problem statement:Predicting insurance 'charges' based on 'age' and 'bmi'

Linear regression

1.Data Collection

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sls
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
df=pd.read_csv(r"C:\Users\DELL\Downloads\insurance.csv")
df
```

Out[1]:

	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 [2]:
```
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 [3]:
```
df.isna().any()
```

Out[3]:
```
age         False
sex         False
bmi         False
children    False
smoker      False
region      False
charges     False
dtype: bool
```

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

In [5]:
```
df.describe()
```

Out[5]:

	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

In [6]:
```
df['region'].value_counts()
```

Out[6]:
```
region
southeast    364
southwest    325
northwest    325
northeast    324
Name: count, dtype: int64
```

In [7]:
```
convert={"sex":{"female":1,"male":0}}
df=df.replace(convert)
df
```

Out[7]:

	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

In [8]:
```
convert={"region":{"southwest":1,"southeast":2,"northwest":3,"northeast":4}}
df=df.replace(convert)
df
```

Out[8]:

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

1338 rows × 7 columns

In [9]:
```
convert={"smoker":{"yes":1,"no":0}}
df=df.replace(convert)
df
```

Out[9]:

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

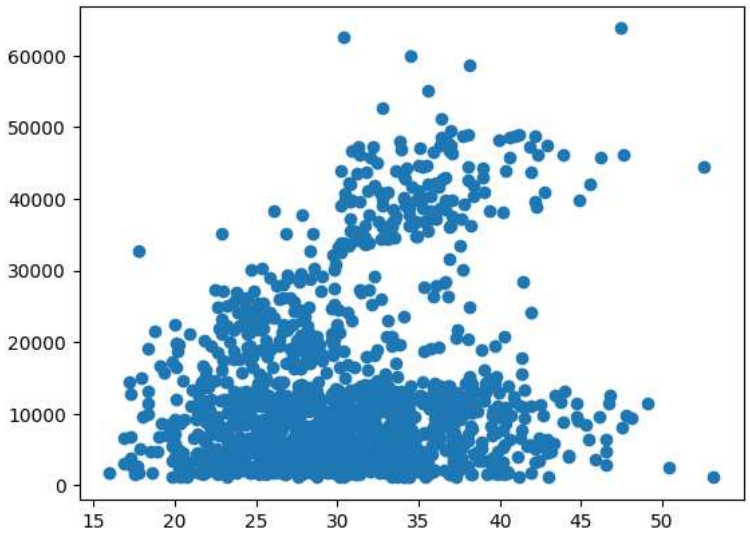
1338 rows × 7 columns

In [ ]:
```

```

3.Data Visualization

In [10]:
```
#checking the relationship between bmi and charges
plt.scatter(df['bmi'],df['charges'])
```

Out[10]:
```
<matplotlib.collections.PathCollection at 0x1f79428c850>
```



In [11]:
```
#identifying the dependent and independent variables
x=df[['bmi']]
y=df['charges']
x.head(10)
```

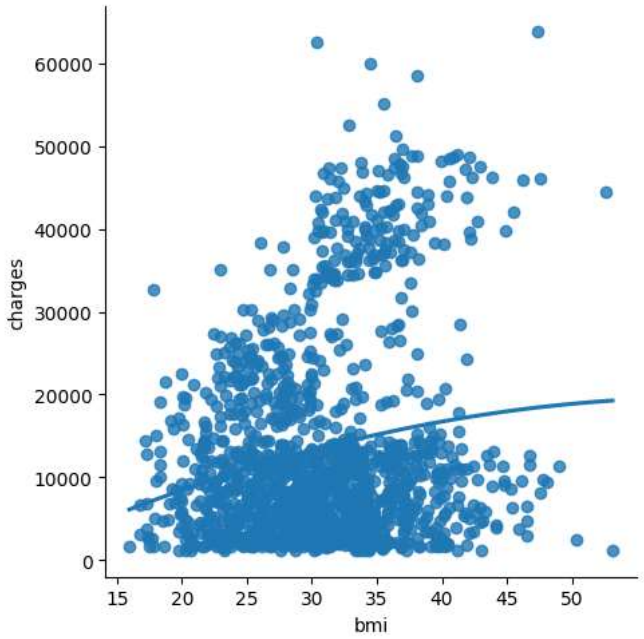
Out[11]:

	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

In [12]:
```
y.head(10)
```

Out[12]:
```
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
Name: charges, dtype: float64
```

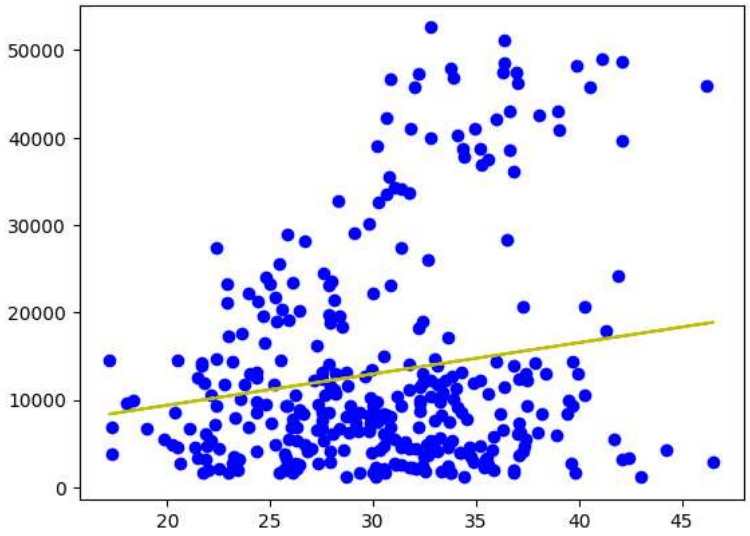
In [13]:
```
sls.lmplot(x='bmi',y='charges',order=2,data=df,ci=None)
plt.show()
```



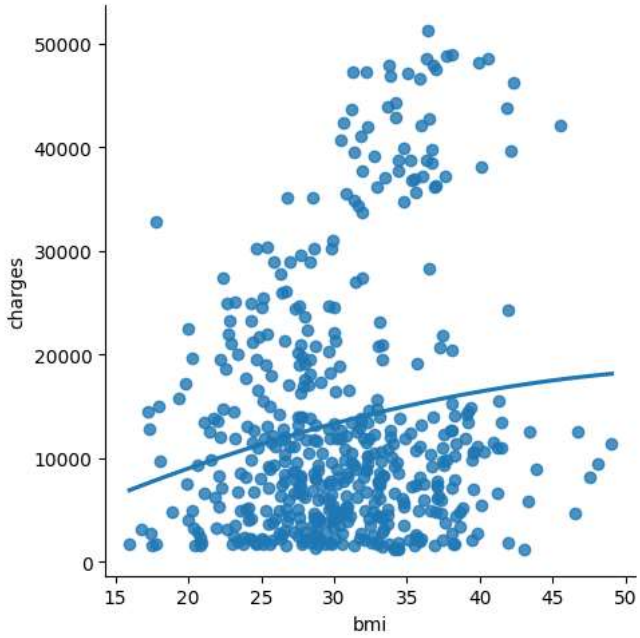
In [14]:
```
#Splitting the data into train and test data
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=0)
lr=LinearRegression()
lr.fit(x_train,y_train)
print(lr.score(x_test,y_test))
```

```
0.050136213258239914
```

In [15]:
```
y_pred=lr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='y')
plt.show()
```



In [16]:
```
#Taking subset of data
df500=df[:][:500]
sls.lmplot(x='bmi',y='charges',order=2,ci=None,data=df500)
plt.show()
```



In [17]:
```
df500.fillna(method='ffill',inplace=True)
```

In [18]:
```
x=np.array(df500["bmi"]).reshape(-1,1)
y=np.array(df500['charges']).reshape(-1,1)
```

In [19]:
```
#Evaluation of model
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)
```

```
0.050136213258239914
```

In [20]:
```
lr.score(x_test,y_test)
```

Out[20]:
```
0.050136213258239914
```

In [21]:
```
#conclusion:the model accuracy is 5%.It is worst fit.
```

Ridge Regression

In [22]:
```
from sklearn.linear_model import Lasso,Ridge
from sklearn.preprocessing import StandardScaler
plt.figure(figsize=(10,10))
sls.heatmap(df500.corr(),annot=True)
plt.show()
```

In [23]:
```
features=df.columns[0:1]
target=df.columns[-1]
```

In [24]:
```
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 [25]:
```
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 [26]:
```
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))
```

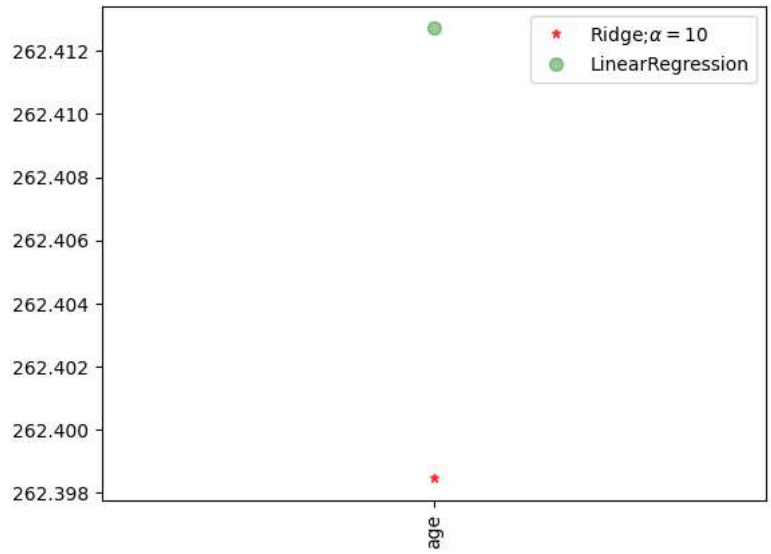
```
Ridge Model:

The train score for ridge model is 0.09109639711159634
The test score for ridge model is 0.08490538609860176
```

In [27]:
```
plt.figure(figsize=(10,10))
```

Out[27]:
```
<Figure size 1000x1000 with 0 Axes>
```

<Figure size 1000x1000 with 0 Axes>

In [29]:
```
plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker="*",markersize=5,color='red',label=r'Ridge;$\alpha=10$',zorder=7)
plot(features,lr.coef_,alpha=0.4,linestyle='none',marker="o",markersize=7,color='green',label='LinearRegression')
xticks(rotation=90)
legend()
show()
```



Lasso Regression

In [30]:
```
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))
```

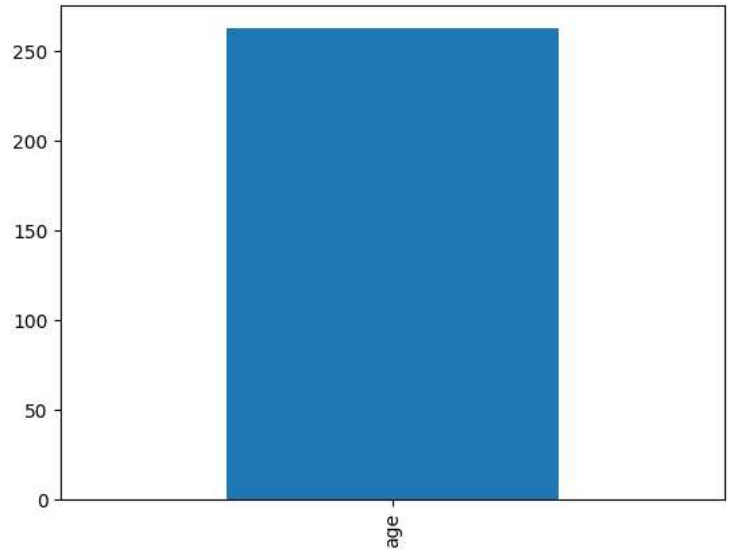
```
Ridge Model:

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

In [31]:
```
plt.figure(figsize=(10,10))
```

Out[31]:
```
<Figure size 1000x1000 with 0 Axes>
```

<Figure size 1000x1000 with 0 Axes>

In [32]:
```
pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "bar")
plt.show()
```



In [33]:
```
from sklearn.linear_model import LassoCV
```

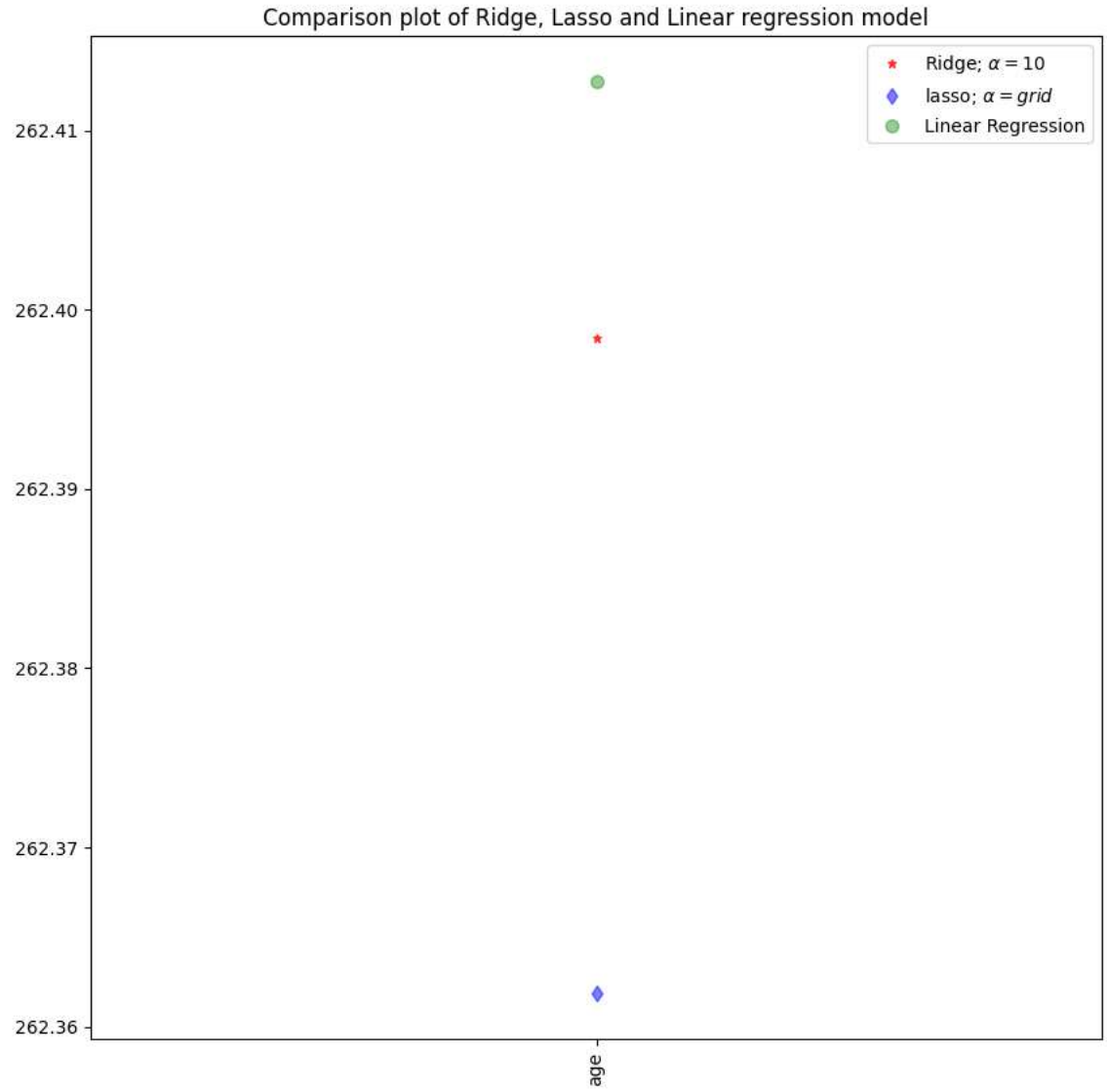
In [34]:
```
#using the linear cv model
from sklearn.linear_model import RidgeCV
#cross validation
ridge_cv=RidgeCV(alphas =[0.0001,0.001,0.01,0.1,1,10]).fit(x_train,y_train)
#score
print(ridge_cv.score(x_train,y_train))
print(ridge_cv.score(x_test,y_test))
```

```
0.09109639711159612
0.08490538609884613
```

In [35]:
```
#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 [36]:
```
igure(figsize = (10, 10))
plot for ridge regression
lot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label=r'Ridge; $\alpha=10$',zorder=7)
plot for lasso regression
lot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r'lasso; $\alpha = grid$')
plot for linear model
lot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear Regression')
te axis
ticks(rotation = 90)
egend()
itle("Comparison plot of Ridge, Lasso and Linear regression model")
how()
```



Elastic Net

In [37]:
```
from sklearn.linear_model import ElasticNet
el=ElasticNet()
el.fit(x_train,y_train)
print(el.coef_)
print(el.intercept_)
```

```
[261.74450967]
3115.0831774262424
```

In [38]:
```
y_pred_elastic=el.predict(x_train)
```

In [39]:
```
mean_squared_error=np.mean((y_pred_elastic-y_train)**2)
print(mean_squared_error)
```

```
135077142.70714515
```

In [40]:
```
el=ElasticNet()
el.fit(x_train,y_train)
print(el.score(x_train,y_train))
```

```
0.09109580670592365
```

In [ ]:
```
#conclusion:the model has 9% accuracy
```

Logistic Regression

In [41]:
```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
df=pd.read_csv(r"C:\Users\DELL\Downloads\insurance.csv")
df
```

Out[41]:

	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 [42]:
```
df.shape
```

Out[42]: (1338, 7)

In [43]:
```
pd.set_option('display.max_rows',10000000000)
pd.set_option('display.max_columns',10000000000)
pd.set_option('display.width',95)
```

In [44]:
```
print('This Dataset has %d rows and %d columns'%(df.shape))
```

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

In [45]:
```
df.head()
```

Out[45]:

	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 [46]:
```
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 [47]:
```
df.describe()
```

Out[47]:

	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

In [48]:
```
df.isnull().sum()
```

Out[48]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

In [49]:
```
convert={"smoker":{"yes":1,"no":0}}
df=df.replace(convert)
df
```

Out[49]:

	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

In [50]:
```
convert={"region":{"southwest":1,"southeast":2,"northwest":3,"northeast":4}}
df=df.replace(convert)
df
```

394	46	female	32.300	2	0	4	9411.005000
395	46	male	19.855	0	0	3	7526.706450
396	43	female	34.400	3	0	1	8522.003000
397	21	male	31.020	0	0	2	16586.497710
398	64	male	25.600	2	0	1	14988.432000
399	18	female	38.170	0	0	2	1631.668300
400	51	female	20.600	0	0	1	9264.797000
401	47	male	47.520	1	0	2	8083.919800
402	64	female	32.965	0	0	3	14692.669350
403	49	male	32.300	3	0	3	10269.460000
404	31	male	20.400	0	0	1	3260.199000
405	52	female	38.380	2	0	4	11396.900200
406	33	female	24.310	0	0	2	4185.097900

In [51]:
```
convert={"sex":{"female":1,"male":0}}
df=df.replace(convert)
df
```

78	22	1	39.805	0	0	4	2755.020950
79	41	1	32.965	0	0	3	6571.024350
80	31	0	26.885	1	0	4	4441.213150
81	45	1	38.285	0	0	4	7935.291150
82	22	0	37.620	1	1	2	37165.163800
83	48	1	41.230	4	0	3	11033.661700
84	37	1	34.800	2	1	1	39836.519000
85	45	0	22.895	2	1	3	21098.554050
86	57	1	31.160	0	1	3	43578.939400
87	56	1	27.200	0	0	1	11073.176000
88	46	1	27.740	0	0	3	8026.666600
89	55	1	26.980	0	0	3	11082.577200
90	21	1	39.490	0	0	2	2026.974100

In [52]:
```
features_matrix=df.iloc[:,0:4]
```

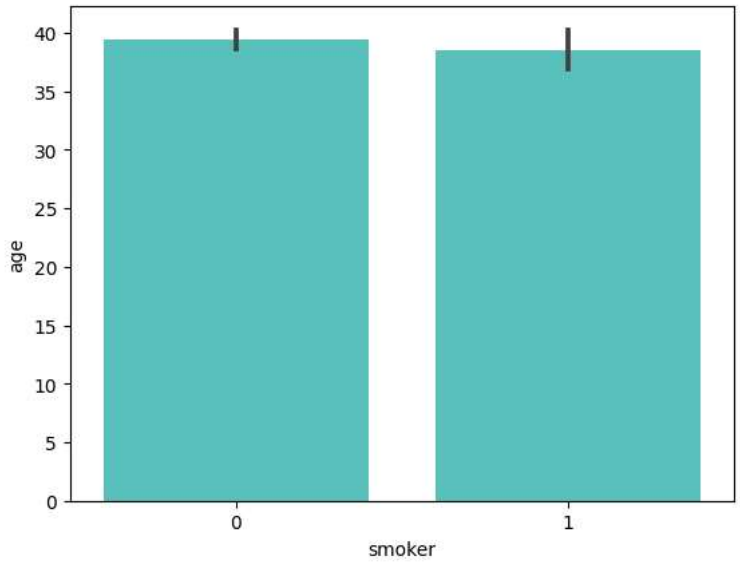
In [53]:
```
target_vector=df.iloc[:,-3]
```

In [54]:
```
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))
```

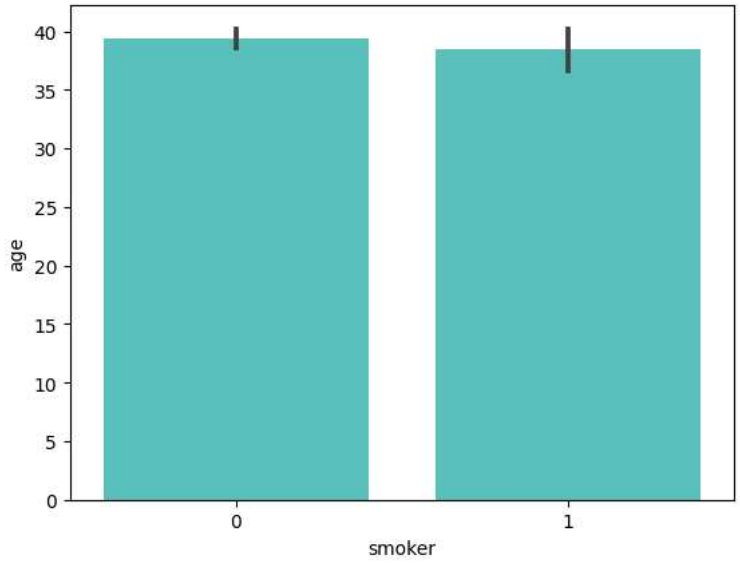
```
The Feature Matrix has 1338 Rows and 4 columns(s)
The Target Matrix has 1338 Rows and 1 columns(s)
```

In [55]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [56]:
```
sns.barplot(x='smoker', y='age', data=df, color="mediumturquoise")
plt.show()
```



In [57]:
```
import matplotlib.pyplot as plt
import seaborn as sns
sns.barplot(x='smoker', y='age', data=df, color="mediumturquoise")
plt.show()
features_matrix_standardized=StandardScaler().fit_transform(features_matrix)
```



In [58]:
```
algorithm=LogisticRegression(max_iter=10000)
```

In [59]:
```
Logistic_Regression_Model=algorithm.fit(features_matrix_standardized,target_vector)
```

In [60]:
```
observation=[[1,0,0.99539,-0.0588]]
```

In [61]:
```
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 [62]:
```
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 [64]:
```
 the probability of the observation we passed belonging to class[0] %s" " "%(algorithm.predict_proba(observation)[0][0]))
l says the probability of the observation we passed belonging to class[1] %s" " "%(algorithm.predict_proba(observation)[0][0])))
```

```
 The Model says the probability of the observation we passed belonging to class[0] 0.8057075871331396
 The Model says the probability of the observation we passed belonging to class[1] 0.8057075871331396
None
```

In [65]:
```
x=np.array(df['age']).reshape(-1,1)
y=np.array(df['smoker']).reshape(-1,1)
```

In [66]:
```
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))
```

```
0.8208955223880597

C:\Users\DELL\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\utils\validation.py:1143: DataConversionWarnin
g: A column-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 [67]:
```
#conclusion:model accuracy is 80%.It is best fit
```

Decision Tree

In [125]:
```
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
df=pd.read_csv(r"C:\Users\DELL\Downloads\insurance.csv")
df
```

32	19	female	28.600	5	no	southwest	4687.797000
33	63	male	28.310	0	no	northwest	13770.097900
34	28	male	36.400	1	yes	southwest	51194.559140
35	19	male	20.425	0	no	northwest	1625.433750
36	62	female	32.965	3	no	northwest	15612.193350
37	26	male	20.800	0	no	southwest	2302.300000
38	35	male	36.670	1	yes	northeast	39774.276300
39	60	male	39.900	0	yes	southwest	48173.361000
40	24	female	26.600	0	no	northeast	3046.062000
41	31	female	36.630	2	no	southeast	4949.758700
42	41	male	21.780	1	no	southeast	6272.477200
43	37	female	30.800	2	no	southeast	6313.759000
44	38	male	37.050	1	no	northeast	6079.671500

In [126]:
```
df.shape
```

Out[126]:
```
(1338, 7)
```

In [127]:
```
df.isna().any()
```

Out[127]:
```
age         False
sex         False
bmi         False
children    False
smoker      False
region      False
charges     False
dtype: bool
```

In [128]:
```
df['region'].value_counts()
```

Out[128]:
```
region
southeast    364
southwest    325
northwest    325
northeast    324
Name: count, dtype: int64
```

In [129]:
```
df['bmi'].value_counts()
```

```
29.640    5
36.850    5
25.300    5
28.700    5
23.655    5
28.785    5
29.700    5
28.975    5
34.200    5
33.345    5
26.030    5
28.025    5
26.315    5
23.180    5
30.400    5
22.515    5
25.840    5
31.920    5
30.780    5
```

In [130]:
```
convert={"sex":{"female":1,"male":0}}
df=df.replace(convert)
df
```

Out[130]:

	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

In [131]:
```
convert={"smoker":{"yes":1,"no":0}}
df=df.replace(convert)
df
```

Out[131]:

	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

In [132]:
```
x=["bmi","children"]
y=["Yes","No"]
all_inputs=df[x]
all_classes=df["sex"]
```

In [133]:
```
(x_train,x_test,y_train,y_test)=train_test_split(all_inputs,all_classes,test_size=0.03)
```

In [134]:
```
clf=DecisionTreeClassifier(random_state=0)
```

In [135]:
```
clf.fit(x_train,y_train)
```

Out[135]:
```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)
```

In [136]:
```
score=clf.score(x_test,y_test)
print(score)
```

```
0.43902439024390244
```

In [137]:
```
#conclusion:this model has 40% accuracy
```

Random Forest

In [138]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt ,seaborn as sns
df=pd.read_csv(r"C:\Users\DELL\Downloads\insurance.csv")
df
```

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

In [139]:
```
df.shape
```

Out[139]:
```
(1338, 7)
```

In [140]:
```
df['region'].value_counts()
```

Out[140]:
```
region
southeast    364
southwest    325
northwest    325
northeast    324
Name: count, dtype: int64
```

In [141]:
```
df['bmi'].value_counts()
```

```
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
24.320    7
34.800    7
27.500    6
19.950    6
29.920    6
```

In [145]:
```
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       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 [146]:
```
n={"smoker":{"yes":1,"no":0}}
df=df.replace(n)
print(df)
```

```
40     24    1  26.600         0       0  northeast   3046.062000
41     31    1  36.630         2       0  southeast   4949.758700
42     41    0  21.780         1       0  southeast   6272.477200
43     37    1  30.800         2       0  southeast   6313.759000
44     38    0  37.050         1       0  northeast   6079.671500
45     55    0  37.300         0       0  southwest  20630.283510
46     18    1  38.665         2       0  northeast   3393.356350
47     28    1  34.770         0       0  northwest   3556.922300
48     60    1  24.530         0       0  southeast  12629.896700
49     36    0  35.200         1       1  southeast  38709.176000
50     18    1  35.625         0       0  northeast   2211.130750
51     21    1  33.630         2       0  northwest   3579.828700
52     48    0  28.000         1       1  southwest  23568.272000
53     36    0  34.430         0       1  southeast  37742.575700
54     40    1  28.690         3       0  northwest   8059.679100
55     58    0  36.955         2       1  northwest  47496.494450
56     58    1  31.825         2       0  northeast  13607.368750
57     18    0  31.680         2       1  southeast  34303.167200
58     53    1  22.880         1       1  southeast  23244.790200
```

In [147]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[147]:
```
RandomForestClassifier
RandomForestClassifier()
```

In [148]:
```
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 [149]:
```
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[149]:
```
GridSearchCV
estimator: RandomForestClassifier
RandomForestClassifier
```

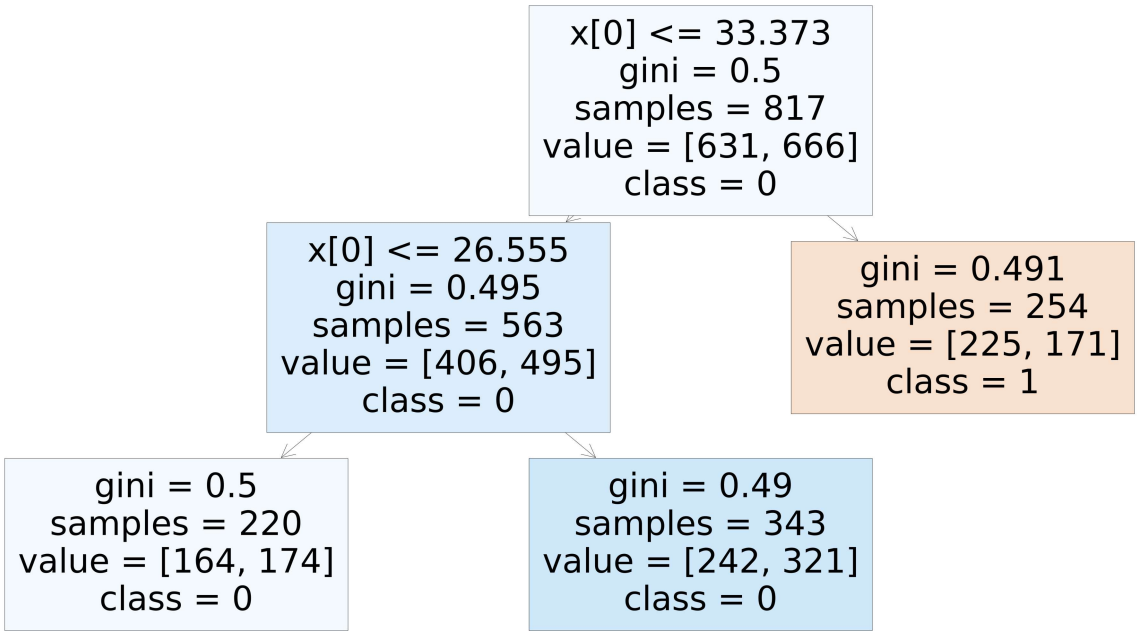
In [150]:
```
grid_search.best_score_
```

Out[150]:
```
0.525056354505507
```

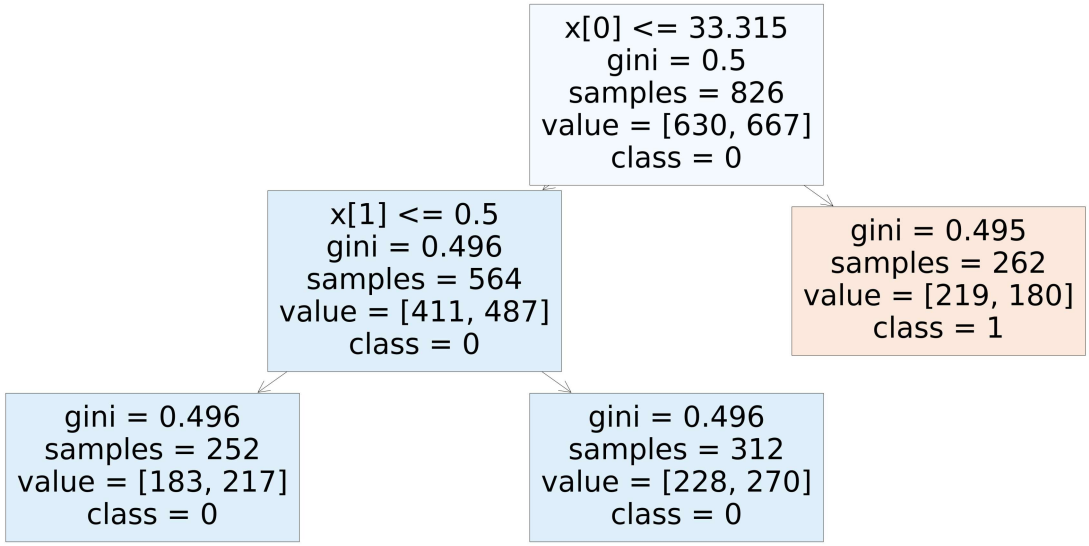
In [151]:
```
rf_best=grid_search.best_estimator_
print(rf_best)
```

```
RandomForestClassifier(max_depth=5, min_samples_leaf=200, n_estimators=25)
```

In [152]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[4],class_names=['1','0'],filled=True);
```



In [153]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(70,30))
plot_tree(rf_best.estimators_[6],class_names=["1","0"],filled=True);
```



In [154]:
```
rf_best.feature_importances_
```

Out[154]:
```
array([0.51284872, 0.48715128])
```

In [155]:
```
rf=RandomForestClassifier(random_state=0)
```

In [156]:
```
rf.fit(x_train,y_train)
```

Out[156]:
```
RandomForestClassifier
RandomForestClassifier(random_state=0)
```

In [157]:
```
score=rf.score(x_test,y_test)
print(score)
```

```
0.3902439024390244
```

In [123]:
```
#conclusion:model has accuracy of 30%.It is best fit.
```

Conclusion

```
#After performing different models we can conclude that "Logistic Regression" is having the highest
accuracy about 80%.
Hence best model that we can consider is Logistic Regression...
```

In [ ]:
```

```