Problem Statement:-To Find the suitable model for a Insurance DataSet

importing Packages

In [1]:

```
#importing packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
```

Data Collection:- To read the dataset

In [2]:

```
df=pd.read_csv(r"C:\Users\lenovo\Downloads\insurance.csv")
df
```

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

Data Cleaning and Preprocessing

In [3]:

```
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 [4]:

```
df.head()
```

Out[4]:

	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 [5]:

```
df.tail()
```

Out[5]:

	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 [6]:

```
df.columns
```

Out[6]:

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

In [7]:

```
df.describe()
```

Out[7]:

	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 [8]:

```
df.shape
```

Out[8]:

```
(1338, 7)
```

To find the duplicate values in Dataset

In [9]:

```
 df.duplicated().sum()
```

Out[9]:

```
1
```

In [10]:

```
df.isnull().sum()
```

Out[10]:

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

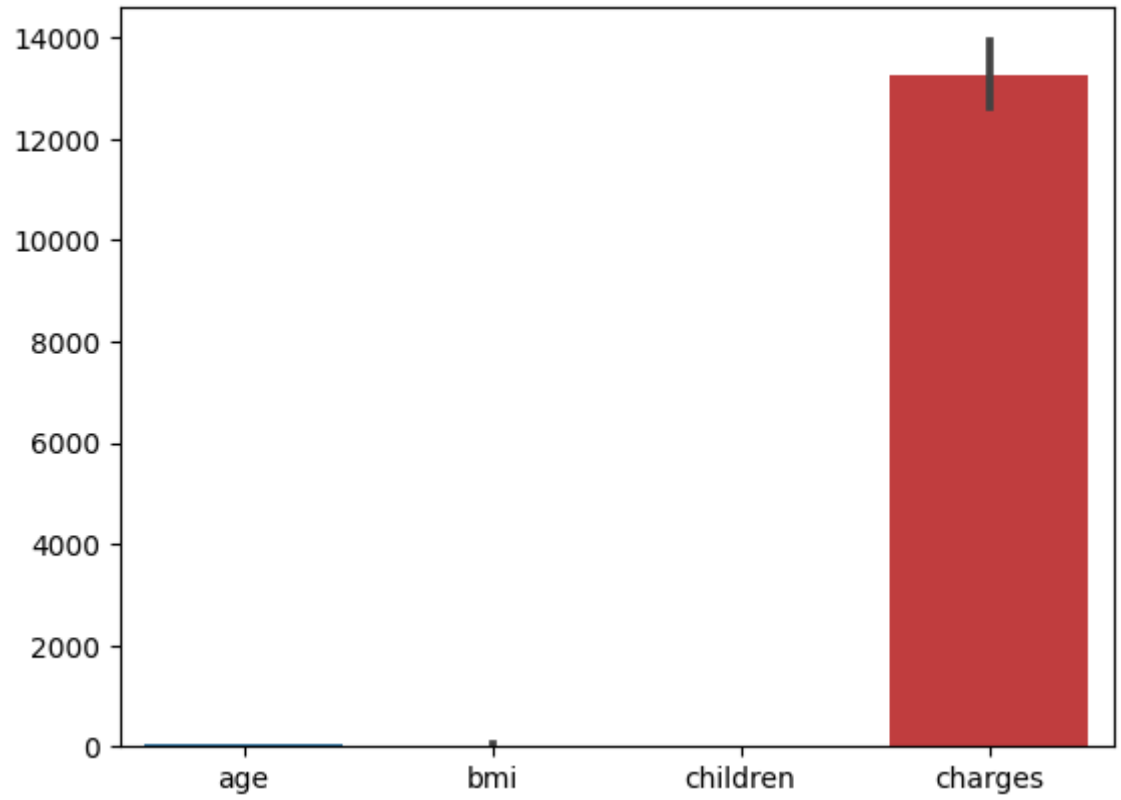
Data Visualization:- To Visualize the data

In [11]:

```
#Exploratory Data Analysis
sns.barplot(df)
```

Out[11]:

```
<Axes: >
```

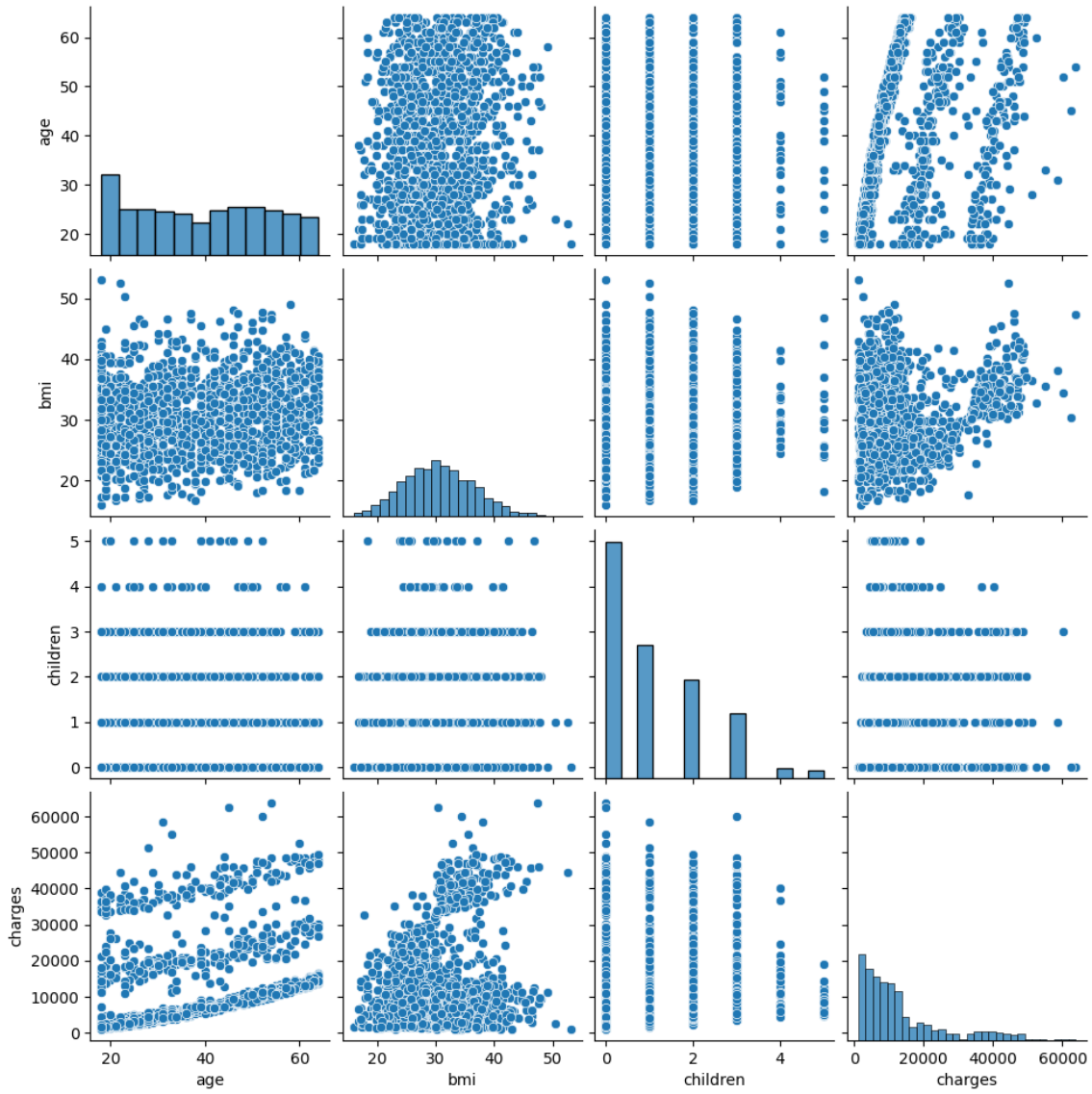


In [12]:

```
sns.pairplot(df)
```

Out[12]:

```
<seaborn.axisgrid.PairGrid at 0x203d1b241d0>
```

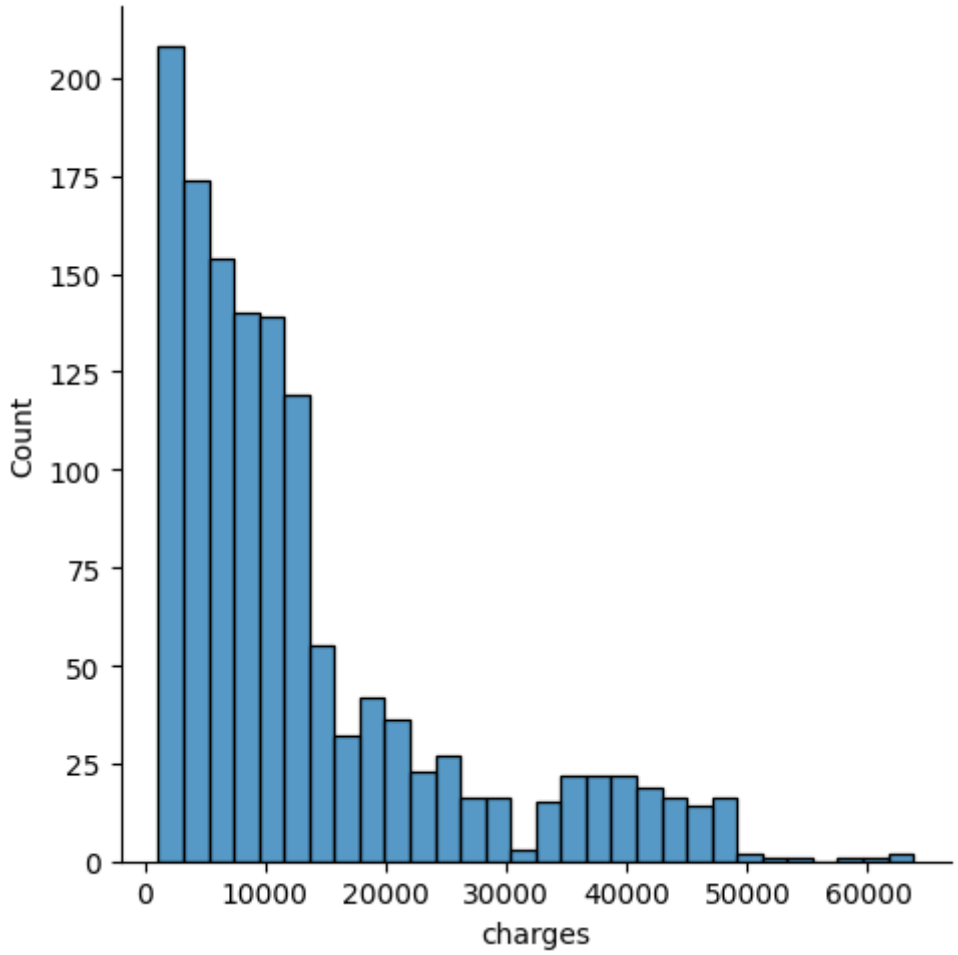


In [13]:

```
sns.displot(df["charges"])
```

Out[13]:

```
<seaborn.axisgrid.FacetGrid at 0x203d4564e50>
```



In [14]:

```
df["region"].value_counts()
```

Out[14]:

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

In [15]:

```
df["smoker"].value_counts()
```

Out[15]:

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

In [16]:

```
df["sex"].value_counts()
```

Out[16]:

```
sex
male      676
female    662
Name: count, dtype: int64
```

In [17]:

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

Out[17]:

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

1338 rows × 7 columns

In [18]:

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

Out[18]:

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

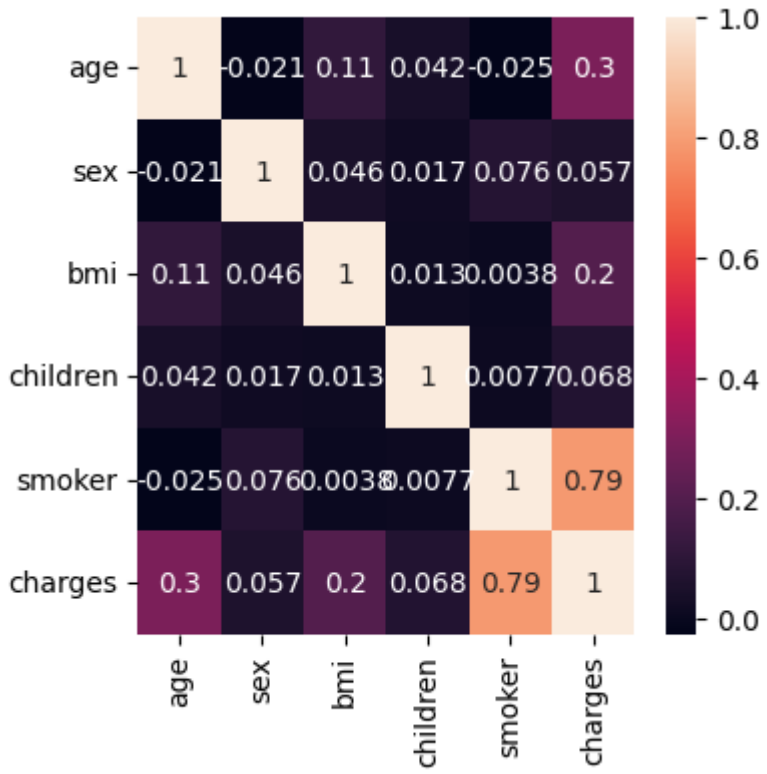
1338 rows × 7 columns

In [19]:

```
idf=df[['age', 'sex', 'bmi', 'children', 'smoker', 'charges']]
plt.figure(figsize=(4,4))
sns.heatmap(idf.corr(),annot=True)
```

Out[19]:

```
<Axes: >
```



Feature Scaling:- To split the data into train data and test data

In [20]:

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

Applying Linear Regression to the DataSet

In [21]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=100)
```

In [22]:

```
from sklearn.linear_model import LinearRegression
regr=LinearRegression()
regr.fit(X_train,y_train)
print(regr.intercept_)
coeff_df=pd.DataFrame(regr.coef_,X.columns,columns=['coefficient'])
coeff_df
```

```
-10719.483493479494
```

Out[22]:

	coefficient
age	259.757578
sex	18.216925
bmi	277.903898
children	461.169867
smoker	23981.741027

In [23]:

```
score=regr.score(X_test,y_test)
print(score)
```

```
0.780095696440481
```

In [24]:

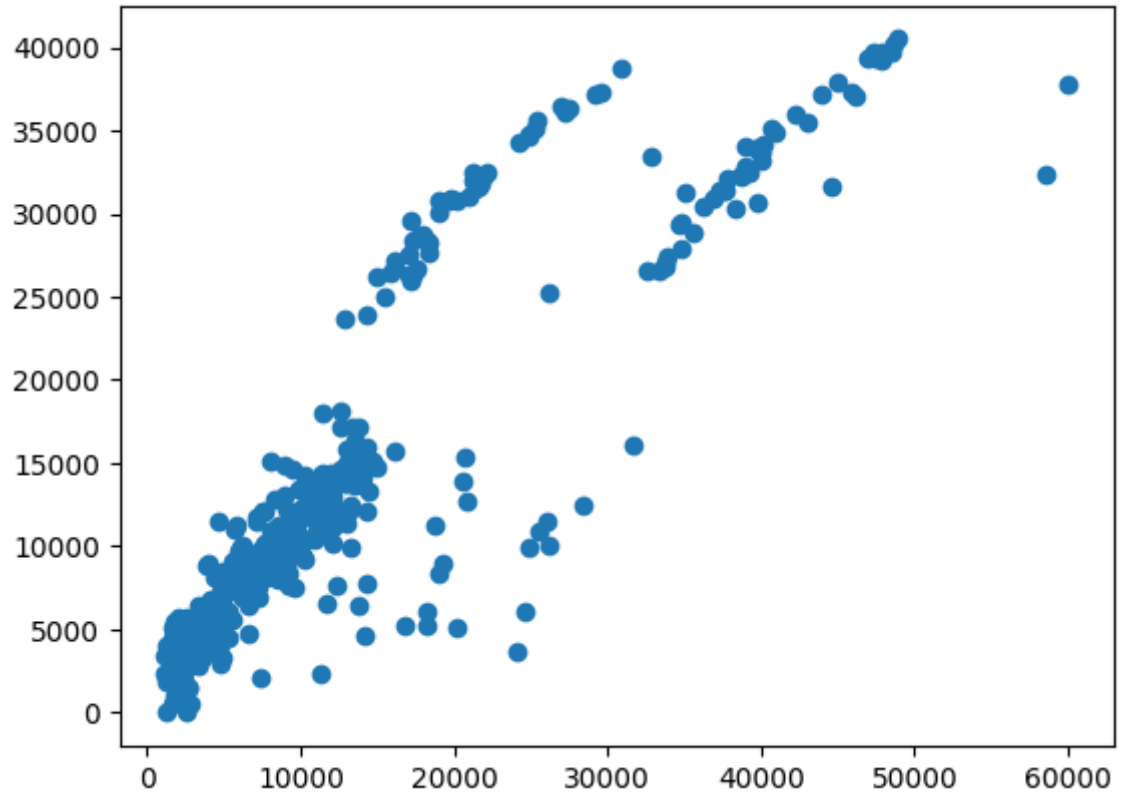
```
pr=regr.predict(X_test)
```

In [25]:

```
plt.scatter(y_test,pr)
```

Out[25]:

```
<matplotlib.collections.PathCollection at 0x203bed22a10>
```



In [26]:

```
x=np.array(df['smoker']).reshape(-1,1)
y=np.array(df['charges']).reshape(-1,1)
df.dropna(inplace=True)
```

In [27]:

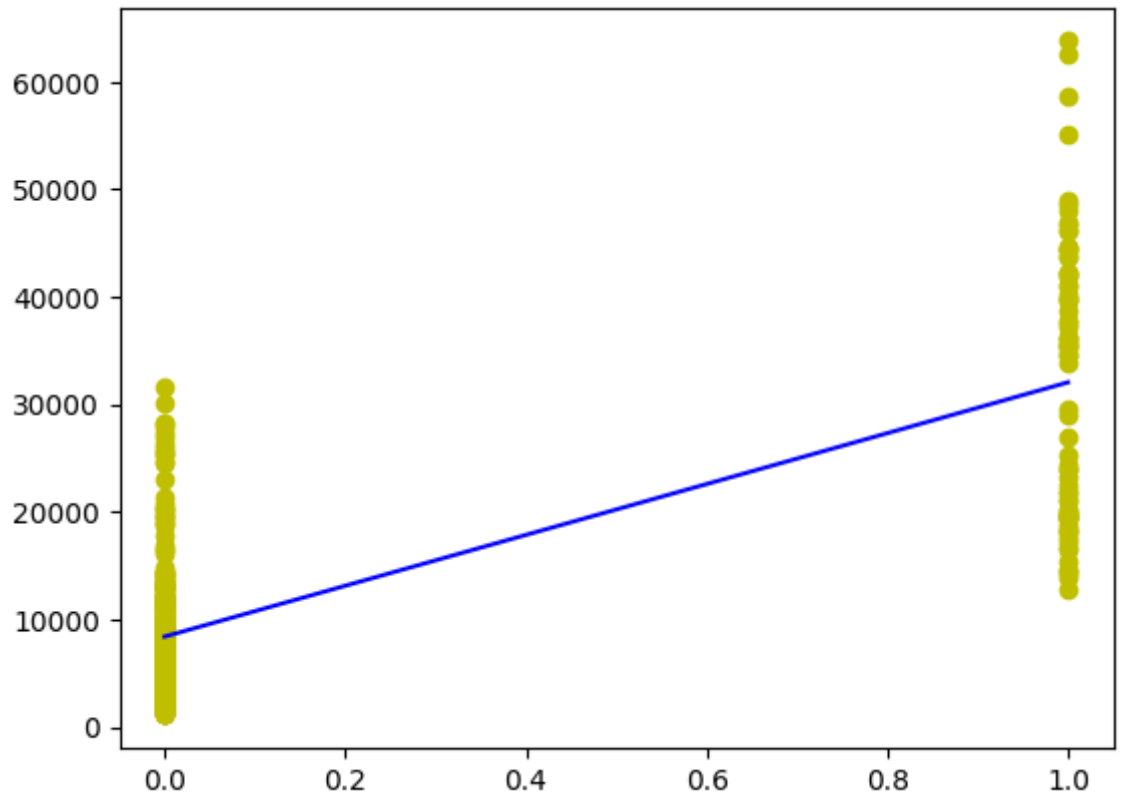
```
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
regr.fit(X_train,y_train)
regr.fit(X_train,y_train)
```

Out[27]:

```
LinearRegression
LinearRegression()
```

In [28]:

```
y_pred=regr.predict(X_test)
plt.scatter(X_test,y_test,color='y')
plt.plot(X_test,y_pred,color='b')
plt.show()
```



In Linear Regression Model we did not get the accuracy. we can implement Logistic Regression.

Applying Logistic Regression to Dataset

In [29]:

```
x=np.array(df['charges']).reshape(-1,1)
y=np.array(df['smoker']).reshape(-1,1)
df.dropna(inplace=True)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=1)
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression(max_iter=10000)
```

In [30]:

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

```
C:\Users\lenovo\AppData\Local\Programs\Python\Python311\Lib\site-packages
\sklearn\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, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[30]:

```
LogisticRegression
LogisticRegression(max_iter=10000)
```

In [31]:

```
LogisticRegression(max_iter=10000)
```

Out[31]:

```
LogisticRegression
LogisticRegression(max_iter=10000)
```

In [32]:

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

```
0.8930348258706468
```

In [33]:

```
pip install statsmodels
```

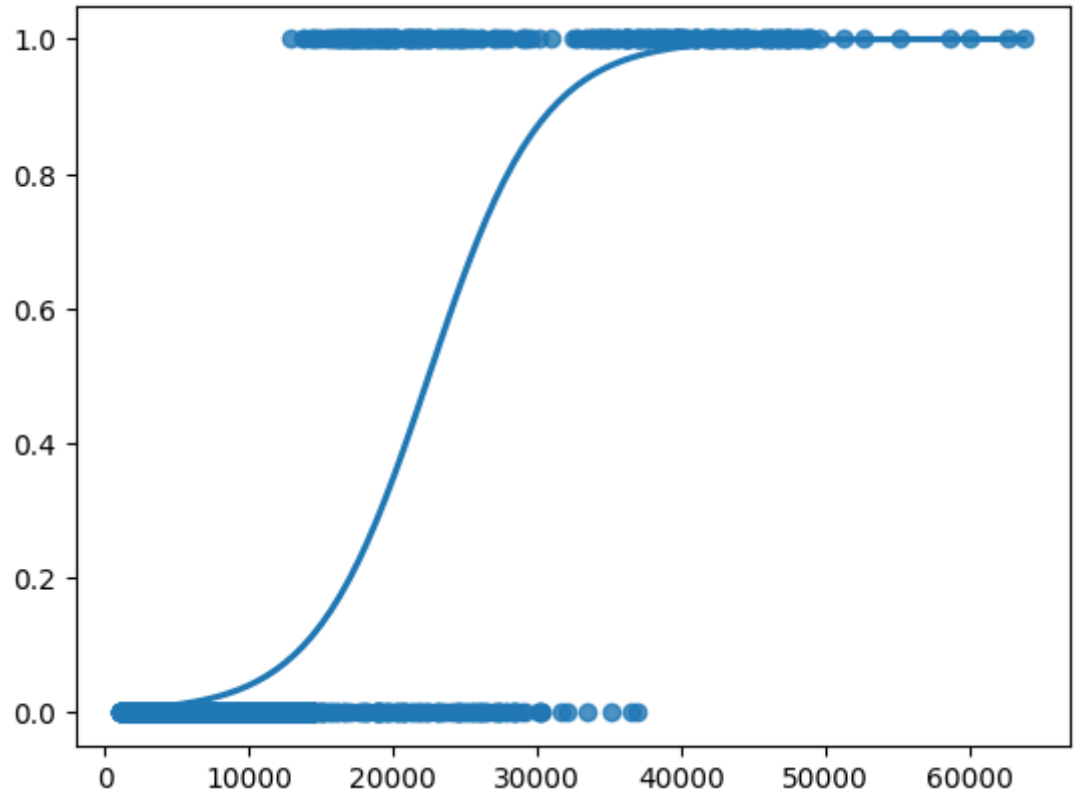
```
Requirement already satisfied: statsmodels in c:\users\lenovo\appdata\loca
l\programs\python\python311\lib\site-packages (0.14.0)
Requirement already satisfied: numpy>=1.18 in c:\users\lenovo\appdata\loca
l\programs\python\python311\lib\site-packages (from statsmodels) (1.24.3)
Requirement already satisfied: scipy!=1.9.2,>=1.4 in c:\users\lenovo\appda
ta\local\programs\python\python311\lib\site-packages (from statsmodels)
(1.10.1)
Requirement already satisfied: pandas>=1.0 in c:\users\lenovo\appdata\loca
l\programs\python\python311\lib\site-packages (from statsmodels) (2.0.1)
Requirement already satisfied: patsy>=0.5.2 in c:\users\lenovo\appdata\loc
al\programs\python\python311\lib\site-packages (from statsmodels) (0.5.3)
Requirement already satisfied: packaging>=21.3 in c:\users\lenovo\appdata
\local\programs\python\python311\lib\site-packages (from statsmodels) (23.
1)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\lenovo\a
ppdata\local\programs\python\python311\lib\site-packages (from pandas>=1.0
->statsmodels) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\lenovo\appdata\loc
al\programs\python\python311\lib\site-packages (from pandas>=1.0->statsmod
els) (2023.3)
Requirement already satisfied: tzdata>=2022.1 in c:\users\lenovo\appdata\l
ocal\programs\python\python311\lib\site-packages (from pandas>=1.0->statsm
odels) (2023.3)
Requirement already satisfied: six in c:\users\lenovo\appdata\local\progra
ms\python\python311\lib\site-packages (from patsy>=0.5.2->statsmodels) (1.
16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [34]:

```
sns.regplot(x=x,y=y,data=df,logistic=True,ci=None)
```

Out[34]:

```
<Axes: >
```



Here,we got best fit curve for Logistic Regression.So we can implement Decision Tree and Random Forest to check accuracy.

Decision Tree

In [35]:

```
from sklearn.tree import DecisionTreeClassifier
clf=DecisionTreeClassifier(random_state=0)
clf.fit(x_train,y_train)
```

Out[35]:

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

In [36]:

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

Out[36]:

DecisionTreeClassifier

DecisionTreeClassifier(random_state=0)

In [37]:

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

```
0.8880597014925373
```

Random Forest

In [38]:

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

```
C:\Users\lenovo\AppData\Local\Temp\ipykernel_30144\4104924521.py:3: DataCo
nversionWarning: A column-vector y was passed when a 1d array was expecte
d. Please change the shape of y to (n_samples,), for example using ravel
().
  rfc.fit(X_train,y_train)
```

Out[38]:

RandomForestClassifier

RandomForestClassifier()

In [39]:

```
RandomForestClassifier()
```

Out[39]:

RandomForestClassifier

RandomForestClassifier()

In [40]:

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

In [41]:

```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rfc,param_grid=params,cv=2,scoring="accuracy")
```

In [42]:

```
grid_search.fit(X_train,y_train)
```

```
C:\Users\lenovo\AppData\Local\Programs\Python\Python311\Lib\site-packag
es\sklearn\model_selection\_validation.py:686: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change
the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\lenovo\AppData\Local\Programs\Python\Python311\Lib\site-packag
es\sklearn\model_selection\_validation.py:686: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change
the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\lenovo\AppData\Local\Programs\Python\Python311\Lib\site-packag
es\sklearn\model_selection\_validation.py:686: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change
the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\lenovo\AppData\Local\Programs\Python\Python311\Lib\site-packag
es\sklearn\model_selection\_validation.py:686: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change
the shape of y to (n_samples,), for example using ravel().
```

In [43]:

```
grid_search.best_score_
```

Out[43]:

```
0.7938034188034188
```

In [44]:

```
rf_best=grid_search.best_estimator_
rf_best
```

Out[44]:

```
RandomForestClassifier
RandomForestClassifier(max_depth=2, min_samples_leaf=5, n_estimators=10)
```

In [45]:

```
RandomForestClassifier(max_depth=2, min_samples_leaf=5, n_estimators=10)
```

Out[45]:

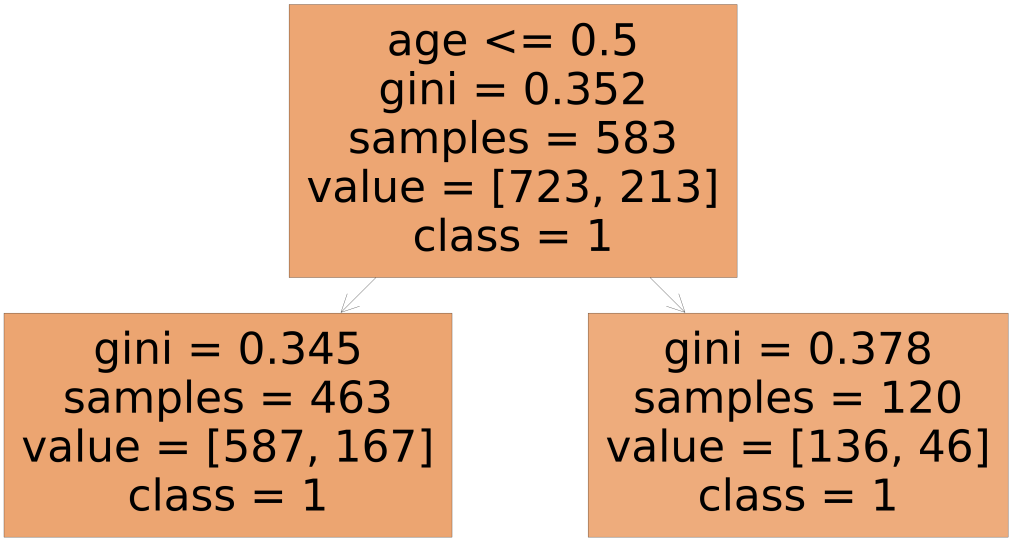
```
RandomForestClassifier
RandomForestClassifier(max_depth=2, min_samples_leaf=5, n_estimators=10)
```

In [46]:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[4],feature_names=X.columns,class_names=['1','0'],filled=Tr
```

Out[46]:

```
[Text(0.5, 0.75, 'age <= 0.5\ngini = 0.352\nsamples = 583\nvalue = [723, 2
13]\nclass = 1'),
 Text(0.25, 0.25, 'gini = 0.345\nsamples = 463\nvalue = [587, 167]\nclass
= 1'),
 Text(0.75, 0.25, 'gini = 0.378\nsamples = 120\nvalue = [136, 46]\nclass =
1')]
```



In [47]:

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

```
0.7985074626865671
```

Conclusion:- Based on accuracy scores of all models that are implemented above we can finaly conclude that "Logistic Regression Model" is best model and fit for the given Insurance DataSet.

In [ ]:

```

```