Problem Statement:To Predict How Best Data Fits

Data Collection

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing,svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [2]:
```
df=pd.read_csv(r"C:\Users\Welcome\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.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]:
```
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]:
```
df.describe
```

Out[5]:
```
<bound method NDFrame.describe of       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 x 7 columns]>
```

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

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

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

Out[8]:
```
age         False
sex         False
bmi         False
children    False
smoker      False
region      False
charges     False
dtype: bool
```

In [9]:
```
df.isna().sum()
```

Out[9]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

In [10]:
```
df['region'].value_counts()
```

Out[10]:
```
region
southeast    364
southwest    325
northwest    325
northeast    324
Name: count, dtype: int64
```

In [11]:
```
convert={"smoker":{"yes":1,"no":0}}
df=df.replace(convert)
df
```

Out[11]:

	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 [39]:
```
convert={"sex":{"female":1,"male":0}}
df=df.replace(convert)
df
```

Out[39]:

	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
...	...	...	...	...	...	...	...
695	26	1	40.185	0	0	northwest	3201.24515
696	53	1	32.300	2	0	northeast	29186.48236
697	41	0	35.750	1	1	southeast	40273.64550
698	56	0	33.725	0	0	northwest	10976.24575
699	23	1	39.270	2	0	southeast	3500.61230

700 rows × 7 columns

In [43]:
```
convert={"region":{"southeast":0,"northeast":1,"southwest":2,"northwest":3}}
df=df.replace(convert)
df
```

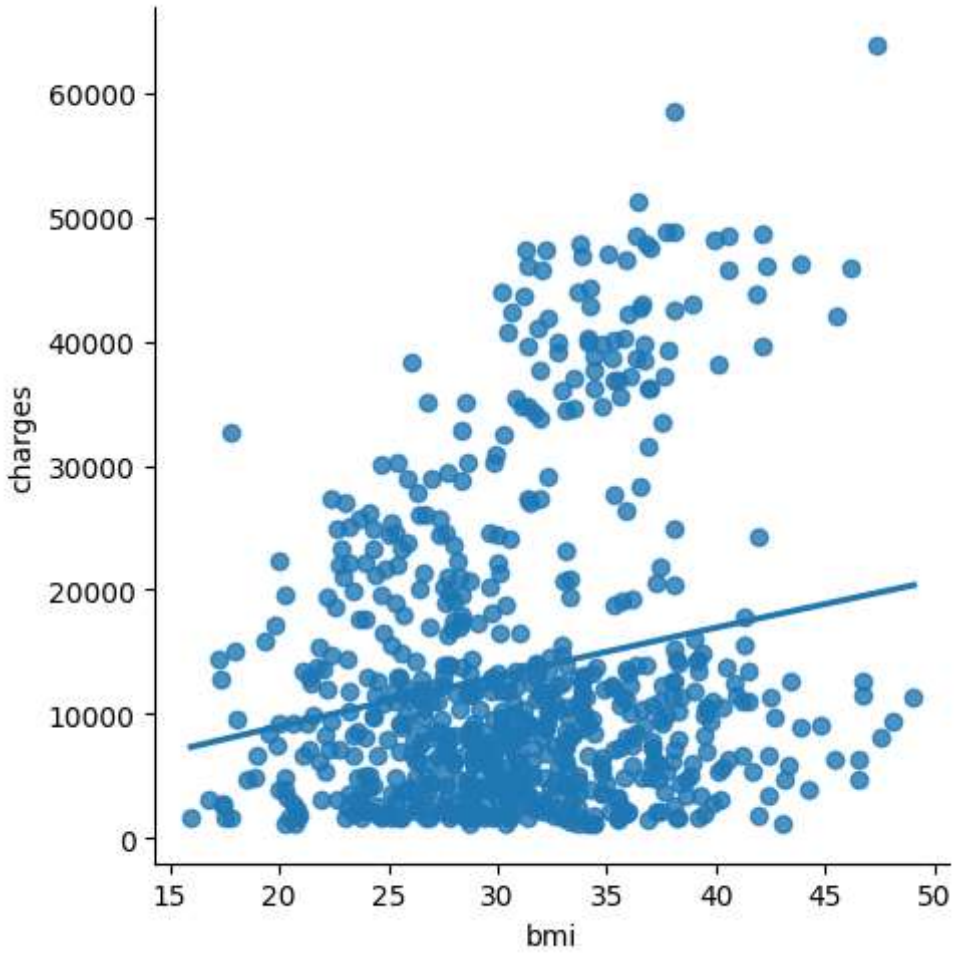
Out[43]:

	age	sex	bmi	children	smoker	region	charges
0	19	1	27.900	0	1	2	16884.92400
1	18	0	33.770	1	0	0	1725.55230
2	28	0	33.000	3	0	0	4449.46200
3	33	0	22.705	0	0	3	21984.47061
4	32	0	28.880	0	0	3	3866.85520
...	...	...	...	...	...	...	...
695	26	1	40.185	0	0	3	3201.24515
696	53	1	32.300	2	0	1	29186.48236
697	41	0	35.750	1	1	0	40273.64550
698	56	0	33.725	0	0	3	10976.24575
699	23	1	39.270	2	0	0	3500.61230

700 rows × 7 columns

Data Visualization

In [44]:
```
sns.lmplot(x='bmi',y='charges',order=2,data=df,ci=None)
plt.show()
```

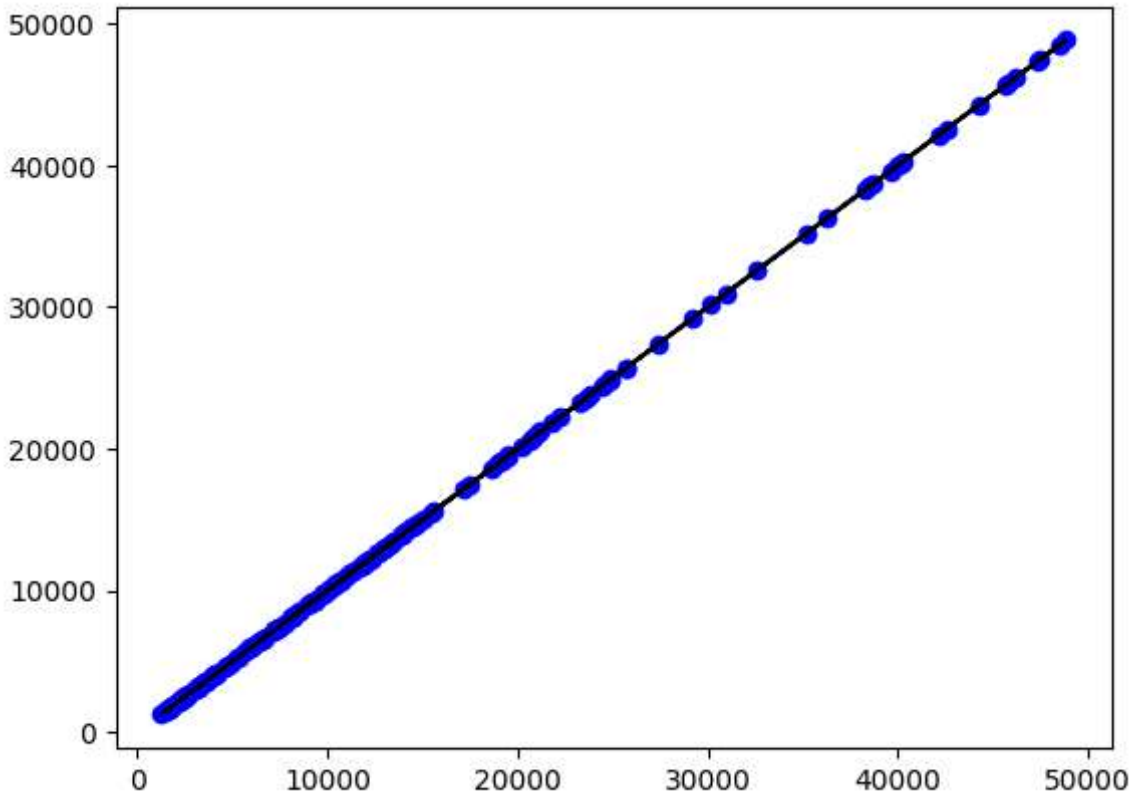


In [45]:
```
x=np.array(df['bmi']).reshape(-1,1)
y=x=np.array(df['charges']).reshape(-1,1)
```

In [46]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=
lr=LinearRegression()
lr.fit(x_train,y_train)
print(lr.score(x_test,y_test))
```

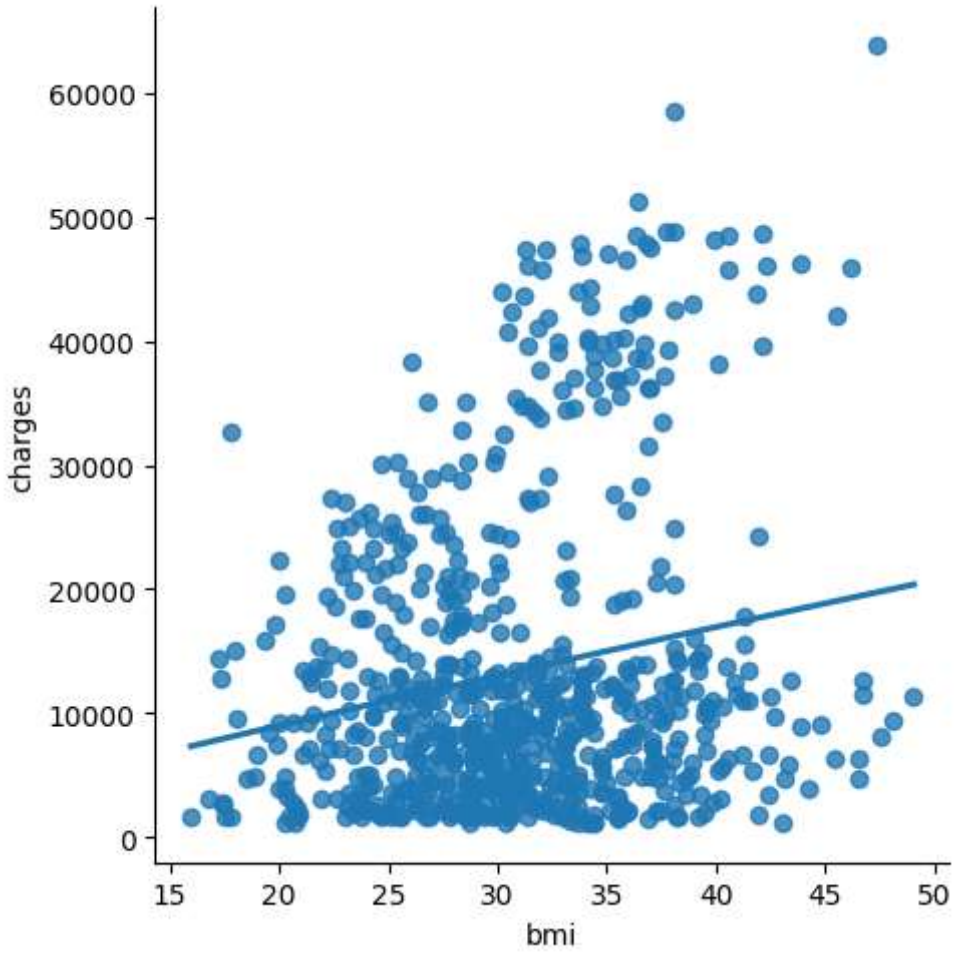
```
1.0
```

In [47]:
```
y_pred=lr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



Work With Subset Data

In [48]:
```
df700=df[:][:700]
sns.lmplot(x='bmi',y='charges',order=2,ci=None,data=df700)
plt.show()
```



In [49]:
```
df700.fillna(method='ffill',inplace=True)
```

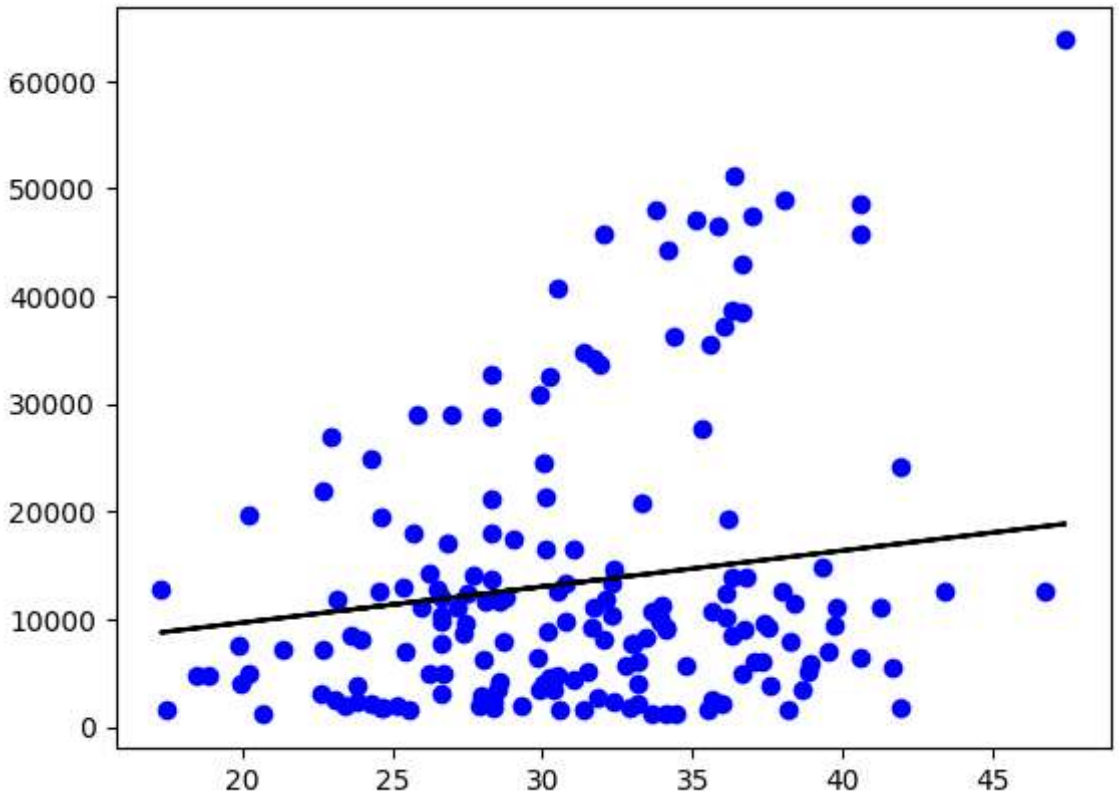
In [50]:
```
x=np.array(df700["bmi"]).reshape(-1,1)
y=np.array(df700['charges']).reshape(-1,1)
```

In [51]:
```
df700.dropna(inplace=True)
```

In [52]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
lr=LinearRegression()
lr.fit(x_train,y_train)
print(lr.score(x_test,y_test))
```

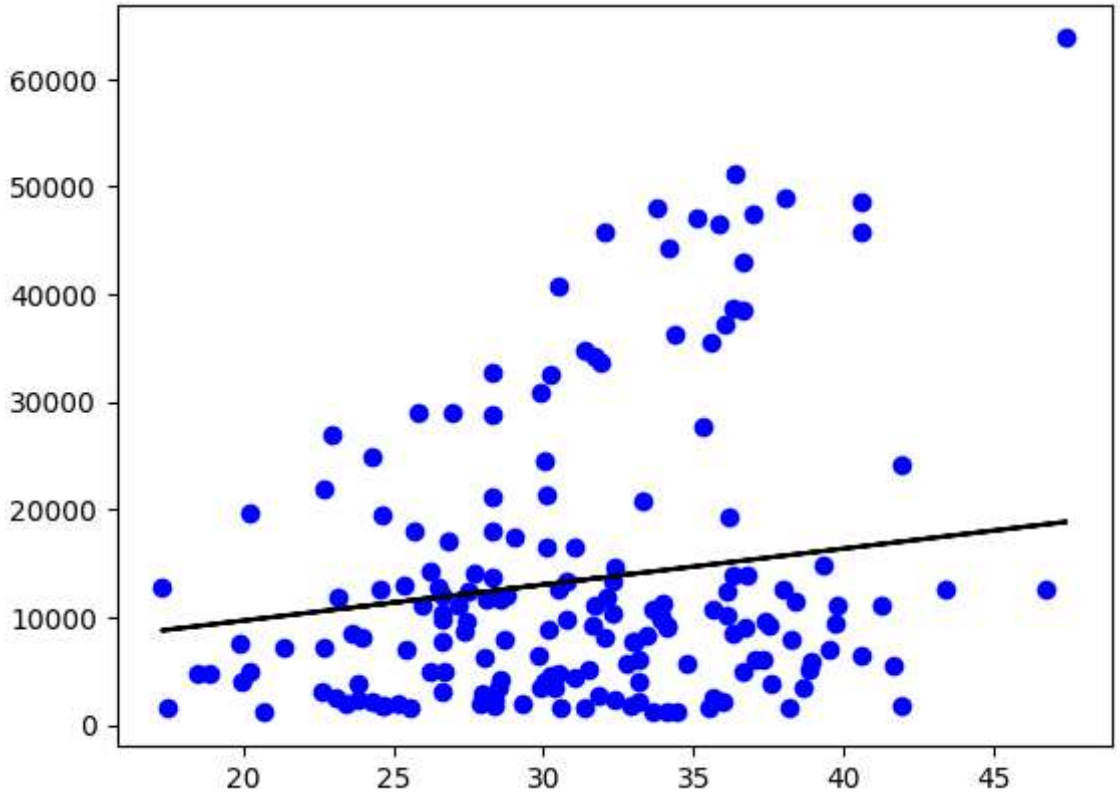
```
0.05538472120602367
```

In [53]:
```
y_pred=lr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



Evaluation Of Model

In [54]:
```
y_pred=lr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```



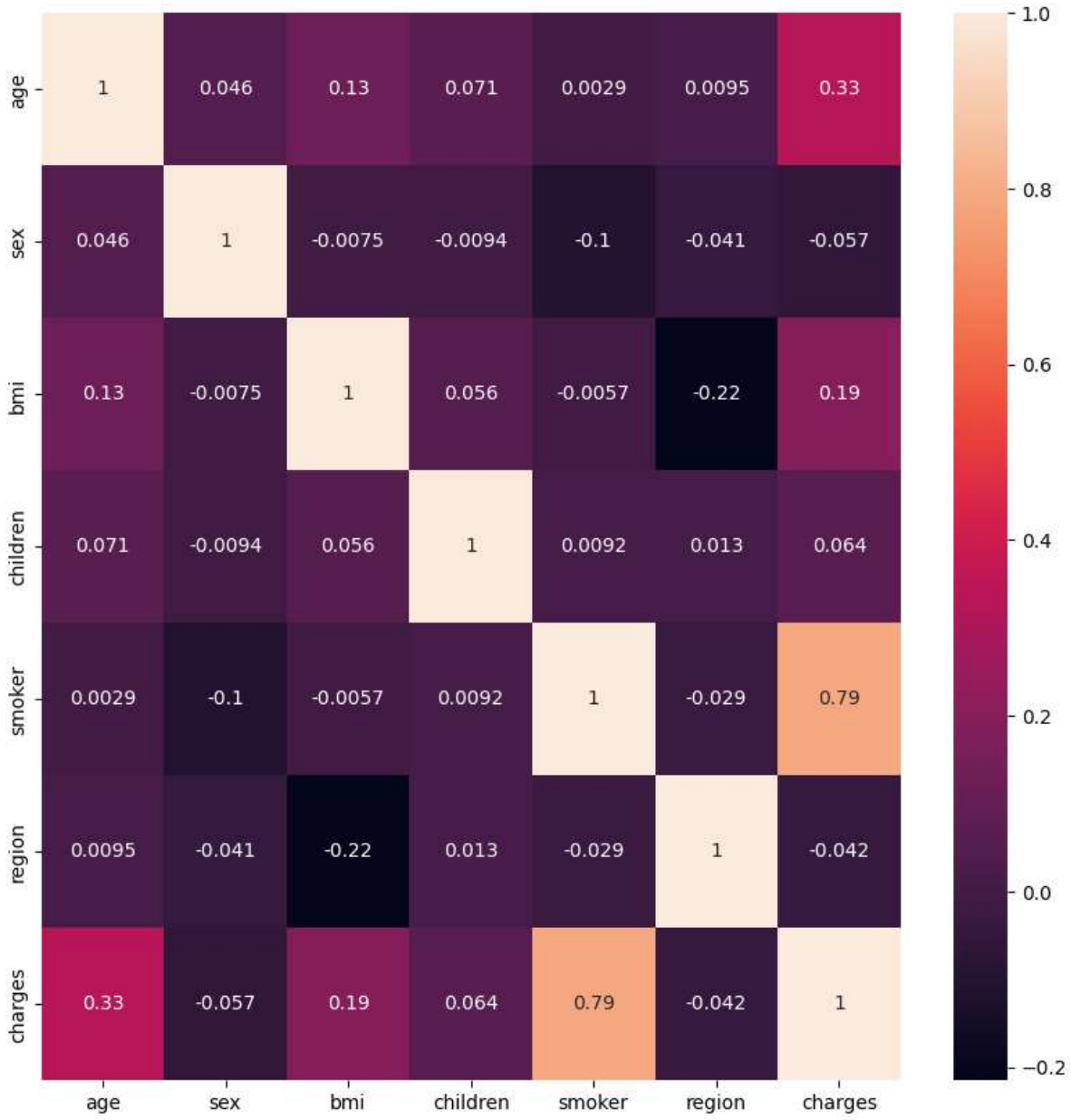
In [55]:
```
lr=LinearRegression()
lr.fit(x_train,y_train)
y_pred=lr.predict(x_test)
r2=r2_score(y_test,y_pred)
print(r2)
```

```
0.05538472120602367
```

Ridge Regression

In [56]:
```
from sklearn.linear_model import Lasso
from sklearn.linear_model import Ridge
from sklearn.preprocessing import StandardScaler
```

In [57]:
```
plt.figure(figsize = (10, 10))
sns.heatmap(df700.corr(), annot = True)
plt.show()
```



In [58]:
```
features=df.columns[0:1]
target=df.columns[-1]
```

In [ ]:
```
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=
print("The dimension of X_train is {}".format(x_train.shape))
print("The dimension of X_test is {}".format(x_test.shape))
```

In [59]:
```
lr = LinearRegression()
#Fit model
lr.fit(x_train, y_train)
#predict
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 0.02860771404610063
The test score for lr model is 0.05538472120602367
```

In [60]:
```
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))
```

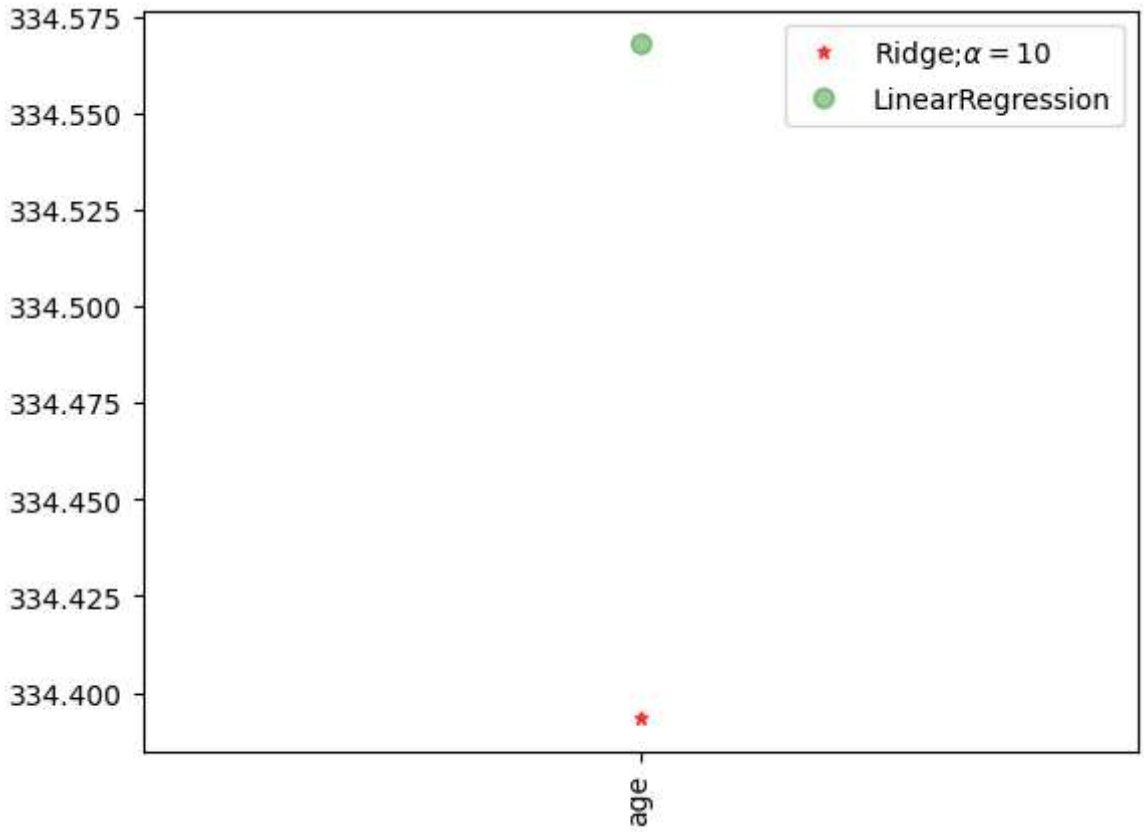
```
Ridge Model:

The train score for ridge model is 0.028607706255655008
The test score for ridge model is 0.05536709747805735
```

In [61]:
```
plt.figure(figsize=(10,10))
```

Out[61]:
```
<Figure size 1000x1000 with 0 Axes>
```

<Figure size 1000x1000 with 0 Axes>

In [62]:
```
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker="*",markersi
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker="o",markersize=7,c
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



Lasso Regression

In [63]:
```
lasso= Lasso(alpha=10)
lasso.fit(x_train,y_train)
#train and test scorefor ridge regression
train_score_ls = lasso.score(x_train, y_train)
test_score_ls= lasso.score(x_test, y_test)
print("\nRidge 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))
```

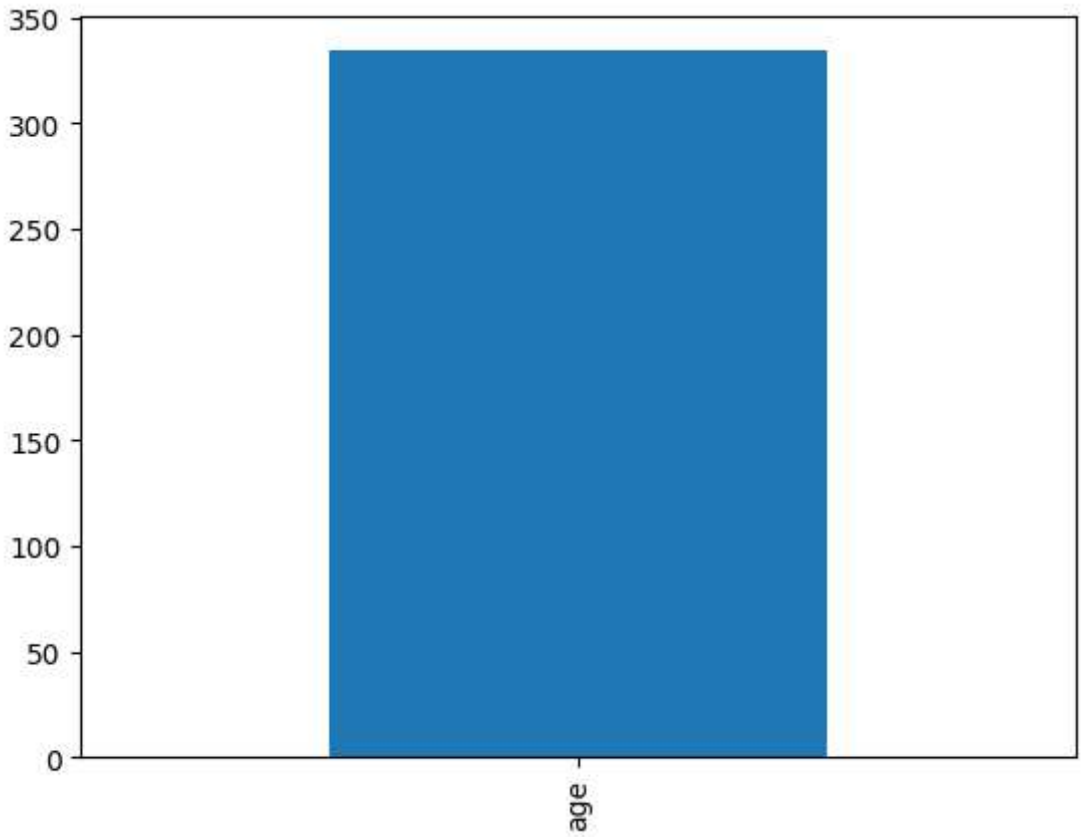
```
Ridge Model:

The train score for lasso model is 0.028607694843243436
The test score for lasso model is 0.05535704638130867
```

In [64]:
```
plt.figure(figsize=(10,10))
```

Out[64]:
```
<Figure size 1000x1000 with 0 Axes>
```

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

In [65]:
```
pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "bar
plt.show()
```



In [66]:
```
from sklearn.linear_model import LassoCV
```

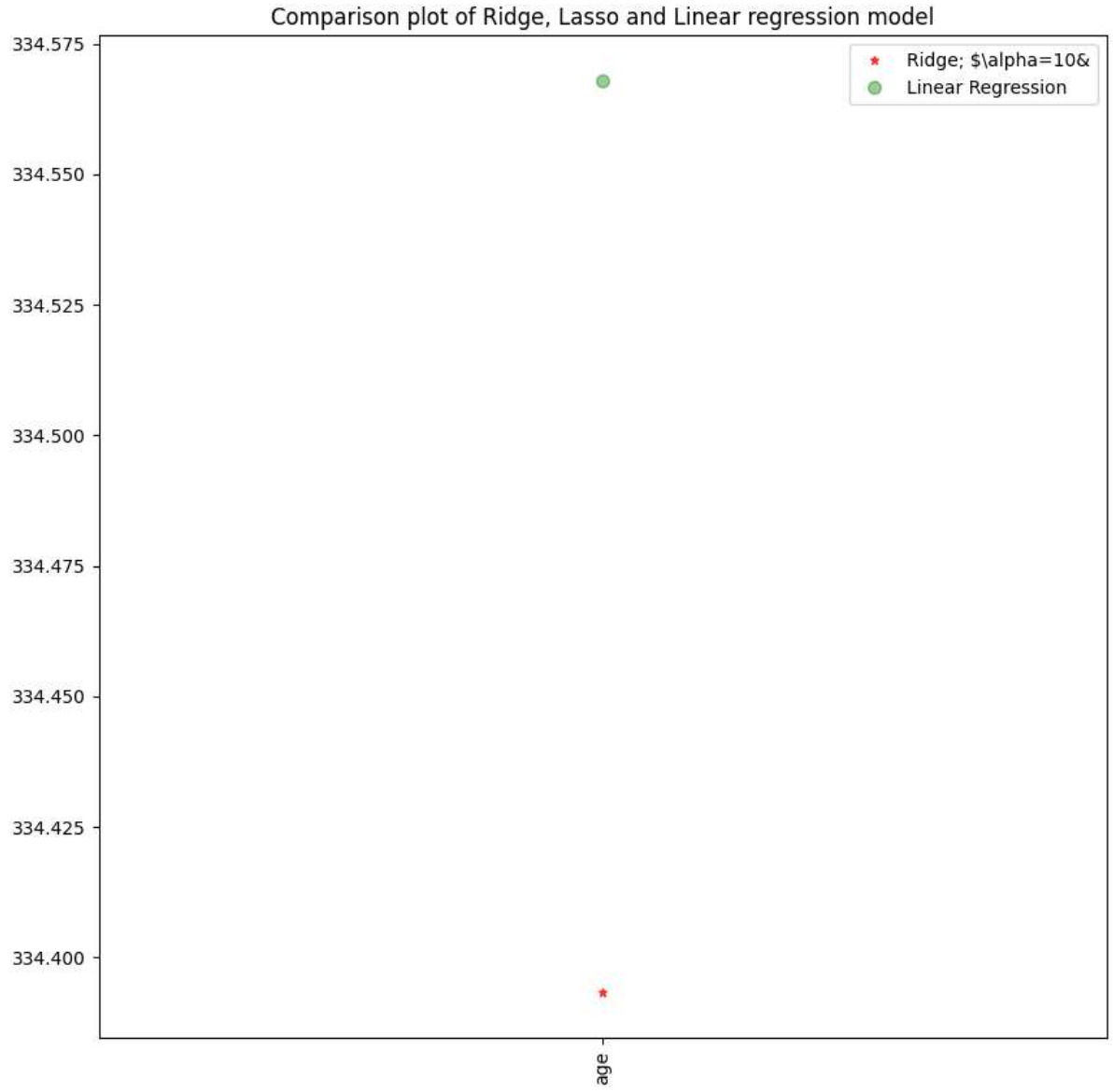
In [67]:
```
#using the linear cv model
from sklearn.linear_model import RidgeCV
#cross validation
ridge_cv=RidgeCV(alphas =[0.0001,0.001,0.01,0.1,1,10]).fit(x_train,y_train)
#score
print(ridge_cv.score(x_train,y_train))
print(ridge_cv.score(x_test,y_test))
```

```
0.028607706255655008
0.05536709747809998
```

In [68]:
```
#using the linear cv model
from sklearn.linear_model import LassoCV
#cross validation
lasso_cv=LassoCV(alphas =[0.0001,0.001,0.01,0.1,1,10]).fit(x_train,y_train)
#score
print(lasso_cv.score(x_train,y_train))
print(lasso_cv.score(x_test,y_test))
```

```
0.02860771404610063
0.05538472092942526

C:\Users\Welcome\AppData\Local\Programs\Python\Python310\lib\site-packages\sk
learn\linear_model\_coordinate_descent.py:1568: DataConversionWarning: A colu
mn-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)
```

In [73]:
```
plt.figure(figsize = (10, 10))
#add plot for ridge regression
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersi
#add plot for lasso regression
#plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,col
#add plot for linear model
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,c
#rotate axis
plt.xticks(rotation = 90)
plt.legend()
plt.title("Comparison plot of Ridge, Lasso and Linear regression model")
plt.show()
```



Elastic Net Regression

In [74]:
```
from sklearn.linear_model import ElasticNet
```

In [75]:
```
el=ElasticNet()
el.fit(x_train,y_train)
print(el.coef_)
print(el.intercept_)
```

```
[330.03091163]
[3122.76162858]
```

In [77]:
```
y_pred_elastic=el.predict(x_train)
```

In [78]:
```
mean_squared_error=np.mean((y_pred_elastic-y_train)**2)
print(mean_squared_error)
```

```
146718205.19247013
```

In [79]:
```
el=ElasticNet()
el.fit(x_train,y_train)
print(el.score(x_train,y_train))
```

```
0.028602453353040902
```

Logistic Regression

In [80]:
```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

In [81]:
```
df=pd.read_csv(r"C:\Users\Welcome\Downloads\insurance.csv")
df
```

Out[81]:

	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 [82]:
```
df.shape
```

Out[82]:
```
(1338, 7)
```

In [83]:
```
pd.set_option('display.max_rows',10000000000)
pd.set_option('display.max_columns',10000000000)
pd.set_option('display.width',95)
```

In [84]:
```
print('This Dataset has %d rows and %d columns'%(df.shape))
```

```
This Dataset has 1338 rows and 7 columns
```

In [85]:
```
df.head()
```

Out[85]:

	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 [86]:
```
df.describe
```

Out[86]:
```
<bound method NDFrame.describe of       age     sex     bmi  children smoke
r     region       charges
0       19  female  27.900         0    yes  southwest  16884.924000
1       18    male  33.770         1     no  southeast   1725.552300
2       28    male  33.000         3     no  southeast   4449.462000
3       33    male  22.705         0     no  northwest  21984.470610
4       32    male  28.880         0     no  northwest   3866.855200
5       31  female  25.740         0     no  southeast   3756.621600
6       46  female  33.440         1     no  southeast   8240.589600
7       37  female  27.740         3     no  northwest   7281.505600
8       37    male  29.830         2     no  northeast   6406.410700
9       60  female  25.840         0     no  northwest  28923.136920
10      25    male  26.220         0     no  northeast   2721.320800
11      62  female  26.290         0    yes  southeast  27808.725100
12      23    male  34.400         0     no  southwest   1826.843000
13      56  female  39.820         0     no  southeast  11090.717800
14      27    male  42.130         0    yes  southeast  39611.757700
15      19    male  24.600         1     no  southwest   1837.237000
16      52  female  30.780         1     no  northeast  10797.336200
```

In [87]:
```
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 [88]:
```
df.isnull().sum()
```

Out[88]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

In [89]:
```
convert={"smoker":{"yes":1,"no":0}}
df=df.replace(convert)
df
```

Out[89]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	1	southwest	16884.924000
1	18	male	33.770	1	0	southeast	1725.552300
2	28	male	33.000	3	0	southeast	4449.462000
3	33	male	22.705	0	0	northwest	21984.470610
4	32	male	28.880	0	0	northwest	3866.855200
5	31	female	25.740	0	0	southeast	3756.621600
6	46	female	33.440	1	0	southeast	8240.589600
7	37	female	27.740	3	0	northwest	7281.505600
8	37	male	29.830	2	0	northeast	6406.410700
9	60	female	25.840	0	0	northwest	28923.136920
10	25	male	26.220	0	0	northeast	2721.320800
11	62	female	26.290	0	1	southeast	27808.725100

In [90]:
```
convert={"sex":{"female":1,"male":0}}
df=df.replace(convert)
df
```

Out[90]:

	age	sex	bmi	children	smoker	region	charges
0	19	1	27.900	0	1	southwest	16884.924000
1	18	0	33.770	1	0	southeast	1725.552300
2	28	0	33.000	3	0	southeast	4449.462000
3	33	0	22.705	0	0	northwest	21984.470610
4	32	0	28.880	0	0	northwest	3866.855200
5	31	1	25.740	0	0	southeast	3756.621600
6	46	1	33.440	1	0	southeast	8240.589600
7	37	1	27.740	3	0	northwest	7281.505600
8	37	0	29.830	2	0	northeast	6406.410700
9	60	1	25.840	0	0	northwest	28923.136920
10	25	0	26.220	0	0	northeast	2721.320800
11	62	1	26.290	0	1	southeast	27808.725100

In [91]:
```
convert={"region":{"southeast":1,"southwest":2,"northeast":3,"northwest":4}}
df=df.replace(convert)
df
```

Out[91]:

	age	sex	bmi	children	smoker	region	charges
0	19	1	27.900	0	1	2	16884.924000
1	18	0	33.770	1	0	1	1725.552300
2	28	0	33.000	3	0	1	4449.462000
3	33	0	22.705	0	0	4	21984.470610
4	32	0	28.880	0	0	4	3866.855200
5	31	1	25.740	0	0	1	3756.621600
6	46	1	33.440	1	0	1	8240.589600
7	37	1	27.740	3	0	4	7281.505600
8	37	0	29.830	2	0	3	6406.410700
9	60	1	25.840	0	0	4	28923.136920
10	25	0	26.220	0	0	3	2721.320800
11	62	1	26.290	0	1	1	27808.725100

In [92]:
```
features_matrix=df.iloc[:,0:4]
```

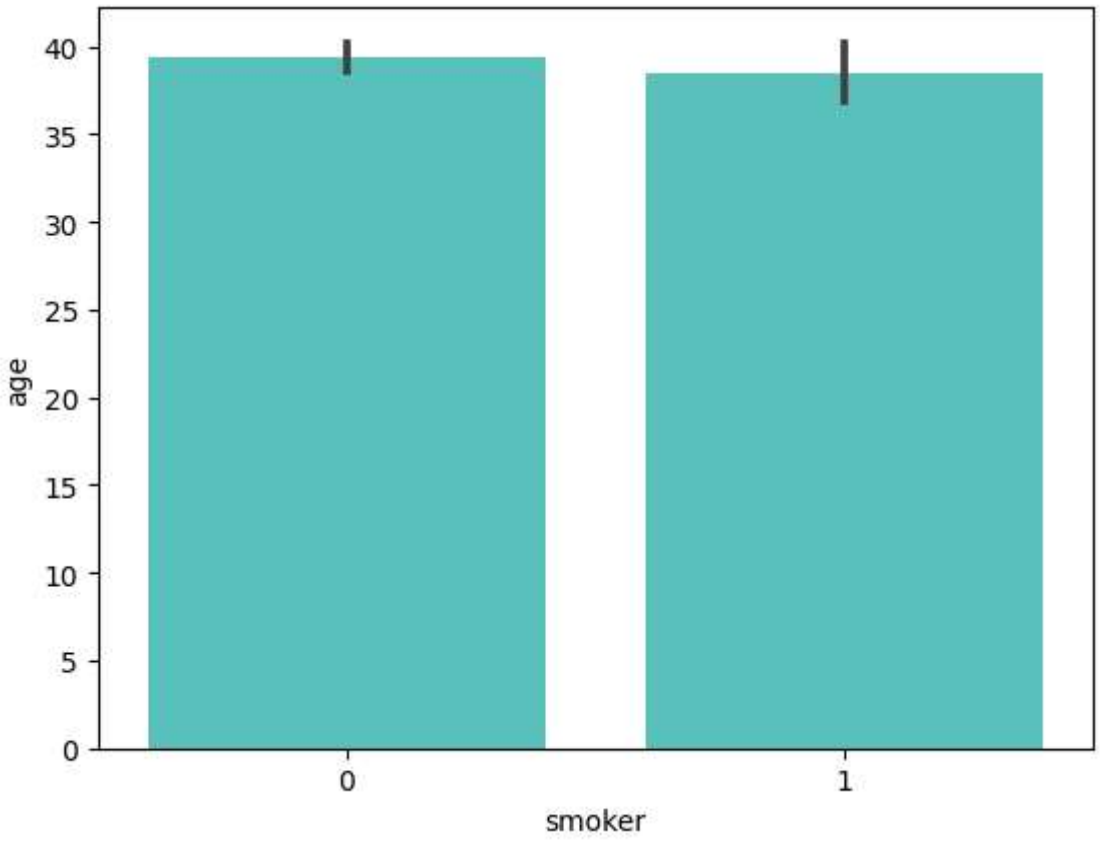
In [93]:
```
target_vector=df.iloc[:,-3]
```

In [94]:
```
print('The Feature Matrix has %d Rows and %d columns(s)'%(features_matrix.shape
print('The Target Matrix has %d Rows and %d columns(s)'%(np.array(target_vector
```

```
The Feature Matrix has 1338 Rows and 4 columns(s)
The Target Matrix has 1338 Rows and 1 columns(s)
```

In [95]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [96]:
```
sns.barplot(x='smoker', y='age', data=df, color="mediumturquoise")
plt.show()
```



In [97]:
```
features_matrix_standardized=StandardScaler().fit_transform(features_matrix)
```

In [98]:
```
algorithm=LogisticRegression(max_iter=10000)
```

In [99]:
```
Logistic_Regression_Model=algorithm.fit(features_matrix_standardized,target_vec
```

In [100]:
```
observation=[[1,0,0.99539,-0.0588]]
```

In [101]:
```
predictions=Logistic_Regression_Model.predict(observation)
print('The model predicted the observation to belong to class %s'%(predictions)
```

```
The model predicted the observation to belong to class [0]
```

In [102]:
```
print('The algoritham was trained to predict one of the two classes:%s'%(algori
```

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

In [104]:
```
print(" " "The Model says the probability of the observation we passed belongir
print()
print(" " "The Model says the probability of the observation we passed belongir
```

```
 The Model says the probability of the observation we passed belonging to cla
ss[0] 0.8057075871331396

 The Model says the probability of the observation we passed belonging to cla
ss[1] 0.19429241286686041
```

In [105]:
```
x=np.array(df['age']).reshape(-1,1)
y=np.array(df['smoker']).reshape(-1,1)
```

In [106]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.05)
lo=LogisticRegression()
lo.fit(x_train,y_train)
print(lo.score(x_test,y_test))
```

```
0.8059701492537313

C:\Users\Welcome\AppData\Local\Programs\Python\Python310\lib\site-packages\sk
learn\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_sampl
es, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Decision Tree

In [107]:
```
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

In [108]:
```
df=pd.read_csv(r"C:\Users\Welcome\Downloads\insurance.csv")
df
```

Out[108]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.924000
1	18	male	33.770	1	no	southeast	1725.552300
2	28	male	33.000	3	no	southeast	4449.462000
3	33	male	22.705	0	no	northwest	21984.470610
4	32	male	28.880	0	no	northwest	3866.855200
5	31	female	25.740	0	no	southeast	3756.621600
6	46	female	33.440	1	no	southeast	8240.589600
7	37	female	27.740	3	no	northwest	7281.505600
8	37	male	29.830	2	no	northeast	6406.410700
9	60	female	25.840	0	no	northwest	28923.136920
10	25	male	26.220	0	no	northeast	2721.320800
11	62	female	26.290	0	yes	southeast	27808.725100

In [109]:
```
df.shape
```

Out[109]:
```
(1338, 7)
```

In [110]:
```
df.isnull().any()
```

Out[110]:
```
age         False
sex         False
bmi         False
children    False
smoker      False
region      False
charges     False
dtype: bool
```

In [111]:
```
df['region'].value_counts()
```

Out[111]:
```
region
southeast    364
southwest    325
northwest    325
northeast    324
Name: count, dtype: int64
```

In [112]:
```
convert={"sex":{"female":1,"male":0}}
df=df.replace(convert)
df
```

Out[112]:

	age	sex	bmi	children	smoker	region	charges
0	19	1	27.900	0	yes	southwest	16884.924000
1	18	0	33.770	1	no	southeast	1725.552300
2	28	0	33.000	3	no	southeast	4449.462000
3	33	0	22.705	0	no	northwest	21984.470610
4	32	0	28.880	0	no	northwest	3866.855200
5	31	1	25.740	0	no	southeast	3756.621600
6	46	1	33.440	1	no	southeast	8240.589600
7	37	1	27.740	3	no	northwest	7281.505600
8	37	0	29.830	2	no	northeast	6406.410700
9	60	1	25.840	0	no	northwest	28923.136920
10	25	0	26.220	0	no	northeast	2721.320800
11	62	1	26.290	0	yes	southeast	27808.725100

In [113]:
```
convert={"smoker":{"yes":1,"no":0}}
df=df.replace(convert)
df
```

Out[113]:

	age	sex	bmi	children	smoker	region	charges
0	19	1	27.900	0	1	southwest	16884.924000
1	18	0	33.770	1	0	southeast	1725.552300
2	28	0	33.000	3	0	southeast	4449.462000
3	33	0	22.705	0	0	northwest	21984.470610
4	32	0	28.880	0	0	northwest	3866.855200
5	31	1	25.740	0	0	southeast	3756.621600
6	46	1	33.440	1	0	southeast	8240.589600
7	37	1	27.740	3	0	northwest	7281.505600
8	37	0	29.830	2	0	northeast	6406.410700
9	60	1	25.840	0	0	northwest	28923.136920
10	25	0	26.220	0	0	northeast	2721.320800
11	62	1	26.290	0	1	southeast	27808.725100

In [114]:
```
x=["bmi","children"]
y=["Yes","No"]
all_inputs=df[x]
all_classes=df["sex"]
```

In [115]:
```
(x_train,x_test,y_train,y_test)=train_test_split(all_inputs,all_classes,test_si
```

In [116]:
```
clf=DecisionTreeClassifier(random_state=0)
```

In [118]:
```
clf.fit(x_train,y_train)
```

Out[118]:
```
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 [119]:
```
score=clf.score(x_test,y_test)
print(score)
```

```
0.3902439024390244
```

Random Forest

In [120]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt ,seaborn as sns
```

In [121]:
```
df=pd.read_csv(r"C:\Users\Welcome\Downloads\insurance.csv")
df
```

Out[121]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.924000
1	18	male	33.770	1	no	southeast	1725.552300
2	28	male	33.000	3	no	southeast	4449.462000
3	33	male	22.705	0	no	northwest	21984.470610
4	32	male	28.880	0	no	northwest	3866.855200
5	31	female	25.740	0	no	southeast	3756.621600
6	46	female	33.440	1	no	southeast	8240.589600
7	37	female	27.740	3	no	northwest	7281.505600
8	37	male	29.830	2	no	northeast	6406.410700
9	60	female	25.840	0	no	northwest	28923.136920
10	25	male	26.220	0	no	northeast	2721.320800
11	62	female	26.290	0	ves	southeast	27808.725100

In [122]:
```
df.shape
```

Out[122]:
```
(1338, 7)
```

In [123]:
```
df['region'].value_counts()
```

Out[123]:
```
region
southeast    364
southwest    325
northwest    325
northeast    324
Name: count, dtype: int64
```

In [124]:
```
df['bmi'].value_counts()
```

Out[124]:
```
bmi
32.300    13
28.310     9
30.495     8
30.875     8
31.350     8
30.800     8
34.100     8
28.880     8
33.330     7
35.200     7
25.800     7
32.775     7
27.645     7
32.110     7
38.060     7
25.460     7
30.590     7
27.360     7
```

In [125]:
```
m={"sex":{"female":1,"male":0}}
df=df.replace(m)
print(df)
```

```
      age  sex     bmi  children smoker     region       charges
0      19    1  27.900         0    yes  southwest  16884.924000
1      18    0  33.770         1     no  southeast   1725.552300
2      28    0  33.000         3     no  southeast   4449.462000
3      33    0  22.705         0     no  northwest  21984.470610
4      32    0  28.880         0     no  northwest   3866.855200
5      31    1  25.740         0     no  southeast   3756.621600
6      46    1  33.440         1     no  southeast   8240.589600
7      37    1  27.740         3     no  northwest   7281.505600
8      37    0  29.830         2     no  northeast   6406.410700
9      60    1  25.840         0     no  northwest  28923.136920
10     25    0  26.220         0     no  northeast   2721.320800
11     62    1  26.290         0    yes  southeast  27808.725100
12     23    0  34.400         0     no  southwest   1826.843000
13     56    1  39.820         0     no  southeast  11090.717800
14     27    0  42.130         0    yes  southeast  39611.757700
15     19    0  24.600         1     no  southwest   1837.237000
16     52    1  30.780         1     no  northeast  10797.336200
17     23    0  23.845         0     no  northeast   2395.171550
```

In [126]:
```
n={"smoker":{"yes":1,"no":0}}
df=df.replace(n)
print(df)
```

```
      age  sex     bmi  children  smoker     region       charges
0      19    1  27.900         0       1  southwest  16884.924000
1      18    0  33.770         1       0  southeast   1725.552300
2      28    0  33.000         3       0  southeast   4449.462000
3      33    0  22.705         0       0  northwest  21984.470610
4      32    0  28.880         0       0  northwest   3866.855200
5      31    1  25.740         0       0  southeast   3756.621600
6      46    1  33.440         1       0  southeast   8240.589600
7      37    1  27.740         3       0  northwest   7281.505600
8      37    0  29.830         2       0  northeast   6406.410700
9      60    1  25.840         0       0  northwest  28923.136920
10     25    0  26.220         0       0  northeast   2721.320800
11     62    1  26.290         0       1  southeast  27808.725100
12     23    0  34.400         0       0  southwest   1826.843000
13     56    1  39.820         0       0  southeast  11090.717800
14     27    0  42.130         0       1  southeast  39611.757700
15     19    0  24.600         1       0  southwest   1837.237000
16     52    1  30.780         1       0  northeast  10797.336200
17     23    0  23.845         0       0  northeast   2395.171550
```

In [127]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[127]:
```
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 [128]:
```
rf=RandomForestClassifier()
params={'max_depth':[2,3,5,20],
 'min_samples_leaf':[5,10,20,50,100,200],
 'n_estimators':[10,25,30,50,100,200]}
```

In [129]:
```
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[129]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [2, 3, 5, 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.

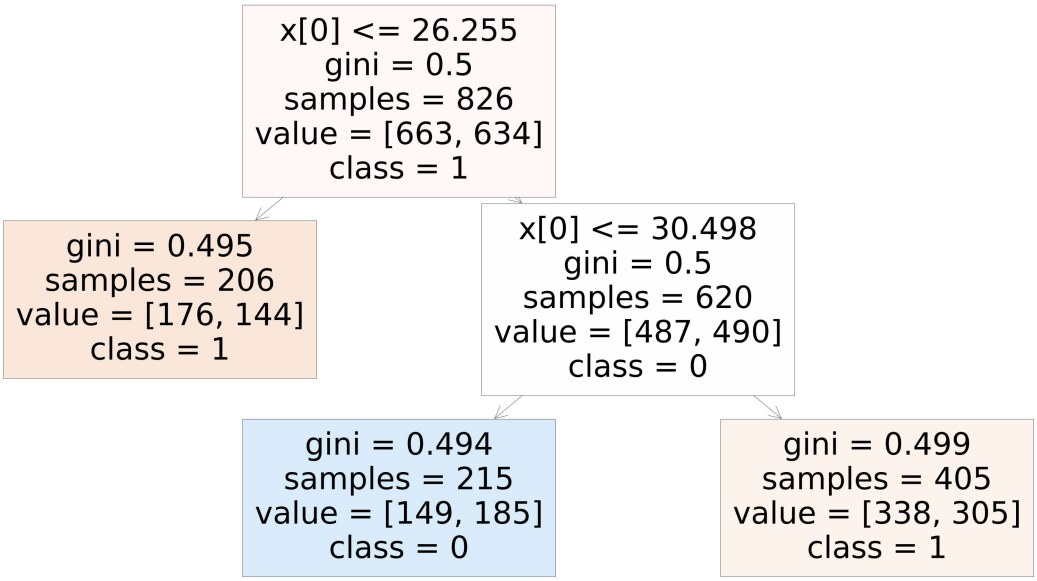
In [130]:
```
grid_search.best_score_
```

Out[130]:
```
0.5227462953451654
```

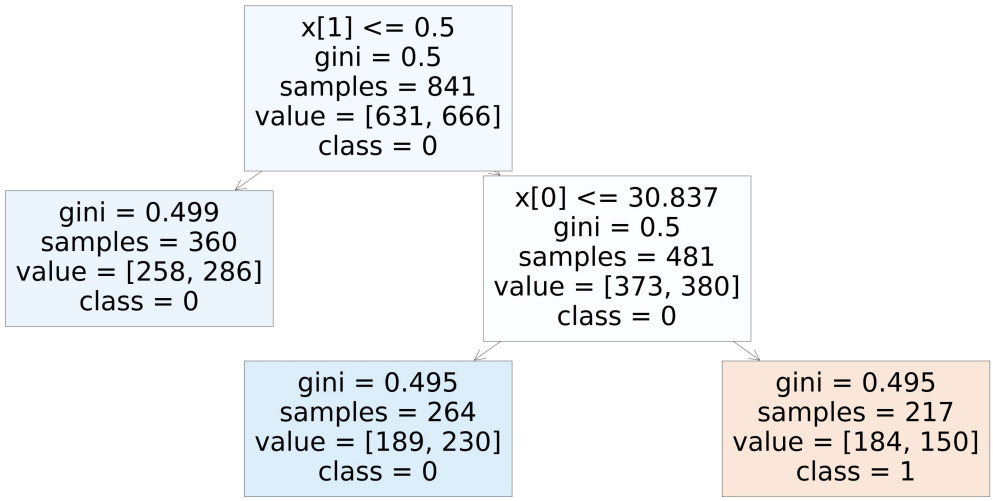
In [131]:
```
rf_best=grid_search.best_estimator_
print(rf_best)
```

```
RandomForestClassifier(max_depth=2, min_samples_leaf=200, n_estimators=25)
```

In [132]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[4],class_names=['1','0'],filled=True);
```



In [133]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(70,30))
plot_tree(rf_best.estimators_[6],class_names=["1","0"],filled=True);
```



In [134]:
```
rf_best.feature_importances_
```

Out[134]:
```
array([0.62283872, 0.37716128])
```

In [135]:
```
rf=RandomForestClassifier(random_state=0)
```

In [136]:
```
rf.fit(x_train,y_train)
```

Out[136]:
```
RandomForestClassifier(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 [137]:
```
score=rf.score(x_test,y_test)
print(score)
```

```
0.2926829268292683
```

Conclusion: For the given insurance data set have performed linear,logistic,random forest and decision tree models of regression and classifications.

#and have conclude that the most accurancy is occured in logistic regression,i.e 80percent #when compare to other regression models. #and concluded that "Logistic Regression" model is fits for the data.

In [ ]:
```

```