Problem statement:

```
To predict how best the data fits and which model suits
```

Linear Regression

Data Collection

In [37]:

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

import warnings
warnings.filterwarnings("ignore")
```

In [38]:

```python
df=pd.read_csv(r"C:\Users\91949\Downloads\insurance.csv")
df
```

Out[38]:

	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 preproccesing

In [39]:

```
df.head()
```

Out[39]:

	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 [40]:

```
df.tail()
```

Out[40]:

	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 [41]:

```
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 [42]:

```
df.shape
```

Out[42]:

```
(1338, 7)
```

In [43]:

```
df.describe()
```

Out[43]:

	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

Finding null values

In [44]:

```
df.isnull().any()
```

Out[44]:

```
age         False
sex         False
bmi         False
children    False
smoker      False
region      False
charges     False
dtype: bool
```

In [45]:

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

Out[45]:

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

Finding Duplicate values

In [46]:

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

Out[46]:

```
1
```

In [47]:

```
df=df.drop_duplicates()
df
```

Out[47]:

	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

1337 rows × 7 columns

In [48]:

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

```
      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

[1337 rows x 7 columns]
```

In [49]:

```
x=df[['bmi']]
y=df['charges']
```

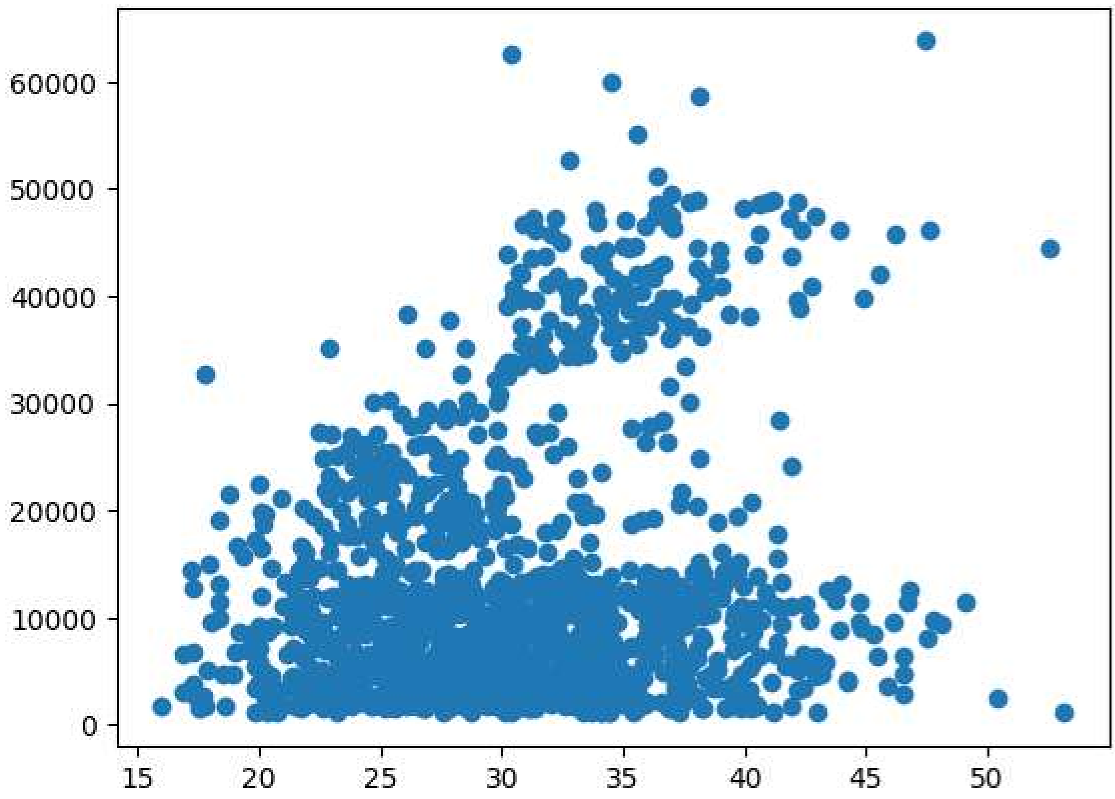
In [50]:

```
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)
```

Data visualization

In [51]:

```
plt.scatter(df['bmi'],df['charges'])
plt.show()
```



In [52]:

```
x.head(20)
```

Out[52]:

	bmi
0	27.900
1	33.770
2	33.000
3	22.705
4	28.880
5	25.740
6	33.440
7	27.740
8	29.830
9	25.840
10	26.220
11	26.290
12	34.400
13	39.820
14	42.130
15	24.600
16	30.780
17	23.845
18	40.300
19	35.300

In [53]:

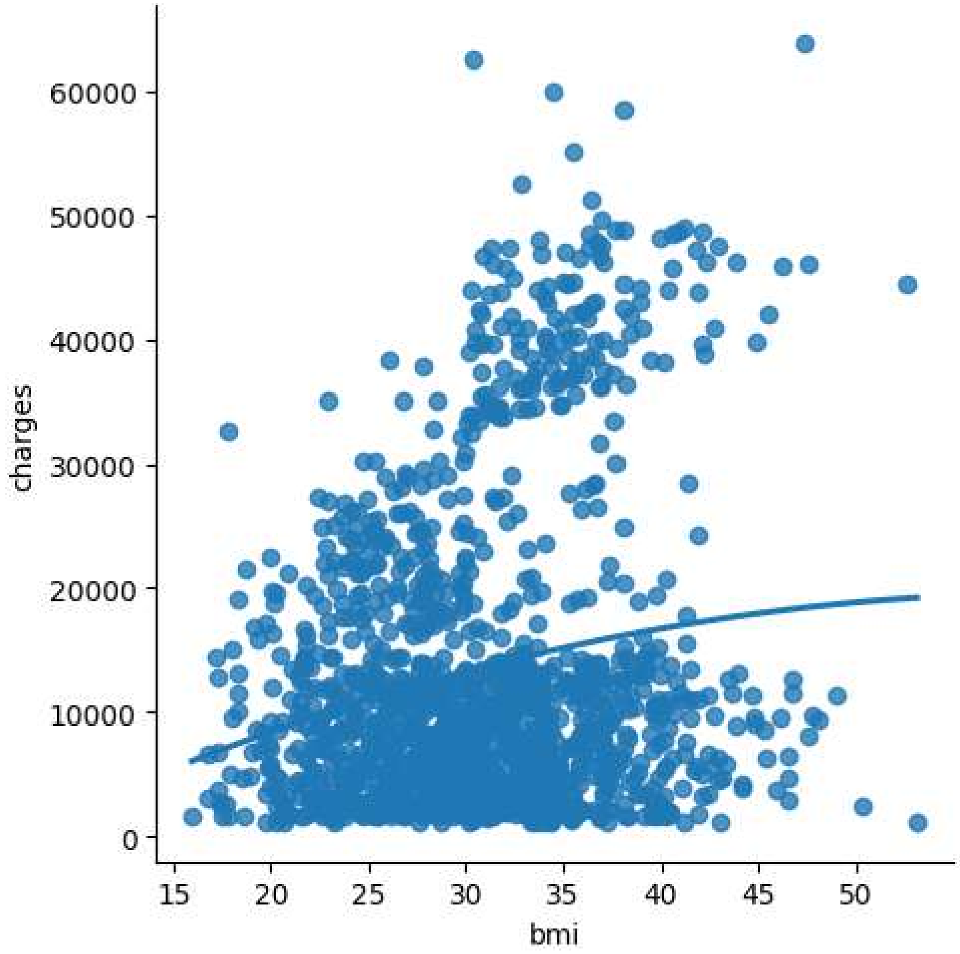
```
y.head(15)
```

Out[53]:

```
0     16884.92400
1      1725.55230
2      4449.46200
3     21984.47061
4      3866.85520
5      3756.62160
6      8240.58960
7      7281.50560
8      6406.41070
9     28923.13692
10     2721.32080
11    27808.72510
12     1826.84300
13    11090.71780
14    39611.75770
Name: charges, dtype: float64
```

In [54]:

```
sns.lmplot(x='bmi',y='charges', order=2,data=df, ci=None)
plt.show()
```



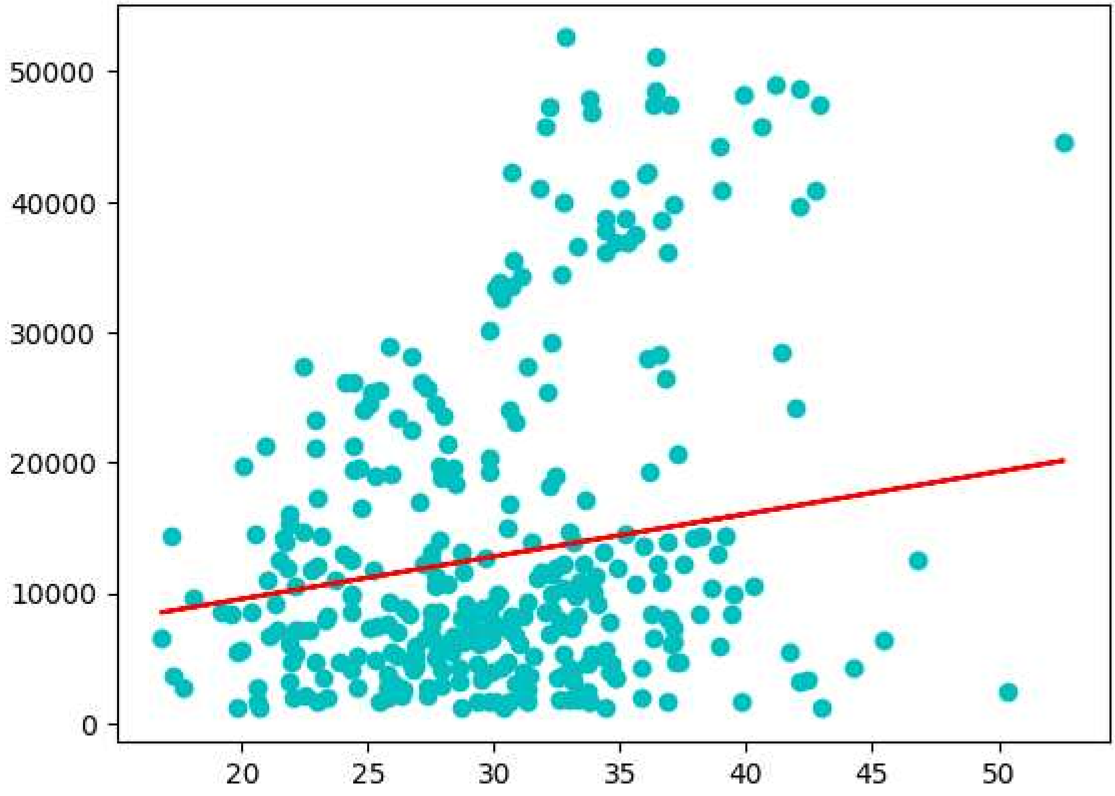
In [55]:

```python
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25, random_state=0)
lr=LinearRegression()
lr.fit(x_train,y_train)
print(lr.score (x_test,y_test))
```

```
0.060963613622574186
```

In [56]:

```python
y_pred=lr.predict(x_test)
plt.scatter(x_test,y_test,color='c')
plt.plot(x_test,y_pred, color='r')
plt.show()
```



In [57]:

```python
df500.fillna (method='ffill', inplace=True)
```

In [58]:

```python
x=np.array(df500["bmi"]).reshape(-1,1)
y=np.array(df500['charges']).reshape(-1,1)
```

In [59]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
lr=LinearRegression()
lr.fit(x_train,y_train)
y_pred=lr.predict(x_test)
r2=r2_score(y_test,y_pred)
print(r2)
```

```
0.060963613622574186
```

Ridge Regression

In [60]:

```
df.columns
```

Out[60]:

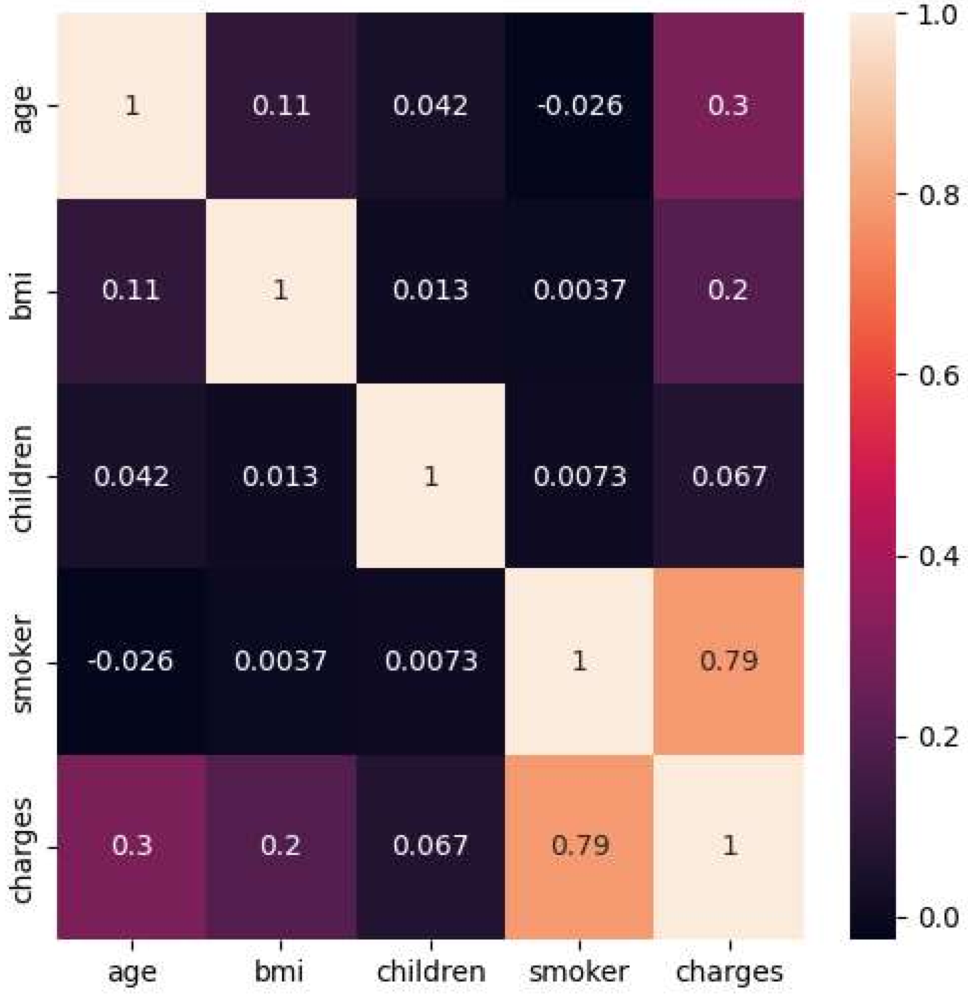
```
Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges'], d
type='object')
```

In [61]:

```
from sklearn.linear_model import Lasso, Ridge
I=df[['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges']]
plt.figure(figsize=(6,6))
sns.heatmap(I.corr(),annot=True)
```

Out[61]:

```
<Axes: >
```



In [62]:

```
features=df.columns [0:1]
target=df.columns[-1]
```

In [63]:

```
x=df[features].values
y=df[target].values
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30, random_state=1)
print("The dimension of X_train is {}".format(x_train.shape))
print("The dimension of X_test is {}".format(x_test.shape))
```

```
The dimension of X_train is (935, 1)
The dimension of X_test is (402, 1)
```

In [64]:

```
lr = LinearRegression()
lr.fit(x_train, y_train)
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 1r model is {}".format(test_score_lr))
```

```
Linear Regression Model:

The train score for lr model is 0.09099234134544743
The test score for 1r model is 0.07338609034045929
```

In [65]:

```
ridgeReg = Ridge(alpha=10)
ridgeReg.fit(x_train,y_train)
#train and test scorefor 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 0.09099234107282062
The test score for ridge model is 0.07338709056396597
```

Lasso Regression

In [66]:

```
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("\nLasso Model:\n")
print("The train score for lasso model is {}".format(train_score_ls))
print("The test score for lasso model is {}".format(test_score_ls))
```

```
Lasso Model:

The train score for lasso model is 0.0909923379381713
The test score for lasso model is 0.07338962361681955
```

Logistic Regression

In [67]:

```
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 [68]:

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

Out[68]:

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

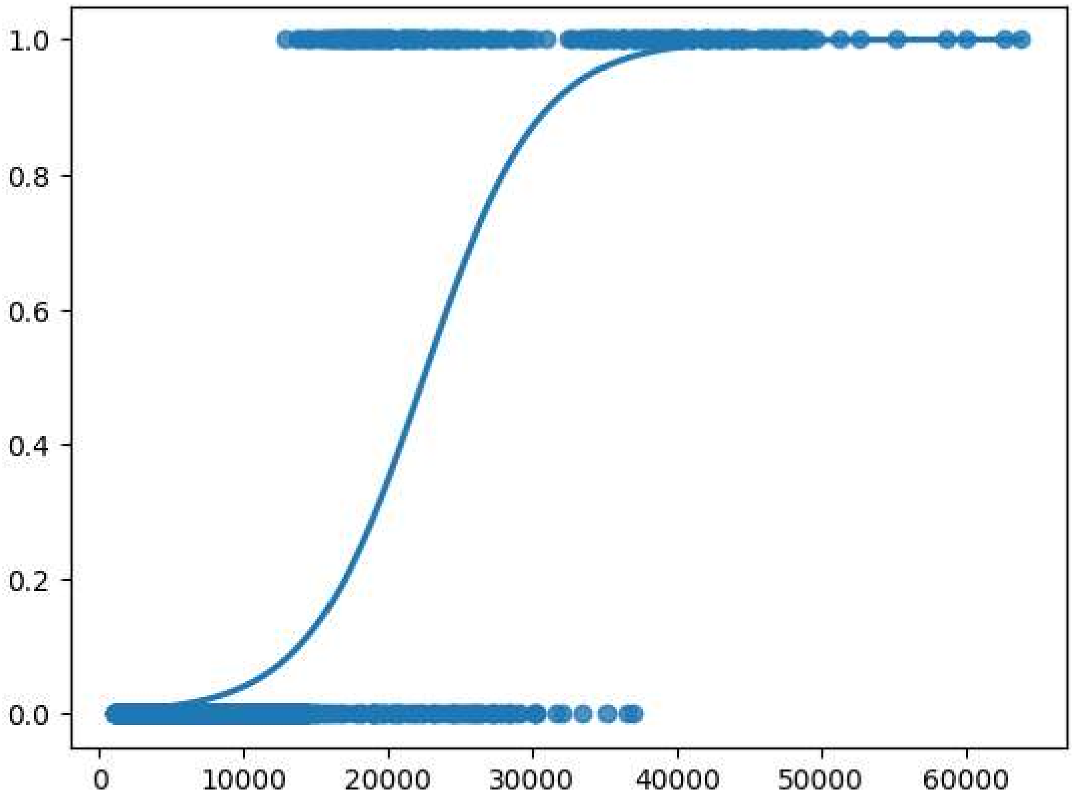
In [69]:

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

```
0.9253731343283582
```

In [70]:

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



Decision Tree

In [71]:

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

Out[71]:

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

In [72]:

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

```
0.900497512437811
```

RANDOM FOREST

In [73]:

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

Out[73]:

```
▾ RandomForestClassifier

RandomForestClassifier()
```

In [74]:

```
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 [75]:

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

In [76]:

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

Out[76]:

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

In [77]:

```
grid_search.best_score_
```

Out[77]:

```
0.9219193250242501
```

In [78]:

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

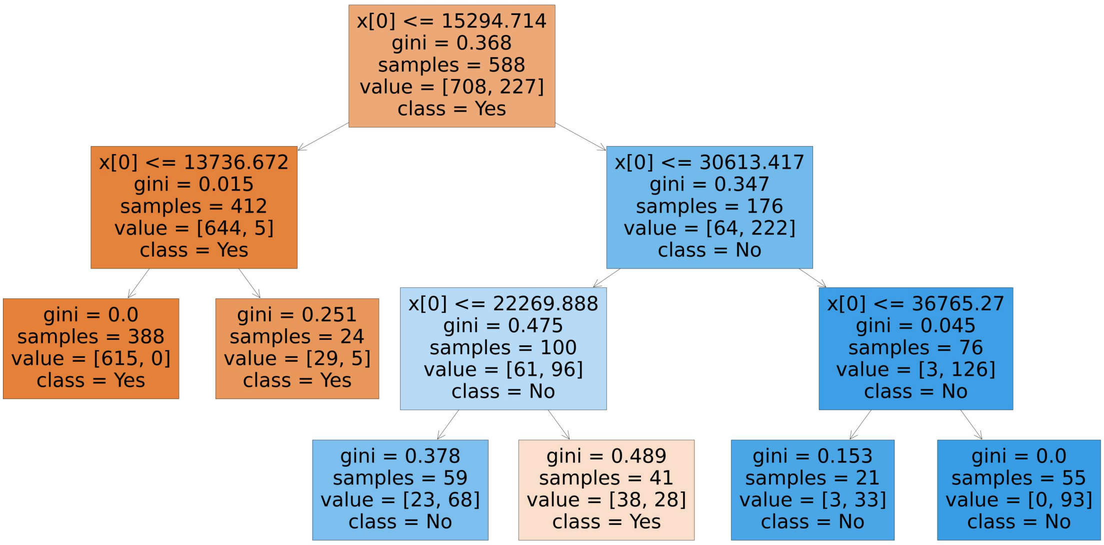
Out[78]:

```
▾                    RandomForestClassifier

RandomForestClassifier(max_depth=3, min_samples_leaf=20)
```

In [79]:

```python
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[5],class_names=['Yes',"No"],filled=True);
```



In [80]:

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

```
0.900497512437811
```

Conclusion:

Finally we conclude that based on the accuracy of all models which we are implemented above the "Logistic Regression" is the best model.