# PROBLEM STATEMENT

For a health insurance company to make money, it needs to collect more in yearly premiums than it spends on medical care to its beneficiaries. As a result, insurers invest a great deal of time and money in developing models that accurately forecast medical expenses for the insured population.

The goal of this analysis is to use patient data to estimate the average medical care expenses for such population segments. These estimates can be used to create actuarial tables that set the price of yearly premiums higher or lower, depending on the expected treatment costs.

# IMPORTING LIBRARIES

In [1]:

```
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 LinearRegression
from sklearn import metrics
```

In [2]:

```
insurance_data = pd.read_csv(r"C:\Users\shaha\OneDrive\Desktop\Excel\insurance.csv")
insurance_data
```

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 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **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]:

```
insurance_data.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]:

```
insurance_data.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]:

```
insurance_data.shape
```

Out[5]:

```
(1338, 7)
```

In [6]:

```
insurance_data.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 [7]:

```
insurance_data.isnull().sum()
```

Out[7]:

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

In [8]:

```
insurance_data.describe()
```

Out[8]:

| | 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 [9]:

```
insurance_data.columns
```

Out[9]:

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

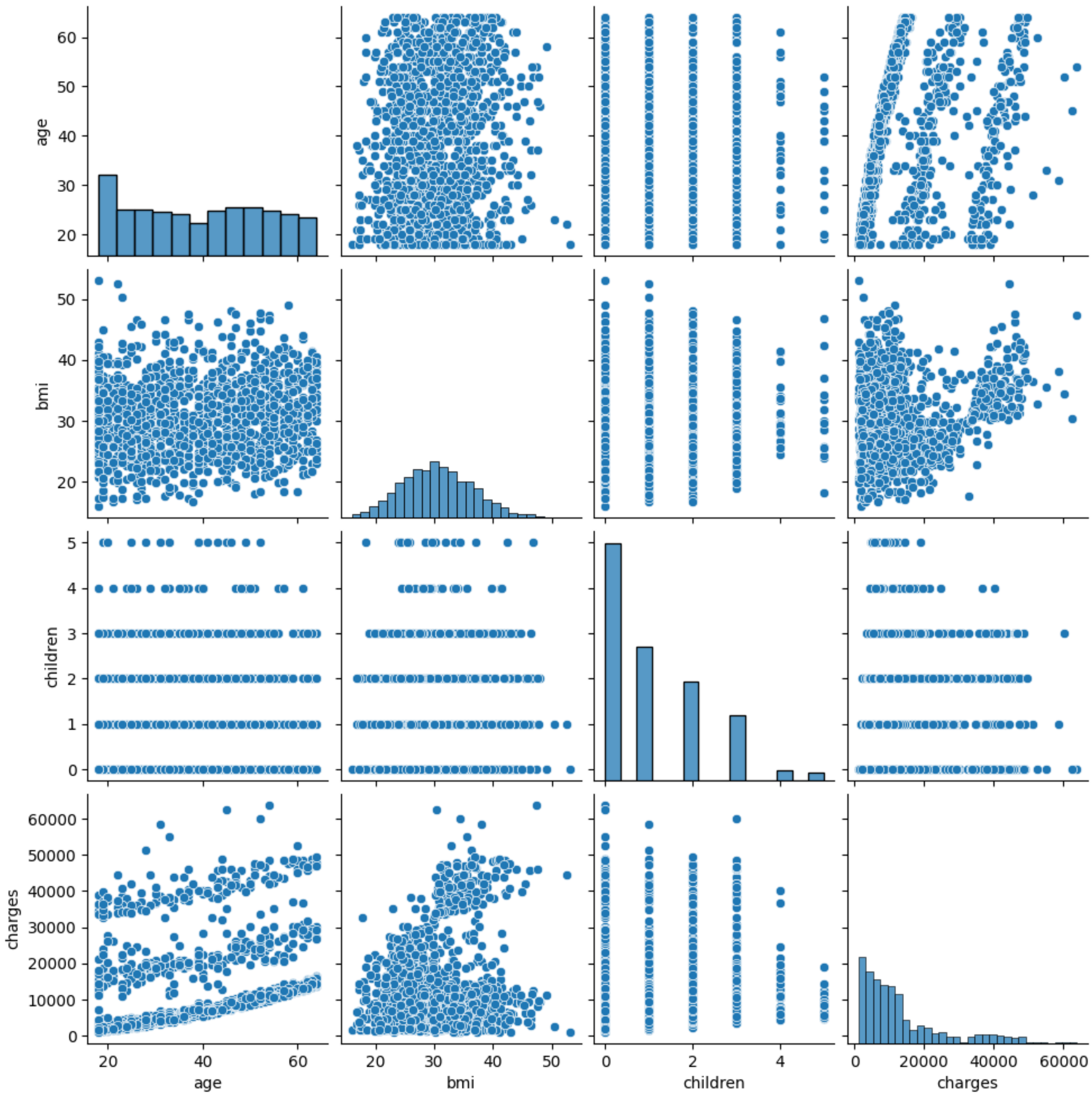
# Exploratory Data Analysis(Linear Regression)

In [10]:

```
sns.pairplot(insurance_data)
```

Out[10]:

<seaborn.axisgrid.PairGrid at 0x25c33ca7b50>

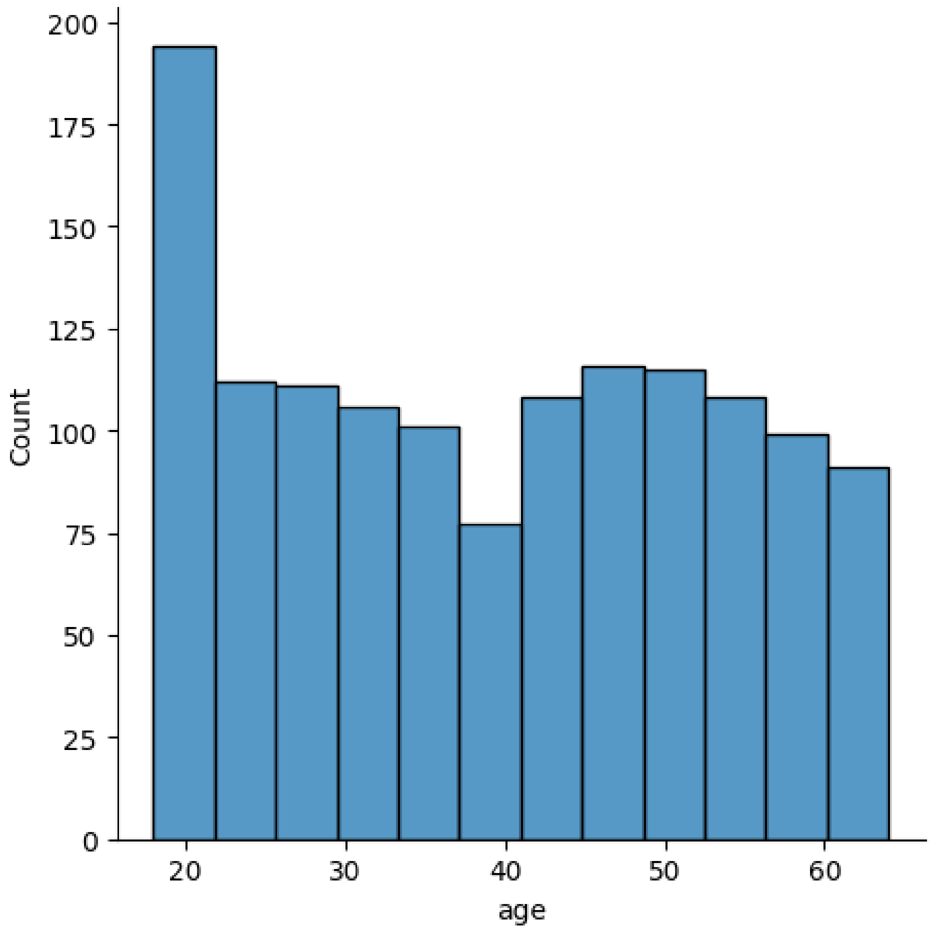


In [11]:

```
sns.displot(insurance_data['age'])
```

Out[11]:

<seaborn.axisgrid.FacetGrid at 0x25c67753550>

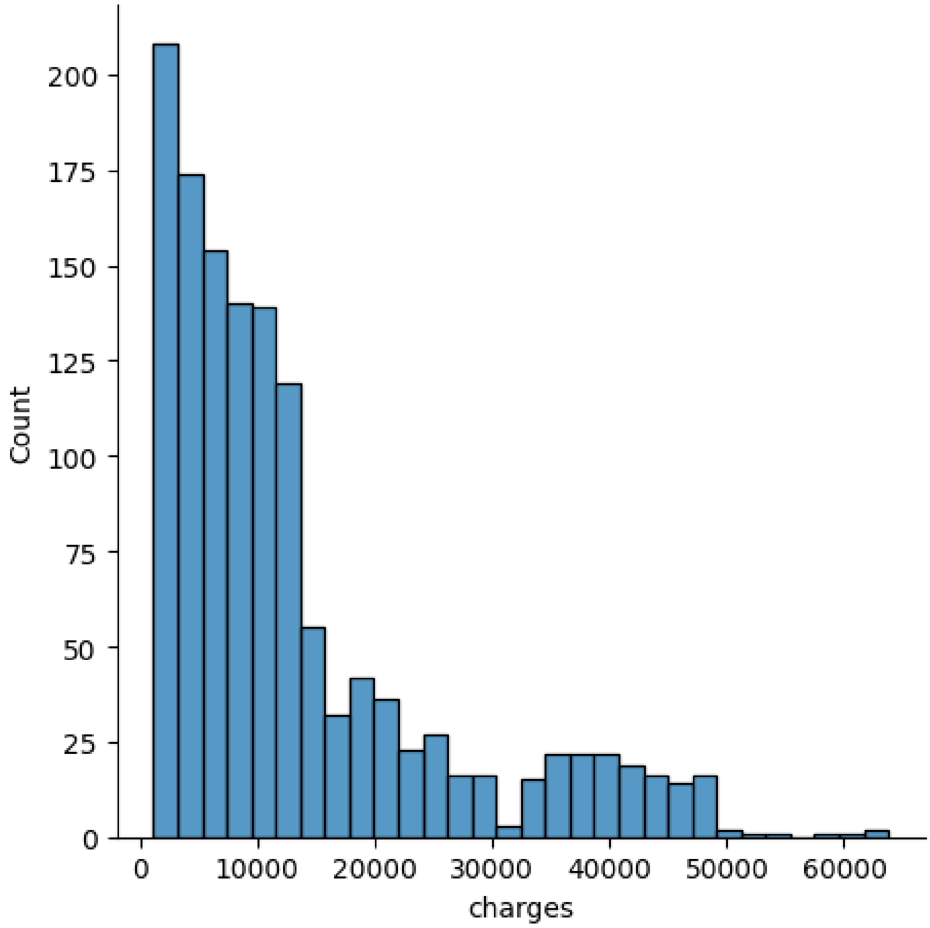


In [12]:

```
sns.displot(insurance_data['charges'])
```

Out[12]:

```
<seaborn.axisgrid.FacetGrid at 0x25c33d5f520>
```



In [13]:

```
s={"sex":{"female":1,"male":0}}
insurance_data=insurance_data.replace(s)
insurance_data
```

Out[13]:

| | age | sex | bmi | children | smoker | region | charges |
|---|---|---|---|---|---|---|---|
| **0** | 19 | 1 | 27.900 | 0 | yes | southwest | 16884.92400 |
| **1** | 18 | 0 | 33.770 | 1 | no | southeast | 1725.55230 |
| **2** | 28 | 0 | 33.000 | 3 | no | southeast | 4449.46200 |
| **3** | 33 | 0 | 22.705 | 0 | no | northwest | 21984.47061 |
| **4** | 32 | 0 | 28.880 | 0 | no | northwest | 3866.85520 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **1333** | 50 | 0 | 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 [14]:

```
som={"smoker":{"yes":1,"no":0}}
insurance_data=insurance_data.replace(som)
insurance_data
```

Out[14]:

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

1338 rows × 7 columns

In [15]:

```
insurance_data['region'].value_counts()
```

Out[15]:

```
region
southeast    364
southwest    325
northwest    325
northeast    324
Name: count, dtype: int64
```

In [16]:

```
reg={"region":{"southeast":1,"southwest":2,"northwest":3,"northeast":4}}
insurance_data=insurance_data.replace(reg)
insurance_data
```

Out[16]:

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

1338 rows × 7 columns

In [17]:

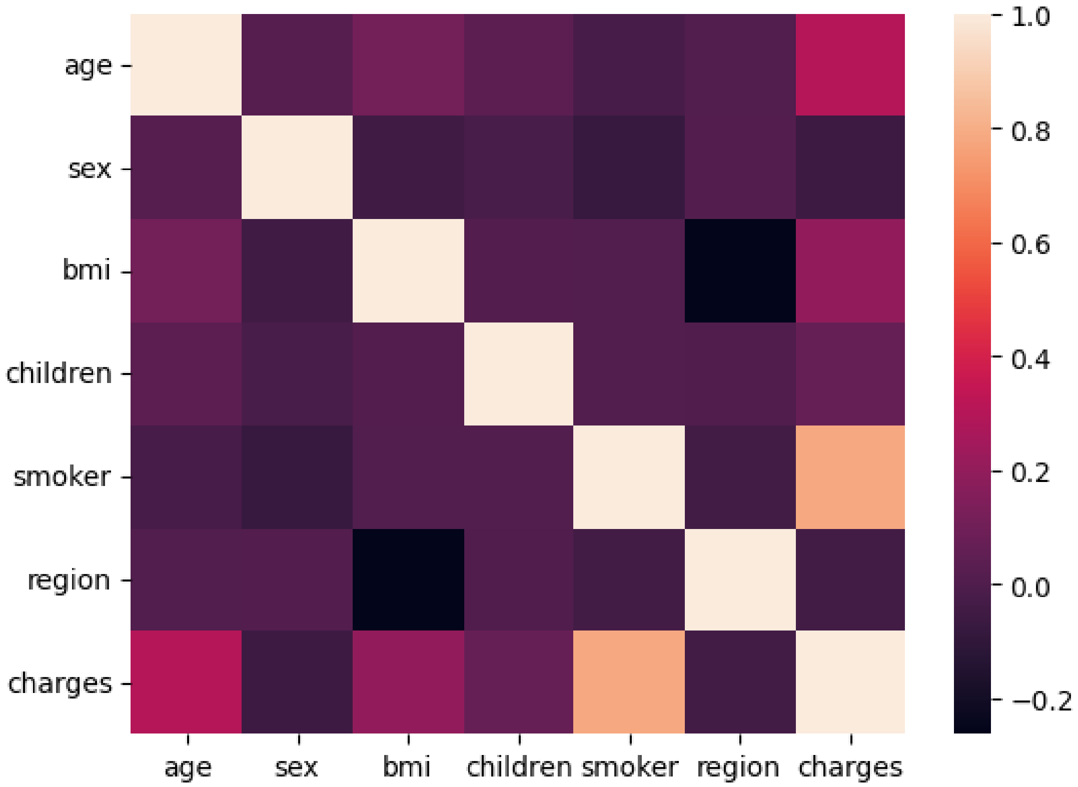
```
df=insurance_data[['age','sex','bmi','children','smoker','region','charges']]
```

In [18]:

```
sns.heatmap(df.corr())
```

Out[18]:

```
<Axes: >
```




# To Train The model

we are going to train linear Regression model.we need to first split up our data into x list that contains the features to train on,and y list with the target variable.

In [19]:

```
x=df[['age','sex','bmi','children','smoker','region']]
y=df['charges']
```

In [20]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=101)
```

In [21]:

```
lm=LinearRegression()
lm.fit(x_train,y_train)
```

Out[21]:

```
▾ LinearRegression
LinearRegression()
```

In [22]:

```
print(lm.intercept_)
```

```
-13563.198964426998
```

In [23]:

```
coeff_df=pd.DataFrame(lm.coef_,x.columns,columns=['coefficient'])
coeff_df
```

Out[23]:

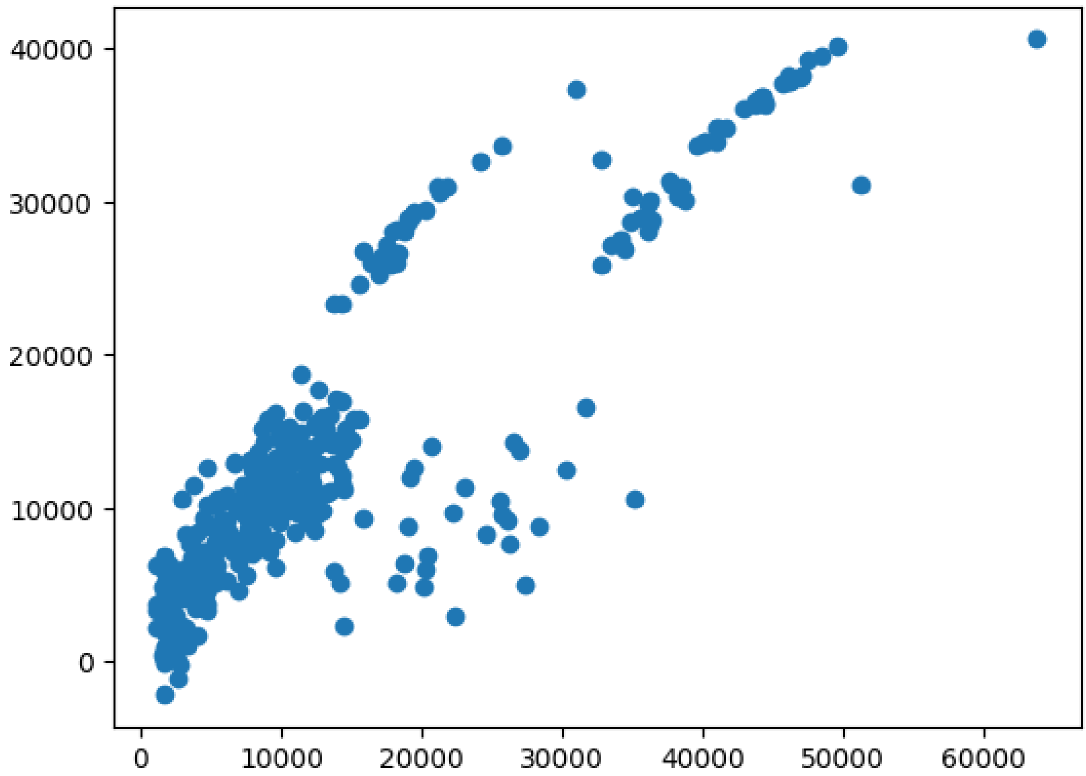
| | coefficient |
|---|---|
| **age** | 240.808027 |
| **sex** | -57.898004 |
| **bmi** | 369.026700 |
| **children** | 494.438485 |
| **smoker** | 23470.199126 |
| **region** | 281.762820 |

In [24]:

```
predictions=lm.predict(x_test)
plt.scatter(y_test,predictions)
```

Out[24]:

```
<matplotlib.collections.PathCollection at 0x25c69494550>
```

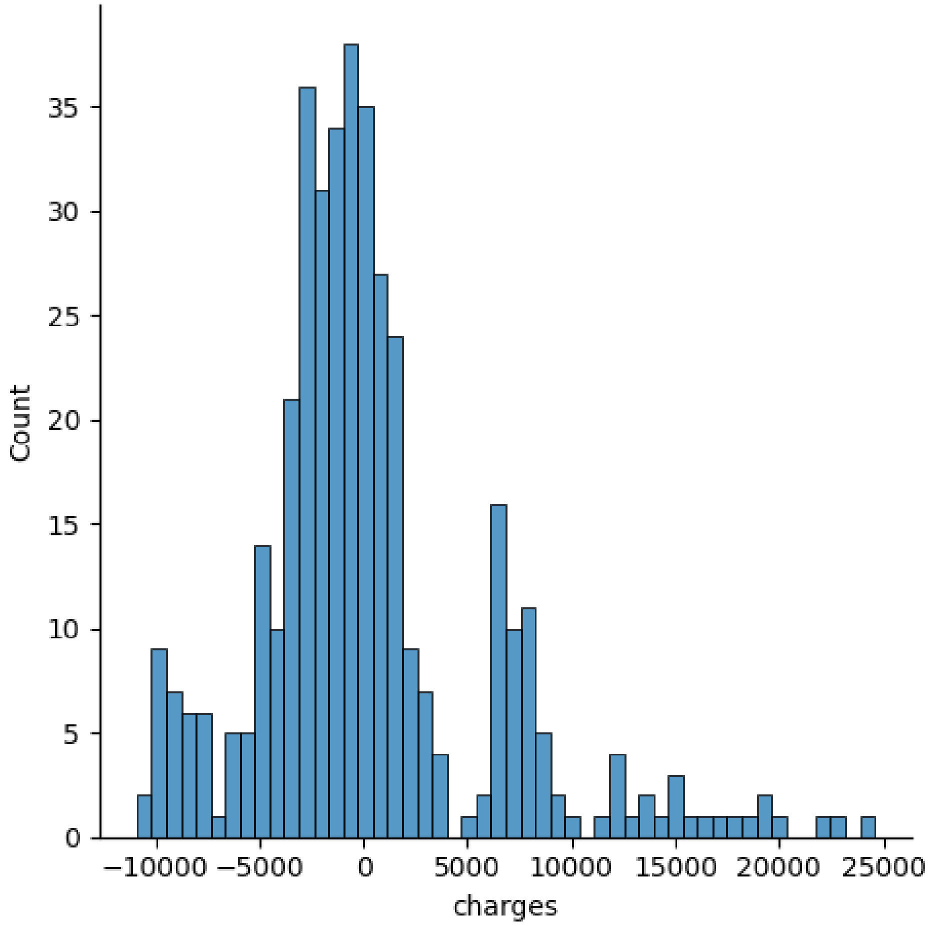


In [25]:

```
sns.displot((y_test-predictions),bins=50)
```

Out[25]:

```
<seaborn.axisgrid.FacetGrid at 0x25c694bc790>
```




In [26]:

```
from sklearn.metrics import r2_score
y_pred=lm.predict(x_test)
r2=r2_score(y_test,y_pred)
print("R2 score:",r2)
```

```
R2 score: 0.7621231693371181
```

# Logistic Regression

In [27]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

In [28]:

```
pd.set_option('display.max_rows',10000000000)
pd.set_option('display.max_columns',10000000000)
pd.set_option('display.width',95)
```

In [29]:

```
print('This DataFrame has %d Rows and %d columns'%(df.shape))
```

```
This DataFrame has 1338 Rows and 7 columns
```

In [30]:

```
features_matrix=df.iloc[:,0:3]
```

In [31]:

```
target_vector=df.iloc[:,-3]
```

In [32]:

```
print('The Features Matrix Has %d Rows And %d Column(s)'%(features_matrix.shape))
print('The Target Matrix Has %d Rows and %d columns(s)'%(np.array(target_vector).reshape(-1,1
```

```
The Features Matrix Has 1338 Rows And 3 Column(s)
The Target Matrix Has 1338 Rows and 1 columns(s)
```

In [33]:

```
features_matrix_standardized=StandardScaler().fit_transform(features_matrix)
```

In [34]:

```
algorithm=LogisticRegression(max_iter=1000000)
```

In [35]:

```
logistic_Regression_Model=algorithm.fit(features_matrix_standardized,target_vector)
```

In [36]:

```
observation=[[1,0,0.3]]
```

In [37]:

```
predictions=logistic_Regression_Model.predict(observation)
print('The Model predicted the observation to belog to class %s'%(predictions))
```

```
The Model predicted the observation to belog to class [0]
```

In [38]:

```
print('The algorithm was trained to predict one of the two classes:%s'%(algorithm.classes_))
```

```
The algorithm was trained to predict one of the two classes:[0 1]
```

In [39]:

```
print("""The model says the probability of the obserbvation we passedbelonging to class['b']is
      %(algorithm.predict_proba(observation)[0][0]))
print()
print("""The model says the probability of the observation we passed belonging to class['g']is
     %(algorithm.predict_proba(observation)[observation[0][1]]))
```

```
The model says the probability of the obserbvation we passedbelonging to class
['b']is 0.806147490103992

The model says the probability of the observation we passed belonging to class
['g']is [0.80614749 0.19385251]
```

In [40]:

```
x=np.array(df['age']).reshape(-1,1)
y=np.array(df['smoker']).reshape(-3,1)
```

In [41]:

```
lr=LogisticRegression()
lr.fit(x,y)
print(lr.score(x,y))
```

```
0.7952167414050823

C:\Users\shaha\AppData\Local\Programs\Python\Python310\lib\site-packages\sklear
n\utils\validation.py:1143: DataConversionWarning: A column-vector y was passed
when a 1d array was expected. Please change the shape of y to (n_samples, ), fo
r example using ravel().
  y = column_or_1d(y, warn=True)
```

# Ridge and Lasso Regression

In [42]:

```
#Ridge Regression MOdel
from sklearn.linear_model import Ridge,RidgeCV,Lasso
from sklearn.preprocessing import StandardScaler
```

In [43]:

```
plt.figure(figsize=(10,10))
```

Out[43]:

```
<Figure size 1000x1000 with 0 Axes>

<Figure size 1000x1000 with 0 Axes>
```

In [44]:

```
features = insurance_data.columns[0:1]
target = insurance_data.columns[-1:]
#x and y values
x = insurance_data[features].values
y = insurance_data[target].values
#splot
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=17)
print("The dimension of x_train is {}".format(x_train.shape))
print("The dimension of x_test is {}".format(x_test.shape))
#Scale features
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

```
The dimension of x_train is (936, 1)
The dimension of x_test is (402, 1)
```

In [45]:

```
#Model
lr = LinearRegression()
#fit model
lr.fit(x_train ,y_train)
#predict
#prediction = lr.predict(x_test)
#actual
actual = y_test
train_score_lr = lr.score(x_train,y_train)
test_score_lr = lr.score(x_test,y_test)
print("\nLinear Regression Model:\n")
print("The train score for lr model is {}".format(train_score_lr))
print("The test score for lr model is {}".format(test_score_lr))
```

```

Linear Regression Model:

The train score for lr model is 0.07447061146193878
The test score for lr model is 0.10891203216512224
```

In [46]:

```
#Ridge Regression Model
ridgeReg = Ridge(alpha=10)
ridgeReg.fit(x_train,y_train)
#train and test score for ridge regression
train_score_ridge = ridgeReg.score(x_train,y_train)
test_score_ridge = ridgeReg.score(x_test,y_test)
print("\nRidge Model:\n")
print("The train score for ridge model is {}".format(train_score_ridge))
print("The test score for ridge model is {}".format(test_score_ridge))
```

```

Ridge Model:

The train score for ridge model is 0.07446228994221393
The test score for ridge model is 0.10855133360950642
```

In [47]:

```
#Lasso Regression model
print("\nLasso Model:\n")
lasso = Lasso(alpha = 10)
lasso.fit(x_train,y_train)
train_score_ls = lasso.score(x_train,y_train)
test_score_ls = lasso.score(x_test,y_test)
print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is {}".format(test_score_ls))
```

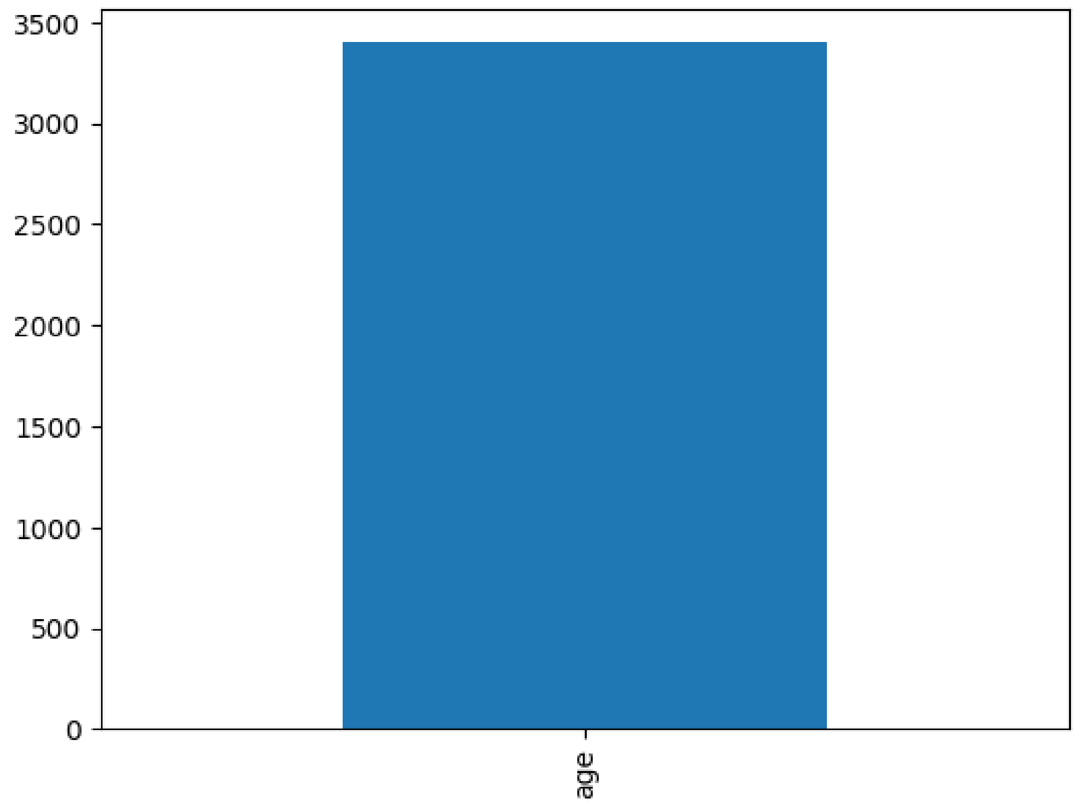
```
Lasso Model:

The train score for ls model is 0.07446997086306062
The test score for ls model is 0.10881427793326703
```

In [48]:

```
pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "bar")
```

Out[48]:

<Axes: >



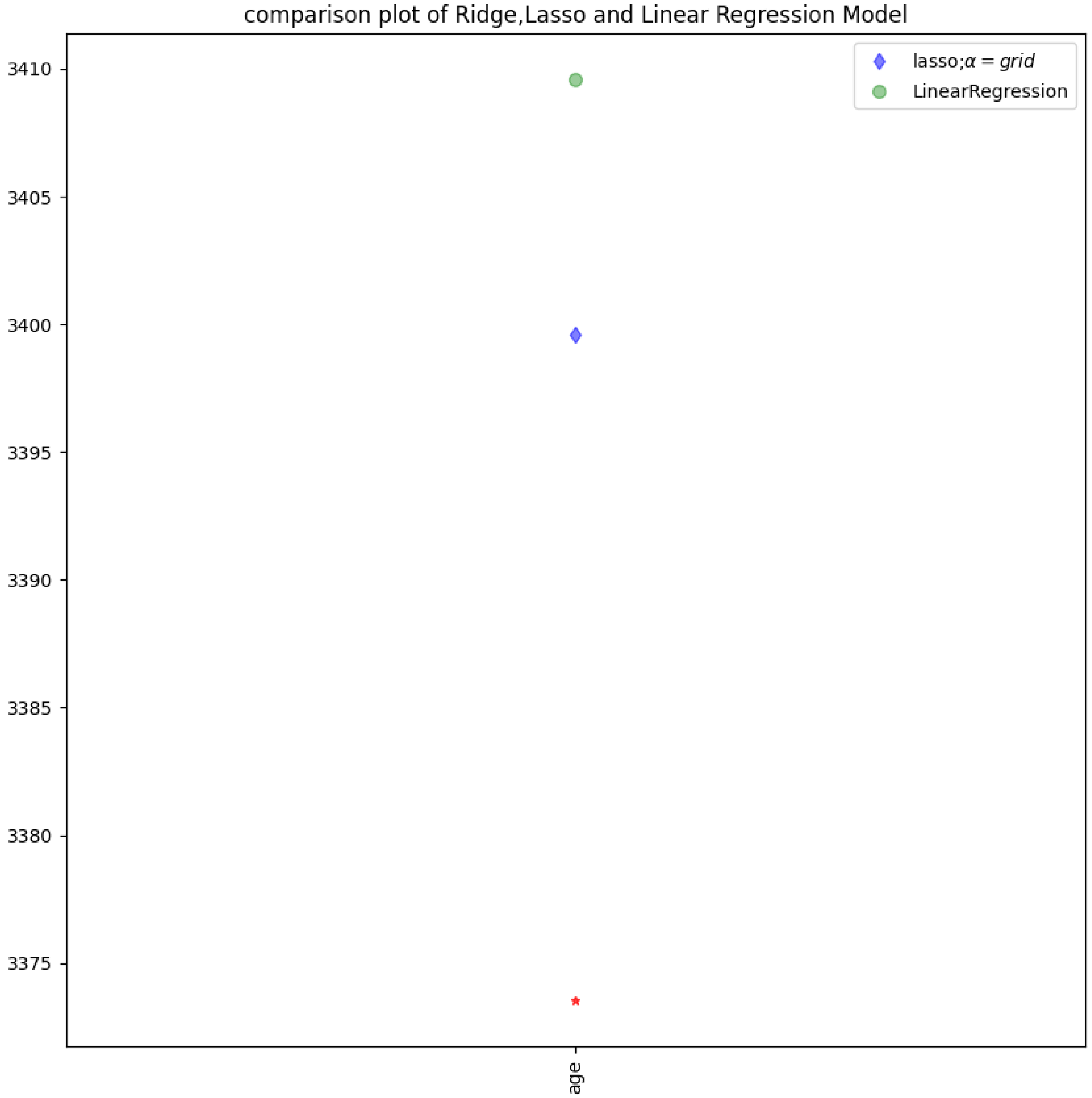
In [49]:

```
#using the linear CV model
from sklearn.linear_model import LassoCV
#Lasso Cross validation
lasso_cv = LassoCV(alphas = [0.0001,0.001,0.01,0.1,1,10],random_state=0).fit(x_train,y_train)
#score
print(lasso_cv.score(x_train,y_train))
print(lasso_cv.score(x_test,y_test))
```

```
0.07446997086306062
0.10881427793326703

C:\Users\shaha\AppData\Local\Programs\Python\Python310\lib\site-packages\sklear
n\linear_model\_coordinate_descent.py:1568: DataConversionWarning: A column-vec
tor y was passed when a 1d array was expected. Please change the shape of y to
(n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [50]:

```
#plot size
plt.figure(figsize=(10,10))
#add plot for ridge regression
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='re
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker = 'o',markersize=7,color='green',
plt.xticks(rotation=90)
plt.legend()
plt.title("comparison plot of Ridge,Lasso and Linear Regression Model")
plt.show()
```



In [51]:

```
#using the linear Cv model
from sklearn.linear_model import RidgeCV
#Ridge cross validation
ridge_cv = RidgeCV(alphas = [0.0001,0.001,0.01,0.1,1,10]).fit(x_train,y_train)
#score
print("The train score for ridge model is {}".format(ridge_cv.score(x_train,y_train)))
print("The train score for ridge model is {}".format(ridge_cv.score(x_test,y_test)))
```

```
The train score for ridge model is 0.07446228994221393
The train score for ridge model is 0.10855133360950775
```

# Elastic Net

In [78]:

```
from sklearn.linear_model import ElasticNet
regr = ElasticNet()
regr.fit(x,y)
print(regr.coef_)
print(regr.intercept_)
regr.score(x,y)
```

```
[-5.41728687e-03 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
 -0.00000000e+00  2.81193545e-05]
0.0440232543556241
```

Out[78]:

```
0.6855923754478445
```

In [53]:

```
y_pred_elastic = regr.predict(x_train)
```

In [54]:

```
mean_squared_error = np.mean((y_pred_elastic-y_train)**2)
print("Mean squared Error on test set",mean_squared_error)
```

```
Mean squared Error on test set 269213980.1466613
```

# Decision Tree

In [55]:

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

In [60]:

```
x=["age","sex","bmi","children","region"]
y=["Yes","No"]
all_inputs=df[x]
all_classes=df["smoker"]
```

In [61]:

```
(x_train,x_test,y_train,y_test)=train_test_split(all_inputs,all_classes,test_size=0.30)
```

In [62]:

```
clf=DecisionTreeClassifier(random_state=0)
```

In [63]:

```
clf.fit(x_train,y_train)
```

Out[63]:

```
▾ DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)
```

In [64]:

```
score=clf.score(x_test,y_test)
print(score)
```

```
0.6467661691542289
```

# Random Forest

In [65]:

```
import matplotlib.pyplot as plt,seaborn as sns
```

In [66]:

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

In [67]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.7)
x_train.shape,x_test.shape
```

Out[67]:

```
((936, 6), (402, 6))
```

In [68]:

```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[68]:

```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [69]:

```
rf=RandomForestClassifier()
```

In [70]:

```
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 [71]:

```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rf,param_grid=params,cv=2,scoring='accuracy')
grid_search.fit(x_train,y_train)
```

Out[71]:

```
▸ GridSearchCV
▸ estimator: RandomForestClassifier
    ▸ RandomForestClassifier
```

In [72]:

```
grid_search.best_score_
```

Out[72]:

```
0.9444444444444444
```

In [73]:

```
rf_best=grid_search.best_estimator_
print(rf_best)
```

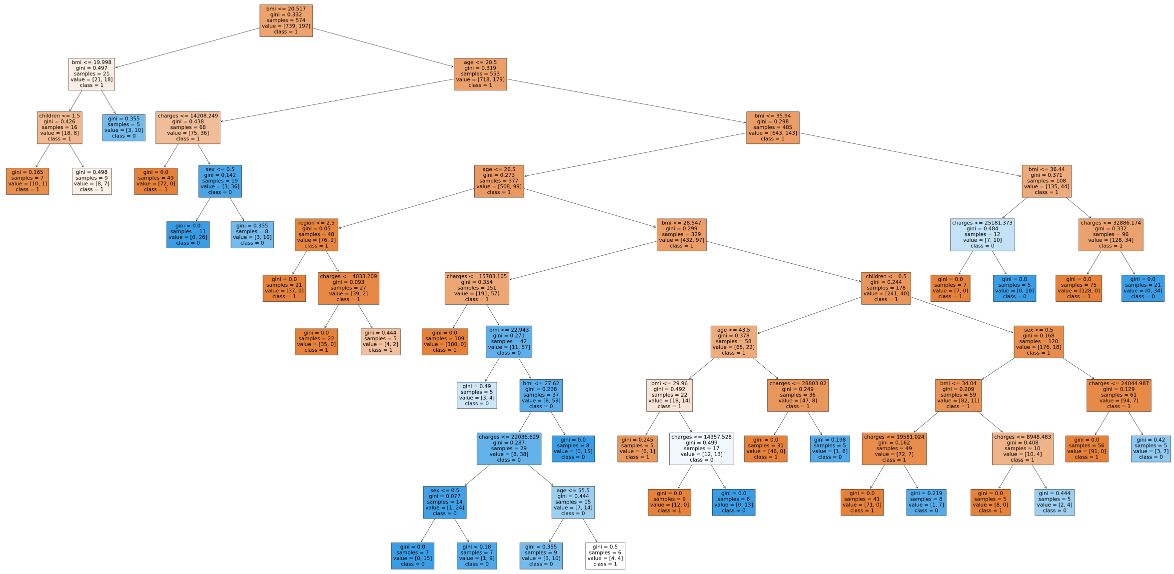
```
RandomForestClassifier(max_depth=20, min_samples_leaf=5, n_estimators=50)
```

In [74]:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[5],feature_names=x.columns,class_names=["1","0"],filled=True)
```

Out[74]:

```
[Text(0.24472128378378377, 0.9545454545454546, 'bmi <= 20.517\ngini = 0.332\nsa
mples = 574\nvalue = [739, 197]\nclass = 1'),
 Text(0.08108108108108109, 0.8636363636363636, 'bmi <= 19.998\ngini = 0.497\nsa
mples = 21\nvalue = [21, 18]\nclass = 1'),
 Text(0.05405405405405406, 0.7727272727272727, 'children <= 1.5\ngini = 0.426\n
samples = 16\nvalue = [18, 8]\nclass = 1'),
 Text(0.02702702702702703, 0.6818181818181818, 'gini = 0.165\nsamples = 7\nvalu
e = [10, 1]\nclass = 1'),
 Text(0.08108108108108109, 0.6818181818181818, 'gini = 0.498\nsamples = 9\nvalu
e = [8, 7]\nclass = 1'),
 Text(0.10810810810810811, 0.7727272727272727, 'gini = 0.355\nsamples = 5\nvalu
e = [3, 10]\nclass = 0'),
 Text(0.4083614864864865, 0.8636363636363636, 'age <= 20.5\ngini = 0.319\nsampl
es = 553\nvalue = [718, 179]\nclass = 1'),
 Text(0.16216216216216217, 0.7727272727272727, 'charges <= 14208.249\ngini = 0.
438\nsamples = 68\nvalue = [75, 36]\nclass = 1'),
 Text(0.13513513513513514, 0.6818181818181818, 'gini = 0.0\nsamples = 49\nvalue
= [72, 0]\nclass = 1'),
 Text(0.1891891891891892, 0.6818181818181818, 'sex <= 0.5\ngini = 0.142\nsample
s = 19\nvalue = [3, 36]\nclass = 0'),
 Text(0.16216216216216217, 0.5909090909090909, 'gini = 0.0\nsamples = 11\nvalue
= [0, 26]\nclass = 0'),
 Text(0.21621621621621623, 0.5909090909090909, 'gini = 0.355\nsamples = 8\nvalu
e = [3, 10]\nclass = 0'),
 Text(0.6545608108108109, 0.7727272727272727, 'bmi <= 35.94\ngini = 0.298\nsamp
les = 485\nvalue = [643, 143]\nclass = 1'),
 Text(0.4239864864864865, 0.6818181818181818, 'age <= 26.5\ngini = 0.273\nsampl
es = 377\nvalue = [508, 99]\nclass = 1'),
 Text(0.2702702702702703, 0.5909090909090909, 'region <= 2.5\ngini = 0.05\nsamp
les = 48\nvalue = [76, 2]\nclass = 1'),
 Text(0.24324324324324326, 0.5, 'gini = 0.0\nsamples = 21\nvalue = [37, 0]\ncla
ss = 1'),
 Text(0.2972972972972973, 0.5, 'charges <= 4033.209\ngini = 0.093\nsamples = 27
\nvalue = [39, 2]\nclass = 1'),
 Text(0.2702702702702703, 0.4090909090909091, 'gini = 0.0\nsamples = 22\nvalue
= [35, 0]\nclass = 1'),
 Text(0.32432432432432434, 0.4090909090909091, 'gini = 0.444\nsamples = 5\nvalu
e = [4, 2]\nclass = 1'),
 Text(0.5777027027027027, 0.5909090909090909, 'bmi <= 28.547\ngini = 0.299\nsam
ples = 329\nvalue = [432, 97]\nclass = 1'),
 Text(0.40540540540540543, 0.5, 'charges <= 15783.105\ngini = 0.354\nsamples =
151\nvalue = [191, 57]\nclass = 1'),
 Text(0.3783783783783784, 0.4090909090909091, 'gini = 0.0\nsamples = 109\nvalue
= [180, 0]\nclass = 1'),
 Text(0.43243243243243246, 0.4090909090909091, 'bmi <= 22.943\ngini = 0.271\nsa
mples = 42\nvalue = [11, 57]\nclass = 0'),
 Text(0.40540540540540543, 0.3181818181818182, 'gini = 0.49\nsamples = 5\nvalue
= [3, 4]\nclass = 0'),
 Text(0.4594594594594595, 0.3181818181818182, 'bmi <= 27.62\ngini = 0.228\nsamp
les = 37\nvalue = [8, 53]\nclass = 0'),
 Text(0.43243243243243246, 0.22727272727272727, 'charges <= 22036.629\ngini =
0.287\nsamples = 29\nvalue = [8, 38]\nclass = 0'),
 Text(0.3783783783783784, 0.13636363636363635, 'sex <= 0.5\ngini = 0.077\nsampl
es = 14\nvalue = [1, 24]\nclass = 0'),
 Text(0.35135135135135137, 0.045454545454545456, 'gini = 0.0\nsamples = 7\nvalu
e = [0, 15]\nclass = 0'),
 Text(0.40540540540540543, 0.045454545454545456, 'gini = 0.18\nsamples = 7\nval
ue = [1, 9]\nclass = 0'),
 Text(0.4864864864864865, 0.13636363636363635, 'age <= 55.5\ngini = 0.444\nsamp
les = 15\nvalue = [7, 14]\nclass = 0'),
 Text(0.4594594594594595, 0.045454545454545456, 'gini = 0.355\nsamples = 9\nval
ue = [3, 10]\nclass = 0'),
```



```
 Text(0.6756756756756757, 0.3181818181818182, 'charges <= 28803.02\ngini = 0.24
9\nsamples = 36\nvalue = [47, 8]\nclass = 1'),
 Text(0.6486486486486487, 0.22727272727272727, 'gini = 0.0\nsamples = 31\nvalue
= [46, 0]\nclass = 1'),
 Text(0.7027027027027027, 0.22727272727272727, 'gini = 0.198\nsamples = 5\nvalu
e = [1, 8]\nclass = 0'),
 Text(0.8783783783783784, 0.4090909090909091, 'sex <= 0.5\ngini = 0.168\nsample
s = 120\nvalue = [176, 18]\nclass = 1'),
 Text(0.8108108108108109, 0.3181818181818182, 'bmi <= 34.04\ngini = 0.209\nsamp
les = 59\nvalue = [82, 11]\nclass = 1'),
 Text(0.7567567567567568, 0.22727272727272727, 'charges <= 19581.024\ngini = 0.
162\nsamples = 49\nvalue = [72, 7]\nclass = 1'),
 Text(0.7297297297297297, 0.13636363636363635, 'gini = 0.0\nsamples = 41\nvalue
= [71, 0]\nclass = 1'),
 Text(0.7837837837837838, 0.13636363636363635, 'gini = 0.219\nsamples = 8\nvalu
e = [1, 7]\nclass = 0'),
 Text(0.8648648648648649, 0.22727272727272727, 'charges <= 8948.483\ngini = 0.4
08\nsamples = 10\nvalue = [10, 4]\nclass = 1'),
 Text(0.8378378378378378, 0.13636363636363635, 'gini = 0.0\nsamples = 5\nvalue
= [8, 0]\nclass = 1'),
 Text(0.8918918918918919, 0.13636363636363635, 'gini = 0.444\nsamples = 5\nvalu
e = [2, 4]\nclass = 0'),
 Text(0.9459459459459459, 0.3181818181818182, 'charges <= 24044.987\ngini = 0.1
29\nsamples = 61\nvalue = [94, 7]\nclass = 1'),
 Text(0.918918918918919, 0.22727272727272727, 'gini = 0.0\nsamples = 56\nvalue
= [91, 0]\nclass = 1'),
 Text(0.972972972972973, 0.22727272727272727, 'gini = 0.42\nsamples = 5\nvalue
= [3, 7]\nclass = 0'),
 Text(0.8851351351351351, 0.6818181818181818, 'bmi <= 36.44\ngini = 0.371\nsamp
les = 108\nvalue = [135, 44]\nclass = 1'),
 Text(0.831081081081081, 0.5909090909090909, 'charges <= 25181.373\ngini = 0.48
4\nsamples = 12\nvalue = [7, 10]\nclass = 0'),
 Text(0.8040540540540541, 0.5, 'gini = 0.0\nsamples = 7\nvalue = [7, 0]\nclass
= 1'),
 Text(0.8581081081081081, 0.5, 'gini = 0.0\nsamples = 5\nvalue = [0, 10]\nclass
= 0'),
 Text(0.9391891891891891, 0.5909090909090909, 'charges <= 32886.174\ngini = 0.3
32\nsamples = 96\nvalue = [128, 34]\nclass = 1'),
 Text(0.9121621621621622, 0.5, 'gini = 0.0\nsamples = 75\nvalue = [128, 0]\ncla
ss = 1'),
 Text(0.9662162162162162, 0.5, 'gini = 0.0\nsamples = 21\nvalue = [0, 34]\nclas
s = 0')]
```

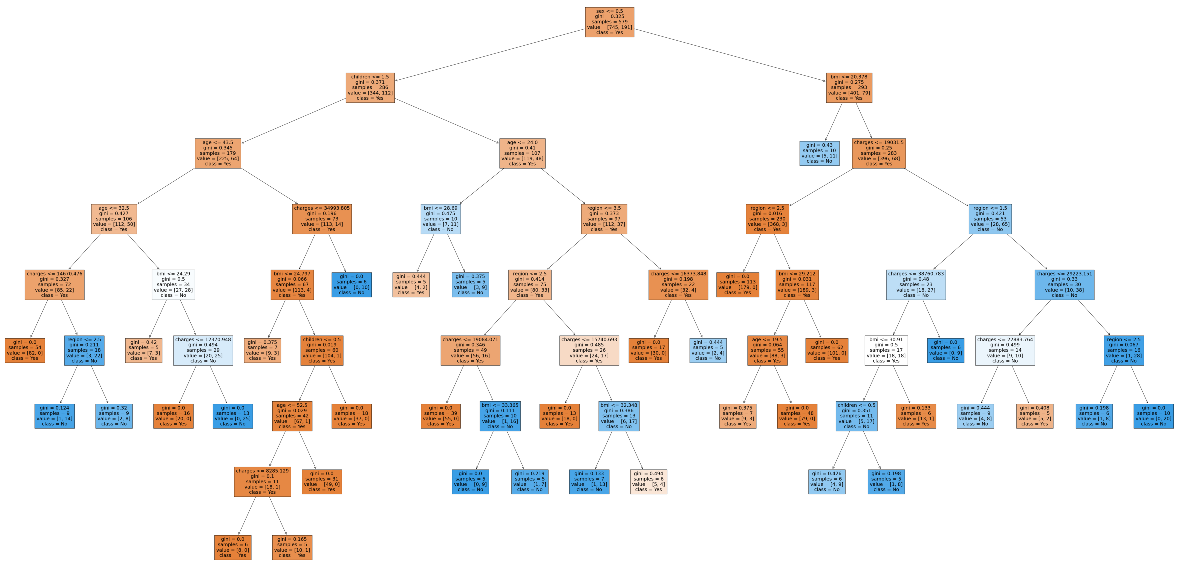
In [75]:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[7],feature_names=x.columns,class_names=["Yes","No"],filled=True
```

Out[75]:

```
[Text(0.5171875, 0.9444444444444444, 'sex <= 0.5\ngini = 0.325\nsamples = 579\n
value = [745, 191]\nclass = Yes'),
 Text(0.315625, 0.8333333333333334, 'children <= 1.5\ngini = 0.371\nsamples = 2
86\nvalue = [344, 112]\nclass = Yes'),
 Text(0.1875, 0.7222222222222222, 'age <= 43.5\ngini = 0.345\nsamples = 179\nva
lue = [225, 64]\nclass = Yes'),
 Text(0.1, 0.6111111111111112, 'age <= 32.5\ngini = 0.427\nsamples = 106\nvalue
= [112, 50]\nclass = Yes'),
 Text(0.05, 0.5, 'charges <= 14670.476\ngini = 0.327\nsamples = 72\nvalue = [8
5, 22]\nclass = Yes'),
 Text(0.025, 0.3888888888888889, 'gini = 0.0\nsamples = 54\nvalue = [82, 0]\ncl
ass = Yes'),
 Text(0.075, 0.3888888888888889, 'region <= 2.5\ngini = 0.211\nsamples = 18\nva
lue = [3, 22]\nclass = No'),
 Text(0.05, 0.2777777777777778, 'gini = 0.124\nsamples = 9\nvalue = [1, 14]\ncl
ass = No'),
 Text(0.1, 0.2777777777777778, 'gini = 0.32\nsamples = 9\nvalue = [2, 8]\nclass
= No'),
 Text(0.15, 0.5, 'bmi <= 24.29\ngini = 0.5\nsamples = 34\nvalue = [27, 28]\ncla
ss = No'),
 Text(0.125, 0.3888888888888889, 'gini = 0.42\nsamples = 5\nvalue = [7, 3]\ncla
ss = Yes'),
 Text(0.175, 0.3888888888888889, 'charges <= 12370.948\ngini = 0.494\nsamples =
29\nvalue = [20, 25]\nclass = No'),
 Text(0.15, 0.2777777777777778, 'gini = 0.0\nsamples = 16\nvalue = [20, 0]\ncla
ss = Yes'),
 Text(0.2, 0.2777777777777778, 'gini = 0.0\nsamples = 13\nvalue = [0, 25]\nclas
s = No'),
 Text(0.275, 0.6111111111111112, 'charges <= 34993.805\ngini = 0.196\nsamples =
73\nvalue = [113, 14]\nclass = Yes'),
 Text(0.25, 0.5, 'bmi <= 24.797\ngini = 0.066\nsamples = 67\nvalue = [113, 4]\n
class = Yes'),
 Text(0.225, 0.3888888888888889, 'gini = 0.375\nsamples = 7\nvalue = [9, 3]\ncl
ass = Yes'),
 Text(0.275, 0.3888888888888889, 'children <= 0.5\ngini = 0.019\nsamples = 60\n
value = [104, 1]\nclass = Yes'),
 Text(0.25, 0.2777777777777778, 'age <= 52.5\ngini = 0.029\nsamples = 42\nvalue
= [67, 1]\nclass = Yes'),
 Text(0.225, 0.16666666666666666, 'charges <= 8285.129\ngini = 0.1\nsamples = 1
1\nvalue = [18, 1]\nclass = Yes'),
 Text(0.2, 0.05555555555555555, 'gini = 0.0\nsamples = 6\nvalue = [8, 0]\nclass
= Yes'),
 Text(0.25, 0.05555555555555555, 'gini = 0.165\nsamples = 5\nvalue = [10, 1]\nc
lass = Yes'),
 Text(0.275, 0.16666666666666666, 'gini = 0.0\nsamples = 31\nvalue = [49, 0]\nc
lass = Yes'),
 Text(0.3, 0.2777777777777778, 'gini = 0.0\nsamples = 18\nvalue = [37, 0]\nclas
s = Yes'),
 Text(0.3, 0.5, 'gini = 0.0\nsamples = 6\nvalue = [0, 10]\nclass = No'),
 Text(0.44375, 0.7222222222222222, 'age <= 24.0\ngini = 0.41\nsamples = 107\nva
lue = [119, 48]\nclass = Yes'),
 Text(0.375, 0.6111111111111112, 'bmi <= 28.69\ngini = 0.475\nsamples = 10\nval
ue = [7, 11]\nclass = No'),
 Text(0.35, 0.5, 'gini = 0.444\nsamples = 5\nvalue = [4, 2]\nclass = Yes'),
 Text(0.4, 0.5, 'gini = 0.375\nsamples = 5\nvalue = [3, 9]\nclass = No'),
 Text(0.5125, 0.6111111111111112, 'region <= 3.5\ngini = 0.373\nsamples = 97\nv
alue = [112, 37]\nclass = Yes'),
 Text(0.45, 0.5, 'region <= 2.5\ngini = 0.414\nsamples = 75\nvalue = [80, 33]\n
class = Yes'),
 Text(0.4, 0.3888888888888889, 'charges <= 19084.071\ngini = 0.346\nsamples = 4
9\nvalue = [56, 16]\nclass = Yes'),
 Text(0.375, 0.2777777777777778, 'gini = 0.0\nsamples = 39\nvalue = [55, 0]\ncl
```

```
ass = Yes'),
 Text(0.425, 0.2777777777777778, 'bmi <= 33.365\ngini = 0.111\nsamples = 10\nva
lue = [1, 16]\nclass = No'),
 Text(0.4, 0.16666666666666666, 'gini = 0.0\nsamples = 5\nvalue = [0, 9]\nclass
= No'),
 Text(0.45, 0.16666666666666666, 'gini = 0.219\nsamples = 5\nvalue = [1, 7]\ncl
ass = No'),
 Text(0.5, 0.3888888888888889, 'charges <= 15740.693\ngini = 0.485\nsamples = 2
6\nvalue = [24, 17]\nclass = Yes'),
 Text(0.475, 0.2777777777777778, 'gini = 0.0\nsamples = 13\nvalue = [18, 0]\ncl
ass = Yes'),
```



```
283\nvalue = [396, 68]\nclass = Yes'),
 Text(0.65, 0.6111111111111112, 'region <= 2.5\ngini = 0.016\nsamples = 230\nva
lue = [368, 3]\nclass = Yes'),
 Text(0.625, 0.5, 'gini = 0.0\nsamples = 113\nvalue = [179, 0]\nclass = Yes'),
 Text(0.675, 0.5, 'bmi <= 29.212\ngini = 0.031\nsamples = 117\nvalue = [189, 3]
\nclass = Yes'),
 Text(0.65, 0.3888888888888889, 'age <= 19.5\ngini = 0.064\nsamples = 55\nvalue
= [88, 3]\nclass = Yes'),
 Text(0.625, 0.2777777777777778, 'gini = 0.375\nsamples = 7\nvalue = [9, 3]\ncl
ass = Yes'),
 Text(0.675, 0.2777777777777778, 'gini = 0.0\nsamples = 48\nvalue = [79, 0]\ncl
ass = Yes'),
 Text(0.7, 0.3888888888888889, 'gini = 0.0\nsamples = 62\nvalue = [101, 0]\ncla
ss = Yes'),
 Text(0.8375, 0.6111111111111112, 'region <= 1.5\ngini = 0.421\nsamples = 53\nv
alue = [28, 65]\nclass = No'),
 Text(0.775, 0.5, 'charges <= 38760.783\ngini = 0.48\nsamples = 23\nvalue = [1
8, 27]\nclass = No'),
 Text(0.75, 0.3888888888888889, 'bmi <= 30.91\ngini = 0.5\nsamples = 17\nvalue
= [18, 18]\nclass = Yes'),
 Text(0.725, 0.2777777777777778, 'children <= 0.5\ngini = 0.351\nsamples = 11\n
value = [5, 17]\nclass = No'),
 Text(0.7, 0.16666666666666666, 'gini = 0.426\nsamples = 6\nvalue = [4, 9]\ncla
ss = No'),
 Text(0.75, 0.16666666666666666, 'gini = 0.198\nsamples = 5\nvalue = [1, 8]\ncl
ass = No'),
 Text(0.775, 0.2777777777777778, 'gini = 0.133\nsamples = 6\nvalue = [13, 1]\nc
lass = Yes'),
 Text(0.8, 0.3888888888888889, 'gini = 0.0\nsamples = 6\nvalue = [0, 9]\nclass
= No'),
 Text(0.9, 0.5, 'charges <= 29223.151\ngini = 0.33\nsamples = 30\nvalue = [10,
38]\nclass = No'),
 Text(0.85, 0.3888888888888889, 'charges <= 22883.764\ngini = 0.499\nsamples =
14\nvalue = [9, 10]\nclass = No'),
 Text(0.825, 0.2777777777777778, 'gini = 0.444\nsamples = 9\nvalue = [4, 8]\ncl
```

In [76]:

```
rf_best.feature_importances_
```

Out[76]:

```
array([0.0544776 , 0.01070401, 0.0733695 , 0.01465252, 0.0126054 ,
       0.83419097])
```

In [77]:

```
imp_df=pd.DataFrame({"varname":x_train.columns,"Imp":rf_best.feature_importances_})
imp_df.sort_values(by="Imp",ascending=False)
```

Out[77]:

| | varname | Imp |
|---|---|---|
| 5 | charges | 0.834191 |
| 2 | bmi | 0.073369 |
| 0 | age | 0.054478 |
| 3 | children | 0.014653 |
| 4 | region | 0.012605 |
| 1 | sex | 0.010704 |

```
ass = No'),
 Text(0.875, 0.2777777777777778, 'gini = 0.408\nsamples = 5\nvalue = [5, 2]\ncl
ass = Yes'),
 Text(0.95, 0.3888888888888889, 'region <= 2.5\ngini = 0.067\nsamples = 16\nval
ue = [1, 28]\nclass = No'),
 Text(0.925, 0.2777777777777778, 'gini = 0.198\nsamples = 6\nvalue = [1, 8]\ncl
ass = No'),
 Text(0.975, 0.2777777777777778, 'gini = 0.0\nsamples = 10\nvalue = [0, 20]\ncl
ass = No')]
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

# Conclusion

In [ ]:

```
From The Given Insurance Dataset,we have formed on different models like Linear
Regression,Logistic Regression,Ridge Regression,
Lasso Regression,Elastic Net,Random Forest,Decision Tree.By observing the score or model
prediction in this models,
In Random forest model have the best score and best accuracy.
```

In [ ]: