MINI PROJECT

PROBLEM STATEMENT:which model is suitable for insurance dataset

importing packages

In [1]:
```
import numpy as ny
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Read the data

In [2]:
```
df=pd.read_csv(r"C:\Users\USER\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 collection 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.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 [6]:
```
df.shape
```

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

In [7]:
```
df.describe()
```

Out[7]:

	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 [8]:
```
df.isna().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.fillna(method='ffill',inplace=True)
```

In [11]:
```
x=ny.array(df['age']).reshape(-1,1)
y=ny.array(df['children']).reshape(-1,1)
```

In [12]:
```
df.dropna(inplace=True)
```

In [13]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
```

Linear Regression

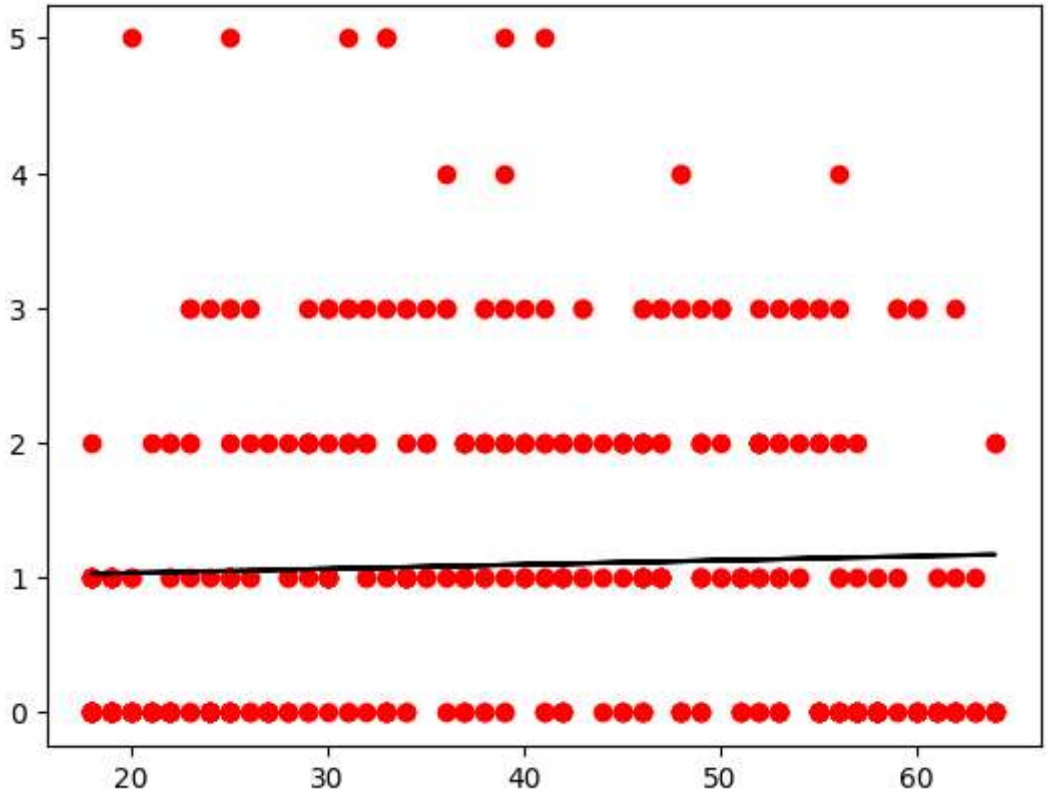
In [14]:
```
from sklearn.linear_model import LinearRegression
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

```
0.0038547098404315694
```

In [15]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
regr.fit(x_train,y_train)
regr.fit(x_train,y_train)
```

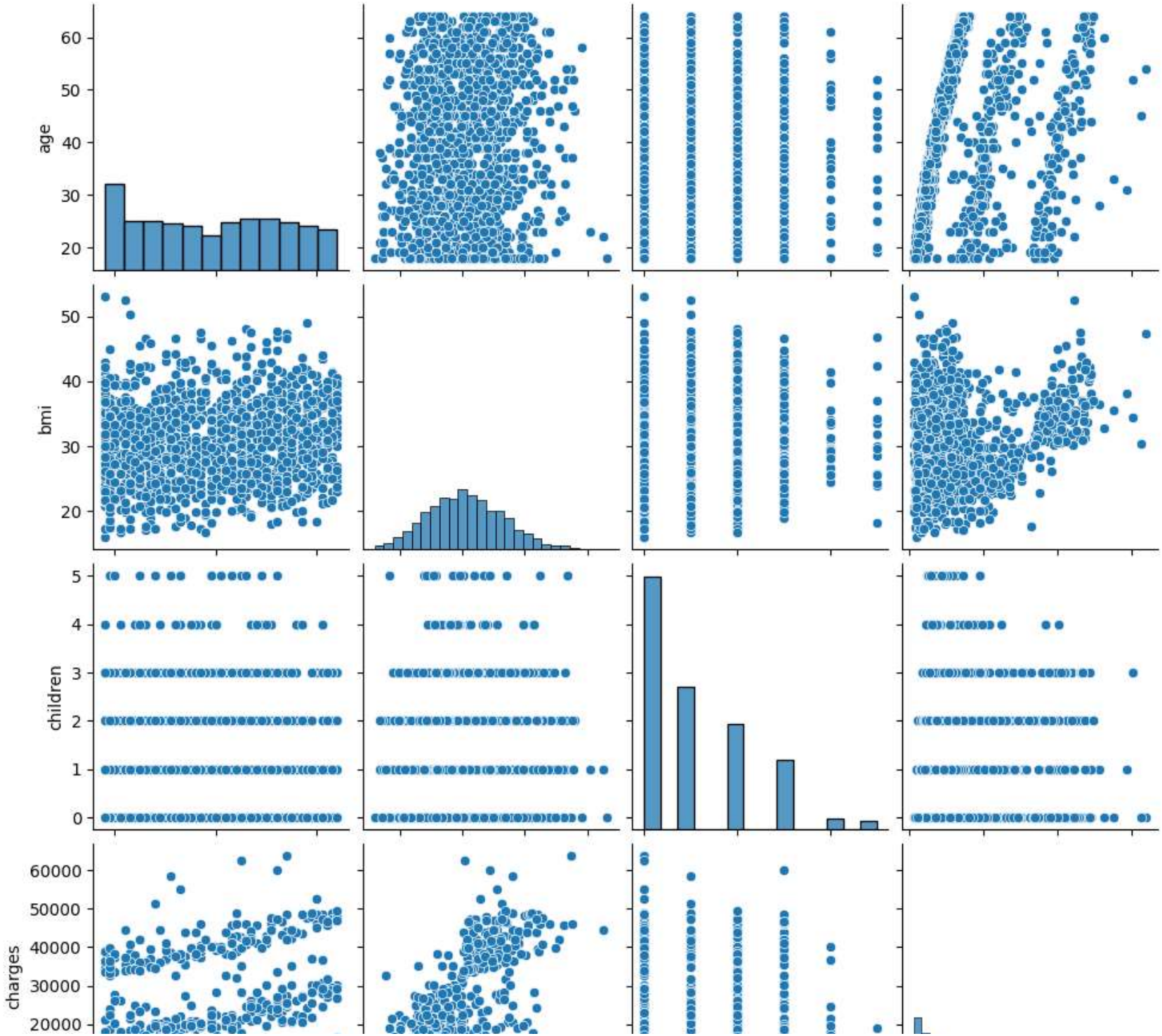
Out[15]:
```
LinearRegression
LinearRegression()
```

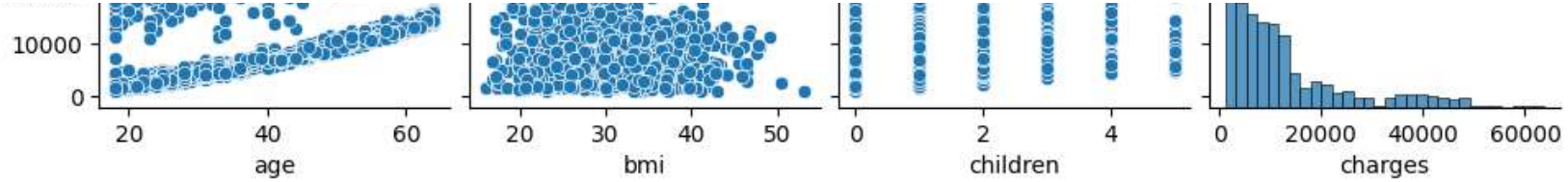
In [16]:
```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='r')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



In [17]:
```
sns.pairplot(df)
```

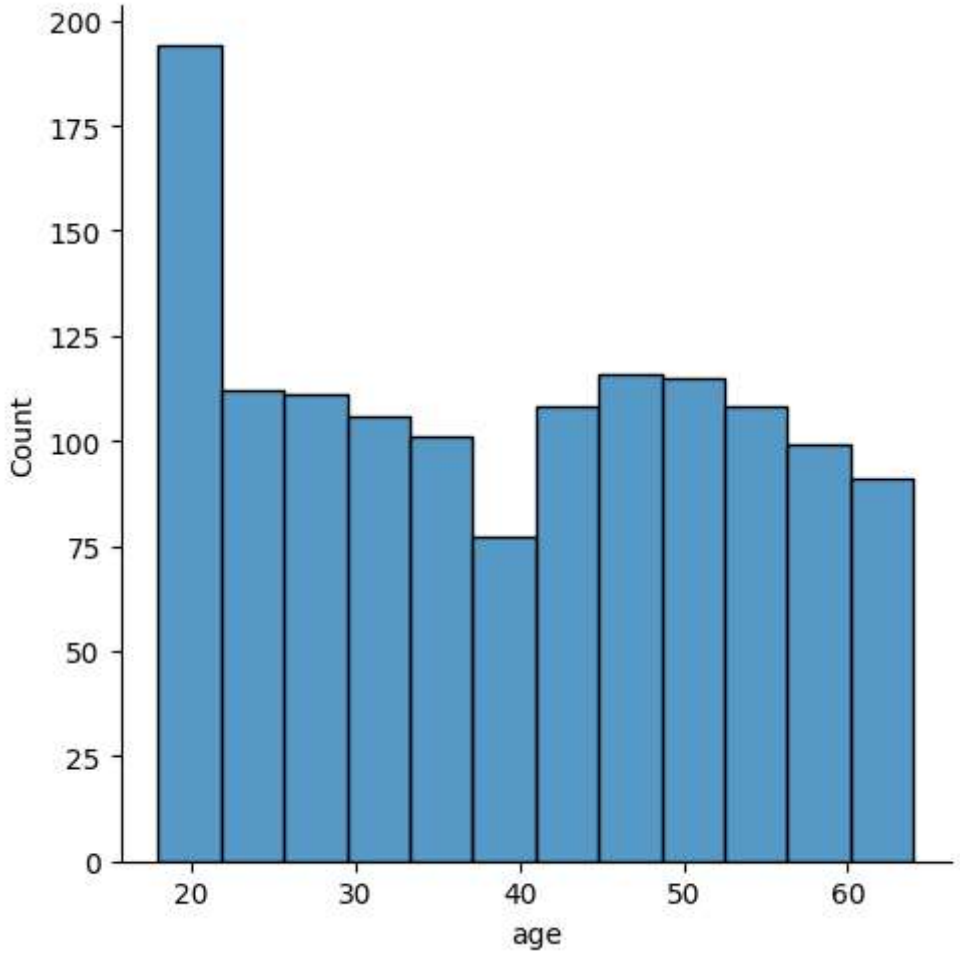
Out[17]:
```
<seaborn.axisgrid.PairGrid at 0x26de3560dd0>
```


age
bmi
children
charges
60
50
40
30
20
50
40
30
20
5
4
3
2
1
0
60000
50000
40000
30000
20000



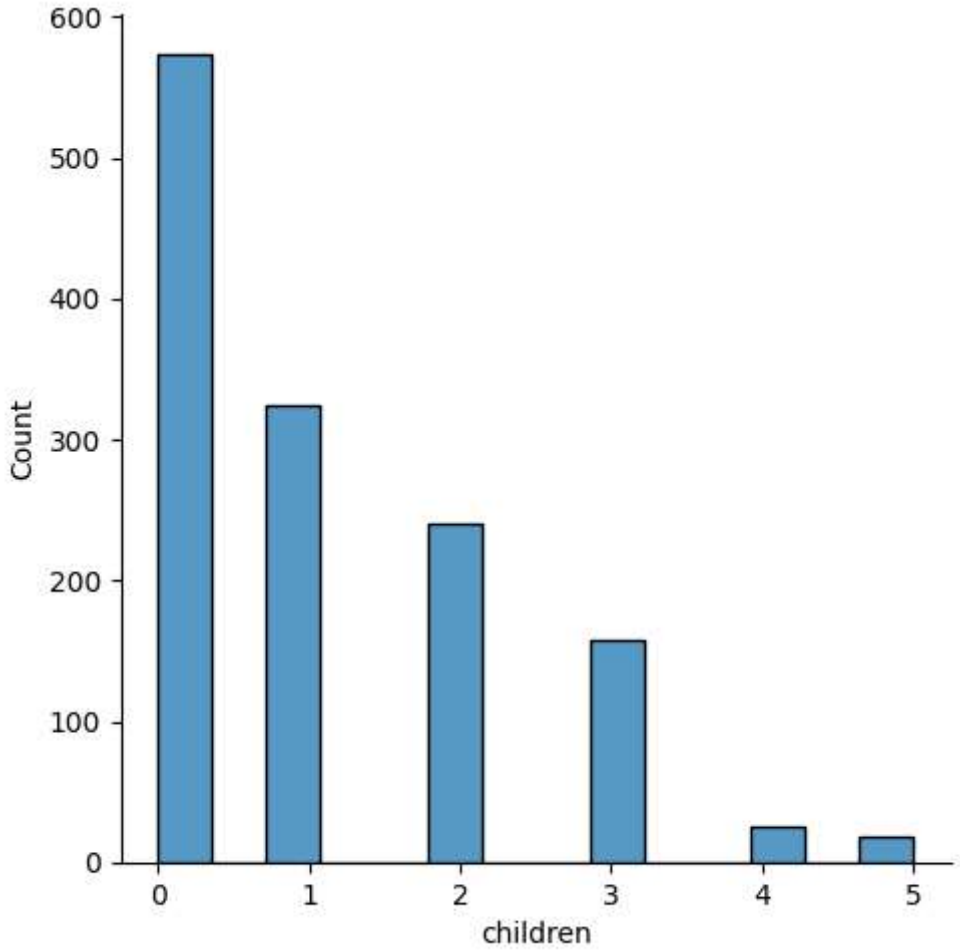
In [18]:
```
sns.displot(df['age'])
```

Out[18]: <seaborn.axisgrid.FacetGrid at 0x26de85780d0>



In [19]:
```
sns.displot(df['children'])
```

Out[19]:
```
<seaborn.axisgrid.FacetGrid at 0x26de8507f90>
```



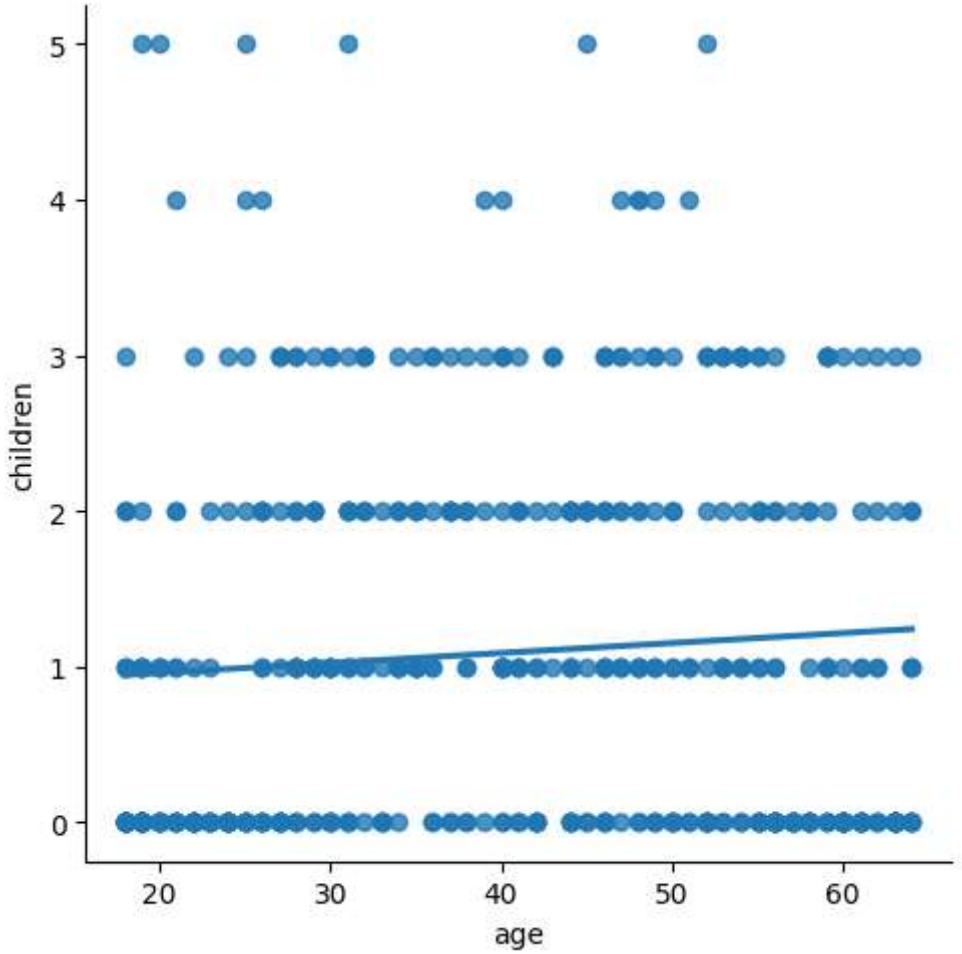
In [20]:
```
plt.figure(figsize=(15,8))
```

Out[20]:
```
<Figure size 1500x800 with 0 Axes>
```

```
<Figure size 1500x800 with 0 Axes>
```

In [21]:
```
df500=df[:][:500]
sns.lmplot(x="age",y="children",data=df500,order=1,ci=None)
```

Out[21]: <seaborn.axisgrid.FacetGrid at 0x26dd8045890>



In [22]:
```
df.fillna(method='ffill',inplace=True)
```

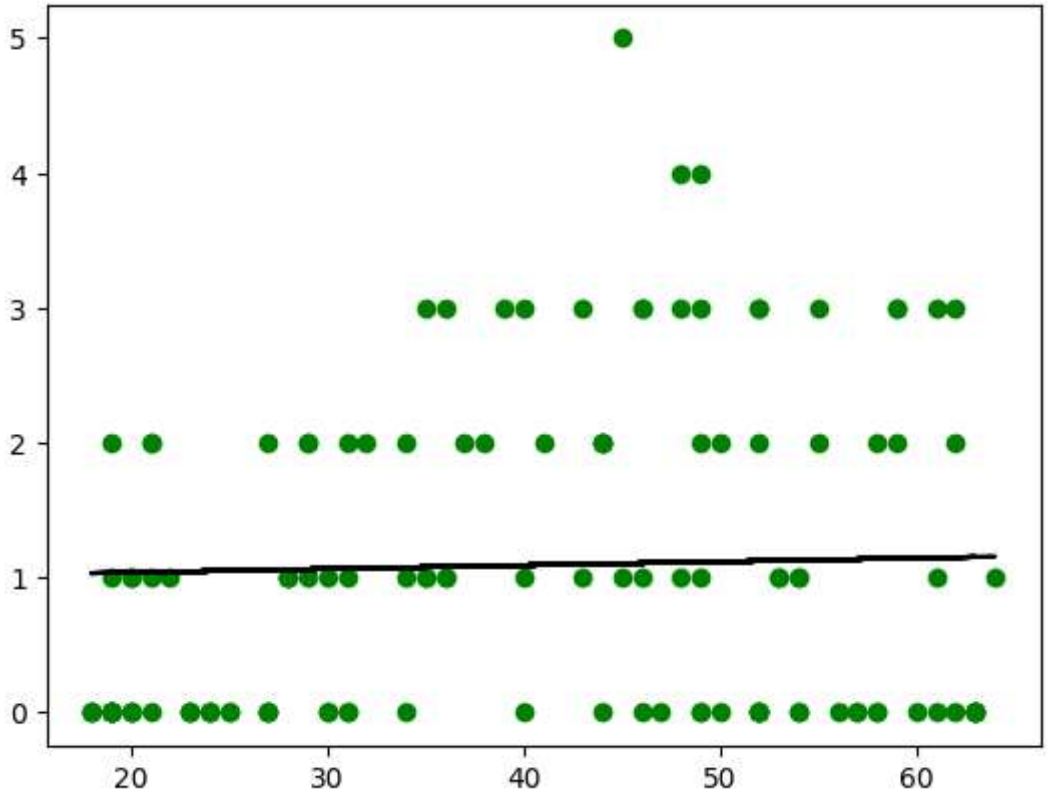
In [23]:
```
x=ny.array(df500['age']).reshape(-1,1)
y=ny.array(df500['children']).reshape(-1,1)
```

In [24]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
```

In [25]:
```
from sklearn.linear_model import LinearRegression
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

```
0.01273421731006008
```

In [26]:
```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='g')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



In [27]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
model=LinearRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
r2=r2_score(y_test,y_pred)
print('r2score:',r2)
```

```
r2score: 0.01273421731006008
```

In [28]:
```
print(model.intercept_)
```

```
[0.98510352]
```

ridge regression

In [29]:
```
from sklearn.linear_model import Ridge,RidgeCV,Lasso
from sklearn.preprocessing import StandardScaler
```

In [30]:
```
df
```

Out[30]:

	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 [31]:
```
df["sex"].value_counts()
```

Out[31]:
```
sex
male      676
female    662
Name: count, dtype: int64
```

In [32]:
```
t={"sex":{"female":1,"male":0}}
df=df.replace(t)
df
```

Out[32]:

	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 [33]:
```
t={"smoker":{"yes":1,"no":2}}
df=df.replace(t)
df
```

Out[33]:

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

1338 rows × 7 columns

In [34]:
```
t={"region":{"northwest":1,"southwest":2,"southeast":3,"northeast":4}}
df=df.replace(t)
df
```

Out[34]:

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

1338 rows × 7 columns

DecisionTree

In [35]:
```
x=["sex","region","smoker"]
y=["f","m"]
all_inputs=df[x]
all_classes=df["sex"]
```

In [36]:
```
from sklearn.model_selection import train_test_split
(x_train,x_test,y_train,y_test)=train_test_split(all_inputs,all_classes,test_size=0.25)
```

In [37]:
```
from sklearn.tree import DecisionTreeClassifier
clf=DecisionTreeClassifier(random_state=0)
```

In [38]:
```
clf.fit(x_train,y_train)
```

Out[38]:
```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)
```

In [39]:
```
score=(clf.score(x_test,y_test))
print(score)
```

```
1.0
```

Data visualization

In [40]:
```
plt.figure(figsize=(10,10))
sns.heatmap(df.corr(),annot=True)
```

Out[40]: <Axes: >


age
sex
bmi
children
smoker
region
arges
1 0.021 0.11 0.042 0.025 -0.0026 0.3
0.021 1 -0.046 -0.017 0.076 -0.0093 -0.057
0.11 -0.046 1 0.013 -0.0038 0.054 0.2
0.042 -0.017 0.013 1 -0.0077 -0.037 0.068
0.025 0.076 -0.0038 -0.0077 1 -0.044 -0.79
-0.0026 -0.0093 0.054 -0.037 -0.044 1 0.05
0.3 -0.057 0.2 0.068 -0.79 0.05 1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

In [41]:
```
df
```

Out[41]:

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

1338 rows × 7 columns

In [42]:
```
features = df.columns[0:2]
target = df.columns[-1]
#X and y values
x = df[features].values
y = df[target].values
#splot
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=17)
print("The dimension of X_train is {}".format(x_train.shape))
print("The dimension of X_test is {}".format(x_test.shape))
#Scale features
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

```
The dimension of X_train is (936, 2)
The dimension of X_test is (402, 2)
```

In [43]:
```
lr = LinearRegression()
#Fit model
lr.fit(x_train, y_train)
#predict
#prediction = lr.predict(X_test)
#actual
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))
#print(lr.score(y_test,prediction))
```

```
Linear Regression Model:

The train score for lr model is 0.08144731818197626
The test score for lr model is 0.1022033122179885
```

In [44]:
```
x_train.shape
```

Out[44]: (936, 2)

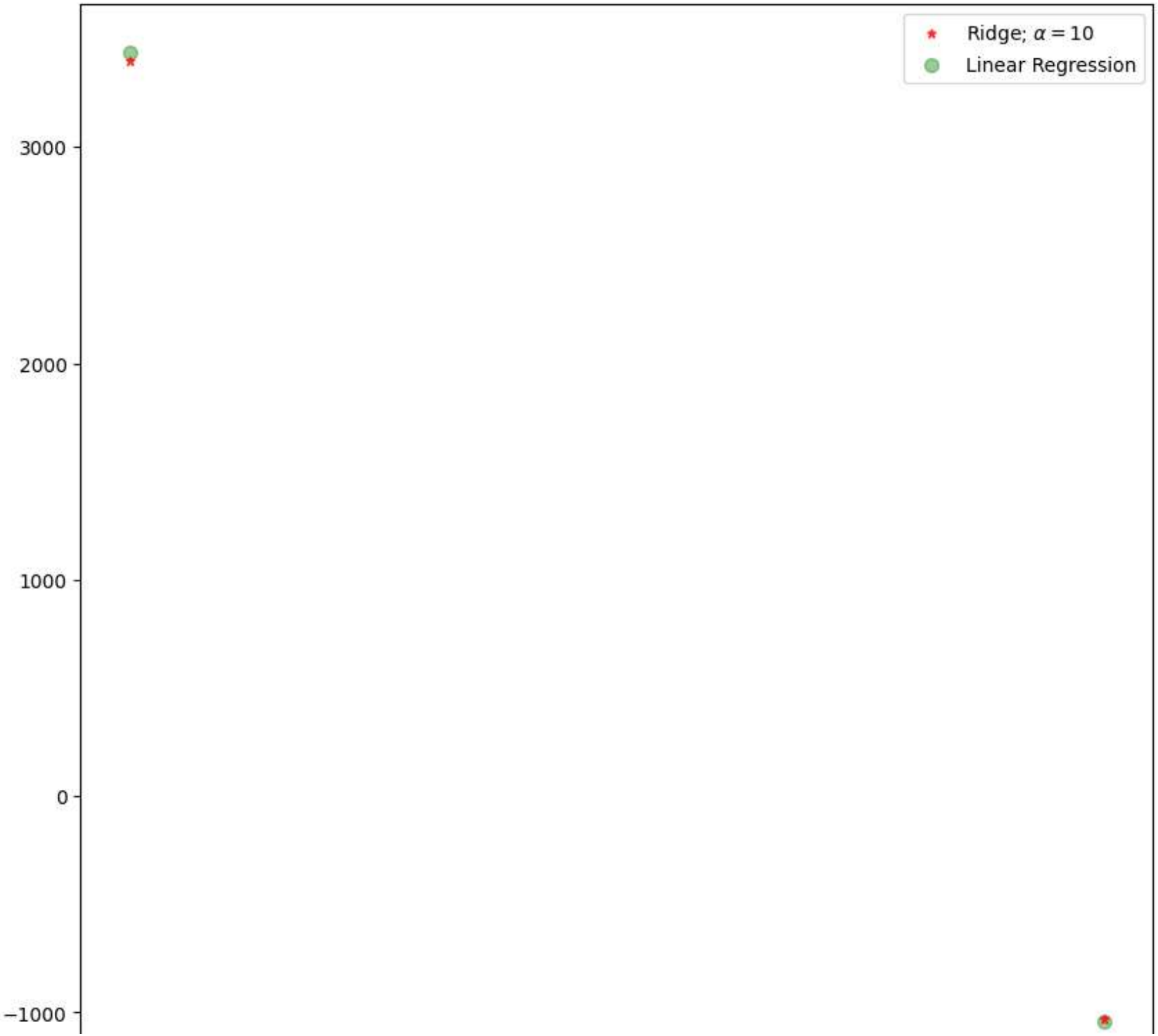
In [45]:
```
y_train.shape
```

Out[45]: (936,)

In [46]:
```
ridgeReg=Ridge(alpha=10)
ridgeReg.fit(x_train,y_train)
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 0.08143796463046804
The test score for ridge model is 0.10202509697425621
```

In [47]:
```
plt.figure(figsize = (10, 10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label=r'Ridge
#plt.plot(rr100.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r'Ridge; $\alpha =
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear Regr
plt.xticks(rotation = 90)
plt.legend()
plt.show()
```


Ridge; $\alpha = 10$
Linear Regression
3000
2000
1000
0
−1000

lasso Regression

In [48]:
```
from sklearn.linear_model import LassoCV
lasso_cv=LassoCV(alphas=[0.0001,0.001,0.01,0.1,1,10],random_state=0).fit(x_train,y_train)
print(lasso_cv.score(x_train,y_train))
print(lasso_cv.score(x_test,y_test))
```

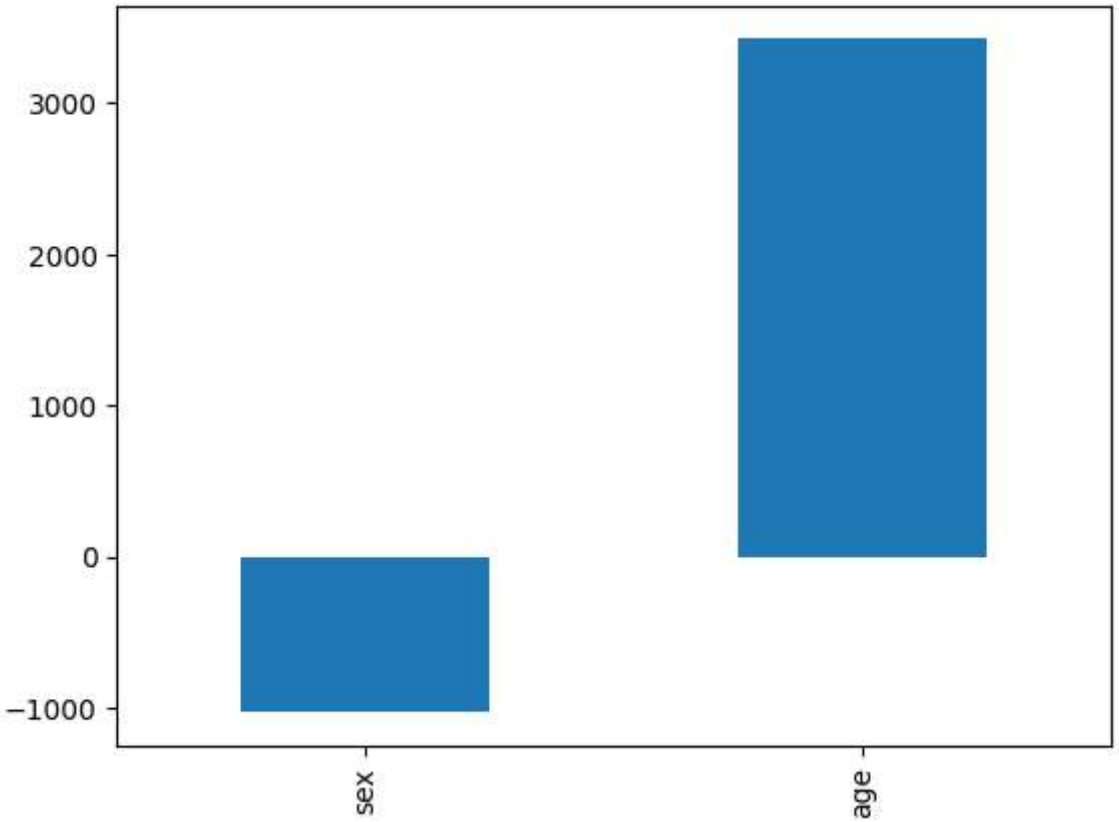
```
0.08144600503720845
0.10226046691368151
```

In [49]:
```
print("\nLasso model:\n")
lasso=Lasso(alpha=10)
lasso.fit(x_train,y_train)
train_score_ls=lasso.score(x_train,y_train)
test_score_ls=lasso.score(x_test,y_test)
print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is {}".format(test_score_ls))
```

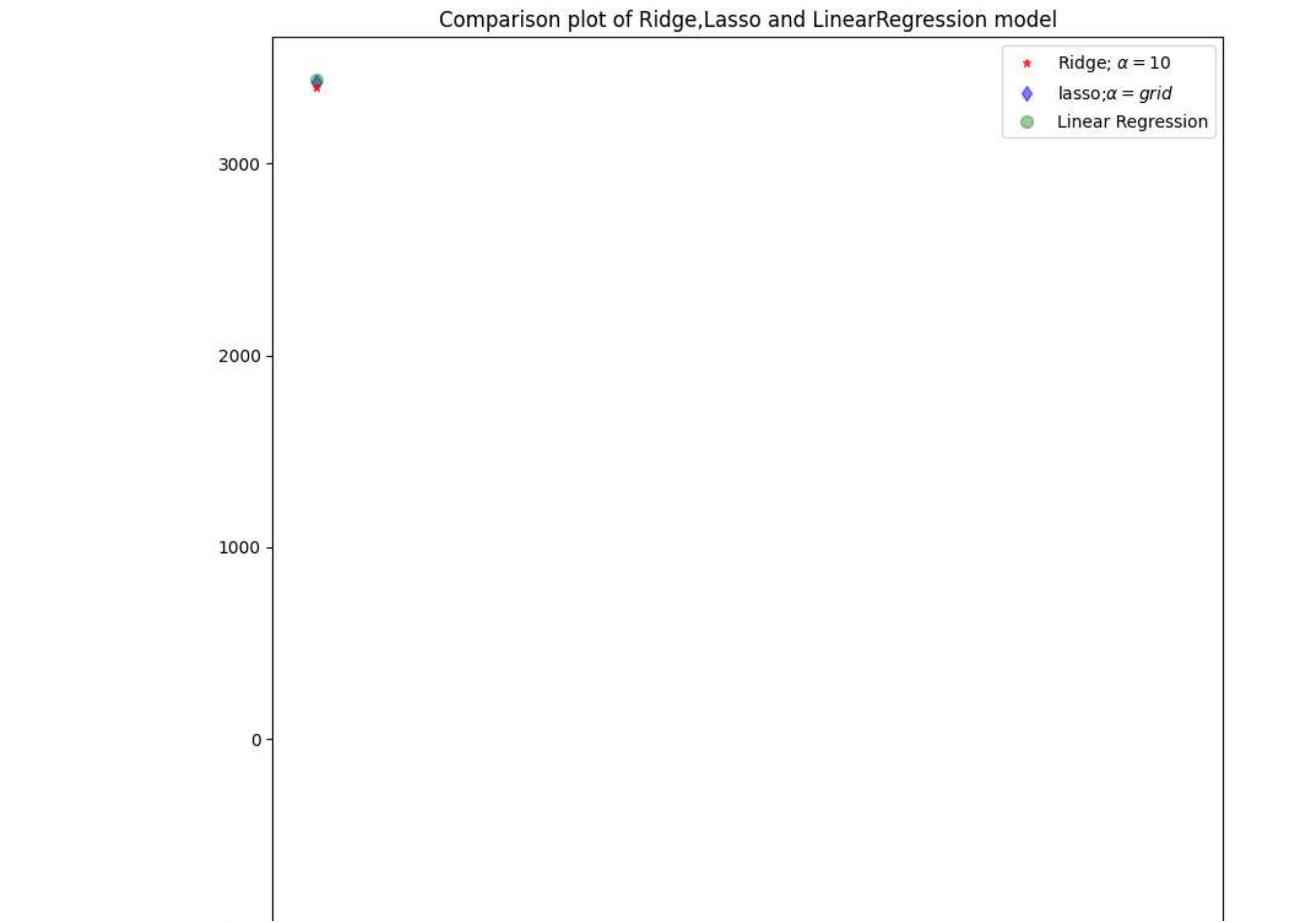
```
Lasso model:

The train score for ls model is 0.08144600503720845
The test score for ls model is 0.10226046691368151
```

In [50]:
```
pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "bar")
```

Out[50]: <Axes: >



In [51]:
```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*', markersize=5, color='red',label=r'Ridg
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r'lasso;$\alpha=
plt.plot(features, lr.coef_, alpha=0.4, linestyle='none', marker='o', markersize=7, color='green',label='Linea
plt.xticks(rotation=90)
plt.legend()
plt.title('Comparison plot of Ridge,Lasso and LinearRegression model')
plt.show()
```


Comparison plot of Ridge,Lasso and LinearRegression model
Ridge; $\alpha = 10$
lasso;$\alpha = grid$
Linear Regression
3000
2000
1000
0

ElasticNet

In [52]:
```
from sklearn.linear_model import ElasticNet
regr=ElasticNet()
regr.fit(x_train,y_train)
print(regr.coef_)
print(regr.intercept_)
```

```
[2283.69089962 -676.93095222]
13823.746181356835
```

In [53]:
```
y_pred_elastic=regr.predict(x_train)
```

In [54]:
```
mean_squared_error=ny.mean((y_pred_elastic- y_train)**2)
print("mean _squared_error",mean_squared_error)
```

```
mean _squared_error 144829222.93161225
```

logistic Regression

In [55]:
```
import re
from sklearn.linear_model import LogisticRegression
```

In [56]:
```
features_matrix=df.iloc[:,0:34]
```

In [57]:
```
target_vector=df.iloc[:,-1]
```

In [58]:
```
print('The features matrix has %d rows and %d column(s)'%(features_matrix.shape))
print('The target matrix has %d rows and %d column(s)'%(ny.array(target_vector).reshape(-1,1).shape))
```

```
The features matrix has 1338 rows and 7 column(s)
The target matrix has 1338 rows and 1 column(s)
```

In [59]:
```
import re
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
%matplotlib inline
digits=load_digits()
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(digits.data,digits.target,test_size=0.30,random_state=2)
print(x_train.shape)
```

```
(1257, 64)
```

In [60]:
```
import re
from sklearn.linear_model import LogisticRegression
x=np.array(df['age']).reshape(-1,1)
y=np.array(df['children']).reshape(-1,1)
df.dropna(inplace=True)
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=1)
LogisticRegr=LogisticRegression(max_iter=10000)
```

In [61]:
```
LogisticRegr.fit(x_train,y_train)
```

```
C:\Users\USER\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 the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[61]:
```
▾ LogisticRegression
LogisticRegression(max_iter=10000)
```

In [62]:
```

LogisticRegr=LogisticRegression(max_iter=10000)
LogisticRegr.fit(x_train,y_train)
print(LogisticRegr.predict(x_test))
```

```
C:\Users\USER\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 the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

In [63]:
```
score=LogisticRegr.score(x_test,y_test)
print(score)
```

```
0.43532338308457713
```

RandomForest

In [64]:
```
x=df.drop('sex',axis=1)
y=df['sex']
```

In [65]:
```
df['smoker'].value_counts()
```

Out[65]:
```
smoker
2    1064
1     274
Name: count, dtype: int64
```

In [66]:
```
r={'sex':{'female':1,'male':2}}
df=df.replace(r)
df
```

Out[66]:

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

1338 rows × 7 columns

In [67]:
```
s={'smoker':{'yes':1,'no':2}}
df=df.replace(s)
df
```

Out[67]:

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

1338 rows × 7 columns

In [68]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.7,random_state=42)
x_train.shape,x_test.shape
```

Out[68]:
```
((401, 6), (937, 6))
```

In [69]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[69]:
```
RandomForestClassifier
RandomForestClassifier()
```

In [70]:
```
rf=RandomForestClassifier()
```

In [71]:
```
params={'max_depth':[2,3,5,10,20],
       'min_samples_leaf':[5,10,20,50,100,200],
       'n_estimators':[10,25,30,50,100,200]}
```

In [79]:
```
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[79]:
```
GridSearchCV
estimator: RandomForestClassifier
RandomForestClassifier
```

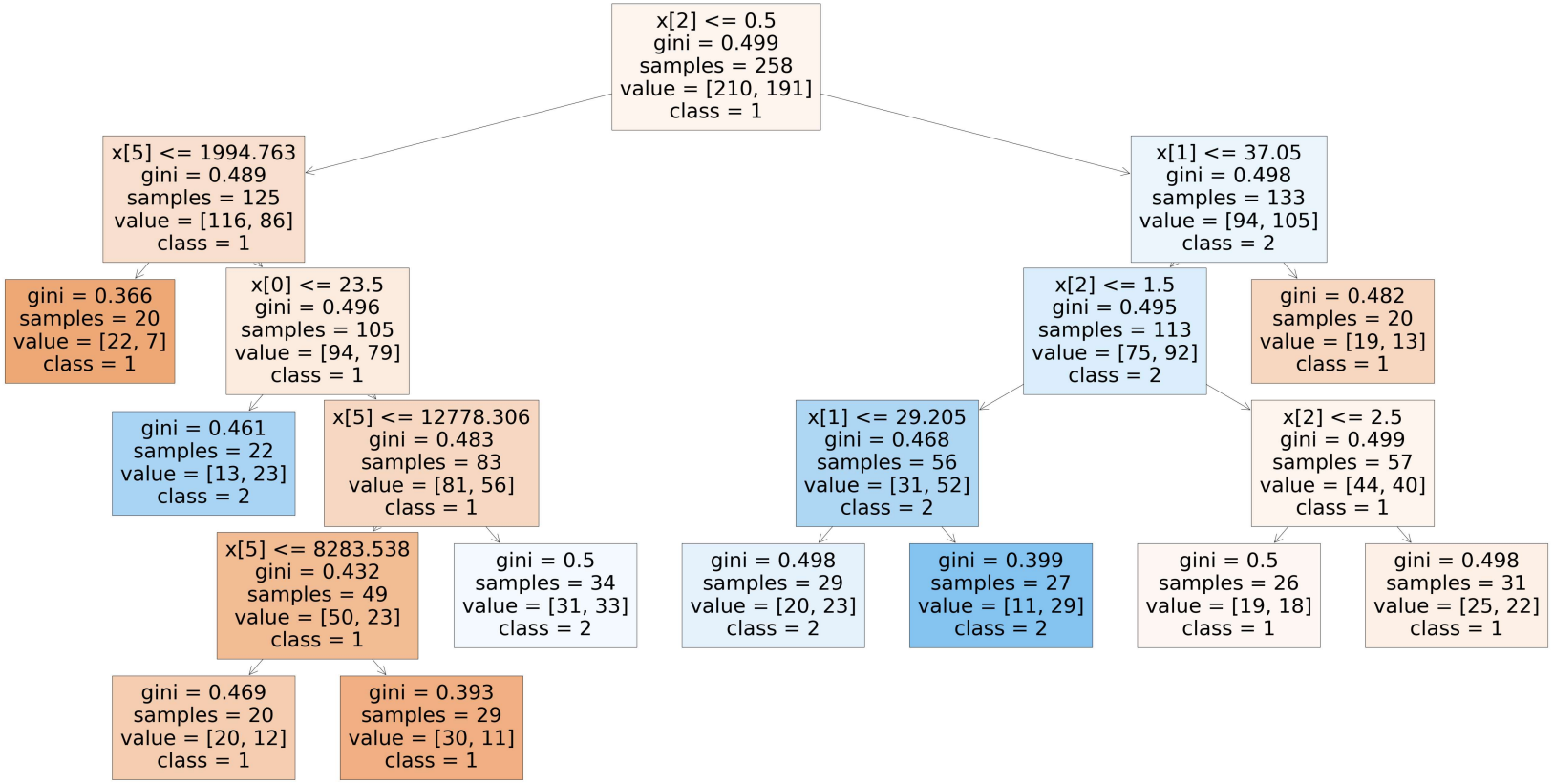
In [80]:
```
grid_search.best_score_
```

Out[80]:
```
0.5162189054726368
```

In [81]:
```
rf_best=grid_search.best_estimator_
print(rf_best)
```

```
RandomForestClassifier(max_depth=10, min_samples_leaf=20, n_estimators=30)
```

In [86]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[5],class_names=['1','2','3'],filled=True);
```



In [87]:
```
rf_best.feature_importances_
```

Out[87]:
```
array([0.25147257, 0.24355536, 0.07936152, 0.00832334, 0.07388058,
       0.34340663])
```

In [84]:
```
imp_df=pd.DataFrame({"varname":x_train.columns,"Imp":rf_best.feature_importances_})
imp_df.sort_values(by="Imp",ascending=False)
```

Out[84]:

	varname	Imp
5	charges	0.343407
0	age	0.251473
1	bmi	0.243555
2	children	0.079362
4	region	0.073881
3	smoker	0.008323

In [85]:
```
score=rfc.score(x_test,y_test)
print(rfc)
```

```
RandomForestClassifier()
```

CONCLUSION

Based on accuracy scores of all models that were implemented we can conclude that "Logistic Regression" is the best model for the given data set