Linear Regression with Medical Cost Personal Datasets

Import Library and Dataset

In [52]:
```
#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')
```

In [53]:
```
# Import dataset
#path ='dataset/'

df = pd.read_csv('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)
```

Out[53]:

	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

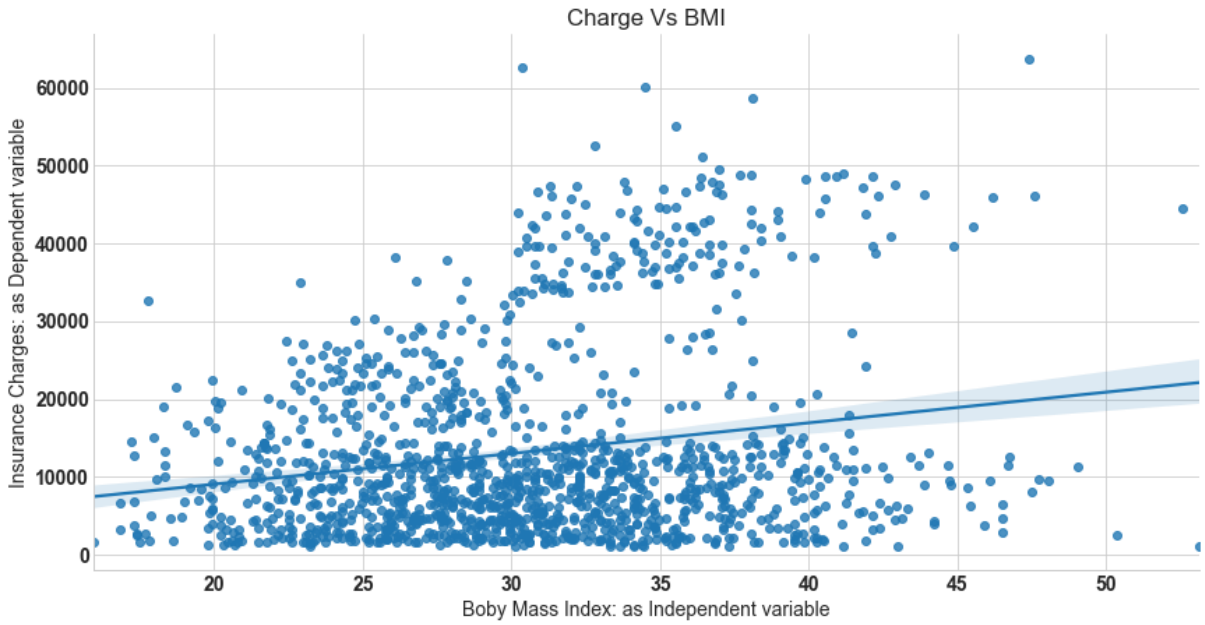
The shape of dataset is (1338,7).So there are $\mathbf{m}$=1338 training exaple and $\mathbf{n}$=7 independent variable.

We'll use multiple linear regression to fit our model since there are several independent variables (age, sex, BMI, children, smoking, and region), while the target variable is charges. The hypothesis function is represented as:

$$\mathbf{h}\theta(\mathbf{xi})=\theta 0+\theta 1\mathbf{age}+\theta 2\mathbf{sex}+\theta 3\mathbf{bmi}+\theta 4\mathbf{children}+\theta 5\mathbf{smoker}+\theta 6\mathbf{region}$$

Plot

In [54]:
```
sns.lmplot(x='bmi',y='charges',data=df,aspect=2,height=6)
plt.xlabel('Boby Mass Index: as Independent variable')
plt.ylabel('Insurance Charges: as Dependent variable')
plt.title('Charge Vs BMI');
```



Exploratory analysis

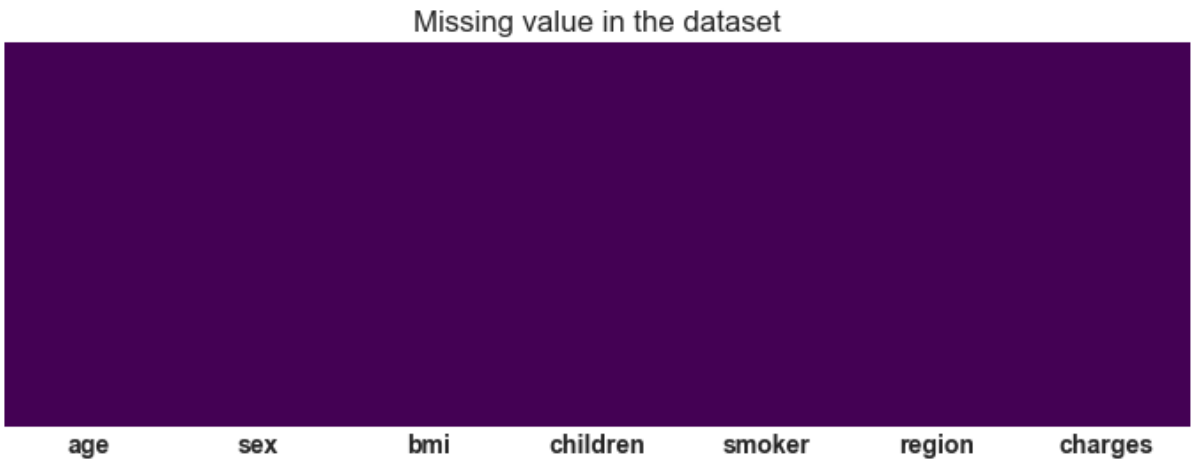
In [55]:
```
df.describe()
```

Out[55]:

	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

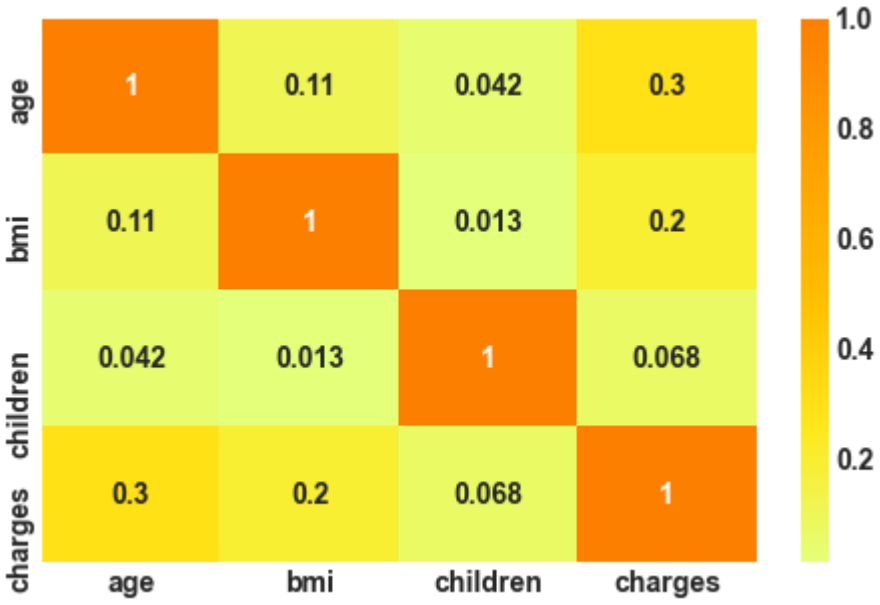
Check for missing value

In [56]:
```
plt.figure(figsize=(12,4))
sns.heatmap(df.isnull(),cbar=False,cmap='viridis',yticklabels=False)
plt.title('Missing value in the dataset');
```



No missing value in the data

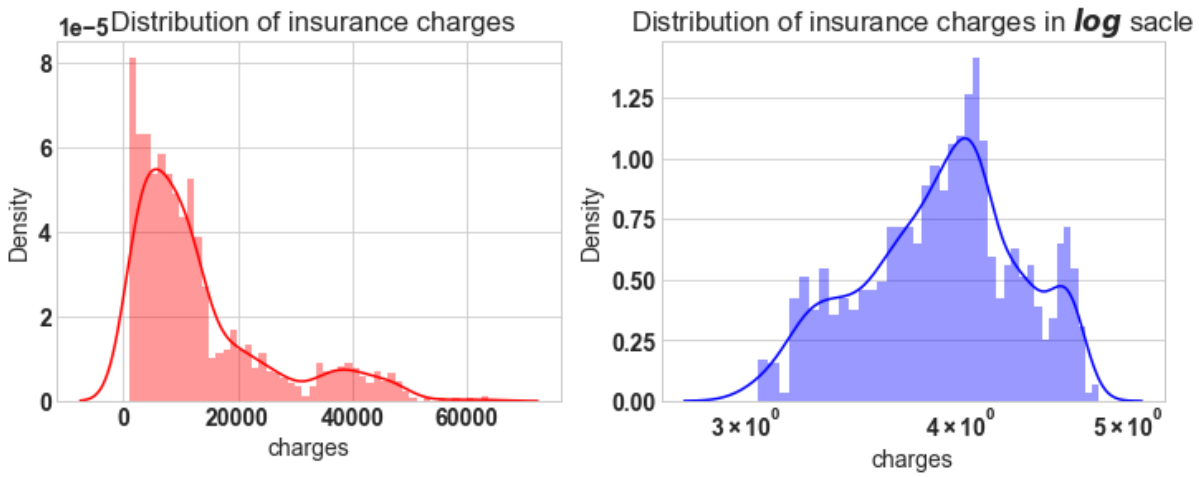
In [57]:
```
# correlation plot
corr = df.corr()
sns.heatmap(corr, cmap = 'Wistia', annot= True);
```



No correlation among valiables.

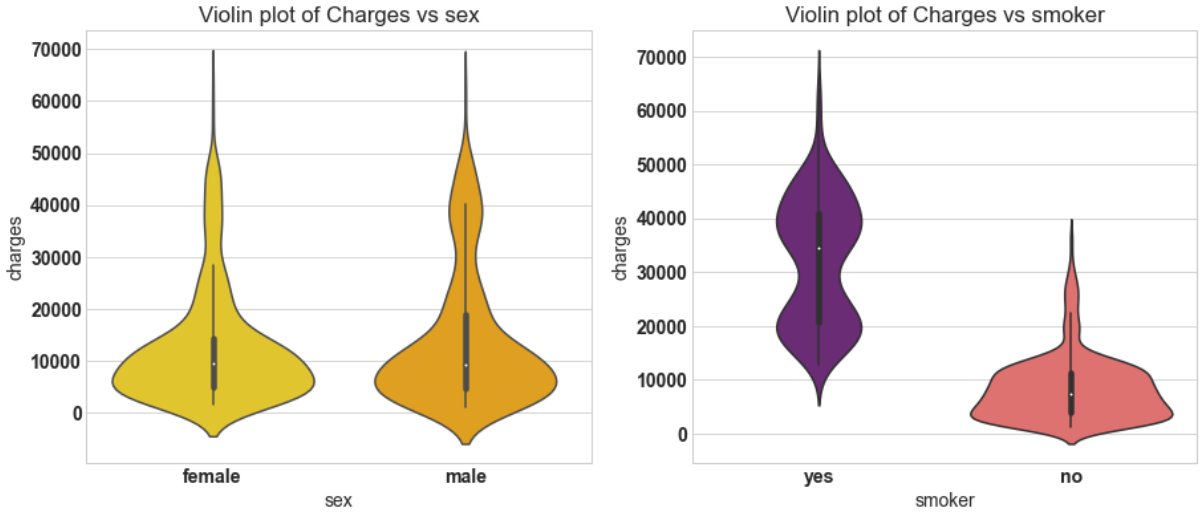
In [58]:
```
f= plt.figure(figsize=(12,4))

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');
```

```
/Users/autumnbrinkerhoff/opt/anaconda3/lib/python3.8/site-packages/seab
orn/distributions.py:2551: FutureWarning: `distplot` is a deprecated fu
nction and will be removed in a future version. Please adapt your code
to use either `displot` (a figure-level function with similar flexibili
ty) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/Users/autumnbrinkerhoff/opt/anaconda3/lib/python3.8/site-packages/seab
orn/distributions.py:2551: FutureWarning: `distplot` is a deprecated fu
nction and will be removed in a future version. Please adapt your code
to use either `displot` (a figure-level function with similar flexibili
ty) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



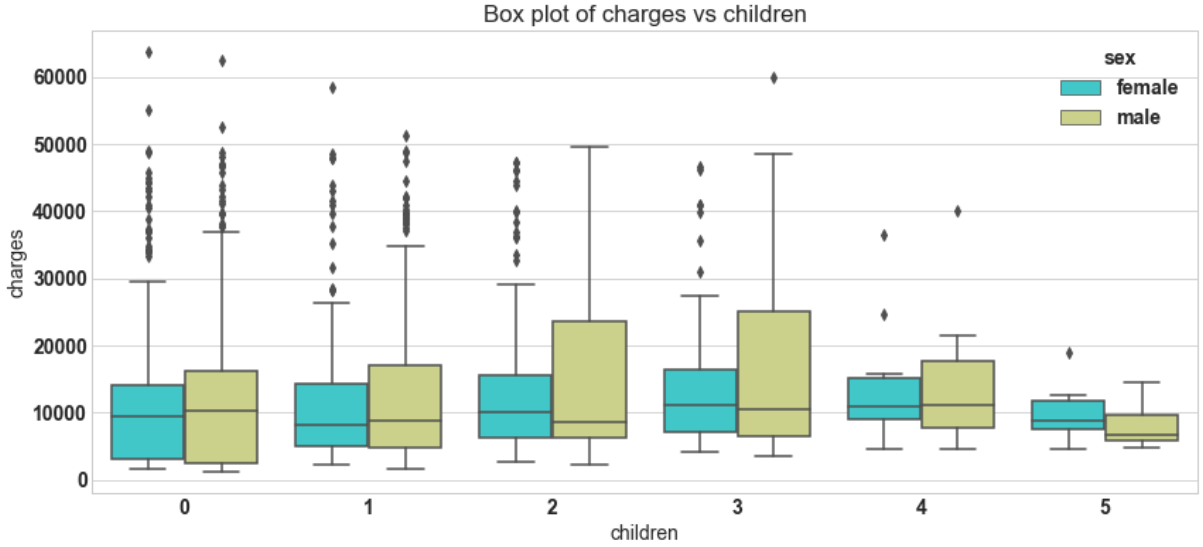
Left plot: Charges range from 1120 to 63500; right-skewed. Right plot: Natural log applied, normal distribution. Target variable charges to be logarithmically transformed for further analysis.

In [59]:
```
f = plt.figure(figsize=(15,6))
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');
```



The left plot shows that insurance charges are about the same for males and females, averaging around 5000 dollars. On the other hand, the right plot shows that smokers pay more for insurance than non-smokers. Smokers' insurance charges vary widely, averaging around 5000 dollars, with a minimum charge of 5000 dollars.

In [60]:
```
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');
```

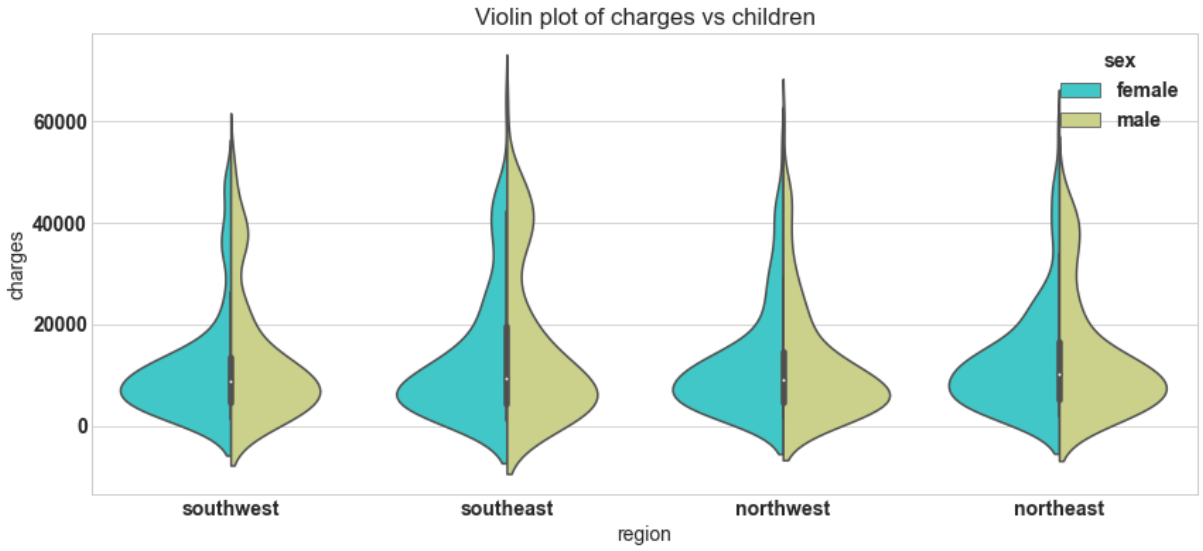


In [61]:
```
df.groupby('children').agg(['mean','min','max'])['charges']
```

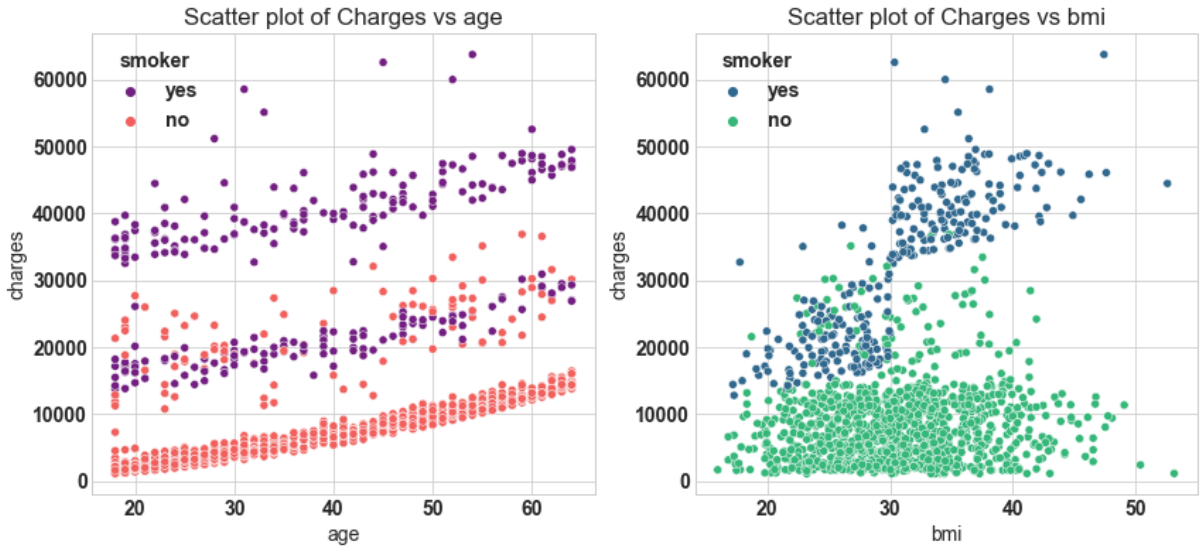
Out[61]:

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

In [62]:
```
plt.figure(figsize=(14,6))
sns.violinplot(x='region', y='charges',hue='sex',data=df,palette='rainbo
plt.title('Violin plot of charges vs children');
```



In [63]:
```
f = plt.figure(figsize=(14,6))
ax = f.add_subplot(121)
sns.scatterplot(x='age',y='charges',data=df,palette='magma',hue='smoker'
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='smoke
ax.set_title('Scatter plot of Charges vs bmi')
plt.savefig('sc.png');
```



The policy requires a minimum age of 18. Different coverage levels: Non-smokers choose 1st/2nd level, and smokers choose 2nd/3rd level. BMI measures body fat based on height and weight, ranging from 16 to 54.

Dummy variables

In [64]:
```
# 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')
```

In [65]:
```
# 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_enco
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)
```

In [66]:
```
## Log transform
df_encode['charges'] = np.log(df_encode['charges'])
```

Train and Test data

In [67]:
```
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,ra
```

Model building

In [68]:
```
# 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.
```

In [69]:
```
# 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,'th
```

In [70]:
```
# Scikit Learn module
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X_train,y_train)

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

Out[70]:

	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

Model evaluation

In [71]:
```
# 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): ',J_mse_sk)
print('R square:',R_square_sk)
```

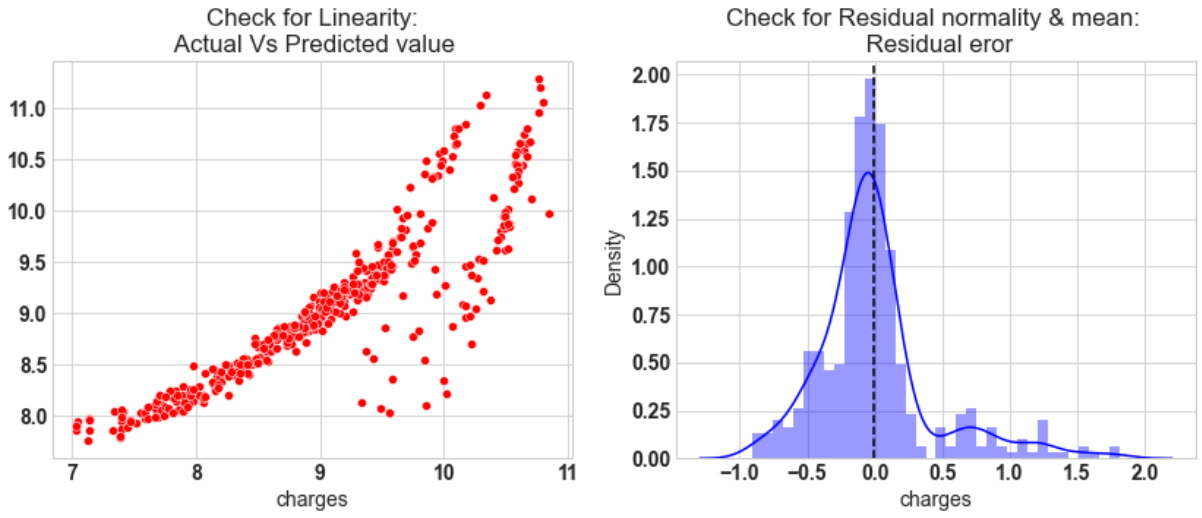
```
The Mean Square Error(MSE):  0.18729622322981904
R square: 0.7795687545055316
```

The model returns R2 value of 77.95%

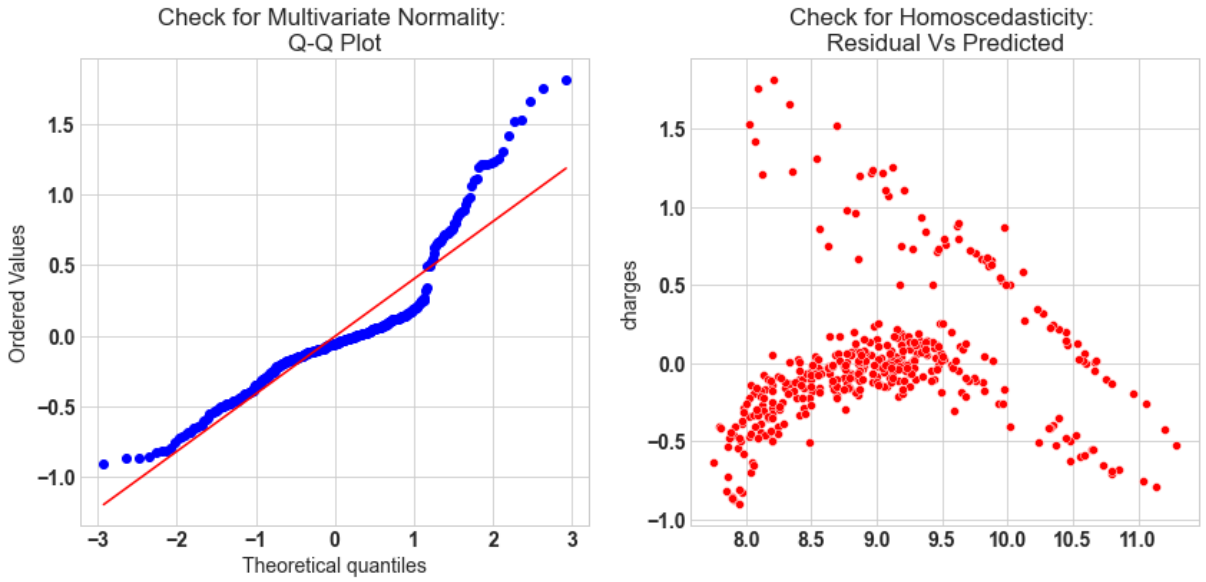
Model Validation

In [72]:
```
# 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');
```

```
/Users/autumnbrinkerhoff/opt/anaconda3/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/Users/autumnbrinkerhoff/opt/anaconda3/lib/python3.8/site-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [73]:
```
# 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=
ax[1].set_title('Check for Homoscedasticity: \nResidual Vs Predicted');
```



In [74]:
```
# Check for Multicollinearity
#Variance Inflation Factor
VIF = 1/(1- R_square_sk)
VIF
```

Out[74]: 4.536561945911133