FIT5196 Assessment 1

Student Name: DISHI JAIN

Student ID: 30759307

Libraries used:

- library(psych) (for describe function)
- library(ggplot2) (for plotting graphs)
- library(reshape2) (for plotting graphs)
- library(gridExtra) (for plotting graphs on a grid)
- library(GGally) (for ggpairs function)
- library(lattice) (for levelplot)
- library(glmnet) (for glmnet in lasso and ridge)
- library(car) (for outlierTest)

1. Libraries Loaded

In [ ]:
```
library(psych)
library(ggplot2)
library(reshape2)
library(gridExtra)
library(GGally)
library(lattice)
library(glmnet)
library(car)
library(leaps)
```

2. DATA EXPLORATION

- Reading datasets

In [ ]:
```
#reading datasets
data_test <- read.csv("test.csv",header=T)
data_train <- read.csv("train.csv",header=T)
```

- Finding dimensions of test dataset

In [ ]:
```
#finding dimesnons of datasets
dim(data_test)
```

This tells us that the test data has 1752 records and 14 columns)

- Finding dimensions of train dataset

In [ ]:
```
dim(data_train)
```

This tells us that the trainng data has 7008 records and 14 columns)

- Taking a look at top rows of test dataset

In [ ]:
```
head(data_test)
```

- Taking a look at top rows of train dataset

In [ ]:
```
head(data_train)
```

In [ ]:
```
#combining the two datasets to get one whole dataset
data <- rbind(data_train, data_test) # join
```

Hence, we get a new dataset called data that is a combination of training and test data

- Taking a look at top rows of entire dataset

In [ ]:
```
head(data)
```

- Finding dimensions of entire dataset

In [ ]:
```
dim(data)
```

This tells us that the data has 8760 records and 14 columns)

In [ ]:
```
str(data)
```

It is known to us that there are no null values present in our dataset.

From the str() we can see each column's type and the kind of values it contains. We know the following things about the given columns -

Date : year-month-day

Rented Bike count : Count of bikes rented at each hour, the response variable

Hour : Hour of the day

Temperature : Temperature in Celsius

Humidity : in %

Windspeed : m/s

Visibility : 10m

Dew point temperature : Celsius

Solar radiation : MJ/m2

Rainfall : mm

Snowfall : cm

Seasons : Winter, Spring, Summer, Autumn

Holiday : Holiday/No holiday, indicating if the corresponding date is a public holiday or not

Functional Day : NoFunc(Non Functional Hours), Fun(Functional hours), the variable indicates the days when the rental bike system does not operate.

Also known to us is that our target variable is Rented.Bike.Count

We need to find how the other columns/predictors affect this target variable

Now to find some interesting observations in our dataset, we will use the summay() to get matrices like max, min, mean, median, etc.

In [ ]:
```
summary(data)
```

To get a more detailed analysis for each column we'll use the describe() on the data

In [ ]:
```
round(describe(data,3))
```

- We see that Rented.Bike.Count and Visibility have very high ranges as compared to the other predictors
- We see that Temperature has a minimum value of -18 degree celcius and a maximum value of 39 degree celcius. Hence we can say that even at such extreme temperatures, bikes are rented.
- We can see that Rented.Bike.Count has the maximum standard deviation and even the maximum standard error.
- We can see that Visibility has the maximum mean and median
- We can see that * indicates the categorical/factor variables which are the Dates, Seasons, Holiday, Functioning.Day
- Mean Absolute Deviation tells us how far “on average” the observations are from the mean. The maximum MAD is observed for Rented.Bike.Count
- Standard error of the mean is seen maximum for Rented.Bike.Count

Now generating boxplot for the entire dataset

In [ ]:
```
boxplot(data, las=2, cex.axis = 1)
```

As above doesn't display nicely, so we plot a boxplot for each column

In [ ]:
```
# ggplot to create graphs

melted_data <- melt(as.data.frame(data[,-2]))
ggplot(melted_data,aes(x = variable,y = value)) +
  facet_wrap(~variable) +
  geom_boxplot()+
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

From the above figure we get to see that Visibility has a very high range and hence only it's boxplot is clearly visible above. The rest of the columns do not have much outliers

However for the categorical coluns we do not generate a boxplot

Next we generate graphs for each column in respect with each other column. This will help us to see correlation relationships between each column/predictor. This will help us to determine how each column is related to one another.

In [ ]:
```
#to determine corelation relationship between all columns
pairs(data, panel = panel.smooth,cex = 0.2)
```

From the above figure we can see that Rented.Bike.Count has a positive corelation with Temperature and Visibility. Temperature and Snowfall have a negative corelation while Temperature and Dew.point.temperature have a positive corelation. Humidity has a negative corelation with Visibility and Solar Radiation while it has a positive corelation with Dew.point.temperature

Visibility and Humidity show a negative corelation while visiblity with wind speed shows a positive corelation. We can also see that hour and solar radiation do not show any corelation as can be seen by the horizontal line.

Now plotting histograms for each predictor to see how they are distributed. Histograms will also help to find the skewness in the data

In [ ]:
```
#plotting histograms for each column to see its distribution
p1<-ggplot(aes(x=Hour), data =  data) +
    geom_histogram(color = I('black'), fill = "red") +
    ggtitle('Hour wise distribution')

p2<-ggplot(aes(x=Temperature), data =  data) +
    geom_histogram(color = I('black'), fill = "red") +
    ggtitle('Temperature wise distribution')

p3<-ggplot(aes(x=Humidity), data =  data) +
    geom_histogram(color = I('black'), fill = "red") +
    ggtitle('Humidity distribution')

p4<-ggplot(aes(x=Wind.speed), data =  data) +
    geom_histogram(color = I('black'), fill = "red") +
    ggtitle('Wind speed distribution')

p5<-ggplot(aes(x=Visibility), data =  data) +
    geom_histogram(color = I('black'), fill = "red") +
    ggtitle('Visibility distribution')

p6<-ggplot(aes(x=Dew.point.temperature), data =  data) +
    geom_histogram(color = I('black'), fill = "red") +
    ggtitle('Dew point temperature  distribution')

p7<-ggplot(aes(x=Solar.Radiation), data =  data) +
    geom_histogram(color = I('black'), fill = "red") +
    ggtitle('Solar radiation distribution')

p8<-ggplot(aes(x=Rainfall), data =  data) +
    geom_histogram(color = I('black'), fill = "red") +
    ggtitle('Rainfall distribution')

p9<-ggplot(aes(x=Snowfall), data =  data) +
    geom_histogram(color = I('black'), fill = "red") +
    ggtitle('Snowfall distribution')


# plot all 9, 3 x 3

grid.arrange(p1, p2, p3, p4, p5, p6, p7, p8, p9, ncol = 3)
```

From the above histograms we can see that-

- Hour is almost evenly distributed.
- Temperature doesn't exhibit any skewness hence we can say that it is normally distributed.
- Humidity has almost an equal distribution of its data.
- Wind speed is right skewed hence we can say that with right-skewed distribution (also known as "positively skewed" distribution), most data falls to the right, or positive side, of the graph's peak. Thus, the histogram skews in such a way that its right side (or "tail") is longer than its left side.
- Visibility is left skewed hence most of the data points lie on the left side
- Dew point temperature is almost normally skewed.
- Rest of the columns i.e. Solar radiation, Rainfall and Snowfall are right skewed graphs.

Now to see how the categorical variables are playing a part we will plot bar charts

To see how each season is contributing date wise we will plot a bar chart that shows the count of each season occurence

In [ ]:
```
#count of rows for seasons
season_plot_count <- ggplot(data, aes(x = as.factor(Seasons))) + geom_bar(color='
geom_text(stat='count',aes(label=..count..),vjust=-1) +
ggtitle("Count of date wise Rented Bikes by Seasons") +
xlab("Seasons") + ylab("Count of date wise Rented Bikes")
#Show the plot
season_plot_count
```

From the above bar chart we see that in all the season i.e. Autumn, Spring, Summer, Winter, the bikes are rented almost in the sme frequency

Now lets plot a bar chart that displays sum of bikes rented for each season

In [ ]:
```
#sum of rented bikes for seasons
sum_by_season <- aggregate(Rented.Bike.Count ~ Seasons, data = data, FUN = sum)
seasons_plot_sum <- ggplot(sum_by_season, aes(x = Seasons, y = Rented.Bike.Count)
                geom_bar(stat = 'identity',color="black",fill="steelblue") + ggti
                xlab("Seasons") + ylab("Bikes rented overall")
#Show the plot
seasons_plot_sum
```

From the above bar chart we can see that the most bikes are rented in the month of Summer. Hence we can say that during summers there is a higher chance of bike renting. Also from the above bar chart we can see that during Winters the least number of bikes are rented. This observation can be used by the bike sharing system to accordingly arrange for bikes during the winter season.

Now, to see how the Holiday column is contributing date wise we will plot a bar chart that shows the count of each Holiday value occurence

In [ ]:
```
#count of rows for holidays
holiday_plot_count <- ggplot(data, aes(x = as.factor(Holiday))) + geom_bar(color=
geom_text(stat='count',aes(label=..count..),vjust=-1) +
ggtitle("Count of date wise Rented Bikes by Holiday/Not Holiday") +
xlab("Holiday or No Holiday") + ylab("Count of date wise Rented Bikes")
#Show the plot
holiday_plot_count
```

From the above bar chart we see that during public Holidays we see a very less count of bikes rented and during normal days i.e. not a public holiday there is a huge count in bike rentals date wise.

Now lets plot a bar chart that displays sum of bikes rented for each Holiday value

In [ ]:
```
#sum of rented bikes for holidays
sum_by_holiday <- aggregate(Rented.Bike.Count ~ Holiday, data = data, FUN = sum)
holiday_plot_sum <- ggplot(sum_by_holiday, aes(x = Holiday, y = Rented.Bike.Count
                geom_bar(stat = 'identity',color="black",fill="green4") + ggtitle
                xlab("Holiday") + ylab("Bikes rented overall")
#Show the plot
holiday_plot_sum
```

From the above bar chart we can see that the most bikes are rented during normal days i.e. not a public holiday. Hence we can say that during normal days there is a higher chance of bike renting. Also from the above bar chart we can see that during holidays the least number of bikes are rented. This observation can be used by the bike sharing system to accordingly arrange for bikes during the holidays season.

Now, to see how the Functioning.Day column is contributing date wise we will plot a bar chart that shows the count of each FunctioningDay value occurence

In [ ]:
```
#count of rows for functioning day yes or no
functioningday_plot_count <- ggplot(data, aes(x = as.factor(Functioning.Day))) +
geom_text(stat='count',aes(label=..count..),vjust=-1) +
ggtitle("Count of date wise Rented Bikes by Functioning Day") +
xlab("Functioning Day Yes or No") + ylab("Count of date wise rented bikes")
#Show the plot
functioningday_plot_count
```

From the above bar chart we see that when the bike sharing system is not functioning we see a very less number of records in the data. This does not indicate tht bikes were rented or not during these dates.

Now lets plot a bar chart that displays sum of bikes rented for each Functioning Day value

In [ ]:
```
#sum of rented bikes for holidays

sum_by_functioning.day <- aggregate(Rented.Bike.Count ~ Functioning.Day, data = d
functioning.day_plot_sum <- ggplot(sum_by_functioning.day, aes(x = Functioning.Da
                geom_bar(stat = 'identity',color="black",fill="yellow") + ggtitle
                xlab("Functioning Day") + ylab("Bikes rented overall")
#Show the plot
functioning.day_plot_sum
```

From the above bar chart we can see that the bikes are rented during functining days only. This chart can be used to detect is there are any defaults in the data. However as 0 bikes are rented in Not Functioing days hence we can see that the data is correct.

Hence when bike systesm functionng is No, then sum of rented bikes is 0 as no bikes are rented

To see a much more detailed relationship between the variables, we can do the followin

In [ ]:
```
ggpairs(data[-1])
```

To get the correlation as numbers we plot a levelplot on our data

In [ ]:
```
#Define you own panel
myPanel <- function(x, y, z, ...) {
    panel.levelplot(x,y,z,...)
    panel.text(x, y, round(z, 2))
}
#Define the color scheme
cols = colorRampPalette(c("red","blue"))
#Plot the correlation matrix.
levelplot(cor(data[2:11]), col.regions = cols(100), main = "correlation", xlab =
          scales = list(x = list(rot = 90)), panel = myPanel)
```

From the above figure, we can clearly see positive and negative correlations between the columns. A positive number indicates a positive corelation whereas a negative number indicates a negative corelation

Now to see how each variable is distributed we will plot the histogram of each variable. However we should also compare its distribution with the distribution of its sqrt and log.

Hence now for each column, we will plot the histogram of the column, the histogram of square root of column and the log of the column. This will help us to determine the distribution and hence will help in a better selection of the variable

In [ ]:
```
#variations of rented bikes
par(mfrow = c(2,2))
hist(data$Rented.Bike.Count, xlab = "Rented.Bike.Count", main="Histogram of Rente
hist(log(data$Rented.Bike.Count), xlab = "Rented.Bike.Count", main="Histogram of
hist(sqrt(data$Rented.Bike.Count), xlab = "Rented.Bike.Count", main="Histogram of
```

From the above figure we can see that sqrt(Rented.Bike.Count) is the most normally distributed

In [ ]:
```
#variations of Hour
par(mfrow = c(2,2))
hist(data$Hour, xlab = "Hour", main="Histogram of Hour")
hist(log(data$Hour), xlab = "Hour", main="Histogram of log(Hour)")
hist(sqrt(data$Hour), xlab = "Hour", main="Histogram of sqrt(Hour)")
```

From the above figure we can see that the hour column itself is the most normally distributed

In [ ]:
```
#variations of Temperature
par(mfrow = c(2,2))
hist(data$Temperature, xlab = "Temperature", main="Histogram of Temperature")
hist(log(data$Temperature), xlab = "Temperature", main="Histogram of log(Temperat
hist(sqrt(data$Temperature), xlab = "Temperature", main="Histogram of sqrt(Temper
```

From the above figure we can see that the temperature column itself is the most normally distributed

In [ ]:
```
#variations of Humidity
par(mfrow = c(2,2))
hist(data$Humidity, xlab = "Humidity", main="Histogram of Humidity")
hist(log(data$Humidity), xlab = "Humidity", main="Histogram of log(Humidity)")
hist(sqrt(data$Humidity), xlab = "Humidity", main="Histogram of sqrt(Humidity)")
```

From the above figure we can see that the humidity column itself is the most normally distributed

In [ ]:
```
#variations of Wind.speed
par(mfrow = c(2,2))
hist(data$Wind.speed, xlab = "Wind.speed", main="Histogram of Wind.speed")
hist(log(data$Wind.speed), xlab = "Wind.speed", main="Histogram of log(Wind.speed
hist(sqrt(data$Wind.speed), xlab = "Wind.speed", main="Histogram of sqrt(Wind.spe
```

From the above figure we can see that the sqrt(Wind.speed) is the most normally distributed

In [ ]:
```
#variations of Visibility
par(mfrow = c(2,2))
hist(data$Visibility, xlab = "Visibility", main="Histogram of Visibility")
hist(log(data$Visibility), xlab = "Visibility", main="Histogram of log(Visibility
hist(sqrt(data$Visibility), xlab = "Visibility", main="Histogram of sqrt(Visibili
```

From the above figure we can see that the visibility column itself is the most normally distributed

In [ ]:
```
#variations of Dew.point.temperature
par(mfrow = c(2,2))
hist(data$Dew.point.temperature, xlab = "Dew.point.temperature", main="Histogram
hist(log(data$Dew.point.temperature), xlab = "Dew.point.temperature", main="Histo
hist(sqrt(data$Dew.point.temperature), xlab = "Dew.point.temperature", main="Hist
```

From the above figure we can see that the dew point temperature column itself is the most normally distributed

In [ ]:
```
#variations of Solar.Radiation
par(mfrow = c(2,2))
hist(data$Solar.Radiation, xlab = "Solar.Radiation", main="Histogram of Solar.Rad
hist(log(data$Solar.Radiation), xlab = "Solar.Radiation", main="Histogram of log(
hist(sqrt(data$Solar.Radiation), xlab = "Solar.Radiation", main="Histogram of sqr
```

From the above figure we can see that the solar radiation column itself is the most normally distributed

In [ ]:
```
#variations of Rainfall
par(mfrow = c(2,2))
hist(data$Rainfall, xlab = "Rainfall", main="Histogram of Rainfall")
hist(log(data$Rainfall), xlab = "Rainfall", main="Histogram of log(Rainfall)")
hist(sqrt(data$Rainfall), xlab = "Rainfall", main="Histogram of sqrt(Rainfall)")
```

From the above figure we can see that the rainfall column itself is the considerably better

In [ ]:
```
#variations of Snowfall
par(mfrow = c(2,2))
hist(data$Snowfall, xlab = "Snowfall", main="Histogram of Snowfall")
hist(log(data$Snowfall), xlab = "Snowfall", main="Histogram of log(Snowfall)")
hist(sqrt(data$Snowfall), xlab = "Snowfall", main="Histogram of sqrt(Snowfall)")
```

From the above figure we can see that log(Snowfall) is the most normally distributed

3. Model Selection

Now we need to generate models that will help us in predicting the value of Rented Bike Count. We can use different models like Linear Regression, Lasso, Ridge or Random Forest. However for this exercise we need to generate only two models. Hence we will create two models i.e. Linear Regression Model and Lasso Model. At the end we will compare the two models based on the error values in prediction i.e. the mean square error. The model which will give the least error will be considered as the best one. Also we generate the models on our training dataset and then use the generated model to predict values from the test dataset.

a) Linear Regression

A Linear Regression Model is created when the target variable is a numerical value. In our case the target variable is the Rented Bike Count which is numerical in nature. Hence we will plot a linear regression model and determine which predictors highly influence the target variable.

To plot the linear regression models, we can first plot the model which will contain all the original predictors. This is represented by . while creating lm

Hence a linear regression model is created below with target variable as Rented.Bike.Count and all the other columns as predictors. We will not give the Date column as a predictor here.

Next to see the statistics of the mode, we will use the summary()

In [ ]:
```
my_fit = lm(Rented.Bike.Count ~ ., data = data_train[,-1])
summary(my_fit)
```

The output contains: residuals, coefficients, residual standard error, R^2, and F-statistic. These are the statistics that we use to assess the accuracy of the model.

Taking significance of 0.05 we can see the p values in the summary to see which predictors are the most significant to the target variable. Here we see that Visibility has a very less significance to the target variable and hence we can exclude it from the model.The lower the p value the better the predictor.

We also saw from the above histograms created in Data Exploration, that sqrt(Wind.speed) and log(Snowfall + 1) had a much more normal distribution as compared to their respective columns indvidually. Hence we can also include these predictors in generating the new lm model.

In [ ]:
```
my_fit2 = lm(Rented.Bike.Count ~ . + sqrt(Wind.speed) + log(Snowfall+1), data = d
summary(my_fit2)
```

In the model above we see that p value of many predictors like Visibility, Dew.point.temperature and Snowfall is slightly high. Hence we can perform analysis on the interaction terms for these column to see whether they increase significance on the target variable.

Also we see that for this model the Adjusted R square value increases which means the model is good. Along with this we also see a decrease in the Residual standard error. Hence including sqrt(Wind.speed) and log(Snowfall + 1) have increased the accuracy of the model.

Now we will create another model which excludes the Visibility and Date columns but also include the sqrt(Wind.speed) and log(Snowfall + 1) predictors.

In [ ]:
```
my_fit3 = lm(Rented.Bike.Count ~ . + sqrt(Wind.speed) + log(Snowfall+1), data = s
summary(my_fit3)
```

In the model above we see that p value of many predictors like Visibility, Dew.point.temperature and Snowfall is still very high. Hence we can perform analysis on the interaction terms for these column to see whether they increase significance on the target variable.

Also we see that for this model the Adjusted R square value remains the same. Along with this we also see a very small amount of increase in the Residual standard error. As result we see a very similar model from abve and hence we should apply more analysis in creating the models.

Now we will create another model which excludes the Visibility and Date columns but will not include the sqrt(Wind.speed) and log(Snowfall + 1) predictors. We will then generate its summary to see the statistical values

In [ ]:
```
my_fit4 = lm(Rented.Bike.Count ~ . , data = subset(data_train, select=c( -Date, -
summary(my_fit4)
```

In the model above we see that p value of many predictors like Dew.point.temperature and Snowfall has reduced hence they have become better predictors. Also we see that most of the predictors have a small p value and hence they show a high significance to the target variable.

Also we see that for this model the Adjusted R square value decreses slightly. Along with this we also see a very small amount of increase in the Residual standard error. Still we can not say that we have a better model, hence more analysis should be applied.

Now we will create another model which excludes the Visibility and Date columns but will not include the sqrt(Wind.speed) and log(Snowfall + 1) predictors. We have also used various other predictors as shown in below code on the basis of the p values generated in the above model. We have not used any interaction terms here yet as I wanted to see first how the model behaves using just sqrt and log predictors. We will then generate its summary to see the statistical values

In [ ]:
```
my_fit5 = lm(Rented.Bike.Count ~ . +  sqrt(Humidity) + sqrt(Wind.speed)  +
               sqrt(Solar.Radiation) + sqrt(Rainfall) + sqrt(Snowfall) +
                log(Humidity+1) + log(Wind.speed+1)  +  log(Snowfall + 1) +
              log(Solar.Radiation+1) + log(Rainfall+1)
             , data = subset(data_train, select=c( -Date , - Visibility)))
summary(my_fit5)
```

In the model above we see that p value of many predictors is less hence hence they have become better predictors. We also see some predictors with a high p value but this can be handled while creating the final lm model. However we see that that most of the predictors have a small p value and hence they show a high significance to the target variable.

Also we see that for this model the Adjusted R square value has increased significantly. Hence we can say that this model has performed better as compared to the above models. Along with this we also see a decrease in the Residual standard error. Hence the error has fallen down which means the model generated is better.

Now we will create another model which excludes the Visibility and Date columns but will not include the sqrt(Wind.speed) and log(Snowfall + 1) predictors. We have also used various other predictors as shown in below code on the basis of the p values generated in the above model. Now I have included some interaction terms based on thie corelation. So Temperature and Dew.point.temperature have a positive corelation with the target variable and hence I have generated an interacton term between these two predictor. Also Temperature and Hour has a positive corelation with the target variable and so I have generated an interaction term of the two. I also saw a negative corelation of Humidity and Rainfall with the target variable and hence I have included an interaction term of these two as well. We will then generate its summary to see the statistical values

In [ ]:
```
my_fit6 <-lm(Rented.Bike.Count ~ . +  sqrt(Wind.speed)  +  log(Snowfall + 1) +
          Temperature:Dew.point.temperature + Temperature:Hour + Humidity:Rainfal
             , data = subset(data_train, select=c( -Date , - Visibility)))
summary(my_fit6)
```

In the model above we see that p value of many predictors is less hence hence they have become better predictors. We see that that most of the predictors have a small p value and hence they show a high significance to the target variable.

Also we see that for this model the Adjusted R square value has decreases slightly and residual error has increased slightly. Hence we can still build a better model.

Now we will create another model which excludes the Visibility and Date columns but will not include the sqrt(Wind.speed) and log(Snowfall + 1) predictors. We have also used various other predictors as shown in below code on the basis of the p values generated in the above model. Now I have included some interaction terms based on thie corelation. So Temperature and Dew.point.temperature have a positive corelation with the target variable and hence I have generated an interacton term between these two predictor. Also Temperature and Hour has a positive corelation with the target variable and so I have generated an interaction term of the two. I also saw a negative corelation of Humidity and Rainfall with the target variable and hence I have included an interaction term of these two as well. I also saw a relation between the p values of certain other predictors which have been used as interaction terms in the code below while creating the lm model. Hence I have included interaction terms like Humidity:Solar.Radiation (both had p value as < 2e-16 in above model), Dew.point.temperature:Hour (as both had p values in terms of e-09) and so on. Hence the final model created looks as shown below.

We will then generate its summary to see the statistical values

In [ ]:
```
final_fit <-lm(Rented.Bike.Count ~ . +  sqrt(Wind.speed)  +  log(Snowfall + 1) +
          Temperature:Dew.point.temperature + Temperature:Hour + Humidity:Rainfal
           Humidity:Solar.Radiation + sqrt(Wind.speed):Dew.point.temperature +
          Wind.speed:Snowfall + Temperature:sqrt(Wind.speed) + Dew.point.temperat
               Humidity:Dew.point.temperature + Dew.point.temperature:Solar.Radia
             , data = subset(data_train, select=c( -Date , - Visibility)))
summary(final_fit)
```

In [ ]:
```
par(mfcol=c(2,2))
plot(final_fit, which = 1)
plot(final_fit, which = 2)
plot(final_fit, which = 3)
plot(final_fit, which = 4)
```

The diagnostic plots show residuals in four different ways.

1. The *residual vs fitted plot*: This plot is used to check the linear assumption. If there is an equal distribution around the horizontal line then we say that there is a linear relationship between

the predictor and target variable. However if the data points are not spread evenly we say there is a non linear relationship. Here we see there is a non linear relationship.

- The normal *Q-Q plot*: The Q-Q plot (i.e., quantile-quantile plot) is a graphical tool which helps us assess if a set of data plausibly came from some theoretical distribution such as a Normal. The plot above will show whether the data points are distributed normally or not. However we see that the residuals are not lying entirely on the straight line.
- The *scale-location plot*: It is used to check if the residuals are lyinh on the horizontal line or not. This is used to check if they are residuals are spread randomly or not. We see that the residuals are spread randomly.

- The *residual-leverage plot*: it helps us identify influential data samples. We will have to check whether the outliers detected are influential points or not. Hence we need to focus only on the influential points as they will be useful while creating the model. In the residual-leverage plot, we look for outlying values at the upper right corner or at the lower right corner. When samples are outside of the Cook's distance the samples are influential to the regression results.

In [ ]:
```
outlierTest(final_fit, cutoff=0.05, digits = 1)
```

Hence 849,4777,2245 are not outliers

Next I have used the step function on the final created model to see if there will be any increased significance.

In [ ]:
```
step_final <- step(final_fit)
summary(step_final)
```

The above model has performed 2 iterations until it achieves a low AIC value. After the secnd iteration we see that AIC value was not reducing by a high margin and hence the step function has stopped. Here we see that the residual error and adjusted R square values are similar as compared to the final_fit model. Hence we can use the final_fit model as the final lm model.

Now we use the anova() to compare the most suitable models. This will help us to compare the p values as the F values of the models.

From the anova table we see that for the final_fit model the p value has reduced which hence shows that it hs a high significance towards the target variable as compared to the rest of the models. Also considering the F value we see that the F value is best for our final model hence we cn say that out of all the models generated, the final_fit is the best one

In [ ]:
```
anova(my_fit5,my_fit6,final_fit)
```

Now to predict the values of the target variable by using our final_fit model we will use the predict()

We will give the final model and the test dat to generate the predicted values. We will then subtract these predicted values from the actual values and then calculate the mean of their squares to finally get the Mean Square Error (MSE). The lower the MSE the better the model was in prediction tasks.

In [ ]:
```
mean((predict(final_fit,data_test) - data_test$Rented.Bike.Count)^2)
```

Regularization and reg subset selection

Now we will do regularizaton and subset selection known as stepwise selection, which explore a far more restricted set of models. The two techniques used are -

- Forward stepwise selection : Starts with one-variable models, gradually add one variable, end with a model including all the specified variables.
- Backward stepwise selection : Starts with a full model, gradually exclude one variable, end with one-variable models.

In [ ]:
```
# regfit.full <- regsubsets(Rented.Bike.Count ~ . , data = data[-1], nvmax = 15)
```

In [ ]:
```
reg_forward <- regsubsets(Rented.Bike.Count ~ ., data = data[-1], nvmax = 15, met
reg_forward_summary <- summary(reg_forward)
```

In [ ]:
```
par(mfrow = c(2, 2))
plot(reg_forward_summary$cp, xlab = "Number of variables", ylab = "C_p", type = '
points(which.min(reg_forward_summary$cp), reg_forward_summary$cp[which.min(reg_fo
plot(reg_forward_summary$bic, xlab = "Number of variables", ylab = "BIC", type =
points(which.min(reg_forward_summary$bic), reg_forward_summary$bic[which.min(reg_
plot(reg_forward_summary$adjr2, xlab = "Number of variables", ylab = "Adjusted R^
points(which.max(reg_forward_summary$adjr2), reg_forward_summary$adjr2[which.max(
plot(reg_forward_summary$rss, xlab = "Number of variables", ylab = "RSS", type =
mtext("Plots of C_p, BIC, adjusted R^2 and RSS for forward stepwise selection", s
```

In [ ]:
```
reg_backward <- regsubsets(Rented.Bike.Count ~ ., data = data[-1], nvmax = 15, me
reg_backward_summary <- summary(reg_backward)
```

In [ ]:
```
par(mfrow = c(2, 2))
plot(reg_backward_summary$cp, xlab = "Number of variables", ylab = "C_p", type =
points(which.min(reg_backward_summary$cp), reg_backward_summary$cp[which.min(reg_
plot(reg_backward_summary$bic, xlab = "Number of variables", ylab = "BIC", type =
points(which.min(reg_backward_summary$bic), reg_backward_summary$bic[which.min(re
plot(reg_backward_summary$adjr2, xlab = "Number of variables", ylab = "Adjusted R
points(which.max(reg_backward_summary$adjr2), reg_backward_summary$adjr2[which.ma
plot(reg_backward_summary$rss, xlab = "Number of variables", ylab = "RSS", type =
mtext("Plots of C_p, BIC, adjusted R^2 and RSS for backward stepwise selection",
```

b) Lasso

In the lasso model creation we will first generate the model on all the original predictors as shown below.

The Lasso approach is used in Machine Learning process to do feature selection automatically. Depending on the shrinkage parameter, Lasso regularizes the coefficient in a way such that the estimated coefficients can be shrunk toward zero. Here we will use glmnet().

In [ ]:
```
#parameters original
train.mat <- model.matrix(Rented.Bike.Count ~ . , data = data_train[-1])[,-1]
test.mat <- model.matrix(Rented.Bike.Count ~ . , data = data_test[-1])[,-1]
grid <- 10^seq(4, -2, length = 100)
```

In [ ]:
```
fit.lasso <- glmnet(train.mat, data_train$Rented.Bike.Count, alpha = 1, lambda =
cv.lasso <- cv.glmnet(train.mat, data_train$Rented.Bike.Count, alpha = 1, lambda
bestlam.lasso <- cv.lasso$lambda.min
bestlam.lasso
```

Hence the best lambda value for lasso generated is approximately 0.06136

In [ ]:
```
pred.lasso <- predict(fit.lasso, s = bestlam.lasso, newx = test.mat)
mean((pred.lasso - data_test$Rented.Bike.Count)^2)
```

Hence the MSE when Lasso is used on all original parameters comes out to be 183220.517 approximately. This is high as compared to the lm model.

Now we will generate a Lasso model based on the parameters used in final_fit lm model.

In [ ]:
```
#parameters of final_fit
train.mat <- model.matrix(Rented.Bike.Count ~ . +  sqrt(Wind.speed)  +  log(Snowf
          Temperature:Dew.point.temperature + Temperature:Hour + Humidity:Rainfal
           Humidity:Solar.Radiation + sqrt(Wind.speed):Dew.point.temperature +
          Wind.speed:Snowfall + Temperature:sqrt(Wind.speed) + Dew.point.temperat
               Humidity:Dew.point.temperature + Dew.point.temperature:Solar.Radia
test.mat <- model.matrix(Rented.Bike.Count ~ . +  sqrt(Wind.speed)  +  log(Snowfa
          Temperature:Dew.point.temperature + Temperature:Hour + Humidity:Rainfal
           Humidity:Solar.Radiation + sqrt(Wind.speed):Dew.point.temperature +
          Wind.speed:Snowfall + Temperature:sqrt(Wind.speed) + Dew.point.temperat
               Humidity:Dew.point.temperature + Dew.point.temperature:Solar.Radia
grid <- 10^seq(4, -2, length = 100)
```

In [ ]:
```
fit.lasso <- glmnet(train.mat, data_train$Rented.Bike.Count, alpha = 1, lambda =
cv.lasso <- cv.glmnet(train.mat, data_train$Rented.Bike.Count, alpha = 1, lambda
bestlam.lasso <- cv.lasso$lambda.min
bestlam.lasso
```

Hence the best lambda value for lasso generated is approximately 0.14175

In [ ]:
```
pred.lasso <- predict(fit.lasso, s = bestlam.lasso, newx = test.mat)
mean((pred.lasso - data_test$Rented.Bike.Count)^2)
```

Hence the MSE when Lasso is used on relevant parameters comes out to be 148112.926 approximately. This is very less as compared to the above Lasso model and similar to the lm model.

Hence comparing the final lm and Lasso models we get that the lm model is better as it has a less MSE value.

Conclusion

As compared from the MSE values we can say that lm model is better. The final model was generated after applying certain interaction terms and new predictors to the original mode. Hence the linear regression model generated was better.

References

References have been taken from the tutorial and stackoverflow.com