Boston House Price Prediction

Maher Daoud

2/6/2022

Overview

The problem on hand is to predict the housing prices of a town or a suburb based on the features of the locality provided to us. In the process, we need to identify the most important features in the dataset. We need to employ techniques of data preprocessing and build a linear regression model that predicts the prices for us.

Data Information

Each record in the database describes a Boston suburb or town. The data was drawn from the Boston Standard Metropolitan Statistical Area (SMSA) in 1970. Detailed attribute information can be found below-

Attribute Information (in order):

- **CRIM:** per capita crime rate by town
- **ZN:** proportion of residential land zoned for lots over 25,000 sq.ft.
- **INDUS:** proportion of non-retail business acres per town
- **CHAS:** Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
- **NOX:** nitric oxides concentration (parts per 10 million)
- **RM:** average number of rooms per dwelling
- **AGE:** proportion of owner-occupied units built prior to 1940
- **DIS:** weighted distances to five Boston employment centres
- **RAD:** index of accessibility to radial highways
- **TAX:** full-value property-tax rate per 10,000 dollars
- **PTRATIO:** pupil-teacher ratio by town
- **LSTAT:** %lower status of the population
- **MEDV:** Median value of owner-occupied homes in 1000 dollars.

Import the Data

```
# In case you face any problem with downloading the dataset
# please use this link to download the dataset
# https://github.com/maherdaoud/Boston-House-Price-Prediction/blob/main/Boston.csv
# then use the following code to read it
# df <- read.csv("C:\download-path\Boston.csv)

# Import the data from Github
urlfile <-
  "https://raw.githubusercontent.com/maherdaoud/Boston-House-Price-Prediction/main/Boston.csv"
df <- data.frame(read_csv(url(urlfile), show_col_types = F), stringsAsFactors = F)
```

Row Data of Boston House Price

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	LSTAT	MEDV
0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	4.98	24.0
0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	9.14	21.6
0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	4.03	34.7
0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	2.94	33.4
0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	5.33	36.2
0.02985	0.0	2.18	0	0.458	6.430	58.7	6.0622	3	222	18.7	5.21	28.7
0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	12.43	22.9
0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311	15.2	19.15	27.1
0.21124	12.5	7.87	0	0.524	5.631	100.0	6.0821	5	311	15.2	29.93	16.5
0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311	15.2	17.10	18.9

Observations:

- The price of the house indicated by the variable MEDV is the target variable and the rest are the independent variables based on which we will predict house price.

Get information about the dataset using the str() method

```
# Print the Structure of the dataset
str(df)
```

```
## 'data.frame':    506 obs. of  13 variables:
##  $ CRIM   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ ZN     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ INDUS  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ CHAS   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ NOX    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ RM     : num  6.58 6.42 7.18 7 7.15 ...
##  $ AGE    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ DIS    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ RAD    : num  1 2 2 3 3 3 5 5 5 5 ...
##  $ TAX    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ PTRATIO: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ LSTAT  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ MEDV   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

```
# compute how many NA we have
colSums(is.na(df))
```

```
##    CRIM      ZN   INDUS    CHAS     NOX      RM     AGE     DIS     RAD     TAX
##       0       0       0       0       0       0       0       0       0       0
## PTRATIO   LSTAT    MEDV
##       0       0       0
```

Observations:

- There are a total of 506 non-null observations in each of the columns. This indicates that there are no missing values in the data.
- Every column in this dataset is numeric in nature.

Analysis

Let’s now check the summary statistics of this dataset

```
# use stat.desc function to find summary statistics
# for all numeric columns in the data set
round(pastecs::stat.desc(df, norm = F),3)
```

```
##                    CRIM        ZN     INDUS    CHAS     NOX        RM       AGE
## nbr.val         506.000   506.000   506.000 506.000 506.000   506.000   506.000
## nbr.null          0.000   372.000     0.000 471.000   0.000     0.000     0.000
## nbr.na            0.000     0.000     0.000   0.000   0.000     0.000     0.000
## min               0.006     0.000     0.460   0.000   0.385     3.561     2.900
## max              88.976   100.000    27.740   1.000   0.871     8.780   100.000
## range            88.970   100.000    27.280   1.000   0.486     5.219    97.100
## sum            1828.443  5750.000  5635.210  35.000 280.676  3180.025 34698.900
## median            0.257     0.000     9.690   0.000   0.538     6.208    77.500
## mean              3.614    11.364    11.137   0.069   0.555     6.285    68.575
## SE.mean           0.382     1.037     0.305   0.011   0.005     0.031     1.251
## CI.mean.0.95      0.751     2.037     0.599   0.022   0.010     0.061     2.459
## var              73.987   543.937    47.064   0.065   0.013     0.494   792.358
## std.dev           8.602    23.322     6.860   0.254   0.116     0.703    28.149
## coef.var          2.380     2.052     0.616   3.672   0.209     0.112     0.410
##                     DIS       RAD         TAX  PTRATIO    LSTAT      MEDV
## nbr.val         506.000   506.000     506.000  506.000  506.000   506.000
## nbr.null          0.000     0.000       0.000    0.000    0.000     0.000
## nbr.na            0.000     0.000       0.000    0.000    0.000     0.000
## min               1.130     1.000     187.000   12.600    1.730     5.000
## max              12.126    24.000     711.000   22.000   37.970    50.000
## range            10.997    23.000     524.000    9.400   36.240    45.000
## sum            1920.292  4832.000  206568.000 9338.500 6402.450 11401.600
## median            3.207     5.000     330.000   19.050   11.360    21.200
## mean              3.795     9.549     408.237   18.456   12.653    22.533
## SE.mean           0.094     0.387       7.492    0.096    0.317     0.409
## CI.mean.0.95      0.184     0.760      14.720    0.189    0.624     0.803
## var               4.434    75.816   28404.759    4.687   50.995    84.587
## std.dev           2.106     8.707     168.537    2.165    7.141     9.197
## coef.var          0.555     0.912       0.413    0.117    0.564     0.408
```

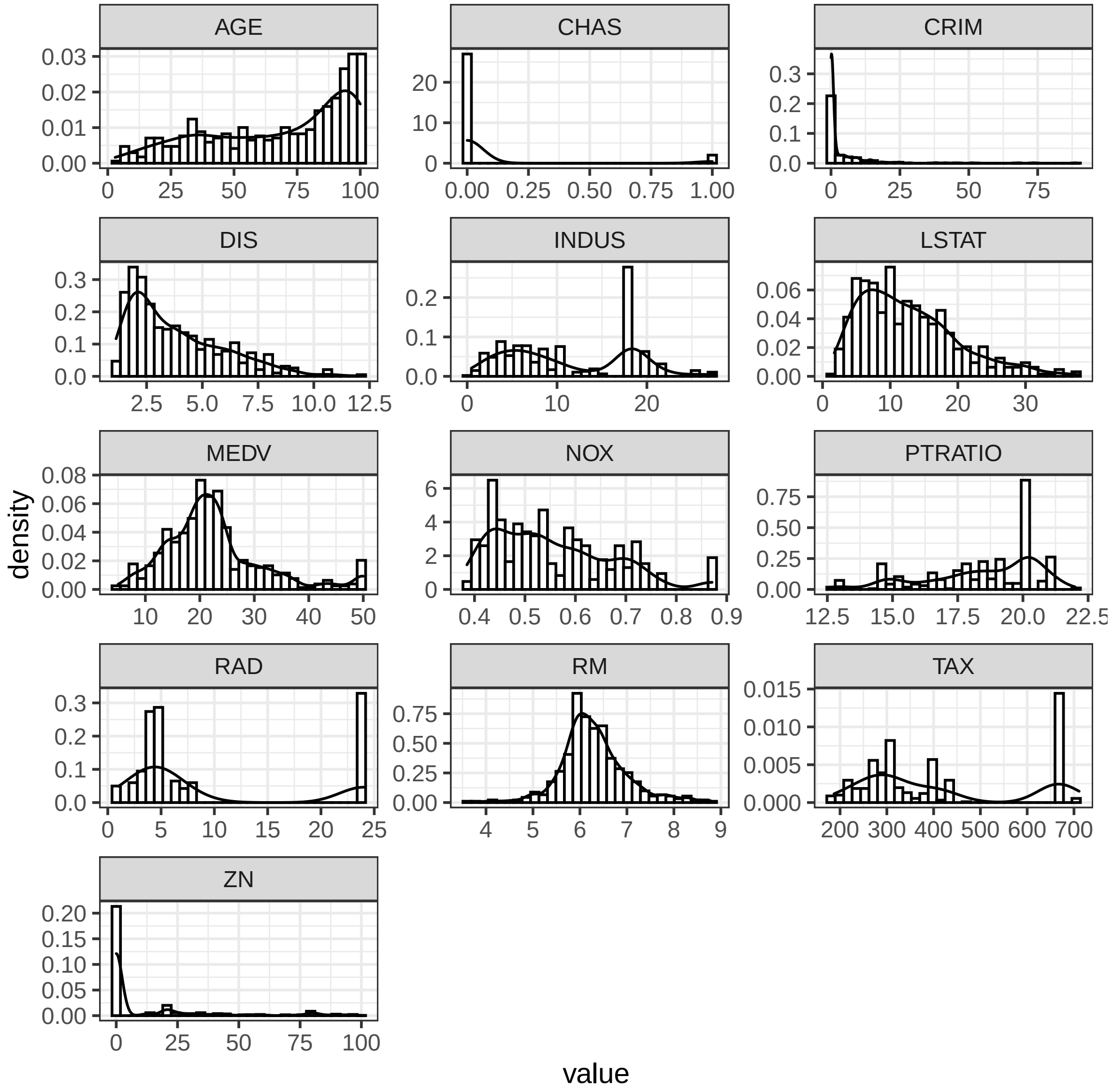
Observations:

- The **50th percentile of ZN** (proportion of residential land zoned for lots over 25,000 sq.ft.) **is 0**. This indicates that at least half the residential plots are under 25,000 sq. ft in area.
- The **75th percentile of CHAS** (Charles River dummy variable) **is 0**. It indicates that the vast majority of these houses are away from the Charles river.
- The **mean house price** is approx. **USD 22,500**, whereas **the median of the house prices** is approx. **USD 21,200**. This indicates that the price distribution is only slightly skewed towards the right side.

Before performing the modeling, it is important to check the univariate distribution of the variables.

Univariate Analysis

```
# Gather all variables in two columns, one for the variable name and one for value using
# gather function
# Plot the distribution fro each variable using geom_histogram
gather(df, cols, value) %>%
  ggplot(aes(x=value)) +
  geom_histogram(aes(y = ..density..), colour = 1, fill = "white") +
  geom_density() +
  facet_wrap(.~cols, scales = "free", nrow = 5) +
  theme_bw()
```



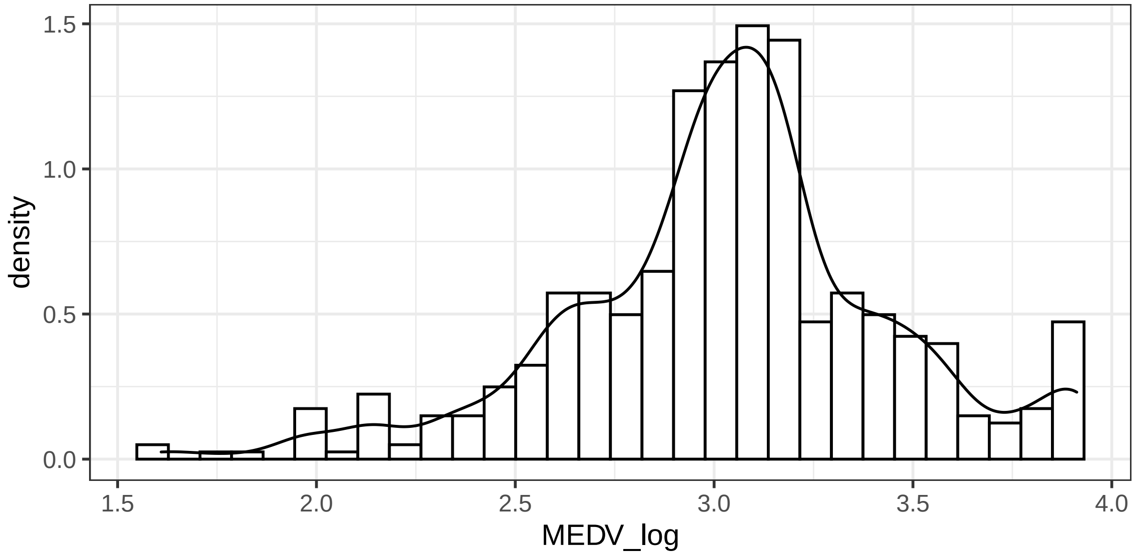
Observations:

- **The variables CRIM and ZN are positively skewed.** This suggests that most of the areas have lower crime rates and most residential plots are under the area of 25,000 sq. ft.
- **The variable CHAS, with only 2 possible values 0 and 1, follows a binomial distribution**, and the majority of the houses are away from Charles river (CHAS = 0).
- The distribution of the variable AGE suggests that many of the owner-occupied houses were built before 1940.
- **The variable DIS** (average distances to five Boston employment centers) **has a nearly exponential distribution**, which indicates that most of the houses are closer to these employment centers.
- **The variables TAX and RAD have a bimodal distribution.**, indicating that the tax rate is possibly higher for some properties which have a high index of accessibility to radial highways.

- The dependent variable MEDV seems to be slightly right skewed.

As the dependent variable is sightly skewed, we will apply a **log transformation on the 'MEDV' column** and check the distribution of the transformed column.

```
# Calculate the log of MEDV
df$MEDV_log <- log(df$MEDV)
# Plot the distribution
ggplot(df, aes(x=MEDV_log)) +
  geom_histogram(aes(y = ..density..), colour = 1, fill = "white") +
  geom_density() +
  theme_bw()
```



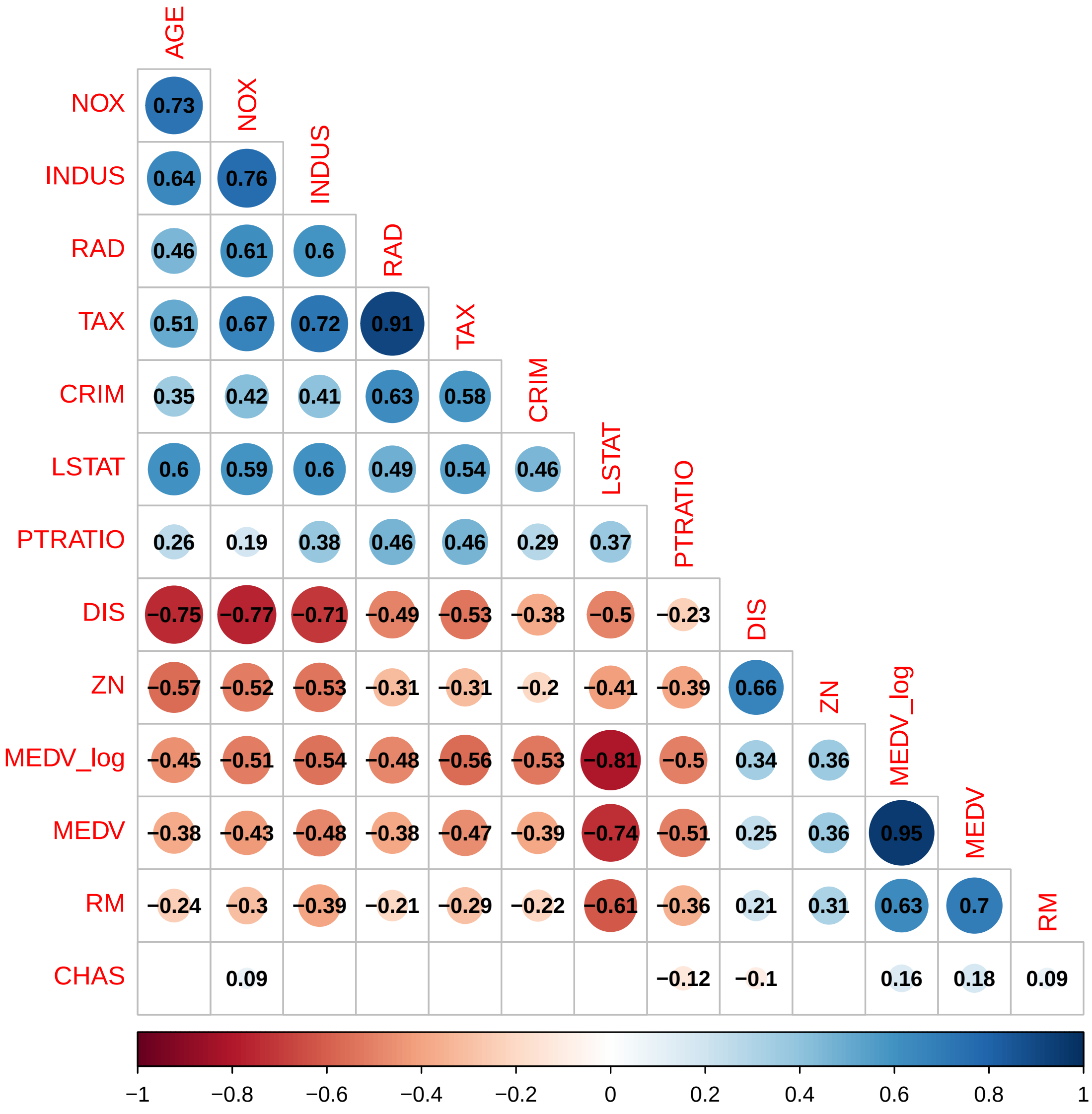
Observations:

- The log-transformed variable (**MEDV_log**) appears to have a **approx. normal distribution and with less skew**, and hence we can proceed.

Before creating the linear regression model, it is important to check the bivariate relationship between the variables. Let's check the same using the heatmap and scatterplot.

Bivariate Analysis

Let's check the correlation using the heatmap



Observations:

- **Significant correlations** are present between **NOX and INDUS** (0.76) - likely because areas with a higher proportion of non-retail industries are likely contributing to Nitric Oxide air pollution
- The variable **DIS has a strong negative correlation with INDUS (-0.71), NOX (-0.77) and AGE (-0.75)**, which are all significantly positively correlated with each other as well. An explanation

for this could be that areas closer to the center of the Boston city/metropolitan area, contain the oldest buildings and factories of importance, and their distance from the five employment centers in the heart of the city is also consequently small.
- Features **RAD and TAX are very strongly correlated (0.91)**.
- **INDUS and TAX** are also significantly correlated (0.70).
- **RM shows a significant positive correlation with MEDV**, likely since the higher the number of rooms per dwelling the more expensive the house, while **LSTAT shows a strong negative linear relationship with MEDV**, showing the likelihood of houses in areas with a higher percentage of lower-status population (poor education, laborers and unskilled employment) to be less expensive.

Linear Model Building - Approach

1. Data preparation
2. Partition the data into train and test set
3. Build model on the train data
4. Cross-validating the model
5. Test the data on test set

Split the dataset Let's split the data into the dependent and independent variables and further split it into train and test set in a ratio of 70:30 for train and test set.

```
# set seed to fix number where be able to have same random set everytime
set.seed(100)
# split using createDataPartition function where train data has 70% of the data
# and test data has 30% of the data
trainIndex <- createDataPartition(df$MEDV_log, p = .7,
                                  list = FALSE,
                                  times = 1)

# Drop MEDV and we will use the log tranformation
df$MEDV <- NULL

# Create training and testing dataset
train <- df[trainIndex,]
test <- df[-trainIndex,]
```

Next, we will check the multicollinearity in the train dataset.

Check for Multicollinearity

- **Multicollinearity** occurs when predictor variables in a regression model are correlated. This correlation is a problem because predictor variables should be independent. If the correlation between variables is high, it can cause problems when we fit the model and interpret the results. When we have multicollinearity in the linear model, the coefficients that the model suggests are unreliable.
- There are different ways of detecting (or testing) multi-collinearity, one such way is Variation Inflation Factor.
- **Variance Inflation factor**: Variance inflation factors measures the inflation in the variances of the regression parameter estimates due to collinearity that exist among the predictors. It is a measure of how much the variance of the estimated regression coefficient Bk is "inflated" by the existence of correlation among the predictor variables in the model.

- General Rule of thumb: If VIF is 1 then there is no correlation among the kth predictor and the remaining predictor variables, and hence the variance of B^k is not inflated at all. Whereas if **VIF exceeds 5 or is close to exceeding 5, we say there is moderate VIF and if it is 10 or exceeding 10, it shows signs of high multi-collinearity.**

```
## use vif function to compute Variance Inflation Factor and test for multicollinearity
check_vif_df <- train
check_vif_df$MEDV_log <- NULL
knitr::kable(usdm::vif(check_vif_df), align = "c") %>%
  kableExtra::kable_minimal(full_width = T, position = "center")
```

Variables	VIF
CRIM	1.916291
ZN	2.347812
INDUS	3.628839
CHAS	1.070909
NOX	4.630511
RM	1.855414
AGE	3.369406
DIS	4.010627
RAD	7.341295
TAX	8.380978
PTRATIO	1.924832
LSTAT	3.002392

- There are two variables with a high VIF - RAD and TAX. Let's remove TAX as it has the highest VIF values and check the multicollinearity again.

Dropping the column 'TAX' from the training data and checking if multicollinearity is removed

Variables	VIF
CRIM	1.915968
ZN	2.167529
INDUS	3.028009
CHAS	1.058080
NOX	4.594750
RM	1.850149
AGE	3.363741
DIS	4.008806
RAD	2.791856
PTRATIO	1.906073
LSTAT	3.002289

Now, we will create the linear regression model as the VIF is less than 5 for all the independent variables, and we can assume that multicollinearity has been removed between the variables.

```


# Remove Tax
train$TAX <- NULL
# create the model
model1 <- lm(MEDV_log ~ ., data = train)
# print summary of the model
summary(model1)
```

Creating linear regression model using lm function

```
##
## Call:
## lm(formula = MEDV_log ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.67645 -0.10927 -0.00364  0.10152  0.81259
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.1924508  0.2342219  17.899  < 2e-16 ***
## CRIM        -0.0091366  0.0017984  -5.080 6.19e-07 ***
## ZN           0.0004043  0.0006645   0.608 0.543346
## INDUS       -0.0031998  0.0025763  -1.242 0.215083
## CHAS         0.1086464  0.0420687   2.583 0.010219 *
## NOX         -0.6281737  0.1890124  -3.323 0.000985 ***
## RM           0.0718205  0.0194037   3.701 0.000250 ***
## AGE          0.0008953  0.0006715   1.333 0.183362
## DIS         -0.0382207  0.0098885  -3.865 0.000133 ***
## RAD          0.0029747  0.0019776   1.504 0.133441
## PTRATIO     -0.0396514  0.0063890  -6.206 1.56e-09 ***
## LSTAT       -0.0331677  0.0024725 -13.415  < 2e-16 ***
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 0.1935 on 344 degrees of freedom
## Multiple R-squared:  0.7788, Adjusted R-squared:  0.7717
## F-statistic: 110.1 on 11 and 344 DF,  p-value: < 2.2e-16
```

Interpreting the Regression Results:

1. **Adjusted. R-squared**: It reflects the fit of the model.
    - R-squared values range from 0 to 1, where a higher value generally indicates a better fit, assuming certain conditions are met.
    - In our case, the value for Adj. R-squared is **0.76**
2. **coeff**: It represents the change in the output Y due to a change of one unit in the variable (everything else held constant).
3. **std err**: It reflects the level of accuracy of the coefficients.
    - The lower it is, the more accurate the coefficients are.
4. **P >|t|**: It is p-value.
    - Pr(>|t|) : For each independent feature there is a null hypothesis and alternate hypothesis Ho : Independent feature is not significant Ha : Independent feature is significant
    - A p-value of less than 0.05 is considered to be statistically significant.
5. **Confidence Interval**: It represents the range in which our coefficients are likely to fall (with a likelihood of 95%).

- Both the **R-squared and Adjusted R-squared of the model are around 76%**. This is a clear indication that we have been able to create a good model that is able to explain variance in the house prices for up to 76%.
- we can examine the significance of the regression model, try dropping insignificant variables.

Dropping the insignificant variables from the above model and creating the regression model again.

Examining the significance of the model

It is not enough to fit a multiple regression model to the data, it is necessary to check whether all the regression coefficients are significant or not. Significance here means whether the population regression parameters are significantly different from zero.

From the above it may be noted that the regression coefficients corresponding to ZN, AGE, and INDUS are not statistically significant at level = 0.05. In other words, the regression coefficients corresponding to these three are not significantly different from 0 in the population. Hence, we will eliminate the three features and create a new model.

```r
# set seed to fix number where be able to have same random set everytime
set.seed(100)
# split using createDataPartition function where train data has 70% of the data
# and test data has 30% of the data
trainIndex <- createDataPartition(df$MEDV_log, p = .7,
                                  list = FALSE,
                                  times = 1)
df_significant <- df
# Drop 'TAX', 'ZN', 'AGE', 'INDUS'
df_significant$TAX <- NULL
df_significant$ZN <- NULL
df_significant$AGE <- NULL
df_significant$INDUS <- NULL

# Create training and testing dataset
train <- df_significant[trainIndex,]
test <- df_significant[-trainIndex,]

# create the model
model2 <- lm(MEDV_log ~ ., data = train)

# print summary of the model
summary(model2)
```

```
##
## Call:
## lm(formula = MEDV_log ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.67114 -0.10440 -0.00471  0.10392  0.82756
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.209564   0.232939  18.072  < 2e-16 ***
## CRIM        -0.009110   0.001786  -5.102 5.55e-07 ***
## CHAS         0.108450   0.041852   2.591 0.009966 **
## NOX         -0.647223   0.173509  -3.730 0.000223 ***
## RM           0.080453   0.018798   4.280 2.42e-05 ***
## DIS         -0.038198   0.008167  -4.677 4.17e-06 ***
## RAD          0.002654   0.001951   1.360 0.174566
## PTRATIO     -0.041788   0.005852  -7.141 5.48e-12 ***
## LSTAT       -0.032278   0.002279 -14.164  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1936 on 347 degrees of freedom
## Multiple R-squared:  0.7765, Adjusted R-squared:  0.7714
## F-statistic: 150.7 on 8 and 347 DF,  p-value: < 2.2e-16
```

Observations:

- We can see that the **R-squared value** and **adjusted R-squared** has not been changed that as what we expect. Now, we will check the linear regression assumptions.

Checking the performance of the model on the train and test data set

Data	RMSE	MAE	MAPE
Train	0.1911638	0.1407065	4.901783
Test	0.2045969	0.1407951	4.848039

Observations:

- RMSE, MAE, and MAPE of train and test data are not very different, indicating that the **model is not overfitting and has generalized well.**

Applying the cross validation technique to improve the model and evaluating it using different evaluation metrics.

```
set.seed(100)
fitControl <- trainControl(method = "cv", number = 10)
fit <- train(MEDV_log ~ .,
         data = train,
         method = "lm",
         trControl = fitControl)
fit
```

```
## Linear Regression
##
## 356 samples
##   8 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 320, 321, 320, 320, 320, 322, ...
## Resampling results:
##
##   RMSE       Rsquared  MAE
##   0.1969757  0.760503  0.1446553
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

Observations:

- The R-squared on the cross validation is 0.760, whereas on the training dataset it was 0.776
- And the MSE on cross validation is 0.147, whereas on the training dataset it was 0.142

We may want to reiterate the model building process again with new features or better feature engineering to increase the R-squared and decrease the MSE on cross validation.

Interpreting Regression Coefficients

With our linear regression model's adjusted R-squared value of around 0.76, we are able to capture **76% of the variation** in our data.

The model indicates that the most significant predictors of the logarithm of house prices are:

NOX: -0.647223

CHAS: 0.108450

RM: 0.080453

PTRATIO: -0.041788

DIS: -0.038198

LSTAT: -0.032278

CRIM: -0.009110

RAD: 0.002654

The p-values for these variables are < 0.05 in our final model, meaning they are statistically significant towards house price prediction.

It is important to note here that the predicted values are log (MEDV) and therefore coefficients have to be converted accordingly by taking their exponent to understand their influence on price.

- The house price decreases with an increase in NOX (nitric oxide concentration). **1 unit increase in the NOX leads to a decrease of** exp(0.647223) ~ **1.91 times the price** of the house when everything else is constant. This is fairly easy to understand as more polluted areas are not desirable to live in and hence cost less.
- The house price increases with an increase in CHAS (Charles River variable). **1 unit increase in CHAS leads to an increase of** exp(0.108) ~ **1.11 times the price** of the house. This is understandable, as houses by the river would be more desirable due to their scenic view, and hence more expensive.
- The house price increases with an increase in RM (average number of rooms per dwelling). **1 unit increase in RM leads to** exp(0.0804) ~ 1.08 times, or a **6% increase in the price of the house** when everything else is constant. Clearly, the higher the average number of rooms per dwelling, the more expensive the house.
- Other variables such as CRIM (per capita crime rate by town), PTRATIO (pupil-teacher ratio by town), DIS (weighted distances to 5 Boston employment centers) and LSTAT (% Lower Status of the population) are all negatively correlated with house price, for differing reasons.
- The RAD variable (index of accessibility to radial highways), with a small coefficient of 0.0026, while being statistically significant, does not appear to have much of an effect on the price of the house.

Let’s now build Non- Linear models like Decision tree and Random forest and check their performance

Building Decision Tree

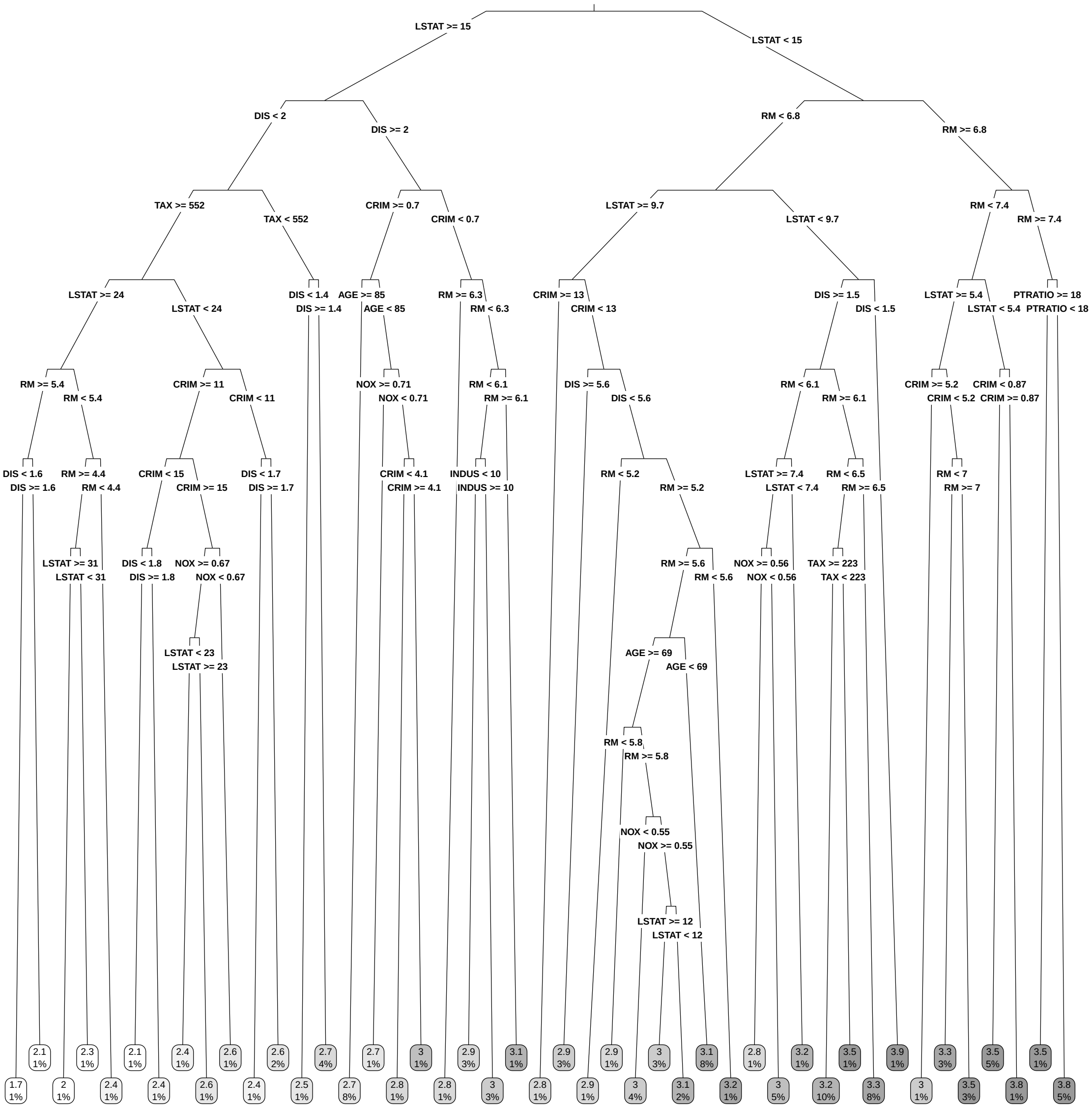
```r
# set seed to fix number where be able to have same random set everytime
set.seed(100)
# split using createDataPartition function where train data has 70% of the data
# and test data has 30% of the data
trainIndex <- createDataPartition(df$MEDV_log, p = .7,
                                  list = FALSE,
                                  times = 1)


# Create training and testing dataset
train <- df[trainIndex,]
test <- df[-trainIndex,]

# create the model
model <- rpart(MEDV_log ~ ., data = train,
               control = rpart.control(minsplit=3 , minbucket=2,
                                       maxdepth=30, cp = 0.001, xval=10))

# print summary of the model
# Compute performence of Model (rpart model)
pref_decision_tree_model <- model_pref(model = model,
                                       x_train = train[,-13], x_test = test[,-13],
           y_train = train$MEDV_log, y_test = test$MEDV_log)
```

Checking Regression Trees model perform on the train and test dataset

Data	RMSE	MAE	MAPE
Train	0.0999833	0.0759148	2.569327
Test	0.2558427	0.1523146	5.495951

Observations:

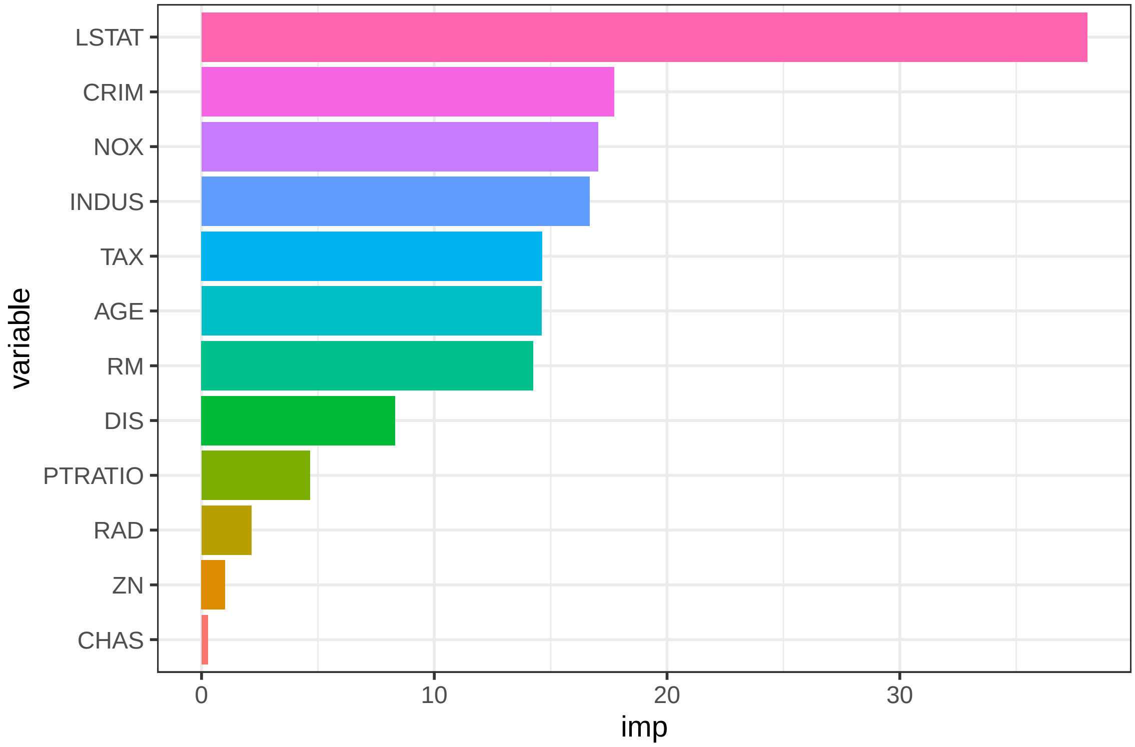
- **The model seem to overfit the data** by giving almost good RMSE result of the train dataset and compare with error on the test dataset.



Observations:

- **The first split is at LSAT<=14.915%**, which signifies that areas where LSAT% or the Lower status of the population resides is <15%, then the prices of the property are high.
- **Other 2 important factors which decide the property rate are DIS and RM**. Houses with no. of rooms >6 are having a high price as compared to other houses, which makes senses as more number of room corresponds to the bigger house and hence large area and thus higher prices.
- **For the area where per capita crime rate is higher the house prices are lower**. This corresponds 2 to things, first is maybe area where low population reside a even smaller crime rate is affecting that area and making it risky to live. Second is maybe there is a populated area and crime rate is also high, but dues some other factors of the property.

Let's plot the feature importance for each variable in the dataset and analyze the variables



Observations:

- As seen from the decision tree above, **LSAT is the more important variable** which affect the price of a house, as area where high profile people are living tend to have a high price of the house.
- Other important features are **RM, CRIM, DIS and NOX level**. These variables collectively signifies that people can pay higher price for the areas where the crime rate is less, which are near to highways and are healthy to live.
- Another important observation can be that, from the decision tree area with low LSAT, high RM and low PTRATIO tend to have a high price for the houses in that area, But here we can see these features are not that important in deciding the price of the house, One of the reason can be that in the decision tree these features are corresponding to the houses with high prices which signifies that these features are only contribution for the area where good profile people are living.

Building Random Forest

```r
# set seed to fix number where be able to have same random set everytime
set.seed(100)
# split using createDataPartition function where train data has 70% of the data
# and test data has 30% of the data
trainIndex <- createDataPartition(df$MEDV_log, p = .7,
                                  list = FALSE,
                                  times = 1)
```

```r
# Create training and testing dataset
train <- df[trainIndex,]
test <- df[-trainIndex,]

# create the model
model <- randomForest::randomForest(MEDV_log ~ ., data = train, mtry=2, ntree=1000)

# print summary of the model
# Compute performence of Model (rpart model)
pref_random_forest_model <- model_pref(model = model,
                                       x_train = train[,-13], x_test = test[,-13],
           y_train = train$MEDV_log, y_test = test$MEDV_log)
```

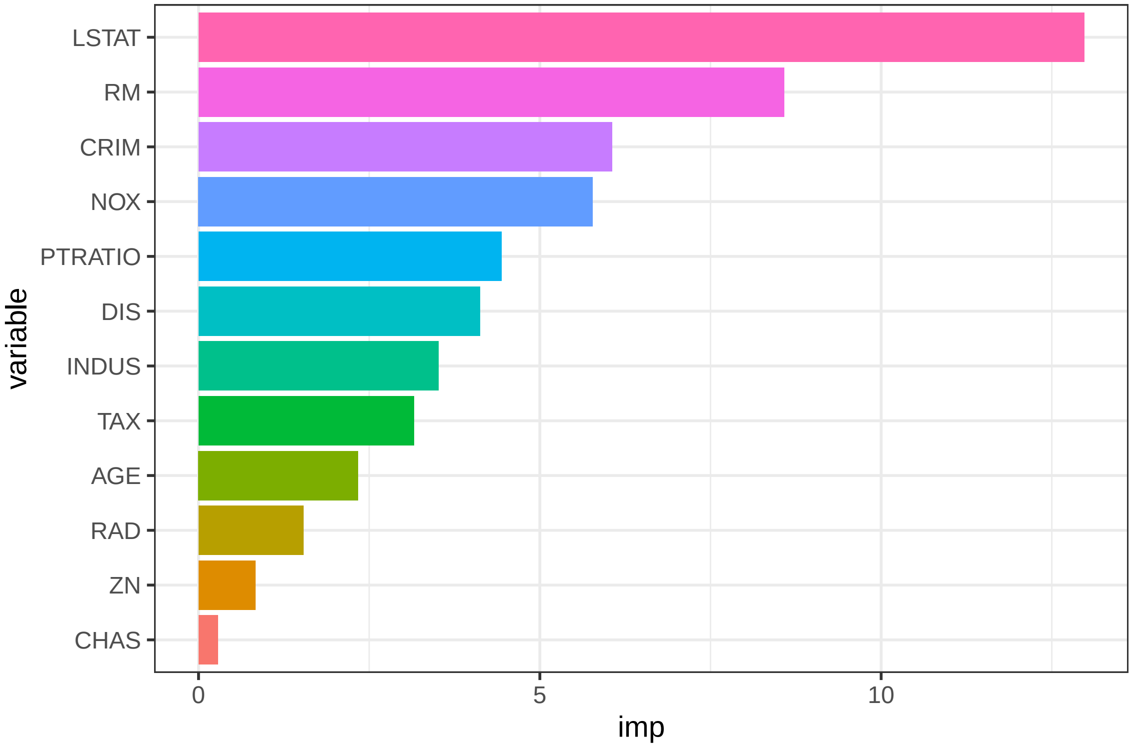
Checking Regression Trees model perform on the train and test dataset

Data	RMSE	MAE	MAPE
Train	0.0980277	0.0645004	2.328720
Test	0.1684565	0.1132508	3.953283

Observations:

- **RMSE, MAE and MAPE for the random forest are very small and are close for both train and test dataset.** Hence model is performing very good and giving generalized results.

Let's plot the feature importance for each variable in the dataset and analyze the variables



Observations:

- The feature importance for decision Tree and Random forest both are approximately same.

Models Performance Comparison

```
## [1] "Linear Regression"
```

```
##    Data      RMSE       MAE     MAPE
## 1 Train 0.1911638 0.1407065 4.901783
## 2  Test 0.2045969 0.1407951 4.848039
```

```
## [1] "Decision tree"
```

```
##    Data       RMSE        MAE     MAPE
## 1 Train 0.09998331 0.07591478 2.569327
## 2  Test 0.25584269 0.15231458 5.495951
```

```
## [1] "Random Forest"
```

```
##    Data       RMSE        MAE     MAPE
## 1 Train 0.09802769 0.06450037 2.328720
## 2  Test 0.16845647 0.11325084 3.953283
```

Observations:

- All the 3 models are performing good and have low RMSE, MAE and MAPE.
- **Decision tree is overfitting a bit as it is giving around 100% results on the train dataset**, which Linear Regression and Random Forest are not over fitting.
- **Random forest is giving the best result of all the 3 models.**

Conclusion

- Our final **Random forest has a Mean Absolute Percentage Error (MAPE) of ~4%** on the test data, which means that **we are able to predict within ~4% of the price value on average**. This is a good model and we can use this model in production.

- We can maybe use Linear regression to get statistical insights about the model and maybe use Random forest in production.

- **Percentage lower-status population has a negative correlation with house price and have the highest importance in deciding the price for the house**, because the definition of the term includes people without or with partial high-school education and male workers classified as laborers; a population segment that is likely to live in less well-off, inexpensive areas due to their lower incomes. Hences houses in such areas would be cheaper to buy as well.

- **Crime rate is negatively correlated with house price and is also an important feature in predicting the house prices**, as neighborhoods and areas with a higher crime rate would clearly be more undesirable and cheaper to buy a house in.

- **The NOX level are highly negatively correlated with the house prices** and is one on the important feature in predicting house prices.This is fairly easy to understand as more polluted areas are not desirable to live in and hence cost less.

- **The pupil-to-teacher ratio is negatively correlated with house price and Decision tree suggested that the lower pupil-to-teacher ratio have higher house prices.**, presumably because students, looking for affordable housing due to their general lack of income and having to pay tuition fees, may be concentrated in less expensive areas. Teachers on the other hand, are paid well and may live in more well-to-do neighborhoods.

- **Distance to employment centers also has a negative correlation with house price**, probably because like many developed cities in the world, Boston has been built from the center radially outwards. The employment centers, being close to the center of the city like many of the oldest and most important of the city's buildings, are areas of prime real estate due to the convenience of being so close, whereas suburban areas further way from the town center that are recently developed, may be less expensive to buy houses.
- Note that I didn't apply tuning techniques in the Random Forest section, I will show that in the coding file :)