DOWNLOADING THE DATA:

[3]:
```
#restart the kernel after installation
!pip install pandas-profiling --quiet
```

[4]:
```
medical_charges_url = 'https://raw.githubusercontent.com/JovianML/opendatasets/master/data/medical-charges.csv'
```

[5]:
```
from urllib.request import urlretrieve
```

[6]:
```
urlretrieve(medical_charges_url, 'medical.csv')
```

[6]:
```
('medical.csv', <http.client.HTTPMessage at 0x170811b5a50>)
```

[7]:
```
import pandas as pd
```

[8]:
```
medical_df = pd.read_csv('medical.csv')
```

[9]:
```
medical_df
```

[9]:

	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
...	...	...	...	...	...	...	...
1333	50	male	30.970	3	no	northwest	10600.54830
1334	18	female	31.920	0	no	northeast	2205.98080
1335	18	female	36.850	0	no	southeast	1629.83350
1336	21	female	25.800	0	no	southwest	2007.94500
1337	61	female	29.070	0	yes	northwest	29141.36030

1338 rows × 7 columns

[10]:
```
medical_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1338 non-null   int64  
 1   sex       1338 non-null   object 
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64  
 4   smoker    1338 non-null   object 
 5   region    1338 non-null   object 
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

[11]:
```
medical_df.describe()
```

[11]:

	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

EXPLORATORY ANALYSIS AND VISUALIZATION:

[12]:
```
!pip install plotly matplotlib seaborn --quiet
```

[13]:
```
import plotly.express as px
import matplotlib
import matplotlib.pyplot as plt
```

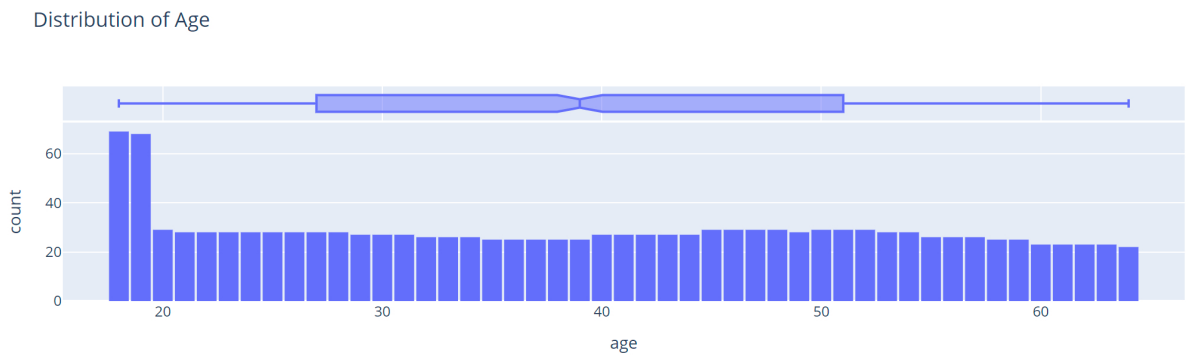
```
import seaborn as sns
%matplotlib inline
```

[14]:
```
sns.set_style('darkgrid')
matplotlib.rcParams['font.size'] = 14
matplotlib.rcParams['figure.figsize'] = (10, 6)
matplotlib.rcParams['figure.facecolor'] = '#00000000'
```

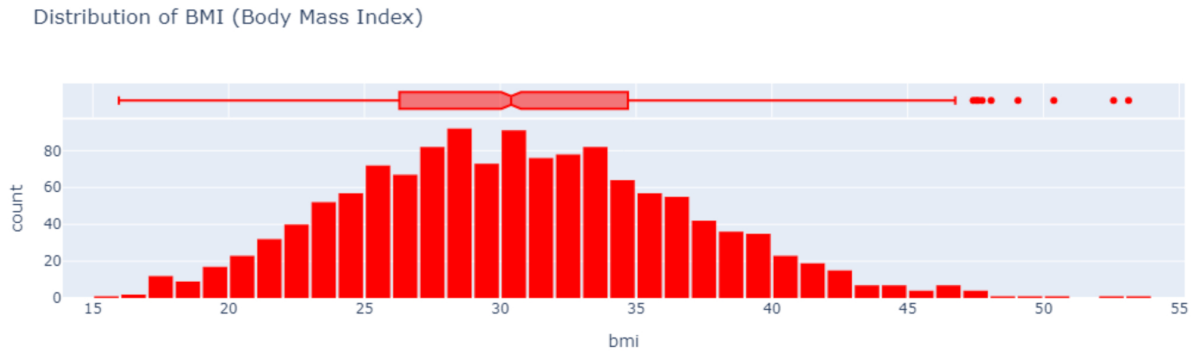
[15]:
```
medical_df.age.describe()
```

[15]:
```
count    1338.000000
mean       39.207025
std        14.049960
min        18.000000
25%        27.000000
50%        39.000000
75%        51.000000
max        64.000000
Name: age, dtype: float64
```

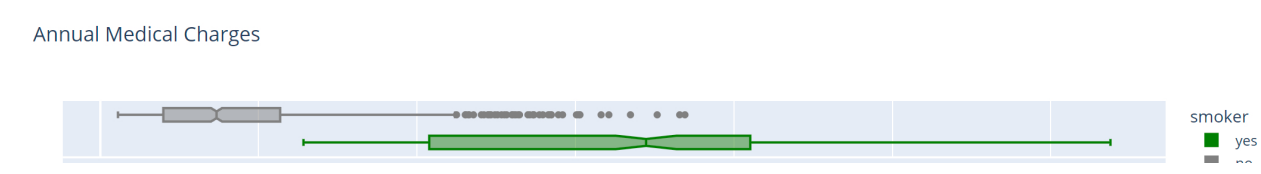
[16]:
```
fig = px.histogram(medical_df,
                   x='age',
                   marginal='box',
                   nbins=47,
                   title='Distribution of Age')
fig.update_layout(bargap=0.1)
fig.show()
```

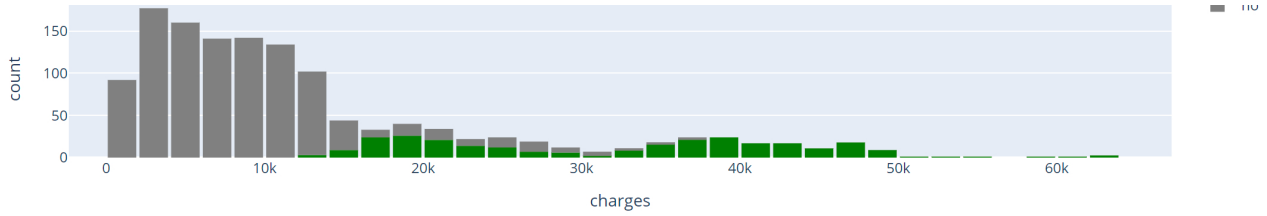


[17]:
```
fig = px.histogram(medical_df,
                   x='bmi',
                   marginal='box',
                   color_discrete_sequence=['red'],
                   title='Distribution of BMI (Body Mass Index)')
fig.update_layout(bargap=0.1)
fig.show()
```

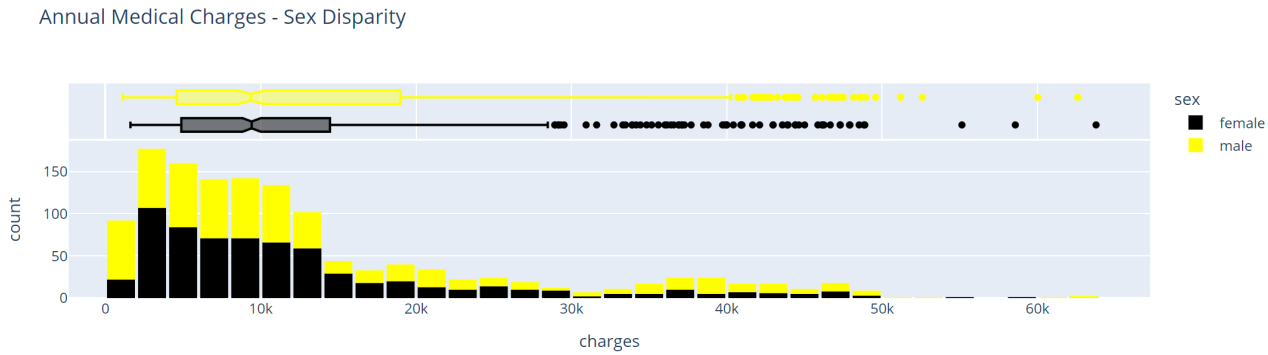


[18]:
```
fig = px.histogram(medical_df,
                   x='charges',
                   marginal='box',
                   color='smoker',
                   color_discrete_sequence=['green', 'grey'],
                   title='Annual Medical Charges')
fig.update_layout(bargap=0.1)
fig.show()
```





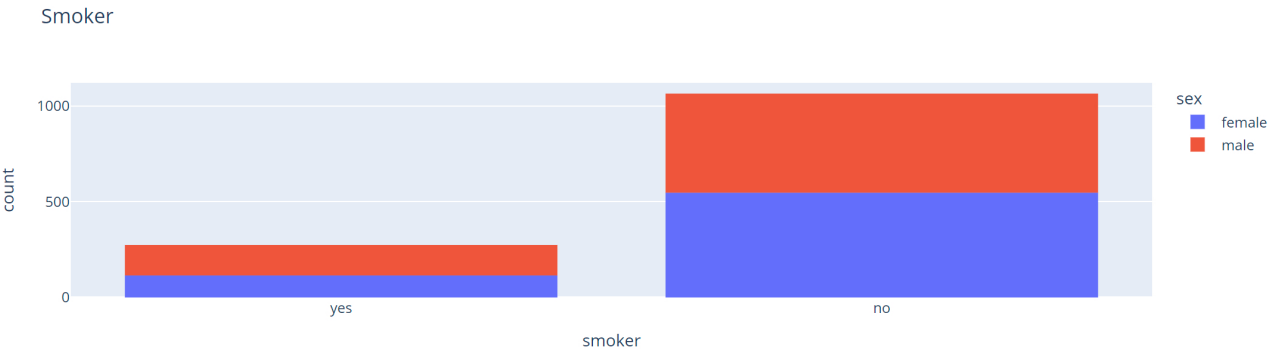
[19]:
```
fig = px.histogram(medical_df,
                   x='charges',
                   marginal='box',
                   color='sex',
                   color_discrete_sequence=['black', 'yellow'],
                   title='Annual Medical Charges - Sex Disparity')
fig.update_layout(bargap=0.1)
fig.show()
```



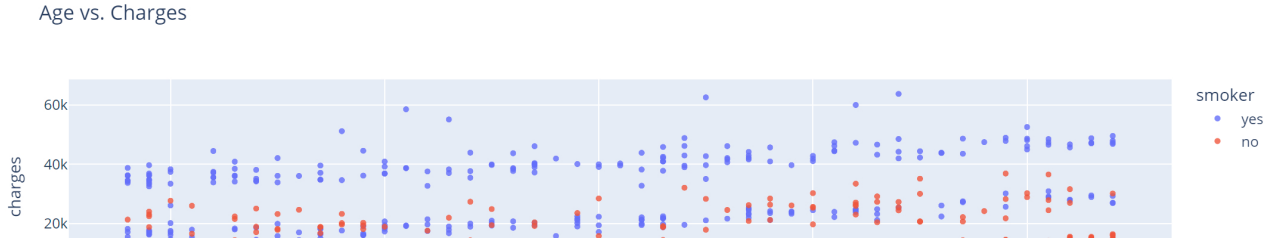
[20]:
```
medical_df.smoker.value_counts()
```

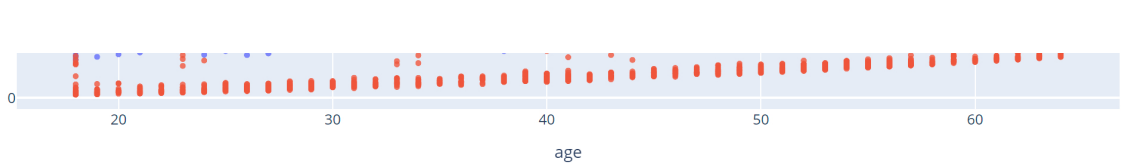
[20]:
```
smoker
no     1064
yes     274
Name: count, dtype: int64
```

[21]:
```
px.histogram(medical_df, x='smoker', color='sex', title='Smoker')
```

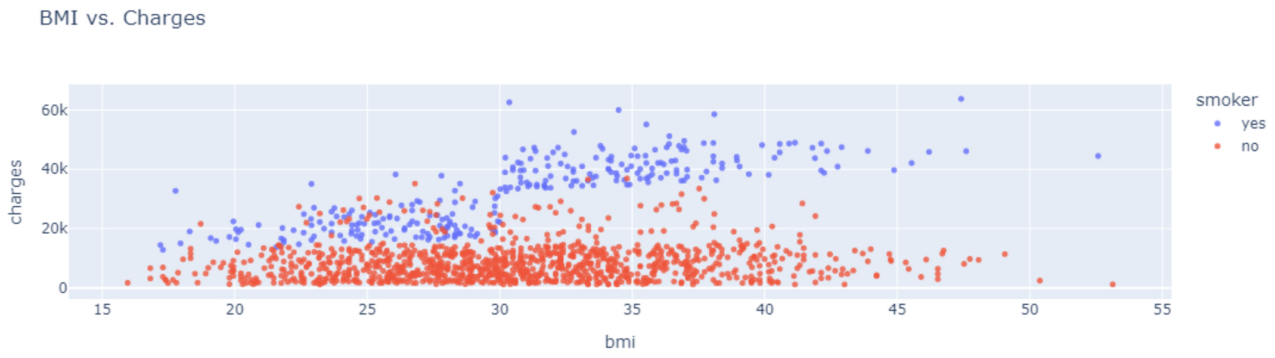


[22]:
```
fig = px.scatter(medical_df,
                 x='age',
                 y='charges',
                 color='smoker',
                 opacity=0.8,
                 hover_data=['sex'],
                 title='Age vs. Charges')
fig.update_traces(marker_size=5)
fig.show()
```

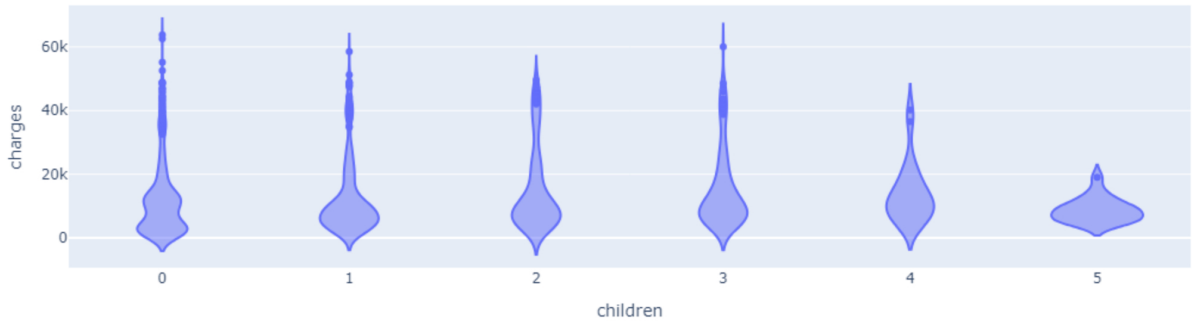




[23]:
```
fig = px.scatter(medical_df,
                 x='bmi',
                 y='charges',
                 color='smoker',
                 opacity=0.8,
                 hover_data=['sex'],
                 title='BMI vs. Charges')
fig.update_traces(marker_size=5)
fig.show()
```



[24]:
```
px.violin(medical_df, x='children', y='charges')
```



CORRELATION:

[25]:
```
medical_df.charges.corr(medical_df.age)
```

[25]:
```
0.2990081933306476
```

[26]:
```
medical_df.charges.corr(medical_df.bmi)
```

[26]:
```
0.19834096883362884
```

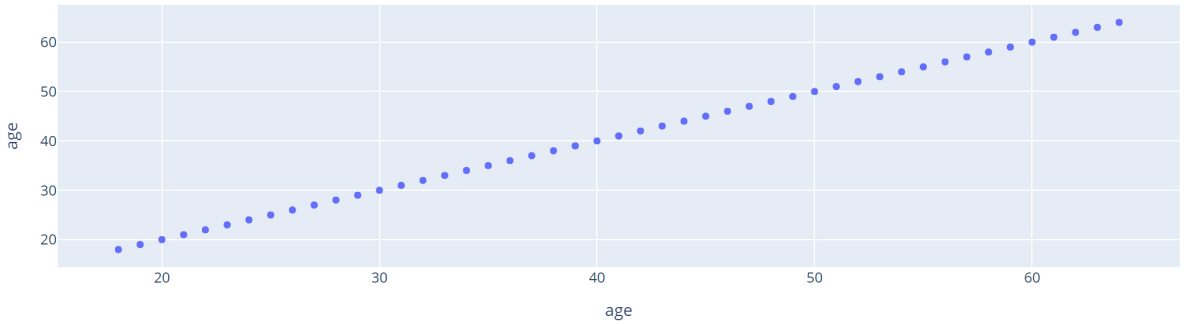
[27]:
```
smoker_values={'no':0, 'yes':1}
smoker_numeric=medical_df.smoker.map(smoker_values)
smoker_numeric
```

[27]:
```
0       1
1       0
2       0
3       0
4       0
       ..
1333    0
1334    0
1335    0
1336    0
1337    1
Name: smoker, Length: 1338, dtype: int64
```

[28]:
```
medical_df.charges.corr(smoker_numeric)
```

[28]:
```
0.7872514304984767
```

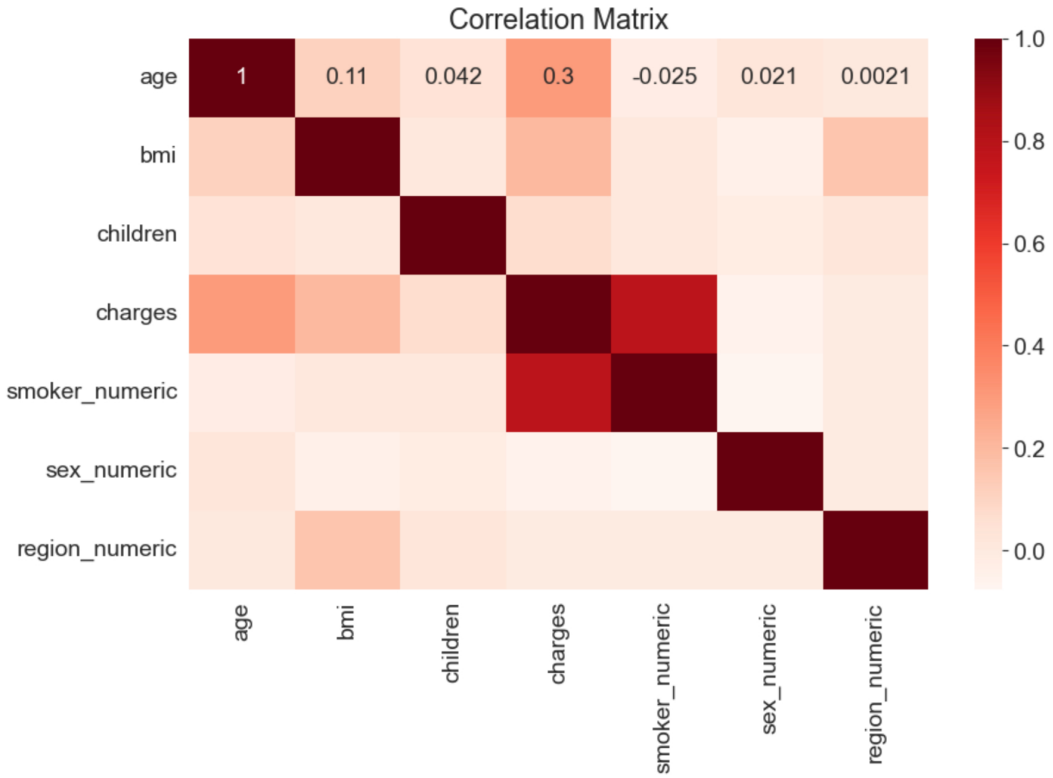
[29]:
```
px.scatter(medical_df, x = 'age', y='age')
```



[30]:
```
medical_df['smoker_numeric'] = medical_df['smoker'].map({'no': 0, 'yes': 1})
medical_df['sex_numeric'] = medical_df['sex'].map({'male': 0, 'female': 1})
medical_df['region_numeric'] = medical_df['region'].astype('category').cat.codes

# Calculate correlations
correlation_matrix = medical_df[['age', 'bmi', 'children', 'charges', 'smoker_numeric', 'sex_numeric', 'region_numeric']].corr()
print(correlation_matrix)
```

```
                     age       bmi  children   charges  smoker_numeric  \
age             1.000000  0.109272  0.042469  0.299008       -0.025019
bmi             0.109272  1.000000  0.012759  0.198341        0.003750
children        0.042469  0.012759  1.000000  0.067998        0.007673
charges         0.299008  0.198341  0.067998  1.000000        0.787251
smoker_numeric -0.025019  0.003750  0.007673  0.787251        1.000000
sex_numeric     0.020856 -0.046371 -0.017163 -0.057292       -0.076185
region_numeric  0.002127  0.157566  0.016569 -0.006208       -0.002181

                sex_numeric  region_numeric
age                0.020856        0.002127
bmi               -0.046371        0.157566
children          -0.017163        0.016569
charges           -0.057292       -0.006208
smoker_numeric    -0.076185       -0.002181
sex_numeric        1.000000       -0.004588
region_numeric    -0.004588        1.000000
```

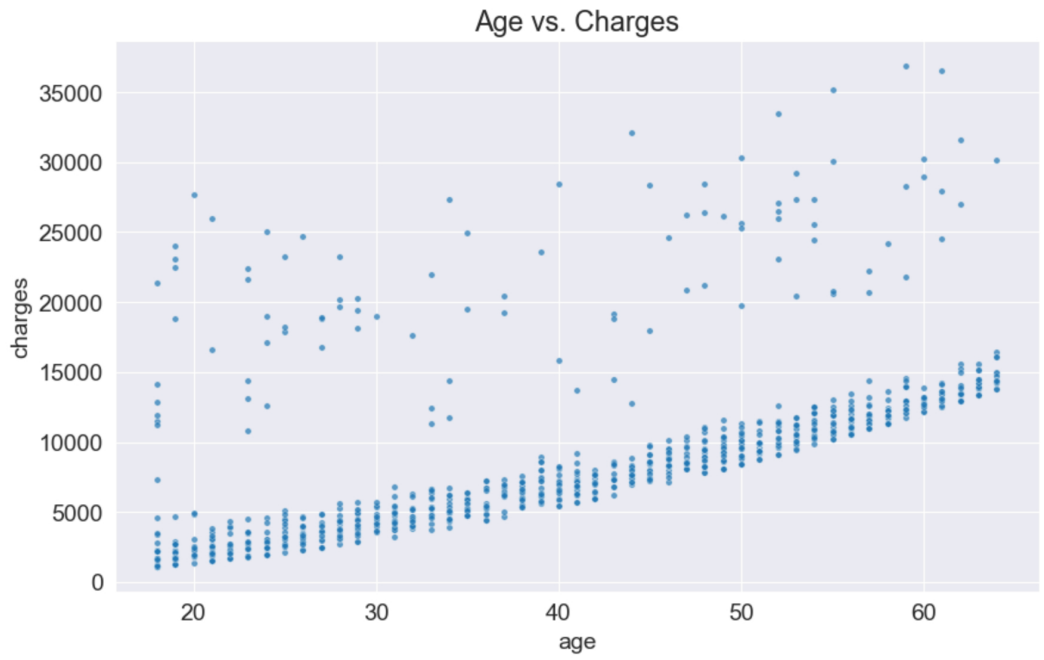
[31]:
```
plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='Reds')
plt.title('Correlation Matrix')
plt.show()
```



LINEAR REGRESSION USING A SINGLE FEATURE:

[32]:
```
non_smoker_df = medical_df[medical_df.smoker == 'no']
```

[33]:
```
plt.title('Age vs. Charges')
sns.scatterplot(data=non_smoker_df, x='age', y='charges', alpha=0.7, s=15);
```



MODEL:

[34]:
```
def estimate_charges(age,w,b):
    return w * age + b
```

[35]:
```
w = 50
b = 100
```

[36]:
```
estimate_charges(30,w,b)
```

[36]:
```
1600
```

[37]:
```
ages=non_smoker_df.age
ages
```

[37]:
```
1       18
2       28
3       33
4       32
5       31
        ..
1332    52
1333    50
1334    18
1335    18
1336    21
Name: age, Length: 1064, dtype: int64
```

[38]:
```
estimated_charges = estimate_charges(ages, w, b)
```

[39]:
```
estimated_charges
```

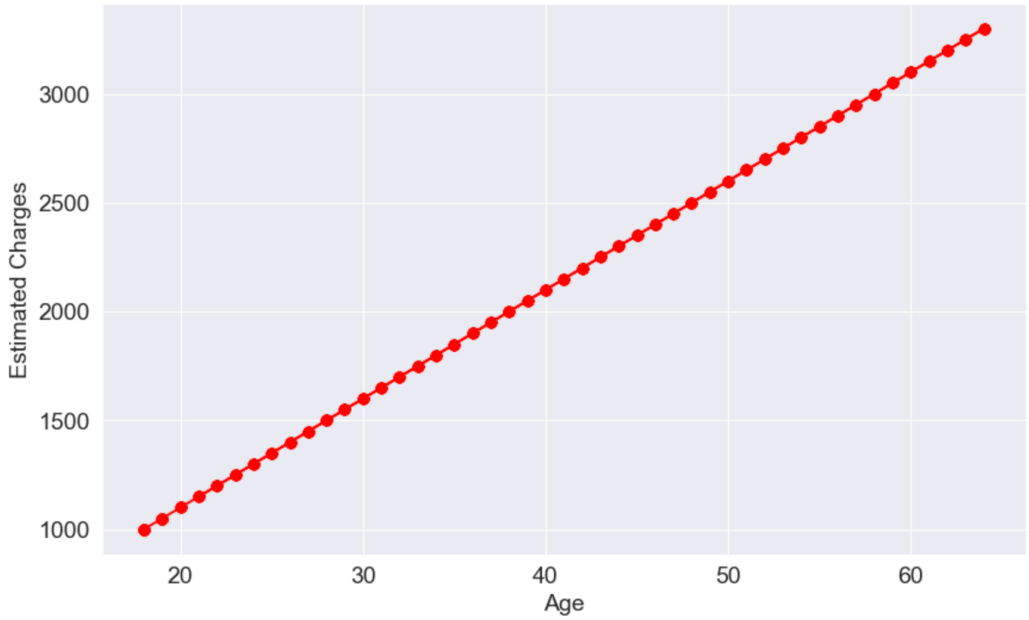
[39]:
```
1       1000
2       1500
3       1750
4       1700
5       1650
        ...
1332    2700
1333    2600
1334    1000
1335    1000
1336    1150
Name: age, Length: 1064, dtype: int64
```

[40]:
```
non_smoker_df.charges
```

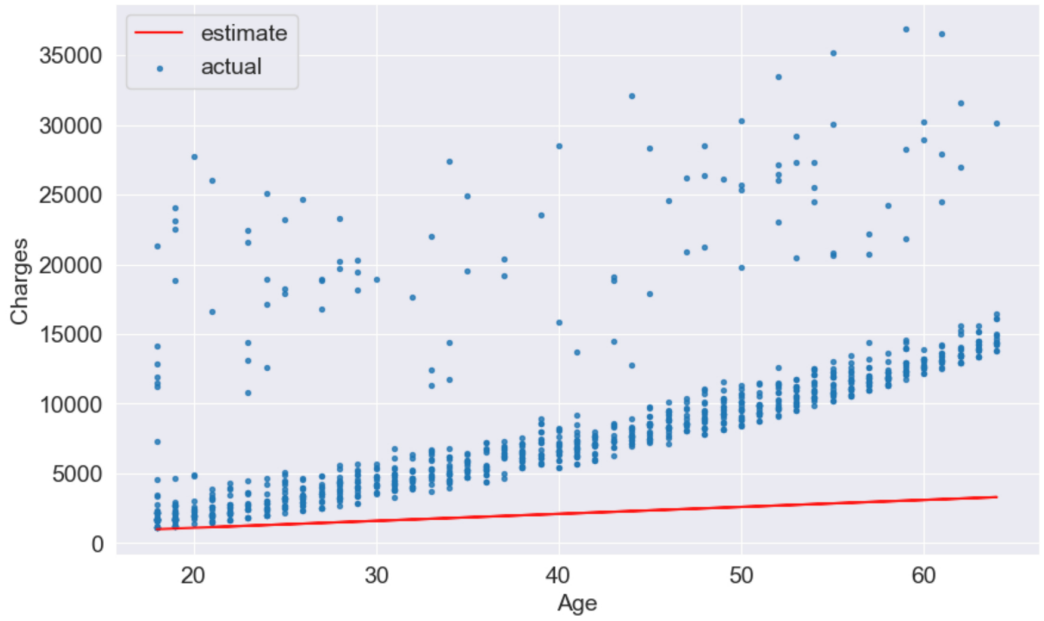
[40]:
```
1        1725.55230
2        4449.46200
3       21984.47061
4        3866.85520
5        3756.62160
           ...
1332    11411.68500
1333    10600.54830
1334     2205.98080
1335     1629.83350
1336     2007.94500
Name: charges, Length: 1064, dtype: float64
```

[41]:
```
plt.plot(ages, estimated_charges, 'r-o');
plt.xlabel('Age');
```

```
plt.xlabel('Age');
plt.ylabel('Estimated Charges');
```

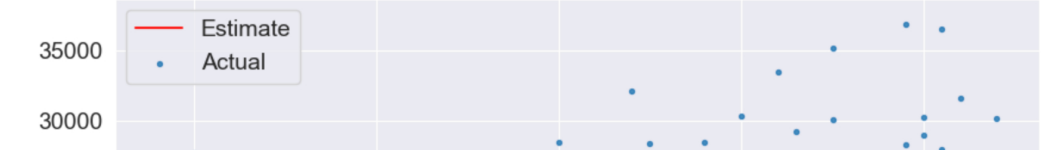


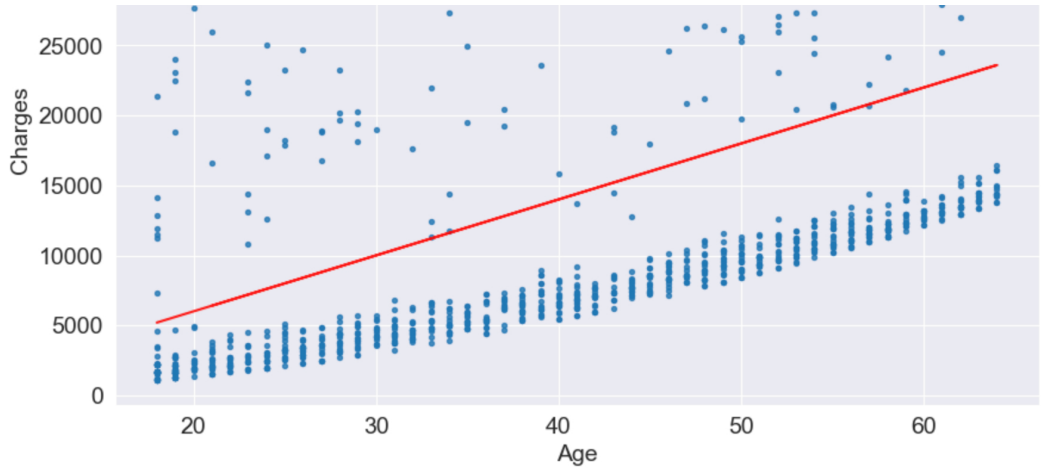
[42]:
```
target=non_smoker_df.charges

plt.plot(ages, estimated_charges, 'r', alpha=0.9)

plt.scatter(ages,target, s=8, alpha=0.8)
plt.xlabel('Age');
plt.ylabel('Charges');
plt.legend(['estimate','actual']);
```



[43]:
```
def try_parameters(w, b):
    ages = non_smoker_df.age
    target = non_smoker_df.charges

    estimated_charges = estimate_charges(ages, w, b)

    plt.plot(ages, estimated_charges, 'r', alpha=0.9);
    plt.scatter(ages, target, s=8,alpha=0.8);
    plt.xlabel('Age');
    plt.ylabel('Charges')
    plt.legend(['Estimate', 'Actual']);
```

[44]:
```
try_parameters(400,-2000)
```





[45]:
```
targets = non_smoker_df.charges
targets
```

[45]:
```
1        1725.55230
2        4449.46200
3       21984.47061
4        3866.85520
5        3756.62160
           ...
1332    11411.68500
1333    10600.54830
1334     2205.98080
1335     1629.83350
1336     2007.94500
Name: charges, Length: 1064, dtype: float64
```

[46]:
```
predictions = estimated_charges
predictions
```

[46]:
```
1       1000
2       1500
3       1750
4       1700
5       1650
        ...
1332    2700
1333    2600
1334    1000
1335    1000
1336    1150
Name: age, Length: 1064, dtype: int64
```

LOSS/COST FUNCTION:

[47]:
```
!pip install numpy --quiet
```

[48]:
```
import numpy as np
```

[49]:
```
def rmse(targets, predictions):
    return(np.sqrt(np.mean(np.square(targets-predictions))))
```

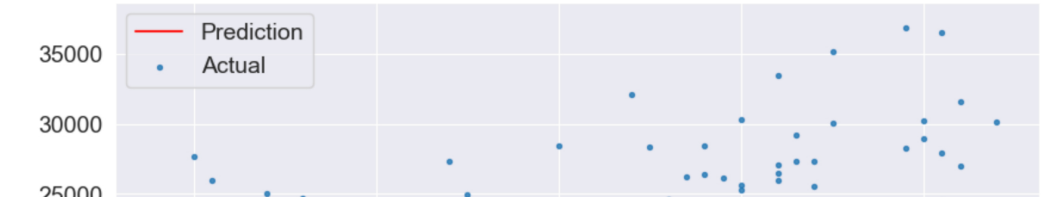
[50]:
```
rmse(targets, predictions)
```

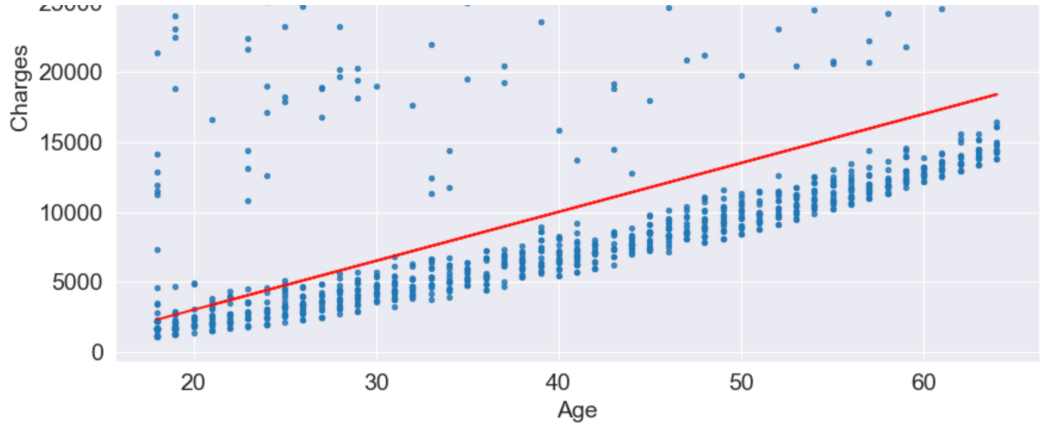
[50]:
```
8461.949562575493
```

[51]:
```
def try_parameters(w, b):
    ages = non_smoker_df.age
    target = non_smoker_df.charges
    predictions = estimate_charges(ages, w, b)

    plt.plot(ages, predictions, 'r', alpha=0.9);
    plt.scatter(ages, target, s=8,alpha=0.8);
    plt.xlabel('Age');
    plt.ylabel('Charges')
    plt.legend(['Prediction', 'Actual']);

    loss = rmse(target, predictions)
    print("RMSE Loss: ", loss)
```

[52]:
```
try_parameters(350, -4000)
```

```
RMSE Loss:  4991.993804156943
```





[53]:
```
!pip install scikit-learn --quiet
```

LINEAR REGRESSION USING SCIKIT LEARN:

[54]:
```
from sklearn.linear_model import LinearRegression
```

[55]:
```
model= LinearRegression()
help(model.fit)
```

```
Help on method fit in module sklearn.linear_model._base:

fit(X, y, sample_weight=None) method of sklearn.linear_model._base.LinearRegression instance
    Fit linear model.

    Parameters
    ----------
    X : {array-like, sparse matrix} of shape (n_samples, n_features)
        Training data.

    y : array-like of shape (n_samples,) or (n_samples, n_targets)
        Target values. Will be cast to X's dtype if necessary.

    sample_weight : array-like of shape (n_samples,), default=None
        Individual weights for each sample.

        .. versionadded:: 0.17
           parameter *sample_weight* support to LinearRegression.

    Returns
    -------
    self : object
        Fitted Estimator.
```

[56]:
```
inputs = non_smoker_df[['age']]
targets = non_smoker_df.charges
print('inputs.shape :', inputs.shape)
print('targets.shape :', targets.shape)
```

```
inputs.shape : (1064, 1)
targets.shape : (1064,)
```

[57]:
```
model.fit(inputs,targets)
```

[57]:
```
▾ LinearRegression
LinearRegression()
```

[58]:
```
model.predict(np.array([[23],[37],[61]]))
```

```
C:\Users\adity\anaconda3\Lib\site-packages\sklearn\base.py:439: UserWarning:

X does not have valid feature names, but LinearRegression was fitted with feature names
```

[58]:
```
array([ 4055.30443855,  7796.78921819, 14210.76312614])
```

[59]:
```
predictions=model.predict(inputs)
predictions
```

[59]:
```
array([2719.0598744 , 5391.54900271, 6727.79356686, ..., 2719.0598744 ,
       2719.0598744 , 3520.80661289])
```

[60]:
```
rmse(targets,predictions)
```

[60]:
```
4662.505766636395
```

[61]:
```
# w
model.coef_
```

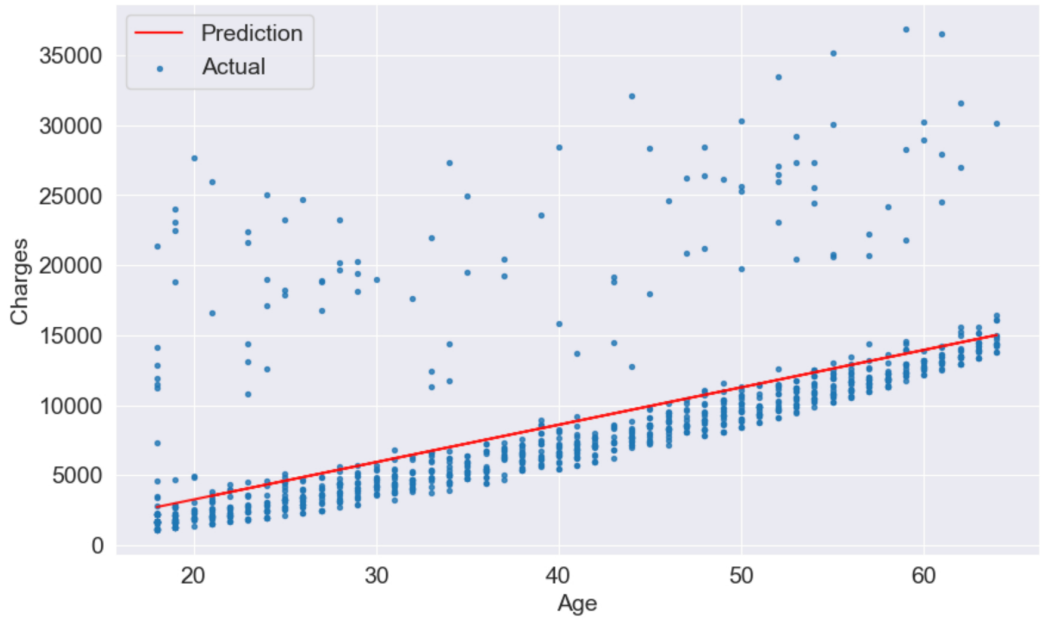
[61]:
```
array([267.24891283])
```

[62]:
```
# b
model.intercept_
```

[62]: -2091.420556650827

[63]:
```
try_parameters(model.coef_,model.intercept_)
```

```
RMSE Loss:  4662.505766636395
```



LINEAR REGRESSION USING MULTIPLE FEATURES:

[64]:
```
# Create inputs and targets
inputs, targets = non_smoker_df[['age', 'bmi']], non_smoker_df['charges']

# Create and train the model
model = LinearRegression().fit(inputs, targets)

# Generate predictions
predictions = model.predict(inputs)

# Compute loss to evalute the model
loss = rmse(targets, predictions)
print('Loss:', loss)
```

```
Loss: 4662.3128354612945
```

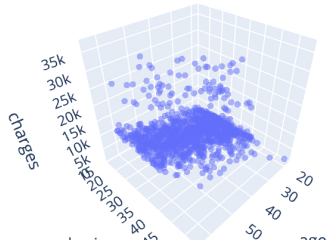
[65]:
```
non_smoker_df.charges.corr(non_smoker_df.bmi)
```

[65]:
```
0.08403654312833271
```

[66]:
```
model.coef_,model.intercept_
```

[66]:
```
(array([266.87657817,   7.07547666]), -2293.6320906488654)
```

[67]:
```
fig = px.scatter_3d(non_smoker_df, x='age', y='bmi', z='charges')
fig.update_traces(marker_size=3, marker_opacity=0.5)
fig.show()
```



[68]:
```
# Create inputs and targets
inputs, targets = non_smoker_df[['age', 'bmi', 'children']], non_smoker_df['charges']

# Create and train the model
model = LinearRegression().fit(inputs, targets)

# Generate predictions
predictions = model.predict(inputs)
```

```
# Compute loss to evalute the model
loss = rmse(targets, predictions)
print('Loss:', loss)
```

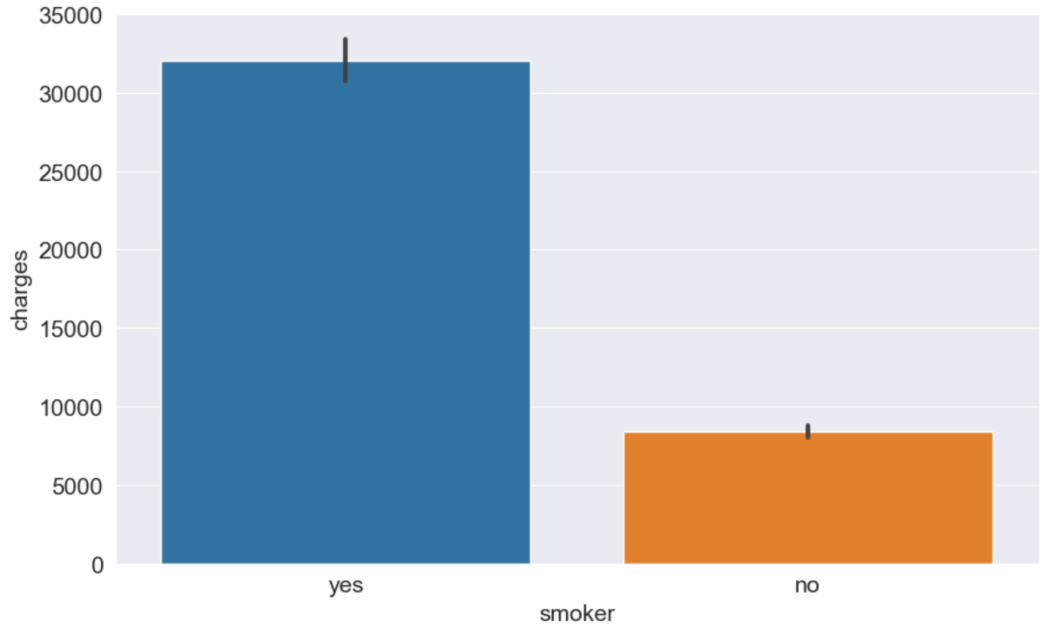
```
Loss: 4608.470405038247
```

[69]:
```
# Create inputs and targets
inputs, targets = medical_df[['age', 'bmi', 'children']], medical_df['charges']

# Create and train the model
model = LinearRegression().fit(inputs, targets)

# Generate predictions
predictions = model.predict(inputs)

# Compute loss to evalute the model
loss = rmse(targets, predictions)
print('Loss:', loss)
```

```
Loss: 11355.317901125973
```

USING CATEGORICAL FEATURES FOR MACHINE LEARNING

[70]:
```
sns.barplot(data=medical_df,x='smoker', y='charges');
```



[71]:
```
smoker_codes= {'no':0, 'yes':1}
medical_df['smoker_code']=medical_df.smoker.map(smoker_codes)
```

[72]:
```
medical_df
```

[72]:

	age	sex	bmi	children	smoker	region	charges	smoker_numeric	sex_numeric	region_numeric	smoker_code
0	19	female	27.900	0	yes	southwest	16884.92400	1	1	3	1
1	18	male	33.770	1	no	southeast	1725.55230	0	0	2	0
2	28	male	33.000	3	no	southeast	4449.46200	0	0	2	0
3	33	male	22.705	0	no	northwest	21984.47061	0	0	1	0
4	32	male	28.880	0	no	northwest	3866.85520	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...
1333	50	male	30.970	3	no	northwest	10600.54830	0	0	1	0
1334	18	female	31.920	0	no	northeast	2205.98080	0	1	0	0
1335	18	female	36.850	0	no	southeast	1629.83350	0	1	2	0
1336	21	female	25.800	0	no	southwest	2007.94500	0	1	3	0
1337	61	female	29.070	0	yes	northwest	29141.36030	1	1	1	1

1338 rows × 11 columns

[73]:
```
medical_df.charges.corr(medical_df.smoker_code)
```

[73]:
```
0.7872514304984767
```

[74]:
```
# Create inputs and targets
inputs, targets = medical_df[['age', 'bmi', 'children', 'smoker_code']], medical_df['charges']

# Create and train the model
model = LinearRegression().fit(inputs, targets)
```

```
# Generate predictions
predictions = model.predict(inputs)

# Compute loss to evalute the model
loss = rmse(targets, predictions)
print('Loss:', loss)
```

```
Loss: 6056.439217188081
```

[75]:
```
sex_codes= {'female':0,'male':1}
medical_df['sex_code']=medical_df.sex.map(sex_codes)
```

[76]:
```
medical_df.charges.corr(medical_df.sex_code)
```

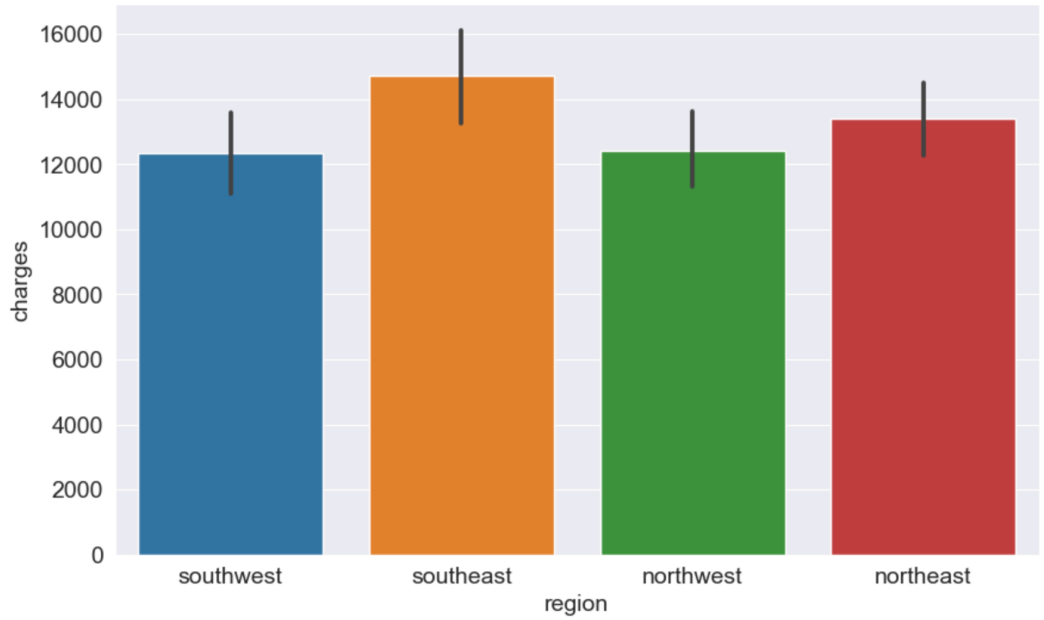
[76]:
```
0.057292062202025366
```

[77]:
```
# Create inputs and targets
inputs, targets = medical_df[['age', 'bmi', 'children', 'smoker_code','sex_code']], medical_df['charges']

# Create and train the model
model = LinearRegression().fit(inputs, targets)

# Generate predictions
predictions = model.predict(inputs)

# Compute loss to evalute the model
loss = rmse(targets, predictions)
print('Loss:', loss)
```

```
Loss: 6056.100708754546
```

ONE HOT ENCODING:

[78]:
```
sns.barplot(data=medical_df, x='region', y='charges');
```



[79]:
```
from sklearn import preprocessing
enc = preprocessing.OneHotEncoder()
enc.fit(medical_df[['region']])
enc.categories_
```

[79]:
```
[array(['northeast', 'northwest', 'southeast', 'southwest'], dtype=object)]
```

[80]:
```
enc.transform([['northeast'],
               ['northwest']]).toarray()
```

```
C:\Users\adity\anaconda3\Lib\site-packages\sklearn\base.py:439: UserWarning:

X does not have valid feature names, but OneHotEncoder was fitted with feature names
```

[80]:
```
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.]])
```

[81]:
```
one_hot=enc.transform(medical_df[['region']]).toarray()
one_hot
```

[81]:
```
array([[0., 0., 0., 1.],
       [0., 0., 1., 0.],
       [0., 0., 1., 0.],
       ...,
       [0., 0., 1., 0.],
       [0., 0., 0., 1.],
       [0., 1., 0., 0.]])
```

[82]:
```
medical_df[['northeast', 'northwest', 'southeast', 'southwest']] = one_hot
```

[83]:
```
medical_df
```

[83]:

	age	sex	bmi	children	smoker	region	charges	smoker_numeric	sex_numeric	region_numeric	smoker_code	sex_code	northeast	northwest	south
0	19	female	27.900	0	yes	southwest	16884.92400	1	1	3	1	0	0.0	0.0	
1	18	male	33.770	1	no	southeast	1725.55230	0	0	2	0	1	0.0	0.0	
2	28	male	33.000	3	no	southeast	4449.46200	0	0	2	0	1	0.0	0.0	
3	33	male	22.705	0	no	northwest	21984.47061	0	0	1	0	1	0.0	1.0	
4	32	male	28.880	0	no	northwest	3866.85520	0	0	1	0	1	0.0	1.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1333	50	male	30.970	3	no	northwest	10600.54830	0	0	1	0	1	0.0	1.0	
1334	18	female	31.920	0	no	northeast	2205.98080	0	1	0	0	0	1.0	0.0	
1335	18	female	36.850	0	no	southeast	1629.83350	0	1	2	0	0	0.0	0.0	
1336	21	female	25.800	0	no	southwest	2007.94500	0	1	3	0	0	0.0	0.0	
1337	61	female	29.070	0	yes	northwest	29141.36030	1	1	1	1	0	0.0	1.0	

1338 rows × 16 columns

[84]:
```
# Create inputs and targets
input_cols = ['age', 'bmi', 'children', 'smoker_code', 'sex_code', 'northeast', 'northwest', 'southeast', 'southwest']
inputs, targets = medical_df[input_cols], medical_df['charges']

# Create and train the model
model = LinearRegression().fit(inputs, targets)

# Generate predictions
predictions = model.predict(inputs)

# Compute loss to evalute the model
loss = rmse(targets, predictions)
print('Loss:', loss)
```

```
Loss: 6041.6796511744515
```

```
MODEL IMPROVEMENTS
```

[85]:
```
model
```

[85]:
```
LinearRegression()
```

[86]:
```
model.predict([[28,30,2,1,0,0,1,0,0]])
```

```
C:\Users\adity\anaconda3\Lib\site-packages\sklearn\base.py:439: UserWarning:

X does not have valid feature names, but LinearRegression was fitted with feature names
```

[86]:
```
array([29875.81463599])
```

[87]:
```
model.coef_
```

[87]:
```
array([  256.85635254,   339.19345361,   475.50054515, 23848.53454191,
       -131.3143594 ,   587.00923503,   234.0453356 ,  -448.01281436,
       -373.04175627])
```

FEATURE SCALING:

[88]:
```
from sklearn.preprocessing import StandardScaler
```

[89]:
```
numeric_cols=['age','bmi','children']
scaler=StandardScaler()
scaler.fit(medical_df[numeric_cols])
```

[89]:
```
StandardScaler()
```

[90]:
```
scaler.mean_
```

[90]:
```
array([39.20702541, 30.66339686,  1.09491779])
```

[91]:
```
scaler.var_
```

[91]:
```
array([197.25385199,  37.16008997,   1.45212664])
```

[92]:
```
medical_df[numeric_cols]
```

[92]:

	age	bmi	children
0	19	27.900	0
1	18	33.770	1

2	28	33.000	3
3	33	22.705	0
4	32	28.880	0
...	...	...	...
1333	50	30.970	3
1334	18	31.920	0
1335	18	36.850	0
1336	21	25.800	0
1337	61	29.070	0

1338 rows × 3 columns

[93]:
```
scaled_inputs = scaler.transform(medical_df[numeric_cols])
scaled_inputs
```

[93]:
```
array([[-1.43876426, -0.45332   , -0.90861367],
       [-1.50996545,  0.5096211 , -0.07876719],
       [-0.79795355,  0.38330685,  1.58092576],
       ...,
       [-1.50996545,  1.0148781 , -0.90861367],
       [-1.29636188, -0.79781341, -0.90861367],
       [ 1.55168573, -0.26138796, -0.90861367]])
```

[94]:
```
cat_cols = ['smoker_code', 'sex_code', 'northeast', 'northwest', 'southeast', 'southwest']
categorical_data = medical_df[cat_cols].values
```

[95]:
```
inputs = np.concatenate((scaled_inputs, categorical_data), axis=1)
inputs
```

[95]:
```
array([[-1.43876426, -0.45332   , -0.90861367, ...,  0.        ,
         0.        ,  1.        ],
       [-1.50996545,  0.5096211 , -0.07876719, ...,  0.        ,
         1.        ,  0.        ],
       [-0.79795355,  0.38330685,  1.58092576, ...,  0.        ,
         1.        ,  0.        ],
       ...,
       [-1.50996545,  1.0148781 , -0.90861367, ...,  0.        ,
         1.        ,  0.        ],
       [-1.29636188, -0.79781341, -0.90861367, ...,  0.        ,
         0.        ,  1.        ],
       [ 1.55168573, -0.26138796, -0.90861367, ...,  1.        ,
         0.        ,  0.        ]])
```

[96]:
```
inputs = np.concatenate((scaled_inputs, categorical_data), axis=1)
targets = medical_df.charges

# Create and train the model
model = LinearRegression().fit(inputs, targets)

# Generate predictions
predictions = model.predict(inputs)

# Compute loss to evalute the model
loss = rmse(targets, predictions)
print('Loss:', loss)
```

```
Loss: 6042.751556200273
```

[97]:
```
weights_df=pd.DataFrame({
    'feature':np.append(numeric_cols+cat_cols,1),
    'weight':np.append(model.coef_,model.intercept_)
})
weights_df.sort_values('weight',ascending=False)
```

[97]:

	feature	weight
5	northeast	1.758467e+17
6	northwest	1.758467e+17
7	southeast	1.758467e+17
8	southwest	1.758467e+17
3	smoker_code	2.385276e+04
0	age	3.608872e+03
1	bmi	2.058490e+03
2	children	5.615551e+02
4	sex_code	-1.670010e+02
9	1	-1.758467e+17

[98]:
```
new_customers=[[28,30,2,1,0,0,1,0,0.]]
```

[99]:
```
scaler.transform([[28,30,2]])
```

```
C:\Users\adity\anaconda3\Lib\site-packages\sklearn\base.py:439: UserWarning:
```

```
X does not have valid feature names, but StandardScaler was fitted with feature names
```

[99]:
```
array([[-0.79795355, -0.10882659,  0.75107928]])
```

[100]:
```
model.predict([[-0.79795355, -0.10882659,  0.75107928,1,0,0,1,0,0]])
```

[100]:
```
array([29696.])
```

CREATING A TEST SET:

[101]:
```
from sklearn.model_selection import train_test_split
```

[102]:
```
inputs_train, inputs_test, targets_train, targets_test = train_test_split(inputs, targets, test_size=0.1)
```

[103]:
```
# Create and train the model
model = LinearRegression().fit(inputs_train, targets_train)

# Generate predictions
predictions_test = model.predict(inputs_test)

# Compute loss to evalute the model
loss = rmse(targets_test, predictions_test)
print('Test Loss:', loss)
```

```
Test Loss: 5672.662886480844
```

[104]:
```
# Generate predictions
predictions_train = model.predict(inputs_train)

# Compute loss to evalute the model
loss = rmse(targets_train, predictions_train)
print('Train Loss:', loss)
```

```
Train Loss: 6083.352524025472
```

[ ]: