PROBLEM STATEMENT:Which model is suitable for insurance dataset

Importing Packages

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt,seaborn as sns
```

In [3]:
```
df=pd.read_csv(r"C:\Users\Dell\Downloads\insurance.csv")
df
```

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
...	...	...	...	...	...	...	...
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 collection and preprocessing

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.shape
```

Out[6]:
```
(1338, 7)
```

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

In [8]:
```
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 [9]:
```
df.describe()
```

Out[9]:

	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

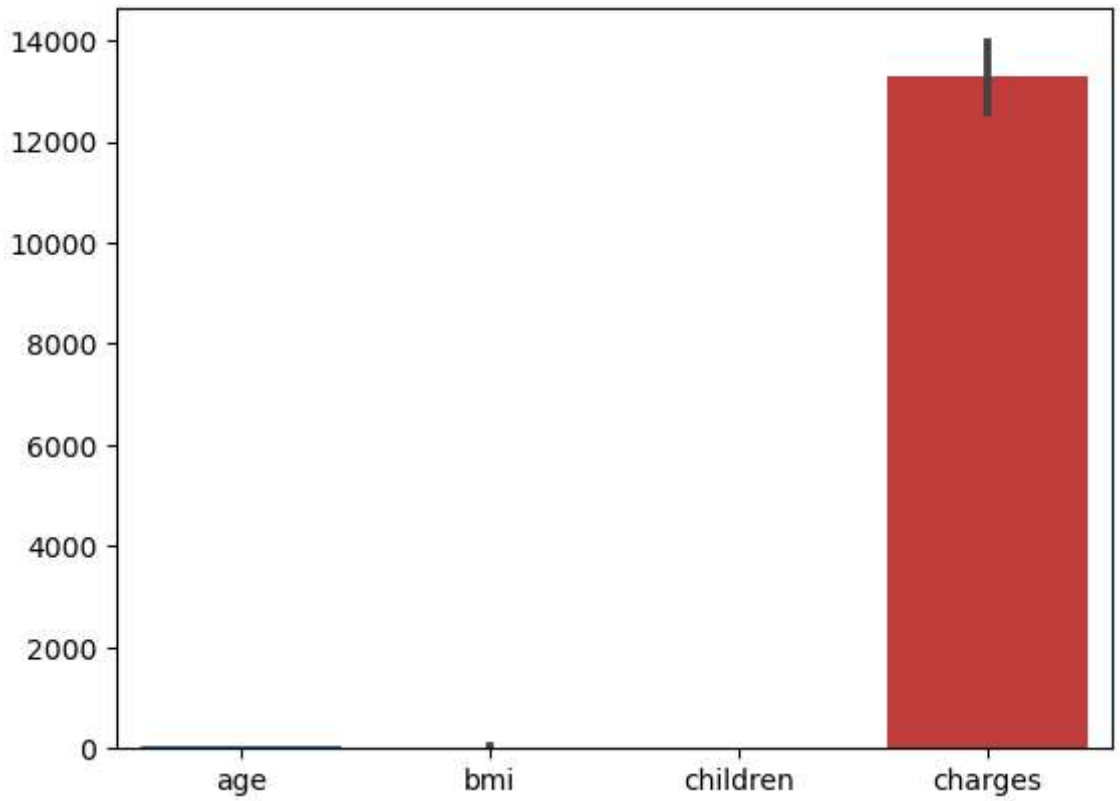
In [10]:
```
df.columns
```

Out[10]:
```
Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges'], dtyp
e='object')
```

Data Visualization

In [11]:
```
sns.barplot(df)
```

Out[11]: <Axes: >



In [12]:
```
smoker={"smoker":{"yes":1,"no":0}}
df=df.replace(smoker)
df
```

Out[12]:

	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 [13]:
```
sex={"sex":{"male":1,"female":0}}
df=df.replace(sex)
df
```

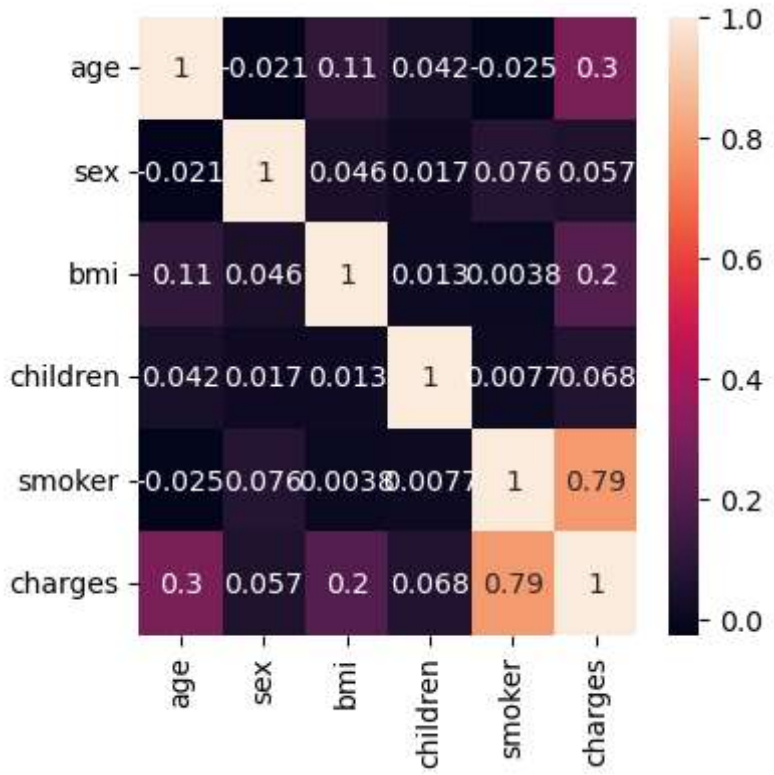
Out[13]:

	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 [15]:
```
idf=df[['age','sex','bmi','children','smoker','charges']]
plt.figure(figsize=(4,4))
sns.heatmap(idf.corr(),annot=True)
```

Out[15]: <Axes: >



Feature Scaling:To Split the data into training data and testing data

In [16]:
```
#training the model
X=df[['age','sex','bmi','children','smoker']]
y=df['charges']
```

Applying Linear Regression

In [17]:
```
#Linear regression
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=
```

In [19]:
```
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=['coefficent'])
coeff_df
```

```
-10719.483493479494
```

Out[19]:

	coefficent
age	259.757578
sex	18.216925
bmi	277.903898
children	461.169867
smoker	23981.741027

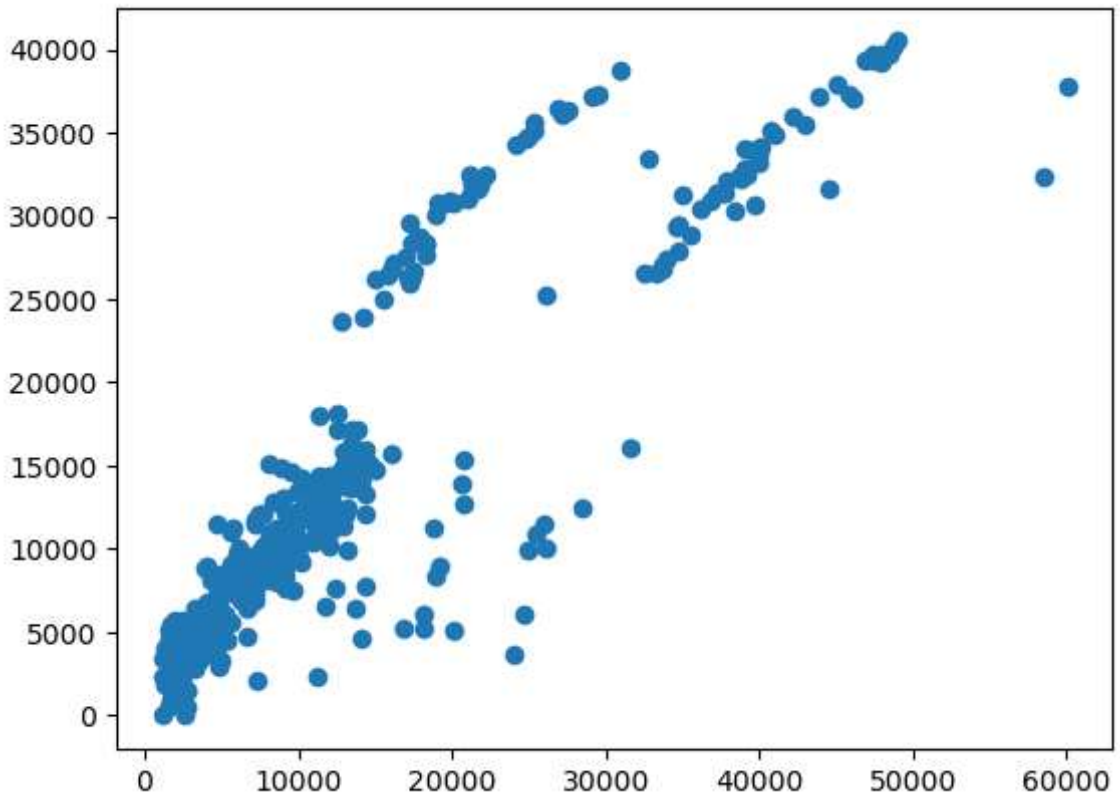
In [20]:
```
score=regr.score(X_test,y_test)
print(score)
```

```
0.780095696440481
```

In [21]:
```
predictions=regr.predict(X_test)
```

In [22]:
```
plt.scatter(y_test,predictions)
```

Out[22]:
```
<matplotlib.collections.PathCollection at 0x22ce3963700>
```



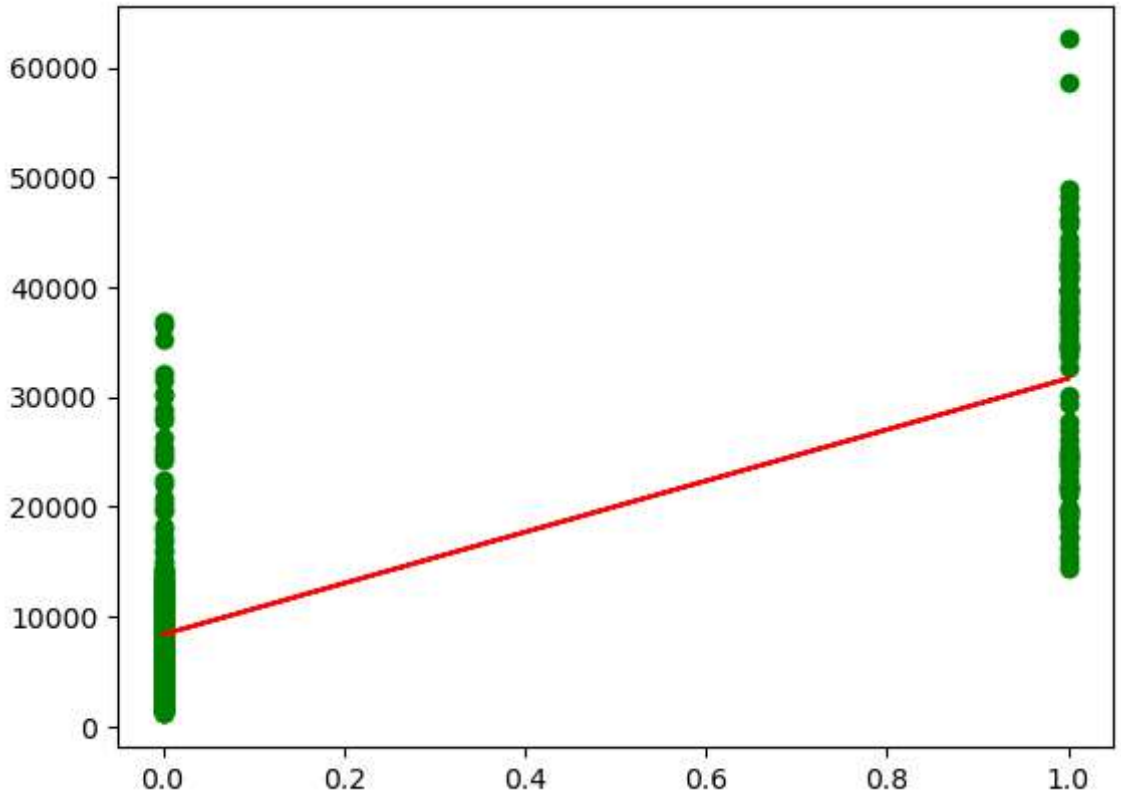
In [23]:
```
x=np.array(df['smoker']).reshape(-1,1)
y=np.array(df['charges']).reshape(-1,1)
df.dropna(inplace=True)
```

In [24]:
```
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[24]:
```
▾ LinearRegression

LinearRegression()
```

In [27]:
```
y_pred=regr.predict(X_test)
plt.scatter(X_test,y_test,color='g')
plt.plot(X_test,y_pred,color='r')
plt.show()
```



Since we did not get the accuracy for Linear Regression now we are going to implement Logistic Regression

Logistic Regression

In [28]:
```
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=
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression(max_iter=10000)
```

In [30]:
```
lr.fit(X_train,y_train)
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:1143:
DataConversionWarning: A column-vector y was passed when a 1d array was expe
cted. Please change the shape of y to (n_samples, ), for example using ravel
().
  y = column_or_1d(y, warn=True)
```

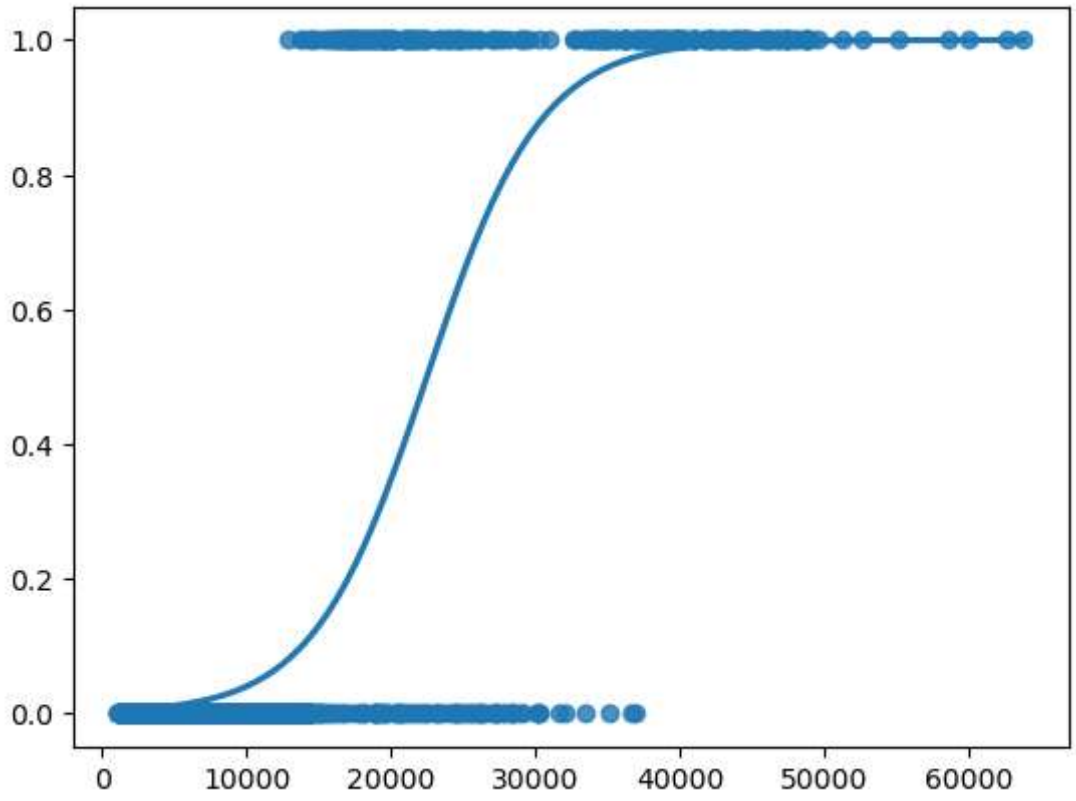
Out[30]:
```
LogisticRegression
LogisticRegression(max_iter=10000)
```

In [32]:
```
score=lr.score(X_test,y_test)
print(score)
```

```
0.8930348258706468
```

In [33]:
```
sns.regplot(x=x,y=y,data=df,logistic=True,ci=None)
```

Out[33]: <Axes: >



We got the best fit curve for Logistic Regression now we are going to check that if we may get better accuracy by implementing Decision Tree

Decision Tree

In [36]:
```
from sklearn.tree import DecisionTreeClassifier
clf=DecisionTreeClassifier(random_state=0)
clf.fit(X_train,y_train)
```

Out[36]:
```
DecisionTreeClassifier

DecisionTreeClassifier(random_state=0)
```

In [37]:
```
score=clf.score(X_test,y_test)
print(score)
```

```
0.8880597014925373
```

CONCLUSION:Based on accuracy scores of all models that were implemented we can conclude that "Logistic regression" is the best model for the given dataset