VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Master of Technology in Data Science Programming

CSI 3003
ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS

DIGITAL ASSIGNMENT – 1

TITLE: ESTIMATING HOUSING PRICE CHANGES IN REAL ESTATE

FACULTY : PROF. ANNAPURNA JONNALAGADDA
SLOT : G1+TG1

TEAM MEMBERS :

- SAI MONICA R. – 20MID0165
- POOJA SREE C. A. – 20MID0204

Problem Description :

Property values in the real estate industry are closely tied to our economy. The goal of this project is to use machine learning to forecast the selling prices of houses based on a variety of financial considerations. Home prices are a significant indicator of the health of the economy, and both buyers and sellers are keenly interested in price points. Despite this, we do not have precise estimates of house costs despite the abundance of data available. In this study, house values will be predicted using explanatory factors that include a wide variety of residential dwelling features. This study tries to create a regression model to accurately predict the cost of the house given its characteristics. (ML and Hybrid ML models are applicable)

Objectives :

1. Estimate the cost of each house's sale.

2. Reduce the discrepancy between the rating's expected and actual values.

Introduction :

In order to make wise financial decisions, it is crucial to assess the market's current state and forecast its performance over the coming few months. In this research, artificial machine learning models are used to assist real estate investors and builders throughout this crucial work.

(a) Significance :

Prospective homeowners, developers, investors, appraisers, tax assessors, and other players in the real estate industry, such as mortgage lenders and insurance, depend on an accurate projection of the property price (Frew and Jud, 2003). Conventional home price forecasting relies on cost and sale price comparisons without an established benchmark or certification procedure. Thus, having a house price prediction model available closes a critical information gap and boosts the effectiveness of the real estate market (Calhoun, 2003). The majority of people in New Zealand are aware of the advantages of home ownership because it is thought to be the most practical and rewarding investment. With almost 70% of its population living in their own homes, New Zealand has one of the highest homeownership rates in the western world. House price has become a major consideration for anyone looking for a home due to New Zealand's booming housing market.

(b) Applications :

Can be useful for middle class and upper-middle-class people who are unable to predict the value of their property in the next 10 to 15 years. It can also be useful for brokers and land dealers in showcasing the best and most affordable real estate properties, if they are on the market. This project can be readily carried out by anyone, which is a benefit to the community as no one will be taken advantage of and everyone can get fair rates for their dream lands. It will assist the government in getting resources closer to public property. The Project can be expanded by merging two or more ML algorithms to attain greater accuracy . Also, we may keep adding features to make the prediction process more effective. The models suggested in the project can be used to forecast a variety of other items whose datasets match the description of the datasets utilised, in addition to housing prices.

(c) Motivation:

E-learning and e-education are heavily affected today. Automation is replacing manual systems everywhere. This project's goal is to forecast house prices in order to lessen the difficulties the consumer would experience. The customer currently approaches a real estate agent to handle his or her investments and recommend appropriate estates for his investments. However, this approach carries some risk because the agent could make a mistaken estate prediction and lose the customer's capital as a result. The manual approach that is still prevalent in the market is risky and out-of-date. There is a need for an updated, automated system to address this flaw. Data mining algorithms can be used to guide investors towards making a suitable real estate investment based on their stated needs. The new system will also save money and time. Its operations will be straightforward. The linear regression algorithm is the basis for the suggested system.

Relevance of the Problem:

Economical and Industrial Problems:

The housing market is intertwined with the economy and industries. Because it can limit employees' mobility and their capacity to take advantage of job possibilities, the availability of inexpensive housing can have a big impact on the economy. The building industry, a significant economic contributor, and the housing market are both tightly related. Hence, creating reliable and effective housing price prediction models can also be advantageous for businesses and industry.

Societal Problem:
Housing availability and affordability are significant societal problems in many places of the world. Accurate housing price forecasting can assist decision-makers and urban planners in crafting housing policies that can have a substantial impact on society.

AI Relation :
AI can aid in the creation of reliable and effective models for house price prediction. Large datasets can be analysed to find patterns that can be utilised to create predictions using machine learning techniques. AI can also assist in determining which characteristics are most crucial for predicting housing values.

Literature survey:

- The existing research papers talks about many ML and DL models but does not accouri the desired / good accuracy
- Methodologies used in the present research papers :

Random Forest Regression:

A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap Aggregation, commonly known as bagging.

The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees. Approach : · Pick at random K data points from the training set. · Build the decision tree associated with those K data points. · Choose the number Ntree of trees you want to build and repeat step 1 & 2. · For a new data point, make each one of your Ntree trees predict the value of Y for the data point, and assign the new data point the average across all of the predicted Y values.

Polynomial Regression :

Polynomial regression is a form of linear regression in which the relationship between the independent variable x and dependent variable y is modeled as an nth degree polynomial. Polynomial regression fits a nonlinear relationship between the value of x and the corresponding conditional mean of y, denoted E(y |x)

KNN Regression:

The K-Nearest Neighbours (KNN) algorithm is a simple, easy to implement supervised machine learning algorithm that can be used to solve both classification and regression problems. The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. KNN captures the idea of similarity (sometimes called distance, proximity, or closeness) with some mathematics we might have learned in our childhood— calculating the distance between points on a graph. There are other ways of calculating distance, and one way might be preferable depending on the problem we are solving. However, the straight - line distance (also called the Euclidean distance) is a popular and familiar choice.

Ordinary Least-Squares Regression:

• In statistics, ordinary least squares (OLS) is a type of linear least squares method for estimating the unknown parameters in a linear regression model. OLS chooses the parameters of a line ar function of a set of explanatory variables by the principle of l east squares: minimizing the sum of the squares of the differences between the observed dependent variable (values of the variable being observed) in the given dataset and those predicted by the linear function.

Insights :

- ➔ The estimates of the unknown parameters obtained from linear least squares regression are the optimal. Estimates from a broad class of possible parameter estimates under the usual assumptions are used for process modelling. It uses data very efficiently. Good results can be obtained with relatively small datasets.
- ➔ The K-Nearest Neighbour (KNN) Classifier is a very simple classifier that works well on basic recognition problems.
- ➔ There is no need for feature normalization. Individual decision trees can be trained in parallel. Random forests are widely used. They reduce overfitting.

Outcomes :

- ➔ **Accurate prediction of housing prices:**

The creation of a machine learning model that can precisely anticipate house values based on the input features in the dataset would be the project's main accomplishment.

➔ **Feature importance analysis:**

The project could also contain a breakdown of the key elements that influence housing price forecasting. Policymakers and urban planners can use this data to better understand the variables influencing housing costs and create strategies to solve them.

➔ **Visualization of the data:**

Data can be represented using visualization techniques in a way that is more understandable and approachable. This can aid stakeholders in comprehending the connections between various attributes and the anticipated house prices.

➔ **Evaluation of different machine learning algorithms:**

The project may also involve comparing various machine learning algorithms to see which one is more effective at forecasting home prices.

ABOUT DATASET:

DATASET DESCRIPTION

This data frame contains the following columns:

- 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 :nitrogen 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 mean of distances to five Boston employment centres.
- ptratio: pupil-teacher ratio by town.
- lstat:lower status of the population (percent).
- rad: index of accessibility to radial highways.
- medv:Median value of owner-occupied homes in $1000s.
- b:where Bk is the proportion of blacks by town.

Implementation:

https://rpubs.com/sai_monica_10/AI_code

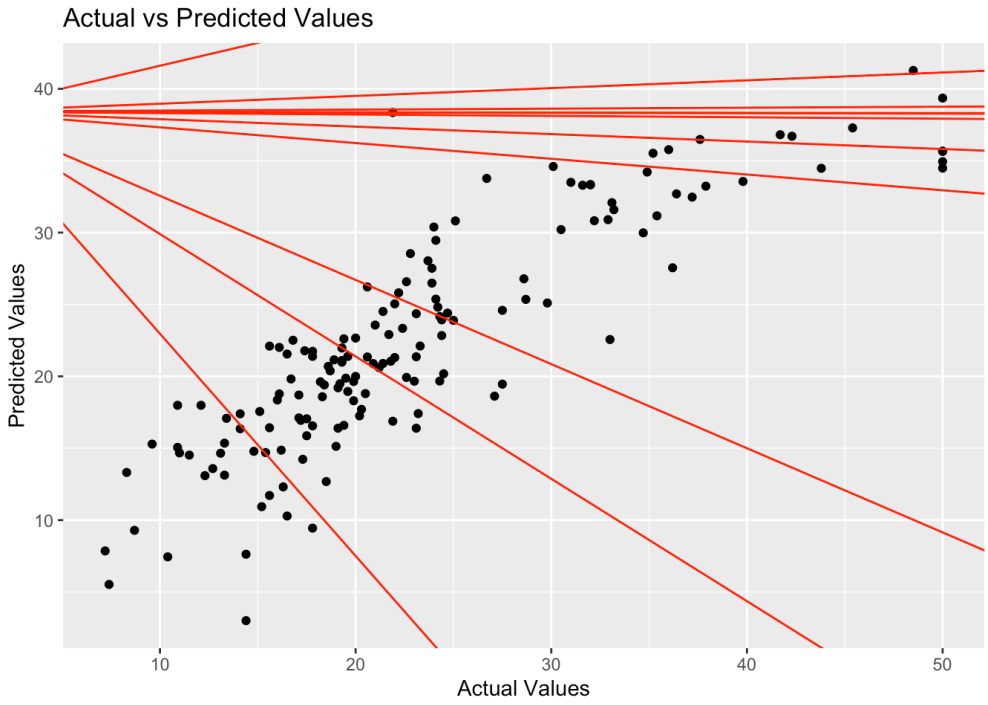
```
#LINEAR REGRESSION

# Load the required libraries
library(MASS)
library(ggplot2)

# Load the dataset
data <- read.csv("/Users/saimonica/Downloads/Boston.csv")

# Remove rows with missing values
data <- na.omit(data)

# Split the dataset into training and testing sets
set.seed(123)
train_index <- sample(1:nrow(data), 0.7*nrow(data))
train_data <- data[train_index, ]
test_data <- data[-train_index, ]

# Fit a linear regression model
lm_model <- lm(medv ~ ., data = train_data)

# Print the summary of the model
summary(lm_model)
```

```
##
## Call:
## lm(formula = medv ~ ., data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.3224  -2.7898  -0.4736   1.5570  24.6878
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  38.417513   6.087489   6.311 8.68e-10 ***
## X            -0.002155   0.002547  -0.846 0.398033
## crim         -0.109511   0.035410  -3.093 0.002149 **
## zn            0.054555   0.016785   3.250 0.001269 **
## indus        -0.052149   0.078799  -0.662 0.508554
## chas          4.079680   1.026026   3.976 8.56e-05 ***
## nox         -14.119739   4.686855  -3.013 0.002785 **
## rm            3.199981   0.500202   6.397 5.25e-10 ***
## age          -0.002898   0.016424  -0.176 0.860032
## dis          -1.545550   0.240687  -6.421 4.56e-10 ***
## rad           0.318610   0.082940   3.841 0.000146 ***
## tax          -0.009904   0.004711  -2.102 0.036249 *
## ptratio      -0.851407   0.160198  -5.315 1.94e-07 ***
## black         0.006875   0.003444   1.996 0.046733 *
## lstat        -0.585453   0.059204  -9.889  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.789 on 339 degrees of freedom
## Multiple R-squared:  0.7336, Adjusted R-squared:  0.7226
## F-statistic: 66.67 on 14 and 339 DF,  p-value: < 2.2e-16
```

```
# Make predictions on the test set
predictions <- predict(lm_model, newdata = test_data)

# Calculate the RMSE value
rmse <- sqrt(mean((test_data$medv - predictions)^2))
rmse
```

```
## [1] 4.792516
```

```
# Plot the actual vs predicted values
ggplot(data = test_data, aes(x = medv, y = predictions)) +
  geom_point() +
  geom_abline(intercept = lm_model$coefficients[1], slope = lm_model$coefficients[-1], color = "red") +
  labs(x = "Actual Values", y = "Predicted Values", title = "Actual vs Predicted Values")
```



```r
#RANDOM FOREST REGRESSION


# Load the required libraries
library(randomForest)
```

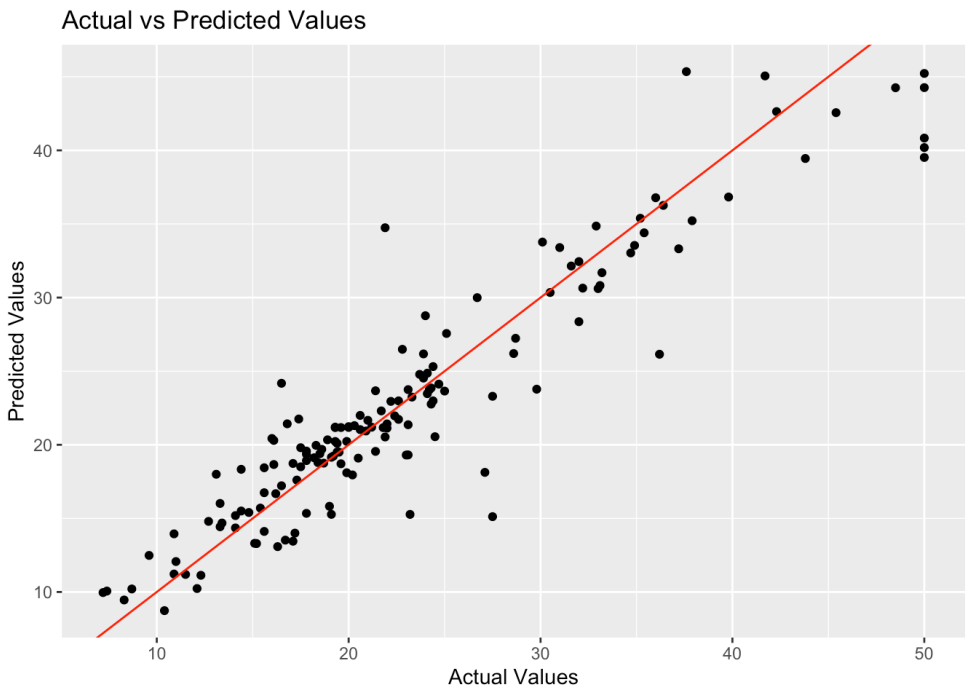
```r
library(ggplot2)

# Load the dataset
data <- read.csv("/Users/saimonica/Downloads/Boston.csv")

# Remove rows with missing values
data <- na.omit(data)

# Split the dataset into training and testing sets
set.seed(123)
train_index <- sample(1:nrow(data), 0.7*nrow(data))
train_data <- data[train_index, ]
test_data <- data[-train_index, ]

# Fit a random forest regression model
rf_model <- randomForest(medv ~ ., data = train_data, ntree = 500, mtry = 3)

# Print the summary of the model
print(rf_model)
```

```
##
## Call:
##  randomForest(formula = medv ~ ., data = train_data, ntree = 500,      mtry = 3)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 10.80667
##                     % Var explained: 86.89
```

```r
# Make predictions on the test set
predictions <- predict(rf_model, newdata = test_data)

# Calculate the RMSE value
rmse <- sqrt(mean((test_data$medv - predictions)^2))
rmse
```

```
## [1] 3.341946
```

```r
# Plot the actual vs predicted values
ggplot(data = test_data, aes(x = medv, y = predictions)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red") +
  labs(x = "Actual Values", y = "Predicted Values", title = "Actual vs Predicted Values")
```




```r
# POLYNOMIAL REGRESSION


# Load the required libraries
library(ggplot2)
library(dplyr)
```

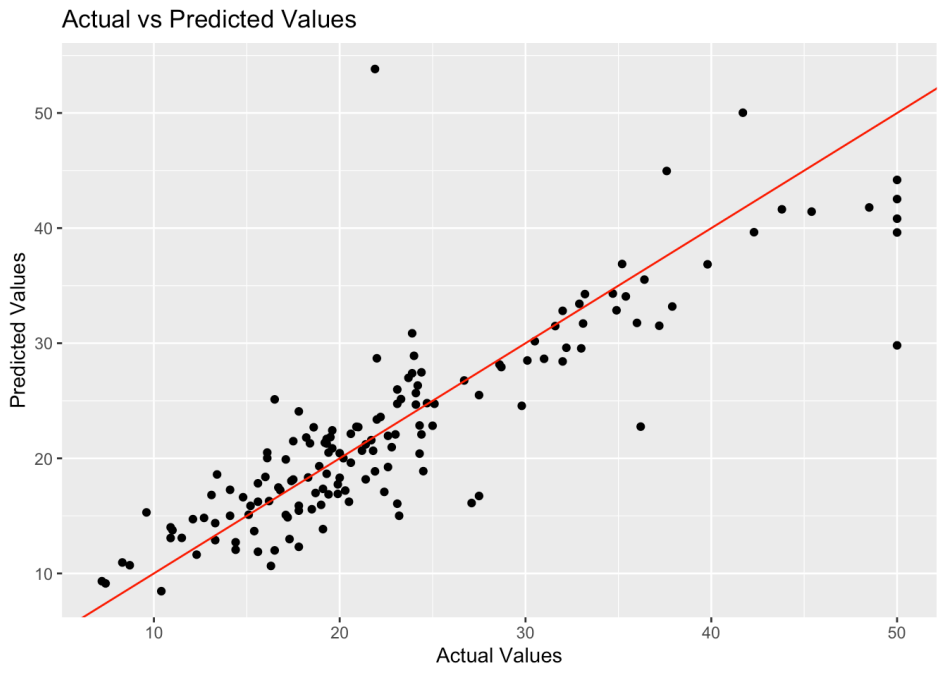
```r
library(tidyr)

# Load the dataset
data <- read.csv("/Users/saimonica/Downloads/Boston.csv")

# Remove rows with missing values
data <- na.omit(data)

# Split the dataset into training and testing sets
set.seed(123)
train_index <- sample(1:nrow(data), 0.7*nrow(data))
train_data <- data[train_index, ]
test_data <- data[-train_index, ]

# Fit a polynomial regression model
poly_model <- lm(medv ~ poly(crim, 2) + poly(rm, 2) + poly(lstat, 2), data = train_data)

# Print the summary of the model
summary(poly_model)
```

```
##
## Call:
## lm(formula = medv ~ poly(crim, 2) + poly(rm, 2) + poly(lstat,
##     2), data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.9760  -2.6130  -0.5454   1.6917  28.8604
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       22.3011     0.2296  97.110  < 2e-16 ***
## poly(crim, 2)1   -24.9266     5.0803  -4.907 1.43e-06 ***
## poly(crim, 2)2     4.8441     4.9742   0.974    0.331
## poly(rm, 2)1      53.1216     5.4679   9.715  < 2e-16 ***
## poly(rm, 2)2      53.4513     4.8648  10.987  < 2e-16 ***
## poly(lstat, 2)1 -84.7066     6.4831 -13.066  < 2e-16 ***
## poly(lstat, 2)2  24.1386     4.8913   4.935 1.25e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.321 on 347 degrees of freedom
## Multiple R-squared:  0.778,  Adjusted R-squared:  0.7742
## F-statistic: 202.7 on 6 and 347 DF,  p-value: < 2.2e-16
```

```
# Make predictions on the test set
predictions <- predict(poly_model, newdata = test_data)

# Calculate the RMSE value
rmse <- sqrt(mean((test_data$medv - predictions)^2))
rmse
```

```
## [1] 4.841934
```

```
# Plot the actual vs predicted values
ggplot(data = test_data, aes(x = medv, y = predictions)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red") +
  labs(x = "Actual Values", y = "Predicted Values", title = "Actual vs Predicted Values")
```



```r
# LASSO REGRESSION


# Load the required libraries
library(glmnet)
```

```r
library(ggplot2)


# Load the dataset
data <- read.csv("/Users/saimonica/Downloads/Boston.csv")
head(data)
```

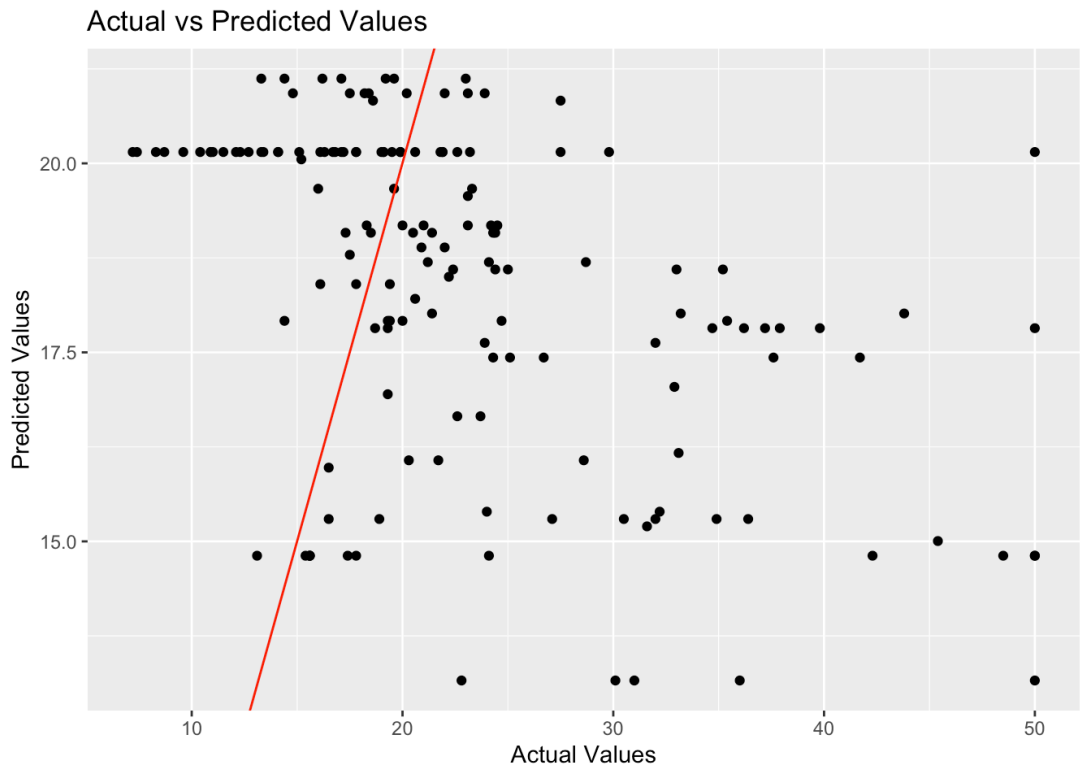
```
##   X    crim zn indus chas   nox    rm  age    dis rad tax ptratio  black lstat
## 1 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90  4.98
## 2 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90  9.14
## 3 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671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

```r
# Remove rows with missing values
data <- na.omit(data)

# Split the dataset into training and testing sets
set.seed(123)
train_index <- sample(1:nrow(data), 0.7*nrow(data))
train_data <- data[train_index, ]
test_data <- data[-train_index, ]

# Fit a Lasso regression model
lasso_model <- cv.glmnet(as.matrix(train_data[, -14]), train_data$ptratio, alpha = 1)

# Print the summary of the model
print(lasso_model)
```

```
##
## Call:  cv.glmnet(x = as.matrix(train_data[, -14]), y = train_data$ptratio,      alpha = 1)
##
## Measure: Mean-Squared Error
##
##      Lambda Index  Measure        SE Nonzero
## min 0.06235    39 0.003949 0.0002877       1
## 1se 0.06235    39 0.003949 0.0002877       1
```

```r
# Make predictions on the test set
predictions <- predict(lasso_model, newx = as.matrix(test_data[, -14]))

# Calculate the RMSE value
rmse <- sqrt(mean((test_data$medv - predictions)^2))
rmse
```

```r
# Plot the actual vs predicted values
ggplot(data = test_data, aes(x = medv, y = predictions)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red") +
  labs(x = "Actual Values", y = "Predicted Values", title = "Actual vs Predicted Values")
```




```
# RIGID REGRESSION


# Load the required libraries
library(glmnet)
library(ggplot2)

# Load the dataset
data <- read.csv("/Users/saimonica/Downloads/Boston.csv")
head(data)
```

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

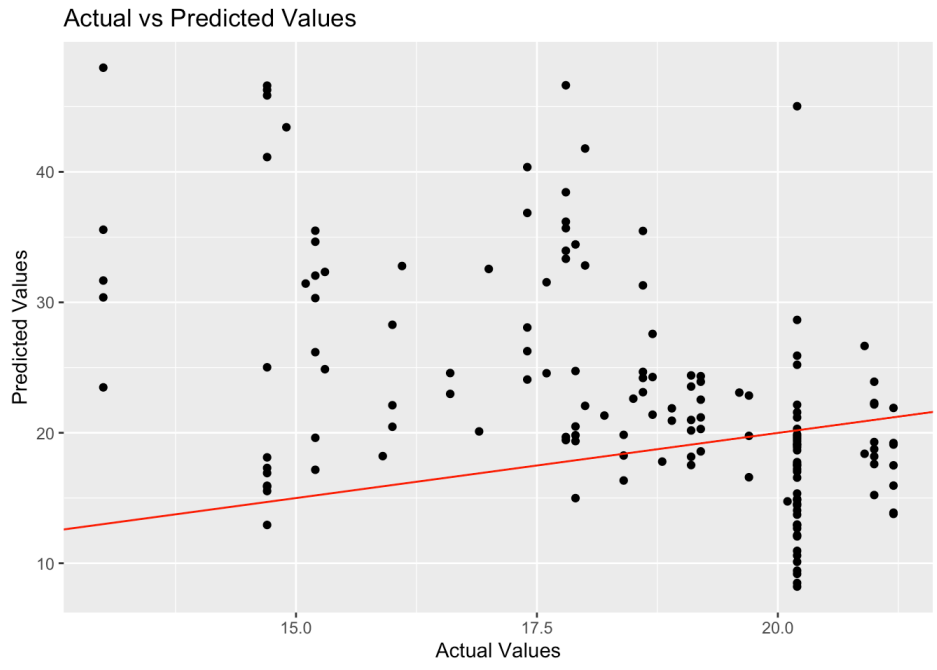
```r
# Remove rows with missing values
data <- na.omit(data)

# Split the dataset into training and testing sets
set.seed(123)
train_index <- sample(1:nrow(data), 0.7*nrow(data))
train_data <- data[train_index, ]
test_data <- data[-train_index, ]

# Fit a Ridge regression model
ridge_model <- cv.glmnet(as.matrix(train_data[, -14]), train_data$medv, alpha = 0)

# Print the summary of the model
print(ridge_model)
```

```
##
## Call:  cv.glmnet(x = as.matrix(train_data[, -14]), y = train_data$medv,      alpha = 0)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure     SE Nonzero
## min 0.9080   100   1.356 0.2321      14
## 1se 0.9965    99   1.539 0.2613      14
```

```r
# Make predictions on the test set
predictions <- predict(ridge_model, newx = as.matrix(test_data[, -14]))

# Calculate the RMSE value
rmse <- sqrt(mean((test_data$ptratio - predictions)^2))
rmse
```

```
## [1] 10.97409
```

```r
# Plot the actual vs predicted values
ggplot(data = test_data, aes(x = ptratio, y = predictions)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red") +
  labs(x = "Actual Values", y = "Predicted Values", title = "Actual vs Predicted Values")
```



```r
# MLP REGRESSION


# Load the required libraries
library(neuralnet)
```

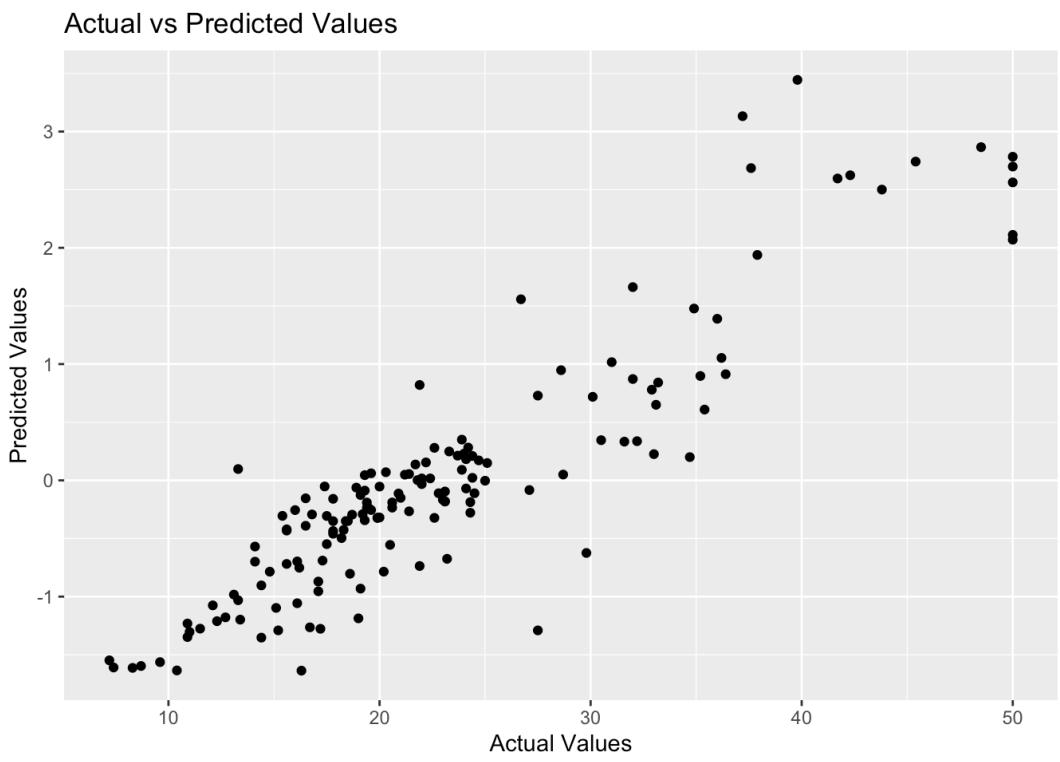
```r
library(ggplot2)


# Load the dataset
data <- read.csv("/Users/saimonica/Downloads/Boston.csv")
head(data)
```

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

```r
# Remove rows with missing values
data <- na.omit(data)

# Split the dataset into training and testing sets
set.seed(123)
train_index <- sample(1:nrow(data), 0.7*nrow(data))
train_data <- data[train_index, ]
test_data <- data[-train_index, ]

# Scale the data
train_data_scaled <- scale(train_data[, -14])
test_data_scaled <- scale(test_data[, -14])

# Fit an MLP regression model
mlp_model <- neuralnet(medv ~ ., data = train_data_scaled, hidden = c(5, 3), linear.output = TRUE)

# Make predictions on the test set
predictions <- predict(mlp_model, test_data_scaled)

# Calculate the RMSE value
rmse <- sqrt(mean((test_data$medv - predictions)^2))
rmse
```

```
## [1] 24.5683
```

```r
# Plot the actual vs predicted values
ggplot(data = test_data, aes(x = medv, y = predictions)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red") +
  labs(x = "Actual Values", y = "Predicted Values", title = "Actual vs Predicted Values")
```



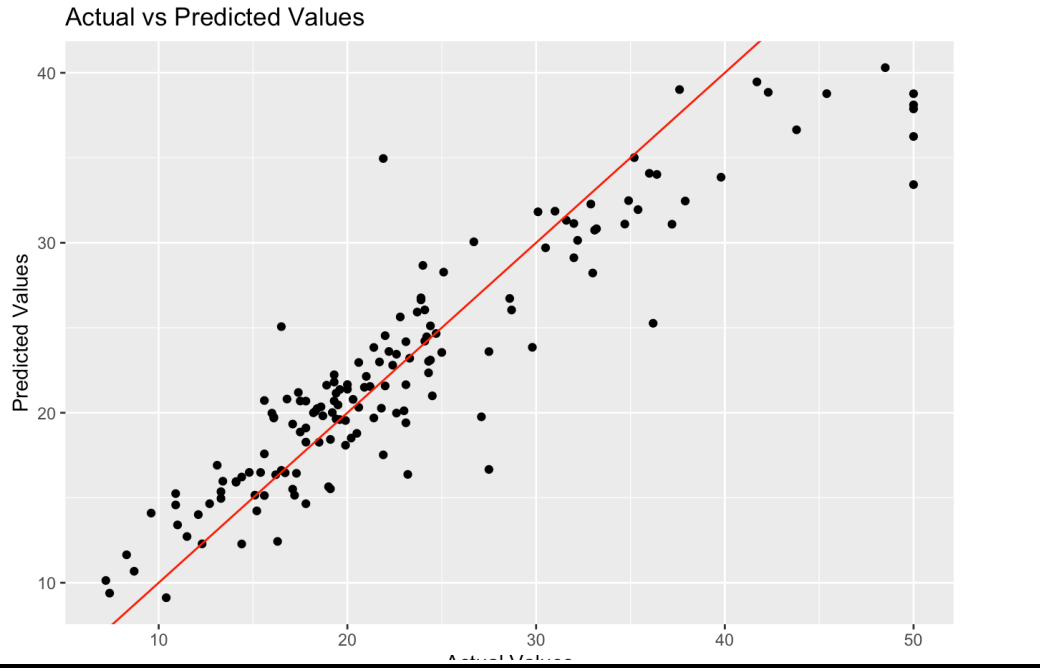

```
# COMBINATION OF RANDOM FOREST AND RIGID REGRESSION

# Load the required libraries
library(randomForest)
library(glmnet)
library(ggplot2)

# Load the dataset
data <- read.csv("/Users/saimonica/Downloads/Boston.csv")
head(data)
```

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

```r
# Remove rows with missing values
data <- na.omit(data)


# Split the dataset into training and testing sets
set.seed(123)
train_index <- sample(1:nrow(data), 0.7*nrow(data))
train_data <- Boston[train_index, ]
test_data <- Boston[-train_index, ]

# Fit a Random Forest model
rf_model <- randomForest(medv ~ ., data = train_data, ntree = 500)

# Extract the predicted values from the Random Forest model
rf_predictions <- predict(rf_model, newdata = test_data)

# Fit a Ridge regression model
ridge_model <- cv.glmnet(as.matrix(train_data[, -14]), train_data$medv, alpha = 0)

# Extract the predicted values from the Ridge regression model
ridge_predictions <- predict(ridge_model, newx = as.matrix(test_data[, -14]))

# Combine the predictions from the Random Forest and Ridge regression models
combined_predictions <- (rf_predictions + ridge_predictions) / 2


# Calculate the RMSE value
rmse <- sqrt(mean((test_data$medv - combined_predictions)^2))
rmse
```

```
## [1] 4.028352
```

```r
# Load the required libraries
library(ggplot2)

# Create a data frame of the actual and predicted values
results <- data.frame(actual = test_data$medv, predicted = combined_predictions)

# Create a scatter plot of the actual vs predicted values
ggplot(data = results, aes(x = actual, y = combined_predictions)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red") +
  labs(x = "Actual Values", y = "Predicted Values", title = "Actual vs Predicted Values")
```



Results :

ALGORITHM	RMSE %
Random forest Regression	4.34
Polynomial Regression	4.84
Lasso Regression	11.60
Regid Regression	10.97
Linear Regression	4.79
Hybrid Model (RFR & LR)	4.02

Contribution from each member of the team:

Sai Monica . R(20MID0165) and Pooja sree C. A. (20MID0204) together brainstormed to come up with the idea of " housing prediction " and coding of the models created and final documentation

Sai Monica . R – searched and analysed what all the ML algorithms that can used for the dataset and what are the Algorithms can be merged to get a better prediction (using R programming)

Pooja Sree C. A. – searched and analysed about all the existing research papers and helped to come up with the problem statement

SWOC Analysis :

Strengths:

- Assist the builder in determining a home's selling price and can assist the buyer in planning the ideal time to buy a home.
- Analysis of location according to the interest rates.
- Estimating the future values of real estate.

Weaknesses:

- Effects of a sudden change in pricing result in less accurate prediction.
- Predicting the prices of houses, which are continuous and real-valued outputs, is a difficulty.

Opportunities:

- New information and prediction technologies.
- Various ML and DL models for accurate prediction.

Challenges:

- Higher Prices of real estate in developed areas.

- More Maintenance costs for Properties with high value and monthly expenses to upkeep the property.

CONCLUSION :

From the above experiment it can be concluded that out of Random Forest regression , Polynomial Regression, Lasso Regression , Rigid Regression , Linear Regression ,MLP Regression, Decision Tree Regression , KNN Regression and the hybrid model (Linear Regression and Random Forest Regression)

We can say that **The Hybrid model** has lower RMSE value which makes it a better fit model

References :

1. Recommender Systems in the Real Estate Market—A Survey
2. Real Estate Data Marketplace.
3. Sampath Kumar et al. / Procedia Computer Science 57(2015).
4. E. Fix and J. L. Hodges Jr, “Discriminatory analysis nonparametric discrimination: consistency properties.
5. A. J. Smola and B. Sch¨olkopf, “A tutorial on support vector regression,” Statistics and Computing.
6. The future of prediction: how Google searchesforeshadow housing prices and quantities.
7. Lifelong property price prediction: A case study of the Toronto real estate market
8. Predicting house prices with spatial dependence: a comparison of alternative methods
9. Spatial and temporal dependence in house price prediction
10. Machine learning for property price predictionand price valuation: a systematic literature review.