In [1]:

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

```

In [20]:

```
data=pd.read_csv(r"C:\Users\samit\OneDrive\Desktop\jupyter\bottle.csv")
data
```

```
C:\Users\samit\AppData\Local\Temp\ipykernel_30204\3524639096.py:1: DtypeW
arning: Columns (47,73) have mixed types. Specify dtype option on import
or set low_memory=False.
  data=pd.read_csv(r"C:\Users\samit\OneDrive\Desktop\jupyter\bottle.csv")
```

Out[20]:

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

In [21]:

```
data.head()
```

Out[21]:

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta	O2Sat
0	1	1	054.0 056.0	19-4903CR-HY-060-0930-05400560-0000A-3	0	10.50	33.440	NaN	25.649	NaN
1	1	2	054.0 056.0	19-4903CR-HY-060-0930-05400560-0008A-3	8	10.46	33.440	NaN	25.656	NaN
2	1	3	054.0 056.0	19-4903CR-HY-060-0930-05400560-0010A-7	10	10.46	33.437	NaN	25.654	NaN
3	1	4	054.0 056.0	19-4903CR-HY-060-0930-05400560-0019A-3	19	10.45	33.420	NaN	25.643	NaN
4	1	5	054.0 056.0	19-4903CR-HY-060-0930-05400560-0020A-7	20	10.45	33.421	NaN	25.643	NaN

5 rows × 74 columns

In [4]:

```
dt.tail()
```

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta
864862	34404	864863	093.4 026.4	20-1611SR-MX-310-2239-09340264-0015A-3	15	17.533	33.3880	5.774	24.15297

Out[4]:

864863 rows × 74 columns

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta
864858	34404	864859	093.4 026.4	20-1611SR-MX-310-2239-09340264-0000A-7	0	18.744	33.4083	5.805	23.87055
864859	34404	864860	093.4 026.4	20-1611SR-MX-310-2239-09340264-0002A-3	2	18.744	33.4083	5.805	23.87072
864860	34404	864861	093.4 026.4	20-1611SR-MX-310-2239-09340264-0005A-3	5	18.692	33.4150	5.796	23.88911
864861	34404	864862	093.4 026.4	20-1611SR-MX-310-2239-09340264-0010A-3	10	18.161	33.4062	5.816	24.01426
864862	34404	864863	093.4 026.4	20-1611SR-MX-310-2239-09340264-0015A-3	15	17.533	33.3880	5.774	24.15297

5 rows × 74 columns

In [22]:

```
data=data[['Salnty','T_degC']]
data.columns=['Sal','Temp']
```

In [23]:

```
data.shape
```

Out[23]:

```
(864863, 2)
```

In [24]:

```
data.columns
```

Out[24]:

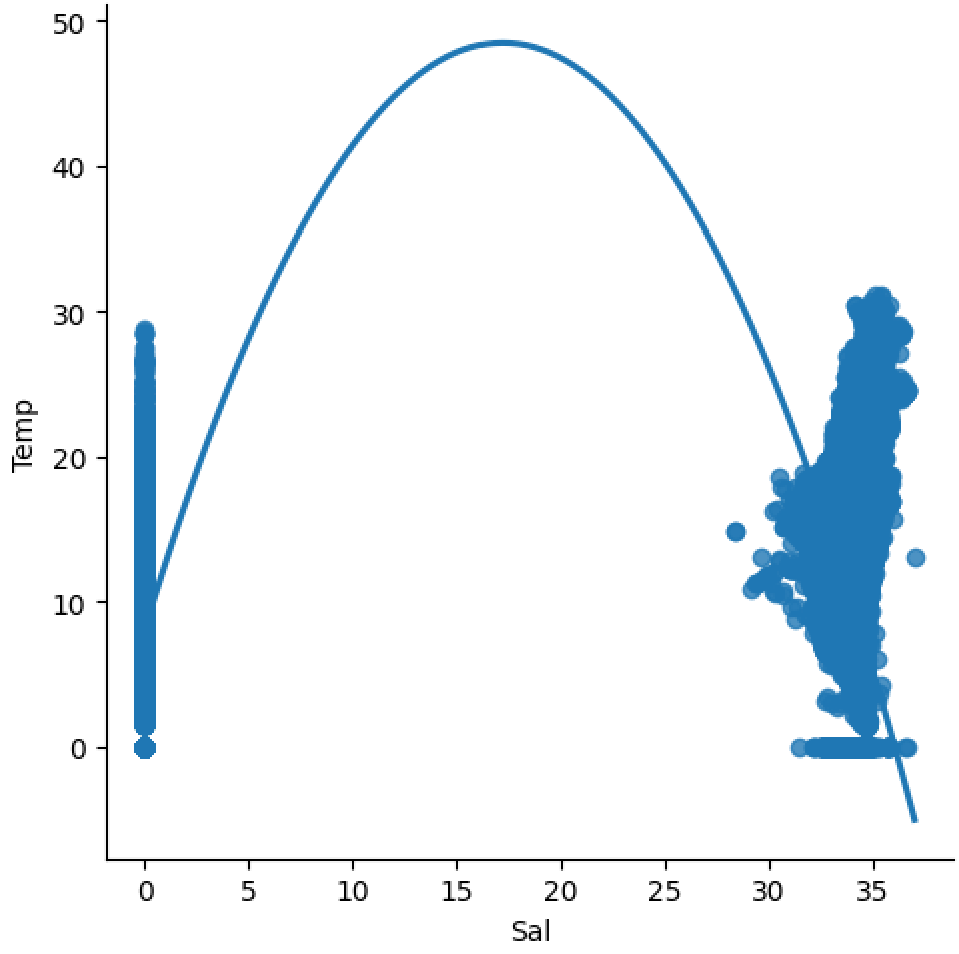
```
Index(['Sal', 'Temp'], dtype='object')
```

In [29]:

```
sns.lmplot(x='Sal',y='Temp',data=dt,order=2,ci=None)

```

Out[29]:

```
<seaborn.axisgrid.FacetGrid at 0x1500f4cb290>
```



In [26]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 864863 entries, 0 to 864862
Data columns (total 2 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Sal     817509 non-null  float64
 1   Temp    853900 non-null  float64
dtypes: float64(2)
memory usage: 13.2 MB
```

In [30]:

```
data.describe()
```

Out[30]:

	Sal	Temp
count	817509.000000	853900.000000
mean	33.840350	10.799677
std	0.461843	4.243825
min	28.431000	1.440000
25%	33.488000	7.680000
50%	33.863000	10.060000
75%	34.196900	13.880000
max	37.034000	31.140000

In [31]:

```
data.fillna(method='ffill')
```

Out[31]:

	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 [32]:

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

```
C:\Users\samit\AppData\Local\Temp\ipykernel_30204\3951631895.py:1: Settin
gWithCopyWarning:
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-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  data.fillna(value=0,inplace=True)
```

In [33]:

```
data.isnull().sum()
```

Out[33]:

```
Sal     0
Temp    0
dtype: int64
```

In [34]:

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

In [35]:

```
data.isna().any()
```

Out[35]:

```
Sal     False
Temp    False
dtype: bool
```

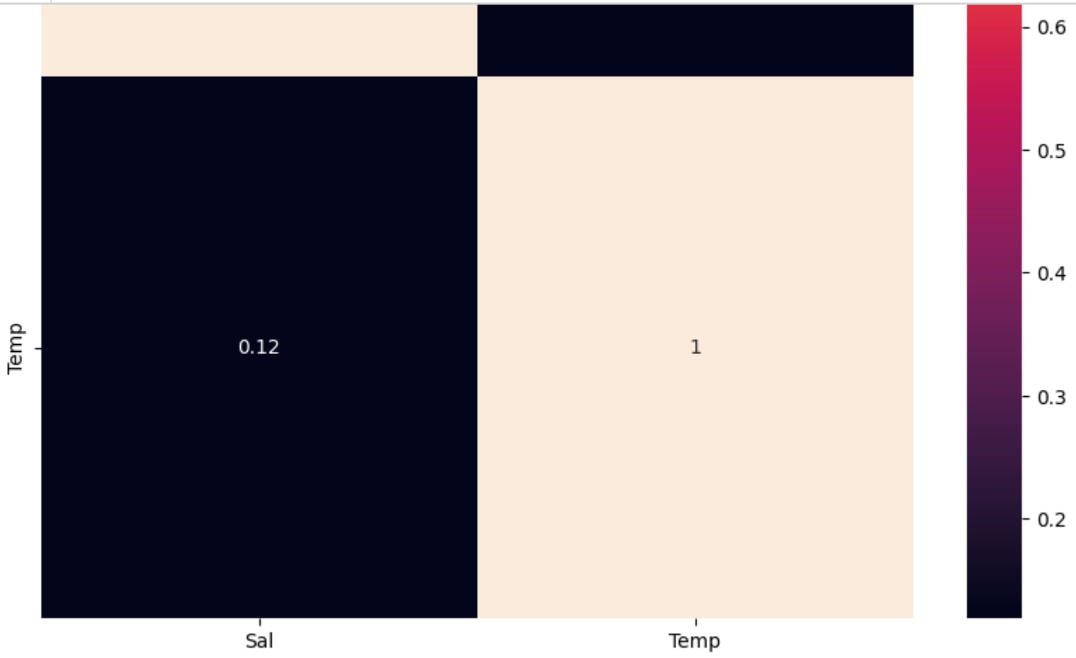
In [36]:

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

```
C:\Users\samit\AppData\Local\Temp\ipykernel_30204\1368182302.py:1: Settin
gWithCopyWarning:
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-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  data.dropna(inplace=True)
```

In [37]:

```
plt.figure(figsize = (10, 10))
sns.heatmap(data.corr(), annot = True)
```



In [38]:

```
features = data.columns[0:2]
target = data.columns[-1]
#X and y values
X = data[features].values
y = data[target].values
#splot
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_sta
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 [39]:

```
#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("\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
```

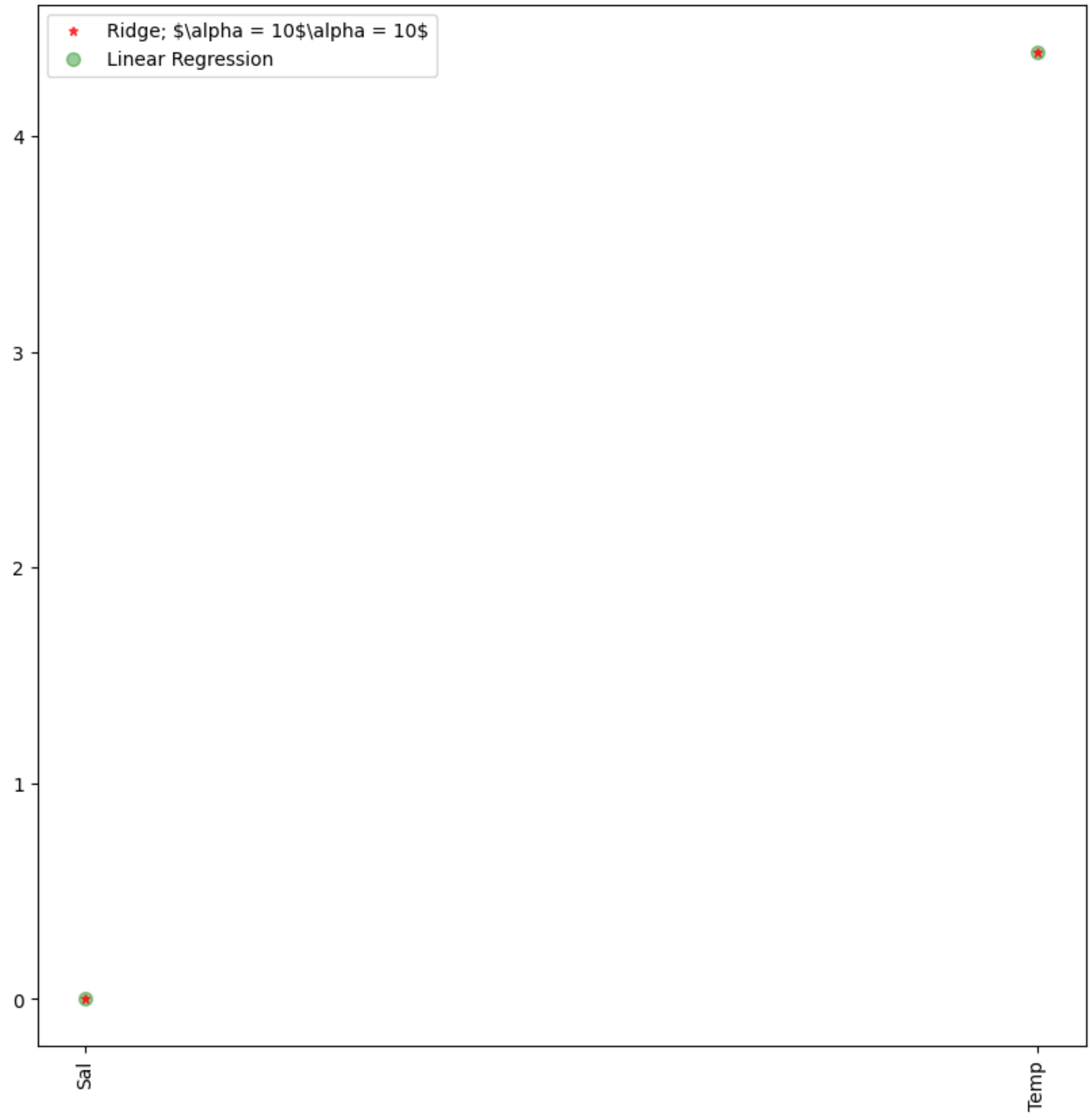
In [40]:

```python
#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 0.999999999723243
The test score for ridge model is 0.9999999997231402
```

In [41]:

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



In [42]:

```
#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))
```

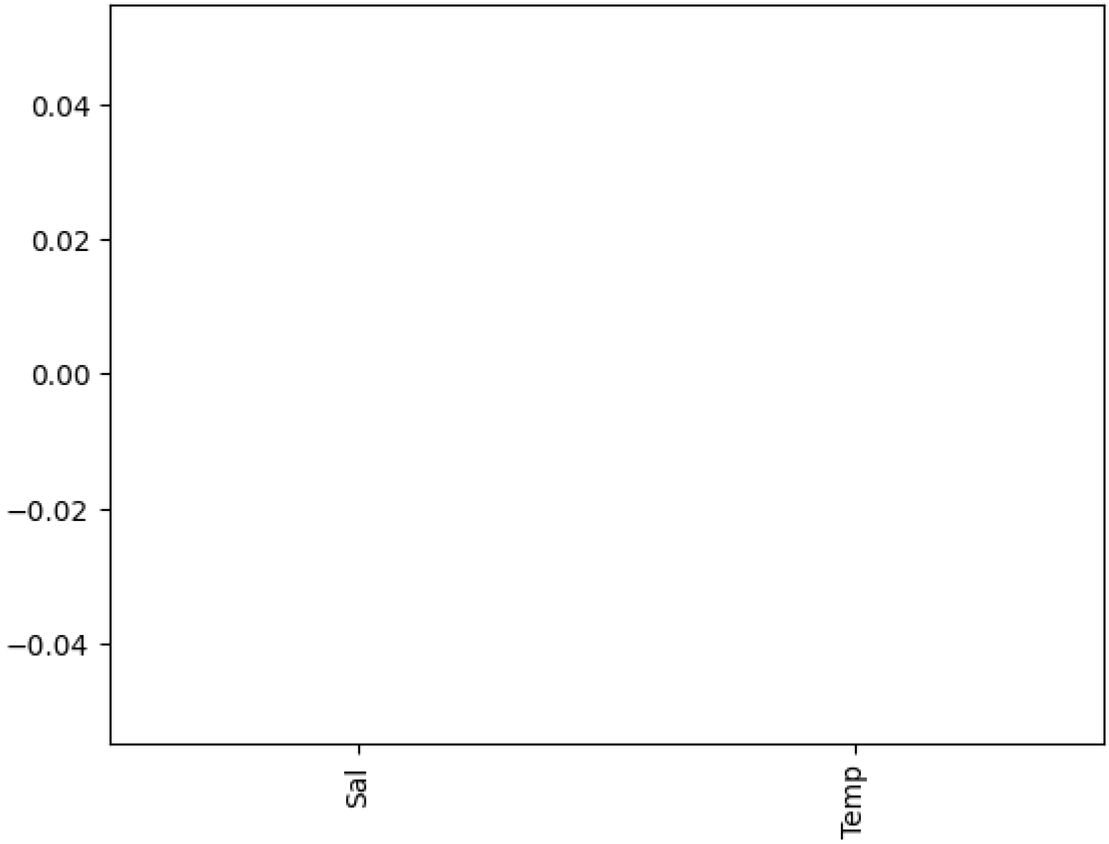
```
Lasso Model: 

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

In [43]:

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

Out[43]:

```
<Axes: >
```



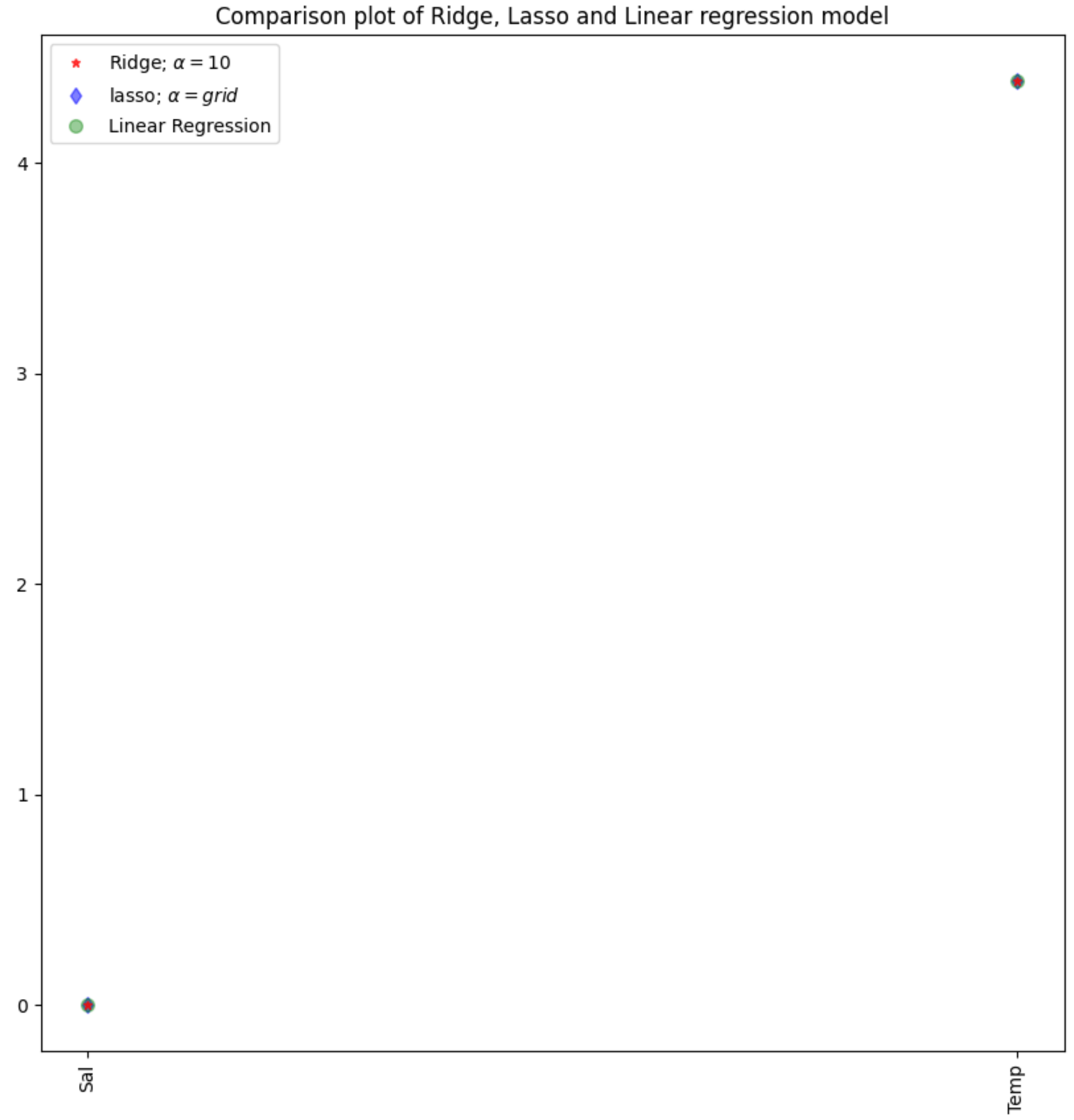
In [44]:

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

```
0.9999999994806811
0.9999999994806712
```

In [45]:

```
#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
#add plot for lasso regression
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='b
#add plot for linear model
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color
#rotate axis
plt.xticks(rotation = 90)
plt.legend()
plt.title("Comparison plot of Ridge, Lasso and Linear regression model")
plt.show()

```



In [46]:

```
#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))
```

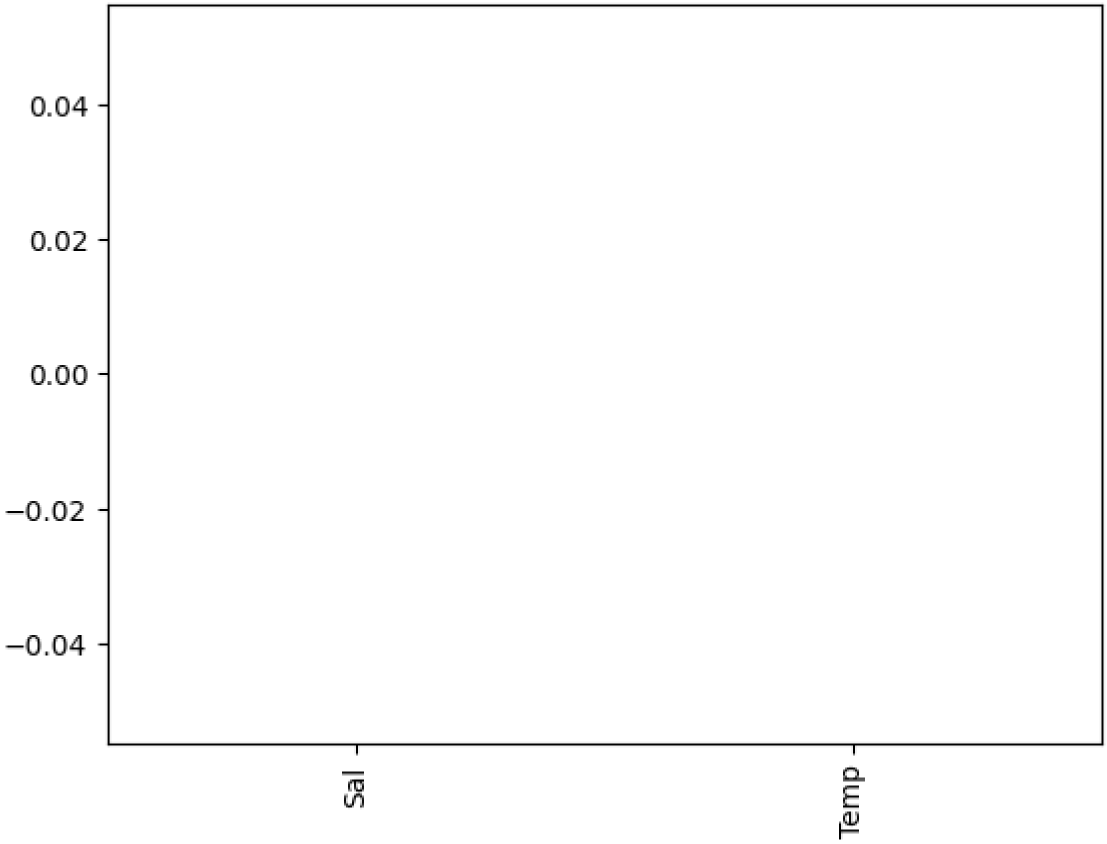
```
Lasso Model:

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

In [47]:

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

Out[47]:

```
<Axes: >
```



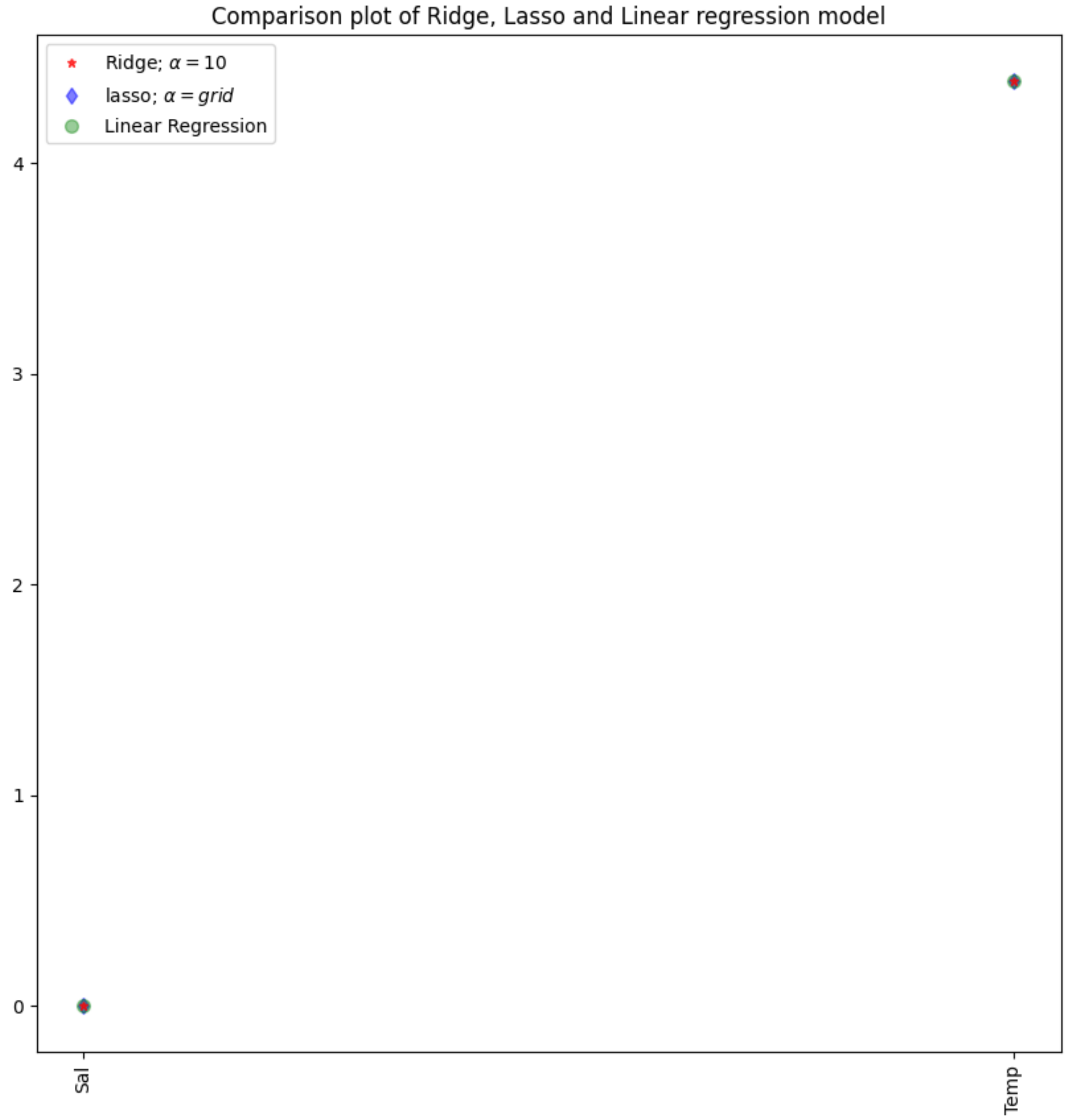
In [48]:

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

```
0.9999999994806811
0.9999999994806712
```

In [49]:

```
#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
#add plot for lasso regression
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='b
#add plot for linear model
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color
#rotate axis
plt.xticks(rotation = 90)
plt.legend()
plt.title("Comparison plot of Ridge, Lasso and Linear regression model")
plt.show()

```



In [50]:

```
#Using the linear CV model
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_trai
print("The train score for ridge model is {}".format(ridge_cv.score(X_test, y_test)
```

```
The train score for ridge model is 0.9999999981135502
The train score for ridge model is 0.99999999811206
```

In [51]:

```
from sklearn.linear_model import ElasticNet
regr=ElasticNet()
regr.fit(X_train,y_train)
print(regr.coef_)
print(regr.intercept_)
```

```
[0.         2.59210994]
10.668516033921204
```

In [52]:

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

In [53]:

```
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 3.225813457818192
```

In [ ]:

```

```