Idea

0.5.2 Build the Model

0.5.3 Model Analysis

HIGH P VALUES

0.6 Linear Regression with Features Selection

0.6.1 Idea

0.6.2 Build the Model

EXHAUSTIVE

0.6.3 Model Analysis

0.7 Linear Regression with Regularization

RIDGE + LASSO

0.7.1 Idea

0.7.2 Build the Model

0.7.3 Model Analysis

0.8 Models Comparaison

USE TEST SET TO COMPARE MODEL