MINI PROJECT

PROBLEM STATEMENT:which model is suitable for Insurance Dataset

Importing Packages

In [1]:

```
import numpy as np
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\arshiha\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.isnull().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=np.array(df["age"]).reshape(-1,1)
```

In [12]:

```
y=np.array(df["children"]).reshape(-1,1)
```

In [13]:

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

Linear Regression

In [14]:

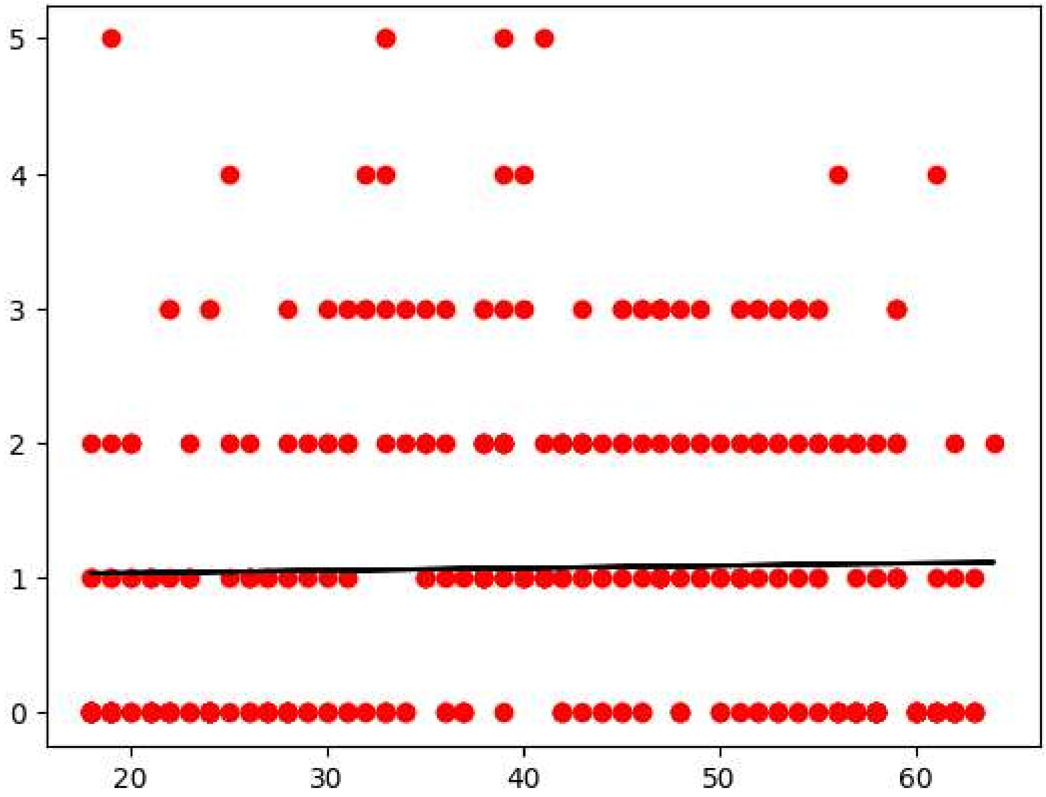
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
```

In [15]:

```
from sklearn.linear_model import LinearRegression
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

```
-0.0017866536487798346
```

In [16]:
```
from sklearn import preprocessing,svm
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color="r")
plt.plot(x_test,y_pred,color="k")
plt.show()
```

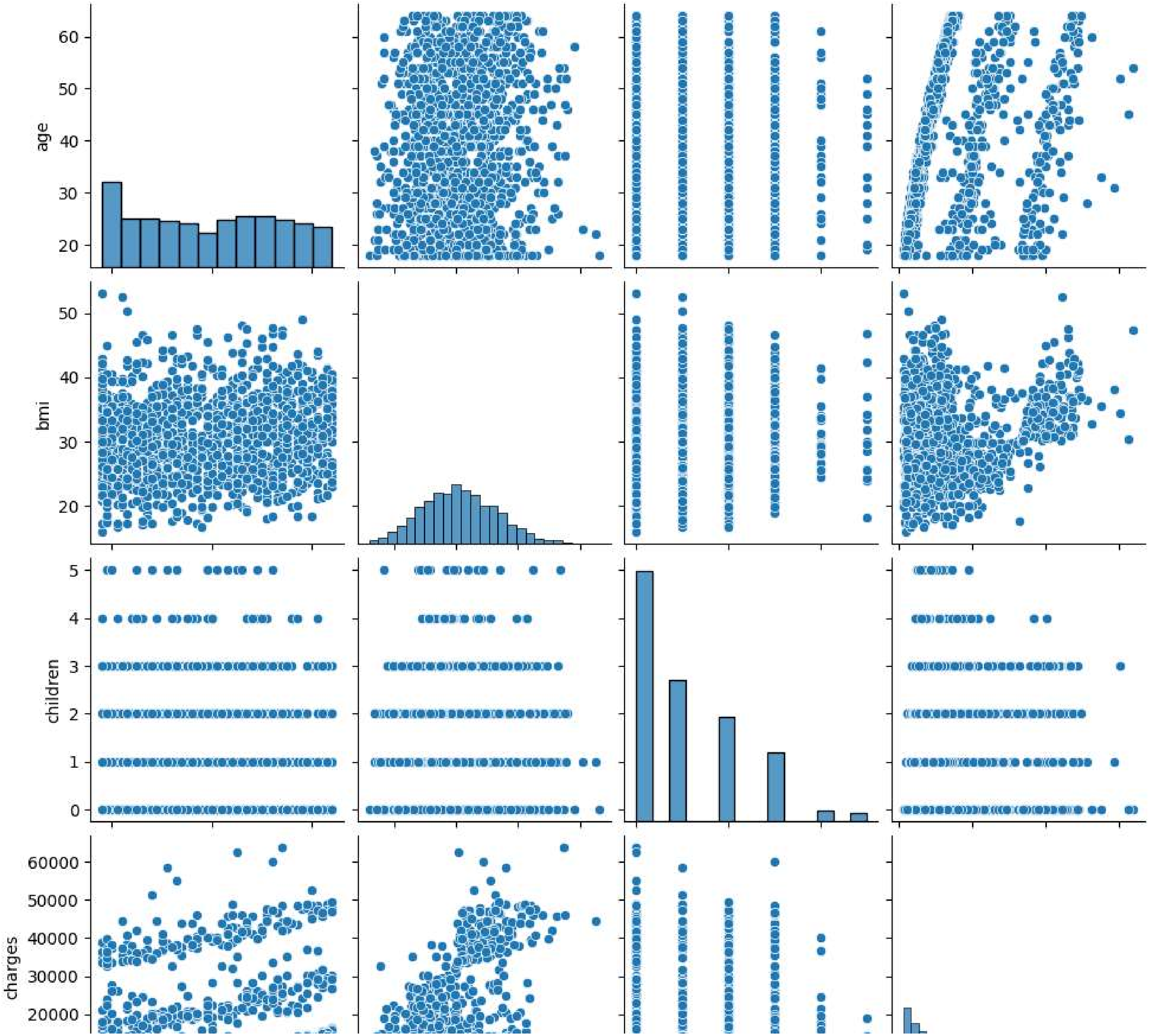


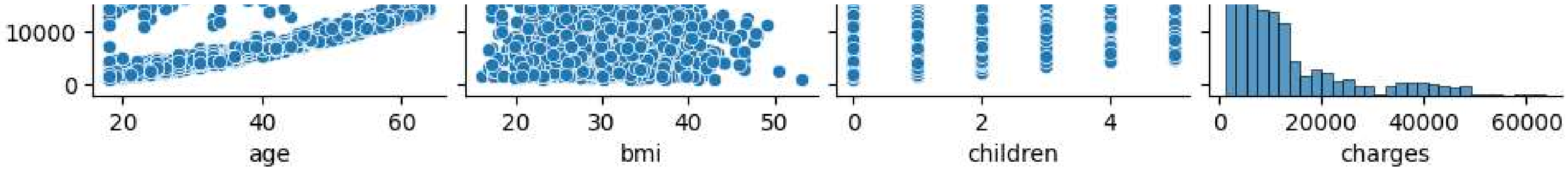
In [17]:

```
sns.pairplot(df)
```

Out[17]:

```
<seaborn.axisgrid.PairGrid at 0x212fab326d0>
```


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

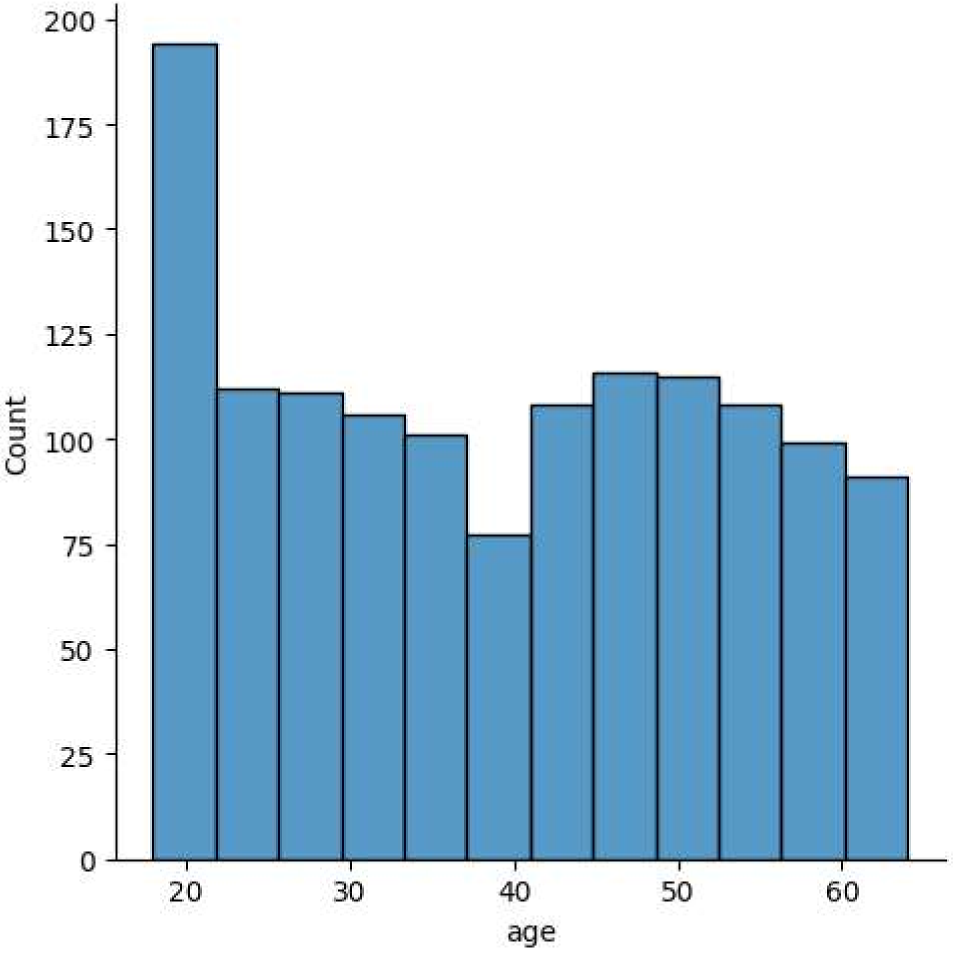


In [18]:

```
sns.displot(df['age'])
```

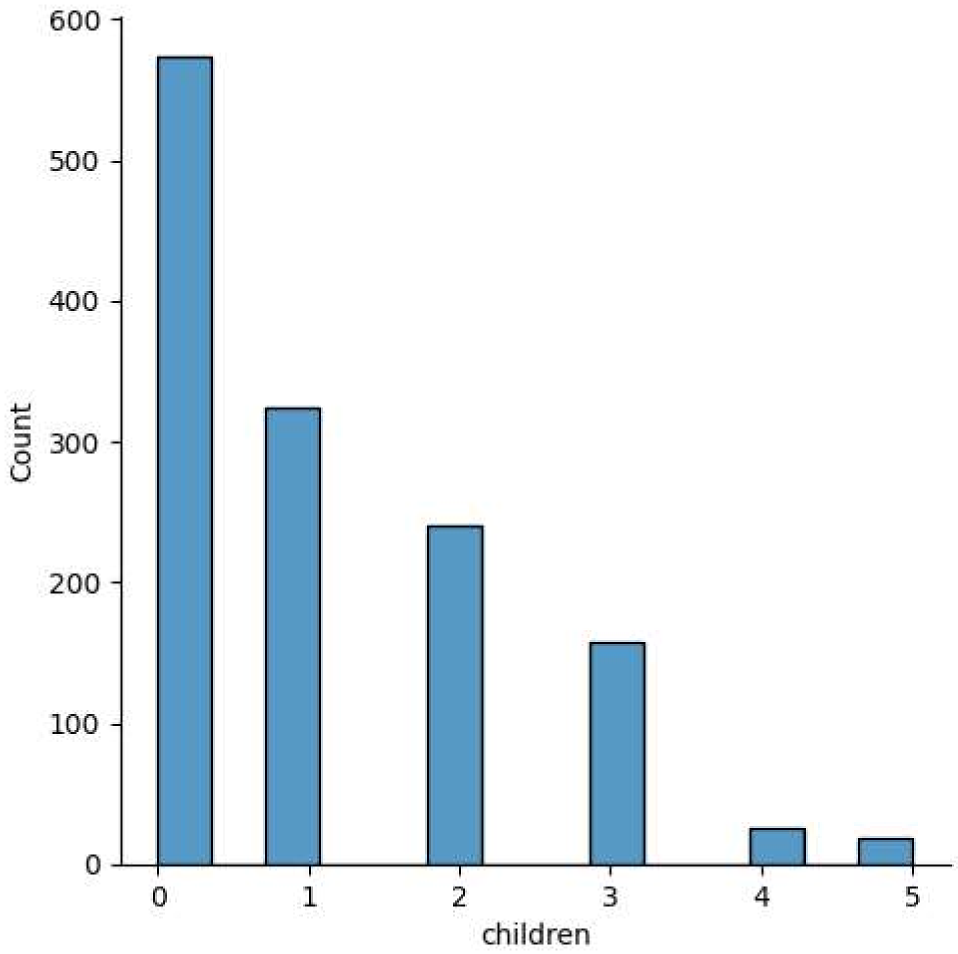
Out[18]:

```
<seaborn.axisgrid.FacetGrid at 0x212ffc64250>
```



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

Out[19]:
```
<seaborn.axisgrid.FacetGrid at 0x212ff639b50>
```



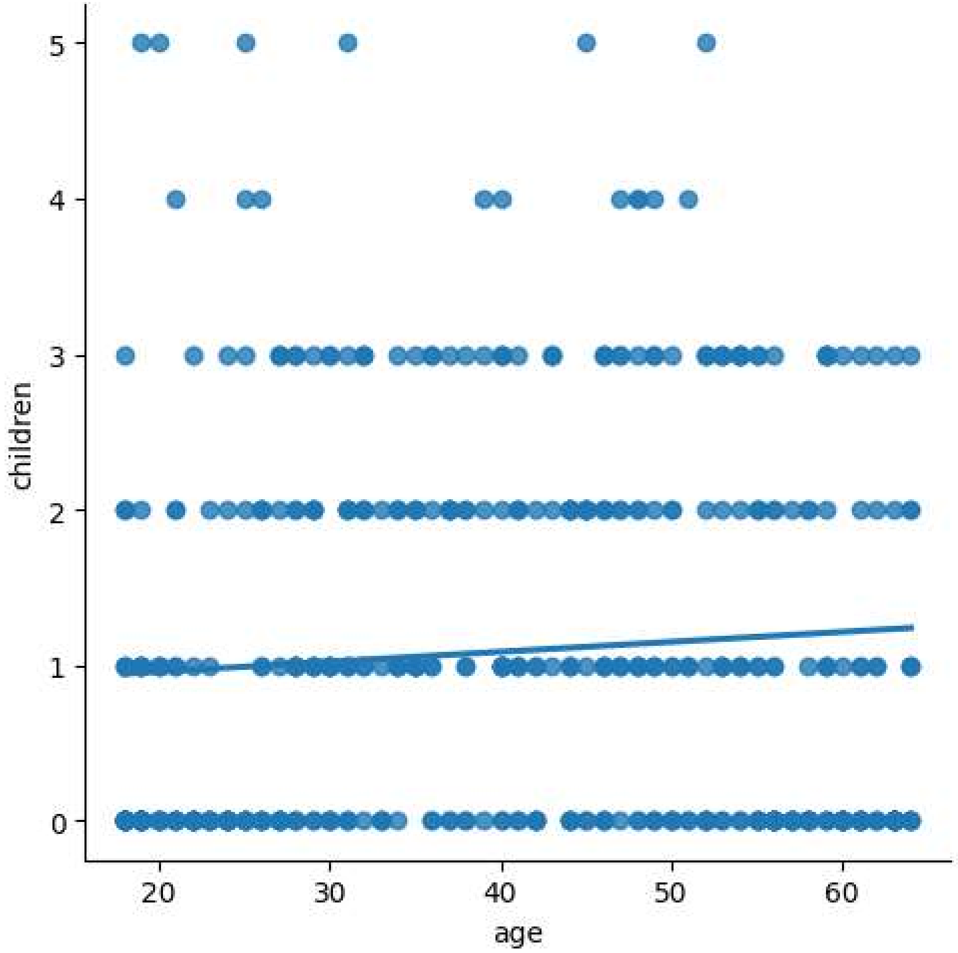
In [20]:
```
plt.figure(figsize=(15,8))
```

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)
```

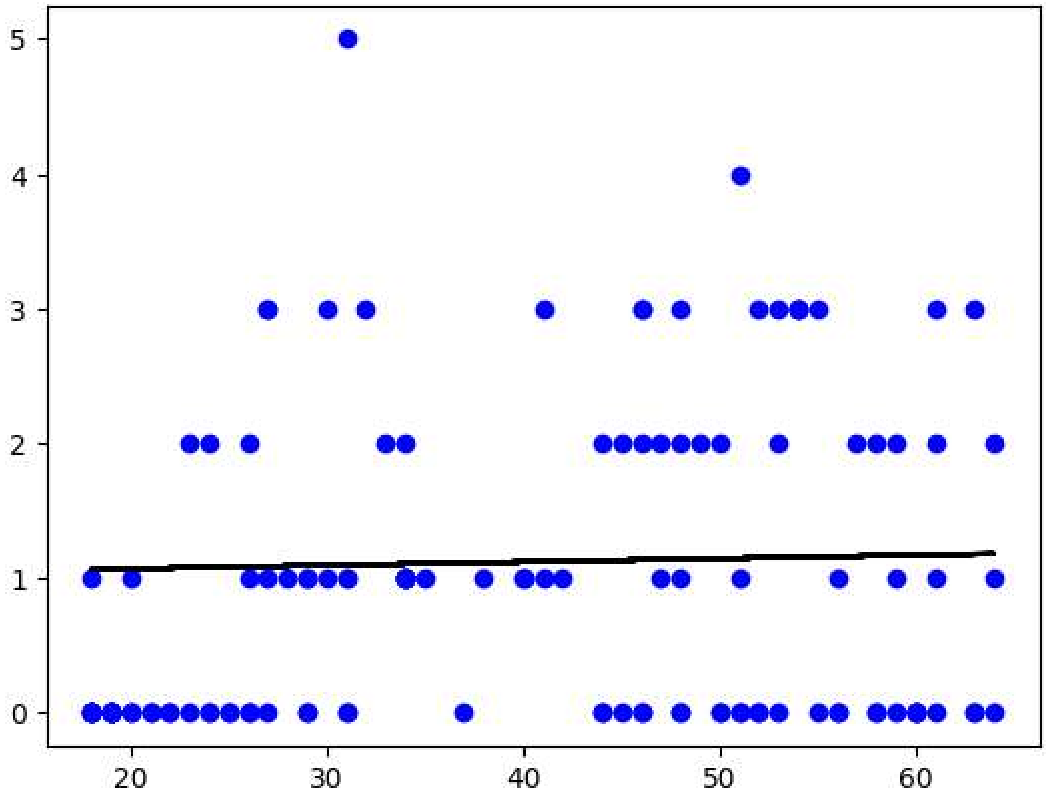
```
Out[21]: <seaborn.axisgrid.FacetGrid at 0x212ffc380d0>
```



In [22]:

```
df500.fillna(method='ffill',inplace=True)
x=np.array(df500['age']).reshape(-1,1)
y=np.array(df500['children']).reshape(-1,1)
df500.dropna(inplace=True)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
regr=LinearRegression()
regr.fit(x_train,y_train)
print("Regression:",regr.score(x_test,y_test))
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```

```
Regression: -0.0042836432283639425
```



In [23]:

```
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("r2 score:",r2)
```

```
r2 score: -0.0042836432283639425
```

Decision Tree

In [24]:

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

In [25]:

```
from sklearn.tree import DecisionTreeClassifier
```

In [26]:

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

Out[26]:

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

In [27]:

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

Out[27]:

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

In [28]:
```
df['region'].value_counts()
```

Out[28]:
```
region
southeast    364
southwest    325
northwest    325
northeast    324
Name: count, dtype: int64
```

In [29]:
```
convert={"sex":{"female":1,"male":2}}
df=df.replace(convert)
df
```

Out[29]:

	age	sex	bmi	children	smoker	region	charges
0	19	1	27.900	0	yes	southwest	16884.92400
1	18	2	33.770	1	no	southeast	1725.55230
2	28	2	33.000	3	no	southeast	4449.46200
3	33	2	22.705	0	no	northwest	21984.47061
4	32	2	28.880	0	no	northwest	3866.85520
...	...	...	...	...	...	...	...
1333	50	2	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 [30]:

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

Out[30]:

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

1338 rows × 7 columns
```

In [31]:

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

Out[31]:

	age	sex	bmi	children	smoker	region	charges
0	19	1	27.900	0	1	1	16884.92400
1	18	2	33.770	1	0	2	1725.55230
2	28	2	33.000	3	0	2	4449.46200
3	33	2	22.705	0	0	3	21984.47061
4	32	2	28.880	0	0	3	3866.85520
...	...	...	...	...	...	...	...
1333	50	2	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 [32]:

```
x=["age","bmi","children"]
y=[1,2]
all_inputs=df[x]
all_classes=df["sex"]
```

In [33]:

```
(x_train,x_test,y_train,y_test)=train_test_split(all_inputs,all_classes,test_size=0.25)
```

In [34]:

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

In [35]:

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

Out[35]:

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

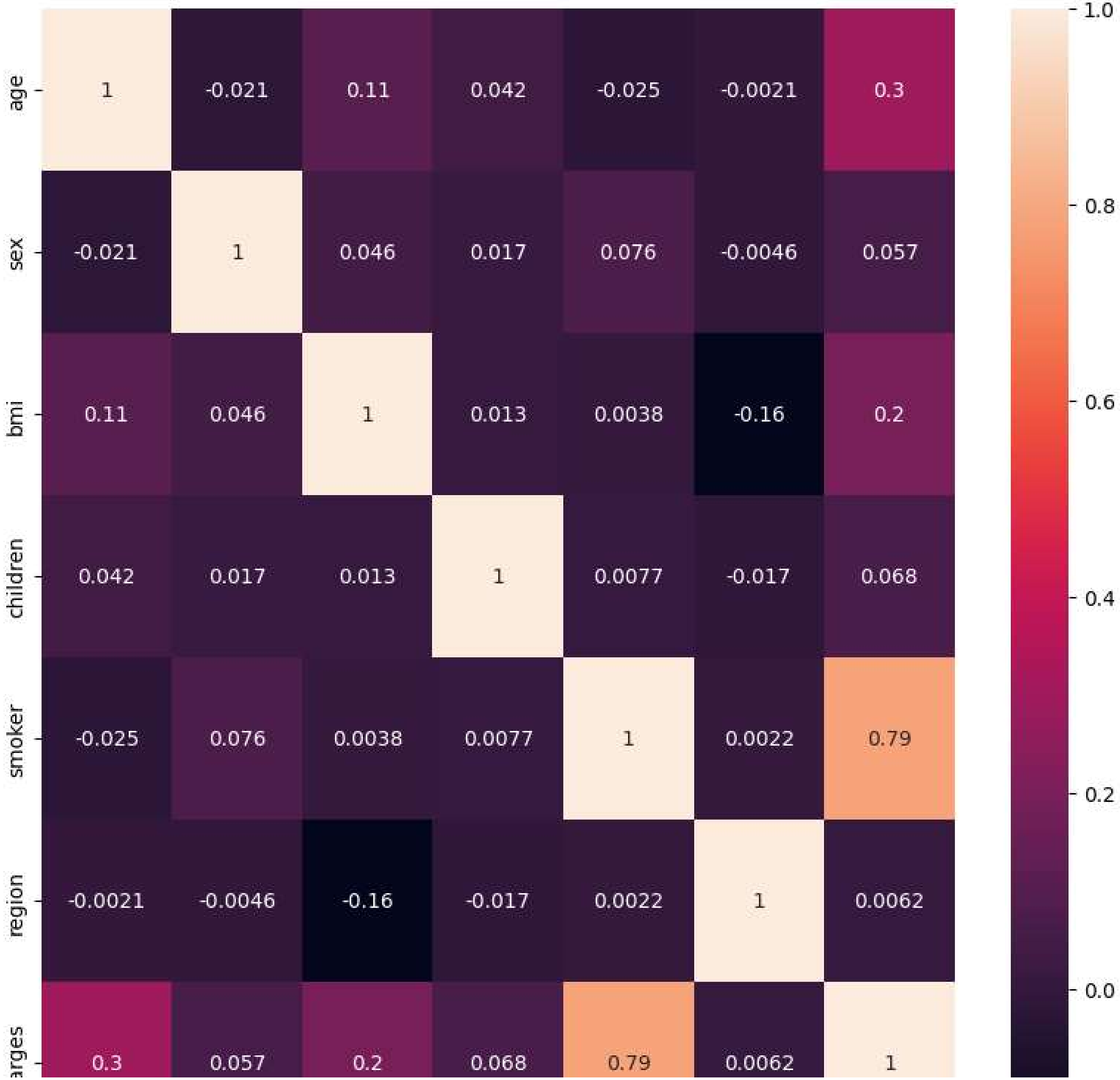
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [36]: clf.score(x_test,y_test)

Out[36]: 0.4835820895522388
```

Data Visualization

In [37]:
```
plt.figure(figsize=(10,10))
sns.heatmap(df.corr(),annot=True)
```

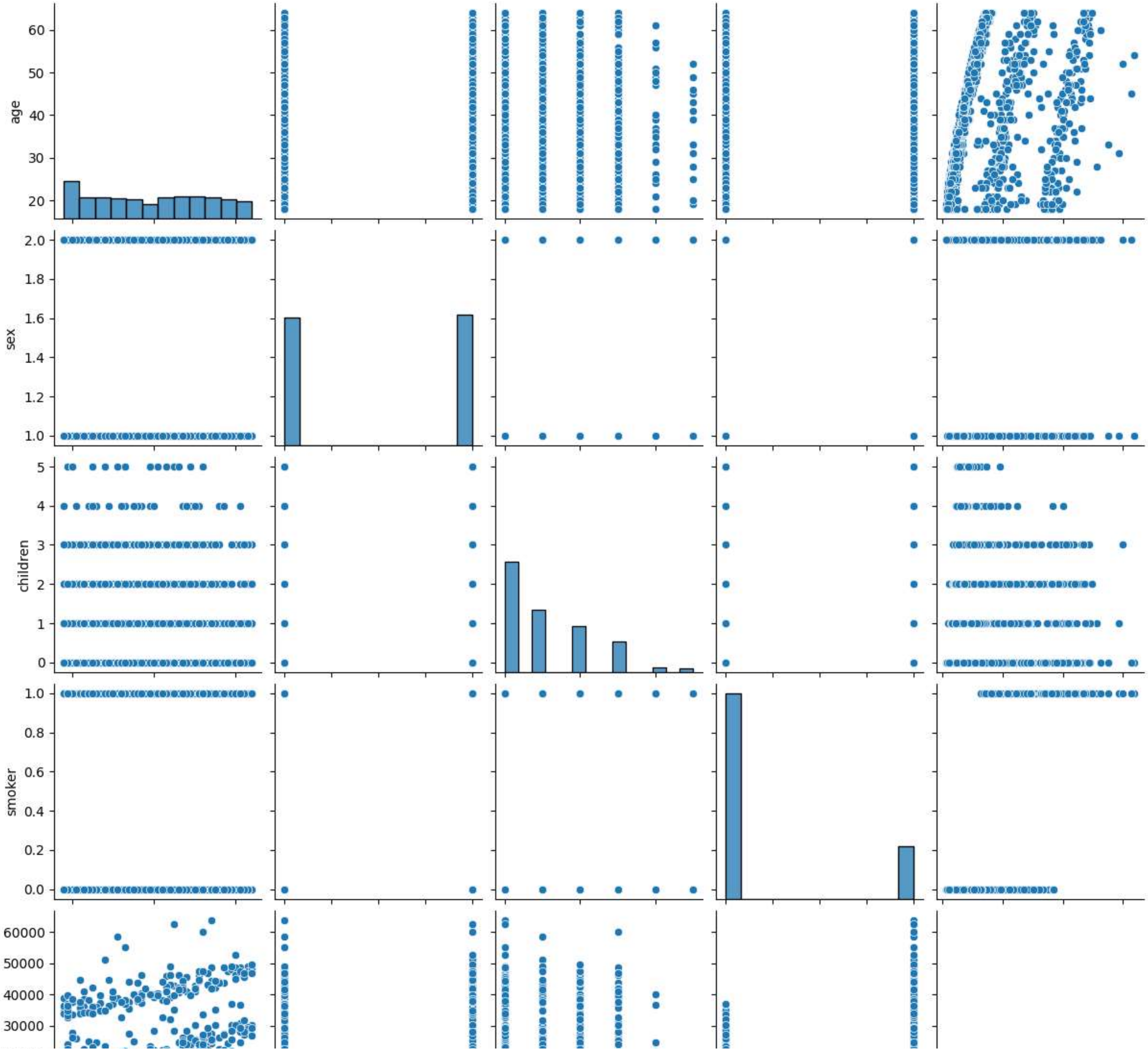
Out[37]:
```
<Axes: >
```

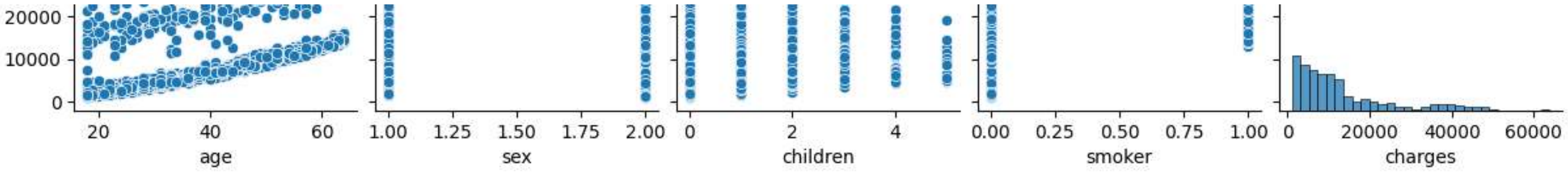

age
1
-0.021
0.11
0.042
-0.025
-0.0021
0.3
sex
-0.021
1
0.046
0.017
0.076
-0.0046
0.057
bmi
0.11
0.046
1
0.013
0.0038
-0.16
0.2
children
0.042
0.017
0.013
1
0.0077
-0.017
0.068
smoker
-0.025
0.076
0.0038
0.0077
1
0.0022
0.79
region
-0.0021
-0.0046
-0.16
-0.017
0.0022
1
0.0062
arges
0.3
0.057
0.2
0.068
0.79
0.0062
1
1.0
0.8
0.6
0.4
0.2
0.0

cha
age
sex
bmi
children
smoker
region
charges

In [38]:
```
df.drop(columns=["region","bmi"],inplace=True)
sns.pairplot(df)
df.smoker=np.log(df.smoker)
```

```
C:\Users\arshiha\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\arraylike.py:396: Ru
ntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```


60
50
age
40
30
20
2.0
1.8
sex
1.6
1.4
1.2
1.0
5
4
children
3
2
1
0
1.0
0.8
smoker
0.6
0.4
0.2
0.0
60000
50000
40000
charges
30000



In [39]:

```
features=df.columns[0:1]
target=df.columns[-1]
x=df[features].values
y=df[target].values
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))
Scaler=StandardScaler()
x_train=Scaler.fit_transform(x_train)
x_test=Scaler.transform(x_test)
```

```
The dimension of x_train is (936, 1)
The dimension of x_test is (402, 1)
```

In [40]:

```
lr = LinearRegression()
lr.fit(x_train,y_train)
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))
```

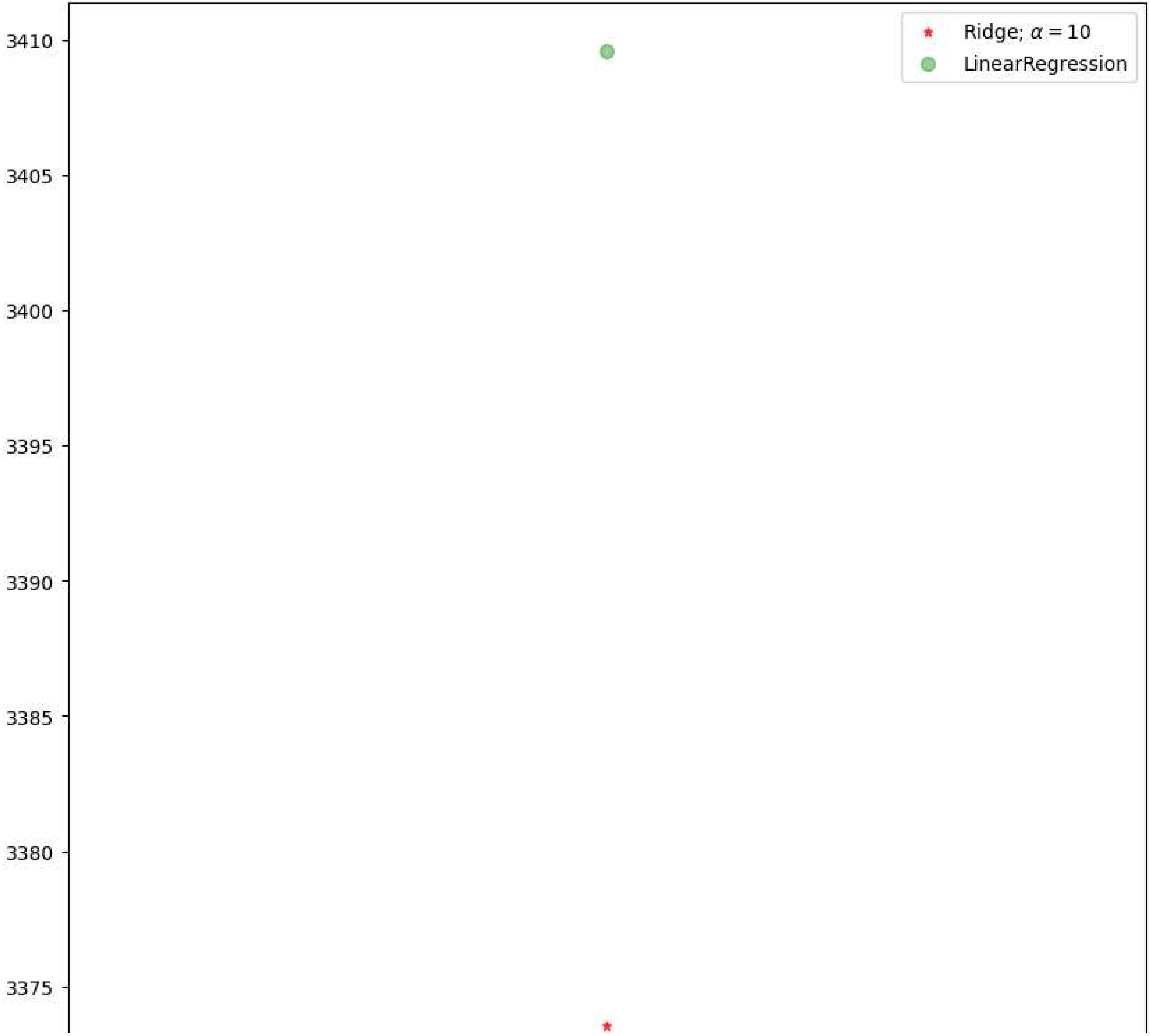
```

Linear Regression Model:

The train score for lr model is 0.07447061146193878
The test score for lr model is 0.10891203216512224
```

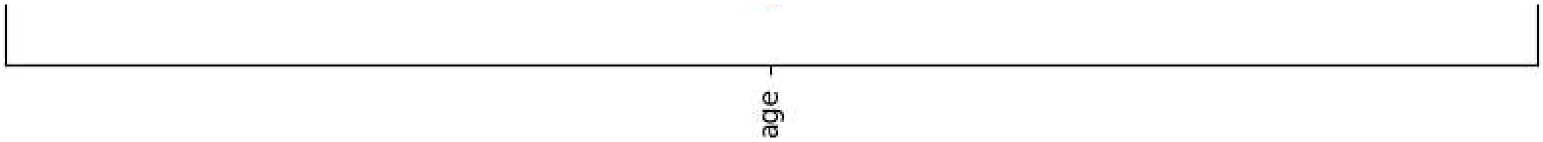
In [41]:
```
ridgeReg = Ridge(alpha=10)
ridgeReg.fit(x_train,y_train)
train_score_ridge = ridgeReg.score(x_train, y_train)
test_score_ridge = ridgeReg.score(x_test, y_test)
print("\nRidge Model:\n")
print("The train score for ridge model is {}".format(train_score_ridge))
print("The test score for ridge model is {}".format(test_score_ridge))
```

```

Ridge Model:

The train score for ridge model is 0.07446228994221393
The test score for ridge model is 0.10855133360950642
```

In [42]:

```
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(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='LinearRegr
plt.xticks(rotation=90)
plt.legend()
plt.show()
```


3410
3405
3400
3395
3390
3385
3380
3375
★ Ridge; $\alpha = 10$
● LinearRegression



Lasso Regression

In [43]:
```
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))
```

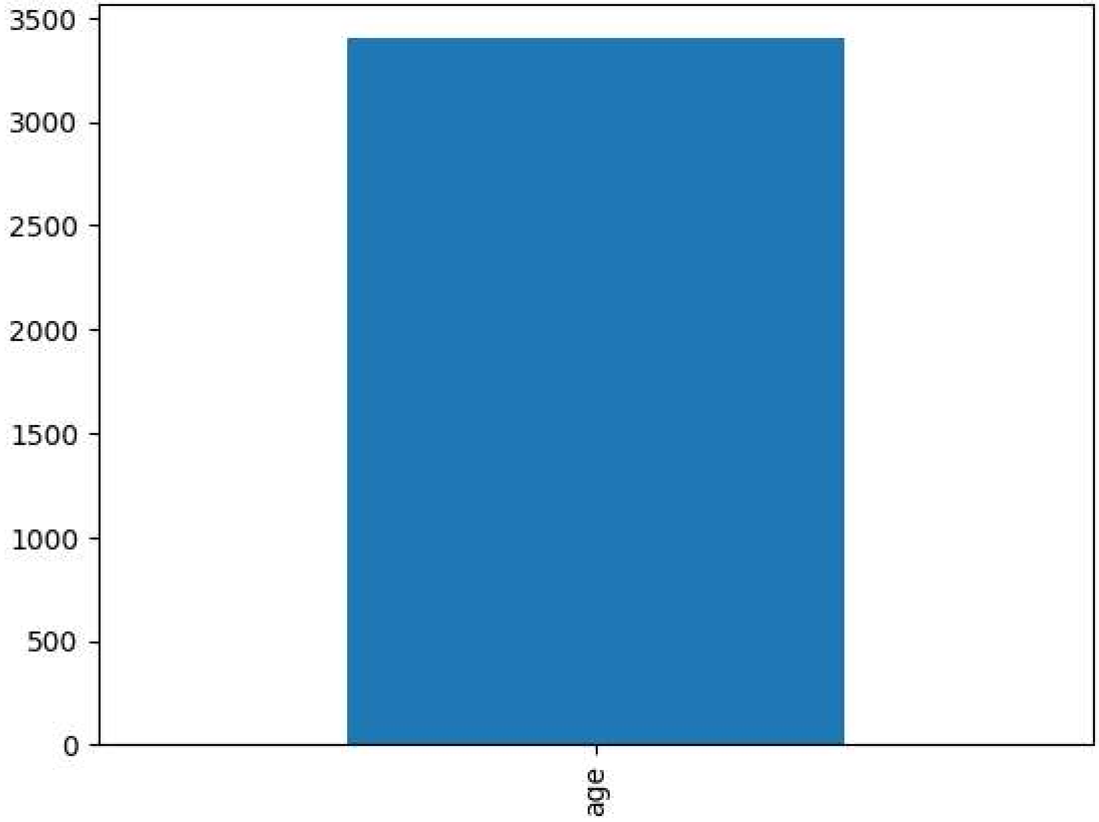
```
0.07446997086306062
0.10881427793326703
```

In [44]:
```
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 0.07446997086306062
The test score for ls model is 0.10881427793326703
```

In [45]:
```
pd.Series(lasso.coef_,features).sort_values(ascending = True).plot(kind = "bar")
```

Out[45]:
```
<Axes: >
```

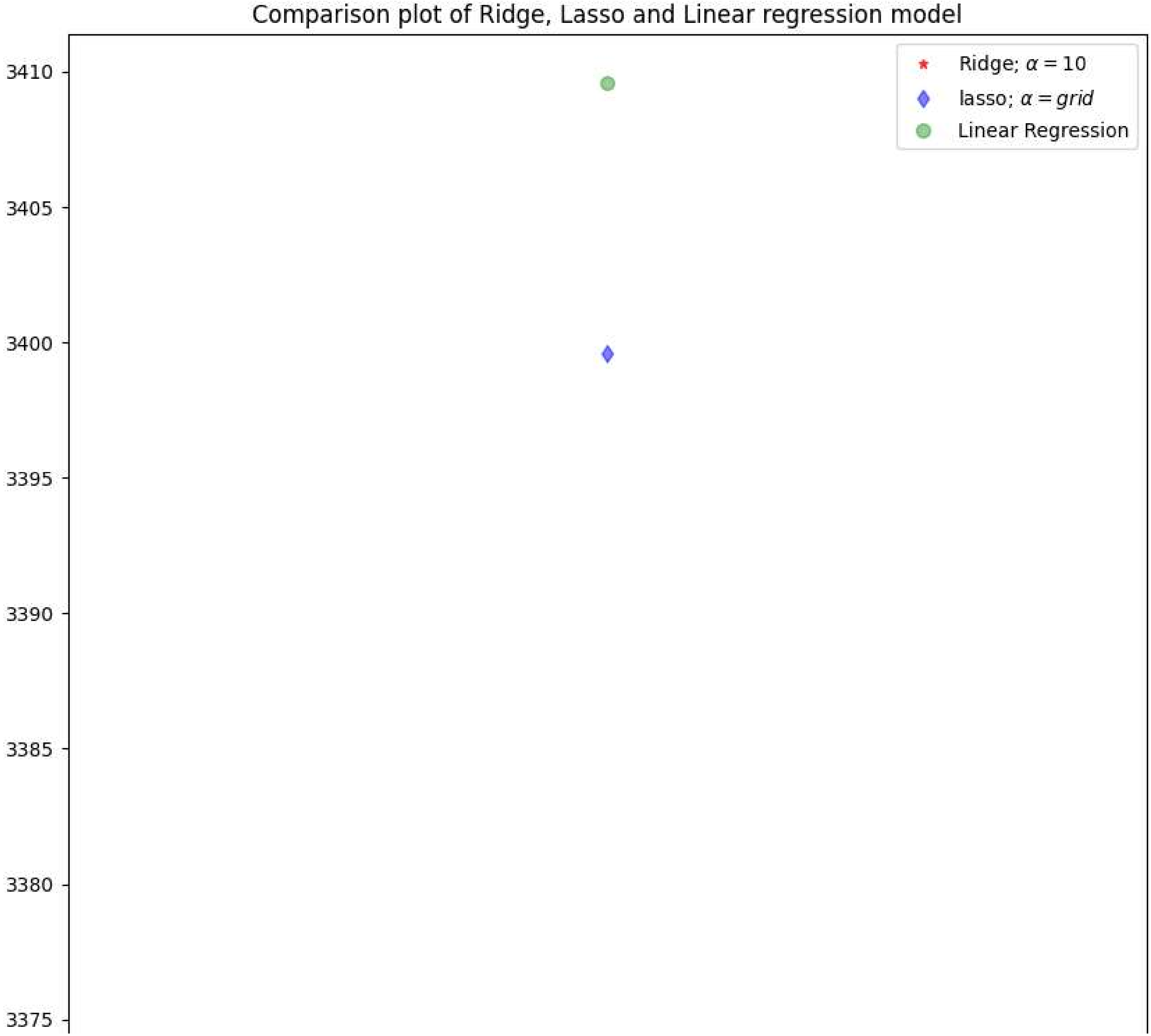


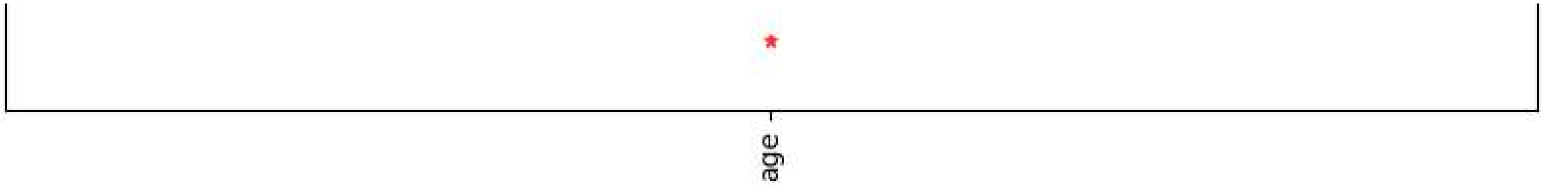
In [46]:
```
from sklearn.linear_model import LassoCV
lasso_cv=LassoCV(alphas = [0.0001, 0.001, 0.01, 0.1, 1, 10],random_state=0).fit(x_train,y_train)
print(lasso_cv.score(x_train,y_train))
print(lasso_cv.score(x_test,y_test))
```

```
0.07446997086306062
0.10881427793326703
```

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(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r'lasso; $\alph
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear Reg
plt.xticks(rotation = 90)
plt.legend()
plt.title("Comparison plot of Ridge, Lasso and Linear regression model")
plt.show()
```


Comparison plot of Ridge, Lasso and Linear regression model
★ Ridge; $\alpha = 10$
◆ lasso; $\alpha = grid$
● Linear Regression
3410
3405
3400
3395
3390
3385
3380
3375



```
In [48]: from sklearn.linear_model import RidgeCV
         ridge_cv = RidgeCV(alphas = [0.0001, 0.001,0.01, 0.1, 1, 10]).fit(x_train,y_train)
         print("The train score for ridge model is {}".format(ridge_cv.score(x_train,y_train)))
         print("The test score for ridge model is {}".format(ridge_cv.score(x_test,y_test)))
```

```
The train score for ridge model is 0.07446228994221393
The test score for ridge model is 0.10855133360950775
```

Elastic Net

```
In [49]: from sklearn.linear_model import ElasticNet
         regr = ElasticNet()
         regr.fit(x,y)
         print(regr.coef_)
         print(regr.intercept_)
```

```
[257.0684655]
3191.532406056682
```

```
In [50]: y_pred_elastic = regr.predict(x_train)
```

```
In [51]: mean_squared_error=np.mean((y_pred_elastic-y_train)**2)
         print("Mean Squared Error on test set",mean_squared_error)
```

```
Mean Squared Error on test set 267460995.25217086
```

In [52]:

```
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()
```

In [53]:

```
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=0.7,random_state=42)
```

In [54]:

```
from sklearn.linear_model import LogisticRegression
```

In [55]:

```
logisticRegr=LogisticRegression(max_iter=10000)
logisticRegr.fit(x_train,y_train)
```

Out[55]:

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

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [56]:

```
print(logisticRegr.predict(x_test))
```

```
[6 9 3 ... 1 0 8]
```

In [57]:

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

```
0.9467408585055644
```

Random Forest

In [58]:

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

Out[58]:

```
RandomForestClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [59]:

```
rf=RandomForestClassifier()
```

In [60]:

```
params={'max_depth':[2,3,4,5,6],
        'min_samples_leaf':[5,10,15,20,50,100],
        'n_estimators':[10,25,30,50,100,200]}
```

In [ ]:

```
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)
```

In [66]:

```
grid_search.best_score_
```

Out[66]:

```
0.9054041029877461
```

In [67]:

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

```
RandomForestClassifier(max_depth=6, min_samples_leaf=5)
```

In [68]:

```
x=df.drop("smoker",axis=1)
y=df["smoker"]
```

CONCLUSION

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