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.preprocessing import StandardScaler
```

In [2]:

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

```
C:\Users\rubin\AppData\Local\Temp\ipykernel_4564\2619180729.py:1: DtypeWarning: Col
umns (47,73) have mixed types. Specify dtype option on import or set low_memory=Fal
se.
  df=pd.read_csv(r"C:\Users\rubin\Downloads\bottle.csv")
```

Out[2]:

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta	O2Sat	...	I
0	1	1	054.0 056.0	19-4903CR-HY-060-0930-05400560-0000A-3	0	10.500	33.4400	NaN	25.64900	NaN	...	
1	1	2	054.0 056.0	19-4903CR-HY-060-0930-05400560-0008A-3	8	10.460	33.4400	NaN	25.65600	NaN	...	
2	1	3	054.0 056.0	19-4903CR-HY-060-0930-05400560-0010A-7	10	10.460	33.4370	NaN	25.65400	NaN	...	
3	1	4	054.0 056.0	19-4903CR-HY-060-0930-05400560-0019A-3	19	10.450	33.4200	NaN	25.64300	NaN	...	
4	1	5	054.0 056.0	19-4903CR-HY-060-0930-05400560-0020A-7	20	10.450	33.4210	NaN	25.64300	NaN	...	
...	...	...	...	...	...	...	...	...	...	...	...	
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	...	
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	...	
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	...	
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	...	

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta	O2Sat	...	I
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	...	

864863 rows × 74 columns

In [3]:
```
df=df[['Salnty','T_degC']]
df.columns=['Sal','Temp']
```

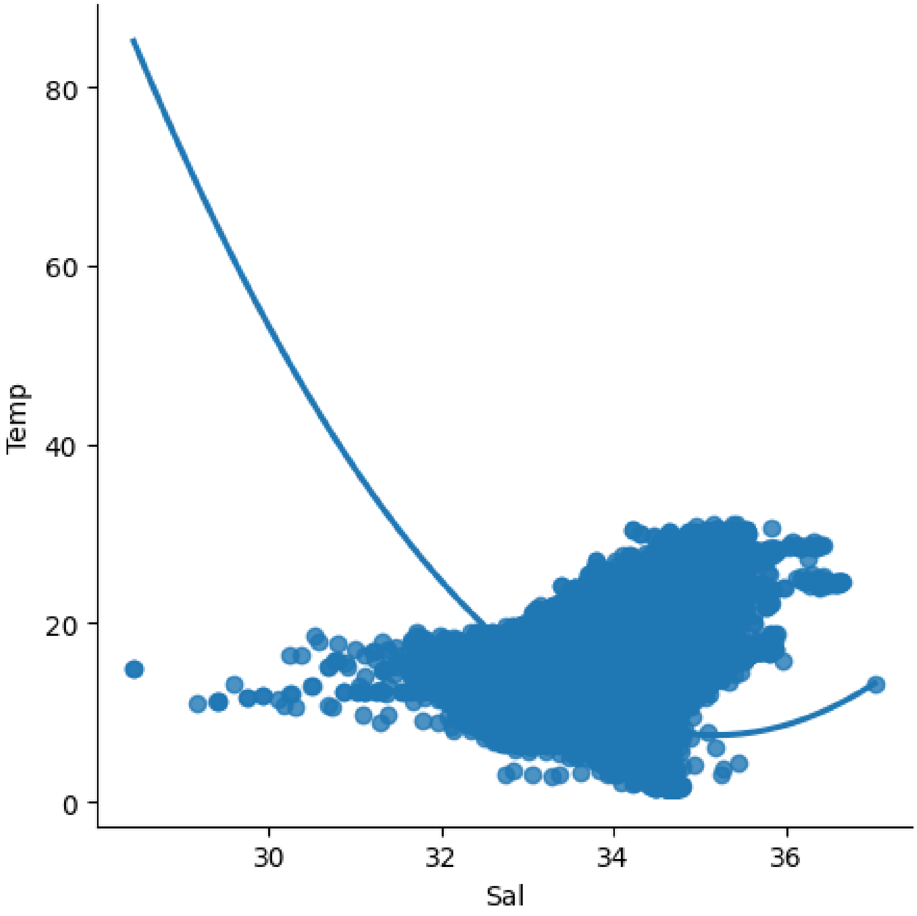
In [4]:
```
df.head()
```

Out[4]:

	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

```
In [5]: sns.lmplot(x='Sal',y='Temp',data=df,order=2,ci=None)
```

```
Out[5]: <seaborn.axisgrid.FacetGrid at 0x11e503f6170>
```



```
In [6]: df.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 [7]:
```
df.describe()
```

Out[7]:

	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 [8]:
```
df.fillna(method='ffill')
```

Out[8]:

	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 [9]:
```
df.fillna(value=0,inplace=True)
```

```
C:\Users\rubin\AppData\Local\Temp\ipykernel_4564\1434098079.py:1: SettingWithCopyWa
rning:
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 [10]:
```
df.isnull().sum()
```

Out[10]:
```
Sal     0
Temp    0
dtype: int64
```

In [11]:
```
x=np.array(df['Sal']).reshape(-1,1)
y=np.array(df['Temp']).reshape(-1,1)
```

In [12]:
```
df.isna().any()
```

Out[12]:
```
Sal     False
Temp    False
dtype: bool
```

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

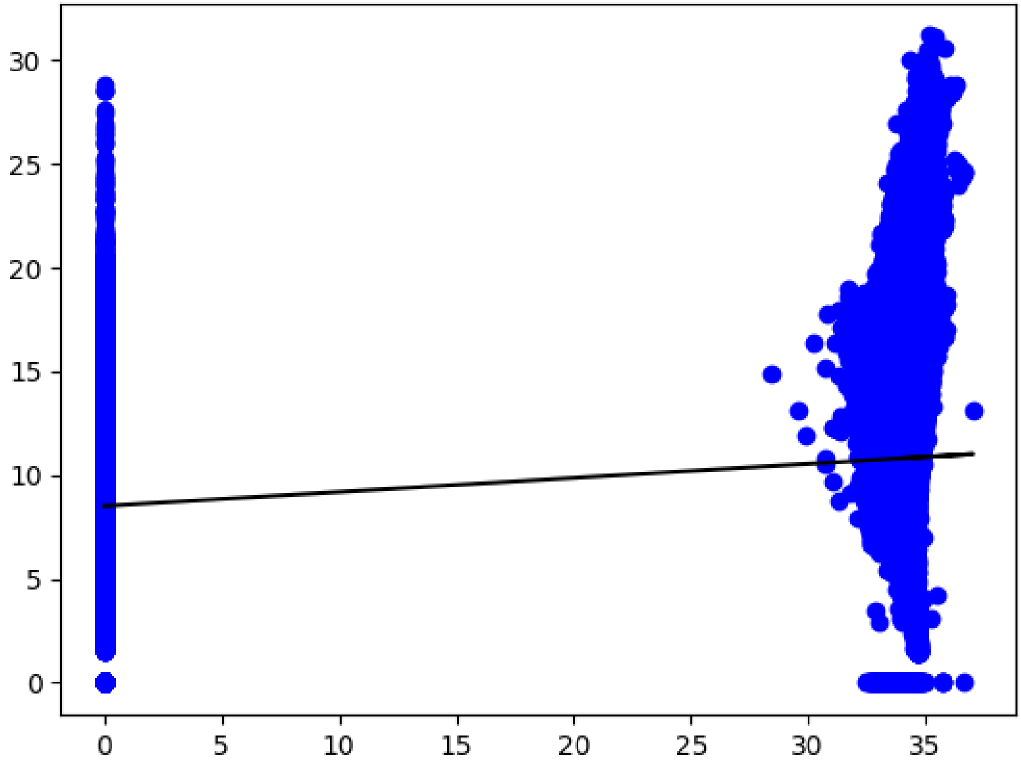
```
C:\Users\rubin\AppData\Local\Temp\ipykernel_4564\1379821321.py:1: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame

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)
reg=LinearRegression()
reg.fit(x_train,y_train)
print(reg.score(x_test,y_test))
```

```
0.014074449093311903
```

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

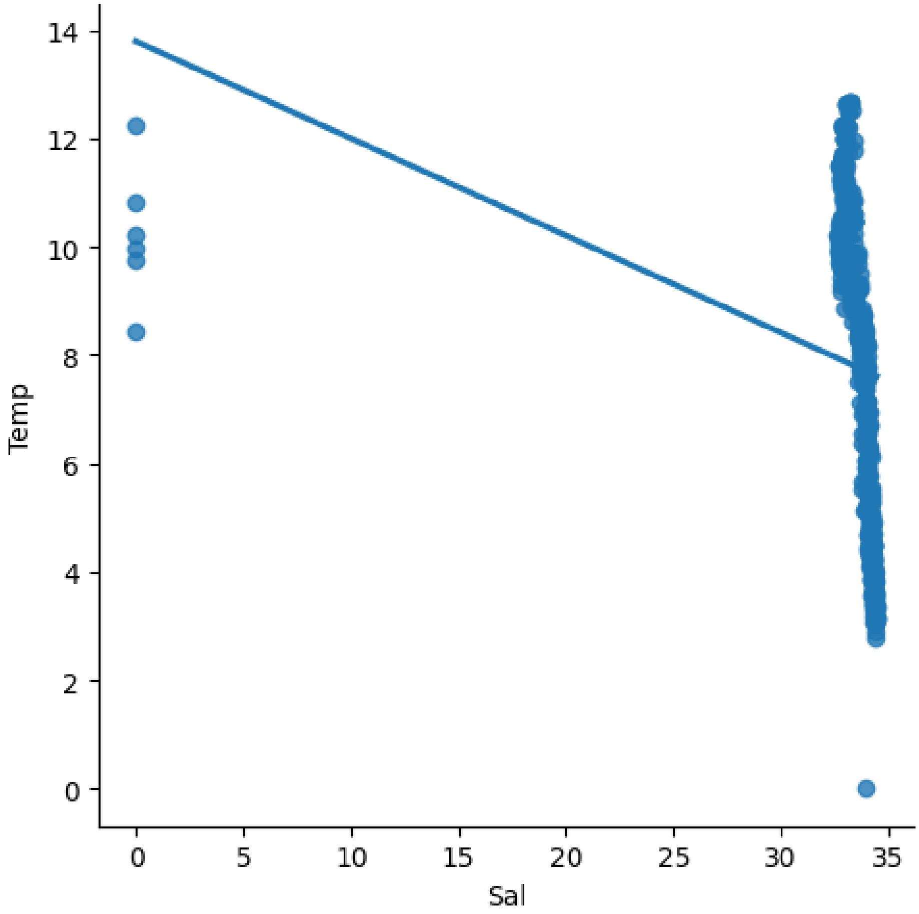


In [17]:

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

Out[17]:

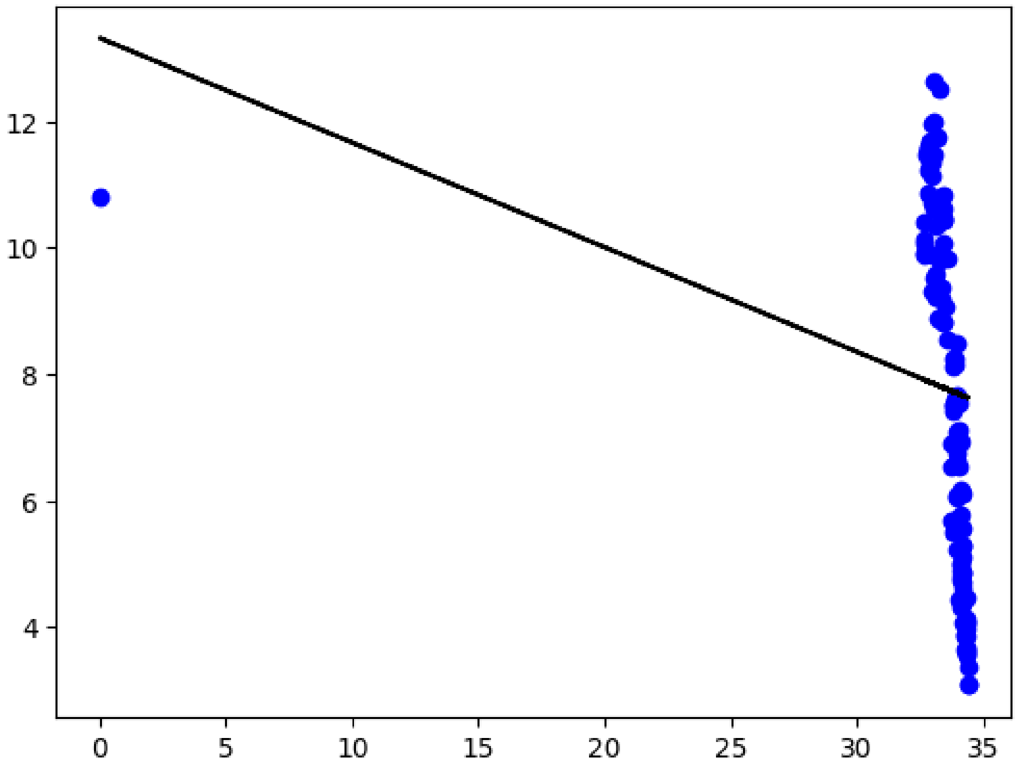
```
<seaborn.axisgrid.FacetGrid at 0x11e51498eb0>
```



In [20]:

```
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: 0.061335622373488774
```



In [21]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

In [22]:

```
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)
```

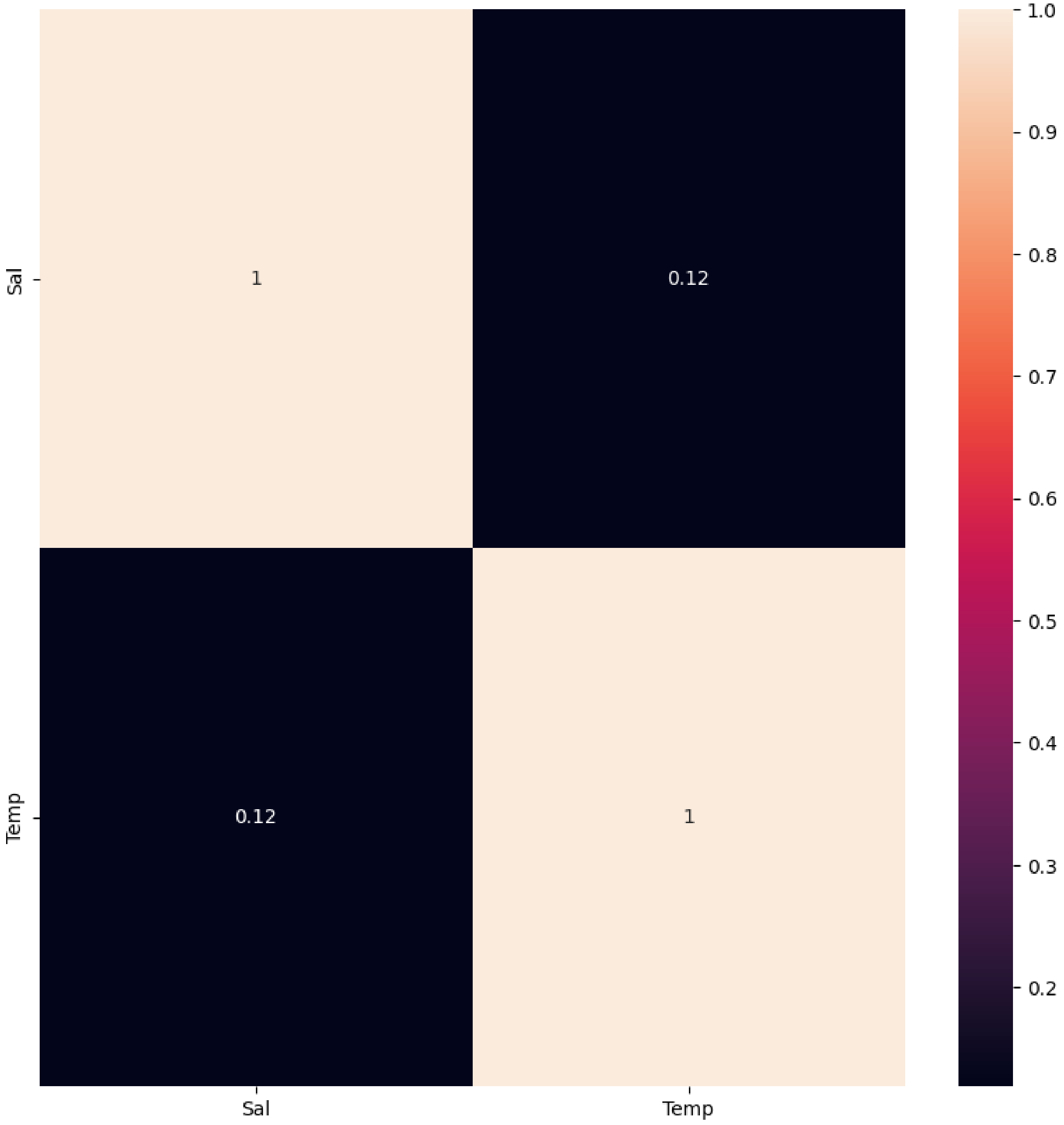
```
R2 score: 0.061335622373488774
```

IMPLEMENTING RIDGE AND LASSO REGRESSION

```
In [23]: from sklearn.linear_model import Ridge
         from sklearn.linear_model import RidgeCV
         from sklearn.linear_model import Lasso
```

```
In [24]: plt.figure(figsize=(10,10))
         sns.heatmap(df.corr(),annot=True)
```

```
Out[24]: <Axes: >
```



In [25]:

```
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_stat
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 [26]:

```
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 [27]:

```
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))
```

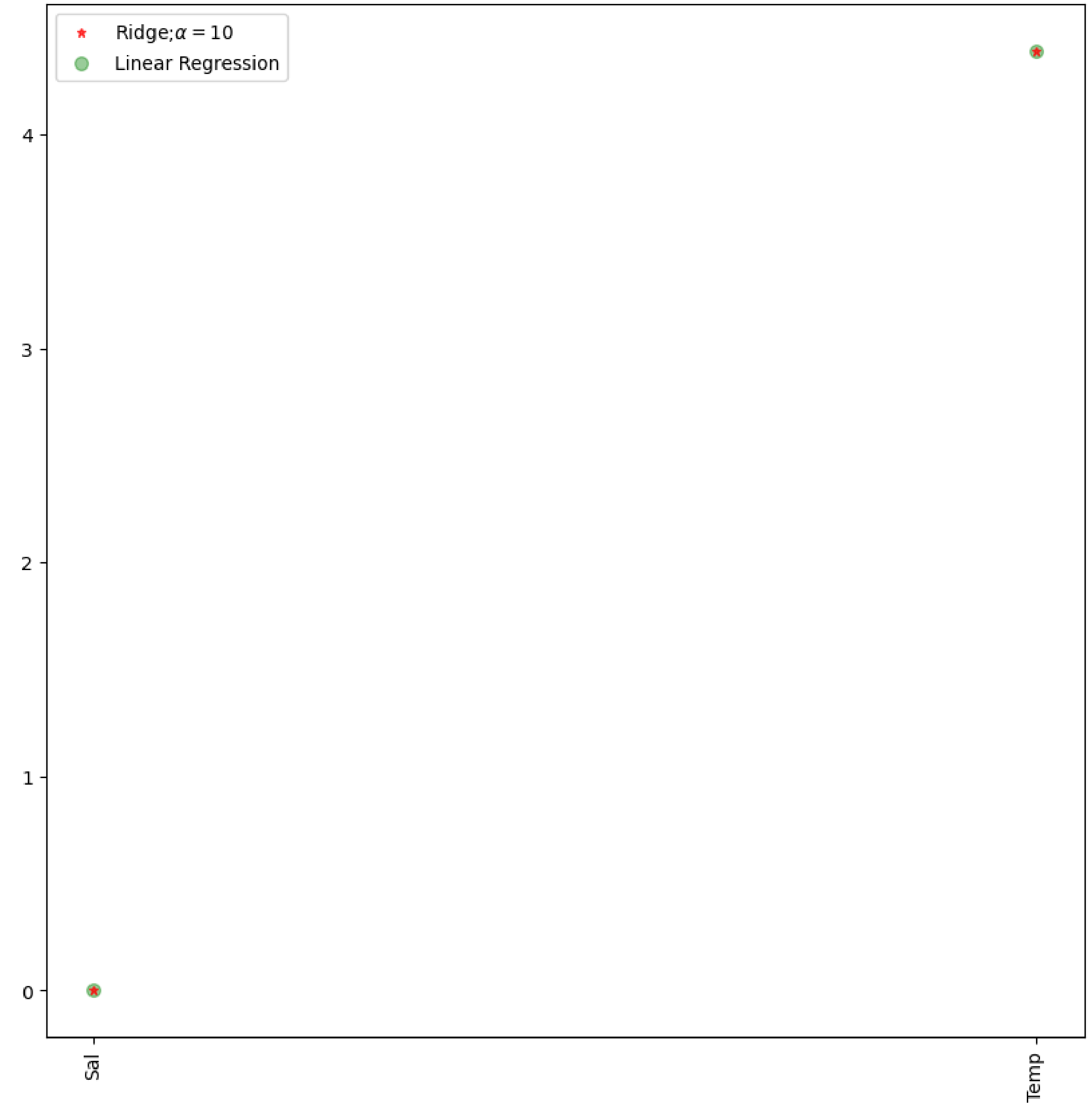
```

Ridge Model:

The train score for ridge model is 0.999999999723243
The test score for ridge model is 0.9999999997231402
```

In [28]:

```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,
#plt.plot(rr100.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color=
plt.xticks(rotation=90)
plt.legend()
#plt.title("comparison plot of Ridge,Lasso and Linear regression model")
plt.show()
```



In [29]:
```
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))
```

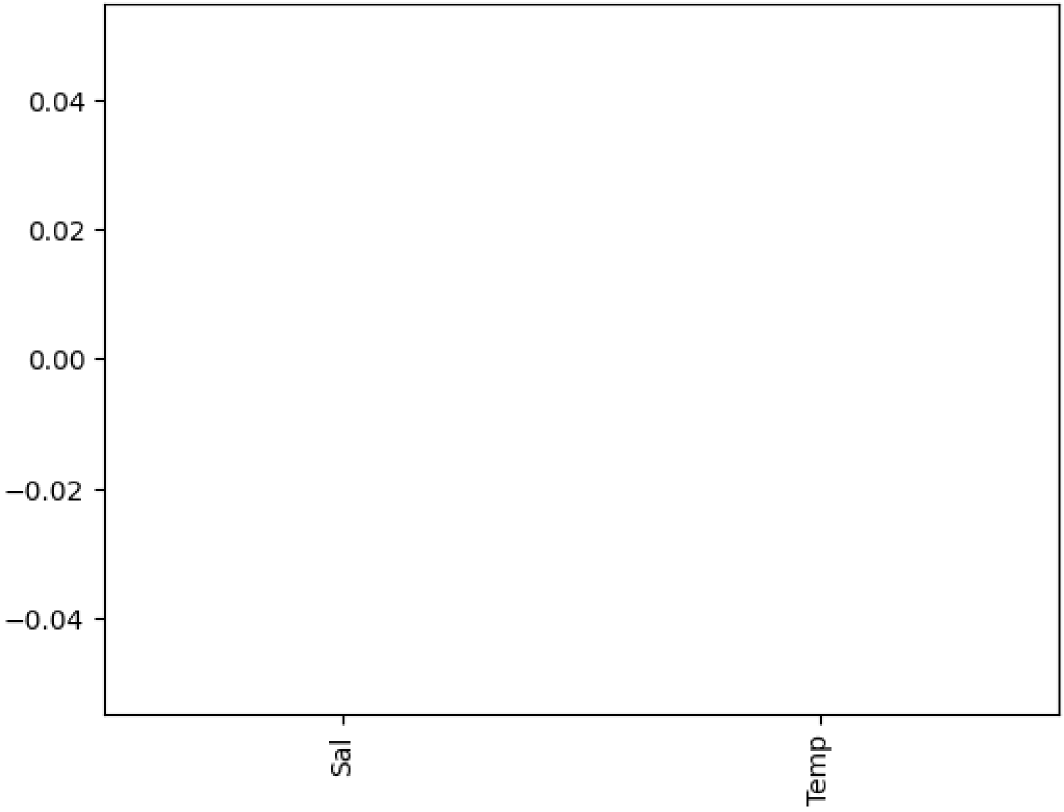
```

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: >
```

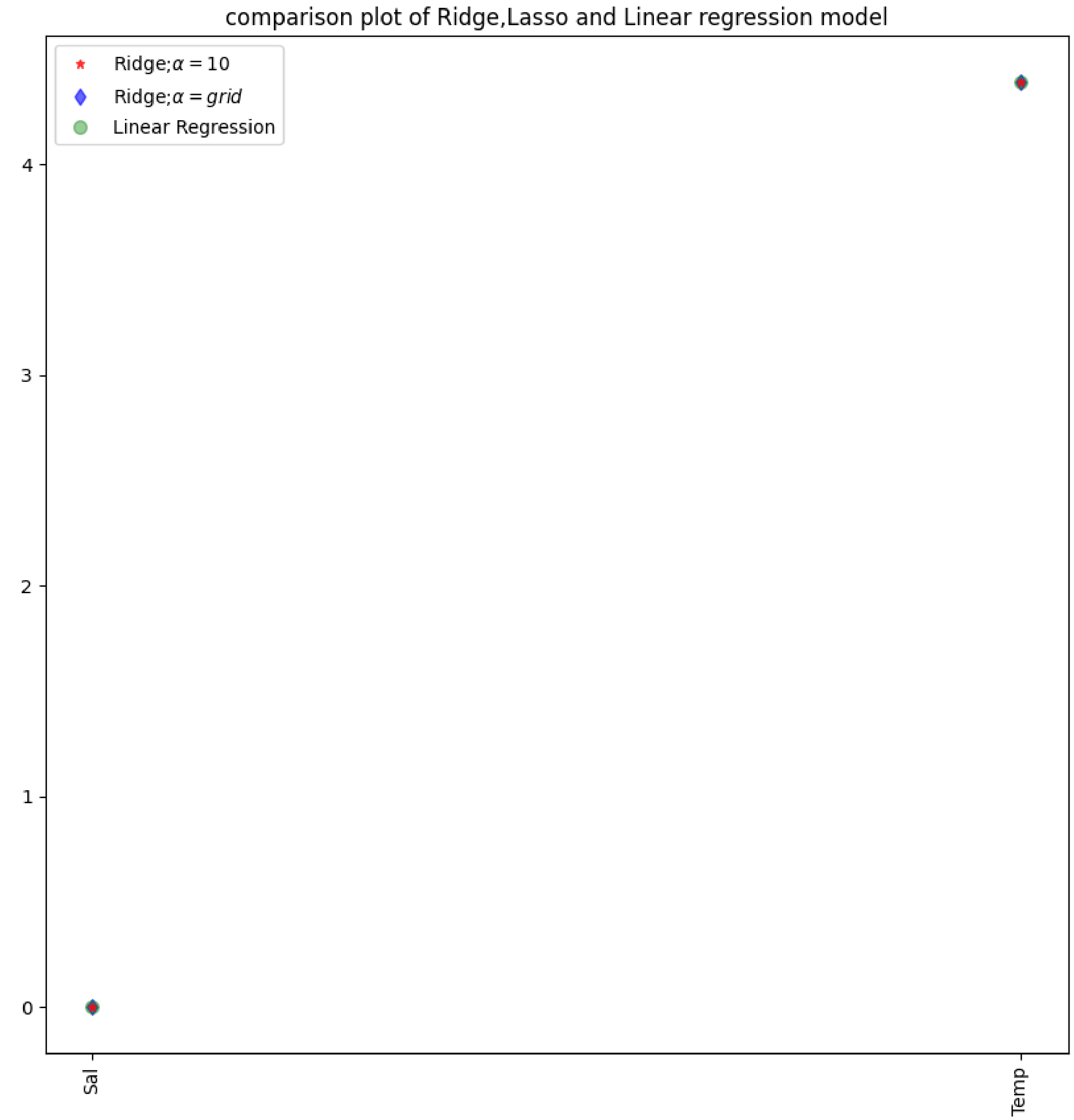


In [35]:
```
from sklearn.linear_model import LassoCV
lasso_cv=LassoCV(alphas=[0.0001,0.001,0.01,0.1,1,10],random_state=0).fit(X_train, y_
print(lasso_cv.score(X_train,y_train))
print(lasso_cv.score(X_test,y_test))
```

```
0.9999999994806811
0.9999999994806712
```

In [36]:

```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,
plt.plot(lasso_cv.coef_,alpha=0.6,linestyle='none',marker='d',markersize=6,color='bl
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color=
plt.xticks(rotation=90)
plt.legend()
plt.title("comparison plot of Ridge,Lasso and Linear regression model")
plt.show()
```



In [37]:

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

ELASTIC NET REGRESSION

In [38]:

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

```
[0.         0.94934511]
0.5401219631063316
```

In [39]:

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

In [40]:

```
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
```

In [ ]: