In [1]:
```
import pandas  as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
plt.rcParams['figure.figsize'] = [8,5]
plt.rcParams['font.size'] =14
plt.rcParams['font.weight']= 'bold'
plt.style.use('seaborn-whitegrid')
```

```
/var/folders/20/prf96pn926n0x2yzn_v2yhbr0000gn/T/ipykernel_17464/356
1387198.py:8: MatplotlibDeprecationWarning: The seaborn styles shipp
ed by Matplotlib are deprecated since 3.6, as they no longer corresp
ond to the styles shipped by seaborn. However, they will remain avai
lable as 'seaborn-v0_8-<style>'. Alternatively, directly use the sea
born API instead.
  plt.style.use('seaborn-whitegrid')
```

In [2]:
```
df = pd.read_csv("/Users/akheruddinahmed/ML/insurance.csv")
print('\nNumber of rows and columns in the data set: ',df.shape)
print('')

df.head()
```

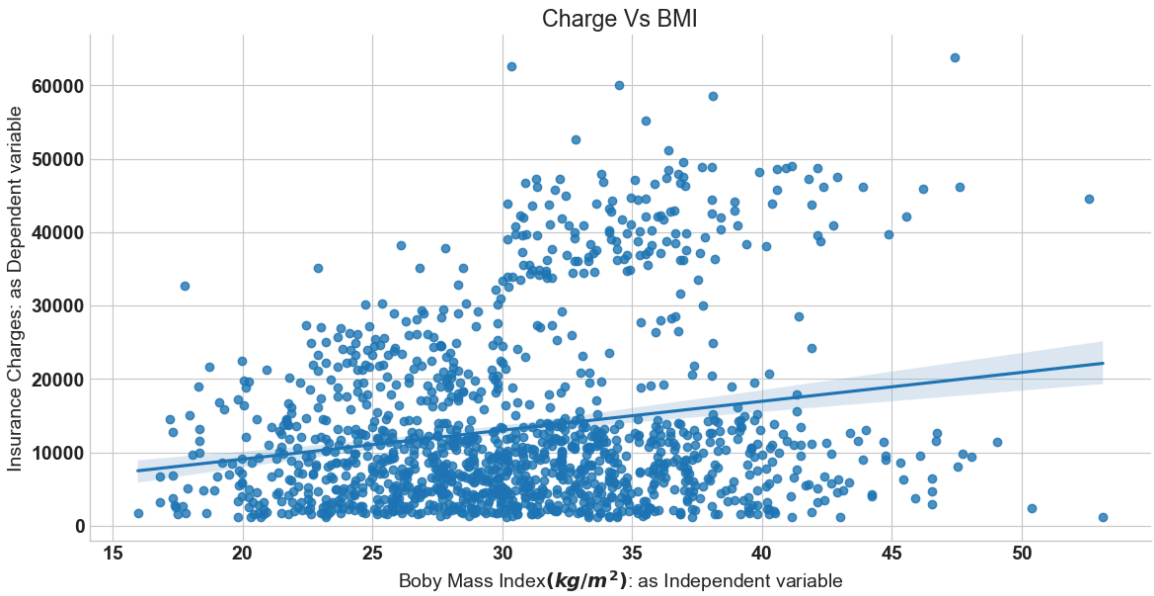
```
Number of rows and columns in the data set:  (1338, 7)
```

Out[2]:

	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

In [3]:
```
""" for our visualization purpose will fit line using seaborn library
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');
```

```
/Users/akheruddinahmed/anaconda3/lib/python3.11/site-packages/seabor
n/axisgrid.py:118: UserWarning: The figure layout has changed to tig
ht
  self._figure.tight_layout(*args, **kwargs)
```

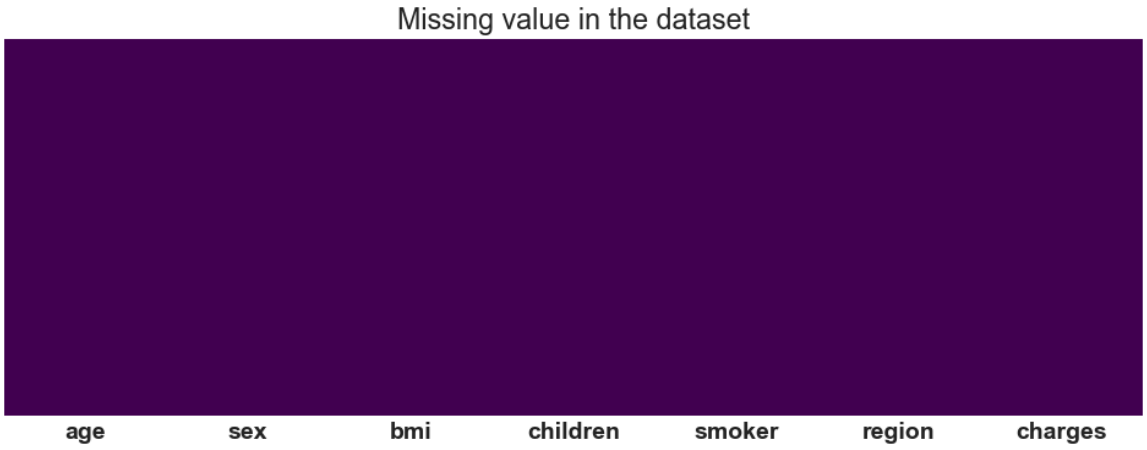


In [4]:
```
df.describe()
```

Out[4]:

	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

In [5]:
```
plt.figure(figsize=(12,4))
sns.heatmap(df.isnull(),cbar=False,cmap='viridis',yticklabels=False)
plt.title('Missing value in the dataset');
```

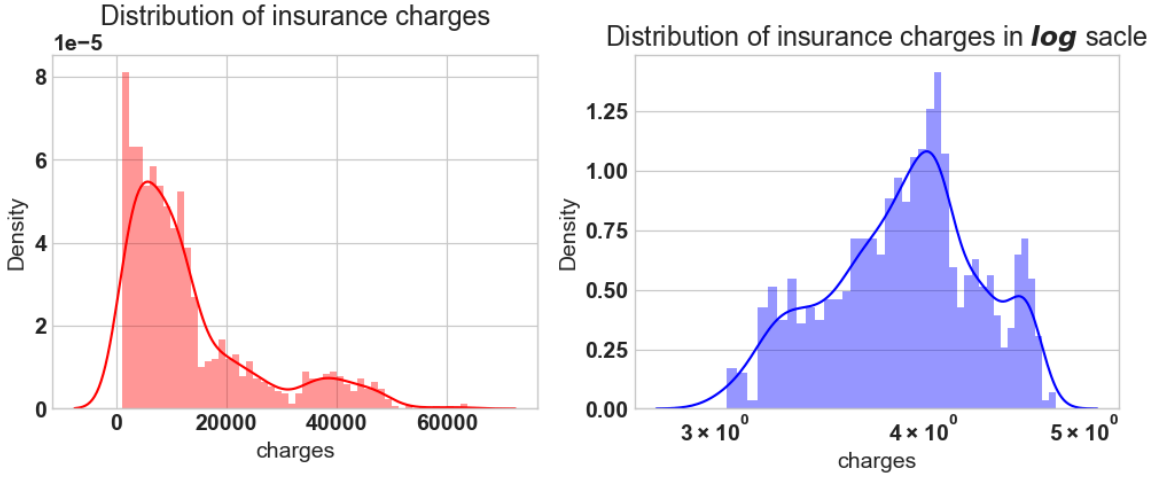


In [6]:
```
# correlation plot
corr = df.corr()
sns.heatmap(corr, cmap = 'Wistia', annot= True);
```

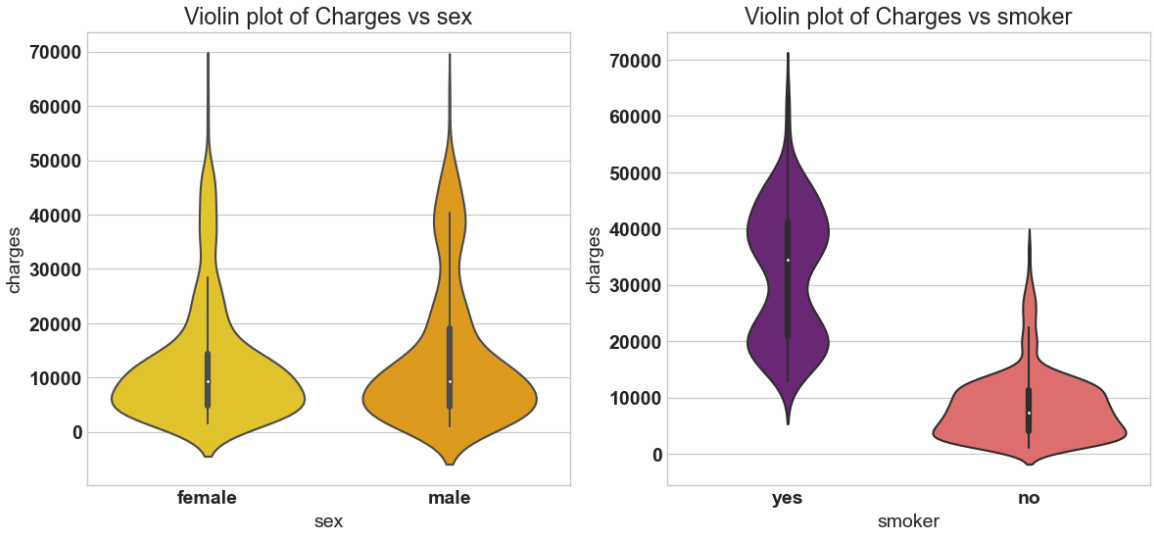
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[6], line 2
      1 # correlation plot
----> 2 corr = df.corr()
      3 sns.heatmap(corr, cmap = 'Wistia', annot= True)

File ~/anaconda3/lib/python3.11/site-packages/pandas/core/frame.py:10054, in DataFrame.corr(self, method, min_periods, numeric_only)
  10052 cols = data.columns
  10053 idx = cols.copy()
> 10054 mat = data.to_numpy(dtype=float, na_value=np.nan, copy=False)
  10056 if method == "pearson":
  10057     correl = libalgos.nancorr(mat, minp=min_periods)

File ~/anaconda3/lib/python3.11/site-packages/pandas/core/frame.py:1838, in DataFrame.to_numpy(self, dtype, copy, na_value)
   1836 if dtype is not None:
   1837     dtype = np.dtype(dtype)
-> 1838 result = self._mgr.as_array(dtype=dtype, copy=copy, na_value=na_value)
   1839 if result.dtype is not dtype:
   1840     result = np.array(result, dtype=dtype, copy=False)

File ~/anaconda3/lib/python3.11/site-packages/pandas/core/internals/managers.py:1732, in BlockManager.as_array(self, dtype, copy, na_value)
   1730         arr.flags.writeable = False
   1731 else:
-> 1732     arr = self._interleave(dtype=dtype, na_value=na_value)
   1733     # The underlying data was copied within _interleave, so no need
   1734     # to further copy if copy=True or setting na_value
   1736 if na_value is not lib.no_default:

File ~/anaconda3/lib/python3.11/site-packages/pandas/core/internals/managers.py:1794, in BlockManager._interleave(self, dtype, na_value)
   1792     else:
   1793         arr = blk.get_values(dtype)
-> 1794     result[rl.indexer] = arr
   1795     itemmask[rl.indexer] = 1
   1797 if not itemmask.all():

ValueError: could not convert string to float: 'female'
```

In [7]:
```
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');
```

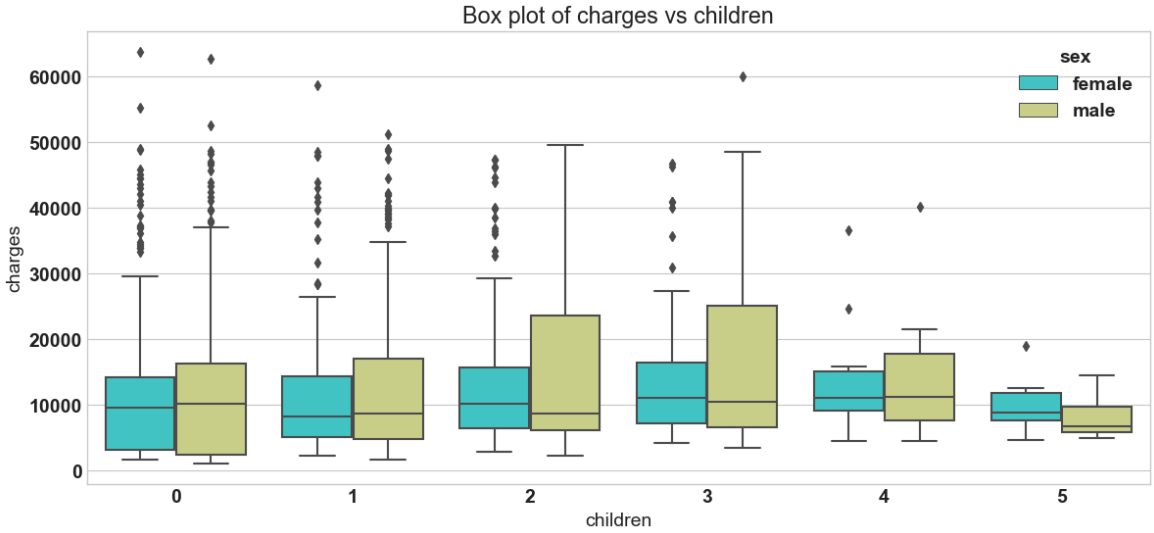
```
/var/folders/20/prf96pn926n0x2yzn_v2yhbr0000gn/T/ipykernel_17464/111
9713767.py:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v
0.14.0.

Please adapt your code to use either `displot` (a figure-level funct
ion with
similar flexibility) or `histplot` (an axes-level function for histo
grams).

For a guide to updating your code to use the new functions, please s
ee
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (ht
tps://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(df['charges'],bins=50,color='r',ax=ax)
/var/folders/20/prf96pn926n0x2yzn_v2yhbr0000gn/T/ipykernel_17464/111
9713767.py:8: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v
0.14.0.

Please adapt your code to use either `displot` (a figure-level funct
ion with
similar flexibility) or `histplot` (an axes-level function for histo
grams).

For a guide to updating your code to use the new functions, please s
ee
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (ht
tps://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(np.log10(df['charges']),bins=40,color='b',ax=ax)
```



In [8]:
```
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');
```



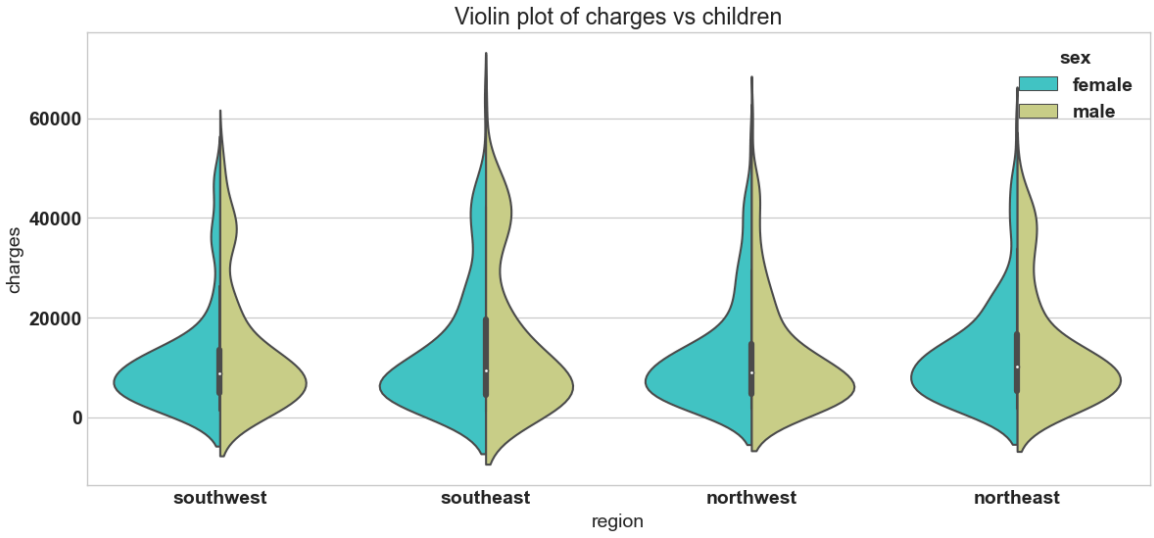
In [9]:
```
plt.figure(figsize=(14,6))
sns.boxplot(x='children', y='charges',hue='sex',data=df,palette='rainb
plt.title('Box plot of charges vs children');
```



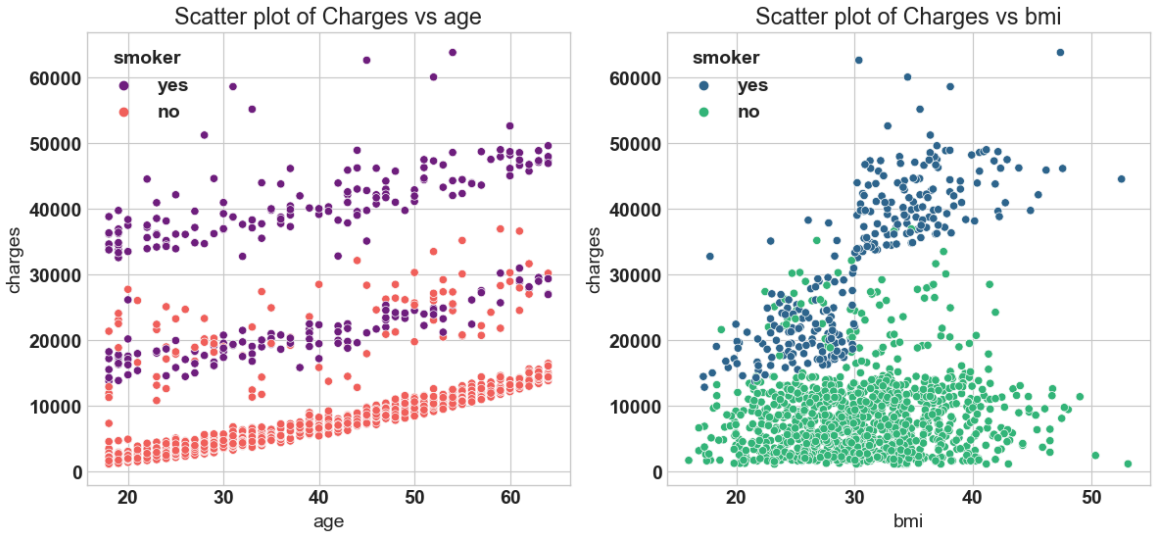
In [10]:
```
df.groupby('children').agg(['mean','min','max'])['charges']
```

```
s.py:482, in WrappedCythonOp._cython_op_ndim_compat(self, values, mi
n_count, ngroups, comp_ids, mask, result_mask, **kwargs)
    481     result_mask = result_mask[None, :]
--> 482 res = self._call_cython_op(
    483     values2d,
    484     min_count=min_count,
    485     ngroups=ngroups,
    486     comp_ids=comp_ids,
    487     mask=mask,
    488     result_mask=result_mask,
    489     **kwargs,
    490 )
    491 if res.shape[0] == 1:

File ~/anaconda3/lib/python3.11/site-packages/pandas/core/groupby/op
s.py:541, in WrappedCythonOp._call_cython_op(self, values, min_coun
t, ngroups, comp_ids, mask, result_mask, **kwargs)
    540 out_shape = self._get_output_shape(ngroups, values)
--> 541 func = self._get_cython_function(self.kind, self.how, value
s.dtype, is_numeric)
```

In [11]:
```
plt.figure(figsize=(14,6))
sns.violinplot(x='region', y='charges',hue='sex',data=df,palette='rain
plt.title('Violin plot of charges vs children');
```



In [12]:
```
f = plt.figure(figsize=(14,6))
ax = f.add_subplot(121)
sns.scatterplot(x='age',y='charges',data=df,palette='magma',hue='smoke
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='smo
ax.set_title('Scatter plot of Charges vs bmi')
plt.savefig('sc.png');
```



In [13]:
```
# 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 [14]:
```
# 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_er
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 [15]:
```
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
```

Out[15]:
```
((-0.01140290617294196, 0.0988096859767545), 0.043649053770664956)
```

In [16]:
```
## Log transform
df_encode['charges'] = np.log(df_encode['charges'])
```

In [17]:
```
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,
```

In [18]:
```
# 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) ), r
```

In [19]:
```
# 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,'
```

In [20]:
```
# 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

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

Out[20]:

	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

In [21]:
```
# 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)
```

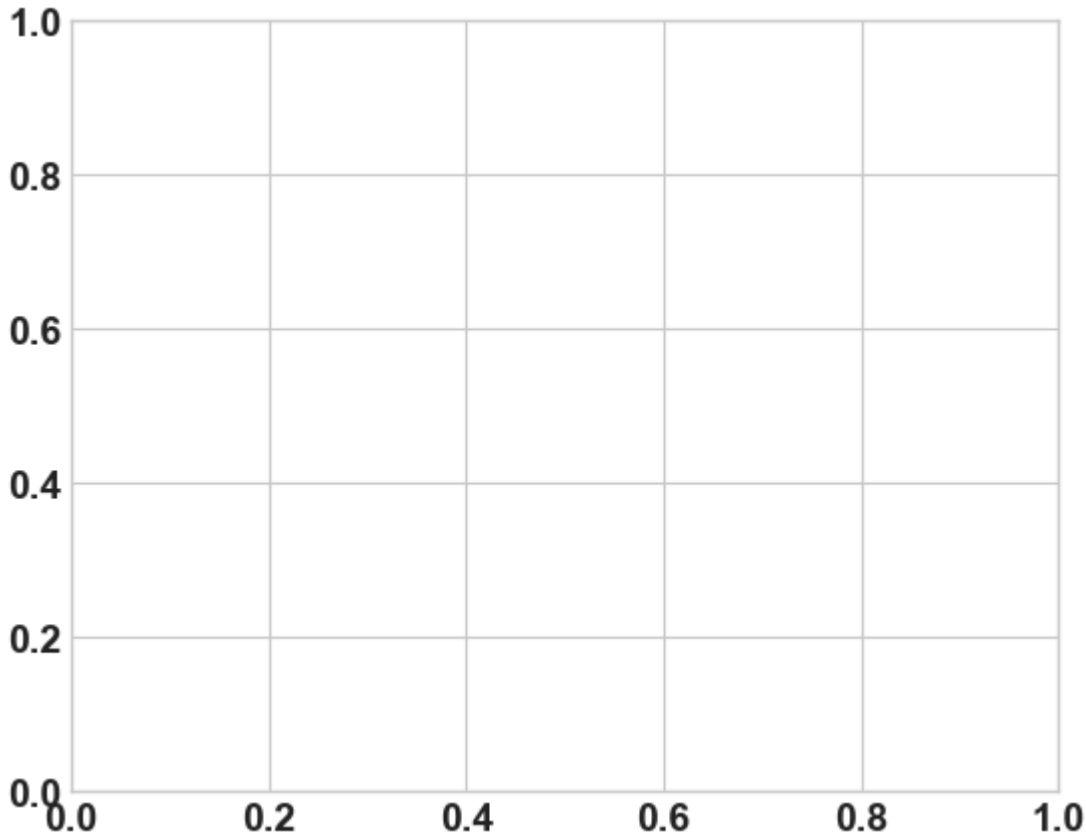
```
The Mean Square Error(MSE) or J(theta) is:  0.18729622322981962
R square obtain for normal equation method is : 0.779568754505531
```

In [22]:
```
# 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)
```

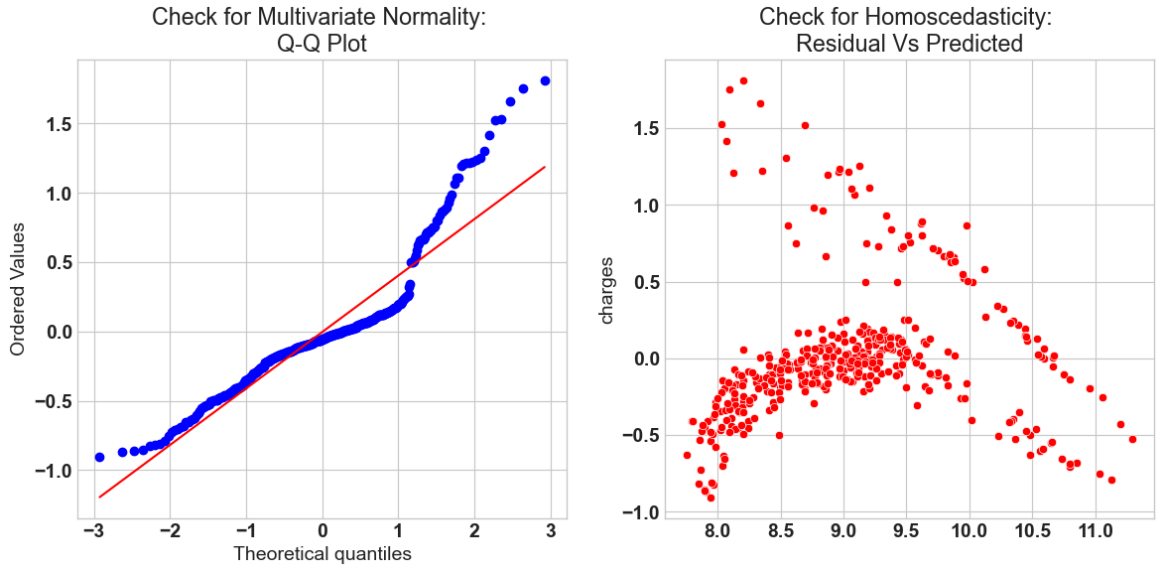
```
The Mean Square Error(MSE) or J(theta) is:  0.18729622322981904
R square obtain for scikit learn library is : 0.7795687545055316
```

In [23]:
```
# 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');
```

```
---------------------------------------------------------------------
-------
TypeError                                 Traceback (most recent cal
l last)
Cell In[23], line 4
      2 f = plt.figure(figsize=(14,5))
      3 ax = f.add_subplot(121)
----> 4 sns.scatterplot(y_test,y_pred_sk,ax=ax,color='r')
      5 ax.set_title('Check for Linearity:\n Actual Vs Predicted val
ue')
      7 # Check for Residual normality & mean

TypeError: scatterplot() takes from 0 to 1 positional arguments but
2 positional arguments (and 1 keyword-only argument) were given
```



In [24]:
```
# 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],col
ax[1].set_title('Check for Homoscedasticity: \nResidual Vs Predicted')
```



In [25]:
```
# Check for Multicollinearity
#Variance Inflation Factor
VIF = 1/(1- R_square_sk)
VIF
```

Out[25]: 4.536561945911133

In [ ]: