PROJECT ON INSURANCE:

PROBLEM STATEMENT:

WHICH MODEL SUITS FOR THE INSURANCE PROBLEM

In [119]:

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

DATA COLLECTION

READ THE DATA

In [120]:

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

Out[120]:

	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 CLEANING

In [121]:

```
df.columns
```

Out[121]:

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

In [122]:
```
df.describe()
```

Out[122]:

	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 [123]:
```
df.tail()
```

Out[123]:

	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 [124]:
```
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 [125]:
```
df.shape
```

Out[125]:
```
(1338, 7)
```

EXPLORATORY DATA ANALYSIS

DATA PREPOCESSING

In [126]:
```
df.isnull().any()
```

Out[126]:
```
age         False
sex         False
bmi         False
children    False
smoker      False
region      False
charges     False
dtype: bool
```

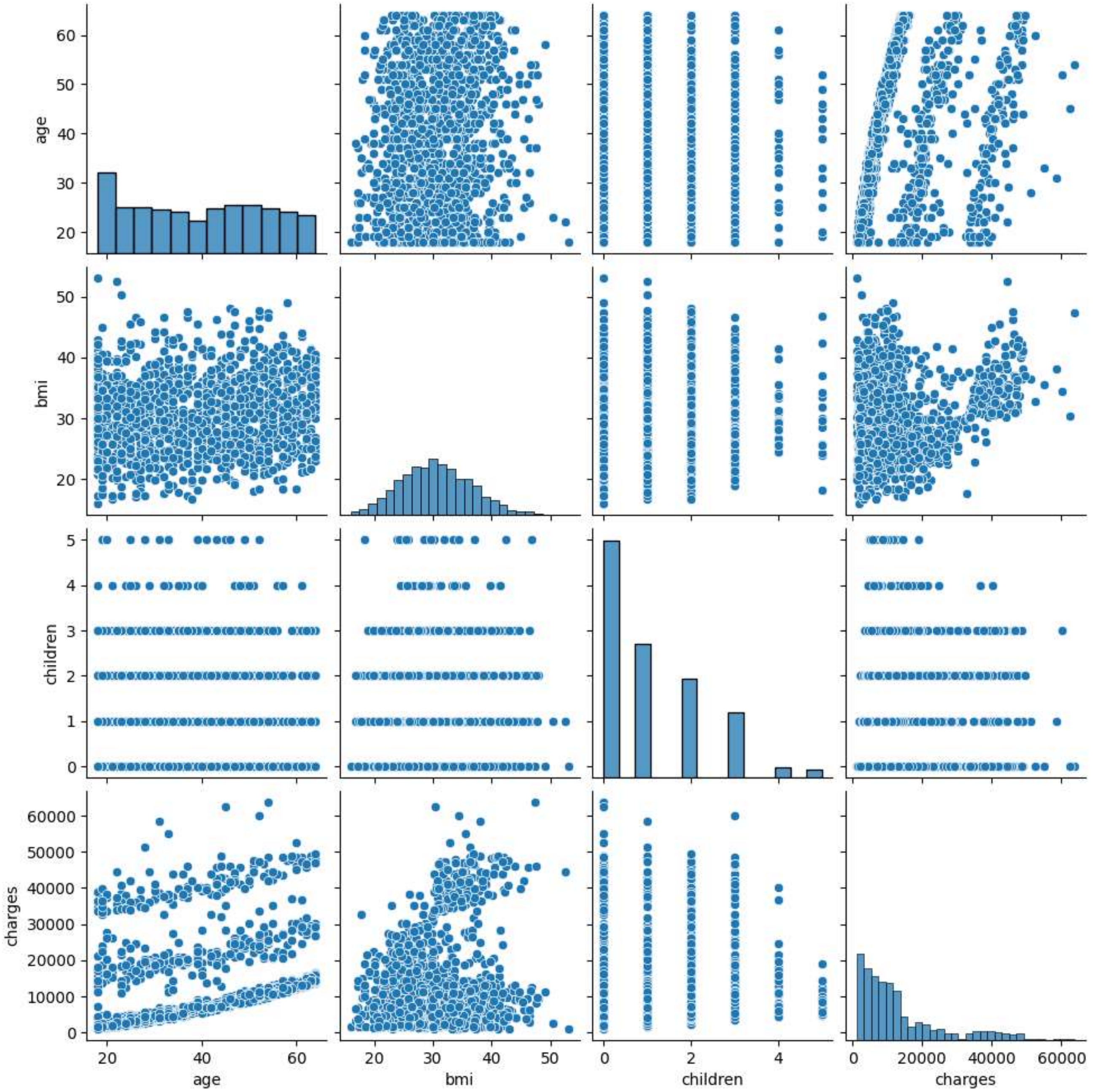
HERE THERE IS NO NULL VALUE.SO THIS IS NOT NECESSARY TO ADJUST THE VALUES

DATA VISUALIZATON

```
In [127]: #plots
```

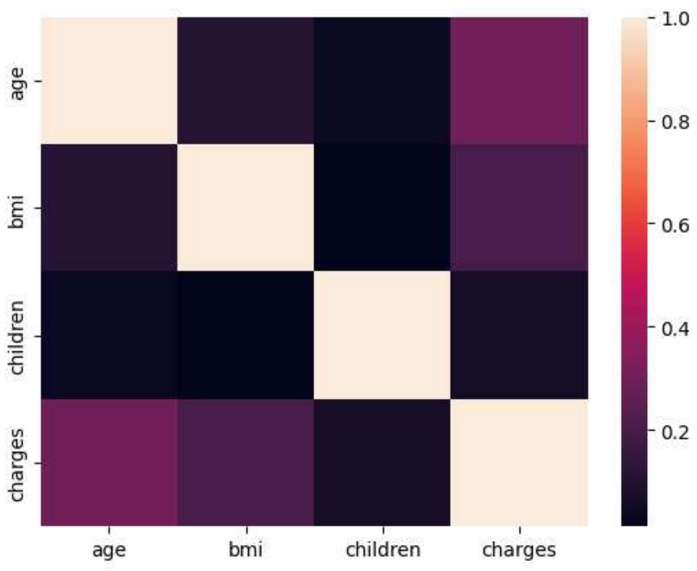
```
In [128]: sns.pairplot(df)
```

```
Out[128]: <seaborn.axisgrid.PairGrid at 0x21f20427040>
```



In [129]:
```
Insurancedf=df[['age','bmi', 'children','charges']]
sns.heatmap(Insurancedf.corr())
```

Out[129]:
```
<Axes: >
```



Feature Scaling:spliting data into train and test data sets

In [130]:
```
x=np.array(df['age']).reshape(-1,1)
y=np.array(df['charges']).reshape(-1,1)
```

In [131]:
```
from sklearn.linear_model import Ridge,RidgeCV,Lasso
from sklearn.preprocessing import StandardScaler
```

DATA MODELING:

first we go through REGRESSION models

1. LINEAR REGRESSION

In [113]:
```
#1.TO MAKE CONVERSIONS TO HAVE SAME DATA TYPE TO FIND REGRESSION MODEL
```

In [134]:
```
A={"smoker":{'yes':1,'no':2}}
df=df.replace(A)
df
```

Out[134]:

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

1338 rows × 7 columns

In [135]:

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

Out[135]:

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

1338 rows × 7 columns

In [136]:

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

Out[136]:

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

1338 rows × 7 columns

In [137]:

```
#2.TO FIND DIMENSIONS OF X and y
features = df.columns[0:2]
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, 2)
The dimension of X_test is (402, 2)
```

In [140]:

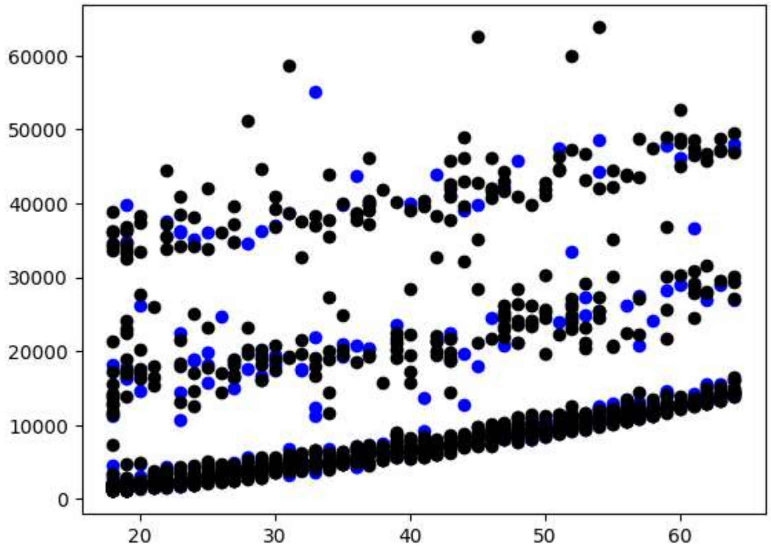
```
#3.TO FIND SCORE OF LinearRegression()
lir = LinearRegression()
lir.fit(X_train, y_train)
lir.score(X_test,y_test)
print(lir.score(X_test,y_test))
#4.TO FIND TRAIN AND TEST SCORE OF LinearRegression()
train_score_lir = lir.score(X_train, y_train)
test_score_lir = lir.score(X_test, y_test)
print("\nLinear Regression Model:\n")
print("The train score for lir model is {}".format(train_score_lir))
print("The test score for lir model is {}".format(test_score_lir))
```

```
0.1022033122179885

Linear Regression Model:

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

graph of linear regression:

In [15]:

```

plt.scatter(x_test,y_test,color='b')
plt.scatter(x_train,y_train,color='k')
plt.show()
```



In the case of smaller dataset Taken(*not necessary)

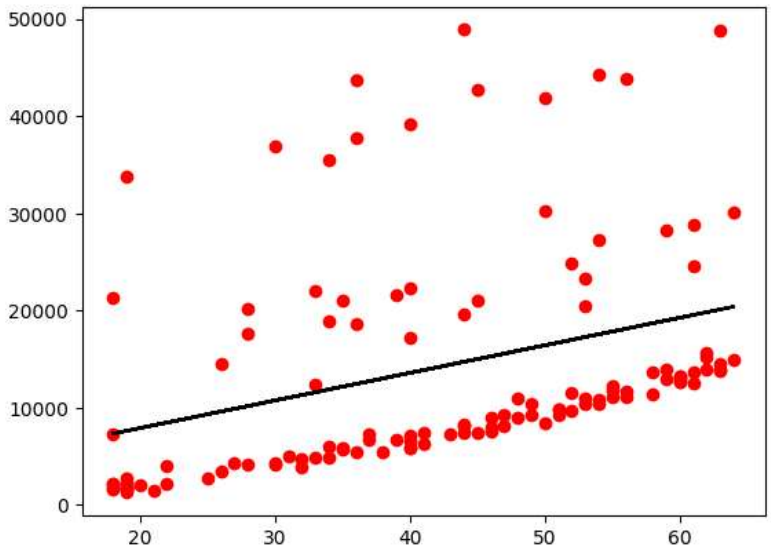
In [16]:

```
df500=df[:][:500]
```

In [17]:

```
df500.fillna(method='ffill',inplace=True)
x=np.array(df500['age']).reshape(-1,1)
y=np.array(df500['charges']).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='r')
plt.plot(x_test,y_pred,color='k')
plt.show()
```

```
Regression: 0.1293688174041372
```



RIDGE REGRESSION

In [141]:

```
#2.TO FIND DIMENSIONS OF X and y
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)
```

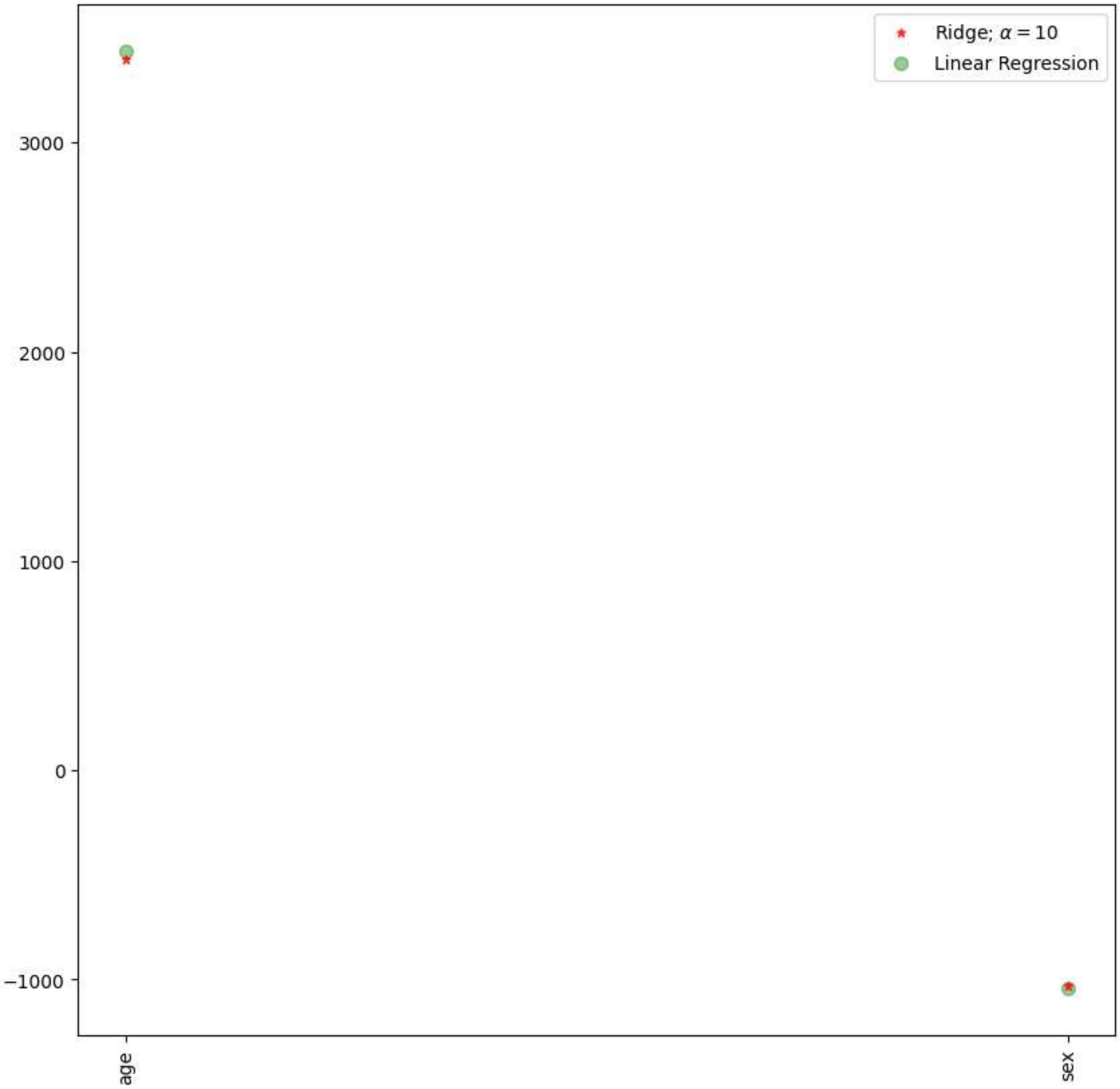
```
The dimension of X_train is (936, 2)
The dimension of X_test is (402, 2)
```

In [33]:

```
ridgeReg = Ridge(alpha=10)
ridgeReg.fit(X_train,y_train)
Print(ridgeReg.score(X_test, y_test))
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 0.10202509697425632
```

In [34]:
```
plt.figure(figsize = (10, 10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label=r'Ridge; $\alpha = 10$
#plt.
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear Regression')
plt.xticks(rotation = 90)
plt.legend()
plt.show()
```



LASSO REGRESSION

In [36]:
```
lr=Lasso(alpha=10)
lr.fit(x_train,y_train)
print(lr.score(x_test,y_test))
```

```
0.10226046691368151
```

In [37]:

```
print("\nLasso Model: \n")
lr = Lasso(alpha = 10)
lr.fit(X_train,y_train)
train_score_ls =lr.score(X_train,y_train)
test_score_ls =lr.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 0.08144600503720834
The test score for ls model is 0.10226046691368151
```

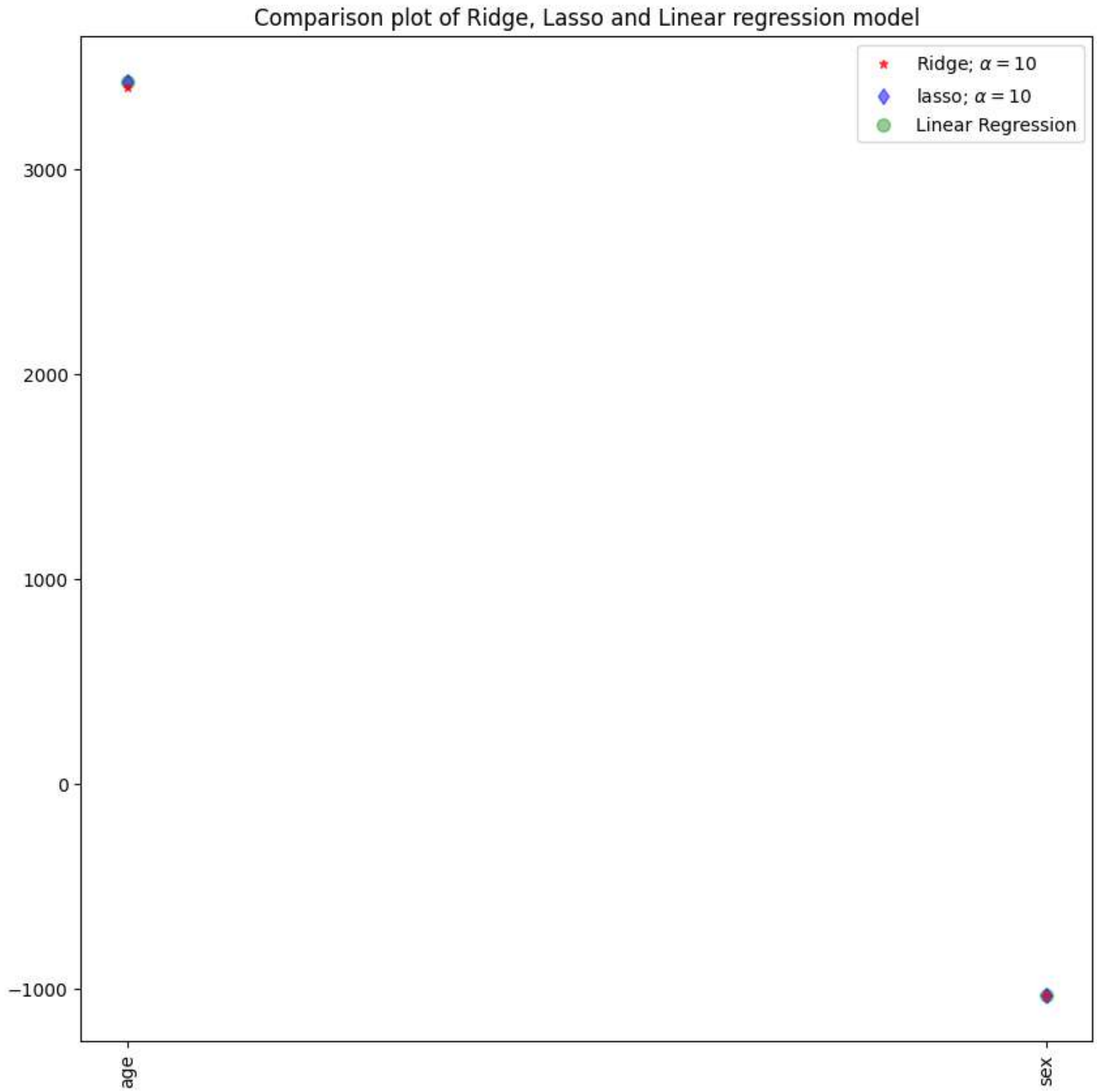
In [44]:

```
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_train, y_train)
#score
print(lasso_cv.score(X_train, y_train))
print(lasso_cv.score(X_test, y_test))
```

```
0.08144600503720834
0.10226046691368151
```

In [45]:

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

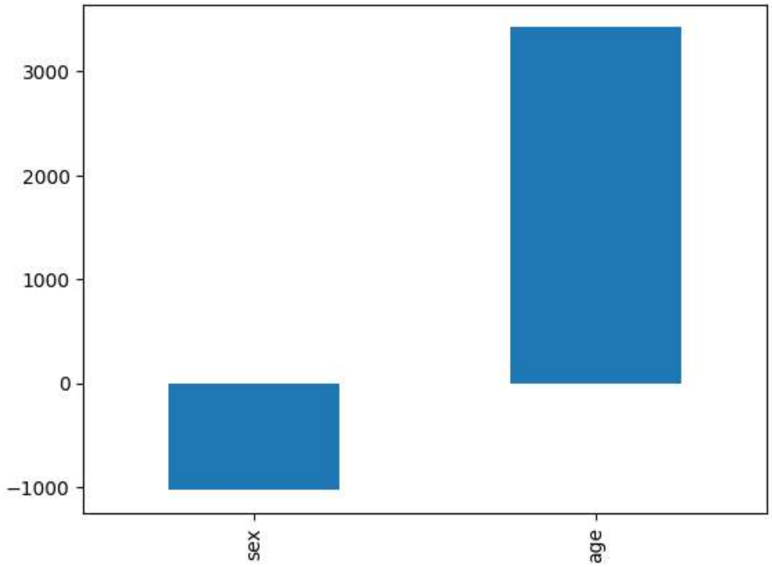


In [47]:

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

Out[47]:

```
<Axes: >
```



ELASTIC NET

In [48]:

```
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[48]:

```
0.09164498902013407
```

Here ridge and lasso rigression have low accuracy.So,Elastic Net also have low accuracy which depends on the values of ridge and lasso.

In [49]:

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

In [50]:

```
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 [51]:

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

In [52]:

```
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 [53]: 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: 8564.838037174472
MSE: 109441886.11907542
RMSE: 10461.447611065852
```

Cross Validation

```
In [54]: ridge_cv=RidgeCV(alphas=[1,10,100]).fit(X_train,y_train)
         print("The train score for ridgecrossvalidation model is : {}".format(ridge_cv.score(X_train,y_train)))
         print("The test score for ridgcrossvalidation model is :{}".format(ridge_cv.score(X_test,y_test)))
```

```
The train score for ridgecrossvalidation model is : 0.08143796463046815
The test score for ridgcrossvalidation model is :0.10202509697425854
```

```
In [55]: lasso_cv=LassoCV(alphas=[1,10,100]).fit(X_train,y_train)
         print(" The train score for lassocrossvalidation model is : {}".format(lasso_cv.score(X_train,y_train)))
         print("The test score for lassocrossvalidation model is:{}".format(lasso_cv.score(X_test,y_test)))
```

```
 The train score for lassocrossvalidation model is : 0.08144600503720834
The test score for lassocrossvalidation model is:0.10226046691368151
```

Here the regression score(accuracy) is low.So,check the categorical models

LOGISTIC REGRESSION

```
In [59]: A={"smoker":{'yes':1,'no':2}}
         df=df.replace(A)
         df
```

Out[59]:

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

1338 rows × 7 columns

```
In [60]: 
         x=df.iloc[:,:-1].values
         y=df.iloc[:,1].values
```

```
In [62]: 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).reshape(-1,1).shape))
```

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

```
In [63]: features_matrix_standardized=StandardScaler().fit_transform(features_matrix)
```

In [64]:
```
algorithm=LogisticRegression(max_iter=100000)
```

In [65]:
```
Logistic_Regression_Model=algorithm.fit(features_matrix_standardized,target_vector)
```

In [66]:
```
observation=[[0,0.01,0.0003,0.0004]]
```

In [67]:
```
predictions=Logistic_Regression_Model.predict(observation)
```

In [68]:
```
print('The Model Predicted The Obsevation To Belong To Class %s'%(predictions))
```

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

In [75]:
```
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 [78]:
```
#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("\nlogistic 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))
```

```

logistic Model:

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

In [79]:
```
x=np.array(df['age']).reshape(-1,1)
y=np.array(df['smoker']).reshape(-1,1)
```

In [80]:
```
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))
```

```
0.8233830845771144
```

```
C:\Users\HP\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\utils\validation.py:1143: DataConversionWarni
ng: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example u
sing ravel().
  y = column_or_1d(y, warn=True)
```

we can apply another regression models also...

Decision Tree Classifier

In [81]:
```
from sklearn.tree import DecisionTreeClassifier
```

```
In [82]: df['sex'].value_counts()
```

```
Out[82]: sex
         1    676
         2    662
         Name: count, dtype: int64
```

```
In [89]: df['bmi'].value_counts()
```

```
Out[89]: bmi
         32.300    13
         28.310     9
         30.495     8
         30.875     8
         31.350     8
                   ..
         46.200     1
         23.800     1
         44.770     1
         32.120     1
         30.970     1
         Name: count, Length: 548, dtype: int64
```

```
In [90]: X=["sex","age","bmi"]
         y=["1","2"]
         all_inputs=df[x]
         all_classes=df["region"]
```

```
In [91]: (X_train,X_test,y_train,y_test)=train_test_split(all_inputs,all_classes,test_size=0.5)
```

```
In [92]: clf=DecisionTreeClassifier(random_state=0)
```

```
In [93]: clf.fit(X_train,y_train)
```

```
Out[93]: ▾        DecisionTreeClassifier

         DecisionTreeClassifier(random_state=0)
```

```
In [94]: score=clf.score(X_test,y_test)
         print(score)

         0.3034379671150972
```

CONCULSION:

```
In [ ]:     It is concluded that regression models:linear,Ridge,Lasso have 10.22% of accuracy,DecisionTreeClassifier has 30% accurac
            logistic regression has 82% accuracy.So,Logistic regression comparatively high with another models.So,Logistic regressio
            best fit for health insurance.
```