# APPENDIX

```
## ----------------------------------Installing necessary libraries-------------------##
library(MASS)
library(ggplot2)
library(caTools)
library(tidyverse)
library(corrplot)
library(lubridate)
library(gridExtra)
library(GGally)
library(corrplot)
library(datarium)
library(ggcorrplot)
library(tidyverse)
library(caret)
library(e1071)
library(randomForest)
library(boot)
library(ipred)
library(dplyr)
library(RColorBrewer)
library(mice)
## --------------reading dataset-------------##
data<-read.csv(file="C:/Users/44776/Desktop/house-data.csv")
```

```
## ---------------TASK 1--------------------##

#Numerical summary of dataset

summary(data)

#Listing the datatype of variables in a dataset

str(data)

#count of row numbers and column numbers in a  dataset

dim(data)

#Viewing the first 5 values in a dataset

head(data)

#viewing the last 5 elements in a dataset

tail(data)

#Displaying column names in dataset

colnames(data)

# Classifying into  numerical and categorical values in a dataset

#Numerical variables selection

numerical_df<-select_if(data,is.numeric)

numerical_df<-data.frame(numerical_df)

head(numerical_df)

#Categorical variables selection

categorical_df<-select_if(data,is.character)

categorical_df<-data.frame(categorical_df)

colnames(categorical_df)

head(categorical_df)
```

```
#printing names of categorical columns

colnames(numerical_df)

#printing names of categorical columns

colnames(categorical_df)

#Counting the number of missing values in each column of dataset

null_values<- colSums(is.na(data))

null_values

#Replacing  NA in Categorical values with its given description

data$Alley<- data$Alley %>% replace(is.na(.),'No Alley Access')

data$PoolQC<- data$PoolQC %>% replace(is.na(.),"No Pool")

data$Fence<- data$Fence %>% replace(is.na(.),'"No Fence"')

data$BsmtQual<- data$BsmtQual %>% replace(is.na(.),'No Basement')

data$BsmtCond<- data$BsmtCond %>% replace(is.na(.),'No Basement')

data$GarageType<- data$GarageType %>% replace(is.na(.),'No Garage')

data$GarageCond<- data$GarageCond %>% replace(is.na(.),'No Garage')

data$MiscFeature<- data$MiscFeature %>% replace(is.na(.),'None')

#checking null values after setting categorical values with proper descriptions for NA

colSums(is.na(data))

# Calculation of mean and standard deviation

mean(data$SalePrice)

sd(data$SalePrice)

#Graphical summary of dataset

#scatterplot with variables Garage area and Sales price

plot(data$GarageArea, data$SalePrice)
```

```
#Plotting the quality of houses using barplot

barplot(table(data$OverallCond),

        main = " Current Condition of the most houses on the market",

        xlab = "Year",

        ylab = "Number of houses",

        col = brewer.pal(8, "PiYG"))

#Plotting the age of houses

barplot(table(data$YearBuilt),

        main = "Year of built in houses",

        xlab = "Year",

        ylab = "Number of houses",

        col = brewer.pal(12, "Greens"))

# Visualizing  median prices per neighborhood

neighborhoodsloc = tapply(data$SalePrice, data$Neighborhood, median)

neighborhoodsloc = sort(neighborhoodsloc, decreasing = TRUE)

dotchart(neighborhoodsloc, pch = 21, bg = "red",

         cex = 0.85,

         xlab="Average price of a house",

         main = "Predicting an affordable neighborhood in buying a house ")

#Filtering numerical values to calculate correlation matrix as correlation works only on
numbers

data_new=select_if(data, is.numeric)

#Filling null values using imputations method

imputations <- mice(data_new,method = 'pmm')

newdataframe=complete(imputations)
```

```
sapply(newdataframe, function(data) sum(is.na(data)))

#positive correlations are displayed in blue color and negative correlations in red color

corrplot(cor(newdataframe))

#calculation of correlation coefficients of different paired variables

cor(newdataframe$LotFrontage, newdataframe$SalePrice)

cor(newdataframe$YearBuilt,newdataframe$SalePrice)

cor(newdataframe$YrSold,newdataframe$SalePrice)

cor(newdataframe$OverallCond,newdataframe$SalePrice)

cor(newdataframe$OverallQual,newdataframe$SalePrice)

cor(newdataframe$BedroomAbvGr,newdataframe$SalePrice)

## ----------------------------------END TASK 1---------------------------------##

##-----------------------------------TASK 2-------------------------------------##

#checking null values after setting categorical values with proper descriptions for NA

colSums(is.na(data))

#Filling null values of two numerical columns using median method

data$LotFrontage[is.na(data$LotFrontage)] <- median(data$LotFrontage, na.rm = TRUE)

df<-data$LotFrontage

head(df)

data$MasVnrArea[is.na(data$MasVnrArea)] <- median(data$MasVnrArea, na.rm = TRUE)

df<-data$MasVnrArea

head(df)

#Categorical variables

categorical_df<-select_if(data,is.character)

categorical_df<-data.frame(categorical_df)
```

```
sum(is.na(categorical_df))

#Numerical Variables

numerical_df<-select_if(data,is.numeric)

numerical_df<-data.frame(numerical_df)

#Checking null values after filling

sum(is.na(numerical_df))

colnames(categorical_df)

#Converting categorical to numerical to fit in classification  machine learning models

design_matrix<- model.matrix(~ ., data = categorical_df)

encoded_cols<- design_matrix[, -1]

#Combined data and doing scaling

newdata<-cbind(numerical_df,encoded_cols)

dim(newdata)

#Categorizing overallcondition as poor,average and good

newdata$OverallCond<- cut(newdata$OverallCond,

                          breaks = c(0, 3, 6, 10),

                          labels = c("Poor", "Average", "Good"))

newdata$OverallCond

#splitting training and testing set

training.obs<- caret::createDataPartition(newdata$OverallCond, p = 0.8, list = FALSE)

train.df<- newdata[training.obs, ]

test.df<- newdata[-training.obs, ]

#Viewing the dimensionality of training and testing set

dim(train.df)
```

```
dim(test.df)

#Fitting the Logistic Regression model on the training set

set.seed(123)

LR<- nnet::multinom(OverallCond ~ ., data = train.df, family = "binomial")

#printing summary of logistic model

summary(LR)

#Predicting test data after logistic fitting

predicted.classes<- predict(LR, test.df)

head(predicted.classes)

#checking accuracy of predicted class using logistic model

mean(predicted.classes == test.df$OverallCond)

#Building a similar model to logistic regression

# Build the decision tree classifier, a similar model to Logistic Regression

tree_model<- rpart(OverallCond ~ ., data = train.df)

# Visualize the decision tree

rpart.plot(tree_model, main = "Decision Tree")

# Build the decision tree classifier

tree_model<- rpart(OverallCond ~ ., data = train.df)

# Make predictions on the testing set

predictions <- predict(tree_model, newdata = test.df, type = "class")

# Evaluate the performance of the decision tree model

confusion_matrix<- table(predictions, test.df$OverallCond)

accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)

accuracy
```

```
## -----------------------------END TASK 2-----------------------------##

##------------------------------TASK 3------------------------------------##

#Converting categorical data to numerical for fitting in models

design_matrix<- model.matrix(~ ., data = categorical_df)

encoded_cols<- design_matrix[, -1]

#Combining both numerical and categorical into single dataset

newdata<-cbind(numerical_df,encoded_cols)

#splitting training and testing set for model fitting

df<-newdata

#Scaling to stable coefficient estimates

scale(df)

trainingobs = sample.split(Y = df$SalePrice, SplitRatio = 0.8)

#subsetting into training data

train_data = df[trainingobs,]

#subsetting into testing data

test_data = df[!trainingobs,]

#checking dimensionality of training and testing set

dim(train_data)

dim(test_data)

#checking outliers

ggplot(data=train_data)+geom_boxplot(aes(x=BedroomAbvGr,y=SalePrice))

# Extracting outliers and creating a new tarin and test subsets from no  outliers data

outliers <- boxplot(train_data$SalePrice, plot = FALSE)$out

outliers_data<- train_data[which(train_data$SalePrice %in% outliers),]
```

```
train_data1 <- train_data[-which(train_data$SalePrice %in% outliers),]

# Create test set

set.seed(123) # for reproducibility

test_indices<- sample(nrow(train_data1), size = 0.2 * nrow(train_data))

test_data1 <- train_data1[test_indices, ]

#plotting data with and without outliers

par(mfrow=c(1, 2))

plot(train_data$BedroomAbvGr, train_data$SalePrice, main="With Outliers",
xlab="BedroomAbvGr", ylab="SalePrice", pch="*", col="green", cex=2)

abline(lm(SalePrice ~ BedroomAbvGr, data=train_data), col="red", lwd=3, lty=2)

# Plot of original data without outliers. Note the change of slope.

plot(train_data1$BedroomAbvGr, train_data1$SalePrice, main="Outliers removed",
xlab="BedroomAbvGr", ylab="SalePrice", pch="*", col="green", cex=2)

abline(lm(SalePrice ~BedroomAbvGr, data=train_data1), col="red", lwd=3, lty=2)

#Fitting model 1 the linear regression model

linearmodel = lm(SalePrice~., data=train_data1)

summary(linearmodel)

# Make predictions on the testing set

predictions1 <- predict(linearmodel, newdata = test_data1)

# Evaluate model 1

rmse1 <- sqrt(mean((predictions1 - test_data1$SalePrice)^2))

r_squared1 <- cor(predictions1, test_data1$SalePrice)^2

mae1 <- mean(abs(predictions1 - test_data1$SalePrice))

set.seed(123)

train_idx<- createDataPartition(data$SalePrice, p = 0.8, list = FALSE)
```

```
train <- data[train_idx, ]

test <- data[-train_idx, ]

# Fit the model 2, the random forest model

randomforestmodel<- randomForest(SalePrice ~ ., data = train, ntree = 500, mtry = 4,
importance = TRUE)

# Predict the sale price of houses in the test set

predictions2 <- predict(randomforestmodel, newdata = test)

# Evaluate model 2

rmse2 <- sqrt(mean((predictions2 - test$SalePrice)^2))

r_squared2 <- cor(predictions2, test$SalePrice)^2

mae2 <- mean(abs(predictions2 - test$SalePrice))

# Print the results of Model 1

cat("Model 1: Linear Regression\n")

cat("RMSE =", round(rmse1, 2), "\n")

cat("R-squared =", round(r_squared1, 2), "\n")

cat("MAE =", round(mae1, 2), "\n\n")

# Print the results of Model 2

cat("Model 2: Random Forest Model \n")

cat("RMSE =", round(rmse2, 2), "\n")

cat("R-squared =", round(r_squared2, 2), "\n")

cat("MAE =", round(mae2, 2), "\n\n")

cat("RMSE =", round(rmse2, 2), "\n")

cat("R-squared =", round(r_squared2, 2), "\n")

cat("MAE =", round(mae2, 2), "\n\n")
```

```
## -------------------------------RESAMPLING---------------------------------------------##

#Applying CV method in linear regression

# Create test set

set.seed(123) # for reproducibility

test_indices<- sample(nrow(train_data1), size = 0.2 * nrow(train_data))

test_data1 <- train_data1[test_indices, ]

lmmodel<- lm(SalePrice ~ ., data = train)

lmmodel

#predicting the test set

mylmpred<-function(object,newdata)

  predict(object,newdata=newdata,type=c("response"))

#printing RMSE error rate of linear model using CV and boot sampling methods

errorest(SalePrice ~.,data=train_data1,model=lm,estimator="cv",predict=mylmpred)

errorest(SalePrice ~.,data=train_data1,model=lm,estimator="boot",predict=mylmpred)

#Applying CV sampling in Random Forest

set.seed(123)

train_idx<- createDataPartition(data$SalePrice, p = 0.8, list = FALSE)

train <- data[train_idx, ]

test <- data[-train_idx, ]

randomforestmodel<- randomForest(SalePrice ~ ., data = train)

randomforestmodel

#predicting using sampling

mypred<-function(object,newdata)

  predict(object,newdata=newdata,type=c("response"))
```

```
#Printing RMSE error rate of Random Forest using both CV and boot sampling method

errorest(SalePrice ~.,data=train,model=randomForest,estimator="cv",predict=mypred)

errorest(SalePrice ~.,data=train,model=randomForest,estimator="boot",predict=mypred)

##--------------------------------------END TASK  3--------------------------------------##

## -------------------------------------TASK 4-------------------------------------------------##

#Predicting Garage Condition with selected significant features in housing dataset

#reading dataset

data<-read.csv(file="C:/Users/44776/Desktop/house-data.csv")

# Select the necessary columns and store them in a new dataset

new_data<- select(data, GarageCond, YearBuilt, GarageType, Fence, Neighborhood,
GarageArea)

#Printing the first few rows of newdata

#With Given description of Categorical values of NA in each categorical set,replacing is done

data<-new_data

data$Fence<- data$Fence %>% replace(is.na(.),"NoFence")

data$GarageType<- data$GarageType %>% replace(is.na(.),"NoGarage")

data$GarageCond<- data$GarageCond %>% replace(is.na(.),"NoGarage")

# Listing first few rows

head(data)

#Converting categorical variables to numbers

data$GarageCond<- as.factor(data$GarageCond)

data$GarageType<- as.factor(data$GarageType)

data$Neighborhood<- as.factor(data$Neighborhood)

data$Fence<-as.factor(data$Fence)

#Visualizing Garage condition values in an overall view
```

```
p <- ggplot(data, aes(x = GarageCond)) +
geom_bar() +
xlab("Garage Condition") +
ylab("Count") +
ggtitle("Distribution of Garage Conditions")
print(p)
# Creating scatter plots for each column to visualize each selected variables with target
Garage condition
for (col in colnames(data)) {
  if (col != "GarageCond") {
    p <- ggplot(data, aes(x = !!sym(col), y = GarageCond)) +
          geom_point() +
          ggtitle(paste("Garage Condition vs. ", col))
    print(p)
  }
}
set.seed(123)
#Splitting training and testing set
trainIndex<- createDataPartition(data$GarageCond, p = .8,
                                  list = FALSE,
                                  times = 1)
train <- data[trainIndex,]
test <- data[-trainIndex,]
```

```
#Fitting a SVM model to predict Garage Condition

svm_model<- svm(GarageCond ~
YearBuilt+GarageType+GarageArea+Fence+Neighborhood, data = train, kernel = "linear")

#Predicting the model with the test results

svm_predictions<- predict(svm_model, test)

#Printing the accuracy using confusion matrix

confusionMatrix(svm_predictions, test$GarageCond)

##-------------------------------END TASK 4--------------------------------------------------##
```