Indian Institute of Technology Kharagpur

Regression Analysis and Time Series Models

MA60280

Course Project

Topic: Forecasting insurance premium expenses based on customer characteristics using multiple linear regression

COURSE INSTRUCTOR:

Prof. Buddhananda Banerjee

GROUP MEMBERS:

S.No.	Names	Roll No.
1	Uday Om Srivastava	22ME3FP49
2	Srajan Kumar Gupta	22ME3FP18
3	Suraj Ghorai	22ME3FP47
4	Adeeba Alam Ansari	22IM3FP17
5	Desham Nihal Reddy	22IM3FP24
6	Hrushabh Prashant Bodhe	22ME3FP27

CONTENTS:

1. INTRODUCTION:

Multiple Linear Regression (MLR), also known as multiple regression, is a statistical method used to model the relationship between one dependent variable and several independent variables. Unlike simple linear regression, which uses just one predictor, MLR allows us to assess how multiple factors together influence an outcome. The goal is to create a model that accurately captures the linear relationships between the variables, enabling better predictions and deeper insights.

In this project, we aim to predict customer expenses based on various features such as Gender, Age, Body Mass Index (BMI), Smoking Status, and Number of Children. These variables are expected to contribute differently to the overall expenses, and our model will help identify and quantify these effects..

The analysis will be conducted using Python, a widely used programming language in data science due to its simplicity and powerful libraries like pandas, scikit-learn, and scipy. We will evaluate the model from both statistical and business viewpoints—using metrics like R-squared and p-values to measure performance, and interpreting the results to gain actionable insights.

Additionally, we will walk through the code step-by-step, explaining the key mathematical concepts and reasoning behind each part of the analysis to ensure a clear understanding of both the implementation and its practical significance.

2. ABOUT THE DATA SET:

This dataset contains a well-rounded set of features essential for analyzing and predicting insurance premium expenses. It includes **demographic variables** such as **age**, a continuous variable, and **gender**, a categorical one. These help capture general trends across different population groups.

Health indicators like **Body Mass Index (BMI)** are included as continuous variables and offer insight into an individual's physical condition, which can significantly impact healthcare costs. **Lifestyle factors** such as **smoking status** and **number of children** provide further context about personal habits and family structure, both of which can influence medical needs and insurance premiums.

Additionally, the dataset accounts for **geographic variation** by categorizing the individual's residence into one of four regions: **Southwest, Southeast, Northwest**, and **Northeast**. This allows for the consideration of regional differences in healthcare access and pricing.

The primary outcome variable is the **annual healthcare expense**, recorded in U.S. dollars. This comprehensive set of features makes the dataset suitable for building robust predictive models and gaining meaningful insights into the factors influencing insurance costs.

3. OBJECTIVE:

The main goal of this project is twofold: first, to gain a comprehensive understanding of the dataset and carry out any required preprocessing to handle inconsistencies or anomalies. Second, to build regression models that can accurately predict housing prices—initially using a single feature and later incorporating multiple features. The performance of these models will be evaluated and compared using metrics such as **R-squared (R^2)**, **Root Mean Squared Error (RMSE)**, and others.

4. ATTRIBUTES:

The dataset comprises various attributes, including both continuous and categorical types. Here is a breakdown of the data attributes:

- **Age:** Records the customer's age in years, representing a continuous variable.
- **Sex:** Indicates the gender of each customer, with only two possible values - Male and Female.
- **Body Mass Index (BMI):** Reflects the BMI of the customer, also a continuous variable.
- **Children:** Indicates the number of children the customer has.
- **Smoker:** Reflects the smoking behavior of the customer.
- **Region:** Specifies the geographical region of the customer's residence, categorized into four groups - Southwest, Southeast, Northwest, and Northeast.
- **Expenses:** Records the customer's annual expenses in dollars.

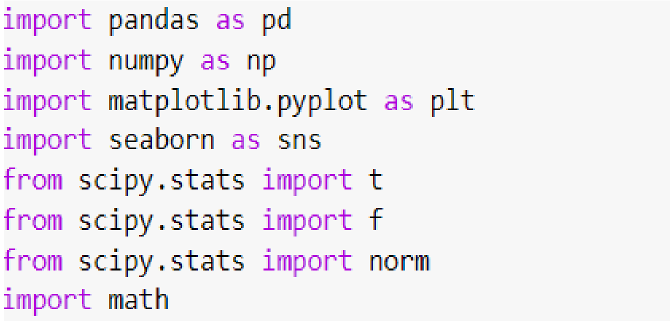
The link to the code : rtsm.ipynb
The link to the dataset: Drive Link

5. DATA ANALYSIS AND PREPROCESSING:

5.1 Materials Considered:

In our analysis, we'll utilize the following Python libraries:

- **Pandas:** Employed for data loading and analysis.
- **Numpy:** Utilized for matrix analysis.
- **Matplotlib:** Employed for plotting curves.
- **Scipy:** Utilized for statistical tables and values.
- **Seaborn:** Employed for data visualization.
- **Math:** Used for verification of assumptions of MLR

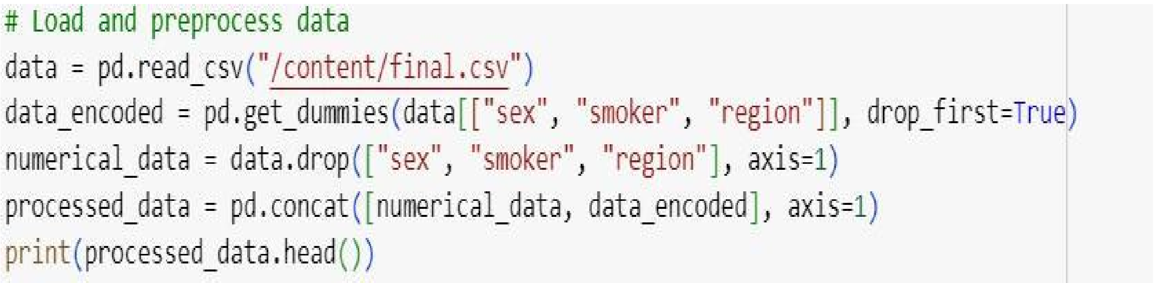


```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import t
from scipy.stats import f
from scipy.stats import norm
import math
```

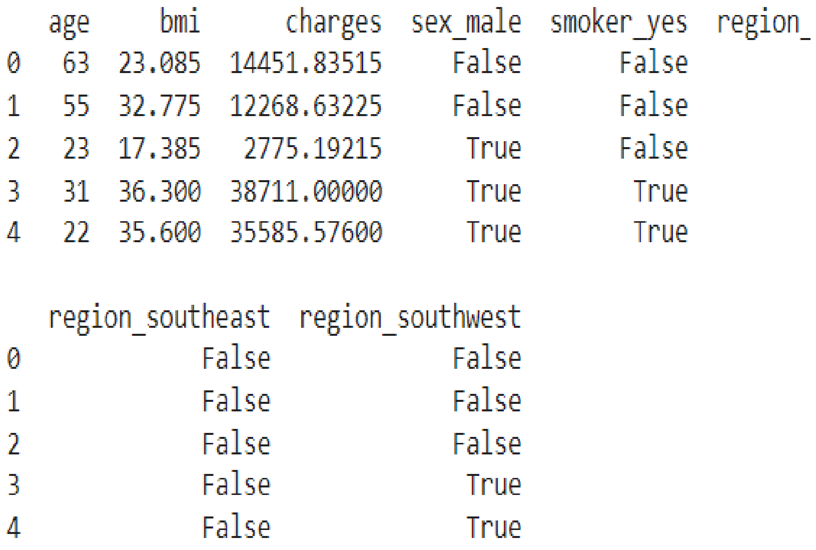
5.2 Preprocessing the Data:

The dataset comprises numerous categorical attributes. Handling these categories necessitates converting them into numerical variables. There are two primary methods for such conversion. The first involves assigning categories to numbers, like A for 1, B for 2, etc. However, this approach presents several challenges. Firstly, determining which numbers correspond to which categories is unclear. Secondly, directly assigning numbers implies an ordinal relationship among the categories, even when it may not exist. The preferred method to address this is through one-hot encoding.

One-hot encoding allows us to incorporate categorical features into regression analysis, as categorical features lacking inherent ordering can shift the regression line parallelly. For instance, if a feature has three categories (A, B, C), it can be encoded as (1,0,0), (0,1,0), (0,0,1), respectively, effectively breaking down the feature into three distinct categories. Notably, when one-hot encoding is applied, an additional term is added to the constant term for each category. Therefore, to accommodate the presence of category A, for instance, we drop the first column from each one-hot vector, converting the encoding to (0,0), (1,0), (0,1). This adjustment ensures that the constant term accommodates the A category itself. Hence, we proceed by dropping the first column from each one-hot vector.



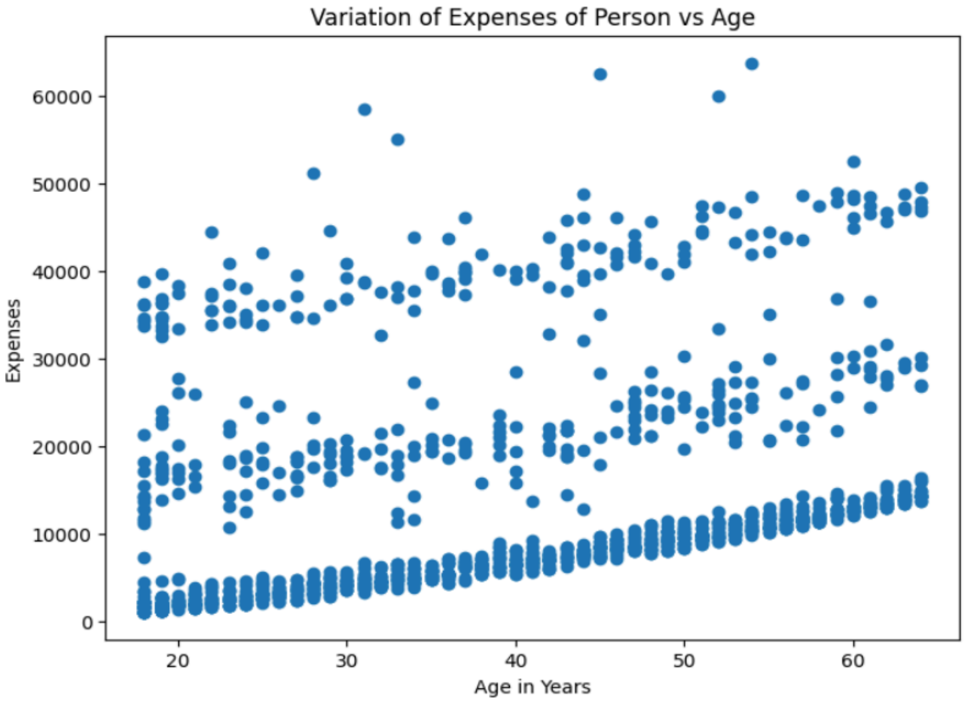
```
# Load and preprocess data
data = pd.read_csv("/content/final.csv")
data_encoded = pd.get_dummies(data[["sex", "smoker", "region"]], drop_first=True)
numerical_data = data.drop(["sex", "smoker", "region"], axis=1)
processed_data = pd.concat([numerical_data, data_encoded], axis=1)
print(processed_data.head())
```

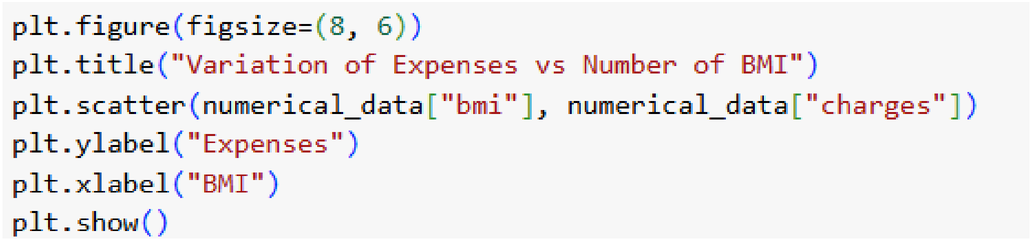


```
   age     bmi      charges  sex_male  smoker_yes  region_
0   63  23.085  14451.83515     False       False
1   55  32.775  12268.63225     False       False
2   23  17.385   2775.19215      True       False
3   31  36.300  38711.00000      True        True
4   22  35.600  35585.57600      True        True

   region_southeast  region_southwest
0             False             False
1             False             False
2             False             False
3             False              True
4             False              True
```

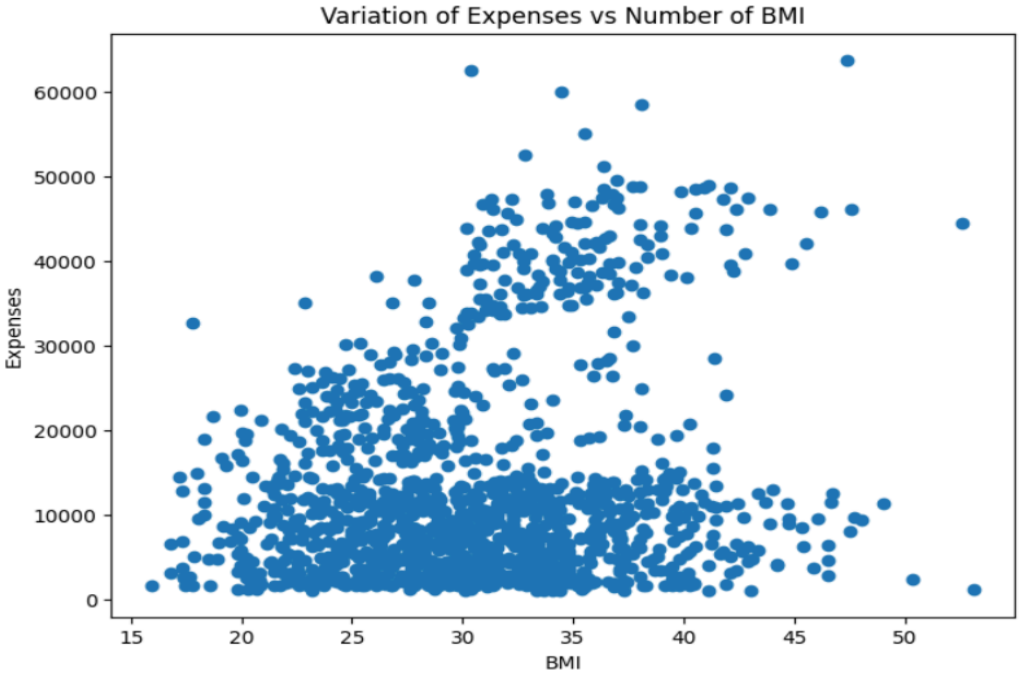
5.2 Exploratory Data Analysis:

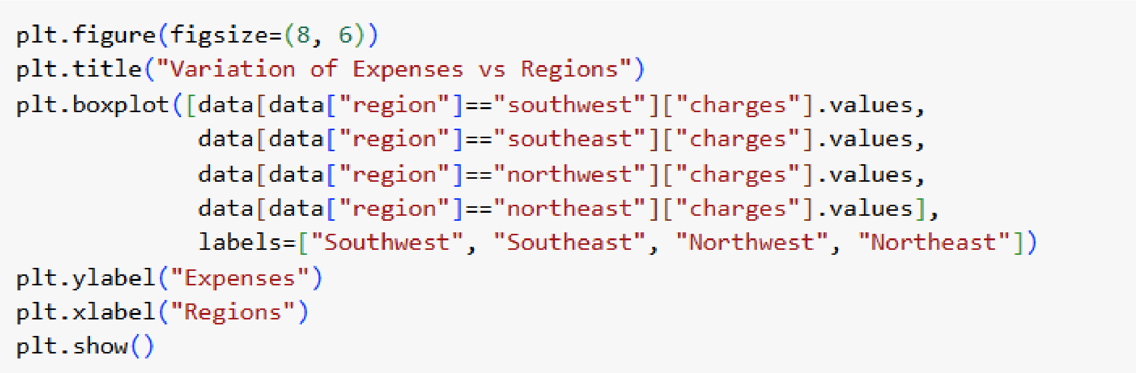
```
# Visualize data
plt.figure(figsize=(8, 6))
plt.title("Variation of Expenses of Person vs Age")
plt.scatter(numerical_data["age"], numerical_data["charges"])
plt.ylabel("Expenses")
plt.xlabel("Age in Years")
plt.show()
```



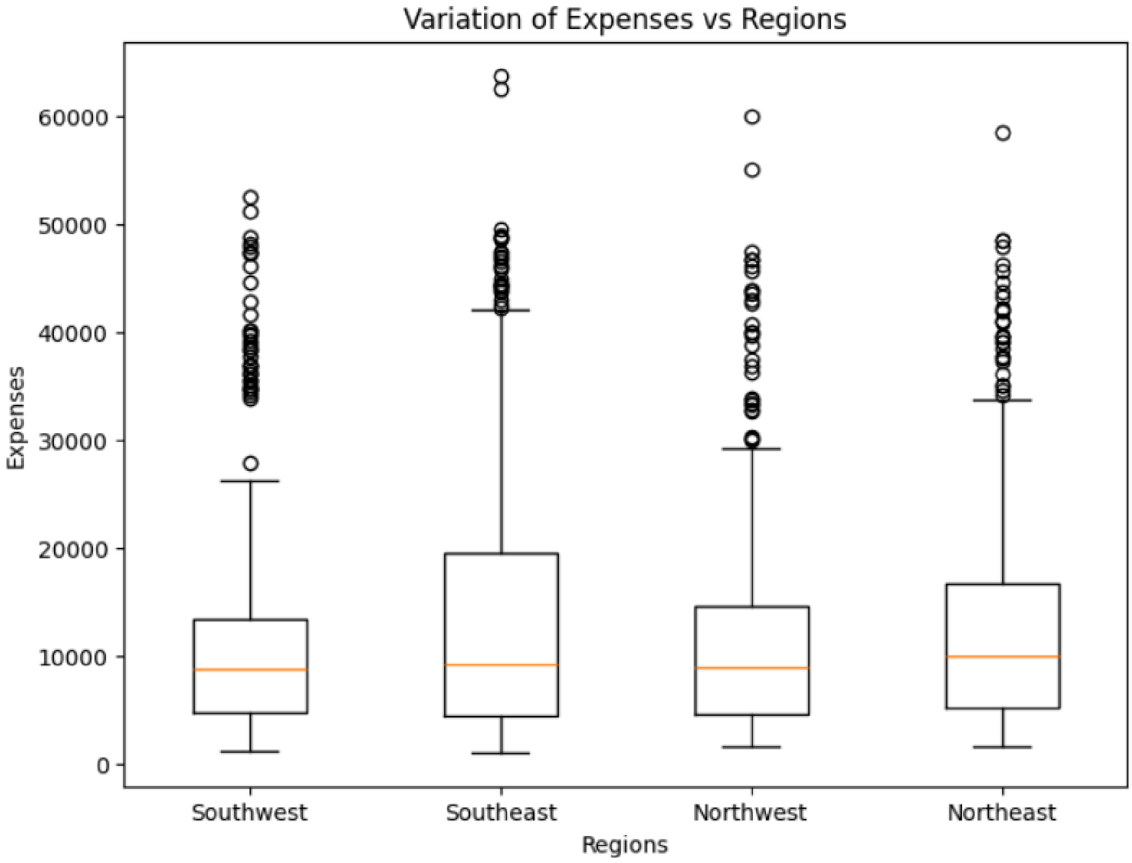


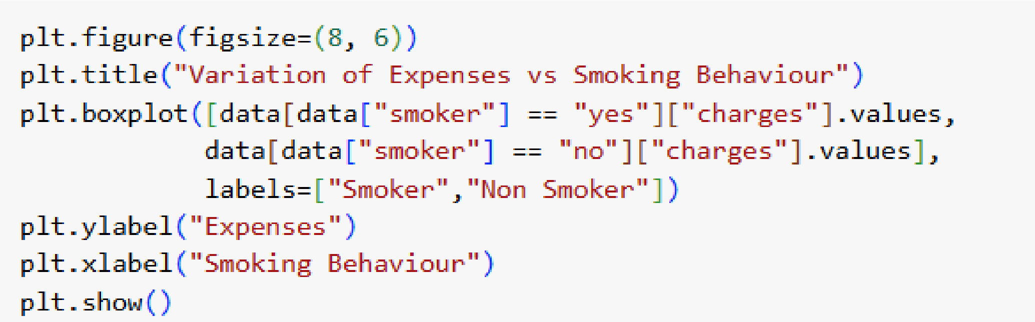
```
plt.figure(figsize=(8, 6))
plt.title("Variation of Expenses vs Number of BMI")
plt.scatter(numerical_data["bmi"], numerical_data["charges"])
plt.ylabel("Expenses")
plt.xlabel("BMI")
plt.show()
```



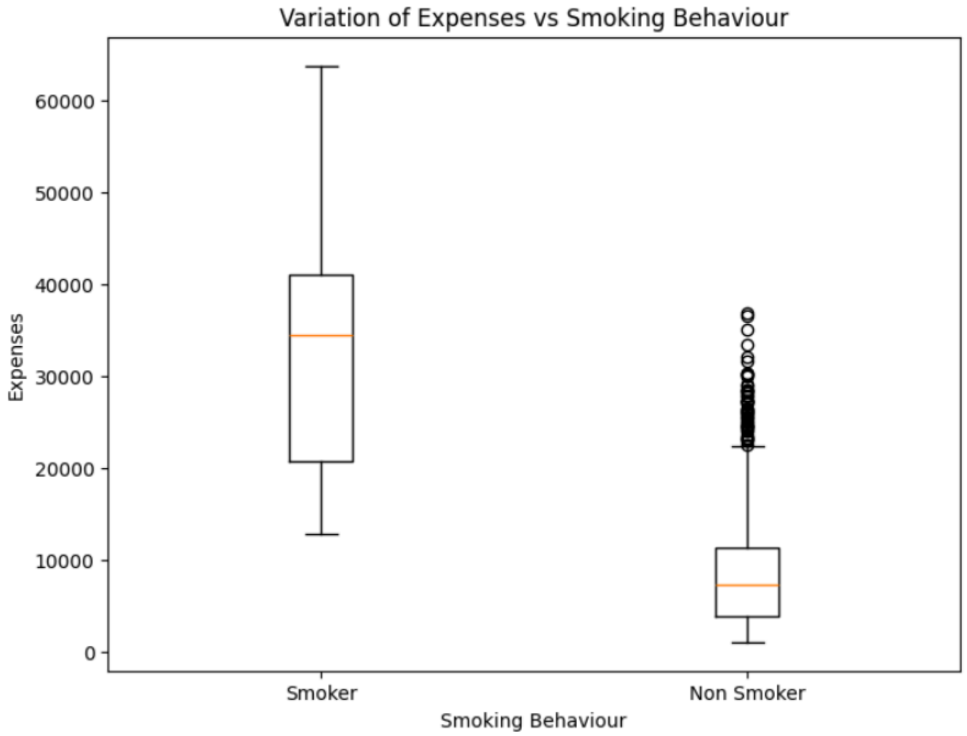


```
plt.figure(figsize=(8, 6))
plt.title("Variation of Expenses vs Regions")
plt.boxplot([data[data["region"]=="southwest"]["charges"].values,
             data[data["region"]=="southeast"]["charges"].values,
             data[data["region"]=="northwest"]["charges"].values,
             data[data["region"]=="northeast"]["charges"].values],
             labels=["Southwest", "Southeast", "Northwest", "Northeast"])
plt.ylabel("Expenses")
plt.xlabel("Regions")
plt.show()
```

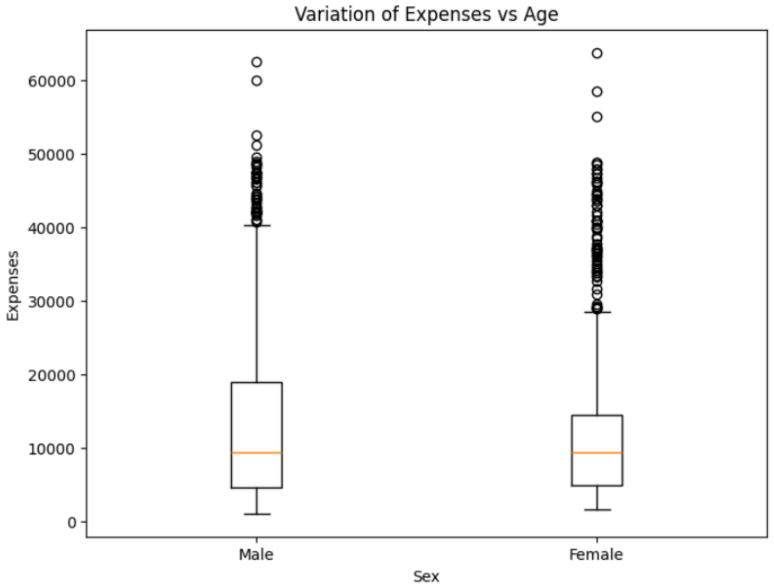


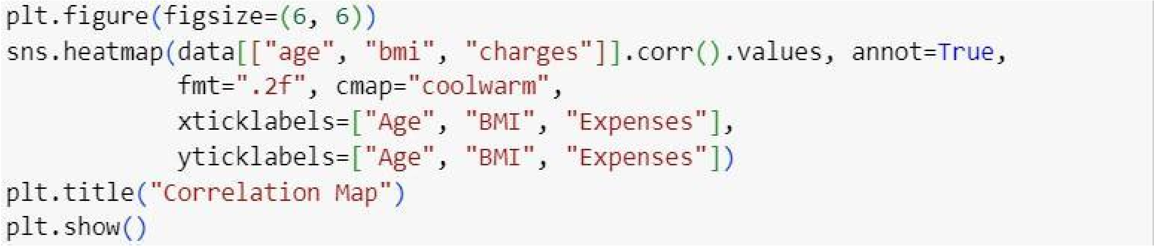


```
plt.figure(figsize=(8, 6))
plt.title("Variation of Expenses vs Smoking Behaviour")
plt.boxplot([data[data["smoker"] == "yes"]["charges"].values,
            data[data["smoker"] == "no"]["charges"].values],
            labels=["Smoker","Non Smoker"])
plt.ylabel("Expenses")
plt.xlabel("Smoking Behaviour")
plt.show()
```

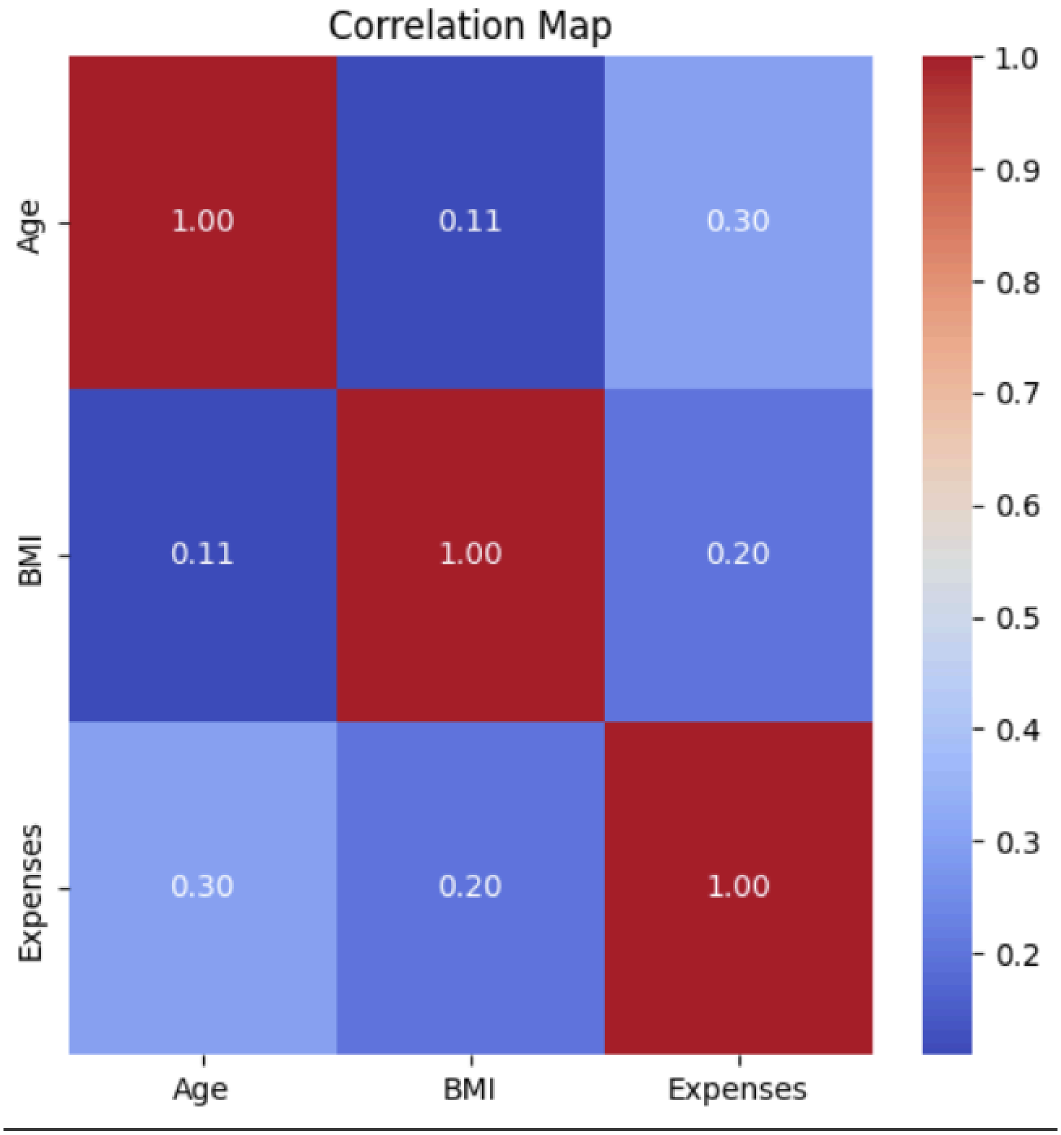


```
plt.figure(figsize=(8, 6))
plt.title("Variation of Expenses vs Age")
plt. boxplot([data[data["sex"] == "male"]["charges"].values,
             data[data["sex"] == "female"]["charges"].values],
             labels=["Male", "Female"])
plt.ylabel("Expenses")
plt.xlabel("Sex")
plt.show()
```





```python
plt.figure(figsize=(6, 6))
sns.heatmap(data[["age", "bmi", "charges"]].corr().values, annot=True,
            fmt=".2f", cmap="coolwarm",
            xticklabels=["Age", "BMI", "Expenses"],
            yticklabels=["Age", "BMI", "Expenses"])
plt.title("Correlation Map")
plt.show()
```

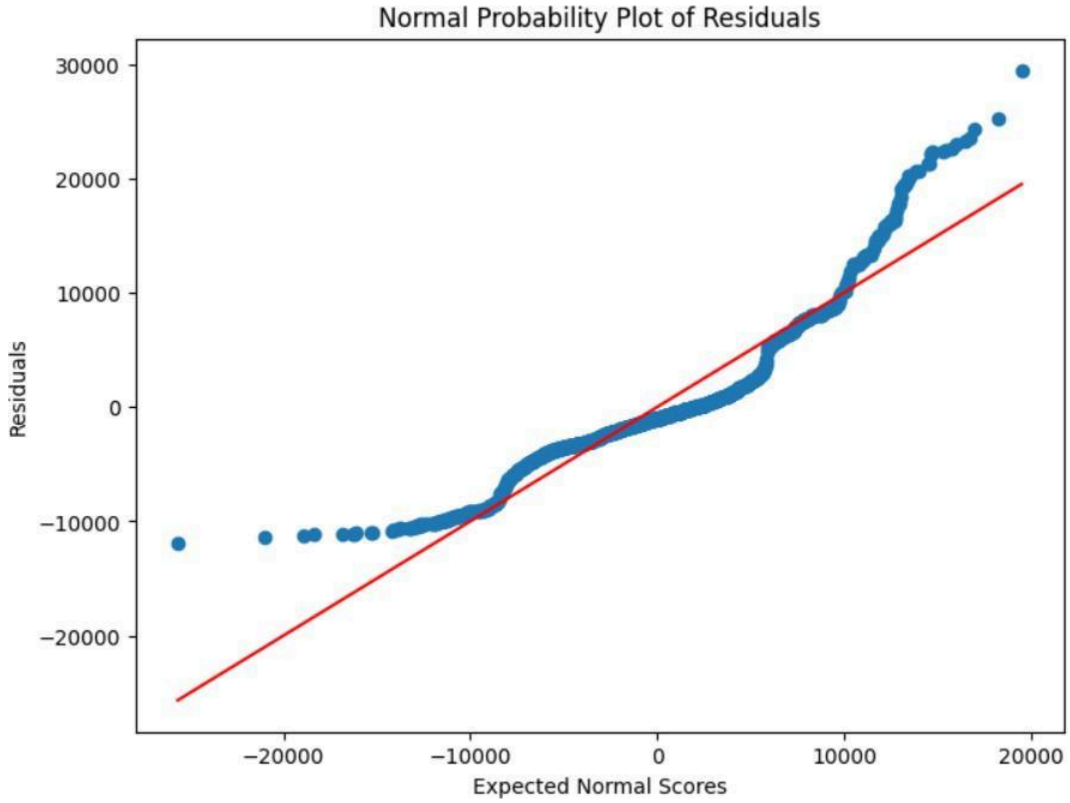


6. VERIFYING THE ASSUMPTIONS OF MULTIPLE LINEAR REGRESSION:

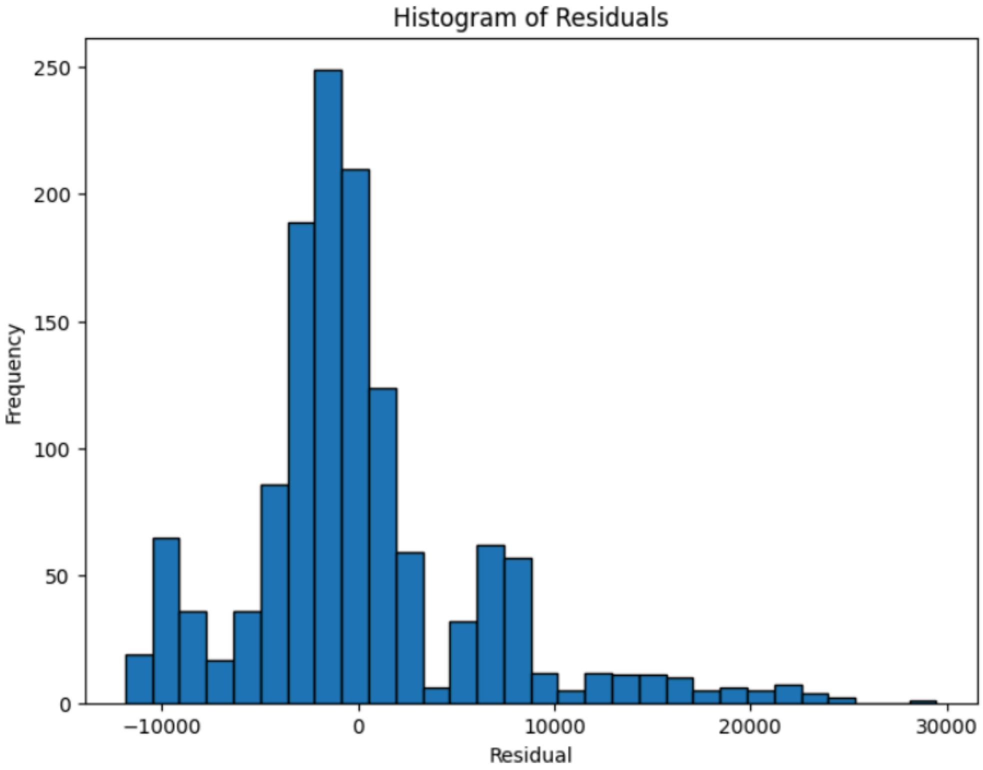
To effectively apply linear regression to our dataset, we first examined whether it satisfies the key underlying assumptions:

- **Normality**: The residuals should follow a normal distribution.
- **Heteroskedasticity**: The variance of the errors should remain constant.
- **Matrix Invertibility**: The feature matrix must be invertible.
- **No Multicollinearity**: Independent variables should not be highly correlated.

To assess the **normality** of residuals, we employed a Q-Q plot. The points in the plot closely followed a straight line, suggesting that the residuals are approximately normally distributed. This supports the assumption of normality, validating one of the key requirements for linear regression.



By plotting the residuals against their frequency, we observed a bell-shaped curve, suggesting that the errors follow a normal distribution.

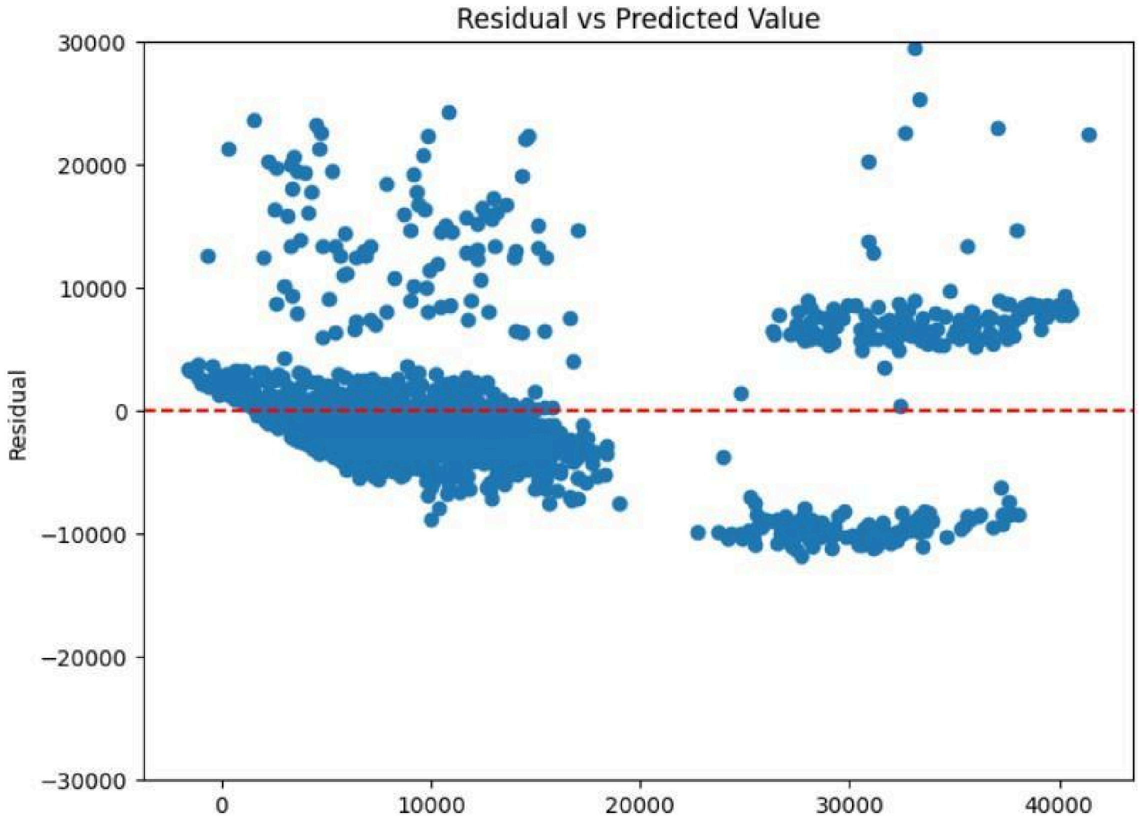


We conducted an analysis to assess **heteroskedasticity**, which refers to the non-constant variance of error terms. By plotting the residuals against their predicted values, we noticed a clear non-linear pattern, particularly evident at higher predicted values. This indicates the potential presence of heteroskedasticity in the dataset.

Additionally, we identified **outliers** within the residuals, highlighted in red on the plot. These were defined as data points that lie more than three standard deviations from zero, suggesting they deviate significantly from the general error pattern.

Next, we turned our attention to verifying the assumption of **matrix invertibility**. To do so, we performed a matrix multiplication between the original feature matrix and its inverse. The result was a non-zero value, confirming that the matrix is indeed invertible.

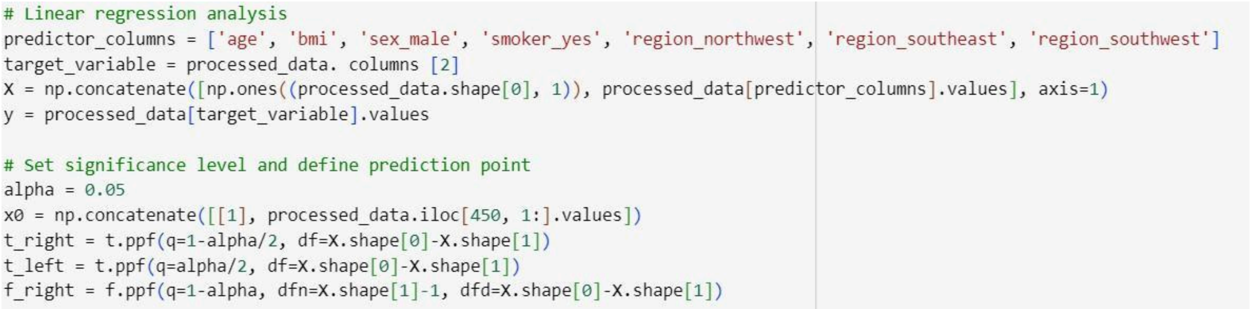
Given that the dataset satisfies key assumptions—**approximately normal error distribution** and the **invertibility of the (X^TX) matrix**—we conclude that it is well-suited for linear regression analysis.



7. REGRESSION ANALYSIS:

For our regression analysis, we employed the **backward selection** method. This approach begins with all predictor variables included in the model, and progressively removes the least significant ones until only statistically meaningful predictors remain. The analysis is conducted at a **95% confidence level**.

As part of the process, we estimate regression coefficients, assess overall model fit, and evaluate the **standard errors** and **p-values** of the coefficients. We also compute **confidence intervals** for the mean predictions, generate **prediction intervals** for individual outcomes, and perform thorough **error analysis** to evaluate model accuracy and reliability.



```
# Linear regression analysis
predictor_columns = ['age', 'bmi', 'sex_male', 'smoker_yes', 'region_northwest', 'region_southeast', 'region_southwest']
target_variable = processed_data. columns [2]
X = np.concatenate([np.ones((processed_data.shape[0], 1)), processed_data[predictor_columns].values], axis=1)
y = processed_data[target_variable].values

# Set significance level and define prediction point
alpha = 0.05
x0 = np.concatenate([[1], processed_data.iloc[450, 1:].values])
t_right = t.ppf(q=1-alpha/2, df=X.shape[0]-X.shape[1])
t_left = t.ppf(q=alpha/2, df=X.shape[0]-X.shape[1])
f_right = f.ppf(q=1-alpha, dfn=X.shape[1]-1, dfd=X.shape[0]-X.shape[1])
```

7.1 Estimating the Coefficients;

The coefficients and their errors are estimated using the following formulas. Firstly, we compute the C matrix as:

$$C = (X^T X)^{-1}$$

Next, we calculate the Hat matrix as:

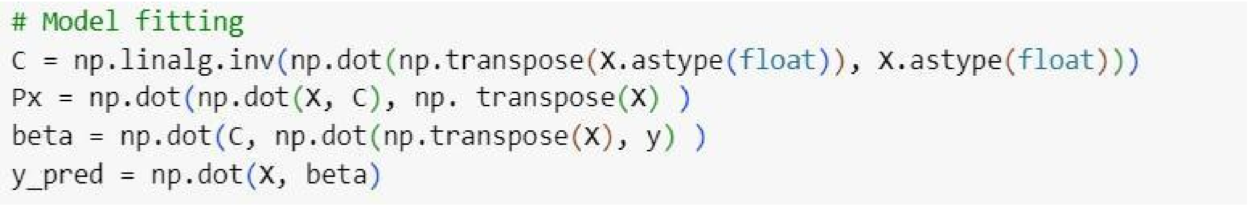
$$P_x = X(X^T X)^{-1} X^T$$

The coefficients are estimated as:

$$\beta = (X^T X)^{-1} X^T y$$

The predicted value of y is given by:

$$\hat{y} = X(X^T X)^{-1} X^T y$$



```
# Model fitting
C = np.linalg.inv(np.dot(np.transpose(X.astype(float)), X.astype(float)))
Px = np.dot(np.dot(X, C), np. transpose(X) )
beta = np.dot(C, np.dot(np.transpose(X), y) )
y_pred = np.dot(X, beta)
```

7.2 Assessing the Model Fit:

The error is defined as the difference between the actual and predicted values:

$$e = y - \hat{y}$$

We predict using the least squares method, defining the Sum of Squares of Regression (SSR), the Sum of Squares of Error (SSE), and the Total Sum of Squares (TSS).

SSE (Sum of Squares of Error) measures the discrepancy between the actual and predicted values:

$$SSE = y^T(I_n - P_x)y$$

SSR (Sum of Squares of Regression) represents the variation explained by the regression model:

$$SSR = y^T[P_x - \frac{1}{n} * 11^T]y$$

TSS (Total Sum of Squares) captures the total variability in the response variable:

$$TSS = y^T[I_n - \frac{1}{n} * 11^T]y$$

To evaluate the regression model, we compute the **MSR (Mean Square of Regression)**:

$$MSR = \frac{SSR}{k}$$

MSE (Mean Square of Error):

$$MSE = \frac{SSE}{n - k - 1}$$

MSS (Mean Sum of Squares):

$$MSS = \frac{TSS}{n - 1}$$

To test the model’s effectiveness and goodness of fit, we calculate the following:

R-squared (R^2): Proportion of variance in the dependent variable explained by the model.

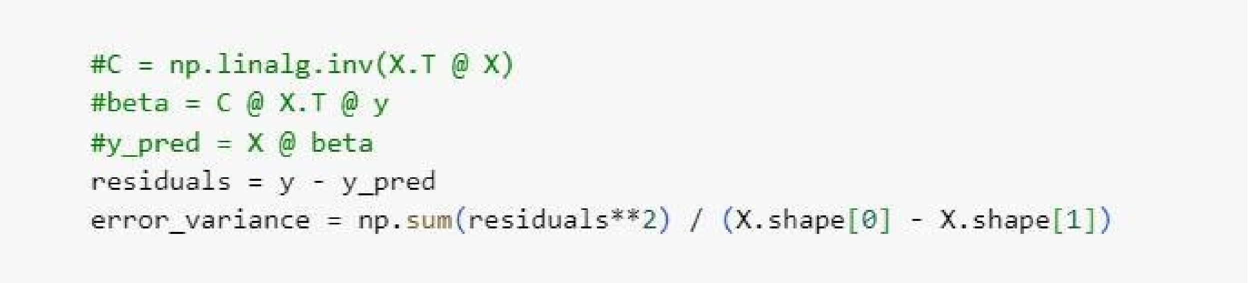
$$R^{2} = 1 - \frac{SSE}{TSS}$$

Adjusted R-squared R^{2}_{adj}: Adjusts R^{2} for the number of predictors in the model.

$$R^{2}_{adj} = 1 - \left[\frac{SSE\,/\,(n-k-1)}{TSS\,/\,(n-1)}\right]$$

F-statistic: Evaluates whether the regression model explains a significant amount of variance.

$$F_{stat} = \frac{MSR}{MSE}$$



```
#C = np.linalg.inv(X.T @ X)
#beta = C @ X.T @ y
#y_pred = X @ beta
residuals = y - y_pred
error_variance = np.sum(residuals**2) / (X.shape[0] - X.shape[1])
```

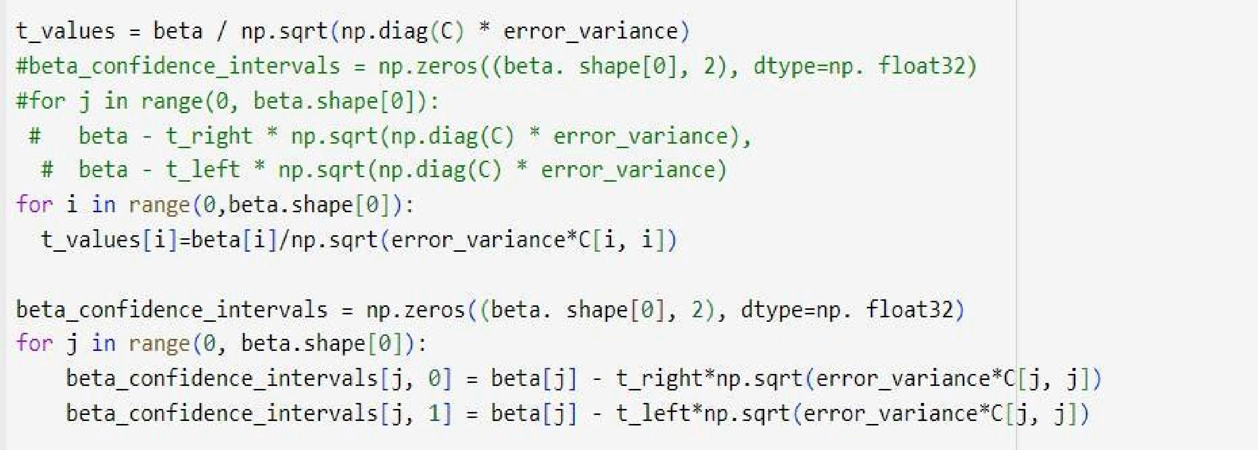
7.3 Test of Hypothesis for the Estimates of Coefficient:

Since all the coefficients are estimated from the given data, the standard error in the estimates is given by

$$t_j = \frac{\beta_j}{\sqrt{\hat{\sigma}^2 C_{jj}}}$$

The confidence interval for the estimates of coefficients is given by

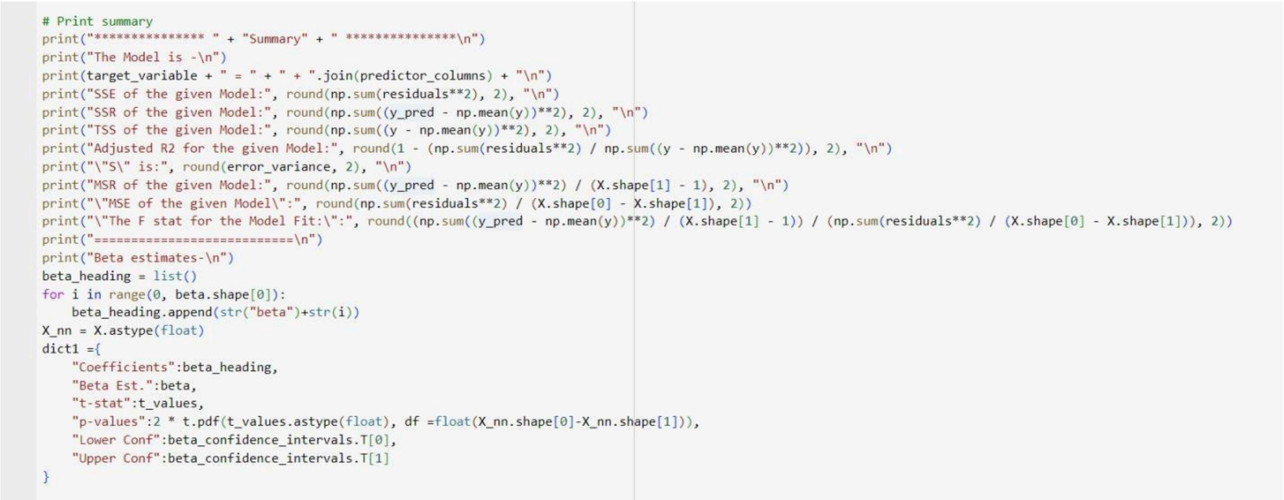
$$\hat{\beta}_j - t_{\alpha/2,n-k-1}\sqrt{\frac{\sigma_b^2}{C_{jj}}} \leq \beta_j \leq \hat{\beta}_j + t_{\alpha/2,n-k-1}\sqrt{\frac{\sigma_b^2}{C_{jj}}}$$



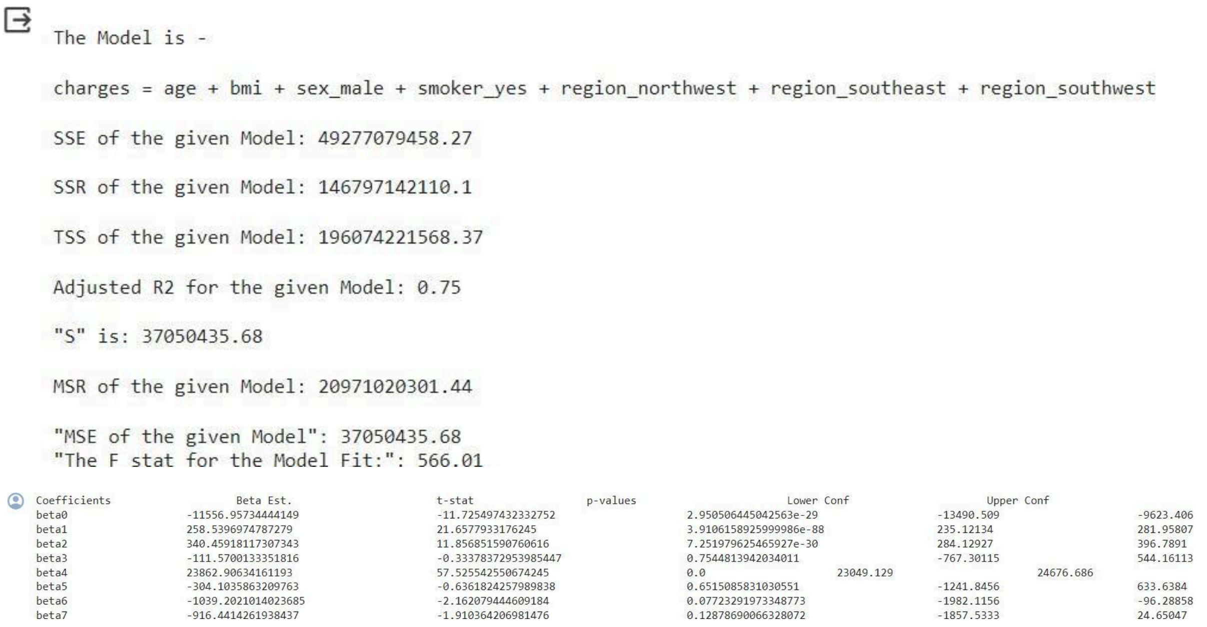
```
t_values = beta / np.sqrt(np.diag(C) * error_variance)
#beta_confidence_intervals = np.zeros((beta. shape[0], 2), dtype=np. float32)
#for j in range(0, beta.shape[0]):
 #   beta - t_right * np.sqrt(np.diag(C) * error_variance),
  #  beta - t_left * np.sqrt(np.diag(C) * error_variance)
for i in range(0,beta.shape[0]):
  t_values[i]=beta[i]/np.sqrt(error_variance*C[i, i])

beta_confidence_intervals = np.zeros((beta. shape[0], 2), dtype=np. float32)
for j in range(0, beta.shape[0]):
    beta_confidence_intervals[j, 0] = beta[j] - t_right*np.sqrt(error_variance*C[j, j])
    beta_confidence_intervals[j, 1] = beta[j] - t_left*np.sqrt(error_variance*C[j, j])
```

7.4 ANOVA Table:

The ANOVA table of our model is as follows



```
# Print summary
print("*************** " + "Summary" + " ***************\n")
print("The Model is -\n")
print(target_variable + " = " + " + ".join(predictor_columns) + "\n")
print("SSE of the given Model:", round(np.sum(residuals**2), 2), "\n")
print("SSR of the given Model:", round(np.sum((y_pred - np.mean(y))**2), 2), "\n")
print("TSS of the given Model:", round(np.sum((y - np.mean(y))**2), 2), "\n")
print("Adjusted R2 for the given Model:", round(1 - (np.sum(residuals**2) / np.sum((y - np.mean(y))**2)), 2), "\n")
print("\"S\" is:", round(error_variance, 2), "\n")
print("MSR of the given Model:", round(np.sum((y_pred - np.mean(y))**2) / (X.shape[1] - 1), 2), "\n")
print("\"MSE of the given Model\":", round(np.sum(residuals**2) / (X.shape[0] - X.shape[1]), 2))
print("\"The F stat for the Model Fit:\":", round((np.sum((y_pred - np.mean(y))**2) / (X.shape[1] - 1)) / (np.sum(residuals**2) / (X.shape[0] - X.shape[1])), 2))
print("===========================\n")
print("Beta estimates-\n")
beta_heading = list()
for i in range(0, beta.shape[0]):
    beta_heading.append(str("beta")+str(i))
X_nn = X.astype(float)
dict1 ={
    "Coefficients":beta_heading,
    "Beta Est.":beta,
    "t-stat":t_values,
    "p-values":2 * t.pdf(t_values.astype(float), df =float(X_nn.shape[0]-X_nn.shape[1])),
    "Lower Conf":beta_confidence_intervals.T[0],
    "Upper Conf":beta_confidence_intervals.T[1]
}
```



```
The Model is -

charges = age + bmi + sex_male + smoker_yes + region_northwest + region_southeast + region_southwest

SSE of the given Model: 49277079458.27

SSR of the given Model: 146797142110.1

TSS of the given Model: 196074221568.37

Adjusted R2 for the given Model: 0.75

"S" is: 37050435.68

MSR of the given Model: 20971020301.44

"MSE of the given Model": 37050435.68
"The F stat for the Model Fit:": 566.01
```

```
Coefficients        Beta Est.               t-stat                 p-values             Lower Conf              Upper Conf
beta0          -11556.95734444149      -11.725497432332752                   2.950506445042563e-29          -13490.509              -9623.406
beta1          258.5396974787279       21.6577933176245                      3.9106158925999986e-88         235.12134               281.95807
beta2          340.45918117307343      11.856851590760616                    7.251979625465927e-30          284.12927               396.7891
beta3          -111.5700133351816      -0.33378372953985447                  0.7544813942034011             -767.30115              544.16113
beta4          23862.90634161193       57.525542550674245                    0.0              23049.129             24676.686
beta5          -304.1035863209763      -0.6361824257989838                   0.6515085831030551             -1241.8456              633.6384
beta6          -1039.2021014023685     -2.162079444609184                    0.07723291973348773            -1982.1156              -96.28858
beta7          -916.4414261938437      -1.910364206981476                    0.12878690066328072            -1857.5333              24.65047
```

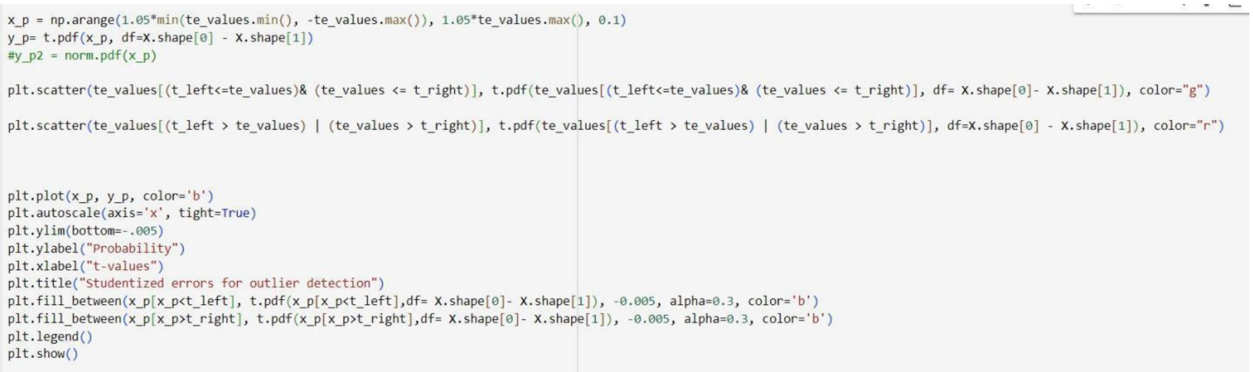
7.5 Mean Confidence Interval and Prediction Intervals:

The Mean confidence interval for a given prediction vector is computed as

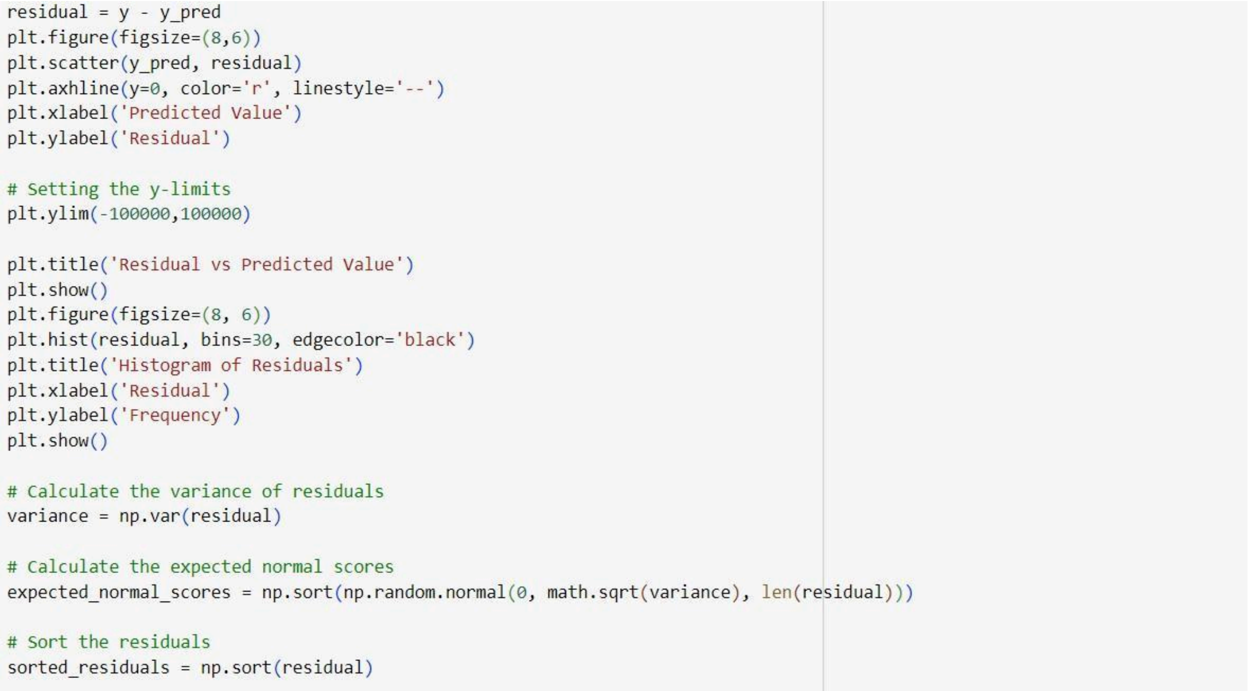
$$\bar{y}_0 - t_{\alpha/2,n-k-1}\sqrt{\sigma_b^2 x_0^T (X^T X)^{-1} x_0} \leq E(y|x) \leq \bar{y}_0 + t_{\alpha/2,n-k-1}\sqrt{\sigma_b^2 x_0^T (X^T X)^{-1} x_0}$$

And last, we have the prediction interval for a given vector given by

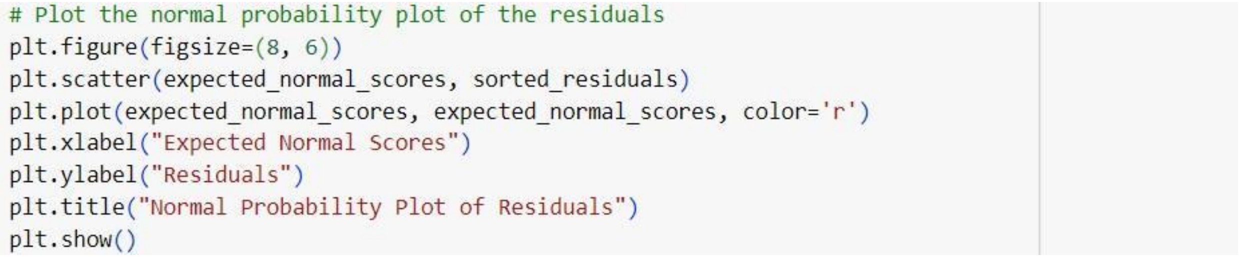
$$\bar{y}_0 - t_{\alpha/2,n-k-1}\sqrt{\sigma_b^2 + \sigma_b^2 x_0^T (X^T X)^{-1} x_0} \leq \bar{y} \leq \bar{y}_0 + t_{\alpha/2,n-k-1}\sqrt{\sigma_b^2 + \sigma_b^2 x_0^T (X^T X)^{-1} x_0}$$



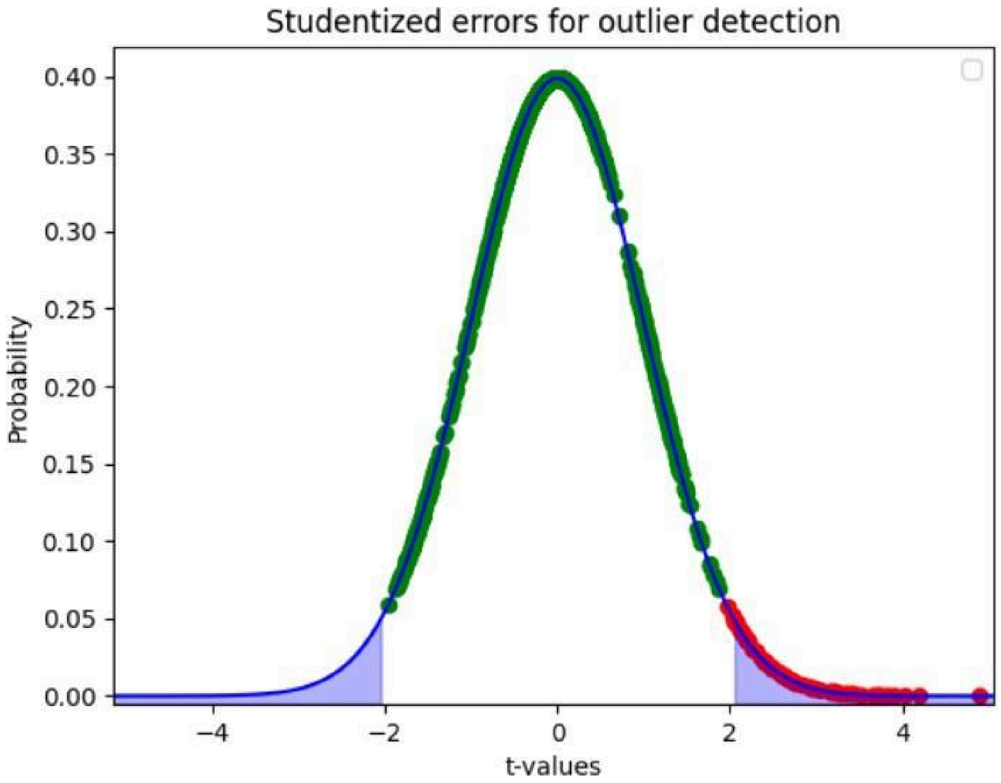
```
x_p = np.arange(1.05*min(te_values.min(), -te_values.max()), 1.05*te_values.max(), 0.1)
y_p= t.pdf(x_p, df=X.shape[0] - X.shape[1])
#y_p2 = norm.pdf(x_p)

plt.scatter(te_values[(t_left<=te_values)& (te_values <= t_right)], t.pdf(te_values[(t_left<=te_values)& (te_values <= t_right)], df= X.shape[0]- X.shape[1]), color="g")

plt.scatter(te_values[(t_left > te_values) | (te_values > t_right)], t.pdf(te_values[(t_left > te_values) | (te_values > t_right)], df=X.shape[0] - X.shape[1]), color="r")



plt.plot(x_p, y_p, color='b')
plt.autoscale(axis='x', tight=True)
plt.ylim(bottom=-.005)
plt.ylabel("Probability")
plt.xlabel("t-values")
plt.title("Studentized errors for outlier detection")
plt.fill_between(x_p[x_p<t_left], t.pdf(x_p[x_p<t_left],df= X.shape[0]- X.shape[1]), -0.005, alpha=0.3, color='b')
plt.fill_between(x_p[x_p>t_right], t.pdf(x_p[x_p>t_right],df= X.shape[0]- X.shape[1]), -0.005, alpha=0.3, color='b')
plt.legend()
plt.show()
```

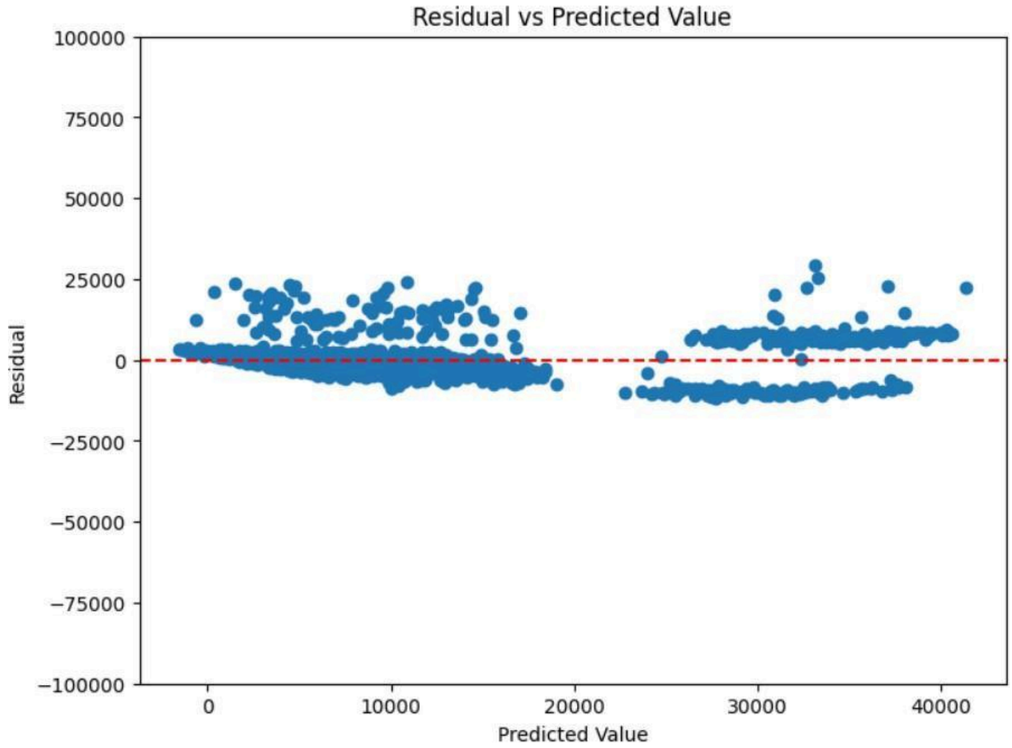


```
residual = y - y_pred
plt.figure(figsize=(8,6))
plt.scatter(y_pred, residual)
plt.axhline(y=0, color='r', linestyle='--')
plt.xlabel('Predicted Value')
plt.ylabel('Residual')

# Setting the y-limits
plt.ylim(-100000,100000)

plt.title('Residual vs Predicted Value')
plt.show()
plt.figure(figsize=(8, 6))
plt.hist(residual, bins=30, edgecolor='black')
plt.title('Histogram of Residuals')
plt.xlabel('Residual')
plt.ylabel('Frequency')
plt.show()

# Calculate the variance of residuals
variance = np.var(residual)

# Calculate the expected normal scores
expected_normal_scores = np.sort(np.random.normal(0, math.sqrt(variance), len(residual)))

# Sort the residuals
sorted_residuals = np.sort(residual)
```



```
# Plot the normal probability plot of the residuals
plt.figure(figsize=(8, 6))
plt.scatter(expected_normal_scores, sorted_residuals)
plt.plot(expected_normal_scores, expected_normal_scores, color='r')
plt.xlabel("Expected Normal Scores")
plt.ylabel("Residuals")
plt.title("Normal Probability Plot of Residuals")
plt.show()
```


Studentized errors for outlier detection
Probability
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−4
−2
0
2
4
t-values


Residual vs Predicted Value
Residual
100000
75000
50000
25000
0
−25000
−50000
−75000
−100000
0
10000
20000
30000
40000
Predicted Value

7.6 Error Analysis:

The distribution of the estimated errors relative to the predicted values shows signs of deviating from normality. Additionally, the error plot hints at the existence of multiple distinct groups within the dataset, potentially arising from underlying cluster patterns. To investigate this further, incorporating more categorical variables could help in effectively distinguishing these subgroups.

Nevertheless, these errors can be standardized using the following formula:

$$se_i = \frac{e_i}{\sqrt{\widehat{\sigma}^2(1 - P_{x,ii})}}$$

For detecting outliers, t-values can be calculated for each error term to evaluate the statistical significance of the deviations.

$$\widehat{\sigma_i}^2 = \frac{SSE - e_i^2}{(n-k-1)(1-P_{x.ii})}$$

$$t_{e,i} = \frac{e_i}{\sqrt{\widehat{\sigma}^2(1 - P_{x,ii})}}$$

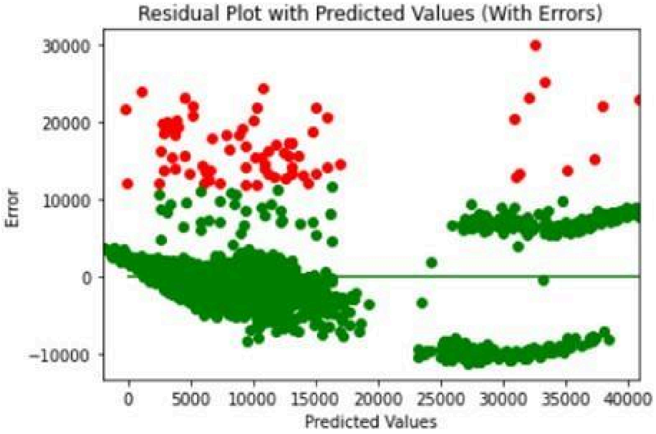
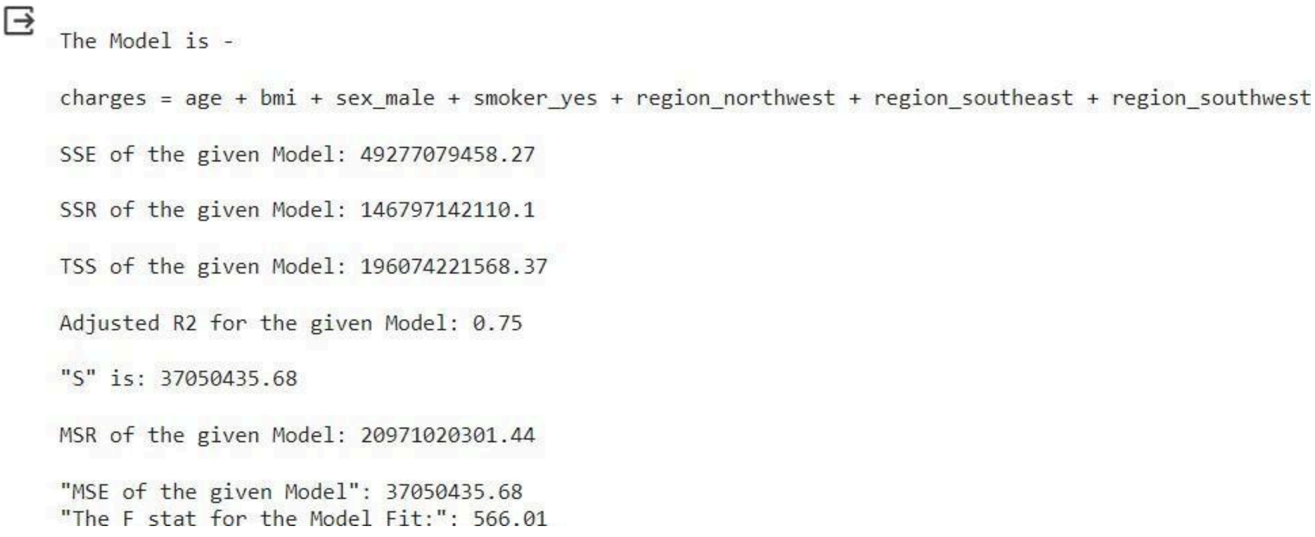


Figure 11: Plotting errors with outliers with the predicted values.

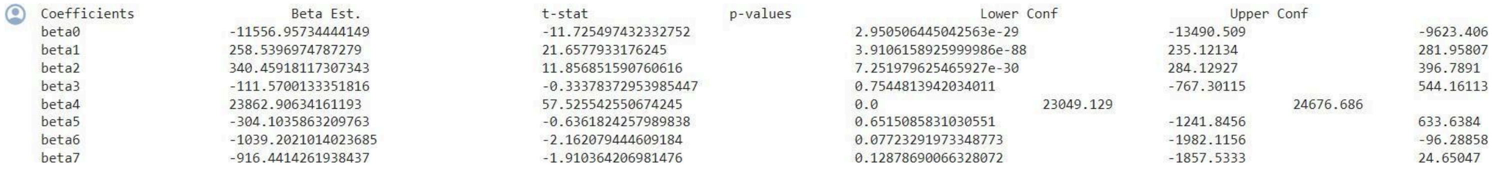
8. BACKWARD FEATURE SELECTION:

We began our Linear Regression analysis using the backward feature selection method, initially including all relevant features in the model. Upon detailed evaluation, we found that three variables—namely, "sex male," "region northwest," and "region southeast"—had coefficient estimates that were not statistically significant. At a 95% confidence level, we cannot confidently state that these coefficients differ from zero.

As a result, we decided to remove the "sex male" variable, as it had the highest p-value among the three, and proceeded to analyze the revised model. The updated results are presented below.



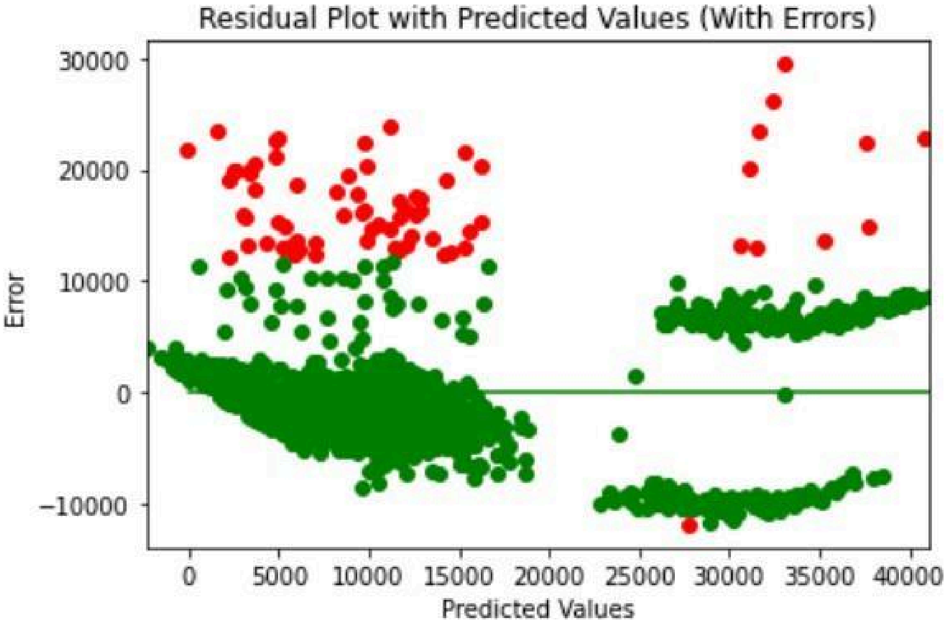
```
The Model is -

charges = age + bmi + sex_male + smoker_yes + region_northwest + region_southeast + region_southwest

SSE of the given Model: 49277079458.27

SSR of the given Model: 146797142110.1

TSS of the given Model: 196074221568.37

Adjusted R2 for the given Model: 0.75

"S" is: 37050435.68

MSR of the given Model: 20971020301.44

"MSE of the given Model": 37050435.68
"The F stat for the Model Fit:": 566.01
```



```
Coefficients                    Beta Est.                       t-stat                  p-values                Lower Conf            Upper Conf
beta0                   -11556.95734444149                      -11.725497432332752                     2.950506445042563e-29                   -13490.509                      -9623.406
beta1                   258.5396974787279                       21.6577933176245                        3.9106158925999986e-88                  235.12134                       281.95807
beta2                   340.45918117307343                      11.856851590760616                      7.251979625465927e-30                   284.12927                       396.7891
beta3                   -111.5700133351816                      -0.33378372953985447                    0.7544813942034011                      -767.30115                      544.16113
beta4                   23862.90634161193                       57.525542550674245                      0.0                     23049.129                       24676.686
beta5                   -304.1035863209763                      -0.6361824257989838                     0.6515085831030551                      -1241.8456                      633.6384
beta6                   -1039.2021014023685                     -2.162079444609184                      0.07723291973348773                     -1982.1156                      -96.28858
beta7                   -916.4414261938437                      -1.910364206981476                      0.12878690066328072                     -1857.5333                      24.65047
```

The key insight here is that a variable previously deemed significant has now lost its significance. Interestingly, all three of these variables were dummy-encoded representations of the same categorical feature, "region." As a result, we chose to drop the "region" variable entirely and evaluated the model once again.

Following this adjustment, all remaining features exhibit statistically significant coefficients, suggesting that the model is now well-aligned with the current set of predictors. While the residual errors have shifted slightly, some variance still remains unexplained. However, it's worth noting that the model's F-statistic has shown a considerable improvement, indicating enhanced overall model performance.



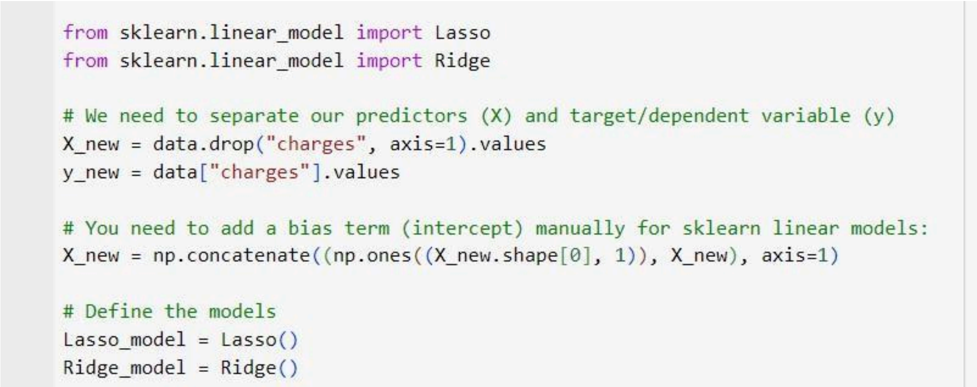
9. OTHER MODELS:

9.1 Lasso and Ridge Regression:

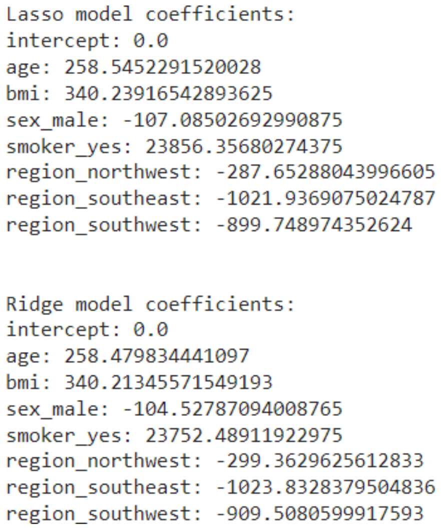
Using backward selection, we arrived at a viable model. However, we sought to further explore how the coefficients differ when applying Lasso Regression.

When comparing the coefficient estimates from Lasso and Ridge regression, we observed that they are quite similar to each other. However, they differ noticeably from the coefficients obtained through backward selection.

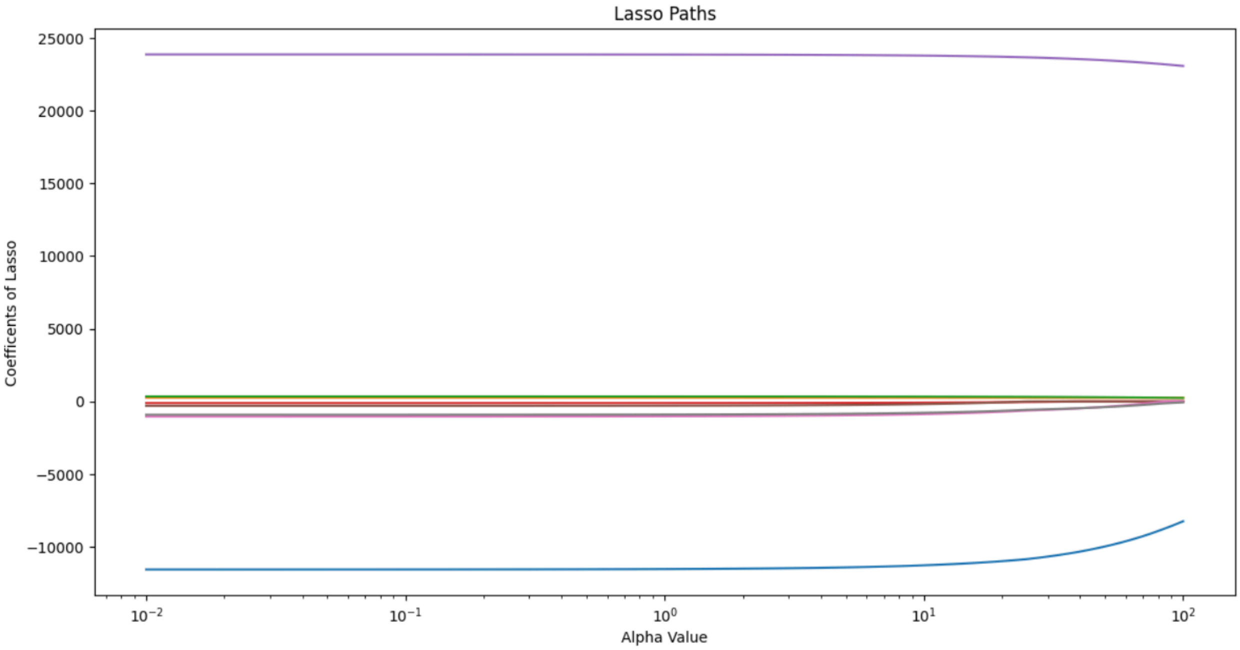
Despite these differences in coefficient values, the R-squared estimates for both Lasso and Ridge models remain consistent with the backward selection model, each achieving an R-squared value of 0.75.

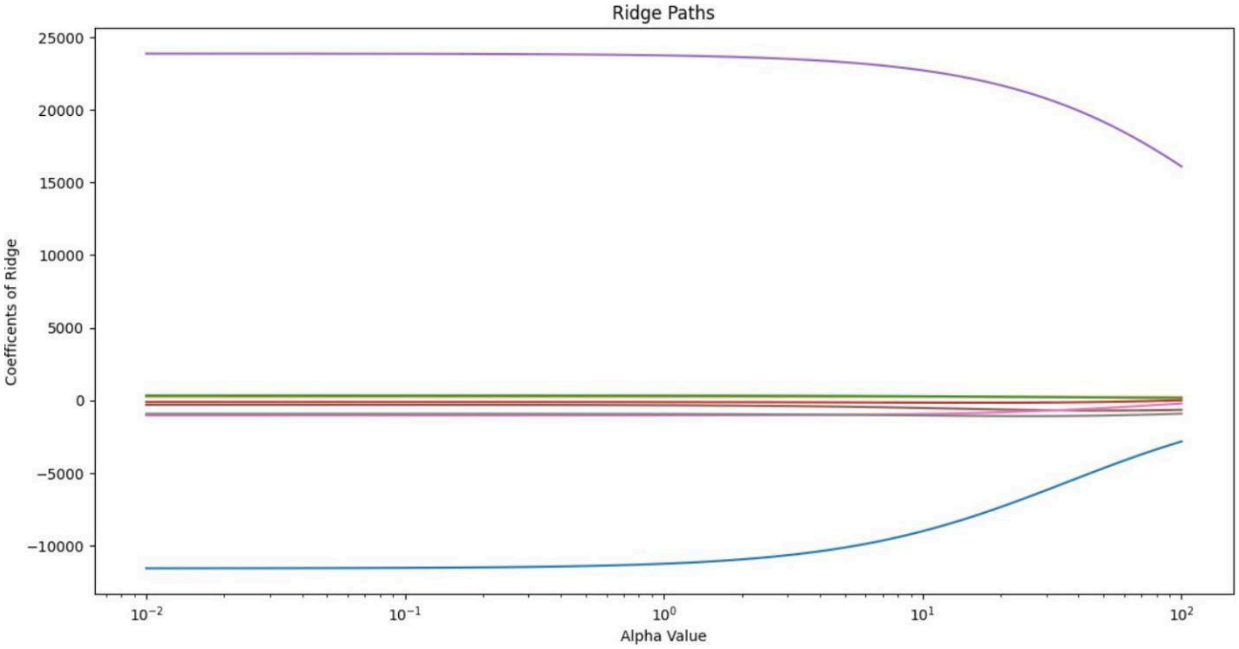


```python
from sklearn.linear_model import Lasso
from sklearn.linear_model import Ridge

# We need to separate our predictors (X) and target/dependent variable (y)
X_new = data.drop("charges", axis=1).values
y_new = data["charges"].values

# You need to add a bias term (intercept) manually for sklearn linear models:
X_new = np.concatenate((np.ones((X_new.shape[0], 1)), X_new), axis=1)

# Define the models
Lasso_model = Lasso()
Ridge_model = Ridge()
```



```
Lasso model coefficients:
intercept: 0.0
age: 258.5452291520028
bmi: 340.23916542893625
sex_male: -107.08502692990875
smoker_yes: 23856.35680274375
region_northwest: -287.65288043996605
region_southeast: -1021.9369075024787
region_southwest: -899.748974352624


Ridge model coefficients:
intercept: 0.0
age: 258.479834441097
bmi: 340.21345571549193
sex_male: -104.52787094008765
smoker_yes: 23752.48911922975
region_northwest: -299.3629625612833
region_southeast: -1023.8328379504836
region_southwest: -909.5080599917593
```


Lasso Paths
Coefficents of Lasso
25000
20000
15000
10000
5000
0
−5000
−10000
Alpha Value
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}

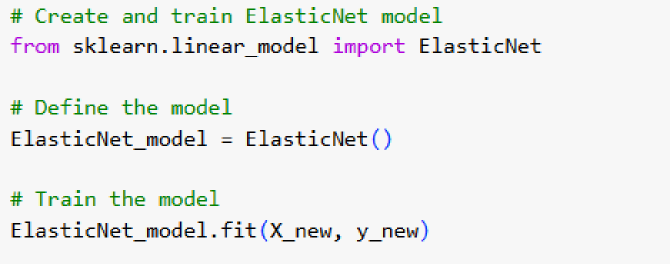

Ridge Paths
Coefficents of Ridge
25000
20000
15000
10000
5000
0
−5000
−10000
Alpha Value
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}

9.2 ElasticNet Regression:

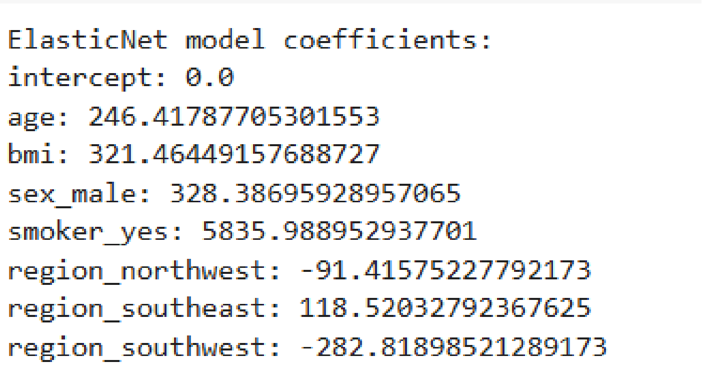
Building on our analysis, we applied Elastic Net Regression to examine how its coefficient estimates compare with those from previous models.

The Elastic Net method, which combines the penalties of both Lasso and Ridge regression, produced coefficients that represent a balance between the sparsity of Lasso and the shrinkage of Ridge. As expected, the resulting coefficients lie between those obtained from the two individual methods.

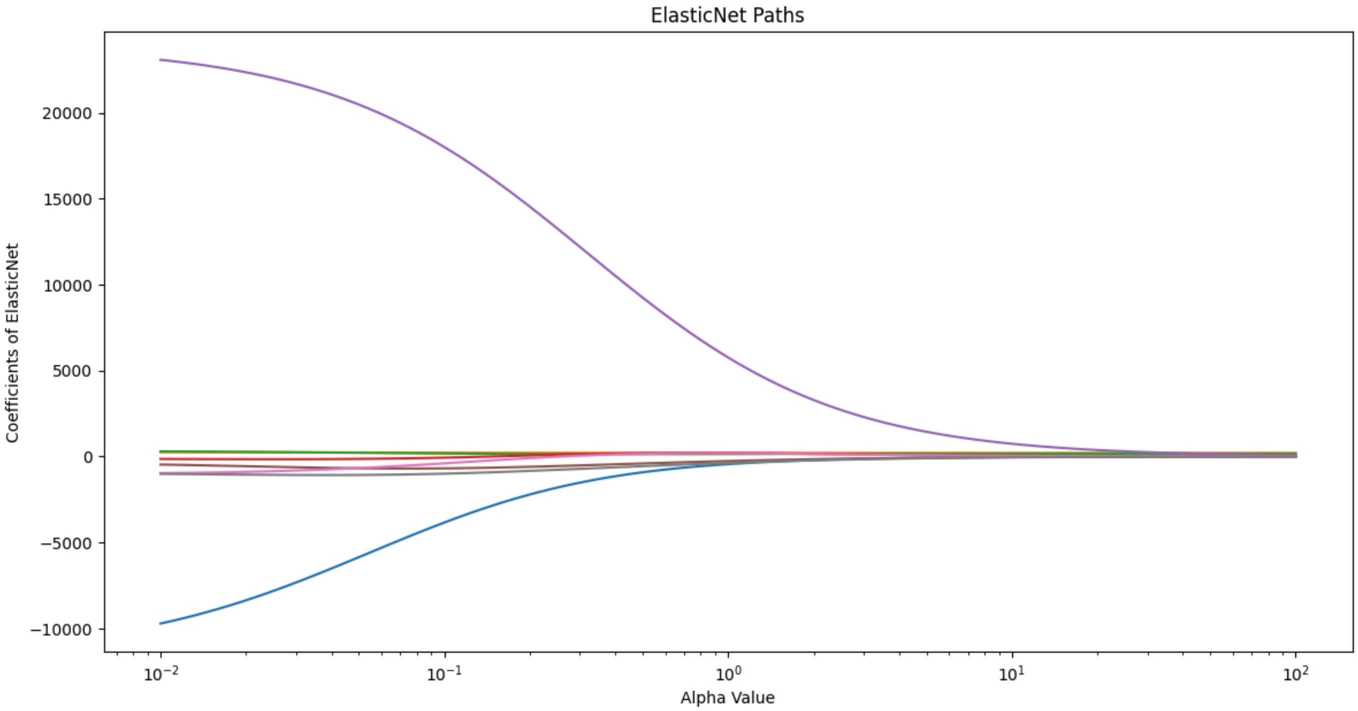
When compared to the backward selection model, the Elastic Net coefficients show notable differences, particularly in how they regularize certain features. Despite these differences, the model's performance remains consistent—Elastic Net achieved an R-squared value of 0.75, aligning with the values observed for Lasso, Ridge, and backward selection models.



```
# Create and train ElasticNet model
from sklearn.linear_model import ElasticNet

# Define the model
ElasticNet_model = ElasticNet()

# Train the model
ElasticNet_model.fit(X_new, y_new)
```



```
ElasticNet model coefficients:
intercept: 0.0
age: 246.41787705301553
bmi: 321.46449157688727
sex_male: 328.38695928957065
smoker_yes: 5835.988952937701
region_northwest: -91.41575227792173
region_southeast: 118.52032792367625
region_southwest: -282.81898521289173
```


ElasticNet Paths
Coefficients of ElasticNet
20000
15000
10000
5000
0
−5000
−10000
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
Alpha Value

10. CONCLUSION:

In conclusion, our exploration of Multiple Linear Regression (MLR) for predicting housing prices has provided meaningful insights and practical outcomes. Through careful data analysis and robust modeling techniques, we successfully built regression models capable of accurately estimating housing prices based on a range of influential features.

The process began with an in-depth examination of the dataset, during which we identified key variables and addressed challenges such as multicollinearity. Using Python and its powerful libraries, we carried out data preprocessing, exploratory analysis, and ensured that the foundational assumptions of MLR were satisfied.

Our modeling approach incorporated backward feature selection, as well as regularization techniques like Lasso and Ridge Regression, enabling us to identify the most impactful predictors and improve model reliability. The backward selection process, in particular, highlighted the relative importance of individual features, guiding us toward a more refined and interpretable model.

By comparing results across different regression methods, we gained a deeper understanding of how regularization affects both coefficient estimates and overall model performance. Despite variations in feature weights, the models consistently demonstrated strong predictive capabilities.

Ultimately, this project underscores the power and flexibility of MLR in predictive modeling, especially within the context of insurance premium estimation. The insights and methodologies developed here lay a solid foundation for future research and real-world applications in insurance, healthcare, and other data-driven domains.