linear-regression-tutorial-2315d0

April 27, 2023

We will build a Linear regression model for Medical cost dataset. The dataset consists of age, sex, BMI(body mass index), children, smoker and region feature, which are independent and charge as a dependent feature. We will predict individual medical costs billed by health insurance.

1 Definition & Working principle

Let's build model using **Linear regression**.

Linear regression is a **supervised learining** algorithm used when target / dependent variable **continues** real number. It establishes relationship between dependent variable y and one or more independent variable x using best fit line. It work on the principle of ordinary least square (OLS) / Mean square errror (MSE). In statistics ols is method to estimated unkown parameter of linear regression function, it's goal is to minimize sum of square difference between observed dependent variable in the given data set and those predicted by linear regression fuction.

1.1 Hypothesis representation

We will use $\mathbf{x_i}$ to denote the independent variable and $\mathbf{y_i}$ to denote dependent variable. A pair of $(\mathbf{x_i}, \mathbf{y_i})$ is called training example. The subscripe $\mathbf{i}$ in the notation is simply index into the training set. We have $\mathbf{m}$ training example then $\mathbf{i} = \mathbf{1}, \mathbf{2}, \mathbf{3}, ...\mathbf{m}$.

The goal of supervised learning is to learn a *hypothesis function* $\mathbf{h}$, for a given training set that can used to estimate $\mathbf{y}$ based on $\mathbf{x}$. So hypothesis fuction represented as

$$\mathbf{h}\left(\mathbf{x_i}\right) = {}_{\mathbf{0}} + {}_{\mathbf{1}}\mathbf{x_i}$$

${}_{\mathbf{0}}, {}_{\mathbf{1}}$ are parameter of hypothesis.This is equation for **Simple / Univariate Linear regression**.

For **Multiple Linear regression** more than one independent variable exit then we will use $\mathbf{x_{ij}}$ to denote indepedent variable and $\mathbf{y_i}$ to denote dependent variable. We have $\mathbf{n}$ independent variable then $\mathbf{j} = \mathbf{1}, \mathbf{2}, \mathbf{3}.....\mathbf{n}$. The hypothesis function represented as

$$\mathbf{h}\left(\mathbf{x_i}\right) = {}_{\mathbf{0}} + {}_{\mathbf{1}}\mathbf{x_{i1}} + {}_{\mathbf{2}}\mathbf{x_{i2}} + ... {}_{\mathbf{j}}\mathbf{x_{ij}} ..... {}_{\mathbf{n}}\mathbf{x_{mn}}$$

${}_{\mathbf{0}}, {}_{\mathbf{1}}, ... {}_{\mathbf{j}} .... {}_{\mathbf{n}}$ are parameter of hypothesis, $\mathbf{m}$ Number of training exaples, $\mathbf{n}$ Number of independent variable, $\mathbf{x_{ij}}$ is $\mathbf{i^{th}}$ training exaple of $\mathbf{j^{th}}$ feature.

1.2 Import Library and Dataset

Now we will import couple of python library required for our analysis and import dataset

[7]:
```
# Import library
import pandas  as pd #Data manipulation
import numpy as np #Data manipulation
import matplotlib.pyplot as plt # Visualization
import seaborn as sns #Visualization
plt.rcParams['figure.figsize'] = [8,5]
plt.rcParams['font.size'] =14
plt.rcParams['font.weight']= 'bold'
plt.style.use('seaborn-whitegrid')
```

[8]:
```
# Import dataset
#path ='dataset/'
path = '../input/'
df = pd.read_csv(path+'insurance.csv')
print('\nNumber of rows and columns in the data set: ',df.shape)
print('')

#Lets look into top few rows and columns in the dataset
df.head()
```

```
Number of rows and columns in the data set:  (1338, 7)
```

[8]:

	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

Now we have import dataset. When we look at the shape of dataset it has return as (1338,7).So there are $\mathbf{m = 1338}$ training exaple and $\mathbf{n = 7}$ independent variable. The target variable here is charges and remaining six variables such as age, sex, bmi, children, smoker, region are independent variable. There are multiple independent variable, so we need to fit Multiple linear regression. Then the hypothesis function looks like

$$\mathbf{h\left(x_i\right) = {}_0 + {}_1 age + {}_2 sex + {}_3 bmi + {}_4 children + {}_5 smoker + {}_6 region}$$

This multiple linear regression equation for given dataset.
If $\mathbf{i = 1}$ then

$$\mathbf{h\left(x_1\right) = {}_0 + {}_1 19 + {}_2 female + {}_3 27.900 + {}_4 1 + {}_5 yes + {}_6 southwest}$$

$$\mathbf{y_1 = 16884.92400}$$

If $\mathbf{i = 3}$ then

$$\mathbf{h\left(x_3\right) = {}_0 + {}_1 28 + {}_2 male + {}_3 33.000 + {}_4 3 + {}_5 no + {}_6 northwest}$$

$$\mathbf{y_3 = 4449.46200}$$

Note: In python index starts from 0.

$$\mathbf{x_1} = \begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & x_{16} \end{pmatrix} = \begin{pmatrix} 19 & \text{female} & 27.900 & 1 & \text{no} & \text{northwest} \end{pmatrix}$$

1.3 Matrix Formulation

In general we can write above vector as

$$\mathbf{x_{ij}} = \begin{pmatrix} \mathbf{x_{i1}} & \mathbf{x_{i2}} & \cdots & \mathbf{x_{in}} \end{pmatrix}$$

Now we combine all aviable individual vector into single input matrix of size (m, n) and denoted it by $\mathbf{X}$ input matrix, which consist of all training exaples,

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ x_{31} & x_{32} & \cdots & x_{3n} \\ \cdot & \cdot & \cdots & \cdot \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{pmatrix}_{(m,n)}$$

We represent parameter of function and dependent variable in vactor form as

$$\theta = \begin{pmatrix} \theta_0 \\ \theta_1 \\ \cdot \\ \cdot \\ \theta_j \\ \cdot \\ \cdot \\ \theta_n \end{pmatrix}_{(n+1,1)} \qquad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ y_i \\ \cdot \\ \cdot \\ y_m \end{pmatrix}_{(m,1)}$$

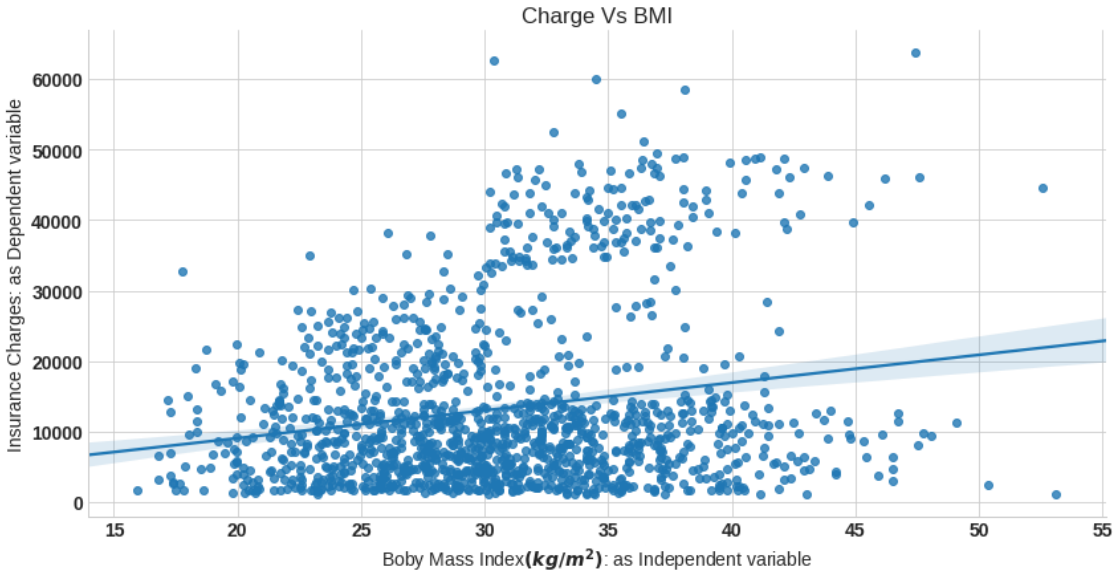
So we represent hypothesis function in vectorize form

$$\mathbf{h\left(x\right) = X}$$

.

[9]:
```
""" for our visualization purpose will fit line using seaborn library only for␣
 ↪bmi as independent variable
and charges as dependent variable"""

sns.lmplot(x='bmi',y='charges',data=df,aspect=2,height=6)
plt.xlabel('Boby Mass Index$(kg/m^2)$: as Independent variable')
plt.ylabel('Insurance Charges: as Dependent variable')
plt.title('Charge Vs BMI');
```

```
/opt/conda/lib/python3.6/site-packages/scipy/stats/stats.py:1713: FutureWarning:
Using a non-tuple sequence for multidimensional indexing is deprecated; use
`arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted
as an array index, `arr[np.array(seq)]`, which will result either in an error or
a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



In above plot we fit regression line into the variables.

1.4 Cost function

A cost function measures how much error in the model is in terms of ability to estimate the relationship between x and y. We can measure the accuracy of our hypothesis function by using a cost function. This takes an average difference of observed dependent variable in the given the dataset and those predicted by the hypothesis function.

$$\mathbf{J(\ )} = \frac{\mathbf{1}}{\mathbf{m}} \sum_{\mathbf{i=1}}^{\mathbf{m}} (\hat{\mathbf{y}}_\mathbf{i} - \mathbf{y}_\mathbf{i})^\mathbf{2}$$

$$\mathbf{J(\ )} = \frac{\mathbf{1}}{\mathbf{m}} \sum_{\mathbf{i=1}}^{\mathbf{m}} (\mathbf{h}\ (\mathbf{x}_\mathbf{i}) - \mathbf{y}_\mathbf{i})^\mathbf{2}$$

To implement the linear regression, take training example add an extra column that is x_0 feature, where $\mathbf{x_0} = \mathbf{1}$. $\mathbf{x_o} = (\ x_{i0}\ x_{i1}\ x_{i2} \cdots x_{mi}\ )$,where $\mathbf{x_{i0}} = \mathbf{0}$ and input matrix will become as

$$\mathbf{X} = \begin{pmatrix} x_{10} & x_{11} & x_{12} & \cdots & x_{1n} \\ x_{20} & x_{21} & x_{22} & \cdots & x_{2n} \\ x_{30} & x_{31} & x_{32} & \cdots & x_{3n} \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ x_{m0} & x_{m1} & x_{m2} & \cdots & x_{mn} \end{pmatrix}_{(m,n+1)}$$

Each of the m input samples is similarly a column vector with n+1 rows x_0 being 1 for our convenience, that is $\mathbf{x_{10}, x_{20}, x_{30}....x_{m0} = 1}$. Now we rewrite the ordinary least square cost function in matrix form as

$$\mathbf{J(\ )} = \frac{\mathbf{1}}{\mathbf{m}}(\mathbf{X\ -y})^{\mathbf{T}}(\mathbf{X\ -y})$$

Let's look at the matrix multiplication concept,the multiplication of two matrix happens only if number of column of firt matrix is equal to number of row of second matrix. Here input matrix $\mathbf{X}$ of size $(\mathbf{m, n+1})$, parameter of function is of size $(n+1, 1)$ and dependent variable vector of size $(\mathbf{m, 1})$. The product of matrix $\mathbf{X_{(m,n+1)\ (n+1,1)}}$ will return a vector of size $(\mathbf{m, 1})$, then product of $(\mathbf{X\ -y})^{\mathbf{T}}_{\mathbf{(1,m)}}(\mathbf{X\ -y})_{\mathbf{(m,1)}}$ will return size of unit vector.

1.5 Normal Equation

The normal equation is an analytical solution to the linear regression problem with a ordinary least square cost function. To minimize our cost function, take partial derivative of $\mathbf{J(\ )}$ with respect to θ and equate to 0. The derivative of function is nothing but if a small change in input what would be the change in output of function.

$$\mathbf{min}_{\mathbf{\ 0,\ 1\cdot\cdot\ n}}\mathbf{J(\ _{0},\ _{1\cdot\cdot\ n})}$$

$$\frac{\mathbf{J(\ _{j})}}{\mathbf{j}} = \mathbf{0}$$

where $\mathbf{j = 0, 1, 2, ...n}$

Now we will apply partial derivative of our cost function,

$$\frac{\mathbf{J(\ _{j})}}{\mathbf{j}} = -\frac{\mathbf{1}}{\mathbf{m}}(\mathbf{X\ -y})^{\mathbf{T}}(\mathbf{X\ -y})$$

I will throw $\frac{\mathbf{1}}{\mathbf{m}}$ part away since we are going to compare a derivative to 0. And solve $\mathbf{J(\ )}$,

$$\begin{aligned}\mathbf{J(\ )} &= (\mathbf{X\ -y})^{\mathbf{T}}(\mathbf{X\ -y}) \\ &= (\mathbf{X\ })^{\mathbf{T}} - \mathbf{y^T})(\mathbf{X\ -y})\end{aligned}$$

$$\begin{aligned}&= (\ ^{\mathbf{T}}\mathbf{X^T} - \mathbf{y^T})(\mathbf{X\ -y}) \\ &= \ ^{\mathbf{T}}\mathbf{X^TX\ - y^TX\ -\ ^{T}X^Ty + y^Ty} \\ &= \ ^{\mathbf{T}}\mathbf{X^TX\ - 2\ ^{T}X^Ty + y^Ty}\end{aligned}$$

Here $\mathbf{y^T_{(1,m)}X_{(m,n+1)\ (n+1,1)} = \ ^{T}_{(1,n+1)}X^T_{(n+1,m)}y_{(m,1)}}$ because unit vector.

$$\begin{aligned}\frac{\mathbf{J(\ )}}{} &= -(\ ^{\mathbf{T}}\mathbf{X^TX\ - 2\ ^{T}X^Ty + y^Ty}) \\ &= \mathbf{X^TX}\frac{^{\mathbf{T}}}{} - \mathbf{2X^Ty}\frac{^{\mathbf{T}}}{} + \frac{\mathbf{y^Ty}}{}\end{aligned}$$

Partial derivative $\frac{x^2}{x} = 2x$, $\frac{kx^2}{x} = kx$, $\frac{Constact}{x} = 0$

$$\frac{J(\,)}{} = X^T X2 \; - 2X^T y + 0$$

$$0 = 2X^T X \; - 2X^T y$$

$$X^T X \; = X^T$$

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

this the normal equation for linear regression

1.6 Exploratory data analysis

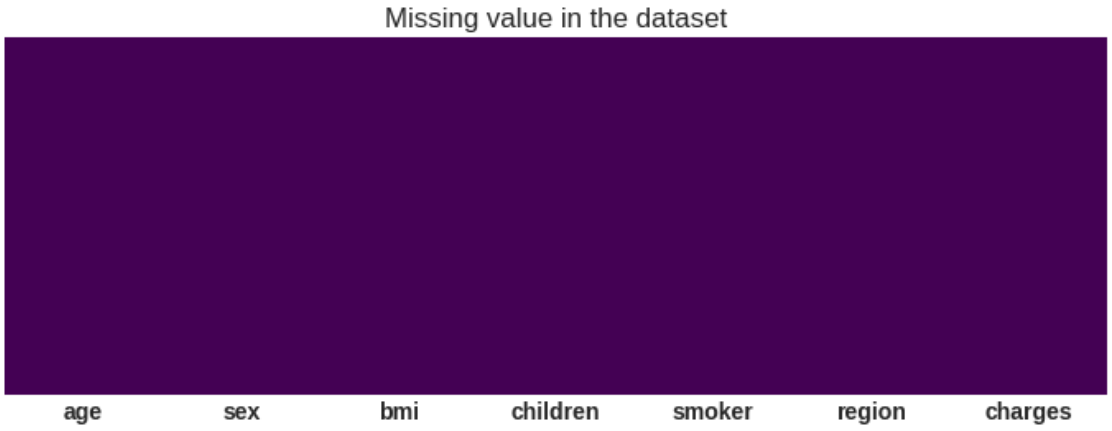
[10]:
```
df.describe()
```

[10]:

	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

1.6.1 Check for missing value

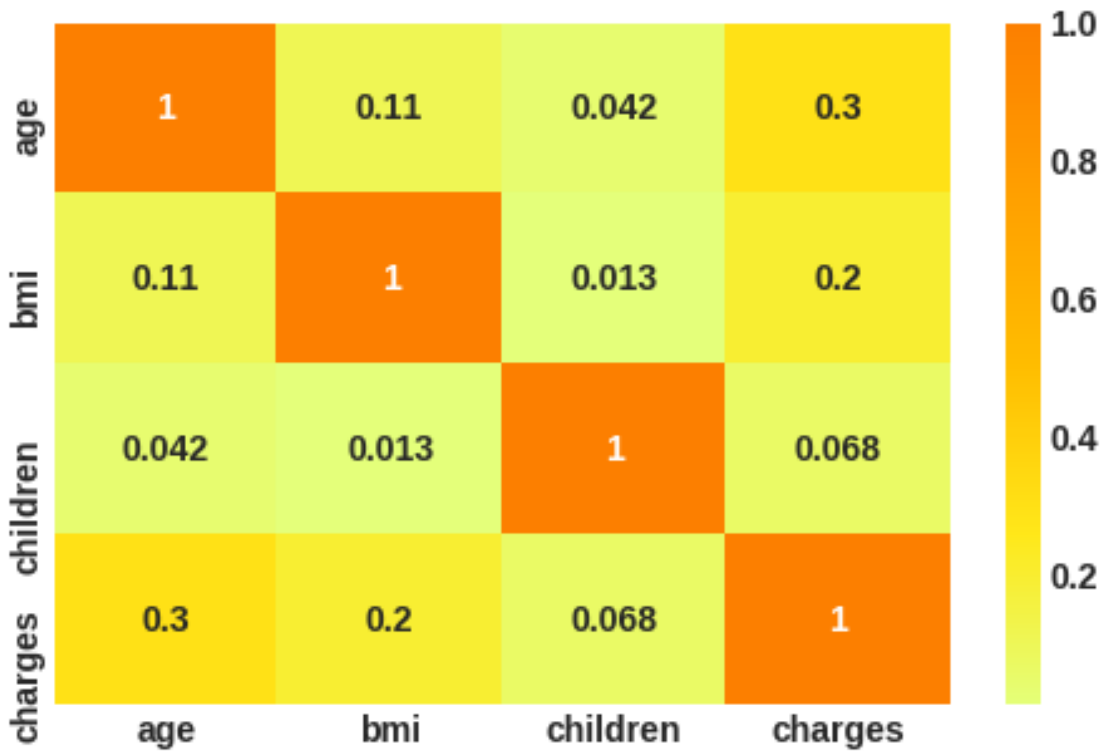
[11]:
```
plt.figure(figsize=(12,4))
sns.heatmap(df.isnull(),cbar=False,cmap='viridis',yticklabels=False)
plt.title('Missing value in the dataset');
```



There is no missing value in the data sex

1.6.2 Plots

[12]:
```
# correlation plot
corr = df.corr()
sns.heatmap(corr, cmap = 'Wistia', annot= True);
```

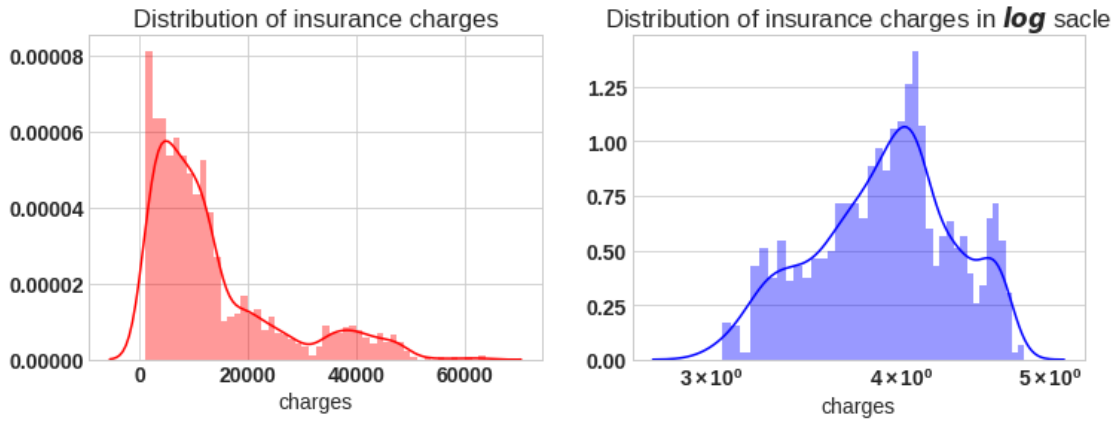


Thier no correlation among valiables.

[13]:
```
f= plt.figure(figsize=(12,4))

ax=f.add_subplot(121)
sns.distplot(df['charges'],bins=50,color='r',ax=ax)
ax.set_title('Distribution of insurance charges')

ax=f.add_subplot(122)
sns.distplot(np.log10(df['charges']),bins=40,color='b',ax=ax)
ax.set_title('Distribution of insurance charges in $log$ sacle')
ax.set_xscale('log');
```

```
/opt/conda/lib/python3.6/site-packages/scipy/stats/stats.py:1713: FutureWarning:
Using a non-tuple sequence for multidimensional indexing is deprecated; use
`arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted
as an array index, `arr[np.array(seq)]`, which will result either in an error or
a different result.
```

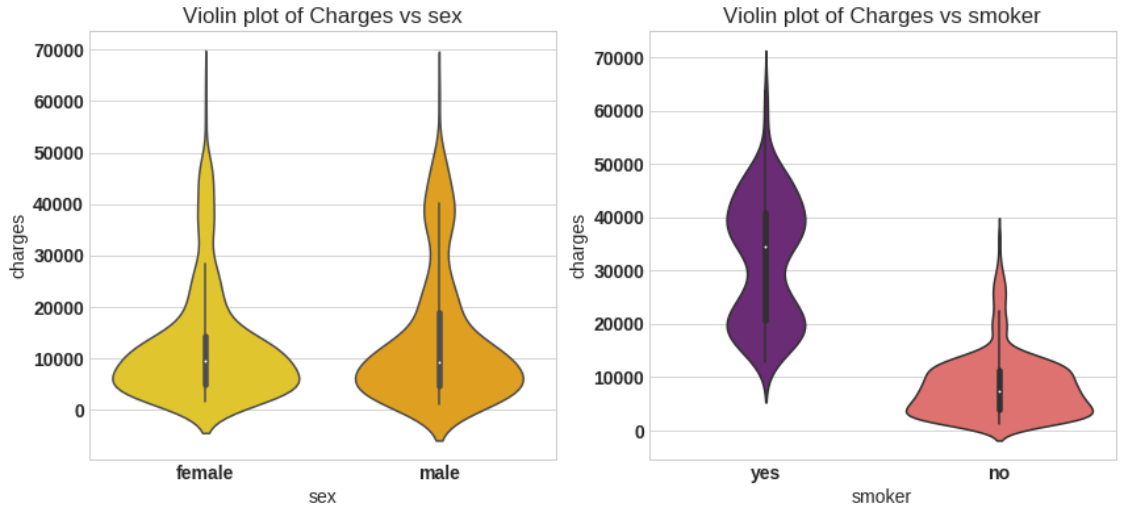
```
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



If we look at the left plot the charges varies from 1120 to 63500, the plot is right skewed. In right plot we will apply natural log, then plot approximately tends to normal. for further analysis we will apply log on target variable charges.

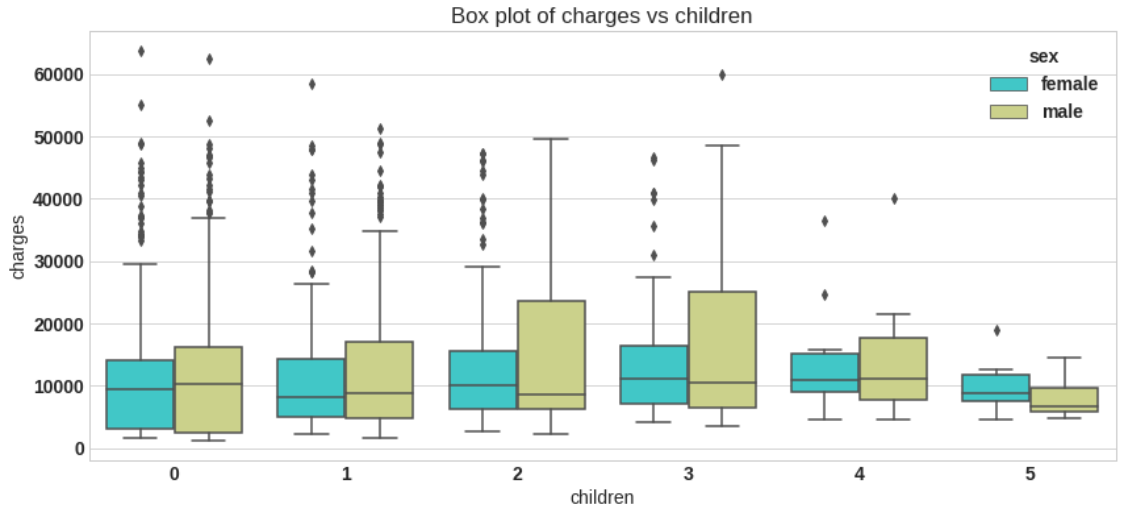
[14]:
```
f = plt.figure(figsize=(14,6))
ax = f.add_subplot(121)
sns.violinplot(x='sex', y='charges',data=df,palette='Wistia',ax=ax)
ax.set_title('Violin plot of Charges vs sex')

ax = f.add_subplot(122)
sns.violinplot(x='smoker', y='charges',data=df,palette='magma',ax=ax)
ax.set_title('Violin plot of Charges vs smoker');
```

```
/opt/conda/lib/python3.6/site-packages/scipy/stats/stats.py:1713: FutureWarning:
Using a non-tuple sequence for multidimensional indexing is deprecated; use
`arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted
as an array index, `arr[np.array(seq)]`, which will result either in an error or
a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



From left plot the insurance charge for male and female is approximatley in same range,it is average around 5000 bucks. In right plot the insurance charge for smokers is much wide range compare to non smokers, the average charges for non smoker is approximately 5000 bucks. For smoker the minimum insurance charge is itself 5000 bucks.

[15]:
```
plt.figure(figsize=(14,6))
sns.boxplot(x='children', y='charges',hue='sex',data=df,palette='rainbow')
plt.title('Box plot of charges vs children');
```



[16]:
```
df.groupby('children').agg(['mean','min','max'])['charges']
```

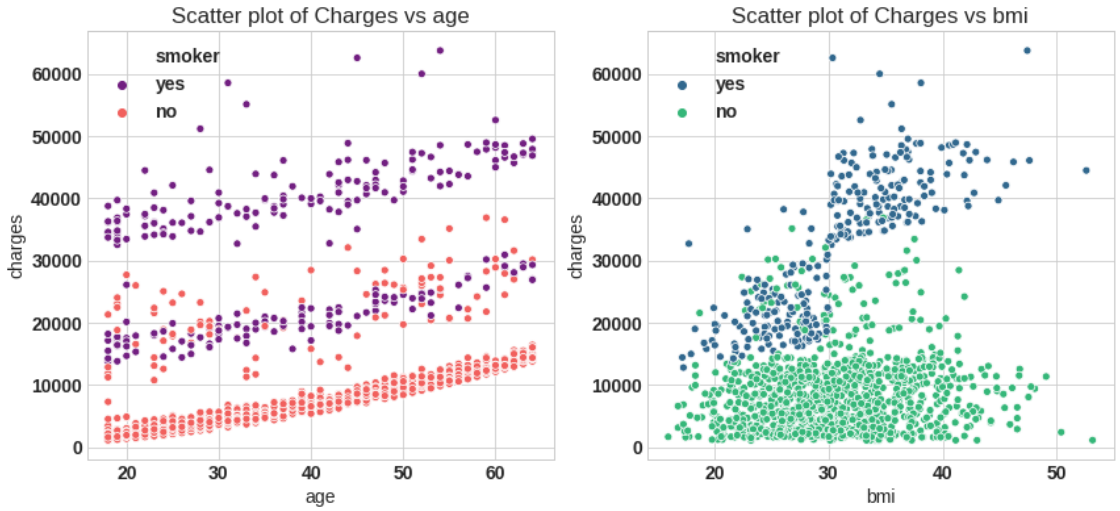
[16]:

children	mean	min	max
0	12365.975602	1121.8739	63770.42801
1	12731.171832	1711.0268	58571.07448
2	15073.563734	2304.0022	49577.66240
3	15355.318367	3443.0640	60021.39897
4	13850.656311	4504.6624	40182.24600
5	8786.035247	4687.7970	19023.26000

[17]:
```
plt.figure(figsize=(14,6))
sns.violinplot(x='region',␣
 ↪y='charges',hue='sex',data=df,palette='rainbow',split=True)
plt.title('Violin plot of charges vs children');
```

```
/opt/conda/lib/python3.6/site-packages/scipy/stats/stats.py:1713: FutureWarning:
Using a non-tuple sequence for multidimensional indexing is deprecated; use
`arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted
as an array index, `arr[np.array(seq)]`, which will result either in an error or
a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

[18]:
```
f = plt.figure(figsize=(14,6))
ax = f.add_subplot(121)
sns.scatterplot(x='age',y='charges',data=df,palette='magma',hue='smoker',ax=ax)
ax.set_title('Scatter plot of Charges vs age')

ax = f.add_subplot(122)
sns.scatterplot(x='bmi',y='charges',data=df,palette='viridis',hue='smoker')
ax.set_title('Scatter plot of Charges vs bmi')
plt.savefig('sc.png');
```



From left plot the minimum age person is insured is 18 year. There is slabs in policy most of non smoker take 1^{st} and 2^{nd} slab, for smoker policy start at 2^{nd} and 3^{rd} slab.

Body mass index (BMI) is a measure of body fat based on height and weight that applies to adult men and women. The minimum bmi is $16kg/m^2$ and maximum upto $54kg/m^2$

1.7 Data Preprocessing

1.7.1 Encoding

Machine learning algorithms cannot work with categorical data directly, categorical data must be converted to number. 1. Label Encoding 2. One hot encoding 3. Dummy variable trap

Label encoding refers to transforming the word labels into numerical form so that the algorithms can understand how to operate on them.

A **One hot encoding** is a representation of categorical variable as binary vectors.It allows the representation of categorical data to be more expresive. This first requires that the categorical values be mapped to integer values, that is label encoding. Then, each integer value is represented as a binary vector that is all zero values except the index of the integer, which is marked with a 1.

The **Dummy variable trap** is a scenario in which the independent variable are multicollinear, a scenario in which two or more variables are highly correlated in simple term one variable can be predicted from the others.

By using *pandas get_dummies* function we can do all above three step in line of code. We will this fuction to get dummy variable for sex, children,smoker,region features. By setting *drop_first =True* function will remove dummy variable trap by droping one variable and original variable.The pandas makes our life easy.

[19]:
```
# Dummy variable
categorical_columns = ['sex','children', 'smoker', 'region']
```

```
df_encode = pd.get_dummies(data = df, prefix = 'OHE', prefix_sep='_',
                columns = categorical_columns,
                drop_first =True,
               dtype='int8')
```

[20]:
```
# Lets verify the dummay variable process
print('Columns in original data frame:\n',df.columns.values)
print('\nNumber of rows and columns in the dataset:',df.shape)
print('\nColumns in data frame after encoding dummy variable:\n',df_encode.
  ↪columns.values)
print('\nNumber of rows and columns in the dataset:',df_encode.shape)
```

```
Columns in original data frame:
 ['age' 'sex' 'bmi' 'children' 'smoker' 'region' 'charges']

Number of rows and columns in the dataset: (1338, 7)

Columns in data frame after encoding dummy variable:
 ['age' 'bmi' 'charges' 'OHE_male' 'OHE_1' 'OHE_2' 'OHE_3' 'OHE_4' 'OHE_5'
 'OHE_yes' 'OHE_northwest' 'OHE_southeast' 'OHE_southwest']

Number of rows and columns in the dataset: (1338, 13)
```

1.7.2 Box -Cox transformation

A Box Cox transformation is a way to transform non-normal dependent variables into a normal shape. Normality is an important assumption for many statistical techniques; if your data isn't normal, applying a Box-Cox means that you are able to run a broader number of tests. All that we need to perform this transformation is to find lambda value and apply the rule shown below to your variable.

$$\begin{cases} \frac{y^{\lambda}-1}{\lambda}, & y_i \neg = 0 \\ \log(y_i) & \lambda = 0 \end{cases}$$

The trick of Box-Cox transformation is to find lambda value, however in practice this is quite affordable. The following function returns the transformed variable, lambda value,confidence interval

[21]:
```
from scipy.stats import boxcox
y_bc,lam, ci= boxcox(df_encode['charges'],alpha=0.05)

#df['charges'] = y_bc
# it did not perform better for this model, so log transform is used
ci,lam
```

[21]:
```
((-0.01140290617294196, 0.0988096859767545), 0.043649053770664956)
```

[22]:
```
## Log transform
df_encode['charges'] = np.log(df_encode['charges'])
```

The original categorical variable are remove and also one of the one hot encode varible column for perticular categorical variable is droped from the column. So we completed all three encoding step by using get dummies function.

1.8 Train Test split

[23]:
```
from sklearn.model_selection import train_test_split
X = df_encode.drop('charges',axis=1) # Independet variable
y = df_encode['charges'] # dependent variable

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.
 ↪3,random_state=23)
```

1.9 Model building

In this step build model using our linear regression equation $= (\mathbf{X^T X})^{-1}\mathbf{X^T y}$. In first step we need to add a feature $\mathbf{x_0 = 1}$ to our original data set.

[24]:
```
# Step 1: add x0 =1 to dataset
X_train_0 = np.c_[np.ones((X_train.shape[0],1)),X_train]
X_test_0 = np.c_[np.ones((X_test.shape[0],1)),X_test]

# Step2: build model
theta = np.matmul(np.linalg.inv( np.matmul(X_train_0.T,X_train_0) ), np.
 ↪matmul(X_train_0.T,y_train))
```

[25]:
```
# The parameters for linear regression model
parameter = ['theta_'+str(i) for i in range(X_train_0.shape[1])]
columns = ['intersect:x_0=1'] + list(X.columns.values)
parameter_df = pd.DataFrame({'Parameter':parameter,'Columns':columns,'theta':
 ↪theta})
```

[26]:
```
# Scikit Learn module
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X_train,y_train) # Note: x_0 =1 is no need to add, sklearn will␣
 ↪take care of it.

#Parameter
sk_theta = [lin_reg.intercept_]+list(lin_reg.coef_)
parameter_df = parameter_df.join(pd.Series(sk_theta, name='Sklearn_theta'))
parameter_df
```

[26]:

	Parameter	Columns	theta	Sklearn_theta
0	theta_0	intersect:x_0=1	7.059171	7.059171
1	theta_1	age	0.033134	0.033134
2	theta_2	bmi	0.013517	0.013517

3	theta_3	OHE_male	-0.067767	-0.067767
4	theta_4	OHE_1	0.149457	0.149457
5	theta_5	OHE_2	0.272919	0.272919
6	theta_6	OHE_3	0.244095	0.244095
7	theta_7	OHE_4	0.523339	0.523339
8	theta_8	OHE_5	0.466030	0.466030
9	theta_9	OHE_yes	1.550481	1.550481
10	theta_10	OHE_northwest	-0.055845	-0.055845
11	theta_11	OHE_southeast	-0.146578	-0.146578
12	theta_12	OHE_southwest	-0.133508	-0.133508

The parameter obtained from both the model are same.So we succefull build our model using normal equation and verified using sklearn linear regression module. Let's move ahead, next step is prediction and model evaluvation.

1.10 Model evaluation

We will predict value for target variable by using our model parameter for test data set. Then compare the predicted value with actual valu in test set. We compute **Mean Square Error** using formula

$$\mathbf{J(\ ) = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2}$$

$\mathbf{R^2}$ is statistical measure of how close data are to the fitted regression line. $\mathbf{R^2}$ is always between 0 to 100%. 0% indicated that model explains none of the variability of the response data around it's mean. 100% indicated that model explains all the variablity of the response data around the mean.

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

SSE = Sum of Square Error
SST = Sum of Square Total

$$\mathbf{SSE = \sum_{i=1}^{m}(\hat{y}_i - y_i)^2}$$

$$\mathbf{SST = \sum_{i=1}^{m}(y_i - \bar{y}_i)^2}$$

Here $\hat{\mathbf{y}}$ is predicted value and $\bar{\mathbf{y}}$ is mean value of $\mathbf{y}$.

[27]:
```
# Normal equation
y_pred_norm =  np.matmul(X_test_0,theta)

#Evaluvation: MSE
J_mse = np.sum((y_pred_norm - y_test)**2)/ X_test_0.shape[0]

# R_square
sse = np.sum((y_pred_norm - y_test)**2)
```

```
sst = np.sum((y_test - y_test.mean())**2)
R_square = 1 - (sse/sst)
print('The Mean Square Error(MSE) or J(theta) is: ',J_mse)
print('R square obtain for normal equation method is :',R_square)
```

```
The Mean Square Error(MSE) or J(theta) is:  0.18729622322982067
R square obtain for normal equation method is : 0.7795687545055299
```

[28]:
```
# sklearn regression module
y_pred_sk = lin_reg.predict(X_test)

#Evaluvation: MSE
from sklearn.metrics import mean_squared_error
J_mse_sk = mean_squared_error(y_pred_sk, y_test)

# R_square
R_square_sk = lin_reg.score(X_test,y_test)
print('The Mean Square Error(MSE) or J(theta) is: ',J_mse_sk)
print('R square obtain for scikit learn library is :',R_square_sk)
```

```
The Mean Square Error(MSE) or J(theta) is:  0.1872962232298189
R square obtain for scikit learn library is : 0.7795687545055318
```

The model returns R^2 value of 77.95%, so it fit our data test very well, but still we can imporve the the performance of by diffirent technique. Please make a note that we have transformer out variable by applying natural log. When we put model into production antilog is applied to the equation.

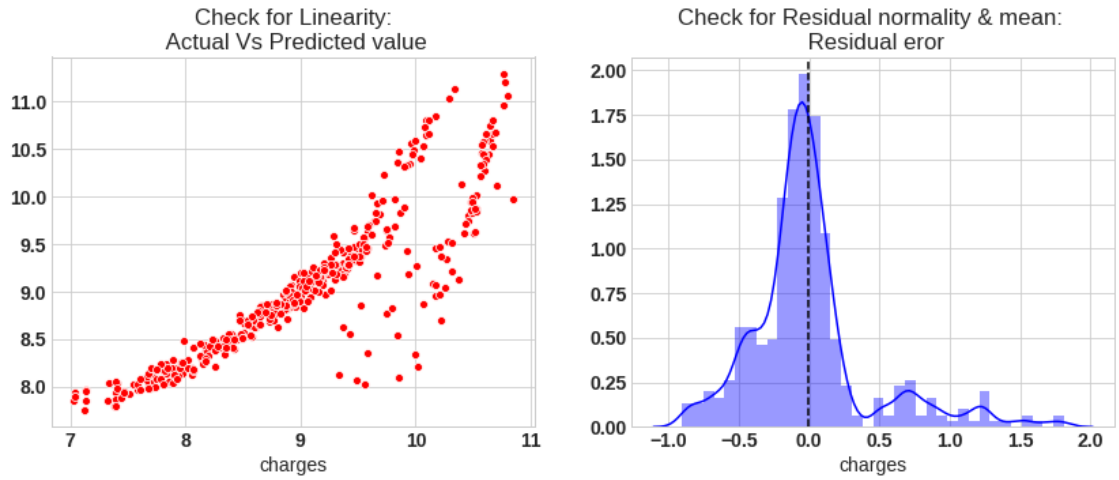
1.11 Model Validation

In order to validated model we need to check few assumption of linear regression model. The common assumption for *Linear Regression* model are following 1. Linear Relationship: In linear regression the relationship between the dependent and independent variable to be *linear*. This can be checked by scatter ploting Actual value Vs Predicted value 2. The residual error plot should be *normally* distributed. 3. The *mean* of *residual error* should be 0 or close to 0 as much as possible 4. The linear regression require all variables to be multivariate normal. This assumption can best checked with Q-Q plot. 5. Linear regession assumes that there is little or no *Multicollinearity in the data. Multicollinearity occurs when the independent variables are too highly correlated with each other. The variance inflation factor* VIF* identifies correlation between independent variables and strength of that correlation. $\mathbf{VIF} = \frac{1}{1-\mathbf{R^2}}$, If VIF >1 & VIF <5 moderate correlation, VIF < 5 critical level of multicollinearity. 6. Homoscedasticity: The data are homoscedastic meaning the residuals are equal across the regression line. We can look at residual Vs fitted value scatter plot. If heteroscedastic plot would exhibit a funnel shape pattern.

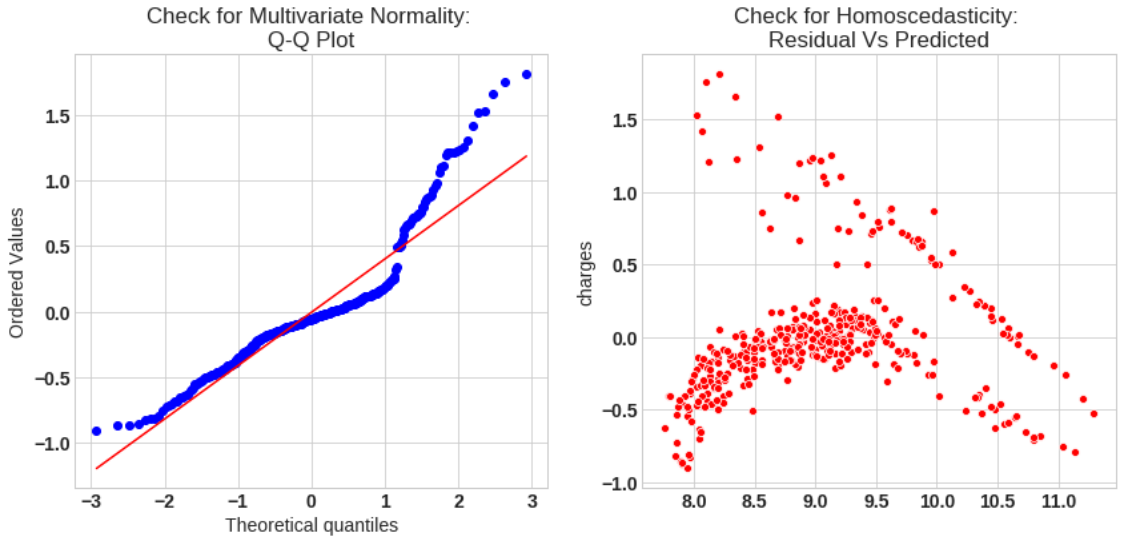
[29]:
```
# Check for Linearity
f = plt.figure(figsize=(14,5))
ax = f.add_subplot(121)
sns.scatterplot(y_test,y_pred_sk,ax=ax,color='r')
```

```
ax.set_title('Check for Linearity:\n Actual Vs Predicted value')

# Check for Residual normality & mean
ax = f.add_subplot(122)
sns.distplot((y_test - y_pred_sk),ax=ax,color='b')
ax.axvline((y_test - y_pred_sk).mean(),color='k',linestyle='--')
ax.set_title('Check for Residual normality & mean: \n Residual eror');
```

```
/opt/conda/lib/python3.6/site-packages/scipy/stats/stats.py:1713: FutureWarning:
Using a non-tuple sequence for multidimensional indexing is deprecated; use
`arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted
as an array index, `arr[np.array(seq)]`, which will result either in an error or
a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



[30]:
```
# Check for Multivariate Normality
# Quantile-Quantile plot
f,ax = plt.subplots(1,2,figsize=(14,6))
import scipy as sp
_,(_,_,r)= sp.stats.probplot((y_test - y_pred_sk),fit=True,plot=ax[0])
ax[0].set_title('Check for Multivariate Normality: \nQ-Q Plot')

#Check for Homoscedasticity
sns.scatterplot(y = (y_test - y_pred_sk), x= y_pred_sk, ax = ax[1],color='r')
ax[1].set_title('Check for Homoscedasticity: \nResidual Vs Predicted');
```



[31]:
```
# Check for Multicollinearity
#Variance Inflation Factor
VIF = 1/(1- R_square_sk)
VIF
```

[31]: 4.536561945911135

The model assumption linear regression as follows 1. In our model the actual vs predicted plot is curve so linear assumption fails 2. The residual mean is zero and residual error plot right skewed 3. Q-Q plot shows as value log value greater than 1.5 trends to increase 4. The plot is exhibit heteroscedastic, error will insease after certian point. 5. Variance inflation factor value is less than 5, so no multicollearity.

1.What do you mean by Linear Regression ?

Ans: Linear regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables. The aim of linear regression is to find the linear equation that best describes the relationship between the variables. The equation takes the form of y = mx + b, where y is the dependent variable, x is the independent variable, m is the slope of the line, and b is the y-intercept.

Linear regression is used to predict the value of the dependent variable (y) based on the value of the independent variable (x) by estimating the values of the slope and intercept. It assumes a linear relationship between the variables, meaning that a change in the independent variable will result in a proportional change in the dependent variable.

Linear regression is a widely used statistical tool in various fields such as finance, economics, biology, engineering, and social sciences to analyze and model data.

2.Why is linear regression called as supervised learning?

Ans: Linear regression is called supervised learning because it involves learning from a labeled

dataset, where the values of both the dependent variable (y) and the independent variable (x) are known. In supervised learning, the model is trained on a labeled dataset to learn the relationship between the input variables (independent variables) and the output variable (dependent variable).

In linear regression, the model learns from the training dataset to find the line of best fit that minimizes the difference between the predicted values and the actual values of the dependent variable. The model is then used to predict the value of the dependent variable for new input values of the independent variable.

Supervised learning is one of the two main types of machine learning, where the model is trained on labeled data. The other type is unsupervised learning, where the model learns to identify patterns and relationships in the data without the use of labeled data.

3.If the input data has some feature which is nominal can that column be used for Solving regression, why?

Ans: Nominal features cannot be directly used in a regression model because they represent categorical data that cannot be ordered or quantified.

However, nominal features can be transformed into numerical features using a process called encoding. There are different types of encoding techniques such as one-hot encoding, ordinal encoding, and binary encoding.

One-hot encoding is a technique that creates a binary vector representation of each category in the nominal feature. For example, if a nominal feature has three categories, it will be transformed into three binary features, where each binary feature represents a category. One-hot encoding allows the nominal feature to be included in a regression model as numerical features.

Ordinal encoding is another technique that assigns a numerical value to each category based on their order. For example, if a nominal feature has three categories, it can be assigned numerical values of 1, 2, and 3. However, ordinal encoding may not be appropriate if the categories in the nominal feature do not have an inherent order.

Binary encoding is a technique that creates binary features for each category in the nominal feature by comparing each category to the mean of the dependent variable.

Therefore, by transforming nominal features into numerical features using one of the encoding techniques mentioned above, they can be used in regression models.

4.What is role of sklearn package in tutorial?

Ans: The scikit-learn (sklearn) package is a widely used machine learning library in Python that provides a range of tools and algorithms for data analysis and machine learning tasks. In the tutorial, the sklearn package is used to build a linear regression model to predict the price of houses based on their characteristics.

The sklearn package provides a wide range of tools for data preprocessing, model selection, and model evaluation. In the tutorial, the sklearn package is used to perform the following tasks:

1. Data preprocessing: The sklearn package provides tools for data preprocessing such as scaling, normalization, and missing value imputation. In the tutorial, the StandardScaler class is used to scale the input features.
2. Model selection: The sklearn package provides tools for model selection such as cross-validation, hyperparameter tuning, and model evaluation metrics. In the tutorial, the

train_test_split function is used to split the dataset into training and testing sets, and the mean squared error (MSE) metric is used to evaluate the performance of the model.

3. Model building: The sklearn package provides a wide range of algorithms for building machine learning models. In the tutorial, the LinearRegression class is used to build a linear regression model.

4. Model evaluation: The sklearn package provides tools for evaluating the performance of a model using various metrics such as accuracy, precision, recall, and F1 score. In the tutorial, the MSE metric is used to evaluate the performance of the linear regression model.

Overall, the sklearn package is used in the tutorial to perform various tasks related to data preprocessing, model selection, model building, and model evaluation, making it a valuable tool for building and evaluating machine learning models.

5.What is violin plot? Explain what information is displayed in any one of the plots shown in notebook.

Ans: A violin plot is a type of data visualization that combines aspects of a box plot and a kernel density plot. It is used to display the distribution of a continuous variable across different categories or groups.

In a violin plot, each group or category is represented by a "violin," which is a mirrored density plot on each side of a box plot. The density plot shows the distribution of the data, while the box plot shows the quartiles of the data and any outliers.

One of the plots shown in the notebook may display the violin plot of the distribution of the target variable (house prices) across different categories of a categorical variable (e.g., the type of house). In this plot, each category of the categorical variable is represented by a separate violin, and the width of the violin indicates the density of data in that category. The thicker parts of the violin indicate that more data points are present in that range of the variable, and the thinner parts indicate that fewer data points are present in that range.

The box in the middle of each violin shows the quartiles of the data distribution, where the bottom of the box represents the 25th percentile, the top of the box represents the 75th percentile, and the line inside the box represents the median of the distribution. Any points outside the whiskers of the box plot are considered outliers and are shown as individual points.

Overall, the violin plot is a useful tool for visualizing the distribution of data across different categories, as it provides a comprehensive view of both the shape and density of the data distribution.