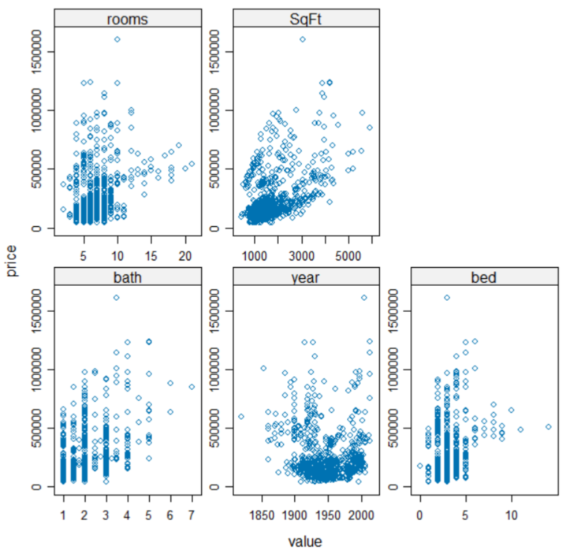
1. Load the dataset into R and remove the street information in order to anonymise the data. Explore the data with str and summary.

```
> setwd("C:/Users/alber/OneDrive/Documentos/UPV/2MU/CDA/Practicals/P5")
> data<-read.csv(file.choose(),header=TRUE, sep=';')
> data$street <- NULL
> head(data)
      zpid zipcode       city state year bath bed rooms SqFt  price
1   956068   35212 Birmingham    AL 1930  2.0   3     7 1732  40745
2   924224   35204 Birmingham    AL 1930  1.0   2     6 1115 205906
3   906733   35215 Birmingham    AL 1982  2.0   3    11 1355  98672
4   964007   35205 Birmingham    AL 1919  2.5   4     7 2876 325474
5 74504350   99801     Juneau    AK 1966  1.0   1     3  476 114726
6 81982160   85037    Phoenix    AZ 2006  3.0   3     5 1652 122241
> str(data)
'data.frame':   799 obs. of  10 variables:
 $ zpid   : int  956068 924224 906733 964007 74504350 81982160 7780105 7792424 7833403 56437339 ...
 $ zipcode: int  35212 35204 35215 35205 99801 85037 85021 85021 85018 2140 ...
 $ city   : chr  "Birmingham" "Birmingham" "Birmingham" "Birmingham" ...
 $ state  : chr  "AL" "AL" "AL" "AL" ...
 $ year   : int  1930 1930 1982 1919 1966 2006 1984 1974 1980 1894 ...
 $ bath   : num  2 1 2 2.5 1 3 5 2 2 3.5 ...
 $ bed    : int  3 2 3 4 1 3 5 2 2 4 ...
 $ rooms  : int  7 6 11 7 3 5 9 4 4 9 ...
 $ SqFt   : int  1732 1115 1355 2876 476 1652 3945 1625 1794 2294 ...
 $ price  : int  40745 205906 98672 325474 114726 122241 573258 239086 304824 885883 ...
> 
```

As we can see, the data is composed of 10 colums without the colum street, in represent the houses in different cities ,and the relevant information about the house such as the number of rooms , beds or price of the house.

2. Print a scatter plot of price versus each other variable so that you can see which variables are likely to be most important.



Do you see any interesting relationship between the price and the other variables?

The price is correlated with the amount of sqft, rooms and bedrooms, as well as baths . less so to the year variable

3.Randomly split the dataset into 75% train and 25% test.

```
> total_rows <- nrow(data)
> train_size <- round(0.75 * total_rows)
> test_size <- total_rows - train_size
> train_indices <- sample(1:total_rows, train_size)
> train_data <- data[train_indices, ]
> test_data <- data[-train_indices, ]
> nrow(data)
[1] 799
> nrow(train_data)
[1] 599
> nrow(test_data)
[1] 200
```

4.Fit several regression models to the training data but only using the numerical attributes:

1. linear model (using the lm function, which fits a linear model using ordinary least squares)
2. regression tree (CART) from the rpart package (set the parameter method to anova in order to produce a CART tree)
3. and a neural network from the nnet package (set the parameters skip and linout- numerical output- to TRUE and size-hidden units- to 12).

```
> # Fit a linear model using lm
> linear_model <- lm(price  ~ zpid+zipcode+year+bath+bed+rooms+SqFt, data = train_data)
>
> # Fit a regression tree (CART) using rpart
> cart_model <- rpart(price ~ zpid+zipcode+year+bath+bed+rooms+SqFt, data = train_data, method = "anova")
>
> # Fit a neural network using nnet
> neural_network <- nnet(price ~ zpid+zipcode+year+bath+bed+rooms+SqFt, data = train_data,
+ size = 12, skip = TRUE, linout = TRUE)
# weights:  116
initial  value 232484950008981248.000000
iter  10 value 10104983466720394.000000
iter  20 value 18560381622437.562500
final  value 17407385833685.863281
converged
```

View the models using the summary method and, additionally, the plot method for the CART tree. Which one is the less informative?

Linear Model

```
> summary(linear_model)

Call:
lm(formula = price ~ zpid + zipcode + year + bath + bed + rooms +
    SqFt, data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max
-407298  -97242  -32012   36785 1228717

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.055e+06  4.956e+05   8.182 1.72e-15 ***
zpid        -4.649e-04  2.664e-04  -1.745 0.081494 .
zipcode     -9.191e-01  2.841e-01  -3.235 0.001283 **
year        -2.043e+03  2.580e+02  -7.917 1.21e-14 ***
bath         6.960e+04  1.293e+04   5.384 1.06e-07 ***
bed         -3.229e+04  9.216e+03  -3.503 0.000494 ***
rooms       -1.587e+03  4.945e+03  -0.321 0.748376
SqFt         1.218e+02  1.524e+01   7.991 7.06e-15 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 171600 on 591 degrees of freedom
Multiple R-squared:  0.364,     Adjusted R-squared:  0.3565
F-statistic: 48.32 on 7 and 591 DF,  p-value: < 2.2e-16
```

CART MODEL:

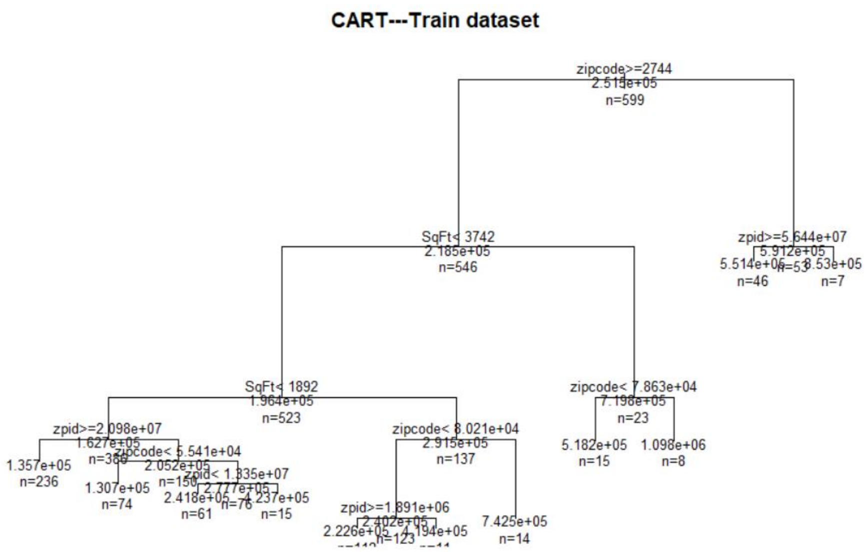
```
> summary(cart_model)
Call:
rpart(formula = price ~ zpid + zipcode + year + bath + bed +
    rooms + SqFt, data = train_data, method = "anova")
  n= 599

          CP nsplit rel error    xerror       xstd
1 0.24520639      0 1.0000000 1.0047874 0.12017030
2 0.22048992      1 0.7547936 0.8388935 0.12247745
3 0.08859897      2 0.5343037 0.5544858 0.08844989
4 0.06405018      4 0.3571058 0.4277611 0.08248701
5 0.02289663      5 0.2930556 0.3604313 0.05072787
6 0.02018837      7 0.2472623 0.3294661 0.04881245
7 0.01455777      8 0.2270739 0.3226233 0.04711552
8 0.01418171      9 0.2125162 0.3114777 0.04730024
9 0.01000000     10 0.1983345 0.2954513 0.04690760

Variable importance
zipcode    SqFt    year    zpid   rooms    bath     bed
     33      24      12       9       8       7       7

Node number 1: 599 observations,    complexity param=0.2452064
  mean=251458.4, MSE=4.569402e+10
  left son=2 (546 obs) right son=3 (53 obs)
  Primary splits:
      zipcode < 2743.5   to the right, improve=0.24520640, (0 missing)
      SqFt    < 3742.5   to the left,  improve=0.20712680, (0 missing)
      bath    < 3.375    to the left,  improve=0.20409950, (0 missing)
      rooms   < 9.5      to the left,  improve=0.09780811, (0 missing)
      bed     < 4.5      to the left,  improve=0.08081959, (0 missing)
  Surrogate splits:
      bed   < 5.5      to the left,  agree=0.938, adj=0.302, (0 split)
      year  < 1886.5   to the right, agree=0.935, adj=0.264, (0 split)
```

The summary function return in text all of the splitted nodes, as we can see the text description is much more detailed that the plot function. Also, the text description returns errors as well as variable importance of each table, we can see that zipcode and SqFt are the most important variables in our prediction model



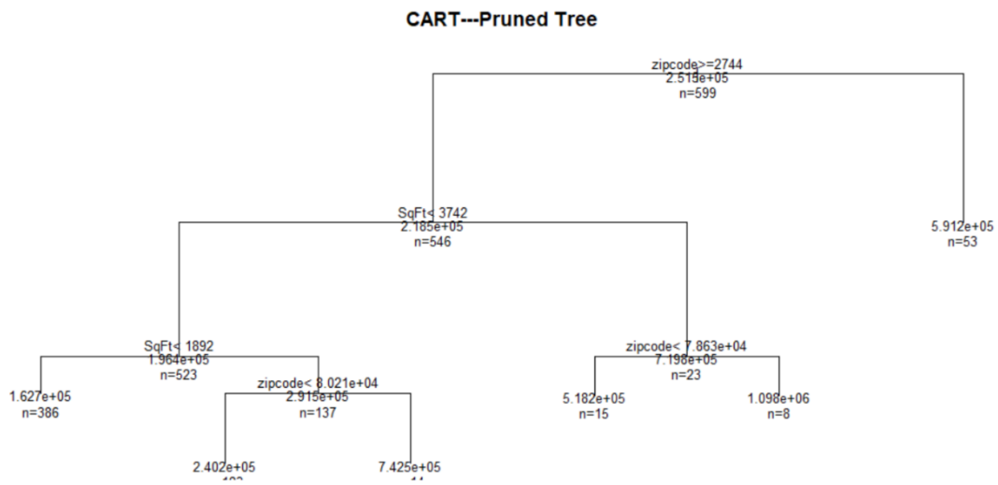
Neural Network Model

```
> summary(neural_network)
a 7-12-1 network with 116 weights
options were - skip-layer connections  linear output units
    b->h1    i1->h1    i2->h1    i3->h1    i4->h1    i5->h1    i6->h1    i7->h1
    -0.60      0.36      0.00      0.29     -0.19     -0.25      0.23     -0.40
    b->h2    i1->h2    i2->h2    i3->h2    i4->h2    i5->h2    i6->h2    i7->h2
     0.44      0.56     -0.37     -0.01      0.57     -0.48     -0.44      0.26
    b->h3    i1->h3    i2->h3    i3->h3    i4->h3    i5->h3    i6->h3    i7->h3
     0.42     -0.01      0.01     -0.59     -0.09     -0.68      0.50      0.13
    b->h4    i1->h4    i2->h4    i3->h4    i4->h4    i5->h4    i6->h4    i7->h4
    -0.30     -0.60      0.46      0.52     -0.40      0.24     -0.54      0.30
    b->h5    i1->h5    i2->h5    i3->h5    i4->h5    i5->h5    i6->h5    i7->h5
    -0.08     -0.01     -0.09      0.22     -0.11     -0.54     -0.28      0.15
    b->h6    i1->h6    i2->h6    i3->h6    i4->h6    i5->h6    i6->h6    i7->h6
    -0.37     -0.17      0.70      0.69     -0.53     -0.49     -0.21      0.36
    b->h7    i1->h7    i2->h7    i3->h7    i4->h7    i5->h7    i6->h7    i7->h7
     0.46      0.03     -0.11      0.21     -0.52      0.67      0.68     -0.67
    b->h8    i1->h8    i2->h8    i3->h8    i4->h8    i5->h8    i6->h8    i7->h8
    -0.14      0.65     -0.24     -0.07      0.52      0.30      0.48     -0.52
    b->h9    i1->h9    i2->h9    i3->h9    i4->h9    i5->h9    i6->h9    i7->h9
    -0.34      0.14     -0.62     -0.57      0.44      0.28     -0.18      0.41
   b->h10   i1->h10   i2->h10   i3->h10   i4->h10   i5->h10   i6->h10   i7->h10
     0.24      0.30      0.34     -0.53     -0.19      0.28      0.39      0.51
   b->h11   i1->h11   i2->h11   i3->h11   i4->h11   i5->h11   i6->h11   i7->h11
    -0.34      0.26     -0.10     -0.09      0.62     -0.64     -0.18     -0.33
   b->h12   i1->h12   i2->h12   i3->h12   i4->h12   i5->h12   i6->h12   i7->h12
    -0.02      0.44      0.11      0.60      0.30      0.06      0.49      0.58
     b->o     h1->o     h2->o     h3->o     h4->o     h5->o     h6->o     h7->o     h8->o     h9->o    h10->o    h11->o    h12->o     i1->o
450520.35 450520.79 450520.40     -0.63     -0.19     -0.16     -0.42 450521.19 450520.51 450520.48 450520.10 450520.89 450520.77      0.00
    i2->o     i3->o     i4->o     i5->o     i6->o     i7->o
    -0.92  -2042.86  69595.61 -32286.65  -1587.09    121.76
```

Prune the regression tree using the prune function and setting the cp parameter to the value you consider is a good balance between complexity and performance (the plotcp function plots tree sizes and relative errors for different values of the complexity parameter). Visualise the new tree.

Pruned Tree

```
> pruned_model <- prune(cart_model, cp = 0.04)
> plot(pruned_model, main="CART---Pruned Tree")
> text(pruned_model,use.n=TRUE, all=TRUE, cex=.8)
>
```



7.Compare how each method works using the Root Mean Square Error (RMSE) and the Mean Absolute Error (MAE) for the training and test data sets. Which model performs better for the training data? And for the test data?

First, we are going to predict the values of the training set.

```
> #Predict
> linear_train_predictions <- predict(linear_model, newdata = train_data)
> cart_train_predictions <- predict(cart_model, newdata = train_data)
> neural_train_predictions <- predict(neural_network, newdata = train_data)
```

We are going to create two functions to calculate the Root Mean Square Error (RMSE) and the Mean Absolute Error (MAE)

```
> calculate_rmse <- function(actual, predicted) {
+   sqrt(mean((actual - predicted)^2))
+ }
>
> calculate_mae <- function(actual, predicted) {
+   mean(abs(actual - predicted))
+ }
```

The results are the following:

```
> #Linear Model
> calculate_rmse(train_data$price, linear_train_predictions)
[1] 170472.1
> calculate_mae(train_data$price, linear_train_predictions)
[1] 114416.5
>
> #CART Model
> calculate_rmse(train_data$price, cart_train_predictions)
[1] 95198.21
> calculate_mae(train_data$price, cart_train_predictions)
[1] 63947
>
> #NN Model
> calculate_rmse(train_data$price, neural_train_predictions)
[1] 170472.1
> calculate_mae(train_data$price, neural_train_predictions)
[1] 114416.5
```

Both the linear and the Neural network model return the same RMSE and MAE , being the CART model the best at predicting the training data.

```
> linear_test_predictions <- predict(linear_model, newdata = test_data)
> cart_test_predictions <- predict(cart_model, newdata = test_data)
> neural_test_predictions <- predict(neural_network, newdata = test_data)
>
> #Linear Model
> calculate_rmse(test_data$price, linear_test_predictions)
[1] 158242.4
> calculate_mae(test_data$price, linear_test_predictions)
[1] 113225.4
>
> #CART Model
> calculate_rmse(test_data$price, cart_test_predictions)
[1] 108354.7
> calculate_mae(test_data$price, cart_test_predictions)
[1] 70295.01
>
> #NN Model
> calculate_rmse(test_data$price, neural_test_predictions)
[1] 158242.4
> calculate_mae(test_data$price, neural_test_predictions)
[1] 113225.4
```

8.Can you improve the results by changing the parameters or trying other methods?

Pruned Model (c=0.04)

```
> pruned_model_predictions <- predict(pruned_model, newdata = test_data)
> calculate_rmse(test_data$price, pruned_model_predictions)
[1] 129009.1
> calculate_mae(test_data$price, pruned_model_predictions)
[1] 92056.35
```

Random Forest

```
> rf_model <- randomForest(price ~ zpid+zipcode+year+bath+bed+rooms+SqFt, data = train_data)
> rf_test_predictions <- predict(rf_model, newdata = test_data)
> calculate_rmse(test_data$price, rf_test_predictions)
[1] 72726.97
> calculate_mae(test_data$price, rf_test_predictions)
[1] 46329.49
```

Ctree (party library)

```
> ctree_model <- ctree(price ~ zpid+zipcode+year+bath+bed+rooms+SqFt, data = train_data)
> ctree_test_predictions <- predict(ctree_model, newdata = test_data)
> calculate_rmse(test_data$price, ctree_test_predictions)
[1] 132672.7
> calculate_mae(test_data$price, ctree_test_predictions)
[1] 90304.71
```

Support Vector Regression

```
> library(e1071)
> svr_model <- svm(price ~ zpid+zipcode+year+bath+bed+rooms+SqFt, data = train_data)
> svr_test_predictions <- predict(svr_model, newdata = test_data)
> calculate_rmse(test_data$price, svr_test_predictions)
[1] 91492.01
> calculate_mae(test_data$price, svr_test_predictions)
[1] 58907.79
```