PROBLEM SATATEMENT:

In this we have to caluclate the how many members of "male smokers"as well as how many memebers of "female smokers"present in the given data set.

Data collection:

In [2]:

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

In [3]:

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

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

Here I import the data using pandas(library) by giving the respected file path.

Data cleaning:

In [4]:

```
data.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 this 'head' command will gives the few rows from the data

In [5]:

```
data.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

'tail' command will gives the few rows from the bottom of data

In [6]:

```
data.describe()
```

Out[6]:

	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

By using 'describe' command i can get the satistical values(such as mean,standerd deviation etc..)

In [7]:

```
data.shape
```

Out[7]:

```
(1338, 7)
```

by using the 'shape' command i can get the no.of rows and no.of columns in the given dataset

Data preprocessing:

In [8]:

```
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
```

by applying 'info' command i can cheack wether the data has any null values or not ,with

In [9]:

```
data['smoker'].value_counts()
```

Out[9]:

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

Now,by using 'value_counts' method i can count the how many members are smoking and how many are not smoking

In [10]:

```
data['sex'].value_counts()
```

Out[10]:

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

here ,i can cheack males and females count

In [11]:

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

Out[11]:

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

now, i can found the how many regions are presented in the given dataset

In [12]:

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

Out[12]:

	age	sex	bmi	children	smoker	region	charges
0	19	0	27.900	0	yes	southwest	16884.92400
1	18	1	33.770	1	no	southeast	1725.55230
2	28	1	33.000	3	no	southeast	4449.46200
3	33	1	22.705	0	no	northwest	21984.47061
4	32	1	28.880	0	no	northwest	3866.85520
...	...	...	...	...	...	...	...
1333	50	1	30.970	3	no	northwest	10600.54830
1334	18	0	31.920	0	no	northeast	2205.98080
1335	18	0	36.850	0	no	southeast	1629.83350
1336	21	0	25.800	0	no	southwest	2007.94500
1337	61	0	29.070	0	yes	northwest	29141.36030

1338 rows × 7 columns

here iam converting the feature of'sex'labels into 0,1 wich is part of decision tree making process

In [13]:

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

Out[13]:

	age	sex	bmi	children	smoker	region	charges
0	19	0	27.900	0	yes	1	16884.92400
1	18	1	33.770	1	no	2	1725.55230
2	28	1	33.000	3	no	2	4449.46200
3	33	1	22.705	0	no	3	21984.47061
4	32	1	28.880	0	no	3	3866.85520
...	...	...	...	...	...	...	...
1333	50	1	30.970	3	no	3	10600.54830
1334	18	0	31.920	0	no	4	2205.98080
1335	18	0	36.850	0	no	2	1629.83350
1336	21	0	25.800	0	no	1	2007.94500
1337	61	0	29.070	0	yes	3	29141.36030

1338 rows × 7 columns

In [14]:

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

Out[14]:

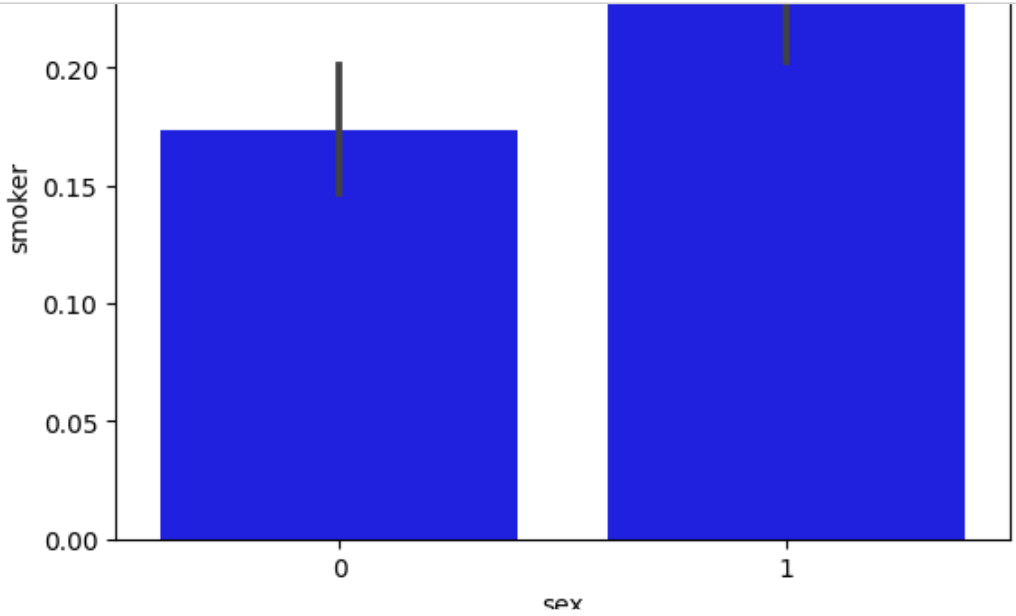
	age	sex	bmi	children	smoker	region	charges
0	19	0	27.900	0	1	1	16884.92400
1	18	1	33.770	1	0	2	1725.55230
2	28	1	33.000	3	0	2	4449.46200
3	33	1	22.705	0	0	3	21984.47061
4	32	1	28.880	0	0	3	3866.85520
...	...	...	...	...	...	...	...
1333	50	1	30.970	3	0	3	10600.54830
1334	18	0	31.920	0	0	4	2205.98080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

1338 rows × 7 columns

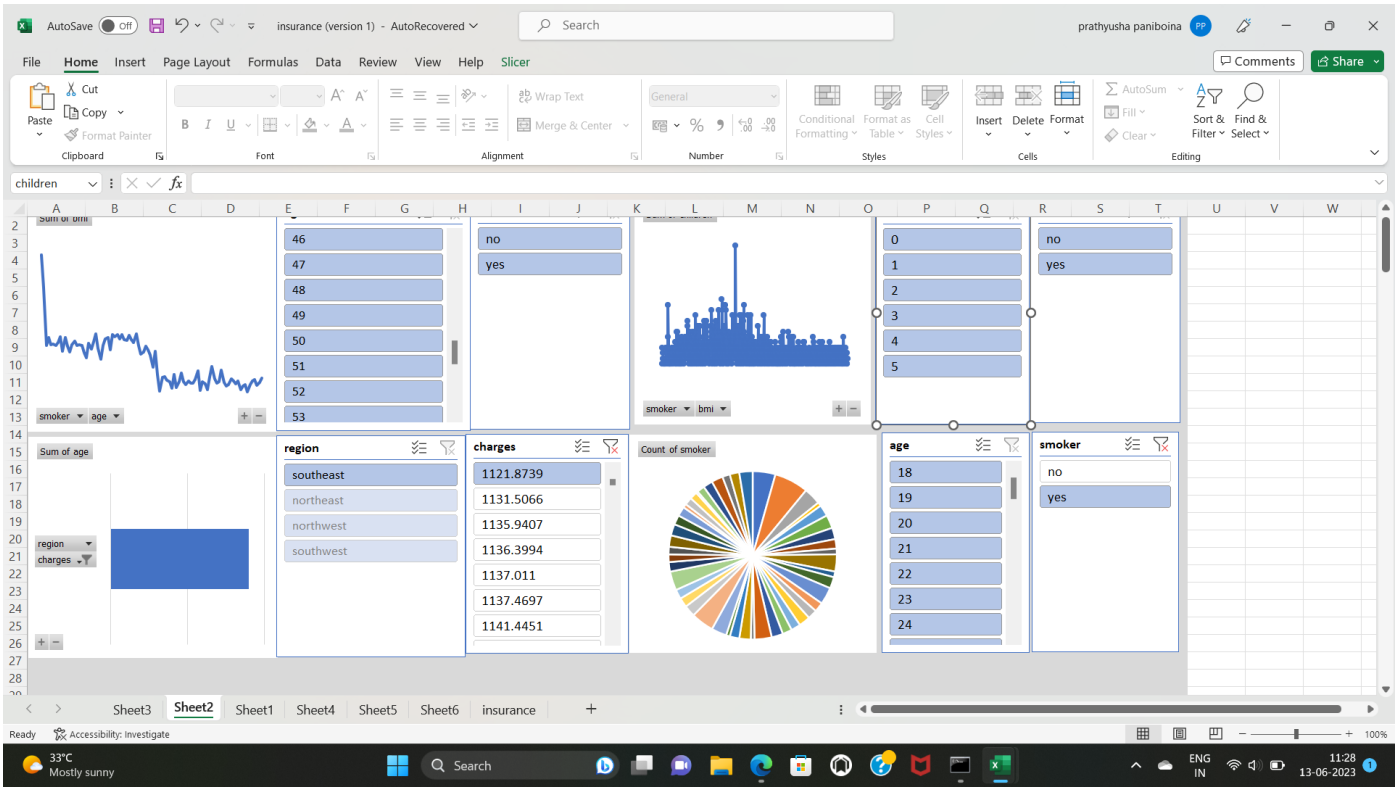
Data visualization:

In [15]:

```
sns.barplot(x='sex',y='smoker',data=data,color='b')
plt.show()
```



Here,i can used bar plot to plot the smokers data with respect to the 'sex'



Data modeling for Decision tree:

In [16]:

```
x=["sex"]
y=["Yes","No"]
all_inputs=data[x]
all_clasess=data["smoker"]
```

In [17]:

```
(x_train,x_test,y_train,y_test)=train_test_split(all_inputs,all_clasess,test_size=0.03)
```

In [18]:

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

In [19]:

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

Out[19]:

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

In [20]:

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

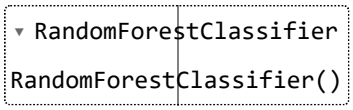
```
0.7317073170731707
```

Randomforest:

In [21]:

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

Out[21]:



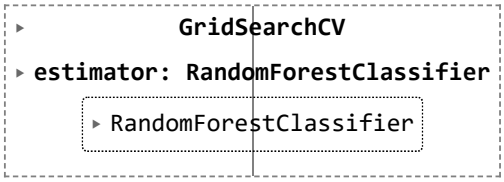
In [22]:

```
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 [23]:

```
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[23]:



In [24]:

```
grid_search.best_score_
```

Out[24]:

```
0.7972248378321825
```

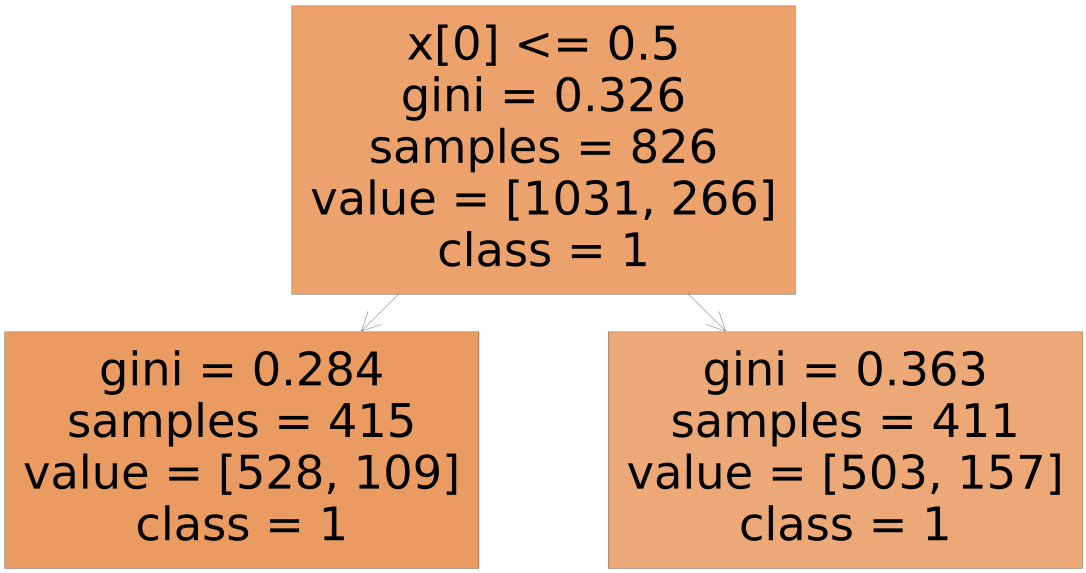
In [25]:

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

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

In [26]:

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



In [27]:

```
rf_best.feature_importances_
```

Out[27]:

```
array([1.])
```

In [28]:

```
rf=RandomForestClassifier(random_state=0)
```

In [29]:

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

Out[29]:

```
RandomForestClassifier
RandomForestClassifier(random_state=0)
```

In [30]:

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

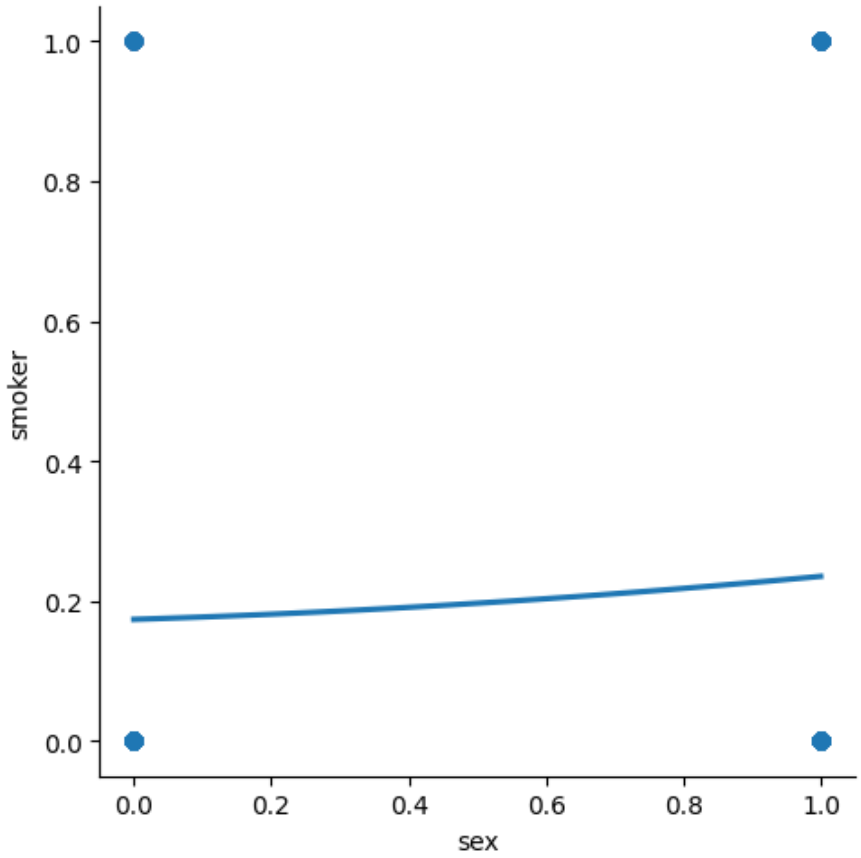
```
0.7317073170731707
```

Linear regression:

In [32]:

```
sns.lmplot(x='sex',y='smoker',order=2,data=data,ci=None)
plt.show()
```

```
C:\Users\Prathyusha\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\r
egression.py:254: RankWarning: Polyfit may be poorly conditioned
  yhat = reg_func(x, y)
```



In [33]:

```
x=np.array(data['sex']).reshape(-1,1)
y=x=np.array(data['smoker']).reshape(-1,1)
```

In [34]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

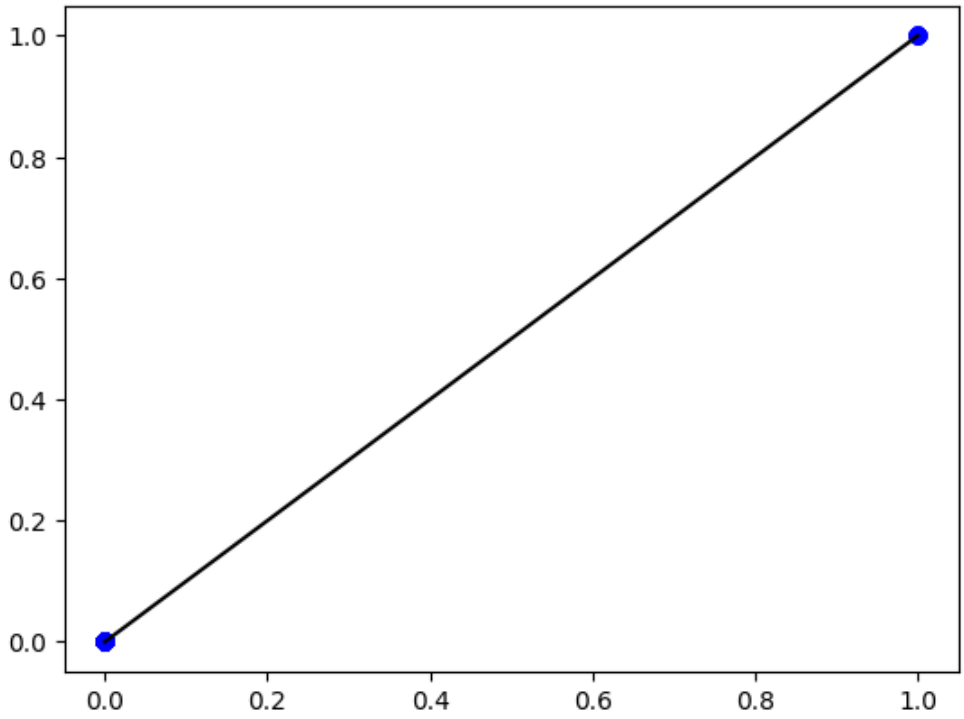
In [35]:

```
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))
```

```
1.0
```

In [36]:

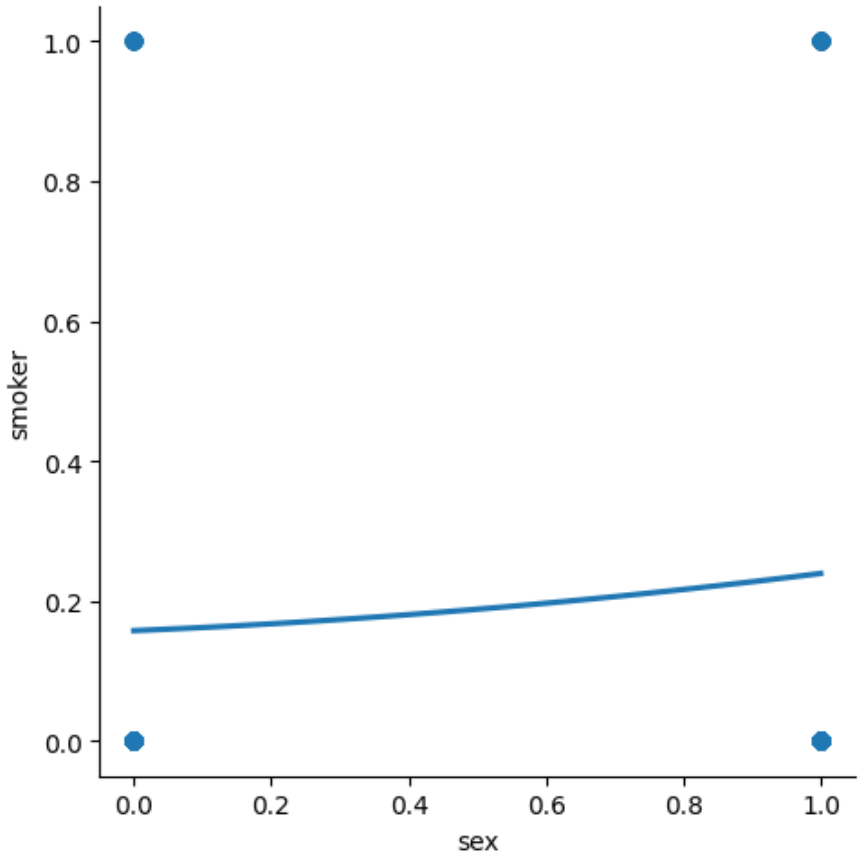
```
y_pred=lr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



In [37]:

```
data700=data[:][:700]
sns.lmplot(x='sex',y='smoker',order=2,ci=None,data=data700)
plt.show()
```

```
C:\Users\Prathyusha\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\r
egression.py:254: RankWarning: Polyfit may be poorly conditioned
  yhat = reg_func(x, y)
```



In [38]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

In [39]:

```
lr=LinearRegression()
lr.fit(x_train,y_train)
y_pred=lr.predict(x_test)
r2=r2_score(y_test,y_pred)
print(r2)
```

```
1.0
```

Ridge regression:

In [41]:

```
from sklearn.linear_model import Lasso,Ridge
from sklearn.preprocessing import StandardScaler
```

In [43]:

```
plt.figure(figsize=(10,10))
sns.heatmap(data700.corr(),annot=True)
plt.show()
```

In [45]:

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

In [46]:

```
x=data[features].values
y=data[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 (936, 1)
The dimension of X_test is (402, 1)
```

In [47]:

```
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 0.0910963973805714
The test score for lr model is 0.08490473916580776
```

In [48]:

```
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(train_score_ridge))
```

```
Ridge Model:

The train score for ridge model is 0.09109639711159634
The test score for ridge model is 0.09109639711159634
```

In [49]:

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

Out[49]:

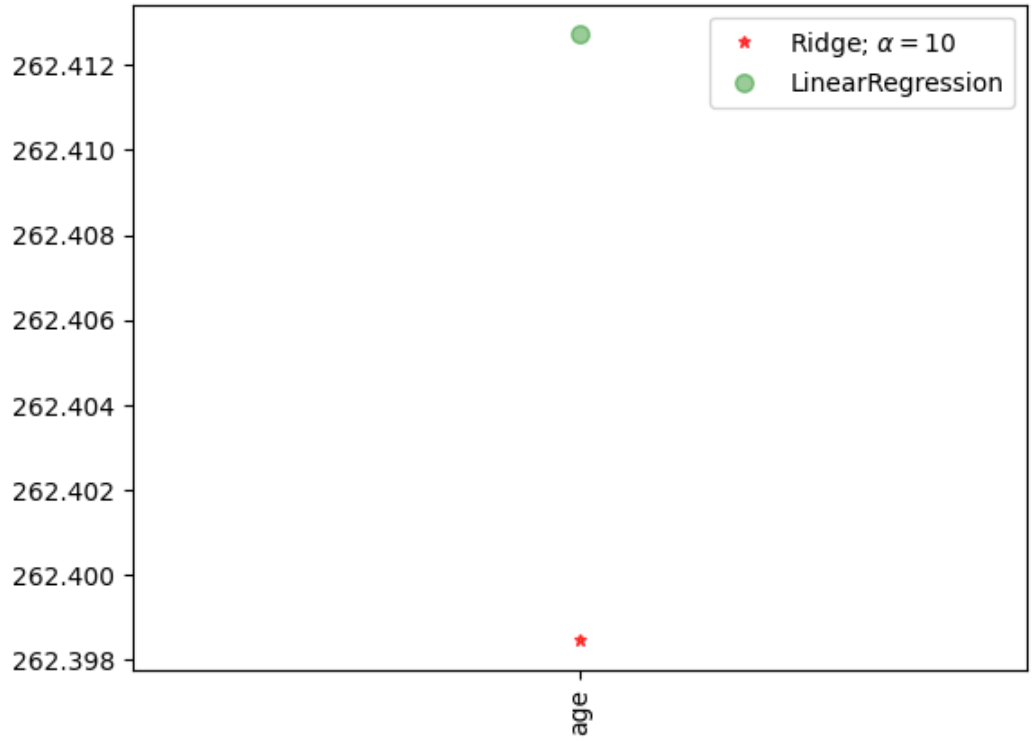
```
<Figure size 1000x1000 with 0 Axes>
```



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

In [50]:

```
.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker="*",markersize=5,color='red',label=r'R
.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker="o",markersize=7,color='green',label='Linear
.xticks(rotation=90)
.legend()
.show()
```



Lasso regression:

In [52]:

```
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 0.09109639395809044
The test score for lasso model is 0.08490704421828055
```

In [53]:

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

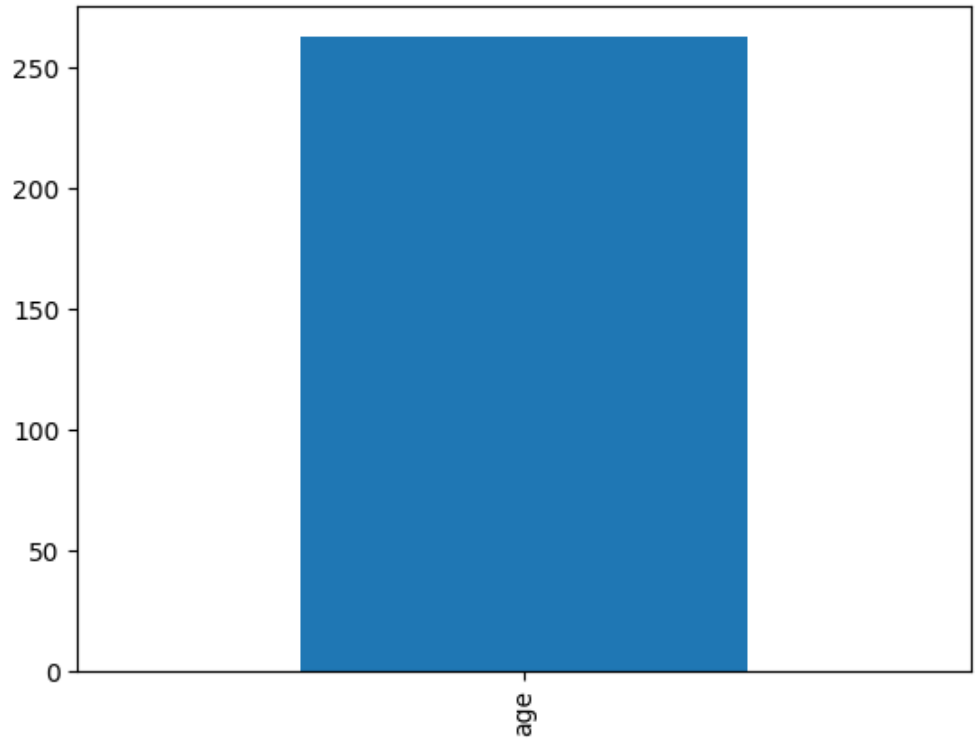
Out[53]:

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


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

In [54]:

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



In [55]:

```
from sklearn.linear_model import LassoCV
```

In [56]:

```
#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))
```

```
0.09109639711159612
0.08490538609884613
```

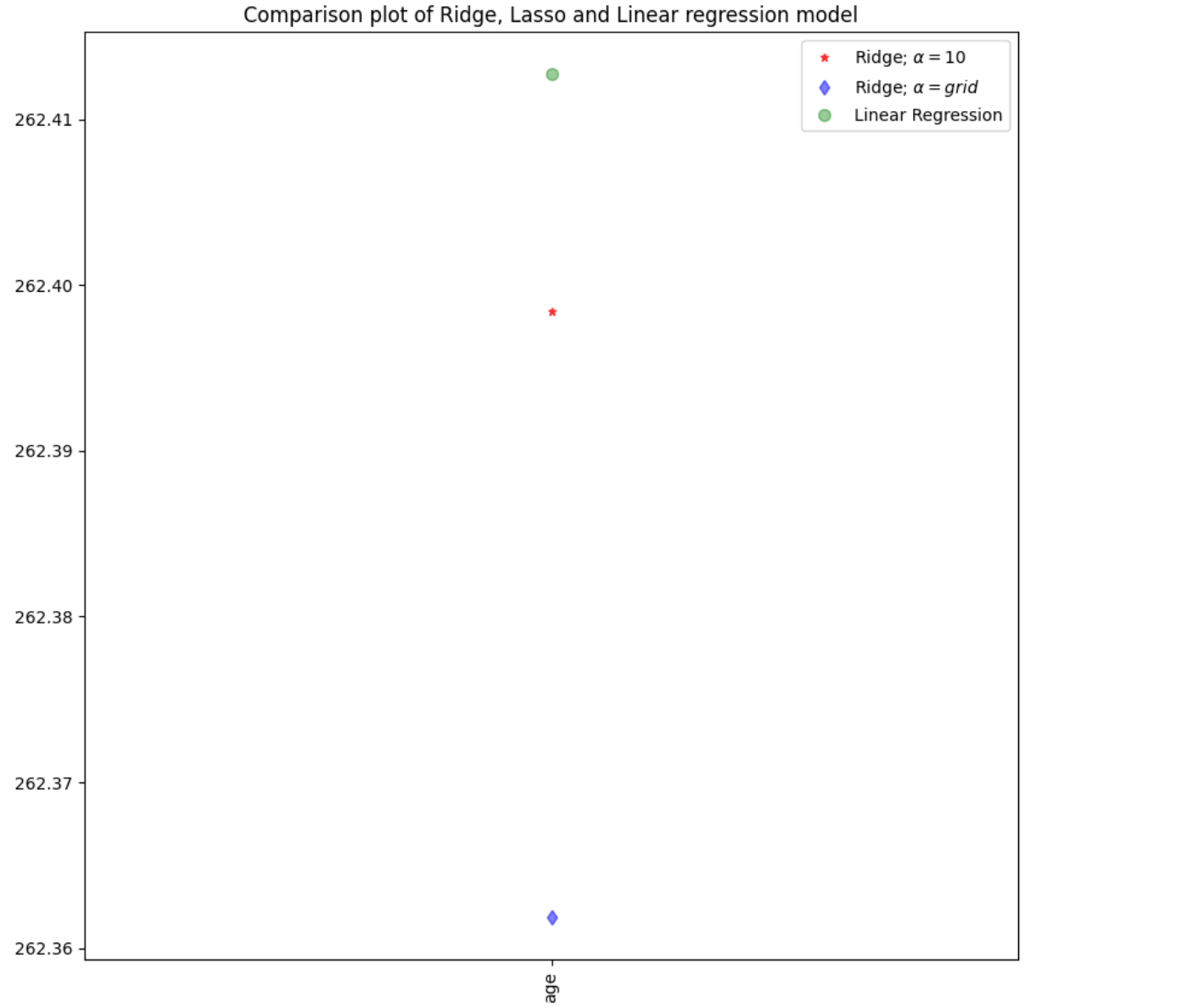
In [57]:

```
#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))
```

```
0.09109639395809044
0.08490704421828055
```

In [58]:

```
t.figure(figsize = (10, 10))
dd plot for ridge regression
t.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label=r'
dd plot for lasso regression
t.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r'Ridge; $
dd plot for linear model
t.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linea
otate axis
t.xticks(rotation = 90)
t.legend()
t.title("Comparison plot of Ridge, Lasso and Linear regression model")
t.show()
```



ElasticNet:

In [60]:

```
from sklearn.linear_model import ElasticNet
```

In [61]:

```
el=ElasticNet()
el.fit(x_train,y_train)
print(el.coef_)
print(el.intercept_)
```

```
[261.74450967]
3115.0831774262424
```

In [62]:

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

In [63]:

```
mean_squared_error=np.mean((y_pred_elastic-y_train)**2)
print(mean_squared_error)
```

```
135077142.70714515
```

In [64]:

```
el=ElasticNet()
el.fit(x_train,y_train)
print(el.score(x_train,y_train))
```

```
0.09109580670592365
```

Logistic regression:

In [66]:

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

In [67]:

```
print('This Dataset has %d rows and %d columns'%(data.shape))
```

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

In [69]:

```
features_matrix=data.iloc[:,0:4]
```

In [71]:

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

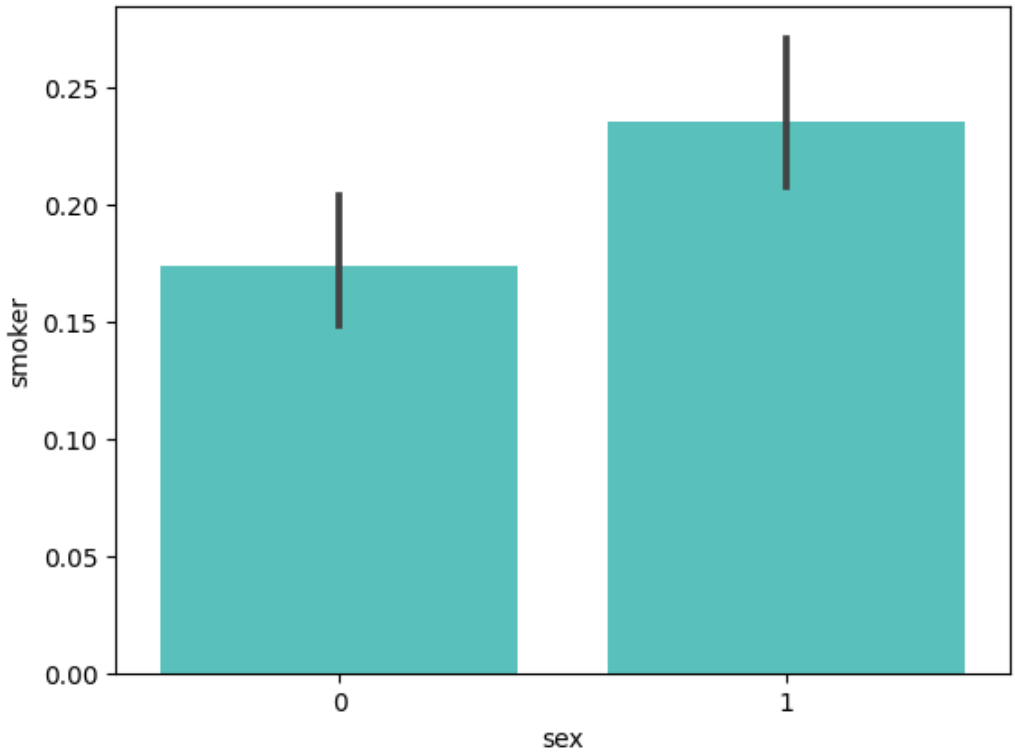
In [72]:

```
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).reshape(-1,1).shape))
```

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

In [78]:

```
sns.barplot(x='sex', y='smoker', data=data, color="mediumturquoise")
plt.show()
```



In [79]:

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

In [81]:

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

In [82]:

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

In [83]:

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

In [84]:

```
observation=[[1,0,0.99539,-0.0588]]
```

In [85]:

```
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 [86]:

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

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

In [87]:

```
print("the model says the probability of the observation we passed belonging to class[0] Is %s" " "%(
print("the model says the probability of the observation we passed belonging to class['1'] Is %s" " "%
```

```
the model says the probability of the observation we passed belonging to class[0] Is 0.
8057075871331396
the model says the probability of the observation we passed belonging to class['1'] Is
0.8057075871331396
```

In [90]:

```
x=np.array(data['sex']).reshape(-1,1)
y=np.array(data['smoker']).reshape(-1,1)
```

In [91]:

```
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))
```

```
0.746268656716418

C:\Users\Prathyusha\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\u
tils\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 ra
vel().
  y = column_or_1d(y, warn=True)
```

Conclusion:

after implementing all the models to the given dataset the "logistic regression" is the best model to the given data set ,which gives the best accuracy to the given dataset.

In [ ]:

```

```