In [76]:

```
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
```

In [77]:

```
df=pd.read_csv(r"C:\Users\teppa\Desktop\AHISAI\bottle.csv")
df
```

```
C:\Users\teppa\AppData\Local\Temp\ipykernel_1224\3415337971.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\teppa\Desktop\AHISAI\bottle.csv")
```

Out[77]:

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta	O2Sat	...	R_PHAEO	R_PRES	R_
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 [78]:
```
df=df[['Salnty','T_degC']]
df.columns=['sal','tem']
```

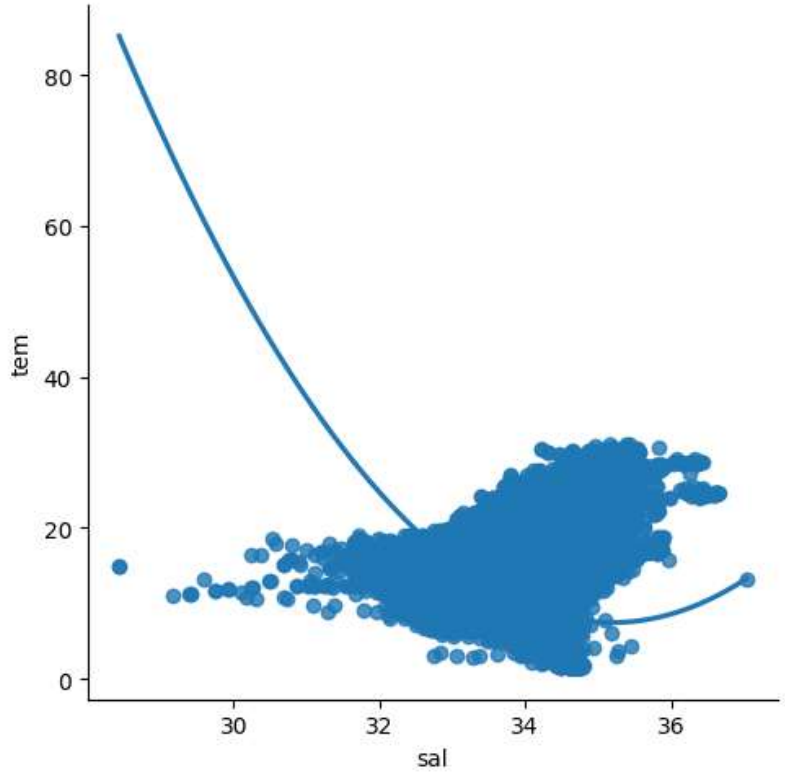
In [79]:
```
df.head()
```

Out[79]:

	sal	tem
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 [80]:
```
sns.lmplot(x='sal',y='tem',data=df,order=2,ci=None)
```

Out[80]:
```
<seaborn.axisgrid.FacetGrid at 0x17c29516aa0>
```



In [81]:
```
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   tem     853900 non-null  float64
dtypes: float64(2)
memory usage: 13.2 MB
```

In [82]:
```
1 df.describe()
```

Out[82]:

	sal	tem
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 [83]:
```
1 df.fillna(method='ffill')
```

Out[83]:

	sal	tem
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 [84]:
```
1 df.fillna(value=0,inplace=True)
```

```
C:\Users\teppa\AppData\Local\Temp\ipykernel_1224\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 [85]:
```
1 df.isnull().sum()
```

Out[85]:
```
sal    0
tem    0
dtype: int64
```

In [86]:
```
1 x=np.array(df['sal']).reshape(-1,1)
2 y=np.array(df['tem']).reshape(-1,1)
```

In [87]:
```
df.dropna(inplace=True)
```

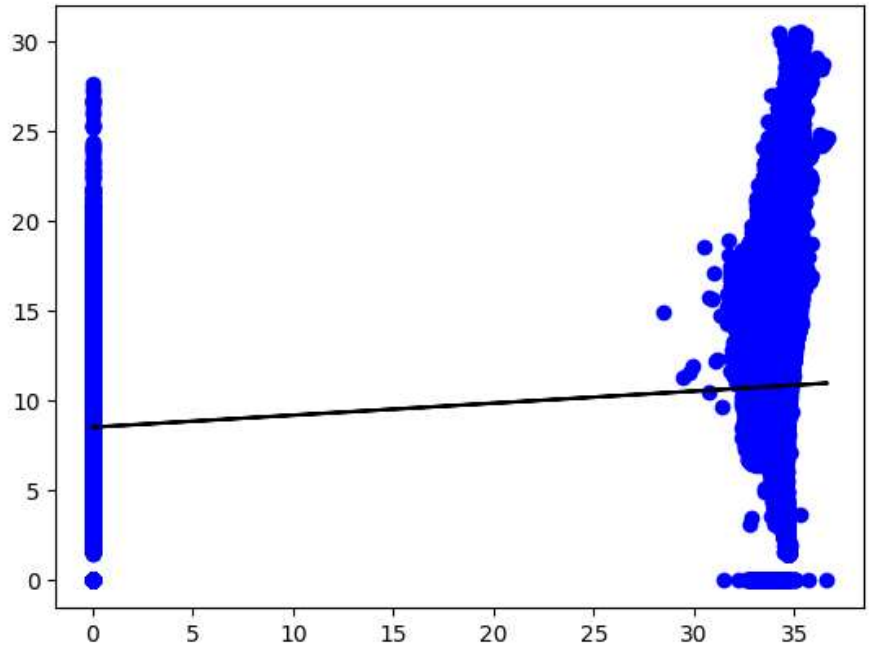
```
C:\Users\teppa\AppData\Local\Temp\ipykernel_1224\1379821321.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

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 [88]:
```
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))
```

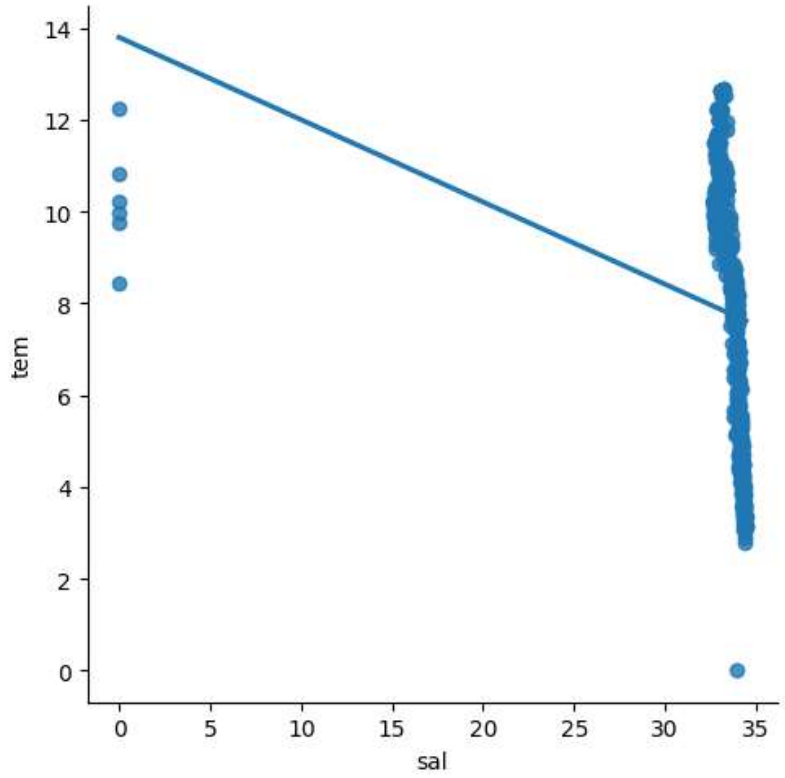
```
0.014828355646333113
```

In [89]:
```
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 [90]:
```
df500=df[:][:500]
sns.lmplot(x='sal',y='tem',data=df500,order=1,ci=None)
```

Out[90]: <seaborn.axisgrid.FacetGrid at 0x17c29516860>

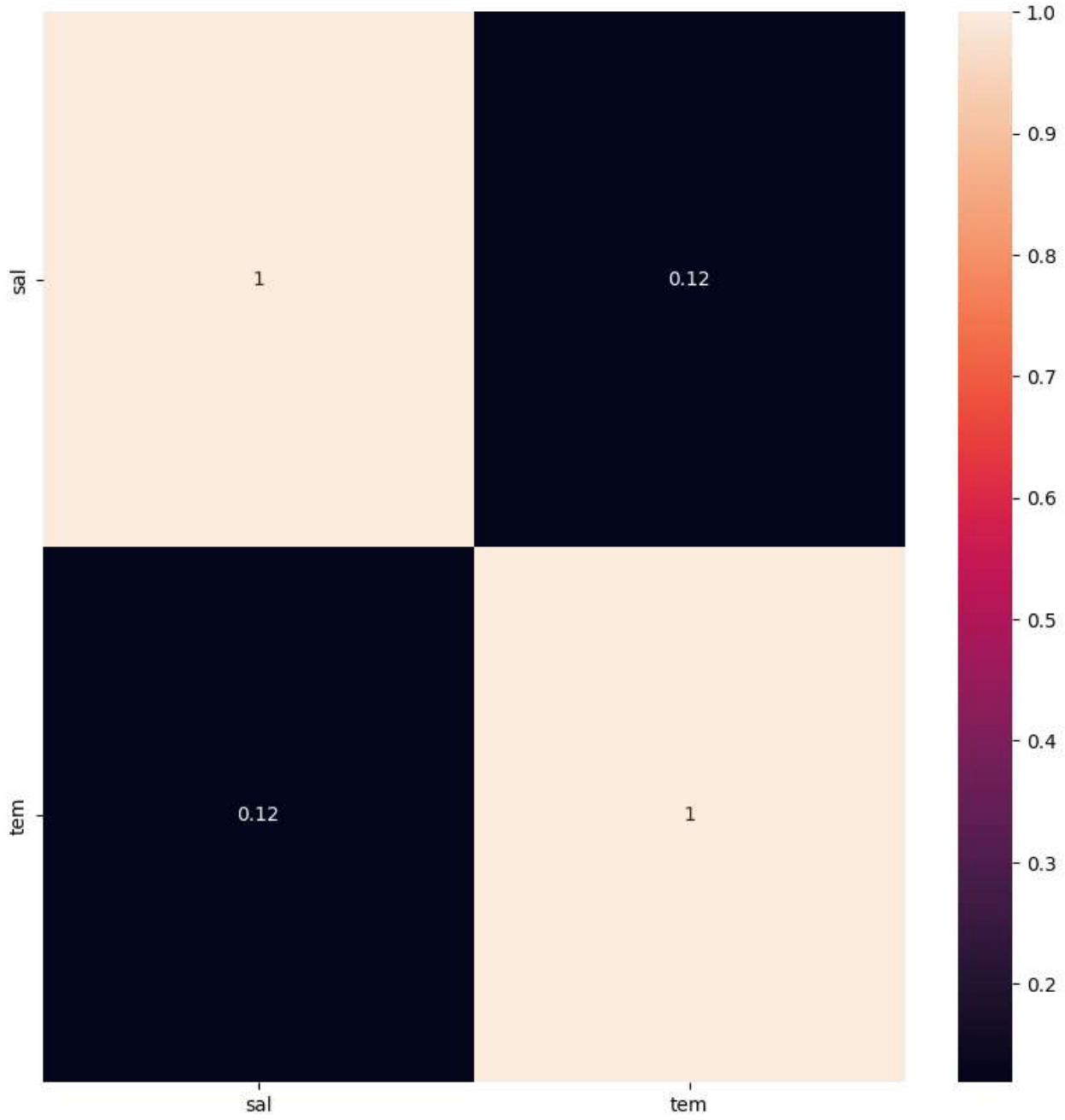


In [91]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
model=LinearRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
r2=r2_score(y_test,y_pred)
print("R2 score:",r2)
```

```
R2 score: 0.014828355646333113
```

In [92]:
```
plt.figure(figsize = (10, 10))
sns.heatmap(df.corr(), annot = True)
```

Out[92]: <Axes: >



In [93]:
```
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=17)
print("The dimension of X_train is {}".format(X_train.shape))
print("The dimension of X_test is {}".format(X_test.shape))
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 [94]:
```
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 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 [95]:
```
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))
```

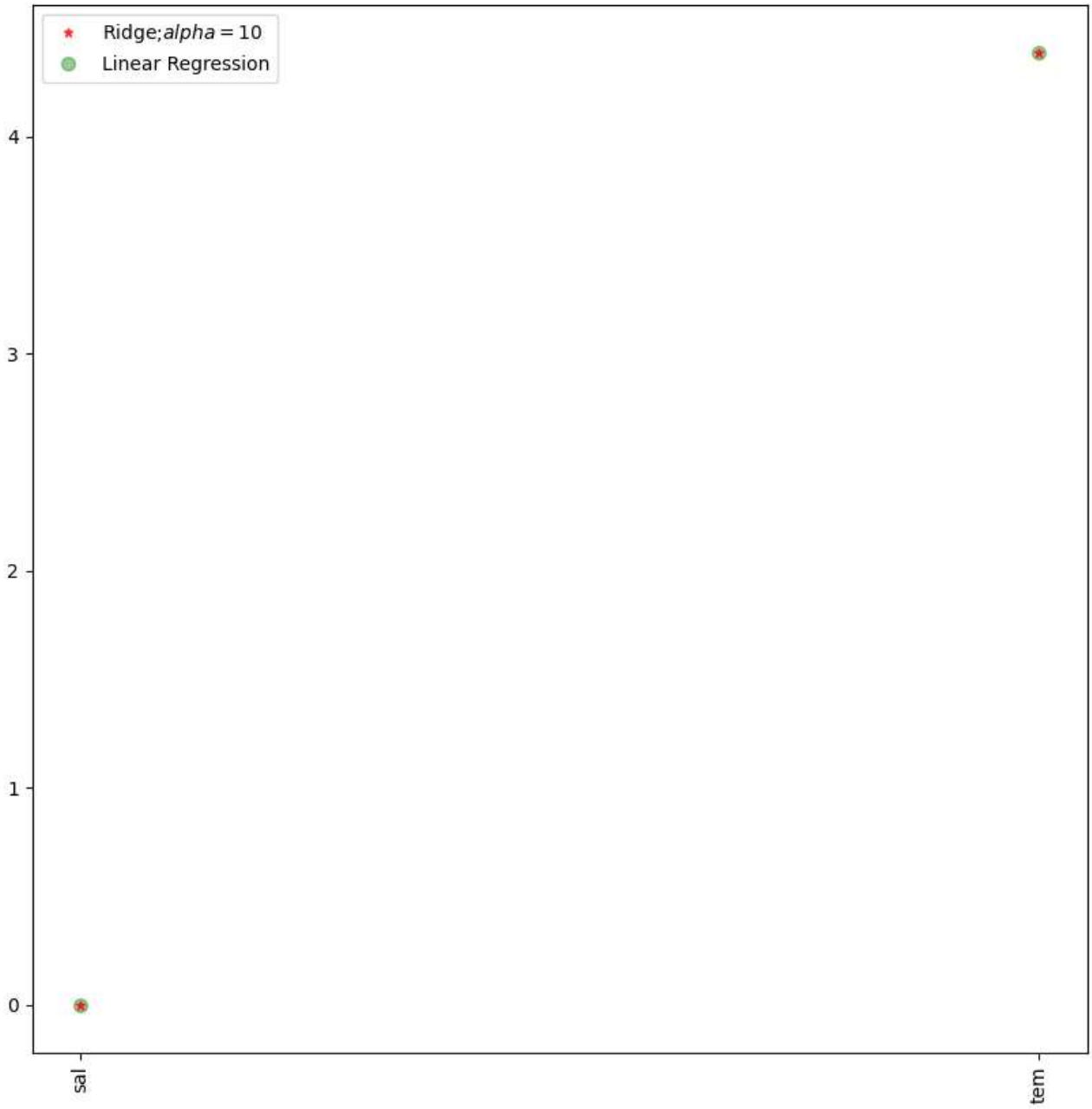
```
Ridge Model:

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

In [96]:
```
from sklearn.linear_model import Ridge,Lasso
from sklearn.preprocessing  import StandardScaler
```

In [98]:

```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='None',marker='*',markersize=5,color='red',lal
plt.plot(features,lr.coef_,alpha=0.4,linestyle="none",marker='o',markersize=7,color='green',label=
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



#Lasso Model

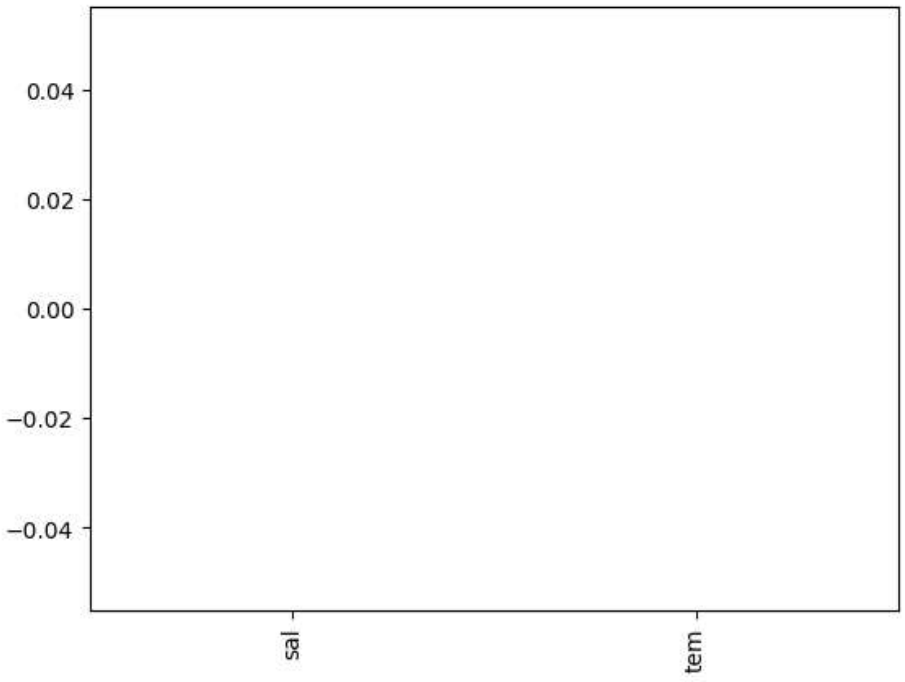
In [99]:
```
print("\nLasso Model: \n")
lasso = Lasso(alpha = 10)
lasso.fit(X_train,y_train)
train_score_ls =lasso.score(X_train,y_train)
test_score_ls =lasso.score(X_test,y_test)
print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is {}".format(test_score_ls))
```

```
Lasso Model:

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

In [100]:
```
pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "bar")
```

Out[100]: <Axes: >



In [101]:
```

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
```

In [102]:
```

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

```
0.9999999994806811
0.9999999994806712
```

In [ ]:
```

```