Problem statement:To predict the best fit data

1. Data Collection

In [3]:

```
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 [4]:

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

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
...	...	...	...	...	...	...	...
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

2. Data Cleaning and Preprocessing

In [5]:

```
df.head()
```

Out[5]:

	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 [6]:

```
df.tail()
```

Out[6]:

	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 [7]:

```
df.shape
```

Out[7]:

```
(1338, 7)
```

In [8]:

```
df.describe
```

Out[8]:

```
<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 [9]:

```
df.info
```

Out[9]:

```
<bound method DataFrame.info 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 [10]:

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

Out[10]:

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

In [11]:

```
df.isna().sum()
```

Out[11]:

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

In [12]:

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

Out[12]:

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

In [13]:

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

Out[13]:

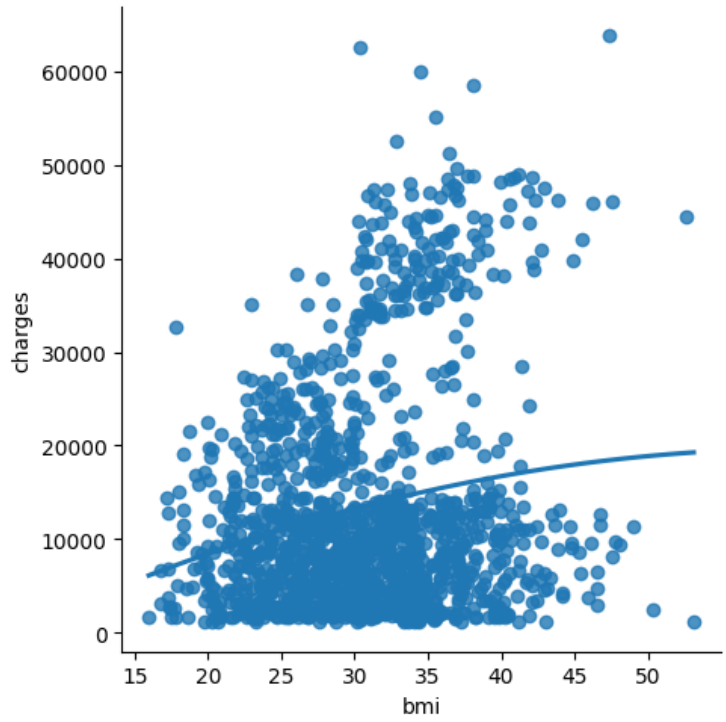
	age	sex	bmi	children	smoker	region	charges
0	19	1	27.900	0	yes	southwest	16884.92400
1	18	0	33.770	1	no	southeast	1725.55230
2	28	0	33.000	3	no	southeast	4449.46200
3	33	0	22.705	0	no	northwest	21984.47061
4	32	0	28.880	0	no	northwest	3866.85520
...	...	...	...	...	...	...	...
1333	50	0	30.970	3	no	northwest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

1338 rows × 7 columns

3. Data Visualization

In [15]:

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



In [16]:

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

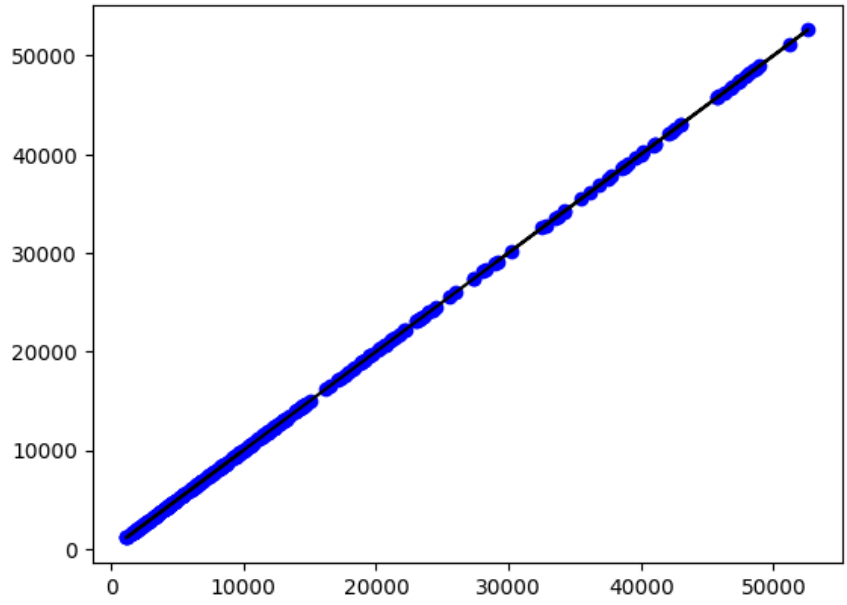
In [17]:

```
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 [18]:

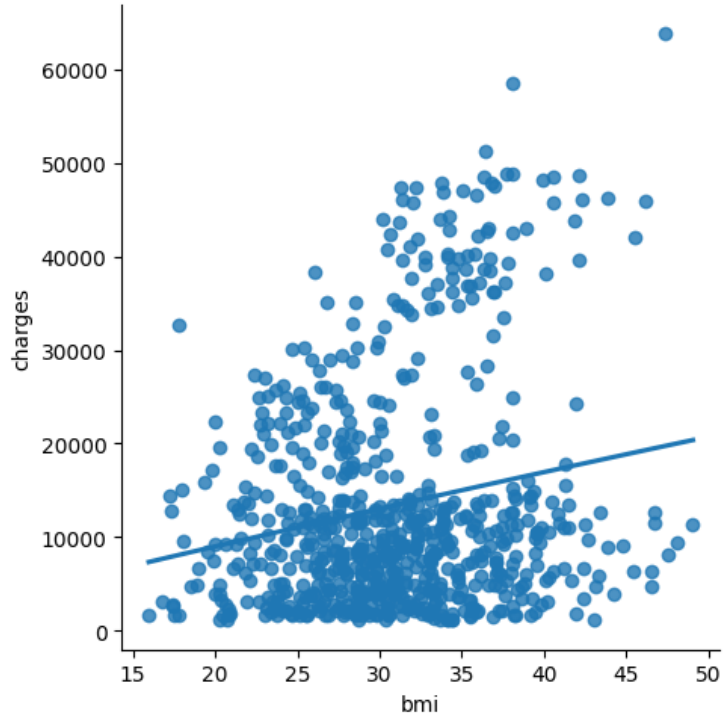
```
y_pred=lr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



working with subset of data

In [20]:

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



In [21]:

```
df700.fillna(method='ffill',inplace=True)
```

In [22]:

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

In [23]:

```
df700.dropna(inplace=True)
```

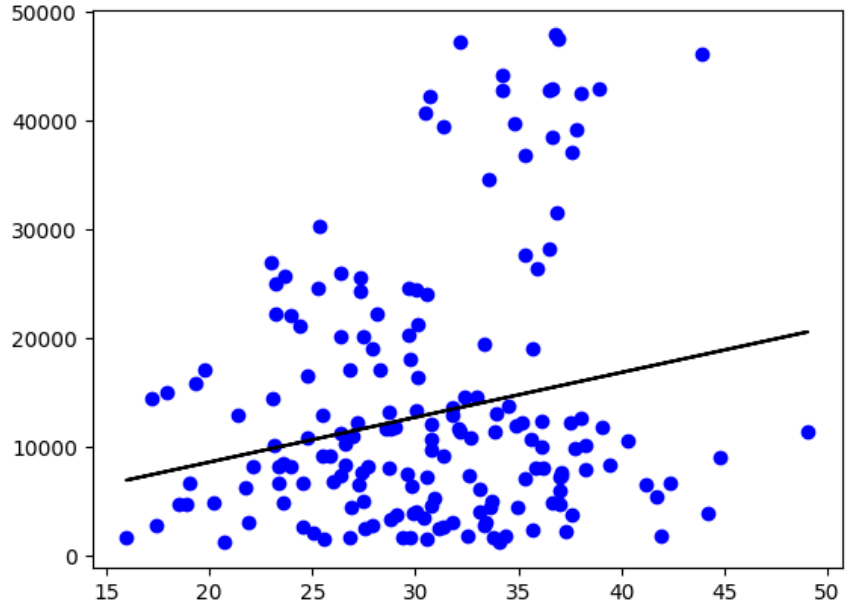
In [24]:

```
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))
```

```
0.02505872132466691
```

In [26]:

```
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 [27]:

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

In [28]:

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

```
0.02505872132466691
```

Ridge Regression

In [29]:

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

In [32]:

```
plt.figure(figsize=(10,10))
sns.heatmap(df700.corr(),annot=True)
plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[32], line 2
      1 plt.figure(figsize=(10,10))
----> 2 sns.heatmap(df700.corr(),annot=True)
      3 plt.show()

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\frame.py:10059, in DataFrame.c
orr(self, method, min_periods, numeric_only)
  10057 cols = data.columns
  10058 idx = cols.copy()
> 10059 mat = data.to_numpy(dtype=float, na_value=np.nan, copy=False)
  10061 if method == "pearson":
  10062     correl = libalgos.nancorr(mat, minp=min_periods)

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\frame.py:1838, in DataFrame.to
_numpy(self, dtype, copy, na_value)
   1836 if dtype is not None:
   1837     dtype = np.dtype(dtype)
-> 1838 result = self._mgr.as_array(dtype=dtype, copy=copy, na_value=na_value)
   1839 if result.dtype is not dtype:
   1840     result = np.array(result, dtype=dtype, copy=False)

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\internals\managers.py:1732, in
BlockManager.as_array(self, dtype, copy, na_value)
   1730         arr.flags.writeable = False
   1731 else:
-> 1732     arr = self._interleave(dtype=dtype, na_value=na_value)
   1733     # The underlying data was copied within _interleave, so no need
   1734     # to further copy if copy=True or setting na_value
   1736 if na_value is not lib.no_default:

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\internals\managers.py:1794, in
BlockManager._interleave(self, dtype, na_value)
   1792     else:
   1793         arr = blk.get_values(dtype)
-> 1794     result[rl.indexer] = arr
   1795     itemmask[rl.indexer] = 1
   1797 if not itemmask.all():

ValueError: could not convert string to float: 'yes'


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

In [33]:

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

In [34]:

```
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 (936, 1)
The dimension of X_test is (402, 1)
```

In [35]:

```
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 [36]:

```
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 0.09109639711159634
The test score for ridge model is 0.08490538609860199
```

In [37]:

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

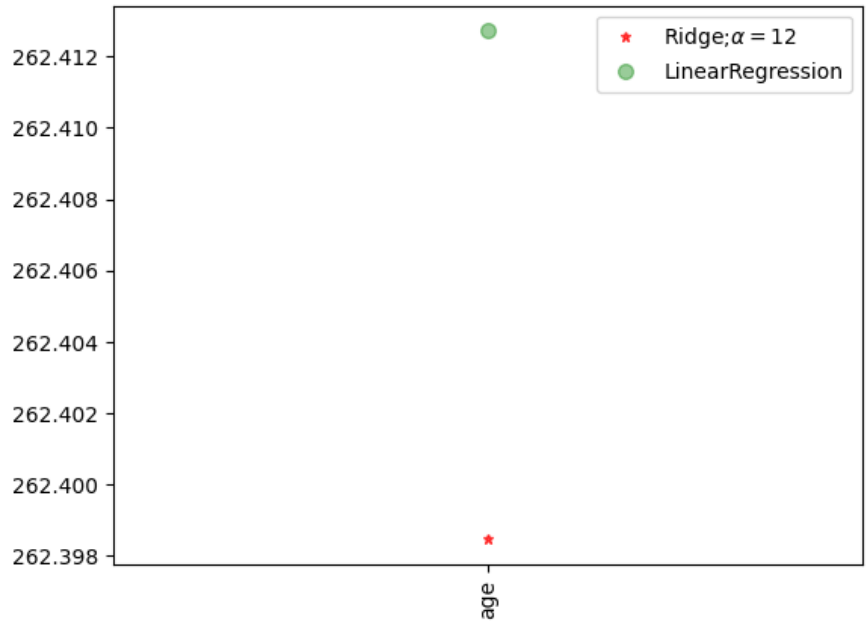
Out[37]:

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

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

In [39]:

```
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker="*",markersize=5,color='red',label=r'Ridge;$\alpha=12$
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker="o",markersize=7,color='green',label='LinearRegression')
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



Lasso Regression

In [40]:

```
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 [41]:

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

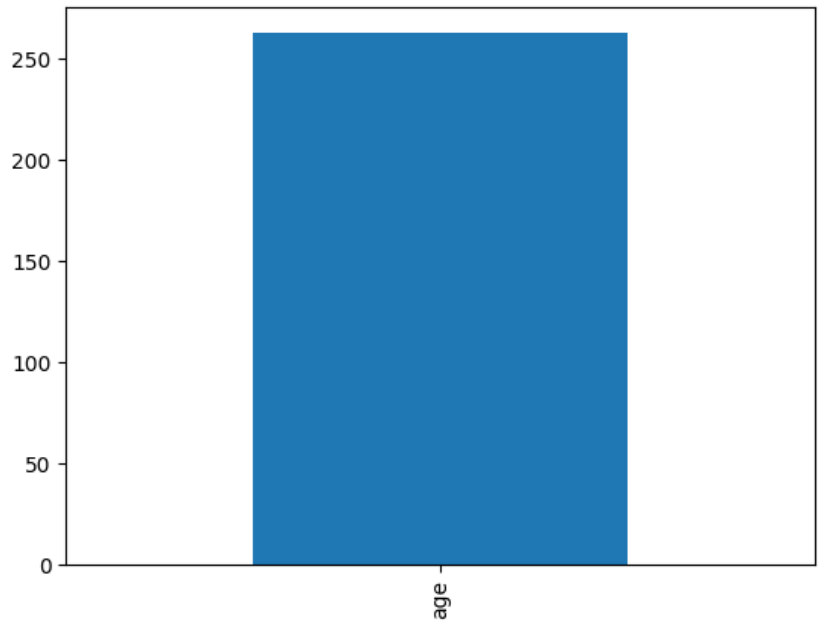
Out[41]:

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

<Figure size 1000x1000 with 0 Axes>

In [42]:

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



In [43]:

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

In [44]:

```
#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
0.08490538609885023
```

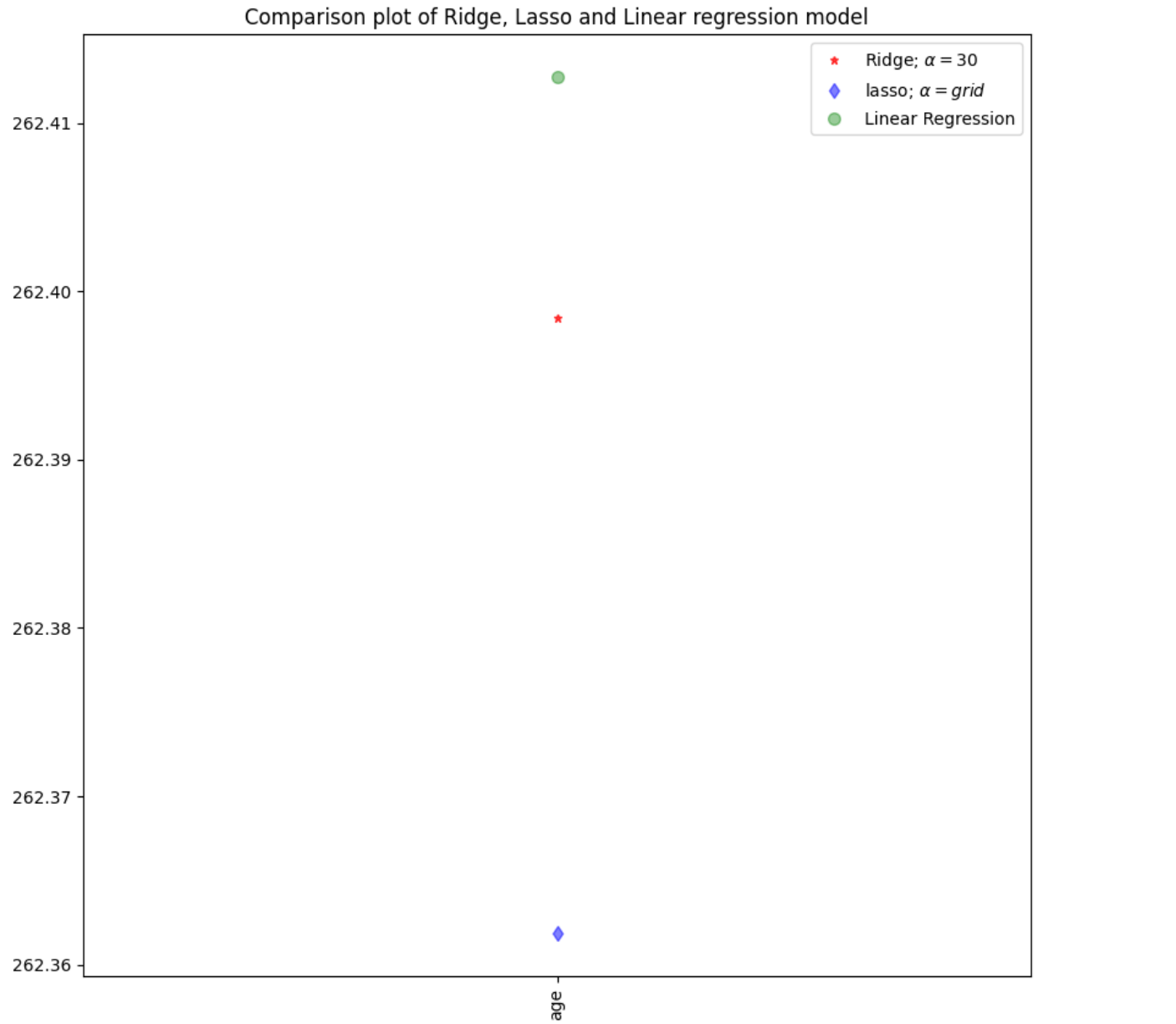
In [45]:

```
#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 [46]:

```
plt.figure(figsize = (10, 10))
#add plot for ridge regression
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label=r'Ridge; $\alpha=30$
#add plot for lasso regression
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r'lasso; $\alpha = grid$')
#add plot for linear model
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear Regression')
#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 [47]:

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

In [48]:

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

```
[261.74450967]
3115.0831774262424
```

In [49]:

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

In [50]:

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

```
135077142.70714515
```

In [51]:

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

```
0.09109580670592365
```

Logistic Regression

In [52]:

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

In [53]:

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

Out[53]:

	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 [54]:

```
df.shape
```

Out[54]:

```
(1338, 7)
```

In [55]:

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

In [56]:

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

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

In [57]:

```
df.head()
```

Out[57]:

	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 [58]:

```
df.tail()
```

Out[58]:

	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 [59]:

```
df.describe
```

```
83     48  female  41.230         4     no  northwest  11033.661700
84     37  female  34.800         2    yes  southwest  39836.519000
85     45    male  22.895         2    yes  northwest  21098.554050
86     57  female  31.160         0    yes  northwest  43578.939400
87     56  female  27.200         0     no  southwest  11073.176000
88     46  female  27.740         0     no  northwest   8026.666600
89     55  female  26.980         0     no  northwest  11082.577200
90     21  female  39.490         0     no  southeast   2026.974100
91     53  female  24.795         1     no  northwest  10942.132050
92     59    male  29.830         3    yes  northeast  30184.936700
93     35    male  34.770         2     no  northwest   5729.005300
94     64  female  31.300         2    yes  southwest  47291.055000
95     28  female  37.620         1     no  southeast   3766.883800
96     54  female  30.800         3     no  southwest  12105.320000
97     55    male  38.280         0     no  southeast  10226.284200
98     56    male  19.950         0    yes  northeast  22412.648500
99     38    male  19.300         0    yes  southwest  15820.699000
100    41  female  31.600         0     no  southwest   6186.127000
101    30    male  25.460         0     no  northeast   3645.089400
102    18  female  30.115         0     no  northeast  21344.846700
```

In [60]:

```
df.info
```

Out[60]:

```
<bound method DataFrame.info of       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
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 [61]:

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

Out[61]:

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

In [62]:

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

35	19	male	20.425	0	0	northwest	1625.433750
36	62	female	32.965	3	0	northwest	15612.193350
37	26	male	20.800	0	0	southwest	2302.300000
38	35	male	36.670	1	1	northeast	39774.276300
39	60	male	39.900	0	1	southwest	48173.361000
40	24	female	26.600	0	0	northeast	3046.062000
41	31	female	36.630	2	0	southeast	4949.758700
42	41	male	21.780	1	0	southeast	6272.477200
43	37	female	30.800	2	0	southeast	6313.759000
44	38	male	37.050	1	0	northeast	6079.671500
45	55	male	37.300	0	0	southwest	20630.283510
46	18	female	38.665	2	0	northeast	3393.356350
47	28	female	34.770	0	0	northwest	3556.922300

In [63]:

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

Out[63]:

	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

In [64]:

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

Out[64]:

	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

In [65]:

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

In [66]:

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

In [67]:

```
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))
```

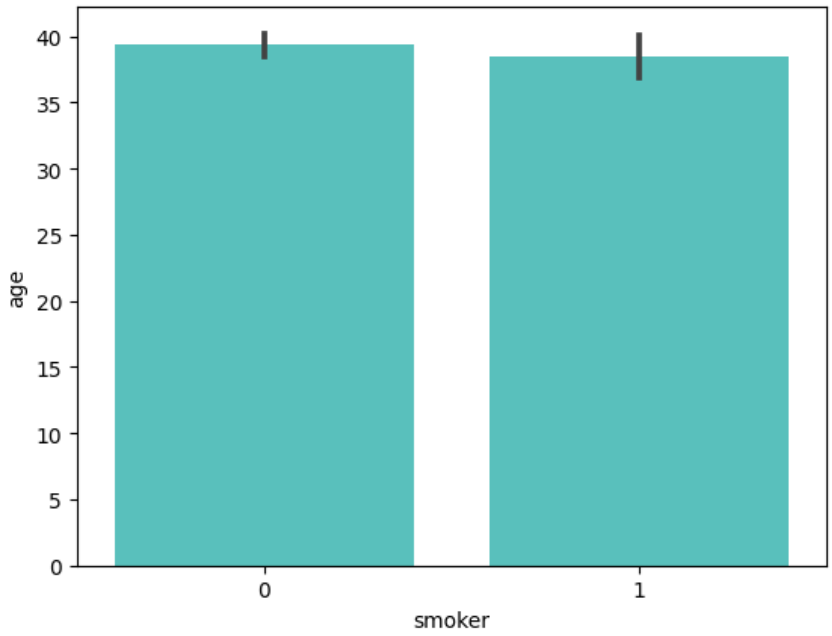
```
The Feature Matrix has 1338 Rows and 4 columns(s)
The Target Matrix has 1338 Rows and 1 columns(s)
```

In [68]:

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

In [69]:

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



In [70]:

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

In [71]:

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

In [72]:

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

In [73]:

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

In [74]:

```
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 [75]:

```
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 [76]:

```
print(" " "The Model says the probability of the observation we passed belonging to class['0'] Is %s" " "%(algorithm.pred
print()
```

```
 The Model says the probability of the observation we passed belonging to class['0'] Is 0.8057075871331396
```

In [77]:

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

In [78]:

```
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\Svijayalakshmi\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 th
e shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Decision tree

In [79]:

```
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 [81]:

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

Out[81]:

	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

In [82]:

```
df.shape
```

Out[82]:

```
(1338, 7)
```

In [83]:

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

Out[83]:

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

In [84]:

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

Out[84]:

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

In [85]:

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

51	21	1	33.630	2	no	northwest	3579.828700
52	48	0	28.000	1	yes	southwest	23568.272000
53	36	0	34.430	0	yes	southeast	37742.575700
54	40	1	28.690	3	no	northwest	8059.679100
55	58	0	36.955	2	yes	northwest	47496.494450
56	58	1	31.825	2	no	northeast	13607.368750
57	18	0	31.680	2	yes	southeast	34303.167200
58	53	1	22.880	1	yes	southeast	23244.790200
59	34	1	37.335	2	no	northwest	5989.523650
60	43	0	27.360	3	no	northeast	8606.217400
61	25	0	33.660	4	no	southeast	4504.662400
62	64	0	24.700	1	no	northwest	30166.618170
63	28	1	25.935	1	no	northwest	4133.641650

In [86]:

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

Out[86]:

	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

In [87]:

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

In [88]:

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

In [89]:

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

In [90]:

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

Out[90]:

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

In [91]:

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

```
0.5121951219512195
```

Random Forest

In [92]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt ,seaborn as sns
```

In [93]:

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

Out[93]:

	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

In [94]:

```
df.shape
```

Out[94]:

```
(1338, 7)
```

In [95]:

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

Out[95]:

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

In [96]:

```
df['bmi'].value_counts()
```

Out[96]:

```
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
```

In [97]:

```
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 [98]:

```
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 [99]:

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

Out[99]:

RandomForestClassifier
RandomForestClassifier()

In [100]:

```
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 [101]:

```
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[101]:

GridSearchCV
estimator: RandomForestClassifier
RandomForestClassifier

In [102]:

```
grid_search.best_score_
```

Out[102]:

```
0.5204398029256787
```

In [103]:

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

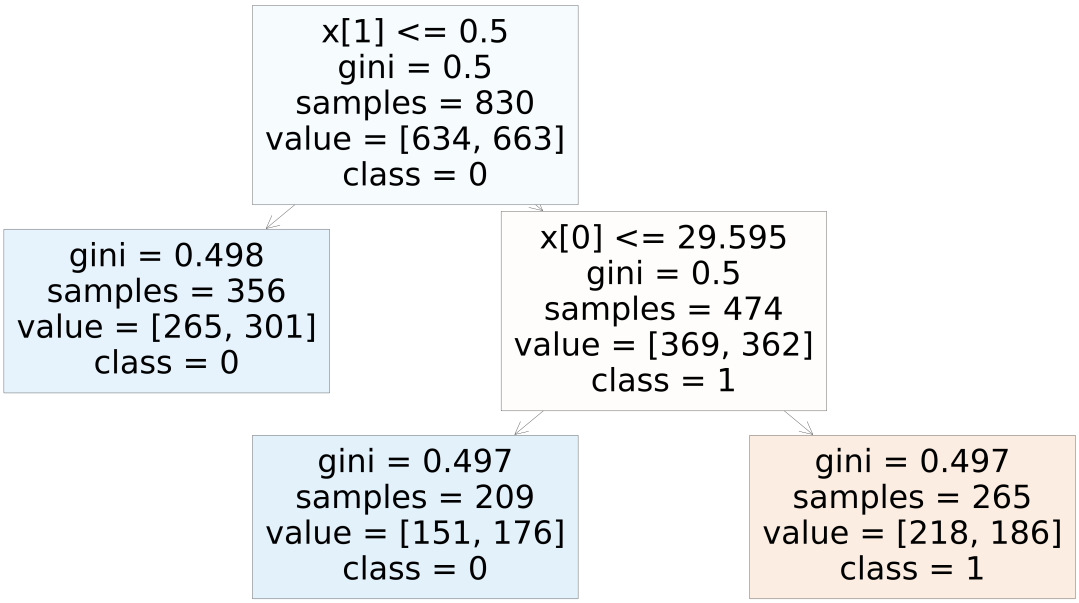
```
RandomForestClassifier(max_depth=3, min_samples_leaf=200, n_estimators=10)
```

In [104]:

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

Out[104]:

```
[Text(0.4, 0.8333333333333334, 'x[1] <= 0.5\ngini = 0.5\nsamples = 830\nvalue = [634, 663]\nclass = 0'),
 Text(0.2, 0.5, 'gini = 0.498\nsamples = 356\nvalue = [265, 301]\nclass = 0'),
 Text(0.6, 0.5, 'x[0] <= 29.595\ngini = 0.5\nsamples = 474\nvalue = [369, 362]\nclass = 1'),
 Text(0.4, 0.16666666666666666, 'gini = 0.497\nsamples = 209\nvalue = [151, 176]\nclass = 0'),
 Text(0.8, 0.16666666666666666, 'gini = 0.497\nsamples = 265\nvalue = [218, 186]\nclass = 1')]
```

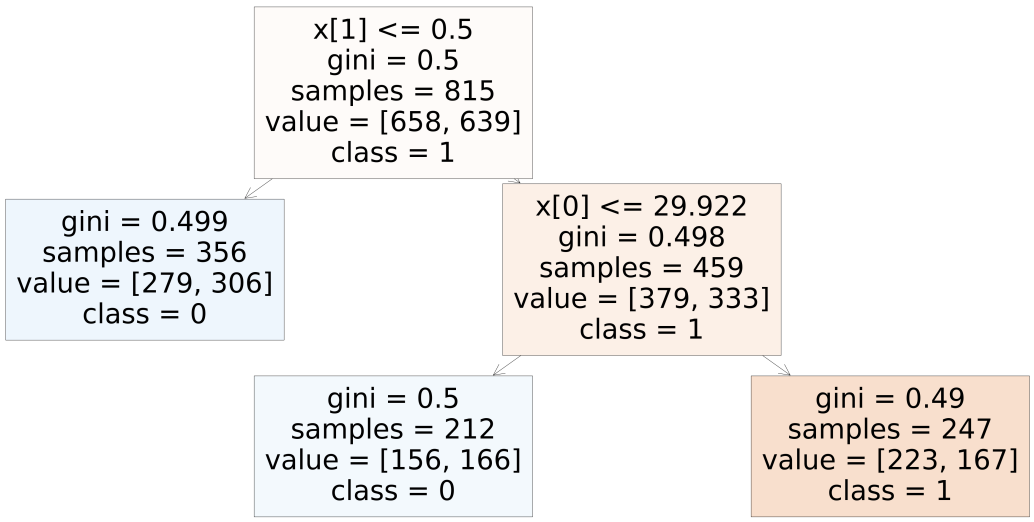


In [105]:

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

Out[105]:

```
[Text(0.4, 0.8333333333333334, 'x[1] <= 0.5\ngini = 0.5\nsamples = 815\nvalue = [658, 639]\nclass = 1'),
 Text(0.2, 0.5, 'gini = 0.499\nsamples = 356\nvalue = [279, 306]\nclass = 0'),
 Text(0.6, 0.5, 'x[0] <= 29.922\ngini = 0.498\nsamples = 459\nvalue = [379, 333]\nclass = 1'),
 Text(0.4, 0.16666666666666666, 'gini = 0.5\nsamples = 212\nvalue = [156, 166]\nclass = 0'),
 Text(0.8, 0.16666666666666666, 'gini = 0.49\nsamples = 247\nvalue = [223, 167]\nclass = 1')]
```



In [106]:

```
rf_best.feature_importances_
```

Out[106]:

```
array([0.53685391, 0.46314609])
```

In [107]:

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

In [108]:

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

Out[108]:

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

In [109]:

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

```
0.4146341463414634
```

In [ ]: