DSA211 Statistical Learning with R

AY 23/24 Term 2
Section: G1
Professor: Kwong Koon Shing

Group Project

Group Members	Student ID
Ariel Wong Wei Lin	01465744
Tan Yu Xuan	01408987
Yip Yu Liang Adrian	01425210
Quek Jia Yang	01373129
Reese Lim Kaylee	01472009

Table of Contents

Prologue

We used the summary function on the Central dataset first and identified that *Tenure, Purchaser* and *Region* are categorical variables while the rest are numerical.

```
> summary(cen)
      Price.V1              Area             Age                Tenure
 Min.   :-1.021308   Min.   : 398.3   Min.   : 0.00   Freehold : 914
 1st Qu.:-0.563140   1st Qu.: 818.1   1st Qu.: 7.00   Leasehold:1586
 Median :-0.261263   Median :1141.0   Median :12.00
 Mean   : 0.000000   Mean   :1279.0   Mean   :13.54
 3rd Qu.: 0.218202   3rd Qu.:1517.7   3rd Qu.:19.00
 Max.   :10.979402   Max.   :7717.8   Max.   :45.00

   Purchaser              Region
 HDB    : 626    Bukit Timah:427
 Private:1874    Bukit Merah:306
                 Toa Payoh  :301
                 Kallang    :217
                 Bishan     :198
                 Queenstown :173
                 (Other)    :878
```

We construct all models on a sample size of 2000 for training, and use the remaining 500 to calculate Mean Squared Error (MSE) of the test set.

Models and Analyses

1. Multiple Linear and Polynomial Regression

Firstly, we run a multiple linear regression taking all variables into account using the lm function. We note that *Area*, *Age*, *Tenure,* and selected *Region* dummy variables are significant with a p-value less than 0.01, while *Purchaser* has a p-value of 0.9256. This seems to suggest that *Purchaser* is a less significant predictor variable in determining a change in *Price*.

We then run through multiple combinations of variables, by varying the terms to include inside interaction terms, and also varying the polynomials of *Area* and *Age*. Furthermore, for the sake of simplicity, we do not include *Region* inside the interaction terms at all. For example, we ran models with interaction terms of *Tenure* and *Area*, *Area* and *Age*, and *Age* and *Purchaser*. We also determined the best polynomial for *Area* and *Age* by running LOOCV on each of them by itself, and determined the best was $Area^3$ and Age^2. For a fairer comparison, we will compare the adjusted R-squared across every model to identify the best fitting model.

Overall, we identified that a multiple regression with the interaction terms of *Area* and *Tenure* is the best, giving rise to an adjusted R-squared of 0.8499. We also calculated the MSE to be 481090502793 for comparison between other models. The coefficients of the best model is as given in Appendix 1.

2. Best Subset Selection

We tried running a best subset selection with 10-fold cross-validation, where we choose a subset of k variables out of p that gives us the best model in terms of smallest MSE. Through cross-validation, we found the model with the smallest MSE of 507174939759 when k = 12 (Appendix 2). Best subset selection involves evaluating all possible combinations of variables (out of 18 variables), and when combined with cross-validation, its computational complexity is very high, since the process needs to be repeated for each fold of the cross-validation. Furthermore, with best subset selection, there is a risk of overfitting the model to the training data, especially when the number of variables is large compared to the number of observations. Finally, its MSE is worse than the initial polynomial multiple regression model. Hence, we do not choose to use the best subset selection for this problem.

3. Ridge Regression

We want to carry out a shrinkage approach to see if Ridge regression could be used to fit all predictors and constrain the coefficient estimates towards zero, to prevent overfitting. The first method we will use for this is ridge regression. Under linear regression, we estimate coefficients based on values that minimise residual sum of squares. Ridge regression also does this, but has an extra penalty term which is determined by the sum of squares of coefficients multiplied by a tuning parameter lambda. The penalty term allows us to take multicollinearity into account as it introduces bias to the model - as lambda increases, bias increases while variance decreases. We carry out cross validation to determine the best value of lambda with the lowest cross validation error. We end up with a lambda of 158285. This lambda then produces a model with an MSE of 519472000000 and coefficient estimates as shown in the figure in Appendix 3. This is worse than the best subset selection MSE = 507174939759. Additionally, the drawback of the ridge regression is that it will include all predictors in the final model. Hence, we also carried out the Lasso.

4. The Lasso

The Lasso can prevent overfitting while overcoming the disadvantage of ridge regression, as it also implements variable selection by shrinking coefficients to exactly 0. Similar to the ridge regression, the Lasso estimates coefficients based on values that minimise residual sum of squares. It also introduces a penalty term based on the absolute values of the coefficients multiplied by a tuning parameter lambda. The penalty term allows the Lasso regression to shrink the coefficients towards 0. When lambda is sufficiently large, coefficients can shrink to exactly 0, giving rise to variable selection. Likewise, cross validation is carried out to determine the best value of lambda with the lowest cross validation error. We get a lambda value of 1778.053 which produces a model with an MSE of 506696419155 and coefficient estimates as shown in Appendix 4. This MSE is lower than that of the model produced by ridge regression, and *Purchaser* is dropped in the final model of the Lasso, verifying its relative insignificance as suggested in the first proposed model.

5. Elastic Net Regression

Following ridge regression and the Lasso, we also decided to try a combination of both regularisation methods by using elastic net regression, which is useful when dealing with multicollinearity and overfitting issues. Given that alpha=0 is set for ridge regression and alpha=1 is set for Lasso, we initially experimented with elastic net regression by taking the middle value and setting alpha=0.5, which produces an MSE of 506676319402 with best value of lambda of 3240.192.

However, since there are different types of elastic net regressions — some being lasso-dominant and others more ridge-dominant — we explored further and considered other cases of 0 < alpha < 1, where alpha need not necessarily be in the middle. Therefore, we considered 2 other alpha values, alpha=0.75 (Lasso-dominant) and alpha=0.25 (Ridge-dominant). When alpha=0.75, it returns an MSE of 506795195894 with a best value of lambda of 2370.737. On the other hand, when alpha=0.25, the model returns an MSE value of 506447279753 and best lambda value of 5380.128.

Our findings show that this project is more suited to adopt a mixture of ridge regression and the Lasso with a more ridge-dominant approach, since the elastic net regression model performs best when alpha is set to 0.25 as it gives the smallest MSE. Appendix 5 shows the final model of elastic net regression, which excludes the *Purchaser* variable.

6. K-nearest Neighbours Algorithm

We then attempted a K-nearest neighbour (KNN) approach, which is a non-parametric method used for both classification & regression. KNN consists of using a distance measure (Euclidean distance, Manhattan, Mikowski & Hamming) to group objects into classes. In our case, we used Euclidean distance based on restrictions of R's KNN package. We chose K = 4 after testing, as higher values increase the MSE value. In regression, the predicted *Price* of the test data set is calculated as the average of the value of the 4 nearest data points in a class. Unlike other methods, KNN is susceptible to large differences in scale of data as, in our case *Area & Age*, where *Area* is especially high. This will cause unequal weight to be placed upon these variables during distance calculation, and choose to scale all predictor columns. The result is that even after scaling data for KNN & comparing it to scaled calculated MSE's of our other models, the MSE = 8800808162478.64 which was still the worst of all of them. The unscaled MSE = 9285272687764.77, which is still the worst. We decided not to pursue this method further. The scaled value can be obtained by uncommenting x[,2:6] <- scale(x[,2:6]).

7. Decision Tree

Given the poor performance of KNN, we decided to apply tree based methods as they are suitable for both regression & classification. Despite these being inferior in accuracy to supervised learning approaches, we believe a decision tree would be suitable for price

predictions, as it is able to handle non-linear & interaction effects, which based on earlier analysis of the linear model, show the highest adjusted R-squared with an interaction effect between *Area* & *Tenure*.
In addition, decision trees are able to handle the mixed data we have in our dataset as *Region Purchase* & *Tenure* are categorical as opposed to the other columns which are continuous. As before, we chose the scale of the *Price* column to have more readable MSE values.

In our first construction of the tree, there were 8 terminal nodes, as seen in Appendix 7, "Regression Tree for Central2024P data".

However, we need to consider the effect of an excessively large number of terminal nodes which generates complexity when fitting the data. This increases MSE and reduces test accuracy beyond a certain number of nodes. We will adopt cross-validation to prune the tree by identifying the largest number of terminal nodes, with the lowest deviance. Based on our codings, having 7 terminal nodes is the best as it gives the lowest deviance of 1.552796e+15. This is better in comparison to having 8 terminal nodes which gives a deviance of 1.565915e+15. See Appendix 7 Cross Validation: Deviance versus Size.

Given this, we now prune the original decision tree using the prune functions within R, and predict the *Price* with this pruned tree to the actual test prices. The result is an MSE of 6.30826e+11, with the tree seen in Appendix 7 "Pruned Regression Tree for Central Data", and the predictions in Appendix 7 "Pruned Tree predictions versus observed prices for test data".

We now rebuild a tree on the full dataset, with the constraint of maximum terminal nodes to 7, and predict its outputs, with comparison to the test dataset. The final tree can be referred to in appendix 7 "Pruned Regression Tree for all Central Data". The MSE = 625661162260, indicating the restriction to a maximum of 7 terminal nodes is ideal. This result is worse than aforementioned models such as multiple polynomial regression, Ridge, Lasso & Elastic Net Regression.

8. Random Forest Ensemble method

Several flaws exist with using decision trees, namely that it runs the risk of overfitting, and is sensitive to changes in training data, resulting in high variance. It is also a greedy algorithm and cannot guarantee the optimal tree.

Our solution is to use the Random Forest (RF) ensemble method. RF is a type of bagging method that involves building multiple decision trees on subsets of data randomly sampled with replacement from the train dataset. Multiple different trees are thus created, trained on a different subset of the training dataset. The final prediction is obtained by averaging predictions across the multiple decision trees created for the final prediction. This has several advantages compared to a single decision tree. Its use of aggregates from multiple trees prevents overfitting, so a more robust model is trained. Additionally, the usage of multiple trees accounts for multiple combinations, simulating new data, meaning it can generalise well to completely novel datasets. Finally, RF ensemble methods work well on high-dimension data, hence it can handle the high number of features we have arising due to the multiple locations we have in the *Region* variable.

In executing the random forest ensemble method, we decided to modify the number of trees and eventually choose 550 trees as increasing it further will decrease our MSE, but raise the risk of overfitting even with Random Forest. In our research the ideal variable count was the square root of our total features to be estimated (18), which we rounded to 4 for variables randomly sampled at each split. In our analysis, the MSE = 221844798581.846 which is the lowest thus far. The diagram of the Random Forest can be seen in Appendix 8 rf_model.

%incMSE refers to the increase in MSE that comes from permuting the values of a predictor variable. The higher the %incMSE, the more important the variable in predicting an accurate response vice versa.

IncNodePurity refers to the increase in node purity when splitting on a particular variable in construction of the tree, which measures homogeneity of observations. The higher the IncNodePurity the better the split, based on that variable, leading to better performance of the random forest model.

In addition, we have also identified the 3 most important variables based on %incMSE & IncNodePurity which are *Area, Region, Age & Tenure* in order of decreasing importance. The least important variable in predicting MSE is *Purchaser* in line with our earlier analysis.

Final Model Selection

Comparing our various models, we end up with the following MSE's across all of them:

1. Multiple Linear & Polynomial Regression: MSE = 481090502793
2. Best Subset Selection: MSE = 507174939759, k = 12
3. Ridge Regression: MSE = 519472000000, Lambda = 158285
4. The Lasso: MSE = 506696419155, Lambda = 1778.053
5. Elastic Net Regression: MSE = 506447279753, Lambda = 5380.128 (alpha = 0.25)
6. KNN: MSE = 1202820924505.22
7. Decision Tree: MSE = 625661162260
8. Random Forest: MSE = 221844798581.846, Trees = 550

Based on all our model training and evaluation, the recommended model for the dataset "Central2024P.csv" is the Random Forest ensemble method, as it has the lowest MSE of all our tested methods.

Appendix

Appendix 1 (Multiple linear and polynomial regression)

Code:

```
set.seed(9876)
cen <- read.csv("Central2024P.csv", stringsAsFactors = TRUE)
train <- sample(1:nrow(cen), 2000)
test <- (-train)
#Best Model according to Adjusted R^2
L3 <- lm(Price~Area*Tenure+Region+Age+Purchaser, cen[train,])
summary(L3)
pred3 <- predict(L3, newdata=cen[test,])
mean((pred3-cen[test, "Price"])^2)
coef(L3)

#Examples of other lm models
summary(lm(Price~.+I(Area^2)+I(Area^3), cen[train,]))
summary(lm(Price~Tenure*Purchaser+Region+Area+Age+I(Area^2), cen[train,]))
summary(lm(Price~Tenure*Purchaser+Region+Area+Age, cen[train,]))
```

Code Output:

```
> set.seed(9876)
> cen <- read.csv("Central2024P.csv", stringsAsFactors = TRUE)
> train <- sample(1:nrow(cen), 2000)
> test <- (-train)
> #Best Model according to Adjusted R^2
> L3 <- lm(Price~Area*Tenure+Region+Age+Purchaser, cen[train,])
> summary(L3)

Call:
lm(formula = Price ~ Area * Tenure + Region + Age + Purchaser,
    data = cen[train, ])

Residuals:
     Min       1Q   Median       3Q      Max
-5421625  -177132    10246   189176 10025849

Coefficients:
                         Estimate Std. Error t value Pr(>|t|)
(Intercept)             -40570.75   87435.09  -0.464 0.642692
Area                      2367.35      32.79  72.201  < 2e-16 ***
TenureLeasehold         233499.07   74386.36   3.139 0.001720 **
RegionBukit Merah       206888.80   71261.22   2.903 0.003734 **
RegionBukit Timah        77466.46   70699.90   1.096 0.273340
RegionGeylang           -70344.66   84950.66  -0.828 0.407734
RegionKallang            26012.77   78429.83   0.332 0.740174
RegionMarine Parade     284265.38   92756.38   3.065 0.002209 **
```

```
RegionNewton           1357491.90  118363.64  11.469  < 2e-16 ***
RegionNovena            166281.26   95616.58   1.739 0.082183 .
RegionOthers            777659.39  114782.88   6.775 1.63e-11 ***
RegionQueenstown        -49387.97   82469.45  -0.599 0.549332
RegionRiver Valley      975827.02  102839.68   9.489  < 2e-16 ***
RegionRochor             70015.97  128011.34   0.547 0.584474
RegionSouthern Islands  418430.77  116502.87   3.592 0.000337 ***
RegionTanglin           791742.64   89432.04   8.853  < 2e-16 ***
RegionToa Payoh        -115697.50   72654.57  -1.592 0.111447
Age                     -36376.26    1980.19 -18.370  < 2e-16 ***
PurchaserPrivate         17580.53   37595.88   0.468 0.640109
Area:TenureLeasehold      -463.02      48.81  -9.487  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 697500 on 1980 degrees of freedom
Multiple R-squared:  0.8513,	Adjusted R-squared:  0.8499
F-statistic: 596.7 on 19 and 1980 DF,  p-value: < 2.2e-16

> pred3 <- predict(L3, newdata=cen[test,])
> mean((pred3-cen[test, "Price"])^2)
[1] 481090502793
> coef(L3)
           (Intercept)                   Area        TenureLeasehold
           -40570.7492              2367.3548            233499.0720
     RegionBukit Merah      RegionBukit Timah          RegionGeylang
           206888.7958             77466.4604            -70344.6571
         RegionKallang    RegionMarine Parade           RegionNewton
            26012.7663            284265.3763           1357491.8971
          RegionNovena           RegionOthers       RegionQueenstown
           166281.2611            777659.3893            -49387.9679
    RegionRiver Valley           RegionRochor RegionSouthern Islands
           975827.0232             70015.9712            418430.7747
         RegionTanglin         RegionToa Payoh                    Age
           791742.6400           -115697.4961            -36376.2573
      PurchaserPrivate   Area:TenureLeasehold
            17580.5330              -463.0165

> #Examples of other lm models
> summary(lm(Price~.+I(Area^2)+I(Area^3), cen[train,]))

Call:
lm(formula = Price ~ . + I(Area^2) + I(Area^3), data = cen[train,
    ])

Residuals:
     Min       1Q   Median       3Q      Max
-4976789  -159611    17575   192216 10129066

Coefficients:
                         Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept)             7.058e+05  1.148e+05   6.146 9.57e-10 ***
Area                    1.452e+03  1.383e+02  10.496  < 2e-16 ***
Age                    -3.388e+04  2.121e+03 -15.974  < 2e-16 ***
TenureLeasehold        -3.589e+05  4.160e+04  -8.629  < 2e-16 ***
PurchaserPrivate        6.962e+03  3.823e+04   0.182  0.85552    
RegionBukit Merah       1.192e+05  7.221e+04   1.651  0.09889 .  
RegionBukit Timah       3.935e+04  7.182e+04   0.548  0.58381    
RegionGeylang          -9.376e+04  8.648e+04  -1.084  0.27840    
RegionKallang          -9.761e+03  7.981e+04  -0.122  0.90268    
RegionMarine Parade     3.006e+05  9.425e+04   3.189  0.00145 ** 
RegionNewton            1.428e+06  1.204e+05  11.857  < 2e-16 ***
RegionNovena            1.221e+05  9.708e+04   1.257  0.20880    
RegionOthers            7.427e+05  1.165e+05   6.372 2.31e-10 ***
RegionQueenstown       -6.246e+04  8.437e+04  -0.740  0.45921    
RegionRiver Valley      1.040e+06  1.041e+05   9.989  < 2e-16 ***
RegionRochor            2.274e+04  1.304e+05   0.174  0.86160    
RegionSouthern Islands  1.243e+05  1.155e+05   1.076  0.28208    
RegionTanglin           8.196e+05  9.079e+04   9.028  < 2e-16 ***
RegionToa Payoh        -9.068e+04  7.383e+04  -1.228  0.21951    
I(Area^2)               2.788e-01  5.409e-02   5.154 2.80e-07 ***
I(Area^3)              -2.660e-05  5.530e-06  -4.811 1.62e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 708400 on 1979 degrees of freedom
Multiple R-squared:  0.8467,	Adjusted R-squared:  0.8452 
F-statistic: 546.6 on 20 and 1979 DF,  p-value: < 2.2e-16

> summary(lm(Price~Tenure*Purchaser+Region+Area+Age+I(Area^2), cen[train,]))

Call:
lm(formula = Price ~ Tenure * Purchaser + Region + Area + Age + 
    I(Area^2), data = cen[train, ])

Residuals:
     Min       1Q   Median       3Q      Max 
-4751055  -174191    20108   196775 10485363 

Coefficients:
                                   Estimate Std. Error t value Pr(>|t|)    
(Intercept)                       2.854e+05  1.037e+05   2.752  0.00597 ** 
TenureLeasehold                  -2.446e+05  7.598e+04  -3.219  0.00131 ** 
PurchaserPrivate                  1.091e+05  6.955e+04   1.569  0.11688    
RegionBukit Merah                 1.477e+05  7.250e+04   2.038  0.04170 *  
RegionBukit Timah                 5.506e+04  7.215e+04   0.763  0.44545    
RegionGeylang                    -7.188e+04  8.682e+04  -0.828  0.40779    
RegionKallang                     1.667e+04  8.012e+04   0.208  0.83519    
RegionMarine Parade               2.893e+05  9.470e+04   3.055  0.00228 ** 
RegionNewton                      1.428e+06  1.211e+05  11.786  < 2e-16 ***
RegionNovena                      1.399e+05  9.760e+04   1.433  0.15193    
RegionOthers                      7.461e+05  1.171e+05   6.369 2.36e-10 ***
```

```
RegionQueenstown                 -2.149e+04  8.433e+04  -0.255  0.79888
RegionRiver Valley                1.051e+06  1.046e+05  10.049  < 2e-16 ***
RegionRochor                      6.355e+04  1.309e+05   0.486  0.62733
RegionSouthern Islands            1.725e+05  1.162e+05   1.485  0.13782
RegionTanglin                     8.413e+05  9.112e+04   9.233  < 2e-16 ***
RegionToa Payoh                  -6.699e+04  7.402e+04  -0.905  0.36553
Area                              2.050e+03  6.149e+01  33.336  < 2e-16 ***
Age                              -3.625e+04  2.076e+03 -17.464  < 2e-16 ***
I(Area^2)                         2.358e-02  1.106e-02   2.132  0.03314 *
TenureLeasehold:PurchaserPrivate -1.508e+05  8.220e+04  -1.835  0.06668 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 711900 on 1979 degrees of freedom
Multiple R-squared:  0.8452,   Adjusted R-squared:  0.8436
F-statistic: 540.3 on 20 and 1979 DF,  p-value: < 2.2e-16

> summary(lm(Price~Tenure*Purchaser+Region+Area+Age, cen[train,]))

Call:
lm(formula = Price ~ Tenure * Purchaser + Region + Area + Age,
    data = cen[train, ])

Residuals:
     Min       1Q   Median       3Q      Max
-4659550  -180644    24109   202650 10509849

Coefficients:
                                   Estimate Std. Error t value Pr(>|t|)
(Intercept)                       194596.61   94628.45   2.056  0.03987 *
TenureLeasehold                  -236626.80   75961.09  -3.115  0.00187 **
PurchaserPrivate                  105668.47   69589.93   1.518  0.12906
RegionBukit Merah                 149887.88   72554.42   2.066  0.03897 *
RegionBukit Timah                  57999.18   72201.55   0.803  0.42190
RegionGeylang                     -62616.77   86784.86  -0.722  0.47068
RegionKallang                      21985.28   80154.67   0.274  0.78389
RegionMarine Parade               284172.02   94759.00   2.999  0.00274 **
RegionNewton                     1407774.85  120886.85  11.645  < 2e-16 ***
RegionNovena                      141711.48   97686.89   1.451  0.14703
RegionOthers                      737737.18  117176.81   6.296 3.75e-10 ***
RegionQueenstown                   -6713.63   84116.83  -0.080  0.93639
RegionRiver Valley               1054502.52  104700.21  10.072  < 2e-16 ***
RegionRochor                       79489.36  130792.46   0.608  0.54342
RegionSouthern Islands            133005.29  114835.34   1.158  0.24691
RegionTanglin                     846677.89   91163.10   9.288  < 2e-16 ***
RegionToa Payoh                   -57367.21   73943.62  -0.776  0.43795
Area                                2168.61      25.91  83.709  < 2e-16 ***
Age                               -37274.15    2020.67 -18.446  < 2e-16 ***
TenureLeasehold:PurchaserPrivate -154727.89   82250.28  -1.881  0.06009 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 712600 on 1980 degrees of freedom
Multiple R-squared:  0.8448,	Adjusted R-squared:  0.8434 
F-statistic: 567.4 on 19 and 1980 DF,  p-value: < 2.2e-16
```

Appendix 2 (Best Subset Selection)

Code:

```
set.seed(9876)
central <- read.csv("Central2024P.csv", stringsAsFactors = TRUE)
library(leaps)
regfit2 <- regsubsets(Price~., data=central, nvmax=18)
regfit2.summary <- summary(regfit2)
plot(regfit2.summary$adjr2, main="Adjusted r^2 plot", xlab="Number of variables", ylab="Adjusted r^2", type="b")
plot(regfit2.summary$cp, main="Cp plot", xlab="Number of variables", ylab="cp", type="b")
plot(regfit2.summary$bic, main="BIC plot", xlab="Number of variables", ylab="BIC", type="b")
b <- which.max(regfit2.summary$adjr2)
c <- which.min(regfit2.summary$cp)
d <- which.min(regfit2.summary$bic)
# Model based on adjusted R square criteria
rsq <- coef(regfit2, b)
print(rsq)
# Model based on Cp criteria
cp <- coef(regfit2, c)
print(cp)
# Model based on BIC criteria
bic <- coef(regfit2, d)
print(bic)

# 10-fold cross validation on best subset
predict.regsubsets <- function(object, newdata, id){
  form <- as.formula(object$call[[2]])
  mat <- model.matrix(form, newdata)
  coefi <- coef(object, id=id)
  xvars <- names(coefi)
  mat[, xvars]%*%coefi
}

k <- 10
set.seed(9876)
folds <- sample(1:k, nrow(central), replace=TRUE)
cv.errors <- matrix(NA, k, 18, dimnames=list(NULL, paste(1:18)))
for (j in 1:k) {
```

```
  best.fit <- regsubsets(Price~., data=central[folds!=j,], nvmax=18)
  for (i in 1:18){
    pred <- predict.regsubsets(best.fit, central[folds==j,], id=i)
    cv.errors[j,i] <- mean((central$Price[folds==j]-pred)^2)
  }
}
# Test error
mean.cv <- apply(cv.errors, 2, mean)
min(mean.cv)
# Model with lowest cross validation error
bb <- which.min(mean.cv)
bssmod <- coef(regfit2, bb)
print(bssmod)
```

Code Output:

```
> set.seed(9876)
> central <- read.csv("Central2024P.csv", stringsAsFactors = TRUE)
> library(leaps)
> regfit2 <- regsubsets(Price~., data=central, nvmax=18)
> regfit2.summary <- summary(regfit2)
> plot(regfit2.summary$adjr2, main="Adjusted r^2 plot", xlab="Number of
variables", ylab="Adjusted r^2", type="b")
> plot(regfit2.summary$cp, main="Cp plot", xlab="Number of variables",
ylab="cp", type="b")
> plot(regfit2.summary$bic, main="BIC plot", xlab="Number of variables",
ylab="BIC", type="b")
> b <- which.max(regfit2.summary$adjr2)
> c <- which.min(regfit2.summary$cp)
> d <- which.min(regfit2.summary$bic)
> # Model based on adjusted R square criteria
> rsq <- coef(regfit2, b)
> print(rsq)
        (Intercept)                Area                 Age 
         304987.321            2169.022          -36864.128 
    TenureLeasehold  RegionBukit Merah       RegionGeylang 
        -354543.134          110757.155          -94923.726 
RegionMarine Parade        RegionNewton        RegionNovena 
         253148.400         1565328.975          119923.266 
       RegionOthers  RegionRiver Valley       RegionTanglin 
         749686.525          935075.025          809621.314 
    RegionToa Payoh 
         -87135.277 
> # Model based on Cp criteria
> cp <- coef(regfit2, c)
> print(cp)
        (Intercept)                Area                 Age 
         304987.321            2169.022          -36864.128 
    TenureLeasehold  RegionBukit Merah       RegionGeylang 
        -354543.134          110757.155          -94923.726 
RegionMarine Parade        RegionNewton        RegionNovena 
```

```
         253148.400        1565328.975         119923.266
       RegionOthers  RegionRiver Valley       RegionTanglin
         749686.525         935075.025         809621.314
    RegionToa Payoh
         -87135.277
> # Model based on BIC criteria
> bic <- coef(regfit2, d)
> print(bic)
        (Intercept)                Area                 Age
         288491.849            2173.528          -36113.552
    TenureLeasehold   RegionBukit Merah RegionMarine Parade
        -374889.486          130237.780          256243.812
       RegionNewton        RegionOthers  RegionRiver Valley
        1563601.519          757482.451          939286.494
      RegionTanglin
         807644.255
>
> # 10-fold cross validation on best subset
> predict.regsubsets <- function(object, newdata, id){
+  form <- as.formula(object$call[[2]])
+  mat <- model.matrix(form, newdata)
+  coefi <- coef(object, id=id)
+  xvars <- names(coefi)
+  mat[, xvars]%*%coefi
+ }
>
> k <- 10
> set.seed(9876)
> folds <- sample(1:k, nrow(central), replace=TRUE)
> cv.errors <- matrix(NA, k, 18, dimnames=list(NULL, paste(1:18)))
> for (j in 1:k) {
+  best.fit <- regsubsets(Price~., data=central[folds!=j,], nvmax=18)
+  for (i in 1:18){
+    pred <- predict.regsubsets(best.fit, central[folds==j,], id=i)
+    cv.errors[j,i] <- mean((central$Price[folds==j]-pred)^2)
+  }
+ }
> # Test error
> mean.cv <- apply(cv.errors, 2, mean)
> min(mean.cv)
[1] 507174939759
> # Model with lowest cross validation error
> bb <- which.min(mean.cv)
> bssmod <- coef(regfit2, bb)
> print(bssmod)
        (Intercept)                Area                 Age
         304987.321            2169.022          -36864.128
    TenureLeasehold   RegionBukit Merah       RegionGeylang
        -354543.134          110757.155          -94923.726
RegionMarine Parade        RegionNewton       RegionNovena
         253148.400         1565328.975         119923.266
       RegionOthers  RegionRiver Valley       RegionTanglin
         749686.525         935075.025         809621.314
```

```
    RegionToa Payoh
         -87135.277
```

Appendix 3 (Ridge Regression)

Code:

```
set.seed(9876)
central <- read.csv("Central2024P.csv", stringsAsFactors = TRUE)
attach(central)
train <- sample(1:nrow(central), 2000)
test <- (-train)
library(glmnet)
x <- model.matrix(Price~., central)[, -1]
y <- central$Price
centraltrain <- central[train,]
centraltest <- central[test,]
trainx <- model.matrix(Price~., centraltrain)[, -1]
trainy <- centraltrain$Price
testx <- model.matrix(Price~., centraltest)[, -1]
testy <- centraltest$Price
ridgemod <- glmnet(trainx, trainy, alpha = 0)
cvout <- cv.glmnet(trainx, trainy, alpha = 0)
lambdarr <- cvout$lambda.min
lambdarr

# Calculating test error
ridgepred <- predict(ridgemod, s = lambdarr, newx = x[test,])
mean((ridgepred-testy)^2)

# Ridge regression model
outrr <- glmnet(x, y, alpha = 0)
rrmodel <- predict(outrr, type = "coefficients", s = lambdarr)[1:19,]
rrmodel[rrmodel!=0]
```

Code Output:

```
> set.seed(9876)
> central <- read.csv("Central2024P.csv", stringsAsFactors = TRUE)
> attach(central)
> train <- sample(1:nrow(central), 2000)
> test <- (-train)
> library(glmnet)
> x <- model.matrix(Price~., central)[, -1]
> y <- central$Price
> centraltrain <- central[train,]
```

```
> centraltest <- central[test,]
> trainx <- model.matrix(Price~., centraltrain)[, -1]
> trainy <- centraltrain$Price
> testx <- model.matrix(Price~., centraltest)[, -1]
> testy <- centraltest$Price
> ridgemod <- glmnet(trainx, trainy, alpha = 0)
> cvout <- cv.glmnet(trainx, trainy, alpha = 0)
> lambdarr <- cvout$lambda.min
> lambdarr
[1] 158285
>
> # Calculating test error
> ridgepred <- predict(ridgemod, s = lambdarr, newx = x[test,])
> mean((ridgepred-testy)^2)
[1] 5.19472e+11
>
> # Ridge regression model
> outrr <- glmnet(x, y, alpha = 0)
> rrmodel <- predict(outrr, type = "coefficients", s = lambdarr)[1:19,]
> rrmodel[rrmodel!=0]
           (Intercept)                  Area                   Age
            529570.255              1924.381            -27406.529
       TenureLeasehold      PurchaserPrivate     RegionBukit Merah
           -371811.706             53870.582             54012.431
     RegionBukit Timah         RegionGeylang         RegionKallang
            -75430.938           -188425.461            -84047.714
   RegionMarine Parade          RegionNewton          RegionNovena
            166561.947           1491826.575             27036.165
          RegionOthers      RegionQueenstown     RegionRiver Valley
            614129.229           -138935.274            858131.708
          RegionRochor RegionSouthern Islands         RegionTanglin
           -123249.440            230595.753            694684.754
       RegionToa Payoh
           -153123.336
```

Appendix 4 (The Lasso)

Code:

```
set.seed(9876)
central <- read.csv("Central2024P.csv", stringsAsFactors = TRUE)
attach(central)
library(glmnet)
train <- sample(1:nrow(central), 2000)
test <- (-train)
x <- model.matrix(Price~., central)[, -1]
y <- central$Price
```

```
centraltrain <- central[train,]
centraltest <- central[test,]
trainx <- model.matrix(Price~., centraltrain)[, -1]
trainy <- centraltrain$Price
testx <- model.matrix(Price~., centraltest)[, -1]
testy <- centraltest$Price
lassomod <- glmnet(trainx, trainy, alpha = 1)
cvout1 <- cv.glmnet(trainx, trainy, alpha = 1)
lambdalasso <- cvout1$lambda.min
lambdalasso

# Test error
lassopred <- predict(lassomod, s = lambdalasso, newx = x[test,])
mean((lassopred-testy)^2)

# The lasso model
outlr <- glmnet(x, y, alpha = 1)
lrmodel <- predict(outlr, type = "coefficients", s = lambdalasso)[1:19,]
lrmodel[lrmodel!=0]
```

Code Output:

```
> set.seed(9876)
> central <- read.csv("Central2024P.csv", stringsAsFactors = TRUE)
> attach(central)
> library(glmnet)
> train <- sample(1:nrow(central), 2000)
> test <- (-train)
> x <- model.matrix(Price~., central)[, -1]
> y <- central$Price
> centraltrain <- central[train,]
> centraltest <- central[test,]
> trainx <- model.matrix(Price~., centraltrain)[, -1]
> trainy <- centraltrain$Price
> testx <- model.matrix(Price~., centraltest)[, -1]
> testy <- centraltest$Price
> lassomod <- glmnet(trainx, trainy, alpha = 1)
> cvout1 <- cv.glmnet(trainx, trainy, alpha = 1)
> lambdalasso <- cvout1$lambda.min
> lambdalasso
[1] 1778.053
>
> # Test error
> lassopred <- predict(lassomod, s = lambdalasso, newx = x[test,])
> mean((lassopred-testy)^2)
[1] 506696419155
>
> # The lasso model
> outlr <- glmnet(x, y, alpha = 1)
```

```
> lrmodel <- predict(outlr, type = "coefficients", s = lambdalasso)[1:19,]
> lrmodel[lrmodel!=0]
          (Intercept)                  Area                   Age
           310677.535              2160.767            -36086.039
      TenureLeasehold      RegionBukit Merah         RegionGeylang
          -354979.350             98973.823            -89809.369
  RegionMarine Parade          RegionNewton          RegionNovena
           234580.741           1546302.832            100410.572
         RegionOthers     RegionQueenstown   RegionRiver Valley
           725095.604            -19342.087            917529.075
        RegionRochor RegionSouthern Islands         RegionTanglin
            -9852.625             44343.389            791096.014
      RegionToa Payoh
           -81996.751
```

Appendix 5 (Elastic Net Regression)

The code below is run using alpha=0.25

Code:

```
central2024 <- read.csv("Central2024P.csv", stringsAsFactors = TRUE)
attach(central2024)
library(glmnet)
set.seed(9876)
train_index <- sample(1:nrow(central2024), 2000)
train_data <- central2024[train_index, ]
test_data <- central2024[-train_index, ]
x.train <- model.matrix(Price ~ Area + Age + Tenure + Purchaser + Region, data =
train_data)[, -1]
y.train <- train_data$Price

#set alpha = 0.5 for Elastic Net (0 for Ridge, 1 for Lasso)
elasticnet <- cv.glmnet(x.train, y.train, alpha = 0.25)

#cross validation results
plot(elasticnet)
bestlambda <- elasticnet$lambda.min
final_model <- glmnet(x.train, y.train, alpha = 0.5, lambda = bestlambda)

# use model for testing
x.test <- model.matrix(Price ~ Area + Age + Tenure + Purchaser + Region, data =
test_data)[,-1]
y.pred <- predict(final_model, newx = x.test)
MSE <- mean((test_data$Price - y.pred)^2)
MSE
bestlambda
model_coefficients <- coef(final_model)
print(model_coefficients)
```

Code output:

```
> central2024 <- read.csv("Central2024P.csv", stringsAsFactors = TRUE)
> attach(central2024)
> library(glmnet)
Loading required package: Matrix
Loaded glmnet 4.1-8
> set.seed(9876)
> train_index <- sample(1:nrow(central2024), 2000)
> train_data <- central2024[train_index, ]
> test_data <- central2024[-train_index, ]
> x.train <- model.matrix(Price ~ Area + Age + Tenure + Purchaser + Region,
data = train_data)[, -1]
> y.train <- train_data$Price

> #set alpha = 0.5 for Elastic Net (0 for Ridge, 1 for Lasso)
> elasticnet <- cv.glmnet(x.train, y.train, alpha = 0.25)

> #cross validation results
> plot(elasticnet)
> bestlambda <- elasticnet$lambda.min
> final_model <- glmnet(x.train, y.train, alpha = 0.5, lambda = bestlambda)

> # use model for testing
> x.test <- model.matrix(Price ~ Area + Age + Tenure + Purchaser + Region,
data = test_data)[,-1]
> y.pred <- predict(final_model, newx = x.test)
> MSE <- mean((test_data$Price - y.pred)^2)
> MSE
[1] 506447279753
> bestlambda
[1] 5380.128
> model_coefficients <- coef(final_model)
> print(model_coefficients)
19 x 1 sparse Matrix of class "dgCMatrix"
                                s0
(Intercept)             3.250865e+05
Area                    2.163697e+03
Age                    -3.642627e+04
TenureLeasehold        -3.606519e+05
PurchaserPrivate        .
RegionBukit Merah       9.293030e+04
RegionBukit Timah       2.464985e+00
RegionGeylang          -9.782176e+04
RegionKallang          -2.124992e+04
RegionMarine Parade     2.243633e+05
RegionNewton            1.354373e+06
RegionNovena            7.458192e+04
RegionOthers            6.692799e+05
RegionQueenstown       -4.151963e+04
RegionRiver Valley      9.969955e+05
RegionRochor            1.089719e+04
RegionSouthern Islands  6.277788e+04
```

```
RegionTanglin           7.852705e+05
RegionToa Payoh        -8.718872e+04
```

Appendix 6 (K-nearest Neighbours)

Code:

```
x <- read.csv("Central2024P.csv", stringsAsFactors = TRUE)
View(x)
library(class)

x$Tenure <- as.numeric(x$Tenure)
x$Purchaser <- as.numeric(x$Purchaser)
x$Region <- as.numeric(x$Region)
#x[,2:6] <- scale(x[,2:6])
View(x)
set.seed(9876)
num_folds <- 5

# Vector to store the mean squared errors for each fold
mse_cv <- numeric(num_folds)

# Perform k-fold cross-validation
for (i in 1:num_folds) {
  # Define the indices for the current fold
  fold_indices <- sample(1:nrow(x), size = nrow(x) / num_folds)

  # Split the data into training and validation sets
  train_fold <- x[-fold_indices, ]
  validation_fold <- x[fold_indices, ]

  # KNN model
  knn_model <- knn(train = train_fold[, -which(names(train_fold) == "Price")],
                      test = validation_fold[, -which(names(validation_fold) ==
"Price")],
                   cl = train_fold$Price,
                   k = 4)  # You can adjust the value of k as needed


  predictions <- as.numeric(knn_model)

  mse_cv[i] <- mean((predictions - validation_fold$Price)^2)
}

# Calculate the average Mean Squared Error across all folds
average_mse_cv <- mean(mse_cv)

print(paste("Average Mean Squared Error (Cross-Validation):", average_mse_cv))
```

Code Output:

```
> x <- read.csv("Central2024P.csv", stringsAsFactors = TRUE)
> library(class)
> x$Tenure <- as.numeric(x$Tenure)
> x$Purchaser <- as.numeric(x$Purchaser)
> x$Region <- as.numeric(x$Region)
> set.seed(9876)
> num_folds <- 5
> # Vector to store the mean squared errors for each fold
> mse_cv <- numeric(num_folds)
> # Perform k-fold cross-validation
> for (i in 1:num_folds) {
+  # Define the indices for the current fold
+  fold_indices <- sample(1:nrow(x), size = nrow(x) / num_folds)
+
+  # Split the data into training and validation sets
+  train_fold <- x[-fold_indices, ]
+  validation_fold <- x[fold_indices, ]
+
+  # KNN model
+    knn_model  <-  knn(train  =  train_fold[,  -which(names(train_fold)  ==
"Price")],
+                    test = validation_fold[, -which(names(validation_fold) ==
"Price")],
+                    cl = train_fold$Price,
+                    k = 4)  # You can adjust the value of k as needed
+
+
+  predictions <- as.numeric(knn_model)
+
+  mse_cv[i] <- mean((predictions - validation_fold$Price)^2)
+ }
> # Calculate the average Mean Squared Error across all folds
> average_mse_cv <- mean(mse_cv)
>    print(paste("Average    Mean    Squared    Error    (Cross-Validation):",
average_mse_cv))
[1] "Average Mean Squared Error (Cross-Validation): 9285272687764.77"
```

Appendix 7 (Decision Tree)

Code:

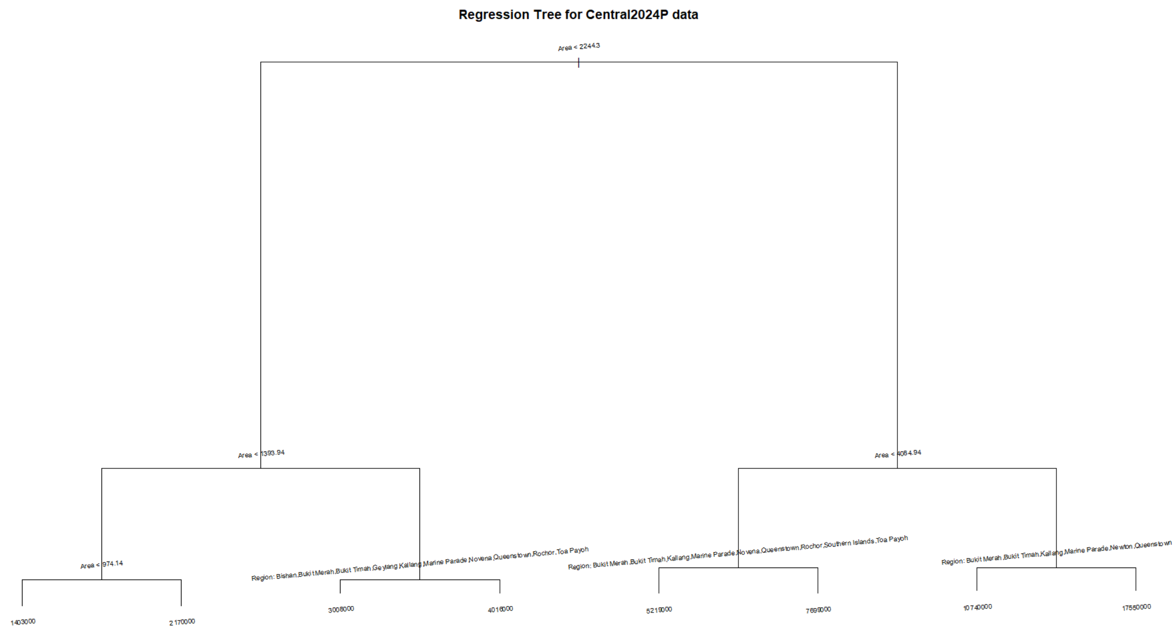
```
#Decision Tree Approach

x <- read.csv("Central2024P.csv", stringsAsFactors = TRUE)
View(x)
#x[,1] <- scale(x[,1])
#attach(x)
library(tree)
```

```
set.seed(9876)
train <- sample(1:nrow(x),2000)
test <- -train

#Initial Tree construction
tree.central <- tree(Price~., data = x, subset = train)
summary(tree.central)
tree.central
plot(tree.central)
title("Regression Tree for Central2024P data")
text(tree.central, pretty = 0, cex = 0.6, srt = 5)

#Finding minimum nodes through cv

cv.central <- cv.tree(tree.central)
cv.central
plot(cv.central$size, cv.central$dev, type="b", main="Cross validation: Deviance
versus Size",
     xlab="Number of terminal nodes", ylab="deviance")
minimum_nodes <- cv.central$size[which.min(cv.central$dev)]
minimum_nodes

#Pruning given minimum nodes

prune.central <- prune.tree(tree.central, best=minimum_nodes)
plot(prune.central)
title ("Pruned Regression Tree for central data")
text(prune.central, pretty=0, cex = 0.6, srt = 5)

predict <- predict(prune.central, newdata = x[test,])
central.test <- x[test, 'Price']

plot(predict, central.test, main="Pruned Tree prediction versus observed prices
for test data",
     xlab="predict Price", ylab="Observed Price")
mean((predict-central.test)^2)
```

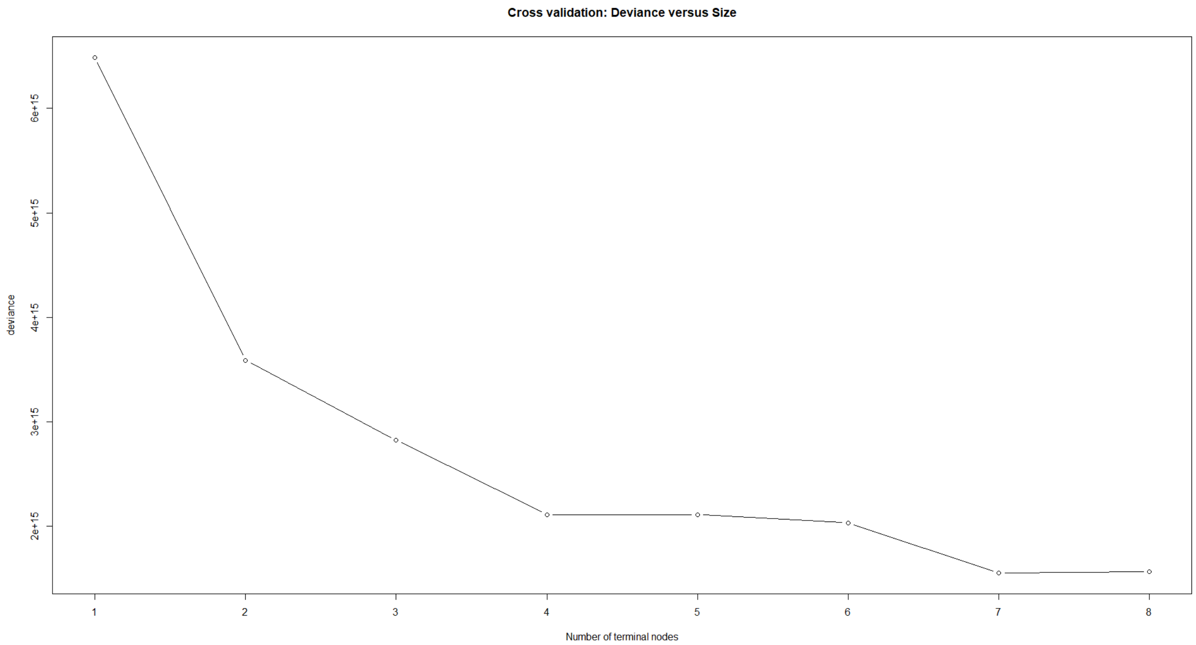
<u>Code Output:</u>

```
> x <- read.csv("Central2024P.csv", stringsAsFactors = TRUE)
> View(x)
> #x[,1] <- scale(x[,1])
> #attach(x)
> library(tree)
> set.seed(9876)
> train <- sample(1:nrow(x),2000)
> test <- -train
> #Initial Tree construction
> tree.central <- tree(Price~., data = x, subset = train)
> summary(tree.central)
```

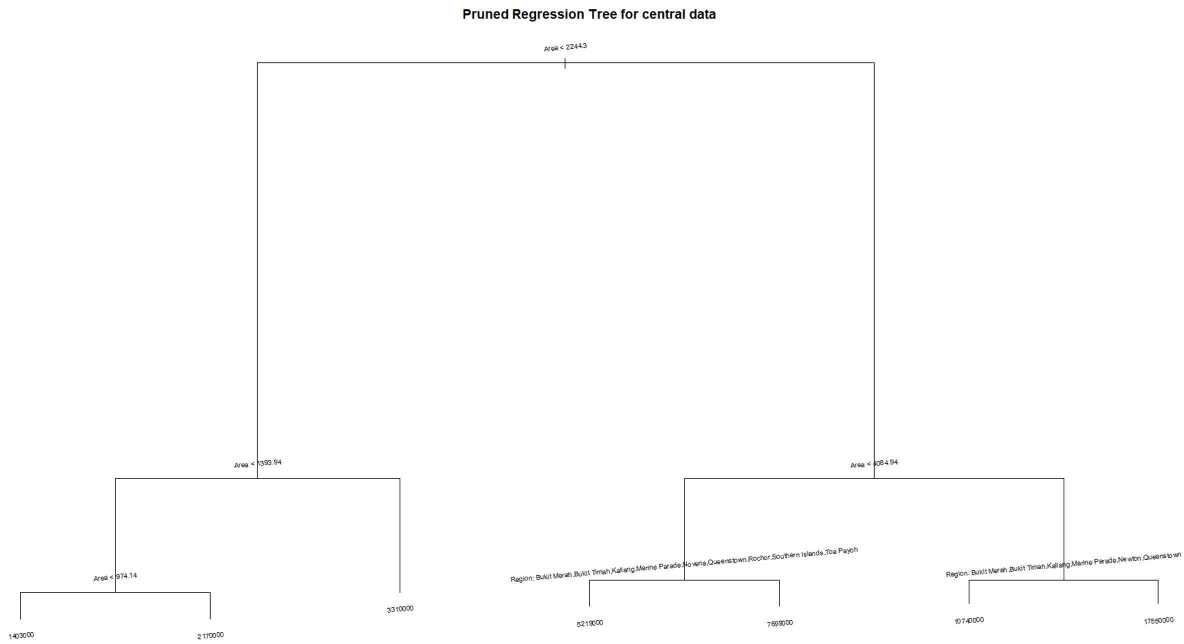
```
Regression tree:
tree(formula = Price ~ ., data = x, subset = train)
Variables actually used in tree construction:
[1] "Area"   "Region"
Number of terminal nodes:  8
Residual mean deviance:  5.499e+11 = 1.095e+15 / 1992
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-3919000  -353900   -36990        0   277200 10300000
> tree.central
node), split, n, deviance, yval
      * denotes terminal node

 1) root 2000 6.480e+15  2522000
   2) Area < 2244.3 1841 1.611e+15  2156000
     4) Area < 1393.94 1368 4.066e+14  1757000
       8) Area < 974.14 737 9.241e+13  1403000 *
       9) Area > 974.14 631 1.139e+14  2170000 *
     5) Area > 1393.94 473 3.557e+14  3310000
          10) Region: Bishan,Bukit Merah,Bukit Timah,Geylang,Kallang,Marine
Parade,Novena,Queenstown,Rochor,Toa Payoh 331 1.195e+14  3008000 *
        11) Region: Newton,Others,River Valley,Southern Islands,Tanglin 142
1.352e+14  4016000 *
   3) Area > 2244.3 159 1.760e+15  6765000
     6) Area < 4084.94 143 6.956e+14  6034000
                   12)   Region:   Bukit   Merah,Bukit   Timah,Kallang,Marine
Parade,Novena,Queenstown,Rochor,Southern   Islands,Toa   Payoh   96   1.800e+14
5219000 *
      13) Region: Newton,Others,River Valley,Tanglin 47 3.218e+14  7699000 *
     7) Area > 4084.94 16 3.063e+14 13290000
                   14)   Region:   Bukit   Merah,Bukit   Timah,Kallang,Marine
Parade,Newton,Queenstown 10 8.227e+13 10740000 *
      15) Region: River Valley,Tanglin 6 5.047e+13 17550000 *
> plot(tree.central)
> title("Regression Tree for Central2024P data")
> text(tree.central, pretty = 0, cex = 0.6, srt = 5)
```



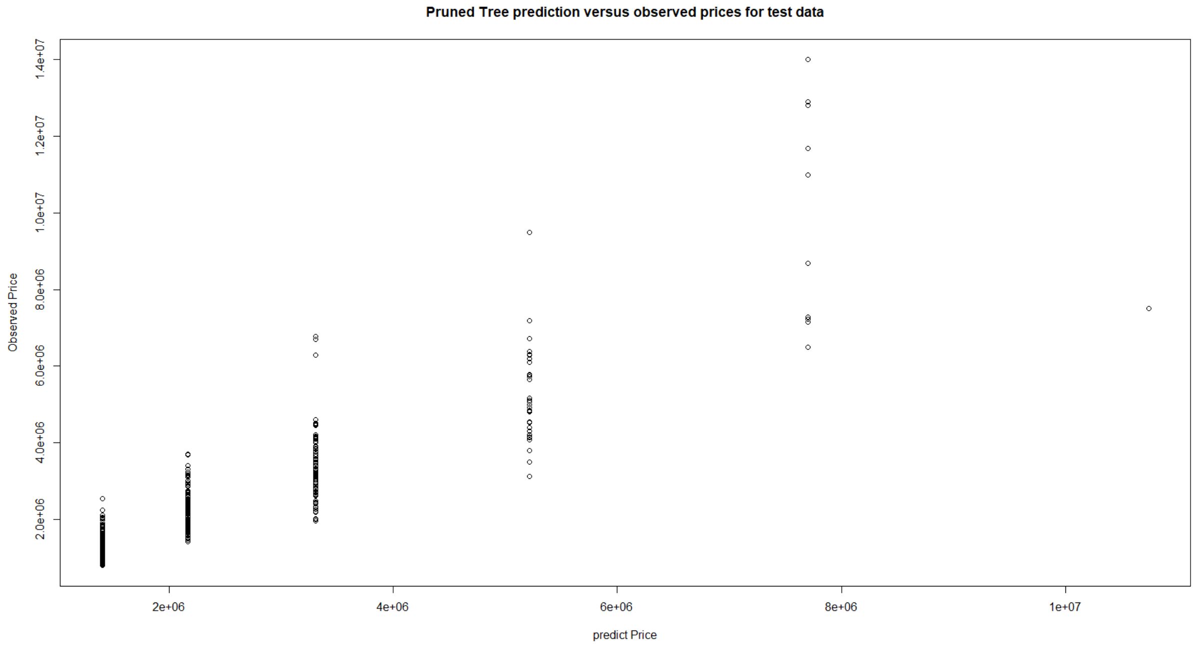

```
> cv.central <- cv.tree(tree.central)
> cv.central
$size
[1] 8 7 6 5 4 3 2 1

$dev
[1]  1.565915e+15  1.552796e+15  2.033781e+15  2.111384e+15  2.111384e+15
2.823611e+15 3.588331e+15
[8] 6.487844e+15

$k
[1]              -Inf  1.010539e+14  1.735927e+14  1.939137e+14  2.003051e+14
7.584704e+14 8.483383e+14
[8] 3.108814e+15

$method
[1] "deviance"

attr(,"class")
[1] "prune"         "tree.sequence"
>  plot(cv.central$size,  cv.central$dev,  type="b",  main="Cross  validation:
Deviance versus Size",
+     xlab="Number of terminal nodes", ylab="deviance")
> minimum_nodes <- cv.central$size[which.min(cv.central$dev)]
> minimum_nodes
[1] 7
```



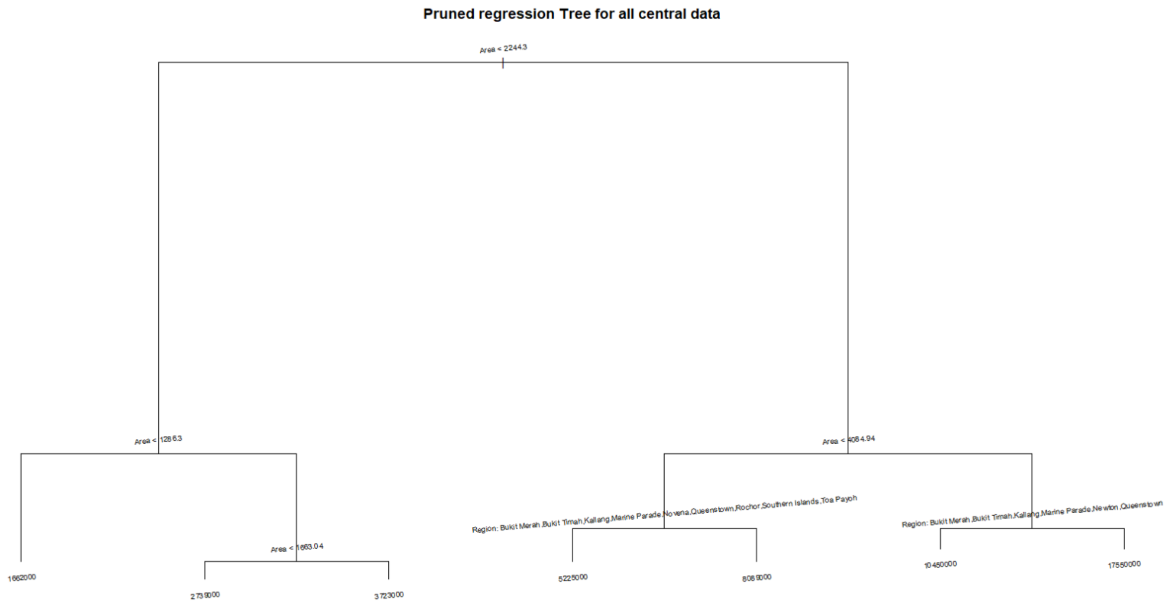

```
> prune.central <- prune.tree(tree.central, best=minimum_nodes)
> plot(prune.central)
> title ("Pruned Regression Tree for central data")
> text(prune.central, pretty=0, cex = 0.6, srt = 5)
```




```
> predict <- predict(prune.central, newdata = x[test,])
> central.test <- x[test, 'Price']
> plot(predict, central.test, main="Pruned Tree prediction versus observed
prices for test data",
+     xlab="predict Price", ylab="Observed Price")
> mean((predict-central.test)^2)
[1] 6.30826e+11
```




```
> #Build model with all data
> tree.centralall <- tree(Price~., data = x)
> prune.centralall <- prune.tree(tree.centralall, best = minimum_nodes)
> plot(prune.centralall)
> title("Pruned regression Tree for all central data")
> text(prune.centralall, pretty=0, cex = 0.6, srt = 5)
```




```
#Testing
> predict2 <- predict(prune.centralall, newdata = x[test,])
> central.test <- x[test, 'Price']
> mean((predict2-central.test)^2)
[1] 625661162260
```

Appendix 8 (Random Forest)

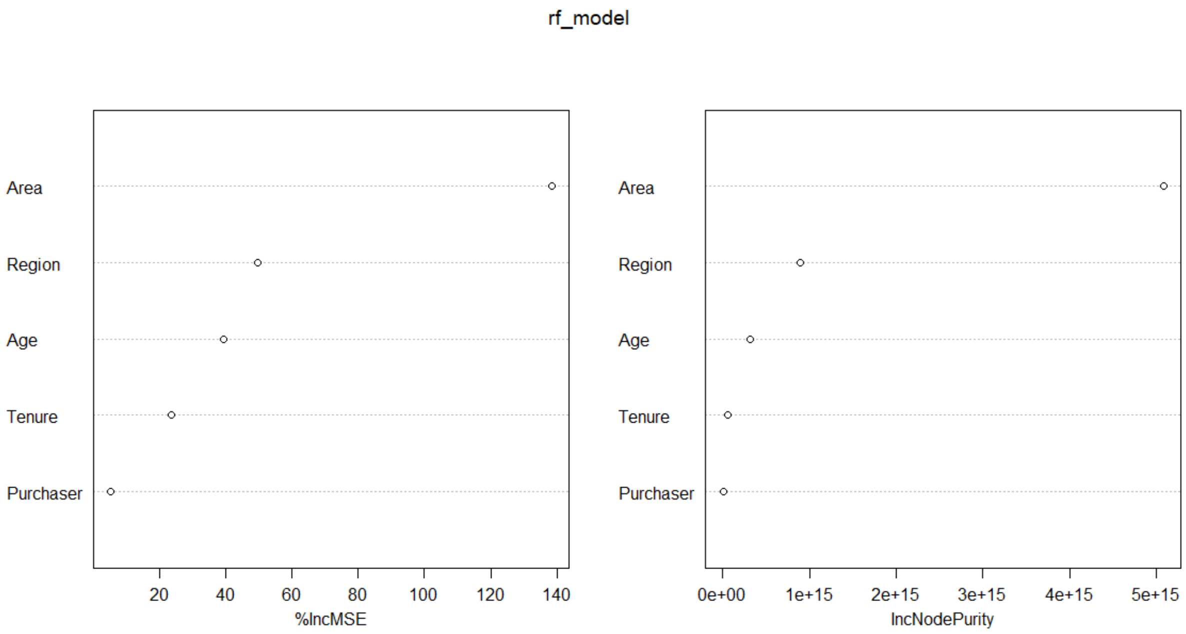
Code:

```
x <- read.csv("Central2024P.csv", stringsAsFactors = TRUE)
library(tree)
set.seed(9876)
train <- sample(1:nrow(x),2000)
test <- -train

#Attempting random forest
library(randomForest)
predictor_variables <- names(x)[-which(names(x) == "Price")]
target_variable <- "Price"

# Train the Random Forest model using only the training data
rf_model <- randomForest(
  formula = as.formula(paste(target_variable, "~ .")),
  data = x[train,],  # Use only the training data
  ntree = 550,  # Number of trees in the forest
  mtry = 4,  # Number of variables randomly sampled as candidates at each split
  importance = TRUE  # Calculate variable importance
)


# Print the model summary
print(rf_model)

# Predict on the test data
predictions <- predict(rf_model, newdata = x[test,])

# Calculate Mean Squared Error
mse <- mean((predictions - x[test,]$Price)^2)
print(paste("Mean Squared Error:", mse))

# Get variable importance measures
importance <- importance(rf_model)
print(importance)

# Plot variable importance
varImpPlot(rf_model)
```

Output Code:

```
> x <- read.csv("Central2024P.csv", stringsAsFactors = TRUE)
> #attach(x)
> library(tree)
> set.seed(9876)
```

```
> train <- sample(1:nrow(x),2000)
> test <- -train
> #Attempting random forest
> library(randomForest)
> predictor_variables <- names(x)[-which(names(x) == "Price")]
> target_variable <- "Price"
> # Train the Random Forest model using only the training data
> rf_model <- randomForest(
+  formula = as.formula(paste(target_variable, "~ .")),
+  data = x[train,],  # Use only the training data
+  ntree = 550,  # Number of trees in the forest
+   mtry = 4,   # Number of variables randomly sampled as candidates at each
split
+  importance = TRUE  # Calculate variable importance
+ )
> # Print the model summary
> print(rf_model)

Call:
 randomForest(formula = as.formula(paste(target_variable, "~ .")),      data
= x[train, ], ntree = 550, mtry = 4, importance = TRUE)
               Type of random forest: regression
                     Number of trees: 550
No. of variables tried at each split: 4

          Mean of squared residuals: 335182286517
                    % Var explained: 89.65
> # Predict on the test data
> predictions <- predict(rf_model, newdata = x[test,])
> # Calculate Mean Squared Error
> mse <- mean((predictions - x[test,]$Price)^2)
> print(paste("Mean Squared Error:", mse))
[1] "Mean Squared Error: 221844798581.846"
> # Get variable importance measures
> importance <- importance(rf_model)
> print(importance)
             %IncMSE IncNodePurity
Area      138.417424  5.082534e+15
Age        39.320319  3.205572e+14
Tenure     23.649001  6.350828e+13
Purchaser   5.333474  6.472522e+12
Region     49.823769  8.945840e+14
> # Plot variable importance
> varImpPlot(rf_model)
```


rf_model
Area
Region
Age
Tenure
Purchaser
20
40
60
80
100
120
140
%IncMSE
Area
Region
Age
Tenure
Purchaser
0e+00
1e+15
2e+15
3e+15
4e+15
5e+15
IncNodePurity