Mini Project on Insurance

1.Problem Statement:which model is suitable best fit model for Insurance

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import preprocessing,svm
from sklearn import metrics
from sklearn.linear_model import RidgeCV
from sklearn.linear_model import LassoCV
from sklearn.linear_model import ElasticNet
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
```

Data collection

Read the data

In [2]:

```
df=pd.read_csv(r"C:\Users\RAMADEVI SURIPAKA\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

2.Data cleaning and Preprocessing

In [3]:

```
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 [4]:

```
df.columns
```

Out[4]:

```
Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges'], dt
ype='object')
```

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]:

	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

To find Duplicate value

In [9]:

```
df.duplicated().sum()
```

Out[9]:

```
1
```

To find unique values

In [10]:

```
df['age'].unique()

df['children'].unique()

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

Out[10]:

```
array([27.9  , 33.77 , 33.   , 22.705, 28.88 , 25.74 , 33.44 , 27.74 ,
       29.83 , 25.84 , 26.22 , 26.29 , 34.4  , 39.82 , 42.13 , 24.6  ,
       30.78 , 23.845, 40.3  , 35.3  , 36.005, 32.4  , 34.1  , 31.92 ,
       28.025, 27.72 , 23.085, 32.775, 17.385, 36.3  , 35.6  , 26.315,
       28.6  , 28.31 , 36.4  , 20.425, 32.965, 20.8  , 36.67 , 39.9  ,
       26.6  , 36.63 , 21.78 , 30.8  , 37.05 , 37.3  , 38.665, 34.77 ,
       24.53 , 35.2  , 35.625, 33.63 , 28.   , 34.43 , 28.69 , 36.955,
       31.825, 31.68 , 22.88 , 37.335, 27.36 , 33.66 , 24.7  , 25.935,
       22.42 , 28.9  , 39.1  , 36.19 , 23.98 , 24.75 , 28.5  , 28.1  ,
       32.01 , 27.4  , 34.01 , 29.59 , 35.53 , 39.805, 26.885, 38.285,
       37.62 , 41.23 , 34.8  , 22.895, 31.16 , 27.2  , 26.98 , 39.49 ,
       24.795, 31.3  , 38.28 , 19.95 , 19.3  , 31.6  , 25.46 , 30.115,
       29.92 , 27.5  , 28.4  , 30.875, 27.94 , 35.09 , 29.7  , 35.72 ,
       32.205, 28.595, 49.06 , 27.17 , 23.37 , 37.1  , 23.75 , 28.975,
       31.35 , 33.915, 28.785, 28.3  , 37.4  , 17.765, 34.7  , 26.505,
       22.04 , 35.9  , 25.555, 28.05 , 25.175, 31.9  , 36.   , 32.49 ,
       25.3  , 29.735, 38.83 , 30.495, 37.73 , 37.43 , 24.13 , 37.145,
       39.52 , 24.42 , 27.83 , 36.85 , 39.6  , 29.8  , 29.64 , 28.215,
```

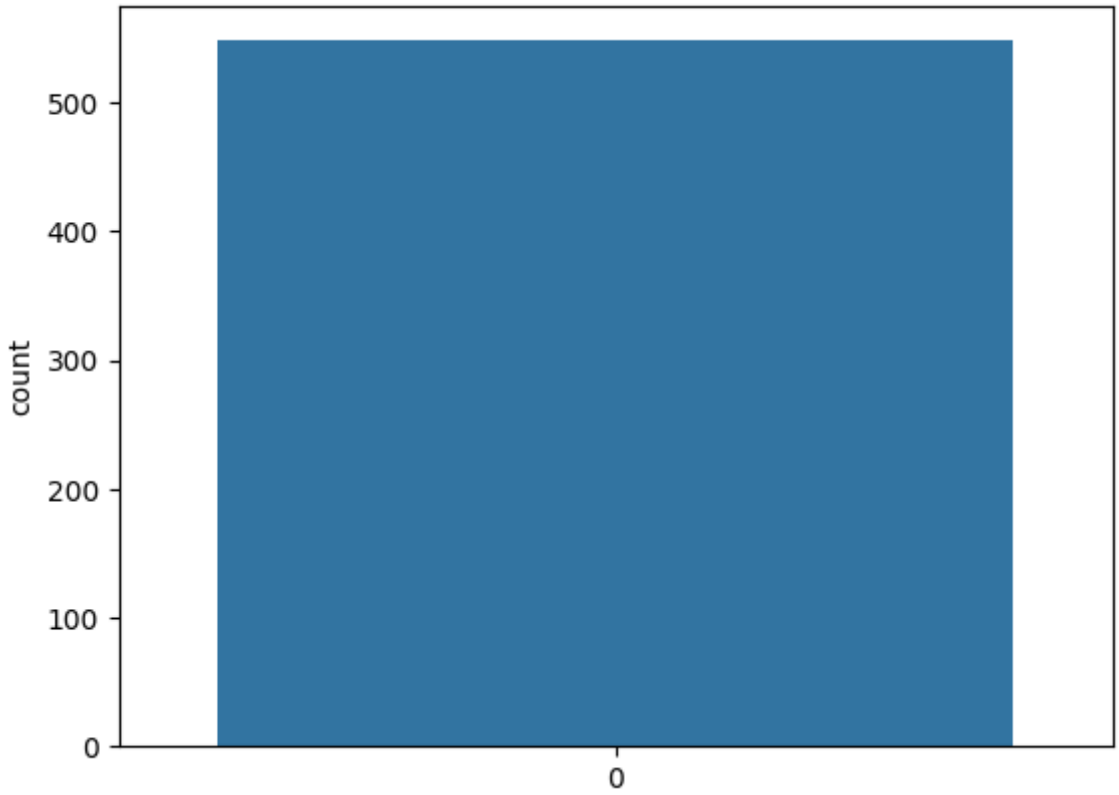
3.Data Visualization:Visualize the unique counts

In [11]:

```
sns.countplot(df['bmi'].unique())
```

Out[11]:

```
<Axes: ylabel='count'>
```



find null values

In [12]:

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

Out[12]:

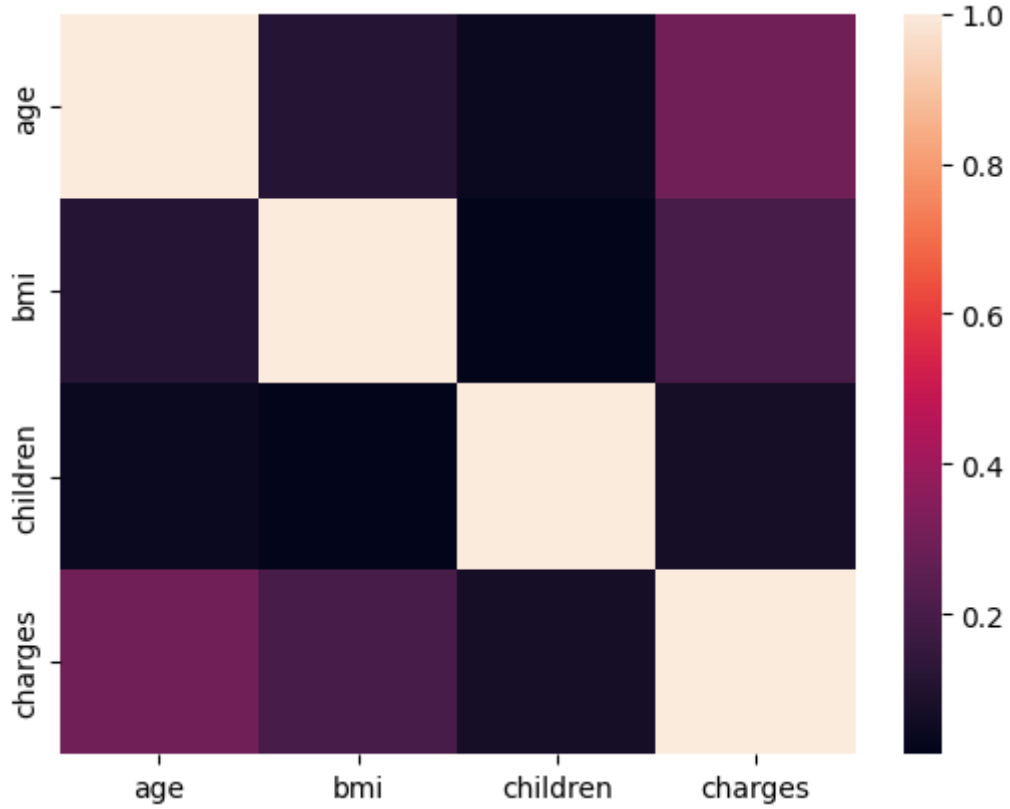
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

In [13]:

```
Insurancedf=df[['age','bmi', 'children','charges']]
sns.heatmap(Insurancedf.corr())
```

Out[13]:

```
<Axes: >
```



to check the null values

In [14]:

```
df.replace(np.nan,'0',inplace = True)
```

In [15]:

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

Out[15]:

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

In the above data thers is no null values by using the formulae:=isnull().sum()

Feature Scaling:To split the data into train and test data

In [16]:

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

In [17]:

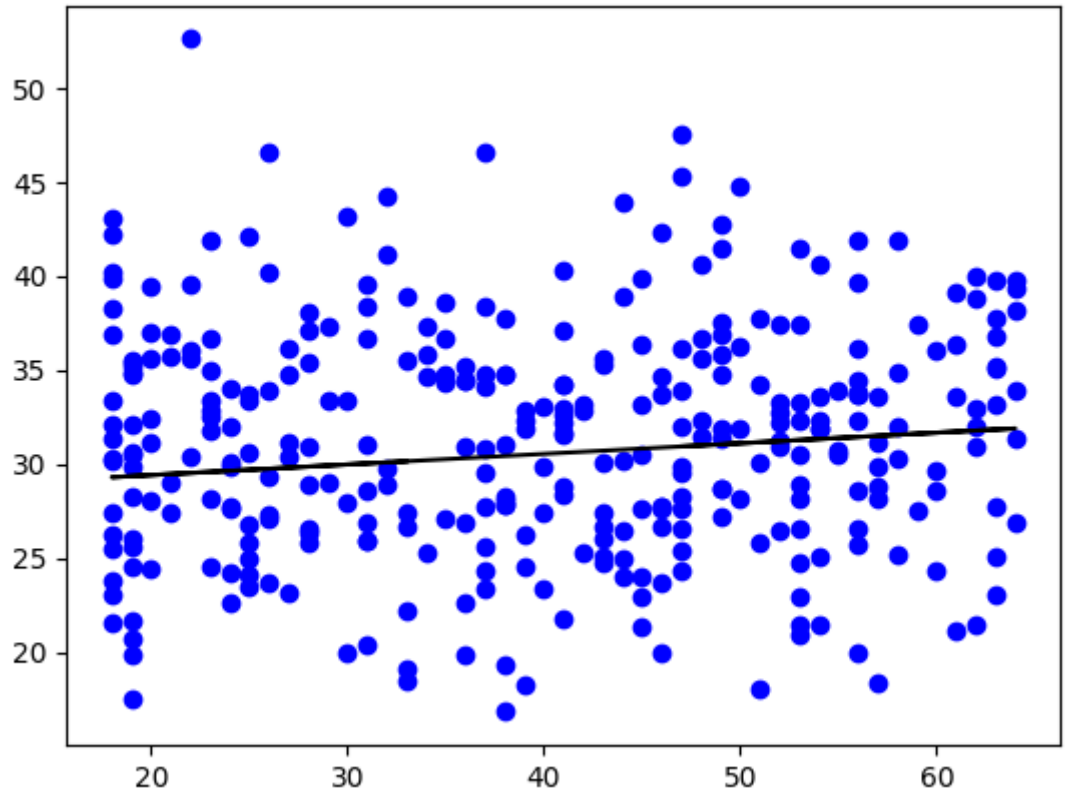
```
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
reg=LinearRegression()
reg.fit(X_train,y_train)
print(reg.score(X_test,y_test))
```

```
-0.01930783753776555
```

splitting of data,I get the value '0'.

In [18]:

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

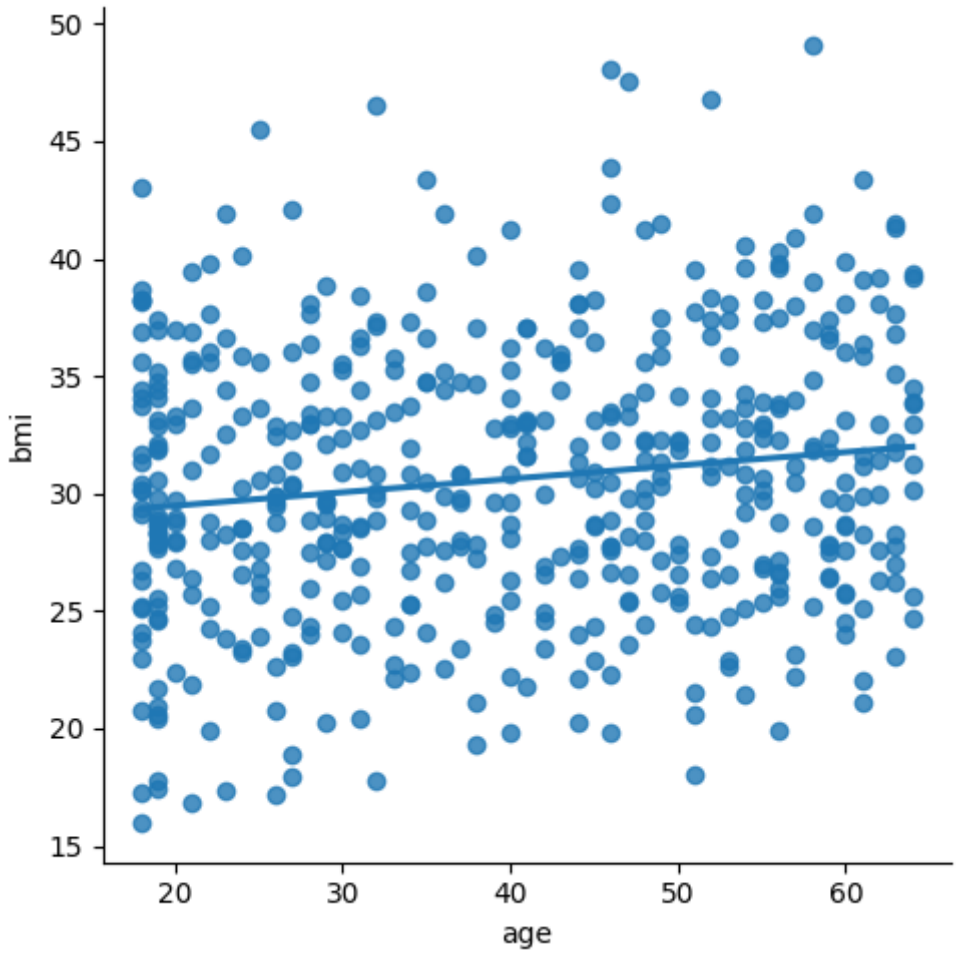


In [19]:

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

Out[19]:

```
<seaborn.axisgrid.FacetGrid at 0x2ab96812f80>
```

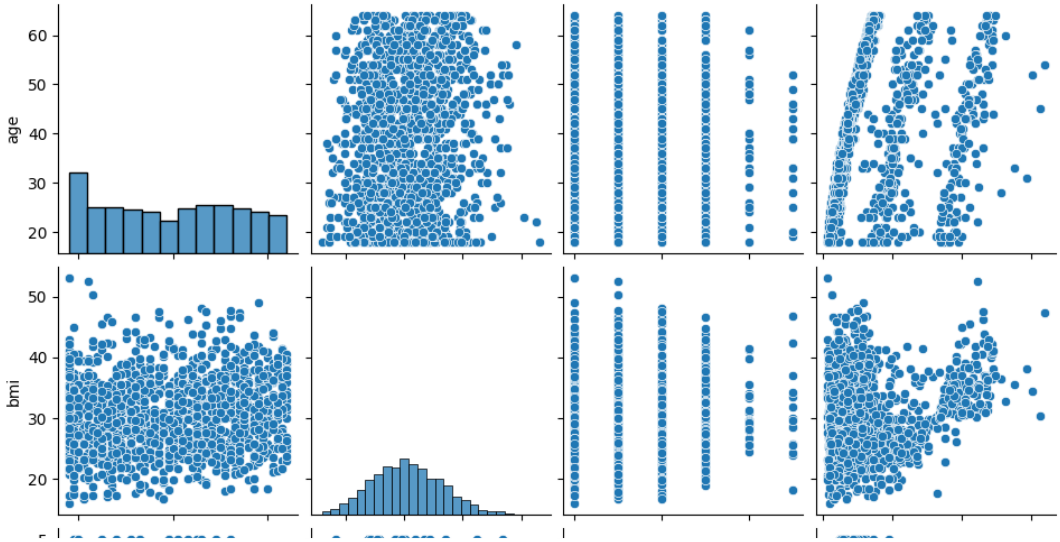


In [20]:

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

Out[20]:

```
<seaborn.axisgrid.PairGrid at 0x2ab9684de70>
```



In [ ]:

```

```

4.Data Modelling:Using Linear,Ridge and Lasso

In [21]:

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

In [22]:

```
T={"sex":{'male':1,'female':2}}
df=df.replace(T)
df
```

Out[22]:

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

1338 rows × 7 columns

In [ ]:

```

```

In [23]:

```
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 [24]:

```
#Model
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("\nRidg 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))
```

```
Ridg Model:

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

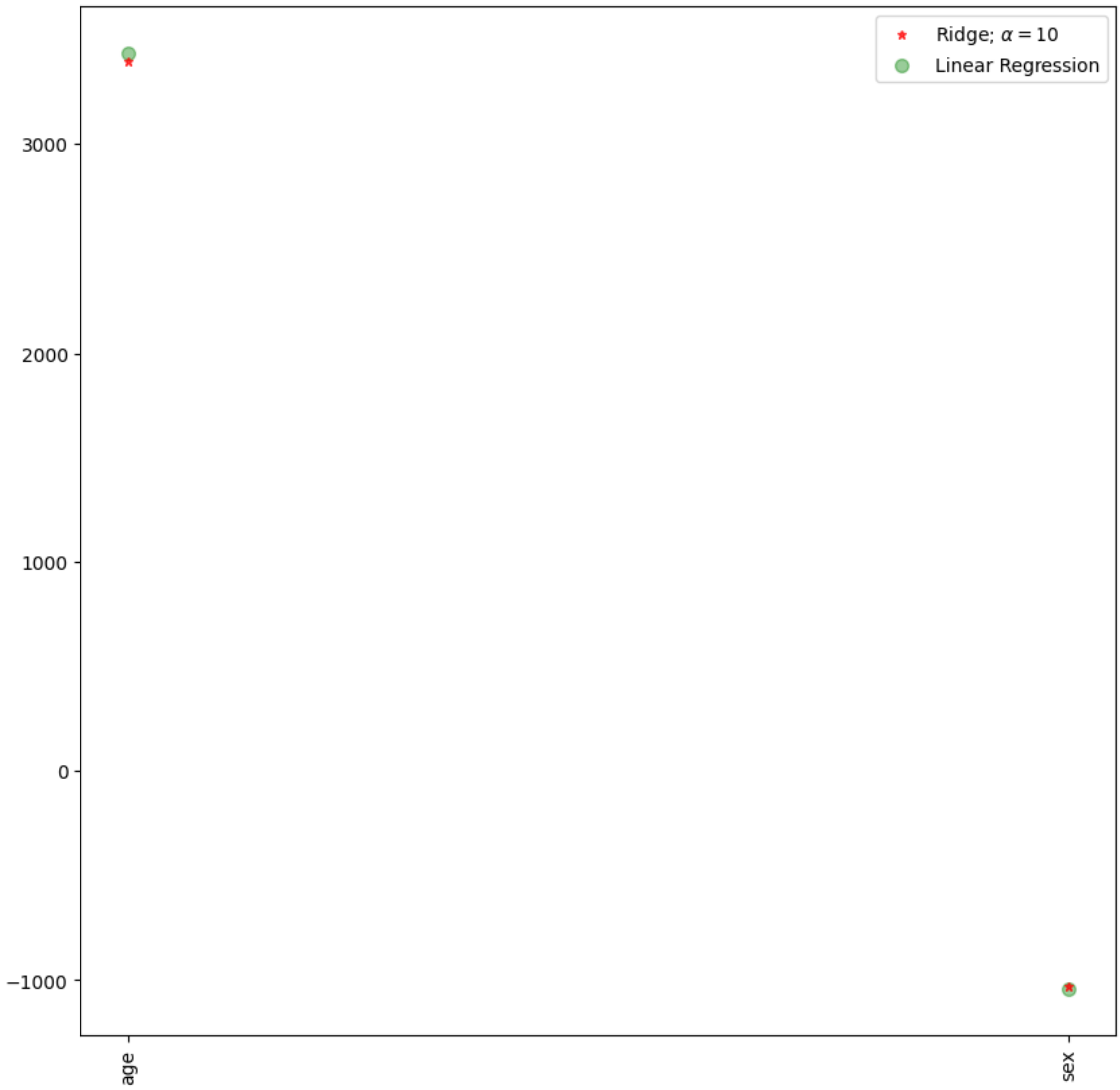
In [25]:

```
#Ridge Regression Model
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 0.08143796463046804
The test score for ridge model is 0.10202509697425621
```

In [26]:

```
plt.figure(figsize = (10, 10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,colo
#lot(rr100.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='gre
plt.xticks(rotation = 90)
plt.legend()
plt.show()
```



In [27]:

```
#Lasso regression model
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))
```

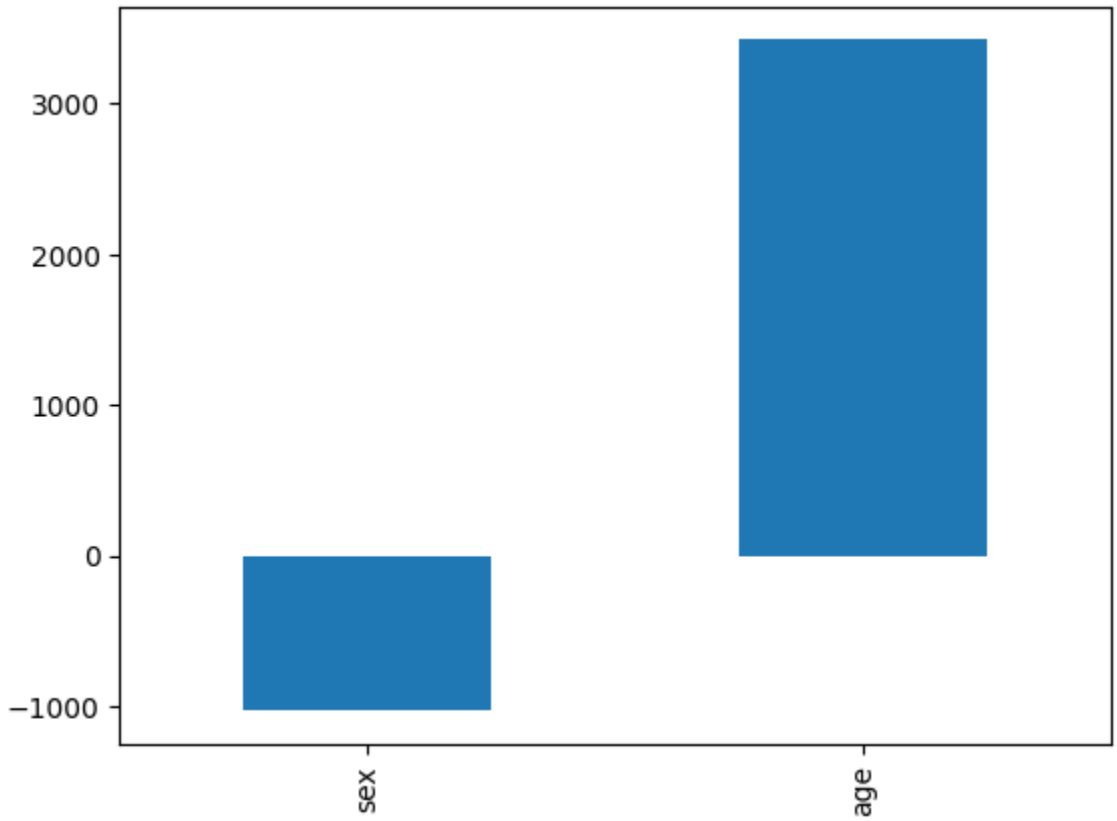
```
Lasso Model:

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

In [28]:

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

Out[28]:

```
<Axes: >
```



In [29]:

```
#Using the linear CV model
from sklearn.linear_model import LassoCV
#Lasso Cross validation
lasso_cv = LassoCV(alphas = [0.0001, 0.001,0.01, 0.1, 1, 10], random_state=0).fit(X_trai
#score
print(lasso_cv.score(X_train, y_train))
print(lasso_cv.score(X_test, y_test))
```

```
0.08144600503720845
0.10226046691368151
```

ElasticNet Regression

In [30]:

```
from sklearn.linear_model import ElasticNet
regr=ElasticNet()
regr.fit(X,y)
print(regr.coef_)
print(regr.intercept_)
regr.score(X,y)
```

```
[ 257.44760657 -511.96287073]
3941.933287820537
```

Out[30]:

```
0.09164498902013407
```

In [31]:

```
y_pred_elastic=regr.predict(X_train)
```

In [32]:

```
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 251337238.06352752
```

5.Data Prediction&Evaluation

In [33]:

```
prediction=lr.predict(X_test)
```

In [34]:

```
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: 0.1022033122179885
```

To find Error

In [35]:

```
print('MAE:',metrics.mean_absolute_error(y_test,prediction))
print('MSE:',metrics.mean_squared_error(y_test,prediction))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
MAE: 8564.630066831065
MSE: 109448853.74996018
RMSE: 10461.780620427871
```

Cross Validation

In [36]:

```
ridge_cv=RidgeCV(alphas=[1,10,100]).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 : 0.08143796463046815
THE TEST SCORE FOR RIDGE MODEL IS :0.10202509697425821
```

In [37]:

```
lasso_cv=LassoCV(alphas=[1,10,100]).fit(X_train,y_train)
print("THE TRAIN SCORE FOR RIDGE MODEL IS : {}".format(lasso_cv.score(X_train,y_train)))
print("THE TEST SCORE FOR RIDGE MODEL IS :{}".format(lasso_cv.score(X_test,y_test)))
```

```
THE TRAIN SCORE FOR RIDGE MODEL IS : 0.08144600503720845
THE TEST SCORE FOR RIDGE MODEL IS :0.10226046691368151
```

In [38]:

```
import pickle
```

In [39]:

```
filename="prediction"
pickle.dump(lr,open(filename,'wb'))
```

Conclusion

In the above project I got same value for four models :Linear for 0.08,0.102;Ridge for 0.08,0.102;Lasso for 0.08,0.102;Elastic for 0.08,0.102.So here I concluded there is no best fit model for the Health Insurance data set. I am going on Logistic Regression

Logistic Regression

In [40]:

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

In [41]:

```
df=pd.read_csv(r"C:\Users\RAMADEVI SURIPAKA\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]:

```
pd.set_option('display.max_row',10000000000)
pd.set_option('display.max_columns',10000000000)
pd.set_option('display.width',95)
print('This dataFrame has %d rows and %d columns'%(df.shape))
```

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

In [43]:

```
T={"smoker":{'yes':1,'no':2}}
df=df.replace(T)
df
```

Out[43]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	1	southwest	16884.924000
1	18	male	33.770	1	2	southeast	1725.552300
2	28	male	33.000	3	2	southeast	4449.462000
3	33	male	22.705	0	2	northwest	21984.470610
4	32	male	28.880	0	2	northwest	3866.855200
5	31	female	25.740	0	2	southeast	3756.621600
6	46	female	33.440	1	2	southeast	8240.589600
7	37	female	27.740	3	2	northwest	7281.505600
8	37	male	29.830	2	2	northeast	6406.410700
9	60	female	25.840	0	2	northwest	28923.136920

In [44]:

```
T={"sex":{'male':1,'female':2}}
df=df.replace(T)
df
```

Out[44]:

	age	sex	bmi	children	smoker	region	charges
0	19	2	27.900	0	1	southwest	16884.924000
1	18	1	33.770	1	2	southeast	1725.552300
2	28	1	33.000	3	2	southeast	4449.462000
3	33	1	22.705	0	2	northwest	21984.470610
4	32	1	28.880	0	2	northwest	3866.855200
5	31	2	25.740	0	2	southeast	3756.621600
6	46	2	33.440	1	2	southeast	8240.589600
7	37	2	27.740	3	2	northwest	7281.505600
8	37	1	29.830	2	2	northeast	6406.410700
9	60	2	25.840	0	2	northwest	28923.136920

In [45]:

```
features_matrix=df.iloc[:,0:4]
target_vector=df.iloc[:,-3]
print('The Features Matrix Has %d Rows And %d Columns(s)'%(features_matrix.shape))
print('The Features Matrix Has %d Rows And %d Columns(s)'%(np.array(target_vector).resha
```

```
The Features Matrix Has 1338 Rows And 4 Columns(s)
The Features Matrix Has 1338 Rows And 1 Columns(s)
```

In [46]:

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

In [47]:

```
algorithm=LogisticRegression(max_iter=100000)
```

In [48]:

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

In [49]:

```
observation=[[0,0.01,0.0003,0.0004]]
```

In [50]:

```
predictions=Logistic_Regression_Model.predict(observation)
```

In [51]:

```
print('The Model Predicted The Obsevation To Belong To Class %s'%(predictions))
```

```
The Model Predicted The Obsevation To Belong To Class [2]
```

In [52]:

```
print(""" The Model Says The Probability Of The Observation We Passed Belonging To Class
print()
print(""" The Model Says The Probability Of The Observation We Passed Belonging To Class
print()
```

```
 The Model Says The Probability Of The Observation We Passed Belonging To
Class['1'] Is 0.2025843785302377

 The Model Says The Probability Of The Observation We Passed Belonging To
Classs['0'] Is 0.7974156214697623
```

In [53]:

```
features = df.columns[0:2]
target = df.columns[-2]
#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 [ ]:

In [54]:

```
#Model
legr = LogisticRegression()
#Fit model
legr.fit(X_train, y_train)
#predict
#prediction = lr.predict(X_test)
#actual
actual = y_test
train_score_legr = legr.score(X_train, y_train)
test_score_legr = legr.score(X_test, y_test)
print("\nRidg Model:\n")
print("The train score for legr model is {}".format(train_score_legr))
print("The test score for legr model is {}".format(test_score_legr))
```

```
Ridg Model:

The train score for legr model is 0.2724358974358974
The test score for legr model is 0.24129353233830847
```

In [55]:

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

In [56]:

```
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.30)
lerg=LogisticRegression()
lerg.fit(X_train,y_train)
print(lerg.score(X_test,y_test))
```

```
0.7910447761194029

C:\Users\RAMADEVI SURIPAKA\AppData\Local\Programs\Python\Python310\lib\sit
e-packages\sklearn\utils\validation.py:1143: DataConversionWarning: A colu
mn-vector y was passed when a 1d array was expected. Please change the sha
pe of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Decision Tree Regression

In [57]:

```
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 [58]:

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

Out[58]:

	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 [59]:

```
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 [60]:

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

Out[60]:

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

In [61]:

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

Out[61]:

```
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 [62]:

```
converter={"sex":{"male":1,"female":2}}
df=df.replace(converter)
df
```

Out[62]:

	age	sex	bmi	children	smoker	region	charges
0	19	2	27.900	0	yes	southwest	16884.924000
1	18	1	33.770	1	no	southeast	1725.552300
2	28	1	33.000	3	no	southeast	4449.462000
3	33	1	22.705	0	no	northwest	21984.470610
4	32	1	28.880	0	no	northwest	3866.855200
5	31	2	25.740	0	no	southeast	3756.621600
6	46	2	33.440	1	no	southeast	8240.589600
7	37	2	27.740	3	no	northwest	7281.505600
8	37	1	29.830	2	no	northeast	6406.410700
9	60	2	25.840	0	no	northwest	28923.136920

In [63]:

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

In [64]:

```
(x_train,x_test,y_train,y_test)=train_test_split(all_inputs,all_classes,test_size=0.5)
```

In [65]:

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

In [66]:

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

Out[66]:

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

In [67]:

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

```
0.312406576980568
```

Conclusion:

In the above project I have done four models,I got same value for four models :Linear for 0.08,0.102;Ridge for 0.08,0.102;Lasso for 0.08,0.102;Elastic for 0.08,0.102.So here I concluded there is no best fit model for the Health Insurance data set. I am going on Logistic Regression.In the Logistic Regression I got accuracy 81,In Decision Tree Regression I got accuracy 31.So,I Concluded That LogisticRegression is the best fit model for the Health Insurance

In [ ]:

```
```