# Indian Institute of Technology, Kharagpur

## Regression and Time Series Models (MA60056)

### Predicting Customer Expenses for Insurance Premium Using Multiple Linear Regression Based on Customer Characteristics

Prof. Buddhananda Banerjee

Course Project

**PGDBA 2022-2024**


Abhirup Saha, 22BM6JP02
Ankit Sonkusare, 22BM6JP07
Sparsh Kansotia, 22BM6JP50
Anant Yadav, 22BM6JP63
Apoorva Tejaswi, 20AG3FP37

# Contents

# Multiple Linear Regression

## 1 Introduction

Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. The goal of multiple linear regression is to model the linear relationship between the explanatory (independent) variables and response (dependent) variables. Here we are trying to predict the expenses of the customer based on several parameters such as Gender, Age, Body Mass Index (BMI), Smoking Behaviour, Number of children, etc.
We shall be performing the regression analysis on Python. The aim of the project is to evaluate the regression from the perspective of its statistics and from the business perspective of the same. We shall be going through the code along with the basic mathematics involved at every step.

## 2 About the Data

**The data contains the following attributes in total. Some of these attributes are continuous in nature while others are ordinal and nominal types. Following are the data attributes -**

1. **Age**: Contains the age of the customer, measured in years. This is a continuous variable.
2. **Sex**: Contains the gender of each of its customers. It can take only 2 data values - Male and Female.
3. **Body Mass Index (BMI)**: Contains the BMI of its customer. This is again a continuous variable.
4. **Children**: Contains the number of children the customer has.
5. **Smoker**: Contains the smoking behaviour, of the customer.
6. **Region**: Contains the region in which the customer lives. This can be divided into 4 categories - Southwest, Southeast, Northwest, and Northeast.
7. **Expenses**: Contains the annual expenses of the customer (in Dollars).

# 3 Regression Analysis

## 3.1 Libraries

**We shall be using the python libraries -**

1. Pandas: For loading in data for analysis.
2. Numpy: For Matrix Analysis
3. Matplotlib: For plotting the curves
4. Scipy: For Statistical tables and values
5. Seaborn: For Data Visualisation

```python
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
```
✓ 28.1s    Python

## 3.2 Data Preprocessing

The data contains several attributes which are categorical in nature. Handling categories require us to convert them to numerical variables. Converting them to numerical can be done in 2 ways, first, we can assign the categories, and numbers, such as A for 1, B for 2, etc. But this has lots of issues. Firstly we don't know which numbers to assign to which categories, secondly assigning numbers directly adds an extra constraint of these categories belonging to some order, even when they won't be. The best way to deal with this is to rather code in a different form - One Hot Encoding.

One-hot encoding technique allows us to regress upon categorical features, as the categorical features with no particular ordering tend to shift the regression line parallel. This way we can incorporate, the modification in each of the constant terms parallelly. For example - If a feature has 3 categories - (A, B, C) then it can be encoded as (1,0,0),(0,1,0),(0,0,1). This way the feature is broken down into 3 categories. Now one important aspect of this is, another term will be added to the constant term when category A is present, same for others. Hence we can drop the column first column and convert the encoding as (0,0), (1,0), (0,1). This way the constant term will accommodate, the A category itself. Hence we went ahead and dropped the first column from each of the one-hot vectors.

```
data_primary = pd.read_csv("insurance.csv")
dummy = pd.get_dummies(data_primary[["sex", "smoker", "region"]], drop_first=True)
data = data_primary.drop(["sex", "smoker", "region"], axis=1)
data = pd.concat((data, dummy), axis=1)
data
```

✓ 3.6s Python

| | age | bmi | children | expenses | sex_male | smoker_yes | region_northwest | region_southeast | region_southwest |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 19 | 27.9 | 0 | 16884.92 | 0 | 1 | 0 | 0 | 1 |
| 1 | 18 | 33.8 | 1 | 1725.55 | 1 | 0 | 0 | 1 | 0 |
| 2 | 28 | 33.0 | 3 | 4449.46 | 1 | 0 | 0 | 1 | 0 |
| 3 | 33 | 22.7 | 0 | 21984.47 | 1 | 0 | 1 | 0 | 0 |
| 4 | 32 | 28.9 | 0 | 3866.86 | 1 | 0 | 1 | 0 | 0 |

Figure 1: One-hot encoded Insurance Data

### 3.3 Exploratory Data Analysis

```
plt.figure(figsize=(6, 4))
plt.title("Variation of Expenses vs Age")
plt.scatter(data.iloc[:, 0], data.iloc[:, 3])
plt.ylabel("Expenses")
plt.xlabel("Age")
plt.show()
```

✓ 2.3s Python

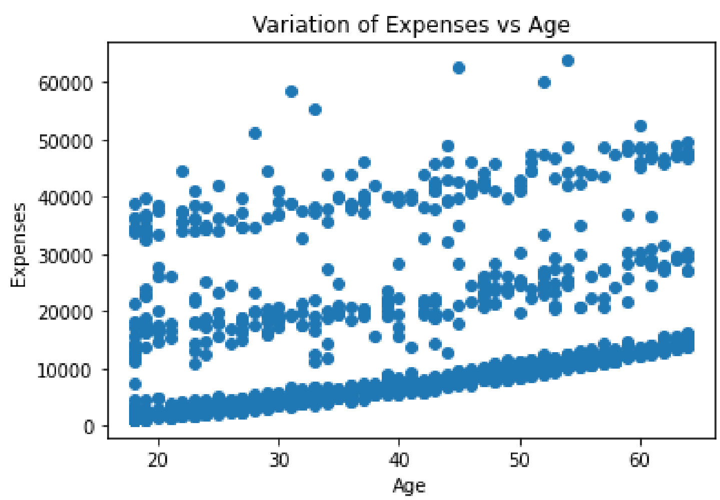



Figure 2: Expenses tend to vary linearly with increasing age, however, there seem to be some parallel lines suggesting the effect of other dependent variables, which are not included here.

```python
plt.figure(figsize=(6, 4))
plt.title("Variation of Expenses vs Number of BMI")
plt.scatter(data.iloc[:, 1], data.iloc[:, 3])
plt.ylabel("Expenses")
plt.xlabel("BMI")
plt.show()
```

✓ 0.3s Python

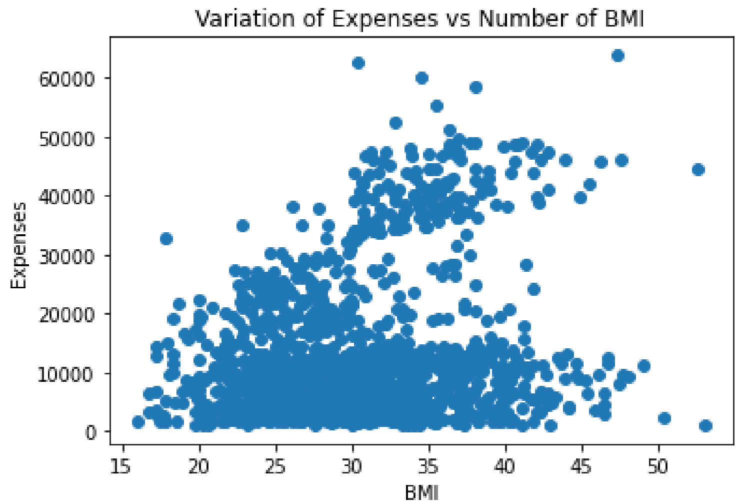



Figure 3: Expenses tend to vary with BMI however there seem to be 2 different trends, one is steep the other one isn't. This suggests the presence of some compounding effect with another variable.

```python
plt.figure(figsize=(6, 4))
plt.title("Variation of Expenses vs Number of Children")
plt.boxplot([data_primary[data_primary["children"]==0]["expenses"].values,
             data_primary[data_primary["children"]==1]["expenses"].values,
             data_primary[data_primary["children"]==2]["expenses"].values,
             data_primary[data_primary["children"]==3]["expenses"].values,
             data_primary[data_primary["children"]==4]["expenses"].values,
             data_primary[data_primary["children"]==5]["expenses"].values],
            labels=range(0,6))
plt.ylabel("Expenses")
plt.xlabel("Number of Children")
plt.show()
```

✓ 0.4s Python

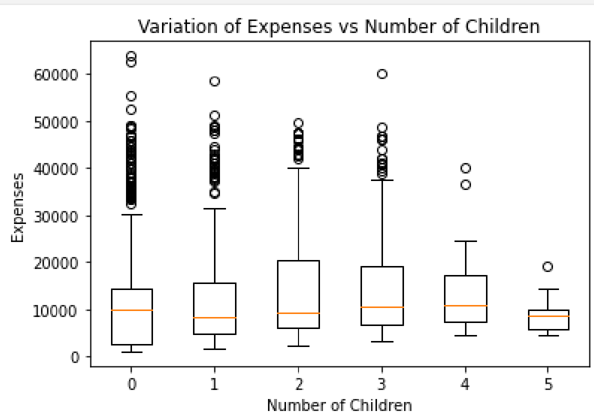



Figure 4: Children were distinct values and had 6 different values. For each of these, we wanted to see the distribution of the expenses, which showed a skewed distribution. This can be inferred from the box plots indicating variation in expenses with different numbers of children.

```python
plt.figure(figsize=(6, 4))
plt.title("Variation of Expenses vs Regions")
plt.boxplot([data_primary[data_primary["region"]=="southwest"]["expenses"].values, data_primary[data_primary["region"]=="southeast"
             data_primary[data_primary["region"]=="northwest"]["expenses"].values, data_primary[data_primary["region"]=="northeast"
            labels=["Southeast", "Southwest", "Northwest", "Northeast"])
plt.ylabel("Expenses")
plt.xlabel("Regions")
plt.show()
```

✓ 0.3s Python

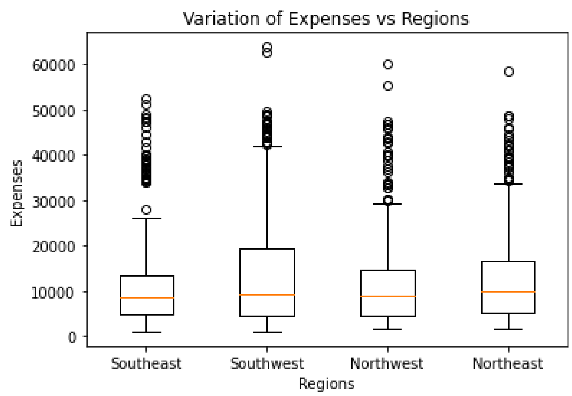



Figure 5: Shows the variation in the expenses region-wise for the different customers. Again we have tried to see the distribution of expenses in different regions.

```python
plt.figure(figsize=(6, 4))
plt.title("Variation of Expenses vs Smoking Behvaiour")
plt.boxplot([data[data["smoker_yes"]==1]["expenses"].values, data[data["smoker_yes"]==0]["expenses"].values], labels=["Smoker", "No
plt.ylabel("Expenses")
plt.xlabel("Smoking Behaviour")
plt.show()
```

✓ 0.1s Python

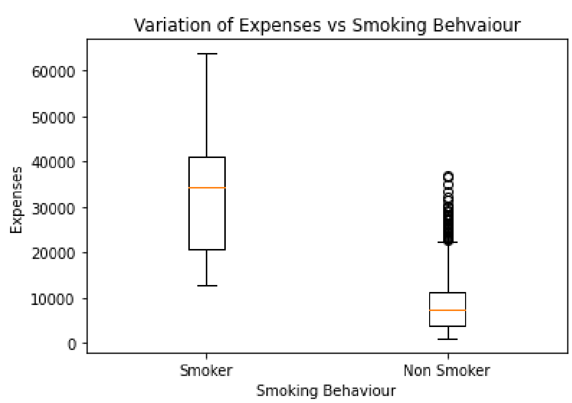



Figure 6: Shows the variation in the expenses for the smoking behavior of the customers. Customers with smoking behavior tend to have higher expenses compared to the ones who don't.

```python
plt.figure(figsize=(6, 4))
plt.title("Variation of Expenses vs Age")
plt.boxplot([data[data["sex_male"]==1]["expenses"].values, data[data["sex_male"]==0]["expenses"].values], labels=["Male", "Female"]
plt.ylabel("Expenses")
plt.xlabel("Sex")
plt.show()
```

✓ 0.1s Python

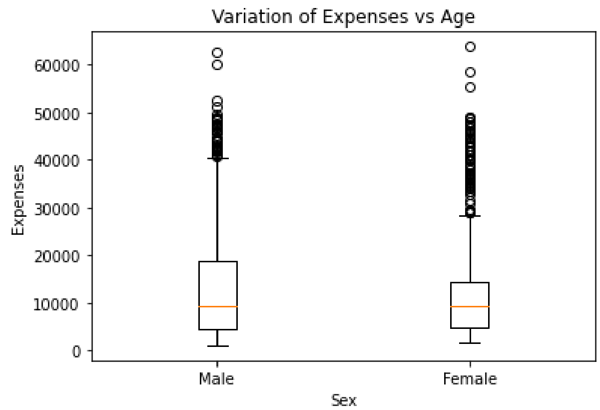



Figure 7: Shows the variation in the expenses with the gender of the customers. There doesn't seem to be much difference apart from the difference in variance between the two.

```python
plt.figure(figsize=(6, 6))
plt.title("Correlation Map")
sns.heatmap(data[["age", "bmi", "children", "expenses"]].corr().values, annot=True,
            xticklabels=["age", "bmi", "children", "expenses"], yticklabels=["age", "bmi", "children", "expenses"])
```

✓ 0.4s Python

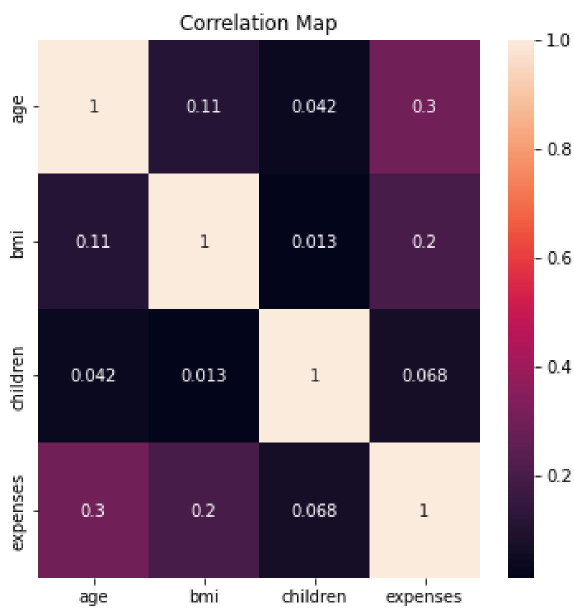



Figure 8: This is the correlation heat map, depicting the correlation between different predictor variables. Here we have considered the correlation among only the continuous variables.

# 4 The Regression Analysis

In the regression analysis, we have chosen the backward selection procedure. We will proceed with all the predictor variables in the first stage and gradually reduce them till we have achieved significant results. For our regression analysis, we shall be assuming a 95% confidence level for our calculations. In our process, we will be estimating the regression coefficients, testing the fit of the models, standard errors and p-values corresponding to the estimated coefficients, mean confidence interval, prediction intervals, and error analysis.

```python
columns_x = ['age', 'bmi', 'children', 'sex_male', 'smoker_yes',
             'region_northwest', 'region_southeast', 'region_southwest']
target = data.columns[3]

X = data[columns_x].to_numpy()
X = np.concatenate((np.ones((X.shape[0], 1)), X), axis=1)
y = data[target].to_numpy()

alpha = 0.05
x0 = np.array(np.concatenate((np.ones(1), 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])
```
Python

## 4.1 Estimating the Coefficients

The Coefficients are estimated along with the errors using the following formulae. We first calculate the C matrix as

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

We then calculate the Hat matrix as

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

The coefficients can be estimated as

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

The predicted value of y is given by

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

```python
# Computing the C Matrix -

C = np.linalg.inv(np.dot(np.transpose(X), X))

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)
```
Python

## 4.2 Estimating the Model Fit

Error is defined as

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

We are predicting using the methods of least squares. Thus we define the Sum of Squares of Regression (SSR), the Sum of squares of Error (SSE), and the Total Sum of Squares.

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

$$SSR = y^T(P_x - \frac{1}{n}11^T)y$$

$$TSS = y^T(I_n - \frac{1}{n}11^T)y$$

The Mean Square of Regression (MSR), Mean Square of Error (MSE), and Mean Sum of Squares are given by for $k$ predictors as

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

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

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

For testing the model of Fit, we shall be computing R-sq., Adjusted R-sq., and F-statistic is given by the following formulae

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

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

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

```python
ones = np.ones((Px.shape[0], 1), dtype=np.float32)
In = np.eye(Px.shape[0])

error = y - y_pred

ones_m = (1/X.shape[0])*np.dot(ones, np.transpose(ones))

SSE = np.dot(np.transpose(y), np.dot(In-Px, y))
TSS = np.dot(np.transpose(y), np.dot(In-ones_m, y))
SSR = np.dot(np.transpose(y), np.dot(Px-ones_m, y))

MSR = SSR/(X.shape[1]-1)
MSE = SSE/(X.shape[0]-X.shape[1])
MSS = TSS/(X.shape[0]-1)

R2 = 1-SSE/TSS
R2_adj = 1-(MSE/MSS)
sigma_2 = MSE
F_stat = MSR/MSE
```

✓ 0.1s Python

## 4.3 Test of Hypothesis for the Estimates of Coefficients

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

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

The confidence interval for the estimates of coefficients is given by

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

```python
t_values = np.zeros(beta.shape[0], dtype=np.float32)

for i in range(0, beta.shape[0]):
    t_values[i] = beta[i]/np.sqrt(sigma_2*C[i, i])

beta_conf = np.zeros((beta.shape[0], 2), dtype=np.float32)

for j in range(0, beta.shape[0]):
    beta_conf[j, 0] = beta[j] - t_right*np.sqrt(sigma_2*C[j, j])
    beta_conf[j, 1] = beta[j] - t_left*np.sqrt(sigma_2*C[j, j])
```

✓ 0.0s Python

## 4.4 ANOVA Table

The ANOVA table of our model is as follows

```python
print("============================== "+"Summary"+" =============================="+"\n")
print("The Model is - \n")
text1 = str(target[0])+" ~ "
for i in columns_x:
    text1=text1+i+" + "
print(text1[:-3]+"\n")
print("SSE of the given Model", SSE.round(2))
print("SSR of the given Model", SSR.round(2))
print("TSS of the given Model", TSS.round(2), "\n")
print("R2 for the given Model", R2.round(2))
print("Adjusted R2 for the given Model", R2.round(2), "\n")
print("s2 =", sigma_2.round(2), "\n")
print("MSR of the given Model", MSR.round(2))
print("MSE of the given Model", MSE.round(2))
print("The F stat for the Model Fit", F_stat.round(2), "\n")
print("=============================="+"======="+"==============================")
print("Beta Estimates of the given Model -\n")
beta_heading = list()
for i in range(0, beta.shape[0]):
    beta_heading.append(str("beta")+"_"+str(i))
dict1 = {
    "Coefficients":beta_heading,
    "Beta Est.":beta,
    "t-stat":t_values,
    "p-values":2*t.pdf(t_values, df=X.shape[0]-X.shape[1]).round(3),
    "Lower Conf":beta_conf.T[0],
    "Upper Conf":beta_conf.T[1]
```

✓ 0.2s Python

```
============================== Summary ==============================

The Model is -

e ~ age + bmi + children + sex_male + smoker_yes + region_northwest + region_southeast + region_southwest

SSE of the given Model 48836507127.82
SSR of the given Model 147237714922.12
TSS of the given Model 196074222049.95

R2 for the given Model 0.75
Adjusted R2 for the given Model 0.75

s2 = 36746807.47

MSR of the given Model 18404714365.27
MSE of the given Model 36746807.47
The F stat for the Model Fit 500.85

=====================================================================
Beta Estimates of the given Model -

  Coefficients     Beta Est.     t-stat  p-values    Lower Conf    Upper Conf
0       beta_0 -11941.562461 -12.088909     0.000 -13879.402344 -10003.722656
1       beta_1    256.839171  21.585659     0.000    233.497086    280.181274
2       beta_2    339.289863  11.863969     0.000    283.187042    395.392670
3       beta_3    475.688916   3.452034     0.002    205.360443    746.017395
4       beta_4   -131.352014  -0.394527     0.738   -784.487549    521.783508
5       beta_5  23847.476695  57.722618     0.000  23037.000000  24657.953125
6       beta_6   -352.790096  -0.740749     0.606  -1287.095459    581.515198
7       beta_7  -1035.595701  -2.163437     0.078  -1974.647827    -96.543564
8       beta_8   -959.305829  -2.007286     0.106  -1896.849976    -21.761742
=====================================================================
```

## 4.5 Mean Confidence Interval and Prediction Intervals

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

$$\widehat{y_0} - t_{\alpha/2,n-k-1}\sqrt{\widehat{\sigma}^2 x_0^T (X^T X)^{-1} x_0} \leq E(y|x) \leq \widehat{y_0} + t_{\alpha/2,n-k-1}\sqrt{\widehat{\sigma}^2 x_0^T (X^T X)^{-1} x_0}$$

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

$$\widehat{y_0} - t_{\alpha/2,n-k-1}\sqrt{\widehat{\sigma}^2 + \widehat{\sigma}^2 x_0^T (X^T X)^{-1} x_0} \leq E(y|x) \leq \widehat{y_0} + t_{\alpha/2,n-k-1}\sqrt{\widehat{\sigma}^2 + \widehat{\sigma}^2 x_0^T (X^T X)^{-1} x_0}$$

```python
mean_conf = np.zeros((1, 2), dtype=np.float32)

mean_x0 = np.dot(x0, beta)
mean_se = np.sqrt(sigma_2*np.dot(x0, np.dot(C, x0)))

mean_conf[0, 0] = mean_x0 - t_right*mean_se
mean_conf[0, 1] = mean_x0 - t_left*mean_se

pred_conf = np.zeros((1, 2), dtype=np.float32)

pred = np.dot(x0, beta)
pred_se = np.sqrt(sigma_2*(1+np.dot(x0, np.dot(C, x0))))

pred_conf[0, 0] = pred - t_right*pred_se
pred_conf[0, 1] = pred - t_left*pred_se
```
✓ 0.0s Python

For the given prediction vector, we got the mean confidence interval as [1538934.1, 5599928.] and the Prediction Interval as [1538899.2, 5599962.5].

## 4.6 Error Analysis

```python
plt.scatter(y_pred, error)
plt.plot([0, y_pred.max()+2], [0, 0], color="g")
plt.autoscale(axis="x", tight=True)
plt.xlabel("Predicted Values")
plt.ylabel("Error")
plt.title("Residual Plot with Predicted Values")
plt.show()
```
✓ 0.3s Python

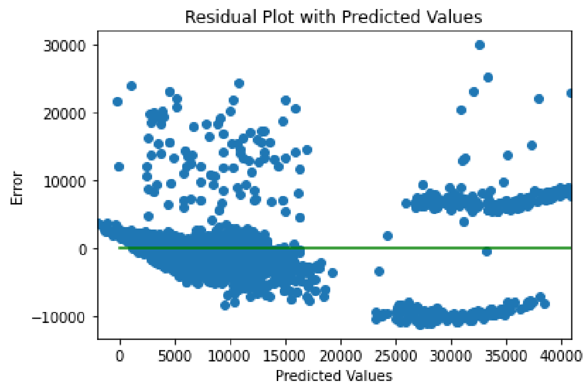



Figure 9: The plot of estimated errors with the predicted values. We observe that these are not normally distributed. Also, the error plot suggests that the given set of data points may be comprised of several different populations due to different cluster formulations. These may be explained further with some additional categorical features to segregate.

However, these errors may be converted to standardized forms and then computed as

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

For outlier detection, we compute the corresponding t-values to each of the error points as

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

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

```python
stand_res = np.zeros(X.shape[0], dtype=np.float32)

for i in range(0, X.shape[0]):
    stand_res[i] = error[i]/np.sqrt(sigma_2*(1-Px[i, i]))

te_values= np.zeros(X.shape[0], dtype=np.float32)

for i in range(0, X.shape[0]):
    sigma_2i = (SSE-np.square(error[i])/(1-Px[i, i]))*(1/(X.shape[0]-X.shape[1]))
    te_values[i] = error[i]/np.sqrt(sigma_2i*(1-Px[i, i]))
```
✓ 0.0s Python

```python
stand_res = np.zeros(X.shape[0], dtype=np.float32)

for i in range(0, X.shape[0]):
    stand_res[i] = error[i]/np.sqrt(sigma_2*(1-Px[i, i]))

te_values= np.zeros(X.shape[0], dtype=np.float32)

for i in range(0, X.shape[0]):
    sigma_2i = (SSE-np.square(error[i])/(1-Px[i, i]))*(1/(X.shape[0]-X.shape[1]))
    te_values[i] = error[i]/np.sqrt(sigma_2i*(1-Px[i, i]))
```
✓ 0.0s Python

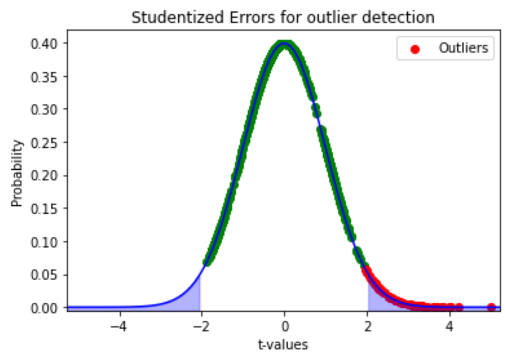



Figure 10: Plotting the t-values for the errors to highlight which ones are outliers and which ones are not.

```python
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",
            label="Outliers")
plt.plot(x_p, y_p, color="b")
#plt.plot(x_p, y_p2, color="g")
plt.autoscale(axis="x", tight=True)
plt.ylim(bottom=-0.005)
plt.xlabel("t-values")
plt.ylabel("Probability")
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, color="b", alpha=0.3)
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, color="b", alpha=0.3)
plt.legend()
plt.show()
```
✓ 0.3s Python

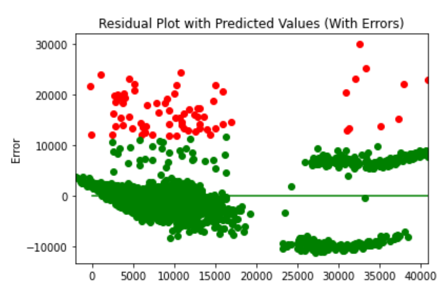



Figure 11: Plotting errors with outliers with the predicted values.

# 5 Backward Feature Selection

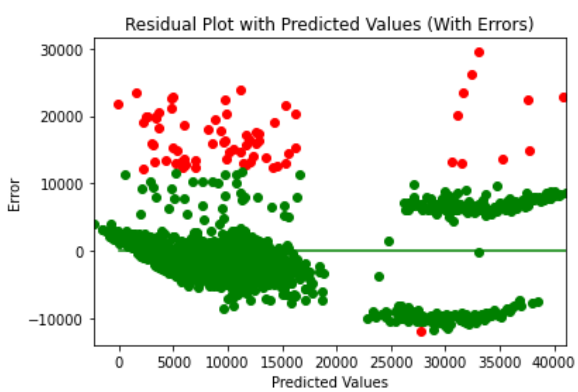
We started our Linear Regression Analysis with the backward feature selection method. Here we initially assumed that all the features were relevant to our model. Now after due analysis, we have concluded that three variables - ***sex_male, region_northwest*** and ***region_southeast***, have coefficient estimates which aren't really significant, and we can't really say with 95% confidence that they aren't equal to 0. Hence in such a scenario, we dropped the ***sex_male*** feature and checked the results. This feature was chosen as it has the maximum p-value. With this, the new results are as follows.

```
============================== Summary ==============================

The Model is -

e ~ age + bmi + children + smoker_yes + region_northwest + region_southeast + region_southwest

SSE of the given Model 48842226837.98
SSR of the given Model 147231995211.97
TSS of the given Model 196074222049.95

R2 for the given Model 0.75
Adjusted R2 for the given Model 0.75

s2 = 36723478.83

MSR of the given Model 21033142173.14
MSE of the given Model 36723478.83
The F stat for the Model Fit 572.74

======================================================================
Beta Estimates of the given Model -

  Coefficients     Beta Est.     t-stat  p-values    Lower Conf    Upper Conf
0       beta_0 -11993.309337 -12.253651     0.000 -13913.378906 -10073.239258
1       beta_1    256.956451  21.609119     0.000    233.629074    280.283844
2       beta_2    338.760947  11.862277     0.000    282.737640    394.784271
3       beta_3    474.754266   3.446855     0.002    204.551743    744.956787
4       beta_4  23835.240835  57.874626     0.000  23027.308594  24643.173828
5       beta_5   -352.008360  -0.739349     0.606  -1286.008301    581.991638
6       beta_6  -1034.933540  -2.162754     0.078  -1973.681152    -96.185944
7       beta_7   -958.630761  -2.006523     0.106  -1895.870605    -21.390968
======================================================================
```

Here the key insight was the fact that a variable that wasn't previously insignificant in the first iteration is now insignificant. Coincidentally, all these 3 variables were dummy encoding for the same variable ***region***. Hence dropping that variable completely, we get the following.

```
============================== Summary ==============================

The Model is -

e ~ age + bmi + children + smoker_yes

SSE of the given Model 49075447085.79
SSR of the given Model 146998774964.15
TSS of the given Model 196074222049.95

R2 for the given Model 0.75
Adjusted R2 for the given Model 0.75

s2 = 36815789.26

MSR of the given Model 36749693741.04
MSE of the given Model 36815789.26
The F stat for the Model Fit 998.2

=====================================================================
Beta Estimates of the given Model -

  Coefficients     Beta Est.     t-stat  p-values     Lower Conf     Upper Conf
0       beta_0 -12105.482135 -12.851459     0.000 -13953.355469 -10257.608398
1       beta_1    257.833746  21.673756     0.000    234.496582    281.170929
2       beta_2    321.938686  11.759828     0.000    268.233673    375.643707
3       beta_3    473.692610   3.437853     0.002    203.388992    743.996216
4       beta_4  23810.317817  57.903381     0.000  23003.632812  24617.001953
=====================================================================
```

Now, none of the variables has insignificant coefficients. This is a good indicator that we have found an appropriate model with respect to our current feature set. However, the errors have changed, but still, we can't say that they are changed much. There is still unexplained behaviour present in the errors. There is also a significant increase in the F-statistic for the model.



# 6 Other Models

## 6.1 Lasso and Ridge Regression

By backward selection, we have reached a solution. However, we want to explore the difference in coefficients with Lasso Regression.

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

X_new = data.drop("expenses", axis=1).values
X_new = np.concatenate((np.ones((X_new.shape[0], 1)), X_new), axis=1)
y_new = data["expenses"].values

Lasso_model = Lasso()
Ridge_model = Ridge()

Lasso_model.fit(X_new, y_new)
Ridge_model.fit(X_new, y_new)
```
✓ 0.1s Python

| | Lasso Estimates | Ridge Estimates |
|---|---|---|
| 0 | 0.000000 | 0.000000 |
| 1 | 256.847634 | 256.779141 |
| 2 | 339.072151 | 339.044548 |
| 3 | 474.864135 | 475.752890 |
| 4 | -126.832858 | -124.257053 |
| 5 | 23840.952846 | 23737.136779 |
| 6 | -336.250487 | -347.636845 |
| 7 | -1018.334842 | -1019.974880 |
| 8 | -942.535919 | -951.978660 |

The Coefficients from the ridge and lasso estimates are similar to one another. However, they differ greatly from the model coefficients obtained from the Backward selection.

For the R-square estimates, both the lasso and ridge are the same as the backward selection models, 0.75

# 7 Conclusion

Out of the three different types of techniques applied, the backward selection seems to have a dominating effect over lasso and ridge for this regression technique, even as the R-square estimates are pretty much the same for all of them because of the model simplicity.

*Thank You*