Regression Analysis

A course project submitted in part fulfilment of the requirement for the completion of an Master's degree in Statistics

(2023-25)

Course Code : SI - 422

by

ANKIT MAURYA (23N0069)

SARANSH TIWARI (23N0065)

SAGAR KUMAR (23N0071)

Under the supervision of

DR. SIULI MUKHOPADHYAY

Department of Mathematics

Indian Institute of Technology, Bombay

Mumbai - 400076

India

April 2024

ACKNOWLEDGEMENT

I extend my sincere appreciation to our project supervisor, Prof. Siuli Mukhopadhyay, for her consistent support and expert guidance throughout this endeavour. Their valuable advice, feedback, and constructive criticism have significantly influenced the trajectory of this project, enhancing its quality. I am deeply thankful for their generous investment of time and expertise.

Additionally, I wish to express profound gratitude to all those who have stood by me during this project. Your encouragement, direction, and aid have been indispensable, in facilitating the successful completion of this endeavour.

Lastly, I am grateful to all the resources and references that have contributed to shaping the ideas and concepts presented in this project. The knowledge and insights gleaned from these sources have played a pivotal role in its success.

Once again, I extend my heartfelt thanks to everyone who has supported me throughout this journey. Your unwavering encouragement and assistance have made this project achievable.

OBJECTIVE

The project aims to develop regression models that best explain the variation in the response variable, denoted as Y, using 10 potential predictor variables. Initially, the individual predictor variables will undergo analysis to determine if any unusual pattern exists that could impact model accuracy, potentially necessitating transformations.

The next step is to fit a comprehensive regression model to the data, incorporating any necessary transformations and all 10 predictor variables. The effectiveness of each predictor variable in explaining variation will be assessed. Variance inflation factors will be computed to identify and eliminate weak predictors as necessary, with the ultimate goal of identifying the most influential factors.

In this process, the full data set is utilized to select the best models corresponding to each criterion: Adjusted R-square, Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), Mallows' statistics, and Predicted Residual Sum of Squares (PRESS residuals). Model selection techniques such as backward elimination, forward selection, and stepwise selection are employed to optimize the model. This approach ensures the most accurate and reliable model is selected for interpreting the factors influencing the response variable.

To ensure the model meets the assumptions of linear regression, a thorough residual analysis will be conducted. This analysis will involve examining quantile plots and scatterplots to assess normality and constant variance of errors, as well as identifying and addressing outliers using Cook's Distance. Of particular interest will be identifying outliers with significant impacts on the regression.

CONTENTS

List of Tables

1 Description of Dataset

1.1 Summary of the Variables

- **Data Format:** The data is available in TXT format. To facilitate analysis, we converted the TXT file into CSV format using Excel.
- **Dataset Composition:** The dataset comprises rows and columns, with each column representing a different variable or attribute of the data. The columns are separated by commas, allowing for easy parsing and interpretation.
- **Initial Approach:** Lacking any prior knowledge about the dataset, our initial approach involved thorough analysis to gain insights.
- **Dataset Details:**
  - Number of Records: 113
  - Input Columns: 10, named [x1, x2, x3, x4, x5, x6, x7, x8, x9, x10]
  - Predictor Variable: [y]

Data Type	Variables
Integer	$x5, x8, x9$
Categorical	$x6, x7$
Float	$x1, x2, x3, x4, x10$

Table 1: Data Types of Variables

	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	y
0	7.13	55.7	9.0	39.6	279	2	4	207	241	60.0	4.1
1	8.82	58.2	3.8	51.7	80	2	2	51	52	40.0	1.6
2	8.34	56.9	8.1	74.0	107	2	3	82	54	20.0	2.7
3	8.95	53.7	18.9	122.8	147	2	4	53	148	40.0	5.6
4	11.20	56.5	34.5	88.9	180	2	1	134	151	40.0	5.7
...	...	...	...	...	...	...	...	...	...	...	...
108	11.80	53.8	9.1	116.9	571	1	2	441	469	62.9	5.7
109	9.50	49.3	42.0	70.9	98	2	3	68	46	22.9	5.8
110	7.70	56.9	12.2	67.9	129	2	4	85	136	62.9	4.4
111	17.94	56.2	26.4	91.8	835	1	1	791	407	62.9	5.9
112	9.41	59.5	20.6	91.7	29	2	3	20	22	22.9	3.1

113 rows × 11 columns

1.2 Missing Values

Handling missing data properly is important because it can affect the accuracy of our analysis. If there are missing values, we might replace them with other values through a process called data imputation. However, if there are very few missing values, we might just remove them from the analysis.

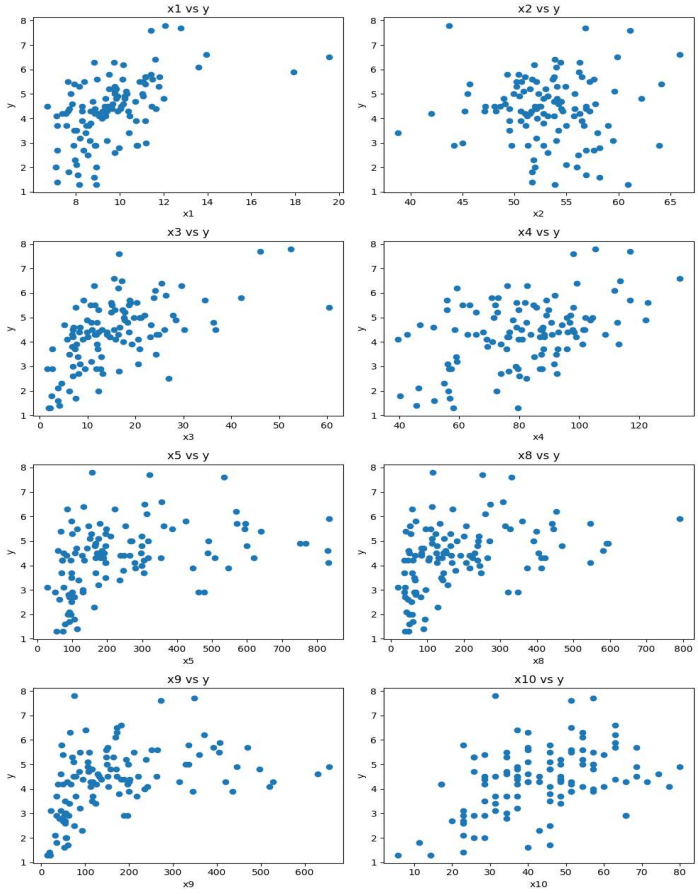
- We checked and since there were no missing values in this dataset, we can move on to the next steps confidently

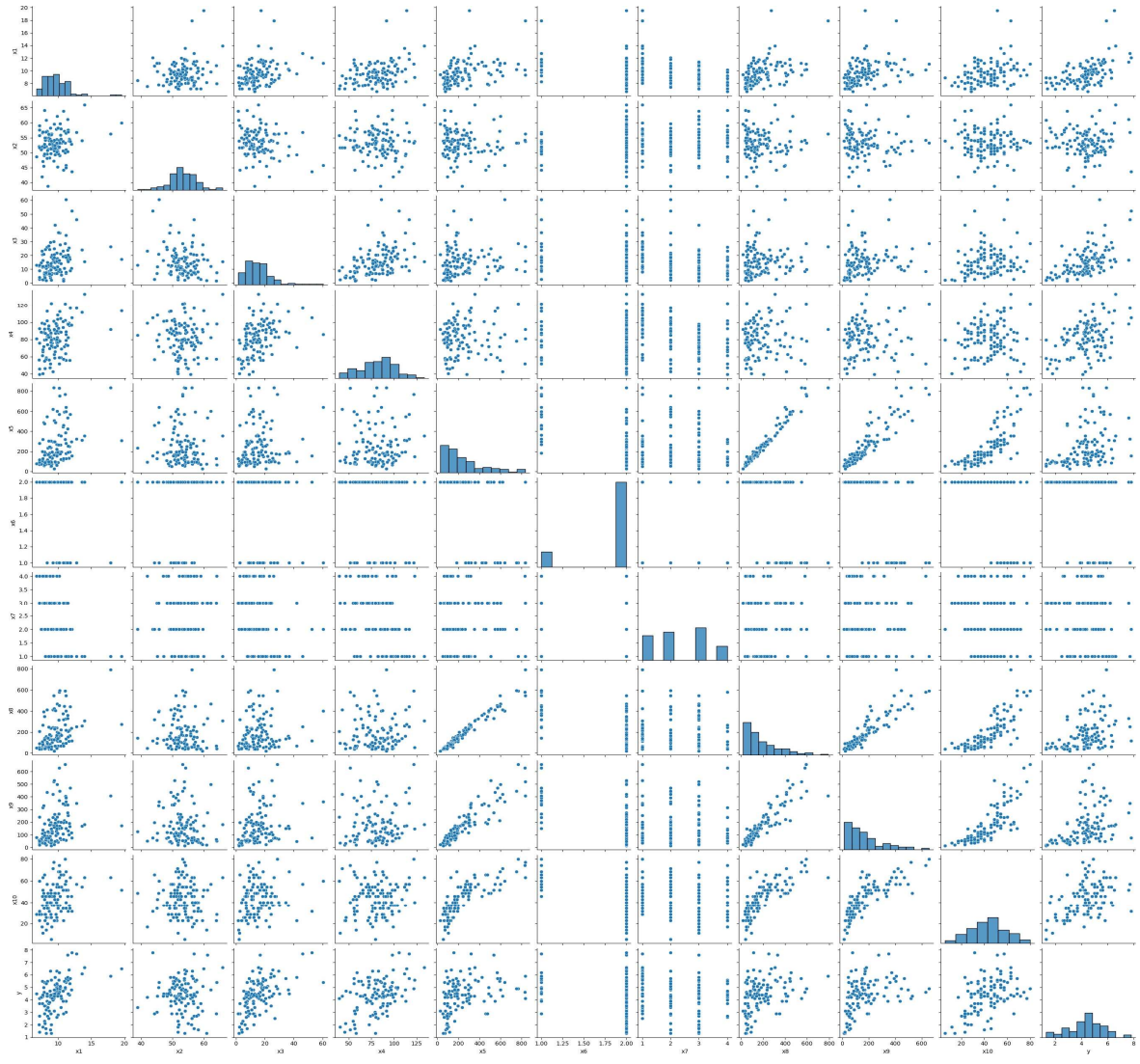
2 Exploratory Data Analysis

To understand the data better, we started by making a **pair plot**.

Pair plot, also known as a scatter-plot matrix, is a matrix of graphs that enables the visualization of the relationship between each pair of variables in a dataset. They facilitate a quick, yet thorough, examination of how variables interact with each other. The major objective at this step was to:

- **Visualize distributions:** Understand the distribution of single variables
- **Identify relationships:** Observe linear or nonlinear relationships between variables





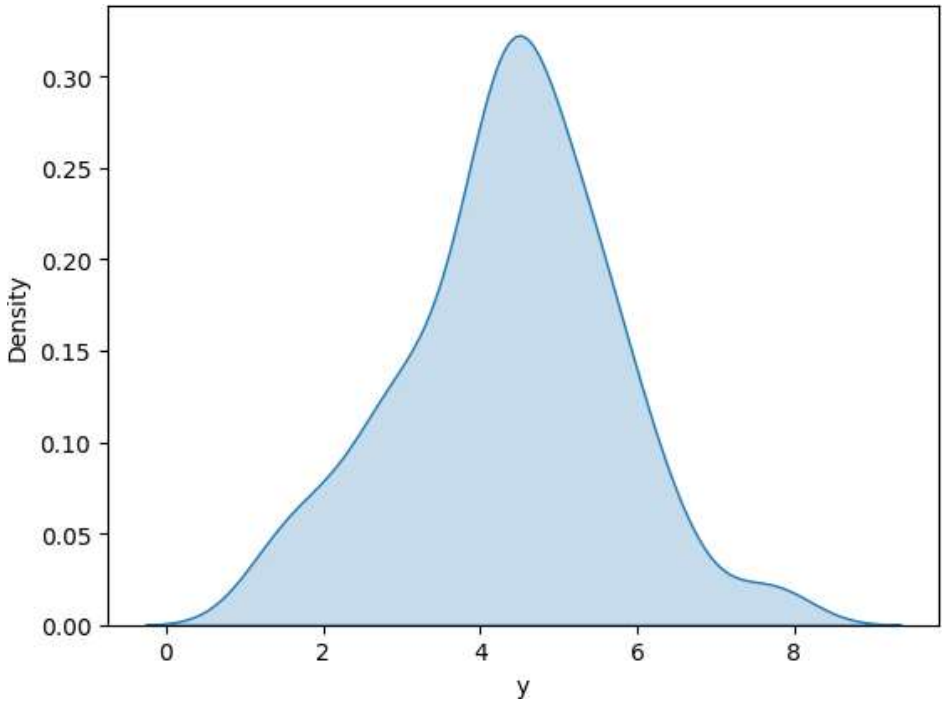
After carefully visualizing this pair plot, the following conclusions were made:-

1. It can be seen that the variables are associated with the y variable, but somewhere, a linear relationship seems missing. May be, some transformation would be needed here.
2. Variables **x1**, **x4**, and **x10** are looking nearly linear and may be beneficial in the further stages of our model-building process.
3. **x5** is linearly related to **x8**, **x9**, and **x10** which may be a threat due to .

2.1 Normality Checking

KDE Plot

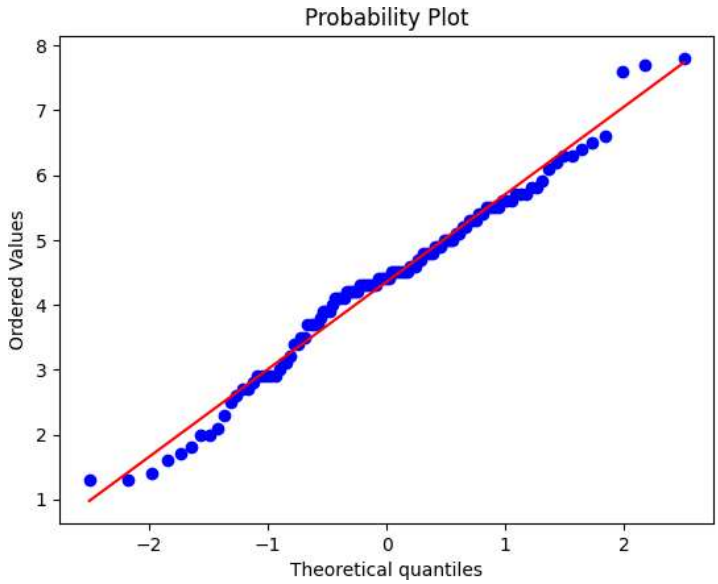
A KDE plot, short for Kernel Density Estimate plot, serves as a visual tool for understanding the distribution of observations within a dataset, similar to a histogram. It utilizes a continuous probability density curve to represent the data across one or more dimensions.



Observation- We can see that the kde plot looks very much similar to normal distribution, but we can not directly comment of the normality assum[tion of y-variable by just seeing seeing this plot. We will have to plot **QQ-Plot** as well.

QQ Plot:

A quantile-quantile plot compares the distribution of data to a theoretical normal distribution. If the data points align with a straight line, it suggests normal distribution.



Observation- A little deviation can be seen and we will see what we have to do for the normality assumption.

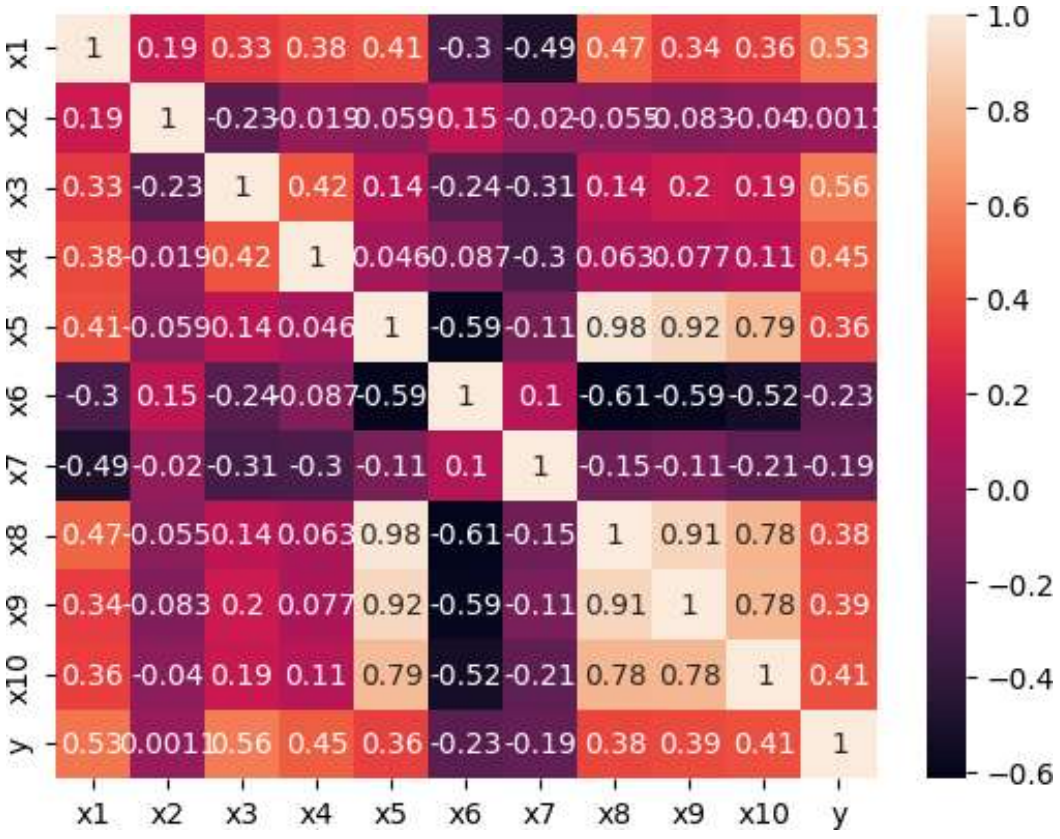
2.2 Correlation

Correlation Heatmap

- A correlation heatmap visually displays the correlation between multiple variables using a color-coded matrix.
- Each variable is represented by a row and column, with cell colors indicating the strength and direction of correlation.

How to Read a Correlation Heatmap?

- Darker colors signify stronger correlations, while lighter colors denote weaker correlations.
- Warm colors like red or orange typically represent positive correlations.
- Cool colors like blue or green typically represent negative correlations.
- The direction and strength of correlation can be inferred by observing the color of each cell in the heatmap.



Observations:

- x_5 exhibits a high correlation with x_8, x_9, and x_{10}.

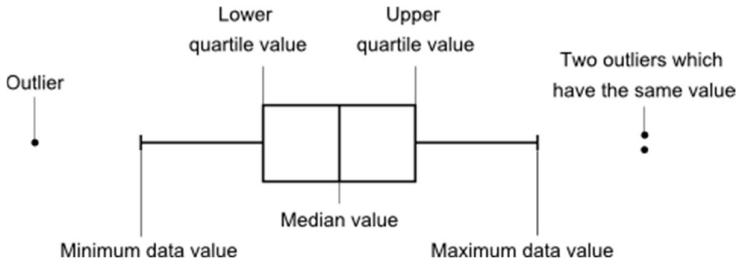
- x_6 demonstrates a moderate correlation with x_5, x_8, and x_9 around 0.60
- x_8 displays a high correlation with x_9 and x_{10} (as it is already correlated with x_5).
- x_9 and x_{10} are highly correlated.

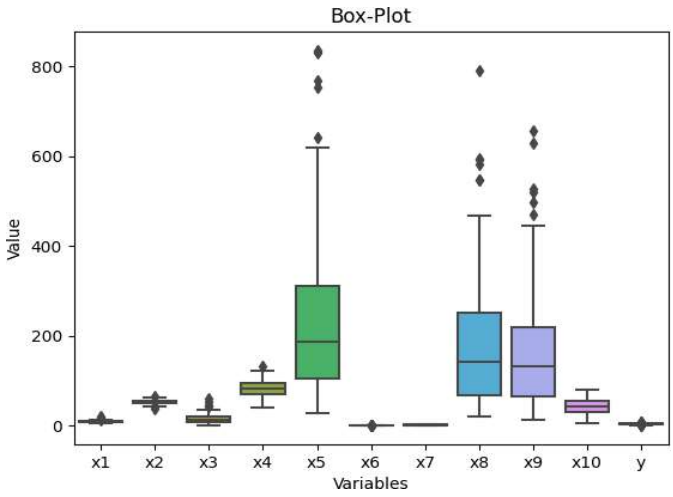
These values suggest the presence of multicollinearity, which can potentially impact the analysis.

Box-Plot

A boxplot, also known as a box-and-whisker plot, is a graphical representation of the distribution of a dataset. It consists of a box that represents the interquartile range (IQR) of the data, with a line inside marking the median. The ”whiskers” extend from the box to the minimum and maximum values of the dataset, excluding outliers. Outliers, if present, are plotted individually as points outside the whiskers. Boxplots are particularly useful for comparing multiple datasets or identifying any unusual observations within a dataset.

Reading a Box and Whisker Plot





Observation:-

- x5, x8 and x9 have some outliers in them
- There is a huge difference between the values of variables such as x5, x8 and x9 : (20-800) while x1/x2 lie in range (20-30) which will make eventually hard to interpret the coefficients and thus, it should be treated as well.

3 Variable Transformation

3.1 One-Hot Encoding

We observed from the pair plot that two input variables are in categorical form: x_6 is labeled as $\{1, 2\}$ only and x_7 is labeled as $\{1, 2, 3, 4\}$ only in the dataset. Therefore, one-hot encoding is applied to these variables.

	x1	x2	x3	x4	x5	x8	x9	x10	y	x7_2	x7_3	x7_4	x6_2
0	7.13	55.7	9.0	39.6	279	207	241	60.0	4.1	0	0	1	1
1	8.82	58.2	3.8	51.7	80	51	52	40.0	1.6	1	0	0	1
2	8.34	56.9	8.1	74.0	107	82	54	20.0	2.7	0	1	0	1
3	8.95	53.7	18.9	122.8	147	53	148	40.0	5.6	0	0	1	1
4	11.20	56.5	34.5	88.9	180	134	151	40.0	5.7	0	0	0	1

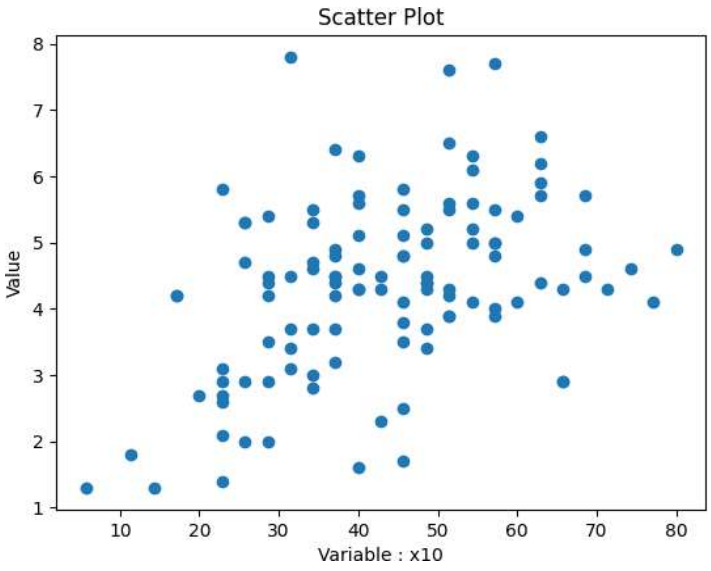
$$x7_2 = \begin{cases} 1 & x7_2 = 2 \\ 1 & x7_2 \neq 2 \end{cases} \qquad x7_3 = \begin{cases} 1 & x7_3 = 3 \\ 1 & x7_3 \neq 3 \end{cases}$$

$$x7_4 = \begin{cases} 1 & x7_4 = 4 \\ 1 & x7_4 \neq 4 \end{cases} \qquad x6_2 = \begin{cases} 1 & x6_2 = 2 \\ 1 & x6_2 \neq 2 \end{cases}$$

3.2 Addressing Linearity Assumption

As observed, the variables exhibit non-linearity with the y-variable. Hence, we aim to determine an appropriate alpha value for each variable through Box-Tidwell transformation to enhance their relationship with the outcome variable.

Scatter plot before applying any transformation -



Procedure

1. **Input Variables**: Begin with the input variables x and y.

2. **Simple Linear Regression**: Initially, fit a simple linear regression model to the data.

3. **Estimate Coefficient**: Obtain the coefficient β_1 (old) from the initial regression model.

4. **Fit Updated Model**: Fit a new linear regression model using the formula $y = \beta_0 + \beta_1^i \times x + \beta_2 \times (x \times \log(x))$.

5. **Calculate Alpha**: Compute the value of α using the formula $\alpha = \frac{\beta_2}{\beta_{1(\text{old})}} + 1$. After applying transformation according to α according to Box Tidwell method:-

Variable	Coefficient
$x1$	-0.5253
$x2$	696.2729
$x3$	-0.0928
$x4$	1.5046
$x5$	-1.5613
$x8$	-0.8494
$x9$	-1.7677
$x10$	-1.5257

We apply following transformation-

Variable	Transformation
$x1$	$x_1^{-0.525}$
$x2$	No Transformation
$x3$	$\log(x_3)$
$x4$	$x_4^{1.5}$
$x5$	$x_5^{-1.56}$
$x8$	$x_8^{-0.84}$
$x9$	$\log(x_9)$
$x10$	$x_{10}^{1.52}$

3.3 Normality assumption

Box-Cox Transformation and Target Variable: Box-Cox transformation is a statistical technique used to transform the target variable of a dataset in order to achieve a normal distribution. The target variable, also known as the dependent variable, is the variable that is being predicted or estimated in a statistical model. By transforming the target variable, Box-Cox helps to improve the predictive power of analytical models by reducing noise and meeting assumptions of normality.

Box-Cox Transformation Equation:

$$w(y) = \begin{cases} \frac{y^{\lambda}-1}{\lambda}, & \text{if } \lambda \neq 0 \\ \ln(y), & \text{if } \lambda = 0 \end{cases}$$

$$\lambda = 1.0944$$

Since the value of λ for the y-variable is 1.09, which is close to 1, there is no need for transformation.

3.4 Multicollinearity

What is a Variation Inflation Factor?

Variance inflation factor(VIF) is a measure of the amount of linear relationship between multiple independent variables in a multiple regression model.

For instance, the VIF for the j^{th} predictor, denoted as VIF_j, is calculated as:

$$VIF_j = \frac{1}{1 - R_j^2} \tag{5}$$

Here, R_j^2 represents the R-squared value obtained by regressing the j^{th} predictor on the remaining predictors.

Variable	VIF
$x1$	2.28
$x2$	1.42
$x3$	1.75
$x4$	1.50
$x5$	6.78
$x8$	16.27
$x9$	4.86
$x10$	4.03
$x7\_2$	1.86
$x7\_3$	2.21
$x7\_4$	1.99
$x6\_2$	1.59

Table 2: Variance Inflation Factor (VIF) Values

Conclusions (before variable transformation) for these VIF values :-

- Variables x_5, x_8, and x_9 exhibit high VIF values, indicating multicollinearity issues.
- The pairplot also reveals a strong correlation between x_5 and x_8, suggesting the need for further treatment of these variables.

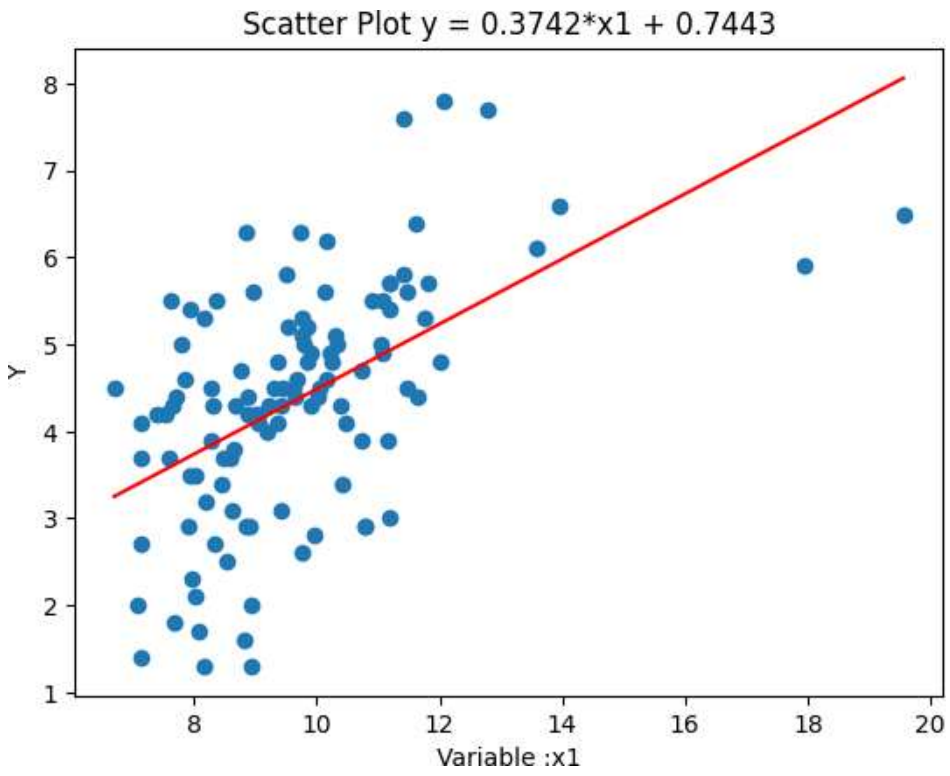
We dropped column [x8]
and VIF of other variable has been reduced -

Variable	VIF
$x1$	2.01
$x2$	1.34
$x3$	1.74
$x4$	1.49
$x5$	2.19
$x9$	5.47
$x10$	3.83
$x7\_2$	1.86
$x7\_3$	2.08
$x7\_4$	1.90
$x6\_2$	1.59

Table 3: Variance Inflation Factor (VIF) Values

Simple Linear Regression

Fitting simple linear regression model of each predictor variable with response variable.



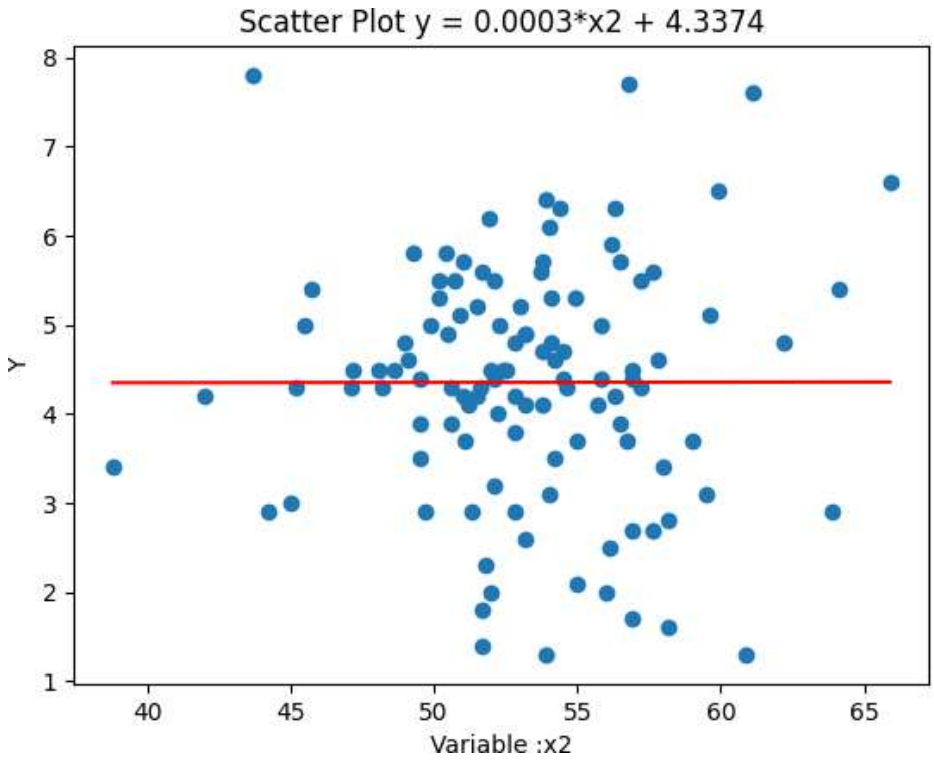
From the p-value, x_1 variable is significant.

OLS Regression Results

Dep. Variable:	y	R-squared:	0.285
Model:	OLS	Adj. R-squared:	0.278
Method:	Least Squares	F-statistic:	44.15
Date:	Thu, 02 May 2024	Prob (F-statistic):	1.18e-09
Time:	00:40:02	Log-Likelihood:	-174.07
No. Observations:	113	AIC:	352.1
Df Residuals:	111	BIC:	357.6
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.7443	0.554	1.344	0.182	-0.353	1.842
x1	0.3742	0.056	6.645	0.000	0.263	0.486

Omnibus:	1.189	Durbin-Watson:	1.821
Prob(Omnibus):	0.552	Jarque-Bera (JB):	1.228
Skew:	-0.237	Prob(JB):	0.541
Kurtosis:	2.811	Cond. No.	51.3



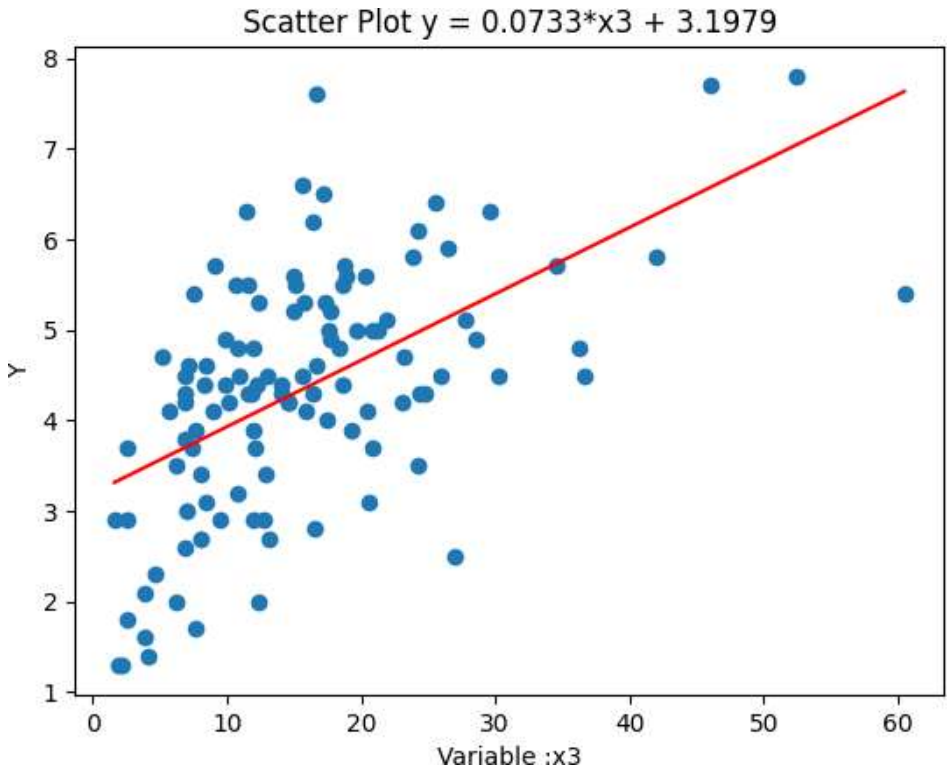
From the p-value, x_2 variable is not significant.

OLS Regression Results

Dep. Variable:	y	R-squared:	0.000
Model:	OLS	Adj. R-squared:	-0.009
Method:	Least Squares	F-statistic:	0.0001326
Date:	Thu, 02 May 2024	Prob (F-statistic):	0.991
Time:	00:40:03	Log-Likelihood:	-192.99
No. Observations:	113	AIC:	390.0
Df Residuals:	111	BIC:	395.4
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	4.3374	1.524	2.846	0.005	1.318	7.357
x2	0.0003	0.029	0.012	0.991	-0.056	0.057

Omnibus:	0.630	Durbin-Watson:	1.779
Prob(Omnibus):	0.730	Jarque-Bera (JB):	0.338
Skew:	-0.119	Prob(JB):	0.845
Kurtosis:	3.122	Cond. No.	643.



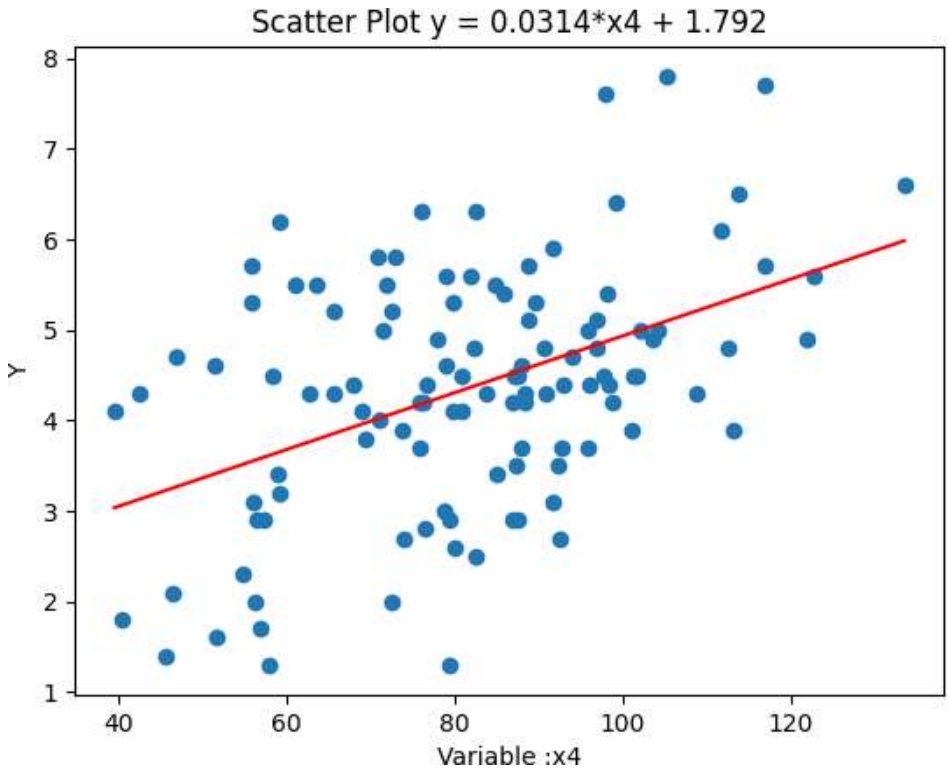
From the p-value, X3 variable is significant.

OLS Regression Results

Dep. Variable:	y	R-squared:	0.313
Model:	OLS	Adj. R-squared:	0.306
Method:	Least Squares	F-statistic:	50.49
Date:	Thu, 02 May 2024	Prob (F-statistic):	1.22e-10
Time:	00:40:03	Log-Likelihood:	-171.80
No. Observations:	113	AIC:	347.6
Df Residuals:	111	BIC:	353.1
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	3.1979	0.194	16.504	0.000	2.814	3.582
x3	0.0733	0.010	7.106	0.000	0.053	0.094

Omnibus:	0.150	Durbin-Watson:	1.981
Prob(Omnibus):	0.928	Jarque-Bera (JB):	0.261
Skew:	-0.079	Prob(JB):	0.878
Kurtosis:	2.826	Cond. No.	34.7



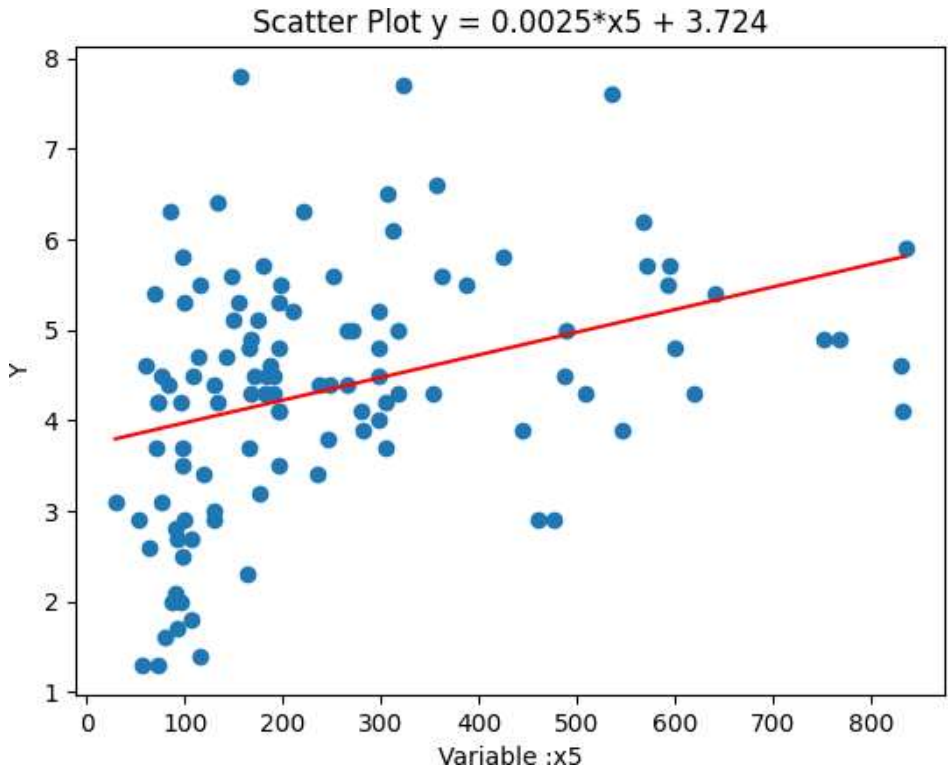
From the p-value, X4 variable is significant.

OLS Regression Results

Dep. Variable:	y	R-squared:	0.206
Model:	OLS	Adj. R-squared:	0.198
Method:	Least Squares	F-statistic:	28.72
Date:	Thu, 02 May 2024	Prob (F-statistic):	4.58e-07
Time:	00:40:03	Log-Likelihood:	-179.98
No. Observations:	113	AIC:	364.0
Df Residuals:	111	BIC:	369.4
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	1.7920	0.491	3.647	0.000	0.818	2.766
x4	0.0314	0.006	5.359	0.000	0.020	0.043

Omnibus:	1.236	Durbin-Watson:	1.780
Prob(Omnibus):	0.539	Jarque-Bera (JB):	1.181
Skew:	0.115	Prob(JB):	0.554
Kurtosis:	2.555	Cond. No.	365.



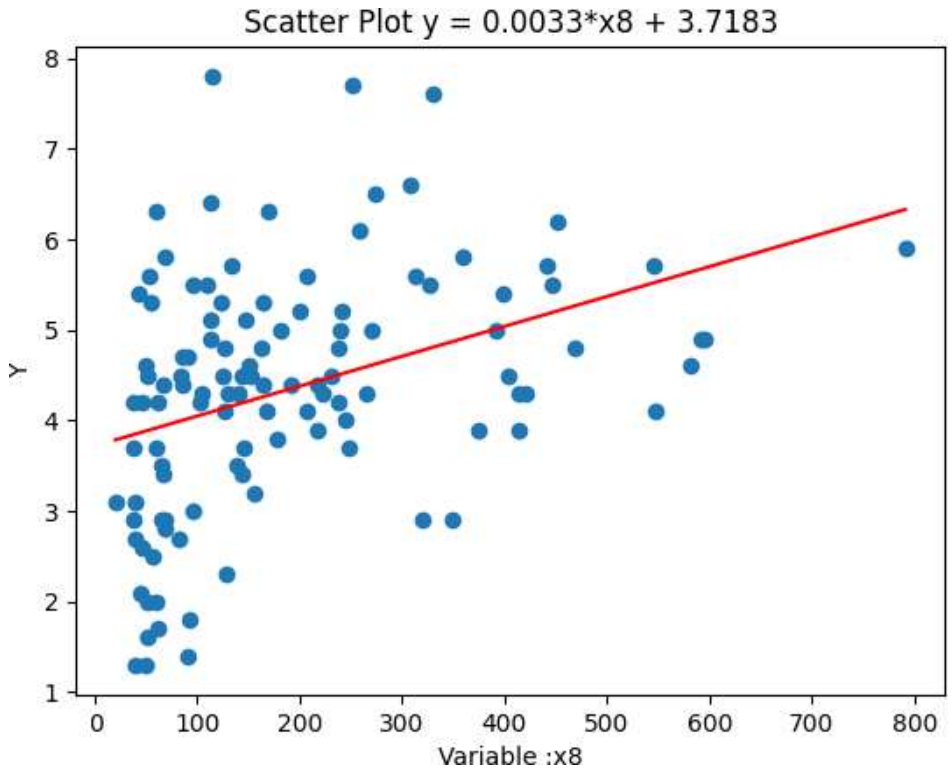
From the p-value, X5 variable is significant.

OLS Regression Results

Dep. Variable:	y	R-squared:	0.129
Model:	OLS	Adj. R-squared:	0.122
Method:	Least Squares	F-statistic:	16.50
Date:	Thu, 02 May 2024	Prob (F-statistic):	9.09e-05
Time:	00:40:03	Log-Likelihood:	-185.15
No. Observations:	113	AIC:	374.3
Df Residuals:	111	BIC:	379.8
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	3.7240	0.195	19.081	0.000	3.337	4.111
x5	0.0025	0.001	4.062	0.000	0.001	0.004

Omnibus:	0.588	Durbin-Watson:	1.818
Prob(Omnibus):	0.745	Jarque-Bera (JB):	0.377
Skew:	0.139	Prob(JB):	0.828
Kurtosis:	3.055	Cond. No.	523.



From the p-value, X8 variable is significant.

Variable :x8

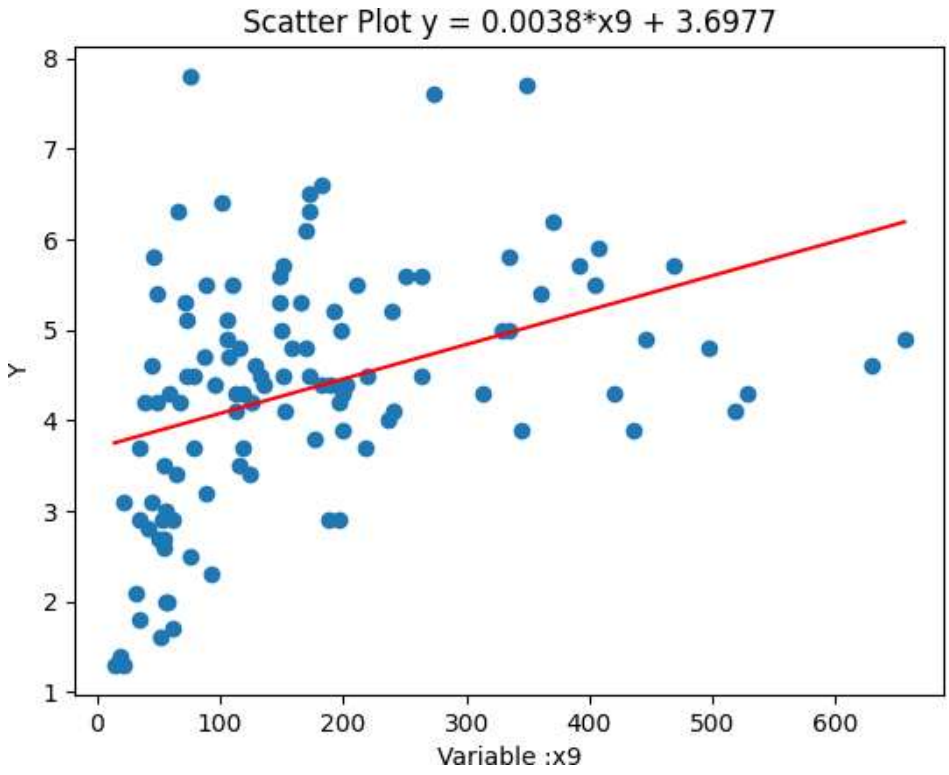
OLS Regression Results

Dep. Variable:	y	R-squared:	0.145
Model:	OLS	Adj. R-squared:	0.138
Method:	Least Squares	F-statistic:	18.90
Date:	Thu, 02 May 2024	Prob (F-statistic):	3.07e-05
Time:	00:40:03	Log-Likelihood:	-184.10
No. Observations:	113	AIC:	372.2
Df Residuals:	111	BIC:	377.7
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	3.7183	0.188	19.829	0.000	3.347	4.090
x8	0.0033	0.001	4.347	0.000	0.002	0.005

Omnibus:	1.150	Durbin-Watson:	1.795
Prob(Omnibus):	0.563	Jarque-Bera (JB):	0.789
Skew:	0.192	Prob(JB):	0.674
Kurtosis:	3.144	Cond. No.	392.

.



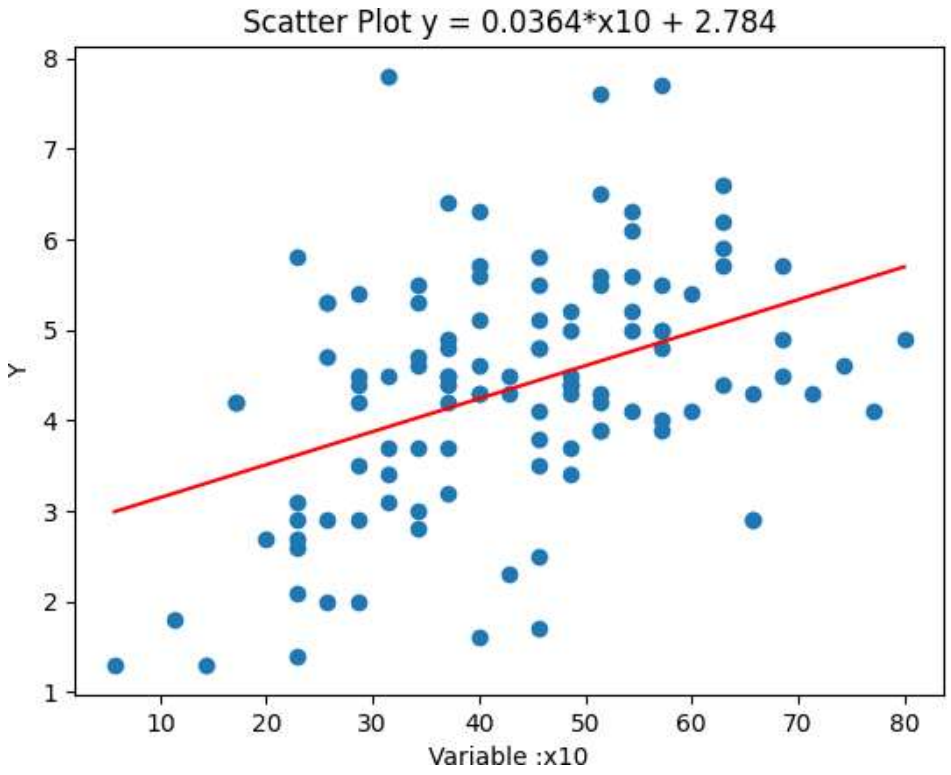
From the p-value, X9 variable is significant

OLS Regression Results

Dep. Variable:	y	R-squared:	0.155
Model:	OLS	Adj. R-squared:	0.148
Method:	Least Squares	F-statistic:	20.40
Date:	Thu, 02 May 2024	Prob (F-statistic):	1.58e-05
Time:	00:40:03	Log-Likelihood:	-183.46
No. Observations:	113	AIC:	370.9
Df Residuals:	111	BIC:	376.4
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	3.6977	0.186	19.839	0.000	3.328	4.067
x9	0.0038	0.001	4.516	0.000	0.002	0.005

Omnibus:	1.515	Durbin-Watson:	1.870
Prob(Omnibus):	0.469	Jarque-Bera (JB):	1.226
Skew:	0.253	Prob(JB):	0.542
Kurtosis:	3.060	Cond. No.	355.



From the p-value, we can say that x_{10} variable is significant.

OLS Regression Results

Dep. Variable:	y	R-squared:	0.170
Model:	OLS	Adj. R-squared:	0.163
Method:	Least Squares	F-statistic:	22.77
Date:	Thu, 02 May 2024	Prob (F-statistic):	5.59e-06
Time:	00:40:04	Log-Likelihood:	-182.44
No. Observations:	113	AIC:	368.9
Df Residuals:	111	BIC:	374.3
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	2.7840	0.349	7.981	0.000	2.093	3.475
x10	0.0364	0.008	4.772	0.000	0.021	0.052

Omnibus:	1.709	Durbin-Watson:	1.853
Prob(Omnibus):	0.425	Jarque-Bera (JB):	1.237
Skew:	0.231	Prob(JB):	0.539
Kurtosis:	3.224	Cond. No.	138.

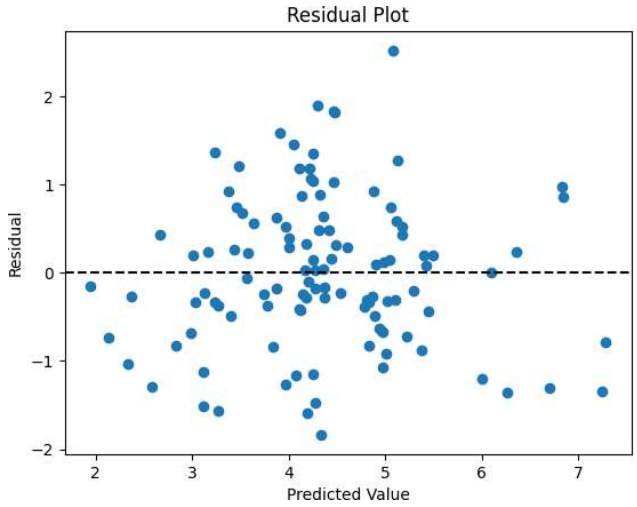
4 Model fitting

Without applying any transformation to the predictor variables, we fitted our full model and obtained the following metrics.

Model Fitting (Full Model)

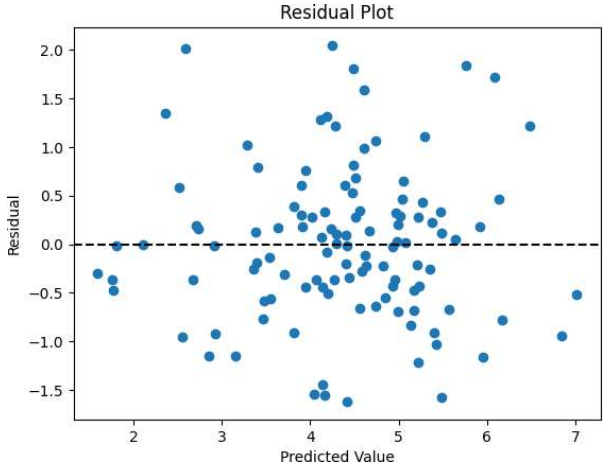
Metric	Value
R-Squared	0.5854
Adjusted R-Squared	0.5356
Mallow's Statistic on Data for full model	13.0
AIC on Data for full model	−8.1868
BIC on Data for full model	27.2692

Residual Plot



* A slight change in the variance of the residuals has been observed when increasing the predicted value.

Residual plot after applying Box Tidwell



The pattern which could have be seen in the scatter plot of residuals earlier now seems sorted after applying transformation according to Box-Tidwell .

5 Best Model Selection

We have used following technique to model selection-

5.1 Using R-Squared

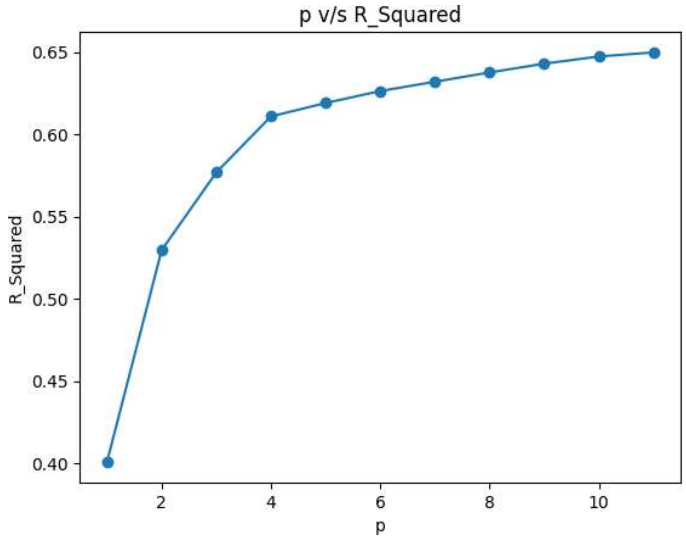
R-squared is a statistical measure that indicates how much of the variation of a dependent variable is explained by an independent variable in a regression model.

$$R^2 = 1 - \frac{\text{SSR}}{\text{SSTO}} \tag{6}$$

p	Variables in Model	R-Squared
1	['x3']	0.400873399
2	['x1', 'x3']	0.529581337
3	['x1', 'x3', 'x9']	0.576796746
4	['x1', 'x3', 'x9', 'x7$_4$']	0.610842599
5	['x1', 'x3', 'x9', 'x7$_4$, x6$_2$']	0.618994976
6	['x1', 'x3', 'x4', 'x9', 'x7$_4$, x6$_2$']	0.626282839
7	['x1', 'x2', 'x3', 'x9', 'x10', 'x7$_2$, x7$_4$']	0.631967363
8	['x1', 'x2', 'x3', 'x4', 'x9', 'x10', 'x7$_2$, x7$_4$']	0.637594367
9	['x1', 'x2', 'x3', 'x4', 'x5', 'x9', 'x10', 'x7$_2$, x7$_4$']	0.642924623
10	['x1', 'x2', 'x3', 'x4', 'x5', 'x9', 'x10', 'x7$_2$, x7$_3$, x7$_4$']	0.647284007
11	['x1', 'x2', 'x3', 'x4', 'x5', 'x9', 'x10', 'x7$_2$, x7$_3$, x7$_4$, x6$_2$']	0.64977429

Table 4: Variations of R-Squared with the Number of Variables

From the table above, we observe that the model with the highest R^2 value is obtained for $p = 11$, which includes the following regressor variables: $['x1', 'x2', 'x3', 'x4', 'x5', 'x9', 'x10', 'x7_2, x7_3, x7_4, x6_2]$.



We observe from the graph that the maximum R-Squared is achieved at $p = 11$.

5.2 Using Adjusted R-Squared

Adjusted R-squared is a modified version of R-squared that has been adjusted for the number of predictors in the model. It increases when the new term improves the model more than would be expected by chance and decreases when a predictor improves the model by less than expected.

$$\text{Adjusted } R^2 = 1 - \frac{(1 - R^2) \times (n - 1)}{n - k - 1}$$

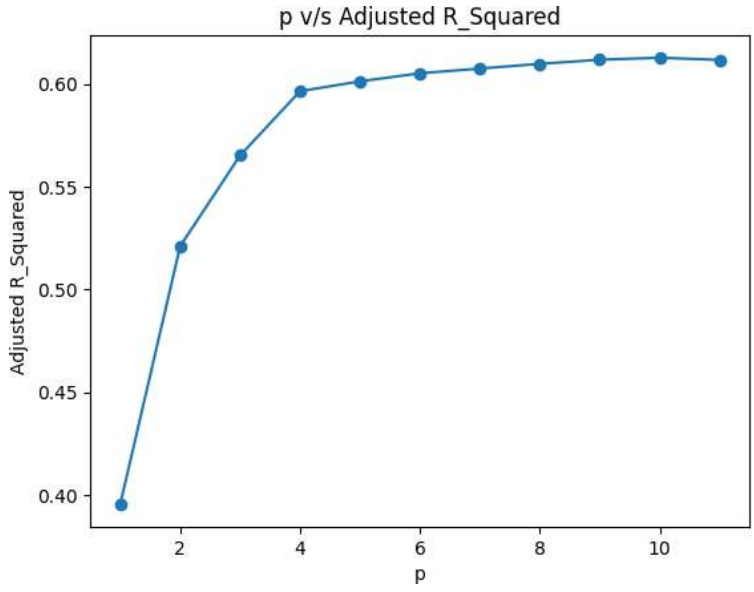
For each regressor, we aim to obtain the maximum R-squared value possible.

From the table above, we observe that the model with the highest $adjusted R^2$ value is obtained for

p	Variables in Model	Adjusted R^2
1	['x3']	0.3955
2	['x1', 'x3']	0.5210
3	['x1', 'x3', 'x9']	0.5651
4	['x1', 'x3', 'x9', 'x7$_4'$]	0.5964
5	['x1', 'x3', 'x9', 'x7$_4'$, 'x6$_2'$]	0.6012
6	['x1', 'x3', 'x4', 'x9', 'x7$_4'$, 'x6$_2'$]	0.6051
7	['x1', 'x2', 'x3', 'x9', 'x10', 'x7$_2'$, 'x7$_4'$]	0.6074
8	['x1', 'x2', 'x3', 'x4', 'x9', 'x10', 'x7$_2'$, 'x7$_4'$]	0.6097
9	['x1', 'x2', 'x3', 'x4', 'x5', 'x9', 'x10', 'x7$_2'$, 'x7$_4'$]	0.6117
10	['x1', 'x2', 'x3', 'x4', 'x5', 'x9', 'x10', 'x7$_2'$, 'x7$_3'$, 'x7$_4'$]	0.6127
11	['x1', 'x2', 'x3', 'x4', 'x5', 'x9', 'x10', 'x7$_2'$, 'x7$_3'$, 'x7$_4'$, 'x6$_2'$]	0.6116

Table 5: Adjusted R^2 Values for Different Models

$p = 10$, which includes the following regressor variables: ['x1', 'x2', 'x3', 'x4', 'x5', 'x9', 'x10', 'x7$_2$', 'x7$_3$', 'x7$_4$']



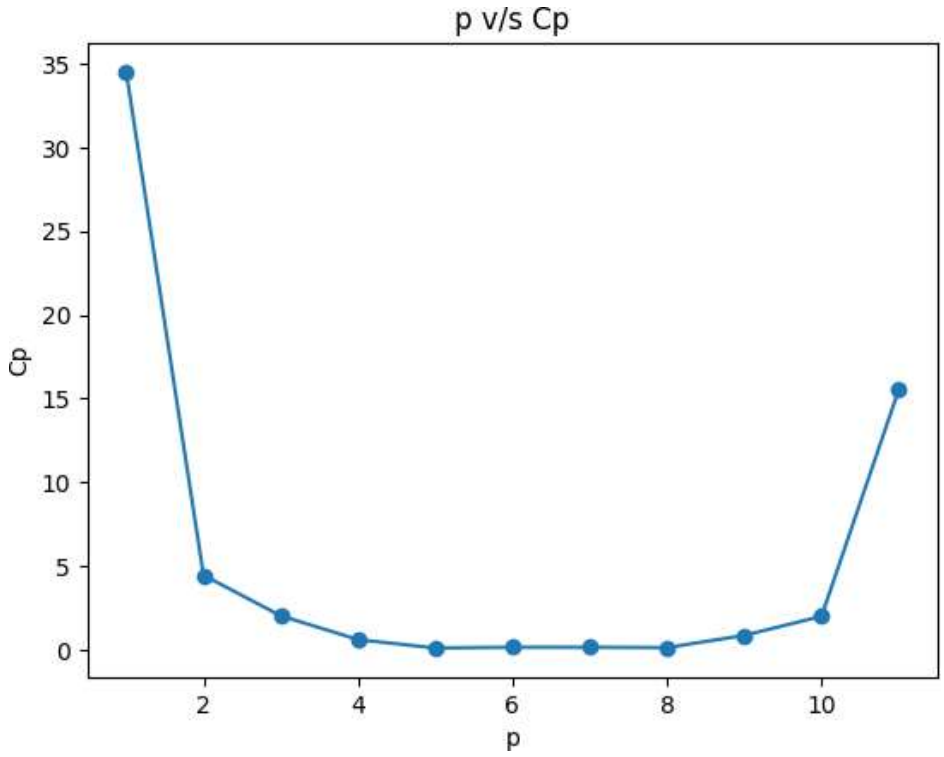
We observe from the graph that the maximum Adjusted R-Squared is achieved at $p = 10$.

5.3 Using Mallow's Statistic (C_p)

p	Variables in Model	Cp
1	['x3']	35.493
2	['x1', 'x3']	6.452
3	['x1', 'x3', 'x10']	5.037
4	['x2', 'x3', 'x4', 'x9']	3.386
5	['x2', 'x3', 'x4', 'x9', '$x6_2'$]	5.113
6	['x1', 'x3', 'x5', 'x10', '$x7_2'$, '$x6_2'$]	5.839
7	['x2', 'x3', 'x4', 'x5', 'x9', '$x7_2'$, '$x6_2'$]	7.160
8	['x2', 'x3', 'x4', 'x9', '$x7_2'$, '$x7_3'$, '$x7_4'$, '$x6_2'$]	8.138
9	['x2', 'x3', 'x4', 'x5', 'x9', 'x10', '$x7_2'$, '$x7_3'$, '$x6_2'$]	8.134
10	['x2', 'x3', 'x4', 'x5', 'x9', 'x10', '$x7_2'$, '$x7_3'$, '$x7_4'$, '$x6_2'$]	7.991
11	['x1', 'x2', 'x3', 'x4', 'x5', 'x9', 'x10', '$x7_2'$, '$x7_3'$, '$x7_4'$, '$x6_2'$]	-4.535

Table 6: Cp Values for Different Models

From the table above, we observe that the model with the highest CP value is obtained for $p = 5$, which includes the following regressor variables: ['$x2'$, '$x3'$, '$x4'$, '$x9'$, '$x6_2'$].



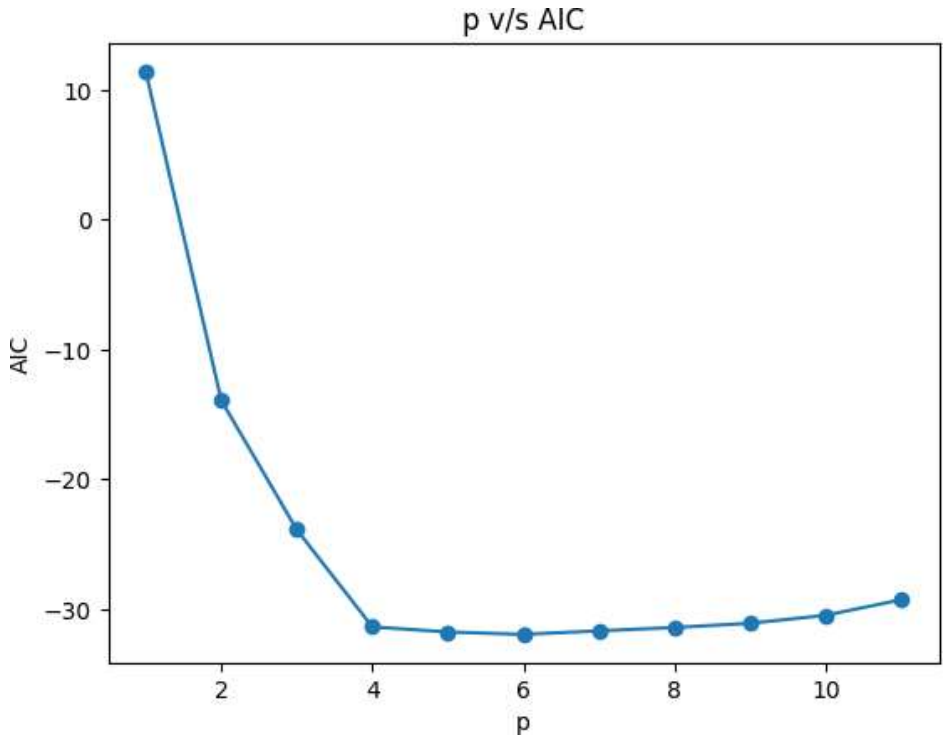
We observe from graph CP is closed to $p = 5$

5.4 Using AIC

p	Variables in Model	AIC
1	['x3']	11.4041
2	['x1', 'x3']	-13.9250
3	['x1', 'x3', 'x9']	-23.8770
4	['x1', 'x3', 'x9', '$x7_4'$]	-31.3542
5	['x1', 'x3', 'x9', '$x7_4'$, '$x6_2'$]	-31.7466
6	['x1', 'x3', 'x4', 'x9', '$x7_4'$, '$x6_2'$]	-31.9290
7	['x1', 'x2', 'x3', 'x9', 'x10', '$x7_2'$, '$x7_4'$]	-31.6610
8	['x1', 'x2', 'x3', 'x4', 'x9', 'x10', '$x7_2'$, '$x7_4'$]	-31.4020
9	['x1', 'x2', 'x3', 'x4', 'x5', 'x9', 'x10', '$x7_2'$, '$x7_4'$]	-31.0764
10	['x1', 'x2', 'x3', 'x4', 'x5', 'x9', 'x10', '$x7_2'$, '$x7_3'$, '$x7_4'$]	-30.4644
11	['x1', 'x2', 'x3', 'x4', 'x5', 'x9', 'x10', '$x7_2'$, '$x7_3'$, '$x7_4'$, '$x6_2'$]	-29.2651

Table 7: AIC Values for Different Models

From the table above, we observe that the model with *AIC* value is minimum for $p = 6$



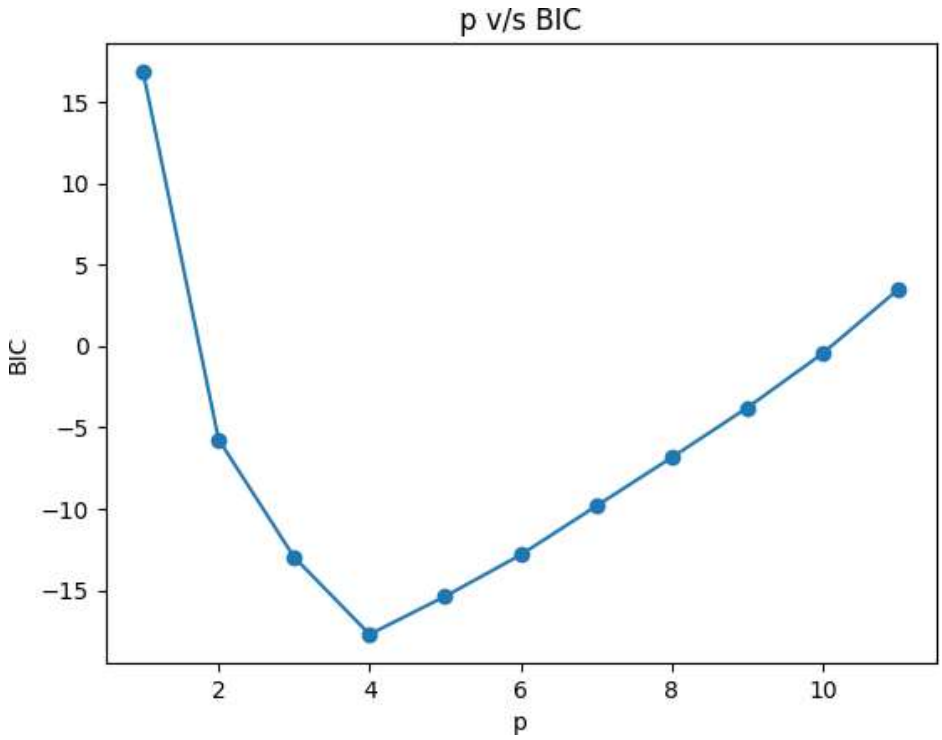
We generally want AIC to be less and it is visible from the graph that AIC is minimum fo r p = 6.

5.5 Using BIC

p	Variables in Model	BIC
1	['x3']	16.8588
2	['x1', 'x3']	-5.7428
3	['x1', 'x3', 'x9']	-12.9675
4	['x1', 'x3', 'x9', '$x7_4'$]	-17.7173
5	['x1', 'x3', 'x9', '$x7_4'$, '$x6_2'$]	-15.3822
6	['x1', 'x3', 'x4', 'x9', '$x7_4'$, '$x6_2'$]	-12.8373
7	['x1', 'x2', 'x3', 'x9', 'x10', '$x7_2'$, '$x7_4'$]	-9.8419
8	['x1', 'x2', 'x3', 'x4', 'x9', 'x10', '$x7_2'$, '$x7_4'$]	-6.8555
9	['x1', 'x2', 'x3', 'x4', 'x5', 'x9', 'x10', '$x7_2'$, '$x7_4'$]	-3.8025
10	['x1', 'x2', 'x3', 'x4', 'x5', 'x9', 'x10', '$x7_2'$, '$x7_3'$, '$x7_4'$]	-0.4632
11	['x1', 'x2', 'x3', 'x4', 'x5', 'x9', 'x10', '$x7_2'$, '$x7_3'$, '$x7_4'$, '$x6_2'$]	3.4636

Table 8: BIC Values for Different Models

From the table above, we observe that the model with *BIC* value is minimum for $p = 4$



We generally want BIC to be less and it is visible from the graph that BIC is minimum for p=4

5.6 Using Backward Elimination Technique

Therefore, by backward elimination technique, the best model selected is having four variables $[x1, x3, x5, x7_4]$.

OLS Regression Results

Dep. Variable:	y	**R-squared (uncentered):**	0.962
Model:	OLS	**Adj. R-squared (uncentered):**	0.960
Method:	Least Squares	**F-statistic:**	687.5
Date:	Sat, 27 Apr 2024	**Prob (F-statistic):**	2.53e-76
Time:	11:39:08	**Log-Likelihood:**	-147.09
No. Observations:	113	**AIC:**	302.2
Df Residuals:	109	**BIC:**	313.1
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
x1	0.6912	0.107	6.430	0.000	0.478	0.904
x3	0.9152	0.130	7.017	0.000	0.657	1.174
x5	-587.4862	134.675	-4.362	0.000	-854.408	-320.564
x7_4	0.4984	0.244	2.043	0.044	0.015	0.982

Omnibus:	1.628	**Durbin-Watson:**	1.975
Prob(Omnibus):	0.443	**Jarque-Bera (JB):**	1.478
Skew:	0.279	**Prob(JB):**	0.477
Kurtosis:	2.952	**Cond. No.**	6.62e+03

Best Model from each Criteria

Criterion	Variables in Model	p	SSE	R-Squared	Adjusted R-Squared	Cp	AIC	BIC
R-Squared	['x1', 'x2', 'x3', 'x4', 'x5', 'x9', 'x10', 'x7_2', 'x7_3', 'x7_4', 'x6_2']	11	70.528	0.649	0.612	-4.535	-29.265	3.464
Adjusted R-Squared	['x1', 'x2', 'x3', 'x4', 'x5', 'x9', 'x10', 'x7_2', 'x7_3', 'x7_4']	10	71.030	0.647	0.613	-5.935	-30.464	-0.463
SSE	['x1', 'x2', 'x3', 'x4', 'x5', 'x9', 'x10', 'x7_2', 'x7_3', 'x7_4', 'x6_2']	11	70.528	0.650	0.612	-4.535	-29.265	3.464
Cp	['x2', 'x3', 'x4', 'x9', 'x6_2']	5	88.605	0.560	0.539	5.113	-15.482	0.882
AIC	['x1', 'x3', 'x4', 'x9', 'x7_4', 'x6_2']	6	75.259	0.626	0.605	-8.870	-31.929	-12.837
BIC	['x1', 'x3', 'x9', 'x7_4']	4	78.368	0.611	0.596	-9.146	-31.354	-17.717
Backward Elimination	['x1', 'x3', 'x5', 'x7_4']	4	79.986	0.603	0.588	-7.209	-29.045	-15.408

Table 9: Criteria for Model Selection

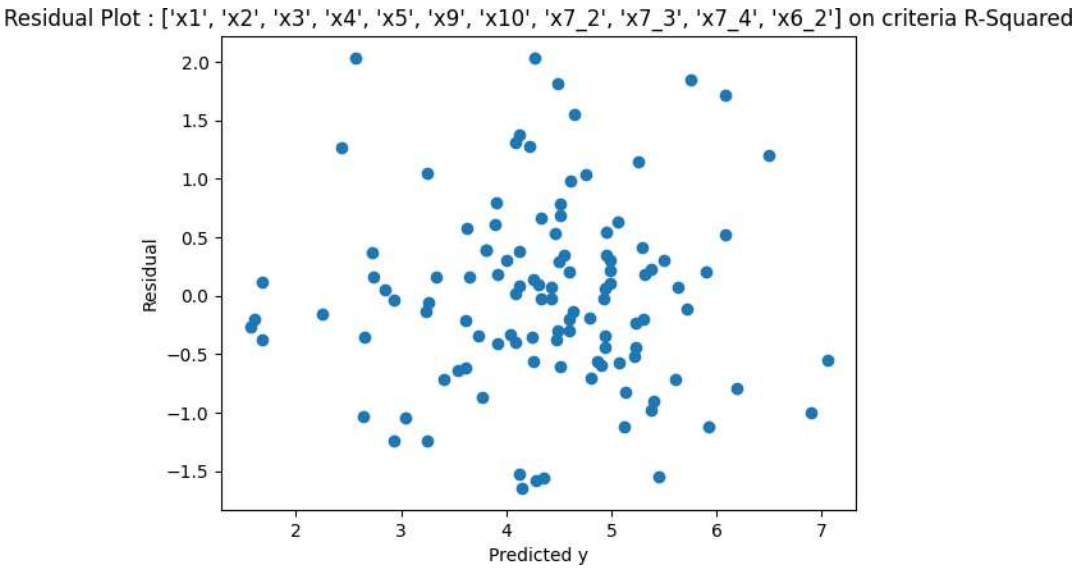
In the above table best variables according to each criterion has been shown.

6 Residual analysis

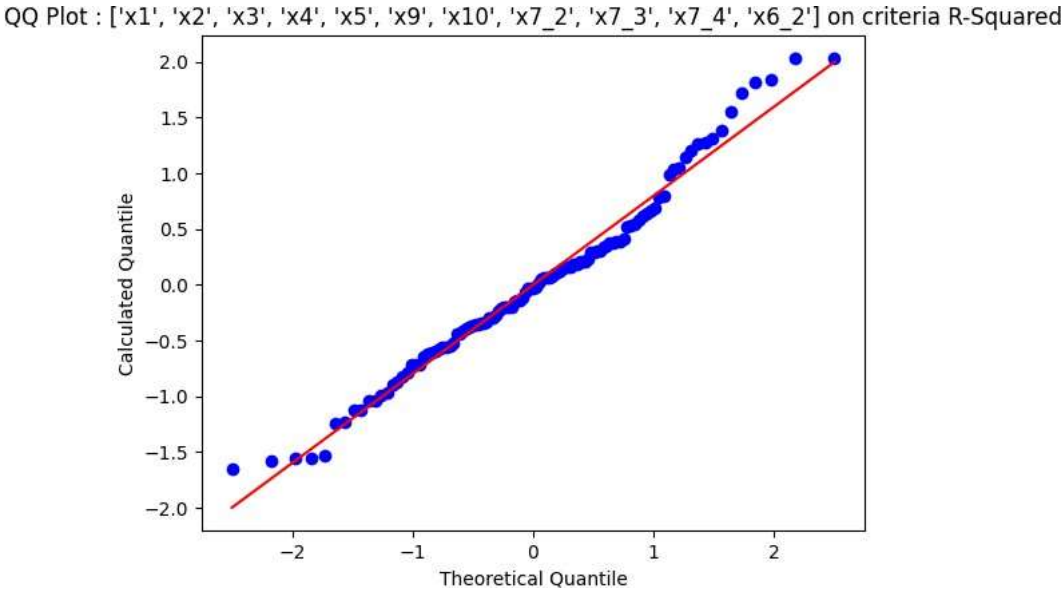
In this section we analyze the assumption of the residual obtain from the model filtted based on table 9.

6.1 Normality Heteroscedasticity

Resdiual and QQ-plot based on best R^2

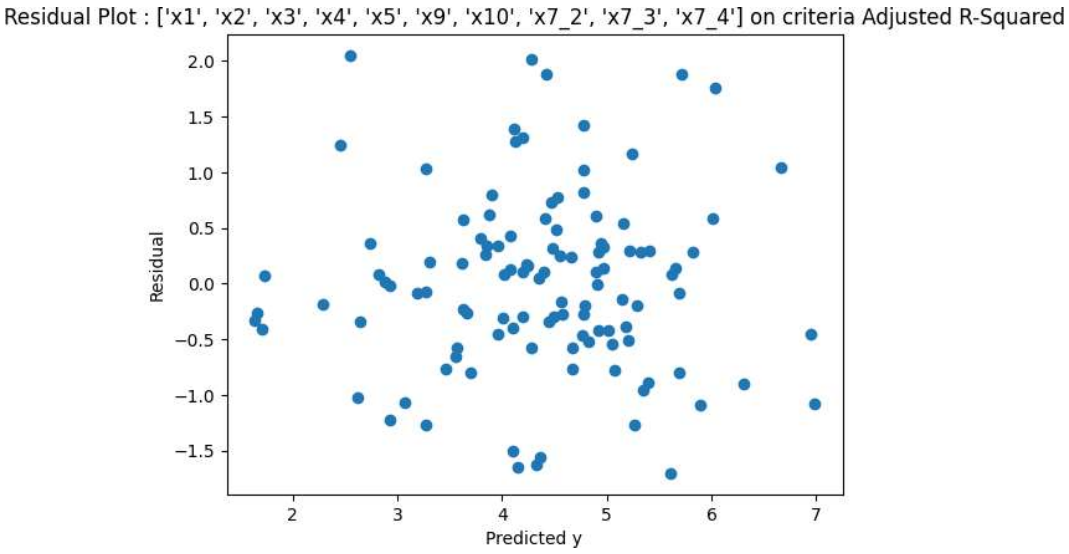


Based on the Brown-Forsythe test, we observe that there exists homoscedasticity in the residual plot.

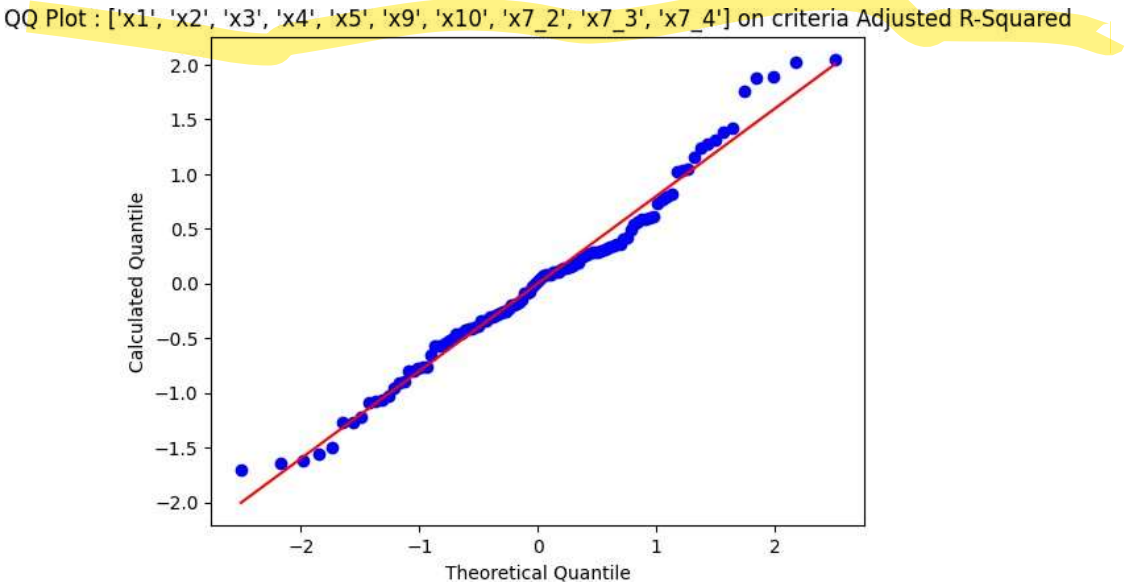


We can see a slight deviation from the normality in QQ-PLOT.

Resdiual and QQ-plot based on best Adjusted R^2

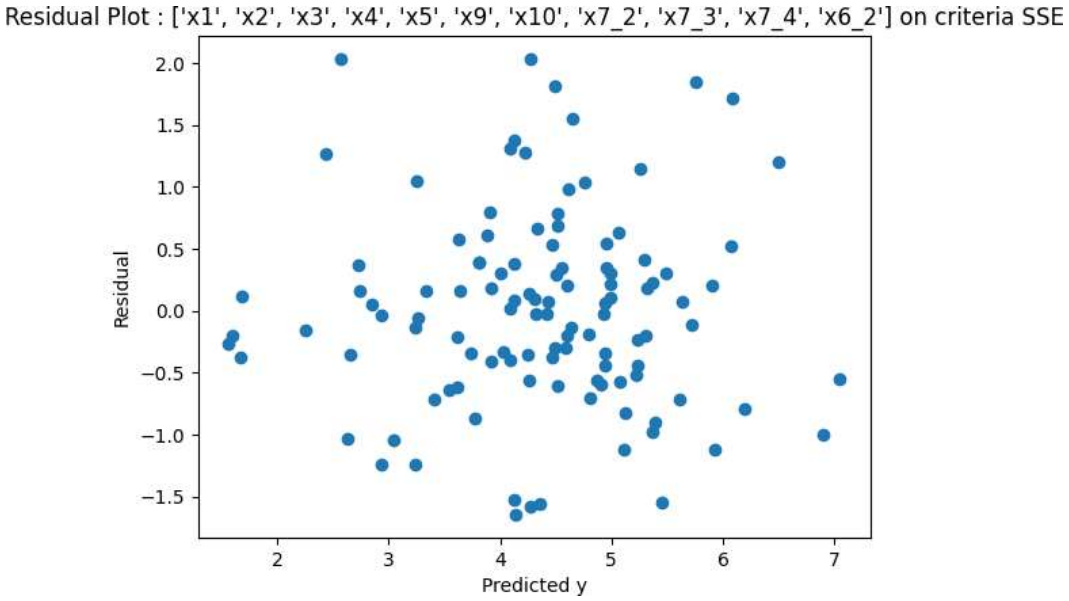


Based on the Brown-Forsythe test, we observe that there exists homoscedasticity in the residual plot.

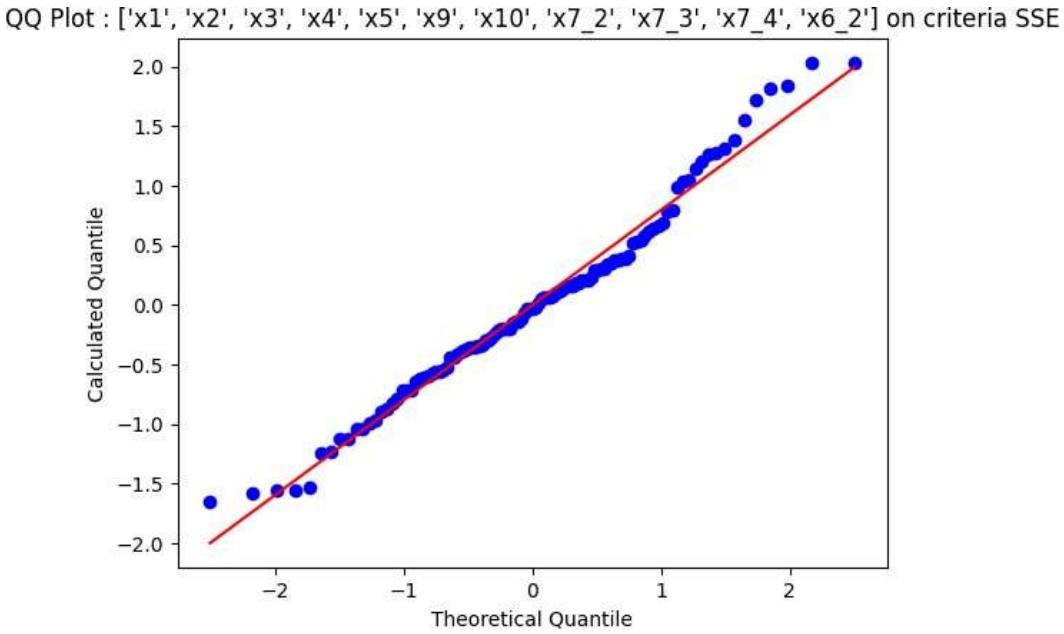


We can see a slight deviation from the normality in QQ-PLOT in right tail.

Resdiual and QQ-plot based on best SSE

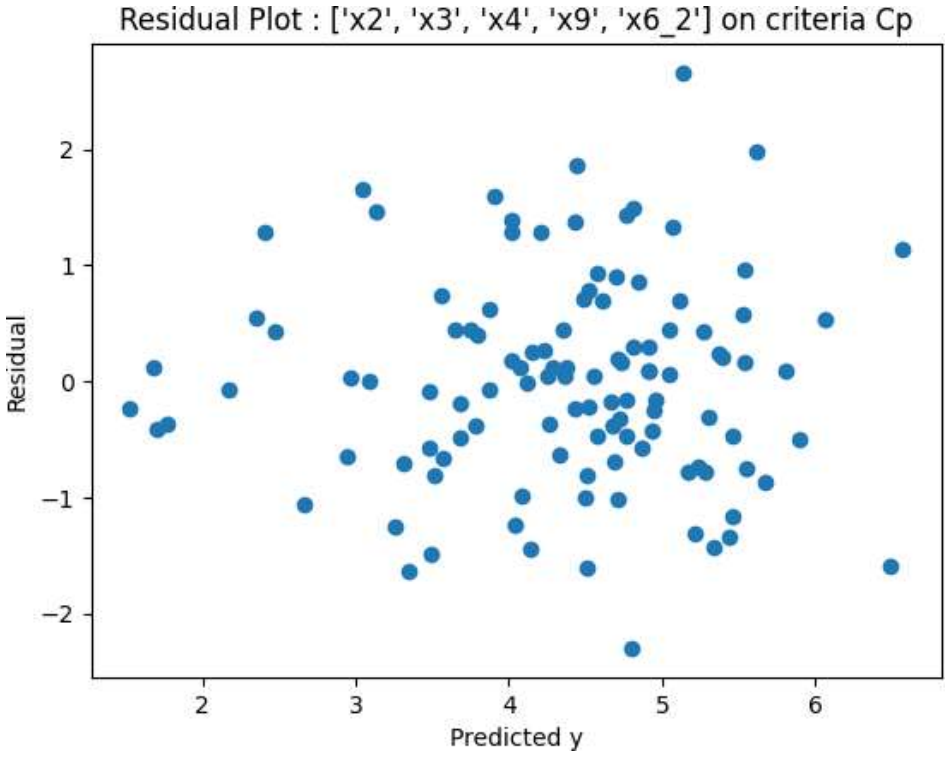


Based on the Brown-Forsythe test, we observe that there exists homoscedasticity in the residual plot.

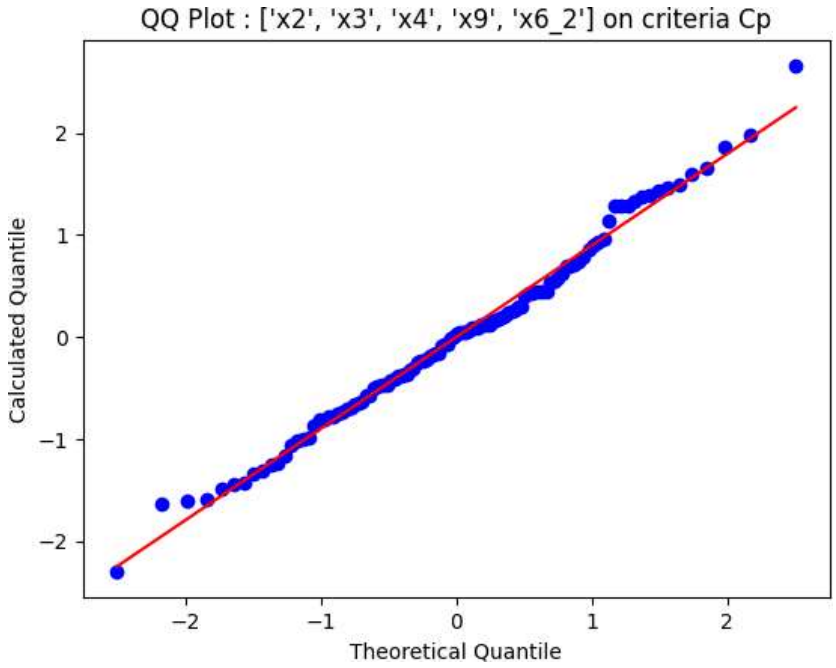


We can see a slight deviation from the normality in QQ-PLOT in right tail.

Resdiual and QQ-plot based on best *CP* criteria

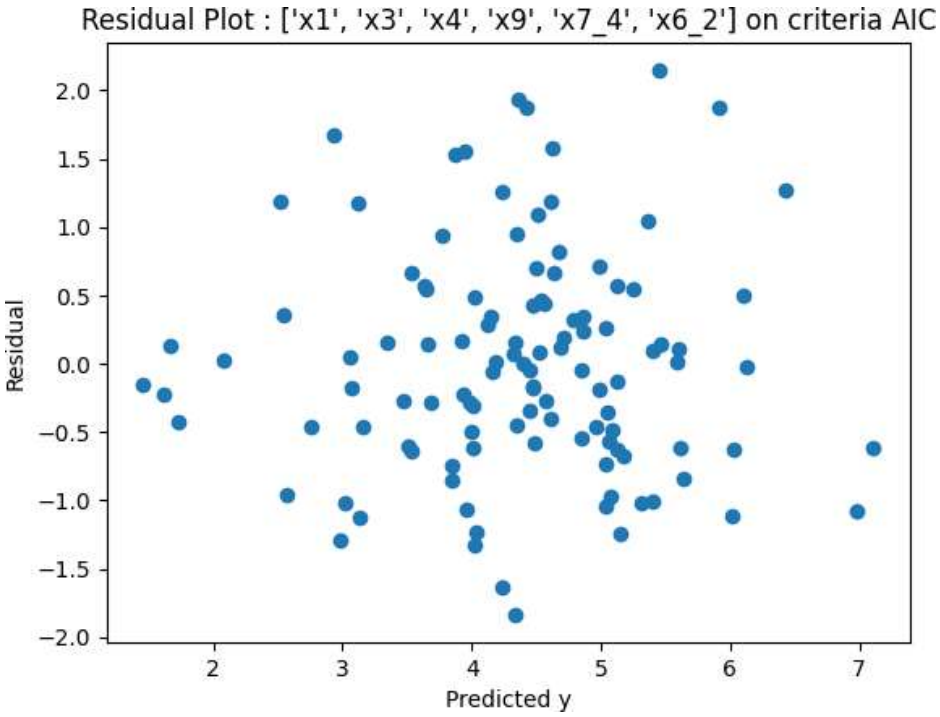


Based on the Brown-Forsythe test, we observe that there exists homoscedasticity in the residual plot.

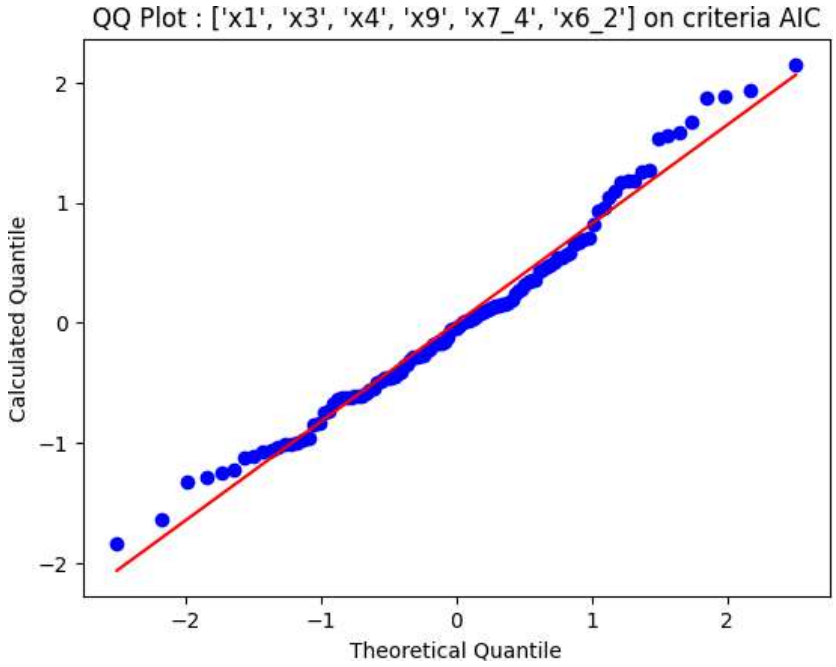


We can see from the QQ-PLOT, Residual follow noramlity asum[tion.

Resdiual and QQ-plot based on best *AIC* criteria

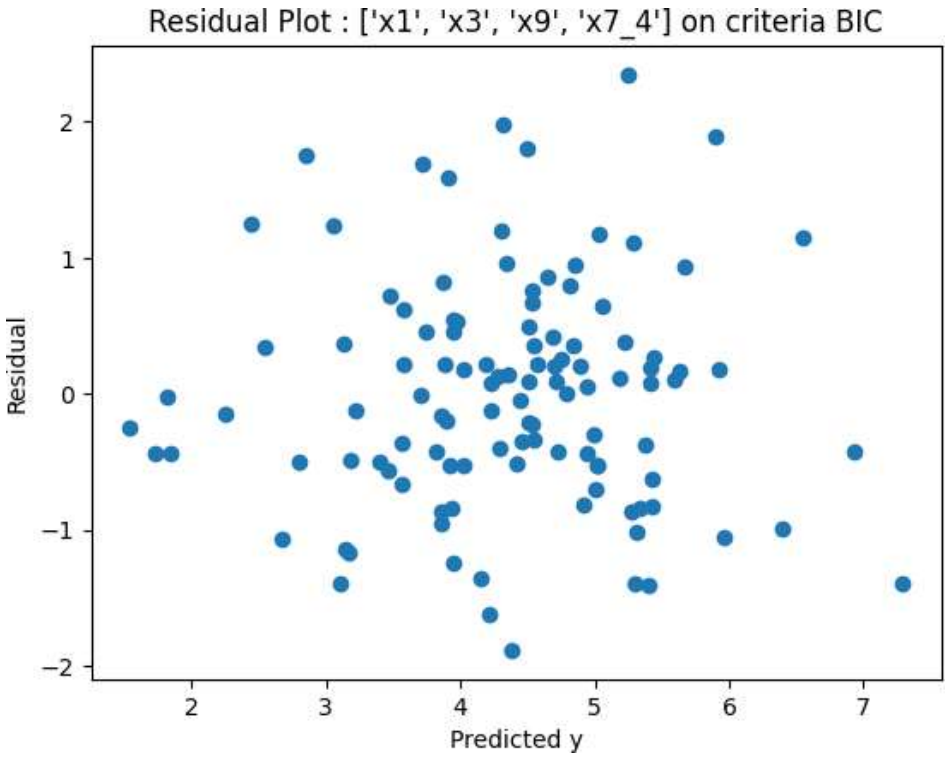


Based on the Brown-Forsythe test, we observe that there exists homoscedasticity in the residual plot.

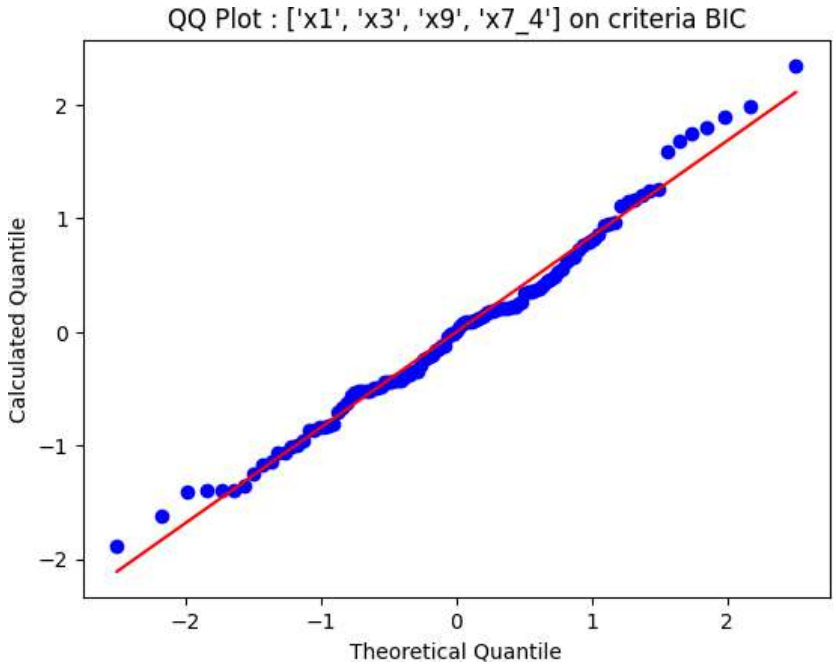


We can see a slight deviation from the normality in QQ-PLOT in right tail

Resdiual and QQ-plot based on best *BIC* criteria

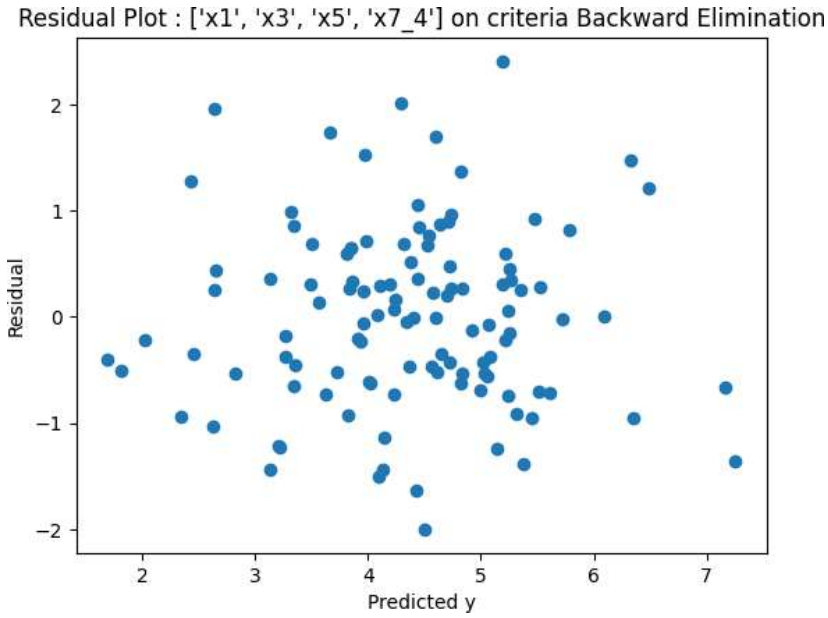


Based on the Brown-Forsythe test, we observe that there exists homoscedasticity in the residual plot.

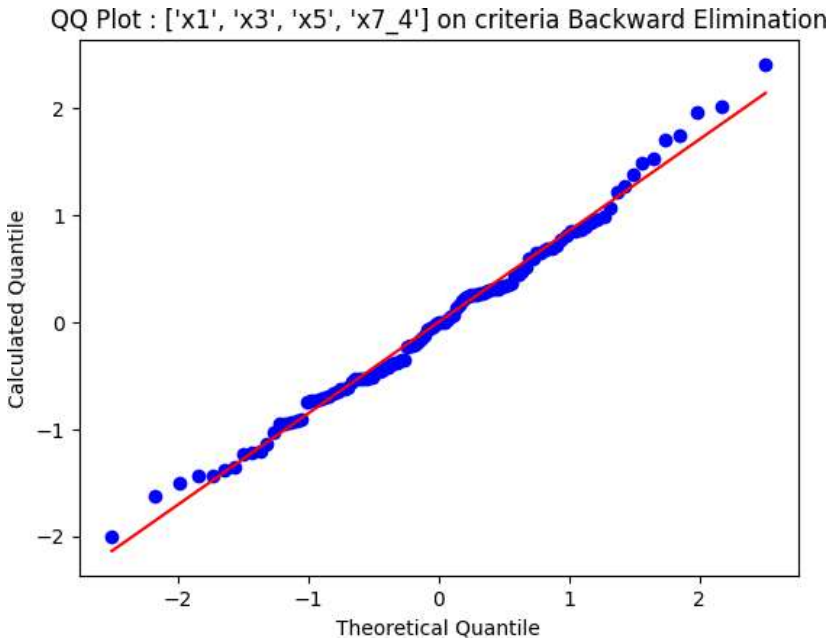


We can see a slight deviation from the normality in QQ-PLOT in it's tail.

Resdiual and QQ-plot based on Backward Elimination



Based on the Brown-Forsythe test, we observe that there exists homoscedasticity in the residual plot.



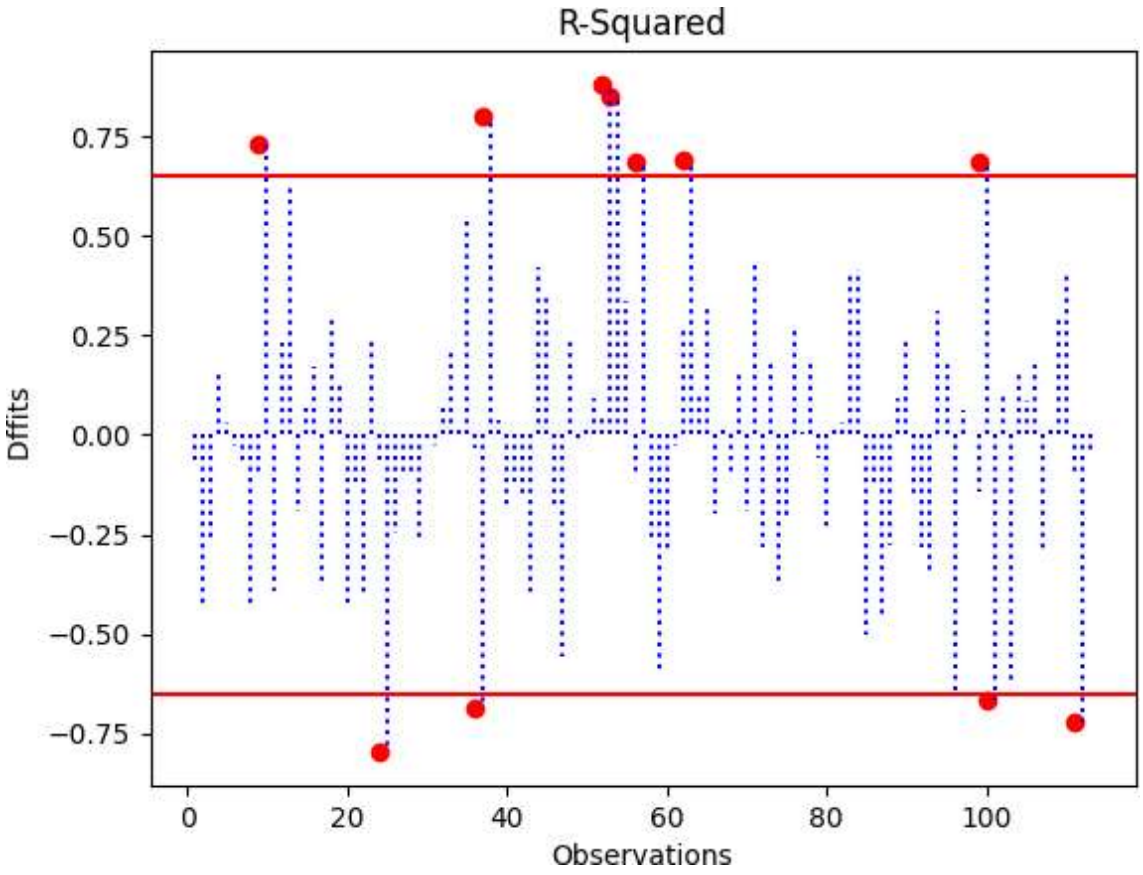
We can see a slight deviation from the normality in QQ-PLOT in it's tail.

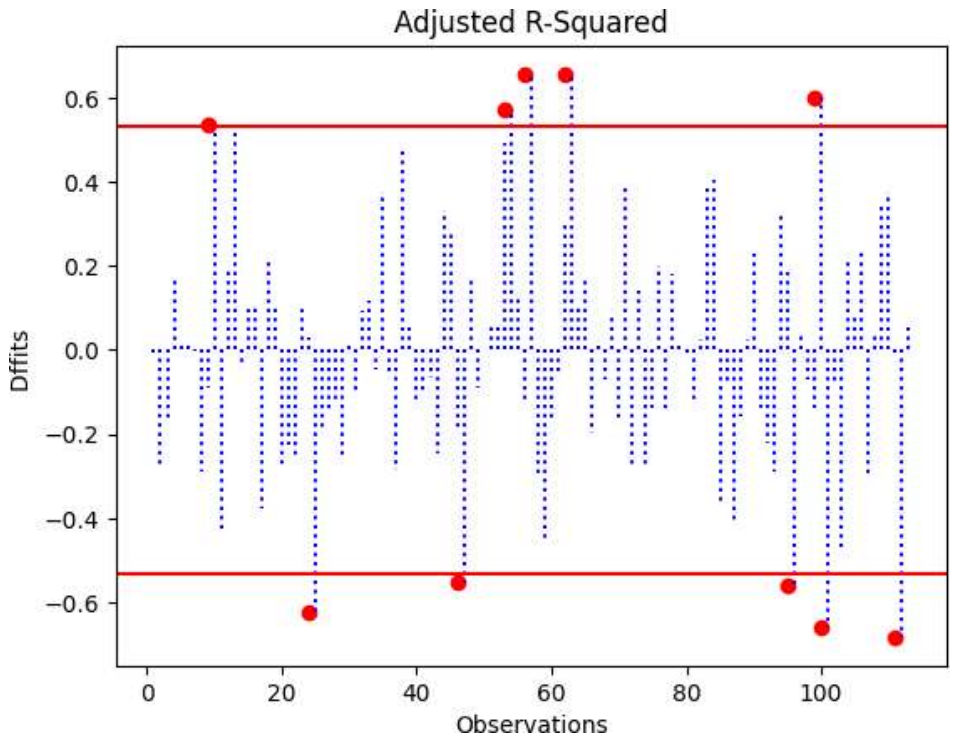
7 Influential Observation Analysis

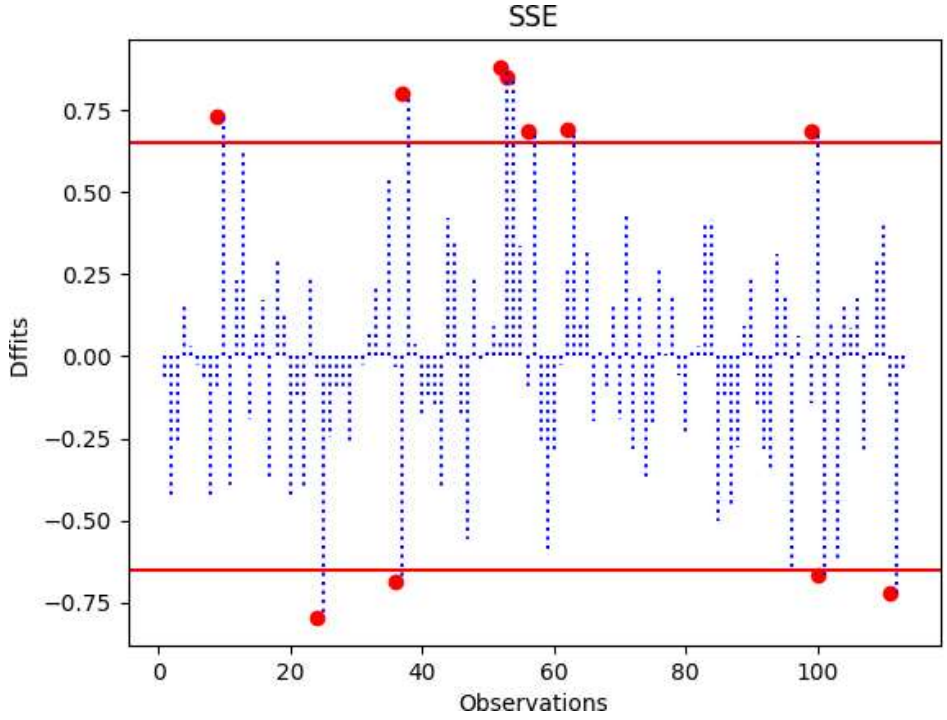
DFFITS

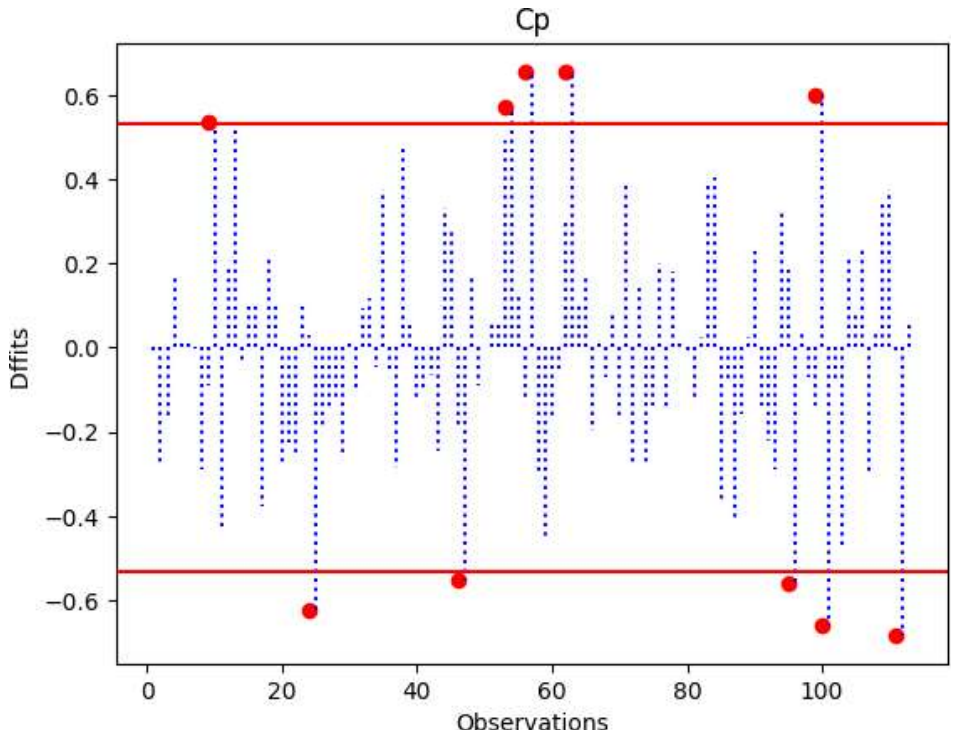
DEFFITS, or externally studentized residuals, measures the influence of each observation on the predicted values. It quantifies how much the predicted values change when that observation is omitted from the analysis.

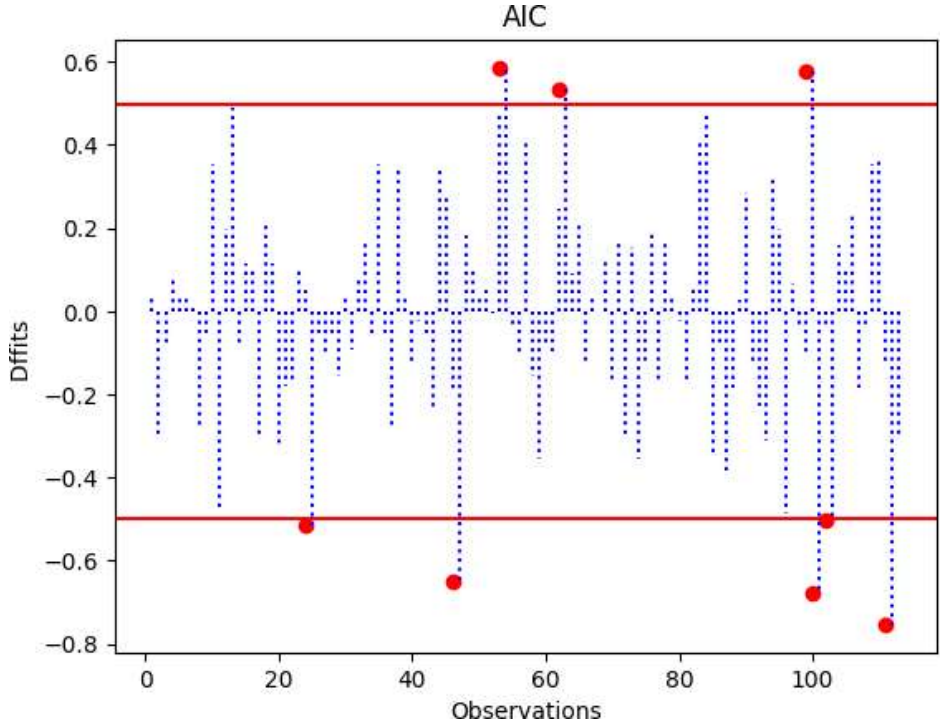
– The red dot points represent influential observations based on DFFITS for the base model according to the given criteria, while the red line indicates the threshold line.

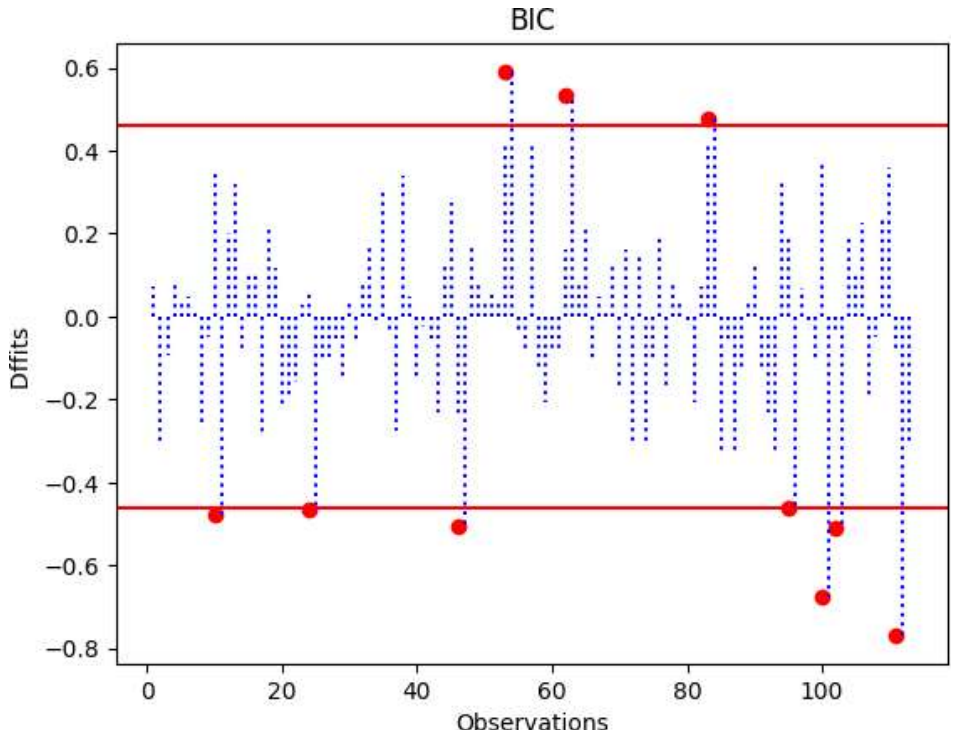


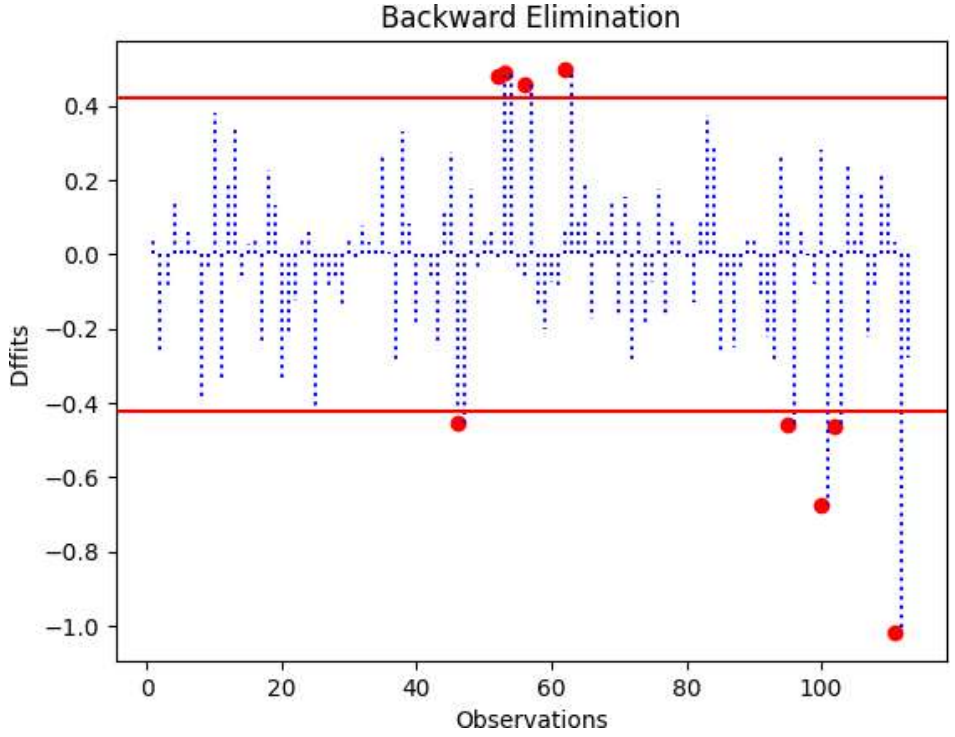

Adjusted R-Squared
Dffits
Observations
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
20
40
60
80
100


SSE
Dffits
Observations
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
0
20
40
60
80
100


Cp
Dffits
Observations
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
20
40
60
80
100


AIC
Dffits
Observations
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
0
20
40
60
80
100



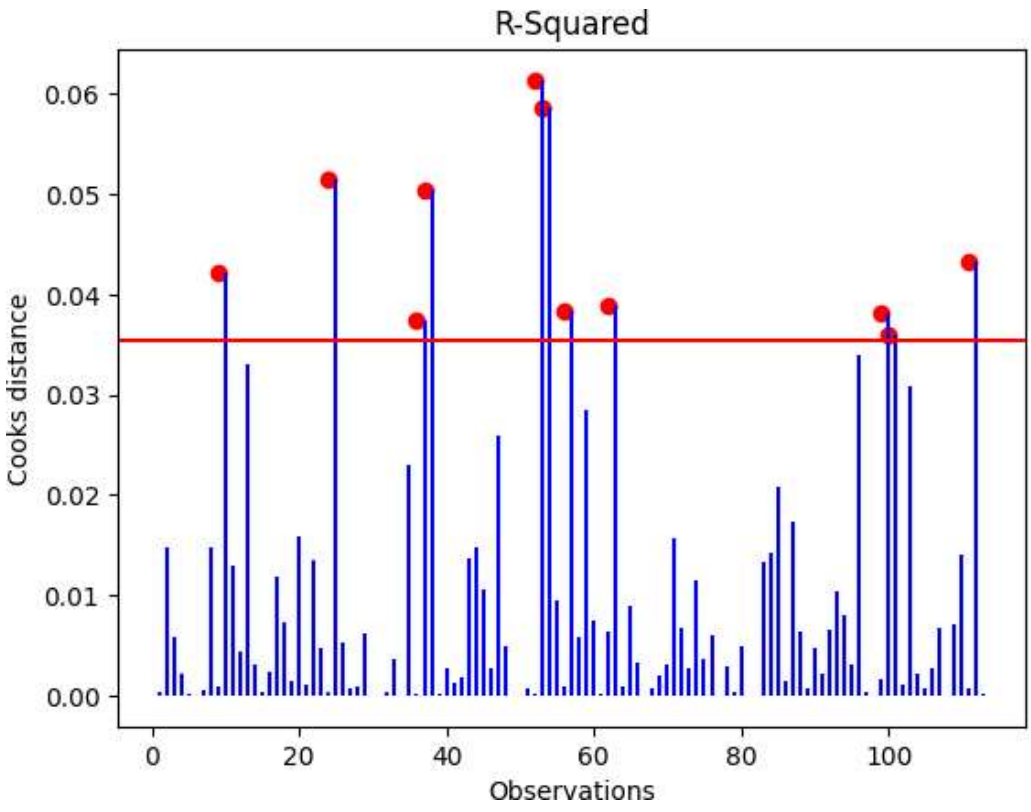


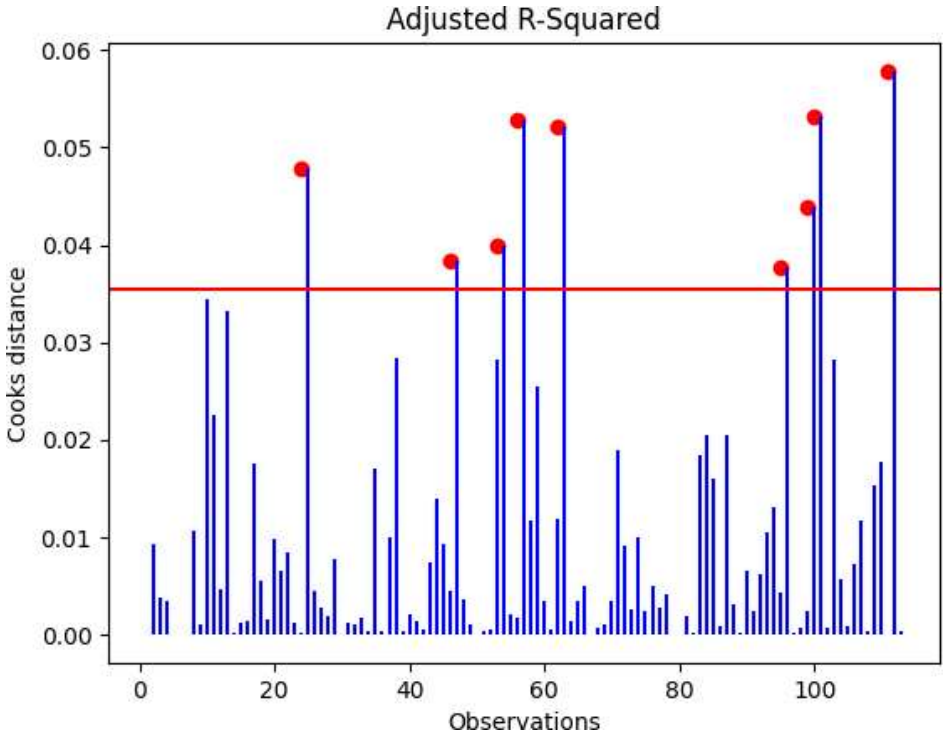
– After removing the influential observations, the adjusted R^2 increases for each model.

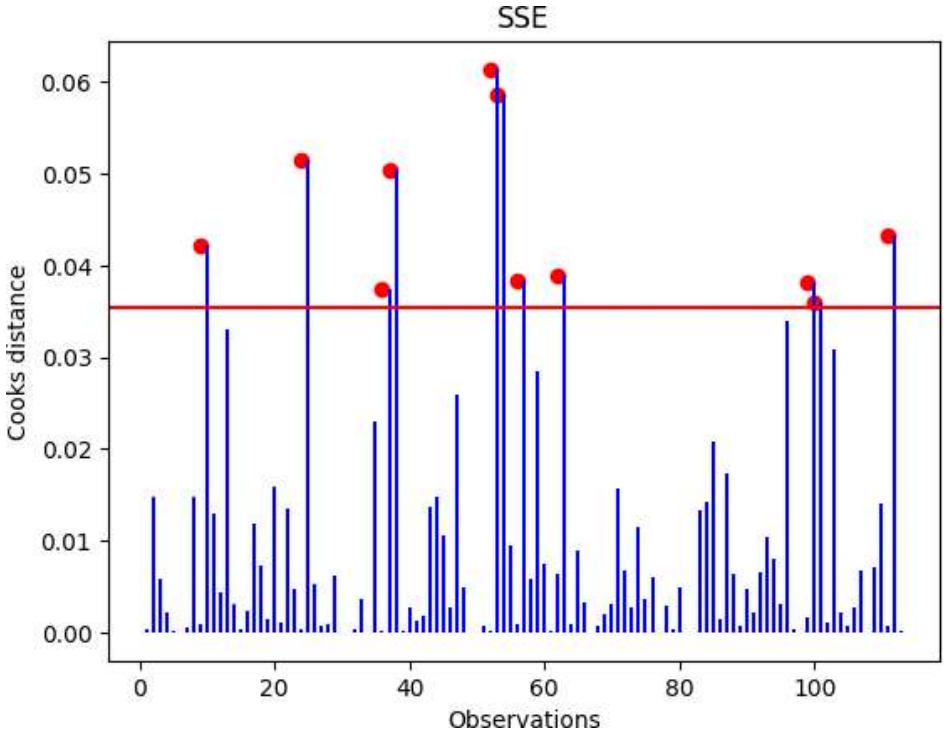
Cook's Distance

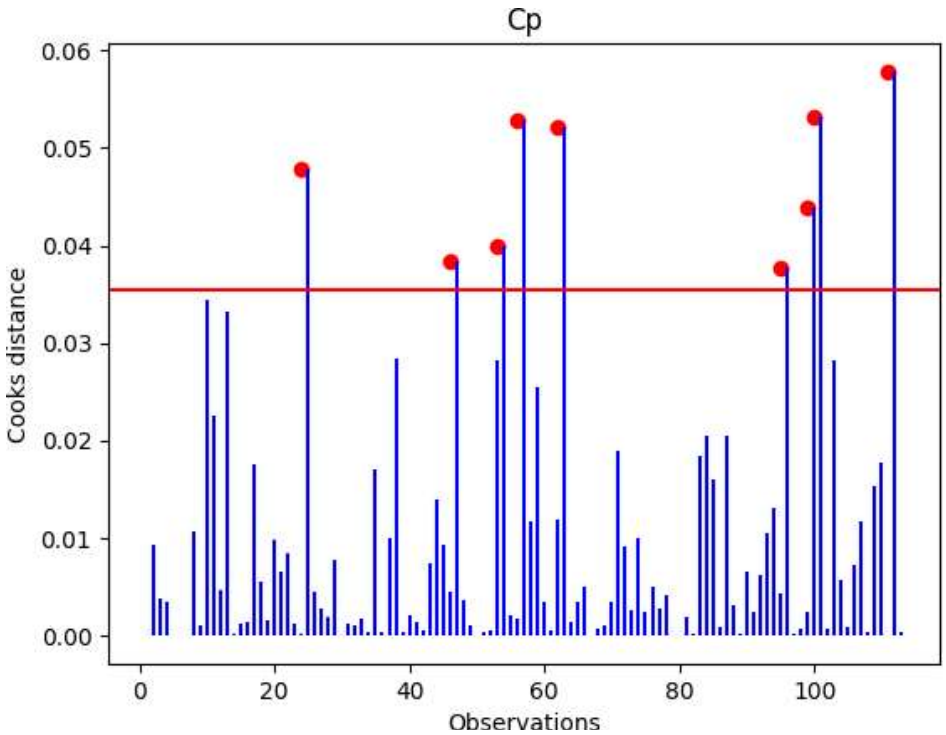
Cook's Distance identifies influential observations in regression analysis. It combines the information from the DFBETAS and DEFFITS to measure the influence of each observation on both the coefficients and the predicted values.

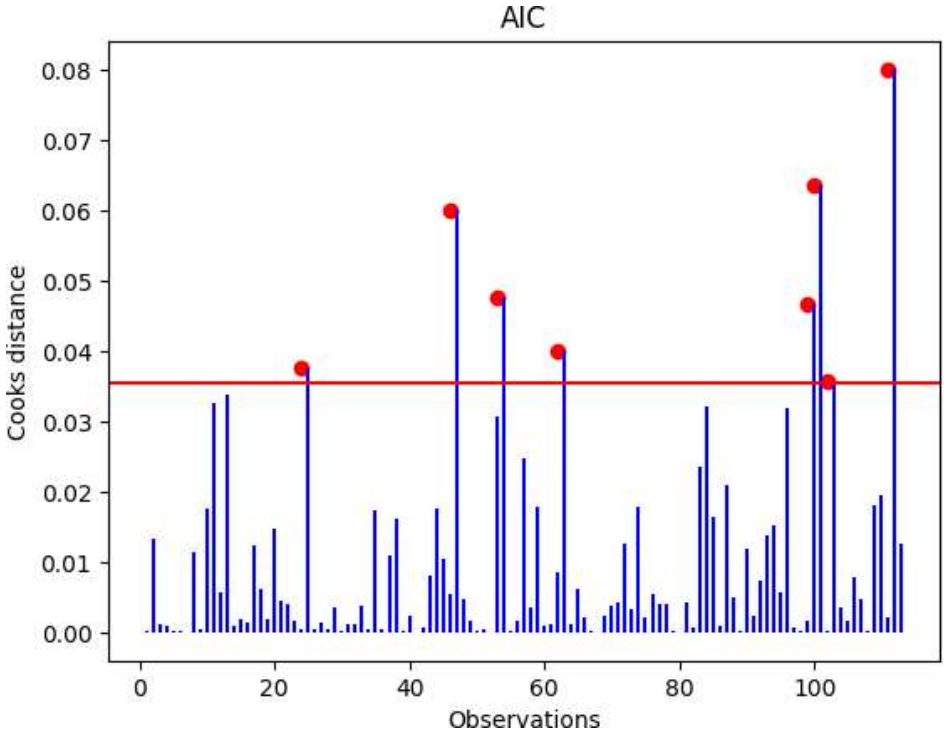
– The red dot points represent influential observations based on DFFITS for the base model according to the given criteria, while the red line indicates the threshold line.

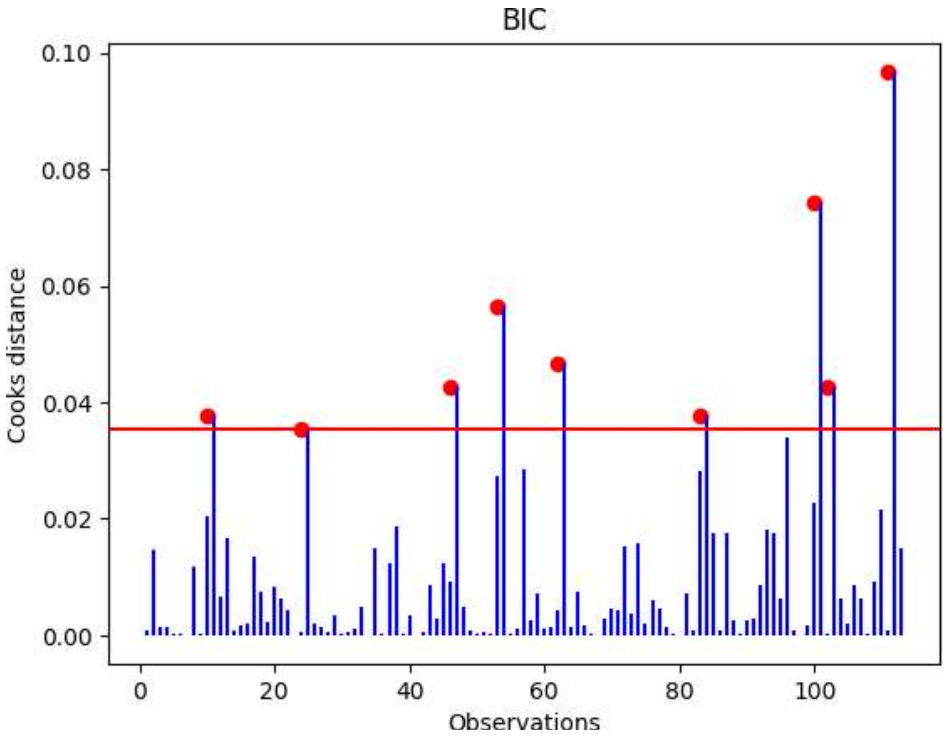


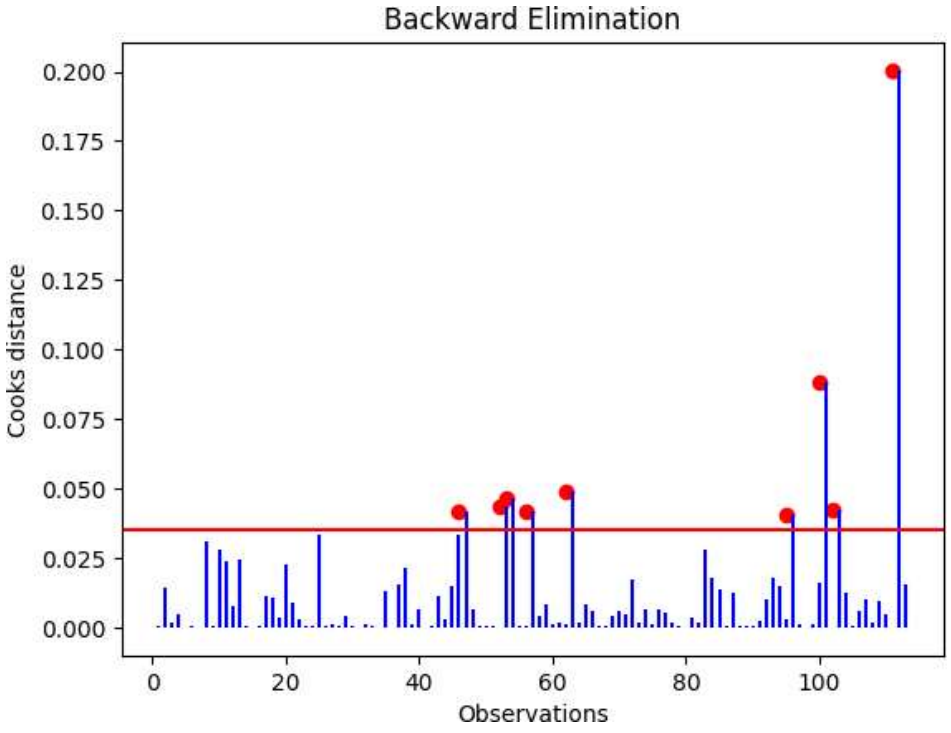

Adjusted R-Squared
Cooks distance
Observations
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0
20
40
60
80
100


SSE
Cooks distance
Observations
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0
20
40
60
80
100


Cp
Cooks distance
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
20
40
60
80
100
Observations


AIC
Cooks distance
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
20
40
60
80
100
Observations





– After removing the influential observations, the adjusted R^2 increases for each model.

8 Final Model & Conclusion

After treating influential observations, the best models and their adjusted R-Squared based on different criteria is :-

Criterion	Model	Adjusted R-Squared
R-Squared	[-5.66 + 1.59*x_1 + 0.02 * x_2 + 0.96 * x_3 + 0.48 * $x7_2$ + 0.38 * $x7_3$ + 1.14 * $x7_4$ + 0.17 * $x6_2$ + 0.000534 * $x4$ + −436.89 * x_5 + 0.000358 * x_10]	0.7257
Adjusted R-Squared	[-6.83 + 1.39*x_1 + 0.034 * x_2 + 0.86 * x_3 + 0.000534 * x_4 − 243.61 * x_5 + 0.49 * x_9 + −0.001338 * x_10 + 0.41 * $x7_2$ + 0.27 * $x7_3$ + 0.99 * $x7_4$]	0.71
SSE	[-5.66 + 1.59*x_1 + 0.02 * x_2 + 0.96 * x_3 + 0.48 * $x7_2$ + 0.38 * $x7_3$ + 1.14 * $x7_4$ + 0.17 * $x6_2$ + 0.000534 * $x4$ + −436.89 * x_5 + 0.000358 * x_10]	0.71
C_p	[-2.21 + 1.42*x_1 + 0.82 * x_3 − 26.54 * x_8 + 0.92 * $x7_4$ + 0.23 * $x6_2$ + 0.15 * x_4 + 0.013 * x_10]	0.65
AIC	[-3.504969 + 1.644007 * x1 + 0.842796 * x3 + 0.798572 * $x7_4$ + 0.255600 * $x6_2$ + 0.120861 * $x4$ + 0.247513 * $x10$]	0.65
BIC	[-3.12 + 1.59*x_1 + 0.84 * x_3 + 0.77 * $x7_4$ + 0.12 * x_4 + 0.20 * x_10]	0.67
Backward Elimination	[-3.73 + 1.68*x_1 + 0.95 * x_3 + 0.16 * x_5 + 0.80 * $x7_4$]	0.69

Table 10: Final Model

Conclusion : Based on the provided data, different models were formed and the best models based on different criterion is also shown above. The model has Adjusted R-Squared approximately equal to 0.68 i.e. it can explain 68% variance in y. Influential Observations and various techniques were also applied and Multicollinearity was also present which was treated as well.

To do's :-

1. Model Fittng (full)
2. **Box-Tidwell** is required or not
3. Normality assumption for y-variable otherwise use Box-Cox Transformation
4. Check for variable importances and then find adequate variables
5. Fit model with adequate variables and then other assumptions - **constant variance**
6. Check for **influential obervations**
7. - Check for interaction terms as well

Importing necessary libraries

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:
```
df=pd.read_excel(r"C:\Users\hp\Downloads\Semester 2 Project\project_dataset.xlsx")
```

In [ ]:
```
df.drop(columns = ["id"], inplace = True)
```

In [ ]:
```
df = df.rename(columns={"x6 " : "x6"})
```

Exploratory Data Analysis

In [ ]:
```
df
```

In [ ]:
```
df.info()
```

In [ ]:
```
df["x6"].unique()
```

Out[ ]:
```
array([2, 1], dtype=int64)
```

In [ ]:
```
df["x7"].unique()
```

Out[ ]:
```
array([4, 2, 3, 1], dtype=int64)
```

So, x6 and x7 are categorical columns.

In [ ]:
```
df.shape
```

Out[ ]:
```
(113, 11)
```

In [ ]:
```
a = list(df.columns)
a = a[:8]
a
```

Out[ ]:
```
['x1', 'x2', 'x3', 'x4', 'x5', 'x8', 'x9', 'x10']
```

In [ ]:
```
# Create a figure and axis objects
fig, axs = plt.subplots(4, 2, figsize=(10, 15))  # 4 rows, 2 columns

# Flatten the axs array to iterate over each subplot
axs = axs.flatten()

# Plot each column of the DataFrame against the 'y' column
for i, column in enumerate(a):
    if column != 'y':  # Skip the 'y' column itself
        ax = axs[i]
        ax.scatter(df[column], df['y'])
        ax.set_title(f'{column} vs y')
        ax.set_xlabel(column)
        ax.set_ylabel('y')

# Adjust layout
plt.tight_layout()

# Show the plot
plt.show()
```

In [ ]:
```
sns.pairplot(df)
```

So, after carefully visualising this pair plot we can make these conclusions

1. We can see that the variables are associated with y variable but, they are not completly linear with each other. Therefore, **Box-Tidwell** Transformation may be needed here.
2. x1, x4 and x10 are looking somewhat linear and may be beneficial in further stage of our model building columns.
3. X5 is linearly related with x8, x9 and x10.

In [ ]:
```
sns.kdeplot(df["y"], fill=True)
```

We can see that the kde plot looks very much similar to normal distribution, but we can not directly comment of the normality assum[tion of y-variable by just seeing seeing this plot. We will have to plot QQ-Plot as well.

In [ ]:
```
from scipy.stats import probplot

probplot(df["y"], dist="norm", plot = plt)
plt.show()
```

A little deviation can be seen and we will see what we have to do for the **normality** assumption.

In [ ]:
```
sns.heatmap(data = df.corr(), annot = True)
```

Conclusions :-

1. x5 can be seen highly correlated with x8, x9 and x10.
2. x6 can be been moderately correlated with x5, x8 and x9. (correlation around 0.6)
3. x8 can be seen highly correlated with x9 and x10. (x5 k saath to hai hi)
4. x9 and x10 are highly correlated.

So, multicollinearity could have been an issue here.

In [ ]:
```
sns.boxplot(df1)

plt.xlabel('Variables')
plt.ylabel('Value')
plt.title('Box-Plot')

plt.show()
```

So, we can see that **x5**, **x8** and **x9** have some outliers in them as well.

One-Hot encoding for x6 and x7

In [ ]:
```
df = pd.get_dummies(df, columns=["x7", "x6"], drop_first=True)
```

In [ ]:
```
df["x6_2"] = df["x6_2"].astype(int)
df["x7_2"] = df["x7_2"].astype(int)
df["x7_3"] = df["x7_3"].astype(int)
df["x7_4"] = df["x7_4"].astype(int)
```

In [ ]:
```
df.head()
```

Out[ ]:

	x1	x2	x3	x4	x5	x8	x9	x10	y	x7_2	x7_3	x7_4	x6_2
0	7.13	55.7	9.0	39.6	279	207	241	60.0	4.1	0	0	1	1
1	8.82	58.2	3.8	51.7	80	51	52	40.0	1.6	1	0	0	1
2	8.34	56.9	8.1	74.0	107	82	54	20.0	2.7	0	1	0	1
3	8.95	53.7	18.9	122.8	147	53	148	40.0	5.6	0	0	1	1
4	11.20	56.5	34.5	88.9	180	134	151	40.0	5.7	0	0	0	1

Box-Tidwell (Linearity Assumption)

As we can see that the varibles are not too linear with the y-variable. So, we will try to find an alpha for the variable using **Box-Tidwell transformation** and see how far we can go.

Procedure :-

```
1. Take input x and y
2. Fit a  simple linear regression model
3. Obtain \beta_1(old)
4. Fit LR Model : **y = \beta_0 + \beta_1(new) * x + \beta_2 * (x * logx)**
5. Find alpha = (\beta_2 / \beta_1(old)) + 1
```

In [ ]:
```
from sklearn.linear_model import LinearRegression
```

In [ ]:
```
import statsmodels.api as sm
from statsmodels.formula.api import ols
```

In [ ]:
```
def box_tidwell(x, y):
    x = sm.add_constant(x)
    lr1 = sm.OLS(y, x).fit()

    beta_1 = lr1.params[-1] # find beta_1

    x["xlogx"] = x.iloc[:, -1] * np.log(x.iloc[:, -1])

    lr2 = sm.OLS(y, x).fit() # Fit second model

    beta_2 = lr2.params[-1] # find beta_2

    # alpha
    alpha  = (beta_2/beta_1) + 1

    return alpha
```

In [ ]:
```
# defining columns or box-tidwell

col = df.columns
col = list(col)

cat_col = ["x6_2", "x7_2", "x7_3", "x7_4"]

for i in cat_col:
    col.remove(i)
# since they were categorical columns

col.remove("y")

col
```

Out[ ]:
```
['x1', 'x2', 'x3', 'x4', 'x5', 'x8', 'x9', 'x10']
```

In [ ]:
```
alphas = {}
for i in col:
    a = box_tidwell(pd.DataFrame(df[i]), pd.DataFrame(df["y"]))
    alphas[i] = a

print(alphas)
```

```
{'x1': -0.5253179340006215, 'x2': 696.2729262185024, 'x3': -0.09282219256014179, 'x4': 1.50455419439544, 'x5': -1.5613257815153019, 'x8': -0.849404288287531, 'x9': -1.7676528018773272, 'x10': -1.5257239771594349}
```

In [ ]:
```
plt.scatter(df["x10"], df["y"])

plt.xlabel('Variable : x10')
plt.ylabel('Value')
plt.title('Scatter Plot')

plt.show()
```

In [ ]:
```
plt.scatter((df["x10"])**(1.525), df["y"])

plt.xlabel('x10^1.525')
plt.ylabel('y')
plt.title('Scatter Plot')

plt.show()
```

x5 - log

Conclusions of Box-Tidwell :-

1. x1^(-0.525) is needed
2. log(x3) is needed
3. log(x9) is needed
4. x10^1.52 is needed
5. x4^1.5 is needed
6. x5^(-1.56) is needed
7. x8^(-0.84) is needed

Linear Model Pipeline

In [ ]:
```
data_for_mse = df.copy()
```

In [ ]:
```
x = data_for_mse.drop(columns=["y"], axis=1)
```

```
y = data_for_mse["y"]
x = sm.add_constant(x)
model = sm.OLS(y, x).fit()
y_pred = model.predict(x)
SSE = np.sum((y - y_pred)**2)
MSE = SSE / (data_for_mse.shape[0] - data_for_mse.shape[1])

print("MSE for full model is : ", MSE)
```

```
MSE for full model is :  0.8350025562952726
```

In [ ]:
```
class LinearModel:
    def __init__(self, data):
        self.data = data
        self.n = self.data.shape[0]
        self.p = self.data.shape[1]
        self.x = data.drop(columns=["y"], axis=1)
        self.y = data["y"]
        self.x = sm.add_constant(self.x)
        self.model = sm.OLS(self.y, self.x).fit()
        self.y_pred = self.model.predict(self.x)
        self.SSE = np.sum((self.y - self.y_pred)**2)
        self.SSR = np.sum((self.y_pred - np.mean(self.y))**2)
        self.SSTO = self.SSE + self.SSR
        self.dfSSE = self.n - self.p
        self.dfSSR = self.p - 1
        self.dfSSTO = self.n - 1
        self.residual = self.y - self.y_pred

    def r_squared(self):
        # Get the R-Squared
        return (1 - self.SSE/self.SSTO)

    def adjustedr_squared(self):
        # Get Adjusted R-Squared
        return (1 - (self.SSE/self.dfSSE)/(self.SSTO/self.dfSSTO))

    def Cp(self, MSE):
        # Get Cp
        return ((self.SSE/MSE) - self.n + 2 * self.p)

    def AIC(self):
        # Get AIC
        return (self.n * np.log(self.SSE/self.n) + 2 * self.p)

    def BIC(self):
        # Get BIC
        return (self.n * np.log(self.SSE/self.n) + np.log(self.n) * self.p)
```

Model Fitting (Full Model, without any changes)

In [ ]:
```
model_full = LinearModel(df)
```

In [ ]:
```
# Print the R-Squared
print(f"R-Squared on Test Data for full model: {model_full.r_squared()}")

# Print the Adjusted R-Squared
print(f"Adjusted R-Squared on Test Data for full model: {model_full.adjustedr_squared()}")

# Print the Adjusted R-Squared
print(f"Mallow's Statistic on Data for full model: {model_full.Cp(MSE)}")

# Print the AIC
print(f"AIC on Data for full model: {model_full.AIC()}")

# Print the BIC
print(f"BIC on Data for full model: {model_full.BIC()}")
```

```
R-Squared on Test Data for full model: 0.5853593752247047
Adjusted R-Squared on Test Data for full model: 0.5356025002516693
Mallow's Statistic on Data for full model: 13.0
AIC on Data for full model: -8.186808173449066
BIC on Data for full model: 27.26923346981137
```

Residual Plot

In [ ]:
```
plt.scatter(model_full.y_pred, (model_full.y - model_full.y_pred))
plt.axhline(0, color='black', linestyle='--')  # Draw a horizontal line at y=0
plt.title("Residual Plot")
plt.ylabel("Residual")
plt.xlabel("Predicted Value")
```

```
plt.show()
```

In [ ]:
```
# Print the Residual Mean
print(f"Residual Mean on Test Data for full model: {np.mean(model_full.residual)}")
```

```
Residual Mean on Test Data for full model: 1.9433815422199644e-14
```

Data Frame for Box-Tidwell

In [ ]:
```
df1 = df.copy()
```

Model Fitting (Model, with Box - Tidwell changes)

Making changes in variables x1 and x3

In [ ]:
```
df1['x1'] = df1['x1']**0.525
df1['x3'] = np.log(df1['x3'])
df1['x9'] = np.log(df1['x9'])
df1['x10'] = df1['x10']**1.52
df1['x4'] = df1['x4']**1.5
df1['x5'] = df1['x5']**(-1.56)
df1['x8'] = df1['x8']**(-0.84)
```

In [ ]:
```
df1
```

In [ ]:
```
model1 = LinearModel(df1)

# Print the R-Squared
print(f"R-Squared on Test Data for full model: {model1.r_squared()}")
```

```
R-Squared on Test Data for full model: 0.6522866508330935
```

In [ ]:
```
# Print the Adjusted R-Squared
print(f"Adjusted R-Squared on Test Data for full model: {model1.adjustedr_squared()}")
```

```
Adjusted R-Squared on Test Data for full model: 0.6105610489330646
```

Residual Plot

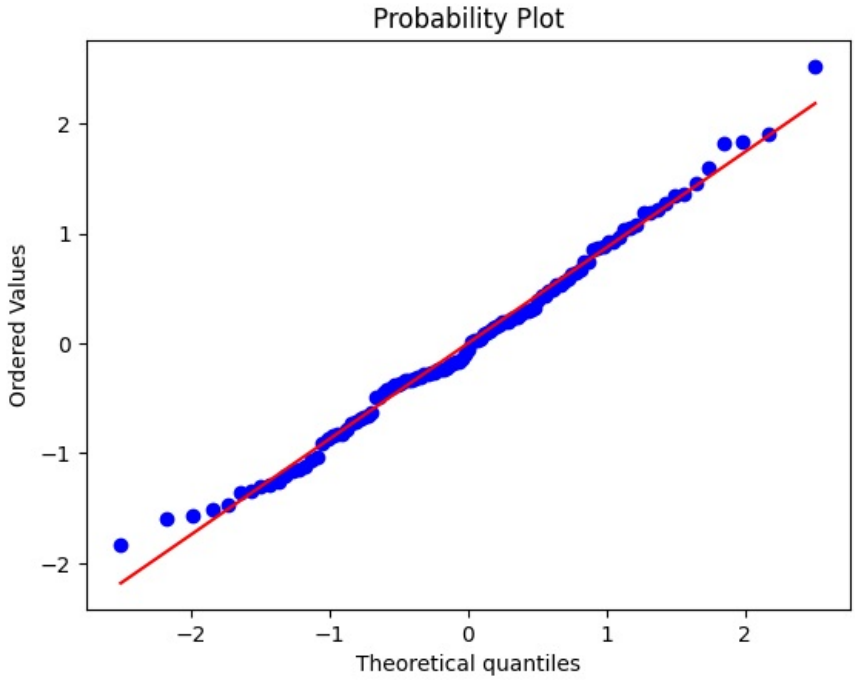
In [ ]:
```
plt.scatter(model1.y_pred, (model1.y - model1.y_pred))
plt.axhline(0, color='black', linestyle='--')  # Draw a horizontal line at y=0
plt.title("Residual Plot")
plt.ylabel("Residual")
plt.xlabel("Predicted Value")
plt.show()
```

In [ ]:
```
# Print the Residual Mean
print(f"Residual Mean on Test Data for full model: {np.mean(model1.residual)}")
```

```
Residual Mean on Test Data for full model: -2.060652533564441e-13
```

Normality Assumption

In [ ]:
```
probplot(model_full.residual, dist="norm", plot = plt)
plt.show()
```



Box-Cox Transformation

In [ ]:
```
from scipy import stats

# Apply Box-Cox transformation
transformed_data, lambda_value = stats.boxcox(df["y"])

print("Lambda value:", lambda_value)
print("Transformed data:", transformed_data)
```

Since, the lambda value for the y-variable is 1.09 which is close to 1, therefore normality assumption is satisfied.

Handling Multicollinearity

Creating function for calculating VIF

In [ ]:
```
def VIF(data):
    model_vif = LinearModel(data)
    return(1 / (1 - model_vif.r_squared()))


def vif_values(data):
    data = data.drop(columns = ["y"], axis = 1)
    VIFs = {}
    for i in data.columns:
        data1 = data.rename(columns = {i : "y"})
        VIFs[i] = VIF(data1)

    return VIFs
```

In [ ]:
```
vif_values(df1)
```

Now, after variable transformations, x8 has to be treated, else we can see that handling the multicollinearity has automatically sorted out.

Conclusions (before variable transformation) for these VIF values :-

1. x5, x8 and x9 have high VIF which can also be seen in the pairplot as well.
2. x5 and x8 are highly correlated with each other and they **have** to treated.

Creating new dataframe after treating multicollinearity as well -

In [ ]:
```
df2 = df1.copy()
```

In [ ]:
```
df2.drop(columns=["x8"], axis=1, inplace=True)
```

In [ ]:
```
vif_values(df2)
```

So, now everything is sorted.

In [ ]:
```
model2 = LinearModel(df2)

# Print the R-Squared
print(f"R-Squared on Test Data for full model: {model2.r_squared()}")
```

```
R-Squared on Test Data for full model: 0.6497742904484571
```

In [ ]:
```
# Print the Adjusted R-Squared
print(f"Adjusted R-Squared on Test Data for full model: {model2.adjustedr_squared()}")
```

```
Adjusted R-Squared on Test Data for full model: 0.611630896338883
```

In [ ]:
```
plt.scatter(model2.y_pred, (model2.y - model2.y_pred))
plt.axhline(0, color='black', linestyle='--')  # Draw a horizontal line at y=0
plt.title("Residual Plot")
plt.ylabel("Residual")
plt.xlabel("Predicted Value")
plt.show()
```

In [ ]:
```
# Print the Residual Mean
print(f"Residual Mean on Test Data for model: {np.mean(model2.residual)}")
```

```
Residual Mean on Test Data for model: 2.929750837213044e-13
```

Single models

In [ ]:
```
# creating the models based on each single x variables
for var in list(df.columns):

    X4 = LinearModel(df[[var,'y']])
    c = np.round(X4.model.params[0] , 4)
    m= np.round(X4.model.params[1] , 4)
    # Generate x values

    x_values = [np.min(df[var]),np.max(df[var])]  # Adjust range as needed

    # Calculate corresponding y values
    y_values = [m * x + c for x in x_values]

    # Plot the line
    plt.plot(x_values, y_values, label='y = {}x + {}'.format(m, c) , color = 'r')


    plt.scatter(df[var], df["y"])
    plt.xlabel(f'Variable :{var}')
    plt.ylabel('Y')
    plt.title(f'Scatter Plot y = {m}*{var} + {c}')

    plt.show()

    print(X4.model.summary())
```

Best Model Selection

1. Using R-Squared
2. Using Adjusted R-Squared
3. Using Mallow's Statistic
4. Using Sequential Selection Techniques (Backward Elimination, Forward Selection)
5. Using AIC-BIC

In [ ]:
```
x = df2.drop(columns=['y'], axis =1)
columns = list(x.columns)

columns
```

Out[ ]:
```
['x1', 'x2', 'x3', 'x4', 'x5', 'x9', 'x10', 'x7_2', 'x7_3', 'x7_4', 'x6_2']
```

Creating Data Frame for all the metrics

In [ ]:
```
import itertools

# to create combinations of variables
```

In [ ]:
```
Variables_in_Model = []
p = []
SSE_p = []
R_Squared = []
```

```
Adjusted_R_Squared = []
Cp = []
AIC = []
BIC = []
```

In [ ]:
```
for i in range(1, 13):
    for predictors in itertools.combinations(columns, i):
        x_subset = x[list(predictors)]
        x_subset['y'] = df2['y']
        eval_model = LinearModel(x_subset)
        Variables_in_Model.append(list(predictors))
        p.append(i)
        SSE_p.append(eval_model.SSE)
        R_Squared.append(eval_model.r_squared())
        Adjusted_R_Squared.append(eval_model.adjustedr_squared())
        Cp.append(eval_model.Cp(MSE))
        AIC.append(eval_model.AIC())
        BIC.append(eval_model.BIC())
```

In [ ]:
```
eval_metrics = pd.DataFrame({"Variables in Model" : Variables_in_Model, "p" : p, "SSE" : SSE_p, "R_Squared" : R_
```

In [ ]:
```
eval_metrics.shape
```

Out[ ]:
```
(2047, 8)
```

Since, we had 11 variable and used atleast 1 variable in the model, therefore total 2^11 - 1 = 2048 - 1 = 2047 models are there.

In [ ]:
```
eval_metrics.to_csv(r"C:\Users\hp\Downloads\Semester 2 Project\model_metrics.csv", sep=',', index=False,header=
```

Model Selection

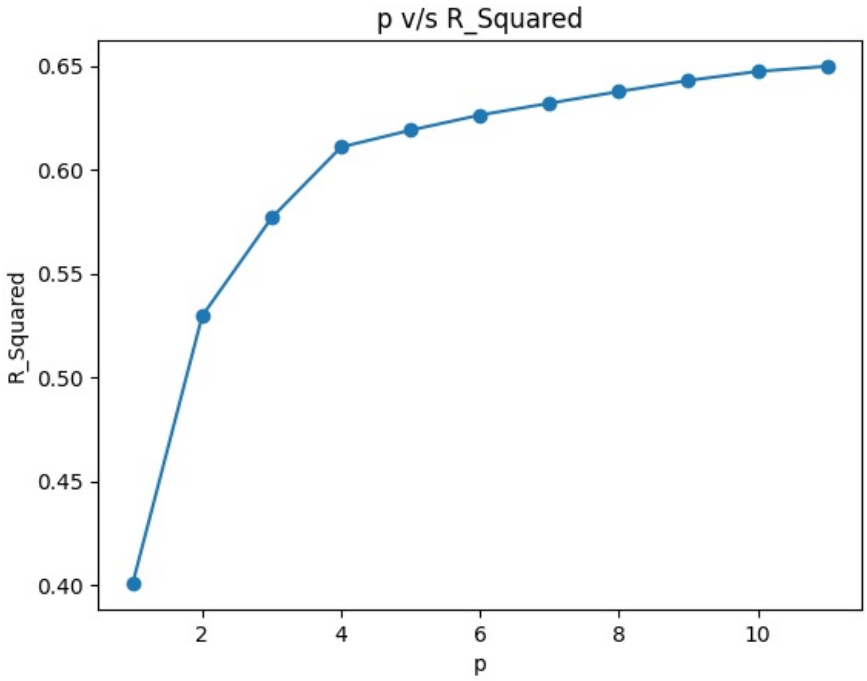
In [ ]:
```
best_models = pd.DataFrame()
```

1. Using R-Squared

In [ ]:
```
p_R_Squared = (eval_metrics.groupby('p')["R_Squared"].max()).reset_index()

index = eval_metrics.loc[eval_metrics['R_Squared'] == eval_metrics['R_Squared'].max()].index[0]

d = (eval_metrics.iloc[index, :]).reset_index()
d = d.T
d.columns = d.iloc[0]
d = d.drop('index')
d["Crietria"] = "R-Squared"

best_models = pd.concat([best_models, d], ignore_index=True)
```

In [ ]:
```
best_rsquared_index = eval_metrics.groupby('p')['R_Squared'].idxmax()
best_rsquared = eval_metrics.loc[best_rsquared_index, ["p", "Variables in Model", "R_Squared"]]

best_rsquared.to_csv(r"C:\Users\hp\Downloads\Semester 2 Project\best_rsquared.csv", sep=',', index=False, heade

best_rsquared
```

In [ ]:
```
plt.scatter(p_R_Squared['p'], p_R_Squared['R_Squared'])
plt.plot(p_R_Squared['p'], p_R_Squared['R_Squared'])

plt.xlabel('p')
plt.ylabel('R_Squared')
plt.title('p v/s R_Squared')

plt.show()
```



Model having all the columns has the best R-Squared.

2. Using Adjusted R-Squared

In [ ]:
```
p_Adjusted_R_Squared = (eval_metrics.groupby('p')["Adjusted R_Squared"].max()).reset_index()

index = eval_metrics.loc[eval_metrics['Adjusted R_Squared'] == eval_metrics['Adjusted R_Squared'].max()].index[

d = (eval_metrics.iloc[index, :]).reset_index()
d = d.T
d.columns = d.iloc[0]
d = d.drop('index')
d["Crietria"] = "Adjusted R-Squared"
d

best_models = pd.concat([best_models, d], ignore_index=True)
```

In [ ]:
```
best_Adjusted_rsquared_index = eval_metrics.groupby('p')['Adjusted R_Squared'].idxmax()
best_Adjusted_rsquared = eval_metrics.loc[best_Adjusted_rsquared_index, ["p", "Variables in Model", "Adjusted R_

best_Adjusted_rsquared.to_csv(r"C:\Users\hp\Downloads\Semester 2 Project\best_Adjusted_rsquared.csv", sep=',', i

best_Adjusted_rsquared
```

In [ ]:
```
plt.scatter(p_Adjusted_R_Squared['p'], p_Adjusted_R_Squared['Adjusted R_Squared'])
plt.plot(p_Adjusted_R_Squared['p'], p_Adjusted_R_Squared['Adjusted R_Squared'])

plt.xlabel('p')
plt.ylabel('Adjusted R_Squared')
plt.title('p v/s Adjusted R_Squared')

plt.show()
```

So, according to Adjusted R-Squared, the model having **8** variables, **[x1, x3, x4, x10, x7_2, x7_3, x7_4, x6_2]** perfroms best for the data.

3. Using SSE

In [ ]:
```
p_SSE = (eval_metrics.groupby('p')["SSE"].min()).reset_index()

index = eval_metrics.loc[eval_metrics['SSE'] == eval_metrics['SSE'].min()].index[0]

d = (eval_metrics.iloc[index, :]).reset_index()
d = d.T
d.columns = d.iloc[0]
d = d.drop('index')
d["Crietria"] = "SSE"
d

best_models = pd.concat([best_models, d], ignore_index=True)
```

In [ ]:
```
best_sse_index = eval_metrics.groupby('p')['SSE'].idxmin()
best_sse = eval_metrics.loc[best_sse_index, ["p", "Variables in Model", "SSE"]]

best_sse.to_csv(r"C:\Users\hp\Downloads\Semester 2 Project\best_sse.csv", sep=',', index=False, header=True)

best_sse
```

Model having all the columns has the best SSE.

In [ ]:
```
plt.plot(p_SSE['p'], p_SSE['SSE'])
plt.scatter(p_SSE['p'], p_SSE['SSE'])

plt.xlabel('p')
plt.ylabel('SSE')
plt.title('p v/s SSE')

plt.show()
```

4. Using Mallow's Statistic (Cp)

In [ ]:
```
p_Cp = eval_metrics[["p", "Cp", "Variables in Model"]]
```

In [ ]:
```
p_Cp["p-Cp"] = abs(p_Cp["p"] - p_Cp["Cp"])
```

In [ ]:
```
index = p_Cp.loc[p_Cp['p-Cp'] == p_Cp['p-Cp'].min()].index[0]

d = (eval_metrics.iloc[index, :]).reset_index()
d = d.T
d.columns = d.iloc[0]
d = d.drop('index')
d["Crietria"] = "Cp"
d

best_models = pd.concat([best_models, d], ignore_index=True)
```

In [ ]:
```
best_cp = p_Cp.groupby('p')['p-Cp'].idxmin()
best_cp = p_Cp.loc[best_cp, ["p", "Variables in Model", "Cp"]]

best_cp.to_csv(r"C:\Users\hp\Downloads\Semester 2 Project\best_cp.csv", sep=',', index=False, header=True)

best_cp
```

In [ ]:
```
p_Cp_ = (p_Cp.groupby('p')["p-Cp"].min()).reset_index()
plt.scatter(p_Cp_['p'], p_Cp_['p-Cp'])
plt.plot(p_Cp_['p'], p_Cp_['p-Cp'])

plt.xlabel('p')
plt.ylabel('Cp')
plt.title('p v/s Cp')

plt.show()
```

Best model using Cp criterian is having **4** variables viz **[x1, x3, x4, x10]**

5. AIC

In [ ]:
```
p_AIC = (eval_metrics.groupby('p')["AIC"].min()).reset_index()

index = eval_metrics.loc[eval_metrics['AIC'] == eval_metrics['AIC'].min()].index[0]

d = (eval_metrics.iloc[index, :]).reset_index()
d = d.T
d.columns = d.iloc[0]
d = d.drop('index')
d["Crietria"] = "AIC"
d

best_models = pd.concat([best_models, d], ignore_index=True)
```

In [ ]:
```
best_AIC = eval_metrics.groupby('p')['AIC'].idxmin()
best_AIC = eval_metrics.loc[best_AIC, ["p", "Variables in Model", "AIC"]]

best_AIC.to_csv(r"C:\Users\hp\Downloads\Semester 2 Project\best_AIC.csv", sep=',', index=False, header=True)

best_AIC
```

In [ ]:
```
plt.scatter(p_AIC['p'], p_AIC['AIC'])
```

```
plt.plot(p_AIC['p'], p_AIC['AIC'])

plt.xlabel('p')
plt.ylabel('AIC')
plt.title('p v/s AIC')

plt.show()
```

Best model using AIC criterian is having **4** variables viz **[x1, x3, x7_4, x10]**

6. BIC

In [ ]:
```
p_BIC = (eval_metrics.groupby('p')["BIC"].min()).reset_index()

index = eval_metrics.loc[eval_metrics['BIC'] == eval_metrics['BIC'].min()].index[0]

d = (eval_metrics.iloc[index, :]).reset_index()
d = d.T
d.columns = d.iloc[0]
d = d.drop('index')
d["Crietria"] = "BIC"
d

best_models = pd.concat([best_models, d], ignore_index=True)
```

In [ ]:
```
best_BIC = eval_metrics.groupby('p')['BIC'].idxmin()
best_BIC = eval_metrics.loc[best_BIC, ["p", "Variables in Model", "BIC"]]

best_BIC.to_csv(r"C:\Users\hp\Downloads\Semester 2 Project\best_BIC.csv", sep=',', index=False, header=True)

best_BIC
```

In [ ]:
```
plt.scatter(p_BIC['p'], p_BIC['BIC'])
plt.plot(p_BIC['p'], p_BIC['BIC'])

plt.xlabel('p')
plt.ylabel('BIC')
plt.title('p v/s BIC')

plt.show()
```

Best model using BIC criterian is having **3** variables viz **[x1, x3, x7_4]**

7. Using Backward Elimination Technique

In [ ]:
```
model = sm.OLS(y, x).fit()
# Perform backward elimination

while True:
    # Extract p-values for each feature
    p_values = model.pvalues

    # Find the feature with the highest p-value
    feature_to_remove = p_values.idxmax()

    # Check if the highest p-value exceeds a significance level (e.g., 0.05)
    if p_values[feature_to_remove] > 0.05:
        # Step 4e: Remove the feature from X and refit the model
        x = x.drop(feature_to_remove, axis=1)
        model = sm.OLS(y, x).fit()
    else:
        break  # Exit the loop when no feature has a p-value above the significance level

# Step 5: Final model with selected features
final_model = model
```

In [ ]:
```
final_model.summary()
```

Therefore, by backward elimination technique, the best model selected is having four variables **[x1, x3, x5, x7_4]**

In [ ]:
```
bm = LinearModel(df2[["x1", "x3", "x5", "x7_4", "y"]])
```

In [ ]:
```
d = pd.DataFrame()
d.loc[0, "Variables in Model"] = 3
d.loc[0, "p"] = int(4)
d.loc[0, "SSE"] = bm.SSE
d.loc[0, "R_Squared"] = bm.r_squared()
d.loc[0, "Adjusted R_Squared"] = bm.adjustedr_squared()
```

```
d.loc[0, "Cp"] = bm.Cp(MSE)
d.loc[0, "AIC"] = bm.AIC()
d.loc[0, "BIC"] = bm.BIC()
d.loc[0, "Crietria"] = "Backward Elimination"

best_models = pd.concat([best_models, d], ignore_index=True)
```

In [ ]:
```
best_models.at[6, "Variables in Model"] = ['x1', 'x3', "x5", "x7_4"]
```

In [ ]:
```
best_models = best_models[['Crietria', 'Variables in Model', 'p', 'SSE', 'R_Squared', 'Adjusted R_Squared',
       'Cp', 'AIC', 'BIC']]
```

In [ ]:
```
best_models
```

In [ ]:
```
best_models.to_csv(r"C:\Users\hp\Downloads\Semester 2 Project\best_models.csv", sep=',', index=False,header=Tru
```

Residual analysis

- Normality
- brown forsyth for heterosedasticity check

Influential observation analysis

- DFBETAS
- DEFFITS
- Cook's

In [ ]:
```
criteria = list(best_models["Crietria"])
```

In [ ]:
```
count = 0
for i in best_models['Variables in Model']:
    x = list(i)
    var = x + ['y']
    model = LinearModel(df2[var])

    plt.scatter(model.y_pred, model.residual)
    plt.xlabel("Predicted y")
    plt.ylabel('Residual')
    plt.title(f'Residual Plot : {x} on criteria {criteria[count]}')

    plt.show()
    probplot(model.residual, dist="norm", plot = plt)
    plt.xlabel("Theoretical Quantile")
    plt.ylabel('Calculated Quantile')
    plt.title(f'QQ Plot : {x} on criteria {criteria[count]}')
    plt.show()
    count += 1
```

In [ ]:
```
# Split the array into two arrays of 50 elements each
from scipy.stats import t
def brown_forsyth(res_trans):
    n = 113
    res_1 = np.array(res_trans)[0:56]
    res_2 = np.array(res_trans)[56 : n]
    e1_tilda = np.median(res_1)
    e2_tilda = np.median(res_2)

    abs_dev_1 = np.abs(res_1 - e1_tilda)
    abs_dev_2 = np.abs(res_2 - e2_tilda)

    d1_bar = np.mean(abs_dev_1)
    d2_bar = np.mean(abs_dev_2)

    s2 = (np.sum((abs_dev_1 - d1_bar)**2) + np.sum((abs_dev_2 - d2_bar)**2))/(n-2)
    ts = (d1_bar - d2_bar)/((s2*(1/56 + (1/(57))))**0.5)
    print(ts)
    alpha=0.05
    df=n-2

    t_tab = t.ppf(1 - alpha/2, df)
    if(abs(ts)>t_tab):
      print("There is heteroscedasticity")
    else:
      print("There is homoscedasticity.")
```

In [ ]:
```
j = 0
for i in best_models['Variables in Model']:
    x = list(i)
    var = x + ['y']
    model = LinearModel(df2[var])
```

```
    residual = model.residual
    print(f'For criteria {criteria[j]} is')
    brown_forsyth(residual)
    j+=1
```

Influential Observation Analysis

In [ ]:
```
from statsmodels.stats.outliers_influence import OLSInfluence
```

In [ ]:
```
def cooks_distance(X):
    # Assuming you have your independent variables in X and dependent variable in y
    n = 113
    # Fit the regression model
    y = df2['y']
    X = df2[X]
    X = sm.add_constant(X)  # Add a constant term if needed
    model = sm.OLS(y, X).fit()

    # Get the OLSInfluence object
    influence = OLSInfluence(model)

    # Calculate Cook's distance
    cooks_d = influence.cooks_distance
    dffits = influence.dffits

    threshold_cooks = 4/n

    # Print Cook's distance for influencial observation

    influencial_cooks = []

    influencial_dffits = []

    print("Index for influential observation using cooks distance:")

    for i, cook_d in enumerate(list(cooks_d[0])):
        if(cook_d > threshold_cooks):
            print(f"Observation index {i}")
            influencial_cooks.append((i,cook_d))

    print("Index for influential observation using dffits:")
    print(f"threshold dffits {dffits[1]}")

    for j, fits in enumerate(list(dffits[0])):
        if(abs(fits) > dffits[1]):
            print(f"Observation index {j}")
            influencial_dffits.append((j, fits))


    # return value tuple of two tuples : 1st for cooks distance ( 1st list of cooks distance , list of tuples o
    #                                  : 2nd for dffits ( ist list of dffit value , list of tuples of influencia
    return (((list(cooks_d[0]),influencial_cooks), (list(dffits[0]) , influencial_dffits)))
```

In [ ]:
```
cooks_distances = {}
count = 0
for i in best_models['Variables in Model']:
   print(f"influential observations in {criteria[count]}")
   cooks_distances[f'{criteria[count]}']= cooks_distance(i)[0]
   count += 1
```

In [ ]:
```
dffits_dict = {}
count = 0
for j in best_models['Variables in Model']:
   print(f"influential observations in {criteria[count]}")
   dffits_dict[f'{criteria[count]}']= cooks_distance(j)[1]
   count += 1
```

In [ ]:
```
best_models['Crietria'][1]
```

Out[ ]:
```
'Adjusted R-Squared'
```

In [ ]:
```
# creating list of index and cooks distance of influential observations

for k in best_models['Crietria']:
    f = []
    s = []
    for i, j in cooks_distances[k][1]:
        f.append(i)
        s.append(j)
    for x, y in zip(np.arange(start=1, stop=114), cooks_distances[k][0]):
```

```
        plt.vlines(x, ymin=0, ymax=y, color='b', linestyle='-')  # Draw vertical lines from each point to x-axi

    plt.axhline(4/113 , color = "r")
    plt.scatter(f,s , color = "r" )
    plt.title(f'{k}')
    plt.ylabel('Cooks distance')
    plt.xlabel('Observations')
    plt.show()
```

In [ ]:
```
# creating list of index and cooks distance of influential observations
thr = [0.6517 , 0.6240, 0.6517, 0.4608, 0.4977, 0.4207 , 0.4207]
count = 0
for k in best_models['Crietria']:
    f = []
    s = []
    for i, j in dffits_dict[k][1]:
        f.append(i)
        s.append(j)
    for x, y in zip(np.arange(start=1, stop=114), dffits_dict[k][0]):
        plt.vlines(x, ymin=0, ymax=y, color='b', linestyle=':')  # Draw vertical lines from each point to x-axi

    plt.axhline(thr[count], color = "r")
    plt.axhline(-thr[count], color = "r")
    plt.scatter(f,s , color = "r" )
    plt.title(f'{k}')
    plt.ylabel('Dffits ')
    plt.xlabel('Observations')
    plt.show()
    count = count +1
```

In [ ]:
```
# Assuming you have your independent variables in X and dependent variable in y
X = df2[['x1', 'x3', 'x9', 'x7_4']]
y = df2['y']
# Fit the regression model
X = sm.add_constant(X)  # Add a constant term if needed
model = sm.OLS(y, X).fit()

# Get the OLSInfluence object
influence = OLSInfluence(model)

# Calculate DFFITS
dffits = influence.dffits  # DFFITS for all observations
# or you can use influence.dffits[0, :] for individual observations

# Calculate DFBETAS
dfbetas = influence.dfbetas  # DFBETAS for all observations

# Print DFFITS and DFBETAS for each observation
# for i, (dfb, dffit) in enumerate(zip(dfbetas, dffits)):
#     print(f"Observation {i}: DFFITS = {dffit}, DFBETAS = {dfb}")

dffits[1]
list(dffits[0])
```

Model Fitting by excluding the influential oberservations

In [ ]:
```
exclude_indexes = [cooks_distances['Adjusted R-Squared'][1][i][0] for i in range(len(cooks_distances['Adjusted R

result = df2[~df2.index.isin(exclude_indexes)]

print(LinearModel(result).adjustedr_squared())
```

```
0.7068610915681984
```

In [ ]:
```
exclude_indexes = [cooks_distances['Adjusted R-Squared'][1][i][0] for i in range(len(cooks_distances['Adjusted R

result = df2[~df2.index.isin(exclude_indexes)]

print(LinearModel(result).adjustedr_squared())
```

```
0.7068610915681984
```

Based on Cook's Distance

In [ ]:
```
for j in cooks_distances.keys():
    exclude_indexes = [cooks_distances[j][1][i][0] for i in range(len(cooks_distances[j][1]))]

    result = df2[~df2.index.isin(exclude_indexes)]

    print(f"Mehnat k baad ka Adjusted R-Squared based on {j} is {LinearModel(result).adjustedr_squared()}")
```

Based on DFFITS Distance

In [ ]:
```
for j in dffits_dict.keys():
    exclude_indexes = [dffits_dict[j][1][i][0] for i in range(len(dffits_dict[j][1]))]

    result = df2[~df2.index.isin(exclude_indexes)]

    print(f"Mehnat k baad ka Adjusted R-Squared based on {j} is {LinearModel(result).adjustedr_squared()}")
```

Final Model after removing influential observations

SSE and R squared

In [ ]:
```
## variables ['x1', 'x2', 'x3', 'x4', 'x5', 'x9', 'x10', 'x7_2', 'x7_3', 'x7_4', 'x6_2']
for k, j in enumerate(dffits_dict.keys()):
    exclude_indexes = [dffits_dict[j][1][i][0] for i in range(len(dffits_dict[j][1]))]

    result = df2[~df2.index.isin(exclude_indexes)]

    print(f"Mehnat k baad ka model coeficients based on {j} is {list(LinearModel(result).model.params)}")

    print(f"Mehnat k baad ka model variables based on {j} is {list(best_models.iloc[k,1])}")
```

In [ ]:
```
# R squared/sse
LinearModel(df2[['x1', 'x2', 'x3', 'x7_2', 'x7_3', 'x7_4', 'x6_2', 'x4', 'x5', 'x10','y']]).model.params
```

Out[ ]:
```
const     -5.665198
x1         1.597776
x2         0.027396
x3         0.963435
x7_2       0.480260
x7_3       0.384884
x7_4       1.141997
x6_2       0.174711
x4         0.000534
x5      -436.893866
x10        0.000358
dtype: float64
```

In [ ]:
```
#Adjusted R squared
LinearModel(df2[ ['x1', 'x2', 'x3', 'x4', 'x5', 'x9', 'x10', 'x7_2', 'x7_3', 'x7_4', 'y']]).model.params
```

Out[ ]:
```
const     -6.833193
x1         1.390107
x2         0.034317
x3         0.869243
x4         0.000534
x5      -243.614570
x9         0.495188
x10       -0.001338
x7_2       0.411510
x7_3       0.270868
x7_4       0.995403
dtype: float64
```

In [ ]:
```
#cp
LinearModel(df2[['x1', 'x3', 'x8', 'x7_4', 'x6_2', 'x4', 'x10', 'y']]).model.params
```

Out[ ]:
```
const    -2.210512
x1        1.420658
x3        0.825063
x8      -26.546144
x7_4      0.927408
x6_2      0.237414
x4        0.154016
x10       0.013759
dtype: float64
```

In [ ]:
```
#AIC
LinearModel(df2[['x1', 'x3', 'x7_4', 'x6_2', 'x4', 'x10','y']]).model.params
```

Out[ ]:
```
const   -3.504969
x1       1.644007
x3       0.842796
x7_4     0.798572
x6_2     0.255600
x4       0.120861
x10      0.247513
dtype: float64
```

In [ ]:
```
#BIC
LinearModel(df2[['x1', 'x3', 'x7_4', 'x4', 'x10', 'y']]).model.params
```