Medical Insurance Cost Prediction

By Hakeem Lawrence

Dataset Link: https://www.kaggle.com/datasets/mirichoi0218/insurance

Importing Libraries

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
```

Data Inspection

In [2]:

```
insurance = pd.read_csv('insurance.csv')
insurance.head(10)
```

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
5	31	female	25.740	0	no	southeast	3756.62160
6	46	female	33.440	1	no	southeast	8240.58960
7	37	female	27.740	3	no	northwest	7281.50560
8	37	male	29.830	2	no	northeast	6406.41070
9	60	female	25.840	0	no	northwest	28923.13692

In [3]:

```
insurance.shape
```

Out[3]:

```
(1338, 7)
```

In [4]:

```
insurance.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
```

In [5]:

```
insurance.isna().sum()
```

Out[5]:

```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

Descriptive Statistics

In [6]:

```
insurance.describe(include=[np.number]).round(3)
```

Out[6]:

	age	bmi	children	charges
count	1338.000	1338.000	1338.000	1338.000
mean	39.207	30.663	1.095	13270.422
std	14.050	6.098	1.205	12110.011
min	18.000	15.960	0.000	1121.874
25%	27.000	26.296	0.000	4740.287
50%	39.000	30.400	1.000	9382.033
75%	51.000	34.694	2.000	16639.913
max	64.000	53.130	5.000	63770.428

In [7]:

```
insurance.columns
```

Out[7]:

```
Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges'], dtype='object')
```

In [8]:

```
#categorical column names
ins_cat_columns = ['sex', 'smoker', 'region']
```

In [9]:

```
# count number of each category in columns

for i in ins_cat_columns:
    print(f'Count of value in {i}')
    print('-'*25)
    print((insurance[i].value_counts(normalize=True).round(3)))
    print(' '*15)
```

```
Count of value in sex
-------------------------
male      0.505
female    0.495
Name: sex, dtype: float64

Count of value in smoker
-------------------------
no     0.795
yes    0.205
```

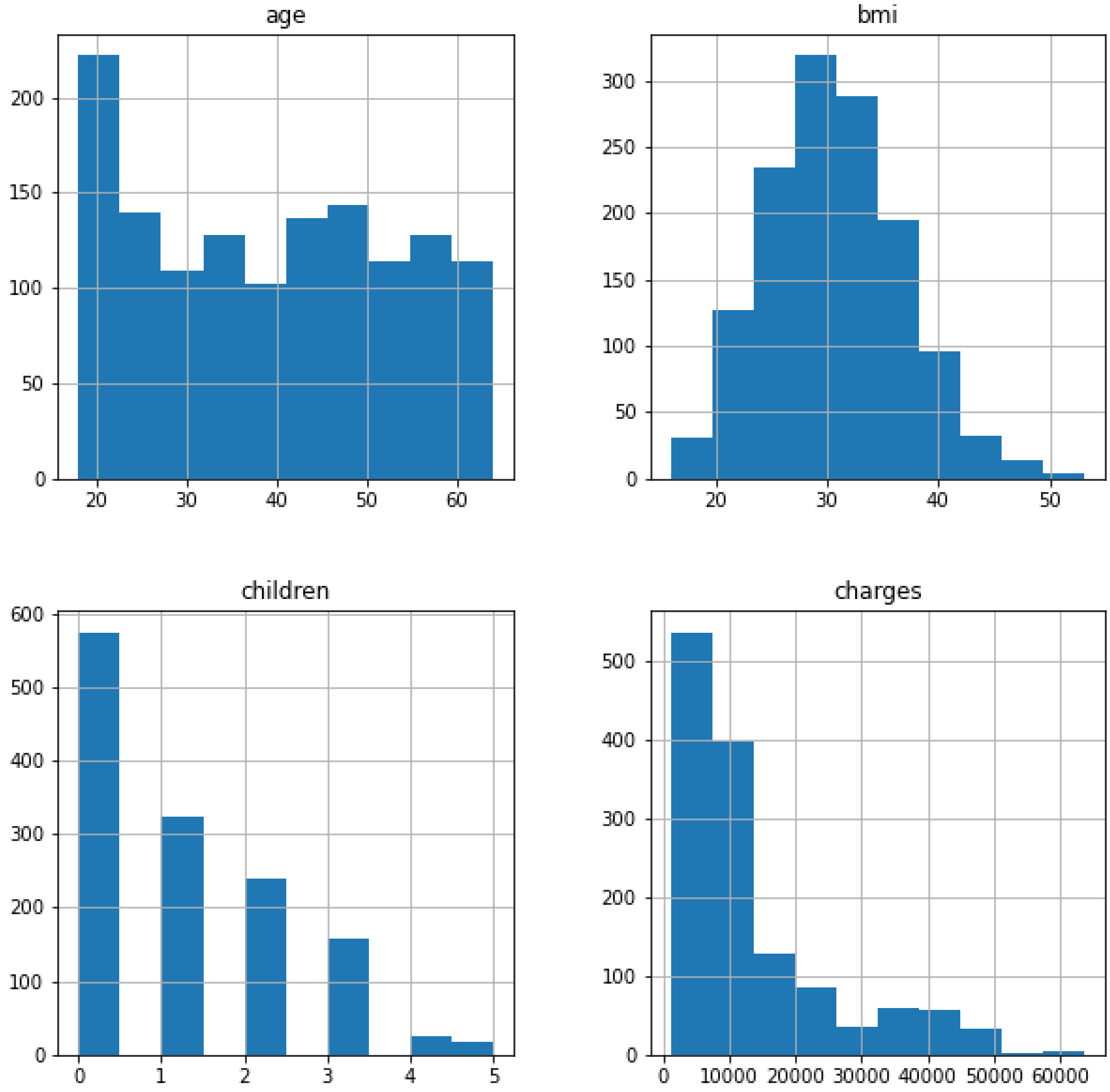
```
Name: smoker, dtype: float64

Count of value in region
-------------------------
southeast    0.272
southwest    0.243
northwest    0.243
northeast    0.242
Name: region, dtype: float64
```

In [10]:

```
#distribution plots of numberical variables

insurance.hist(figsize=(10,10))
```

Out[10]:

```
array([[<AxesSubplot:title={'center':'age'}>,
        <AxesSubplot:title={'center':'bmi'}>],
       [<AxesSubplot:title={'center':'children'}>,
        <AxesSubplot:title={'center':'charges'}>]], dtype=object)
```



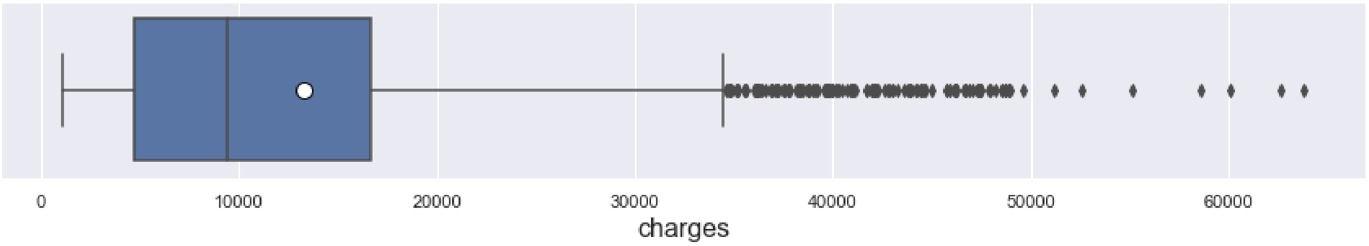
In [11]:

```
#histogram of target variable

sns.set(rc={'figure.figsize':(15,2)})
sns.boxplot(data=insurance, x='charges',
            showmeans=True,
            meanprops={'marker':'o',
                       'markerfacecolor':'white',
                       'markeredgecolor':'black',
                       'markersize': '10'})
```

```
plt.xlabel('charges', fontsize=16)
```

Out[11]:

```
Text(0.5, 0, 'charges')
```

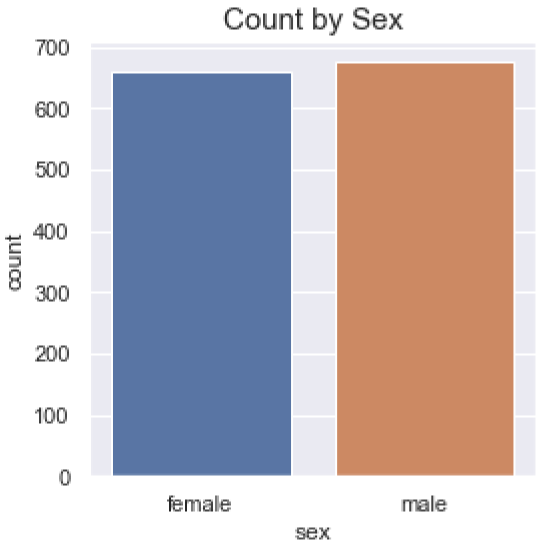


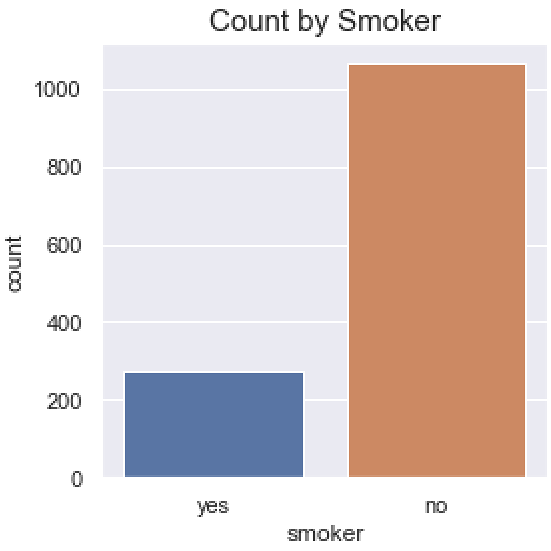
In [12]:

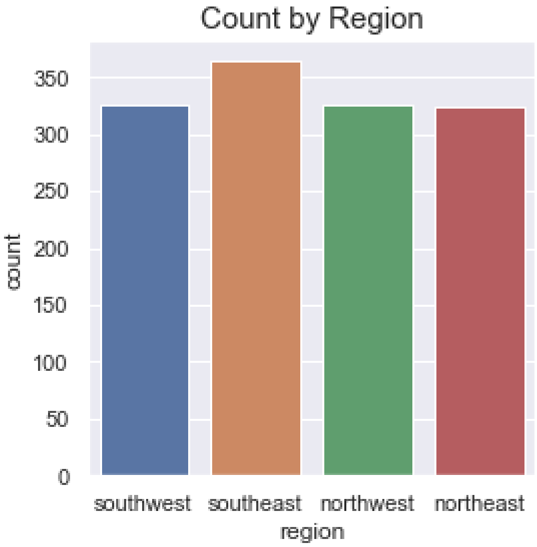
```
#creating function to display counts of categorical variables

def counts(var):
    plt.figure(figsize=(4,4))
    for i in ins_cat_columns:
        sns.countplot(data=insurance, x=var)
        plt.title(f'Count by {var.title()}', size=15)
```

In [13]:

```
#showing counts of categorical variables

for col in ins_cat_columns:
    counts(col)
```







In [14]:

```
#creating function to display distribution of variables by column

def distributions(variable):
    for i in ins_cat_columns:
        sns.displot(data=insurance, x=variable, hue=i,
                    height=4, aspect=2, kind='kde')

        plt.xlabel(variable, size=12)
        plt.ylabel('Count', size=12)
        plt.xticks(size=12)
        plt.title(f'Distribution of {variable.title()} by {i.title()}', size=16)
        plt.show()
```

In [15]:

```
insurance.columns
```

Out[15]:

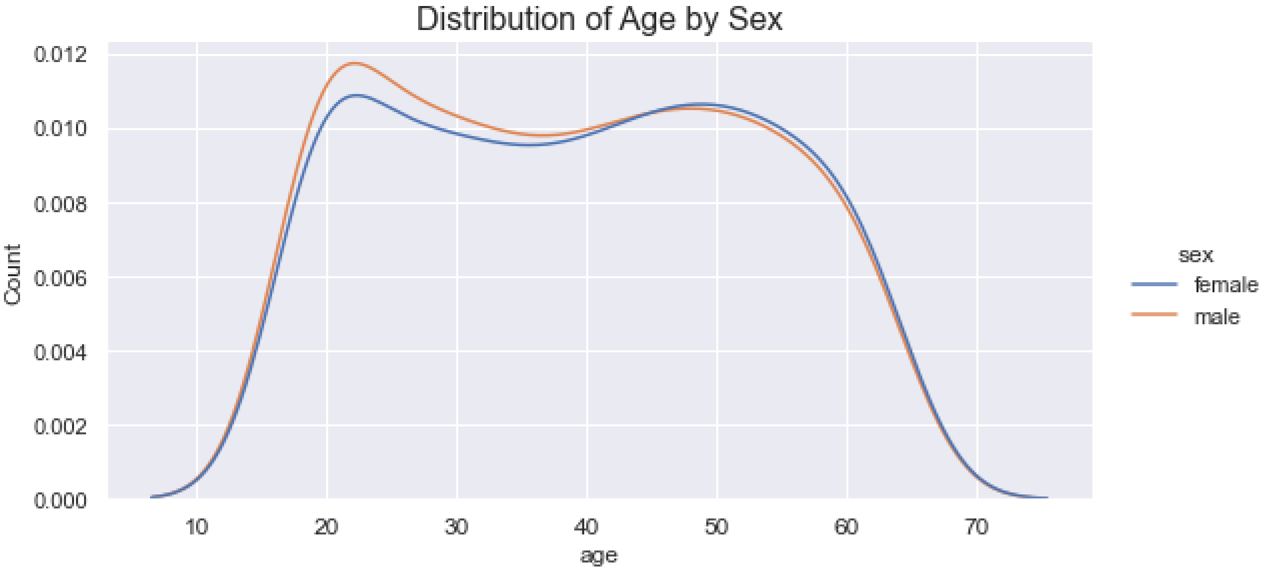
```
Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges'], dtype='object')
```

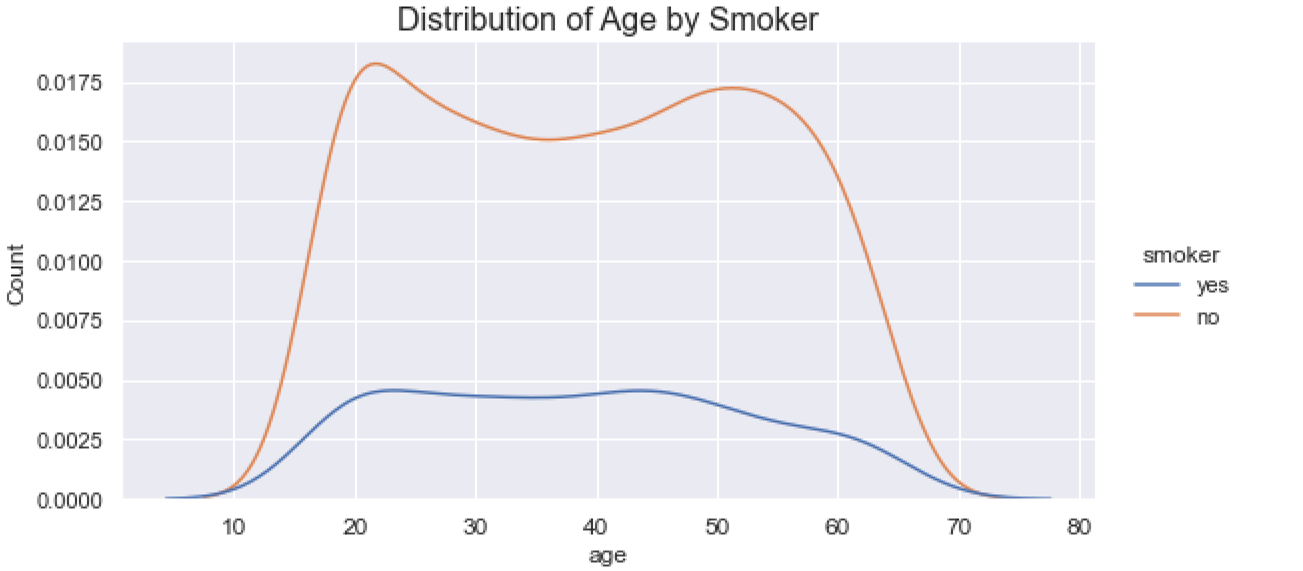
In [16]:

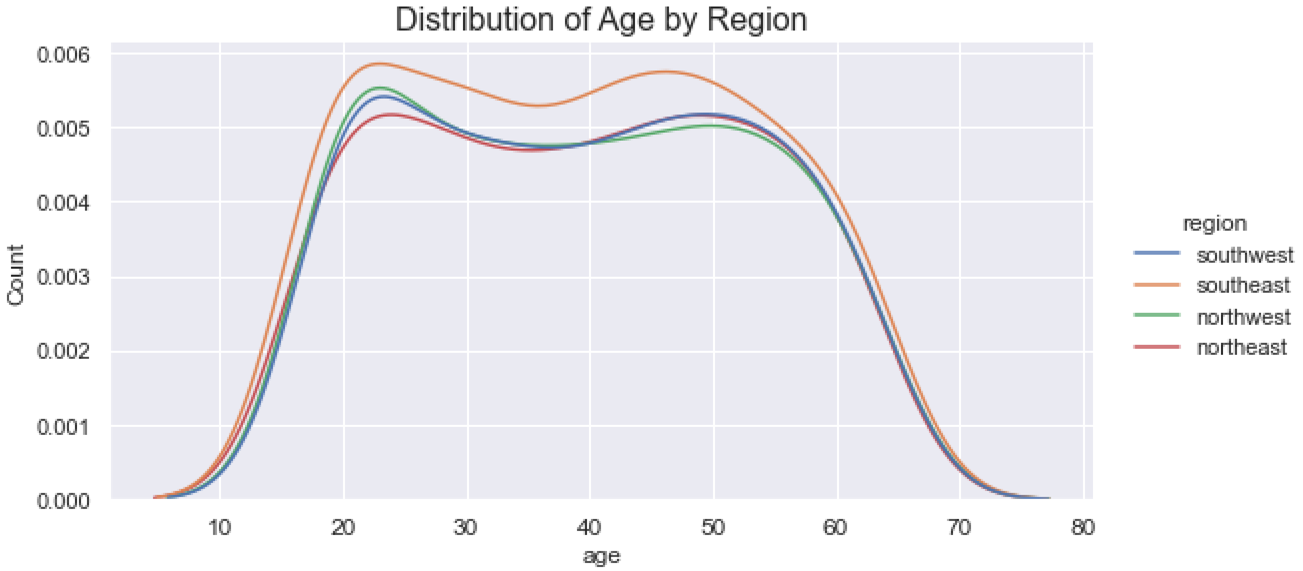
```
ins_num_columns= ['age', 'bmi', 'children', 'charges']
```

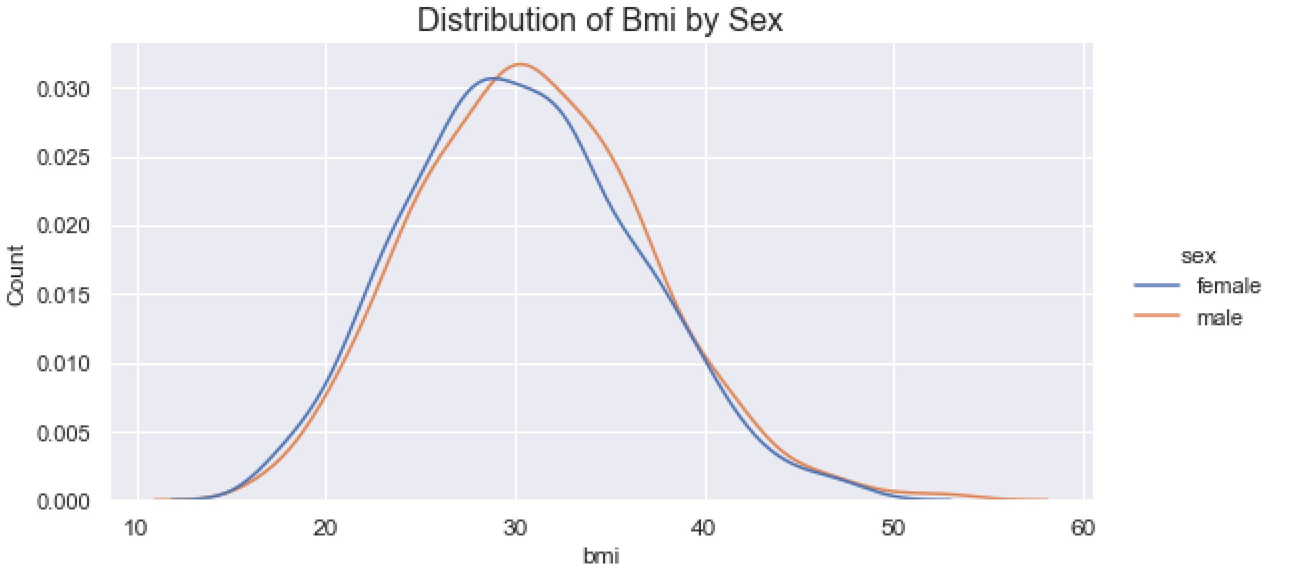
In [17]:

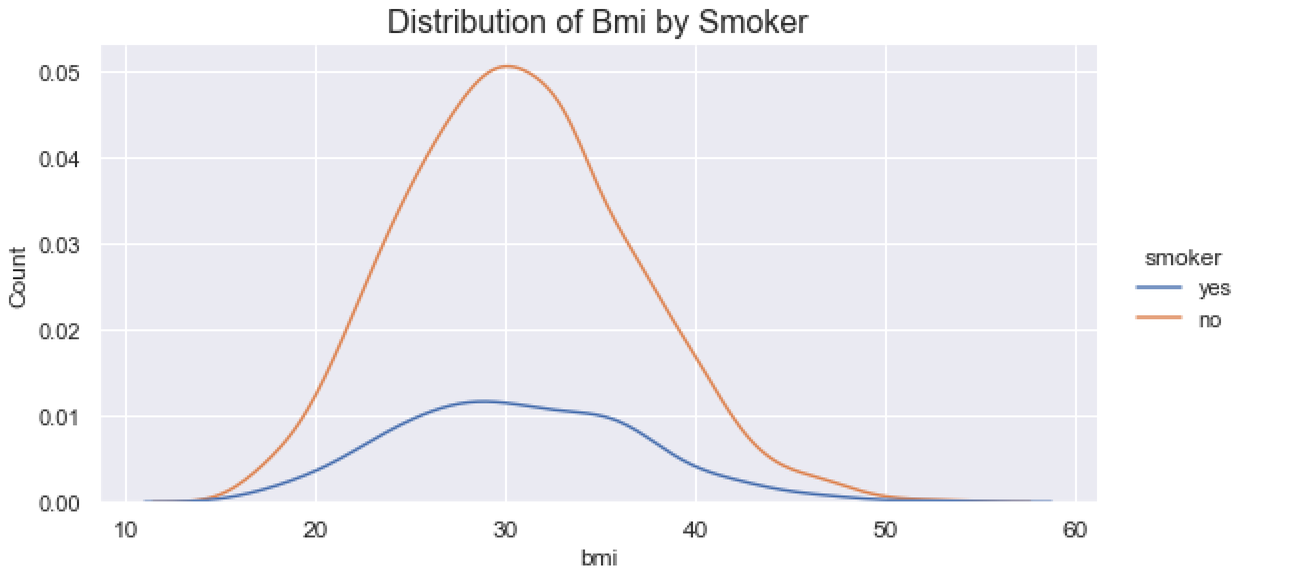
```
for col in ins_num_columns:
    distributions(col)
```

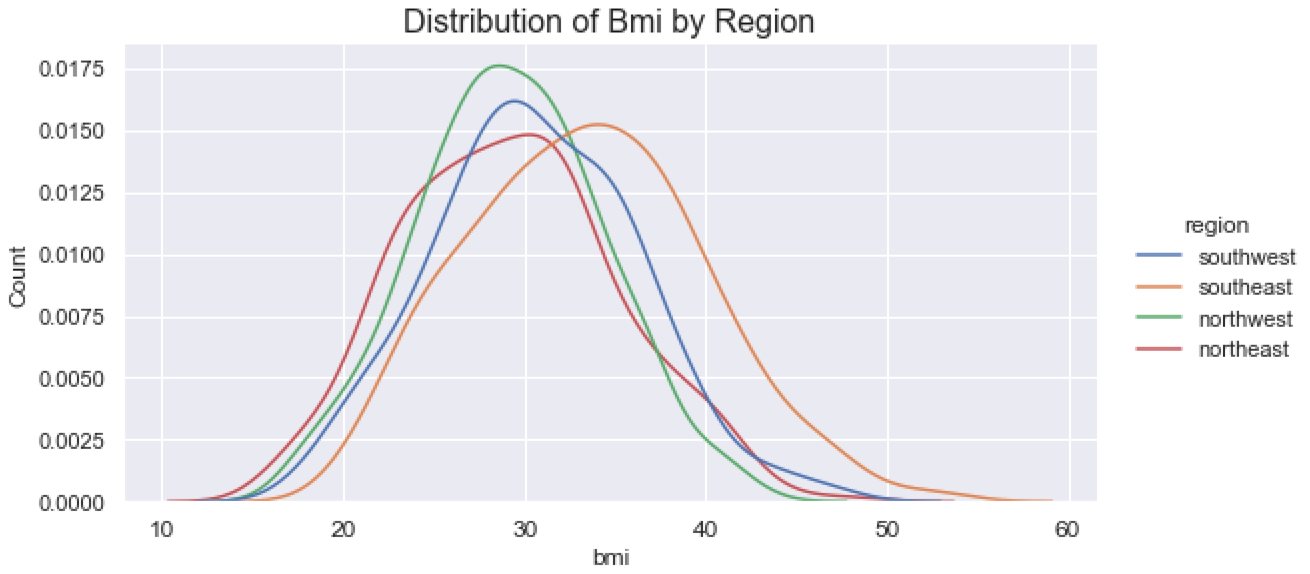


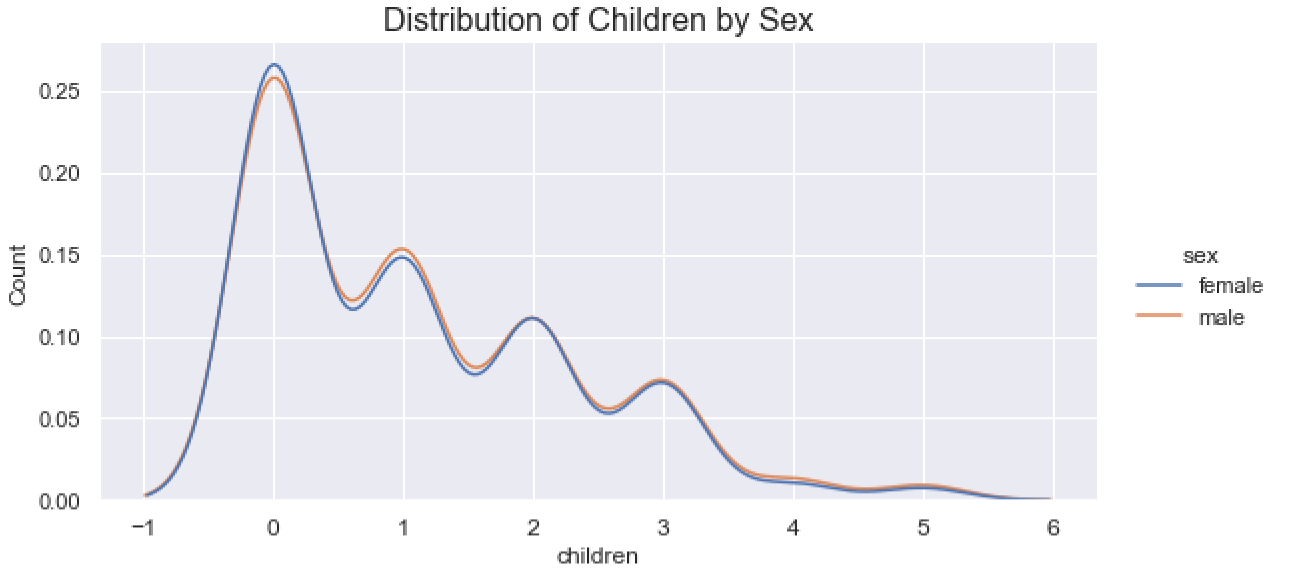

Distribution of Age by Smoker


Distribution of Age by Region

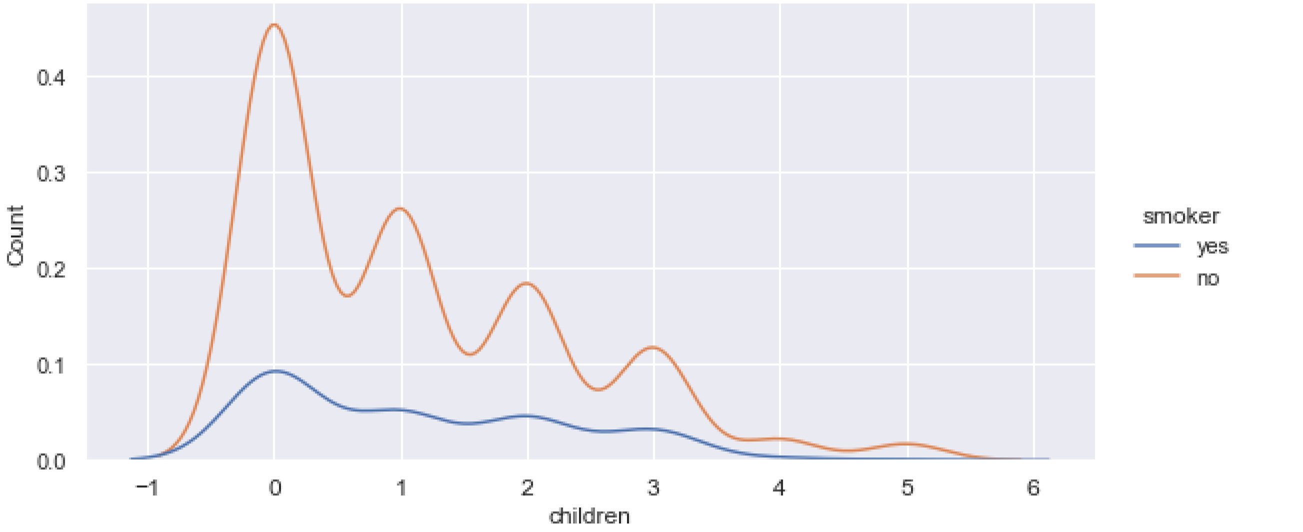

Distribution of Bmi by Sex


Distribution of Bmi by Smoker
Count
0.00
0.01
0.02
0.03
0.04
0.05
10
20
30
40
50
60
bmi
smoker
yes
no

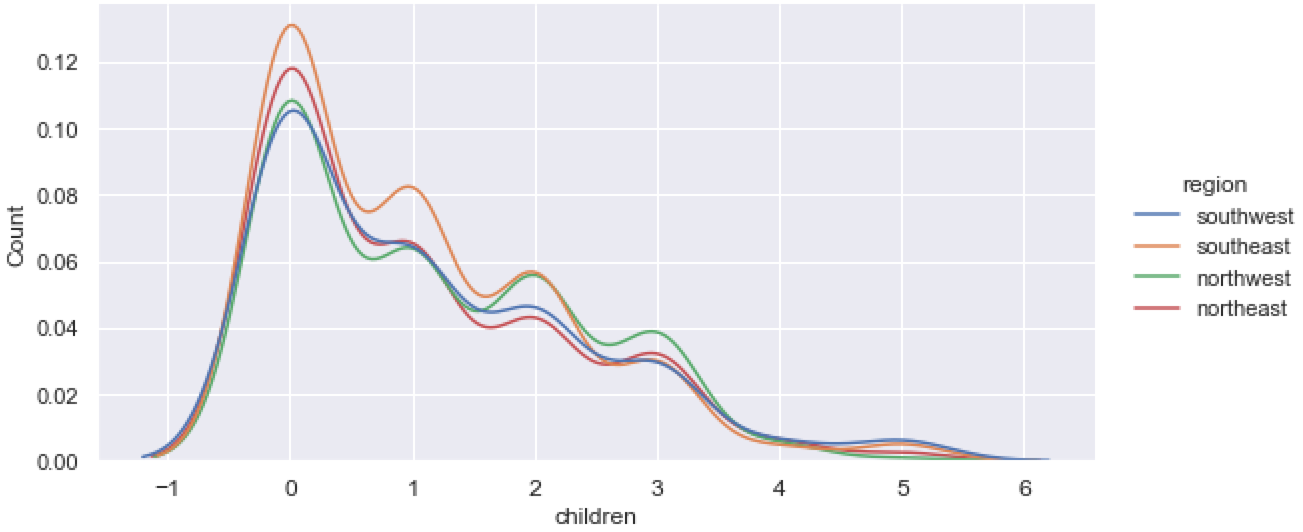

Distribution of Bmi by Region
Count
0.0000
0.0025
0.0050
0.0075
0.0100
0.0125
0.0150
0.0175
10
20
30
40
50
60
bmi
region
southwest
southeast
northwest
northeast


Distribution of Children by Sex
Count
0.00
0.05
0.10
0.15
0.20
0.25
−1
0
1
2
3
4
5
6
children
sex
female
male

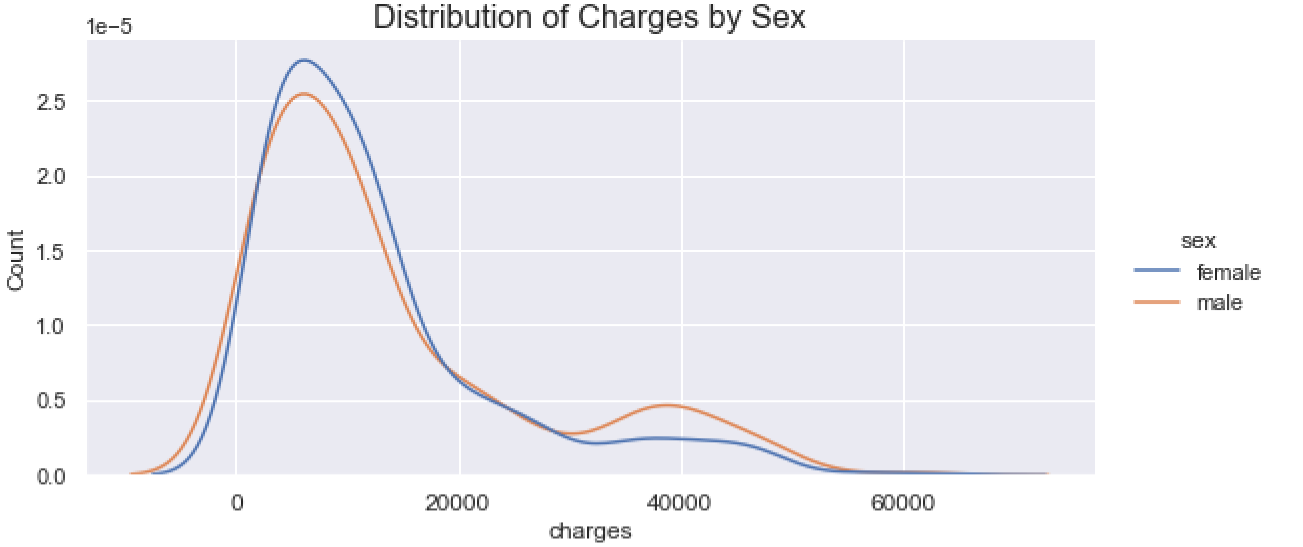
Distribution of Children by Smoker

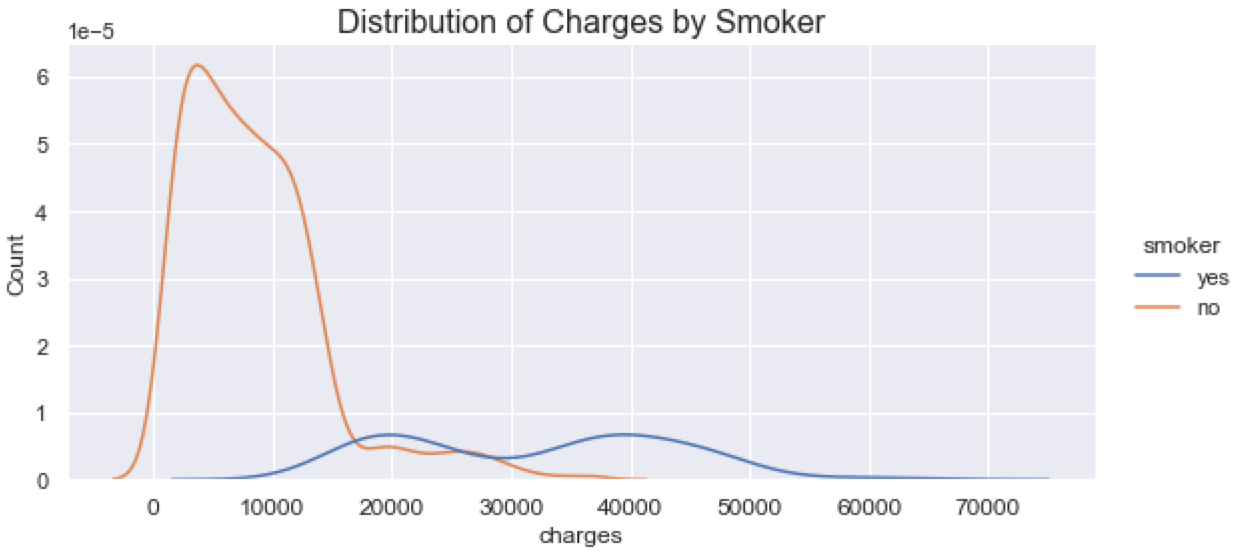


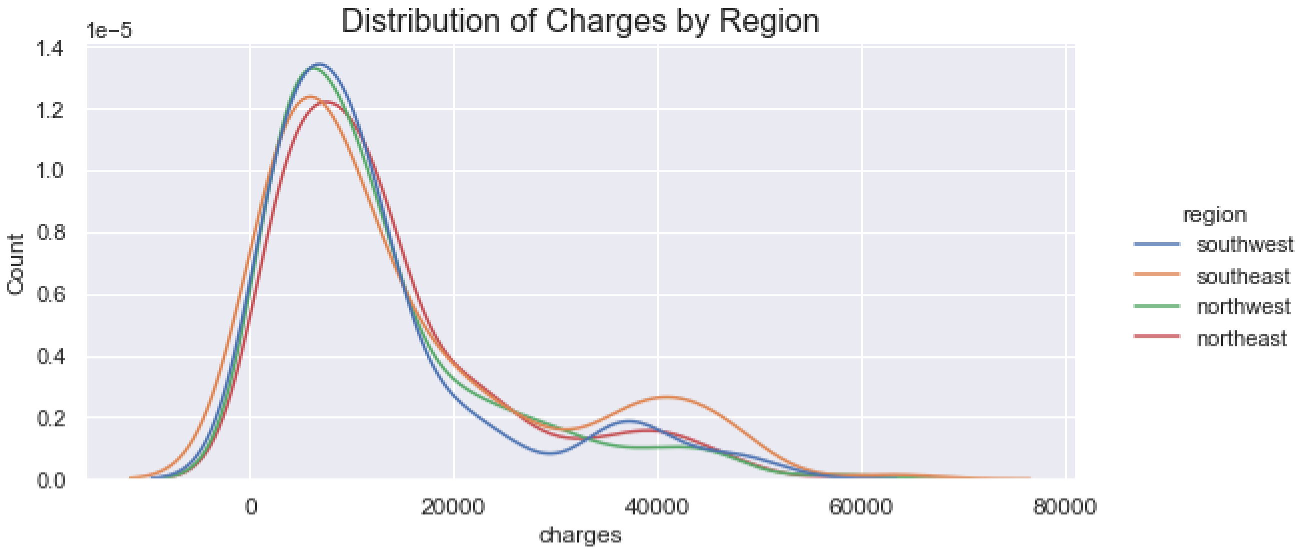
Distribution of Children by Region



Distribution of Charges by Sex

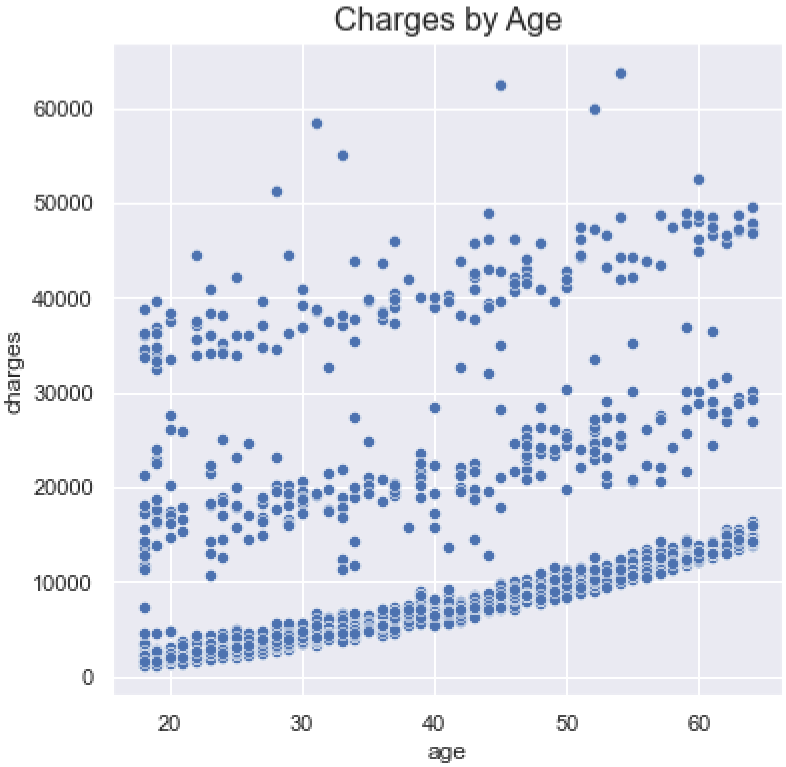


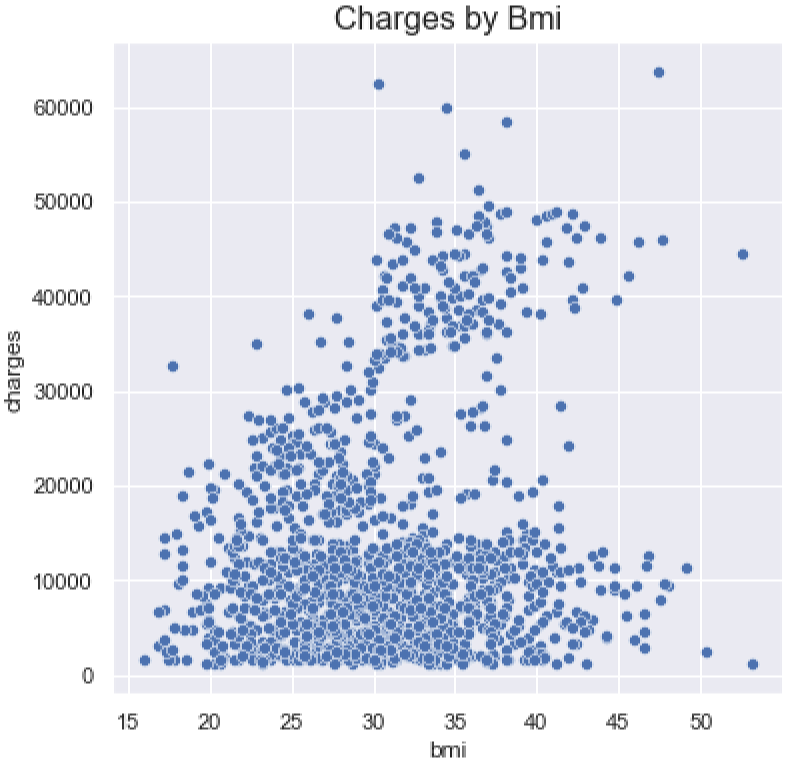


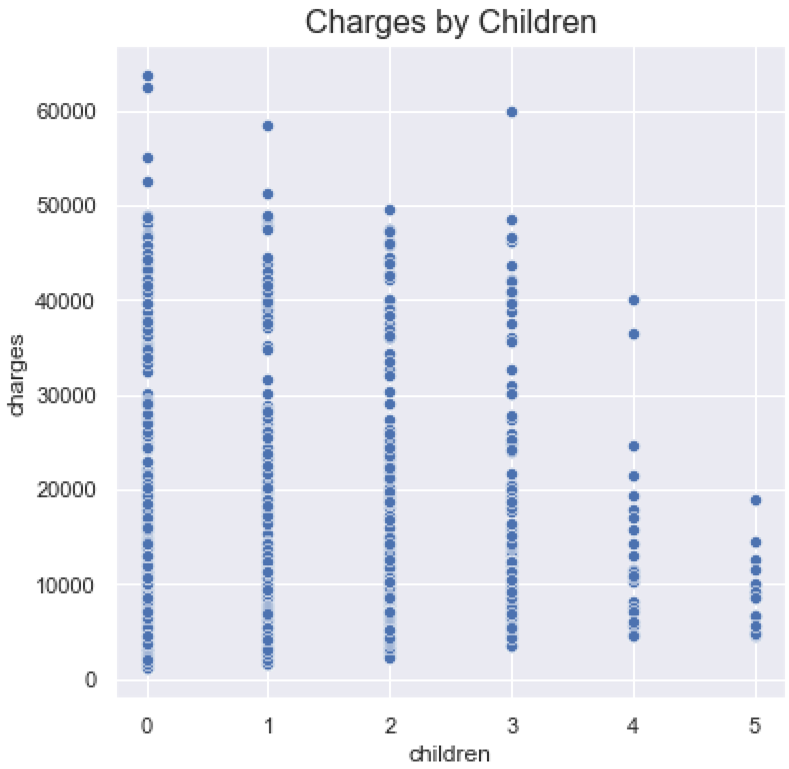


In [18]:

```
#creating function to display relationship between quant variables

def scatter(var):
    if var != 'charges':
        plt.figure(figsize=(6,6))
        sns.scatterplot(x=var, y='charges', data=insurance)
        plt.title(f'Charges by {var.title()}', size=16)


for i in ins_num_columns:
    scatter(i)
```


Charges by Age
charges
60000
50000
40000
30000
20000
10000
0
20
30
40
50
60
age


Charges by Bmi
charges
60000
50000
40000
30000
20000
10000
0
15
20
25
30
35
40
45
50
bmi

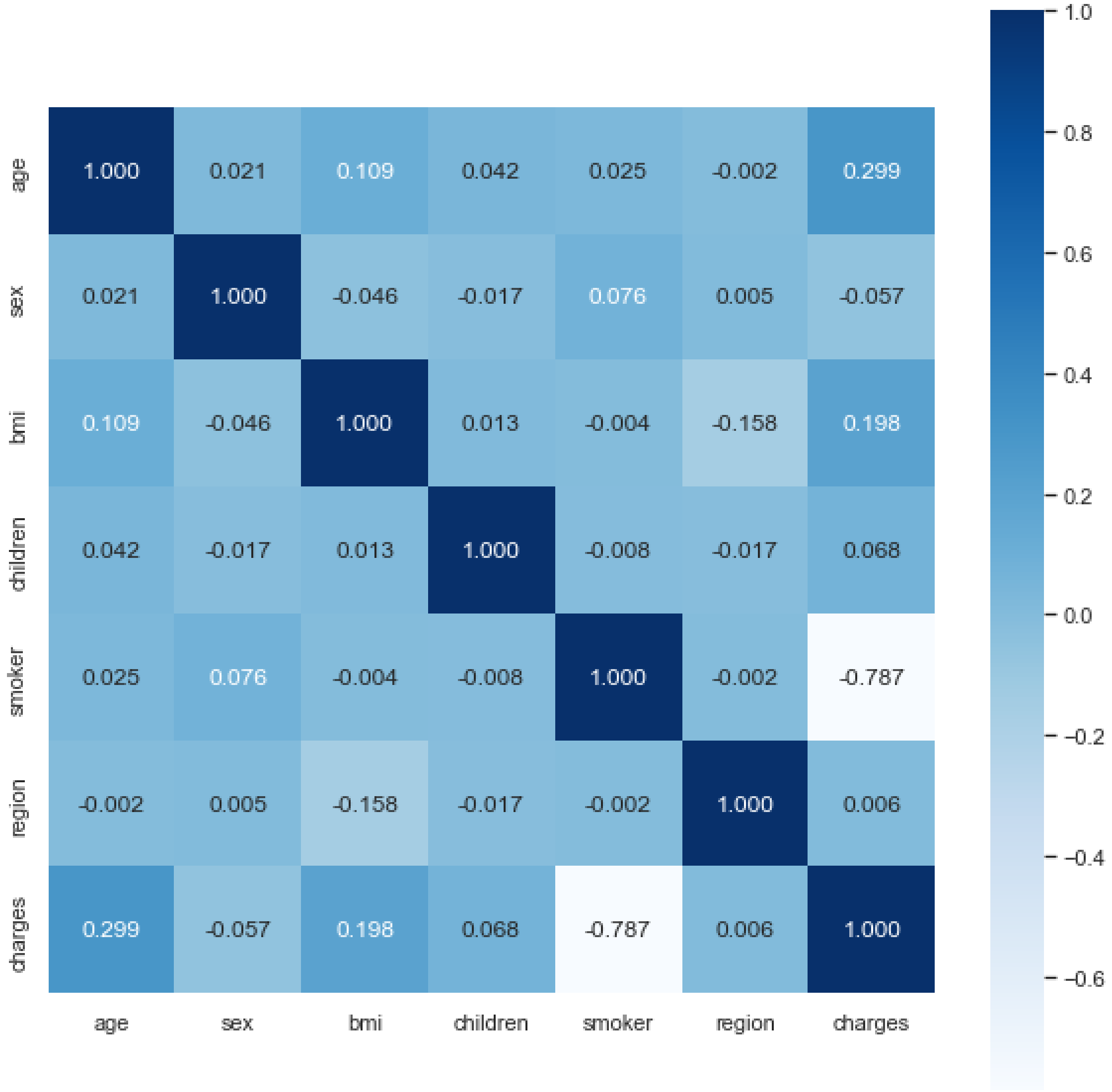


In [31]:

```
#viewing correlation between variables

corr= insurance.corr()

plt.figure(figsize=(10,10))
sns.heatmap(corr,
            cbar=True,
           square=True, fmt='.3f',
            annot=True,
            annot_kws={'size':12},
            cmap='Blues')
```

Out[31]:

```
<AxesSubplot:>
```



	age	sex	bmi	children	smoker	region	charges
age	1.000	0.021	0.109	0.042	0.025	-0.002	0.299
sex	0.021	1.000	-0.046	-0.017	0.076	0.005	-0.057
bmi	0.109	-0.046	1.000	0.013	-0.004	-0.158	0.198
children	0.042	-0.017	0.013	1.000	-0.008	-0.017	0.068
smoker	0.025	0.076	-0.004	-0.008	1.000	-0.002	-0.787
region	-0.002	0.005	-0.158	-0.017	-0.002	1.000	0.006
charges	0.299	-0.057	0.198	0.068	-0.787	0.006	1.000

Pre-Processing

Encoding Categorical Features

In [20]:
```
#encoding insurance column
insurance.replace({'sex':{'male':0,'female':1}}, inplace=True)

#encoding the smoker column
insurance.replace({'smoker':{'yes':0,'no':1}}, inplace=True)

#encoding region column
insurance.replace({'region':{'southwest':0,'southeast':1, 'northwest':2, 'northeast':3}}
```

Splitting Features from Target

In [21]:
```
X= insurance.drop(columns='charges', axis=1)
Y= insurance['charges']
```

Training Model

In [22]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size= .2, random_state=2)
```

In [23]:
```
print(X.shape, X_train.shape, X_test.shape)
```

```
(1338, 6) (1070, 6) (268, 6)
```

In [24]:

```
linearmodel = LinearRegression()
```

In [25]:

```
linearmodel.fit(X_train, Y_train)
```

Out[25]:

```
▾ LinearRegression
LinearRegression()
```

Model Evaluation

In [26]:

```
#predicting on training data

train_predict = linearmodel.predict(X_train)
```

In [27]:

```
#R2 value

r2_train = metrics.r2_score(Y_train, train_predict)
print('R2 value:', r2_train)
```

```
R2 value: 0.7519923667088932
```

In [28]:

```
# predicting on test data

test_predict = linearmodel.predict(X_test)
```

In [29]:

```
#R2 value

r2_test = metrics.r2_score(Y_test, test_predict)
print('R2 value:', r2_test)
```

```
R2 value: 0.7445422986536503
```

Prediction System

In [30]:

```
input_data = (31,1,25.740,0,1,1) #charge = 3756.62160


#changing input data to np array

input_np = np.asarray(input_data)

#reshape np array

input_reshaped = input_np.reshape(1,-1)

prediction = linearmodel.predict(input_reshaped).round(2)
# print(prediction)

charge = 3756.62160
diff = (prediction - charge).round(2)
# print(diff)


print(f"The predicted insurance cost is ${prediction[0]}\n")
print(f"The prediction is off by ${diff[0]} from the actual value")
```

```
The predicted insurance cost is $3911.45

The prediction is off by $154.83 from the actual value
```

```
C:\Users\hakee\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not
have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```