Selection of model ideal

Code

Load Libraries

Hide

```
library(tidymodels) # basic libraries
```

Load Data

Hide

```
df<-read.csv("rent-amount-clear.csv")
```

Data Preprocessing

One Hot Encoding

t is used for qualitative variables, for example if the house is furnished or not. Dummy variables are created according to the number of categories where a 1 is assigned when the condition is met. While in the opposite case it is filled with 0, in this case it is a binary variable. It doesn't make sense to create a new column, just assign it a 1 where the condition is met. Since it is perfectly understood.

Hide

```
ohe_binary<-function(x,class_category){

  ifelse(x==class_category,1,0)
}
```

Hide

```
attach(df)
```

Hide

```
ohe_animal<-mapply(ohe_binary,animal,"acept")
ohe_furniture<-mapply(ohe_binary,furniture,"furnished")
ohe_city<-mapply(ohe_binary,city,"yes")
```

One Hot Binary Transform

Hide

```
df<- df %>%

  mutate(animal=ohe_animal) %>%
  mutate(furniture=ohe_furniture) %>%
  mutate(city=ohe_city)
```

With the mutate function we perform the transformation of variables.

Hide

```
df %>% select(city,animal,furniture) %>% head()
```

	city <dbl>	animal <dbl>	furniture <dbl>
1	1	1	1
2	0	1	0
3	1	1	1
4	1	1	0
5	1	0	0
6	1	1	0

6 rows

Data scaling.

We do it so that the variables can be comparable to each other. Where for each observation the average will be subtracted and divided by the standard deviation.

Hide

```
scaled_data<-function(data){

  data<-as.matrix(data) # transform data  to matrix

  data<-apply(data,MARGIN = 2,FUN=scale) # scaler data

  data<-as.vector(data) # transform data to vector

  return(data)
}
```

Hide

```
scaler_many<-function(data){

  columns<-c("rooms","bathrooms","floor","parking.spaces")

  for(col in columns){
    data[,col]<-scaled_data(data[,col])
  }

  return(data)
}
```

Scale Data

Hide

```
df<-scaler_many(df)
```

Save DataFrame

Hide

```
write.csv(df,"rent-amount-clear-preprocesing.csv",row.names = FALSE)
```

Hide

```
library(caret) # machine learning library
```

Split Data

We split the data, to see if our model is capable of solving new predictions, with data that it has never seen.

Hide

```
set.seed(2018) # random state

training.ids<-createDataPartition(rent.amount,p=0.7,list = F)

train_data<-df[training.ids,] # select train data index
test_data<-df[-training.ids,] # select test data index
```

Hide

```
dim(train_data)
```

```
[1] 4256   10
```

Hide

```
dim(test_data)
```

```
[1] 1824   10
```

We have 4256 observations to train the model. And the rest to perform a validation, to see if the model is capable of generalizing the problem.

Cross Validation

It is a technique that consists of making sub samples in the data, where each sample would be evaluated. In order to see the average generalization of the model.

Penalty methods.

- **L1:** It consists of **minimizing** the **weight** of the **coefficient** for the variables that are not so significant, but complement the prediction. It is used when we know that all the variables influence the variable to be predicted. The closer the alpha value is to 0, the more influence this regularization method will have.
- **L2: Cancels the 0** for the variables that are **not so relevant.** It is used when we do not know if all the variables are of vital importance for the prediction. The closer the alpha value, the greater the L2 penalty.
- **L1_L2:** It is a **combination** of the **two regularization techniques.** With alpha value of 0.5 combine the two regularization methods.

We will use the **L1 method**, since we know perfectly well that all the variables contribute to the result.

Hide

```
train_control<-trainControl(method = "cv",number = 10)
```

Hide

```
alphas<-c(0,0.01,0.001)
lambdas<-c(0.001,0.014)
params<-expand.grid(alpha=alphas,lambda=lambdas)
```

Hide

```
cv_fit<-train(rent.amount~.,data=train_data,
              trControl=train_control,
              tuneGrid=params,
              method="glmnet")
```

Performance metrics

- **Mean Square Error** Measures the average error between the predicted and original values. It is very sensitive to outliers, but it is good for making sure that our model does not have predictions that are too far out.
- **Mean Absolute Error** It is similar to the MSE with the difference that the MAE is robust with the predictions with outliers, so it is more complicated to determine if our model generates predictions.
- R^2 It measures the degree of fit between the predictions and the original value. It is measured from 0 to 1, the closer it is to 1, the greater the accumulation of the predictions with respect to the original value.

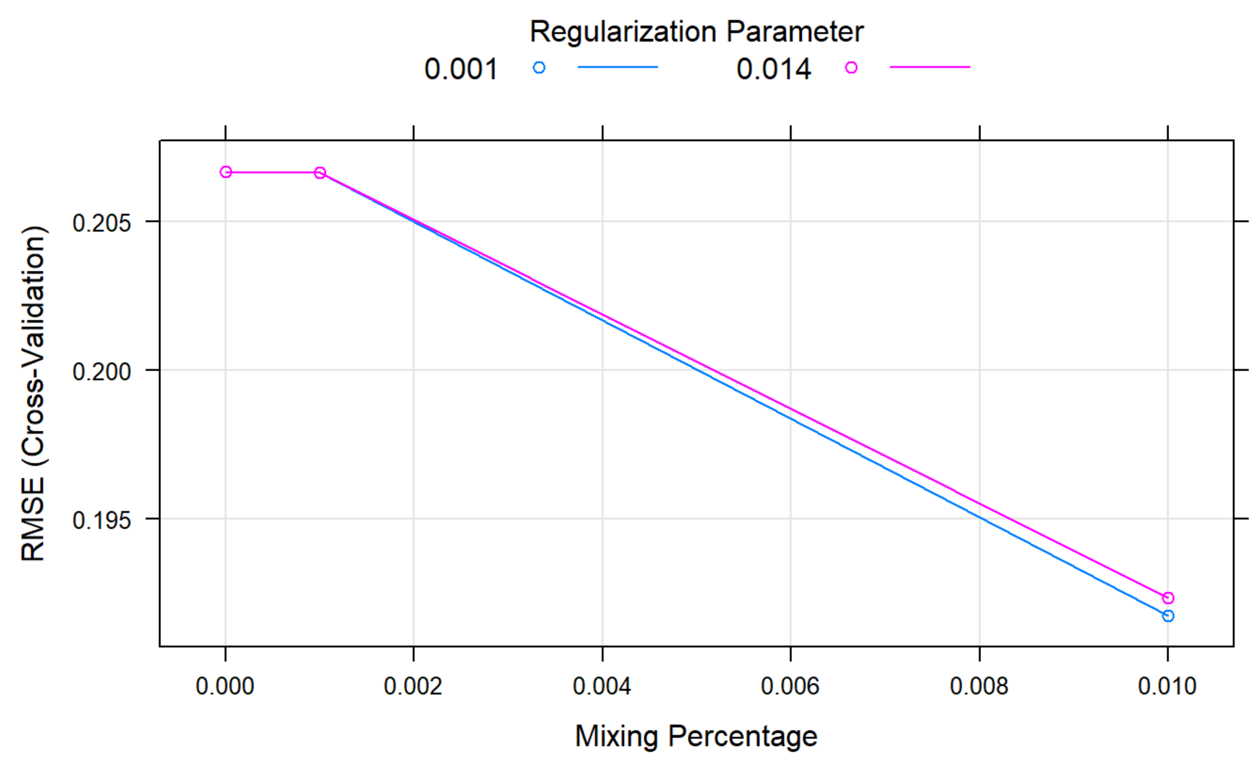
Hide

```
cv_fit
```

```
glmnet

4256 samples
   9 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3831, 3831, 3831, 3830, 3830, 3831, ...
Resampling results across tuning parameters:

  alpha  lambda  RMSE       Rsquared   MAE
  0.000  0.001   0.2066773  0.9325097  0.1516342
  0.000  0.014   0.2066773  0.9325097  0.1516342
  0.001  0.001   0.2066389  0.9325535  0.1515934
  0.001  0.014   0.2066389  0.9325535  0.1515934
  0.010  0.001   0.1917232  0.9391582  0.1172533
  0.010  0.014   0.1923295  0.9389158  0.1209572

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were alpha = 0.01 and lambda = 0.001.
```

Hide

```
plot(cv_fit)
```



Best Model

Hide

```
cv_fit$bestTune
```

	alpha <dbl>	lambda <dbl>
5	0.01	0.001

1 row