In [3]:
```
#step 1:importing
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
#reading the data set
```

In [4]:
```
#step2:reading data set
df=pd.read_csv(r"C:\Users\ubinl\Downloads\bottle.csv.zip")
df
```

```
C:\Users\ubinl\AppData\Local\Temp\ipykernel_29804\3036292363.py:2: DtypeWarni
ng: Columns (47,73) have mixed types. Specify dtype option on import or set l
ow_memory=False.
  df=pd.read_csv(r"C:\Users\ubinl\Downloads\bottle.csv.zip")
```

Out[4]:

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta	O2S
0	1	1	054.0 056.0	19-4903CR-HY-060-0930-05400560-0000A-3	0	10.500	33.4400	NaN	25.64900	Na
1	1	2	054.0 056.0	19-4903CR-HY-060-0930-05400560-0008A-3	8	10.460	33.4400	NaN	25.65600	Na
2	1	3	054.0 056.0	19-4903CR-HY-060-0930-05400560-0010A-7	10	10.460	33.4370	NaN	25.65400	Na
3	1	4	054.0 056.0	19-4903CR-HY-060-0930-05400560-0019A-3	19	10.450	33.4200	NaN	25.64300	Na
4	1	5	054.0 056.0	19-4903CR-HY-060-0930-05400560-0020A-7	20	10.450	33.4210	NaN	25.64300	Na
...	...	...	...	...	...	...	...	...	...	
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.
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.
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.
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.

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta	O2S
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.

864863 rows × 74 columns

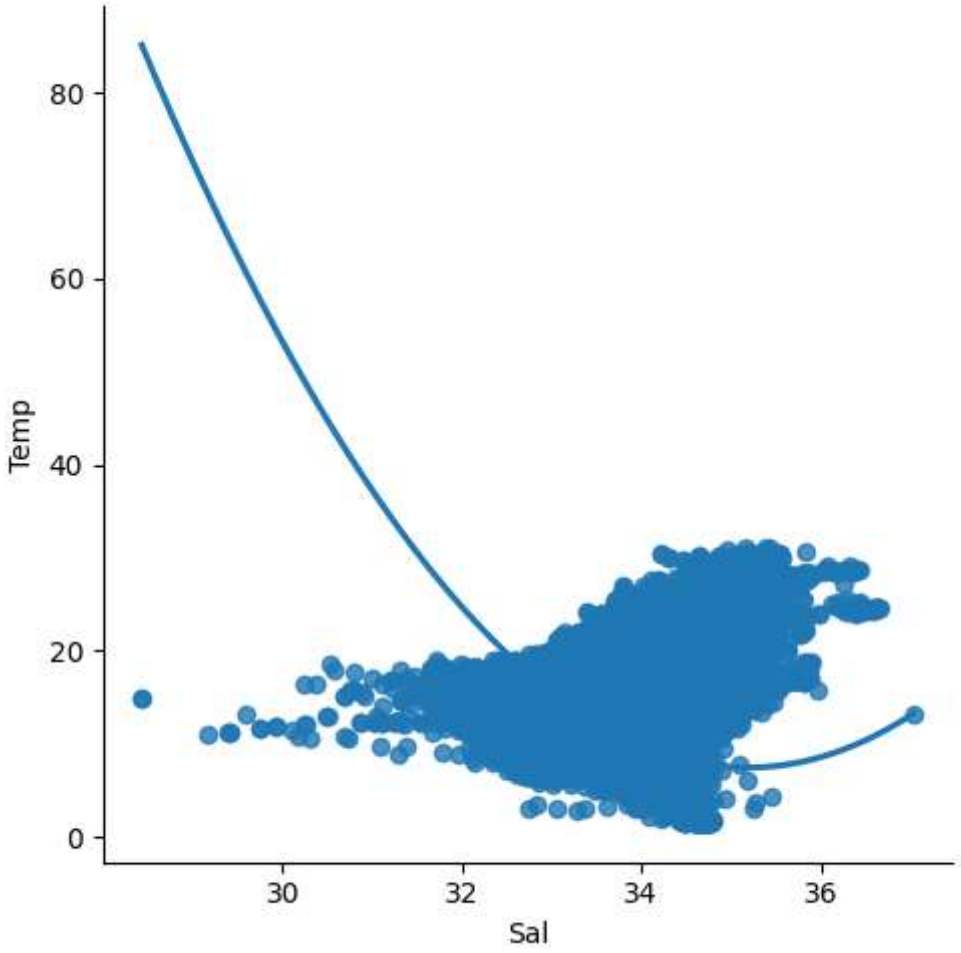
In [5]:
```
df=df[['Salnty','T_degC']]
df.columns=['Sal','Temp']
df.head(10)
```

Out[5]:

	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

In [6]:
```
#step 3:exploring
sns.lmplot(x="Sal",y="Temp",data=df,order=2,ci=None)
```

Out[6]: <seaborn.axisgrid.FacetGrid at 0x1b8fe608990>



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.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
```

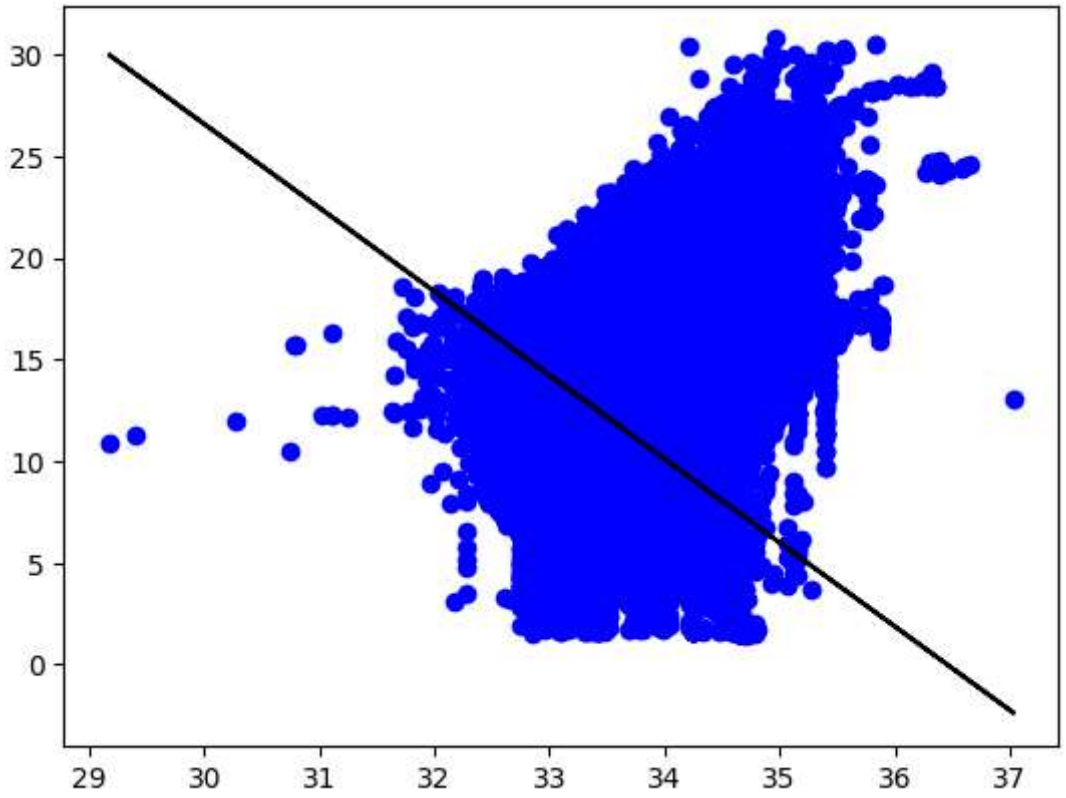
In [9]:
```
#step 4:
df.fillna(method='ffill',inplace=True)
```

```
C:\Users\ubinl\AppData\Local\Temp\ipykernel_29804\3632936489.py:2: SettingWit
hCopyWarning:
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/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  df.fillna(method='ffill',inplace=True)
```

In [10]:
```
#step 5:training model
x=np.array(df['Sal']).reshape(-1,1)
y=np.array(df['Temp']).reshape(-1,1)
#seperating
#column
df.dropna(inplace=True)
#droping values
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
#spliting data
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

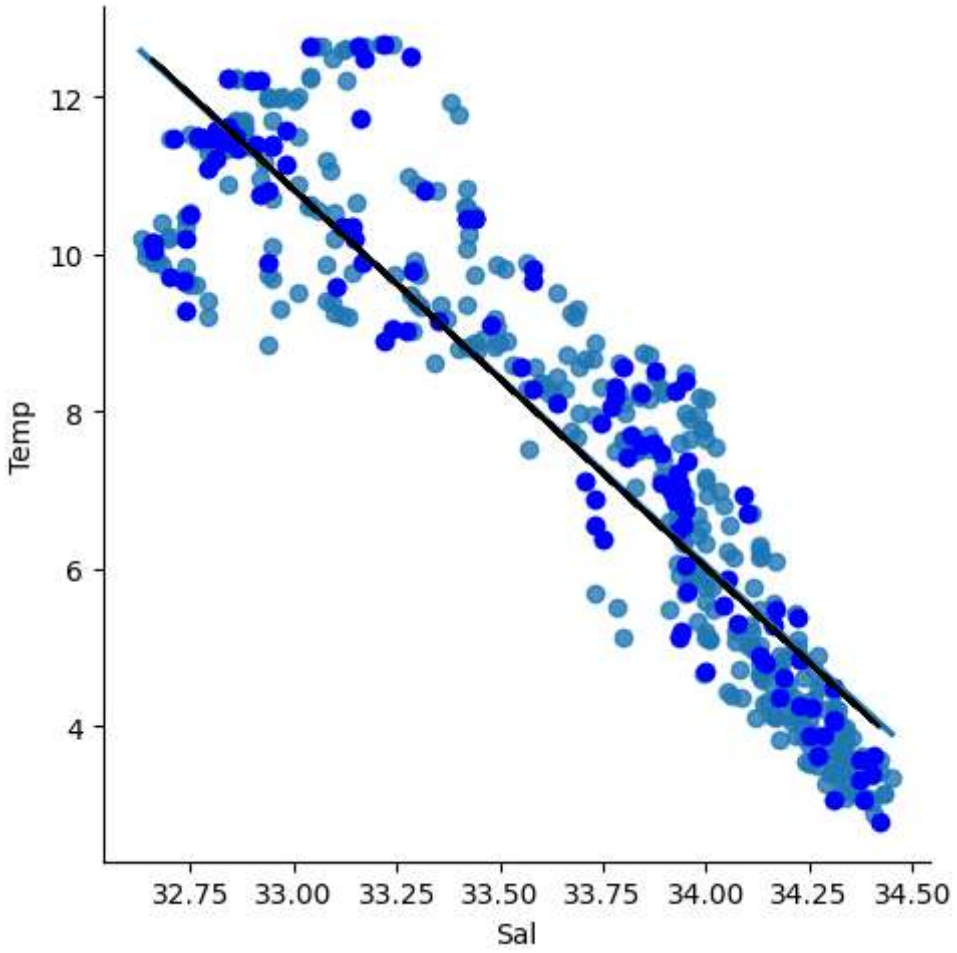
```
0.20122480857385217

C:\Users\ubinl\AppData\Local\Temp\ipykernel_29804\59502318.py:6: SettingWithC
opyWarning:
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/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  df.dropna(inplace=True)
```

In [11]:
```
#step 6:exploring results
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
#scatter
```



In [12]:
```
#step7:working with a smaller data set
df500=df[:][:500]
#selecting
sns.lmplot(x="Sal",y="Temp",data=df500,order=1,ci=None)
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: 0.8363544393576057
```



In [13]:
```
#step 8:
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
#train
model=LinearRegression()
model.fit(x_train,y_train)
#evaluate
y_pred=model.predict(x_test)
r2=r2_score(y_test,y_pred)
print("r2_score:",r2)
```

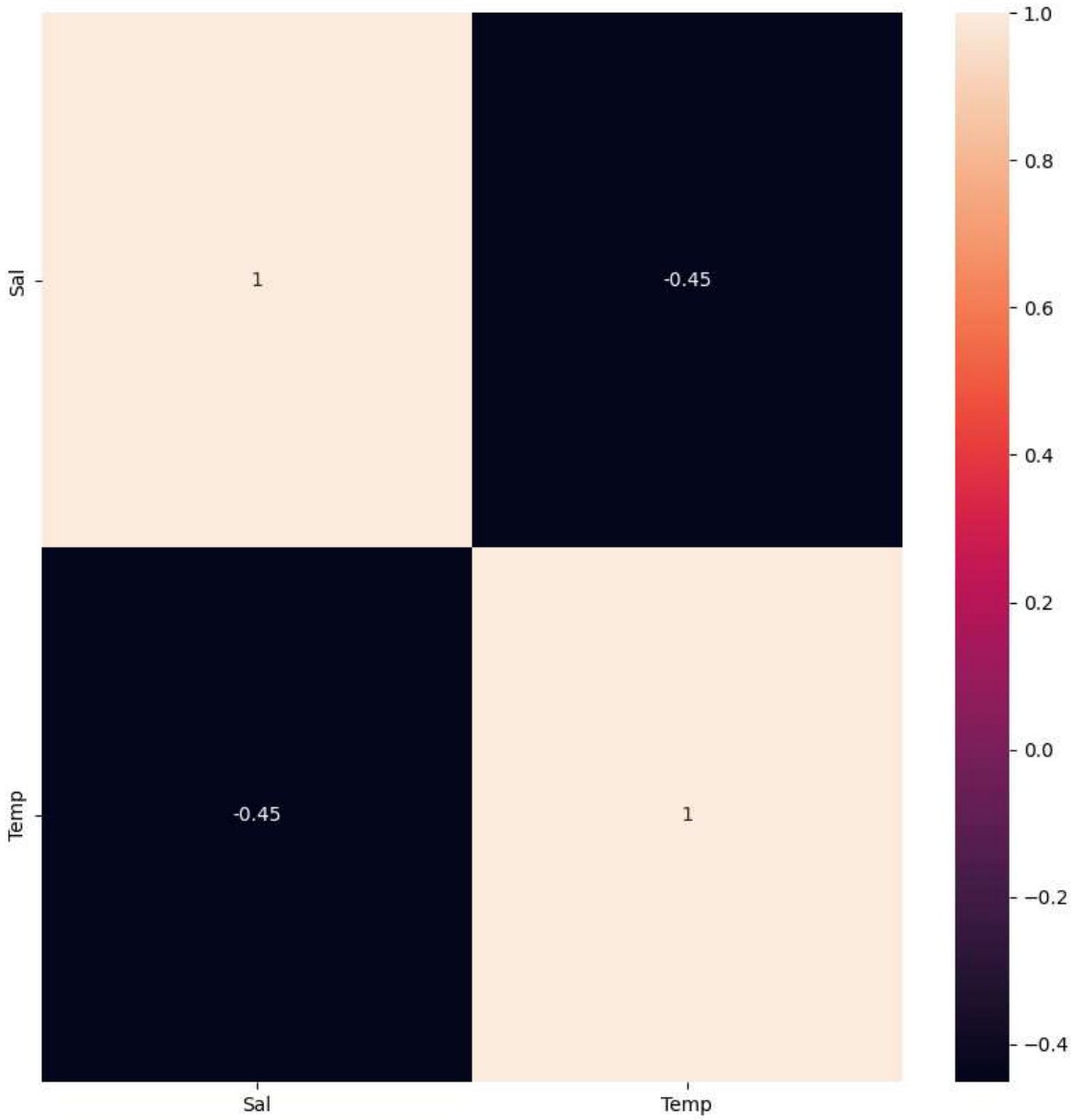
```
r2_score: 0.8363544393576057
```

In [21]:
```
#step 9:conclusion:
#Dataset we have taken is poor for linear model but with smaller data it works
```

In [22]:
```
from sklearn.linear_model import Ridge,RidgeCV ,Lasso
from sklearn.preprocessing import StandardScaler
```

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

Out[23]: <Axes: >



In [24]:
```
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=
print("The dimension of x_train is {}".format(x_train.shape))
print("The dimension of x_train is {}".format(x_test.shape))
```

```
The dimension of x_train is (605404, 2)
The dimension of x_train is (259459, 2)
```

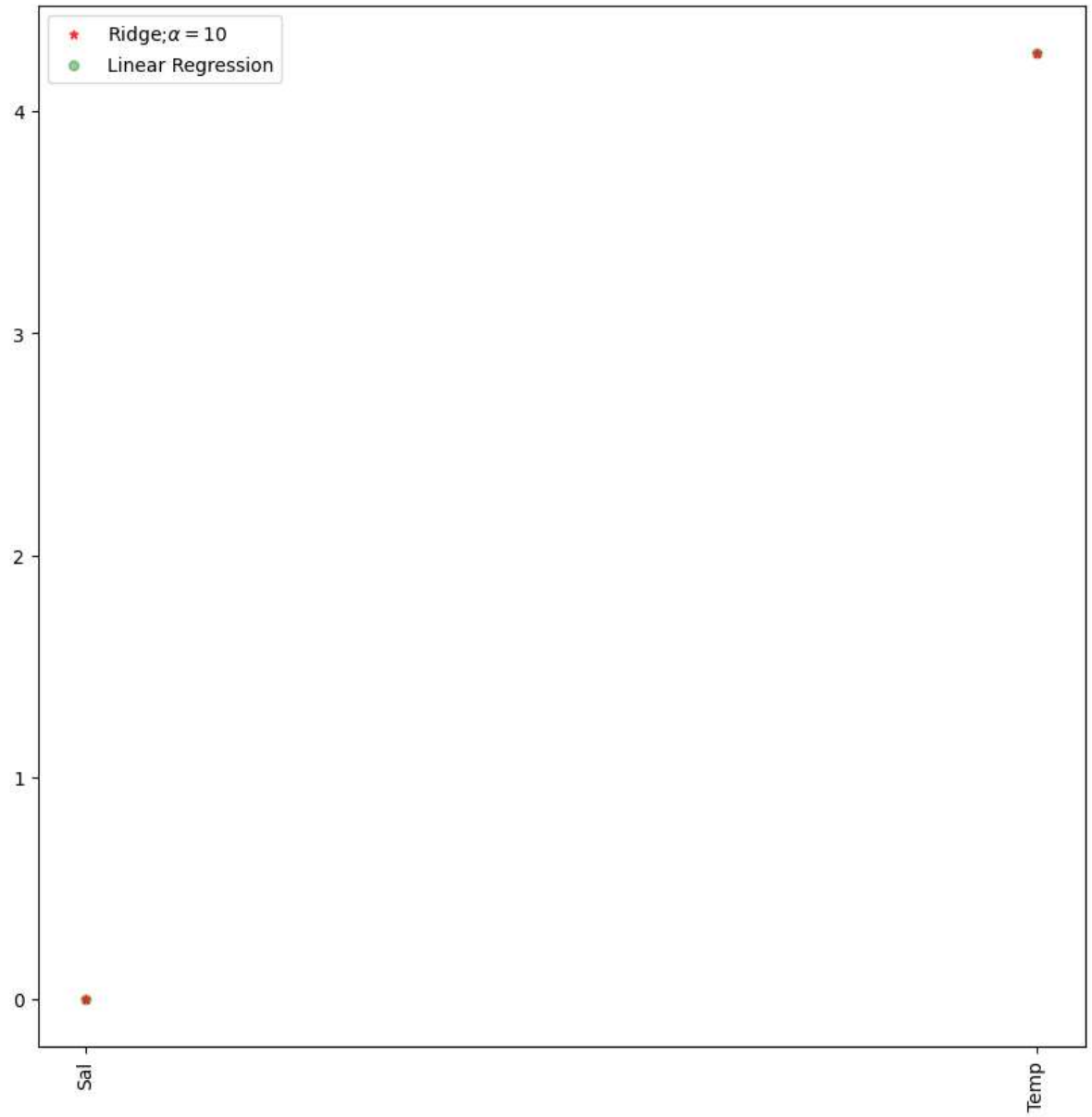
In [25]:
```
scaler=StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.transform(x_test)
lr=LinearRegression()
lr.fit(x_train,y_train)
actual=y_test
train_score_lr=lr.score(x_train,y_train)
test_score_lr=lr.score(x_test,y_test)
print("\nLinear Regression Model:\n")
print("The train score of lr model is {}".format(train_score_lr))
print("The test score of lr model is {}".format(test_score_lr))
```

```
Linear Regression Model:

The train score of lr model is 1.0
The test score of lr model is 1.0
```

In [26]:
```
r=Ridge(alpha=10)
r.fit(x_train,y_train)
train_score_ridge=r.score(x_train,y_train)
test_score_ridge=r.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 0.9999999996569116
The test score for ridge model is 0.9999999996561358
```

In [27]:
```
plt.figure(figsize=(10,10))
plt.plot(features,r.coef_,alpha=0.7,linestyle='None',marker='*',markersize=5,c
plt.plot(features,lr.coef_,alpha=0.4,linestyle='None',marker='o',markersize=5,
plt.xticks(rotation=90)
plt.legend()
plt.show()
```

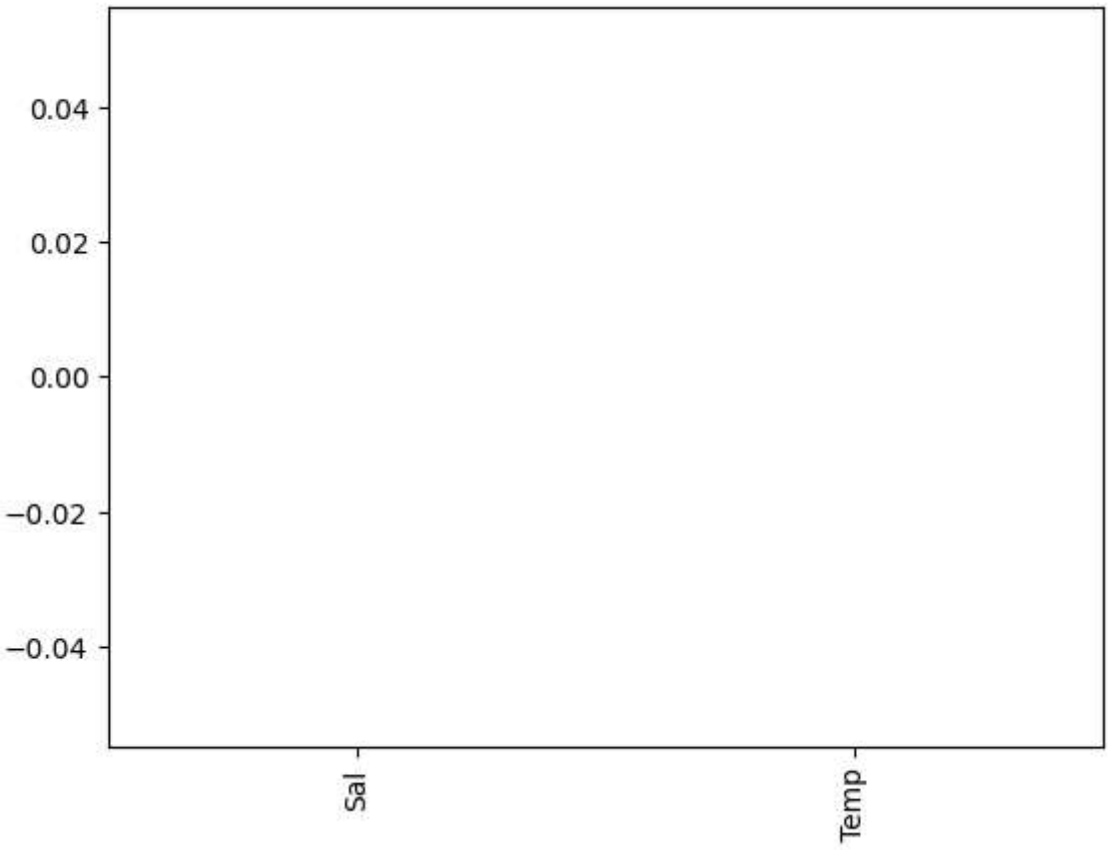


In [28]:
```
l=Lasso(alpha=10)
l.fit(x_train,y_train)
train_score_ls=l.score(x_train,y_train)
test_score_ls=l.score(x_test,y_test)
print("\nRidge model:\n")
print("The train score for ridge model is {}".format(train_score_ls))
print("The test score for ridge model is {}".format(test_score_ls))
```

```
Ridge model:

The train score for ridge model is 0.0
The test score for ridge model is -9.467790479389393e-06
```

In [29]:
```
pd.Series(l.coef_,features).sort_values(ascending=True).plot(kind='bar')
```

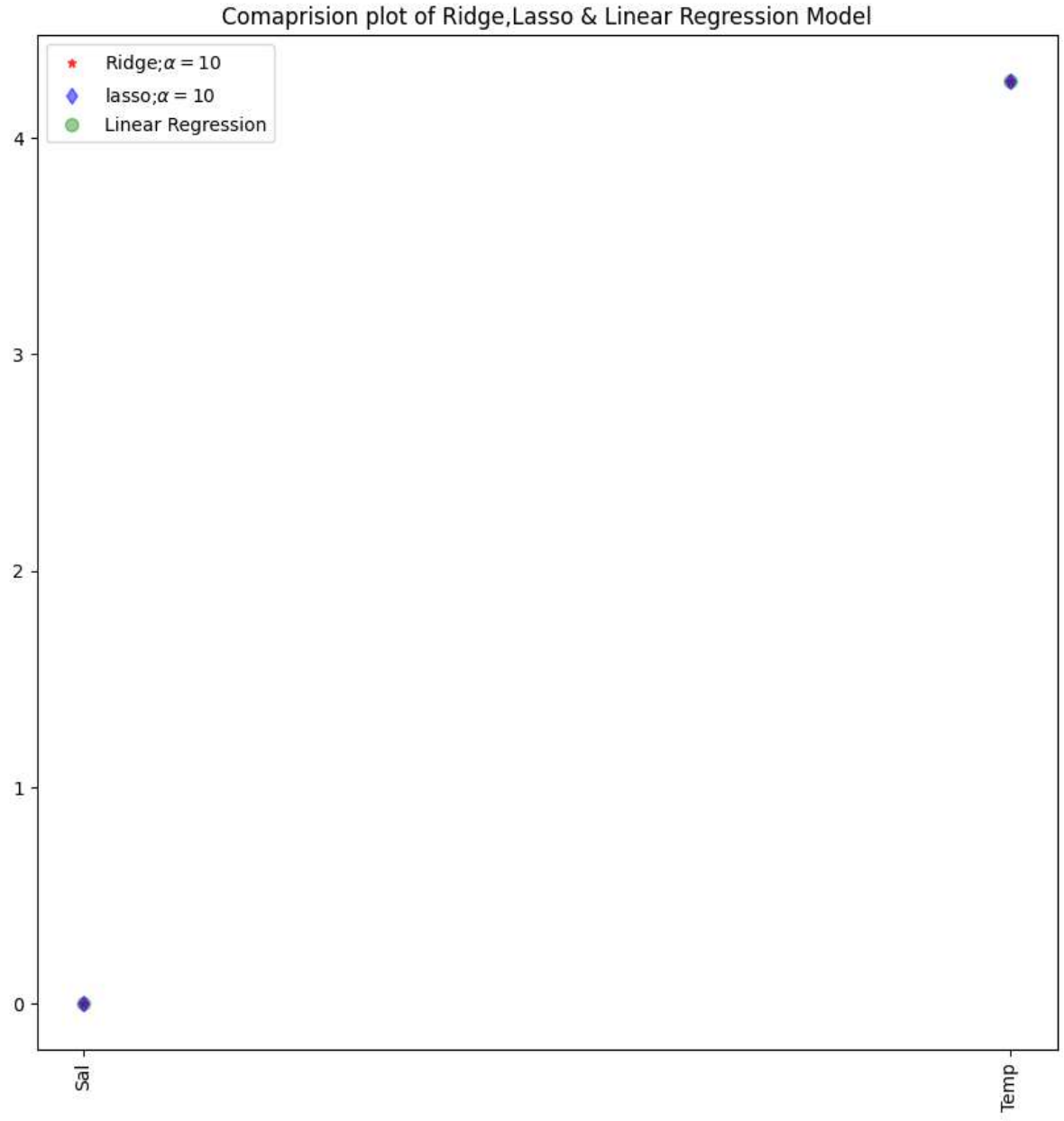
Out[29]: <Axes: >



In [30]:
```
from sklearn.linear_model import LassoCV
lc=LassoCV(alphas=[0.0001,0.001,0.01,0.1,0,1,10],random_state=0).fit(x_train,y
print(lc.score(x_train,y_train))
print(lc.score(x_test,y_test))
```

```
C:\Users\ubinl\AppData\Local\Programs\Python\Python311\Lib\site-packages\skle
arn\linear_model\_coordinate_descent.py:617: UserWarning: Coordinate descent
without L1 regularization may lead to unexpected results and is discouraged.
Set l1_ratio > 0 to add L1 regularization.
  model = cd_fast.enet_coordinate_descent_gram(
C:\Users\ubinl\AppData\Local\Programs\Python\Python311\Lib\site-packages\skle
arn\linear_model\_coordinate_descent.py:617: UserWarning: Coordinate descent
without L1 regularization may lead to unexpected results and is discouraged.
Set l1_ratio > 0 to add L1 regularization.
  model = cd_fast.enet_coordinate_descent_gram(
C:\Users\ubinl\AppData\Local\Programs\Python\Python311\Lib\site-packages\skle
arn\linear_model\_coordinate_descent.py:617: UserWarning: Coordinate descent
without L1 regularization may lead to unexpected results and is discouraged.
Set l1_ratio > 0 to add L1 regularization.
  model = cd_fast.enet_coordinate_descent_gram(
C:\Users\ubinl\AppData\Local\Programs\Python\Python311\Lib\site-packages\skle
arn\linear_model\_coordinate_descent.py:617: UserWarning: Coordinate descent
without L1 regularization may lead to unexpected results and is discouraged.
Set l1_ratio > 0 to add L1 regularization.
  model = cd_fast.enet_coordinate_descent_gram(
C:\Users\ubinl\AppData\Local\Programs\Python\Python311\Lib\site-packages\skle
arn\linear_model\_coordinate_descent.py:617: UserWarning: Coordinate descent
without L1 regularization may lead to unexpected results and is discouraged.
Set l1_ratio > 0 to add L1 regularization.
  model = cd_fast.enet_coordinate_descent_gram(
C:\Users\ubinl\AppData\Local\Programs\Python\Python311\Lib\site-packages\skle
arn\linear_model\_coordinate_descent.py:1712: UserWarning: With alpha=0, this
algorithm does not converge well. You are advised to use the LinearRegression
estimator
  model.fit(X, y)
C:\Users\ubinl\AppData\Local\Programs\Python\Python311\Lib\site-packages\skle
arn\linear_model\_coordinate_descent.py:631: UserWarning: Coordinate descent
with no regularization may lead to unexpected results and is discouraged.
  model = cd_fast.enet_coordinate_descent(

0.9999999999628355
0.9999999999627442
```

In [31]:
```
plt.figure(figsize=(10,10))
plt.plot(features,r.coef_,alpha=0.7,linestyle='None',marker='*',markersize=5,c
plt.plot(features,lc.coef_,alpha=0.5,linestyle='None',marker='d',markersize=6,
plt.plot(features,lr.coef_,alpha=0.4,linestyle='None',marker='o',markersize=7,
plt.xticks(rotation=90)
plt.title("Comaprision plot of Ridge,Lasso & Linear Regression Model")
plt.legend()
plt.show()
```



In [32]:
```
from sklearn.linear_model import RidgeCV
rc=RidgeCV(alphas=[0.0001,0.001,0.01,0.1,1,10]).fit(x_train,y_train)
print("\nRidge model:\n")
print("The train score for ridge model is {}".format(rc.score(x_train,y_train)
print("The test score for ridge model is {}".format(rc.score(x_test,y_test)))
```

```
Ridge model:

The train score for ridge model is 0.9999999443609283
The test score for ridge model is 0.9999999444198527
```

In [33]:
```
from sklearn.linear_model import ElasticNet
e=ElasticNet()
e.fit(x,y)
print(e.coef_)
print(e.intercept_)
```

```
[-0.          0.94635903]
0.5788601900287595
```

In [34]:
```
y_pred_elastic=e.predict(x_train)
mse=np.mean((y_pred_elastic-y_train)**2)
print("Mean squared error on test set",mse)
```

```
Mean squared error on test set 115.36385388404192
```