International Journal of Computer Trends and Technology
ISSN: 2231–2803 / https://doi.org/10.14445/22312803/IJCTT-V72I11P106

Volume 72 Issue 11, 39-62, November 2024
© 2024 Seventh Sense Research Group®

Original Article

Investigating the Relationship between Fuel Consumption and Fuel Properties: A Regression Analysis

Asish Pradhan

Technical Architect at Dell Technologies, USA.

Corresponding Author : Asish_Pradhan@Dell.com

Received: 21 September 2024 Revised: 24 October 2024 Accepted: 13 November 2024 Published: 30 November 2024

Abstract *- This study investigates the relationship between fuel consumption and fuel properties, including vehicle type, cetane number, density, viscosity, initial boiling point, final boiling point, and flash point. A linear regression analysis was conducted to identify the most significant predictors of fuel consumption. The results show that vehicle type, cetane number, and initial boiling point are the most significant predictors of fuel consumption. The study also found that the interaction between vehicle type and initial boiling point significantly impacts fuel consumption. This study's findings can inform the development of more efficient and environmentally friendly vehicles.*

Keywords *- Fuel consumption, Exploratory data analysis, Cross Validation, Linear regression, Multivariate Analysis, Regression Analysis.*

1. Introduction

The transportation sector significantly contributes to greenhouse gas emissions, with heavy-duty vehicles being a major source of emissions. With the advancement of civilization and the rise in population, there is a heavy set of vehicles that cause pollution. This pollution does have an impact on humans, birds, and other living beings. To ensure safety and sustenance, this study tries to identify the specific fuel properties that impact fuel consumption and minimizing it. Most research focuses on reducing emissions. However, based on the understanding of the data, it is noticed that the impact of fuel properties, such as cetane number, density, viscosity, boiling point, flash point, aromatics, etc., on engine efficiency, thereby affecting fuel consumption. This study uses the fuel consumption dataset to investigate the relationship between fuel consumption and fuel properties in heavy-duty vehicles [1]. The results will provide insights into the impact of fuel properties on fuel consumption and provide strategies to reduce fuel consumption and emissions in this sector. Various methodologies investigated the relationship between fuel consumption and fuel properties. The model specification was based on a comprehensive analysis of all attributes, including cetane number, density, viscosity, engine type, load capacity, driving conditions, and fuel properties. The model was estimated using Ordinary Least Squares (OLS) regression, and the results were checked for adequacy using plots and diagnostics. Model transformations were applied to address potential issues with normality and homoscedasticity, including square root and log transformations. The introduction of quadratic terms was also considered to account for potential non-linear relationships. The best regressors were selected using all possible regressions and stepwise regression, and the variance influencing factor (VIF) was checked to identify potential multicollinearity issues. The model was validated using multiple techniques, including redoing regressions with new possible models and analyzing the results. Based on the results obtained, the best model has been selected. For this selected model, adequacy and validations were verified using plots and diagnostics. The main goal of this study is to investigate the relationship between fuel consumption and fuel properties and identify the primary factor that influences fuel consumption. This will help the environment be pollution-free, aid manufacturers in developing better vehicles that reduce fuel consumption and inform strategies to reduce fuel consumption and emissions. By employing a range of methodologies, including model specification, parameter estimation, model adequacy checking, model transformation, and model validation, this study aims to provide a comprehensive understanding of the relationship between fuel consumption and fuel properties in heavy-duty vehicles.

2. Exploratory Data Analysis (EDA)

2.1. Dataset

The dataset used for this analysis is a collection of several observations of fuel consumption and related attributes for heavy-duty vehicles. This data was obtained from a research study on polycyclic aromatic hydrocarbons (PAHs) emissions from heavy-duty diesel vehicles [1]. In this prior research document, 47 variables, fuel parameters, and PAH contents


This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)

were analyzed, which resulted in a data matrix of 376 observations. With the application of Principal Component Analysis (PCA), it was found that the fuel aromatic contents and initial and final distillation boiling points affect fuel PAH contents. However, at this point, there is little to no mention of other fuel properties affecting fuel PAH. Therefore, a subset of fuel attributes present in the given dataset is used for this study [6]

Table. 1 Variables & their description

Attribute	Description
y	Fuel consumption (in g/km)
x_1	Vehicle type (binary: 0 = bus, 1 = truck)
x_2	Cetane number
x_3	Density (g/L, 15 °C)
x_4	Viscosity (KV, 40 °C)
x_5	Initial boiling point (degrees C)
x_6	Final boiling point (degrees C)
x_7	Flash point (degrees C)
x_8	Total aromatics (percent)

2.2. Descriptive Analysis

2.2.1. Target Variable: Fuel consumption (y)

Based on this EDA, fuel consumption is a continuous variable with a slightly skewed distribution. Most data points are concentrated around the mean, with a few outliers at the higher end of the range. There is a moderate positive correlation between fuel consumption and cetane number and viscosity and a moderate negative correlation with density and flash point.

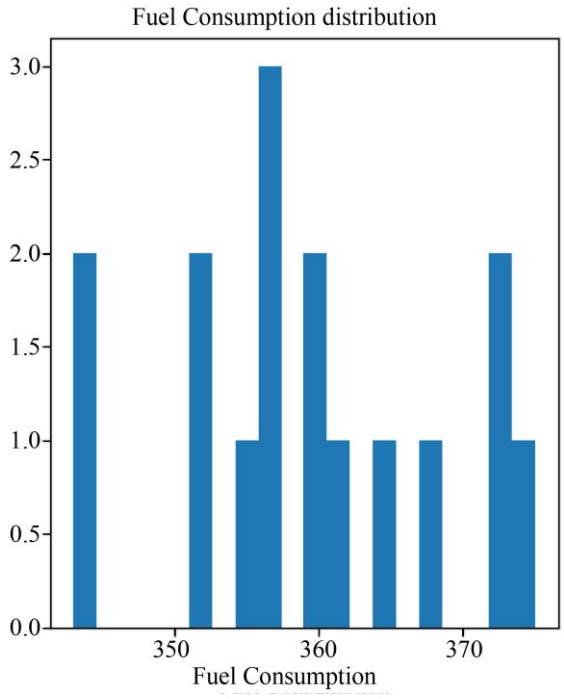

Fig. 1 Distribution of target variable fuel consumption

2.2.2. Vehicle Type Vs Fuel consumption

The vehicle type variable (x_1), is categorical, either 0 or 1. 0 represents a bus, and 1 represents a truck. In the current dataset, both have equal numbers of observations. The correlation coefficient with the target variable y is -0.2345, which shows a weak relationship between them.

2.2.3. Cetane Number Vs Fuel consumption

The variable x2, the Cetane number, has a slightly skewed distribution with a mean of 49.812 and a median of 49.15, indicating a minor skew in data. The correlation coefficient with y is -0.07, representing almost no relationship with the target variable.

2.2.4. Density Vs Fuel consumption

The variable x3, density, has a slightly skewed distribution with a mean of 820.412 and a median of 817.25. The correlation coefficient with y is 0.162. It appears to be a little dependency on the target variable y.

2.2.5. Viscosity Vs Fuel consumption

The variable x4, viscosity, has a slightly skewed distribution with a mean of 1.98, a median of 2.10, and 75% of data have values more than 2.14. The correlation coefficient with y is -0.04, showing a negligible relationship with y.

2.2.6. Initial Boiling Point Vs Fuel consumption

The variable x5, Initial boiling point, has a bit of skewed distribution with a mean of 195.62 and a median of 185. The correlation coefficient with y is -0.591. This implies a strong reverse relationship between x5 and y.

2.2.7. Final Boiling Point Vs Fuel consumption

The variable x6, the final boiling point, has a slightly left-skewed distribution with a mean of 296.12 and a median of 299. Mean value is slightly less than the median, so the data is left skewed. The correlation coefficient with y is 0.588. This suggests that variable x6 has a strong positive relationship with target variable y.

2.2.8. Flash Point Vs Fuel consumption

The variable x7, Flash point, has a slightly skewed distribution with a mean of 76.87 and a median of 75. The correlation coefficient with y is 0.507 showing a positive relationship with y.

2.2.9. Total Aromatics Vs Fuel consumption

The variable x8, Total aromatics, has a left-skewed distribution with a mean of 18.8 and a median of 20.25. The correlation coefficient with y is 0.505, which shows a strong positive relationship with variable y.

2.2.10. Vehicle Type Vs Initial Boiling Point

Vehicle type and initial boiling point x5 seem evenly distributed, with a few missing data in between. At this point,

it's hard to guess how the initial boiling point and the vehicle types are related unless more domain knowledge is gathered the correlation analysis revealed a strong positive relationship between density and viscosity, with a coefficient of 0.82, indicating that as one increases, the other tends to increase. Furthermore, moderate positive correlations were found between total aromatics and density and between total aromatics and final boiling point, suggesting that these variables are related and may be used to predict each other. Additionally, the negative correlation between the initial and final boiling points implies that as the initial boiling point increases, the final boiling point tends to decrease, which may be due to the chemical composition of the substance or the manufacturing process used. Overall, these correlations provide valuable insights into the relationships between these variables and can be used to improve the understanding of the underlying physical and chemical properties of the substances being studied.

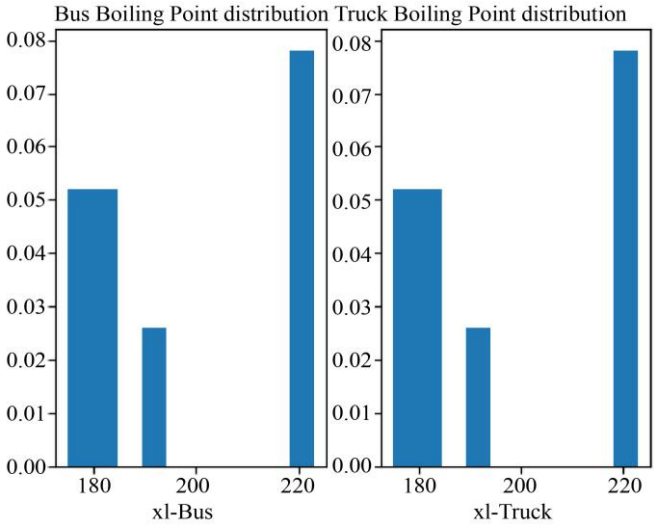

Fig. 2 Distribution of initial boiling point for both vehicle types

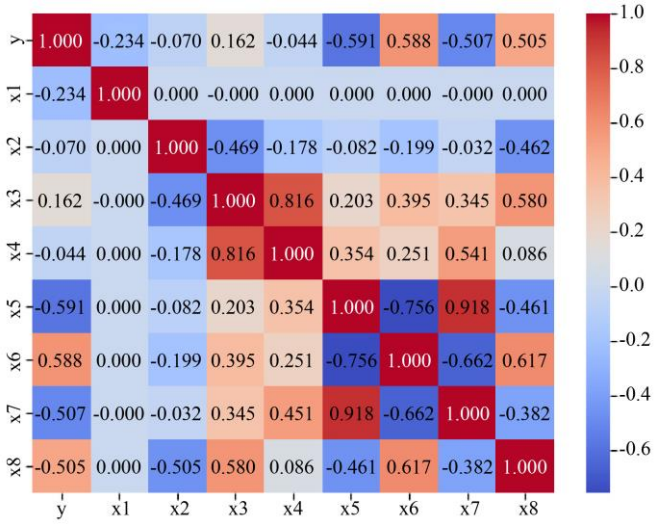

Fig. 3 Correlation matrix

3. Methodology

The goal is to build a predictive model that accurately predicts fuel consumption based on the given dataset. To achieve this, a model-building pipeline is built that iteratively adds and refines the regressors to achieve the best possible model.

3.1. K-Fold Cross Validation

The cross-validation technique measures the model against unseen data while dealing with a defined dataset. This helps avoid overfitting and gives an insight into behaving against unseen datasets. In general, 20% of data is set aside for final model validation, and the remaining 80% of data is set to undergo cross-validation. In a K-fold cross-validation method, one-fold is set aside for validation, and the remaining K-1 fold of data is used for training. Stratification (binning in regression) ensures data in each fold is evenly distributed. In each iteration, the sum model is tested. The overall model's performance is found by taking the average of the metrics, such as R^2 and MSE. [8]

3.2. Multiple Linear Regression

Multiple linear regression is a statistical technique used to analyze the relationship between a dependent variable y and two or more independent variables $x_1, x_2, \dots, x_n$. The goal is to predict the value of y based on the values of the independent variables.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \epsilon \quad (1)$$

Where:

- y is the dependent variable
- $x_1, x_2, \dots, x_n$ are the independent variables
- β_0 is the intercept or constant term
- $\beta_0, \beta_1 \dots \beta_n$ are the slope coefficients
- ϵ is the error term

3.3. Interpretation of the Above Multiple Linear Regression Equation

The slope coefficients (β) represent the change in the dependent variable for a one-unit change in the independent variable while holding all other independent variables constant. The intercept (β_0) represents the value of the dependent variable when all independent variables are equal to zero. The R-squared value represents the proportion of the variance in the dependent variable explained by the independent variables.

3.4. Assumptions in Multiple Linear Regression

- Linearity: The relationship between the dependent variable and independent variables should be linear.
- Independence: Each observation should be independent of the others.
- Homoscedasticity: The variance of the residuals should be constant across all levels of the independent variables.

- Normality: The residuals should be normally distributed.
- No multicollinearity: The independent variables should not be highly correlated. [2,12]

3.5. Advantages & Disadvantages of Multiple Linear Regression

Multiple linear regression is a powerful tool for analyzing the relationship between multiple variables, identifying the relative importance of each variable and creating a linear equation that predicts the dependent variable. However, it requires a large sample size. It can be sensitive to outliers, making it essential to carefully evaluate the model and ensure that the data is representative of the population. Additionally, the model can be difficult to interpret, especially when multiple independent variables exist.

3.6. Define Base Model (with all variables)

This step involves taking all variables in the dataset for regression. Specify the dependent variable (y) and independent variables $(x_1, x_2, \ldots, x_n)$.

$$y = f(x_1, x_2, \ldots, x_n) \quad (2)$$

3.7. Parameter Estimation

Estimate the coefficients of the linear equation using a suitable estimation method. This research uses Ordinary Least Square Regression (OLS), a simple and common technique for parameter estimation. OLS regression is well suited as it adheres to the required assumptions in linear regression (3.3). This is also an unbiased and consistent estimator, as the estimator's expected value is the same as the true value of the parameter, and it tends to converge as the data size increases. This research is based on a limited dataset; therefore, OLS regression would be better suited to this case. When a larger dataset is available, the non-linearity analysis may be studied as part of future work. $\beta_0, \beta_1, \ldots, \beta_n = (X'X)^{-1}X'y$. This step involves using statistical techniques to estimate the coefficients of the linear equation. These coefficients signify the relationship between the independent variable(s) and the dependent variable.

- $X'X$: the covariance matrix of the independent variables.
- $(X'X)^{-1}$: the inverse of the covariance matrix of the independent variables.
- $X'y$: the vector of observations of the dependent variable.

3.8. Model Adequacy Checking

This step involves checking whether the residuals are normally distributed, an assumption of many statistical tests.

$$W = \Sigma(\varepsilon i - \varepsilon)^2 \quad (3)$$

$$W \sim N(0, \sigma^2) \quad (4)$$

W is the sum of squares of the residuals. The residuals are normally distributed with mean 0 and variance σ^2.

3.8. Model Transformation

This step involves transforming the independent variables to meet the assumptions of multiple linear regression. Square root transformation was applied to the dependent variable (y) to meet the normality assumption, and then log transformation was applied to the same dependent variable(y).

$$\sqrt{(y)} = \beta_0 + \beta_1 x_1 + \beta_n x_2 + \ \ldots + \beta_n x_n + \ \varepsilon \quad (5)$$

$$log(y) = \beta_0 + \beta_1 x_1 + \beta_n x_2 + \ \ldots + \beta_n x_n + \ \varepsilon \quad (6)$$

3.9. Non-Linear Relationship Verification

As none of the above transformations didn't help in improving the model, it is essential to check for non-linearity in the relationships between the dependent variable and the predictor variables. F-test and t-test statistics are generally used to determine if the relationships are linear or non-linear. To test this, quadratic terms are introduced, and this process is also impractical to verify each quadratic term with a smaller dataset since that doesn't have much information to extract through relationship plots. There are a few other advanced techniques that can be leveraged to analyze the non-linearity as a future scope.

3.10. Variable Selection – All Possible Regressions

In the above-described methods, all variables are used to verify if a better model was obtained or not. It's necessary to evaluate several combinations of the variables and select the subset of variables that best predicts the outcome variable. As part of all possible regression methods, all possible combinations of predictor variables are considered to find the best fit. Equations that include one, two, three, and so on of the predictor variables are examined, and the best model is found based on an evaluation criterion. This is a challenging and time-consuming task because if there are K predictor variables, there are 2^K possible equations to consider. It's impractical to analyze all of them, so the following analysis plots are used to help.

3.10.1. Coefficient of Multiple Determination(R_p^2)

In general, a better R^2 value decides the goodness of a model; however, it is also necessary to access the R^2 value when a regressor is added or removed from a model equation and make a judgment of whether to consider the regressor or not. Adj R^2 could also be helpful in this purpose of taking or leaving the regressor. The R^2 value for up to p regressors, i.e. R_p^2 is calculated as

$$R_p^2 = \frac{SS_R(p)}{SS_r} = 1 - \left(\frac{SS_{Res}(p)}{SS_T}\right) \quad (7)$$

Where $SS_R(p)$ is the regression sum of squares and $SS_{Res}(p)$ denotes the residual sum of squares for a subset model having p terms. Note that R_p^2 value is calculated for each value of p, one for each possible subset model of size p . As p increases, the R_p^2 value also increases and reaches its maximum when $p = K + 1$. Therefore, adding up regressors

is beneficial until adding additional variables is not useful anymore by just providing a small increase in R_p^2.

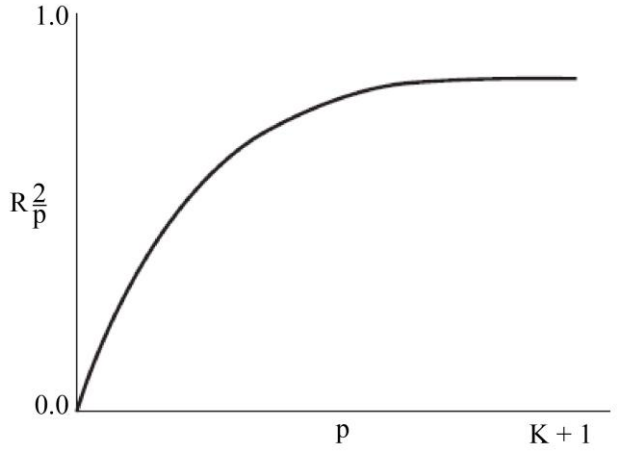

Fig. 3 R_p^2 Vs p plot

3.10.2. Residual Mean Square $MS_{Res}(p)$

$MS_{Res(p)}$ is the reverse of above R_p^2 criterion. Model equations are aimed to achieve the lower MS_{Res} value. A better model has the comparatively lowest $MS_{Res(p)}$ value. In this criterion, the change in $MS_{Res(p)}$ is accessed while adding or deleting a regressor, and then a judgmental discussion is taken on whether to include or exclude the regressor.

$$MS_{Res(p)} = \frac{SS_{Res(p)}}{n-p} \quad (8)$$

As the value of p increases, the sum of squares of residuals $SS_{Res(p)}$ decreases. The $MS_{Res(p)}$ value decreases in the beginning, then stabilizes, and may increase eventually. In the process of adding regressors from a model equation, when the reduction in $SS_{Res(p)}$is not sufficient to compensate for the loss of one degree of freedom in the denominator, the $MS_{Res(p)}$ begins the upward movement.

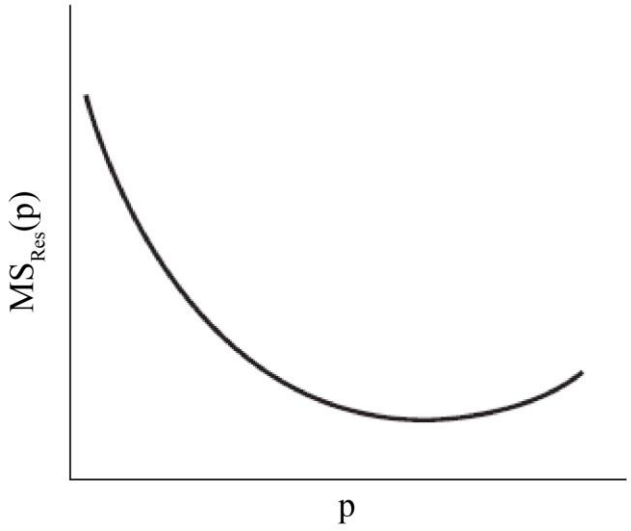

Fig. 4 $MS_{Res(p)}$ Vs p plot

In summary in a $MS_{Res(p)}$ Vs p plot below points is looked at in order to select the best set of regressors.

- Point where $MS_{Res(p)}$ holds the minimum value
- The value of p such that $MS_{Res(p)}$) is approximately equal to MS_{Res} for the full model
- A value of p near the point where the smallest $MS_{Res(p)}$ moves upward

3.10.3. Mallows' s C_p Statistic C_p

In the C_p criterion, a plot of C_p as a function of p can be helpful to visualize it better. Regression equations with slight bias will have values of Cp that fall near the line $Cp = p$ (point A in the figure below), while those equations with substantial bias will fall above this line. Generally, models with small values of Cp and those lying below the $Cp = p$ line represent better models.

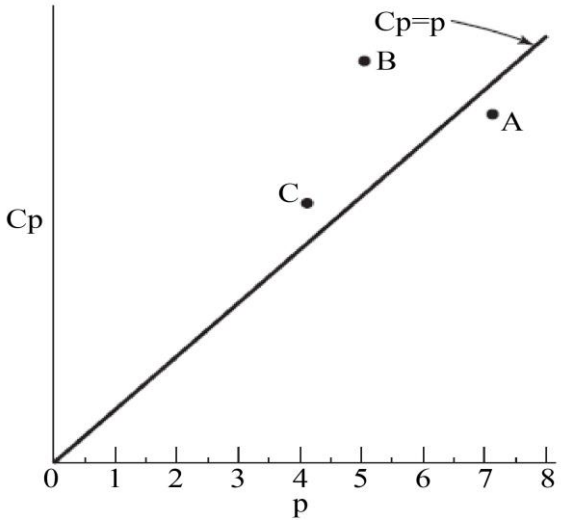

Fig. 5 C_p Vs p plot

3.11. Variable Selection – Forward Selection

Forward selection is a method for selecting the most important predictor variables in regression analysis. It starts with an empty model and adds one predictor variable at a time, selecting the one with the highest correlation coefficient with the response variable. The process continues until a stopping criterion is reached. The advantages of forward selection include ease of implementation and flexibility in choosing the evaluation criterion.

$$F = \frac{SS_{R(X_2|X_1)}}{MS_{Res(x1,x2)}} \quad (9)$$

In general, at each step, the regressor having the highest partial correlation with y (or equivalently, the largest partial F statistic given the other regressors already in the model) is added to the model if its partial F statistic exceeds the preselected entry-level. F_{IN}. It continues until a stopping criterion is reached, such as all or a specified number of predictor variables or F statistic value doesn't exceed F_{IN}.

3.12. Variable Selection – Backward Elimination

Backward elimination is a method for selecting the most important predictor variables in regression analysis by iteratively removing the least important ones. It starts with a full model and removes regressors one by one based on their partial F-statistic, stopping when a stopping criterion is reached. Backward elimination has advantages, such as handling correlated predictor variables and detecting interactions, but it can be computationally intensive. It's a useful alternative to forward selection and can be combined with other methods like stepwise regression and Lasso regression.

3.13. Variable Selection – Stepwise Regression

Stepwise regression is a method that combines forward and backward elimination to select the most important predictor variables in regression analysis. It iteratively adds or removes variables based on their partial F-statistic, stopping when a criterion is reached. Advantages include flexibility, handling of correlated variables, and detection of interactions. However, limitations include sensitivity to variable order and the potential for overfitting.

4. Results and Discussion

The analysis started with a base model by taking all the regressors. The model equation, therefore, was $y = C(x_1) + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8$ (C refers to a categorical variable) Upon running the regression, it is observed that all the p-values are more than the significance level $= 0.05$.

Therefore, none of the regressors are statistically significant, and about 50% of the data are away from the 45-degree line in the QQ plot.[5] As part of the square root transformation, all of the y data are now replaced with their corresponding square roots, and the regression process continues.

OLS Regression Results

Dep. Variable:	y	R-squared:	0.505
Model:	OLS	Adj. R-squared:	-0.060
Method:	Least Squares	F-statistic:	0.8941
Date:	Mon, 11 Nov 2024	Prob (F-statistic):	0.565
Time:	00:41:43	Log-Likelihood:	-52.384
No. Observations:	16	AIC:	122.8
Df Residuals:	7	BIC:	129.7
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-312.8951	1696.890	-0.184	0.859	-4325.403	3699.612
x1	-4.2500	4.832	-0.880	0.408	-15.676	7.176
x2	0.1587	0.963	0.165	0.874	-2.118	2.436
x3	1.0272	2.902	0.354	0.734	-5.836	7.890
x4	-8.6450	45.290	-0.191	0.854	-115.739	98.449
x5	-0.4320	0.925	-0.467	0.655	-2.619	1.755
x6	-0.1368	1.186	-0.115	0.911	-2.941	2.667
x7	-0.3184	3.278	-0.097	0.925	-8.070	7.434
x8	-0.5242	2.245	-0.233	0.822	-5.833	4.784

Omnibus:	8.202	Durbin-Watson:	3.234
Prob(Omnibus):	0.017	Jarque-Bera (JB):	1.735
Skew:	-0.000	Prob(JB):	0.420
Kurtosis:	1.387	Cond. No.	6.31e+05

Fig. 6 OLS regression results for the base model

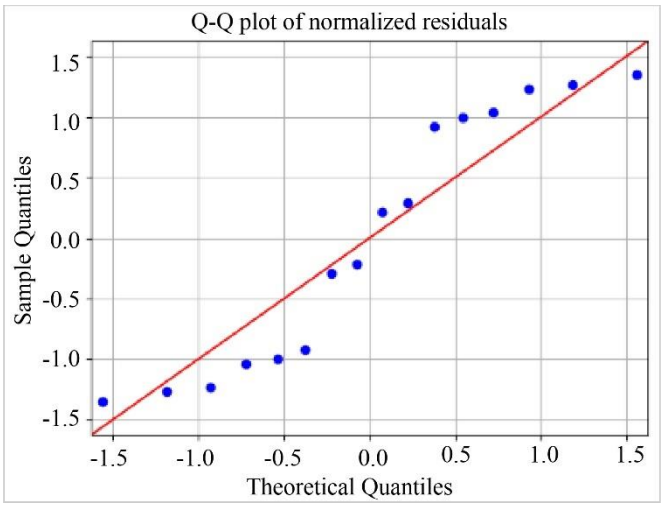

Fig. 7 Residual plot of base model

OLS Regression Results

Dep. Variable:	y	R-squared:	0.504
Model:	OLS	Adj. R-squared:	-0.062
Method:	Least Squares	F-statistic:	0.8903
Date:	Mon, 11 Nov 2024	Prob (F-statistic):	0.568
Time:	01:05:05	Log-Likelihood:	5.7598
No. Observations:	16	AIC:	6.480
Df Residuals:	7	BIC:	13.43
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.0826	44.816	0.024	0.981	-104.890	107.055
x1	-0.1095	0.128	-0.858	0.419	-0.411	0.192
x2	0.0041	0.025	0.163	0.875	-0.056	0.064
x3	0.0273	0.077	0.356	0.732	-0.154	0.209
x4	-0.2331	1.196	-0.195	0.851	-3.062	2.595
x5	-0.0114	0.024	-0.469	0.654	-0.069	0.046
x6	-0.0036	0.031	-0.116	0.911	-0.078	0.070
x7	-0.0084	0.087	-0.097	0.925	-0.213	0.196
x8	-0.0139	0.059	-0.235	0.821	-0.154	0.126

Omnibus:	8.115	Durbin-Watson:	3.235
Prob(Omnibus):	0.017	Jarque-Bera (JB):	1.728
Skew:	-0.000	Prob(JB):	0.422
Kurtosis:	1.390	Cond. No.	6.31e+05

Fig. 8 OLS regression results with sqrt(y)

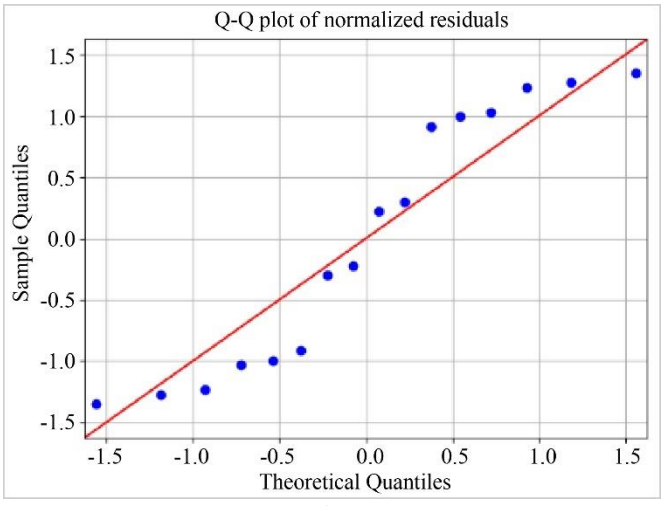

Fig. 9 Residual plot of the model with sqrt(y)

Results with log transformation on target variable y below.

OLS Regression Results

Dep. Variable:	y	R-squared:	0.503
Model:	OLS	Adj. R-squared:	-0.064
Method:	Least Squares	F-statistic:	0.8866
Date:	Mon, 11 Nov 2024	Prob (F-statistic):	0.570
Time:	01:09:03	Log-Likelihood:	41.721
No. Observations:	16	AIC:	-65.44
Df Residuals:	7	BIC:	-58.49
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	3.9833	4.735	0.841	0.428	-7.213	15.180
x1	-0.0113	0.013	-0.837	0.430	-0.043	0.021
x2	0.0004	0.003	0.161	0.877	-0.006	0.007
x3	0.0029	0.008	0.358	0.731	-0.016	0.022
x4	-0.0251	0.126	-0.199	0.848	-0.324	0.274
x5	-0.0012	0.003	-0.470	0.653	-0.007	0.005
x6	-0.0004	0.003	-0.117	0.910	-0.008	0.007
x7	-0.0009	0.009	-0.097	0.925	-0.023	0.021
x8	-0.0015	0.006	-0.237	0.820	-0.016	0.013

Omnibus:	8.025	Durbin-Watson:	3.236
Prob(Omnibus):	0.018	Jarque-Bera (JB):	1.720
Skew:	-0.000	Prob(JB):	0.423
Kurtosis:	1.394	Cond. No.	6.31e+05

Fig. 10 OLS regression results with log(y)

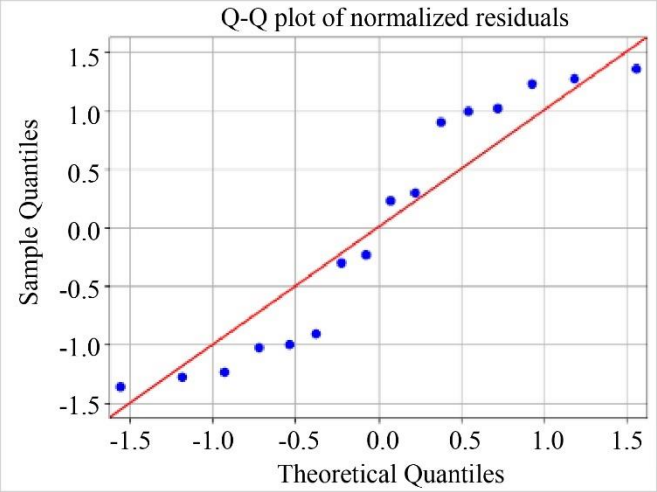

Fig. 11 Residual plot of the model with log(y)

4.1. Best Regressor Selection

4.1.1. All Possible Regressions

Examining the above plot of R_p^2 versus p, after three regressor sets $[x_1, x_3, x_5]$ in the model, there is little or nothing much to be gained in terms of R_p^2 by introducing additional variables. Therefore, x_1, x_3, x_5 are selected as the regressors. The minimum residual mean square model is the model, with $p = 4$ with $MS_{Res}(4) = 56.5183$. In the $MS_{Res}(p)$ Vs p plot, the model that minimizes $MS_{Res}(p)$ also maximizes the adjusted R^2. There are few other regressors with comparable $MS_{Res}(p)$ value. However, x_1, x_3 and x_5 are selected as the regressors as this set has the lowest $MS_{Res}(p)$. From examining the C_p versus p, it is found there are quite a few models that have comparable C_p value. Analyzing the C_p values obtained by running the program, the regressors x_1, x_3, x_5 are selected, even though the sub-model doesn't have the smallest C_p (in general, the smallest C_p is preferred.). This decision is taken as in both the previous R_p^2 *and* $MS_{Res}(p)$ methods incline more towards having $[x_1, x_3, x_5]$ as the regressors and also in the C_p criterion check it's C_p value is close to the smallest C_p even though it's not the smallest.

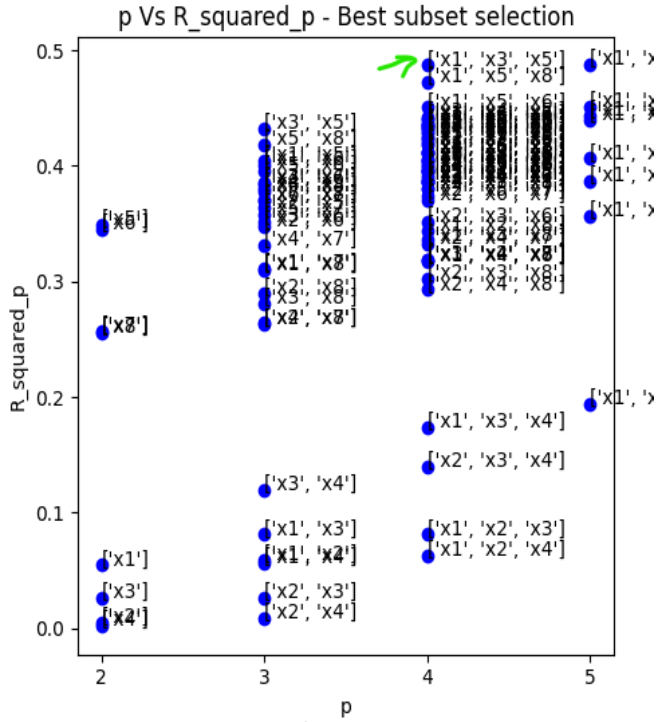

Fig. 12 R_p^2 versus p plot

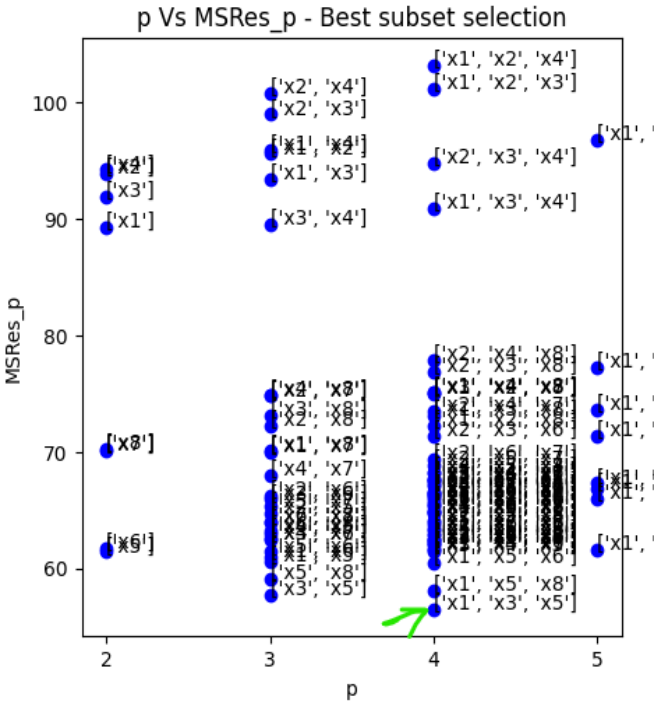

Fig. 13 $MS_{Res}(p)$ versus p plot

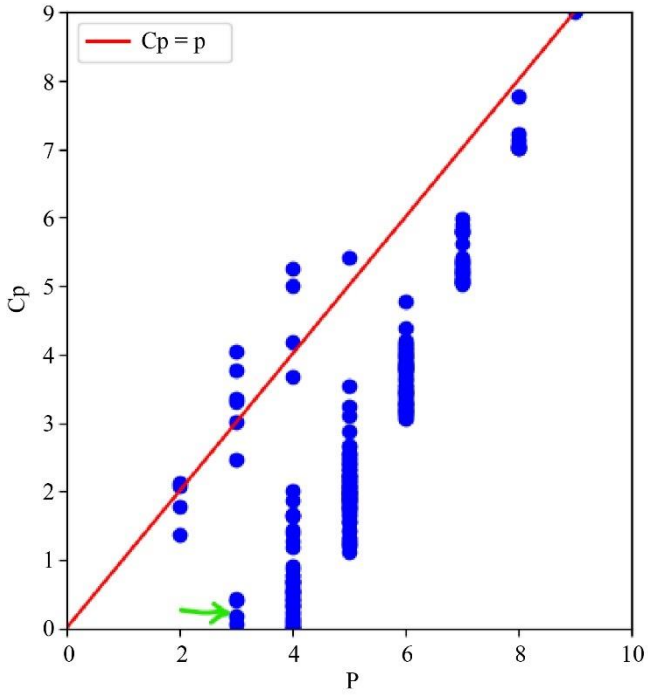

Fig. 14 C_p versus p plot

4.1.2. Stepwise Regressions

Table 1. Algorithm used and the regressors selected

Selection Algorithm	Selected Regressors
Forward Selection	x_3, x_5
Backward Elimination	x_5
Stepwise Regression	x_5

By analyzing the R^2, MS_{Res} and C_p values obtained from the sub-models, it is evident that the model with x_1, x_3, x_5 as regressors predict better compared to the other sub models.

Table 2. Regressors used in the model and their regression statistics

Regressors in sub-model	R_p^2	C_p	MS_{Res}
x1, x3, x5	0.486878	-0.737994	56.5183
x3, x5	0.432215	-1.96438	57.7284
x5	0.34927	-2.79049	61.4359

(Complete set of regressors sets (2^8) with R^2, MS_{Res} and C_p can be found out by running the program in the appendix)

4.1.3. Check Variance Influencing Factor (VIF)

Table 3. VIF of each regressor in the sub-model

Regressors in sub-model	VIF x_1	VIF x_3	VIF x_5
x_1, x_3, x_5	1.9966	97.7009	96.6813
x_3, x_5	NA	96.6812	96.6812

The VIF values for x_3 and x_5 seem to be very high in both models. This indicates a possibility of multicollinearity in the model. While comparing between the sub-models based on their R^2, MS_{Res} and C_p, the model with x_1, x_3 and $x_5$appears to be doing better, and that of the VIF value for x_1 is lower. Therefore, regressors x_1, x_3 and x_5 can be taken as the final set of regressors for the final model. The regression process continued with x_1, x_3 and x_5 as the regressors. There are at least 4 combinations of models (including interaction terms) to come up with the best model.

$$y = C(x_1) + x_3 + x_5$$

$$y = C(x_1) + x_3 + x_5 + C(x_1) * x_3$$

$$y = C(x_1) + x_3 + x_5 + C(x_1) * x_5$$

$$y = C(x_1) + x_3 + x_5 + x_3 * x_5$$

Table 4. Models and their predictability

Model	R^2	$R^2_{Prediction}$
$y = C(x_1) + x_3 + x_5$	46%	26.68%
$y = C(x_1) + x_3 + x_5 + C(x_1) * x_3$	46.2%	18.13%
$y = C(x_1) + x_3 + x_5 + C(x_1) * x_5$	88.7%	87.12%
$y = C(x_1) + x_3 + x_5 + x_3 * x_5$	46.3%	31.82%

OLS Regression Results

Dep. Variable:	y	R-squared:	0.887
Model:	OLS	Adj. R-squared:	0.822
Method:	Least Squares	F-statistic:	13.70
Date:	Mon, 11 Nov 2024	Prob (F-statistic):	0.00201
Time:	09:16:55	Log-Likelihood:	-30.559
No. Observations:	12	AIC:	71.12
Df Residuals:	7	BIC:	73.54
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	271.9233	105.846	2.569	0.037	21.637	522.210
C(x1)	-120.3087	22.877	-5.259	0.001	-174.403	-66.214
x3	0.2520	0.133	1.893	0.100	-0.063	0.567
x5	-0.5973	0.081	-7.337	0.000	-0.790	-0.405
C(x1)*x5	0.5933	0.115	5.139	0.001	0.320	0.866

Omnibus:	0.341	Durbin-Watson:	1.973
Prob(Omnibus):	0.843	Jarque-Bera (JB):	0.455
Skew:	0.040	Prob(JB):	0.797
Kurtosis:	2.050	Cond. No.	7.69e+04

Fig. 15 OLS regression result of the final model

Looking at the normal probability plot, it is evident that the points are closer to the 45-degree line than the original normality plots. The residual plot looks good as the points are distributed evenly. Though two possible outliers are visible, no action is necessary at this point because the QQ residual plot looks good and has a very small dataset of only 16 data points. So, it's better to keep all the data points. Based on the available data and the analysis $y = x_1 + x_3 + x_5 + x_1 * x_5 + \varepsilon$, determined to be the best model. The prediction probability $R^2 = 88.7\%$. The $Adj\ R^2$ value is close to R^2 signifying the regressors contribute well to the final model. F statistic is greater than 1, and the p-value is less than the threshold of 0.05, which signifies there is a good amount of relationship between the target variable and the feature variables. The

equation for the fitted line is $\hat{y} = 271.92 - 120.30x_1 + 0.252x_3 - 0.5973x_5 + 0.5933x_1x_5$. At this point, with the available training data, regressors (x_1),density (x_3) and initial boiling point (x_5) together can make the model prediction well.

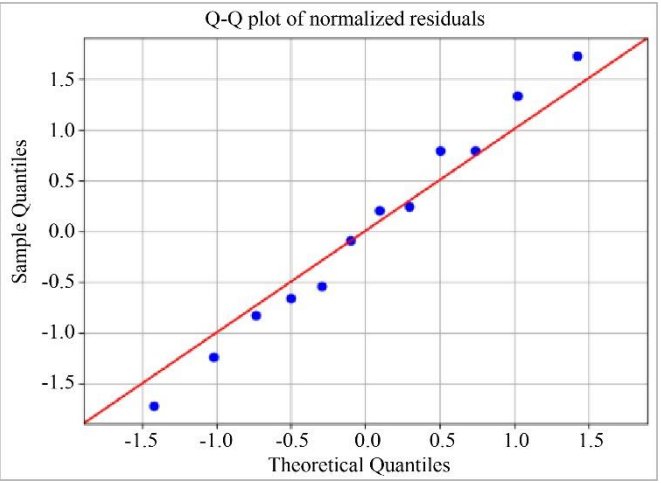

Fig. 16 Residual plot of the final model

4.1.4. Quantifying the Environmental implications

The reduction in fuel consumption has a direct impact on the environment. For every unit increase in fuel density, fuel consumption increases by 0.2474g. This means that a 1% reduction in fuel density would result in a 0.002474g decrease in fuel consumption, leading to a corresponding reduction in greenhouse gas emissions. Additionally, one centigrade increase in initial boiling point results in a 0.5973g decrease in fuel consumption, reducing air pollution, water pollution, land use, and biodiversity impacts. This reduction in fuel consumption has a significant positive impact on the environment, reducing the amount of pollutants released into the atmosphere and preserving natural habitats.

5. Future Work

5.1. Inclusion of other Fuel Attributes

This research is limited to understanding the impact of the above-mentioned fuel properties, such as cetane number, density, viscosity, boiling point, flash point, aromatics, etc., on fuel consumption. However, various external factors, such as fuel type, driving conditions, road conditions, etc., and fuel contents, such as NO_x, CO_2, etc., play a significant role in fuel consumption. More studies need to be carried out based on these factors to get a better understanding of all factors influencing fuel consumption and thereby taking action to reduce pollution.

5.2. Use of Advanced Techniques

The utilization of advanced techniques to further analyze the relationship between fuel density and consumption is necessary. Non-linear analysis using parametric and non-parametric models, such as polynomial regression, exponential regression, and power regression, can provide a more accurate representation of the complex relationships between these variables. Additionally, machine learning algorithms like gradient descent can be employed to identify patterns and trends in the data that may not be apparent through traditional statistical methods. By incorporating these advanced techniques, researchers can gain a deeper understanding of the underlying relationships between fuel density and consumption and develop more effective models for predicting and optimizing fuel consumption.

6. Conclusion

This study investigated the relationship between fuel consumption and various fuel properties, including vehicle type, cetane number, density, viscosity, initial boiling point, final boiling point, flash point, and total aromatics. The results of the correlation analysis revealed a strong negative correlation between fuel consumption and initial boiling point, indicating that initial boiling point could be useful in reducing fuel consumption. The regression analysis revealed a significant relationship between fuel consumption and the predictor variables, with the best-fit model indicating that fuel consumption is influenced by vehicle type, density, initial boiling point, and the interaction between vehicle type and initial boiling point. The model validation showed that the model is a good fit for the data and can be used to make accurate predictions about fuel consumption. Using several regressor selection methods, vehicle type (x_1), density (x_3), and initial boiling point (x_5) found the most significant and important variables with the given dataset in explaining the variability in fuel consumption.

The final model, $y = x_1 + x_3 + x_5 + x_1 * x_5$, suggests that the vehicle type, density, initial boiling point, and the interaction between vehicle type and initial boiling point influence fuel consumption. This model can be used to predict fuel consumption based on the values of the predictor variables and develop strategies to reduce fuel consumption. This study provides valuable insights into the relationships between fuel consumption and fuel properties and highlights the importance of considering the interactions between these variables when optimizing fuel consumption. The findings of this study can be used to inform the development of more efficient and environmentally friendly fuels and to optimize fuel consumption in various applications. Additionally, the results of this study can be used as a basis for further research on the topic, such as investigating the effects of other fuel properties on fuel consumption or examining the impact of fuel consumption on different types of vehicles.

Funding Statement

This study was conducted without any external funding. The research was carried out solely for the purpose of personal interest and curiosity, without any financial support or sponsorship from any external organizations or individuals.

References

[1] Roger Westerholm, and Hang Li, "A Multivariate Statistical Analysis of Fuel-Related Polycyclic Aromatic Hydrocarbon Emissions from Heavy-Duty Diesel Vehicles." *Environmental Science & Technology*, vol. 28, no. 5, pp. 965-972, 1994. [CrossRef] [Google Scholar] [Publisher Link]

[2] Saurabh Kumar, and Avinash Sinha, "Predicting Used Car Prices with Regression Techniques," *International Journal of Computer Trends and Technology*, vol. 72, no. 6, pp. 132-141, 2024. [CrossRef] [Publisher Link]

[3] Trishit Banerjee, "Forecasting Apple Inc. Stock Prices Using S&P500– An OLS Regression Approach with Structural Break," *2020 IEEE 1st International Conference for Convergence in Engineering (ICCE)*, Kolkata, India, pp. 306-310, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[4] Yanming Yang, "Prediction and Analysis of Aero-Material Consumption based on Multivariate Linear Regression Model," *2018 IEEE 3rd International Conference on Clouds Computing and Big Data Analysis (ICCCBDA)*, Chengdu, China, pp. 628-632, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[5] Renyan Jiang, Pei Li, and Kunpeng Zhang, "Quantile-Quantile Plot of Folded-Normal Distribution and its Applications in Reliability and Quality Modeling," *2024 10th International Symposium on System Security, Safety, and Reliability (ISSSR)*, Xiamen, China, pp. 44-50, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[6] Douglas C. Montgomery, Elizabeth A. Peck, and G. Geoffrey Vining, *Introduction to Linear Regression Analysis*, John Wiley & Sons, United States, pp. 1-672, 2015. [Google Scholar] [Publisher Link]

[7] Michael H. Kutner, *Applied Linear Statistical Models*, 5th ed., McGraw-Hill, pp.1-396, 2005. [Google Scholar] [Publisher Link]

[8] Ethm Alpaydin, *Introduction to Machine Learning*, 4th ed., MIT Press, pp. 1-712, 2020. [Google Scholar] [Publisher Link]

[9] Eric Matthes, *Python Crash Course, A Hands-On, Project-Based Introduction to Programming*, 2nd ed., No Starch Press, pp. 1-544, 2019. [Google Scholar] [Publisher Link]

[10] Lynette Cheah et al., *Factor of Two: Halving the Fuel Consumption of New U.S. Automobiles by 2035*, Reducing Climate Impacts in the Transportation Sector, pp. 49-71, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[11] Jim Frost, Model Specification: Choosing the Best Regression Model, Statistics by Jim. [Online]. Available: https://statisticsbyjim.com/regression/model-specification-variable-selection/

[12] Sara Stoudt, The Origins of Ordinary Least Squares Assumptions, Feature Column, 2022. [Online]. Available: https://mathvoices.ams.org/featurecolumn/2022/03/01/ordinary-least-squares/

Appendix 1

Statistics of sub-models

Number of Regressors	p	Regressors in the model	SSRes p	R_squae p	Adj R square p	MSRes p	Cp
1	2	['x1']	1249.5000	0.0547	-0.0129	89.2500	1.3790
1	2	['x2']	1315.3046	0.0049	-0.0662	93.9503	2.0836
1	2	['x3']	1286.9184	0.0264	-0.0432	91.9227	1.7796
1	2	['x4']	1319.2105	0.0019	-0.0694	94.2293	2.1254
1	2	['x5']	860.1029	0.3493	0.3028	61.4359	-2.7905
1	2	['x6']	865.0580	0.3455	0.2988	61.7899	-2.7374
1	2	['x7']	982.6454	0.2566	0.2035	70.1890	-1.4784
1	2	['x8']	984.2976	0.2553	0.2021	70.3070	-1.4607
2	3	['x1', 'x2']	1243.0546	0.0595	-0.0851	95.6196	3.3100
2	3	['x1', 'x3']	1214.6684	0.0810	-0.0604	93.4360	3.0060
2	3	['x1', 'x4']	1246.9605	0.0566	-0.0886	95.9200	3.3518
2	3	['x1', 'x5']	787.8529	0.4039	0.3122	60.6041	-1.5641
2	3	['x1', 'x6']	792.8080	0.4002	0.3079	60.9852	-1.5110
2	3	['x1', 'x7']	910.3954	0.3112	0.2053	70.0304	-0.2520
2	3	['x1', 'x8']	912.0476	0.3100	0.2038	70.1575	-0.2343
2	3	['x2', 'x3']	1286.8523	0.0264	-0.1234	98.9886	3.7789
2	3	['x2', 'x4']	1310.9797	0.0081	-0.1444	100.8446	4.0373
2	3	['x2', 'x5']	841.4976	0.3633	0.2654	64.7306	-0.9897
2	3	['x2', 'x6']	861.9746	0.3479	0.2475	66.3057	-0.7704
2	3	['x2', 'x7']	972.8944	0.2639	0.1507	74.8380	0.4172
2	3	['x2', 'x8']	939.4744	0.2892	0.1799	72.2673	0.0594
2	3	['x3', 'x4']	1163.8195	0.1195	-0.0160	89.5246	2.4615

2	3	['x3', 'x5']	750.4695	0.4322	0.3449	57.7284	-1.9644
2	3	['x3', 'x6']	857.3497	0.3514	0.2516	65.9500	-0.8200
2	3	['x3', 'x7']	812.0833	0.3856	0.2911	62.4679	-1.3047
2	3	['x3', 'x8']	950.2303	0.2811	0.1705	73.0946	0.1746
2	3	['x4', 'x5']	818.7473	0.3806	0.2853	62.9806	-1.2333
2	3	['x4', 'x6']	813.3361	0.3847	0.2900	62.5643	-1.2912
2	3	['x4', 'x7']	883.6632	0.3314	0.2286	67.9741	-0.5382
2	3	['x4', 'x8']	974.1056	0.2630	0.1496	74.9312	0.4302
2	3	['x5', 'x6']	798.7773	0.3957	0.3027	61.4444	-1.4471
2	3	['x5', 'x7']	849.0222	0.3577	0.2588	65.3094	-0.9091
2	3	['x5', 'x8']	769.2151	0.4180	0.3285	59.1704	-1.7637
2	3	['x6', 'x7']	832.5670	0.3701	0.2732	64.0436	-1.0853
2	3	['x6', 'x8']	821.6033	0.3784	0.2828	63.2003	-1.2027
2	3	['x7', 'x8']	832.2658	0.3703	0.2735	64.0204	-1.0886
3	4	['x1', 'x2', 'x3']	1214.6023	0.0811	-0.1487	101.2169	5.0053
3	4	['x1', 'x2', 'x4']	1238.7297	0.0628	-0.1715	103.2275	5.2636
3	4	['x1', 'x2', 'x5']	769.2476	0.4180	0.2725	64.1040	0.2367
3	4	['x1', 'x2', 'x6']	789.7246	0.4025	0.2531	65.8104	0.4559
3	4	['x1', 'x2', 'x7']	900.6444	0.3186	0.1482	75.0537	1.6436
3	4	['x1', 'x2', 'x8']	867.2244	0.3439	0.1799	72.2687	1.2858
3	4	['x1', 'x3', 'x4']	1091.5695	0.1741	-0.0323	90.9641	3.6879
3	4	['x1', 'x3', 'x5']	678.2195	0.4869	0.3586	56.5183	-0.7380
3	4	['x1', 'x3', 'x6']	785.0997	0.4060	0.2575	65.4250	0.4064
3	4	['x1', 'x3', 'x7']	739.8333	0.4403	0.3003	61.6528	-0.0783
3	4	['x1', 'x3', 'x8']	877.9803	0.3357	0.1697	73.1650	1.4009
3	4	['x1', 'x4', 'x5']	746.4973	0.4352	0.2940	62.2081	-0.0069
3	4	['x1', 'x4', 'x6']	741.0861	0.4393	0.2991	61.7572	-0.0649
3	4	['x1', 'x4', 'x7']	811.4132	0.3861	0.2326	67.6178	0.6882
3	4	['x1', 'x4', 'x8']	901.8556	0.3177	0.1471	75.1546	1.6566
3	4	['x1', 'x5', 'x6']	726.5273	0.4503	0.3129	60.5439	-0.2207
3	4	['x1', 'x5', 'x7']	776.7722	0.4123	0.2654	64.7310	0.3173
3	4	['x1', 'x5', 'x8']	696.9651	0.4727	0.3409	58.0804	-0.5373
3	4	['x1', 'x6', 'x7']	760.3170	0.4248	0.2810	63.3598	0.1411
3	4	['x1', 'x6', 'x8']	749.3533	0.4331	0.2913	62.4461	0.0237
3	4	['x1', 'x7', 'x8']	760.0158	0.4250	0.2812	63.3346	0.1378
3	4	['x2', 'x3', 'x4']	1137.5658	0.1393	-0.0758	94.7972	4.1804
3	4	['x2', 'x3', 'x5']	750.1030	0.4325	0.2906	62.5086	0.0317
3	4	['x2', 'x3', 'x6']	856.9727	0.3516	0.1895	71.4144	1.1760
3	4	['x2', 'x3', 'x7']	798.2187	0.3961	0.2451	66.5182	0.5469
3	4	['x2', 'x3', 'x8']	922.3496	0.3022	0.1277	76.8625	1.8760
3	4	['x2', 'x4', 'x5']	807.6841	0.3889	0.2362	67.3070	0.6482
3	4	['x2', 'x4', 'x6']	812.7108	0.3851	0.2314	67.7259	0.7021
3	4	['x2', 'x4', 'x7']	882.1314	0.3326	0.1658	73.5110	1.4454
3	4	['x2', 'x4', 'x8']	934.7633	0.2928	0.1160	77.8969	2.0089
3	4	['x2', 'x5', 'x6']	797.1277	0.3969	0.2461	66.4273	0.5352
3	4	['x2', 'x5', 'x7']	826.8348	0.3744	0.2180	68.9029	0.8533
3	4	['x2', 'x5', 'x8']	767.5349	0.4193	0.2741	63.9612	0.2183
3	4	['x2', 'x6', 'x7']	832.3171	0.3703	0.2129	69.3598	0.9120
3	4	['x2', 'x6', 'x8']	794.8246	0.3987	0.2483	66.2354	0.5106
3	4	['x2', 'x7', 'x8']	818.6713	0.3806	0.2258	68.2226	0.7659
3	4	['x3', 'x4', 'x5']	737.7077	0.4419	0.3023	61.4756	-0.1010
3	4	['x3', 'x4', 'x6']	787.2208	0.4044	0.2555	65.6017	0.4291
3	4	['x3', 'x4', 'x7']	811.4878	0.3861	0.2326	67.6240	0.6890

3	4	['x3', 'x4', 'x8']	900.8902	0.3184	0.1480	75.0742	1.6462
3	4	['x3', 'x5', 'x6']	745.6760	0.4358	0.2948	62.1397	-0.0157
3	4	['x3', 'x5', 'x7']	749.3358	0.4331	0.2913	62.4447	0.0235
3	4	['x3', 'x5', 'x8']	745.4387	0.4360	0.2950	62.1199	-0.0182
3	4	['x3', 'x6', 'x7']	809.9736	0.3872	0.2340	67.4978	0.6728
3	4	['x3', 'x6', 'x8']	778.0683	0.4113	0.2642	64.8390	0.3311
3	4	['x3', 'x7', 'x8']	804.7946	0.3911	0.2389	67.0662	0.6173
3	4	['x4', 'x5', 'x6']	798.6237	0.3958	0.2447	66.5520	0.5512
3	4	['x4', 'x5', 'x7']	818.4796	0.3808	0.2260	68.2066	0.7638
3	4	['x4', 'x5', 'x8']	755.2663	0.4286	0.2857	62.9389	0.0870
3	4	['x4', 'x6', 'x7']	779.4957	0.4103	0.2628	64.9580	0.3464
3	4	['x4', 'x6', 'x8']	777.7035	0.4116	0.2645	64.8086	0.3272
3	4	['x4', 'x7', 'x8']	799.2207	0.3953	0.2442	66.6017	0.5576
3	4	['x5', 'x6', 'x7']	793.2047	0.3999	0.2499	66.1004	0.4932
3	4	['x5', 'x6', 'x8']	754.2588	0.4293	0.2867	62.8549	0.0762
3	4	['x5', 'x7', 'x8']	764.2920	0.4218	0.2772	63.6910	0.1836
3	4	['x6', 'x7', 'x8']	785.6531	0.4056	0.2570	65.4711	0.4123
4	5	['x1', 'x2', 'x3', 'x4']	1065.3158	0.1940	-0.0991	96.8469	5.4068
4	5	['x1', 'x2', 'x3', 'x5']	677.8530	0.4872	0.3007	61.6230	1.2581
4	5	['x1', 'x2', 'x3', 'x6']	784.7227	0.4063	0.1904	71.3384	2.4024
4	5	['x1', 'x2', 'x3', 'x7']	725.9687	0.4508	0.2510	65.9972	1.7733
4	5	['x1', 'x2', 'x3', 'x8']	850.0996	0.3568	0.1230	77.2818	3.1024
4	5	['x1', 'x2', 'x4', 'x5']	735.4341	0.4436	0.2413	66.8576	1.8746
4	5	['x1', 'x2', 'x4', 'x6']	740.4608	0.4398	0.2361	67.3146	1.9285
4	5	['x1', 'x2', 'x4', 'x7']	809.8814	0.3873	0.1645	73.6256	2.6718
4	5	['x1', 'x2', 'x4', 'x8']	862.5133	0.3474	0.1102	78.4103	3.2353
4	5	['x1', 'x2', 'x5', 'x6']	724.8777	0.4516	0.2522	65.8980	1.7616
4	5	['x1', 'x2', 'x5', 'x7']	754.5848	0.4291	0.2215	68.5986	2.0797
4	5	['x1', 'x2', 'x5', 'x8']	695.2849	0.4740	0.2827	63.2077	1.4447
4	5	['x1', 'x2', 'x6', 'x7']	760.0671	0.4250	0.2158	69.0970	2.1384
4	5	['x1', 'x2', 'x6', 'x8']	722.5746	0.4533	0.2545	65.6886	1.7369
4	5	['x1', 'x2', 'x7', 'x8']	746.4213	0.4353	0.2299	67.8565	1.9923
4	5	['x1', 'x3', 'x4', 'x5']	665.4577	0.4965	0.3135	60.4962	1.1254
4	5	['x1', 'x3', 'x4', 'x6']	714.9708	0.4591	0.2624	64.9973	1.6555
4	5	['x1', 'x3', 'x4', 'x7']	739.2378	0.4407	0.2373	67.2034	1.9154
4	5	['x1', 'x3', 'x4', 'x8']	828.6402	0.3731	0.1451	75.3309	2.8726
4	5	['x1', 'x3', 'x5', 'x6']	673.4260	0.4905	0.3052	61.2205	1.2107
4	5	['x1', 'x3', 'x5', 'x7']	677.0858	0.4877	0.3015	61.5533	1.2499
4	5	['x1', 'x3', 'x5', 'x8']	673.1887	0.4907	0.3055	61.1990	1.2081
4	5	['x1', 'x3', 'x6', 'x7']	737.7236	0.4419	0.2389	67.0658	1.8991
4	5	['x1', 'x3', 'x6', 'x8']	705.8183	0.4660	0.2718	64.1653	1.5575
4	5	['x1', 'x3', 'x7', 'x8']	732.5446	0.4458	0.2442	66.5950	1.8437
4	5	['x1', 'x4', 'x5', 'x6']	726.3737	0.4504	0.2506	66.0340	1.7776
4	5	['x1', 'x4', 'x5', 'x7']	746.2296	0.4354	0.2301	67.8391	1.9902
4	5	['x1', 'x4', 'x5', 'x8']	683.0163	0.4832	0.2953	62.0924	1.3134
4	5	['x1', 'x4', 'x6', 'x7']	707.2457	0.4649	0.2703	64.2951	1.5728
4	5	['x1', 'x4', 'x6', 'x8']	705.4535	0.4663	0.2722	64.1321	1.5536
4	5	['x1', 'x4', 'x7', 'x8']	726.9707	0.4500	0.2500	66.0882	1.7840
4	5	['x1', 'x5', 'x6', 'x7']	720.9547	0.4545	0.2562	65.5413	1.7196
4	5	['x1', 'x5', 'x6', 'x8']	682.0088	0.4840	0.2964	62.0008	1.3026
4	5	['x1', 'x5', 'x7', 'x8']	692.0420	0.4764	0.2860	62.9129	1.4100
4	5	['x1', 'x6', 'x7', 'x8']	713.4031	0.4603	0.2640	64.8548	1.6387
4	5	['x2', 'x3', 'x4', 'x5']	732.3446	0.4459	0.2444	66.5768	1.8415

4	5	['x2', 'x3', 'x4', 'x6']	768.3738	0.4187	0.2073	69.8522	2.2273
4	5	['x2', 'x3', 'x4', 'x7']	792.7565	0.4002	0.1821	72.0688	2.4884
4	5	['x2', 'x3', 'x4', 'x8']	891.2011	0.3257	0.0806	81.0183	3.5425
4	5	['x2', 'x3', 'x5', 'x6']	745.3324	0.4361	0.2310	67.7575	1.9806
4	5	['x2', 'x3', 'x5', 'x7']	748.1710	0.4340	0.2281	68.0155	2.0110
4	5	['x2', 'x3', 'x5', 'x8']	743.0781	0.4378	0.2334	67.5526	1.9565
4	5	['x2', 'x3', 'x6', 'x7']	798.1847	0.3961	0.1765	72.5622	2.5465
4	5	['x2', 'x3', 'x6', 'x8']	766.3723	0.4202	0.2093	69.6702	2.2059
4	5	['x2', 'x3', 'x7', 'x8']	785.0343	0.4061	0.1901	71.3668	2.4057
4	5	['x2', 'x4', 'x5', 'x6']	797.1199	0.3969	0.1776	72.4654	2.5351
4	5	['x2', 'x4', 'x5', 'x7']	807.5687	0.3890	0.1668	73.4153	2.6470
4	5	['x2', 'x4', 'x5', 'x8']	753.7036	0.4298	0.2224	68.5185	2.0702
4	5	['x2', 'x4', 'x6', 'x7']	767.5021	0.4193	0.2082	69.7729	2.2180
4	5	['x2', 'x4', 'x6', 'x8']	762.0422	0.4235	0.2138	69.2766	2.1595
4	5	['x2', 'x4', 'x7', 'x8']	785.7123	0.4056	0.1894	71.4284	2.4130
4	5	['x2', 'x5', 'x6', 'x7']	790.2118	0.4021	0.1847	71.8374	2.4612
4	5	['x2', 'x5', 'x6', 'x8']	749.9598	0.4326	0.2263	68.1782	2.0302
4	5	['x2', 'x5', 'x7', 'x8']	763.5928	0.4223	0.2122	69.4175	2.1761
4	5	['x2', 'x6', 'x7', 'x8']	770.2947	0.4172	0.2053	70.0268	2.2479
4	5	['x3', 'x4', 'x5', 'x6']	737.5067	0.4420	0.2391	67.0461	1.8968
4	5	['x3', 'x4', 'x5', 'x7']	737.2280	0.4422	0.2394	67.0207	1.8938
4	5	['x3', 'x4', 'x5', 'x8']	733.1577	0.4453	0.2436	66.6507	1.8503
4	5	['x3', 'x4', 'x6', 'x7']	777.7074	0.4116	0.1976	70.7007	2.3273
4	5	['x3', 'x4', 'x6', 'x8']	777.0274	0.4121	0.1983	70.6389	2.3200
4	5	['x3', 'x4', 'x7', 'x8']	799.1956	0.3954	0.1755	72.6541	2.5574
4	5	['x3', 'x5', 'x6', 'x7']	739.8918	0.4402	0.2367	67.2629	1.9224
4	5	['x3', 'x5', 'x6', 'x8']	744.3966	0.4368	0.2320	67.6724	1.9706
4	5	['x3', 'x5', 'x7', 'x8']	745.3464	0.4361	0.2310	67.7588	1.9808
4	5	['x3', 'x6', 'x7', 'x8']	774.6406	0.4139	0.2008	70.4219	2.2944
4	5	['x4', 'x5', 'x6', 'x7']	757.9088	0.4266	0.2181	68.9008	2.1153
4	5	['x4', 'x5', 'x6', 'x8']	753.4383	0.4300	0.2227	68.4944	2.0674
4	5	['x4', 'x5', 'x7', 'x8']	755.2610	0.4286	0.2208	68.6601	2.0869
4	5	['x4', 'x6', 'x7', 'x8']	769.5661	0.4178	0.2060	69.9606	2.2401
4	5	['x5', 'x6', 'x7', 'x8']	750.6067	0.4321	0.2256	68.2370	2.0371
5	6	['x1', 'x2', 'x3', 'x4', 'x5']	660.0946	0.5006	0.2509	66.0095	3.0679
5	6	['x1', 'x2', 'x3', 'x4', 'x6']	696.1238	0.4733	0.2100	69.6124	3.4537
5	6	['x1', 'x2', 'x3', 'x4', 'x7']	720.5065	0.4549	0.1823	72.0506	3.7148
5	6	['x1', 'x2', 'x3', 'x4', 'x8']	818.9511	0.3804	0.0706	81.8951	4.7689
5	6	['x1', 'x2', 'x3', 'x5', 'x6']	673.0824	0.4908	0.2361	67.3082	3.2070
5	6	['x1', 'x2', 'x3', 'x5', 'x7']	675.9210	0.4886	0.2329	67.5921	3.2374
5	6	['x1', 'x2', 'x3', 'x5', 'x8']	670.8281	0.4925	0.2387	67.0828	3.1829
5	6	['x1', 'x2', 'x3', 'x6', 'x7']	725.9347	0.4508	0.1762	72.5935	3.7729
5	6	['x1', 'x2', 'x3', 'x6', 'x8']	694.1223	0.4748	0.2123	69.4122	3.4323
5	6	['x1', 'x2', 'x3', 'x7', 'x8']	712.7843	0.4607	0.1911	71.2784	3.6321
5	6	['x1', 'x2', 'x4', 'x5', 'x6']	724.8699	0.4516	0.1774	72.4870	3.7615
5	6	['x1', 'x2', 'x4', 'x5', 'x7']	735.3187	0.4437	0.1655	73.5319	3.8734
5	6	['x1', 'x2', 'x4', 'x5', 'x8']	681.4536	0.4844	0.2266	68.1454	3.2966
5	6	['x1', 'x2', 'x4', 'x6', 'x7']	695.2521	0.4740	0.2110	69.5252	3.4444
5	6	['x1', 'x2', 'x4', 'x6', 'x8']	689.7922	0.4781	0.2172	68.9792	3.3859
5	6	['x1', 'x2', 'x4', 'x7', 'x8']	713.4623	0.4602	0.1903	71.3462	3.6394
5	6	['x1', 'x2', 'x5', 'x6', 'x7']	717.9618	0.4568	0.1852	71.7962	3.6875
5	6	['x1', 'x2', 'x5', 'x6', 'x8']	677.7098	0.4873	0.2309	67.7710	3.2565
5	6	['x1', 'x2', 'x5', 'x7', 'x8']	691.3428	0.4769	0.2154	69.1343	3.4025

5	6	['x1', 'x2', 'x6', 'x7', 'x8']	698.0447	0.4719	0.2078	69.8045	3.4743
5	6	['x1', 'x3', 'x4', 'x5', 'x6']	665.2567	0.4967	0.2450	66.5257	3.1232
5	6	['x1', 'x3', 'x4', 'x5', 'x7']	664.9780	0.4969	0.2453	66.4978	3.1202
5	6	['x1', 'x3', 'x4', 'x5', 'x8']	660.9077	0.5000	0.2500	66.0908	3.0766
5	6	['x1', 'x3', 'x4', 'x6', 'x7']	705.4574	0.4663	0.1994	70.5457	3.5537
5	6	['x1', 'x3', 'x4', 'x6', 'x8']	704.7774	0.4668	0.2002	70.4777	3.5464
5	6	['x1', 'x3', 'x4', 'x7', 'x8']	726.9456	0.4500	0.1750	72.6946	3.7837
5	6	['x1', 'x3', 'x5', 'x6', 'x7']	667.6418	0.4949	0.2423	66.7642	3.1487
5	6	['x1', 'x3', 'x5', 'x6', 'x8']	672.1466	0.4915	0.2372	67.2147	3.1970
5	6	['x1', 'x3', 'x5', 'x7', 'x8']	673.0964	0.4908	0.2361	67.3096	3.2072
5	6	['x1', 'x3', 'x6', 'x7', 'x8']	702.3906	0.4686	0.2029	70.2391	3.5208
5	6	['x1', 'x4', 'x5', 'x6', 'x7']	685.6588	0.4812	0.2219	68.5659	3.3417
5	6	['x1', 'x4', 'x5', 'x6', 'x8']	681.1883	0.4846	0.2269	68.1188	3.2938
5	6	['x1', 'x4', 'x5', 'x7', 'x8']	683.0110	0.4833	0.2249	68.3011	3.3133
5	6	['x1', 'x4', 'x6', 'x7', 'x8']	697.3161	0.4724	0.2086	69.7316	3.4665
5	6	['x1', 'x5', 'x6', 'x7', 'x8']	678.3567	0.4868	0.2302	67.8357	3.2635
5	6	['x2', 'x3', 'x4', 'x5', 'x6']	732.3002	0.4460	0.1689	73.2300	3.8411
5	6	['x2', 'x3', 'x4', 'x5', 'x7']	732.3084	0.4460	0.1689	73.2308	3.8412
5	6	['x2', 'x3', 'x4', 'x5', 'x8']	727.6026	0.4495	0.1743	72.7603	3.7908
5	6	['x2', 'x3', 'x4', 'x6', 'x7']	758.3774	0.4262	0.1393	75.8377	4.1203
5	6	['x2', 'x3', 'x4', 'x6', 'x8']	761.5426	0.4238	0.1358	76.1543	4.1542
5	6	['x2', 'x3', 'x4', 'x7', 'x8']	784.1670	0.4067	0.1101	78.4167	4.3964
5	6	['x2', 'x3', 'x5', 'x6', 'x7']	736.8091	0.4426	0.1638	73.6809	3.8894
5	6	['x2', 'x3', 'x5', 'x6', 'x8']	742.7705	0.4380	0.1571	74.2771	3.9532
5	6	['x2', 'x3', 'x5', 'x7', 'x8']	742.4922	0.4383	0.1574	74.2492	3.9502
5	6	['x2', 'x3', 'x6', 'x7', 'x8']	765.2747	0.4210	0.1315	76.5275	4.1941
5	6	['x2', 'x4', 'x5', 'x6', 'x7']	755.8954	0.4281	0.1422	75.5895	4.0937
5	6	['x2', 'x4', 'x5', 'x6', 'x8']	749.8964	0.4326	0.1490	74.9896	4.0295
5	6	['x2', 'x4', 'x5', 'x7', 'x8']	753.6381	0.4298	0.1447	75.3638	4.0695
5	6	['x2', 'x4', 'x6', 'x7', 'x8']	748.4563	0.4337	0.1506	74.8456	4.0141
5	6	['x2', 'x5', 'x6', 'x7', 'x8']	747.8240	0.4342	0.1513	74.7824	4.0073
5	6	['x3', 'x4', 'x5', 'x6', 'x7']	736.8914	0.4425	0.1637	73.6891	3.8902
5	6	['x3', 'x4', 'x5', 'x6', 'x8']	730.8385	0.4471	0.1706	73.0838	3.8254
5	6	['x3', 'x4', 'x5', 'x7', 'x8']	731.9447	0.4462	0.1693	73.1945	3.8373
5	6	['x3', 'x4', 'x6', 'x7', 'x8']	761.3614	0.4240	0.1360	76.1361	4.1522
5	6	['x3', 'x5', 'x6', 'x7', 'x8']	730.4133	0.4474	0.1711	73.0413	3.8209
5	6	['x4', 'x5', 'x6', 'x7', 'x8']	744.1177	0.4370	0.1555	74.4118	3.9676
6	7	['x1', 'x2', 'x3', 'x4', 'x5', 'x6']	660.0502	0.5006	0.1677	73.3389	5.0675
6	7	['x1', 'x2', 'x3', 'x4', 'x5', 'x7']	660.0584	0.5006	0.1677	73.3398	5.0675
6	7	['x1', 'x2', 'x3', 'x4', 'x5', 'x8']	655.3526	0.5042	0.1736	72.8170	5.0172
6	7	['x1', 'x2', 'x3', 'x4', 'x6', 'x7']	686.1274	0.4809	0.1348	76.2364	5.3467
6	7	['x1', 'x2', 'x3', 'x4', 'x6', 'x8']	689.2926	0.4785	0.1308	76.5881	5.3806
6	7	['x1', 'x2', 'x3', 'x4', 'x7', 'x8']	711.9170	0.4614	0.1023	79.1019	5.6228
6	7	['x1', 'x2', 'x3', 'x5', 'x6', 'x7']	664.5591	0.4972	0.1620	73.8399	5.1157
6	7	['x1', 'x2', 'x3', 'x5', 'x6', 'x8']	670.5205	0.4927	0.1545	74.5023	5.1796

6	7	['x1', 'x2', 'x3', 'x5', 'x7', 'x8']	670.2422	0.4929	0.1549	74.4714	5.1766
6	7	['x1', 'x2', 'x3', 'x6', 'x7', 'x8']	693.0247	0.4757	0.1261	77.0027	5.4205
6	7	['x1', 'x2', 'x4', 'x5', 'x6', 'x7']	683.6454	0.4828	0.1380	75.9606	5.3201
6	7	['x1', 'x2', 'x4', 'x5', 'x6', 'x8']	677.6464	0.4873	0.1455	75.2940	5.2559
6	7	['x1', 'x2', 'x4', 'x5', 'x7', 'x8']	681.3881	0.4845	0.1408	75.7098	5.2959
6	7	['x1', 'x2', 'x4', 'x6', 'x7', 'x8']	676.2063	0.4884	0.1473	75.1340	5.2405
6	7	['x1', 'x2', 'x5', 'x6', 'x7', 'x8']	675.5740	0.4889	0.1481	75.0638	5.2337
6	7	['x1', 'x3', 'x4', 'x5', 'x6', 'x7']	664.6414	0.4972	0.1619	73.8490	5.1166
6	7	['x1', 'x3', 'x4', 'x5', 'x6', 'x8']	658.5885	0.5017	0.1695	73.1765	5.0518
6	7	['x1', 'x3', 'x4', 'x5', 'x7', 'x8']	659.6947	0.5009	0.1682	73.2994	5.0637
6	7	['x1', 'x3', 'x4', 'x6', 'x7', 'x8']	689.1114	0.4786	0.1311	76.5679	5.3786
6	7	['x1', 'x3', 'x5', 'x6', 'x7', 'x8']	658.1633	0.5021	0.1701	73.1293	5.0473
6	7	['x1', 'x4', 'x5', 'x6', 'x7', 'x8']	671.8677	0.4917	0.1528	74.6520	5.1940
6	7	['x2', 'x3', 'x4', 'x5', 'x6', 'x7']	731.0911	0.4469	0.0781	81.2323	5.8281
6	7	['x2', 'x3', 'x4', 'x5', 'x6', 'x8']	726.8809	0.4501	0.0834	80.7645	5.7830
6	7	['x2', 'x3', 'x4', 'x5', 'x7', 'x8']	727.2434	0.4498	0.0830	80.8048	5.7869
6	7	['x2', 'x3', 'x4', 'x6', 'x7', 'x8']	746.3690	0.4353	0.0589	82.9299	5.9917
6	7	['x2', 'x3', 'x5', 'x6', 'x7', 'x8']	729.4028	0.4482	0.0803	81.0448	5.8100
6	7	['x2', 'x4', 'x5', 'x6', 'x7', 'x8']	737.6990	0.4419	0.0698	81.9666	5.8989
6	7	['x3', 'x4', 'x5', 'x6', 'x7', 'x8']	728.5370	0.4488	0.0813	80.9486	5.8008
7	8	['x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'x7']	658.8411	0.5015	0.0654	82.3551	7.0545
7	8	['x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'x8']	654.6309	0.5047	0.0714	81.8289	7.0094
7	8	['x1', 'x2', 'x3', 'x4', 'x5', 'x7', 'x8']	654.9934	0.5044	0.0708	81.8742	7.0133
7	8	['x1', 'x2', 'x3', 'x4', 'x6', 'x7', 'x8']	674.1190	0.4900	0.0437	84.2649	7.2181
7	8	['x1', 'x2', 'x3', 'x5', 'x6', 'x7', 'x8']	657.1528	0.5028	0.0678	82.1441	7.0364
7	8	['x1', 'x2', 'x4', 'x5', 'x6', 'x7', 'x8']	665.4490	0.4965	0.0560	83.1811	7.1253
7	8	['x1', 'x3', 'x4', 'x5', 'x6', 'x7', 'x8']	656.2870	0.5035	0.0690	82.0359	7.0272

7	8	['x2', 'x3', 'x4', 'x5', 'x6', 'x7', 'x8']	726.0000	0.4507	-0.0299	90.7500	7.7736
8	9	['x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'x7', 'x8']	653.7500	0.5054	-0.0599	93.3929	9.0000

Appendix 2

Python Code

Start.py

```
import pandas as pd
from tabulate import tabulate
import matplotlib.pyplot as plt
import numpy as np
from ParametersEstimator import ParametersEstimator
import BestRegressorFinder as bestregressor
import ModelAdequacyChecker as modeladequacychecker
from statsmodels.stats.outliers_influence import variance_inflation_factor

def main():
    df = load_data()

    #visualize original data
    visualize_data(df)

    #step# 1
    print('Checking the model with all the regressors... section 6.1 of the report')
    regressors = ['x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'x7', 'x8']
    params_estimator = ParametersEstimator(df, regressors, 'y ~ C(x1) + x2 + x3 + x4 + x5 + x6 + x7 + x8')
    params_estimator.do_parameter_estimations()
    print('Model adequacy checking... Section 6.3 of the report')
    modeladequacychecker.check_model_adequacy(params_estimator.regression_result)

    # #step# 2 square root transformation
    print('Square root transformation... section 6.4.1 of the report')
    df1 = df.copy()
    df1['y'] = (np.sqrt(df1['y']))
    regressors = ['x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'x7', 'x8']
    params_estimator = ParametersEstimator(df1, regressors, 'y ~ C(x1) + x2 + x3 + x4 + x5 + x6 + x7 + x8')
    params_estimator.do_parameter_estimations()
    print('Model adequacy checking... Section 6.4.1 of the report')
    modeladequacychecker.check_model_adequacy(params_estimator.regression_result)
    #
    # step# 2 log transformation
    print('Log transformation... section 6.4.2 of the report')
    df1 = df.copy()
    df1['y'] = (np.log(df1['y']))
    regressors = ['x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'x7', 'x8']
    params_estimator = ParametersEstimator(df1, regressors, 'y ~ C(x1) + x2 + x3 + x4 + x5 + x6 + x7 + x8')
    params_estimator.do_parameter_estimations()
    modeladequacychecker.check_model_adequacy(params_estimator.regression_result)

    #step# 3 Quadratric a)
    print('Checking model with quadratric terms... section 6.4.3 of the report')
    params_estimator = ParametersEstimator(df, regressors, 'y ~ C(x1) + x2*x2 + x3 + x4 + x5 + x6 + x7 + x8')
    params_estimator.do_parameter_estimations()
    print('Model adequacy checking... Section 6.4.2 of the report')
    modeladequacychecker.check_model_adequacy(params_estimator.regression_result)

    # step# 3 Quadratric b)
    print('Checking model with quadratric terms... section 6.4.3 of the report')
    params_estimator = ParametersEstimator(df, regressors, 'y ~ C(x1) + x2 + x3*x3 + x4 + x5 + x6 + x7 + x8')
    params_estimator.do_parameter_estimations()
    print('Model adequacy checking... Section 6.4.3 of the report')
    modeladequacychecker.check_model_adequacy(params_estimator.regression_result)

    # step# 4 Best regressors
```

```
    print('An attempt to get best set of regressors... section 6.5 of the report')
    best_regressors = bestregressor.get_best_regressors(df, regressors)
    print(tabulate(best_regressors, headers='keys', tablefmt='psql', showindex=False))

    #
    regressors_sets_to_consider = [['x1', 'x3', 'x5'], ['x3', 'x5'], ['x5']]
    for regressors_set in regressors_sets_to_consider:
        print(tabulate(get_vif_values(df[regressors_set]), headers='keys', tablefmt='psql', showindex=False))

    ##Model validation split data into 75%-25%
    print('Analysis on different combinations of best regressor sets... section 6.5.3 of the report')
    split_at = int(len(df)*0.75)
    estimation_df = df.sample(split_at, random_state=3)
    prediction_df = pd.concat([df, estimation_df]).drop_duplicates(keep=False)

    estimation_df = pd.DataFrame(data=estimation_df.values,columns=df.columns)
    prediction_df = pd.DataFrame(data=prediction_df.values,columns=df.columns)

    regressors_sets_to_consider = [['C(x1)','x3','x5'],['C(x1)','x3','x5','C(x1)*x3'],
                                   ['C(x1)','x3','x5','C(x1)*x5'],['C(x1)','x3','x5','x3*x5']]
    #
    for regressors_set in regressors_sets_to_consider:
        params_estimator = ParametersEstimator(estimation_df, regressors_set)
        params_estimator.do_parameter_estimations()
        print('Model adequacy checking for model {} ... Section 6.5.3 of the report'.format(regressors_set))
        modeladequacychecker.check_model_adequacy(params_estimator.regression_result)
        PRESS, R_square_prediction = validate_model(prediction_df, params_estimator.regression_result);
        print('PRESS Statistics = {} R_square_prediction = {}'.format(np.round(PRESS, 4), R_square_prediction))

    #residual analysis of the final model with complete data
    print('Analysis on the final model ... section 6.5.3.1 of the report')
    params_estimator = ParametersEstimator(df, regressors_sets_to_consider[2], 'y ~ C(x1) + x3 + x5 + C(x1) * x5')
    params_estimator.do_parameter_estimations()
    print('Model adequacy checking for the final model {} ... Section 6.5.3.2 of the
report'.format(regressors_sets_to_consider[2]))
    modeladequacychecker.check_model_adequacy(params_estimator.regression_result)
    PRESS, R_square_prediction = validate_model(df, params_estimator.regression_result);
    print('PRESS Statistics = {} R_square_prediction = {}'.format(np.round(PRESS, 4), R_square_prediction))



def validate_model(prediction_df,regression_result):

    fitted_value_prediction_data_list = regression_result.predict(prediction_df)

    PRESS = np.sum(np.square(prediction_df['y'] - fitted_value_prediction_data_list))
    y_bar = np.mean(prediction_df['y'])
    SST = np.sum(np.square(prediction_df['y'] - y_bar))
    R_square_prediction = str(np.round((1 - (PRESS / SST)) * 100, 2)) + '%'
    return PRESS, R_square_prediction

def plot_y_x(x, y, xlabel, ylabel, title):
    plt.figure(figsize=(8, 5))
    plt.plot(x, y, 'b.',markersize=10)
    plt.xlabel(xlabel, fontsize=15)
    plt.ylabel(ylabel, fontsize=15)
    plt.title(title, fontsize=18)
    plt.grid(True)

def get_vif_values(features_df):
    vif = pd.DataFrame()
    if len(features_df.columns)>1:
        vif["VIF Factor"] = [variance_inflation_factor(features_df.values, i) for i in range(features_df.shape[1])]
        vif["features"] = features_df.columns
    return vif

def visualize_data(df):
    plot_y_x(df['x1'], df['y'],'x1','y','x1 vs y')
    plot_y_x(df['x2'], df['y'],'x2','y','x2 vs y')
```

```
    plot_y_x(df['x3'], df['y'],'x3','y','x3 vs y')
    plot_y_x(df['x4'], df['y'],'x4','y','x4 vs y')

    plot_y_x(df['x5'], df['y'], 'x5', 'y', 'x5 vs y')
    plot_y_x(df['x6'], df['y'], 'x6', 'y', 'x6 vs y')
    plot_y_x(df['x7'], df['y'], 'x7', 'y', 'x7 vs y')
    plot_y_x(df['x8'], df['y'], 'x8', 'y', 'x8 vs y')

    plt.show()

def load_data():
    # Creating pd DataFrames
    # Table B.18
    y = [343, 356, 344, 356, 352, 361, 372, 355, 375, 359, 364, 357, 368, 360, 372, 352]
    x1 = [0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]
    x2 = [52.8, 52.8, 50.0, 50.0, 47.2, 47.2, 47.0, 47.0, 48.3, 48.3, 44.7, 44.7, 55.7, 55.7, 52.8, 52.8]
    x3 = [811.7, 811.7, 821.3, 821.3, 832.0, 832.0, 831.3, 831.3, 836.8, 836.8, 808.3, 808.3, 808.7, 808.7, 813.2,
          813.2]
    x4 = [2.11, 2.11, 2.11, 2.11, 2.09, 2.09, 2.26, 2.26, 2.47, 2.47, 1.41, 1.41, 1.44, 1.44, 1.96, 1.96]
    x5 = [220, 220, 223, 223, 221, 221, 190, 190, 180, 180, 180, 180, 176, 176, 175, 175]
    x6 = [261, 261, 260, 260, 261, 261, 323, 323, 364, 364, 300, 300, 299, 299, 301, 301]
    x7 = [87, 87, 87, 87, 92, 92, 75, 75, 71, 71, 64, 64, 64, 64, 75, 75]
    x8 = [1.8, 1.8, 16.6, 16.6, 23.0, 23.0, 25.1, 25.1, 26.1, 26.1, 20.0, 20.0, 20.5, 20.5, 17.3, 17.3]

    data = {'x1': x1, 'x2': x2, 'x3': x3, 'x4': x4,
            'x5': x5, 'x6': x6, 'x7': x7, 'x8': x8,
            'y': y}
    df = pd.DataFrame(data=data)

    print(tabulate(df, headers='keys', tablefmt='psql', showindex=False))

    return df


if __name__ == "__main__":
    main()
```

ParametersEstimator.py

```
from statsmodels.formula.api import ols
import numpy as np


class ParametersEstimator:
    'default constructor'
    def __init__(self, df, regressors, formula=''):
        self.alpha = 0.05
        self.df = df
        self.regressors = regressors
        if len(formula) == 0:
            self.formula = formula = 'y ~ ' + ('+'.join(list(regressors)))
        self.formula = formula
        self.regression_result = self.fit_selected_model(df,regressors,formula)

    def do_parameter_estimations(self):

        for counter in range(1,len(self.regression_result.pvalues)):
        #for regressor in self.regressors:
            pvalue = np.round(self.regression_result.pvalues[counter], 4)
            tvalue = np.round(self.regression_result.tvalues[counter], 4)

            if pvalue > self.alpha:
                print('tvalue = {}, pvalue = {}. {} is not significant.'.format(tvalue, pvalue,
[self.regressors[counter-1]]))
            else:
                print('tvalue = {}, pvalue = {}. {} is significant.'.format(tvalue, pvalue,
[self.regressors[counter-1]]))
```

```
    def fit_selected_model(self, df, regressors, formula=''):
        if len(formula) == 0:
            formula = 'y ~ ' + ('+'.join(list(regressors)))
        regression_result = ols(formula=formula, data=df).fit()
        print(regression_result.summary(xname=['Intercept']+regressors))
        return regression_result
```

ModelAdequacyChecker.py

```
import pandas as pd
from tabulate import tabulate
import matplotlib.pyplot as plt
import statsmodels.api as sm

def check_model_adequacy(regression_result):
    plot_residuals(regression_result)

def plot_residuals(regression_result):
    # Construct a normal probability plot of the residuals. Does there seem to be any
    # problem with the normality assumption?
    plot_normal_probability(regression_result)

    # Construct and interpret a plot of the residuals versus the predicted response.
    # Fitted vs. residuals
    plot_fitted_vs_residual(regression_result.fittedvalues, regression_result.resid, 'Fitted values',
                            'Residuals', 'Fitted vs. Residuals plot')

    # Compute the studentized residuals and the R - student residuals for this model.
    regression_influence = regression_result.get_influence()
    studentized_residuals = regression_influence.resid_studentized
    r_student_residuals = regression_influence.resid_studentized_external

    # Compute all other residuals(e.g., PRESS)
    press_residuals = regression_influence.resid_studentized_external

    residuals_data = {
        'Studentized Residuals': studentized_residuals,
        'R-student Residuals': r_student_residuals,
        'Press Residuals': press_residuals
    }
    residuals_df = pd.DataFrame(data=residuals_data)
    # start the dataframe index 1 for visual purpose
    residuals_df.index += 1
    print(tabulate(residuals_df, headers='keys', tablefmt='psql', showindex=True))

    # Fitted vs. studentized residuals
    plot_fitted_vs_residual(regression_result.fittedvalues, studentized_residuals, 'Fitted values',
                            'Studentized Residuals', 'Fitted vs. Studentized Residuals plot')
    # Fitted vs. r-studentized residuals
    plot_fitted_vs_residual(regression_result.fittedvalues, r_student_residuals, 'Fitted values',
                            'R Student Residuals', 'Fitted vs. R Student Residuals plot')
    # Fitted vs. press residuals
    plot_fitted_vs_residual(regression_result.fittedvalues, press_residuals, 'Fitted values',
                            'Press Residuals', 'Fitted vs. Press Residuals plot')

    plt.show()

def plot_normal_probability(regression):
    # Histogram of normalized residuals
    plt.figure(figsize=(8, 5))
    plt.hist(regression.resid_pearson, bins=20, edgecolor='k')
    plt.xlabel('Normalized residuals', fontsize=15)
    plt.title("Histogram of normalized residuals", fontsize=18)

    # Q - Q plot of the residuals
    plt.figure(figsize=(8, 5))
    probplot = sm.ProbPlot(regression.resid, fit='True')
```

```
    fig = probplot.qqplot(line='45')
    plt.title('Q-Q plot of normalized residuals')
    plt.grid(True)

def plot_fitted_vs_residual(x, y, xlabel, ylabel, title):
    plt.figure(figsize=(8, 5))
    plt.scatter(x=x, y=y, edgecolor='k')
    xmin = min(x)
    xmax = max(x)
    plt.hlines(y=0, xmin=xmin, xmax=xmax, color='red', linestyle='-', lw=2)
    plt.xlabel(xlabel, fontsize=15)
    plt.ylabel(ylabel, fontsize=15)
    plt.title(title, fontsize=18)
    plt.grid(True)
```

BestRegressorFinder.py

```
import itertools
from tabulate import tabulate
from statsmodels.formula.api import ols
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy import stats


def get_best_regressors(df,regressors):

    X = df.iloc[:, :-1]
    y = df.iloc[:, -1:]

    best_subset_selection(df,regressors)

    alpha = 0.25
    regressors_to_retain1 = stepwise_forward_selection(df,regressors, alpha)

    alpha = 0.10
    regressors_to_retain2 = stepwise_backward_elemination(df,regressors, alpha)

    alpha = 0.15
    regressors_to_retain3 = stepwise_regression(df,regressors, alpha)

    return pd.DataFrame({ 'forward_selection': [','.join(regressors_to_retain1)],
             'backward_elemination': [','.join(regressors_to_retain2)],
             'stepwise_regression': [','.join(regressors_to_retain3)],
             })


def fit_selected_model(df, regressors):
    formula = 'y ~ ' + ('+'.join(list(regressors)))
    regression_result = ols(formula=formula, data=df).fit()
    equation = 'y = ' + str(np.round(regression_result.params['Intercept'], 4))
    for reg in regressors:
        coefficient = np.round(regression_result.params[reg], 4)
        if coefficient > 0:
            equation += '+' + str(coefficient) + reg
        else:
            equation += str(coefficient) + reg
    rsquare_percentage = str(np.round(regression_result.rsquared * 100, 2)) + '%'
    return equation, rsquare_percentage


def stepwise_regression(df,all_regressors, threshold):
    y = df['y']
    included = []

    while True:
```

```
        changed = False
        n = len(y)
        # forward step
        excluded = list(set(all_regressors) - set(included))
        new_tval = pd.Series(index=excluded, dtype=float)
        included_reg_tval = pd.Series(index=excluded, dtype=float)
        models_df = pd.DataFrame(columns=['regressors', 'model'])
        regressors_under_test_list, models_list = [], []

        for new_column in excluded:
            regressors_under_test = included + [new_column]
            regressors = '+'.join(regressors_under_test)
            formula = 'y ~ ' + regressors
            if ('x1*x2' in regressors_under_test) and \
                (('x1' not in regressors_under_test) or ('x2' not in regressors_under_test)):
                continue;
            model = ols(formula=formula, data=df).fit()
            new_tval[new_column] = np.abs(model.tvalues[new_column.replace('*', ':')])
            for reg in included:
                included_reg_tval[reg] = np.abs(model.tvalues[new_column.replace('*', ':')])

            regressors_under_test_list.append(','.join(regressors_under_test))
            models_list.append(model)

        models_df['regressors'] = regressors_under_test_list;
        models_df['model'] = models_list;

        best_tval = new_tval.max()
        regressors_without_intercept = model.model.exog_names[1:]
        dof = n - (len(regressors_without_intercept) + 1)
        t_in = np.abs(np.round(stats.t.ppf(threshold / 2, dof), 4))
        t_out = np.abs(np.round(stats.t.ppf(threshold / 2, dof), 4))

        if best_tval > t_in:
            best_feature = new_tval.index[new_tval.argmax()]
            for reg in included:
                regressors_under_test = ','.join(included + [best_feature])
                index = models_df.index[models_df['regressors'] == regressors_under_test]
                model_info_for_regressors_under_test = models_df.loc[index]['model'].values[0]
                if np.abs(model_info_for_regressors_under_test.tvalues[reg]) < t_out:
                    included.remove(reg)
            included.append(best_feature)
            changed = True
        if not changed:
            break
    return included


def stepwise_backward_elemination(df,all_regressors, threshold_out):
    y = df['y']
    included = all_regressors
    K = len(included)

    while True:
        changed = False
        # backward step
        regressors = '+'.join(included)
        formula = 'y ~ ' + regressors

        model = ols(formula=formula, data=df).fit()
        r = K - len(included)
        # model = sm.OLS(y, sm.add_constant(pd.DataFrame(X[included]))).fit()
        p = K + 1 - r
        n = len(y)
        dof = n - p
        t_out = np.abs(np.round(stats.t.ppf(threshold_out / 2, dof), 4))
        # use all coefs except intercept
        tvalues = pd.Series(index=included, dtype=float)
        for reg in included:
```

```
            tvalues[reg] = (np.abs(model.tvalues.iloc[1:][reg.replace('*', ':')]))
        worst_tval = tvalues.min()
        if worst_tval < t_out:
            changed = True
            worst_feature = tvalues.index[tvalues.argmin()]
            included.remove(worst_feature)
        if not changed:
            break
    return included


def stepwise_forward_selection(df,all_regressors, threshold_in):
    y = df['y']
    included = []

    while True:
        changed = False
        n = len(y)
        # forward step
        excluded = list(set(all_regressors) - set(included))
        new_tval = pd.Series(index=excluded, dtype=float)
        for new_column in excluded:
            regressor_set = included + [new_column]
            if ('x1*x2' in regressor_set) and (('x1' not in regressor_set) or ('x2' not in regressor_set)):
                continue;
            regressors = '+'.join(regressor_set)
            formula = 'y ~ ' + regressors
            model = ols(formula=formula, data=df).fit()
            new_tval[new_column] = np.abs(model.tvalues[new_column.replace('*', ':')])
        best_tval = new_tval.max()
        regressors_without_intercept = model.model.exog_names[1:]
        dof = n - (len(regressors_without_intercept) + 1)
        t_in = np.abs(np.round(stats.t.ppf(threshold_in / 2, dof), 4))

        if best_tval > t_in:
            best_feature = new_tval.index[new_tval.argmax()]
            included.append(best_feature)
            changed = True
        if not changed:
            break
    return included


def best_subset_selection(df,complete_regressors):
    X = df.iloc[:, :-1]
    y = df.iloc[:, -1:]
    K = len(X.columns)  # number of regressors
    n = len(y)  # number of observations

    SSRes_list, R_squared_list, feature_list, MSRes_list = [], [], [], []
    Adj_R_squared_list, num_regressors = [], []

    for r in range(1, len(complete_regressors) + 1):
        for regressor_set in itertools.combinations(complete_regressors, r):
            regressors = '+'.join(list(regressor_set))
            formula = 'y ~ ' + regressors
            regression_model = ols(formula=formula, data=df).fit()
            p = len(regressor_set) + 1
            R_squared_list.append(regression_model.rsquared)
            r_sqr_adj = 1 - ((n - 1) / (n - p)) * (1 - regression_model.rsquared)
            Adj_R_squared_list.append(r_sqr_adj)
            SSRes_list.append(regression_model.ssr)
            feature_list.append(list(regressor_set))
            MSRes_list.append(regression_model.ssr / (n - p))
            num_regressors.append(len(regressor_set))

    p_list = np.array(num_regressors) + 1
    # lets take sigmasquare MSRes of full model, last element in the list
    hat_sigma_squared = MSRes_list[len(MSRes_list) - 1]  # same as np.min(SSRes_list)/(n - K - 1)
```

```
    C_p_list = SSRes_list / hat_sigma_squared - n + 2 * p_list

    regression_summary_df = pd.DataFrame(
        {'Number_of_Regressors': num_regressors,
         'p': p_list,
         'Regressors_in_the_model': feature_list,
         'SSRes_p': SSRes_list,
         'R_squared_p': R_squared_list,
         'Adj_R_square_p': Adj_R_squared_list,
         'MSRes_p': MSRes_list,
         'C_p': C_p_list
         })

    print(tabulate(regression_summary_df, headers='keys', tablefmt='psql', showindex=False))

    # last element contains full model information
    R_K_plus_1_Square = regression_summary_df['R_squared_p'][regression_summary_df.shape[0] - 1]

    alpha = 0.05
    F = np.round(stats.f.ppf(q=1 - alpha, dfn=K, dfd=(n - K - 1)), 4)
    d_alpha_n_k = K * F / (n - K - 1)

    R_0_Square = np.round(1 - (1 - R_K_plus_1_Square) * (1 + d_alpha_n_k), 4)

    regression_summary_df.where(regression_summary_df['R_squared_p'] > R_0_Square)[
        ['Regressors_in_the_model', 'R_squared_p']].dropna()

    # p vs R square
    plot_figure(regression_summary_df['p'], regression_summary_df['R_squared_p'],
                regression_summary_df['Regressors_in_the_model'], 'p', 'R_squared_p',
                'p Vs R_squared_p - Best subset selection')
    # p vs MSRes
    plot_figure(regression_summary_df['p'], regression_summary_df['MSRes_p'],
                regression_summary_df['Regressors_in_the_model'], 'p', 'MSRes_p',
                'p Vs MSRes_p - Best subset selection')
    # p vs Cp
    plot_Cp_vs_p_figure(regression_summary_df)

    plt.show()


def plot_figure(x, y, annotation_list, xlabel, ylabel, title):
    fig = plt.figure(figsize=(10, 8))
    ax = fig.add_subplot(111)

    ax.scatter(x, y, color='b')
    for i, txt in enumerate(annotation_list):
        ax.annotate(txt, (x[i], y[i]))

    ax.set_xlabel(xlabel)
    ax.set_ylabel(ylabel)
    ax.set_title(title)

def plot_Cp_vs_p_figure(regression_summary_df):
    fig = plt.figure(figsize=(15, 12))
    f, (ax, ax2) = plt.subplots(2,1,sharex=True)


    #top part
    uppper_plot_data = regression_summary_df.where(regression_summary_df['C_p'] > max(regression_summary_df.p))\
                        [['p', 'C_p']].dropna()
    ax.scatter(uppper_plot_data['p'], uppper_plot_data['C_p'], color='b')
    for i, txt in enumerate(regression_summary_df.Regressors_in_the_model):
         ax.annotate(txt, (regression_summary_df.p[i], regression_summary_df.C_p[i]))
    ax.set_xlim(0)
    ax.set_ylim([max(regression_summary_df.p),max(regression_summary_df.C_p)])
    ax.set_ylabel('Cp')
    ax.set_title('p Vs Cp - Best subset selection')
```

```
#bottom part
lower_plot_data = regression_summary_df.where(regression_summary_df['C_p'] <= max(regression_summary_df.p)) \
    [['p', 'C_p']].dropna()
ax2.scatter(lower_plot_data['p'], lower_plot_data['C_p'], color='b')
for i, txt in enumerate(regression_summary_df.Regressors_in_the_model):
    ax2.annotate(txt, (regression_summary_df.p[i], regression_summary_df.C_p[i]))
cp_p_line_data_p = [0,max(regression_summary_df.p)]
cp_p_line_data_cp = [0, regression_summary_df.C_p[len(regression_summary_df.C_p) - 1]] #cp of full model
ax2.plot(cp_p_line_data_p, cp_p_line_data_cp,  color='r', label='Cp = p')
ax2.set_xlim([0,max(regression_summary_df.p)+1])
ax2.set_ylim([0,max(regression_summary_df.p)])
ax2.set_xlabel('p')
ax2.set_ylabel('Cp')
ax2.legend(loc='upper left')

# hide the spines between ax and ax2
ax.spines['bottom'].set_visible(False)
ax2.spines['top'].set_visible(False)
ax.xaxis.tick_top()
ax.tick_params(labeltop=False)
ax2.xaxis.tick_bottom()
```