Insurance Dataset

Linear Regression

Problem Statement: To find how charges are varying based on the selected features.

In [1]:
```
# importing the necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [79]:
```
# Data Collection-reading the file
df=pd.read_csv(r"C:\Users\yoshitha lakshmi\OneDrive\Desktop\python\insurance.csv")
df
```

Out[79]:

	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

In [3]:
```
# Data cleaning and Preprocessing
df.head()
```

Out[3]:

	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 [4]:
```
1 df.tail()
```

Out[4]:

	age	sex	bmi	children	smoker	region	charges
1333	50	male	30.97	3	no	northwest	10600.5483
1334	18	female	31.92	0	no	northeast	2205.9808
1335	18	female	36.85	0	no	southeast	1629.8335
1336	21	female	25.80	0	no	southwest	2007.9450
1337	61	female	29.07	0	yes	northwest	29141.3603

In [5]:
```
1 df.describe()
```

Out[5]:

	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 [6]:
```
1 df.shape
```

Out[6]: (1338, 7)

In [7]:
```
1 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
```

In [8]:
```
1 df['sex'].value_counts()
```

Out[8]:
```
sex
male      676
female    662
Name: count, dtype: int64
```

In [9]:
```
convert={"sex":{"female":1,"male":2}}
df=df.replace(convert)
df
```

Out[9]:

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

1338 rows × 7 columns

In [80]:
```
convert={'smoker':{'yes':1,'no':2}}
df=df.replace(convert)
df
```

Out[80]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	1	southwest	16884.92400
1	18	male	33.770	1	2	southeast	1725.55230
2	28	male	33.000	3	2	southeast	4449.46200
3	33	male	22.705	0	2	northwest	21984.47061
4	32	male	28.880	0	2	northwest	3866.85520
...	...	...	...	...	...	...	...
1333	50	male	30.970	3	2	northwest	10600.54830
1334	18	female	31.920	0	2	northeast	2205.98080
1335	18	female	36.850	0	2	southeast	1629.83350
1336	21	female	25.800	0	2	southwest	2007.94500
1337	61	female	29.070	0	1	northwest	29141.36030

1338 rows × 7 columns

In [11]:
```
df=df.drop('region',axis=1)
```

In [12]:
```
df=df.drop('children',axis=1)
```

In [13]:
```
features=df.columns[0:2]
```

In [14]:
```
target=df.columns[-1]
```

In [15]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [16]:
```
x=np.array(df[features])
y=np.array(df[target])
```

In [17]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.7)
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_train,y_train))
```

```
0.08396311752167784
```

In [18]:
```
print(regr.intercept_)
```

```
1440.2203473920035
```

In [19]:
```
coeff_df=pd.DataFrame(regr.coef_)
coeff_df
```

Out[19]:

	0
0	250.860087
1	1409.875055

Conclusion

The accuracy for this Dataset is very low while using LinearRegression. Accuracy = 0.0835

Ridge Regression

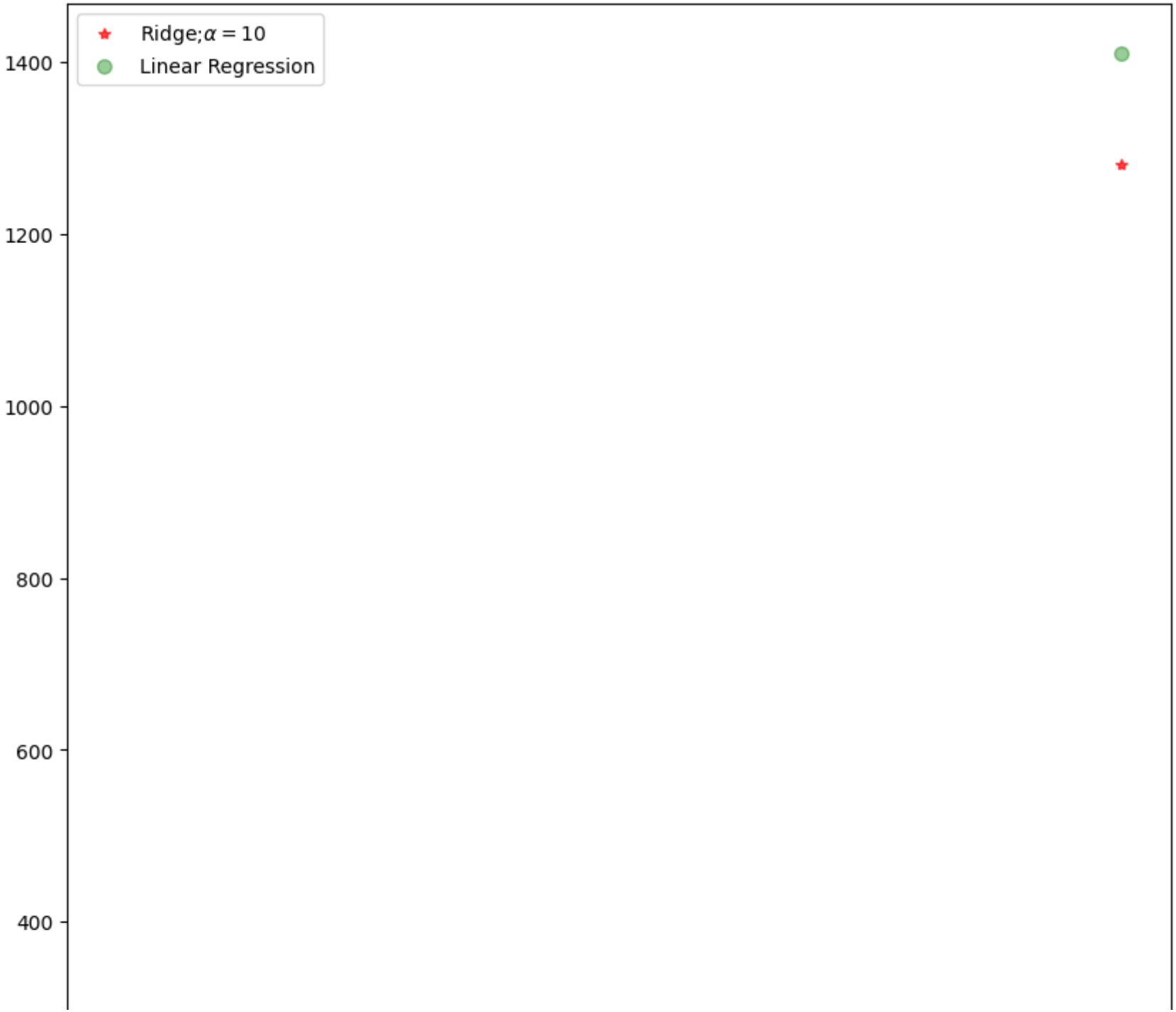
In [20]:
```
from sklearn.linear_model import Ridge,Lasso,RidgeCV
```

In [21]:
```
ridgeReg = Ridge(alpha=10)
ridgeReg.fit(x_train,y_train)
train_score_ridge = ridgeReg.score(x_train,y_train)
test_score_ridge = ridgeReg.score(x_test,y_test)

print('\nRidge Model\n')
print('Train score for ridge model is {}'.format(train_score_ridge))
print('Test score for ridge model is{}'.format(test_score_ridge))
```

```
Ridge Model

Train score for ridge model is 0.08393600452093364
Test score for ridge model is0.09723547364784435
```

In [22]:

```
plt.figure(figsize = (10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=6,color='red',label=r'Ridge;$\
plt.plot(features,regr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear Regr
plt.xticks(rotation=90)
plt.legend()
plt.show()
```


Ridge;$\alpha = 10$
Linear Regression
1400
1200
1000
800
600
400

Conclusion

For Ridge Regression also the Accuracy value is very low.

Train score for ridge model is 0.08393600452093364

Test score for ridge model is0.09723547364784435

Logostic Regression

Problem Statement: To find smokers count based on the features - sex, age.

In [23]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

In [24]:
```
df=pd.read_csv(r"C:\Users\yoshitha lakshmi\OneDrive\Desktop\python\insurance.csv")
df
```

Out[24]:

	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

In [25]:
```
features=df.columns[0:3]
```

In [26]:
```
target=df.columns[-3]
```

In [27]:
```
from sklearn.datasets import load_digits
digits=load_digits()
```

In [28]:
```
print(digits.data.shape)
print(digits.target.shape)
```

```
(1797, 64)
(1797,)
```

In [29]:
```
x_train,x_test,y_train,y_test=train_test_split(digits.data,digits.target,test_size=0.7,random_state=2)
lor=LogisticRegression(max_iter=10000)
lor.fit(x_train,y_train)
```

Out[29]:
```
LogisticRegression(max_iter=10000)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [30]:
```
score=lor.score(x_test,y_test)
print(score)
```

```
0.9523052464228935
```

Conclusion

Using Logistic Regression score is little high compared to Linear Regression. So, further process continued in Logistic Regression. Accuracy = 0.9523052464228935

Decision Tree

In [31]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

In [32]:
```
df=pd.read_csv(r"C:\Users\yoshitha lakshmi\OneDrive\Desktop\python\insurance.csv")
df
```

Out[32]:

	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

In [33]:
```
df['sex'].value_counts()
```

Out[33]:
```
sex
male      676
female    662
Name: count, dtype: int64
```

In [34]:
```
1 df['bmi'].value_counts()
```

Out[34]:
```
bmi
32.300    13
28.310     9
30.495     8
30.875     8
31.350     8
          ..
46.200     1
23.800     1
44.770     1
32.120     1
30.970     1
Name: count, Length: 548, dtype: int64
```

In [35]:
```
convert={'sex':{'female':0,'male':1}}
df=df.replace(convert)
df
```

Out[35]:

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

1338 rows × 7 columns

In [36]:
```
X=['age','sex']
y=['yes','no']
all_inputs=df[X]
all_classes=df['smoker']
```

In [37]:
```
X_train,x_test,y_train,y_test=train_test_split(all_inputs,all_classes,test_size=0.7)
```

In [38]:
```
clf=DecisionTreeClassifier(random_state=0)
```

In [56]:
```
clf.fit(X_train,y_train)
```

Out[56]:
```
DecisionTreeClassifier(random_state=0)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [57]:
```
score=clf.score(X_train,y_train)
print(score)
```

```
0.770573566084788
```

Random Forest

In [69]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt,seaborn as sns
from sklearn.model_selection import train_test_split
```

In [70]:
```
df=pd.read_csv(r"C:\Users\yoshitha lakshmi\OneDrive\Desktop\python\insurance.csv")
df
```

Out[70]:

	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

In [71]:
```
1 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
```

In [72]:
```
1 x=df.drop('smoker',axis=1)
2 y=df['smoker']
```

In [73]:
```
convert={'sex':{'female':0,'male':1}}
df=df.replace(convert)
df
```

Out[73]:

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

1338 rows × 7 columns

In [74]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(X_train,y_train)
```

Out[74]:
```
RandomForestClassifier()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [75]:
```
score=rfc.score(x_test,y_test)
print(score)
```

```
0.7502668089647813
```

In [83]:
```
params={'max_depth':[2,3,5,10,20],
       'min_samples_leaf':[5,10,20,50,100,200],
       'n_estimators':[10,25,30,50,100,200]}
```

In [85]:
```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rfc,param_grid=params,cv=2,scoring="accuracy")
grid_search.fit(X_train,y_train)
```

Out[85]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [2, 3, 5, 10, 20],
                         'min_samples_leaf': [5, 10, 20, 50, 100, 200],
                         'n_estimators': [10, 25, 30, 50, 100, 200]},
             scoring='accuracy')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [86]:
```
grid_search.best_score_
```

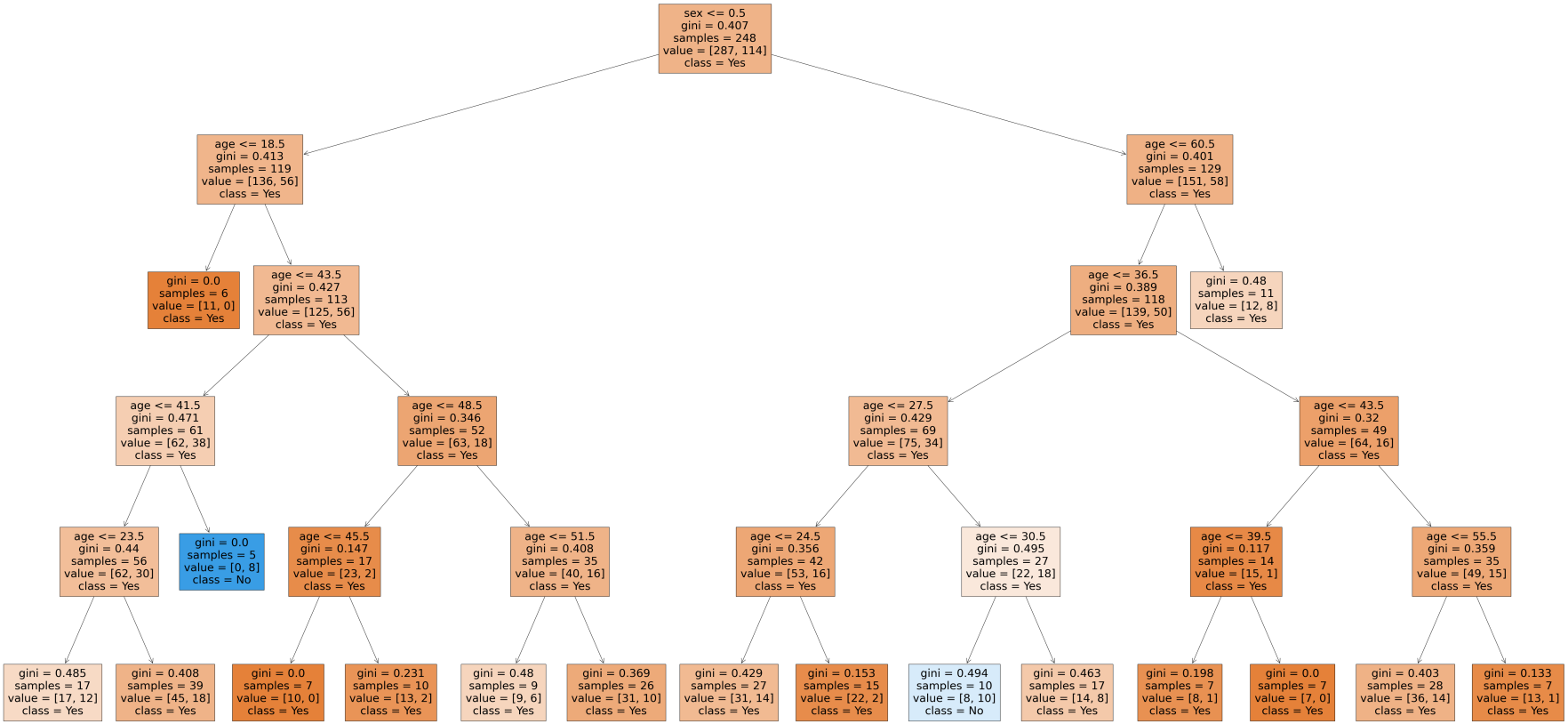
Out[86]:
```
0.7481343283582089
```

In [87]:
```
rf_best=grid_search.best_estimator_
```

In [88]:

```
from sklearn.tree import plot_tree
from sklearn.tree import DecisionTreeClassifier
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[5],feature_names=x.columns,class_names=['Yes','No'],filled=True)
```

Out[88]:
```
[Text(0.45535714285714285, 0.9166666666666666, 'sex <= 0.5\ngini = 0.407\nsamples = 248\nvalue = [287, 114]\nclass
= Yes'),
 Text(0.16071428571428573, 0.75, 'age <= 18.5\ngini = 0.413\nsamples = 119\nvalue = [136, 56]\nclass = Yes'),
 Text(0.125, 0.5833333333333334, 'gini = 0.0\nsamples = 6\nvalue = [11, 0]\nclass = Yes'),
 Text(0.19642857142857142, 0.5833333333333334, 'age <= 43.5\ngini = 0.427\nsamples = 113\nvalue = [125, 56]\nclass
= Yes'),
 Text(0.10714285714285714, 0.4166666666666667, 'age <= 41.5\ngini = 0.471\nsamples = 61\nvalue = [62, 38]\nclass =
Yes'),
 Text(0.07142857142857142, 0.25, 'age <= 23.5\ngini = 0.44\nsamples = 56\nvalue = [62, 30]\nclass = Yes'),
 Text(0.03571428571428571, 0.08333333333333333, 'gini = 0.485\nsamples = 17\nvalue = [17, 12]\nclass = Yes'),
 Text(0.10714285714285714, 0.08333333333333333, 'gini = 0.408\nsamples = 39\nvalue = [45, 18]\nclass = Yes'),
 Text(0.14285714285714285, 0.25, 'gini = 0.0\nsamples = 5\nvalue = [0, 8]\nclass = No'),
 Text(0.2857142857142857, 0.4166666666666667, 'age <= 48.5\ngini = 0.346\nsamples = 52\nvalue = [63, 18]\nclass = Y
es'),
 Text(0.21428571428571427, 0.25, 'age <= 45.5\ngini = 0.147\nsamples = 17\nvalue = [23, 2]\nclass = Yes'),
 Text(0.17857142857142858, 0.08333333333333333, 'gini = 0.0\nsamples = 7\nvalue = [10, 0]\nclass = Yes'),
 Text(0.25, 0.08333333333333333, 'gini = 0.231\nsamples = 10\nvalue = [13, 2]\nclass = Yes'),
 Text(0.35714285714285715, 0.25, 'age <= 51.5\ngini = 0.408\nsamples = 35\nvalue = [40, 16]\nclass = Yes'),
 Text(0.32142857142857145, 0.08333333333333333, 'gini = 0.48\nsamples = 9\nvalue = [9, 6]\nclass = Yes'),
 Text(0.39285714285714285, 0.08333333333333333, 'gini = 0.369\nsamples = 26\nvalue = [31, 10]\nclass = Yes'),
 Text(0.75, 0.75, 'age <= 60.5\ngini = 0.401\nsamples = 129\nvalue = [151, 58]\nclass = Yes'),
 Text(0.7142857142857143, 0.5833333333333334, 'age <= 36.5\ngini = 0.389\nsamples = 118\nvalue = [139, 50]\nclass =
Yes'),
 Text(0.5714285714285714, 0.4166666666666667, 'age <= 27.5\ngini = 0.429\nsamples = 69\nvalue = [75, 34]\nclass = Y
es'),
 Text(0.5, 0.25, 'age <= 24.5\ngini = 0.356\nsamples = 42\nvalue = [53, 16]\nclass = Yes'),
 Text(0.4642857142857143, 0.08333333333333333, 'gini = 0.429\nsamples = 27\nvalue = [31, 14]\nclass = Yes'),
 Text(0.5357142857142857, 0.08333333333333333, 'gini = 0.153\nsamples = 15\nvalue = [22, 2]\nclass = Yes'),
 Text(0.6428571428571429, 0.25, 'age <= 30.5\ngini = 0.495\nsamples = 27\nvalue = [22, 18]\nclass = Yes'),
 Text(0.6071428571428571, 0.08333333333333333, 'gini = 0.494\nsamples = 10\nvalue = [8, 10]\nclass = No'),
 Text(0.6785714285714286, 0.08333333333333333, 'gini = 0.463\nsamples = 17\nvalue = [14, 8]\nclass = Yes'),
 Text(0.8571428571428571, 0.4166666666666667, 'age <= 43.5\ngini = 0.32\nsamples = 49\nvalue = [64, 16]\nclass = Ye
s'),
 Text(0.7857142857142857, 0.25, 'age <= 39.5\ngini = 0.117\nsamples = 14\nvalue = [15, 1]\nclass = Yes'),
 Text(0.75, 0.08333333333333333, 'gini = 0.198\nsamples = 7\nvalue = [8, 1]\nclass = Yes'),
 Text(0.8214285714285714, 0.08333333333333333, 'gini = 0.0\nsamples = 7\nvalue = [7, 0]\nclass = Yes'),
 Text(0.9285714285714286, 0.25, 'age <= 55.5\ngini = 0.359\nsamples = 35\nvalue = [49, 15]\nclass = Yes'),
 Text(0.8928571428571429, 0.08333333333333333, 'gini = 0.403\nsamples = 28\nvalue = [36, 14]\nclass = Yes'),
 Text(0.9642857142857143, 0.08333333333333333, 'gini = 0.133\nsamples = 7\nvalue = [13, 1]\nclass = Yes'),
 Text(0.7857142857142857, 0.5833333333333334, 'gini = 0.48\nsamples = 11\nvalue = [12, 8]\nclass = Yes')]
```



In [91]:
```
imp_df=pd.DataFrame({"varname":X_train.columns,"Imp":rf_best.feature_importances_})
imp_df.sort_values(by="Imp",ascending=False)
```

Out[91]:

	varname	Imp
0	age	0.963054
1	sex	0.036946

Coclusion

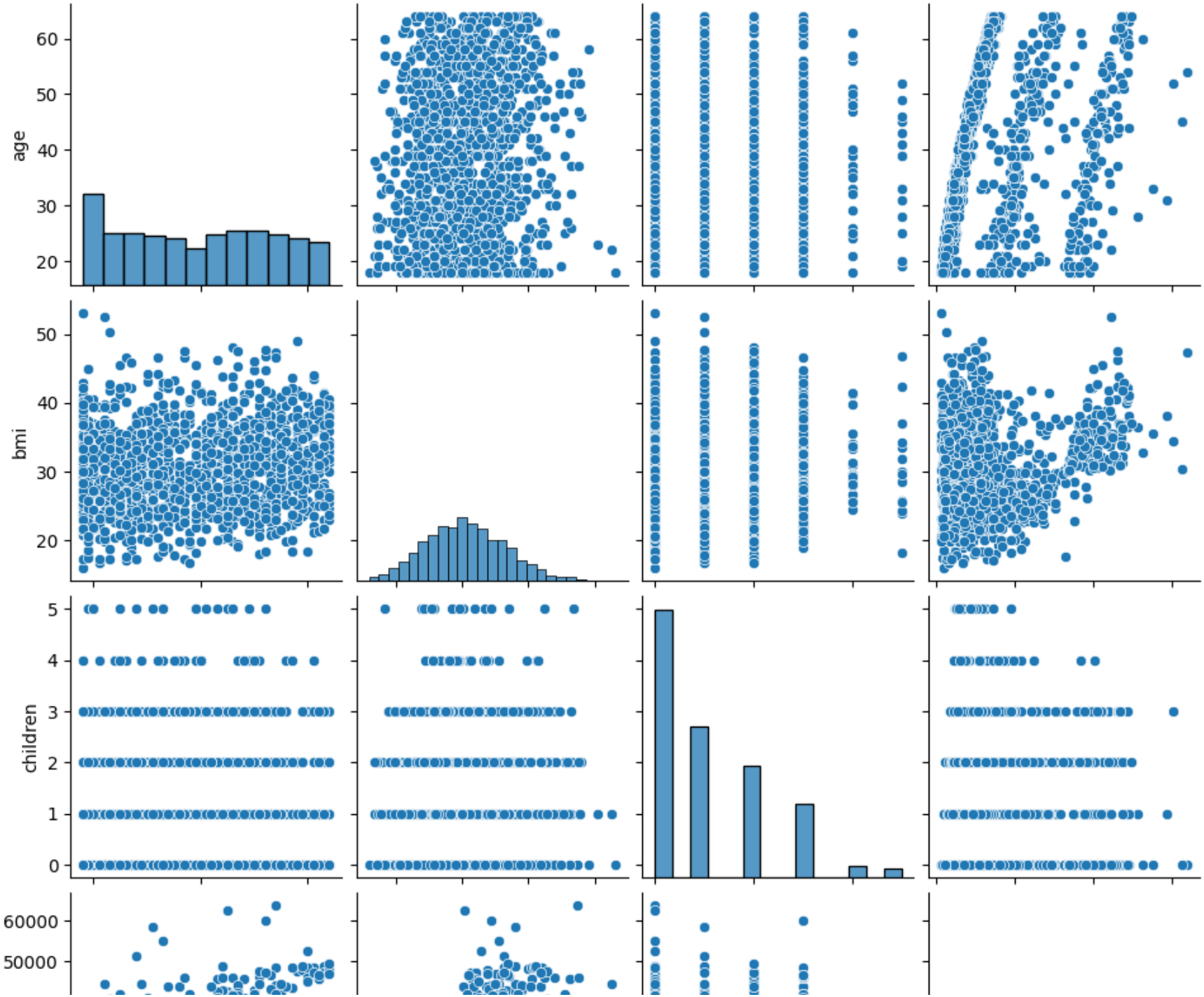
For both Decision Tree and Random Forest

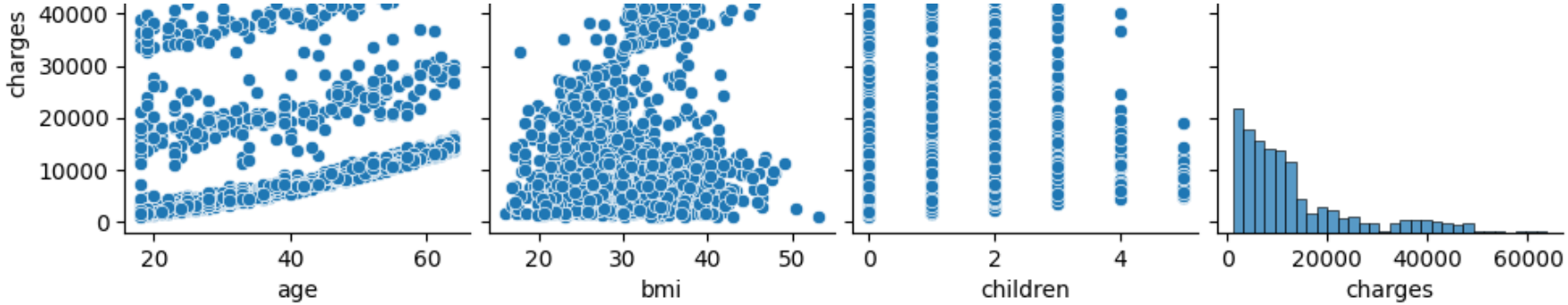
Based on our Problem Statement we classified data and build using Random Forest.

Exploratory Data Analysis

In [46]:
```
1 sns.pairplot(df)
```

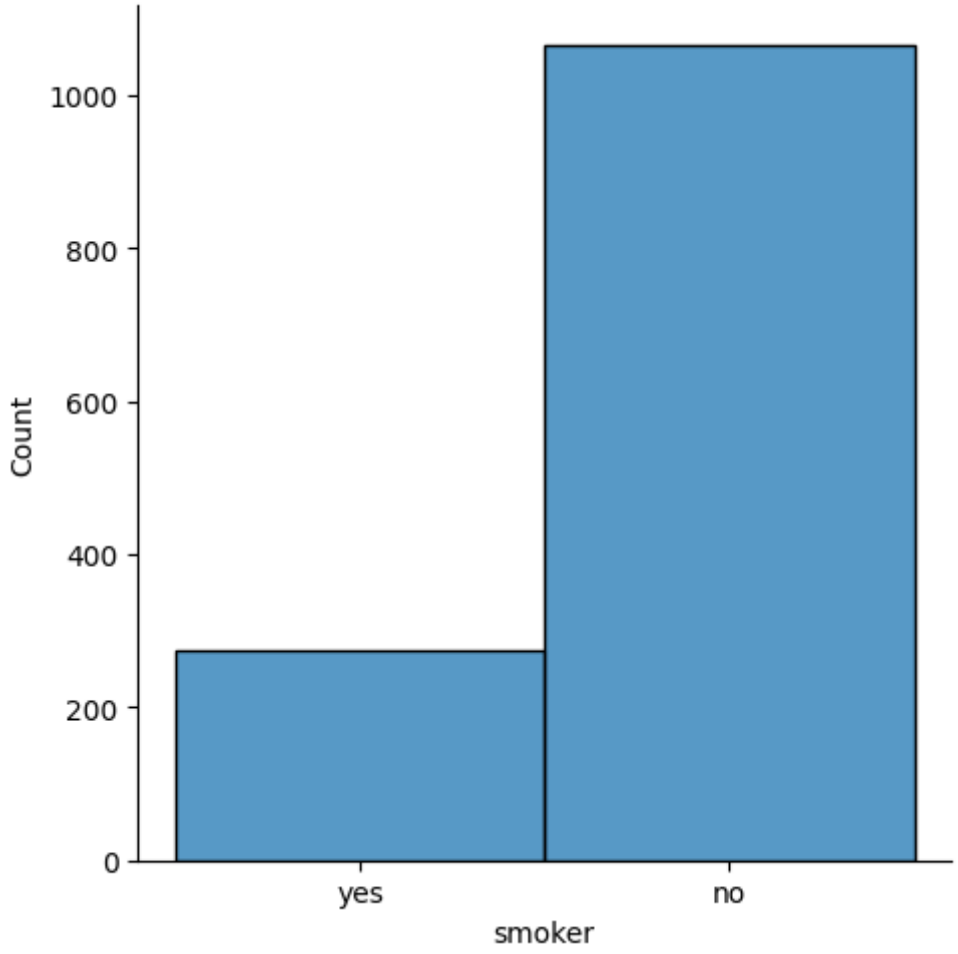
Out[46]:
```
<seaborn.axisgrid.PairGrid at 0x14cff9bb340>
```


age
bmi
children
60
50
40
30
20
50
40
30
20
5
4
3
2
1
0
60000
50000


charges
40000
30000
20000
10000
0
20
40
60
age
20
30
40
50
bmi
0
2
4
children
0
20000
40000
60000
charges

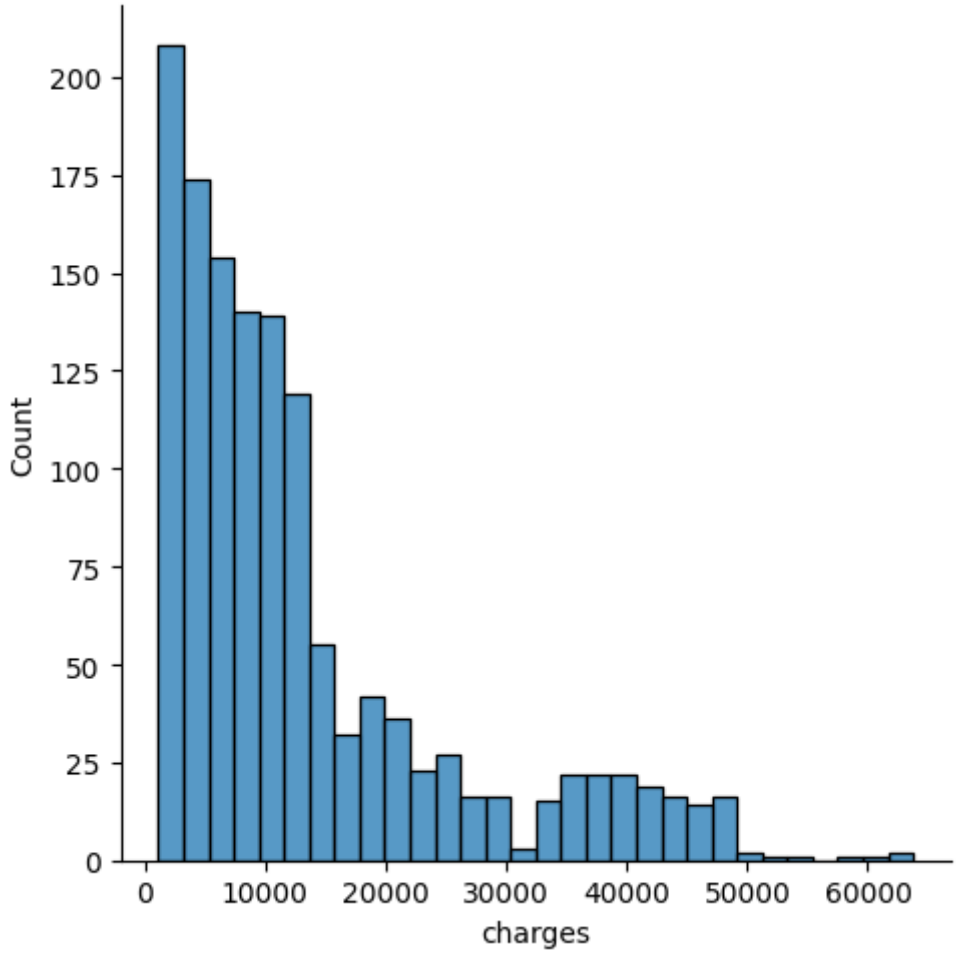
In [47]:
```
1 sns.displot(df['smoker'])
```

Out[47]: <seaborn.axisgrid.FacetGrid at 0x14c85185f60>



In [48]:
```
1 sns.displot(df['charges'])
```

Out[48]: <seaborn.axisgrid.FacetGrid at 0x14c85143940>



Conclusion

Using Exploratory Analysis, the relation between features has discovered.