**UNIVERSITY OF ESSEX**

**DEPARTMENT OF MATHEMATICAL SCIENCE**

# MA321 – APPLIED STATISTICS

## GROUP PROJECT REPORT

**DHEERAJ LAKKAKULA – DL22170 - 2202392**

## CONTRIBUTION LIST

**QUESTION 1: DHEERAJ LAKKAKULA**

**QUESTION 2: SANJAY RAMASAMY**

**QUESTION 3: LINGESWARAN KUMAR**

**QUESTION 4: DHEERAJ LAKKAKULA**

**REPORT : EQUAL CONTRIBUTION**

**Abstract**

The analysis of the house data set provides valuable insights into the real estate market, allowing the individuals to make good decisions about buying or selling properties. This Data analysis makes it possible to identify trends and patterns within the data. This report includes descriptive statistics, visual summaries and analysis of the attributes of house data.

**Table of Contents**

Introduction ....3
Data Exploration....3
Handling Missing values ....4
Checking for Outliers ....6
Correlation Analysis of house data....7
Logistic Regression ....9
Naive Bayes Classifier ....10
Linear Regression....10
Random Forest ....11
Re-sampling Methods ....11
Further Analysis on the house data....11
Summary ....12
References ....12

**Word count: 2922**

**Introduction:**

As Data Analysts at an estate agency, we have performed the data analysis on the given house data set using R Programming and Machine Learning algorithms. By using this we can predict house prices in future and can observe how the current housing market is doing. Before performing some machine learning algorithms, it is mandatory to perform EDA, Data Visualization and cleaning processes. The given "house_data.csv" data set has a total of 1460 observations and 51 variables. After Exploring the data using EDA there can be some missing values and outliers in the given data are handled with a suitable imputation method based on the missing values in the data can lead to degrading the accuracy.

**Data Exploration:**

This is the first step in every data analysis that involves loading the dataset and viewing the dimensions and structure of the data. As R is a popular open-source programming language with powerful built-in libraries. Some of the data explorations we carried out are like getting the dimensions, structure, and an overview count of numeric and character variables of the data. We summarise the numeric and character variables and store them in a separate data frame. Finding missing values and visualizing the data using a boxplot for checking outliers and some important variables like a sale price. This gives a clear and comprehensive overview of the dataset. This also includes Data visualization where we can get more insights into the data and the relation of the attributes shown in **Figure 1**.

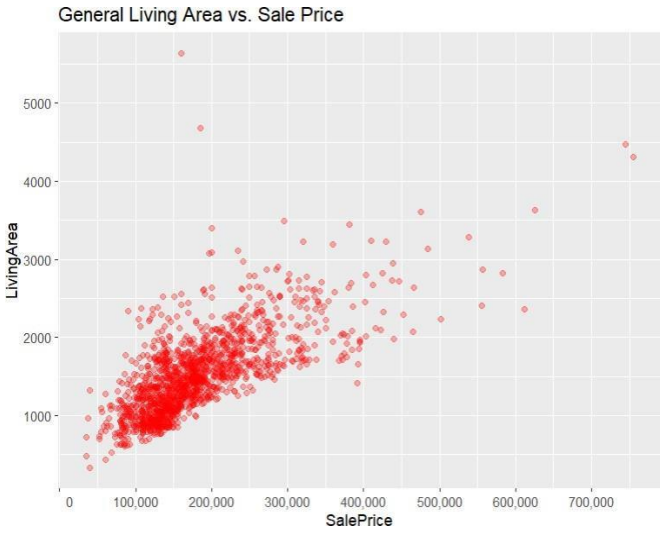



**Figure 1: Scatter plot (Living Area Vs SalePrice)**

The above figure shows how the sale price and Living area of the house are related to each other. By this, we can say that an increase in Living Area also increases the sale price, but some variables do not follow this trend and we consider them as outliers. In the house-dataset, we have 23 numerical and 28-character observations, in which character observations are converted into categorical observations for memory efficiency and to give better performance with better visualization helping us to improve the accuracy of the model. The presence of missing values in the observation will affect the accuracy of the machine-learning model and leads to accurate predictions. So, these missing data are handled by imputing them based on the context of the data. The following **Table 1** shows the percentage of missing values in the dataset.

| Variable | Missing Percentage |
|---|---|
| PoolQC | 0.995205479 |
| MiscFeature | 0.963013699 |
| Alley | 0.937671233 |
| Fence | 0.807534247 |
| LotFrontage | 0.17739726 |
| GarageType | 0.055479452 |
| GarageCond | 0.055479452 |
| BsmtQual | 0.025342466 |
| BsmtCond | 0.025342466 |
| MasVnrArea | 0.005479452 |

**Table 1: Percentage of missing values**

In the above table, the columns in red colour represent that they have missing values above 80%. So, these missing values are handled based on context of the data to get a good statistical summary and the accuracy of the model we implemented on the data. There are several methods to impute these missing values.

**Handling Missing values:**

Missing data can lead to inaccuracy of prediction. In order to get good results the data has to be handled efficiently using imputation methods such as Mean, Median, Mode or assigning a new group value for missing categorical variables. Data cleaning is an important step in any data analysis or machine learning process. Based on the given dataset there are 5910 missing values. we have carried out some steps to handle these missing value as follows.

Dropping those columns which has missing values above 80% will not affect the accuracy. The imputation method is applied to the columns that are below 80%.In the house_dataset, there are four columns PoolQC, Alley, MiscFeature and Fence which exceed 80% of missing values as shown in **Figure 2**. So dropping those columns will be a good option.

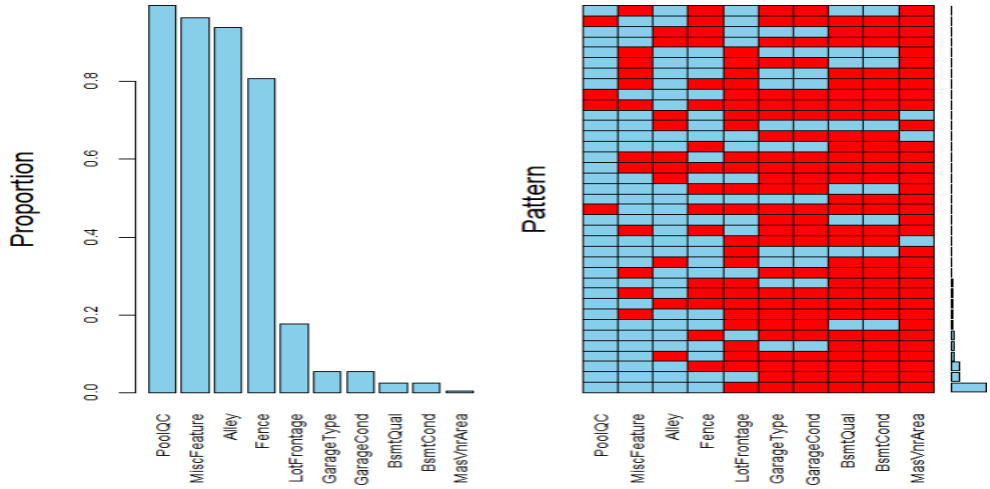



**Figure 2: Percentage of Missing values**

The above figure shows the distribution of missing values. The missing values in LotFrontage, GarageType, GarageCond, BsmtQual, BsmtCond and MasVnrArea are below 80% .We can use imputation methods to handle a numerical variable, like mean and median or assign a new group it depends on the context of the data. In this, we imputed the LotFrontage using the Median value by observing it as in Figure 3. We can see that is a continuous and skewed distribution towards the right, this indicates it has outliers so the median will be the best to deal with outliers. By using **median()**, we have imputed the value for the LotFrontage. For MasVarArea we have imputedit with mean value as the data is slightly skewed and it is not exactly as symmetric as it startsfrom a point of the Y-axis and smoothly down towards the X-axis as shown in **Figure 3**. So, in this case, we use either median or mean to impute the values.

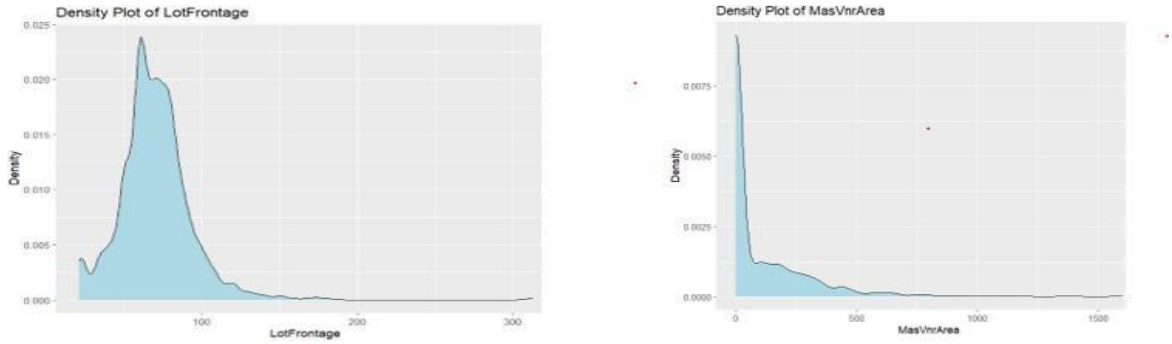



**Figure 3: showing the density plots for LotFrontage and MasVnrArea**

As mean is the more informative measure of central tendency, we use mean to impute the MasVnrArea. For the categorical variables like GarageType, GarageCond, BsmtCond, and BsmtType, we imputed them as No garage and No Basement respectively.

**Checking for Outliers:**

These observations are significantly different from other observations in the dataset. It is important to check for any outliers in the dataset due to the high impact on the results. We can detect them by using IQR and data visualizations. The data points that fall outside the IQR values are considered outliers. In the house_dataset, we have used a boxplot to detect the outliers and remove them using IQR. We have found an outlier in the SalePrice by using a boxplot which can be obtained by using **ggplot2** library as shown in **Figure 4.**

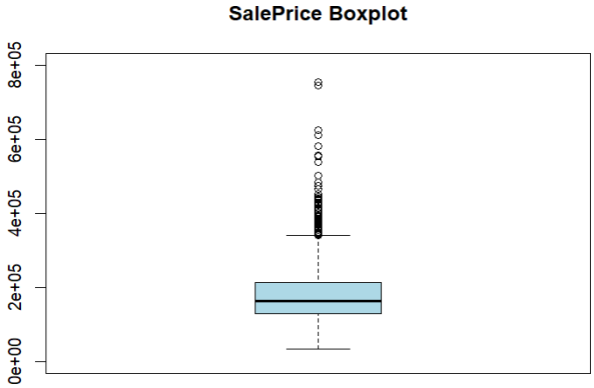



**Figure 4: Boxplot of SalePrice**

This plot shows the line inside the box is the median of the data. The whiskers extend from the box to show the range of the data. By default, the whiskers extend to 1.5 times the interquartile range (IQR), which is the distance between Q1 and Q3. Any data points that lie outside the whiskers are considered outliers. These outliers may indicate extreme values or errors in the data.By getting the summary of the sale price using Summary(), we can get to know the descriptive statistics of the SalePrice such as median, mean, 1st quartile etc. as shown in **Table 2**.

| Min | 1st Quartile | Median | Mean | 3rd Quartile | Max |
|---|---|---|---|---|---|
| 34900 | 129975 | 163000 | 180921 | 214000 | 755000 |

**Table 2: Descriptive statistics of SalePrice**

Now by using SalesPrice, we can find IQR to remove the outliers by below calculations:

IQR_SalePrice = 214000 - 129975

Lower = 214000 -1.5*IQR_SalePrice

Upper = 129975 + 1.5*IQR_SalePrice

By filtering the data using sale price by **SalePrice >Lower & SalePrice < Upper** we can remove the outliers. To visualize this, we can use a boxplot as shown in **Figure 5** there are no outliers in the SalePrice of the house.

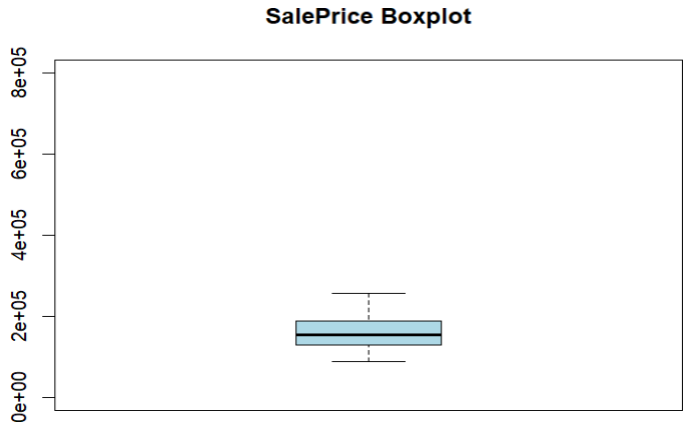



**Figure 5: Boxplot of SalePrice (without outliers)**

The Identified outliers in the dataset are not removed because of the smaller amount of data (1460*51) and removing may lead to loss of information. Instead, we removed only the missing values that are above 80%.

Now the dimensions of data are changed i.e., (1460*47) as we removed four columns that havemissing values greater than 80%.

**Correlation Analysis of house data:**

The main goal of correlation analysis is to identify any relationships between different variables in the dataset. We can calculate the correlation coefficients of each variable that lies between 0 and 1 as shown in Figure 6. We plot the correlation plot using corrplot() library. This correlation analysis are done on numerical variables in the house-dataset. In this correlation plot, we have removed column 'Id' as it doesn't give any insights into the data. We have also taken the variables where the correlation coefficients lie between 0.5 to -0.5.

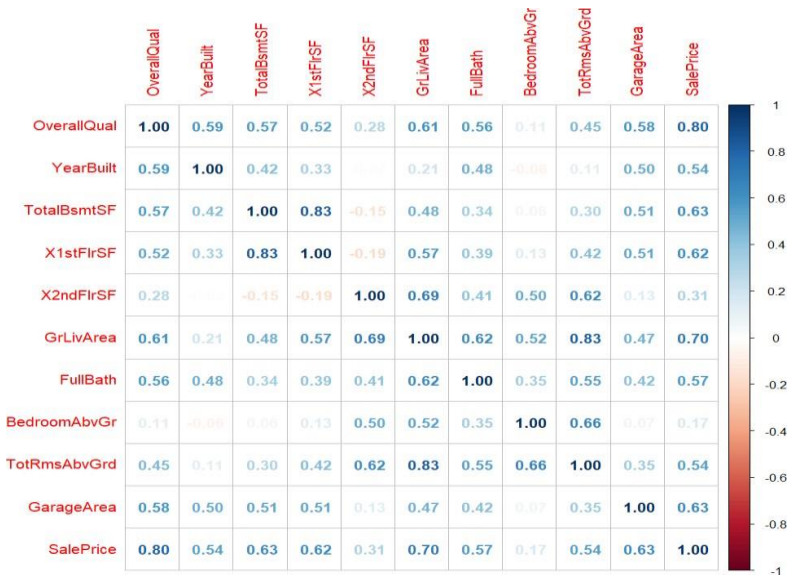



**Figure 6: Correlation analysis of house dataset**

From this correlation plot, we can say that SalePrice and OverallQual have the highest 0.80 correlation coefficient which means these two variables are highly related to each other also we can see that GrLivArea (Living Area) and the sale price have a correlation coefficient of 0.70 which is high. Similarly, we can find which variables are highly correlated to each other. In case of Positive correlation, if one variable is increasing and the other also will follow the same trend. For negative correlation, it will be vice-versa. By this plot, we can easily visualize how two variables are correlated to each other. There is a statistical term called multicollinearity means a high degree of correlation between two or more predictor variables which will be a problem in finding the results. Overall, after performing Data exploration, Data cleaning and Data visualization techniques on the house_dataset, we can finalize the variables present in the dataset and their relation to each other. Now the dataset was relativelyclean with few missing values that are also imputed with the appropriate methods like Mean and median as discussed in the data cleaning section. Now our data is ready to do further analysis like predicting some target variables based on some dependent variables. We can also perform the techniques like Regression and Classification to get more insights into the data. In this house_dataset we are using classification techniques like Logistic regression and Naïve Bayes for predicting the overall Condition of the house and using Regression algorithms such as Linear regression and Random Forest to predict the sale price of the housebased on some features.

Before performing these, we divided the houses into three categories namely "Poor", "Average" and "Good" which are based on the Overall Condition of the housewhich defines the condition in form of a rating i.e., a numeric variable. Before applying any of the machine-learning models convert the newly created column and character to a factor variable, and split the cleaned dataset into train and test parts. Mostly 80% for training and 20% for testing the model by using the Caret library.

**Logistic Regression:**

In the house_dataset we have created a new column based on the overall condition of the house and named it New_cond with 3 values poor, good, and average. Logistic regression is used when the dependent variable has only two factors. We used Multinomial Logistic regression to predict the New_cond of the house. This can be used for classification problems to make predictions on new data by estimating the probabilities of each category based on the predictor variables. It can be often used in the field of marketing where there are multiple categories of interest. We need to train the data on this model by specifying the dependent variable i.e., New_cond and measuring the performance metrics like accuracy, F1-score, and AUC_ROC as shown in Table 3 using test data we got an accuracy of 0.959.

| Category | Accuracy | F1_score | AUC_ROC |
|---|---|---|---|
| Average | 0.959 | 0.973 | 0.974 |
| Good | 0.959 | 0.959 | 1 |
| Poor | 0.959 | 0.364 | 0.868 |

**Table 3: performance metrics of the logistic regression**

Based on these metrics for each category, we can say that the model is good at predicting the different categories of the New_Cond. The F1 score considers both the precision and recall of the model. A higher F1 score indicates a better balance between Precision and Recall. AUC-ROC is a metric that checks how well the model can differentiate between the classes. A big score of it indicates the model has a better capacity to distinguish the classes. So, this model is good at accuracy and able to differentiate between Poor, Average, and Good classes.

**Naive Bayes Classifier:**

This is used for classification tasks though it's simple and computationally. It works by Bayes' theorem to calculate the probability of each class. This algorithm is well suited for predicting the overall condition of houses because this problem involves data with both categorical and continuous variables, such as the age of the house, number of bedrooms. This algorithm can handle mixed data, and provide accurate predictions. The model can get an accuracy of 0.835. Apart from this we also found some metrics as shown in **Table 4**.

| **Metric** | Accuracy | Precision | Recall | F1 Score | AUC_ROC |
|---|---|---|---|---|---|
| **Value** | 0.8350515 | 0.8053097 | 0.9784946 | 0.8834951 | 0.9287796 |

**Table 4: Performance of Naive Bayes**

By these metrics, we can compare both the model's performance. By considering the accuracy of the models we can say that Logistic regression outperforms Naive bayes.

**Linear Regression:**

This statistical model is used to find the relationship between the dependent variables and independent variables. It uses the '**lm()**' function to fit the model. In this analysis, we predicted the SalePrice of the house using a linear regression algorithm. Before applying the model, we need to split the data into train and test data and preprocess the data. We also drop the factor columns with lesser levels, they are not necessary to produce better results. In this we are predicting the SalePriceof the. Differences between the actual and predicted SalePrice can visualize using ggplot2 in R as shown in **Figure 7.** we can observe that most of the occurrences of dark blue indicate the actual and predicted sale prices are the same and some occurrences in red and light grey are different in the predicted SalePrice of the house.

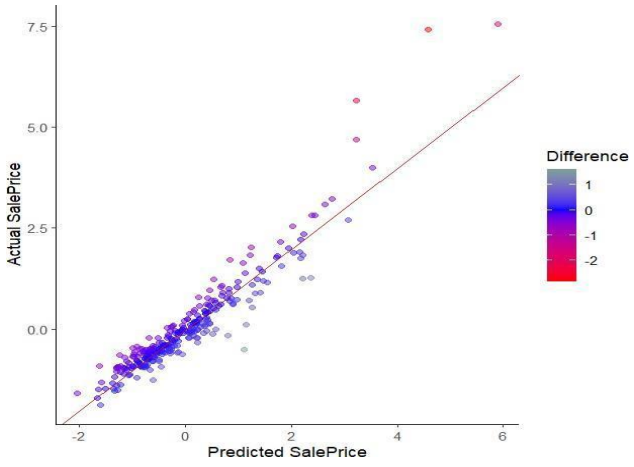



**Figure 7: Difference between the actual and predicted SalePrice**

**Random Forest:**

This model is used for regression tasks that involve building decision trees, by creating many decision trees each tree trained on randomly selected subsets of training data and predictor variables to get better accuracy. The house dataset has both continuous and categorical data, the random forest algorithm works well in mixed types of data sets. Once the model is trained by this algorithm, we can predict the SalePrice, to know the accuracy of the model we find some metrics like R-Squared value and Root Mean Square Error (RMSE). Among these, the R-Squared (R2) value shows much variance in SalePrice. The higher R2 value indicates that the model fits the data well. By using these metrics, we can evaluate the performance of the above two models as shown in **Table 5**.

| | RMSE | R2 |
|---|---|---|
| **Linear Regression** | 0.3947491 | 0.896292 |
| **Random Forest** | 0.13947 | 0.832769 |

**Table 5: Performance metrics of both models**

By viewing these performance metrics, the R2 value is higher for the Linear regression. So, in predicting the SalePrice of the house linear regression outperforms random forest.

**Re-sampling Methods:**

These methods involve statistical techniques that estimate the performance of the model by randomly sampling subsets of the data. We have used cross-validation to estimate the misclassification error of the random forest model by repeatedly splitting the data into training and testing sets to evaluate the performance of the model on new data. In order to apply this method we set the estimator argument to **"cv"** by default which uses 10-fold cross-validation. The RMSE value obtained from this estimation is 29053.34, this shows the average sale price will differ by 29053.34 compared to the actual sale price. Similarly, we implemented the same using the bootstrapping technique to get a misclassification error and got an RMSE value of 29890.71. So by using these resampling methods, we have validated the model to ensure that the model can work on new or unseen data.

**Further Analysis on the house data:**

In further analysis, we thought of how the living area of house will vary based on we find a relationship between the Living area and SalePrice of the house. We predicted the living area's size with the house's sale price. As discussed in thecorrelation analysis, there is a strong correlation between the living area, the SalePrice, Lot Area, Total bathrooms, Total Bedrooms, Kitchen and Total Rooms.

So, it's good to use all these as independent variables to predict the living area of the house using linear regression.The R2 value for this model is 0.8611 which shows that this model is a good fit for this data.In this whole process correlation analysis helped a lot in determining the independent variables for predicting the living area of the house.

**Summary:**

By using this house dataset, we analyzed the information for predicting the sale price and condition of houses based on their features, such as location, living area, and a number of bedrooms using machine-learning techniques such as linear regression and random forest to make accurate predictions. This analyzed house data has practical applications in real-life scenarios such as buying or selling a house, property assessment, and real estate investment. Making informed decisions based on accurate predictions can help individuals avoid financial risks and make profitable investments.

**References :**

Chicco, D., Warrens, M.J. and Jurman, G., 2021. The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation. PeerJ Computer Science, 7, p.e623.

Shrestha, N., 2020. Detecting multicollinearity in regression analysis. American Journal of Applied Mathematics and Statistics, 8(2), pp.39-42.

Wang, Yijia, and Qiaotong Zhao. "House Price Prediction Based on Machine Learning: A Case of King County." 2022 7th International Conference on Financial Innovation and Economic Development (ICFIED 2022). Atlantis Press, 2022.

MA321 – Applied Statistics Lab notes by Dr. Osama Mahmoud

https://www.analyticsvidhya.com/blog/2020/12/predicting-using-linear-regression-in-r/

https://www.r-bloggers.com/2021/04/random-forest-in-r/

https://stats.oarc.ucla.edu/r/dae/multinomial-logistic-regression/

# Appendix:

**## R-Script**

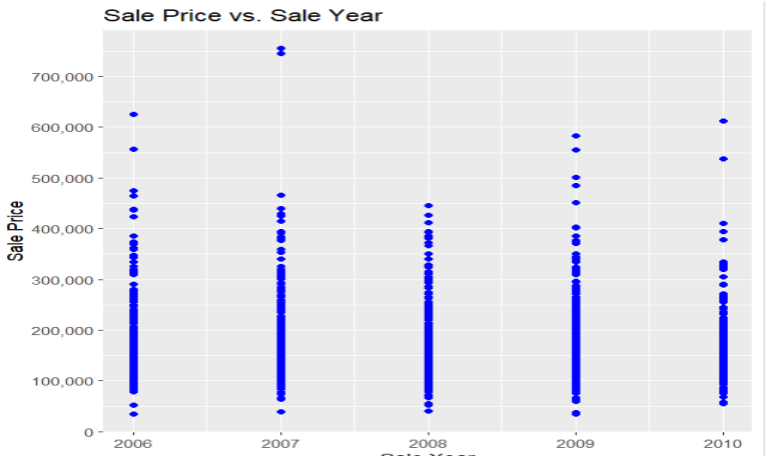
```
#install required packages

install.packages("dplyr")
install.packages("corrplot")
install.packages("VIM")
install.packages("MICE")
install.packages("psych")
install.packages("ggstatsplot")
install.packages("caret")
install.packages("randomForest")
install.packages("e1071")
install.packages("knitr")
install.packages("ipred")

#Attaching the required libraries

library(dplyr)
library(VIM)
library(mice)
library(psych)
library(ggstatsplot)
library(caret)
library(nnet)
library(randomForest)
library(e1071)
library(corrplot)
library(knitr)
library(ipred)
#Loading house_data set

house_data <- read.csv(file.choose())

#Exploring the house_data

#View its dimensions
dim(house_data)

#Getting column names
names(house_data)

#display the structure of the whole data set
str(house_data)

#getting head and tail of the data
head(house_data,5)
tail(house_data,5)
```

```
#Checking for Missing values and percentage of missing values

colSums(is.na(house_data))
print("Percentage of Missing Values")
colMeans(is.na(house_data))*100

missing_cols<-names(which(colSums(is.na(house_data)) > 0))
missing_cols

#To visualize how the missing values are distributed
missing_plot <- aggr(house_data[missing_cols], col=c('red','skyblue'),
                     numbers=TRUE, sortVars=TRUE,
                     labels=names(house_data[missing_cols]), cex.axis=.7,
                     gap=3, ylab=c("Proportion","Pattern"))

#Removing the columns with missing values equal or greater than 80%
miss_high <- names(which(colMeans(is.na(house_data))>0.8))
miss_high

house_data_new<-house_data[, which(colMeans(!is.na(house_data)) >=0.8)]
colMeans(is.na(house_data_new))
names(house_data_new)

temp_house_data<- house_data_new


#Data Analysis and summaries

#numerical data

data_num2 <- select_if(temp_house_data, is.numeric)
dim(data_num2)
names(data_num2)

summary(data_num2)

#correlations

correlations <- cor(na.omit(data_num2[,-1]))

#correlation plot using numerical varaibles
row <- apply(correlations, 1, function(x) sum(x > 0.5 | x < -0.5) > 1)
correlations<- correlations[row ,row ]
corrplot(correlations, method="number")

#converting all character to factor varaible

for (col in names(temp_house_data)) {
  if (is.character(temp_house_data[[col]])) {
    temp_house_data[[col]] <- as.factor(temp_house_data[[col]])
  }
}
```

```
#selecting only factor (categorical) varaibles
cat <- select_if(temp_house_data, is.factor)
summary(cat)

str(temp_house_data)
#plotting the histogram for every categorical varaibles
for (var in names(cat)) {
  plot<-ggplot(temp_house_data, aes(x=get(var))) +
    geom_bar() +
    labs(title=paste("Histogram of", var), x=var, y="Frequency")
  print(plot)
}

#Exploring Sale price column by its histogram
plot<-ggplot(temp_house_data, aes(x=SalePrice)) +
geom_histogram(fill="blue",binwidth=10000) +
  labs(title=paste("Histogram of SalePrice"), x="SalePrice", y="Frequency")+
scale_x_continuous(breaks = seq(0, 800000, by = 100000), labels = function(x) format(x,
big.mark = ",", scientific = FALSE))
print(plot)

#using kable function to get the descriptive statistics of sale price in table format
kable(as.array(summary(temp_house_data$SalePrice)),
      caption = 'Summary of Sales Price')
```

```
> #using kable function to get the descriptive statistics of sale price in table format
> kable(as.array(summary(temp_house_data$SalePrice)),
+       caption = 'Summary of Sales Price')


Table: Summary of Sales Price

|Var1    |     Freq|
|:-------|--------:|
|Min.    |  34900.0|
|1st Qu. | 129975.0|
|Median  | 163000.0|
|Mean    | 180921.2|
|3rd Qu. | 214000.0|
|Max.    | 755000.0|
>
```

```
# Create a scatter plot (year_sold vs saleprice)
plot <- ggplot(temp_house_data, aes(x = YrSold, y = SalePrice)) +
  geom_point(col = "blue") +
  labs(title = "Sale Price vs. Sale Year", x = "Sale Year", y = "Sale Price") +
scale_y_continuous(breaks = seq(0, 800000, by = 100000), labels = function(x) format(x,
big.mark = ",", scientific = FALSE))
print(plot)
```




```
#Scatter plot for living area vs sale_price
plot<-ggplot(temp_house_data, aes(x=SalePrice, y=GrLivArea)) +
  geom_point(colour="red", alpha=0.3)+
  labs(title = "General Living Area vs. Sale Price", x="SalePrice", y="LivingArea")+
scale_x_continuous(breaks = seq(0, 800000, by = 100000), labels = function(x) format(x,
big.mark = ",", scientific = FALSE))
print(plot)

#boxplot for visualazing outliers:
boxplot(house_data$SalePrice, main = "SalePrice Boxplot",
        boxwex = 0.5, # sets the width of the box
        whisklty = 2, # sets the line type of the whiskers
        col = "lightblue", # sets the color of the box and whiskers
        ylim = c(0, 800000)
)

# Getting the summary of saleprice column to know quartile values
summary(temp_house_data$SalePrice)

#Finding Inter Quartile Range(IQR) to remove outliers

IQR_SalePrice =  214000 - 129975

lower= 214000 -1.5*IQR_SalePrice
upper = 129975 + 1.5*IQR_SalePrice



#Removing outliers
temp_new= subset(temp_house_data, SalePrice >lower & SalePrice < upper)

boxplot(temp_new$SalePrice, main = "SalePrice Boxplot",
        boxwex = 0.5, # sets the width of the box
        whisklty = 2, # sets the line type of the whiskers
        col = "lightblue", # sets the color of the box and whiskers
        ylim = c(0, 800000)
)
```

```
#DATA CLEANING

###Dealing with Missing values

#imputing the LotFrontage with the MEDIAN value

house_data_new$LotFrontage[which(is.na(house_data_new$LotFrontage))] <-
median(house_data_new$LotFrontage,na.rm = T)


#Imputing the MasVnrArea with MEAN value

house_data_new$MasVnrArea[which(is.na(house_data_new$MasVnrArea))] <-
mean(house_data_new$MasVnrArea,na.rm=T)

#imputing the character variables as given in data description and converting them as factor
variables
house_data_new$BsmtQual[which(is.na(house_data_new$BsmtQual))] <- "No Basement"
house_data_new$BsmtQual <- as.factor(house_data_new$BsmtQual)

house_data_new$BsmtCond [which(is.na(house_data_new$BsmtCond ))] <- "No Basement"
house_data_new$BsmtCond  <- as.factor(house_data_new$BsmtCond)


house_data_new$GarageType [which(is.na(house_data_new$GarageType))] <- "No Garage"
house_data_new$GarageType  <- as.factor(house_data_new$GarageType)


house_data_new$GarageCond [which(is.na(house_data_new$GarageCond))] <- "No Garage"
house_data_new$GarageCond <- as.factor(house_data_new$GarageCond)

#checking the structure of the dataset
str(house_data_new)


#converting all Character variable to Factor variable in the dataframe

cleaned_data <- house_data_new %>%
  mutate_if(sapply(house_data_new, is.character), as.factor)

str(cleaned_data)

#checking for missing values after imputation of all missing values
missing_cols<-names(which(colSums(is.na(cleaned_data)) > 0))
missing_cols

print("Percentage of Missing Values")
colMeans(is.na(cleaned_data))*100

#assigning the cleaned_data(without any missing values) to another dataframe.
new_cleaned_data2 <- cleaned_data
```

##########################Question 2#####################################

```
#Dividing the houses based on the Overall condition of the house into Poor,Good, Average
#based on the conditions
cleaned_house_data <- new_cleaned_data2 %>%
  mutate(New_cond= case_when(
    new_cleaned_data2$OverallCond>=1 & new_cleaned_data2$OverallCond<=3 ~ "Poor",
    new_cleaned_data2$OverallCond>=4 & new_cleaned_data2$OverallCond<=6~ "Average",
    new_cleaned_data2$OverallCond>=7 & new_cleaned_data2$OverallCond<=10 ~ "Good"))
cleaned_house_data

str(cleaned_house_data)

#getting the newly created column New_cond which has created based on the overallcondition
of house as a factor variable
cleaned_house_data$New_cond <- as.factor(cleaned_house_data$New_cond)


#Removing the Id column
cleaned_house_data <- cleaned_house_data[,-1]
#Splitting the dataset
# Set the random seed for reproducibility
set.seed(123)

# Split the data into training and testing sets into 80 to 20 set using caret package
trainIndex <- createDataPartition(cleaned_house_data$New_cond, p = 0.8, list = FALSE)
train <- cleaned_house_data[trainIndex, ]
test <- cleaned_house_data[-trainIndex, ]

# Fit a multinomial logistic regression model to the training data
model <- multinom(New_cond ~ ., data = train, family = binomial)
summary(model)

mlr_pred <- predict(model, newdata = test)
head(mlr_pred)
head(train$New_cond)
mean(mlr_pred == test$New_cond)
```

```
> mlr_pred <- predict(model, newdata = test)
> head(mlr_pred)
[1] Average Average Average Average Average Average
Levels: Average Good Poor
> head(train$New_cond)
[1] Good    Average Average Average Average Average
Levels: Average Good Poor
> mean(mlr_pred == test$New_cond)
[1] 0.9587629
>
```

```
# Create a confusion matrix
conf_matrix <- confusionMatrix(mlr_pred, test$New_cond,mode="everything")

# Predict the class probabilities for the test data
mlr_prob <- predict(model, newdata = test, type = "probs")
```

```
# Calculate the accuracy
acc <- conf_matrix$overall["Accuracy"]
library(pROC)

# Calculate the F1 score
f1 <- conf_matrix$byClass,"F1"]

# Calculate the AUC-ROC for each category
auc <- sapply(levels(test$New_cond), function(x) {
  roc(test$New_cond == x, mlr_prob[, x])$auc
})

# Combine the metrics into a table
metrics <- data.frame(Category = levels(test$New_cond),
                      Accuracy = round(acc, 3),
                      F1_score = round(f1, 3),
                      AUC_ROC = round(auc, 3))
print(metrics)
```

```
> print(metrics)
  Category Accuracy F1_score AUC_ROC
1  Average    0.959    0.973   0.974
2     Good    0.959    0.959   1.000
3     Poor    0.959    0.364   0.868
>
```

```
library(pROC)
install.packages("pROC")
###Naive bayes

NB_model = naiveBayes(New_cond ~ ., data = train)
NB_model     # The Y values are the means and standard deviations of the predictors within
each class.
NB_model.predicted = predict(NB_model,type="class", newdata = test)
NB_prob =predict(model, newdata = test, type = "probs")

# confusion matrix
conf_matrix<-table(NB_model.predicted, test$New_cond)
mean(NB_model.predicted== test$New_cond)

accuracy <- (conf_matrix[1,1] + conf_matrix[2,2] + conf_matrix[3,3]) / sum(conf_matrix)
precision <- conf_matrix[1,1] / (conf_matrix[1,1] + conf_matrix[2,1] + conf_matrix[3,1])
recall <- conf_matrix[1,1] / (conf_matrix[1,1] + conf_matrix[1,2] + conf_matrix[1,3])
f1_score <- 2 * (precision * recall) / (precision + recall)
auc_roc <-multiclass.roc(test$New_cond, NB_prob)


# Print results in a table
result_table <- data.frame(metric = c("Accuracy", "Precision", "Recall", "F1
Score","AUC_ROC"),
                           value = c(accuracy, precision, recall, f1_score,auc_roc$auc))

result_table
```

```
> result_table
     metric     value
1  Accuracy 0.8350515
2 Precision 0.8053097
3    Recall 0.9784946
4  F1 Score 0.8834951
5   AUC_ROC 0.9287796
>
```

##########Question 3a)##########

#Predicting the SalePrice of house using Linear Regression

#filtering some of the data to fit linear regression model

# removing unused factor levels from the column/ removing the unused column

cleaned_house_data<- new_cleaned_data2

#removing the columns which are very less dependent on the target variable
cleaned_house_data = subset(cleaned_house_data, select = -c(Utilities))

cleaned_house_data = subset(cleaned_house_data, select = -c(Condition2))

#Removing the lease used factor levels in the data
cleaned_house_data <- cleaned_house_data %>%
  filter(Exterior1st!='AsphShn' & Exterior1st!='CBlock') %>%
filter(Exterior1st!='ImStucc')%>% filter(Functional!='Sev')%>%filter(Heating!='Floor' &
Heating!='OthW')%>%
filter(ExterCond!='Po')%>%filter(BsmtCond!='Po')%>%filter(Condition1!='RRNe')

#splitting the dataset for training the model and testing the model
set.seed(123)
train_index <- createDataPartition(cleaned_house_data$SalePrice, p = 0.8, list = FALSE)
train_data <- cleaned_house_data[train_index, ]
test_data <- cleaned_house_data[-train_index, ]

# Preprocessing the data using center and scale method to get better results
pre_proces<- preProcess(train_data, method = c("center", "scale"))
train_data <- predict(pre_proces, train_data)
test_data <- predict(pre_proces, test_data)

library("ggplot2")

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

# Evaluate the model
predictions <- predict(lm_model, newdata = test_data)
mse <- mean((test_data$SalePrice - predictions)^2)

```
rmse <- sqrt(mse)
r_squared <- cor(predictions, test_data$SalePrice)^2

# Print the results
cat("Mean squared error: ", mse, "\n")
cat("R-squared value: ", r_squared, "\n")
```

```
> cat("Mean squared error: ", mse, "\n")
Mean squared error:  0.1558268
> cat("R-squared value: ", r_squared, "\n")
R-squared value:  0.8962916
```

```
test_data <- test_data %>%
  mutate(diff = predictions - SalePrice)


ggplot(test_data, aes(x = predictions, y = SalePrice, color = diff)) +
  geom_point(alpha = 0.5) +
  geom_abline(intercept = 0, slope = 1, color = "red") +
  scale_color_gradient2(low = "red", mid = "blue", high = "green", midpoint = 0) +
  labs(x = "Predicted SalePrice", y = "Actual SalePrice", color = "Difference") +
  theme_classic()

##Random forest
library(caret)
set.seed(123)
#splitting the dataset into tarin and test

train_index <- createDataPartition(cleaned_house_data$SalePrice, p = 0.8, list = FALSE)
training <- cleaned_house_data[train_index, ]
testing <- cleaned_house_data[-train_index, ]
library(randomForest)

#Training the model using house_data
house_model <- randomForest(SalePrice~.,
                            data = training)
# Predict using the test set

prediction <- predict(house_model,testing)

# Evaluating the value root mean square(RMSE)

x <- prediction
y <- testing$SalePrice
a <- sqrt(sum((log(x)-log(y))^2)/length(y))
rmse <- round(a, digits = 5)
rmse
#calculating Mean absolute error
MAE <- mean(abs(prediction - testing$SalePrice))
MAE
#calculating sum of squares of residuals
```

```
res <- sum((testing$SalePrice - prediction)^2)
#calculating total sum of price (difference b/w saleprice and mean of saleprice)
tot <- sum((testing$SalePrice - mean(testing$SalePrice))^2)
#calculating the r2 value which always lies in o to 1.
R2 <- 1 - (res / tot)
R2

cat("Root mean squared error: ", rmse, "\n")
cat("R-squared value: ", R2, "\n")
```

```
> cat("Root mean squared error: ", rmse, "\n")
Root mean squared error:  0.14794
> cat("R-squared value: ", R2, "\n")
R-squared value:  0.8571166
>
```

```
##################################Question 3b)#####################################

library(ipred)
###Applying Re-sampling Methods to calculate the Misclassification error.

#defining a function that uses predict function inside
mypredict <- function(object, newdata)
  predict(object, newdata = newdata, type = c("response"))

#calculate the misclassification error using cross fold validation for randomForest algorithm
used
#to predict saleprice
cv_result<-errorest(SalePrice ~ ., data=cleaned_house_data, model = randomForest,
                    estimator = "cv", predict= mypredict)
cv_result
summary(cv_result)
```

```
> cv_result

Call:
errorest.data.frame(formula = SalePrice ~ ., data = cleaned_house_data,
    model = randomForest, predict = mypredict, estimator = "cv")

          10-fold cross-validation estimator of root mean squared error

Root mean squared error:  28608.88
```

```
#Calculating the Miscalssification error using BootStrapping for randomForest algorithm used
#to predict SalePrice
boot_result <- errorest(SalePrice ~ ., data=cleaned_house_data, model = randomForest,
                        estimator = "boot", predict= mypredict)
boot_result
```

```
> boot_result

Call:
errorest.data.frame(formula = SalePrice ~ ., data = cleaned_house_data,
    model = randomForest, predict = mypredict, estimator = "boot")

        Bootstrap estimator of root mean squared error
        with 25 bootstrap replications

Root mean squared error:  30296.6
```

####################################Question 4#################################

#Predicting the size of the living area based on sale_price of the house using

#splitting the size of the train and test dataset
set.seed(100)
trainIndex <- createDataPartition(cleaned_house_data$GrLivArea, p = 0.8, list = FALSE)
train <- house_data[trainIndex, ]
test <- house_data[-trainIndex, ]

# Fit a random forest model to predict living area based on the parameters which are realted to the
#target varaible

names(house_data_new)

model <- lm(GrLivArea ~ SalePrice+LotArea+FullBath+BedroomAbvGr+KitchenAbvGr+TotRmsAbvGrd+OverallQual+X2ndFlrSF
          , data = train)

# Predict living area for the test set based on the model
predicted_living_area <- predict(model, newdata = test)

# Compute the mean squared error
mse <- mean((predicted_living_area - test$GrLivArea)^2)

# Print the mean squared error
print(paste("Mean squared error:", mse))
r_squared <- cor(predicted_living_area, test$GrLivArea)^2

# Print the results
cat("Mean squared error: ", mse, "\n")
cat("R-squared value: ", r_squared, "\n")

```
> cat("Mean squared error: ", mse, "\n")
Mean squared error:  44096.11
> cat("R-squared value: ", r_squared, "\n")
R-squared value:  0.8611957
>
```