Harvard Data Science Capstone Project - DIY: Boston housing

Jose Angulo

4/21/2024

1. Introduction

We will be using the Boston housing dataset which is one of the most popular datasets to be used for machine learning.

1.a. Structure of the Report

Section 1 - Introduction and Objective

Section 2 - Data Preparation

Section 3 - Exploratory Data Analysis

Section 4- Development of the models

Section 5 - Conclusion

Section 6 - Limitations

Section 7 - References

1.b. Objective

The objective is to build a machine learning model to predict the Median value (MEDV) of owner occupied homes in Boston based on the available features.

1.c. Overview

The data set has been downloaded from Kaggle.

Since Kaggle does not allow us to download the files directly, I have downloaded the file to my github and here is the link to the file:

https://github.com/rrao2511/CYO-Harvard-Capstone-Project/raw/main/housing.csv

The original Boston housing dataset contains 506 samples and 14 variables.

For the purpose of this report we will be looking only at a subset of the original Boston housing dataset.

Our dataset contains 489 samples and 4 variables which are explained below:

MEDV – Median Value of Owner occupied homes

RM - Average number of rooms per dwelling

LSTAT - % lower status of population

PT RATIO - Pupil teacher ratio by town

The goal of our analysis is to select the best prediction model which can predict the Median value of owner occupied homes in Boston.

1.d. Approach

The steps we will take for this project are:

Look at the data structure

Data Preparation and Cleaning

Exploring the data

Development of models

Results

Recommendation of the Model

Limitations of the Model

Start of Script

2. Data Preparation and Data Preprocessing

Dataset Source:

Kaggle is an online platform for data scientists and machine learning students.

This particular dataset- Boston housing has been downloaded from Kaggle.

First step - download the Packages needed for this analysis and load the libraries.

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org")
if(!require(corrplot)) install.packages("corrplot", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: corrplot
## corrplot 0.92 loaded
```

```
if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: randomForest
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
```

```
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
if(!require(rpart)) install.packages("rpart", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: rpart
```

```
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
if(!require(rpart.plot)) install.packages("rpart.plot", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: rpart.plot
```

```
if(!require(e1071)) install.packages("e1071", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: e1071
```

```
library(tidyverse)
library(ggplot2)
library(caret)
library(dplyr)
library(corrplot)
library(randomForest)
library(rpart)
library(rpart.plot)
library(e1071)
```

Download the dataset

Since Kaggle does not allow us to download the files directly, have downloaded the file to my github and here is the link to the file:

https://github.com/Jarkeld/Harvard-Capstone-project---Boston-Houding-dataset---/blob/823681e9c5c133d1b039fde8ac9cc
housing.csv

Reading the data from the csv file

```
boston_housing<-read.csv("https://raw.githubusercontent.com/Jarkeld/Harvard-Capstone-project---Boston-H
```

For the purpose of this analysis we are looking at a subset of the Boston housing set

First lets look at the data set - checking the dimension.

This dataset has 489 observations and 4 columns. This is a subset of the original Kaggle dataset.

There are 4 columns and the details of the column are shown below.We will be using all the 4 columns for our analysis.

We will also look at the structure of the dataset, head and the summary.

Explanation of Column names and details

RM - Average number of rooms per dwelling

LSTAT - % lower status of population

PT Ratio - Pupil teacher ratio by town

MEDV - Median Value of owner occupied homes in $1000s.

```
dim(boston_housing)
```

```
## [1] 489   4
```

```
str(boston_housing)
```

```
## 'data.frame':    489 obs. of  4 variables:
##  $ RM     : num  6.58 6.42 7.18 7 7.15 ...
##  $ LSTAT  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ PTRATIO: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ MEDV   : num  504000 453600 728700 701400 760200 ...
```

```
head(boston_housing)
```

```
##      RM LSTAT PTRATIO   MEDV
## 1 6.575  4.98    15.3 504000
## 2 6.421  9.14    17.8 453600
## 3 7.185  4.03    17.8 728700
## 4 6.998  2.94    18.7 701400
## 5 7.147  5.33    18.7 760200
## 6 6.430  5.21    18.7 602700
```

```
summary(boston_housing)
```

```
##        RM            LSTAT          PTRATIO          MEDV
##  Min.   :3.561   Min.   : 1.98   Min.   :12.60   Min.   : 105000
##  1st Qu.:5.880   1st Qu.: 7.37   1st Qu.:17.40   1st Qu.: 350700
##  Median :6.185   Median :11.69   Median :19.10   Median : 438900
##  Mean   :6.240   Mean   :12.94   Mean   :18.52   Mean   : 454343
##  3rd Qu.:6.575   3rd Qu.:17.12   3rd Qu.:20.20   3rd Qu.: 518700
##  Max.   :8.398   Max.   :37.97   Max.   :22.00   Max.   :1024800
```

Cleaning up the data

Since this dataset is already clean, data cleaning was not needed and it could be used directly for analysis.

Check to see if there are duplicate values and also any missing values.

```
sum(duplicated(boston_housing))
```

```
## [1] 0
```

```
sum(is.na(boston_housing))
```

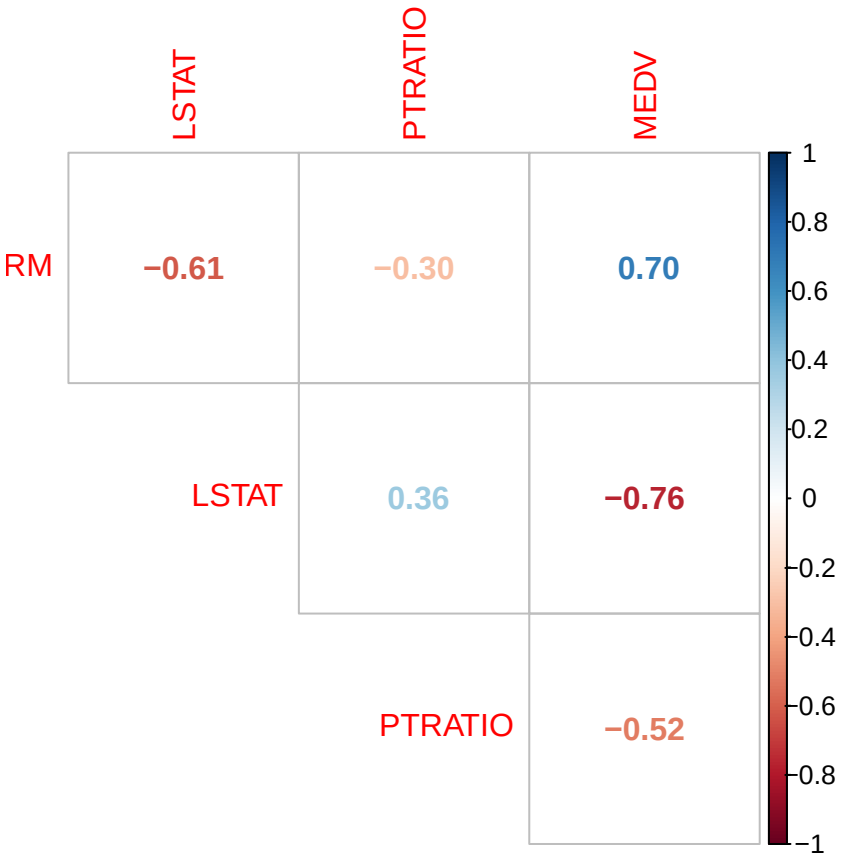
```
## [1] 0
```

3. Exploratory Data Analysis using Data Visualization

Before we start building the model we will understand the data set by doing some Exploratory Data Analysis.

Check the correlation between variables by plotting a correlation graph

```
corrplot(cor(boston_housing), method = "number", type = "upper", diag = FALSE)
```



From correlation matrix, we observe that:

Both RM and LSTAT have a strong correlation with MEDV.

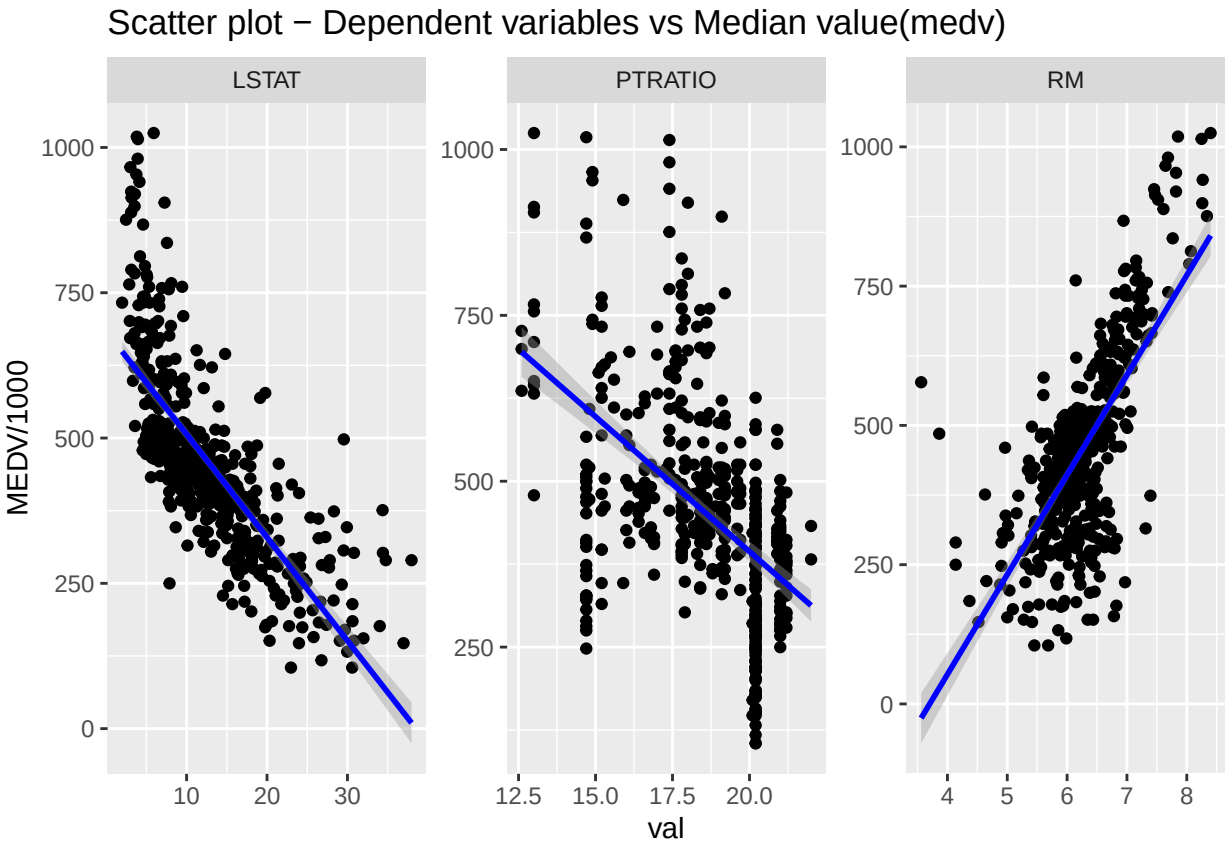
Median value of owner-occupied homes (in 1000$) increases as average number of rooms per dwelling increases and it decreases if percent of lower status population in the area increases.

PT Ratio has a positive correlation with LSTAT

Next lets look at Scatter plots to show relationship between Median value and variables

```
boston_housing%>%
  gather(key, val,-MEDV) %>%
  ggplot(aes(x = val, y = MEDV/1000))+
  geom_point()+
  stat_smooth(method = "lm", se = TRUE, col ="blue") +
  facet_wrap(~key, scales = "free")+
  theme_grey()+
  ggtitle("Scatter plot - Dependent variables vs Median value(medv)")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



From the plots we see that RM and LSTAT have a strong correlation with Median value.

The Median value prices increases as the RM value increases linearly.

The Median value prices tend to decrease with an increase in LSTAT

4. Developing the Models

We will use three different models for this project:

Decision trees, Random Forest and Support Vector Machine.

We will evaluate the models using Root Mean Squared Error (RMSE).

The model that best fits the data will be selected.

First we need to split the data into train sets and test sets:

Data is split into train and test sets - 80:20

```
set.seed(123)
bh_index<- sample(nrow(boston_housing),nrow(boston_housing)*.80)
bh_train<- boston_housing[bh_index,]
bh_test<- boston_housing[-bh_index,]
```

Model Development

Model 1 - Decision trees

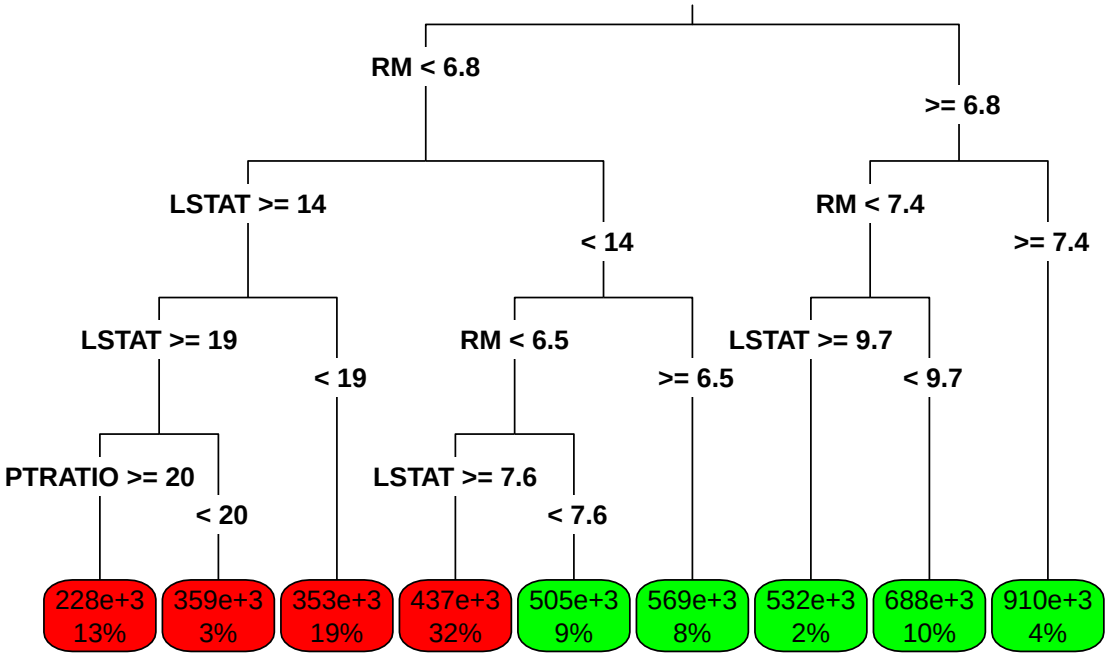
Steps - we will create the model using Decision trees on the

train set, plot the decision tree, validate on the test set

and finally calculate the RMSE.

```
bhtree.fit<- rpart(MEDV~., data= bh_train)

rpart.plot(bhtree.fit, type = 3, box.palette = c("red", "green"), fallen.leaves = TRUE)
```




```
tree.pred<- predict(bhtree.fit, newdata = bh_test)

tree.rmse<- sqrt(mean((bh_test$MEDV- tree.pred)^2))

cat("Decision Tree RMSE", round(tree.rmse,2),"\n")
```

```
## Decision Tree RMSE 79926.41
```

Model 2 - Random Forest

Steps - we will create the model using Random forest on train

set, validate on the test set and finally calculate the

RMSE.

```
rf.fit<- randomForest(MEDV~., data= bh_train, ntree= 500, mtry = 3)

rf.pred<- predict(rf.fit, newdata = bh_test)

rf.rmse<- sqrt(mean((bh_test$MEDV - rf.pred)^2))


cat("Random Forest RMSE", round(rf.rmse,2), "\n")
```

```
## Random Forest RMSE 71804.79
```

Model 3 - Support Vector Machines (SVM)

Steps - we will create the model using SVM on train

set, validate on the test set and finally calculate the

RMSE.

```
svm.fit<- svm(MEDV~., data = bh_train, kernel= "linear", cost =1)

svm.pred<- predict(svm.fit, newdata = bh_test)

svm.rmse<- sqrt(mean((svm.pred- bh_test$MEDV)^2))


cat("SVM RMSE:", svm.rmse, "\n")
```

```
## SVM RMSE: 93155.68
```

5. Conclusion

Create a table for the RMSE values of Decision trees, Random Forest and SVM

```
results_table<- data.frame(Model = c("Decision Tree", "Random Forest", "SVM"),
                           RMSE= c(tree.rmse,rf.rmse,svm.rmse ))
```

Based on the above results here are our observations:

a) The random forest model has the lowest RMSE value of 71805,indicating that it may be the best model for predicting the median value of owner-occupied homes in Boston.

b) Whereas the Decision tree and the SVM models have higher RMSE values of 79926 and 93156 respectively, indicating that they may not be the best choice for predicting the median value of owner occupied homes.

6. Limitations of the Model

We need to be cautious we need to be cautious when drawing conclusions based on RMSE values alone, as there may be other factors to consider such as model complexity, interpretability, and computational efficiency

Random forest models can be further improved with hyperparameters tuning.

But on account of shortage of time this was not attempted.

Similarly the SVM model could be tuned further by changing the parameters and the kernel function.

But on account of shortage of time this was not attempted.

========== END OF FILE ==========