```r
#load BostonHousing Data

#Machine Learning Benchmark Problems

#A collection of artificial and real-world machine learning benchmark problems,

install.packages("mlbench")

library(mlbench)

#data manipulation library

install.packages("dplyr")

library(dplyr)

#powerful graphics language for creating elegant and complex plots

install.packages("ggplot2")

install.packages("reshape2")

library(ggplot2)

#library which makes easy to transform data between wide and long formats.

library(reshape2)

data("BostonHousing")

housing <- BostonHousing

View(housing)

str(housing)



#ggplot

housing %>%

  #The infix operator %>% is not part of base R, but is in fact defined by the package magrittr (CRAN) and is heavily used by dplyr

  ggplot(aes(x = medv)) +

  stat_density() +

  labs(x = "Median Value ($1000s)", y = "Density", title = "Density Plot of Median Value House Price in Boston") +

  theme_minimal()



#summary
```

```
summary(housing$medv)
```

```
#predicted V/S original
housing %>%
  select(c(crim, rm, age, rad, tax, lstat, medv)) %>%
  melt( id.vars = "medv") %>%
  ggplot(aes(x = value, y = medv, colour = variable)) +
  geom_point(alpha = 0.7) +
  stat_smooth(aes(colour = "black")) +
  facet_wrap(~variable, scales = "free", ncol = 2) +
  labs(x = "Variable Value", y = "Median House Price ($1000s)") +
  theme_minimal()
```

```
#Set a seed of 123 and split your data into a train and test set using a 75/25 split. You may find the caret library helpful here.
install.packages("caret")
library("caret")
set.seed(123) #random number geneartion
to_train <- createDataPartition(y = housing$medv, p = 0.75, list = FALSE)
to_test<-createDataPartition(y=housing$medv, p=0.25,list=FALSE)
train <- housing[to_train, ]
test <- housing[to_test, ]
```

```
#fit a linear model
first_lm <- lm( medv ~ crim +rm +tax +lstat, data = train)
```

```
#Obtain an r-squared value for your model and examine the diagnostic plots found by plotting your linear model.
```

```
lm1_rsqu <- summary(first_lm)$r.squared

print(paste("First linear model has an r-squared value of ", round(lm1_rsqu, 3), sep = ""))

## [1] "First linear model has an r-squared value of 0.672"

#plot(first_lm)



#Fix few problems

second_lm <- lm(log(medv) ~ crim +rm + tax +lstat, data = train)


lm2_rsqu <- summary(second_lm)$r.squared

print(paste("Our second linear model has an r-squared value of ", round(lm2_rsqu, 3), sep = ""))



#One assumption of a linear model is that the mean of the residuals is zero.

abs(mean(second_lm$residuals))



#Create a data frame of your predicted values and the original values

predicted <- predict(second_lm, newdata = test)

results <- data.frame(predicted = exp(predicted), original = test$medv)



#Plot this to visualize the performance of your model.

results %>%

  ggplot(aes(x = predicted, y = original)) +

  geom_point() +

  stat_smooth() +

  labs(x = "Predicted Values", y = "Original Values", title = "Predicted vs. Original Values") +

  theme_minimal()
```