A Study of different ML Models on the Boston dataset

Tan Yao Hui

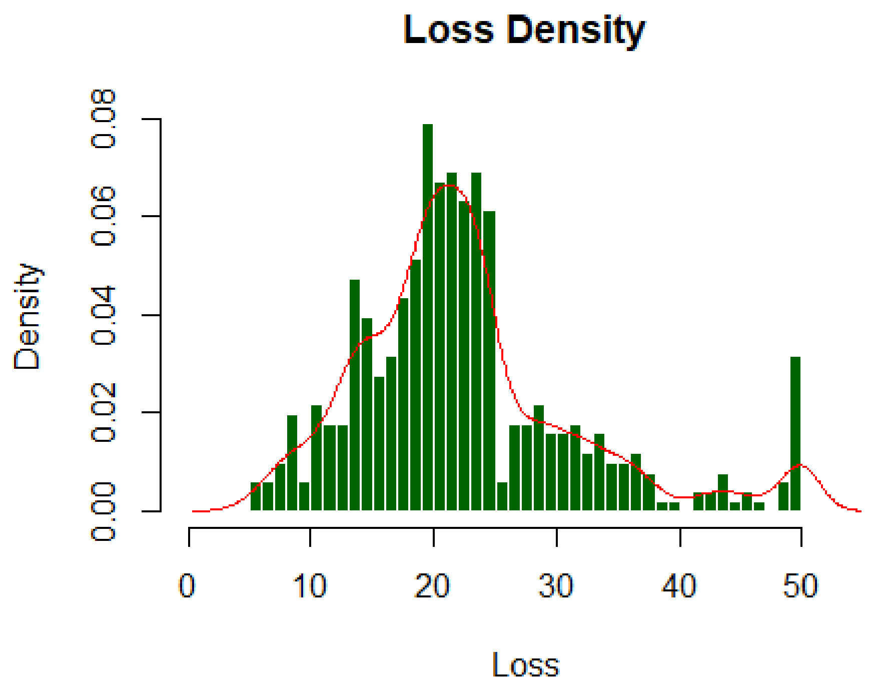
In this project, we will review the performance of linear regression, SVM, decision tree, Random Forests and Gradient Tree Boosting models and also consider the cost of fitting and training these models in our discussion. In each section, we will show the steps taken to encode/ preprocess the data before fitting the model and their rationale.

Data Exploration

In this section, we will plot out the distribution of the target variable. This will help us to choose the best link function for the linear regression model later.

```
data(Boston)
head(Boston)

##      crim zn indus chas   nox    rm  age    dis rad tax ptratio  black
lstat
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90
4.98
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90
9.14
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83
4.03
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63
2.94
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90
5.33
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12
5.21
##   medv
## 1 24.0
## 2 21.6
## 3 34.7
## 4 33.4
## 5 36.2
## 6 28.7
```

Plot histogram and empirical pdf:

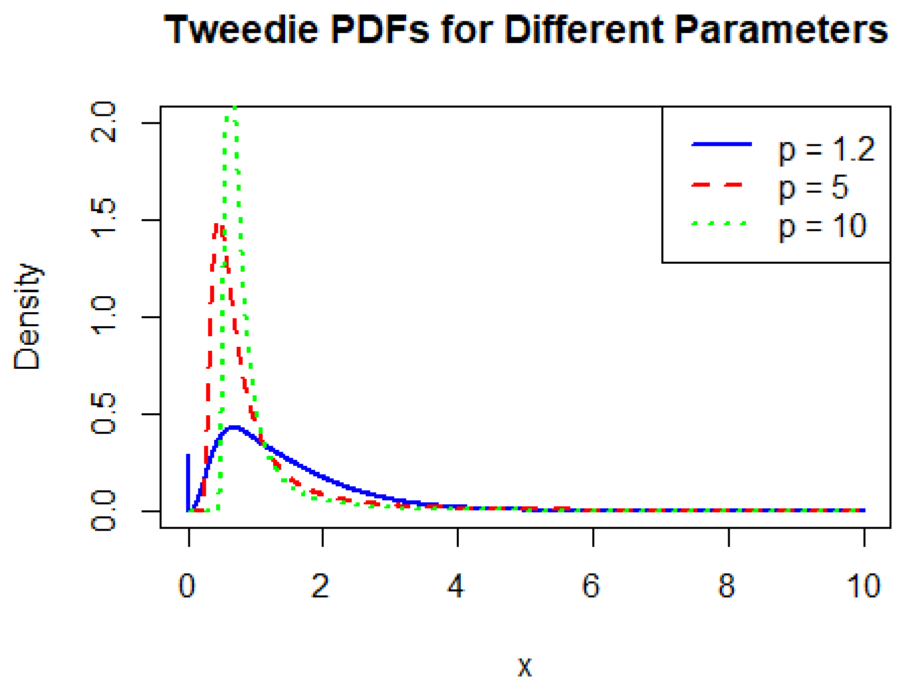


The only categorical variable in the data set is CHAS- this saves us the effort from performing preprocessing or encoding for the variables.

Linear Regression

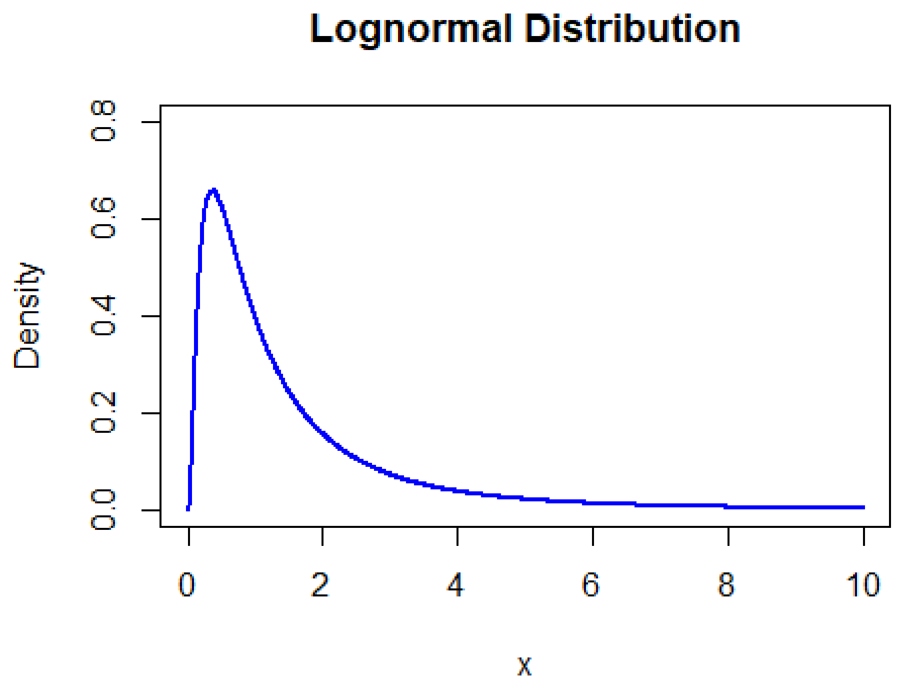
Firstly, we check whether Tweedie Distribution is suitable for our data. Tweedie Regression is a general case for many different distributions like normal, Compound Poisson, Inverse Gaussian, etc so it is a good place to start our search for a suitable link function for our GLM.

We plot out Tweedie Distribution with different power parameters:



Notice that the spike at 0 is higher for lower p. The spike moves to the right as p increases. The Tweedie Regression is suitable for insurance losses where there is a significant loss at 0 due to most policies not having a claim (Zero-inflated claims). Our target variable does not have a spike at 0, therefore Tweedie may not be a good choice for the link function!

We then plot out the pdf of the lognormal distribution to check whether the log function is a good candidate for the link function:



The log link (well in fact its the inverse of the link) converts our predictions into a output scale as the target variable (0,+infinity)

Model Validation with Test Set

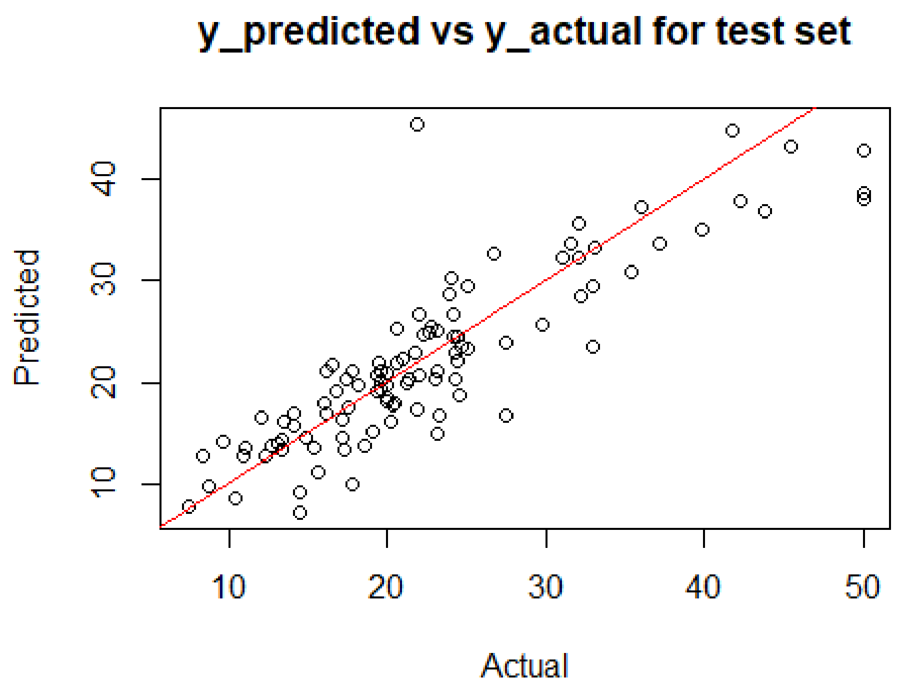
We partition the data into train and test set, with 80% of the data in train and the rest in test:

```
## [1] 404  14

## [1] 102  14
```

1. Linear Regression

```
logNormGLM=glm(medv~., data=train_data,family = gaussian(link="log"))
summary(logNormGLM)

##
## Call:
## glm(formula = medv ~ ., family = gaussian(link = "log"), data =
train_data)
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.5235291  0.2208390  15.955  < 2e-16 ***
## crim        -0.0096822  0.0027901  -3.470 0.000578 ***
## zn           0.0010549  0.0004514   2.337 0.019944 *
## indus        0.0023511  0.0025200   0.933 0.351398
## chas         0.0946490  0.0272138   3.478 0.000562 ***
## nox         -0.5585254  0.1725819  -3.236 0.001314 **
## rm           0.1389202  0.0161143   8.621  < 2e-16 ***
## age          0.0007976  0.0005386   1.481 0.139397
## dis         -0.0441978  0.0078348  -5.641 3.25e-08 ***
## rad          0.0159178  0.0031145   5.111 5.03e-07 ***
## tax         -0.0004453  0.0001649  -2.700 0.007246 **
## ptratio     -0.0325204  0.0052273  -6.221 1.27e-09 ***
## black        0.0004521  0.0001866   2.423 0.015846 *
## lstat       -0.0380261  0.0026024 -14.612  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 16.01159)
##
##     Null deviance: 34108.3  on 403  degrees of freedom
## Residual deviance:  6244.5  on 390  degrees of freedom
## AIC: 2282.7
##
## Number of Fisher Scoring iterations: 8
```

We fit a Linear Regression model with log link to the data. Almost all variables except for indus (proportion of non-retail business acres per town) and age (proportion of owner-occupied units built prior to 1940) are significant. nox (nitric oxides concentration (parts per 10 million)) has the strongest negative effect on medv (Median value of owner-occupied homes in $1000’s)-more polluted areas may be industrial/ agricultural land and thus have a lower median value.



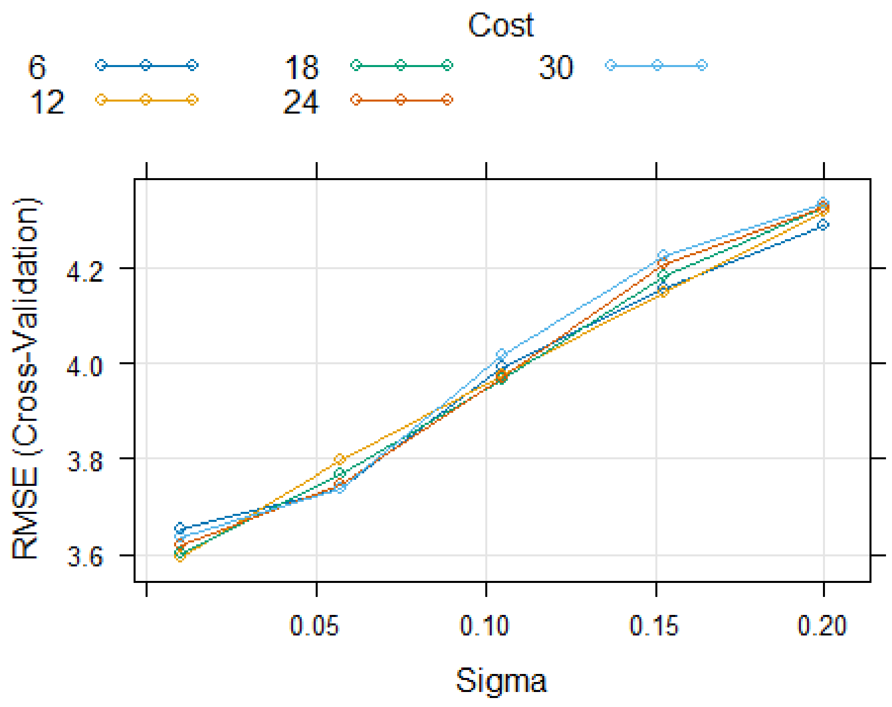
```
## [1] "MSE for Linear Regression with log transform: 21.0233316717663"
```

2. SVM

We normalize the data using the train data so that each feature has mean zero and variance one. Then we use the mean and variance calculated from the train set to preprocess the test set to avoid data leakage.

We perform 5-fold CV to tune hyperparameters for the SVM.In this case, we are using a Gaussian Kernel as it is more flexible and in general perform well on most datasets. For the Gaussian Kernel, we need to tune for sigma (scaling parameter for the distance) and C (cost for the wrongly classified observations). A smaller value of sigma (scaling for distance) - implies we give less weight to observations further away. Note that its sigma not gamma which is usually used in Scikit-Learn.

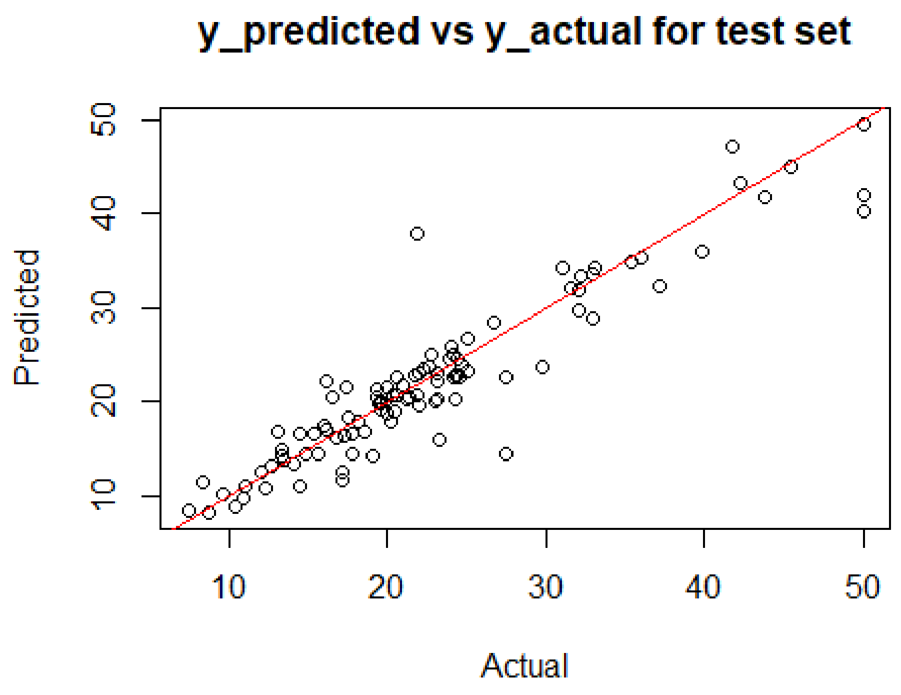
```
##   sigma  C
## 2  0.01 12
```



High levels of regularization perform better with low sigma (assign less weight to points further away - getting an accurate estimate and higher regularization on weights- reduces variance in prediction and prevent overfitting) We choose sigma=0.06 and C=30 (svm performance significantly better with sigma around 0.06 with both C=12/C=30 performing similarly)

Fitting the SVM

```
##
## Call:
## svm(formula = medv ~ ., data = train_data_normalized, gamma = 0.06,
##     cost = 30, scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  30
##       gamma:  0.06
##     epsilon:  0.1
##
##
## Number of Support Vectors:  386
```



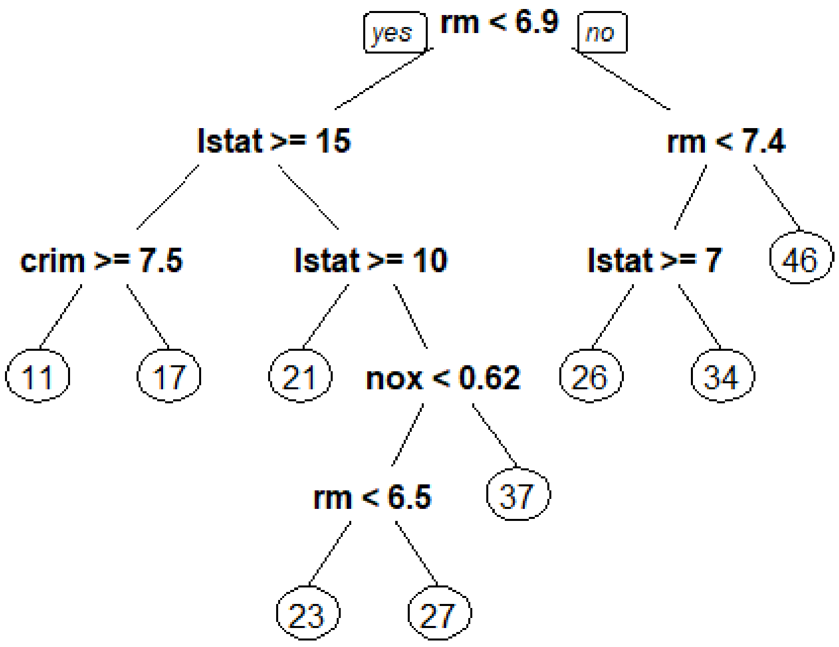
```
## [1] "MSE for SVM: 11.0674406114298"
```

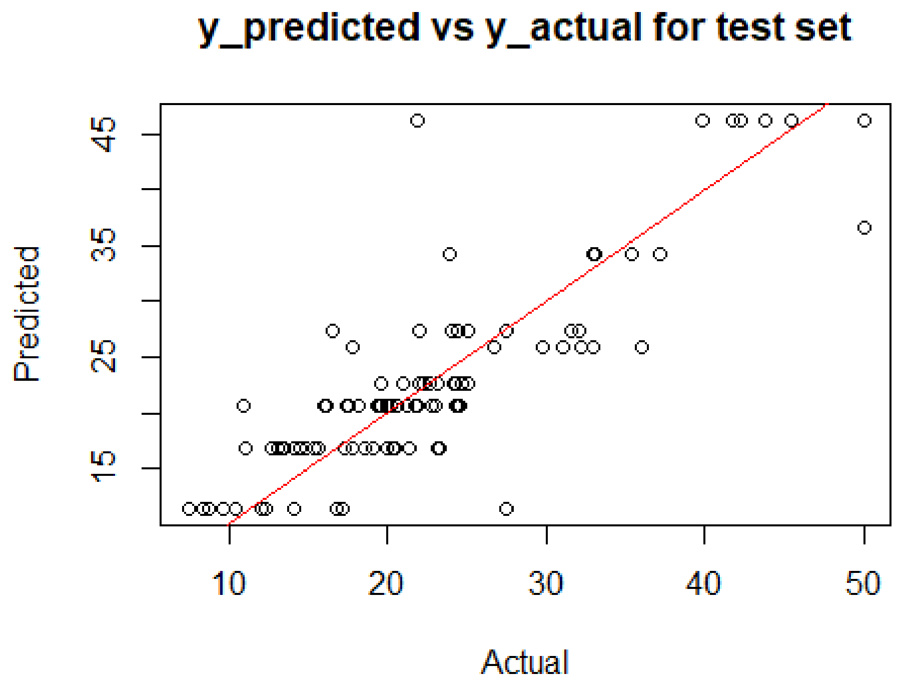
It seems that the performance is better than the results from the Linear Regression with log transform. However,complicated models does not always yield better results as they may overfit on unseen data.

3. Decision Tree

In this section, we try fitting a decision tree to the data as decision trees are known to be robust to hyperparameter choice and usually performs well on most datasets. Therefore, other than Linear Regression, Decision Tree is also a good choice for the baseline model.

Visualization of decision tree:





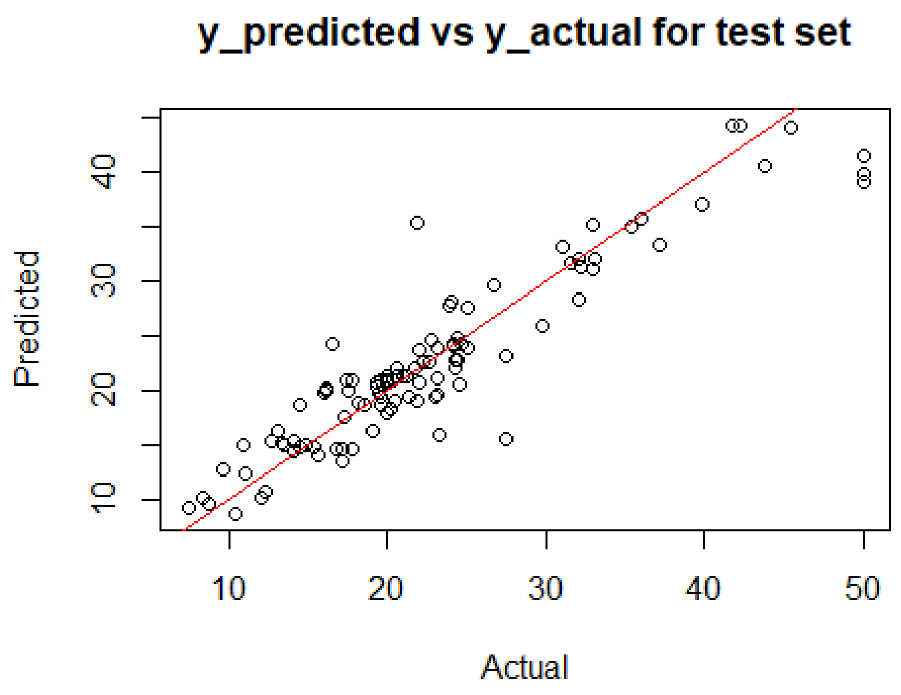
```
## [1] "MSE for decision tree: 24.5104460847889"
```

Note that the MSE of the decision tree is worse than the linear regression model in part 1- more flexible models do not always perform better than weaker and simpler ones! Therefore, comparing multiple models and thinking about how their performance can be improved upon is vital! For example, we notice that the decision tree performs well on the train set but its predictive power does not generalize well to an unseen test set.

4. Random Forest

Building on the previous section, we can try Random Forests to reduce forecast variability and increase the generalization ability of trees on unseen data- most leaves in our tree have around 5% of observations, while the leaf with least observations has 2.5% of observations-> showing that decision trees are susceptible to overfitting on data.

```
##
## Call:
##  randomForest(formula = medv ~ ., data = train_data)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 10.79007
##                     % Var explained: 87.22
```



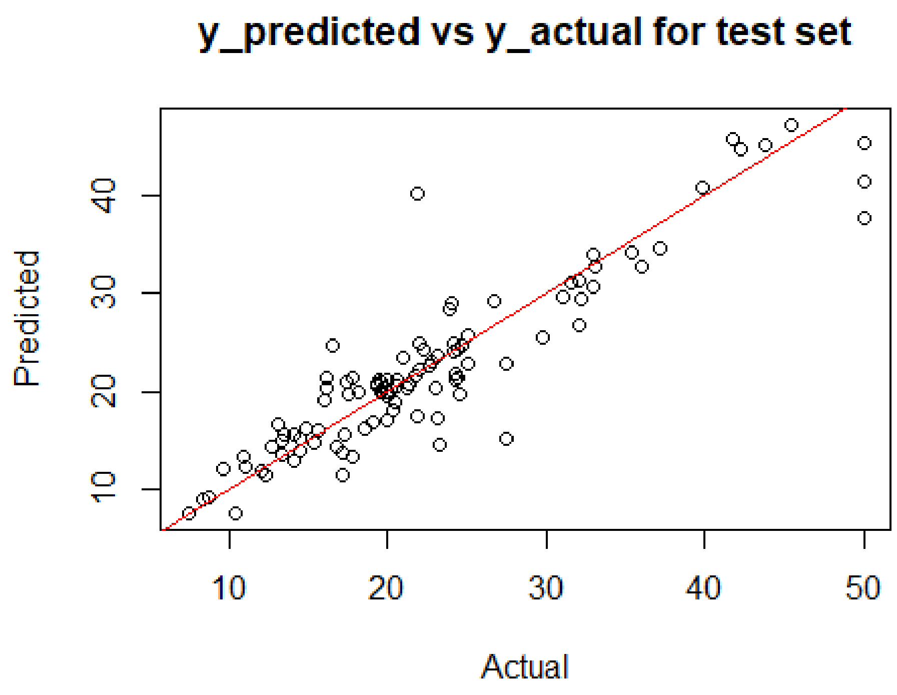
```
## [1] "MSE for random forest: 11.715906685778"
```

The Random Forest showed much promising results compared to the dismal performance by the decision tree. Surprisingly, its test MSE is one of the lowest among all models! Training weaker models and aggregating their results reduces variance and also ensures that the pattern in the data is strong enough for most bootstrapped predictors to capture it, thus avoiding overfitting.

5. Gradient Boosting Machine

Boosted Tree may also perform well as additional trees can help to make more granular predictions for nodes with a large number of observations(the leaf with most observations has around 12.5% of total observations)

```
## Distribution not specified, assuming gaussian ...

## gbm(formula = medv ~ ., data = train_data)
## A gradient boosted model with gaussian loss function.
## 100 iterations were performed.
## There were 13 predictors of which 11 had non-zero influence.

## Using 100 trees...
```



```
## [1] "MSE for Gradient Boosted Machine: 14.114363327315"
```

Gradient Boosting Machine improves on the performance of the decision tree, reporting a MSE around 15. The GBM model is able to perform better than decision trees due to its ability to perform more granular predictions for nodes with a large number of observations, thus improving accuracy. It also uses an out-of-bag set to determine whether to grow new trees, hence improving the ability to generalize to unseen data (if there is sufficient data in the test set to capture the characteristics of the entire population). GBM is also fairly robust to overfitting since the subsequent trees are fit on the residuals of the last prediction (which means that we are trying to reduce the prediction error at each step)- observations with large prediction errors are weighted more in the subsequent trees.

Summary of Results

```
##                                Model      MSE
## 1 Linear Regression(Lognormal Link) 21.02333
## 2                                SVM 11.06744
## 3                      Decision Tree 24.51045
## 4 Random Forest (500 trees default) 11.71591
## 5           Gradient Boosted Machine 14.11436
```

From this report, we learnt that using Linear Regression or GLM is always a good choice as baseline predictor of a dataset. After studying the performance of the LR model, we can gain insight on what models to try to improve on the baseline performance.

For example, if the target variable has a peak at zero and is heavy tailed as in insurance losses, trying neighbour-based methods may help to improve predictions as losses may be random for a similar group of drivers- hence the predictor variables supplied are useless in predicting losses. We can also use decision trees to determine which policies are bad risks that we want to remove from the portfolio has they are likely to have extremely high claims.

Plotting out the correlation or other dependence measures for the predictor variables with the target variable can help us to determine whether regression models will perform well- for example, if there is no clear relationship between target and predictors, maybe trees can capture the complex nature of the data generating process better than regression models.

The best model is the SVM regression model- however, note that SVM performance may be weak if there are too many indicator functions from one-hot encoding (usually tree-based models perform better than SVM since they can handle high dimensionality due to OHE better) SVM usually performs better when the data share a similar scale- remember to normalize your data before training the model. SVM also may not perform well when the dataset has many indicators and numerical variables- RBF and the polynomial kernels are more suited for continuous data, and are unable to handle both categorical and continuous data.

The Random Forest model ranked second, at a slight disadvantage against the SVM. Training on bootstrapped samples of the data seems to yield good results and is able to overcome the problem of overfitting in decision trees, hence generating predictions that generalize well to unseen data.

The Gradient Boosting Machine model also presents a formidable tool to deal with the Boston dataset, with the third lowest MSE and a large improvement over the decision tree model.

In a nutshell, we learnt that testing models of different natures is vital for us to pick the best performing model for a dataset. Cross validation is also important for certain models like SVM to choose the best hyperparameters for a dataset.