University of Essex
Department of Mathematical Sciences

MA321 - Applied Statistics

Rojitha Repalle (2201010)

Sharon Machado(2202055)

Yash Mhatre (2201772)

Supervisor: **Oludare Ariyo**

March 20, 2023

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

The price of houses increases every year, There is a need for a system that predicts the price and overall condition of a house based on given factors. This report presents an analysis of the given dataset describing the condition of a house based on factors such as the year built, overall quality, sale price, number of rooms, and many more. The primary aim of this analysis is to classify the overall condition of a house as Poor, Good, or Average, based on a rating scale ranging from 1 to 10, and to predict the sale price of the house using various machine learning models. Classification is a technique that helps to categorize data into distinct groups. The main intent of a classification problem is to determine which category the new data will belong to. The dataset is carefully evaluated for missing values as it captures critical information about the features of a house. The numerical values are imputed to keep the dataset's structure intact and ready for further processing.

Contribution of Team Members:

Question 1 - Yash Mhatre, Rojitha Repalle, Sharon Machado

Question 2 - Yash Mhatre, Rojitha Repalle, Sharon Machado

Question 3 - Yash Mhatre, Rojitha Repalle, Sharon Machado

Question 4 - Yash Mhatre, Rojitha Repalle, Sharon Machado

Report, Presentation - Yash Mhatre, Rojitha Repalle, Sharon Machado

CHAPTER 2

Descriptive Statistics of Dataset

The House dataset has 1460 records with 51 distinct features out of which 'SalePrice' is the primary response variable. The variable 'ID' is eliminated as it does not contribute to the model building. There are 23 numeric columns and 28 character columns. On careful observation, it can be seen that the numerical columns can be further classified as continuous and categorical variables. Four of the variables - Fence, MiscFeature, PoolQC, and Alley have more than 75% of missing values but these values are not null and carry meaningful information according to the descriptive description of the housing dataset. The Descriptive statistics of the numerical columns can be seen in Table 3.1.

The histogram for the response variable is shown in Figure 3.1. It can be seen that most of the houses are below 200000, indicating that most of the houses are in the lesser price range. The maximum sale price of the house in the dataset is 755000 and the minimum is 34900. It is evident that the count of houses decreased with the increase in price after 200000. Figure 3.4 displays the box plot of some of the variables with respect to the response variable. In the SaleCondition vs SalePrice plot, it can be seen that there are 6 varieties of sale conditions. The majority are of type partial and the fewer ones are of a type family. Outliers for all the types except AdjLand.

The boxplots are drawn for some variables to check for outliers in them. They are shown in fig 3.2

Table 2.1: Descriptive Statistics of numerical columns

Features	Min	1st Qu	Median	Mean	3rd Qu	Max
Id	1	365.8	730.5	730.5	1095.2	1460
LotFrontage	21	59	69	70.05	80	313
LotArea	1300	7554	9478	10517	11602	215245
OverallQual	1	5	6	6.099	7	10
OverallCond	1	5	5	5.575	6	9
YearBuilt	1872	1954	1973	1971	2000	2010
MasVnrArea	0	0	0	103.7	166	1600
TotalBsmtSF	0	795.8	991.5	1057.4	1298.2	6110
X1stFlrSF	334	882	1087	1163	1391	4692
X2ndFlrSF	0	0	0	347	728	2065
LowQualFinSF	0	0	0	5.845	0	572
GrLivArea	334	1130	1464	1515	1777	5642
FullBath	0	1	2	1.565	2	3
BedroomAbvGr	0	2	3	2.866	3	8
KitchenAbvGr	0	1	1	1.047	1	3
TotRmsAbvGrd	2	5	6	6.518	7	14
Fireplaces	0	0	0	0.613	1	3
GarageArea	0	334.5	480	473	576	1418
PoolArea	0	0	0	2.759	0	738
MiscVal	0	0	0	43.49	0	15500
MoSold	1	5	6	6.322	8	12
YrSold	2006	2007	2008	2008	2009	2010
SalePrice	34900	129975	163000	180921	214000	755000

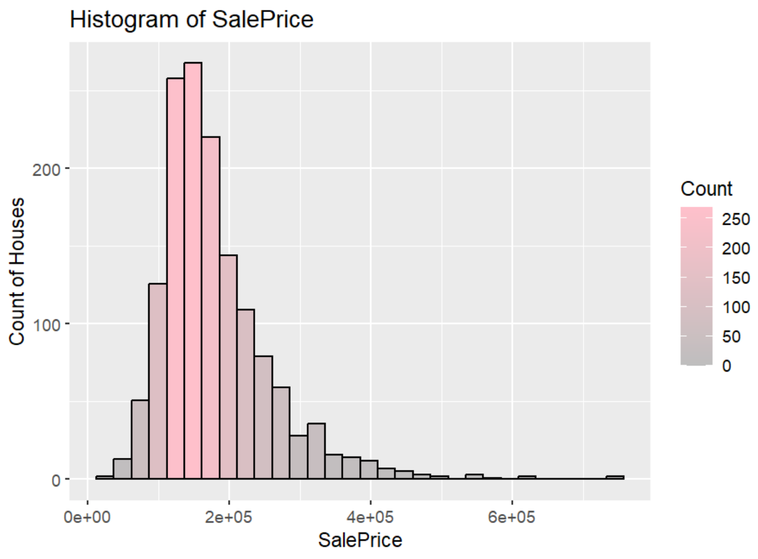

Figure 2.1: Histogram of SalePrice

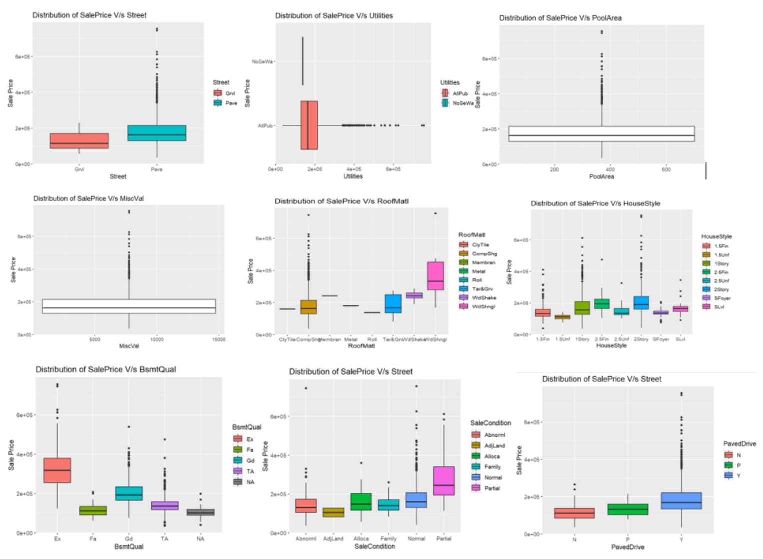

Figure 2.2: Boxplots of few predictor variables

2.1 Dealing with Missing Values

There are missing values present in the character and the numerical variables which cannot be eliminated as they carry meaningful explanations about the houses. The NAs present in the character columns is replaced with 'Not Available' to retain the original information. Two numerical columns - LotFrontage & MasVnrArea contained null values. Mice imputation is performed on these two features to impute the null values. Mice imputation is a statistical method used to fill in missing values in a dataset. The constraint for mice imputation is that it assumes that the missing data are missing at random (MAR) which is true with the dataset provided. The features - Street, PoolArea, MiscVal, Utilities, and RoofMat have categorical outliers and can be viewed in Figure 3.2 as well. Therefore, these variables are dropped and the rest of the predictor variables are considered for further analysis.

2.2 Dealing with Multicollinearity

Collinearity [1] is a situation where two or more variables are closely related to one another. The correlation matrix is performed on the continuous variables to find the highly correlated predictor variables as it can lead to the problem of multicollinearity and overfitting. The highly correlated variables having a threshold 0.6 is discarded. Variance Inflation Factors (VIFs) [2] measure the correlation among independent variables in the least squares regression

models. VIF 5 indicates that the features are highly correlated and can reduce the adequacy of the model. A linear model is instantiated with the continuous variables with respect to the response variable and VIF is calculated. The variables with VIF 10(X1stFlrSF, X2ndFlrSF, GrLivArea, TotalBsmtSF, LotFrontage, and LotArea) are eliminated and the resultant variables are stored in a character data frame. As the dataset consists of categorical and numerical data, it is important to transform the categorical columns to numeric factors in order to apply machine learning algorithms and train the dataset. Label Encoding [3] refers to converting the values in categorical character columns into a numeric form to be ready to be given into the machine-readable form. Label encoding is performed on the character columns and then the resultant is combined with the numerical columns using cbind() function.
The Chi-squared test [4] is used to find out the statistically significant variables and to check the relevance between two variables. The Chi-square uses a p-value which determines the significance of the features. The following categorical variables OverallQual, OverallCond, FullBath, BedroomAbvGr, TotRmsAbvGrd, GarageType, BsmtCond, KitchenAbvGr, Neighborhood, and ExterCond are found to be statistically significant by the Chi-square test as the p-value is less than 0.05. Thus, these columns are considered for our final dataset.

CHAPTER 3

Classification of Houses

The objective of classification is to find out the overall condition of the house. The 'Overall-Cond' rates the overall condition of the house. It is an ordinal categorical variable ranging from values between 1 to 10 describing the overall condition of the house from very poor to excellent. The houses are classified based on the given condition:

- Poor if the overall condition is between 1 to 3.
- Average if the overall condition is between 4 and 6.
- Good if the overall condition is between 7 and 10.

This classification methodology helps in simplifying the evaluation process and makes it easier to interpret the results. There are 1130 average, 299 good, and 31 poor condition houses after the classification as shown in Figure 4.1.

3.1 Logistic Regression Model

Logistic regression [5] estimates the probability of an event occurring. Multinomial logistic regression is implemented as there are more than two predictor variables in our model. The 'glment' method and multinomial family are used to build a logistic regression model. To ensure the reliability of the model, bootstrapping re-sampling is employed, which involved randomly selecting subsets of data from the original data set and then constructing a model on each subset. A multinomial logistic regression model is built using bootstrapping resampling with 25 reps on the samples and produced an accuracy of 77.45%. The confusion matrix for the logistic regression model is shown in table 4.1.

3.2 Decision Tree

Another method employed to predict the overall condition of the house is a decision tree. It is a simple to implement and a majorly used model. The overfitting and size of the decision tree are controlled by the complex parameter(cp) which helps in reducing the issues of overfitting. The cross-validation (cv) method is used which involves resampling without replacement. The decision tree is implemented using cv with 10-fold cross-validation and produced an accuracy of 79%. The confusion matrix for the decision tree model is shown in table 4.2.

	Poor	Average	Good
Poor	0	0	0
Average	2	73	14
Good	0	5	6

Table 3.1: Confusion Matrix for Decision Tree

	Poor	Average	Good
Poor	0	0	0
Average	2	74	16
Good	1	4	5

Table 3.2: Confusion Matrix for Logistic Regression model

CHAPTER 4

Prediction of House Prices

In this section, the algorithms used for building the prediction models are discussed. The dataset is partitioned into two parts: 80% for training and 20 % for testing.

4.1 Algorithms used

The two popular machine learning models - Random Forest and Support Vector Machines(SVM) are implemented to predict house prices with the help of the selected variables.

4.1.1 Random Forest Model

In this approach, the algorithm creates several decision trees and averages their predictions to come up with a final prediction. The function set.seed() sets the initial number used to generate a sequence of random numbers to ensure to get same results every time when the function is called. The function randomForest() is used and trained on the training data on the SalePrice variable, stating that SalePrice is the dependent variable and all the other variables in the dataset are independent variables. The house prices for the test data is predicted using the trained model. The plot for the same is shown in figure 5.1. The performance metrics: R-Squared, Root Mean Square Error(RMSE) and Mean Absolute Error(MAE) is calculated and shown in Table 5.1. The R-Squared, RMSE and MAE for Random forest are 0.849709, 29357.97 and 19636.87 respectively.

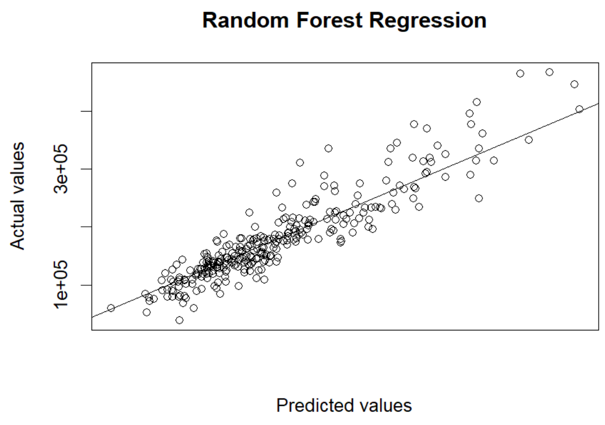

Figure 4.1: Plot for Random Forest Regression

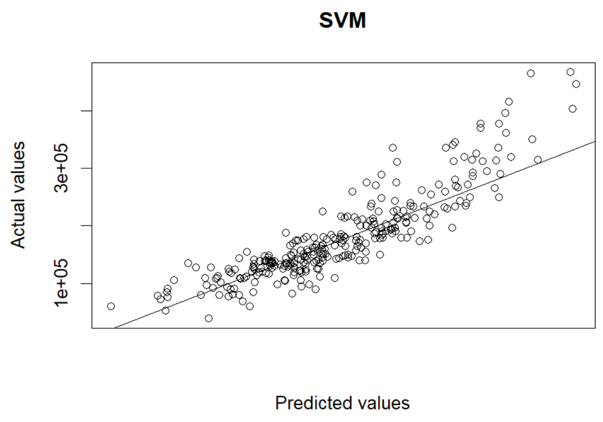

Figure 4.2: Plot for SVM

4.1.2 SVM

SVMs are particularly useful when the data has a non-linear relationship between the dependent and independent variables. In the case of house price prediction, there may be complex relationships between features such as location, year built, and size of the property, making SVMs an appropriate choice for modeling this data.

The SVM model uses a linear kernel to predict the sale price of houses based on various features in the training data. The code uses 10-fold cross-validation with 3 repeats for training the model and applies to center and scaling as pre-processing steps. It also performs hyperparameter tuning by testing 10 different combinations of hyperparameters using the tune length parameter. Once the model is trained, it is used to predict SalePrice values for the test data as depicted in figure 5.2. The performance metrics: R-Squared, Root Mean Square Error(RMSE), and Mean Absolute Error(MAE) are calculated and shown in Table

5.1. The R-Squared, RMSE, and MAE for Random forest are 0.74682, 38141.6, and 23636.08 respectively.

4.1.3 Output Interpretation

RMSE and RSquared give detailed insights with respect to the accuracy of the model to fit the dataset. The RMSE describes the ability of our model to predict the value of âSalePriceâ. RSquared shows how well our model explains the variation in the response variable. MAE represents the average of the absolute difference between the actual and predicted values in the dataset.

4.2 Re-Sampling Methods to estimate test error

4.2.1 Cross-Validation(CV)

Cross-validation (CV) is a technique used in ML to evaluate the performance of a predictive model on an independent data set. The errorest() function is used from ′ipred′ package to evaluate the performance of Random Forest and SVM model.
The function uses the "cv" estimator for the random forest to perform k-fold cross-validation with k=10, which is set using the control.errorest() function. It calculates various performance metrics for each fold of the cross-validation, and returns an object of class "errorest" which is the sampling test error. The output shows that the estimated RMSE is 34,869.22, which means that on average, the model's predictions are off by $34,869.22 in terms of the SalePrice variable. This indicates that the model's predictions are not highly accurate, and there is still room for improvement. For the SVM model, it is similar to the random forest model, but the model is specified using the "lm" method instead of "randomForest". This is because the "lm" method in the "train" function was used to fit the SVM model. The errorest() function calculates the error estimates for each fold of the cross-validation. The RMSE for SVM model came upto the value 40649.4.

4.2.2 Error estimation using BootStrap

Bootstrap is a resampling technique used to estimate the variability of a statistical estimate and to evaluate how robust a model is. It can be used to estimate the prediction error of a model by generating multiple random samples with replacement from the original data set and fitting the model on each bootstrap sample.
The random forest regression model uses the "boot" estimator to perform bootstrap resampling. The estimated root mean squared error is 36238.1, which means that, on average, the predicted values are off by around $36,238.1 from the actual values. This value can be used to compare the performance of different models or to evaluate the performance of the same model with different parameters or data.For the SVM model, the "lm" method instead of "randomForest". This is because the "lm" method in the "train" function was used to fit the SVM model. The errorest() function calculates the error estimates for each bootstrap sample. THE RMSE is 41143.88.

CHAPTER 5

Research Question

Can we cluster houses based on their qualities to discover similar groups of properties that can help real estate brokers or purchasers target specific sorts of properties more effectively? To understand and solve the problem, we'll employ the K-means clustering technique. The best number of groups (k) must first be determined using an Elbow plot. By reducing the within-cluster sum of squares, this plot will assist us in selecting the best k value. We must analyze the clusters we get. We achieve this by assigning each data point (house) to its corresponding cluster number, then using the summary function of the dplyr package As the k-means algorithm converges, the within-cluster sum of squares (WSS), which is calculated as the sum of the squared distances between each observation and its assigned centroid, can be used to assess the clustering's quality. The values are assigned to clusters and it is iterated until all the values are assigned for 'i' values with 'i' centers. The vector stores the calculated within-cluster sum of squares(WSS).
An Elbow plot is used to determine that helps to determine the optimal number of clusters to use in a clustering algorithm. it is commonly used in an unsupervised way in machine learning to identify patterns within data. The WCSS drops as the number of clusters rises because each cluster has fewer points and is more densely packed. it shows where adding more clusters does not significantly reduce the sum of squares within a cluster.
K-Means model is trained which is an unsupervised Machine Learning algorithm, In this Centre is the number of clusters we want to create that is optimal k, iter.max initializes the maximum number of iterations

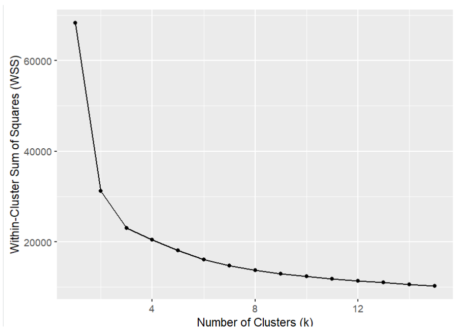

Figure 5.1: Elbow plot

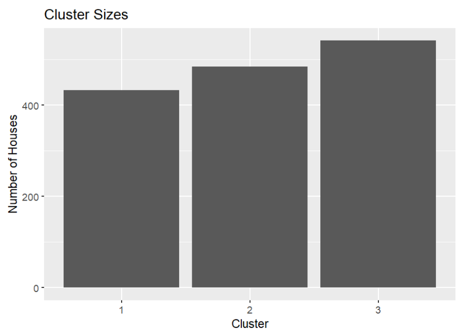

Figure 5.2: Clusters Kmeans

The provided code labels each data point with the cluster to which it belongs and uses the summary function of the dplyr package to calculate the average of each feature for each cluster in order to assess the clusters produced by the clustering algorithm, This produces a summary of the clusters, After getting the summary it shows the number of houses in each cluster and how are they
The Count(cluster) is used to count the number of observations in each cluster, And the bar chart is created with ggplot package The y-axis represents the number of observations in each cluster, while the x-axis represents the cluster column. This makes it possible to study data patterns and understand the characteristics of each cluster In general, cluster analysis can be a useful tool for uncovering insights and understanding relationships among variables.

CHAPTER 6

Appendix

6.1 R Code:

```
# Import libraries
library(randomForest)
library(xtable)
library(ggplot2)
library(dplyr)
library(mice)
library(pastecs)
library(caret)
library(car)
library(ggcorrplot)
library(MASS)
library(ipred)
library(tree)
library(Hmisc)
library(corrplot)
library(kernlab)
library(party)
library(tidyr)
library(tidyverse)
library(ggplot2)
library(cluster)

# Loading the csv file into a data frame
house_df = read.csv("C:\\Users\\Sharon\\OneDrive\\Documents\\Essex\\Spring Term\\Applied Statistics\\Assignment 2\\Group assignment 20222023-20230314\
house_data <- house_df
dim(house_data)

##################### Q1 #####################

# Numerical summary of data
house_data_summary<-summary(house_data)
View(house_data_summary)

house_data_num<-names(which(sapply(house_data,is.numeric)))
house_data_num  # 23 numerical columns

house_data_char<-names(which(sapply(house_data,is.character)))
```

```
house_data_char # 28 character columns

# Graphical Summary

# Understanding the nature of the response variable
ggplot(data=house_data, aes(SalePrice)) +
  geom_histogram(col="black", aes(fill=..count..)) +
  scale_fill_gradient("Count", low="grey", high="pink") +
  labs(title = "Histogram of SalePrice", x = "SalePrice", y = "Count of Houses")
summary(house_data$SalePrice)

IQR <- IQR(house_data$SalePrice,na.rm = TRUE)
StdDev <- sd(house_data$SalePrice,na.rm = TRUE)
Median = median(house_data$SalePrice,na.rm = TRUE)
Mean = mean(house_data$SalePrice,na.rm = TRUE)

#Density function and histogram of response Variable
density_plot = ggplot(house_data, aes(x=SalePrice)) +
  geom_histogram(aes(y=..density..), # Histogram with density instead of count on y-axis
                 binwidth=.5,
                 colour="black", fill="grey") +

  geom_density(alpha=.2, fill="#FFCC0011") + geom_vline(aes(xintercept=Median),
                                                        color="violet", size=1)
density_plot+geom_vline(aes(xintercept=Mean), color="red", linetype="dashed", size=1)

#For Skewness and Kurtosis - Nature of distribution
library(moments)
skewness(house_data$SalePrice,na.rm = TRUE)
kurtosis(house_data$SalePrice,na.rm = TRUE)

# Boxplots of few variables and to check for outliers

#1
par( mfrow= c(3,2) )
ggplot(data=house_data, aes(y= SalePrice, x=Street, fill=Street) ) +
  geom_boxplot() +
  ggtitle("Distribution of SalePrice V/s Street") +
  ylab("Sale Price") +
  xlab("Street")
#2
ggplot(data=house_data, aes(x= SalePrice, y=Utilities, fill=Utilities ) ) +
  geom_boxplot() +
  ggtitle("Distribution of SalePrice V/s Utilities") +
  ylab("Sale Price") +
  xlab("Utilities")
#3
ggplot(data=house_data, aes(y= SalePrice, x=PoolArea, fill=PoolArea) ) +
  geom_boxplot() +
  ggtitle("Distribution of SalePrice V/s PoolArea") +
  ylab("Sale Price") +
  xlab("PoolArea")
#4
ggplot(data=house_data, aes(y= SalePrice, x=MiscVal, fill=MiscVal) ) +
  geom_boxplot() +
  ggtitle("Distribution of SalePrice V/s MiscVal") +
  ylab("Sale Price") +
  xlab("MiscVal")
#5
ggplot(data=house_data, aes(y= SalePrice, x=RoofMatl, fill=RoofMatl) ) +
  geom_boxplot() +
  ggtitle("Distribution of SalePrice V/s RoofMatl") +
  ylab("Sale Price") +
  xlab("RoofMatl")
#6
ggplot(data=house_data, aes(y= SalePrice, x=HouseStyle, fill=HouseStyle) ) +
```

```r
  geom_boxplot() +
  ggtitle("Distribution of SalePrice V/s HouseStyle") +
  ylab("Sale Price") +
  xlab("HouseStyle")
#7
ggplot(data=house_data, aes(y= SalePrice, x=BsmtQual, fill=BsmtQual) ) +
  geom_boxplot() +
  ggtitle("Distribution of SalePrice V/s BsmtQual") +
  ylab("Sale Price") +
  xlab("BsmtQual")
#8
ggplot(data=house_data, aes(y= SalePrice, x=SaleCondition, fill=SaleCondition) ) +
  geom_boxplot() +
  ggtitle("Distribution of SalePrice V/s Street") +
  ylab("Sale Price") +
  xlab("SaleCondition")
#9
ggplot(data=house_data, aes(y= SalePrice, x=PavedDrive, fill=PavedDrive) ) +
  geom_boxplot() +
  ggtitle("Distribution of SalePrice V/s Street") +
  ylab("Sale Price") +
  xlab("PavedDrive")


# Replacing NA for character columns with meaningful values
house_data <- house_data %>% mutate_if(is.character, ~replace_na(.,"Not Available"))
colSums(is.na(house_data))

# Handling missing values for numerical columns with mice
house_data = mice(house_data, m = 10, seed=500) # cart method
house_data = complete(house_data)
house_data = na.omit(house_data)

colSums(is.na(house_data))
dim(house_data)

# Dropping ID column
house_data <- house_data[, !(colnames(house_data) %in% c("Id"))]

# Dropping columns having categorical outliers
house_data <- house_data[, !(colnames(house_data) %in% c("Street","Utilities","PoolArea","MiscVal","RoofMatl"))]
dim(house_data)

# Subsetting data based on catgorical-character, categorical-numerical and continuous
contin <- subset(house_data, select=
                       c("LotFrontage","LotArea","YearBuilt","MasVnrArea","TotalBsmtSF","X1stFlrSF",
                         "X2ndFlrSF","GrLivArea","GarageArea","SalePrice"))

c_numerical <-subset(house_data, select=
                                 c("OverallQual","OverallCond","LowQualFinSF","FullBath","BedroomAbvGr",
                                   "KitchenAbvGr","TotRmsAbvGrd","Fireplaces","MoSold", "YrSold"))

c_character <- house_data[unlist(lapply(house_data, is.character))]
# Correlation matrix
cormat <- round(cor(contin), 1) # compute the correlation matrix

# Correlation Matrix Plot
corrplot.mixed(cor(cormat),
               lower = FALSE,
               upper = "circle",
               tl.col = "black",addCoef.col = 1,
               number.cex = 0.5,tl.cex = 0.35)

# Variables with threshold greater than 0.6
var <- findCorrelation(cormat, cutoff=0.6)
varnames <- names(contin)[var]
```

```
print(varnames)

# Instantiating a linear model
m = lm(SalePrice ~ ., data = contin)
summary(m)
vif(m)

# Eliminating the variables with VIF > 10 and the highly correlated features
contin <- contin[, !(colnames(contin) %in%
                               c("X1stFlrSF","X2ndFlrSF","GrLivArea","TotalBsmtSF","LotFrontage","LotArea"))]

# Label Encoding - Categorical character columns
c_character <- c_character %>% mutate_if(is.character, as.factor)

# Function to perform label encoding
label_encode <- function(x) {
  levels <- unique(x)
  labels <- seq_along(levels)
  names(labels) <- levels
  return(labels[x])
}

# Apply label encoding to categorical variables
c_character <- c_character %>% mutate_if(is.factor, label_encode)
View(c_character)

# Concatenating numerical categorical features with character categorical features
cat <- cbind(c_character, c_numerical)
View(cat)

# Chi square test
round(mapply(function(x, y) chisq.test(x, y)$p.value, cat[, -35],
             MoreArgs=list(categorical[,1])),3)

# selecting columns which have p-values less than 0.05
cat <- subset(cat, select=
                        c("OverallQual","OverallCond","FullBath","Neighborhood","BedroomAbvGr",
                          "KitchenAbvGr","TotRmsAbvGrd","ExterCond","GarageType", "BsmtCond"))
# Combined dataset
f_data <- cbind(cat,contin)
View(f_data)



###################### Q2 ######################

#dividing OverallCond into poor,average, good based on their value
f_data$OverallCond = cut(f_data$OverallCond,
                          breaks = c(1, 4, 7, 11),
                          labels = c("Poor", "Average", "Good"),
                          right = FALSE)

ggplot(f_data) +
  geom_bar(aes(x = OverallCond), stat = "count") +
  geom_text(stat = "count", aes(x = OverallCond, y = ..count.., label = ..count..), vjust = -0.5)

# Replica of final dataset
m_data <- f_data

set.seed(100)
# Multinomial Logistic Regression
trctrl_logistic = trainControl(method = "boot", savePredictions=TRUE, search = "random")
multlogreg = train(OverallCond ~ ., data = m_data, method = "glmnet", trControl=trctrl_logistic,
                family="multinomial", tune_length = 0)
confusionMatrix(multlogreg)
#The confusion matrix contain percentages - 74.7 in the (2,2) position suggests that 74.7% of the samples were predicted as "Average" and correctly cl
```

```r
cm <- confusionMatrix(multlogreg)
cm$table <- round(cm$table, digits = 0)
print(cm)

# Decision Tree
set.seed(100)

d_tree = trainControl(method = "cv", number = 10, savePredictions=TRUE)
d_tree_fit = train(OverallCond ~., data = m_data, method = "rpart", trControl=d_tree, tuneLength =
                     0)
cm1 <- confusionMatrix(d_tree_fit)
cm1$table <- round(cm$table, digits = 0)
print(cm)


###################### Q3 ######################

# Dividing the data for training - 80% and testing - 20%
data_partition <- createDataPartition(f_data$SalePrice, p = .8,list = FALSE, times = 1)
train_data  <- f_data[data_partition, ]
test_data <- f_data[-data_partition, ]

# Random Forest Regressor for prediction of SalePrice
RFR <- randomForest(SalePrice ~ . ,data = train_data, importance = T)
# Observing the importance of each variable
imp <- importance(RFR)

# Predict using test data
RFR_Pred <-predict(RFR, newdata = test_data)
RFR_Pred

varImp <- data.frame(Variables = row.names(imp),
                            Importance = round(imp[ ,'%IncMSE'],2))
varImp

# Metrics
data.frame(
  R_Squared = R2(RFR_Pred, test_data$SalePrice),
  Root_Mean_Square_Error = RMSE(RFR_Pred , test_data$SalePrice),
  Mean_Absolute_Error = MAE(RFR_Pred , test_data$SalePrice)
)

# SVM
train_control <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
SVM <- train(SalePrice ~., data = train_data, method = "svmLinear",
             trControl=train_control,
             preProcess = c("center", "scale"),
             tuneLength = 10)

SVMPred <- predict(SVM, newdata = test_data)
SVMPred

# Metrics
data.frame(
  R_Squared = R2(SVMPred, test_data$SalePrice),
  Root_Mean_Square_Error = RMSE(SVMPred, test_data$SalePrice),
  Mean_Absolute_Error = MAE(SVMPred, test_data$SalePrice)
)

plot(SVMPred, test_data$SalePrice,
     xlab="Predicted values",ylab="Actual values", xaxt="n", main = "SVM")
abline(a=0,b=1)

plot(RFR_Pred,test_data$SalePrice,
     xlab="Predicted values",ylab="Actual values", xaxt="n", main = "Random Forest Regression")
abline(a=0,b=1)
```

```r
# Sampling methods:

# Cross Validation(CV):

# Random forest Regression
rferror <- errorest(SalePrice ~ ., data=f_data, model=randomForest,estimator= c("cv"),est.para=control.errorest(k=10), predict= RFR_Pred)
rferror

# SVM
svmerror <- errorest(SalePrice~ .,data=f_data,model=lm,estimator= c("cv"),est.para=control.errorest(k=10), predict= SVMPred)
svmerror

# Error estimation using BootStrap:

# Random forest
rferrorboot <- errorest(SalePrice ~ ., data=f_data,model=randomForest,estimator= "boot", predict= RFR_Pred)
rferrorboot

# SVM
svmerrorboot <- errorest(SalePrice~ .,data=f_data,model=lm,estimator= "boot", predict= SVMPred)
svmerrorboot


##################### Q4 #####################

## Normalize the dataset
contin_1 <- apply(contin, 2, function(x) scale(x, center = FALSE, scale = max(x) - min(x)))
f_data1 <- cbind(cat,contin_1)

# K-Means Clustering

## Determine the optimal number of clusters (k)
wss <- c()
for (i in 1:15) {
  kmeans_model <- kmeans(f_data1, centers = i, nstart = 25, iter.max = 50)
  wss[i] <- kmeans_model$tot.withinss
}

## Plot the Elbow Method
elbow_plot <- data.frame(k = 1:15, wss = wss)
ggplot(elbow_plot, aes(x = k, y = wss)) + geom_point() + geom_line() + xlab("Number of Clusters (k)") + ylab("Within-Cluster Sum of Squares (WSS)")

## Train the K-Means model
optimal_k <- 3 # Based on the elbow plot, choose the optimal value for k
kmeans_model <- kmeans(f_data1, centers = optimal_k, nstart = 25, iter.max = 50)

# Analyze the clusters
f_data1$cluster <- as.factor(kmeans_model$cluster)
cluster_summary <- f_data1 %>% group_by(cluster) %>% summarise_all(list(mean))
print(cluster_summary)

cluster_sizes <- f_data1 %>% count(cluster)
ggplot(cluster_sizes, aes(x = cluster, y = n)) + geom_bar(stat = "identity") + xlab("Cluster") + ylab("Number of Houses") + ggtitle("Cluster Sizes")
```

Bibliography

[1] www.sciencedirect.com. (n.d.). Collinearity - an overview | ScienceDirect Topics. [online] Available at: Collinearity

[2] Frost, J. (2020). Variance Inflation Factors (VIFs). [online] Statistics By Jim. Available at: Variable Inflation Factor

[3] GeeksforGeeks. (2018). ML | Label Encoding of datasets in Python. [online] Available at: Label Encoding

[4] Turney, S. (2022). Chi-Square Tests | Types, Formula & Examples. [online] Scribbr. Available at: Chi-sqaured test

[5] www.ibm.com. (n.d.). What is Logistic regression? | IBM. [online] Available at: Logistic Regression