Multiple Regression Model

Applied Statistical Methods Assignment

BITS Pilani
Dubai Campus

Project By:

Anushka Patil
(2017A7PS0233U)

1. INTRODUCTION

Multiple linear regression attempts to model the relationship between two or more explanatory(independent) variables and a response (dependent) variable by fitting a linear equation to observed data.

- The dependent features are called the dependent variables, outputs, or responses.
- The independent features are called the independent variables, inputs, or predictors.

It is used to answer whether some phenomeneon is influenced by several other variables. It is also used for forecasting a response. Linear regression calculated the estimated weights or predicted weights of the regressors, where regressors are the independent variables that are used to predict the values.

Two things that we need to consider when we choose the coefficients are

- The independent variable must have a strong correlation with the dependent variable.
- The independent variable should not have a good correlation with any other independent variable.

a. SIMPLE LINEAR REGRESSION Vs. MULTIPLE LINEAR REGRESSION

Simple or univariate linear regression is the simplest case of linear regression with a single independent variable.(Here, p=1 in the equation given in section 1.2)
Whereas, Multiple or multivariate linear regression is a case of linear regression with two or more independent variables.

b. BASIC REGRESSION MODEL

Multiple regression estimates the β's in the equation

$$\mathbf{Y=\beta_0 + \beta_1 x_{1j} + \beta_2 x_{x2j} + \beta_3 x_{3j} + ... + \beta_p x_{pj} + \varepsilon_j}$$

where,

- the x's are the independent variables
- Y is the dependent variable
- j represents the observation (row) number
- β's are the unknown regression coefficients ((population parameter)
- εj is the error (residual) of observation j

The j^{th} regression coefficient β represents the expected change in y per unit change in j^{th} independent variable X_j. Assuming $E(\varepsilon) = 0$,

$$\beta_j = (\partial E(y))/\partial X_j$$

A model is said to be linear when it is linear in parameters. In such a case $(\partial E(y)/\partial X_j$ should not depend on any β's.

c. FORMULA

$$\hat{y}_j = b_0 + b_1 x_{1j} + b_2 x_{2j} + \ldots + b_p x_{pj}$$

where,

- b is an estimate of the unknown regression coefficient β
- ŷ is the estimated value of the dependent variable Y

Although the regression problem may be solved by a number of techniques, the most-used method is least squares. In least squares regression analysis, the b's are selected so as to minimize the sum of the squared residuals. This set of b's is not necessarily the set you want, since they may be distorted by outliers--points that are not representative of the data. Robust regression, an alternative to least squares, seeks to reduce the influence of outliers.

d. SAMPLE RESIDUAL

A large part of a regression analysis consists of analyzing the sample residuals, ej, defined as

$$e_j = y_j - \hat{y}_j$$

It is a measure of the variability in the dependent variable not explained by the regression model.

e. COEFFICIENT OF DETERMINATION (R^2 or R Squared)

It helps to understand which amount of variation in y can be explained by the dependence on x using the particular regression model. Closer the R^2 value to 1, the better the fit. It means, the model can better predict the output(y) for the input(x).

$$SSE = \Sigma (y - \hat{y})^2$$

$$SSR = \Sigma (\hat{y} - Ý)^2$$

$$SST = \Sigma (y - Ý)^2$$

$$R^2 = SSR / SST$$

where,

- SSE is the sum of squares due to error
- SSR is the sum of squares due to regression
- SST is the total sum of squares
- ŷ is the predicted value of the dependent variable,
- Ý is the dependent variable mean
- y is the dependent variable raw score.

f. ADJUSTED R SQUARED

R-squared value can only increase as predictors are added to the regression model. This increase is artificial when predictors are not actually improving the model's fit.
To remedy this, a related statistic, Adjusted R-squared, incorporates the model's degrees of freedom. Adjusted R-squared will decrease as predictors are added if the increase in model fit does not make up for the loss of degrees of freedom. Likewise, it will increase as predictors are added if the increase in model fit is worthwhile.

g. CORRELATION COEFFICIENT

Pearson's product moment correlation coefficient (r) is given as a measure of linear association between the two variables:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(Y_i - \bar{Y})^2}}$$

h. ROOT MEAN SQUARE ERROR

Root Mean Square Error is the standard deviation of the residuals(prediction errors). Residuals are a measure of how far from the regression line data points are. The lower the RMSE value, the model would be fit well.

$$RMSErrors = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}}$$

i. DEGREES OF FREEDOM

- Degrees of Freedom for Regression Model: DFR= p – 1
- Degrees of Freedom for Error: DFE= n - p
- Degrees of Freedom Total: DFT= n - 1

j. F-TEST

It helps in understanding whether the equation as a whole is statistically significant in explaining Y or not. Here are the five steps of overall F-test for regression

- State the null and alternative hypothesis

  H_0: $\beta_1 = \beta_2 = ... = \beta_p = 0$
  H_1: at least one $\beta_i \neq 0$, $i = 1, ..., k$

- Compute the test-statistic assuming that the null hypothesis is true

  F = MSR/MSE (explained variance/unexplained variance)
  where,

MSR = SSR/DFR (refer to 1.5,1.9)
MSE = SSE/DFE (refer to 1.5,1.9)

- Find a (1- α)100% confidence interval I, for (DFR, DFE) degrees of freedom using an F-table or statistical software.
- Accept the null hypothesis if F belongs to the confidence interval I; else reject H_0.
- Determine p-value

k. T-TEST

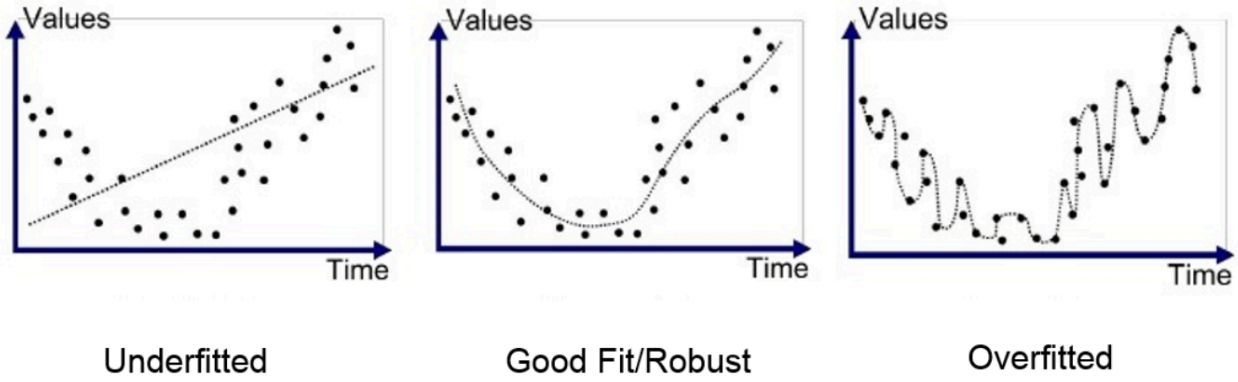
The t-test behaves differently from the F-test. It looks at the relationship between the target variable, and every predictor variable, independently.
Here, b_i is the population parameter of the i^{th} variable.

Ho: $b_i = 0$
H_1: b_i != 0 (not equal to 0)

l. OVERFITTING AND UNDERFITTING OF THE MODEL

Overfitting happens when the model learns the existing data too well. Complex models, which have many features, are often prone to overfitting. If the model is overfitted, the R^2 value would be low when using with new data.

When the model can't accurately capture the dependencies among data, the underfitting will happen. It often yields a low R^2 with known data and bad generalization capabilities when applied with new data.



m. MULTICOLLINEARITY (PROBLEM WITH MULTIPLE REGRESSION)

Collinearity, or multicollinearity, is the existence of near-linear relationships among the set of independent variables. When two variables are highly corelated they are basically measuring the same thing. The presence of multicollinearity causes all kinds of problems with regression analysis, so you could say that we assume the data do not exhibit it.

Signs of multicollinearity include:

- None of the t-ratios of the coefficients are statistically significant, but the F-test for the equation as a whole is significant.
- Adding an additional independent variable to the equation radically changes either the size or the sign (plus or minus) of the coefficients associated with the other independent variables.

2. APPLICATIONS OF MULTIPLE REGRESSION

a. BUSINESS FIELD

Here are some of the ways in which linear regression is used in business:

- **For Predictive Analysis**

    1. **Ex:** Imagine you want to estimate the demand of your customer. Or you want to predict the sales of a particular item in the future.
    2. **Ex:** Insurance companies rely heavily on regression to understand the number of claims at any given time.

Sales of Products

Predict economic growth

Oil price prediction

House price prediction

- **Operational Efficiency** (For optimizing the business)

    1. **Ex:** A baker can estimate the right temperature for its oven, to increase the shelf life of the cookies.
    2. **Ex:** In BPOs/KPOs, we can analyse the relationship between the wait times of caller and the number of complaints.
    3. **Ex:** An organisation can use linear regression to figure out how much they would pay to a new joinee based on the years of experience.

Salary Estimation

- **Supporting Decisions and Avoiding Errors** (Regression analysis can take the guesswork out of the picture and can help the management to support their business decisions.)

- **Provides New Insights** (With ever-growing data, it would help us to get new insights. Analysing the website sales data can tell us what day of the week, what hours of day, is most profitable.)

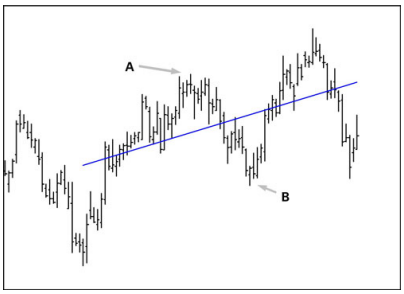
b. ASTRONOMY

Five methods for obtaining multiple linear regression fits to bivariate data with unknown or insignificant measurement errors are: ordinary least-squares (OLS) regression of Y on X OLS regression of X on Y, the bisector of the two OLS lines, orthogonal regression, and 'reduced major-axis' regression. These methods have been used by various researchers in observational astronomy, most importantly in cosmic distance scale applications. Formulas for calculating the slope and intercept coefficients and their uncertainties are given for all the methods, including a new general form of the OLS variance estimates.

c. FINANCIAL MARKETS

- Linear regression is a useful measure for technical and quantitative analysis in financial markets.
- Plotting stock prices along a normal distribution—bell curve—can allow traders to see when a stock is overbought or oversold.
- Using linear regression, a trader can identify key price points—entry price, stop-loss price, and exit prices.
- A stock's price and time period determine the system parameters for linear regression, making the method universally applicable.

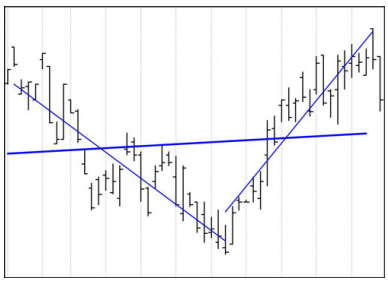
d. IDENTIFYING MARKET TRENDS



On a trading chart, one can draw a line (called the *linear regression line*) that goes through the center of the price series, which one can analyze to identify trends in price. Although we can't technically draw a straight line through the center of each trading chart price bar, the linear regression line minimizes the distance from itself to each price close along the line and thus provides a way to evaluate trends.

- **Pros:** A linear regression is the true, pure trendline. If you accept the core concept of technical analysis, that a trend will continue in the same direction, at least for a while, then you can extend the true trendline and obtain a forecast.

- **Cons:** The bad news is that the linear regression line can slope this way or that way or no way (horizontal), depending on where you start and stop drawing. If you take a V-shaped price series like the one in this figure (to the right) and draw a single linear regression line, you get nothing. So careful analysis is needed.



e. METEOROLOGICAL DATA ANALYSIS AND QUALITY ASSESSMENT IN A TERRAIN

Multivariate linear regression analysis of meteorological has been tested as a tool for both data quality assessment and as a method for objectively analyzing data in complex terrain. A least-squares fit to the pressure, temperature and dew point data is accomplished by assuming a linear variance of the surface data (e.g. temperature) in x, y, and z space. The linear regression analysis provides a tool for

- assessing the quality of data and
- objectively analyzing surface meteorological data (e.g. as input into atmospheric models)

Objective schemes commonly used in meteorological analysis use a weighting of values of nearby data to determine an estimate at a given location. Many of these schemes are used for analysis of meteorological variables on quasi-horizontal surfaces in which weights are dependent on the horizontal distance an observation is from an analysis point.

3. DEFINING THE PROBLEM AND DATA SET

a. PROBLEM AND SOURCE OF DATA

This report would focus on using multiple linear regression techniques for **Predicting individual medical costs billed by health insurance.** This would be a helpful prediction analysis as healthcare costs are rising as we move ahead in time and no matter what, these costs are unavoidable.

Hypothesis that we would be testing is whether the medical bills are dependent on factors like age, sex, body mass index, no of children, smoker/non-smoker and the region of USA that an individual lives in.

The dataset that I used in this report was obtained from Kaggle. This dataset was published by Packt Publishing online under public domain for the use of readers of book 'Machine Learning with R by Brett Lantz'. The author of the Kaggle dataset formatted the dataset form the source and made it available on Kaggle.

Please find the dataset file attached in the mail along with this report.

b. DATA SET DISCRIPTION

- No of observation (rows): 1338
- Attributes (columns): 7

Independent variables:

i. **age**: age of the individual
ii. **sex**: insurance contractor gender, female, male
iii. **bmi**: Body mass index, providing an understanding of body, weights that are relatively high or low relative to height, objective index of body weight (kg / m ^ 2) using the ratio of height to weight, ideally 18.5 to 24.9
iv. **children**: Number of children covered by health insurance / Number of dependents
v. **smoker**: Smoking
vi. **region**: the beneficiary's residential area in the US, northeast, southeast, southwest, northwest.

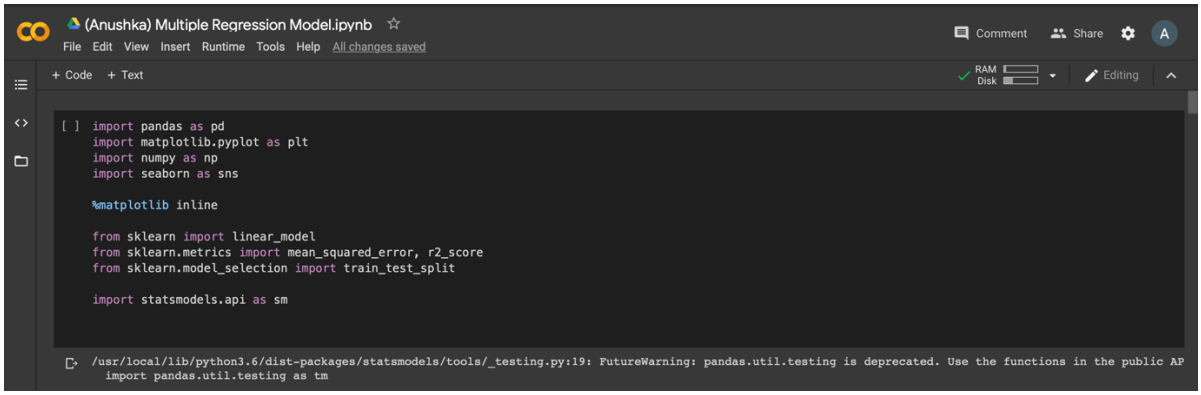
Dependent variable:

charges: Individual medical costs billed by health insurance

4. PROGRAMMING LANGUAGE, LIBRARIES AND APPROACH USED

Programming language that I would be using to solve this linear regression problem is Python. It is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable and hence I would be using this language for dealing with this huge dataset. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

Following screenshot shows the libraries and packages that I would be importing for this project:



a. PACKAGES USED

- **Pandas** is a powerhouse tool that allows you to do anything and everything with tabular or columnar data sets -- analyzing, organizing, sorting, filtering, aggregating, cleaning, calculating, and more!

- **Matplotlib** is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays. One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

- **Numerical Python (numpy)** is a powerful library which efficiently performs matrix operations faster and exceed the python capabilities of data processing.

- **Seaborn** is a library for making statistical graphics in Python. It is built on the top of matplotlib and closely integrated with pandas data structures.

- **Scikit-learn (sklearn)** is doing machine learning with emphasis on predictive modeling with often large and sparse data.

- **Statsmodels** is doing "traditional" statistics and econometrics, with much stronger emphasis on parameter estimation and (statistical) testing.

b. APPROACH USED IN THIS REPORT

In this report I would be solving the problem with two methods using, Scikit -learn and statsmodel.

- I will start by fitting the model using **SKLearn**. After I fit the model, unlike with statsmodels, SKLearn does not automatically print the concepts or have a method like summary. So we have to print the coefficients,intercepts etc. separately.
- After fitting the model with SKLearn, I fit the model using **statsmodels**. Unlike SKLearn, statsmodels doesn't automatically fit a constant, so you need to use the method **sm.add_constant(X)** in order to add a constant. Adding a constant, while not necessary, makes your line fit much better.
- Coefficients can be obtained pretty easily from SKLearn, so the main benefit of statsmodels is the other statistics it provides.
- One of the assumptions of a simple linear regression model is normality of our data. The statistics in the summary table in statsmodel are testing the normality of our data.
- If the Prob(Omnibus) is very small, and I took this to mean <.05 as this is standard statistical practice, then our data is not normal. This is a more precise way than graphing our data to determine if our data is normal.
- Statsmodels also helps us determine which of our variables are statistically significant through the p-values. If our p-value is <.05, then that variable is statistically significant. This is a useful tool to tune your model.

Therefore, SKLearn has more useful features, but statsmodels is a good method to analyze your data before you put it into your model.

5. BUILDING THE MULTIPLE REGRESSION MODEL

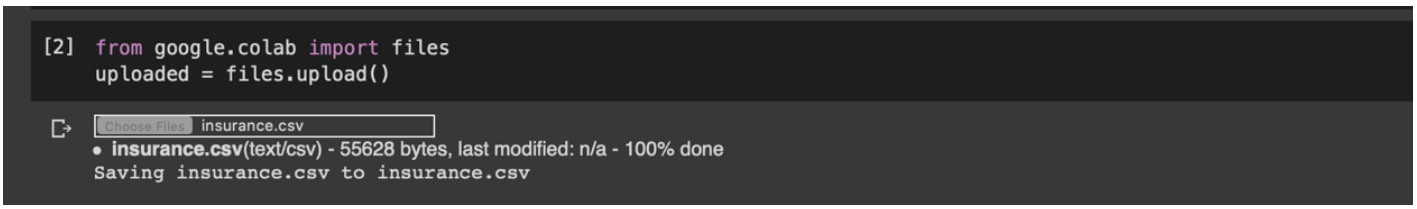
5.1. IMPORTING LIBRARIES

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

%matplotlib inline

from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split

import statsmodels.api as sm
```

5.2. IMPORTING THE DATASET FILE

File name: insurance.csv

```
[2] from google.colab import files
    uploaded = files.upload()
```



5.3. DATA DESCRIPTION AND CHECKING FOR MISSING VALUES

- Importing the data into a dataframe and printing the first five entries of the dataframe using function head().Here, age, sex, bmi, children, smoker and region would be independent variables and charges would be dependent variable.

```
[3] df=pd.read_csv("insurance.csv")
    df.head()
```

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

- Description of the dataset (continuous) attributes.

```
[4] df.describe()
```

	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

- shape function prints the number of rows(observations) and columns(attributes) of the dataframe

```
[5] df.shape
(1338, 7)
```

- ‘non-null’ implies there are no missing values. Now, linear regression can be done on numerical or continuous attributes but not on object type attributes such as sex, smoker and region. Therefore, we would be converting these object type variables into int values (0/1) in preprocessing section.

```
[6] df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
age         1338 non-null int64
sex         1338 non-null object
bmi         1338 non-null float64
children    1338 non-null int64
smoker      1338 non-null object
region      1338 non-null object
charges     1338 non-null float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

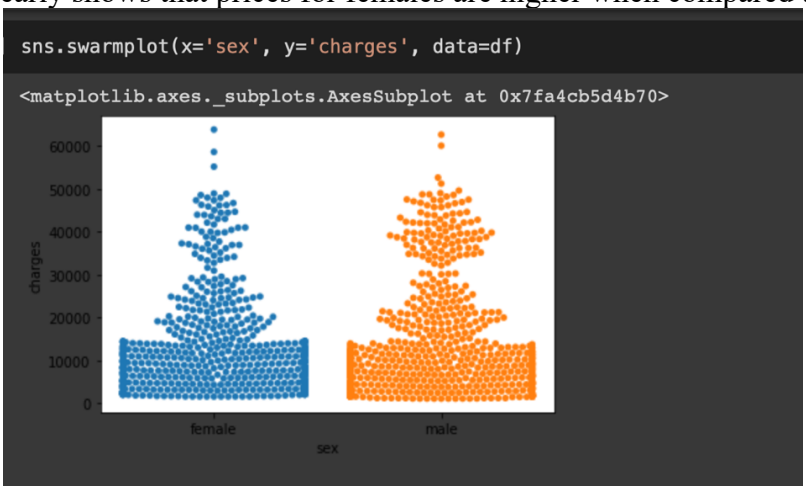
5.4. GRAPHICALLY STUDYING INDEPENDNET VARIABLES Vs DEPENDENT VARIABLE

In this section, I have used matplotlib and seaborn to plot different graphs for better understanding of the variability of the dependent variable based on the independent variables.

- **Sex Vs. Charges**

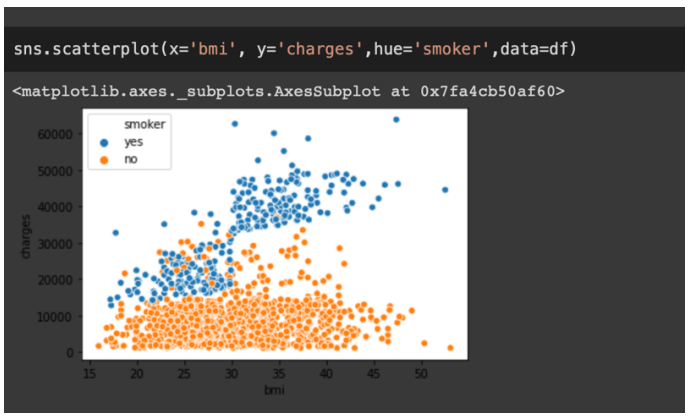
KDE Plot described as Kernel Density Estimate is used for visualizing the Probability Density of a continuous variable. It depicts the probability density at different values in a continuous variable.

The following graph clearly shows that prices for females are higher when compared to males.



- **Smoker Vs. Charges**

Following two graphs implies that the medical bills for a smoker is much more than that of a non-smoker.

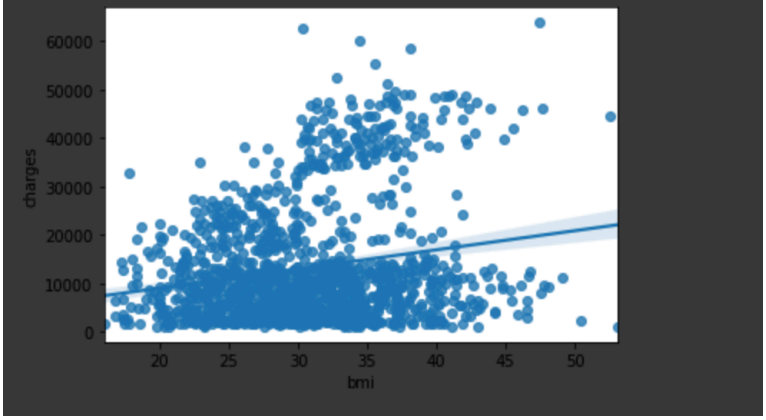


- **BMI vs. Charges**

Left graph implies that charges increase with increase in bmi, basically with increase in weight. Right graph details this representation and divides the composition into male and female.

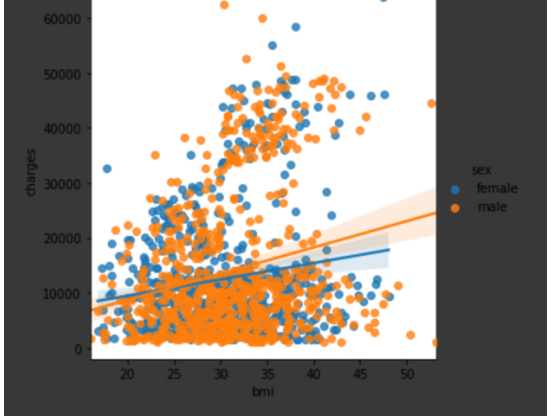
```
sns.regplot(x='bmi',y='charges',data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa4cb4c70b8>
```




```
sns.lmplot(x='bmi',y='charges',hue='sex',data=df)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fa4cb3f8e80>
```

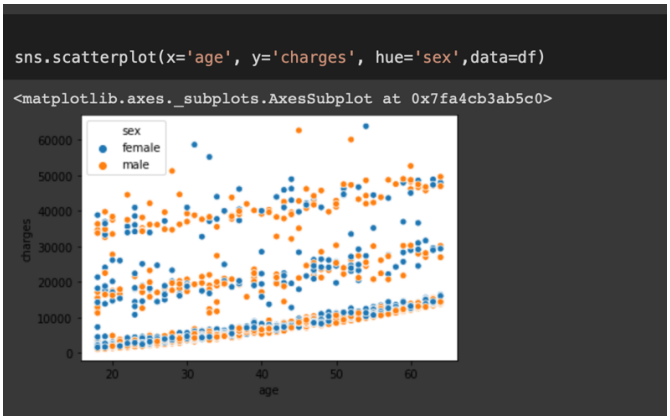


- **Age Vs. Charges**

It can be inferred that charges increase with the age of the individual.

```
sns.scatterplot(x='age', y='charges', hue='sex',data=df)
```

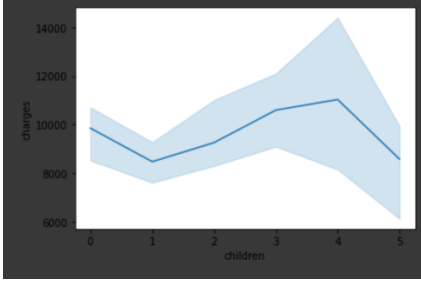
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa4cb3ab5c0>
```



- **Children Vs. Charges**

Charges don't show a prominent trend against the number of children of an individual.

```
sns.lineplot(x='children', y='charges',  estimator=np.median, data=df);
```

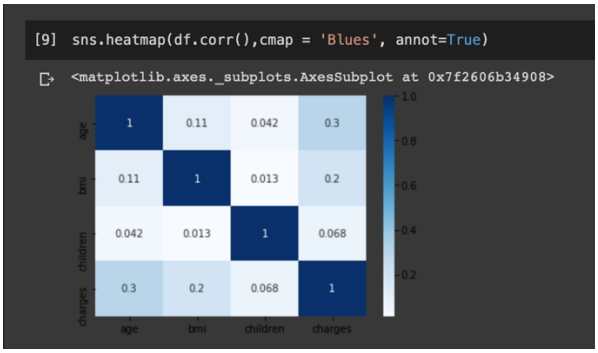



```
df.groupby('children')['charges'].agg('median')
```

```
children
0     9856.95190
1     8483.87015
2     9264.97915
3    10600.54830
4    11033.66170
5     8589.56505
Name: charges, dtype: float64
```

5.5. CORRELATION MATRIX TO CHECK FOR COLLINEARITY

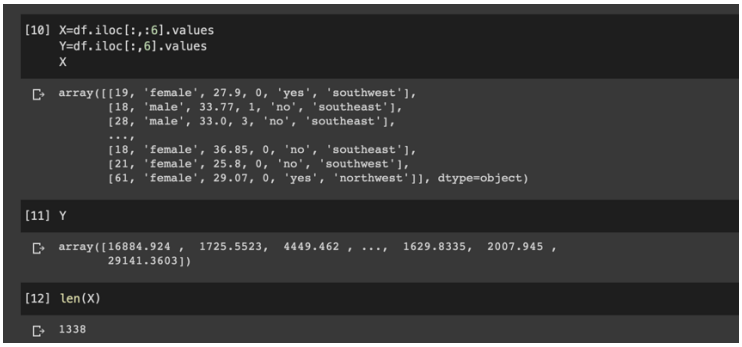
Correlation can be calculated only for the continuous numeric attributes. In the following matrix, no two variables show a high correlation (close to one) that means no two variables represent the same thing.



5.6. SEPERATING DEPENDENT AND INDEPENDENT VARIABLES

Column [0]: age, Column [1]: sex, Column [2]: bmi, Column [3]: children, Column [4]: smoker, Column [5]: region, Column [6]: charges

Array X: contains '0' to '6-1', i.e. 6 attribute columns which constitute all the independent variables.
Array Y: contains charges attribute values.



5.7. USING SKLEARN

5.7.1. PREPROCESSING

5.7.1.1. LABEL ENCODER

Sklearn provides a very efficient tool for encoding the levels of categorical features into numeric values. LabelEncoder encode labels with a value between 0 and n_classes-1 where n is the number of distinct labels. If a label repeats it assigns the same value to as assigned earlier.
Using this:

Column [1]: Sex is represented as

- Male:1
- Female:0

Column [4]: Smoker is represented as

- yes:1
- no:0

Column [5]: Region is represented as

- Northeast:0
- Northwest:1
- Southeast:2
- Southwest:3

Output with these values in the array can be seen in the screenshot below.

```
[13] from sklearn.preprocessing import LabelEncoder
     le=LabelEncoder()

[14] X[:,5]=le.fit_transform(X[:,5])
     X[:,1]=le.fit_transform(X[:,1])
     X[:,4]=le.fit_transform(X[:,4])
     X

array([[19, 0, 27.9, 0, 1, 3],
       [18, 1, 33.77, 1, 0, 2],
       [28, 1, 33.0, 3, 0, 2],
       ...,
       [18, 0, 36.85, 0, 0, 2],
       [21, 0, 25.8, 0, 0, 3],
       [61, 0, 29.07, 0, 1, 1]], dtype=object)
```

5.7.1.2. ONE HOT ENCODER

What one hot encoding does is, it takes a column which has categorical data, which has been label encoded and then splits the column into multiple columns. The numbers are replaced by 1s and 0s, depending on which column has what value. In our case, we'll get four new columns, one for each region.

Column 0-4 becomes column 4-8 and four columns are appended before these giving a total of 9 columns.

- Column [0]: Northeast
- Column [1]: Northwest
- Column [2]: Southeast
- Column [3]: Southwest

```
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
ct = ColumnTransformer([("region", OneHotEncoder(),[5])], remainder="passthrough")
X = ct.fit_transform(X)
X

array([[0.0, 0.0, 0.0, ..., 27.9, 0, 1],
       [0.0, 0.0, 1.0, ..., 33.77, 1, 0],
       [0.0, 0.0, 1.0, ..., 33.0, 3, 0],
       ...,
       [0.0, 0.0, 1.0, ..., 36.85, 0, 0],
       [0.0, 0.0, 0.0, ..., 25.8, 0, 0],
       [0.0, 1.0, 0.0, ..., 29.07, 0, 1]], dtype=object)

[16] X.shape

(1338, 9)
```

Dummy Variable Trap:
The Dummy variable trap is a scenario where there are attributes which are highly correlated (Multicollinear) and one variable predicts the value of others. When we use one hot encoding for handling the categorical data, then one dummy variable (attribute) can be predicted with the help of other dummy variables. Hence, one dummy variable is highly correlated with other dummy variables. Using all dummy variables for regression models lead to **dummy variable trap**. So, the regression models should be designed excluding one dummy variable.

```
[17] # avoid from dummy variable trap
     X=X[:,1:]
     X

array([[0.0, 0.0, 1.0, ..., 27.9, 0, 1],
       [0.0, 1.0, 0.0, ..., 33.77, 1, 0],
       [0.0, 1.0, 0.0, ..., 33.0, 3, 0],
       ...,
       [0.0, 1.0, 0.0, ..., 36.85, 0, 0],
       [0.0, 0.0, 1.0, ..., 25.8, 0, 0],
       [1.0, 0.0, 0.0, ..., 29.07, 0, 1]], dtype=object)

[18] X.shape

(1338, 8)
```

5.7.2. TRAIN AND TEST MODEL SPLITTING

The dataset is split into train and test set. The model is trained using train test and then values are predicted using this model with test set as input.

Train set: 80% of 1338 (1070 observations)
Test set: 20% of 1338 (268 observations)

- X_train: 1070 observations and 8 independent variables.
- Y_train: 1070 observations and dependent variable.
- X_test: 268 observations and 8 independent variables.
- Y_test: 268 observations and dependent variable.

```
[19] X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=0)

[24] print(X_train.shape)
     print(Y_train.shape)
     print(X_test.shape)
     print(Y_test.shape)

     (1070, 8)
     (1070,)
     (268, 8)
     (268,)
```

5.7.3. COEFFICIENT AND DEPENDENT VARIABLE PREDICTION

- Training of the model using LinearRegression() function

```
[25] model = linear_model.LinearRegression()
     model.fit(X_train, Y_train)

     LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

- Intercept and coefficient values

```
[26] print('Intercept: \n', model.intercept_)

     Intercept:
      -11828.073537474787

[27] coeff_df = pd.DataFrame(model.coef_, ['region1','region2','region3','age', 'sex', 'bmi', 'children', 'smoker'], columns=['Coefficient'])
     coeff_df
```

	Coefficient
region1	-260.132732
region2	-913.278834
region3	-761.948706
age	253.700500
sex	-15.463728
bmi	335.962814
children	436.910121
smoker	23605.017267

- Predicted y values saved in an array Y_pred

```
[28] Y_pred = model.predict(X_test)
     Y_pred

     array([ 9.25496051e+03, 1.08433751e+03, 9.80417085e+03,
             3.77104596e+03, 1.04318587e+04, 9.00931722e+03, 4.00749509e+04,
             1.56889543e+04, 1.38794545e+04, 2.47597127e+04, 5.16638285e+03,
             1.26109277e+04, 3.07691018e+04, 3.35498325e+04, 3.67154946e+03,
             3.97568613e+03, 3.98729942e+03, 3.05285774e+04, 3.95053023e+04,
             2.78105036e+04, 5.09258923e+03, 1.06042481e+04, 7.82952256e+03,
             3.59255553e+03, 1.02128745e+04, 5.72038147e+03, 3.42627499e+03,
             3.30210242e+04, 3.84738218e+04, 1.60534782e+04, 7.16491905e+03,
```

- Comparison between Actual and Predicted values

5.7.4. INFERENCE USING GRAPHICAL REPRESENTATION

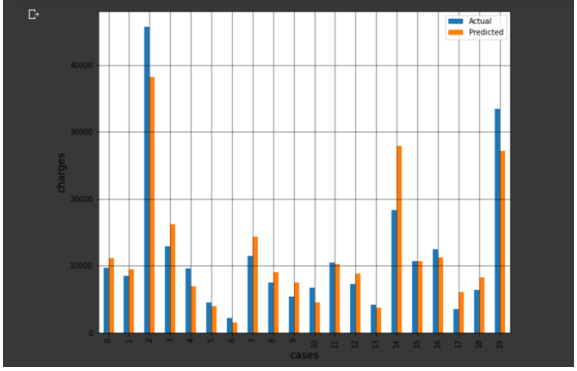
- Comparison between Actual and Predicted values

df2 holds first 20 values of the dataframe with actual and predicted columns for visual representation.

```
[29] df1 = pd.DataFrame({'Actual': Y_test, 'Predicted': Y_pred})
     df2 = df1.head(20)
     df2
```

	Actual	Predicted
0	9724.53000	11169.927119
1	8547.69130	9486.709085
2	45702.02235	38181.123053
3	12950.07120	16266.313289
4	9644.25250	6914.648007
5	4500.33925	3963.484876
6	2198.18985	1579.396913
7	11436.73815	14385.256569
8	7537.16390	9012.579696
9	5425.02335	7508.460677
10	6753.03800	4491.767282
11	10493.94580	10279.583941
12	7337.74800	8801.297506
13	4185.09790	3798.020130
14	18310.74200	27926.200966
15	10702.64240	10715.115820
16	12523.60480	11288.975621
17	3490.54910	6105.017680
18	6457.84340	8241.041174
19	33475.81715	27144.508950

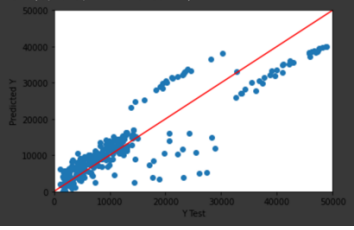
```
[30] import matplotlib.pyplot as plt
     df2.plot(kind='bar',figsize=(10,8))
     plt.grid(which='major', linestyle='-', linewidth='0.5', color='black')
     plt.grid(which='minor', linestyle=':', linewidth='0.5', color='white')
     plt.xlabel('cases',color='black',fontsize=14)
     plt.ylabel('charges',color='black',fontsize = 14)

     plt.show()
```



- Predicted Y Vs. Actual Y

The points lying on the line y=x are the ones where predicted value is the same as the actual value.

```
[31] plt.scatter(Y_test,Y_pred)

     plt.ylim(0,50000)
     plt.xlim(0,50000)

     x = np.linspace(0, 500000, 30)
     plt.plot(x, x + 0,'-r', linestyle='solid')

     plt.xlabel('Y Test')
     plt.ylabel('Predicted Y')

     Text(0, 0.5, 'Predicted Y')
```



5.7.5. EVALUATION METRICS

- R^2 value of 0.7999 is obtained, which means 79.99% of the variability in the dependent variable(charges) can be explained by the independent variables.

```
[32] from sklearn import metrics
     print('Mean Absolute Error:', metrics.mean_absolute_error(Y_test, Y_pred))
     print('Mean Squared Error:', metrics.mean_squared_error(Y_test, Y_pred))
     print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(Y_test, Y_pred)))

     print('rsquare (Coefficient of determination))',r2_score(Y_test, Y_pred))

     Mean Absolute Error: 3933.2726494052367
     Mean Squared Error: 31827950.2295238
     Root Mean Squared Error: 5641.6265588501865
     rsquare (Coefficient of determination)) 0.7999876970680435
```

5.8. USING STATSMODELS

5.8.1. ORDINARY LEAST SQUARE METHOD USING BACKWARD ELIMINATION

In statistics, **ordinary least squares** (**OLS**) is a type of linear least squares method for estimating the unknown parameters in a linear regression model. OLS chooses the parameters of a linear function of a set of explanatory variables by the principle of least squares: minimizing the sum of the squares of the differences between the observed dependent variable (values of the variable being observed) in the given dataset and those predicted by the linear function.

To perform this, all values are converted into float64 type first.

- Backward Elimination:

The attribute with the highest P-value is removed in every successive step from the X_opt and we stop when the R-squared value starts decreasing.

```
[33] X.shape
     (1338, 8)

[34] X = X.astype('float64')
```

- X2 has the appended constant column[0] as discussed in section 4.2.

```
X2 = sm.add_constant(X)
X_opt = X2[:, [0, 1, 2, 3, 4, 5, 6, 7, 8]]
regressor_OLS = sm.OLS(endog = Y, exog = X_opt).fit()
regressor_OLS.summary()
```

OLS Regression Results

Dep. Variable:	y	R-squared:	0.751
Model:	OLS	Adj. R-squared:	0.749
Method:	Least Squares	F-statistic:	500.8
Date:	Tue, 31 Mar 2020	Prob (F-statistic):	0.00
Time:	14:33:05	Log-Likelihood:	-13548.
No. Observations:	1338	AIC:	2.711e+04
Df Residuals:	1329	BIC:	2.716e+04
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1.194e+04	987.819	-12.086	0.000	-1.39e+04	-1e+04
x1	-352.9639	476.276	-0.741	0.459	-1287.298	581.370
x2	-1035.0220	478.692	-2.162	0.031	-1974.097	-95.947
x3	-960.0510	477.933	-2.009	0.045	-1897.636	-22.466
x4	256.8564	11.899	21.587	0.000	233.514	280.199
x5	-131.3144	332.945	-0.394	0.693	-784.470	521.842
x6	339.1935	28.599	11.860	0.000	283.088	395.298
x7	475.5005	137.804	3.451	0.001	205.163	745.838
x8	2.385e+04	413.153	57.723	0.000	2.3e+04	2.47e+04

Omnibus:	300.366	Durbin-Watson:	2.088
Prob(Omnibus):	0.000	Jarque-Bera (JB):	718.887
Skew:	1.211	Prob(JB):	7.86e-157
Kurtosis:	5.651	Cond. No.	311.

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

```
[37] X_opt = X2[:, [0, 1, 2, 3, 4, 6, 7, 8]]
     regressor_OLS = sm.OLS(endog = Y, exog = X_opt).fit()
     regressor_OLS.summary()
```

OLS Regression Results

Dep. Variable:	y	R-squared:	0.751
Model:	OLS	Adj. R-squared:	0.750
Method:	Least Squares	F-statistic:	572.7
Date:	Tue, 31 Mar 2020	Prob (F-statistic):	0.00
Time:	14:35:52	Log-Likelihood:	-13548.
No. Observations:	1338	AIC:	2.711e+04
Df Residuals:	1330	BIC:	2.715e+04
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1.199e+04	978.762	-12.250	0.000	-1.39e+04	-1.01e+04
x1	-352.1821	476.120	-0.740	0.460	-1286.211	581.847
x2	-1034.3601	478.537	-2.162	0.031	-1973.130	-95.590
x3	-959.3747	477.778	-2.008	0.045	-1896.656	-22.094
x4	256.9736	11.891	21.610	0.000	233.646	280.301
x5	338.6646	28.559	11.858	0.000	282.639	394.690
x6	474.5665	137.740	3.445	0.001	204.355	744.778
x7	2.384e+04	411.856	57.875	0.000	2.3e+04	2.46e+04

Omnibus:	300.735	Durbin-Watson:	2.089
Prob(Omnibus):	0.000	Jarque-Bera (JB):	720.516
Skew:	1.212	Prob(JB):	3.48e-157
Kurtosis:	5.654	Cond. No.	309.

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

```
[38] X_opt = X2[:, [0, 2, 3, 4, 6, 7, 8]]
     regressor_OLS = sm.OLS(endog = Y, exog = X_opt).fit()
     regressor_OLS.summary()
```

OLS Regression Results

Dep. Variable:	y	R-squared:	0.751
Model:	OLS	Adj. R-squared:	0.750
Method:	Least Squares	F-statistic:	668.3
Date:	Tue, 31 Mar 2020	Prob (F-statistic):	0.00
Time:	14:35:58	Log-Likelihood:	-13548.
No. Observations:	1338	AIC:	2.711e+04
Df Residuals:	1331	BIC:	2.715e+04
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1.217e+04	949.538	-12.812	0.000	-1.4e+04	-1.03e+04
x1	-858.4696	415.206	-2.068	0.039	-1672.998	-43.941
x2	-782.7452	413.756	-1.892	0.059	-1594.430	28.940
x3	257.0064	11.889	21.617	0.000	233.683	280.330
x4	338.6413	28.554	11.860	0.000	282.625	394.657
x5	471.5441	137.656	3.426	0.001	201.498	741.590
x6	2.384e+04	411.659	57.921	0.000	2.3e+04	2.47e+04

Omnibus:	300.125	Durbin-Watson:	2.092
Prob(Omnibus):	0.000	Jarque-Bera (JB):	716.587
Skew:	1.211	Prob(JB):	2.48e-156
Kurtosis:	5.643	Cond. No.	295.

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly

```
X_opt = X2[:, [0, 2, 4, 6, 7, 8]]
regressor_OLS = sm.OLS(endog = Y, exog = X_opt).fit()
regressor_OLS.summary()
```

OLS Regression Results

Dep. Variable:	y	R-squared:	0.750
Model:	OLS	Adj. R-squared:	0.749
Method:	Least Squares	F-statistic:	799.7
Date:	Tue, 31 Mar 2020	Prob (F-statistic):	0.00
Time:	14:37:58	Log-Likelihood:	-13550.
No. Observations:	1338	AIC:	2.711e+04
Df Residuals:	1332	BIC:	2.714e+04
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1.228e+04	948.666	-12.940	0.000	-1.41e+04	-1.04e+04
x1	-578.8620	388.384	-1.490	0.136	-1340.774	183.050
x2	257.1365	11.901	21.607	0.000	233.791	280.482
x3	333.4448	28.449	11.721	0.000	277.635	389.255
x4	468.0668	137.777	3.397	0.001	197.783	738.350
x5	2.385e+04	412.023	57.895	0.000	2.3e+04	2.47e+04

Omnibus:	302.906	Durbin-Watson:	2.089
Prob(Omnibus):	0.000	Jarque-Bera (JB):	726.891
Skew:	1.220	Prob(JB):	1.44e-158
Kurtosis:	5.662	Cond. No.	294.

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe

5.8.2. CHOOSING OPTIMAL SET OF INDEPENDENT VARIABLES AND BUILDING THE MODEL

- The third case above is chosen as the optimal case and two new models are built
  i) regressor_OLS_optimal using statsmodel
  ii) model2 using sklearn

```
[40] X_opt = X2[:, [0, 2, 3, 4, 6, 7, 8]]

[41] X_train2, X_test2, Y_train2, Y_test2 = train_test_split(X_opt, Y, test_size = 0.2, random_state = 0)
     model2 = linear_model.LinearRegression()
     model2.fit(X_train2, Y_train2)
     Y_pred2 = model2.predict(X_test2)
     Y_pred2

  array([11179.78396531,  9478.43960021, 38323.79532847, 16408.17920146,
          7036.28878939,  3837.87153244,  1436.15007609, 14507.76397291,
          9131.84254997,  7642.04803594,  4348.54524981, 10401.03254095,
          8794.02242961,  3787.08759217, 28064.52785452, 10706.01217353,
         11428.61646478,  5978.97671757,  8376.22947501, 27021.75728786,
         33785.0115461 , 14478.74114159, 11600.75786066, 32133.57489599,
```

```
[42] print(X_train2.shape)
     print(Y_train2.shape)
     print(X_test2.shape)
     print(Y_test2.shape)

  (1070, 7)
  (1070,)
  (268, 7)
  (268,)
```

```
X_opt = X2[:, [0, 2, 3, 4, 6, 7, 8]]
regressor_OLS_optimal = sm.OLS(endog = Y, exog = X_opt).fit()
regressor_OLS_optimal.summary()
```

OLS Regression Results

Dep. Variable:	y	R-squared:	0.751
Model:	OLS	Adj. R-squared:	0.750
Method:	Least Squares	F-statistic:	668.3
Date:	Tue, 31 Mar 2020	Prob (F-statistic):	0.00
Time:	14:35:58	Log-Likelihood:	-13548.
No. Observations:	1338	AIC:	2.711e+04
Df Residuals:	1331	BIC:	2.715e+04
Df Model:	6		
Covariance Type:	nonrobust		

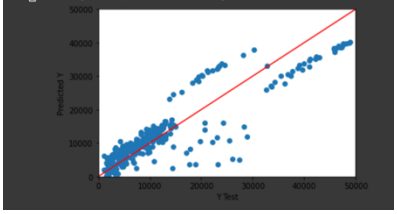
	coef	std err	t	P>|t|	[0.025	0.975]
const	-1.217e+04	949.538	-12.812	0.000	-1.4e+04	-1.03e+04
x1	-858.4696	415.206	-2.068	0.039	-1672.998	-43.941
x2	-782.7452	413.756	-1.892	0.059	-1594.430	28.940
x3	257.0064	11.889	21.617	0.000	233.683	280.330
x4	338.6413	28.554	11.860	0.000	282.625	394.657
x5	471.5441	137.656	3.426	0.001	201.498	741.590
x6	2.384e+04	411.659	57.921	0.000	2.3e+04	2.47e+04

Omnibus:	300.125	Durbin-Watson:	2.092
Prob(Omnibus):	0.000	Jarque-Bera (JB):	716.587
Skew:	1.211	Prob(JB):	2.48e-156
Kurtosis:	5.643	Cond. No.	295.

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

- The graph and metrics obtained using this method is very similar to what we obtained using sklearn.

```
[43] plt.scatter(Y_test2,Y_pred2)

     plt.ylim(0,50000)
     plt.xlim(0,50000)

     x = np.linspace(0, 500000, 30)
     plt.plot(x, x + 0,'-r', linestyle='solid')

     plt.xlabel('Y Test')
     plt.ylabel('Predicted Y')

  Text(0, 0.5, 'Predicted Y')
```



5.9.RESIDUAL ANALYSIS OF THE MODEL BUILT USING STATSMODEL PACKAGE

5.9.1. CALCULATE RESIDUAL METRICS

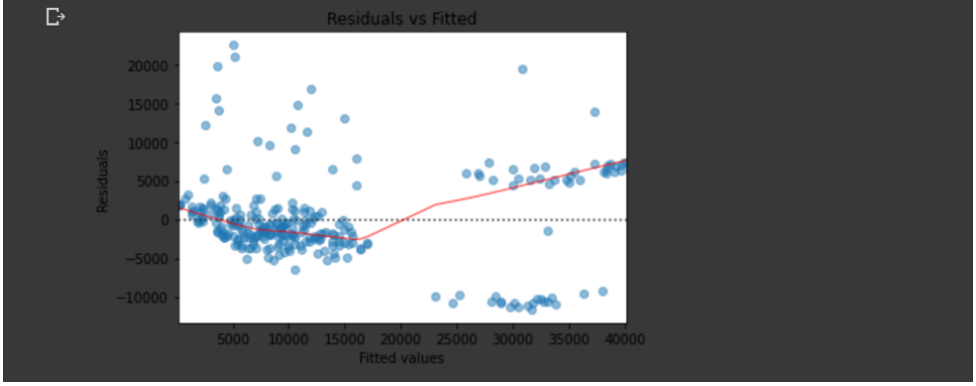
```
[45] residuals= Y_test2-Y_pred2
```

```
[46]
     # normalized residuals
     model_norm_residuals = regressor_OLS_optimal.get_influence().resid_studentized_internal
     # absolute squared normalized residuals
     model_norm_residuals_abs_sqrt = np.sqrt(np.abs(model_norm_residuals))
     # absolute residuals
     model_abs_resid = np.abs(residuals)
     # leverage, from statsmodels internals
     model_leverage = regressor_OLS_optimal.get_influence().hat_matrix_diag
```

5.9.2. RESIDUAL Vs. FITTED VALUES GRAPH

When conducting a residual analysis, a "**residuals versus fits plot**" is the most frequently created plot. It is a scatter plot of residuals on the *y* axis and fitted values (estimated responses) on the *x* axis. The plot is used to detect non-linearity, unequal error variances, and outliers.

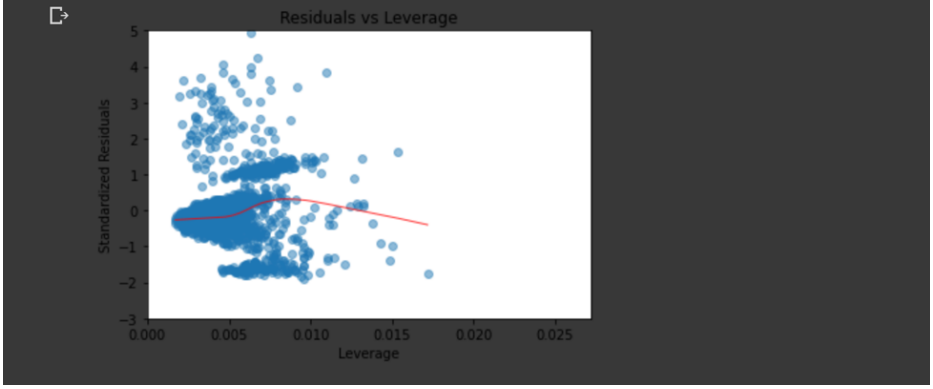
```
[47] plot1 = plt.figure()
     plot1.axes[0] = sns.residplot(Y_pred2, residuals,
                                   lowess=True,
                                   scatter_kws={'alpha': 0.5},
                                   line_kws={'color': 'red', 'lw': 1, 'alpha': 0.8})

     plot1.axes[0].set_title('Residuals vs Fitted')
     plot1.axes[0].set_xlabel('Fitted values')
     plot1.axes[0].set_ylabel('Residuals');
```



5.9.3. RESIDUAL Vs. LEVERAGE GRAPH

The Residuals vs. Leverage plots helps you identify influential data points on your model. Outliers can be influential, though they don't necessarily have to it and some points within a normal range in your model could be very influential.

```
[48] plot2 = plt.figure();
     plt.scatter(model_leverage, model_norm_residuals, alpha=0.5);
     sns.regplot(model_leverage, model_norm_residuals,
                 scatter=False,
                 ci=False,
                 lowess=True,
                 line_kws={'color': 'red', 'lw': 1, 'alpha': 0.8});
     plot2.axes[0].set_xlim(0, max(model_leverage)+0.01)
     plot2.axes[0].set_ylim(-3, 5)
     plot2.axes[0].set_title('Residuals vs Leverage')
     plot2.axes[0].set_xlabel('Leverage')
     plot2.axes[0].set_ylabel('Standardized Residuals');
```



6. INFERENCE

6.1. Model1 made using SKLEARN

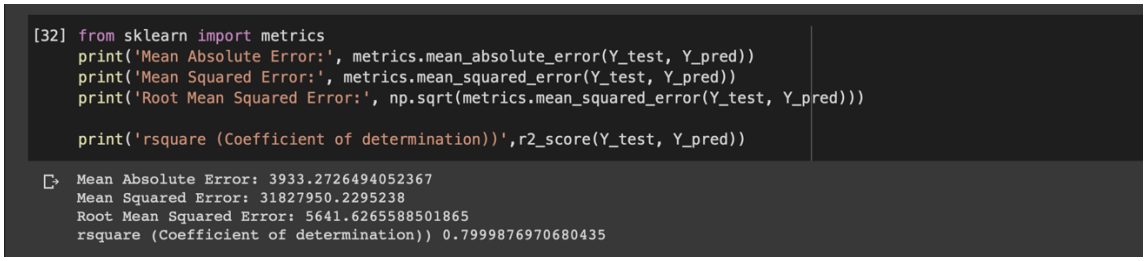
Refer to section 5.7.3 to find out how we obtained these values.

Attribute	Var.	value
Intercept	b_0	-11828.0735
Region1	b_1	-260.132
Region2	b_2	-913.278
Region3	b_3	-761.948
age	b_4	253.700
sex	b_5	-15.463
bmi	b_6	335.962
children	b_7	436.910
smoker	b_8	23605.017

Therefore, the equation is:

$$\hat{y}_j = -11828.073 - 260.132\,x_1 - 913.278\,x_2 - 761.948\,x_3 + 253.700\,x_4 - 15.463x_5 + 335.962x_6 + 436.910x_7 + 23605.017x_8$$

- We can say that with different regions have different level of impacts on the charges. The charges decrease based on the region.
- With one unit increase in age, the charges increase by 253.700 units when all other attributes are kept constant.
- Sex value is '1' for males and '0' for females. Therefore, the result implies that the charges for males are less than that of females.
- With one unit increase in bmi, the charges increase by 335.962 units when all other attributes are kept constant.
- With one unit increase in children, the charges increase by 436.910 units when all other attributes are kept constant.
- Smoker attribute is '1' if a person is a smoker and '0' for a non-smoker. Therefore, for a smoker, charges increase drastically by 23605.017 units.



- R^2 value of 0.7999 is obtained, which means 79.99% of the variability in the dependent variable(charges) can be explained by the independent variables.
- Mean absolute Error obtained is 3933.272 which is 29% of the mean charges value (obtained in 5.3), which is not that great but is good enough to predict approximate values.

- **Note:** region 1 is Northwest, region 2 is Southeast and region 3 is Southwest. (Northeast was dropped because of dummy trap).

6.2. regressor_OLS_optimal model made using STATSMODELs.

Model2 is built in sklearn using regressor_OLS_optimal model made using statsmodels.
Only major difference between model1 and model2 is that attributes 'sex' and 'region1' are dropped for better fit.

```
X_opt = X2[:, [0, 2, 3, 4, 6, 7, 8]]
regressor_OLS_optimal = sm.OLS(endog = Y, exog = X_opt).fit()
regressor_OLS_optimal.summary()
```

OLS Regression Results

Dep. Variable:	y	R-squared:	0.751
Model:	OLS	Adj. R-squared:	0.750
Method:	Least Squares	F-statistic:	668.3
Date:	Tue, 31 Mar 2020	Prob (F-statistic):	0.00
Time:	14:35:58	Log-Likelihood:	-13548.
No. Observations:	1338	AIC:	2.711e+04
Df Residuals:	1331	BIC:	2.715e+04
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1.217e+04	949.538	-12.812	0.000	-1.4e+04	-1.03e+04
x1	-858.4696	415.206	-2.068	0.039	-1672.998	-43.941
x2	-782.7452	413.756	-1.892	0.059	-1594.430	28.940
x3	257.0064	11.889	21.617	0.000	233.683	280.330
x4	338.6413	28.554	11.860	0.000	282.625	394.657
x5	471.5441	137.656	3.426	0.001	201.498	741.590
x6	2.384e+04	411.659	57.921	0.000	2.3e+04	2.47e+04

Omnibus:	300.125	Durbin-Watson:	2.092
Prob(Omnibus):	0.000	Jarque-Bera (JB):	716.587
Skew:	1.211	Prob(JB):	2.48e-156
Kurtosis:	5.643	Cond. No.	295.

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Attribute	Var.	value
Intercept	b_0	-1.217×10^4
Region2	b_1	-858.4696
Region3	b_2	-782.745
age	b_3	257.0064
bmi	b_4	338.641
children	b_5	471.5441
smoker	b_6	2.384×10^4

Therefore, the equation is:

$$\hat{y}_j = -1.217 \times 10^4 - 858.4696x_1 - 782.745x_2 - 257.0064x_3 + 338.641x_4 + 471.5441x_5 + (2.384 \times 10^4)x_6$$

- Both the regions have different level of impacts on the charges. The charges decrease based on the region. Region2(Southeast) is cheaper than Region3(Southwest).
- With one unit increase in age, the charges increase by 257.0064 units when all other attributes are kept constant.
- With one unit increase in bmi, the charges increase by 338.641 units when all other attributes are kept constant.
- With one unit increase in children, the charges increase by 471.5441 units when all other attributes are kept constant.
- Smoker attribute is '1' if a person is a smoker and '0' for a non-smoker. Therefore, for a smoker, charges increase drastically by 23605.017 units.

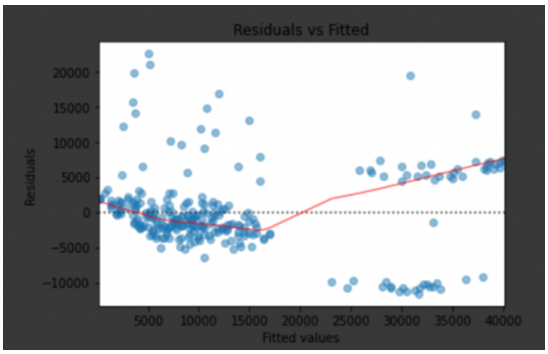
- R^2 value of 0.751 is obtained, which means 75.1% of the variability in the dependent variable(charges) can be explained by the independent variables.
- The P-value for each term tests the null hypothesis that the coefficient is equal to zero (no effect). A low P-value (< 0.05) indicates that you can reject the null hypothesis. In other words, a predictor that has a low P-value is likely to be a meaningful addition to your model because changes in the predictor's value are related to changes in the response variable. From the table, looking at the P-values it shows all attributes are a meaningful addition to our model except Region3.
- F-test:
  i) H_0: $\beta_1 = \beta_2 = ... = \beta_p = 0$
  ii) H_1: at least one $\beta_i \neq 0$, $i = 1, ..., k$
  F-statistic *being a huge value, we can reject* H_0 *that means all beta's have a non-zero value. All the independent variables define variability in the dependent variable.*

6.3. RESIDUAL ANALYSIS

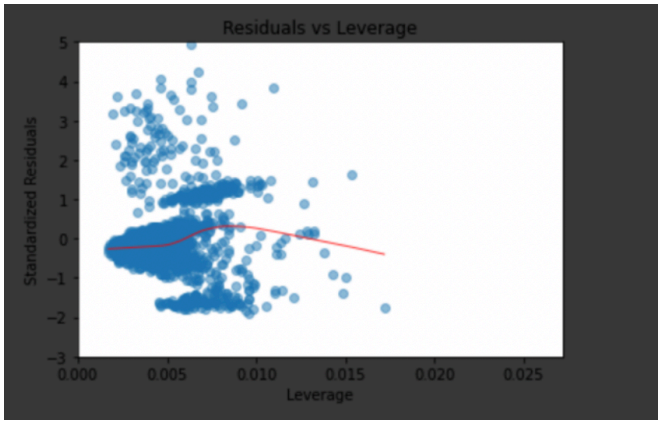
6.3.1. RESIDUAL Vs. FITTED

A good residual vs fitted plot has three characteristics:

- The residuals "bounce randomly" around the 0 line. This suggests that the assumption that the relationship is linear is reasonable. (some of our residual values are far away from the 0 line)
- The residuals values should be equally and randomly spaced around the 0 line.(this is not the ideal case as our fit is not a straight line. This can be so because of the outliers present)
- No one residual "stands out" from the basic random pattern of residuals. This suggests that there are no outliers.(In our case some of the residual values stand out at the extreme top and bottom. This is so because of inefficient preprocessing)



6.3.2. RESIDUAL Vs. LEVERAGE



- The Residuals vs. Leverage plots helps you identify influential data points on your model. Outliers can be influential, though they don't necessarily have to it and some points within a normal range in your model could be very influential.
- Ideally it should be a horizontal line near the 0-line.
- Our result shows that because of some of the outliers being influential points, our line is slightly curved.