

```
In [1]: import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing,svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.linear_model import Ridge
from sklearn.preprocessing import StandardScaler
```

```
In [2]: df=pd.read_csv(r"C:\Users\DELL\Downloads\bottle1.csv")
df
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_7216\3734436603.py:1: DtypeWarning: Columns (47,73) have
mixed types. Specify dtype option on import or set low_memory=False.
df=pd.read_csv(r"C:\Users\DELL\Downloads\bottle1.csv")
```

Out[2]:

| | Cst_Cnt | Btl_Cnt | Sta_ID | Depth_ID | Depthm | T_degC | Salnty | O2ml_L | STheta | O2Sat | ... | R_PHAEAO | R_PRES |
|--------|---------|---------|----------------|--|--------|--------|---------|--------|----------|--------|-----|----------|--------|
| 0 | 1 | 1 | 054.0 056.0 | 19- 4903CR- HY-060- 0930- 05400560- 0000A-3 | 0 | 10.500 | 33.4400 | NaN | 25.64900 | NaN | ... | NaN | 0 |
| 1 | 1 | 2 | 054.0 056.0 | 19- 4903CR- HY-060- 0930- 05400560- 0008A-3 | 8 | 10.460 | 33.4400 | NaN | 25.65600 | NaN | ... | NaN | 8 |
| 2 | 1 | 3 | 054.0 056.0 | 19- 4903CR- HY-060- 0930- 05400560- 0010A-7 | 10 | 10.460 | 33.4370 | NaN | 25.65400 | NaN | ... | NaN | 10 |
| 3 | 1 | 4 | 054.0 056.0 | 19- 4903CR- HY-060- 0930- 05400560- 0019A-3 | 19 | 10.450 | 33.4200 | NaN | 25.64300 | NaN | ... | NaN | 19 |
| 4 | 1 | 5 | 054.0 056.0 | 19- 4903CR- HY-060- 0930- 05400560- 0020A-7 | 20 | 10.450 | 33.4210 | NaN | 25.64300 | NaN | ... | NaN | 20 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 864858 | 34404 | 864859 | 093.4 026.4 | 20- 1611SR- MX-310- 2239- 09340264- 0000A-7 | 0 | 18.744 | 33.4083 | 5.805 | 23.87055 | 108.74 | ... | 0.18 | 0 |
| 864859 | 34404 | 864860 | 093.4 026.4 | 20- 1611SR- MX-310- 2239- 09340264- 0002A-3 | 2 | 18.744 | 33.4083 | 5.805 | 23.87072 | 108.74 | ... | 0.18 | 2 |
| 864860 | 34404 | 864861 | 093.4 026.4 | 20- 1611SR- MX-310- 2239- 09340264- 0005A-3 | 5 | 18.692 | 33.4150 | 5.796 | 23.88911 | 108.46 | ... | 0.18 | 5 |
| 864861 | 34404 | 864862 | 093.4 026.4 | 20- 1611SR- MX-310- 2239- 09340264- 0010A-3 | 10 | 18.161 | 33.4062 | 5.816 | 24.01426 | 107.74 | ... | 0.31 | 10 |
| 864862 | 34404 | 864863 | 093.4 026.4 | 20- 1611SR- MX-310- 2239- 09340264- 0015A-3 | 15 | 17.533 | 33.3880 | 5.774 | 24.15297 | 105.66 | ... | 0.61 | 15 |

864863 rows × 74 columns



```
In [3]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 864863 entries, 0 to 864862
Data columns (total 74 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Cst_Cnt          864863 non-null   int64  
 1   Btl_Cnt          864863 non-null   int64  
 2   Sta_ID           864863 non-null   object  
 3   Depth_ID         864863 non-null   object  
 4   Depthm           864863 non-null   int64  
 5   T_degC           853900 non-null   float64 
 6   Salnty           817509 non-null   float64 
 7   O2ml_L           696201 non-null   float64 
 8   STtheta           812174 non-null   float64 
 9   O2Sat            661274 non-null   float64 
 10  Oxy_µmol/Kg      661268 non-null   float64 
 11  BtlNum           118667 non-null   float64 
 12  RecInd           864863 non-null   int64  
 13  T_prec            853900 non-null   float64 
 14  T_qual            23127 non-null   float64 
 15  S_prec            817509 non-null   float64 
 16  S_qual            74914 non-null   float64 
 17  P_qual            673755 non-null   float64 
 18  Q_qual            184676 non-null   float64 
 19  SThtaq            65823 non-null   float64 
 20  O2Satq            217797 non-null   float64 
 21  ChlorA            225272 non-null   float64 
 22  Chlqua            639166 non-null   float64 
 23  Phaeop             225271 non-null   float64 
 24  Phaqua            639170 non-null   float64 
 25  PO4uM              413317 non-null   float64 
 26  PO4q               451786 non-null   float64 
 27  SiO3uM            354091 non-null   float64 
 28  SiO3qu            510866 non-null   float64 
 29  NO2uM              337576 non-null   float64 
 30  NO2q               529474 non-null   float64 
 31  NO3uM              337403 non-null   float64 
 32  NO3q               529933 non-null   float64 
 33  NH3uM              64962 non-null   float64 
 34  NH3q               808299 non-null   float64 
 35  C14As1             14432 non-null   float64 
 36  C14A1p             12760 non-null   float64 
 37  C14A1q             848605 non-null   float64 
 38  C14As2             14414 non-null   float64 
 39  C14A2p             12742 non-null   float64 
 40  C14A2q             848623 non-null   float64 
 41  DarkAs              22649 non-null   float64 
 42  DarkAp              20457 non-null   float64 
 43  DarkAq              840440 non-null   float64 
 44  MeanAs              22650 non-null   float64 
 45  MeanAp              20457 non-null   float64 
 46  MeanAq              840439 non-null   float64 
 47  IncTim              14437 non-null   object  
 48  LightP              18651 non-null   float64 
 49  R_Depth             864863 non-null   int64  
 50  R_TEMP              853900 non-null   float64 
 51  R_POTEMP            818816 non-null   float64 
 52  R_SALINITY           817509 non-null   float64 
 53  R_SIGMA              812007 non-null   float64 
 54  R_SVA                812092 non-null   float64 
 55  R_DYNHT              818206 non-null   float64 
 56  R_O2                 696201 non-null   float64 
 57  R_O2Sat              666448 non-null   float64 
 58  R_SI03              354099 non-null   float64 
 59  R_PO4                413325 non-null   float64 
 60  R_NO3                337411 non-null   float64 
 61  R_NO2                337584 non-null   float64 
 62  R_NH4                64982 non-null   float64 
 63  R_CHLA              225276 non-null   float64
```

```

64 R_PHAEO          225275 non-null  float64
65 R_PRES           864863 non-null  int64
66 R_SAMP            122006 non-null  float64
67 DIC1              1999 non-null  float64
68 DIC2              224 non-null   float64
69 TA1               2084 non-null  float64
70 TA2               234 non-null   float64
71 pH2                10 non-null   float64
72 pH1                84 non-null   float64
73 DIC Quality Comment 55 non-null  object
dtypes: float64(64), int64(6), object(4)
memory usage: 488.3+ MB

```

In [4]: `df.describe()`

Out[4]:

| | Cst_Cnt | Btl_Cnt | Depthm | T_degC | Salnty | O2ml_L | STheta | C |
|--------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|-----------|
| count | 864863.000000 | 864863.000000 | 864863.000000 | 853900.000000 | 817509.000000 | 696201.000000 | 812174.000000 | 661274.00 |
| mean | 17138.790958 | 432432.000000 | 226.831951 | 10.799677 | 33.840350 | 3.392468 | 25.819394 | 57.10 |
| std | 10240.949817 | 249664.587269 | 316.050259 | 4.243825 | 0.461843 | 2.073256 | 1.167787 | 37.09 |
| min | 1.000000 | 1.000000 | 0.000000 | 1.440000 | 28.431000 | -0.010000 | 20.934000 | -0.10 |
| 25% | 8269.000000 | 216216.500000 | 46.000000 | 7.680000 | 33.488000 | 1.360000 | 24.965000 | 21.10 |
| 50% | 16848.000000 | 432432.000000 | 125.000000 | 10.060000 | 33.863000 | 3.440000 | 25.996000 | 54.40 |
| 75% | 26557.000000 | 648647.500000 | 300.000000 | 13.880000 | 34.196900 | 5.500000 | 26.646000 | 97.60 |
| max | 34404.000000 | 864863.000000 | 5351.000000 | 31.140000 | 37.034000 | 11.130000 | 250.784000 | 214.10 |

8 rows × 70 columns

In [5]: `df.isna().any()`

```

Out[5]: Cst_Cnt      False
        Btl_Cnt     False
        Sta_ID      False
        Depth_ID    False
        Depthm     False
        ...
        TA1         True
        TA2         True
        pH2         True
        pH1         True
        DIC Quality Comment  True
Length: 74, dtype: bool

```

In [6]: `df.isnull().sum()`

```

Out[6]: Cst_Cnt      0
        Btl_Cnt     0
        Sta_ID      0
        Depth_ID    0
        Depthm     0
        ...
        TA1         862779
        TA2         864629
        pH2         864853
        pH1         864779
        DIC Quality Comment  864808
Length: 74, dtype: int64

```

```
In [7]: df=df[['Salnty','T_degC']]  
df.columns=['Sal','Temp']
```

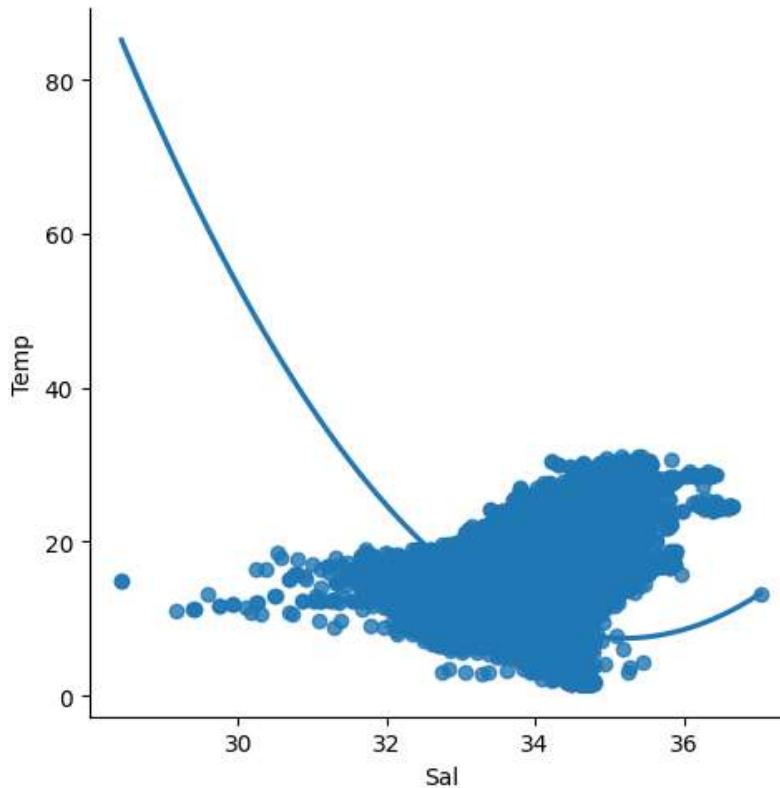
```
In [8]: df.head(20)
```

Out[8]:

| | Sal | Temp |
|----|--------|-------|
| 0 | 33.440 | 10.50 |
| 1 | 33.440 | 10.46 |
| 2 | 33.437 | 10.46 |
| 3 | 33.420 | 10.45 |
| 4 | 33.421 | 10.45 |
| 5 | 33.431 | 10.45 |
| 6 | 33.440 | 10.45 |
| 7 | 33.424 | 10.24 |
| 8 | 33.420 | 10.06 |
| 9 | 33.494 | 9.86 |
| 10 | 33.510 | 9.83 |
| 11 | 33.580 | 9.67 |
| 12 | 33.640 | 9.50 |
| 13 | 33.689 | 9.32 |
| 14 | 33.847 | 8.76 |
| 15 | 33.860 | 8.71 |
| 16 | 33.876 | 8.53 |
| 17 | NaN | 8.45 |
| 18 | 33.926 | 8.26 |
| 19 | 33.980 | 7.96 |

```
In [9]: sns.lmplot(x='Sal',y='Temp',data=df,order=2,ci=None)
```

```
Out[9]: <seaborn.axisgrid.FacetGrid at 0x22e4b682b50>
```



```
In [10]: df.fillna(method='ffill')
```

```
Out[10]:
```

| | Sal | Temp |
|--------|---------|--------|
| 0 | 33.4400 | 10.500 |
| 1 | 33.4400 | 10.460 |
| 2 | 33.4370 | 10.460 |
| 3 | 33.4200 | 10.450 |
| 4 | 33.4210 | 10.450 |
| ... | ... | ... |
| 864858 | 33.4083 | 18.744 |
| 864859 | 33.4083 | 18.744 |
| 864860 | 33.4150 | 18.692 |
| 864861 | 33.4062 | 18.161 |
| 864862 | 33.3880 | 17.533 |

864863 rows × 2 columns

```
In [11]: df.fillna(value=0,inplace=True)
```

C:\Users\DELL\AppData\Local\Temp\ipykernel_7216\1434098079.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
df.fillna(value=0,inplace=True)
```

```
In [12]: x=np.array(df['Sal']).reshape(-1,1)
y=np.array(df['Temp']).reshape(-1,1)
```

```
In [13]: df.dropna(inplace=True)
```

C:\Users\DELL\AppData\Local\Temp\ipykernel_7216\1379821321.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

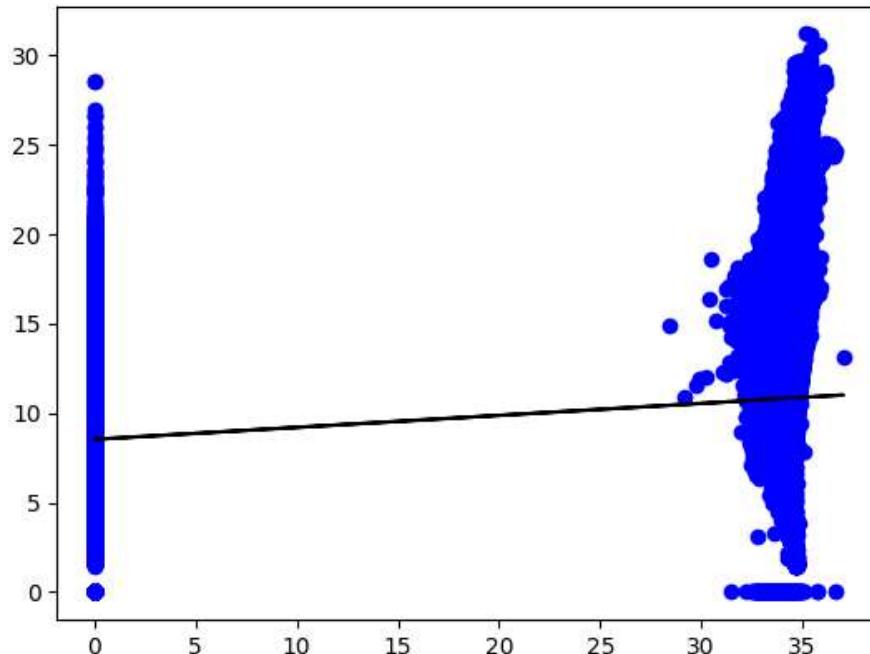
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
df.dropna(inplace=True)

```
In [14]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
```

```
In [15]: regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

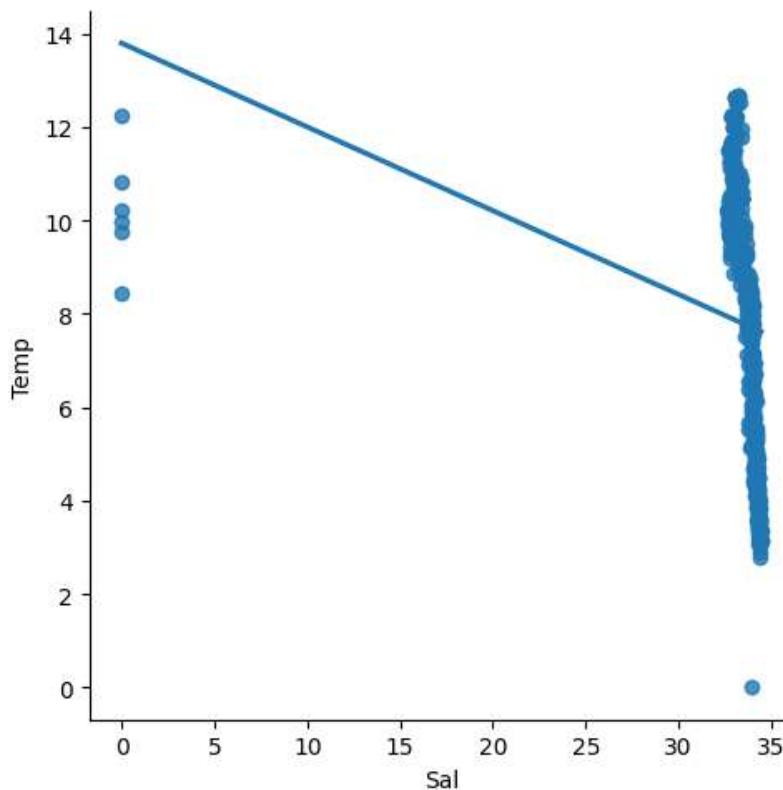
0.014887373627622047

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



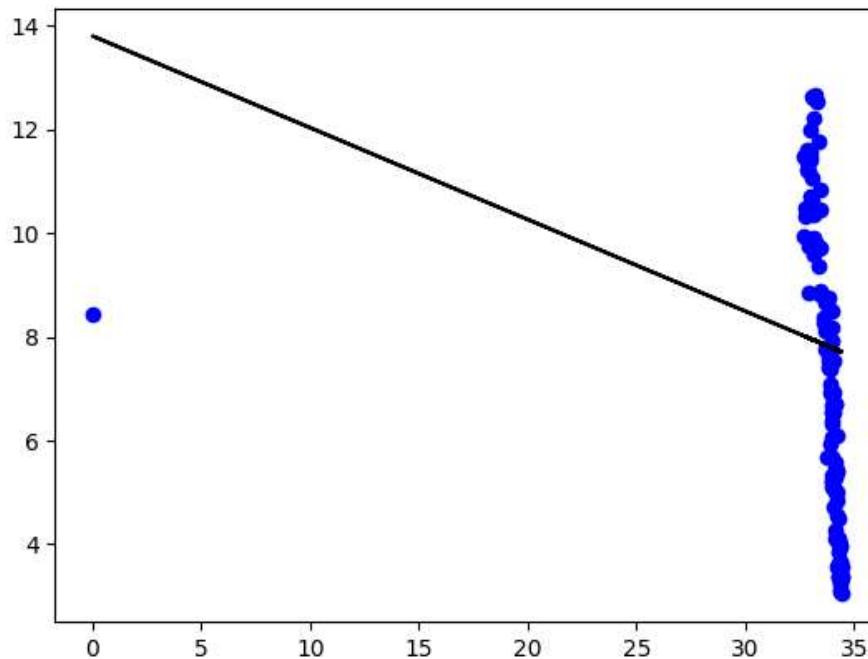
```
In [17]: df500=df[:][:500]
sns.lmplot(x='Sal',y='Temp',data=df500,order=1,ci=None)
```

```
Out[17]: <seaborn.axisgrid.FacetGrid at 0x22e25c60c50>
```



```
In [18]: df500.fillna(method='ffill',inplace=True)
x=np.array(df500['Sal']).reshape(-1,1)
y=np.array(df500['Temp']).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: 0.022037637344194616



```
In [19]: 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)
```

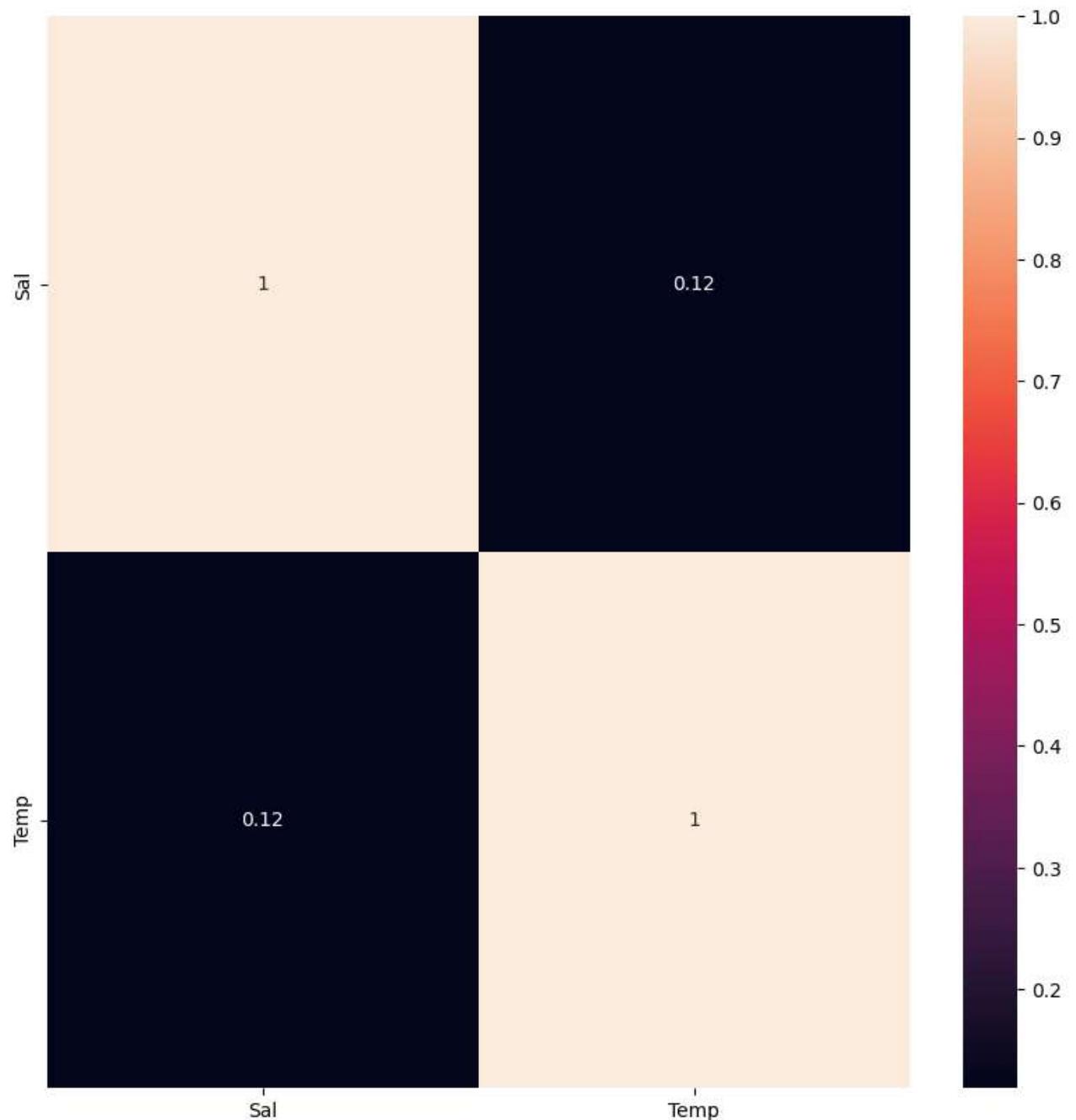
R2 score: 0.022037637344194616

```
In [20]: #conclusion: Linear regression is not fit for the model
```

Ridge and lasso

```
In [21]: plt.figure(figsize = (10, 10))
sns.heatmap(df.corr(), annot = True)
```

Out[21]: <Axes: >



```
In [22]: 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 (605404, 2)
The dimension of X_test is (259459, 2)

```
In [23]: 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))
```

Linear Regression Model:

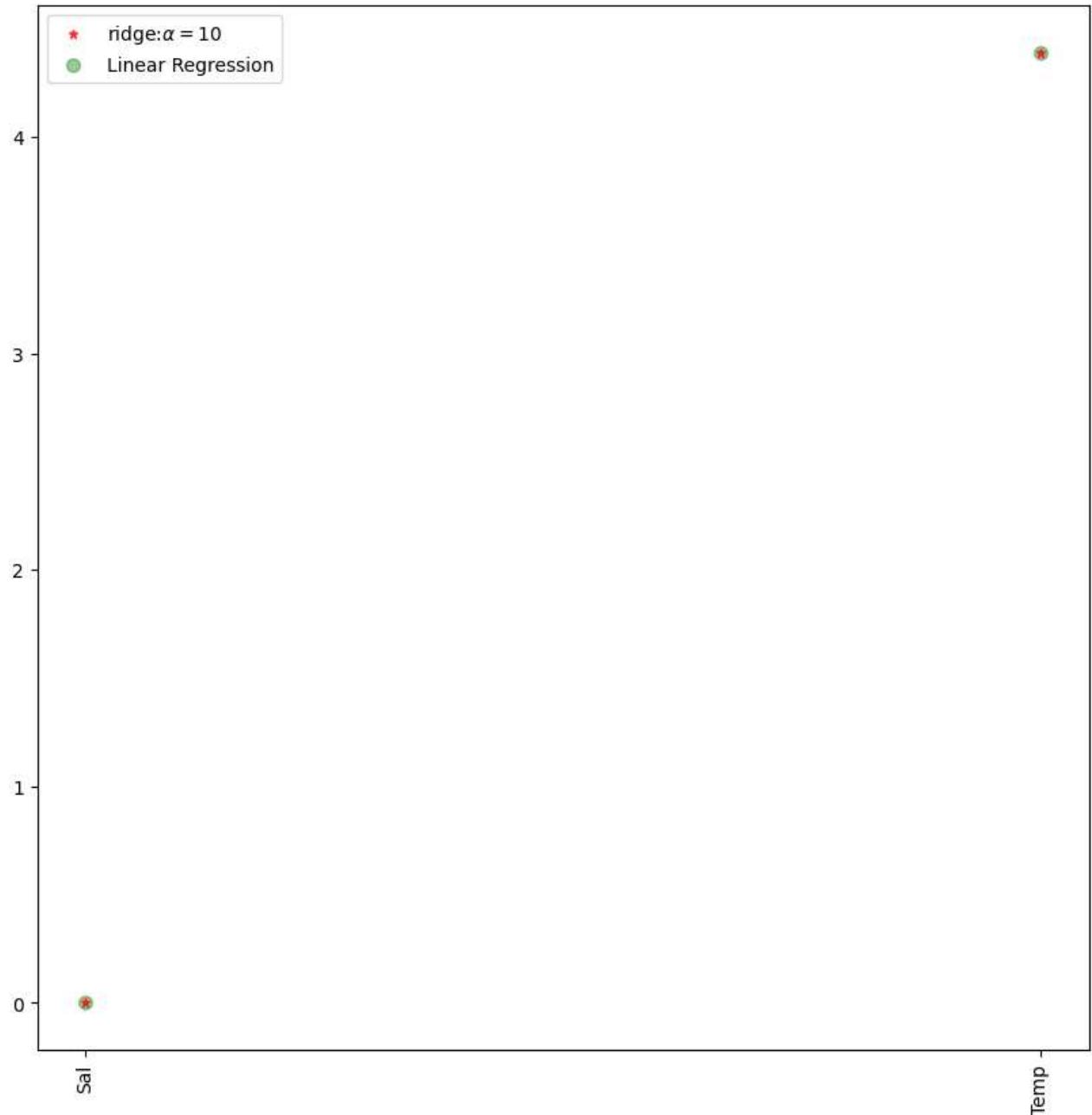
The train score for lr model is 1.0
The test score for lr model is 1.0

```
In [24]: 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 0.999999999723243
The test score for ridge model is 0.9999999997231402

```
In [25]: plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label='Ridge')
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear')
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



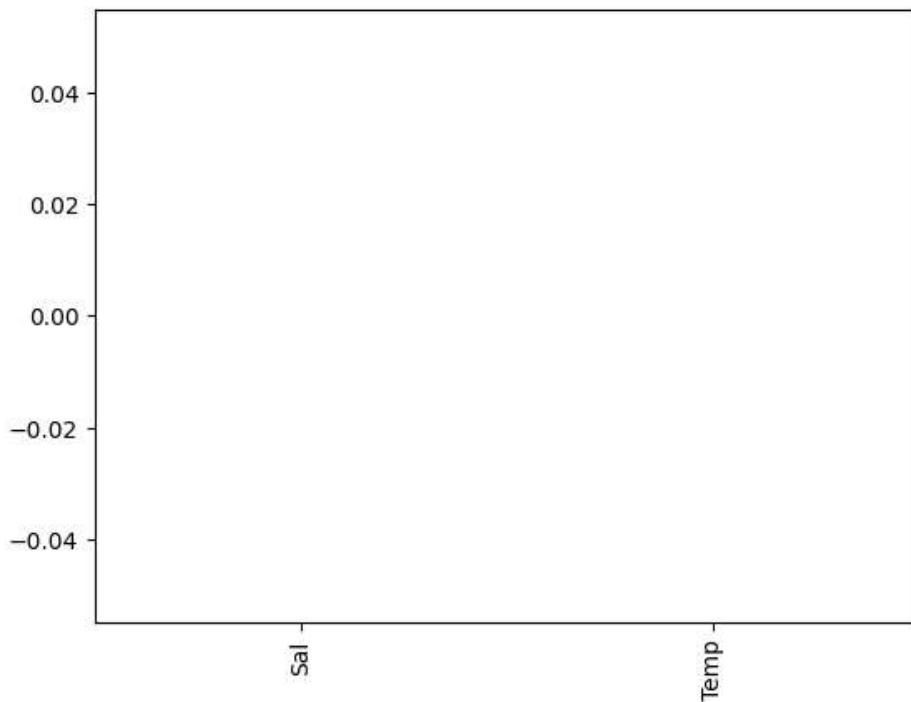
```
In [26]: #Lasso regression model
print("\nLasso Model: \n")
lasso = Lasso(alpha = 10)
lasso.fit(X_train,y_train)
train_score_ls = lasso.score(X_train,y_train)
test_score_ls = lasso.score(X_test,y_test)
print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is {}".format(test_score_ls))
```

Lasso Model:

The train score for ls model is 0.0
 The test score for ls model is -1.9031696447013857e-05

```
In [27]: pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "bar")
```

Out[27]: <Axes: >



```
In [28]: 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))
```

0.999999994806811
 0.999999994806712

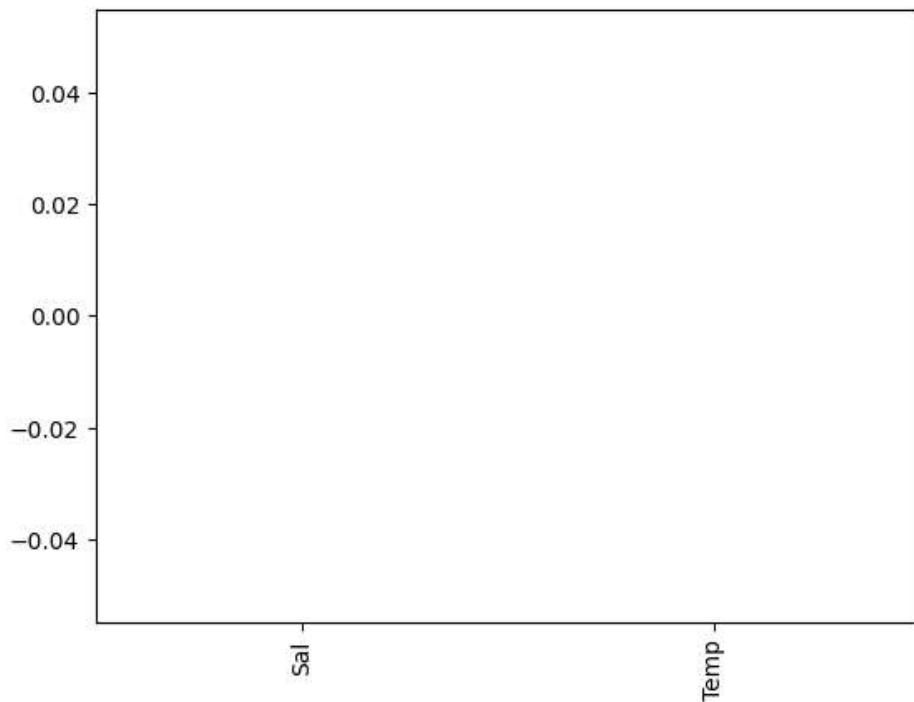
```
In [29]: #Lasso regression model
print("\nLasso Model: \n")
lasso = Lasso(alpha = 10)
lasso.fit(X_train,y_train)
train_score_ls = lasso.score(X_train,y_train)
test_score_ls = lasso.score(X_test,y_test)
print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is {}".format(test_score_ls))
```

Lasso Model:

The train score for ls model is 0.0
The test score for ls model is -1.9031696447013857e-05

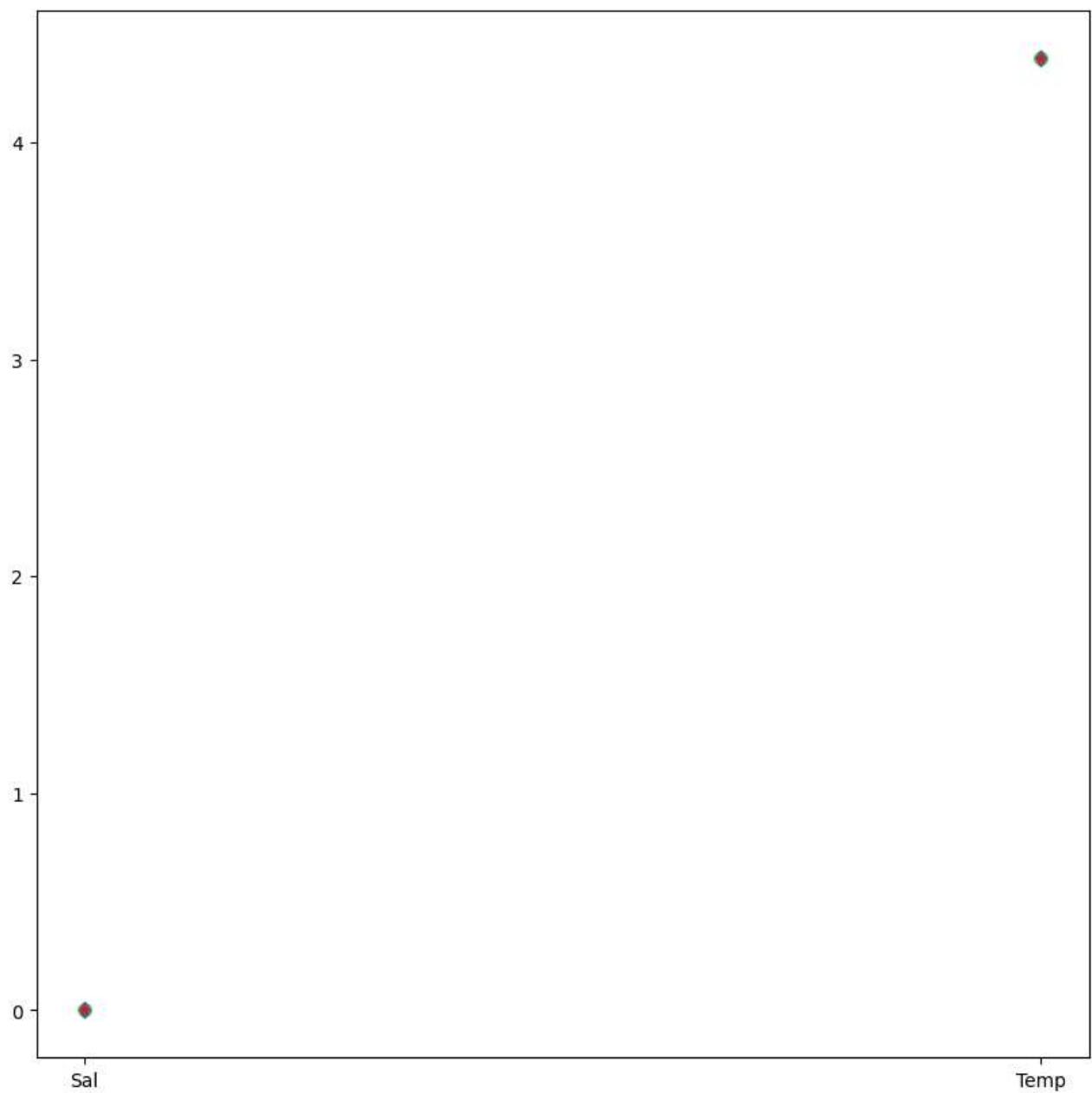
```
In [30]: pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "bar")
```

Out[30]: <Axes: >



```
In [31]: #plot size
plt.figure(figsize = (10, 10))
#add plot for ridge regression
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label='Ridge')
#add plot for lasso regression
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label='Lasso')
#add plot for linear model
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear')

Out[31]: [<matplotlib.lines.Line2D at 0x22e28b657d0>]
```



Elastic Net

```
In [32]: from sklearn.linear_model import ElasticNet
regr=ElasticNet()
regr.fit(X,y)
print(regr.coef_)
print(regr.intercept_)

[0.          0.94934511]
0.5401219631063316
```

```
In [33]: y_pred_elastic=regr.predict(X_train)
```

```
In [34]: 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 114.40984808660129

dataset-2

```
In [35]: #vehicles dataset
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing,svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.linear_model import Ridge
from sklearn.preprocessing import StandardScaler

d=pd.read_csv(r"C:\Users\DELL\Downloads\fiat500_VehicleSelection_Dataset1.xlsx")
d
```

Out[35]:

| | ID | model | engine_power | age_in_days | km | previous_owners | lat | lon | price |
|------|------|--------|--------------|-------------|--------|-----------------|-----------|-----------|-------|
| 0 | 1 | lounge | 51 | 882 | 25000 | 1 | 44.907242 | 8.611560 | 8900 |
| 1 | 2 | pop | 51 | 1186 | 32500 | 1 | 45.666359 | 12.241890 | 8800 |
| 2 | 3 | sport | 74 | 4658 | 142228 | 1 | 45.503300 | 11.417840 | 4200 |
| 3 | 4 | lounge | 51 | 2739 | 160000 | 1 | 40.633171 | 17.634609 | 6000 |
| 4 | 5 | pop | 73 | 3074 | 106880 | 1 | 41.903221 | 12.495650 | 5700 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 1533 | 1534 | sport | 51 | 3712 | 115280 | 1 | 45.069679 | 7.704920 | 5200 |
| 1534 | 1535 | lounge | 74 | 3835 | 112000 | 1 | 45.845692 | 8.666870 | 4600 |
| 1535 | 1536 | pop | 51 | 2223 | 60457 | 1 | 45.481541 | 9.413480 | 7500 |
| 1536 | 1537 | lounge | 51 | 2557 | 80750 | 1 | 45.000702 | 7.682270 | 5990 |
| 1537 | 1538 | pop | 51 | 1766 | 54276 | 1 | 40.323410 | 17.568270 | 7900 |

1538 rows × 9 columns

In [36]: `d.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1538 entries, 0 to 1537
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   ID               1538 non-null    int64  
 1   model             1538 non-null    object  
 2   engine_power      1538 non-null    int64  
 3   age_in_days       1538 non-null    int64  
 4   km                1538 non-null    int64  
 5   previous_owners   1538 non-null    int64  
 6   lat               1538 non-null    float64 
 7   lon               1538 non-null    float64 
 8   price              1538 non-null    int64  
dtypes: float64(2), int64(6), object(1)
memory usage: 108.3+ KB
```

In [37]: `d.describe()`

Out[37]:

| | ID | engine_power | age_in_days | km | previous_owners | lat | lon | price |
|-------|-------------|--------------|-------------|---------------|-----------------|-------------|-------------|--------------|
| count | 1538.000000 | 1538.000000 | 1538.000000 | 1538.000000 | 1538.000000 | 1538.000000 | 1538.000000 | 1538.000000 |
| mean | 769.500000 | 51.904421 | 1650.980494 | 53396.011704 | 1.123537 | 43.541361 | 11.563428 | 8576.003901 |
| std | 444.126671 | 3.988023 | 1289.522278 | 40046.830723 | 0.416423 | 2.133518 | 2.328190 | 1939.958641 |
| min | 1.000000 | 51.000000 | 366.000000 | 1232.000000 | 1.000000 | 36.855839 | 7.245400 | 2500.000000 |
| 25% | 385.250000 | 51.000000 | 670.000000 | 20006.250000 | 1.000000 | 41.802990 | 9.505090 | 7122.500000 |
| 50% | 769.500000 | 51.000000 | 1035.000000 | 39031.000000 | 1.000000 | 44.394096 | 11.869260 | 9000.000000 |
| 75% | 1153.750000 | 51.000000 | 2616.000000 | 79667.750000 | 1.000000 | 45.467960 | 12.769040 | 10000.000000 |
| max | 1538.000000 | 77.000000 | 4658.000000 | 235000.000000 | 4.000000 | 46.795612 | 18.365520 | 11100.000000 |

In [38]: `d.isna().any()`

Out[38]:

| | |
|-----------------|-------|
| ID | False |
| model | False |
| engine_power | False |
| age_in_days | False |
| km | False |
| previous_owners | False |
| lat | False |
| lon | False |
| price | False |

dtype: bool

In [39]: `d.isnull().sum()`

Out[39]:

| | |
|-----------------|---|
| ID | 0 |
| model | 0 |
| engine_power | 0 |
| age_in_days | 0 |
| km | 0 |
| previous_owners | 0 |
| lat | 0 |
| lon | 0 |
| price | 0 |

dtype: int64

In [40]: `d.isnull()`

Out[40]:

| | ID | model | engine_power | age_in_days | km | previous_owners | lat | lon | price |
|------|-------|-------|--------------|-------------|-------|-----------------|-------|-------|-------|
| 0 | False | False | False | False | False | False | False | False | False |
| 1 | False | False | False | False | False | False | False | False | False |
| 2 | False | False | False | False | False | False | False | False | False |
| 3 | False | False | False | False | False | False | False | False | False |
| 4 | False | False | False | False | False | False | False | False | False |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 1533 | False | False | False | False | False | False | False | False | False |
| 1534 | False | False | False | False | False | False | False | False | False |
| 1535 | False | False | False | False | False | False | False | False | False |
| 1536 | False | False | False | False | False | False | False | False | False |
| 1537 | False | False | False | False | False | False | False | False | False |

1538 rows × 9 columns

In [41]: `d.loc[:10, ["ID", "price"]]`

Out[41]:

| | ID | price |
|----|----|-------|
| 0 | 1 | 8900 |
| 1 | 2 | 8800 |
| 2 | 3 | 4200 |
| 3 | 4 | 6000 |
| 4 | 5 | 5700 |
| 5 | 6 | 7900 |
| 6 | 7 | 10750 |
| 7 | 8 | 9190 |
| 8 | 9 | 5600 |
| 9 | 10 | 6000 |
| 10 | 11 | 8950 |

In [42]: `d=d[["engine_power", "price"]]
d.columns=["engine", "price"]`

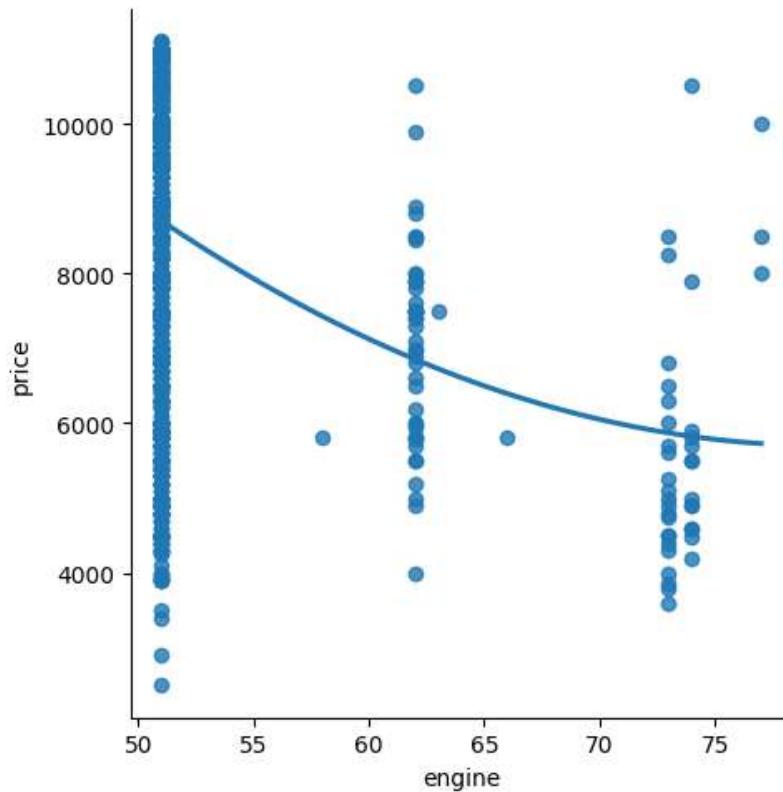
```
In [43]: d.head(10)
```

Out[43]:

| | engine | price |
|---|--------|-------|
| 0 | 51 | 8900 |
| 1 | 51 | 8800 |
| 2 | 74 | 4200 |
| 3 | 51 | 6000 |
| 4 | 73 | 5700 |
| 5 | 74 | 7900 |
| 6 | 51 | 10750 |
| 7 | 51 | 9190 |
| 8 | 73 | 5600 |
| 9 | 51 | 6000 |

```
In [44]: sns.lmplot(x='engine',y='price',data=d,order=2,ci=None)
```

Out[44]: <seaborn.axisgrid.FacetGrid at 0x22e503e0b50>



In [45]: `d.fillna(method='ffill')`

Out[45]:

| | engine | price |
|------|--------|-------|
| 0 | 51 | 8900 |
| 1 | 51 | 8800 |
| 2 | 74 | 4200 |
| 3 | 51 | 6000 |
| 4 | 73 | 5700 |
| ... | ... | ... |
| 1533 | 51 | 5200 |
| 1534 | 74 | 4600 |
| 1535 | 51 | 7500 |
| 1536 | 51 | 5990 |
| 1537 | 51 | 7900 |

1538 rows × 2 columns

In [46]: `x=np.array(d['engine']).reshape(-1,1)`
`y=np.array(d['price']).reshape(-1,1)`

In [47]: `d.dropna(inplace=True)`

C:\Users\DELL\AppData\Local\Temp\ipykernel_7216\1307611603.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

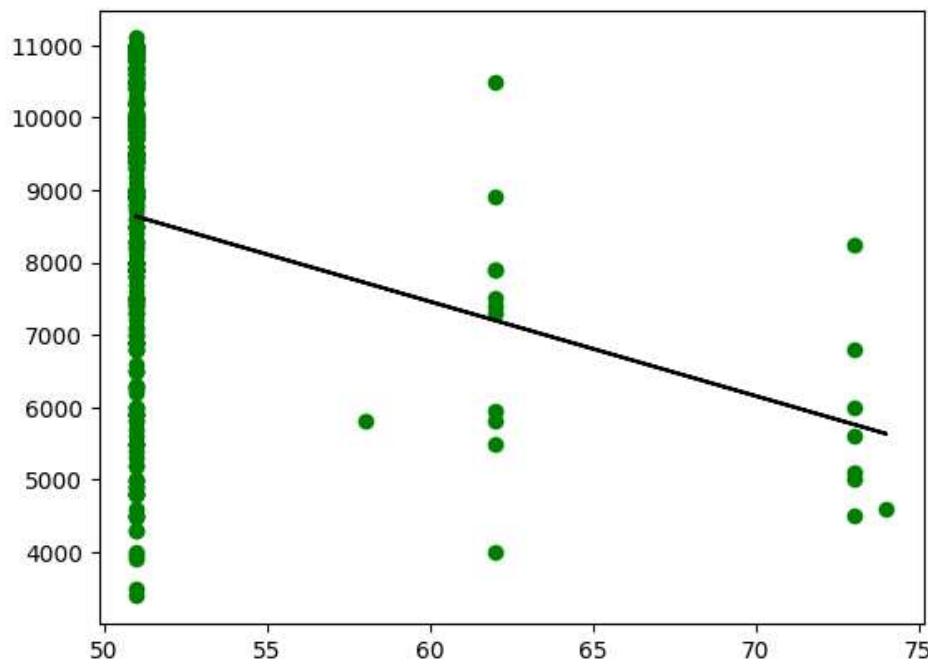
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
`d.dropna(inplace=True)`

In [48]: `x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)`

In [49]: `regr=LinearRegression()`
`regr.fit(x_train,y_train)`
`print(regr.score(x_test,y_test))`

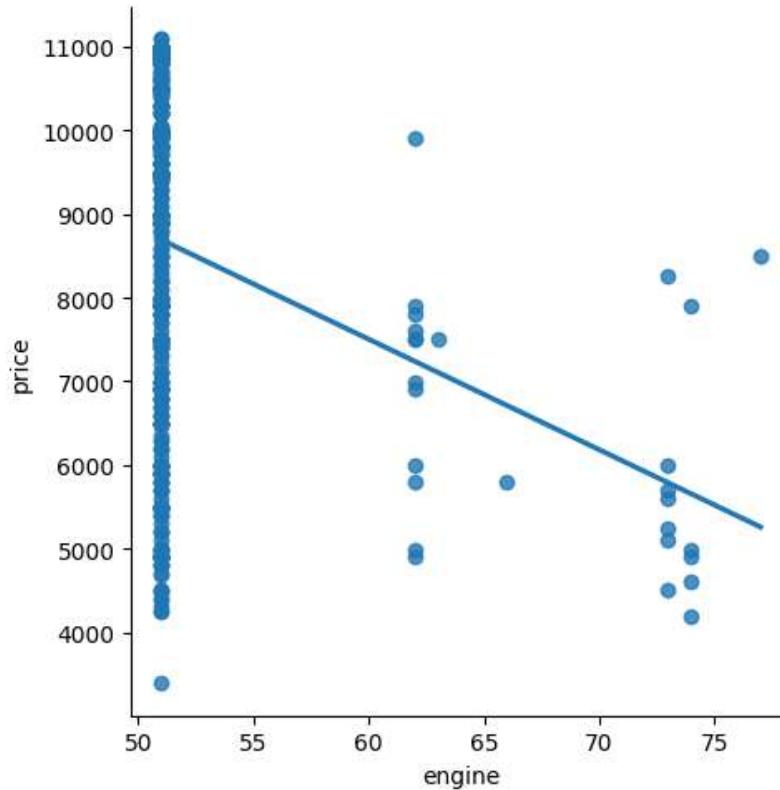
0.06526406815257191

```
In [50]: y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='g')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



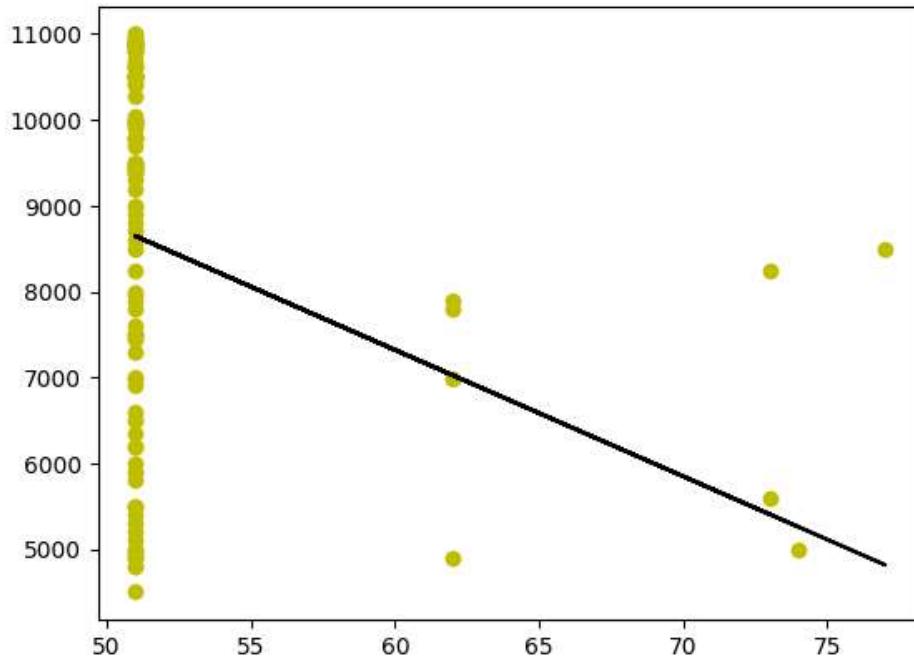
```
In [51]: d500=d[:, :500]
sns.lmplot(x='engine',y='price',data=d500,order=1,ci=None)
```

```
Out[51]: <seaborn.axisgrid.FacetGrid at 0x22e28b65750>
```



```
In [52]: d500.fillna(method='ffill',inplace=True)
x=np.array(d500['engine']).reshape(-1,1)
y=np.array(d500['price']).reshape(-1,1)
d500.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='y')
plt.plot(x_test,y_pred,color='k')
plt.show()
```

Regression: 0.02369782656887498



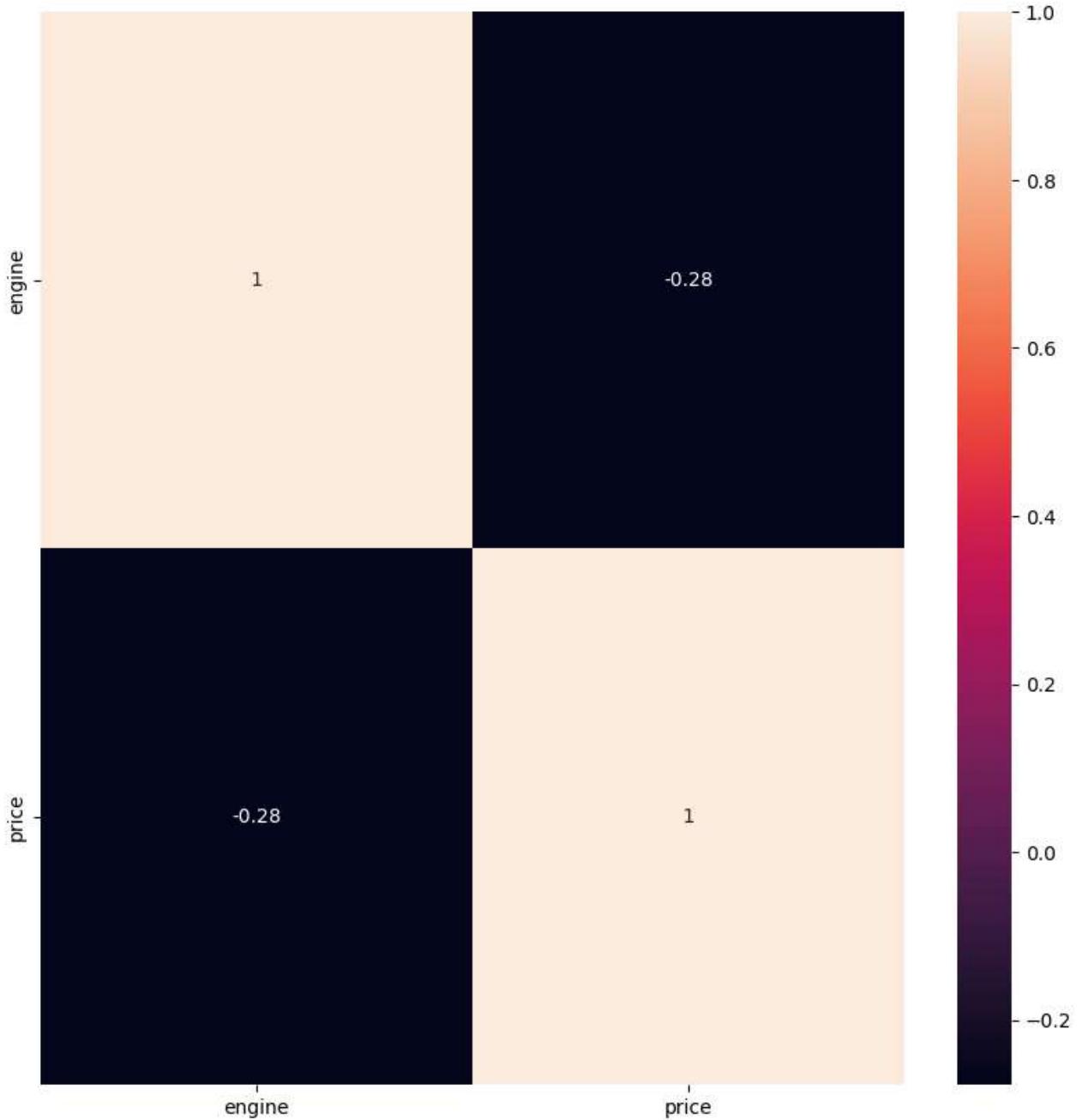
```
In [53]: from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
model=LinearRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
r2=r2_score(y_test,y_pred)
print("R2 score:",r2)
```

R2 score: 0.02369782656887498

Ridge regression and lasso

```
In [54]: plt.figure(figsize = (10, 10))
sns.heatmap(d.corr(), annot = True)
```

```
Out[54]: <Axes: >
```



```
In [55]: features = d.columns[0:2]
target = d.columns[-1]
#X and y values
X = d[features].values
y = d[target].values
#split
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 (1076, 2)
The dimension of X_test is (462, 2)

```
In [56]: 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))
```

Linear Regression Model:

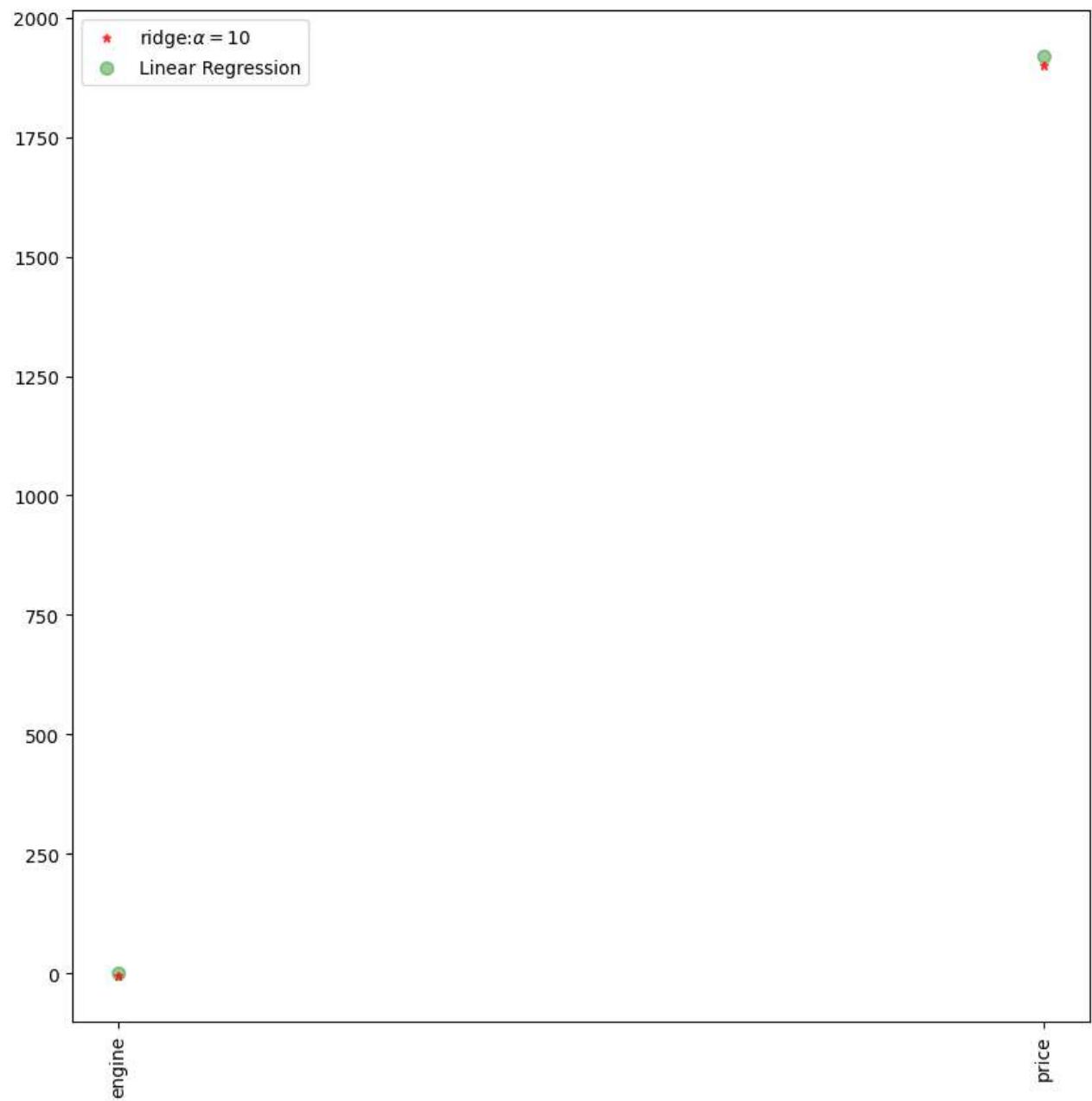
The train score for lr model is 1.0
The test score for lr model is 1.0

```
In [57]: 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 0.9999088581979684
The test score for ridge model is 0.9999100853681022

```
In [58]: plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label='Ridge')
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()
```



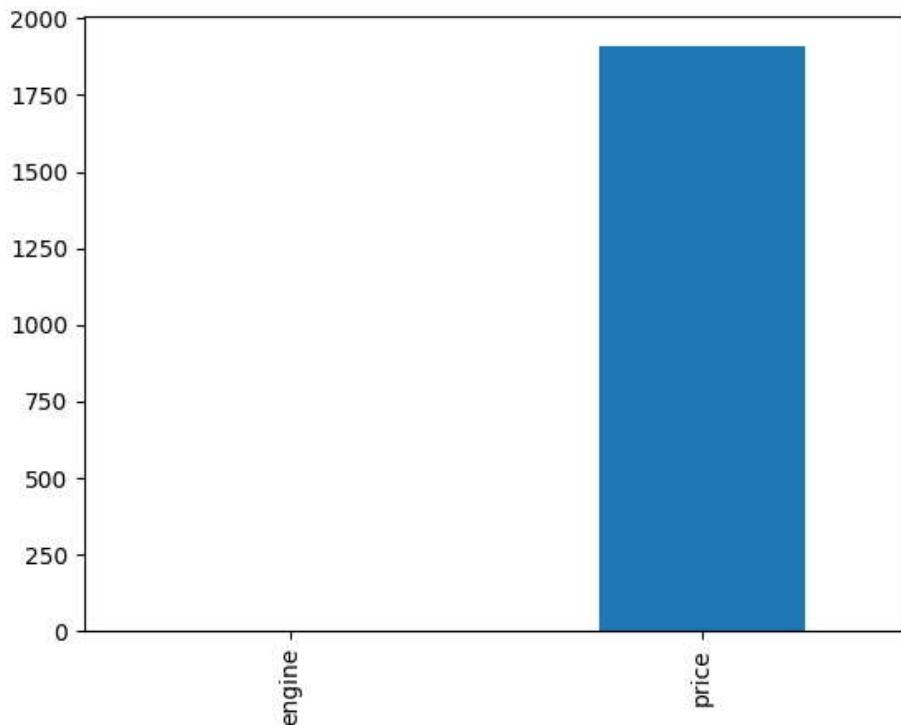
```
In [59]: #Lasso regression model
print("\nLasso Model: \n")
lasso = Lasso(alpha = 10)
lasso.fit(X_train,y_train)
train_score_ls = lasso.score(X_train,y_train)
test_score_ls = lasso.score(X_test,y_test)
print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is {}".format(test_score_ls))
```

Lasso Model:

The train score for ls model is 0.9999728562194999
 The test score for ls model is 0.9999728508562553

```
In [60]: pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "bar")
```

Out[60]: <Axes: >



```
In [61]: 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))
```

0.9999999999501757
 0.9999999999638806

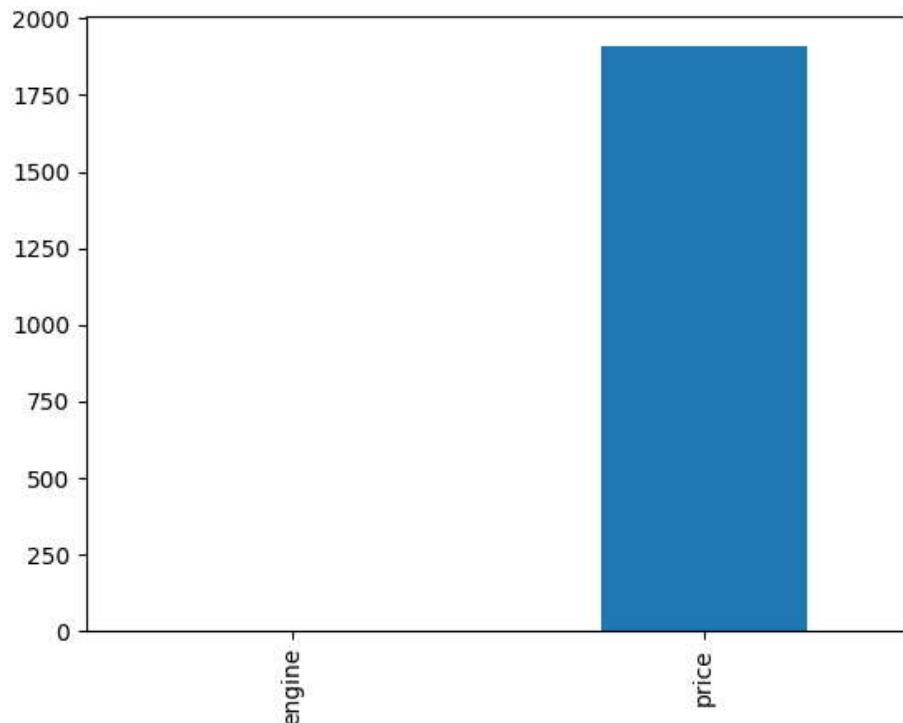
```
In [62]: #Lasso regression model
print("\nLasso Model: \n")
lasso = Lasso(alpha = 10)
lasso.fit(X_train,y_train)
train_score_ls = lasso.score(X_train,y_train)
test_score_ls = lasso.score(X_test,y_test)
print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is {}".format(test_score_ls))
```

Lasso Model:

The train score for ls model is 0.9999728562194999
The test score for ls model is 0.9999728508562553

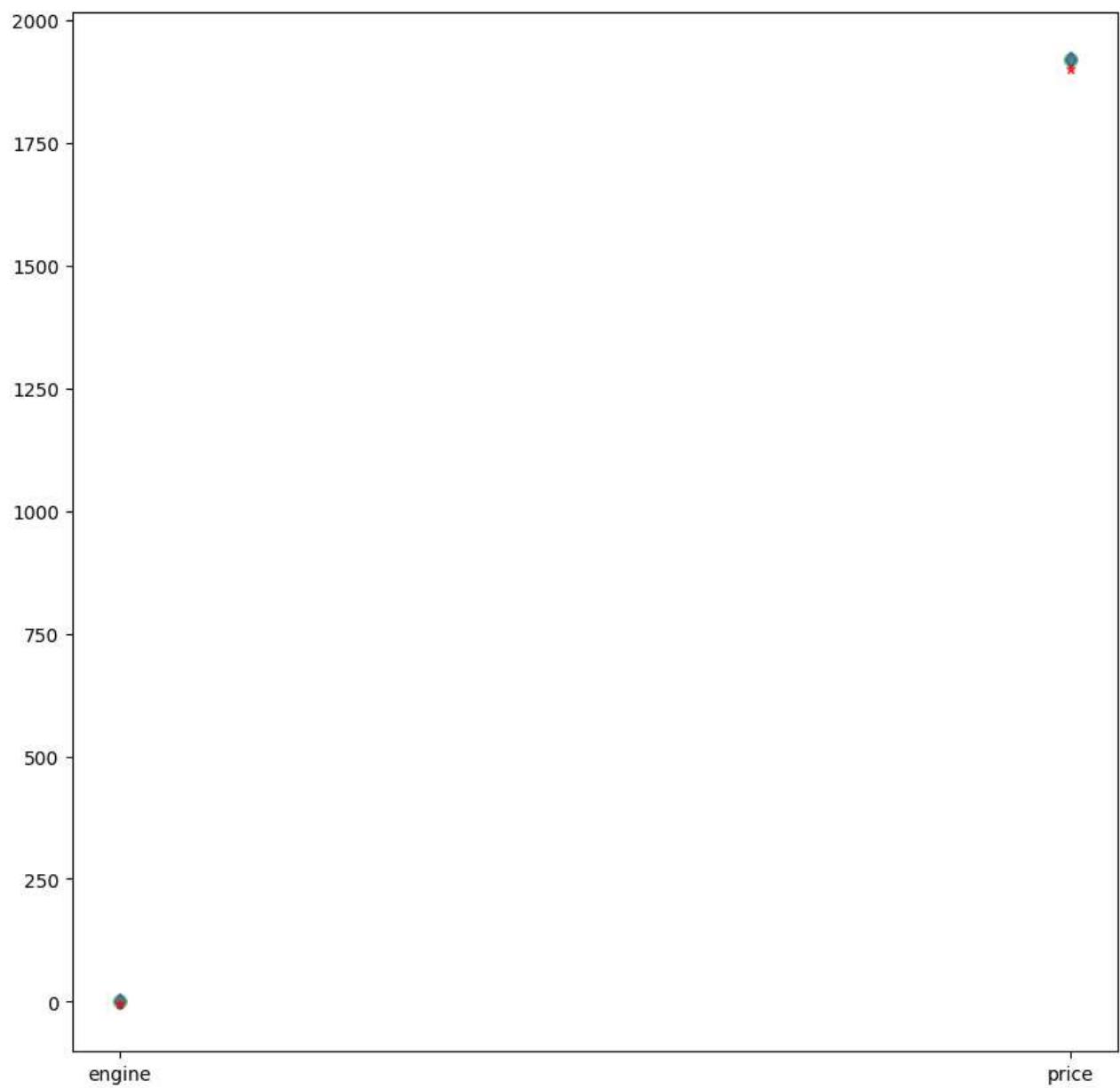
```
In [63]: pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "bar")
```

Out[63]: <Axes: >



```
In [64]: #plot size
plt.figure(figsize = (10, 10))
#add plot for ridge regression
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label='Ridge')
#add plot for lasso regression
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label='Lasso')
#add plot for linear model
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear')

Out[64]: [<matplotlib.lines.Line2D at 0x22e4f51b490>]
```



Elastic Net

```
In [66]: from sklearn.linear_model import ElasticNet
regr=ElasticNet()
regr.fit(X,y)
print(regr.coef_)
print(regr.intercept_)

[-0.          0.99999973]
0.002280249860632466
```

```
In [67]: y_pred_elastic=regr.predict(X_train)
```

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

dataset-3

```
In [68]: #dataset-3
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing,svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.linear_model import Ridge
from sklearn.preprocessing import StandardScaler
ds=pd.read_csv(r"C:\Users\DELL\Downloads\data.csv")
ds
```

Out[68]:

| | date | price | bedrooms | bathrooms | sqft_living | sqft_lot | floors | waterfront | view | condition | sqft_above | sq |
|------|---------------------|--------------|----------|-----------|-------------|----------|--------|------------|------|-----------|------------|------|
| 0 | 2014-05-02 00:00:00 | 3.130000e+05 | 3.0 | 1.50 | 1340 | 7912 | 1.5 | 0 | 0 | 3 | 1340 | 1340 |
| 1 | 2014-05-02 00:00:00 | 2.384000e+06 | 5.0 | 2.50 | 3650 | 9050 | 2.0 | 0 | 4 | 5 | 3370 | 3370 |
| 2 | 2014-05-02 00:00:00 | 3.420000e+05 | 3.0 | 2.00 | 1930 | 11947 | 1.0 | 0 | 0 | 4 | 1930 | 1930 |
| 3 | 2014-05-02 00:00:00 | 4.200000e+05 | 3.0 | 2.25 | 2000 | 8030 | 1.0 | 0 | 0 | 4 | 1000 | 1000 |
| 4 | 2014-05-02 00:00:00 | 5.500000e+05 | 4.0 | 2.50 | 1940 | 10500 | 1.0 | 0 | 0 | 4 | 1140 | 1140 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 4595 | 2014-07-09 00:00:00 | 3.081667e+05 | 3.0 | 1.75 | 1510 | 6360 | 1.0 | 0 | 0 | 4 | 1510 | 1510 |
| 4596 | 2014-07-09 00:00:00 | 5.343333e+05 | 3.0 | 2.50 | 1460 | 7573 | 2.0 | 0 | 0 | 3 | 1460 | 1460 |
| 4597 | 2014-07-09 00:00:00 | 4.169042e+05 | 3.0 | 2.50 | 3010 | 7014 | 2.0 | 0 | 0 | 3 | 3010 | 3010 |
| 4598 | 2014-07-10 00:00:00 | 2.034000e+05 | 4.0 | 2.00 | 2090 | 6630 | 1.0 | 0 | 0 | 3 | 1070 | 1070 |
| 4599 | 2014-07-10 00:00:00 | 2.206000e+05 | 3.0 | 2.50 | 1490 | 8102 | 2.0 | 0 | 0 | 4 | 1490 | 1490 |

4600 rows × 18 columns

In [69]: `ds.describe()`

Out[69]:

| | price | bedrooms | bathrooms | sqft_living | sqft_lot | floors | waterfront | view | cond |
|--------------|--------------|-------------|-------------|--------------|--------------|-------------|-------------|-------------|-------------|
| count | 4.600000e+03 | 4600.000000 | 4600.000000 | 4600.000000 | 4.600000e+03 | 4600.000000 | 4600.000000 | 4600.000000 | 4600.000000 |
| mean | 5.519630e+05 | 3.400870 | 2.160815 | 2139.346957 | 1.485252e+04 | 1.512065 | 0.007174 | 0.240652 | 3.45 |
| std | 5.638347e+05 | 0.908848 | 0.783781 | 963.206916 | 3.588444e+04 | 0.538288 | 0.084404 | 0.778405 | 0.67 |
| min | 0.000000e+00 | 0.000000 | 0.000000 | 370.000000 | 6.380000e+02 | 1.000000 | 0.000000 | 0.000000 | 1.00 |
| 25% | 3.228750e+05 | 3.000000 | 1.750000 | 1460.000000 | 5.000750e+03 | 1.000000 | 0.000000 | 0.000000 | 3.00 |
| 50% | 4.609435e+05 | 3.000000 | 2.250000 | 1980.000000 | 7.683000e+03 | 1.500000 | 0.000000 | 0.000000 | 3.00 |
| 75% | 6.549625e+05 | 4.000000 | 2.500000 | 2620.000000 | 1.100125e+04 | 2.000000 | 0.000000 | 0.000000 | 4.00 |
| max | 2.659000e+07 | 9.000000 | 8.000000 | 13540.000000 | 1.074218e+06 | 3.500000 | 1.000000 | 4.000000 | 5.00 |

◀ ▶

In [70]: `ds.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4600 entries, 0 to 4599
Data columns (total 18 columns):
 #   Column           Non-Null Count  Dtype  
 ---  --  
 0   date             4600 non-null   object 
 1   price            4600 non-null   float64
 2   bedrooms          4600 non-null   float64
 3   bathrooms         4600 non-null   float64
 4   sqft_living       4600 non-null   int64  
 5   sqft_lot          4600 non-null   int64  
 6   floors            4600 non-null   float64
 7   waterfront        4600 non-null   int64  
 8   view              4600 non-null   int64  
 9   condition         4600 non-null   int64  
 10  sqft_above        4600 non-null   int64  
 11  sqft_basement    4600 non-null   int64  
 12  yr_built          4600 non-null   int64  
 13  yr_renovated     4600 non-null   int64  
 14  street             4600 non-null   object 
 15  city              4600 non-null   object 
 16  statezip          4600 non-null   object 
 17  country            4600 non-null   object 
dtypes: float64(4), int64(9), object(5)
memory usage: 647.0+ KB
```

In [71]: `ds.isna().any()`

```
Out[71]: date      False
price      False
bedrooms   False
bathrooms  False
sqft_living False
sqft_lot   False
floors     False
waterfront False
view       False
condition  False
sqft_above False
sqft_basement False
yr_built   False
yr_renovated False
street     False
city       False
statezip   False
country    False
dtype: bool
```

In [72]: `ds.isnull().sum()`

Out[72]:

| | date | 0 |
|---------------|------|---|
| price | 0 | |
| bedrooms | 0 | |
| bathrooms | 0 | |
| sqft_living | 0 | |
| sqft_lot | 0 | |
| floors | 0 | |
| waterfront | 0 | |
| view | 0 | |
| condition | 0 | |
| sqft_above | 0 | |
| sqft_basement | 0 | |
| yr_built | 0 | |
| yr_renovated | 0 | |
| street | 0 | |
| city | 0 | |
| statezip | 0 | |
| country | 0 | |
| dtype: int64 | | |

In [73]: `ds.loc[:10, ["price", "floors"]]`

Out[73]:

| | price | floors |
|----|-----------|--------|
| 0 | 313000.0 | 1.5 |
| 1 | 2384000.0 | 2.0 |
| 2 | 342000.0 | 1.0 |
| 3 | 420000.0 | 1.0 |
| 4 | 550000.0 | 1.0 |
| 5 | 490000.0 | 1.0 |
| 6 | 335000.0 | 1.0 |
| 7 | 482000.0 | 2.0 |
| 8 | 452500.0 | 1.0 |
| 9 | 640000.0 | 1.5 |
| 10 | 463000.0 | 1.0 |

In [74]: `ds=ds[["price", "floors"]]
ds.columns=["pri", "flo"]`

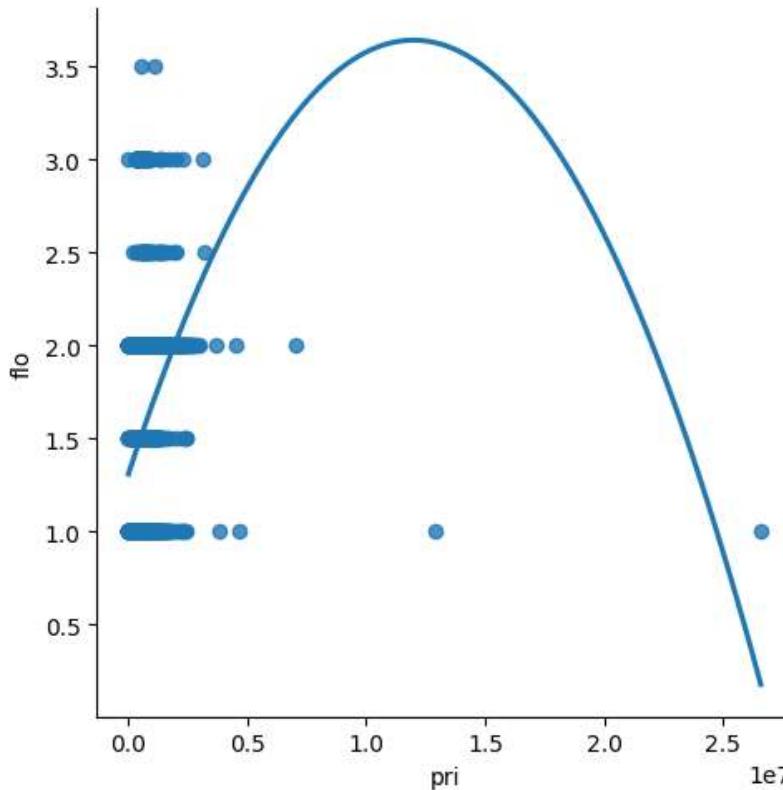
In [75]: `ds.head(10)`

Out[75]:

| | pri | flo |
|---|-----------|-----|
| 0 | 313000.0 | 1.5 |
| 1 | 2384000.0 | 2.0 |
| 2 | 342000.0 | 1.0 |
| 3 | 420000.0 | 1.0 |
| 4 | 550000.0 | 1.0 |
| 5 | 490000.0 | 1.0 |
| 6 | 335000.0 | 1.0 |
| 7 | 482000.0 | 2.0 |
| 8 | 452500.0 | 1.0 |
| 9 | 640000.0 | 1.5 |

```
In [76]: sns.lmplot(x='pri',y='flo',data=ds,order=2,ci=None)
```

```
Out[76]: <seaborn.axisgrid.FacetGrid at 0x22e25ee5e90>
```



```
In [77]: ds.fillna(method='ffill')
```

```
Out[77]:
```

| | pri | flo |
|------|--------------|-----|
| 0 | 3.130000e+05 | 1.5 |
| 1 | 2.384000e+06 | 2.0 |
| 2 | 3.420000e+05 | 1.0 |
| 3 | 4.200000e+05 | 1.0 |
| 4 | 5.500000e+05 | 1.0 |
| ... | ... | ... |
| 4595 | 3.081667e+05 | 1.0 |
| 4596 | 5.343333e+05 | 2.0 |
| 4597 | 4.169042e+05 | 2.0 |
| 4598 | 2.034000e+05 | 1.0 |
| 4599 | 2.206000e+05 | 2.0 |

4600 rows × 2 columns

```
In [78]: x=np.array(ds['pri']).reshape(-1,1)
y=np.array(ds['flo']).reshape(-1,1)
```

```
In [79]: ds.dropna(inplace=True)
```

C:\Users\DELL\AppData\Local\Temp\ipykernel_7216\2725967003.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

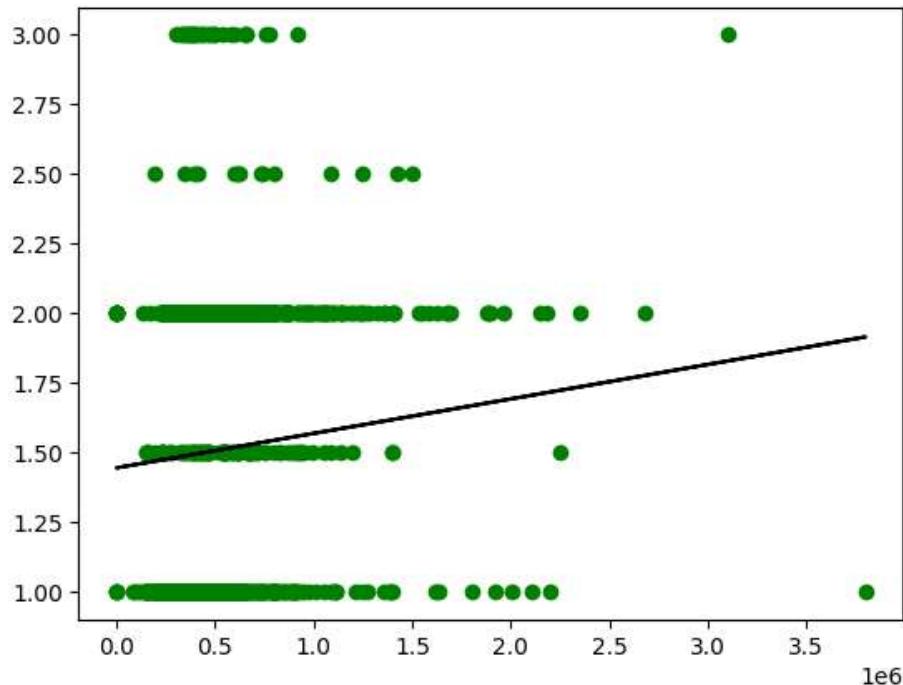
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
ds.dropna(inplace=True)

```
In [80]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
```

```
In [81]: regr=LinearRegression()  
regr.fit(x_train,y_train)  
print(regr.score(x_test,y_test))
```

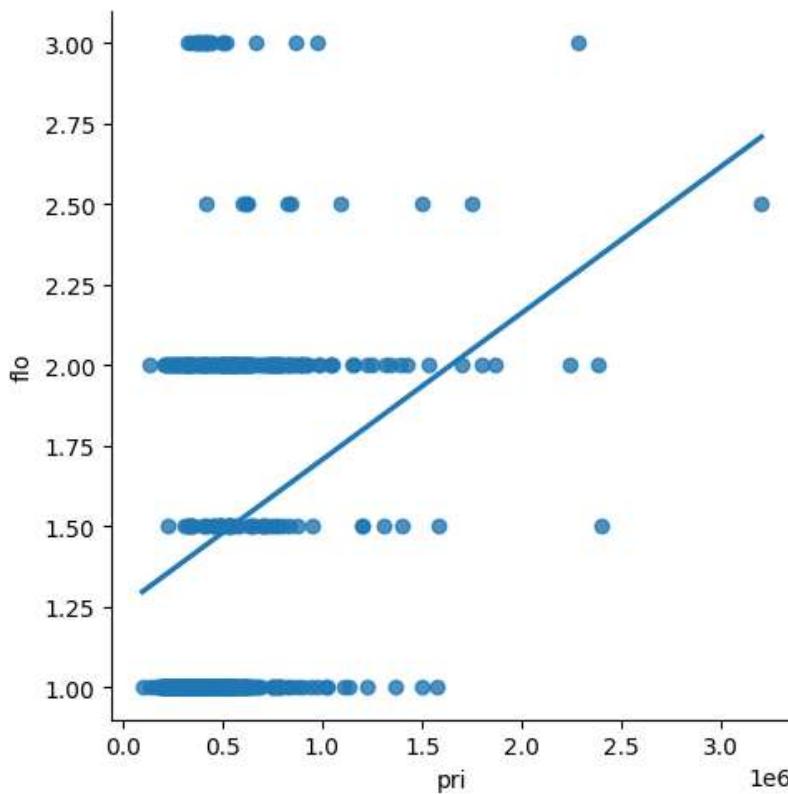
0.02841230483499435

```
In [82]: y_pred=regr.predict(x_test)  
plt.scatter(x_test,y_test,color='g')  
plt.plot(x_test,y_pred,color='k')  
plt.show()
```



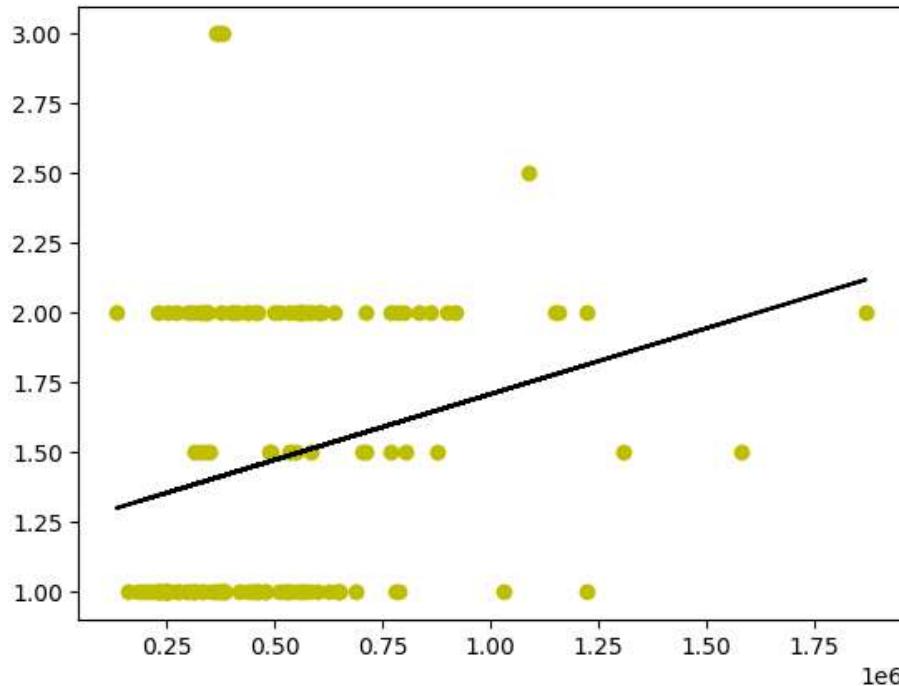
```
In [83]: ds500=ds[:, :500]
sns.lmplot(x='pri', y='flo', data=ds500, order=1, ci=None)
```

```
Out[83]: <seaborn.axisgrid.FacetGrid at 0x22e50006910>
```



```
In [84]: ds500.fillna(method='ffill',inplace=True)
x=np.array(ds500['pri']).reshape(-1,1)
y=np.array(ds500['flo']).reshape(-1,1)
ds500.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='y')
plt.plot(x_test,y_pred,color='k')
plt.show()
```

Regression: 0.03942436503719848



```
In [85]: 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)
```

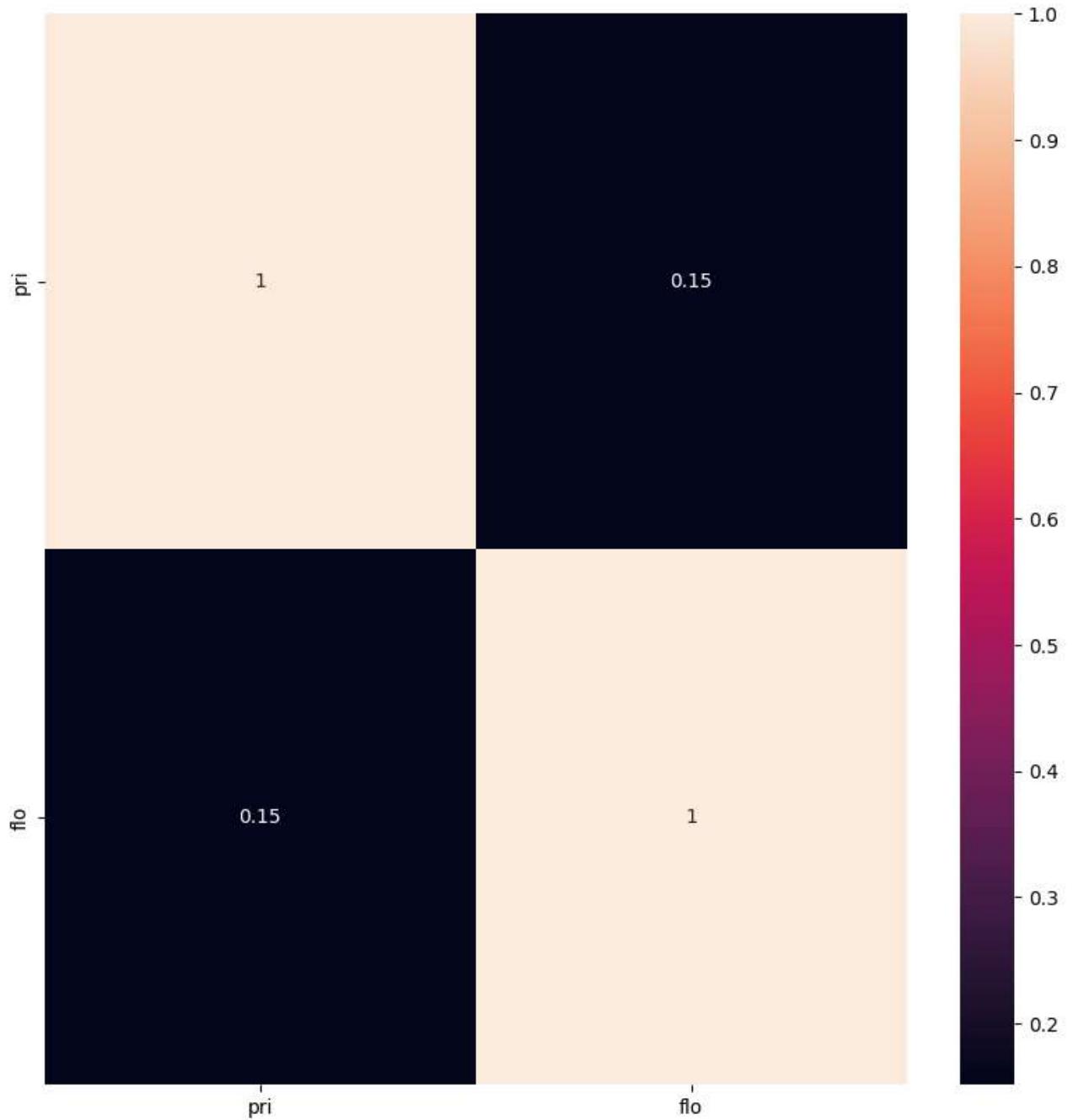
R2 score: 0.03942436503719848

```
In [86]: #conclusion:the model is best fit for linear regression
```

Ridge and lasso

```
In [87]: plt.figure(figsize = (10, 10))
sns.heatmap(ds.corr(), annot = True)
```

```
Out[87]: <Axes: >
```



```
In [88]: features = ds.columns[0:2]
target = ds.columns[-1]
#X and y values
X = ds[features].values
y = ds[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 (3220, 2)
The dimension of X_test is (1380, 2)

```
In [89]: 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))
```

Linear Regression Model:

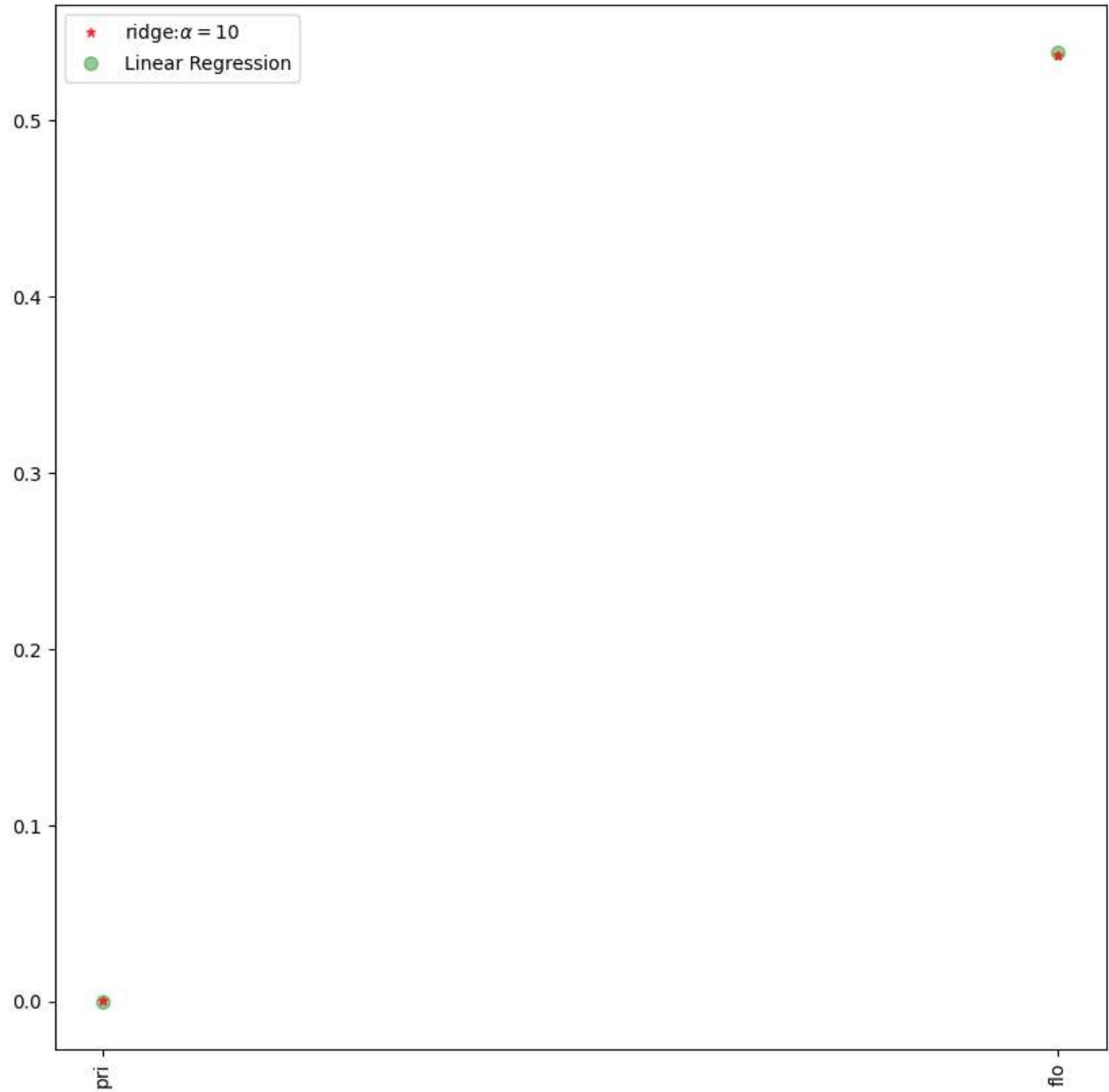
The train score for lr model is 1.0
The test score for lr model is 1.0

```
In [90]: 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 0.9999902268703251
The test score for ridge model is 0.9999903260461295

```
In [91]: plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label='Ridge')
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear')
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



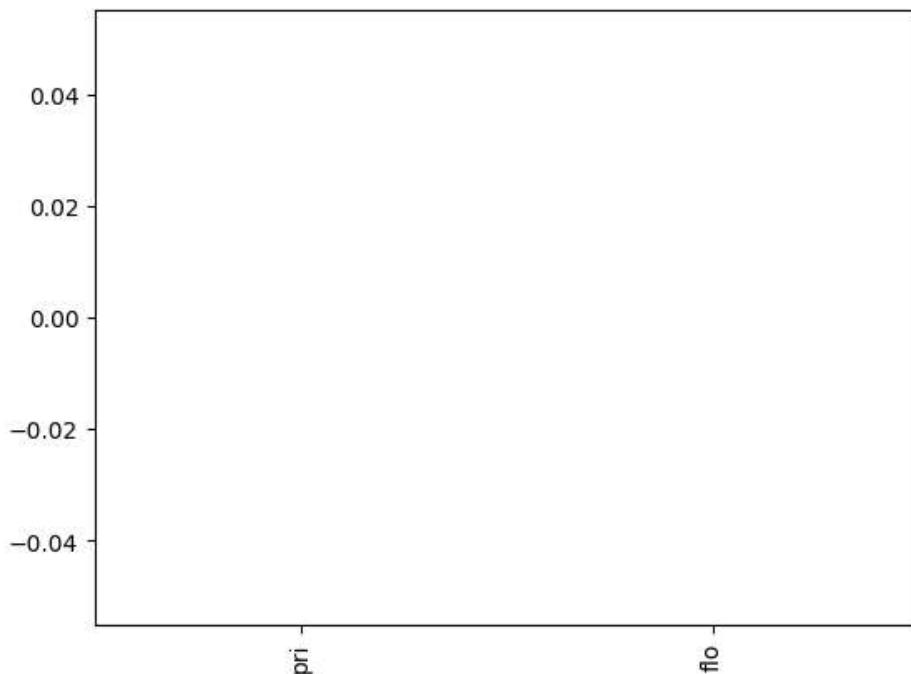
```
In [92]: #Lasso regression model
print("\nLasso Model: \n")
lasso = Lasso(alpha = 10)
lasso.fit(X_train,y_train)
train_score_ls = lasso.score(X_train,y_train)
test_score_ls = lasso.score(X_test,y_test)
print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is {}".format(test_score_ls))
```

Lasso Model:

The train score for ls model is 0.0
The test score for ls model is -0.000324420787036761

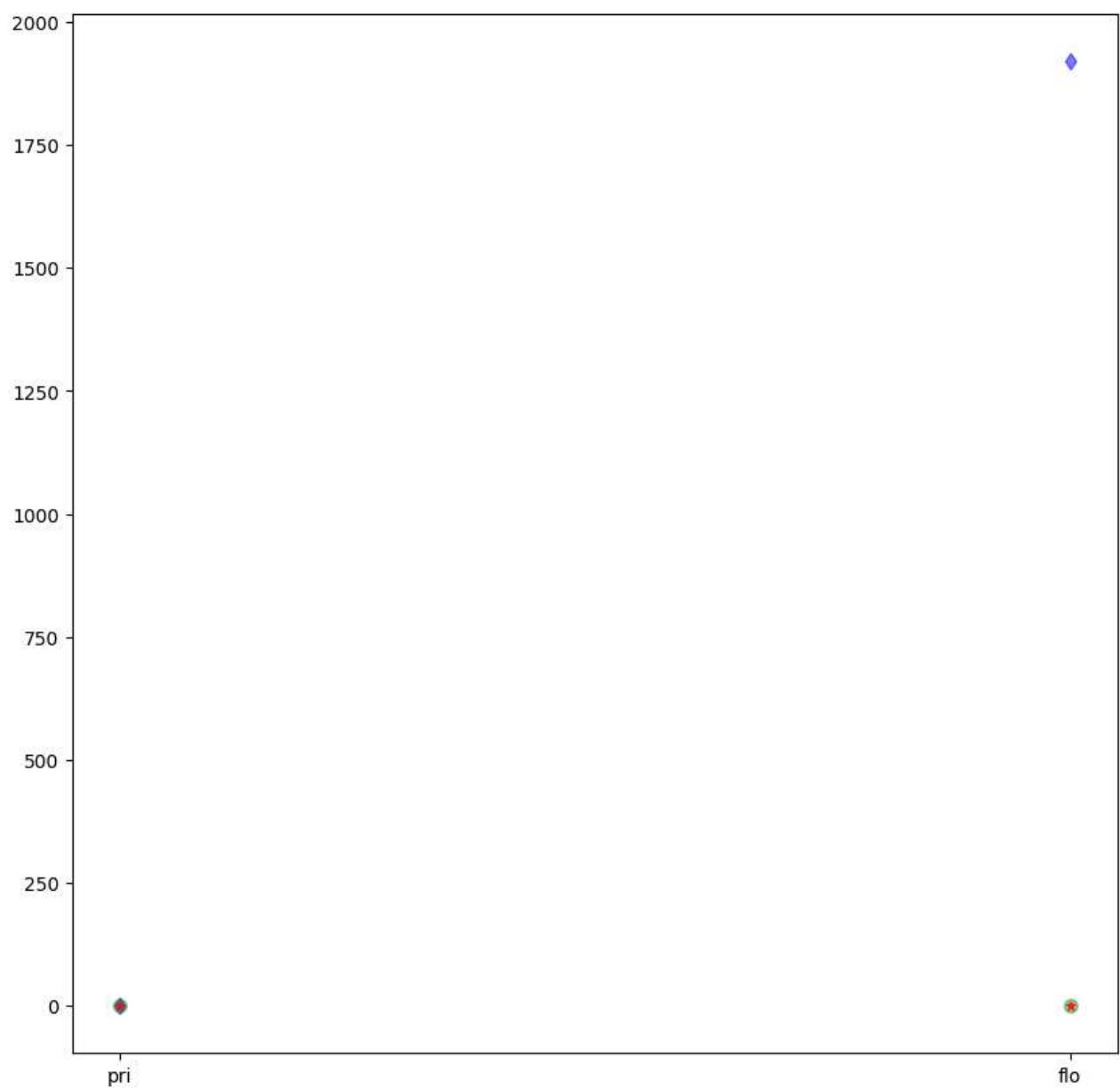
```
In [93]: pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "bar")
```

Out[93]: <Axes: >



```
In [94]: #plot size
plt.figure(figsize = (10, 10))
#add plot for ridge regression
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label='Ridge')
#add plot for lasso regression
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label='Lasso')
#add plot for linear model
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear')

Out[94]: [<matplotlib.lines.Line2D at 0x22e4f853cd0>]
```



Elastic Net

```
In [95]: from sklearn.linear_model import ElasticNet  
regr=ElasticNet()  
regr.fit(X,y)  
print(regr.coef_ )  
print(regr.intercept_ )
```

```
[1.44596814e-07 0.00000000e+00]  
1.432253127854822
```

```
In [96]: y_pred_elastic=regr.predict(X_train)
```

```
In [97]: 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 0.2959706931737603
```