Linear Regression- Salinity and temperature

In [22]:

```
#step 1-importing all the requried libraries
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 [2]:

```
#Step-2: Reading the Dataset
df=pd.read_csv(r"C:\Users\jyothi reddy\Downloads\bottle.csv.zip")
df
```

```
C:\Users\jyothi reddy\AppData\Local\Temp\ipykernel_13408\715749884.py:2: DtypeWarning: Co
lumns (47,73) have mixed types. Specify dtype option on import or set low_memory=False.
  df=pd.read_csv(r"C:\Users\jyothi reddy\Downloads\bottle.csv.zip")
```

Out[2]:

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depth	T_degC	Salnty	O2ml_L	STheta	O2Sat	...	R_PHAEO
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
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
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
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
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
...	...	...	...	...	...	...	...	...	...	...	...	...
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
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
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
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
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

864863 rows × 74 columns

In [4]:
```
df=df[['Salnty','T_degC']]
df.columns=['Sal','Temp']
```

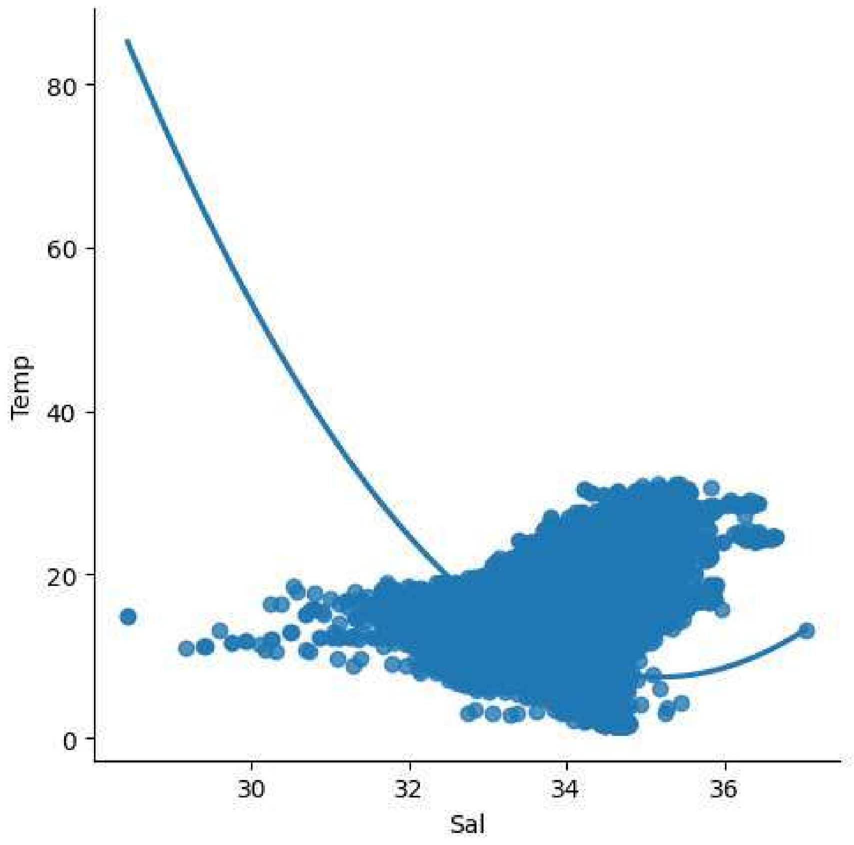
In [5]:
```
df.head()
```

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

In [6]:
```
#Step-3: Exploring the Data Scatter - plotting the data scatter

sns.lmplot(x="Sal",y="Temp", data = df, order = 2, ci = None)
df.describe()
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.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 [10]:
```
#Step-4: Data cleaning - Eliminating NaN OR missing input numbers

df.fillna(method ='ffill', inplace = True)
```

```
C:\Users\jyothi reddy\AppData\Local\Temp\ipykernel_13408\3532286049.py:3: SettingWithCopy
Warning:
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_g
uide/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(method ='ffill', inplace = True)
```

In [11]:
```
# Step-5: Training Our Model
X = np.array(df['Sal']).reshape(-1, 1)
y = np.array(df['Temp']).reshape(-1, 1)
#Seperating the data into independent and dependent variables and convert
#Now each dataset contains only one column
```

In [12]:
```
df.dropna(inplace = True)
```

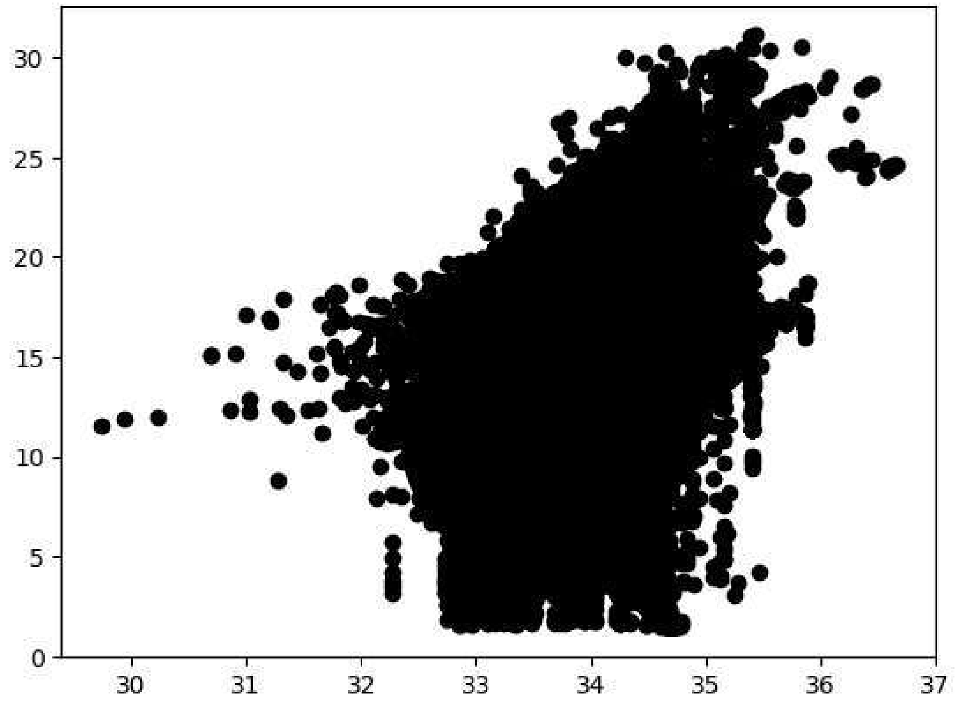
```
C:\Users\jyothi reddy\AppData\Local\Temp\ipykernel_13408\1791587065.py:1: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/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 [13]:
```
X_train,X_test,y_train,y_test = train_test_split(X, y, test_size = 0.25)
# Splitting the data into training data and test data
regr = LinearRegression()
regr.fit(X_train, y_train)
print(regr.score(X_test, y_test))
```

```
0.20113565587943782
```

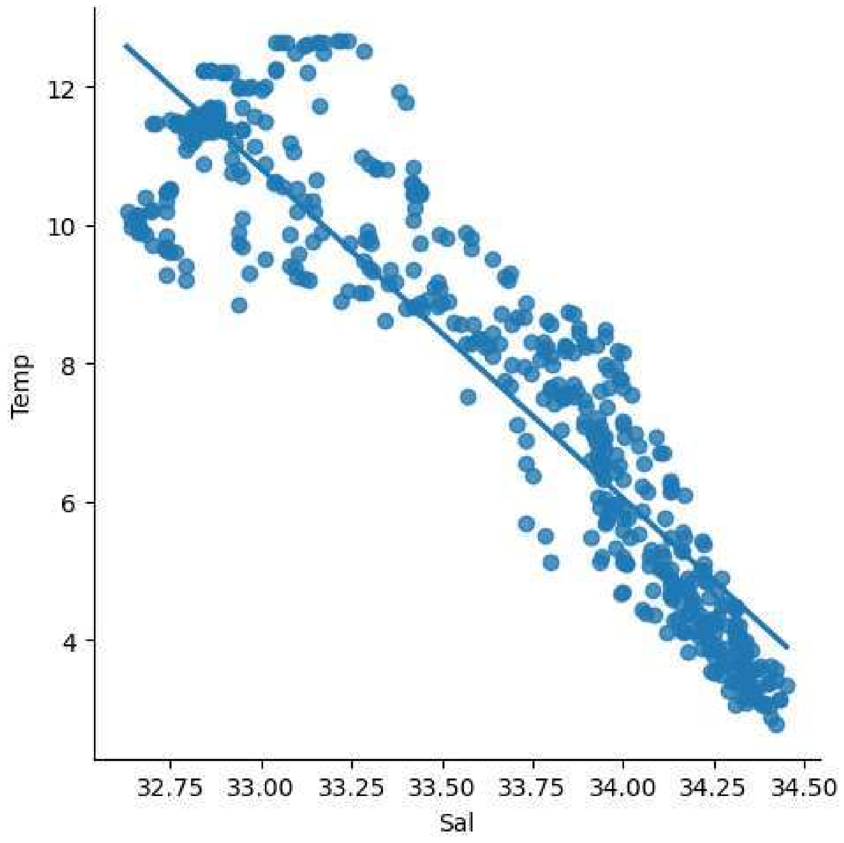
In [14]:

```
#step-6: Exploring Our Results
y_pred = regr.predict(X_test)
plt.scatter(X_test, y_test, color = 'k')
plt.show()
# Data scatter of predicted valueS
```



In [15]:
```
# Step-7: Working with a smaller Dataset
df500 = df[:][:500]
# Selecting the 1st 500 rows of the data
sns.lmplot(x ="Sal", y ="Temp", data = df500, order = 1, ci = None)
```

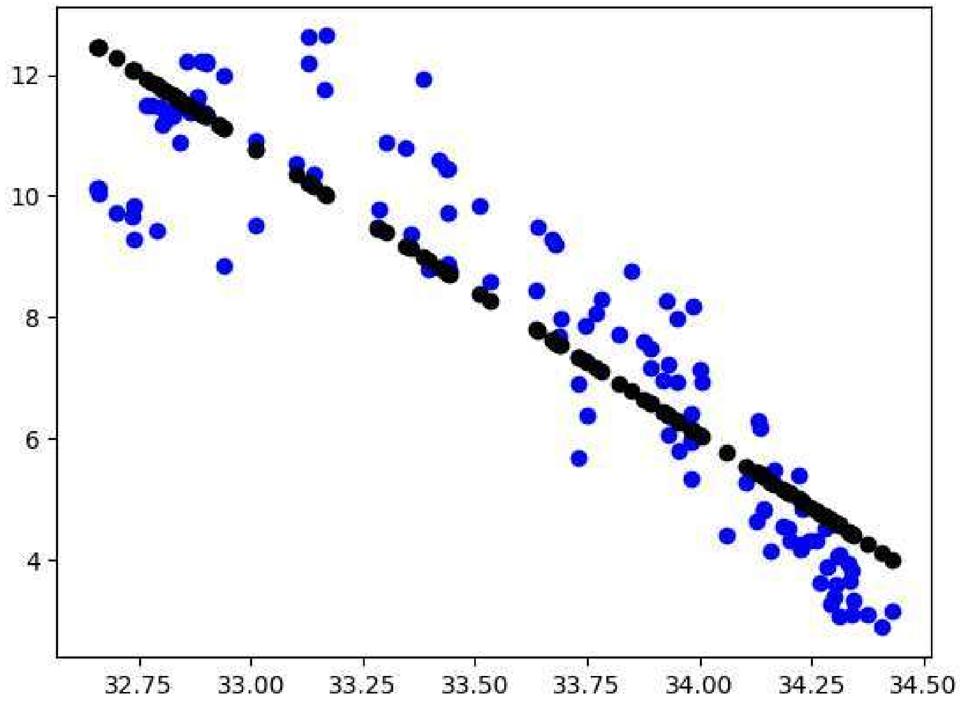
Out[15]:
```
<seaborn.axisgrid.FacetGrid at 0x133c17d0a50>
```



In [16]:

```
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.scatter(X_test, y_pred, color = 'k')
plt.show()
```

```
Regression: 0.846425087424722
```



In [17]:

```
#Step-8: Evaluation of model
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
#Train the model
model = LinearRegression()
model.fit(X_train, y_train)
#Evaluating the model on the test set
y_pred = model.predict(X_test)
r2 = r2_score(y_test, y_pred)
print("R2 score:",r2)
```

```
R2 score: 0.846425087424722
```

#step 9-conclusion: #Data set we have taken is poor for linear model but with the smaller data works well with Linear Model

In [ ]: