Models, Graphics and LinAlg

Gurleen Kaur(300327252)

30/01/2022

```
library(tidyverse)

library(visdat)

library(rsample)

library(recipes)

library(caret)

library(matlib)

library(pracma)

library(EigenR)

library(Matrix)

library(psych)
```

Introduction

Housing costs play a significant role in decision-making at all levels, from recent graduates to policymakers. This study is carried out on housing data in order to forecast home prices.

1 Modeling

1.1 Cleaning and preparing

```
#replacing blanks with NA's

housingdata <- read.csv( "housingdata.csv",na.strings = c("", "NA"))



# deleting irrelevant columns
```

```
housingdata <- housingdata %>%
               select(-c(Order,PID,X,X.1)) %>%
               rename(unknown_variable = X.2)

# converting characters into factors

housingdata<- as.data.frame(unclass(housingdata),stringsAsFactors=TRUE)
```

Exploratory Data Analysis

Overview of the Dataset

It can be seen this data set has 2930 rows and 80 columns. An overview of variables can be seen below:

```
# Dimensions and column names of dataset

dim(housingdata)
```

```
## [1] 2930   80
```

```
# Nature of Variables

str(housingdata)
```

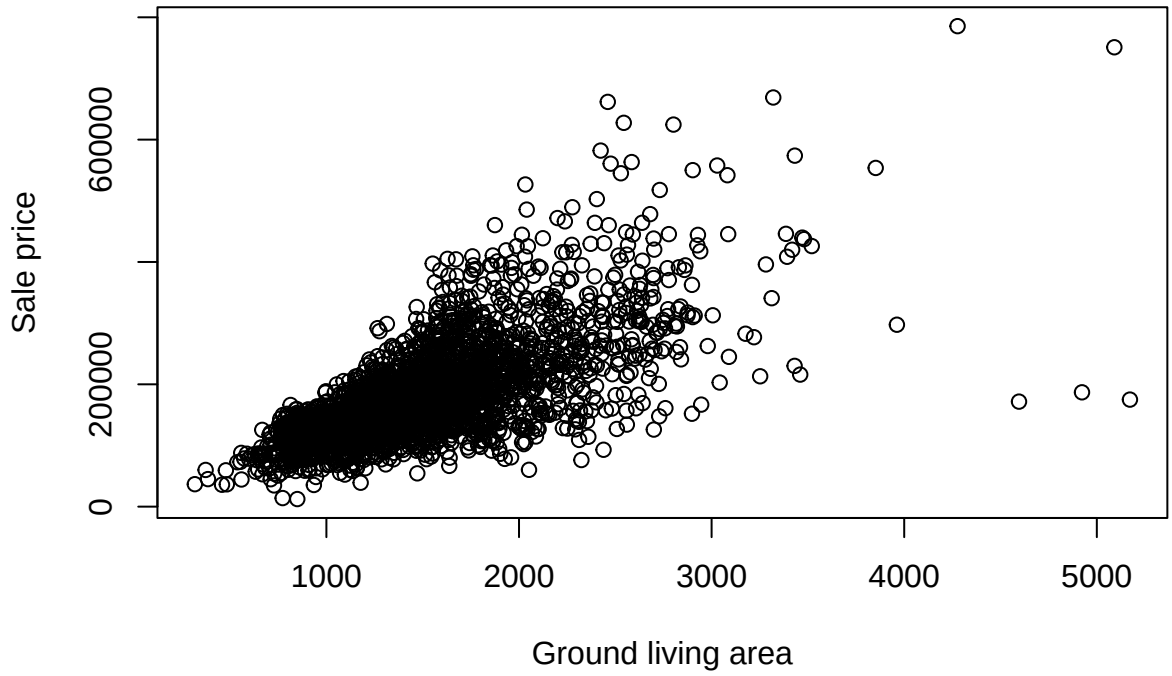
```
## 'data.frame':    2930 obs. of  80 variables:
##  $ MS.SubClass     : int  20 20 20 20 60 60 120 120 120 60 ...
##  $ MS.Zoning       : Factor w/ 7 levels "A (agr)","C (all)",..: 6 5 6 6 6 6 6 6 6 6 ...
##  $ Lot.Frontage    : num  143.8 81.6 79.4 91.1 75.5 ...
##  $ Lot.Area        : num  32405 11622 13982 11383 14107 ...
##  $ Street          : Factor w/ 2 levels "Grvl","Pave": 2 2 NA 2 2 2 2 2 2 2 ...
##  $ Alley           : Factor w/ 2 levels "Grvl","Pave": NA NA NA NA NA NA NA NA NA NA ...
##  $ Lot.Shape       : Factor w/ 4 levels "IR1","IR2","IR3",..: 1 4 1 4 1 1 4 1 1 4 ...
##  $ Land.Contour    : Factor w/ 4 levels "Bnk","HLS","Low",..: 4 4 4 4 4 4 4 2 4 4 ...
##  $ Utilities       : Factor w/ 3 levels "AllPub","NoSeWa",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Lot.Config      : Factor w/ 5 levels "Corner","CulDSac",..: 1 5 1 1 5 5 5 5 5 5 ...
##  $ Land.Slope      : Factor w/ 3 levels "Gtl","Mod","Sev": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Neighborhood    : Factor w/ 28 levels "Blmngtn","Blueste",..: 16 16 16 16 9 9 25 NA 25 9 ...
##  $ Condition.1     : Factor w/ 9 levels "Artery","Feedr",..: 3 2 3 3 3 3 NA 3 3 3 ...
##  $ Condition.2     : Factor w/ 8 levels "Artery","Feedr",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ Bldg.Type       : Factor w/ 5 levels "1Fam","2fmCon",..: NA NA 1 1 1 1 5 5 5 1 ...
##  $ House.Style     : Factor w/ 8 levels "1.5Fin","1.5Unf",..: 3 3 3 3 6 6 3 3 3 6 ...
##  $ Overall.Qual    : int  6 5 NA 7 NA 6 8 8 8 7 ...
##  $ Overall.Cond    : int  5 6 6 5 5 6 5 5 5 5 ...
##  $ Year.Built      : int  1960 1961 1958 1968 1997 1998 2001 1992 1995 1999 ...
##  $ Year.Remod.Add  : int  1960 1961 1958 1968 1998 1998 2001 1992 1996 1999 ...
##  $ Roof.Style      : Factor w/ 6 levels "Flat","Gable",..: 4 2 4 4 2 2 2 2 2 2 ...
##  $ Roof.Matl       : Factor w/ 7 levels "ClyTile","CompShg",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ Exterior.1st    : Factor w/ 16 levels "AsbShng","AsphShn",..: 4 14 15 4 14 14 6 7 6 14 ...
##  $ Exterior.2nd    : Factor w/ 17 levels "AsbShng","AsphShn",..: 11 15 16 NA 15 15 6 7 NA 15 ...
```

```
##  $ Mas.Vnr.Type    : Factor w/ 5 levels "BrkCmn","BrkFace",..: 5 4 2 4 4 2 4 4 4 4 ...
##  $ Mas.Vnr.Area    : int  112 0 108 0 0 20 0 0 0 0 ...
##  $ Exter.Qual      : Factor w/ 4 levels "Ex","Fa","Gd",..: 4 4 4 3 4 4 3 3 3 4 ...
##  $ Exter.Cond      : Factor w/ 5 levels "Ex","Fa","Gd",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ Foundation      : Factor w/ 6 levels "BrkTil","CBlock",..: 2 2 2 2 3 3 3 3 3 3 ...
##  $ Bsmt.Qual       : Factor w/ 5 levels "Ex","Fa","Gd",..: 5 5 5 5 3 5 3 3 3 5 ...
##  $ Bsmt.Cond       : Factor w/ 5 levels "Ex","Fa","Gd",..: 3 5 5 5 5 5 5 5 5 5 ...
##  $ Bsmt.Exposure   : Factor w/ 4 levels "Av","Gd","Mn",..: 2 4 4 4 NA NA NA 4 4 4 ...
##  $ BsmtFin.Type.1  : Factor w/ 6 levels "ALQ","BLQ","GLQ",..: 2 NA 1 1 3 3 3 1 3 6 ...
##  $ BsmtFin.SF.1    : Factor w/ 1021 levels "-1","#VALUE!",..: 730 570 NA 48 NA 694 710 384 111 4 ...
##  $ BsmtFin.Type.2  : Factor w/ 6 levels "ALQ","BLQ","GLQ",..: 6 4 6 6 6 6 6 6 6 6 ...
##  $ BsmtFin.SF.2    : int  0 144 NA 0 0 0 0 0 0 0 ...
##  $ Bsmt.Unf.SF     : Factor w/ 1167 levels "#VALUE!","0",..: 618 NA 645 73 83 524 873 119 587 4 ...
##  $ Total.Bsmt.SF   : int  1080 882 1329 2110 928 926 1338 1280 1595 994 ...
##  $ Heating         : Factor w/ 6 levels "Floor","GasA",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ Heating.QC      : Factor w/ 5 levels "Ex","Fa","Gd",..: 2 5 5 1 3 1 1 1 1 3 ...
##  $ Central.Air     : Factor w/ 2 levels "N","Y": NA 2 2 2 2 2 2 2 2 2 ...
##  $ Electrical      : Factor w/ 5 levels "FuseA","FuseF",..: 5 5 5 NA 5 5 5 5 5 5 ...
##  $ X1st.Flr.SF     : int  1602 898 1315 2125 855 885 1387 1334 1807 1138 ...
##  $ X2nd.Flr.SF     : int  0 0 0 0 701 678 0 0 0 776 ...
##  $ Low.Qual.Fin.SF : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Gr.Liv.Area     : int  1788 888 1327 2061 1611 1553 1165 1320 1792 1842 ...
##  $ Bsmt.Full.Bath  : int  1 0 0 1 0 0 1 0 1 0 ...
##  $ Bsmt.Half.Bath  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Full.Bath       : int  1 1 1 2 2 2 2 2 2 2 ...
##  $ Half.Bath       : int  0 0 1 1 1 1 0 0 0 1 ...
##  $ Bedroom.AbvGr   : int  3 2 3 3 3 3 2 2 2 3 ...
##  $ Kitchen.AbvGr   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Kitchen.Qual    : Factor w/ 5 levels "Ex","Fa","Gd",..: 5 5 3 1 5 3 3 3 3 3 ...
##  $ TotRms.AbvGrd   : int  7 5 6 8 6 7 6 5 5 7 ...
##  $ Functional      : Factor w/ 8 levels "Maj1","Maj2",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ Fireplaces      : int  2 0 0 2 1 1 0 0 1 1 ...
##  $ Fireplace.Qu    : Factor w/ 5 levels "Ex","Fa","Gd",..: 3 NA NA 5 5 3 NA NA 5 5 ...
##  $ Garage.Type     : Factor w/ 6 levels "2Types","Attchd",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ Garage.Yr.Blt   : int  1960 1961 1958 1968 1997 1998 2001 1992 1995 1999 ...
##  $ Garage.Finish   : Factor w/ 3 levels "Fin","RFn","Unf": 1 3 3 1 1 1 1 2 2 1 ...
##  $ Garage.Cars     : int  2 1 1 2 2 2 2 2 2 2 ...
##  $ Garage.Area     : int  528 730 312 522 482 470 582 506 608 442 ...
##  $ Garage.Qual     : Factor w/ 5 levels "Ex","Fa","Gd",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ Garage.Cond     : Factor w/ 5 levels "Ex","Fa","Gd",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ Paved.Drive     : Factor w/ 3 levels "N","P","Y": 2 3 3 3 3 3 3 3 3 3 ...
##  $ Wood.Deck.SF    : int  210 140 393 0 212 360 0 0 237 140 ...
##  $ Open.Porch.SF   : int  62 0 36 0 34 36 0 82 152 60 ...
##  $ Enclosed.Porch  : int  0 0 0 0 0 0 170 0 0 0 ...
##  $ unknown_variable: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Screen.Porch    : int  0 120 0 0 0 0 0 144 0 0 ...
##  $ Pool.Area       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Pool.QC         : Factor w/ 4 levels "Ex","Fa","Gd",..: NA NA NA NA NA NA NA NA NA NA ...
##  $ Fence           : Factor w/ 4 levels "GdPrv","GdWo",..: NA 3 NA NA 3 NA NA NA NA NA ...
##  $ Misc.Feature    : Factor w/ 5 levels "Elev","Gar2",..: NA NA 2 NA NA NA NA NA NA NA ...
##  $ Misc.Val        : int  0 0 12500 0 0 0 0 0 0 0 ...
##  $ Mo.Sold         : int  5 6 6 4 3 6 4 1 3 6 ...
##  $ Yr.Sold         : int  2010 2010 2010 2010 2010 2010 2010 2010 2010 2010 ...
##  $ Sale.Type       : Factor w/ 10 levels "COD","Con","ConLD",..: 10 10 10 10 10 10 10 10 10 10 ...
```

```
##  $ Sale.Condition  : Factor w/ 6 levels "Abnorml","AdjLand",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ SalePrice       : int  209200 107700 163600 257900 184400 208700 213700 193600 241700 191200 ...
```

Taking a step further, I decided to see how the ground living area impacts sale price of a house.

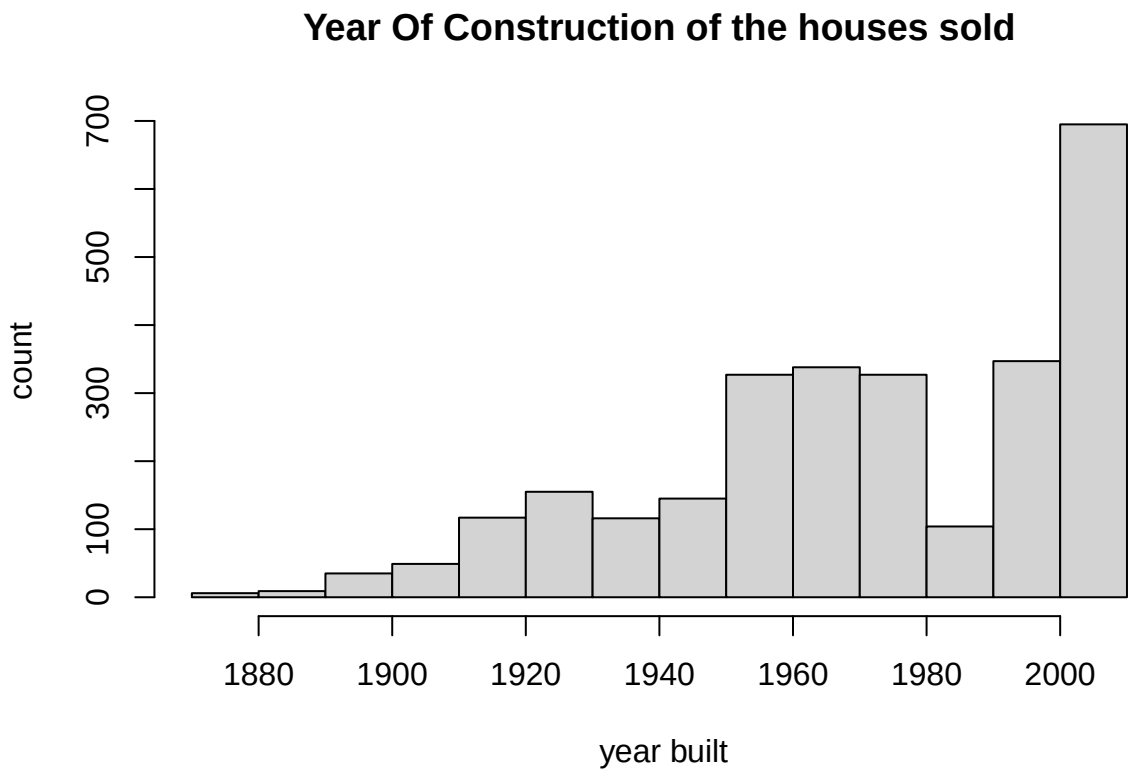
```
options(scipen = 1000000)

plot ( x = housingdata$Gr.Liv.Area, y = housingdata$SalePrice,
       xlab = "Ground living area", ylab = "Sale price" )
```



The above scatter plot reveals that as ground living area increased sale price of a house also increased.

Investigating the age of the homes sold could be instructive. The great majority of houses were built between 1950 and 2010, as shown in the figure below. A significant drop occurs in the 1980s, which can be ascribed to fewer homes being built during that time period due to the housing market meltdown. Some of the houses were constructed over a century ago. The majority of the houses were constructed in the year 2000.

```
hist(housingdata$Year.Built, main = "Year Of Construction of the houses sold",
       xlab = " year built", ylab = "count")
```



Following this, I wanted to explore number of NA's in the dataset.

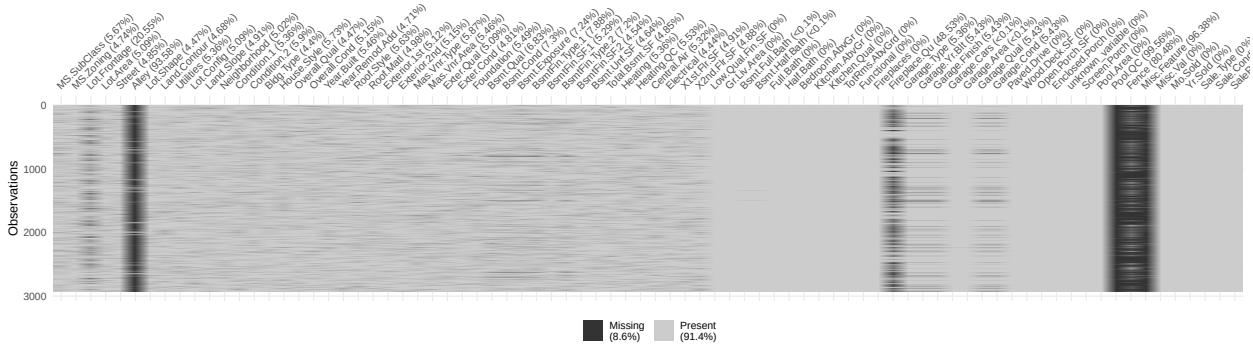
```
# Number Of NA's in data

sum(is.na(housingdata))
```

```
## [1] 20253
```

Visualising Missing Data

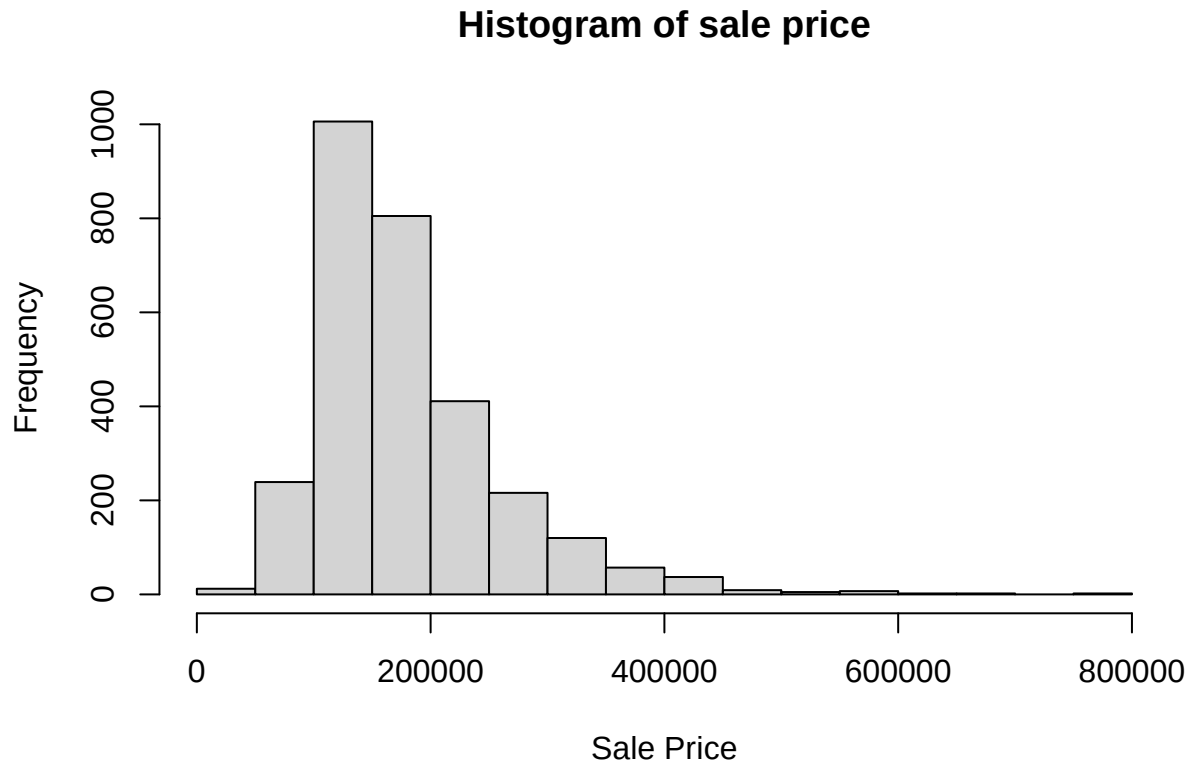
After examining the visualization of missing data, I came across many columns which have missing values for which I will use bag impute to impute numerical data as it will impute missing values by taking into consideration all actual values, which will be a good estimate for imputing missing values. And, for categorical data will do mode imputation, which will fill missing values according to most often occurring categorical values. For label encoding, will choose step integer which will keep my dimensions of data set the same and will convert data into a set of integers based on the levels of categorical data.

```
# visualization Of missing data

vis_missingdata <- vis_miss(housingdata)

vis_missingdata
```



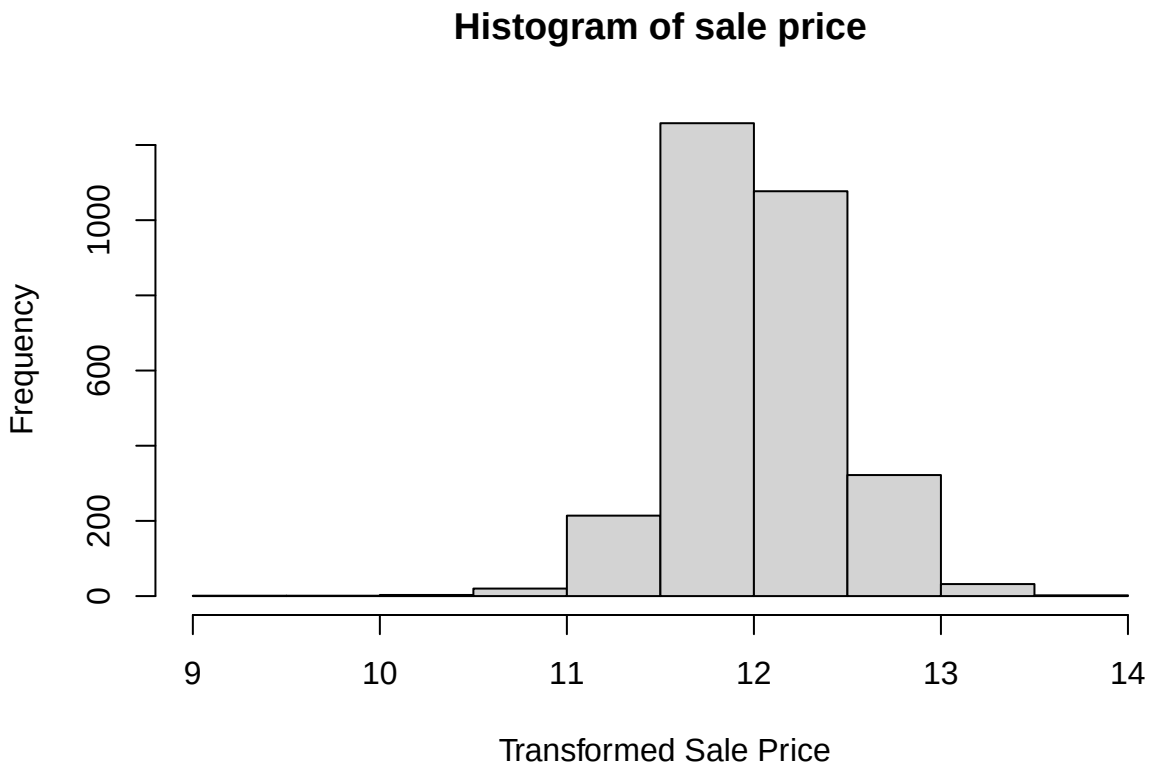
There is a need for transforming sale price as we can see from the histogram it is skewed to one side. Transformation can help in making our predictions better and can make sales price distribution normal.

```
#Visualising Sale Price

options(scipen = 100000)

hist ( housingdata$SalePrice, main = "Histogram of sale price",
       xlab = "Sale Price", ylab = "Frequency" )
```




```
#log transformation of saleprice

saleprice_transformed <- log(housingdata$SalePrice)
```

```r
#normally distributed saleprice

hist(saleprice_transformed, main = "Histogram of sale price",
       xlab = " Transformed Sale Price", ylab = "Frequency")
```



1.2 Design

```r
#Splitting data

set.seed(123)
house_split <- initial_split( data = housingdata, prop = 7/10)

#Training data

house_train <- training(house_split)

#Testing data

house_test <- testing(house_split)
```

```
#preprocessing data

house_blueprint <- recipe(SalePrice~., data = house_train) %>%

                   step_nzv(all_predictors()) %>%

                   step_impute_bag(all_numeric_predictors()) %>%

                   step_impute_mode(all_nominal_predictors()) %>%

                   step_integer(all_nominal_predictors()) %>%

                   step_log(all_outcomes()) %>%

                   step_center(all_numeric_predictors(),-Garage.Yr.Blt,-Yr.Sold,
                               -Year.Built,
                               -Year.Remod.Add) %>%

                   step_scale(all_numeric_predictors(),-Garage.Yr.Blt,-Yr.Sold,
                              -Year.Built,-Year.Remod.Add)


#filtering near zero variance

caret::nearZeroVar(house_train, saveMetrics= TRUE) %>% filter(nzv)
```

```
##                   freqRatio percentUnique zeroVar  nzv
## Street             278.00000    0.09751341   FALSE TRUE
## Land.Contour        22.01250    0.19502682   FALSE TRUE
## Utilities         1935.00000    0.14627011   FALSE TRUE
## Land.Slope          23.78205    0.14627011   FALSE TRUE
## Condition.2        189.70000    0.39005363   FALSE TRUE
## Roof.Matl          137.00000    0.24378352   FALSE TRUE
## Bsmt.Cond           20.26744    0.24378352   FALSE TRUE
## BsmtFin.Type.2      25.01493    0.29254022   FALSE TRUE
## BsmtFin.SF.2       576.00000    9.41004388   FALSE TRUE
## Heating            100.68421    0.29254022   FALSE TRUE
## Low.Qual.Fin.SF    504.50000    1.41394442   FALSE TRUE
## Kitchen.AbvGr       21.25000    0.19502682   FALSE TRUE
## Functional          39.72917    0.39005363   FALSE TRUE
## Garage.Qual         21.00000    0.24378352   FALSE TRUE
## Garage.Cond         32.56140    0.24378352   FALSE TRUE
## Enclosed.Porch     116.20000    7.36226231   FALSE TRUE
## unknown_variable   675.33333    0.97513408   FALSE TRUE
## Screen.Porch       169.18182    5.07069722   FALSE TRUE
## Pool.Area         2042.00000    0.48756704   FALSE TRUE
## Misc.Feature        24.00000    0.24378352   FALSE TRUE
## Misc.Val           141.00000    1.51145783   FALSE TRUE
```

```
prepare_house_train <- prep( x =  house_blueprint, training = house_train)

#baked data
```

```
bake_house_train <- bake( object = prepare_house_train, new_data = house_train)
```

```
cross_validation <- trainControl(method = "repeatedcv",number = 10,repeats = 5 )


grid_search <- expand.grid(k = seq(5, 15, by = 1))
```

Difference between random grid search and grid search is in random search from range of possible hyperparameter values any value of hyperparameter is randomly selected and explored but in grid search it searches across many hyperparameter values and gives the average error rate of each parameter value in order to minimize the error.

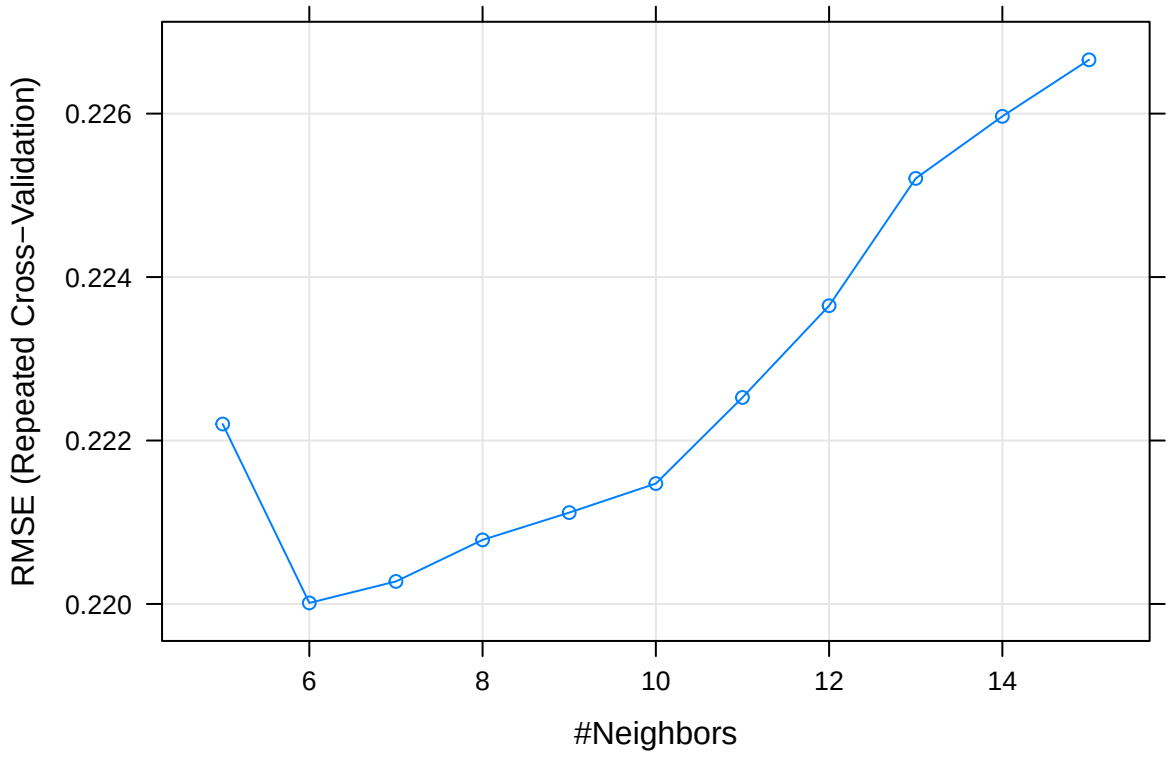
For train control function there are various resampling methods available like cv(cross-validation), repeated cv(repeated cross-validation), boot, boot632, optimism boot, boot_all, LOOCV, LGOCV, timeslice, adaptive_cv.

1.3 Fitting

```
knn_model_a1 <- train(SalePrice~.,data = bake_house_train,method = "knn",
                      trControl = cross_validation, tuneGrid = grid_search,
                      metric = "RMSE")

knn_model_a1
```

```
## k-Nearest Neighbors
##
## 2051 samples
##   58 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 1846, 1847, 1846, 1845, 1845, 1846, ...
## Resampling results across tuning parameters:
##
##   k   RMSE       Rsquared   MAE
##    5  0.2222019  0.7173450  0.1530862
##    6  0.2200129  0.7240291  0.1522371
##    7  0.2202758  0.7245016  0.1525844
##    8  0.2207843  0.7244602  0.1535720
##    9  0.2211180  0.7242053  0.1546046
##   10  0.2214733  0.7239216  0.1552495
##   11  0.2225267  0.7217620  0.1561188
##   12  0.2236496  0.7193951  0.1568093
##   13  0.2252069  0.7154917  0.1578909
##   14  0.2259667  0.7138583  0.1586296
##   15  0.2266585  0.7124865  0.1593670
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 6.
```

```
plot(knn_model_a1)
```



The optimal model is found at k = 6, which means model will make a prediction for a given observation based on the average of the response values for the 6 observations in the training data most similar to the observation being predicted.

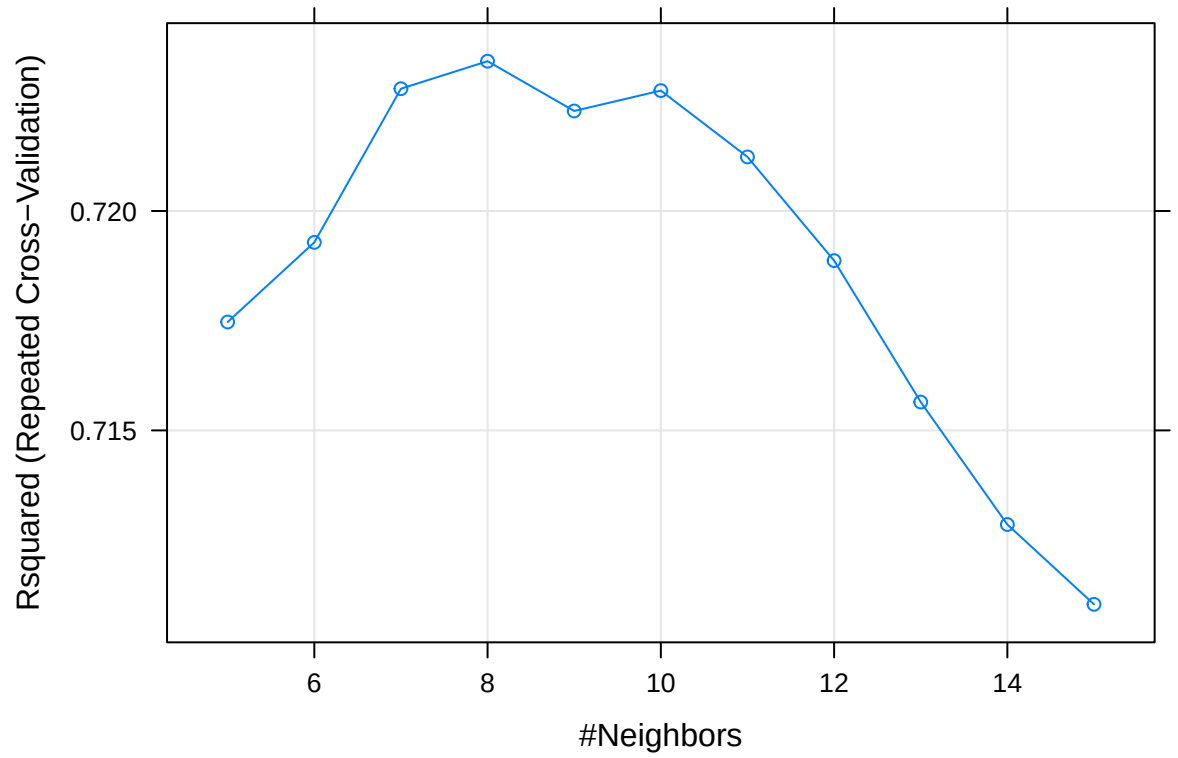
I used Root mean squared error metric and objective is to minimize it. There are many other metrics available like MSE, MAE, RMSLE, R squared. Yes, I tried other metrics as well. There wasn't any difference in the results given by these metrics all gave the same value for k except the R squared, which gave value of k as 8.

```
knn_model_a2 <- train(SalePrice~.,data = bake_house_train,method = "knn",
                      trControl = cross_validation, tuneGrid = grid_search,
                      metric = "Rsquared")

knn_model_a2
```

```
## k-Nearest Neighbors
##
## 2051 samples
##   58 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 1844, 1846, 1846, 1846, 1845, 1847, ...
## Resampling results across tuning parameters:
##
```

```
##    k   RMSE       Rsquared   MAE
##    5  0.2218175  0.7174708  0.1526072
##    6  0.2215095  0.7192850  0.1526895
##    7  0.2206802  0.7227874  0.1528513
##    8  0.2208660  0.7234151  0.1536379
##    9  0.2215732  0.7222798  0.1548181
##   10  0.2217126  0.7227445  0.1554974
##   11  0.2225774  0.7212326  0.1563284
##   12  0.2236172  0.7188695  0.1567793
##   13  0.2249674  0.7156463  0.1577259
##   14  0.2260974  0.7128546  0.1585182
##   15  0.2268874  0.7110362  0.1592865
##
## Rsquared was used to select the optimal model using the largest value.
## The final value used for the model was k = 8.
```

```
plot(knn_model_a2)
```

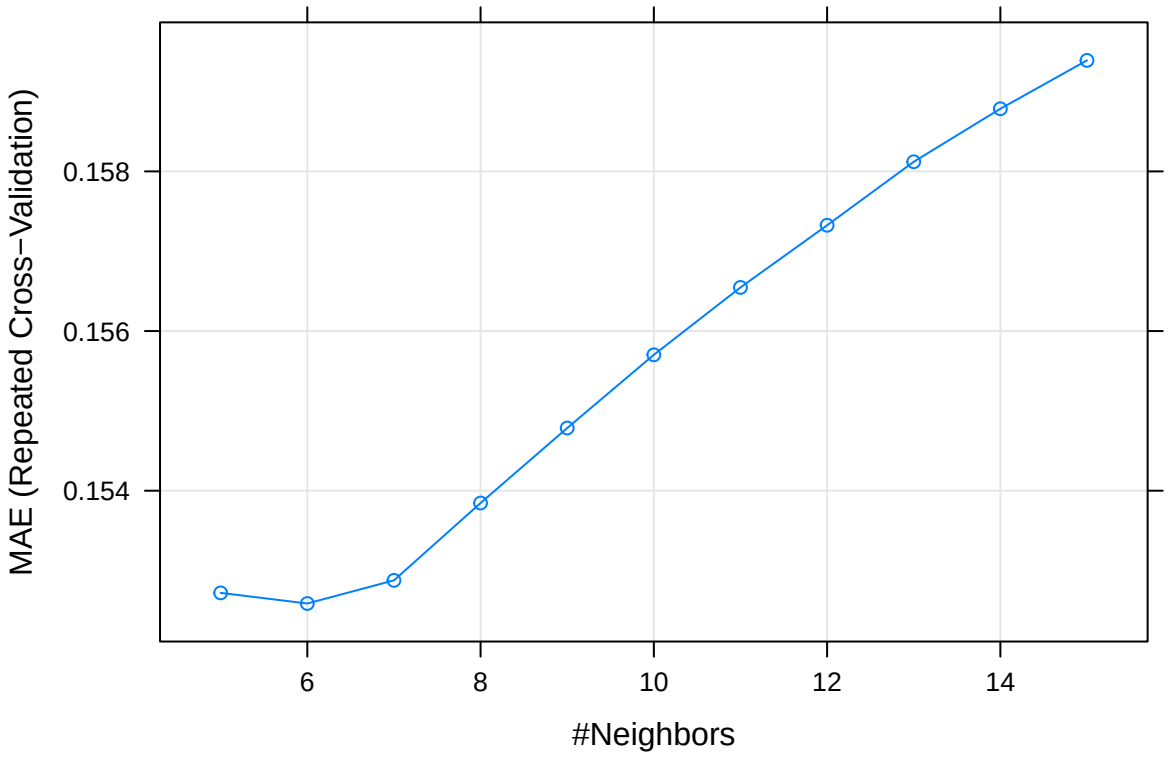



```
knn_model_a3 <- train(SalePrice~.,data = bake_house_train,method = "knn",
                      trControl = cross_validation, tuneGrid = grid_search,
                      metric = "MAE")

knn_model_a3
```

```
## k-Nearest Neighbors
```

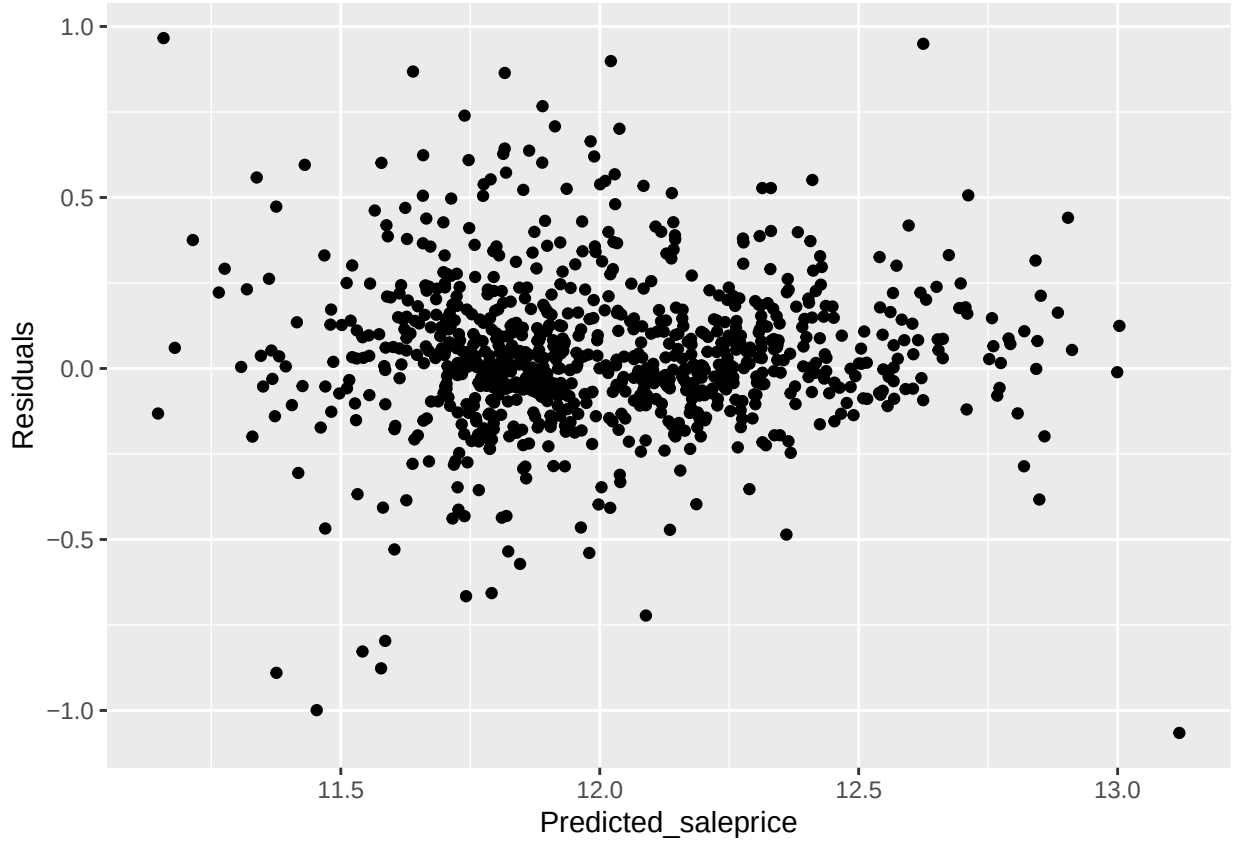
```
##
## 2051 samples
##   58 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 1847, 1844, 1847, 1846, 1845, 1846, ...
## Resampling results across tuning parameters:
##
##   k   RMSE       Rsquared   MAE
##    5  0.2210724  0.7194660  0.1527199
##    6  0.2201792  0.7230504  0.1525871
##    7  0.2203264  0.7236560  0.1528769
##    8  0.2207499  0.7236133  0.1538458
##    9  0.2211587  0.7233513  0.1547856
##   10  0.2216267  0.7228370  0.1557023
##   11  0.2226689  0.7208796  0.1565465
##   12  0.2239775  0.7178062  0.1573252
##   13  0.2252131  0.7149920  0.1581206
##   14  0.2261400  0.7129263  0.1587847
##   15  0.2267617  0.7116067  0.1593908
##
## MAE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 6.
```

```
plot(knn_model_a3)
```



1.4 Final Testing

```
bake_house_test <- bake(prepare_house_train,new_data = house_test)



#predicting sale price

Predicted_saleprice <- predict(knn_model_a1,bake_house_test)


bake_house_test <- bake_house_test %>%
                   mutate(Residuals = SalePrice - Predicted_saleprice,
                          Predicted_saleprice)

# plotting residuals

ggplot(data=bake_house_test,mapping = aes(x=Predicted_saleprice,y=Residuals)) +
         geom_point()
```




```
#summary of residuals

summary(bake_house_test$Residuals)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
```

```
## -1.06561 -0.06918  0.02736  0.04280  0.14496  0.96599
```

From the above plot, we can see that there is a random scatterness in the plot which is also known as homoscedasticity which is good for the model.

2 Graphical Exploration

Introduction

Gdp, fertlity rate, infant mortality do have an impact on global policies.Global policies are made taking into consideration these factors.

For this purpose, analysis is done on Global Economic and Social data. Various relationships between variable is explored to determine the changes needed in global polcies or not.

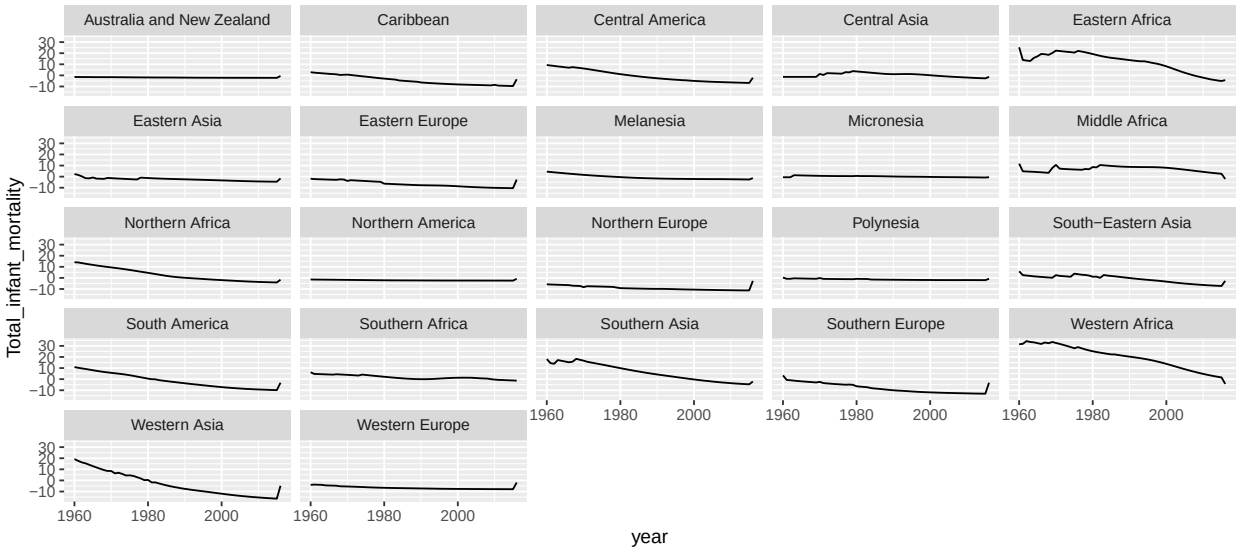
```
global_economic <- read.csv("global economic.csv")


#imputing missing values

global_economic$infant_mortality[is.na(global_economic$infant_mortality)] <-
  median(global_economic$infant_mortality,na.rm = TRUE)

global_economic$gdp[is.na(global_economic$gdp)] <-
  median(global_economic$gdp,na.rm = TRUE)

global_economic$fertility[is.na(global_economic$fertility)] <- median(global_economic$fertility,na.rm =


# scaling  numerical data

global_economic_scaled <- scale(global_economic[3:7], center=TRUE, scale=TRUE)

global_economic_scaled <- as.data.frame(global_economic_scaled )


global_economic_2 <- global_economic %>%
```

```
updated_global_economic <- global_economic_2 %>%

                           group_by(year,region) %>%

                       summarise(Total_infant_mortality = sum(infant_mortality),

                       Total_gdp = sum(gdp),Total_fertility = sum(fertility),

                       Total_life_expectency = sum(life_expectancy))
```

```
## `summarise()` has grouped output by 'year'. You can override using the `.groups` argument.
```

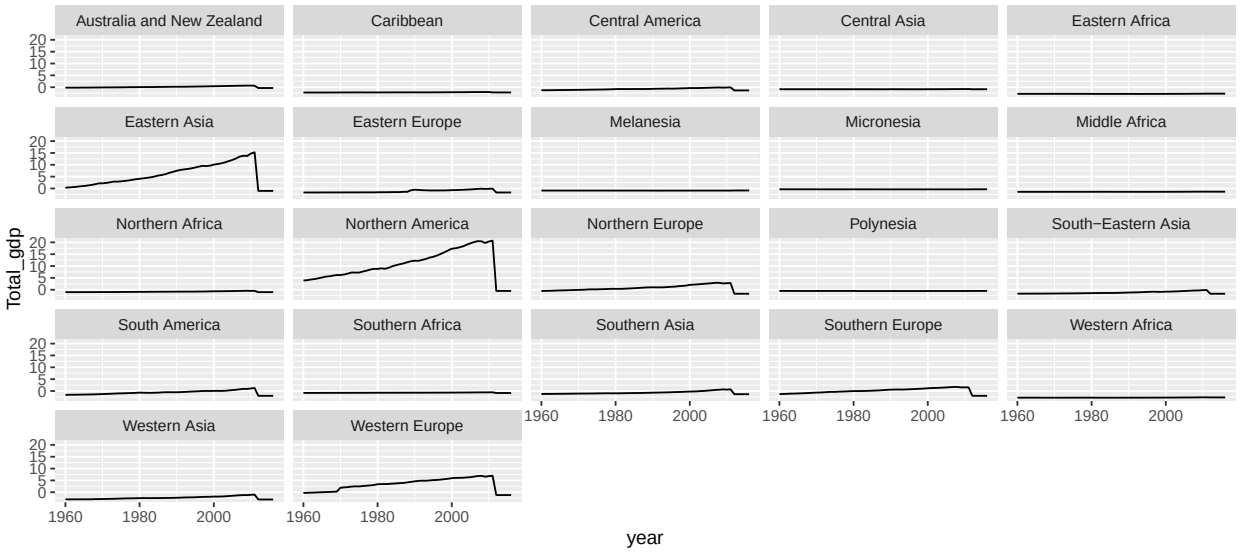
From the below given graph, we can see that year after year infant mortality keeps on getting decreasing for different regions that is Eastern Africa, South America, Southern Asia, Western Africa and Western Asia. It was stable for Australia and New Zealand, Micronesia and Polynesia.

```
visual_mortality <- ggplot( data = updated_global_economic) +
                    geom_line( mapping = aes( x = year, y = Total_infant_mortality)) +
                             facet_wrap(~region)

visual_mortality
```



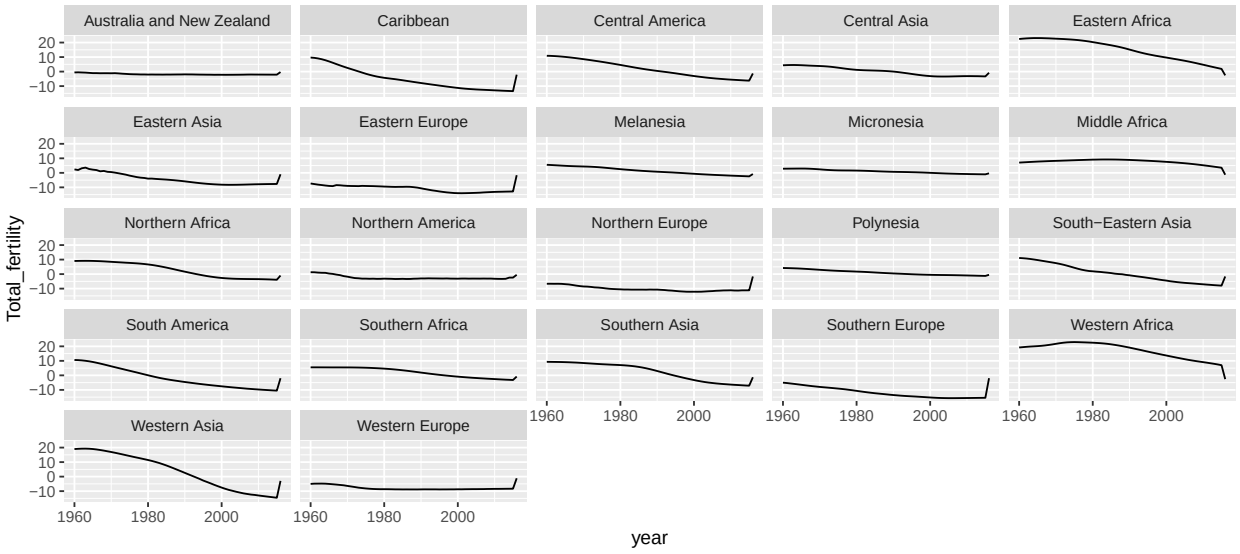
After examining this plot, we can see that as we move on gdp keeps on growing for various regions. For those regions who have stable GDP, government can spend on improving infrastructure and can provide tax cuts and rebates.

```
visual_gdp <- ggplot( data = updated_global_economic) +

  geom_line( mapping = aes( x = year, y = Total_gdp)) +

              facet_wrap(~region)
visual_gdp
```

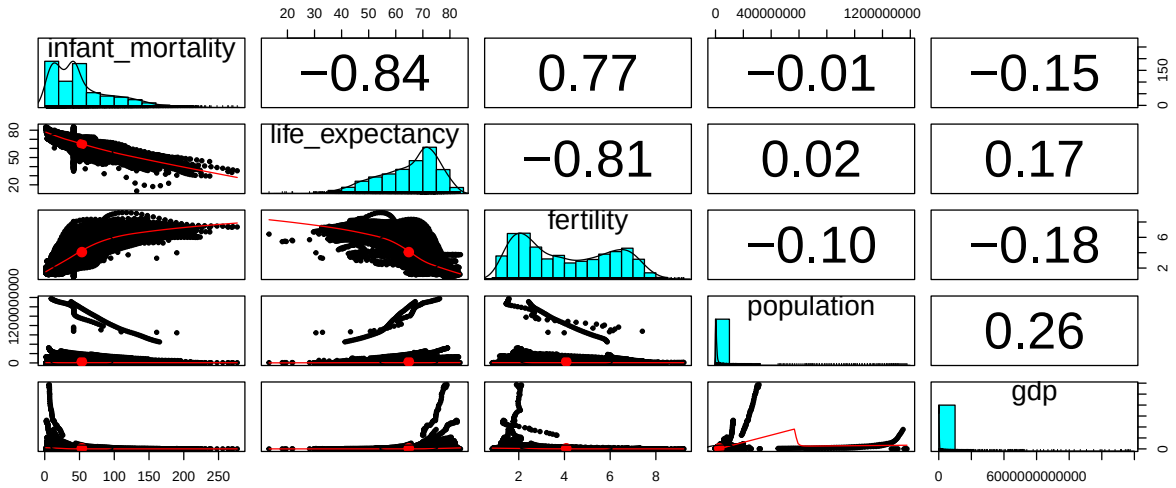


Year after year we can see there is a dip in the fertility rate. For that specific measures or policies can be made by government to increase fertility rate like baby bonuses, family allowances, maternal, paternal and parental leave, tax incentives and flexible work schedules.

```
visual_fertility <- ggplot( data = updated_global_economic  ) +

  geom_line( mapping = aes( x = year, y = Total_fertility)) +
   facet_wrap(~region)

visual_fertility
```




```
pairs.panels(global_economic[3:7])
```

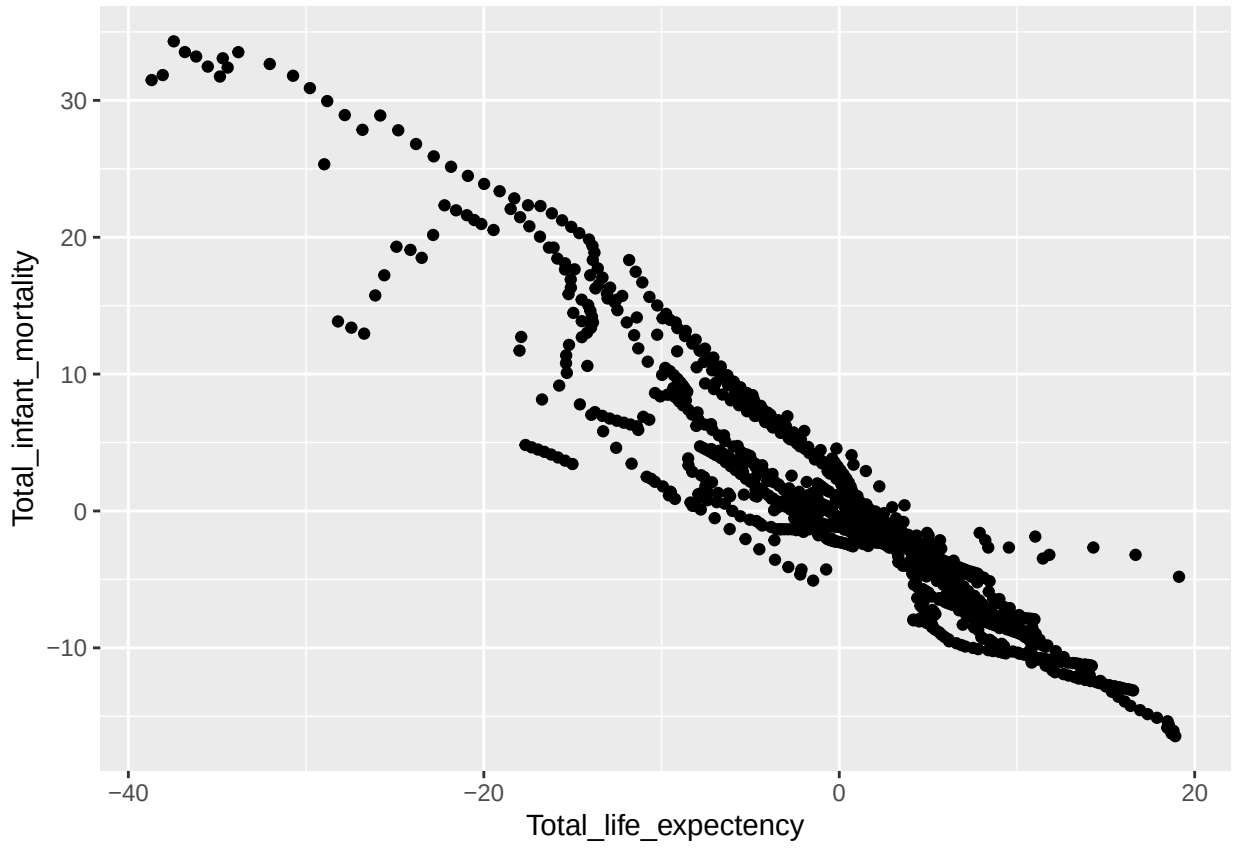


From the above diagram, we can see that there exists a negative correlation among infant mortality, fertility and life expectancy.

To have a more closer look, we can have a look at the ggplot:

As life expectency increases, there is a significant drop in infant mortality which confirms that there exists a negative relationship between them.

```
ggplot( data = updated_global_economic  ) +

  geom_point( mapping = aes( x =  Total_life_expectency, y = Total_infant_mortality))
```




```
gdp_transformed <- global_economic %>% mutate(gdp_per_capita = (gdp/population)/365 )
```

3 Linear Algebra

3.1

$$\begin{aligned} 2x + 3y &= 4 \\ x - 2y &= 3 \end{aligned}$$

Ans:

x= 17/7 y= -2/7

```
Matrix_A <- matrix(data=c(2,1,3,-2),nrow=2,ncol=2)

Matrix_A
```

```
##      [,1] [,2]
## [1,]    2    3
## [2,]    1   -2
```

```
#determinant of matrix

det(Matrix_A)
```

```
## [1] -7
```

```
#inverse

matrix_inverse <- inv(Matrix_A)

matrix_inverse
```

```
##           [,1]       [,2]
## [1,] 0.2857143  0.4285714
## [2,] 0.1428571 -0.2857143
```

```
b <- matrix(data = c(4,3),nrow = 2, ncol = 1)
b
```

```
##      [,1]
## [1,]    4
## [2,]    3
```

```
solving_matrix <- matrix_inverse %*% b

solving_matrix
```

```
##            [,1]
## [1,]  2.4285714
## [2,] -0.2857143
```

The determinant of matrix is not equal to zero which means that this matrix is invertible or this matrix transformation can be undone.

3.3

```
Matrix_B <- matrix(data=c(1,2,3,-6),nrow=2,ncol=2)

nullspace(Matrix_B)
```

```
## NULL
```

3.4

```
Matrix_B <- matrix(data=c(1,2,3,-6),nrow=2,ncol=2)

Matrix_B
```

```
##      [,1] [,2]
## [1,]    1    3
## [2,]    2   -6
```

```
x <- Matrix_A %*% Matrix_B

x
```

```
##      [,1] [,2]
## [1,]    8  -12
## [2,]   -3   15
```

```
# 2 non zero rows
rref(x)
```

```
##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1
```

```
rankMatrix(x)
```

```
## [1] 2
## attr(,"method")
## [1] "tolNorm2"
## attr(,"useGrad")
## [1] FALSE
## attr(,"tol")
## [1] 0.00000000000000004440892
```

3.5

```
eigen <- eigen(Matrix_A)

eigen
```

```
## eigen() decomposition
## $values
## [1] -2.645751  2.645751
##
## $vectors
##            [,1]      [,2]
## [1,] -0.5424768 0.9776088
## [2,]  0.8400708 0.2104307
```

```
Matrix_A %*% eigen$vectors
```

```
##           [,1]      [,2]
## [1,]  1.435259 2.5865097
## [2,] -2.222618 0.5567473
```

When we multiply matrix A with the eigen vectors it does not change it's directions but only scales it.

4 PCA using the Social Data

4.1

```
Youth_data <- read_csv("youthsocialnetworking.csv")



Youth_data$age[is.na(Youth_data$age)] <- median(Youth_data$age,na.rm = TRUE)
```

4.3

```
#removing grad year and age dependent variable

Youth_data_1 <- Youth_data[-1,-2]

scaled_youth <- scale(Youth_data_1,center = TRUE,scale = TRUE)

covarince_matrix <- cov(scaled_youth)

eigen_matrix <- eigen(covarince_matrix)
```

4.5

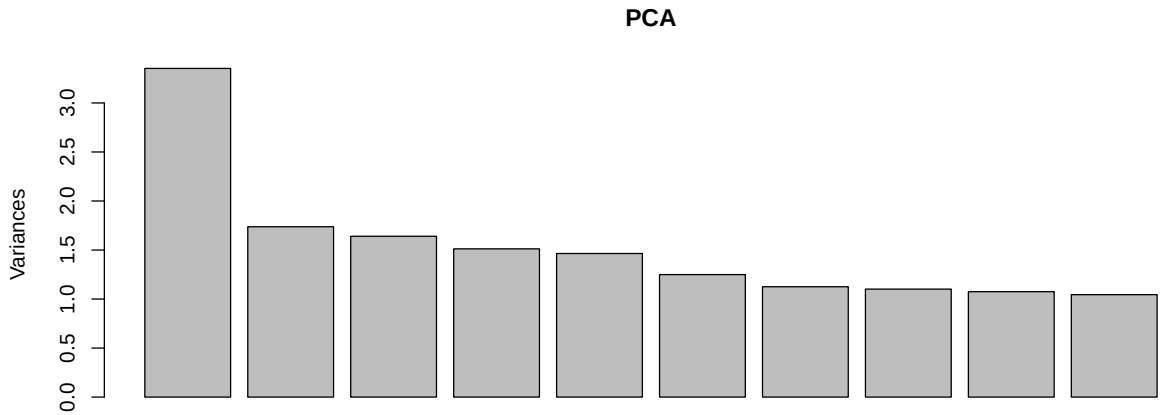
```
PCA <- prcomp(Youth_data_1, center = TRUE, scale = TRUE)



summary(PCA)
```

```
## Importance of components:
##                           PC1     PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     1.83093 1.31814 1.28077 1.22957 1.21014 1.11775 1.06083
## Proportion of Variance 0.08596 0.04455 0.04206 0.03877 0.03755 0.03203 0.02886
## Cumulative Proportion  0.08596 0.13051 0.17257 0.21133 0.24888 0.28092 0.30977
```

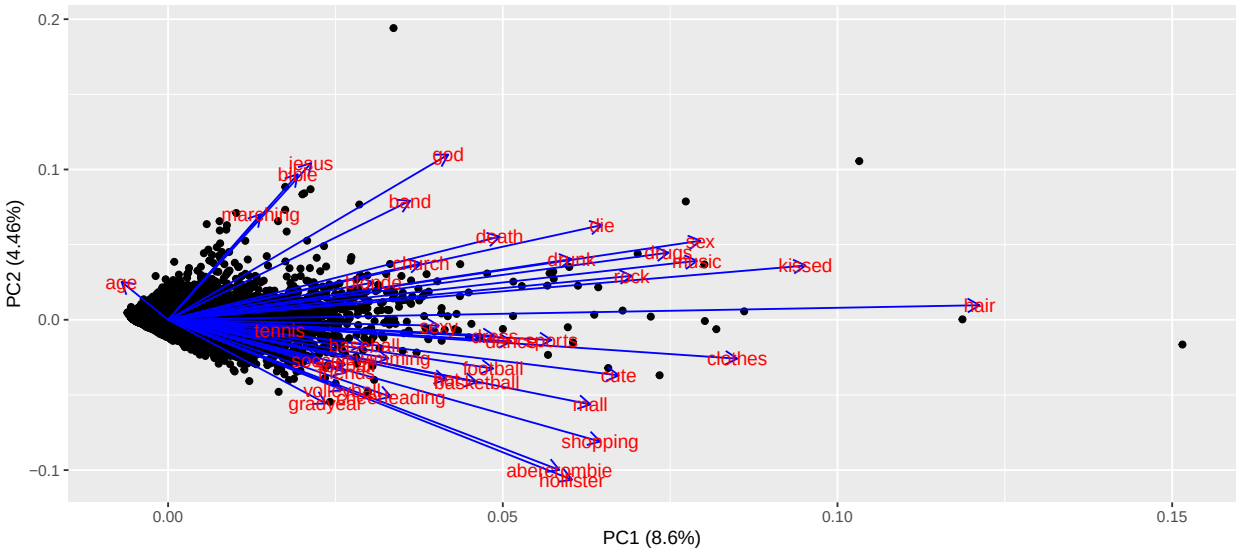
```
##                             PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     1.04945 1.03680 1.02216 1.01542 1.00626 0.99161 0.98653
## Proportion of Variance 0.02824 0.02756 0.02679 0.02644 0.02596 0.02521 0.02496
## Cumulative Proportion  0.33801 0.36558 0.39237 0.41880 0.44477 0.46998 0.49493
##                            PC15    PC16    PC17    PC18    PC19   PC20    PC21
## Standard deviation     0.98265 0.97630 0.97061 0.96936 0.95780 0.9492 0.94277
## Proportion of Variance 0.02476 0.02444 0.02416 0.02409 0.02352 0.0231 0.02279
## Cumulative Proportion  0.51969 0.54413 0.56829 0.59238 0.61591 0.6390 0.66180
##                            PC22    PC23   PC24    PC25   PC26    PC27    PC28
## Standard deviation     0.92988 0.92743 0.9220 0.91263 0.9072 0.89550 0.89242
## Proportion of Variance 0.02217 0.02205 0.0218 0.02136 0.0211 0.02056 0.02042
## Cumulative Proportion  0.68397 0.70603 0.7278 0.74918 0.7703 0.79084 0.81126
##                            PC29    PC30    PC31    PC32    PC33    PC34    PC35
## Standard deviation     0.88345 0.87833 0.86932 0.86586 0.85364 0.84388 0.80474
## Proportion of Variance 0.02001 0.01978 0.01938 0.01922 0.01868 0.01826 0.01661
## Cumulative Proportion  0.83127 0.85106 0.87043 0.88966 0.90834 0.92660 0.94321
##                            PC36    PC37   PC38    PC39
## Standard deviation     0.79750 0.75804 0.7202 0.69690
## Proportion of Variance 0.01631 0.01473 0.0133 0.01245
## Cumulative Proportion  0.95951 0.97425 0.9876 1.00000
```

4.6

```
library(ggplot2)
library(ggfortify)



screeplot(PCA, type = "barplot")
```




```
autoplot(PCA, loadings = TRUE, loadings.colour = 'blue',
         loadings.label = TRUE, loadings.label.size = 4)
```



4.7

PCA focuses on narrowing the feature space, allowing the majority of the information or variability in the data set to be explained with fewer features; in the case of PCA, these new features will also be uncorrelated. This can aid in the description of numerous aspects in our data collection, as well as the removal of multicollinearity, which can increase forecast accuracy.Naturally, the first PC (PC1) captures the greatest amount of variation, followed by PC2, PC3, and so on.

The PVE for each individual PC is displayed on a scree plot. The majority of scree plots have a similar form, starting high on the left, quickly descending, and then levelling out at some point. This is because the first component typically explains a large portion of the variability, the next few components a moderate proportion, and the final components only explain a small portion of the overall variability.