6_Machine Learning GLS

Simone Zanetti

4/7/2018

APPROACH

The **approach** related to the machine learning analysis on this project regards a **supervised problem**. In fact, both the value of the dependent variable and the independent variables are known, making possible to split the data into train data and test data. From this point of view, the analysis represents a **problem of regression** where the **goal is the attempt to define a model which can predict the value of the number of deliveries per hour that a driver can perform** based on the parameter defined by the independent variable which will be estabilished to be more significant.

VARIABLES CONSIDERED

In particular, the independent variable that will be taken in consideration to start the analysis are:

independent variables	description
tot_packloaded	total packs loaded on the van each morning
packarrived_total	total packs arrived for each driver/zone
tot_services	total services done
pickedup_total	total packs picked up
tot_weight_pack	total weight of the packs picked up
del_zone1,2,3	deliveries performed in each of the three area of the city

MODELS INVOLVED

The model that will be analysed in this project are the

- **LINEAR REGRESSION**
  - **Traditional**
  - **Lasso**
  - **Ridge**
- **REGRESSION TREE**

Each model will be analysed throught a comparison between the mean of the squared difference between the predicted results on the test data and the effective values of the dependent variable on those test data.

Before the analysis, it will be necessary to fix the issue that regards the missing values that are found on the variable packarrived_total.

ANALYSIS

Before to start, it will be necessary to **optimise the dataset** in order to avoid to have unuseful variable that are unnecessary for the analysis.

```
data2 <- aggregate_data_last
str(data2)

# deleting the column about driver_code
data2 <- data2[,-1]

# delete the input of my response variable

data2 <- data2[,-c(1,3)]
```

SOLVE THE PROBLEM WITH MISSING VALUES

The missing values will be **treated like a response variable of a test data**. In this way, it will be possible to obtain a predictive model using the train data (the data where I have the values of the dependent variable available) and obtain the missing value with that model.

```
data.missed <- data2[which(is.na(data2$packarrived_total)),]
index.missed <- which(is.na(data2$packarrived_total))
head(data.missed)

data.complete <- data2[-index.missed,]  # complete cases only
```

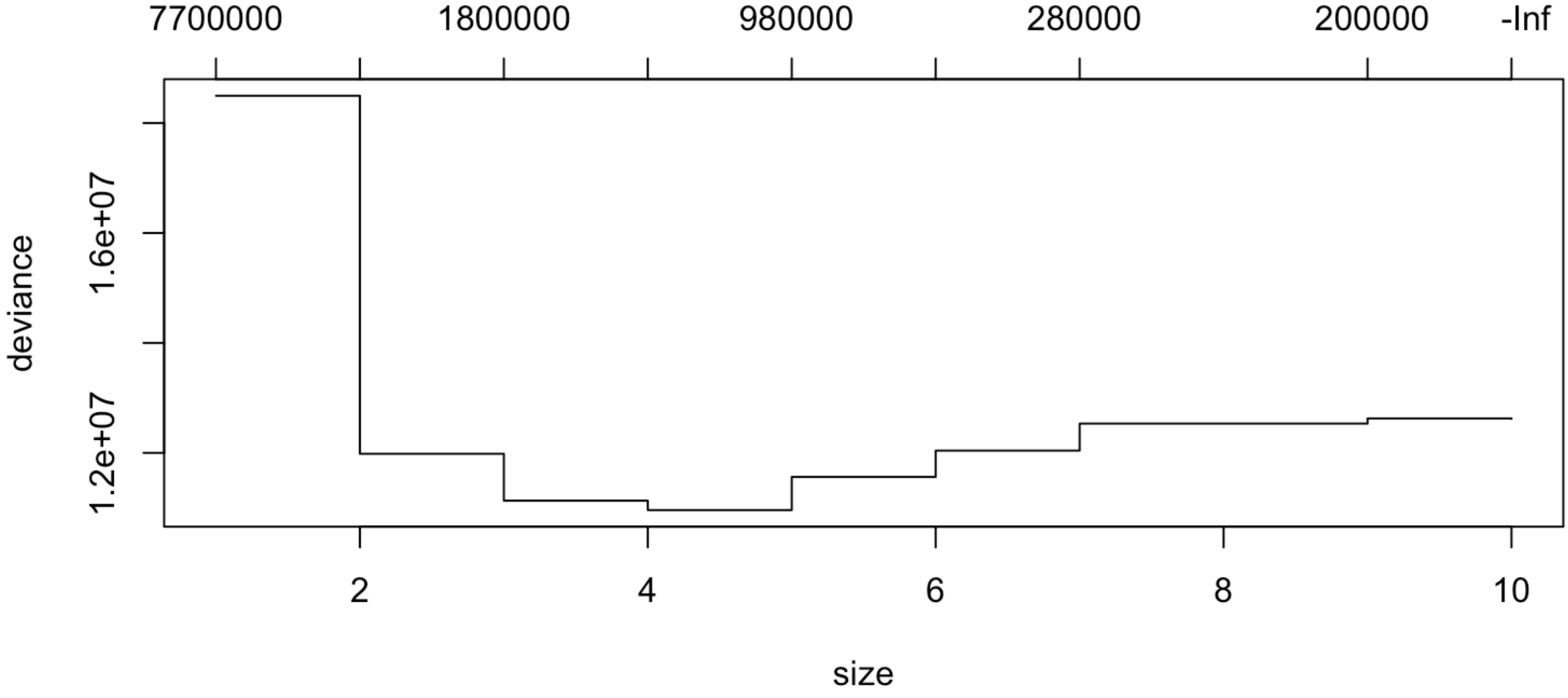
In order to perform this operation, the **REGRESSION TREE** model will be applied

```
library(tree)
```

```
# creating train and test set
set.seed(1234)
train.index <- sample(x = 1:nrow(data.complete), size = 50, replace = F)
train <- data.complete[train.index, ]
test <- data.complete[-train.index,]

tree_complete <- tree(packarrived_total ~., data = train,
                 control = tree.control(nobs = NROW(train), minsize = 2, mindev =
0.01))
```

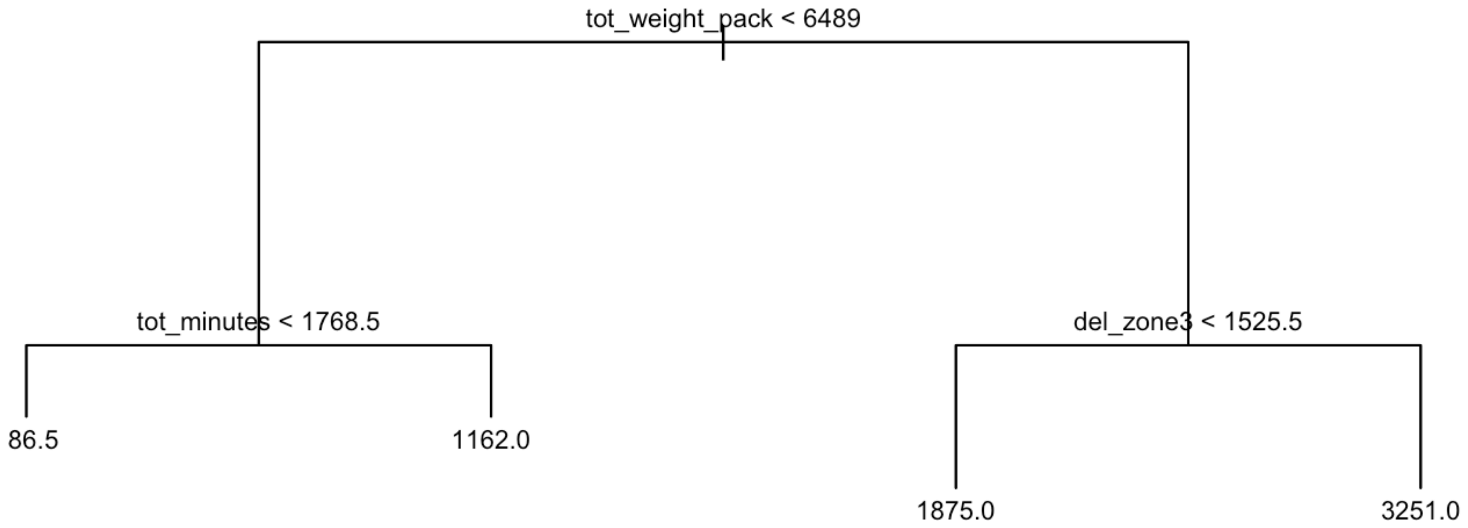
Cross-validation can be useful to recombine the train data in order to permorm deeper analysis, overcoming the issue that can be caused by the fact that the dataset does not contain a big amount of observation.

```
set.seed(99)
pruning <- cv.tree(tree_complete, FUN = prune.tree, K = 10)
plot(pruning)
```



The graph allows to identify the depth that minimises the loss function

```
J <- pruning$size[pruning$dev == min(pruning$dev)]
J # In order to verify the depth of the tree that minimises the loss function
m.tree <- prune.tree(tree_complete, best = J)
plot(m.tree)
text(m.tree, cex = .7)
```



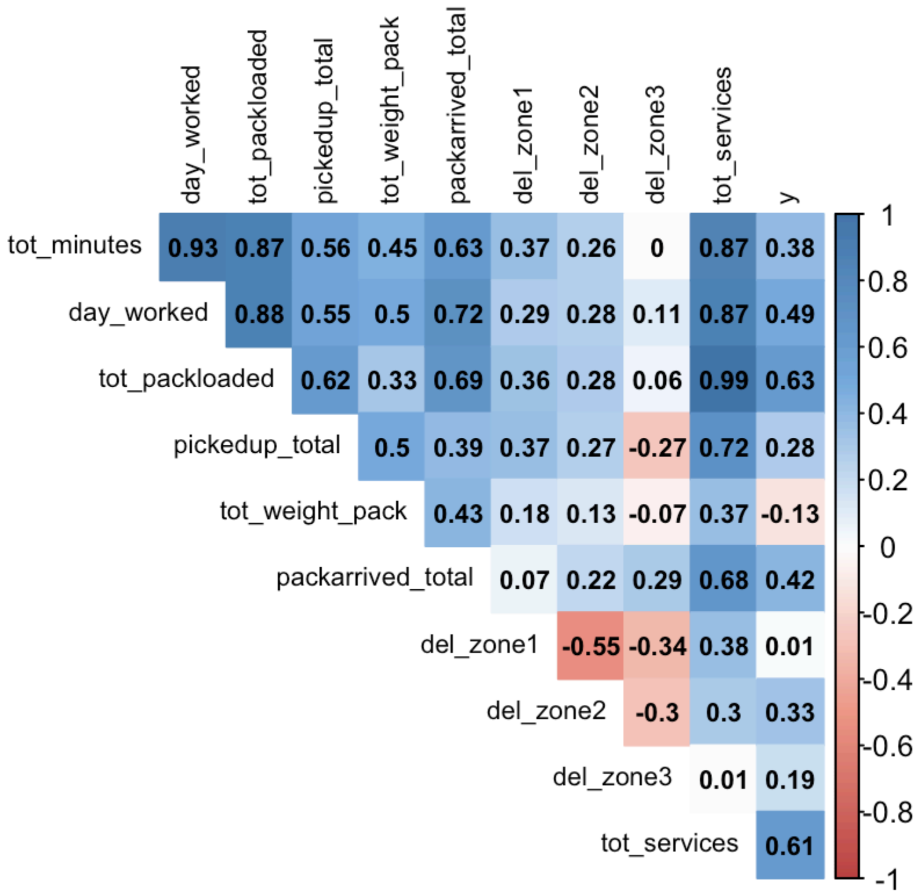
```
# predictions for the packarrived_total variable
pred <- predict(m.tree, newdata = data.missed[,-which(names(data.missed) == "packa
rrived_total")])

# replacing NA's with our predictions
data2$packarrived_total[index.missed] <- pred
```

WORK ON PREDICTIVE MODELS

The **first step** is to observe the **correlation table** to verify the risk of collinearity between variables

```
library(corrplot)
```

```
corrtab <- cor(data2)
col <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD", "#4477AA"))
corrplot(corrtab, method = "color", col = col(200),
         type = "upper", number.cex = .7, tl.cex = .7,
         addCoef.col = "black", # Add coefficient of correlation
         tl.col = "black", tl.srt = 90, # Text label color and rotation
         # hide correlation coefficient on the principal diagonal
         diag = FALSE)
```



The plot allows to observe that tot_services, day_worked and tot_packloaded will suffer of almost perfect collinearity. In fact, they essentially bring the same information in modeling the variability of the response. Also tot_services and pickedup_total have high correlation.

FIX THE ISSUES BEFORE THE ANALYSIS

1) INFORMATION RELATED TO THE ZONE

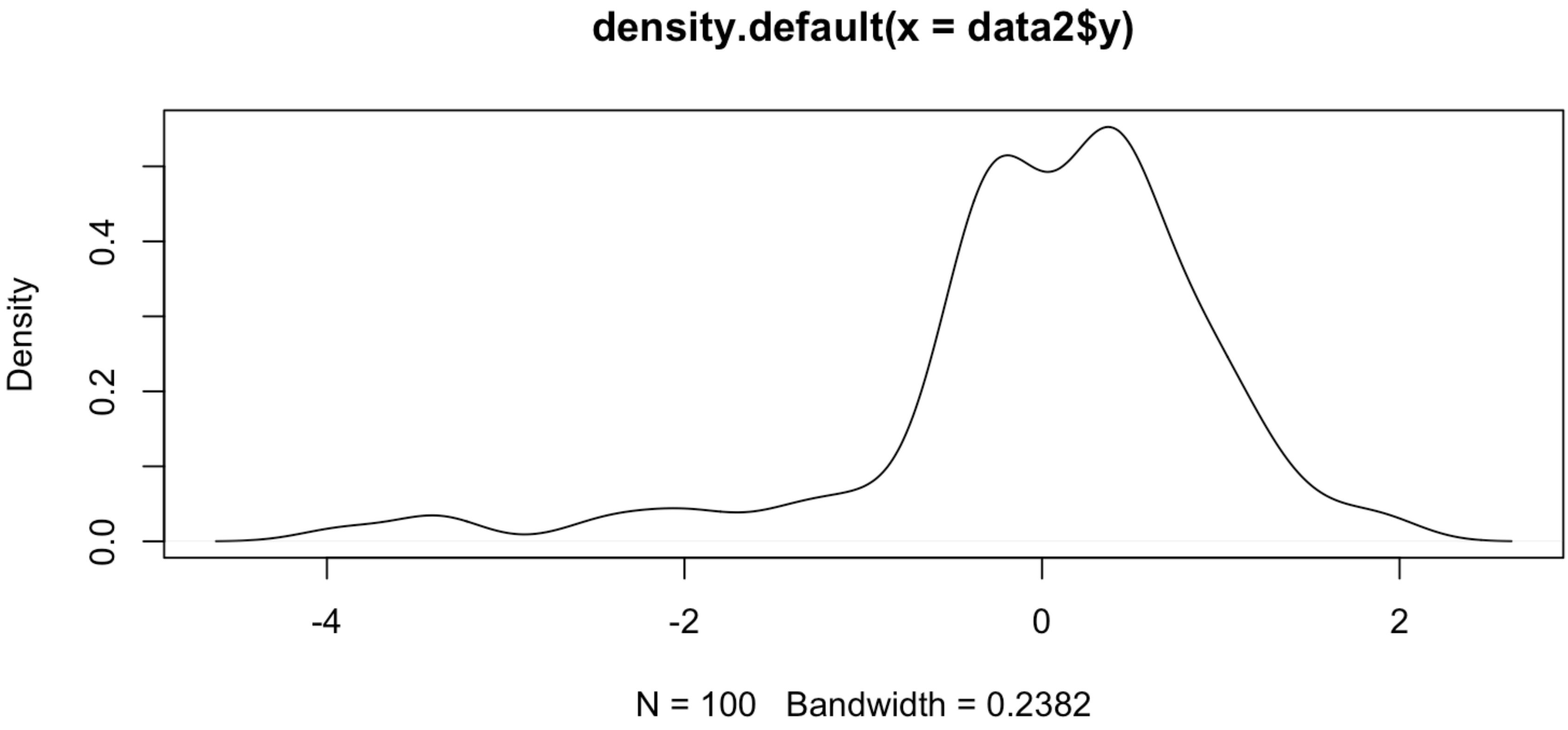
It may be **more informative** to use the **zone information as a categorical variable.** In this way we can highlight the importance of being in zone 1/2/3

```
names(data2)
data2$zone <- apply(X = data2[,7:9], MARGIN = 1, FUN = which.max)
data2$zone <- factor(data2$zone, levels = 1:3, labels = c("Zone 1", "Zone 2", "Zon
e 3"))
table(data2$zone)

# deleting original columns about zones
data2 <- data2[,-c(7,8,9)]
```

2) NECESSITY OF STANDARDIZATION

The **problem** that can verify during the application of predictive models can be the big **difference of the scale** of the different variables. As a consequence, each variable will be **standardized.**

```
# scaling the dataset (each column minus its mean divided by its std deviation)
categorical.index <- which(names(data2) == "zone")
data2[,-categorical.index] <- as.data.frame(scale(data2[,-categorical.index]))
head(data2)

plot(density(data2$y))
```



LINEAR MODEL: TRADITIONAL

Now, we can think about a simple linear model, as the value of the y can be also negative

```
m1 <- lm(y ~ ., data = data2)
summary(m1)
```

```
##
## Call:
## lm(formula = y ~ ., data = data2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.59939 -0.27385  0.01505  0.34987  1.48021
##
## Coefficients: (1 not defined because of singularities)
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -0.18878    0.11440  -1.650  0.10235
## tot_minutes       -0.81716    0.19200  -4.256 5.04e-05 ***
## day_worked         0.67117    0.21289   3.153  0.00219 **
## tot_packloaded     0.89840    0.17037   5.273 8.99e-07 ***
## pickedup_total     0.06662    0.09582   0.695  0.48866
## tot_weight_pack   -0.38037    0.08726  -4.359 3.43e-05 ***
## packarrived_total -0.03174    0.09994  -0.318  0.75152
## tot_services            NA         NA      NA       NA
## zoneZone 2         0.25348    0.14921   1.699  0.09277 .
## zoneZone 3         0.38693    0.21932   1.764  0.08104 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6221 on 91 degrees of freedom
## Multiple R-squared:  0.6442, Adjusted R-squared:  0.6129
## F-statistic:  20.6 on 8 and 91 DF,  p-value: < 2.2e-16
```

As expected, the tot_services variable shows perfect collinearity. Trainting the model without that variable.

```
m2 <- lm(y ~ . - tot_services, data = data2)
summary(m2)

# backward selection <- I delete the variable packarrived_total after observed the
P value of it, which is the highest.
m3 <- lm(y ~ . - tot_services - packarrived_total, data = data2)
summary(m3)

# last step <- remove day_worked. It is an attempt based on common sense, as total
minutes and worked hour represent the same value. On the other side, picked up tot
al that does not show significance, should be significant as it can represent the
time lost on the deliveries in order to perform picking up of goods.

m4 <- lm(y ~ . - tot_services - packarrived_total - day_worked,
                   data = data2)
summary(m4)

# we observed that the zone3 value becomes statistically significant.
# However, pickedup_total does not have effects on the model, so it has to be remo
ved.
linear.model <- lm(y ~ . - tot_services - packarrived_total - day_worked - pickedu
p_total,
                   data = data2)
```

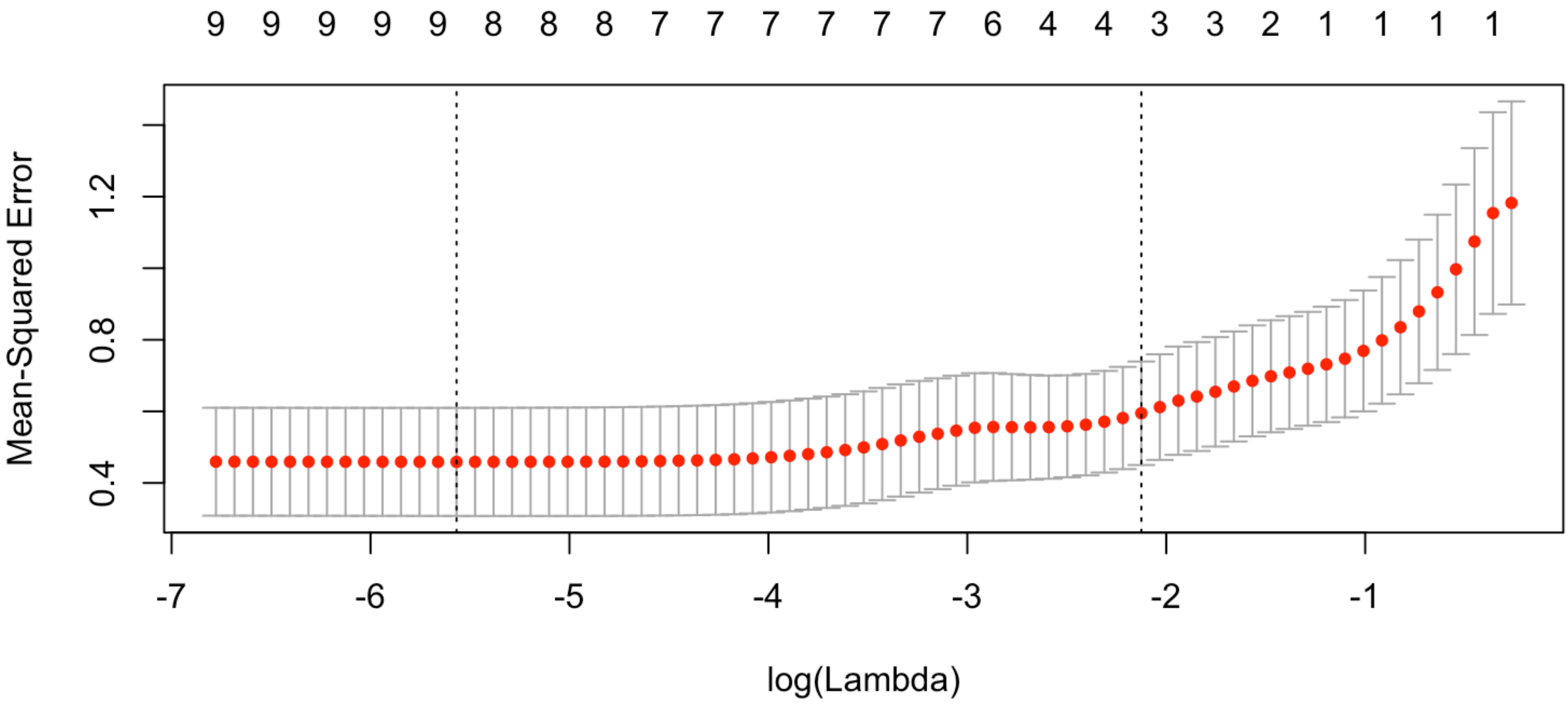
```
summary(linear.model)
```

```
##
## Call:
## lm(formula = y ~ . - tot_services - packarrived_total - day_worked -
##     pickedup_total, data = data2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.70408 -0.30518  0.00408  0.39491  1.54785
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -0.22273    0.11356  -1.961 0.052802 .
## tot_minutes     -0.42082    0.14360  -2.931 0.004247 **
## tot_packloaded   1.13333    0.13539   8.371 5.29e-13 ***
## tot_weight_pack -0.26858    0.07437  -3.612 0.000491 ***
## zoneZone 2       0.27238    0.15150   1.798 0.075404 .
## zoneZone 3       0.52620    0.19764   2.662 0.009128 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6463 on 94 degrees of freedom
## Multiple R-squared:  0.6034, Adjusted R-squared:  0.5823
## F-statistic:  28.6 on 5 and 94 DF,  p-value: < 2.2e-16
```

PENALISED LINEAR MODEL (LASSO)

```
# Obtain train and test

set.seed(1234)
train.index <- sample(x = 1:nrow(data2), size = 70, replace = F)
train <- data2[train.index,]
test <- data2[-train.index,]

# Try lasso with cross-validation
# The pack requires to manually obtain the vector of response variable and a matri
x of the
# independent variables
train.y <- train$y
train.X <- model.matrix(~., data = train[,-which(names(train) == "y")])

test.y <- test$y
test.X <- model.matrix(~., data = test[,-which(names(test) == "y")])
```

```
library(glmnet)
m.lasso <- cv.glmnet(train.X[,-1], train.y, alpha = 1 ) # default K= 10
plot(m.lasso)
```



The lines identify: 1.the lambda value with lowest mean squared error.
2.the lambda value with smallest confidence interval.
So, I choose the second.

```
coef(m.lasso)  # it gives back the values of the coefficients corresponding to the
chosen lambda
```

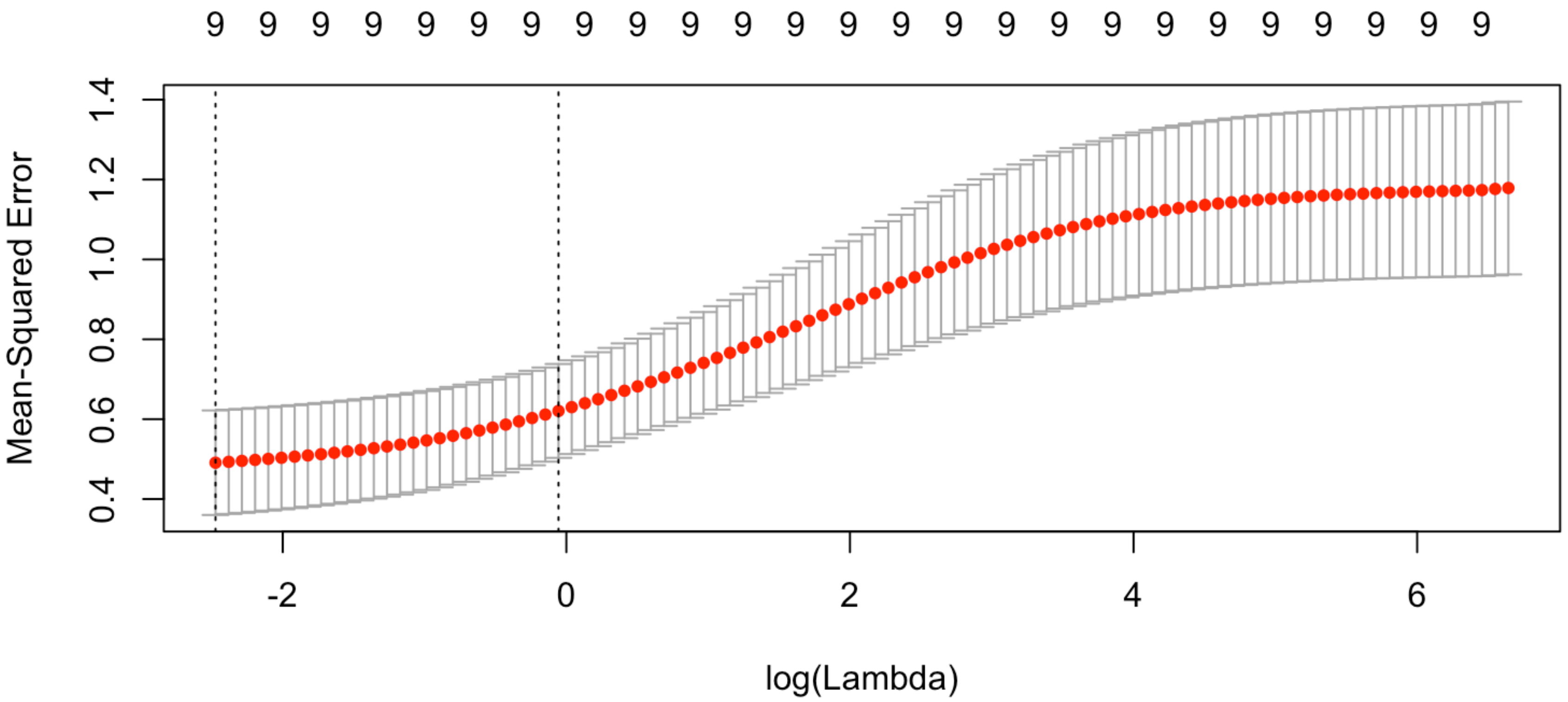
```
## 10 x 1 sparse Matrix of class "dgCMatrix"
##                              1
## (Intercept)        -0.06629278
## tot_minutes         .
## day_worked          .
## tot_packloaded      0.70615462
## pickedup_total      .
## tot_weight_pack    -0.16599369
## packarrived_total   .
## tot_services        .
## zoneZone 2          .
## zoneZone 3          0.19987436
```

```
# I predict on my test data
p.lasso <- predict(m.lasso, newx = test.X[,-1])

# Now I create a parameter for future comparisons between the different models
mse.lasso <- mean((p.lasso - test.y)^2)
```

```
mse.lasso
```

PENALISED LINEAR MODEL(RIDGE)

```
m.ridge <- cv.glmnet(train.X[,-1], train.y, alpha = 0 )
plot(m.ridge)
```



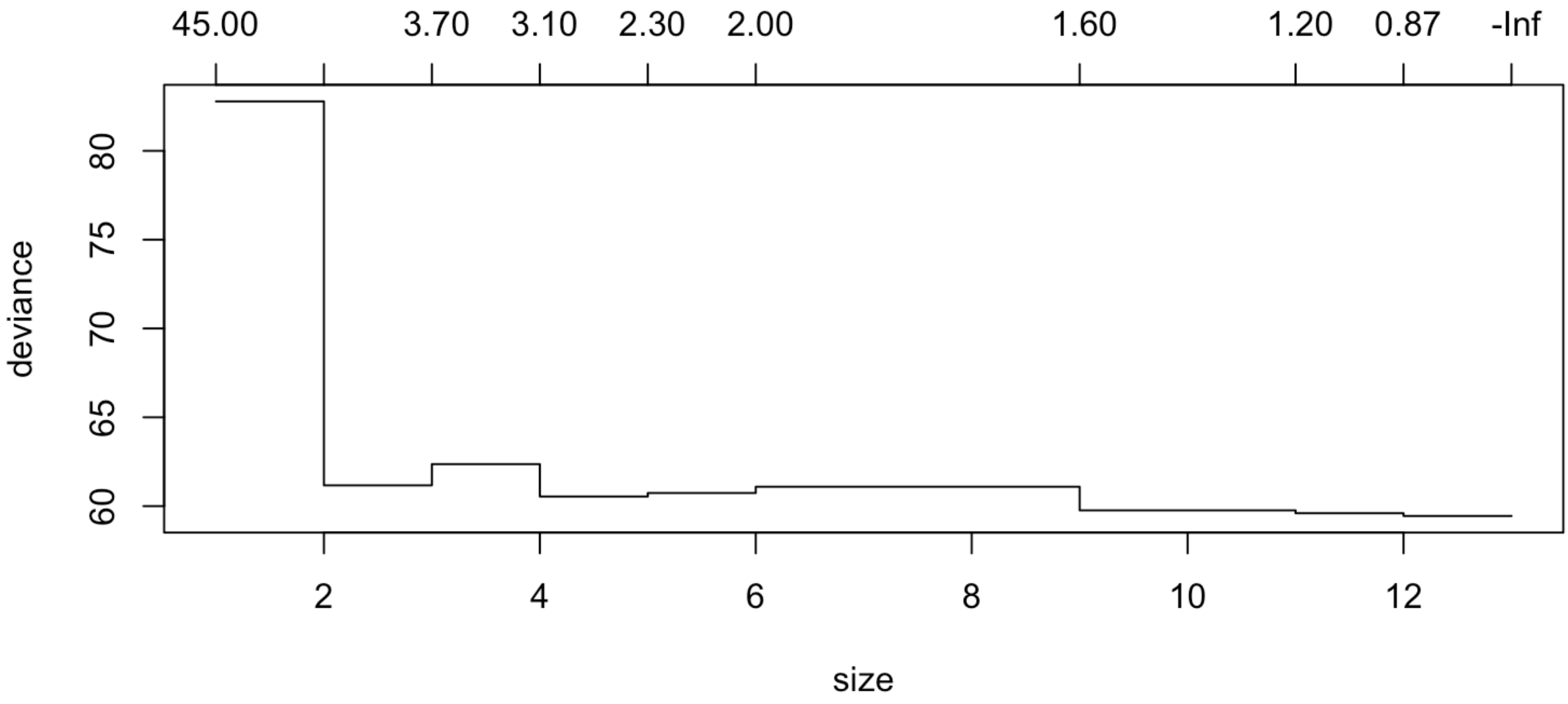
```
coef(m.ridge)     # I chose the second one

coef(m.ridge, s = "lambda.min")
coef(m.ridge, s = "lambda.1se")

# I predict on my test data
p.ridge <- predict(m.ridge, newx = test.X[,-1])
mse.ridge <- mean((p.ridge - test.y)^2)
```

```
mse.ridge
```

```
## [1] 0.4931289
```

REGRESSION TREE

```
tree_complete <- tree(y ~., data = train,
                    control = tree.control(nobs = NROW(train), minsize = 2, mindev =
0.01))

# pruning using cross-validation
set.seed(99)
pruning <- cv.tree(tree_complete, FUN = prune.tree, K = 10)
plot(pruning)
```



The plot represents a situation where the result is not clear. In fact, the deviance does not increase again at the increase of the depth of the tree. As a consequence, there is a need to arbitrary chose a value that can be able to increase the capacity of prediction avoiding the risk of overfitting that can happen as long as the number of chosen branches increase.

```
J <- pruning$size[pruning$dev == min(pruning$dev)]
J
```

```
## [1] 12
```

```
m.tree <- prune.tree(tree_complete, best = 4)


pred <- predict(m.tree, newdata = test[,-which(names(data.missed) == "y")])

mse.tree <- mean((pred - test$y)^2)
mse.tree
```

```
## [1] 0.9085693
```

```
J <- pruning$size[pruning$dev == min(pruning$dev)]
J
```

```
## [1] 12
```

```
m.tree <- prune.tree(tree_complete, best = 6)


pred <- predict(m.tree, newdata = test[,-which(names(data.missed) == "y")])

mse.tree <- mean((pred - test$y)^2)
mse.tree
```

```
## [1] 0.9364772
```

```
J <- pruning$size[pruning$dev == min(pruning$dev)]
J
```

```
## [1] 12
```

```
m.tree <- prune.tree(tree_complete, best = 8)


pred <- predict(m.tree, newdata = test[,-which(names(data.missed) == "y")])

mse.tree <- mean((pred - test$y)^2)
mse.tree
```

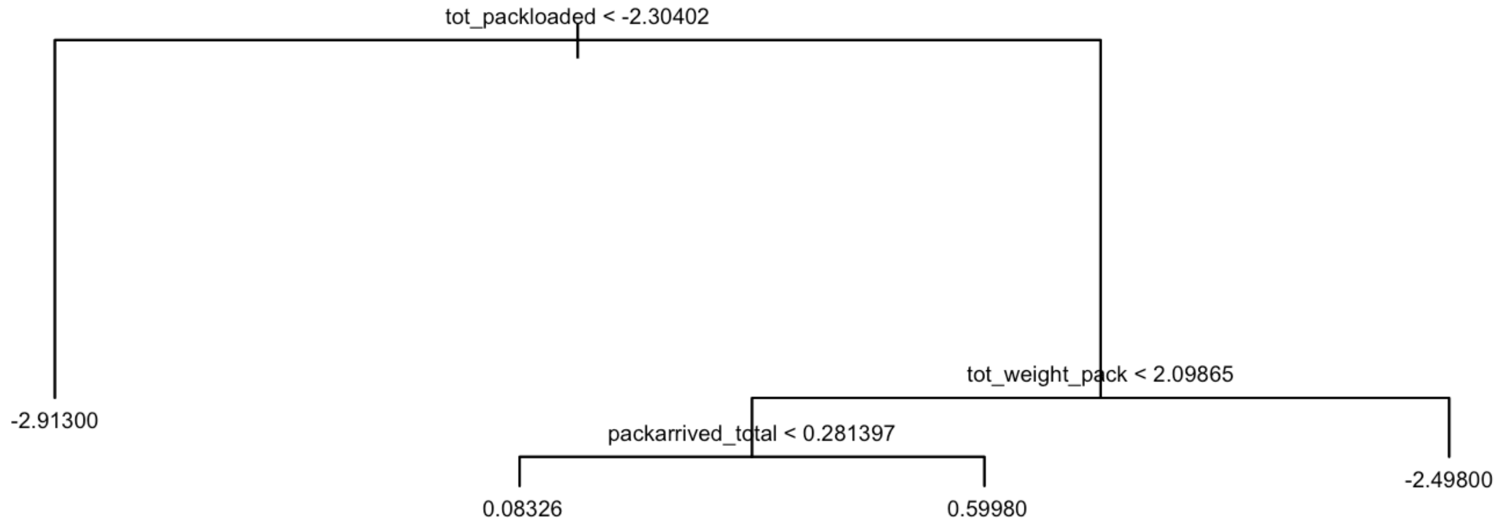
```
## [1] 0.9736361
```

The number has been arbitrary chosen to be 4

```
J <- pruning$size[pruning$dev == min(pruning$dev)]
J
```

```
## [1] 12
```

```
m.tree <- prune.tree(tree_complete, best = 4)

plot(m.tree)
text(m.tree, cex = .6)
```


tot_packloaded < -2.30402
tot_weight_pack < 2.09865
-2.91300
packarrived_total < 0.281397
-2.49800
0.08326
0.59980

```
pred <- predict(m.tree, newdata = test[,-which(names(data.missed) == "y")])
```

```
mse.tree <- mean((pred - test$y)^2)
mse.tree
```

```
best_model_fit <- c(mse.lasso,mse.ridge,mse.tree)
names(best_model_fit) <- c("mse.lasso","mse.ridge","mse.tree")
best_model_fit
```

```
## mse.lasso mse.ridge  mse.tree
## 0.5102480 0.4931289 0.9085693
```

The comparison between the model suggests the **Penalised Linear Model Ridge** to be the best to predict the number of deliveries per hours.