```
real_estate<-read.csv("/Users/nivedita_Christ/Downloads/Real estate valuation .csv")
library(psych)
library(DataExplorer)
library(car)#scatterplot, vif
library(lmtest) #Autocorrelation
library(MASS) #Step-wise regression
library(Metrics) #Loss/Cost function
library(glmnet)
library(dplyr)

#Structure
plot_str(real_estate)
str(real_estate)

#Descriptive Statistics
summary(real_estate)


#Distribution
plot_density(real_estate)
plot_correlation(real_estate)

#Missing Values
plot_missing(real_estate)

#Outlier Detection
boxplot_result<-boxplot(real_estate)
outlier<-boxplot_result$out
if (length(outliers) > 0) {
  print(paste("Outliers:", outliers))
} else {
  print("No outliers detected by IQR method.")
}
real_estate_clean <- real_estate[-outliers, ]
str(real_estate_clean)

#Duplicates
duplicates<-sum(duplicated(real_estate_clean))
print(duplicates)


#Data partitioning
set.seed(1234)
real_estate_mixed<-real_estate_clean[order(runif(413)),]
```

```
real_estate_training<-real_estate_mixed[1:289,]
real_estate_testing<-real_estate_mixed[289:413,]

#Build a full model
real_estate_lm_full<-lm(House.price.of.unit.area~.,data=real_estate_training)
summary(real_estate_lm_full)

#Best Fit
real_estate_step<-stepAIC(real_estate_lm_full,direction="backward")

#Build the Reduced Model
real_estate_reduced<-lm(House.price.of.unit.area ~ Transaction.date + House.age +
Distance.to.the.nearest.MRT.station +
                    Number.of.convenience.stores + Latitude, data=real_estate_training)
summary(real_estate_reduced)



#--------------------------------------------------------------------------
#Full Model diagnostics
plot(real_estate_lm_full)

#Residual vs fitted:
#!. Outward funneling effect
#2. Inward funneling effect
#3. Oscilating effect:
#4. U- shaped curve: Quadratic curve. (2 degree polynomial)

#Any point falling beyond the cook's distance point, there is an indication of an outlier.
#Variations- Assignanle or Random Cause of variation:

#Model diagnostics-2 VIF To Check Multi-collinearity.
#Check for multi-collinearity
vif(real_estate_lm_full)
#vif value should be strictly between 1 and 10. It is an indication of absence of multicollinearity.

#Auto-correlation of residuals (Durbin-Watson Test)
durbinWatsonTest(real_estate_lm_full)
dwtest(real_estate_lm_full)
#DW Statistics 2(1-ϱ) where row(ϱ ) stands for correlation
#where -1 <=ϱ<= +1
#if ϱ = -1, DW= 4, ϱ=0 DW= 2, ϱ=1 DW = 0

#CR Plot- Component Residual Plot to check linearity
```

```
crPlots(real_estate_lm_full)

#Non-constant Variance
ncvTest(real_estate_lm_full)
#H0: The errors have constant variance

#---------------------------------------------------------------------------------
#FULL MODEL PREDICTION
real_estate_prediction<-predict(real_estate_lm_full, newdata = real_estate_testing)
real_estate_prediction

#Actual Vs Predicted
newtest_pred<-cbind(real_estate_testing, real_estate_prediction)
head(newtest_pred)
mse(real_estate_testing$House.price.of.unit.area,real_estate_prediction)
------------------------------------------------------------------------------------------

#REDUCED MODEL PREDICTION
real_estate_prediction1<-predict(real_estate_reduced, newdata = real_estate_testing)
real_estate_prediction1

#Actual Vs Predicted
newtest_pred1<-cbind(real_estate_testing, real_estate_prediction1)
head(newtest_pred1)
mse(real_estate_testing$House.price.of.unit.area,real_estate_prediction1)
-------------------------------------------------------
# GRAPHS
#Predict using the full model
pred_full <- predict(real_estate_lm_full, newdata = real_estate_testing)
# Plot actual vs predicted
plot(real_estate_testing$House.price.of.unit.area, pred_full,
    main = "Actual vs Predicted (Full Model)",
    xlab = "Actual House Price",
    ylab = "Predicted House Price",
    col = "blue")
abline(0, 1, col = "red")  # Add a 45-degree line for reference


# Predict using the reduced model
  pred_reduced <- predict(real_estate_reduced, newdata = real_estate_testing)

 # Plot actual vs predicted
plot(real_estate_testing$House.price.of.unit.area, pred_reduced,
      main = "Actual vs Predicted (Reduced Model)",
```

```
      xlab = "Actual House Price",
      ylab = "Predicted House Price",
      col = "blue")
 abline(0, 1, col = "red")  # Add a 45-degree line for reference
-------------------------------------------------
pred_full <- predict(real_estate_lm_full, newdata = real_estate_testing)

 # Calculate MSE for full model
 mse_full <- mean((real_estate_testing$House.price.of.unit.area - pred_full)^2)

 # Calculate RMSE for full model
 rmse_full <- sqrt(mse_full)

 cat("Full Model:\n")
 cat(paste("Mean Squared Error (MSE):", mse_full, "\n"))
 cat(paste("Root Mean Squared Error (RMSE):", rmse_full, "\n\n"))

 # Predict using the reduced model on testing data
 pred_reduced <- predict(real_estate_reduced, newdata = real_estate_testing)

 # Calculate MSE for reduced model
 mse_reduced <- mean((real_estate_testing$House.price.of.unit.area - pred_reduced)^2)

 # Calculate RMSE for reduced model
 rmse_reduced <- sqrt(mse_reduced)

 cat("Reduced Model:\n")
 cat(paste("Mean Squared Error (MSE):", mse_reduced, "\n"))
 cat(paste("Root Mean Squared Error (RMSE):", rmse_reduced, "\n\n"))
---------------------------------------------------------------------------------
 #LASSO REGRESSION
  # Load required libraries
  library(glmnet)
 library(dplyr)

 # Assuming 'real_estate_clean' is your cleaned dataset
 # Create Model matrix (including dummy variables for State)
 X <- model.matrix(House.price.of.unit.area ~ ., data = real_estate_clean)
 Y <- real_estate_clean$House.price.of.unit.area

 # Define the lambda sequence (regularization parameter)
 lambda <- 10^seq(10, -2, length = 100)

 # Split the data into training and testing sets
```

```
set.seed(567)
part <- sample(2, nrow(X), replace = TRUE, prob = c(0.7, 0.3))
X_train <- X[part == 1, ] # Training Data
X_test <- X[part == 2, ]  # Testing Data
Y_train <- Y[part == 1]   # Training Data
Y_test <- Y[part == 2]    # Testing Data

# Fit Lasso regression model
lasso_reg <- glmnet(X_train, Y_train, alpha = 1, lambda = lambda)

# Perform cross-validation to select the best lambda (minimizing MSE)
lasso_reg_cv <- cv.glmnet(X_train, Y_train, alpha = 1)

# Get the optimal lambda
best_lambda <- lasso_reg_cv$lambda.min
print(paste("Optimal Lambda (min MSE):", best_lambda))

# Predict on the testing set using the best lambda
lasso_pred <- predict(lasso_reg, s = best_lambda, newx = X_test)

# Calculate Mean Squared Error (MSE)
mse <- mean((Y_test - lasso_pred)^2)
print(paste("Mean Squared Error (MSE):", mse))

# Calculate R-squared value
sst <- sum((Y_test - mean(Y_test))^2)
sse <- sum((Y_test - lasso_pred)^2)
r2 <- 1 - (sse / sst)
print(paste("R-squared (R^2):", r2))

# Summary of Lasso regression model
summary(lasso_reg)

# Predict on the testing set using the best lambda
lasso_pred <- predict(lasso_reg, s = best_lambda, newx = X_test)

# Create a data frame for actual and predicted values
predictions <- data.frame(Actual = Y_test, Predicted = lasso_pred)

# Plot actual vs predicted
plot(predictions$Actual, predictions$Predicted,
    main = "Actual vs Predicted (Lasso Regression)",
    xlab = "Actual House Price",
    ylab = "Predicted House Price",
```

```
     col = "black")
  abline(0, 1, col = "red")  # Add a 45-degree line for reference

  rmse <- sqrt(mse)
  cat("Root Mean Squared Error (RMSE):", rmse, "\n")
  n <- length(Y_test)
  p <- ncol(X_test) - 1  # Number of predictors excluding intercept
  adj_r2 <- 1 - (sse / (n - p - 1)) / (sst / (n - 1))
  cat("Adjusted R-squared:", adj_r2, "\n")
--------------------------------------------------------
   #RIDGE REGRESSION
   library(dplyr)
  library(glmnet)


  #View the first few rows of the dataset
  head(real_estate_clean)

  #Create Model matrix (including dummy variables for State)
  X<-model.matrix(House.price.of.unit.area ~., real_estate_clean)
  print(X)

  #Separate the target variable
  Y<-real_estate_clean$House.price.of.unit.area

  #Define the lambda sequence
  lambda<-10^seq(10,-2,length=100)
  print(lambda)

  #Split the data into training and validation sets
  set.seed(567)
  part<-sample(2, nrow(X), replace=TRUE, prob=c(0.7,0.3))
  X_train<-X[part==1, ] #Training Data
  X_test<-X[part==2, ] #Testing Data
  Y_train<-Y[part==1] #Training Data
  Y_test<-Y[part==2] #Testing Data

  #Perform Ridge Regression
  ridge_reg<-glmnet(X_train, Y_train, alpha=0, lambda=lambda)
  summary(ridge_reg)

  #Find the best lambda via cross-validation
  ridge_reg1<-cv.glmnet(X_train, Y_train, alpha=0)
  bestlam<-ridge_reg1$lambda.min
```

```
print(bestlam)

#Predict on the validation set
ridge.pred<-predict(ridge_reg, s=bestlam, newx=X_test)

#Calculate the mean squared error
mse1<-mean((Y_test - ridge.pred)^2)
print(paste("Mean Squared Error: ",mse))

#Calculate R squared Value
sst<-sum((Y_test - mean(Y_test))^2)
sse<-sum((Y_test - ridge.pred)^2)
r2<- 1 - (sse/sst)
print(paste("R square: ",r2))

# Predict on the testing set using the best lambda
ridge_pred <- predict(ridge_reg, s = bestlam, newx = X_test)

# Create a data frame for actual and predicted values
predictions <- data.frame(Actual = Y_test, Predicted = ridge_pred)

# Plot actual vs predicted
plot(predictions$Actual, predictions$Predicted,
    main = "Actual vs Predicted (Ridge Regression)",
    xlab = "Actual House Price",
    ylab = "Predicted House Price",
    col = "brown")
abline(0, 1, col = "red")  # Add a 45-degree line for reference


rmse1 <- sqrt(mse1)
cat("Root Mean Squared Error (RMSE):", rmse1, "\n")

# Calculate Adjusted R-squared
n <- length(Y_test)
p <- ncol(X_test) - 1  # Number of predictors excluding intercept
adj_r2 <- 1 - (sse / (n - p - 1)) / (sst / (n - 1))
cat("Adjusted R-squared:", adj_r2, "\n")
----------------------
```