DATE:- 7-6-2023 ____**RELATIONSHIP B/W SALINITY & WATER TEMP** *USING* ***ELASTIC NET***

[2]:
```
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.preprocessing import StandardScaler
```

[4]:
```
df=pd.read_csv(r"/content/bottle.csv")
df
```

[4]:

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	\
0	1	1	054.0 056.0	19-4903CR-HY-060-0930-05400560-0000A-3	
1	1	2	054.0 056.0	19-4903CR-HY-060-0930-05400560-0008A-3	
2	1	3	054.0 056.0	19-4903CR-HY-060-0930-05400560-0010A-7	
3	1	4	054.0 056.0	19-4903CR-HY-060-0930-05400560-0019A-3	
4	1	5	054.0 056.0	19-4903CR-HY-060-0930-05400560-0020A-7	
…	…	…	…	…	
19279	633	19280	102.0 064.0	19-4909CR-HY-255-0424-10200640-0250A-7	
19280	633	19281	102.0 064.0	19-4909CR-HY-255-0424-10200640-0288A-3	
19281	633	19282	102.0 064.0	19-4909CR-HY-255-0424-10200640-0300A-7	
19282	633	19283	102.0 064.0	19-4909CR-HY-255-0424-10200640-0383A-3	
19283	633	19284	102.0 064.0	19-4909CR-HY-255-0424-10200640-0400	

	Depthm	T_degC	Salnty	O2ml_L	STheta	O2Sat	…	R_PHAEO	R_PRES	\
0	0.0	10.50	33.440	NaN	25.649	NaN	…	NaN	0.0	
1	8.0	10.46	33.440	NaN	25.656	NaN	…	NaN	8.0	
2	10.0	10.46	33.437	NaN	25.654	NaN	…	NaN	10.0	
3	19.0	10.45	33.420	NaN	25.643	NaN	…	NaN	19.0	
4	20.0	10.45	33.421	NaN	25.643	NaN	…	NaN	20.0	
…	…	…	…	…	…	…	…	…	…	
19279	250.0	7.82	34.155	1.61	26.641	24.1	…	NaN	251.0	
19280	288.0	7.64	34.270	1.00	26.758	14.9	…	NaN	290.0	
19281	300.0	7.54	34.268	0.93	26.771	13.9	…	NaN	302.0	
19282	383.0	6.77	34.200	0.67	26.825	9.8	…	NaN	385.0	
19283	NaN	NaN	NaN	NaN	NaN	NaN	…	NaN	NaN	

	R_SAMP	DIC1	DIC2	TA1	TA2	pH2	pH1	DIC Quality Comment
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

```
4          NaN   NaN   NaN  NaN  NaN  NaN  NaN                    NaN
…        …   …   …   …   …  …  …                    …
19279      NaN   NaN   NaN  NaN  NaN  NaN  NaN                    NaN
19280      NaN   NaN   NaN  NaN  NaN  NaN  NaN                    NaN
19281      NaN   NaN   NaN  NaN  NaN  NaN  NaN                    NaN
19282      NaN   NaN   NaN  NaN  NaN  NaN  NaN                    NaN
19283      NaN   NaN   NaN  NaN  NaN  NaN  NaN                    NaN

[19284 rows x 74 columns]
```

[5]:
```
df=df[['Salnty','T_degC']]
df.columns=['Sal','Temp']
```

[6]:
```
df.head()
```

[6]:
```
       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
```

[7]:
```
df.info()
```

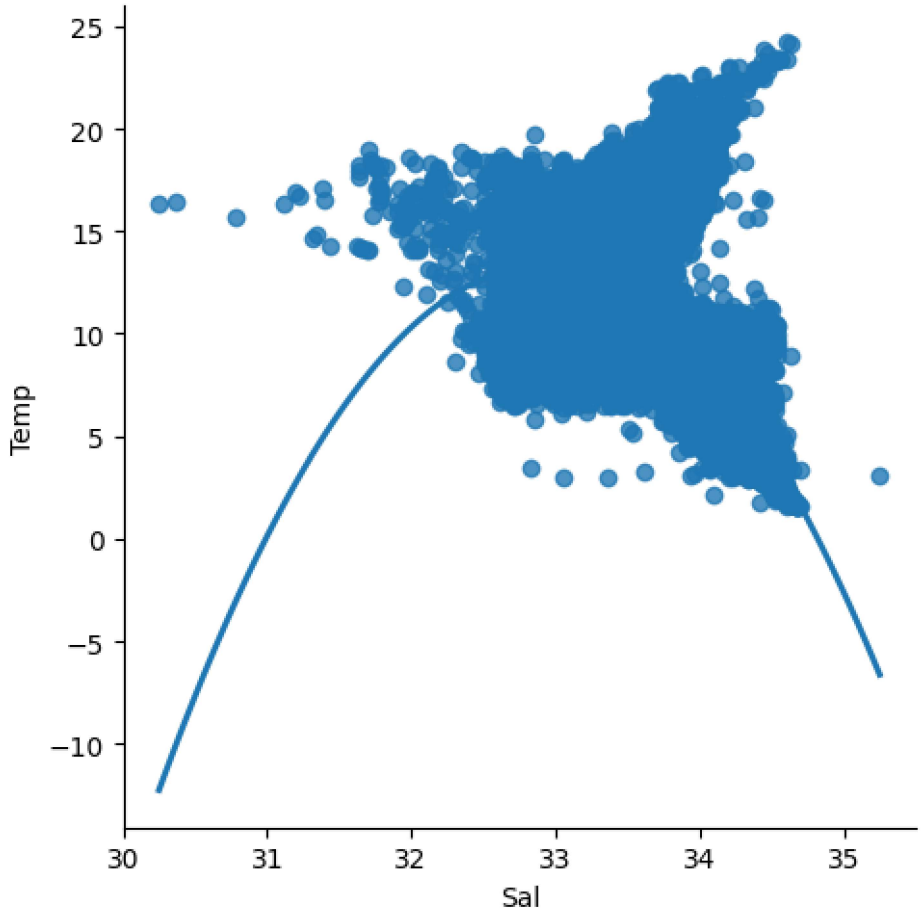
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19284 entries, 0 to 19283
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Sal     18941 non-null  float64
 1   Temp    19178 non-null  float64
dtypes: float64(2)
memory usage: 301.4 KB
```

[8]:
```
df.describe()
```

[8]:
```
                Sal          Temp
count  18941.000000  19178.000000
mean      33.818976      9.109888
std        0.549559      4.523740
min       30.250000      1.540000
25%       33.488000      5.180000
50%       33.933000      8.200000
75%       34.289000     12.480000
max       35.250000     24.200000
```

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

[9]: <seaborn.axisgrid.FacetGrid at 0x7f7044397d60>



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

[10]:

	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
…	…	…
19279	34.155	7.82
19280	34.270	7.64
19281	34.268	7.54
19282	34.200	6.77
19283	34.200	6.77

[19284 rows x 2 columns]

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

```
<ipython-input-11-08528570881a>: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
  df.fillna(value=0,inplace=True)
```

[12]:
```
df.isnull().sum()
```

[12]:
```
Sal     0
Temp    0
dtype: int64
```

[13]:
```
x=np.array(df['Sal']).reshape(-1,1)
y=np.array(df['Temp']).reshape(-1,1)
```

[14]:
```
df.isna().any()
```

[14]:
```
Sal     False
Temp    False
dtype: bool
```

[15]:
```
df.dropna(inplace=True)
```

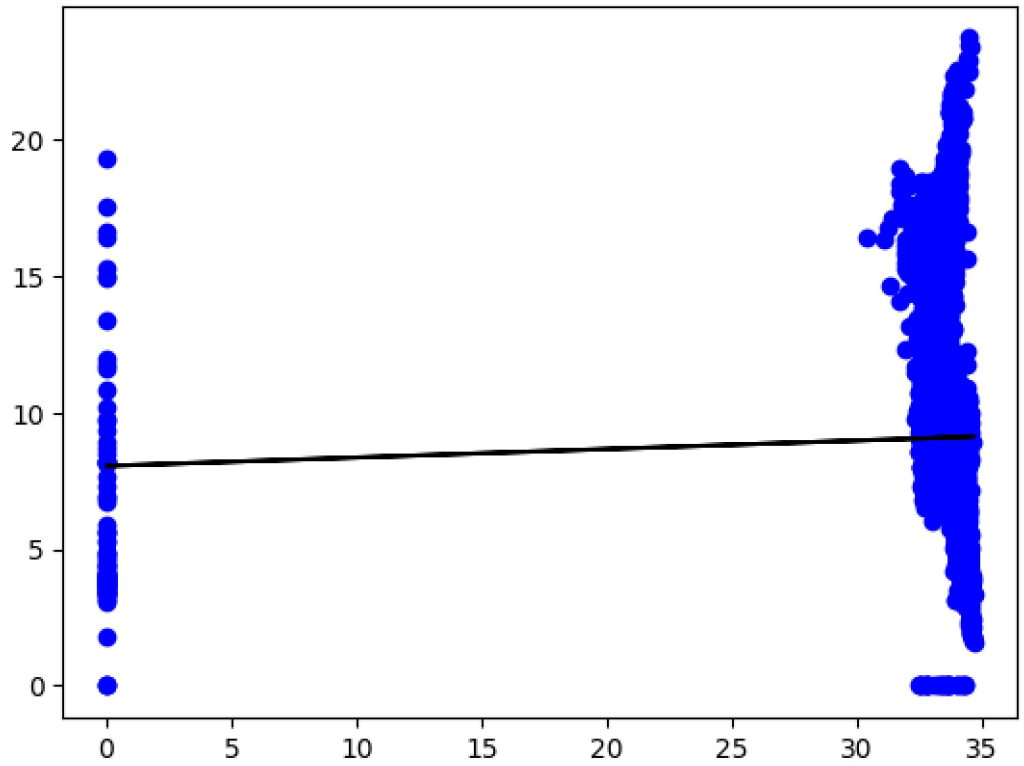
```
<ipython-input-15-c64f9f573c18>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

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

[16]:
```
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))
```

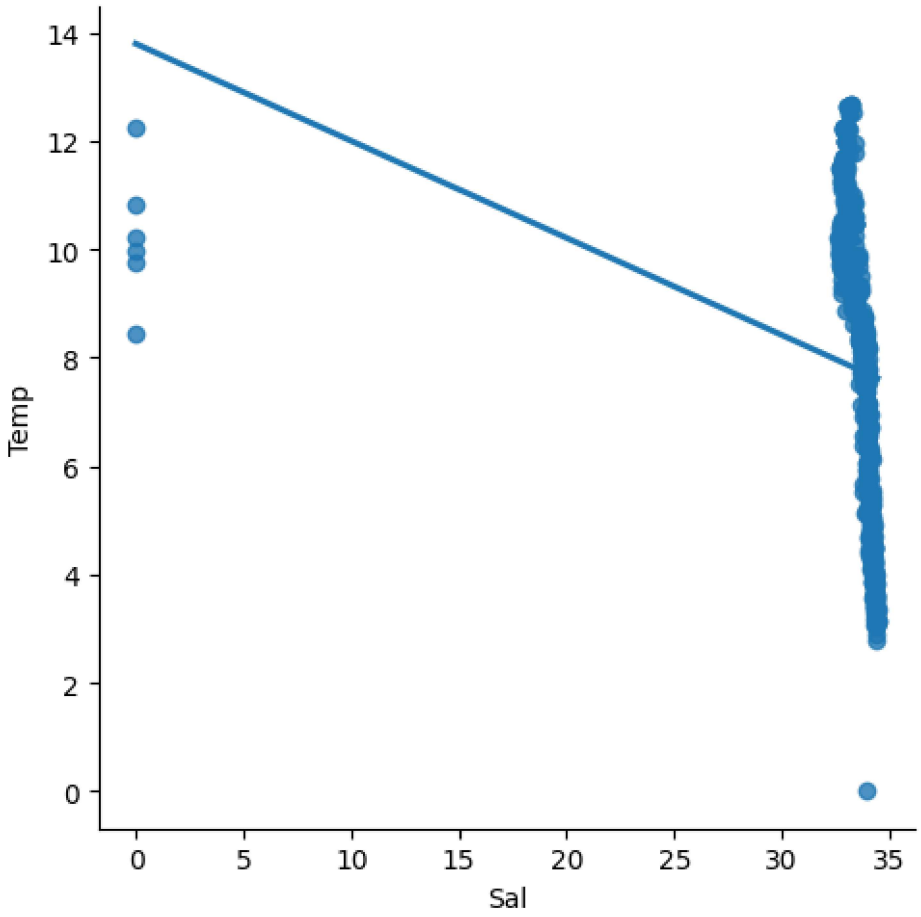
```
-0.000568849449205544
```

[17]:
```
y_pred=reg.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



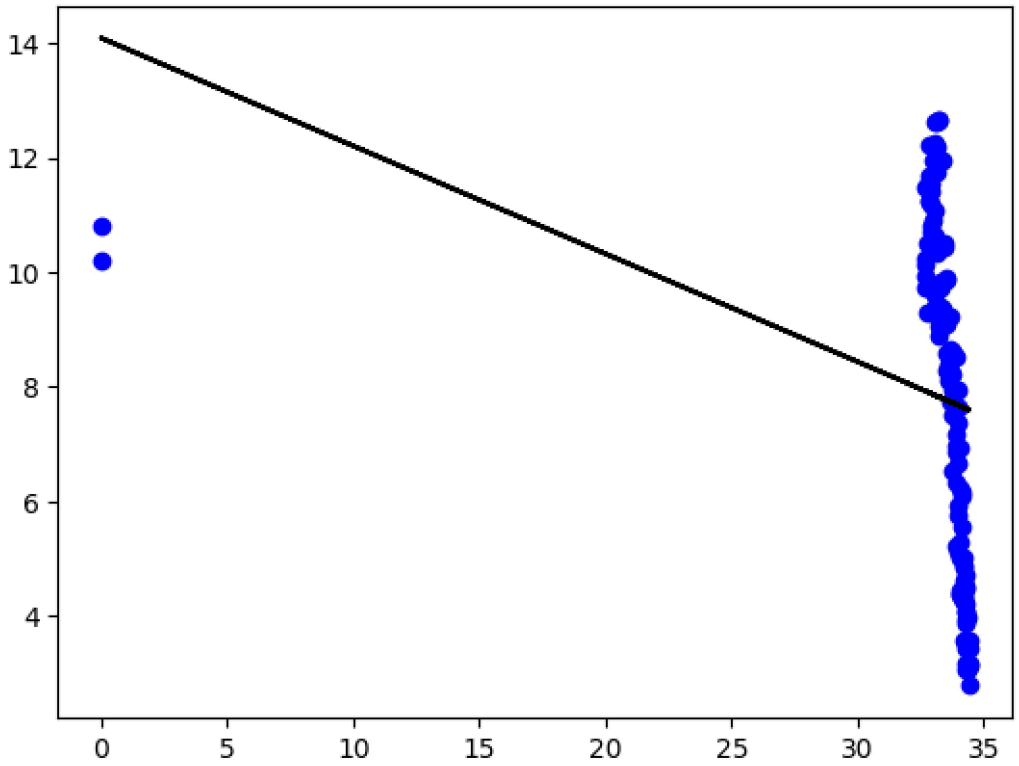
[18]:
```
df500=df[:][:500]
sns.lmplot(x='Sal',y='Temp',data=df500,order=1,ci=None)
```

[18]: <seaborn.axisgrid.FacetGrid at 0x7f704343d840>



[19]:
```
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)
reg=LinearRegression()
reg.fit(x_train,y_train)
print("Regresion:",reg.score(x_test,y_test))
y_pred=reg.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```

```
Regresion: 0.0518585039577697
```



[20]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

[21]:
```
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)
```

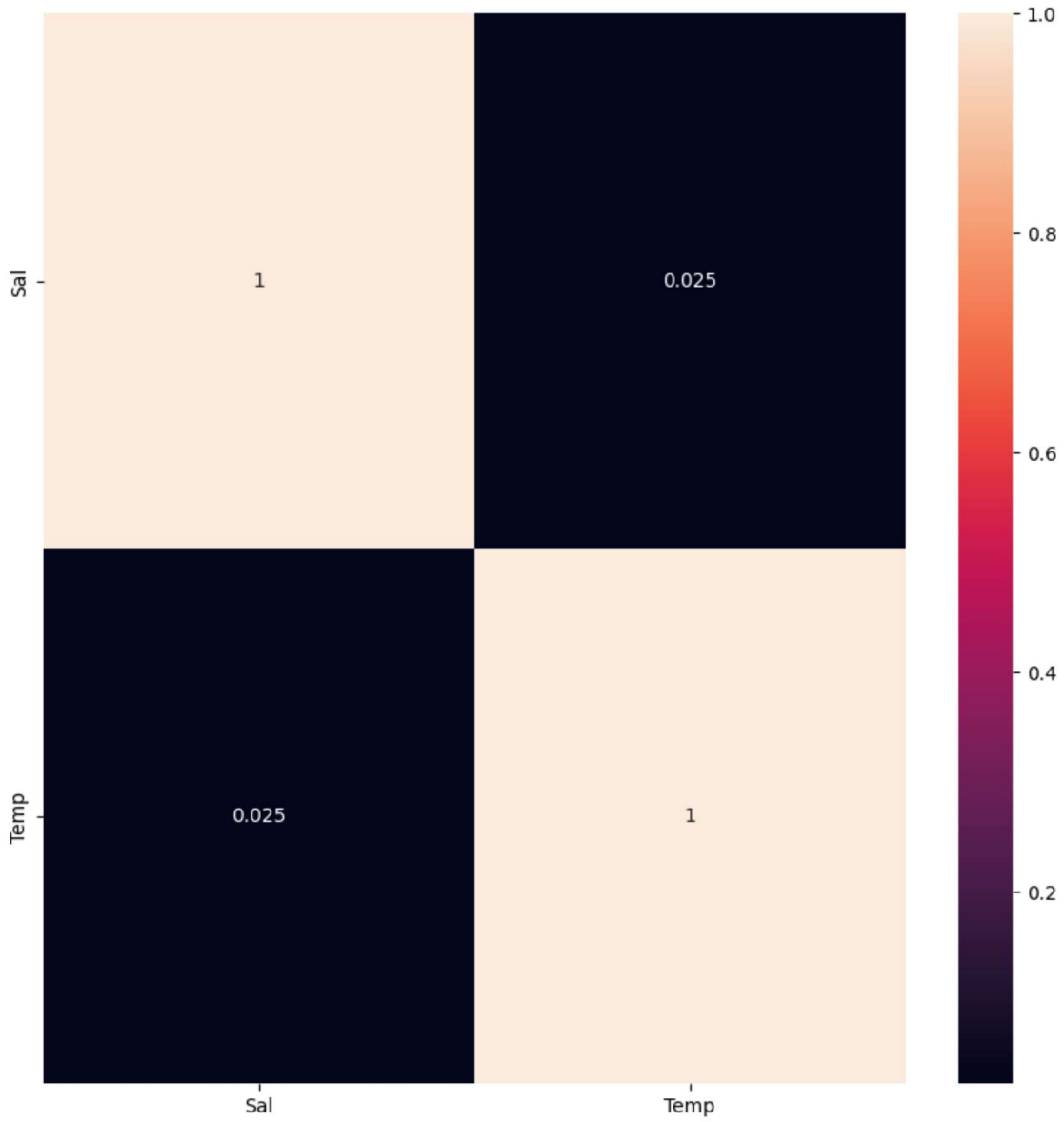
```
R2 score: 0.0518585039577697
```

#-> IMPLEMENTING RIDGE AND LASSO REGRESSION

[22]:
```
from sklearn.linear_model import Ridge
from sklearn.linear_model import RidgeCV
from sklearn.linear_model import Lasso
```

[23]:
```
plt.figure(figsize=(10,10))
sns.heatmap(df.corr(),annot=True)
```

[23]:
```
<Axes: >
```



[31]:
```
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=2)
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 (13498, 2)
The dimension of X_test is (5786, 2)
```

[32]:
```
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
```

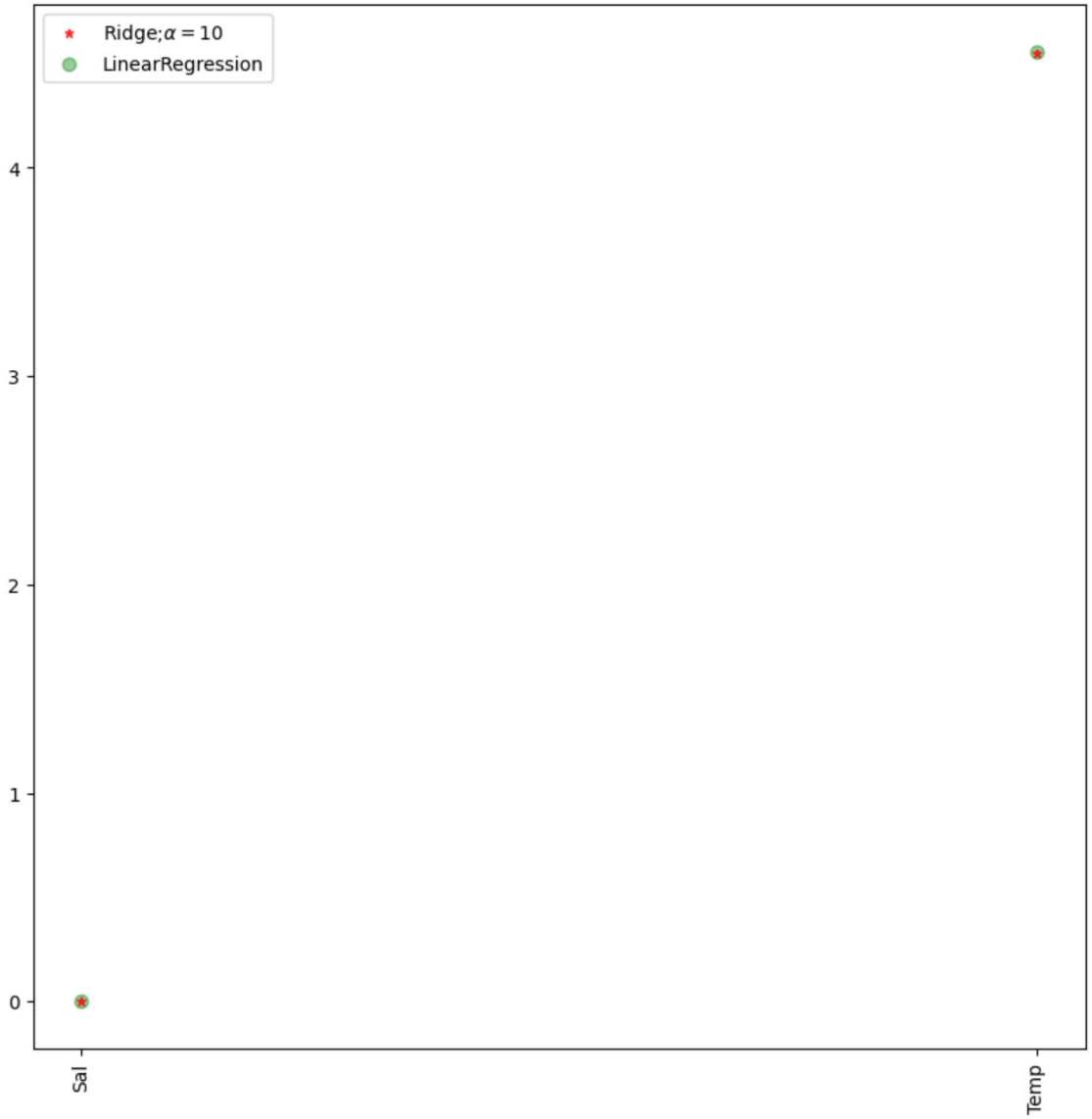
[44]:
```
ridgeReg=Ridge(alpha=10)
ridgeReg.fit(X_train,y_train)
train_score_ridge=ridgeReg.score(X_train,y_train)
test_score_ridge=ridgeReg.score(X_test,y_test)
print("\nRidge Model:\n")
print("The train score for ridge model is {}".format(train_score_ridge))
print("The test score for ridge model is {}".format(test_score_ridge))
```

```
Ridge Model:

The train score for ridge model is 0.9999994518012152
The test score for ridge model is 0.9999994522224434
```

[48]:
```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.
 ↪7,linestyle='none',marker='*',markersize=5,color='red',label=r'Ridge;
 ↪$\alpha=10$',zorder=7)
#plt.plot(rr100.coef_,alpha=0.
 ↪5,linestyle='none',marker='d',markersize=6,color='blue',Label=r'Ridge;
 ↪$\alpha=100$')
```

```
plt.plot(features,lr.coef_,alpha=0.
  ↪4,linestyle='none',marker='o',markersize=7,color='green',label␣
  ↪='LinearRegression')
plt.xticks(rotation=90)
plt.legend()
#plt.title("comparison plot of Ridge,Lasso and Linear regression model")
plt.show()
```



[49]:
```
print("\nLasso Model: \n")
lasso = Lasso(alpha = 10)
lasso.fit(X_train,y_train)
train_score_ls =lasso.score(X_train,y_train)
```

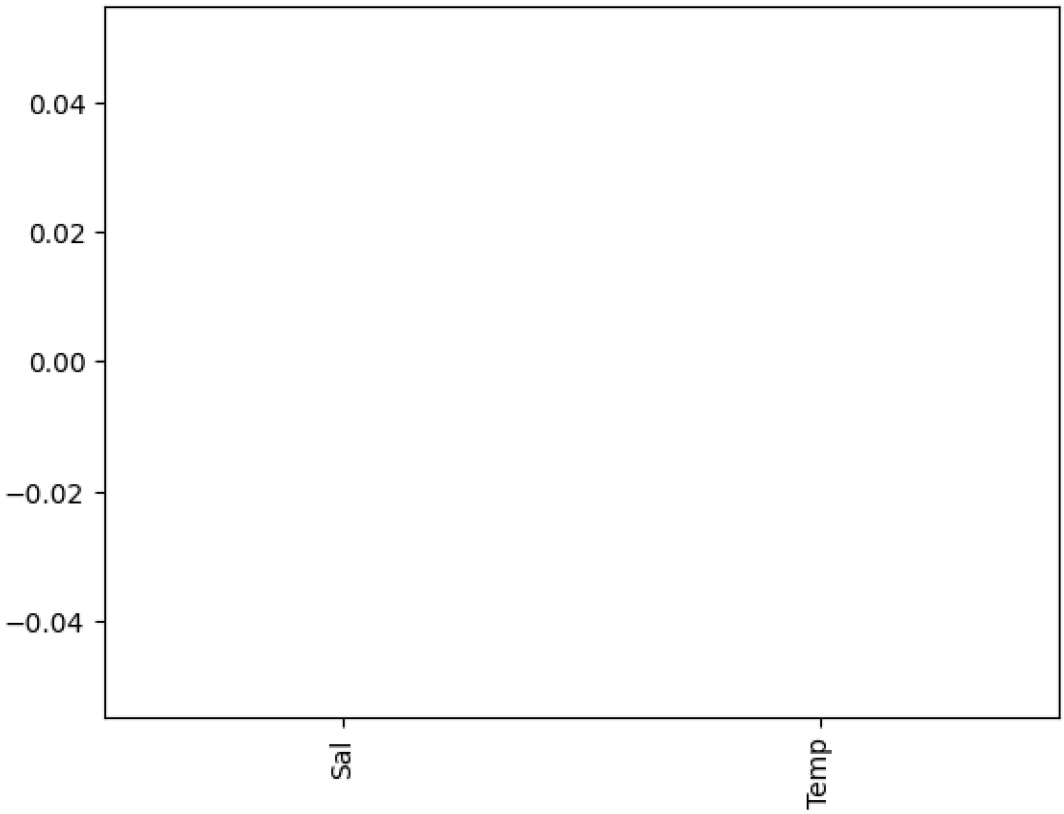
```
test_score_ls =lasso.score(X_test,y_test)
print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is {}".format(test_score_ls))
```

```
Lasso Model:

The train score for ls model is 0.0
The test score for ls model is -0.00017709697264689517
```

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

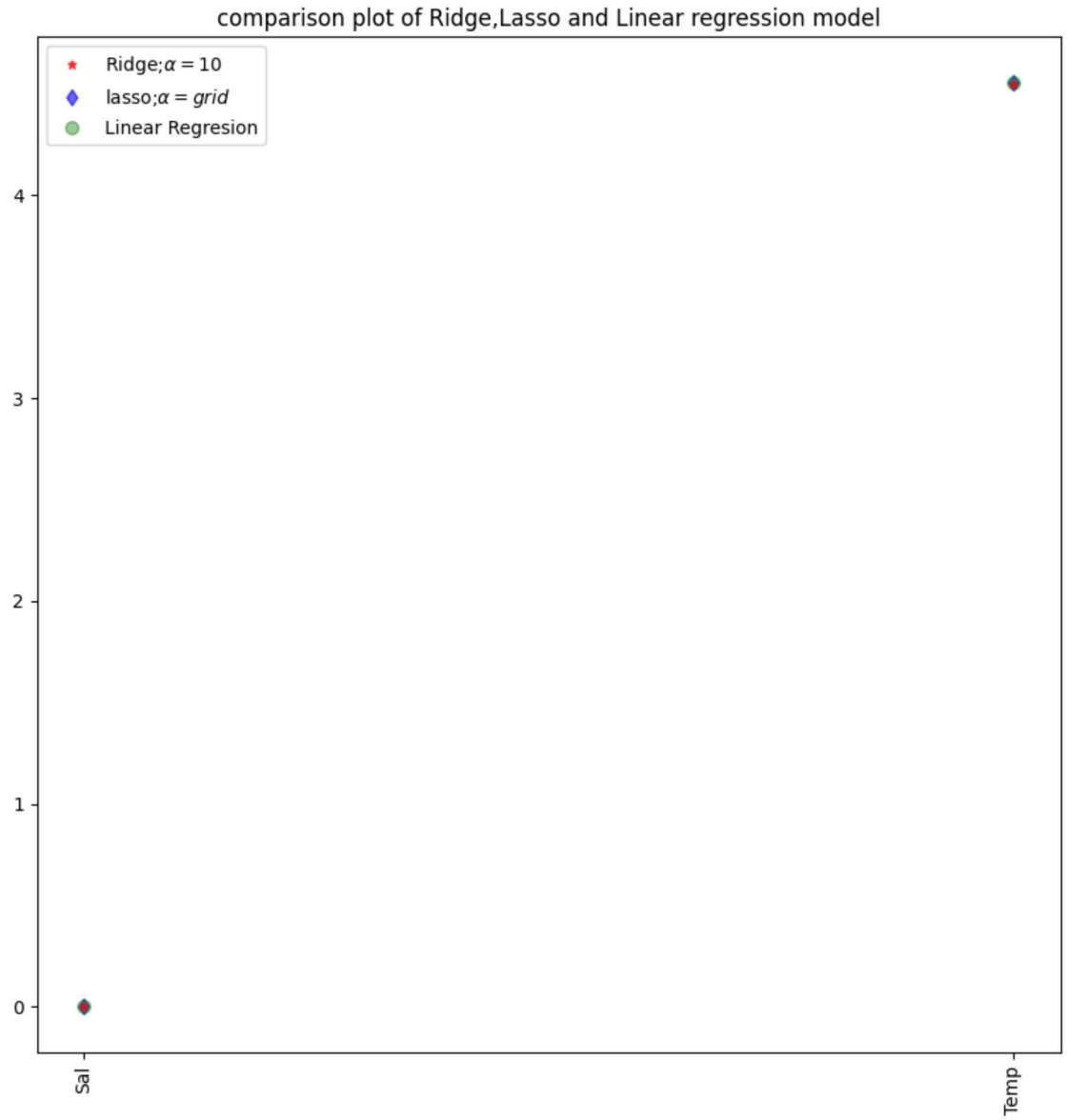
[50]: <Axes: >



[53]:
```
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))
```

```
0.9999999995176235
0.999999999517538
```

[56]:
```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.
  ↪7,linestyle='none',marker='*',markersize=5,color='red',label=r'Ridge;
  ↪$\alpha=10$',zorder=7)
plt.plot(lasso_cv.coef_,alpha=0.
  ↪6,linestyle='none',marker='d',markersize=6,color='blue',label=r'lasso;
  ↪$\alpha=grid$')
plt.plot(features,lr.coef_,alpha=0.
  ↪4,linestyle='none',marker='o',markersize=7,color='green',label='Linear␣
  ↪Regresion')
plt.xticks(rotation=90)
plt.legend()
plt.title("comparison plot of Ridge,Lasso and Linear regression model")
plt.show()
```



[58]:
```
from sklearn.linear_model import RidgeCV
#Ridge Cross validation
ridge_cv = RidgeCV(alphas = [0.0001, 0.001,0.01, 0.1, 1, 10]).fit(X_train,␣
 ↪y_train)
#score
print("The train score for ridge model is {}".format(ridge_cv.score(X_train,␣
 ↪y_train)))
print("The train score for ridge model is {}".format(ridge_cv.score(X_test,␣
 ↪y_test)))
```

```
The train score for ridge model is 0.9999999999999973
```

```
The train score for ridge model is 0.9999999999999973
```

#-> ELASTIC NET REGRESSION

[59]:
```
from sklearn.linear_model import ElasticNet
regr=ElasticNet()
regr.fit(X,y)
print(regr.coef_)
print(regr.intercept_)
```

```
[0.         0.95306122]
0.4252565866429965
```

[60]:
```
y_pred_elastic=regr.predict(X_train)
```

[61]:
```
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 87.83194014034463
```