3/30/2022

University of Essex

MA321 Group Coursework

2021-22 - House Data Analysis

**Contents**

Abstract…………………………………………………………………………………………………………………………………………….3

Introduction………………………………………………………………………………………………………………………………………3

Question 1…………………………………………………………………………………………………………………………………………4

Question 2…………………………………………………………………………………………………………………………………………8

Question 3……………………………………………………………………………………………………………………………………….11

Question 4……………………………………………………………………………………………………………………………………….13

Appendix

Word Count : 3636

**Abstract:**

House data is combination of various features. There are two response variables overall condition and Saleprice. There are several other features which effect the response variables. In the analysis we did we made some conclusions for dataset. Predicted overall condition using classification and sale price using regression. We have also performed some analysis using clustering.

**Introduction**

The dataset given for the analysis consists of 1460 observations and 51 variables which describe the overall condition of a house depending on various factors such as lot frontage, year built, sale price, number of rooms etc.

We will load data into R by using function read.csv () or by importing directly by tab. Head () and Tail () function returns the first or last parts of a vector, matrix, table, data frame or function. Summary () is a generic function used to produce result summaries of the results of various model fitting functions.

The problem statement requires to classify the overall condition of a house as Poor, Good, Average based on the rating from 1 to 10. Classification is a technique for classifying data into a set of categories. The principal purpose of a classification problem is to determine which category or class new data will belong to. The dataset is loaded into R using read.csv () function. The libraries like dplyr , mice which are useful in building a model are loaded into R.

The data set is checked for missing values. Missing values cause certain problems on the decisions taken on analysing the data. However, excluding missing data from the evaluation causes loss of data and results in inaccurate conclusions. The ideal way to deal with missing values is to impute them with certain values which complete the dataset without disturbing the original structure.

**Question** 1:

The given house dataset have 1460 rows and 51 columns. There are 23 numeric columns and 28 character columns.

**Summarize the missing values in the data.**

There are four categorical columns which have more than 75% of NA values.

* Alley: indicates the type of alley access
* PoolQC: Pool Quality
* Fence: Fence Quality
* MiscFeature: Miscellaneous features not covered in other categories

The missing values indicate that many of the houses do not have alley access, no pool, no fence and no elevator, 2nd garage, shed or tennis court that is covered by the MiscFeature.

**Sale Price Histogram**

Chart, histogram

Description automatically generated

From the above histogram we can say that most of the houses are in less price range which is between zero and 200000. Maximum value of sale price is 755000. Minimum is 34900. It is clear that count of houses decreased with the increase in price after 200000.

Now let us analyse some of the important features of dataset and how they are related with sale price based on box plots. All the plots are included in appendix.

**Roof Style vs Sale Price(Fig 1)**

There are 6types of roof styles. Hip style have most range and Shed range is used least. There are outliers for Gable and Hip style roofs.

**Roof Malt vs Sale Price (Fig 2)**

There are 8 types of Roof malts. Among them WdShngl have most range and membrane, metal, roll and clytile is least range. There are outliers for compshg.

**Paved Drive vs Sale Price(Fig 3)**

There are three types of paved drives. Most prince range is for drive is of type Y and least range is for type P. There are outliers for N and Y types.

**Neighborhood vs Sale price(Fig 4)**

There are 13 types of neighborhoods. Houses in Stone Br have major price range and least range in NPKVill. There are outliers for almost all columns.

**Distribution of LotFrontage vs Sale Price(Fig 5)**

This is a numerical column and most of the houses are in range of 0 to 250000. There are manty outliers for this feature.

**Lot Config vs Sale Price(Fig 6)**

There are 5 types of lot configs. CulDsac have wide price range and least range is for FR2. There are outliers for all features except FR3.

**House Style vs Sale Price(Fig 7)**

There are 8 types of house styles. 2.5Fn type have wide price range and least are SFoyer. There are outliers for all features except 1.5Unf.

**Heating vs Sale Price(Fig 8)**

There are 6 types of heating. GasW have wide price range and least floor. GasA has outliers.

**Foundation vs Sale Price(Fig 9)**

There are 6types of foundation. Stone have wide price range. Least range is of type Pconc. There are outliers for types BrkTil,CBlock,PConc,Stab.

**ExterQual vs SalePrice(Fig 10)**

There are 4 types of ExterQuals. Type Ex have wide price range. Least range is of type Fa. There are outliers for all the columns.

**ExterCond vs SalePrice(Fig 11)**

There are 4types of ExterCond. Type Ex have wide price range. Least range is of type Gd. There are outliers for Gd and TA.

**Condition 2 vs SalePrice(Fig 12)**

There are 8 types in condition 2. Type PosN have wide price range. Least range of types POSA,RRAe and RRAn. There are outliers for type Norm.

**Condition1 vs SalePrice(Fig 13)**

There are 9 types in condition1. Mostly used of type RRNn. RRNe is the least used. There are outliers RRNe and RRNn.

**Bldg vs SalePrice(Fig 14)**

There are 5 types of buildings. Most of the building is of type 1Farm and least used is of type 2fmCon. There are outliers for all types of except Twnhs

**SaleCondition vs SalePrice (Fig 15)**

There are 6 types of saleconditions. Most of it are of type partial. Least is of type family. There are outliers for all type except AdjLand.

**Street vs Sale Price(fig 16)**

There are 2 types of street. Pave had wide price range and Grvl had less price range. There are outliers for pave.

**Principle Component Analysis(PCA)**

Principal Component Analysis, or PCA, is a dimensionality-reduction method that is often used to reduce the dimensionality of big dat, by changing a large set of variables into a smaller one that still have most of the information in the large set.

Summary of PCA is included in appendix. Below is the plot for PCA.

Chart, scatter chart

Description automatically generated

**Question 2:**

First, we were given a condition to split the OverallCond column in the data

* 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.

First we have added a column called overallcond1 and placed categories according to the above condition. We have Average 1130 Good 299 Poor 31 according to our data

We have relevelled the overallcond1 column. Relevelling Reorders the levels of an existing factor. The reference level is moved to the first position and the others retain their original order.

After using the mice package we have still nulls for the char columns, so we have replaced the NA values with below keywords

BsmtQual BsmtCond "No basement"

GarageType GarageCond "No garage"

Now we were asked to use Logistic Regression and predict the overallcond column.

Logistic Regression:

Logistic Regression is a measurable model that in its fundamental structure utilizes a strategic capacity to show a twofold reliant variable, albeit a lot more complicated augmentations exist. In relapse investigation, calculated relapse (or logit relapse) is assessing the boundaries of a strategic model.

• Strategic relapse is a grouping strategy, normally utilized in

twofold grouping issues - class names {0,1}

• The model deals with likelihood - it yields the likelihood 𝑝 𝐗

that the info 𝐗 is the class mark 1: 𝑝(𝐗) = Pr(𝑌 = 1|𝐗)

• A calculated relapse model is characterized as

.• 𝑝(𝑿) is likewise called strategic capacity

Calculated Relapse

• Strategic Relapse is a summed up direct model

• Chances: 𝑝 𝐗1−𝑝 𝐗 = 𝑒𝛽0+𝐗𝛃

• log-chances or logit

𝑙𝑜𝑔(𝑝 𝐗1−𝑝 𝐗) = 𝛽0 + 𝐗𝛃, direct in 𝐗

We haven’t used all the columns to fit the model, the overallcond column is dependent only on few columns which are mentioned below

OverallQual, SalePrice, YearBuilt, house\_data$RoofStyle

Since we have used 4 variables our linear equation becomes

Y= β0 + β1X1+ β2X2+ β3X3+ β4X4

After fitting the model and checking the summary we can observe below findings:

1-Every one-unit change in gre will build the log chances of getting concede by 0.002, and its p-esteem shows that it is fairly huge in deciding the concede.

2-Every unit expansion in GPA builds the log chances of getting concede by 0.80 and p-esteem shows that it is fairly huge in deciding the concede.

3-The distinction between Residual Deviance and AIC lets us know that the model is a solid match. More prominent the distinction better the model. Invalid abnormality is the worth when you just have capture in your situation without any factors and Residual Deviance is the worth when you are considering every one of the factors. It's a good idea to think about the model great assuming that distinction is sufficiently large

We have assigned the values predicted overallcond by logistic regression model to the predicted model. The overall condition has changed few values before and after logistic regression

Part b:

Now we are using Naive Bayes as the method as second method to predict the overallcond column

Naive Bayes Classification:

Naive Bayes is a straightforward strategy for developing classifiers: models that relegate class marks to issue occasions, addressed as vectors of component values, where the class names are drawn from some limited set. There is certainly not a solitary calculation for preparing such classifiers, yet a group of calculations in light of a typical guideline: all guileless Bayes classifiers expect that the worth of a specific element is autonomous of the worth of some other element, given the class variable. For instance, a natural product might be viewed as an apple on the off chance that it is red, round, and around 10 cm in measurement. A guileless Bayes classifier thinks about every one of these highlights to contribute autonomously to the likelihood that this organic product is an apple, no matter what any potential connections between the shading, roundness, and distance across highlights.

Using the split function, we have divided the data into training and testing data with 80% and 20% ratio. Then we have feed the training data to the Naïve bayes model. Then we have predicted the overallcond of the testing data by using above model. We have obtained the below confusion matrix. Below are the interpretations of the summary of Naïve Bayes and the confusion matrix

A picture containing text

Description automatically generatedThe above Naïve bayes classifier has an accuracy of 0.8629 and Kappa score of 0.629. Hence we can assume that the Naïve Bayes classifier is a good model to predict the overallcond of the houses.

Text

Description automatically generated

**Question 3:**

**3.1) Predicting House Prices**

In this part, we have two different ML models in order to predict house prices with the help of the best variables. The first one is a multiple Linear Regression model and the second one is Random Forrest Model. This dataset consists of 1460 instances of house data with 51 variables having 23 as numeric datatype and 28 object types.

**3.1.1) Linear Regression Model**

Linear Regression Model comes under supervised learning problems to investigate the relationship among multiple features and their relationship with the target variable i.e. Sales Price. This relationship is learned by the model and helps us in order to make the prediction with fewer errors.

Y = β0 + β1X1 +... + βnXn + e

Where,

β0 is the intercept,

βn is the coefficient of the slop.

E is the error.

After selecting the model, we had done the pre-processing part. Converting the object type variable into a numeric variable by calling the dummies method and we have also applied the normalization technique to make the numeric variables comparable. Thereafter, we have imputed the dataset with the help of MICE which will help our model to learn properly with minimum error. In order to Train and Test our dataset, we have split our dataset into training and testing datasets in the ratio of 80% training and 20% test sets. Investigation for a variance has also been done on the dataset and features having close to zero variation are removed from the dataset

We have trained the model with the help of multiple iterations and selected the best features for the model-building to predict the sales price. For predicting the sales prices, we have used the best-trained model on the test dataset. The model is giving good accuracy as it gave fewer errors while predicting the sales price. Residual standard error: 0.40740 on 1164 degrees of freedom, Multiple R-squared: 0.7505, Adjusted R-squared: 0.7496, F-statistic: 375.2 on 4 and 1164 DF are showing the best accuracy by the model. RMSE Score and R- Square Scores are 0.4485 and 0.7928978 respectively and by looking at the error value following model is tend to be more accurate among all the models.

**3.1.2) Random Forrest**

Random Forest techniques are used in the industry because of their compatibility and scalability without losing the accuracy of the model and are the best alternative to tree-based models which are more prone to overfitting. There are many used cases in the industry for example Amazon and Facebook use this model for the customers.

We have built a random forest model with the most important features and cross-validation with Root Mean Squared Error and R – Squared values for Mtry. After multiple imputations, we have found that 9 out of 10 times the Root Mean Squared Error and R – Squared values for Mtry are the lowest and highest. In order to develop the model, we had to tweak the parameters and evaluated it on a test set. As we have normalized our dataset while doing the pre-processing, the Root Mean Squared Error value is coming out to be on the lower side.

Root Mean Squared Error and R – Squared values for random forests are coming out to be 0.39662.75 and 0.80144 respectively. It also shows that the Random Forest model gives us the best model among other implemented ones.

**3.2) Re-sampling methods to estimate the test errors**

**3.2.1) Error estimation using Cross-validation (CV)**

For error estimating we have first used the cross-validation resampling method and have applied it to test both Random Forest and Linear Regression Models on sample datasets. We have selected the 10-cross validation with parameters as control.errorest (k= 10).

There are two ways to measure the model error i.e., Root Mean Squared Error and Mean Absolute Error. The lower they are the better our model is. We have divided our dataset into train and test sets in 80% and 20% respectively. Here we will mainly deal with the Root Mean Squared Error and Mean Absolute Error as they are the more accurate and help us in determining the accuracy of the model. Root Mean Squared Error value is 0.23 and Mean Absolute Error is 0.294.

**3.2.1) Error estimation using BootStrap**

For error estimating Now, we will use the Bootstrap resampling method and have applied it to test both Random Forest and Linear Regression Models on sample datasets. We have selected the boot as parameters as a model estimator.

Decision tree models can be generated with the help of bootstrap. This method resamples from the original dataset with a replacement technique. It is one of the easiest resampling techniques to use There are two ways to measure the model error i.e., Root Mean Squared Error and Mean Absolute Error. The lower they are the better our model is. We have divided our dataset into train and test sets in 80% and 20% respectively. Here we will mainly deal with the Root Mean Squared Error and Mean Absolute Error as they are the more accurate and help us in determining the accuracy of the model. Root Mean Squared Error value is 0.473 and Mean Absolute Error is 0.324

**Question 4:**

Clustering is one of the most popular and commonly used unsupervised learning techniques used in machine learning. In clustering or cluster analysis in R, we attempt to**group objects with similar traits and features together**, such that a larger set of objects is divided into smaller sets of objects.

We can also investigate the relationship between the year a house is built with the overall quality and overall condition of a house. This information can be useful because if it is revealed that the variable ‘YearBuilt’ has a positive relationship with some of the negative features e.g. a poor ‘OverallQual’ score then this could be something that the government would want to look into to resolve this issue. Our initial hypothesis is that the later a house is built, the better the overall condition and overall quality.

As the dataset has a large number of variables, there will be some which provide little information in regard to our problem.

**METHODOLOGY**

As from clustering methodology I decided to perform analysis on K-means clustering. Here we use factoextra and cluster libraries to perform the clustering analysis to the dataset. To perform k-means clustering in R we can use the built-in **kmeans()** function.

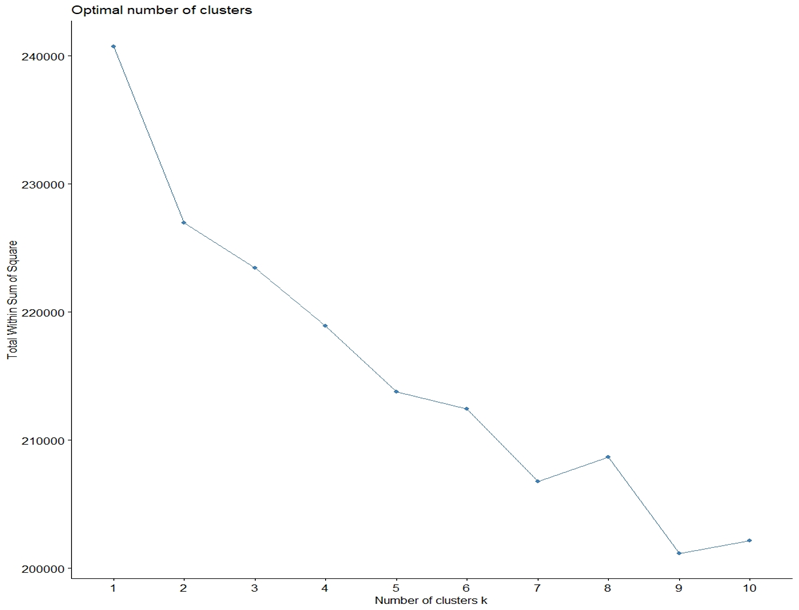
**kmeans(data, centers, nstart)**

**data:** Name of the dataset.

**centers:** The number of clusters, denoted *k*.

**nstart:** The number of initial configurations. Because it’s possible that different initial starting clusters can lead to different results, it’s recommended to use several different initial configurations. The k-means algorithm will find the initial configurations that lead to the smallest within-cluster variation.

**1)Number of Clusters vs. the Total Within Sum of Squares**

****

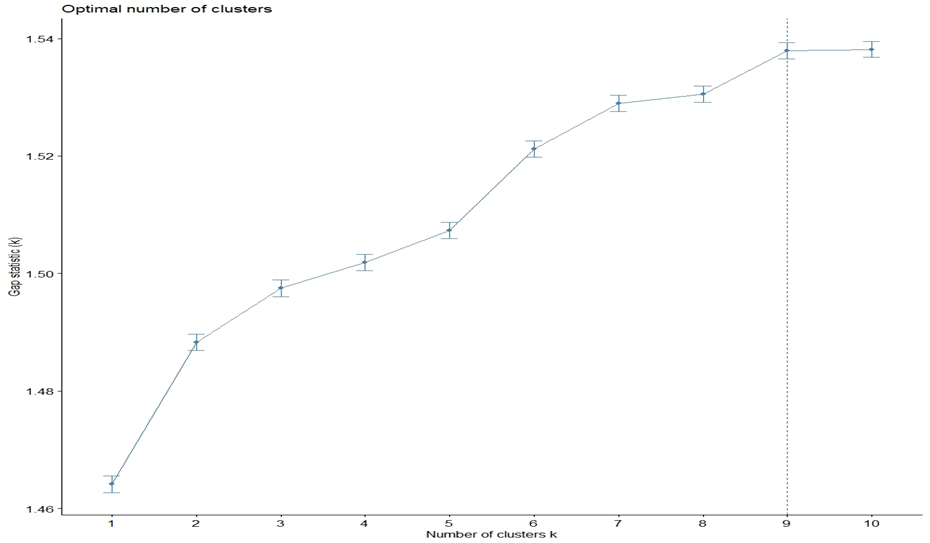
Typically when we create this type of plot we look for an “elbow” where the sum of squares begins to “bend” or level off. This is typically the optimal number of clusters. For this plot it appear that there is a bit of an elbow or “bend” at k = 9 clusters.

**2. Number of Clusters vs. Gap Statistic**

Another way to determine the optimal number of clusters is to use a metric known as the [gap statistic](http://web.stanford.edu/~hastie/Papers/gap.pdf), which compares the total intra-cluster variation for different values of k with their expected values for a distribution with no clustering.

We can calculate the gap statistic for each number of clusters using the **clusGap()** function from the *cluster* package along with a plot of clusters vs. gap statistic using the **fviz\_gap\_stat()** function.

Here from the below plot we can see that plots are plotted between gap statistic (k) and number of clusters k. The number of clusters that are taken should be considered depends upon the gap statistic(k). From the below plot the number of clusters that should be considered is 9. This is because the gap statistic (k) is high when number of clusters are 9.



**3.Perform K-Means Clustering with Optimal *K***

We can perform k-means clustering on the dataset using the optimal value for *k* of 9:

**#make this example reproducible**

**set.seed(1)**

**#perform k-means clustering with k = 9 clusters**

**km <- kmeans(stand\_house\_datA, centers = 9, nstart = 25)**

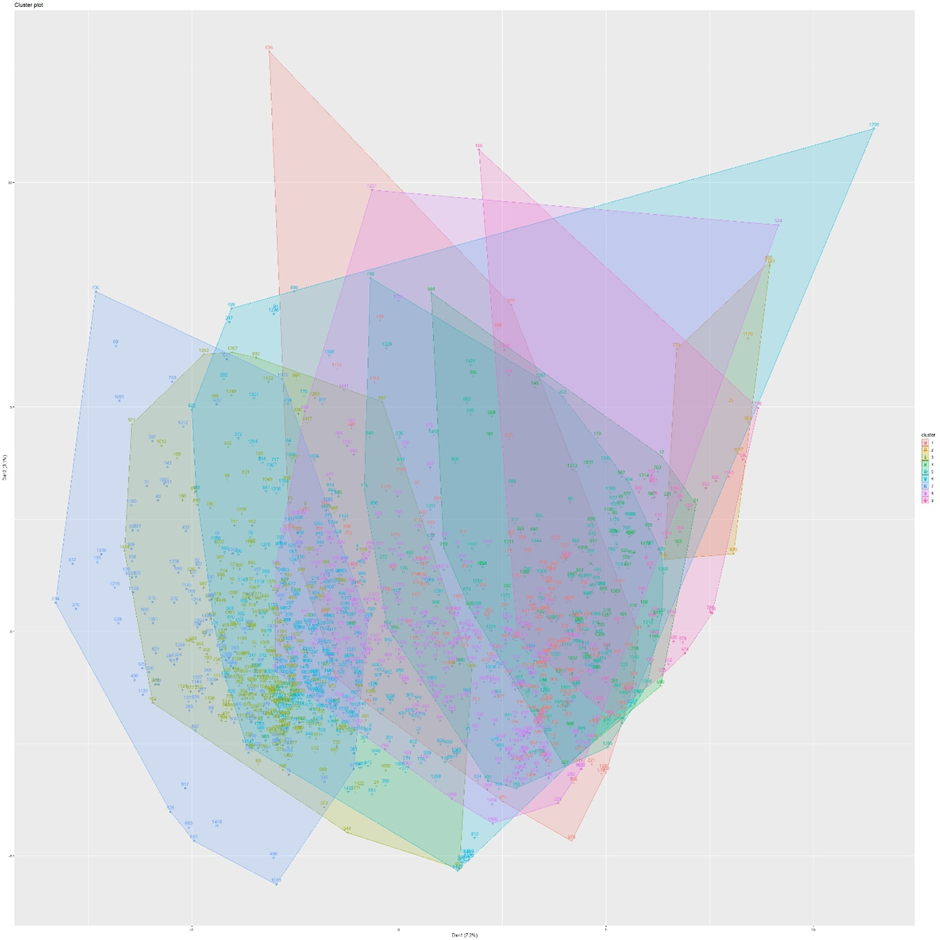
**#view results**

**km**

**data=**The dataset we used to perform kmeans clustering.

**Centres=**Number of clusters.

**Nstart=** The number of initial configurations. Because it’s possible that different initial starting clusters can lead to different results, it’s recommended to use several different initial configurations. The k-means algorithm will find the initial configurations that lead to the smallest within-cluster variation.



### ****Pros & Cons of K-Means Clustering:****

K-means clustering offers the following benefits:

* It is a fast algorithm.
* It can handle large datasets well.
* High Performance K-Means algorithm has linear time complexity and it can be used with large datasets conveniently.
* With unlabelled big data K-Means offers many insights and benefits as an unsupervised clustering algorithm.

**Disadvantages of k-means clustering:**

* One of the inconsistencies of K-Means algorithm is that results will differ based due to random centroid initialization. Unless you pick the centroids at fixed positions, which is not a common practice K-Means can come up with different clusters after its iterations.
* K-Means generates spherical clusters. So, if you have overlapping clusters or arbitrary shapes K-Means won't be able to cluster those.

**Appendix:**

library(ggplot2)  
library(dplyr)  
library(tidyverse) # for tidy data packages, automatically loads dplyr  
library(magrittr) # for piping  
install.packages("roperators")  
library(roperators)  
install.packages("ExPanDaR")  
library(purrr)  
library(ipred)  
library(tree)  
library(rpart)  
library(ada)  
library(readr)  
library(ExPanDaR)  
library(ggplot2)  
library(ISLR)  
library(nnet)  
library(roperators)  
house\_data <- read.csv(file = "C:/Users/91960/OneDrive/MA321-Applied Statistics/Assessment/MA321\_Group\_coursework-20220329/house-data.csv")  
  
#summary of house data  
summary(house\_data)  
  
num\_data<-names(which(sapply(house\_data,is.numeric)))  
  
char\_data<-names(which(sapply(house\_data,is.character)))  
  
#plots  
  
ggplot(data=house\_data, aes(SalePrice)) +  
  geom\_histogram(col="red", aes(fill=..count..)) +  
  scale\_fill\_gradient("Count", low="white", high="red") +  
  labs(title = "Sale price histogram", x = "Sale price", y = "Count")  
summary(house\_data$SalePrice)  
  
#Boxplots  
#1  
ggplot(data=house\_data, aes(y= SalePrice, x=HouseStyle, fill=HouseStyle) ) +  
  geom\_boxplot() +  
  ggtitle("Distribution of HouseStyle") +    
  ylab("Sale Price") +  
  xlab("HouseStyle")  
#2  
ggplot(data=house\_data, aes(y= SalePrice, x=RoofMatl, fill=RoofMatl ) ) +  
  geom\_boxplot() +  
  ggtitle("Distribution of RoofMatl") +    
  ylab("Sale Price") +  
  xlab("RoofMatl")  
#3  
ggplot(data=house\_data, aes(x= SalePrice, y=Heating, fill=Heating ) ) +  
  geom\_boxplot() +  
  ggtitle("Distribution of Heating") +    
  ylab("Sale Price") +  
  xlab("Heating")  
#4  
ggplot(data=house\_data, aes(y= SalePrice, x=Condition1, fill=Condition1) ) +  
  geom\_boxplot() +  
  ggtitle("Distribution of Condition1") +    
  ylab("Sale Price") +  
  xlab("Condition1")  
#5  
ggplot(data=house\_data, aes(y= SalePrice, x=Condition2, fill=Condition2) ) +  
  geom\_boxplot() +  
  ggtitle("Distribution of Condition2") +    
  ylab("Sale Price") +  
  xlab("Condition2")  
#6  
ggplot(data=house\_data, aes(y= SalePrice, x=RoofStyle, fill=RoofStyle) ) +  
  geom\_boxplot() +  
  ggtitle("Distribution of RoofStyle") +    
  ylab("Sale Price") +  
  xlab("RoofStyle")  
#7  
ggplot(data=house\_data, aes(y= SalePrice, x=LotFrontage, fill=LotFrontage) ) +  
  geom\_boxplot() +  
  ggtitle("Distribution of lotfontage") +    
  ylab("Sale Price") +  
  xlab("Lotfrontage")  
#8  
ggplot(data=house\_data, aes(y= SalePrice, x=ExterQual, fill=ExterQual ) ) +  
  geom\_boxplot() +  
  ggtitle("Distribution of ExterQual") +    
  ylab("Sale Price") +  
  xlab("ExterQual")  
#9  
ggplot(data=house\_data, aes(y= SalePrice, x=BldgType, fill=BldgType) ) +  
  geom\_boxplot() +  
  ggtitle("Distribution of BldgType") +    
  ylab("Sale Price") +  
  xlab("BldgType")  
#10  
ggplot(data=house\_data, aes(y= SalePrice, x=ExterCond, fill=ExterCond) ) +  
  geom\_boxplot() +  
  ggtitle("Distribution of ExterCond") +    
  ylab("Sale Price") +  
  xlab("ExterCond")  
#11  
ggplot(data=house\_data, aes(y= SalePrice, x=Foundation, fill=Foundation) ) +  
  geom\_boxplot() +  
  ggtitle("Distribution of Foundation") +    
  ylab("Sale Price") +  
  xlab("Foundation")  
#12  
ggplot(data=house\_data, aes(y= Neighborhood, x=SalePrice, fill=Neighborhood) ) +  
  geom\_boxplot() +  
  ggtitle("Distribution of Neighborhood") +    
  ylab("Sale Price") +  
  xlab("Neighborhood")  
#13  
ggplot(data=house\_data, aes(y= SalePrice, x=LotConfig, fill=LotConfig) ) +  
  geom\_boxplot() +  
  ggtitle("Distribution of LotConfig") +    
  ylab("Sale Price") +  
  xlab("LotConfig")  
#14  
ggplot(data=house\_data, aes(y= SalePrice, x=Street, fill=Street) ) +  
  geom\_boxplot() +  
  ggtitle("Distribution of Street") +    
  ylab("Sale Price") +  
  xlab("Street")  
#15  
ggplot(data=house\_data, aes(y= SalePrice, x=PavedDrive, fill=PavedDrive) ) +  
  geom\_boxplot() +  
  ggtitle("Distribution of PavedDrive") +    
  ylab("Sale Price") +  
  xlab("PavedDrive")  
#16  
ggplot(data=house\_data, aes(y= SalePrice, x=SaleCondition, fill=SaleCondition) ) +  
  geom\_boxplot() +  
  ggtitle("Distribution of SaleCondition") +    
  ylab("Sale Price") +  
  xlab("SaleCondition")  
  
  
str(house\_data)  
  
  
  
#Data pre processing  
data <- read.csv(file = "C:/Users/91960/OneDrive/MA321-Applied Statistics/Assessment/MA321\_Group\_coursework-20220329/house-data.csv")  
library(mice)  
dim(data)  
  
str(data)  
  
  
  
  
#Consider columns with more than 75% na  
sapply(data, function(x) mean([is.na](http://is.na/)(x)))  
#Alley, PoolQC, Fence,MiscFeature  
  
drop <- c("Alley", "PoolQC","Fence","MiscFeature")  
data = data[,!(names(data) %in% drop)]  
  
library(mice)  
md.pattern(data)  
imputed\_Data <- mice(data,method = 'pmm')  
  
new\_data\_4=complete(imputed\_Data)  
#summary\_table<-prepare\_descriptive\_table(df,format = "html")  
  
new\_data\_4 = new\_data\_4[,!(names(new\_data\_4) %in% c("Id"))]  
  
house\_data=new\_data\_4  
house\_data$BsmtQual[[is.na](http://is.na/)(house\_data$BsmtQual)] <- "No basement"  
  
house\_data$BsmtCond[[is.na](http://is.na/)(house\_data$BsmtCond)] <- "No basement"  
  
house\_data$GarageType[[is.na](http://is.na/)(house\_data$GarageType)] <- "No garage"  
  
house\_data$GarageCond[[is.na](http://is.na/)(house\_data$GarageCond)] <- "No garage"  
  
house\_num\_data<-names(which(sapply(house\_data,is.numeric)))  
  
house\_char\_data<-names(which(sapply(house\_data,is.character)))  
  
house\_cat<-house\_data[,!(names(house\_data) %in% house\_num\_data)]  
  
house\_num<-house\_data[,(names(house\_data) %in% house\_num\_data)]  
  
  
#data one hot encoding  
data\_new <- sapply(house\_cat, unclass)  
  
house\_dummies <- as.data.frame(model.matrix(~.-1, house\_cat))  
  
  
house\_data\_numeric=as.data.frame(lapply(house\_data[c(1,2,11,12,13,17,23,25,26,27,28,29,30,31,33,35,37,40,41,42,43,46)], as.numeric))  
new\_data=cbind(house\_dummies, house\_data\_numeric)  
#Standardized data  
standardisedconcentrations <- as.data.frame(scale(house\_num))  
house\_data.pca <- prcomp(standardisedconcentrations, scale. = T)  
plot(house\_data.pca, loadings = TRUE)  
  
summary(house\_data.pca ) # as before look at the proportions of variance  
screeplot(house\_data.pca , type="lines") # plot the variances in decreasing order  
(house\_data.pca $sdev)^2 # Kaiser's criterion: retain  principal component variance >1  
library(devtools)  
install\_github("vqv/ggbiplot")  
library(ggbiplot)  
  
ggbiplot(house\_data.pca)  
  
################################################################  
  
#Task 2  
  
###DIvided the OverallCond into 3 categories (Average, Good and Poor)  
  
library(ISLR)  
house\_data$overallcond1 <- ifelse(house\_data$OverallCond < 4, "Poor",  
                                  ifelse(house\_data$OverallCond < 7, "Average", "Good"))  
  
house\_data$overallcond1<- factor(house\_data$overallcond1)  
house\_data$overallcond1  
  
  
  
  
### The Multinomial Logistic Regression#####  
  
house\_data$overallcond1 = relevel(house\_data$overallcond1, ref = "Average")  
  
mlogi <- multinom(house\_data$overallcond1 ~ house\_data$OverallQual  
                  + house\_data$SalePrice + house\_data$YearBuilt+ +house\_data$RoofStyle, data = house\_data)  
  
summary(mlogi)  
  
  
### the coefficient####  
exp(coef(mlogi))  
  
  
house\_data$predicted <- predict(mlogi, newdata = house\_data, "class")  
table(house\_data$overallcond1)  
  
table(house\_data$predicted)  
  
house\_data  
  
install.packages("e1071")  
install.packages("caTools")  
install.packages("caret")  
  
# Loading package  
library(e1071)  
library(caTools)  
library(caret)  
  
# Splitting data into training and testing data  
split <- sample.split(house\_data, SplitRatio = 0.8)  
train\_data <- subset(house\_data, split == "TRUE")  
test\_data <- subset(house\_data, split == "FALSE")  
  
  
# Fitting Naive Bayes Model  
# to training dataset  
set.seed(120)  # Setting Seed  
classifier\_NB <- naiveBayes(overallcond1 ~ ., data = train\_data)  
summary(classifier\_NB)  
  
# Predicting on test data'  
y\_pred <- predict(classifier\_NB, newdata = test\_data)  
  
# Confusion Matrix  
confusion\_matrix <- table(test\_data$overallcond1, y\_pred)  
confusion\_matrix  
  
# Model Evaluation  
confusionMatrix(confusion\_matrix)  
  
#Task 3  
#Initialising Required Libraries for Regression  
library(caret)  
library(randomForest)  
library(gbm)  
library(caTools)  
library(class)  
library(MASS)  
library(ipred)  
  
#Reading the data from csv file  
house\_data<-read.csv("house-data.csv",row.names=1)  
  
#Creating the data partition using 80%-20% combination  
training\_samples <- createDataPartition(house\_data$SalePrice, p = .8,  
                                        list = FALSE,  
                                        times = 1)  
training\_data  <- house\_data[training\_samples, ]  
test\_data <- house\_data[-training\_samples, ]  
  
#1.Linear Model - Model for Predicting Sales  
  
#training the Data  
linear\_model<-lm(SalePrice~ OverallQual+GrLivArea+GarageArea+TotalBsmtSF, data = training\_data)  
  
summary(linear\_model)  
  
#Prediction function  
linear\_prediction<-predict(linear\_model,test\_data)  
  
#RMSE score  
RMSE\_score<-RMSE(linear\_prediction,test\_data$SalePrice)  
RMSE\_score  
  
#R2 score  
R2\_score<-R2(linear\_prediction,test\_data$SalePrice)  
R2\_score  
  
#2.Random forest regressor - Model for Predicting Sales  
  
#Traning the Data  
RandomForest\_model <- train(SalePrice~ OverallQual+GrLivArea+GarageArea+TotalBsmtSF, data = training\_data, method = "rf", metric="RMSE",ntree = 100,verbose = TRUE)  
summary(RandomForest\_model)  
  
#model prediction  
RandomForest\_prediction<-predict(RandomForest\_model,test\_data)  
  
#RSME score  
RMSE\_score<-RMSE(RandomForest\_prediction,test\_data$SalePrice)  
RMSE\_score  
  
#R2 score  
R2\_score<-R2(RandomForest\_prediction,test\_data$SalePrice)  
R2\_score  
  
  
#Sampling Methods:  
  
#Error estimation using Cross validation (CV):  
  
#Random forest  
errorest(SalePrice~ OverallQual+GrLivArea+GarageArea+TotalBsmtSF, data=house\_data,model=randomForest,estimator= c("cv"),est.para=control.errorest(k=10), predict= RandomForest\_prediction)  
#Linear model  
errorest(SalePrice~ OverallQual+GrLivArea+GarageArea+TotalBsmtSF, data=house\_data,model=lm,estimator= c("cv"),est.para=control.errorest(k=10), predict= linear\_prediction)  
  
#Error estimation using BootStrap:  
  
#Linear model  
  
errorest(SalePrice~ OverallQual+GrLivArea+GarageArea+TotalBsmtSF, data=house\_data,model=lm,estimator= "boot", predict= linear\_prediction)  
  
#Random forest  
errorest(SalePrice~ OverallQual+GrLivArea+GarageArea+TotalBsmtSF, data=house\_data,model=randomForest,estimator= "boot", predict= RandomForest\_prediction)  
  
#Task 4  
  
  
  
num\_features<- names(which(sapply(house\_data, is.numeric)))  
cat\_features<- names(which(sapply(house\_data, is.character)))  
  
house\_num <- house\_data[, names(house\_data) %in% num\_features]  
  
house\_cat <- house\_data[, !(names(house\_data) %in% num\_features)]  
#remove rows with missing values  
house\_data <- na.omit(house\_data)  
house\_dummies <- as.data.frame(model.matrix(~.-1, house\_cat))  
dim(house\_dummies)  
house\_data1 <- cbind(house\_num,house\_dummies)  
head(house\_data1)  
stand\_house\_datA=standardizedData<-as.data.frame(scale(house\_data1))  
fviz\_nbclust(stand\_house\_datA, kmeans, method = "wss")  
#calculate gap statistic based on number of clusters  
gap\_stat <- clusGap(stand\_house\_datA,  
                    FUN = kmeans,  
                    nstart = 25,  
                    K.max = 50,  
                    B = 50)  
#plot number of clusters vs. gap statistic  
fviz\_gap\_stat(gap\_stat)  
#make this example reproducible  
set.seed(1)  
  
#perform k-means clustering with k = 9 clusters  
km <- kmeans(house\_data1, centers = 9, nstart = 25)  
  
#view results  
km  
  
#plot results of final k-means model  
fviz\_cluster(km, data = house\_data1)

Chart, box and whisker chart

Description automatically generated

**Fig 1**

Chart, box and whisker chart

Description automatically generated

**Fig 2.**

Chart, box and whisker chart

Description automatically generated

**Fig 3**

Chart

Description automatically generated

**Fig 4**

Chart, histogram

Description automatically generated

**Fig 5**

Chart, box and whisker chart

Description automatically generated

**Fig 6**

Chart, box and whisker chart

Description automatically generated

**Fig 7**

Chart, box and whisker chart

Description automatically generated

**Fig 8**

Chart, box and whisker chart

Description automatically generated

**Fig 9**

Chart, box and whisker chart

Description automatically generated

**Fig 10**

Chart, box and whisker chart

Description automatically generated

**Fig 11**

**Fig 12**Chart, box and whisker chart

Description automatically generated

Chart, box and whisker chart

Description automatically generated

**Fig 13**

Chart, box and whisker chart

Description automatically generated

**Fig 14**

Chart, box and whisker chart

Description automatically generated

**Fig 15**

Chart, box and whisker chart

Description automatically generated

**Fig 16**

**Summary for PCA**

**A screenshot of a computer

Description automatically generated with low confidence**