Linear Regression Model Step 1: PROBLEM STATEMENT - HOW BEST FIT THE DATASET ?

In [1]:
```
#Step 1: Importing All the Required 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\butyl\OneDrive\Desktop\Pri\Consensus\Assignment\GIST\DA
df
```

Out[2]:

	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	10.500	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	19-4903CR-HY-060-	10	10.460	33.4370	NaN	25.65400	NaN

In [3]:
```
df= df[['Salnty', 'T_degC']]

# Taking only the selected two attributes from the dataset

df.columns = ['Sal', 'Temp']

# Renaming the columns for easier writing of the code
```

In [4]:
```
df.head(10)

# Displaying only the 1st  rows along with the column names
```

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
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

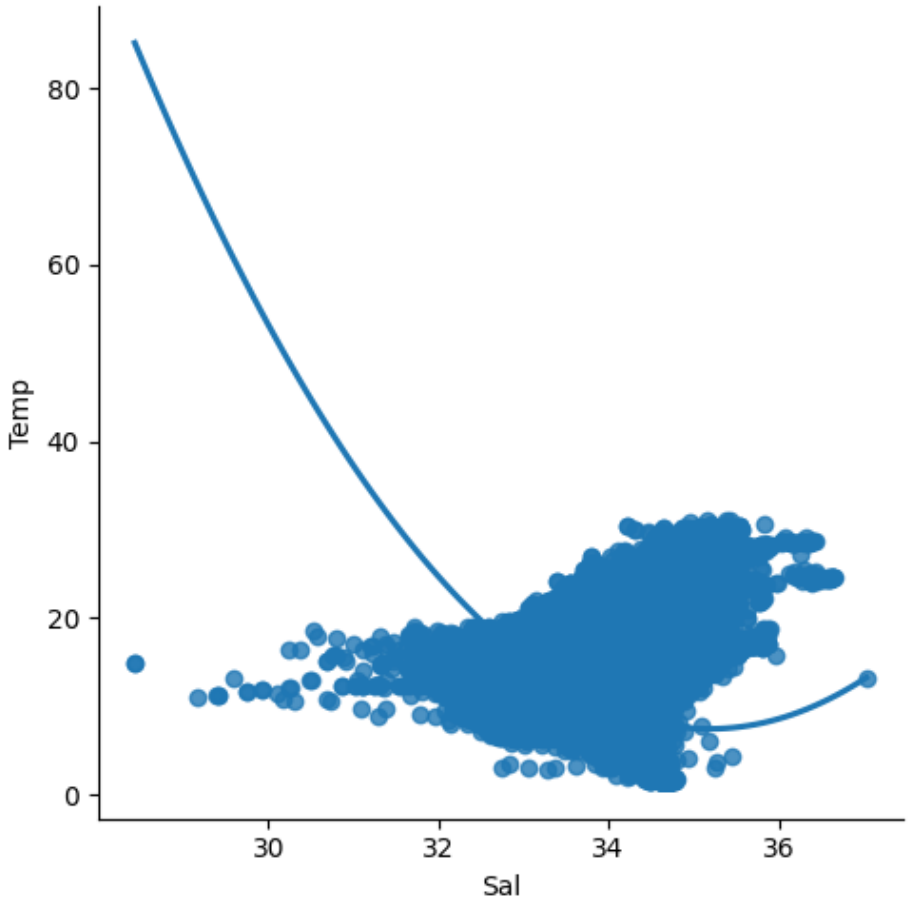
Out[4]:

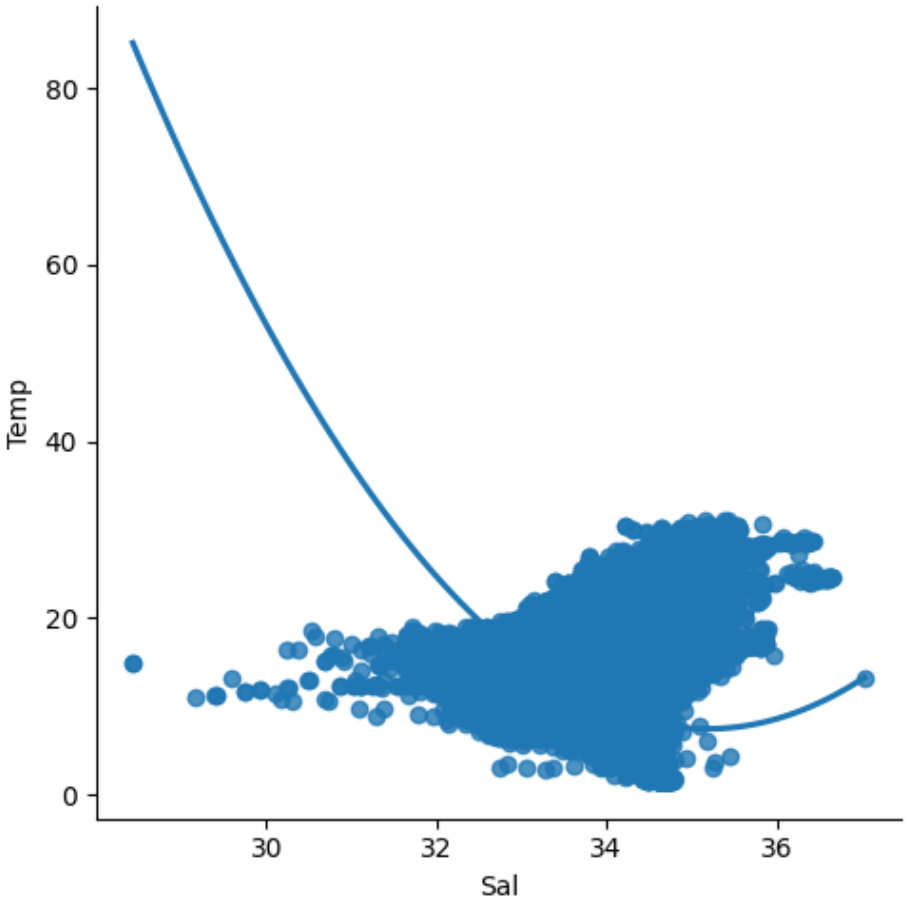
	Sal	Temp
0	33.440	10.50
1	33.440	10.46
2	33.437	10.46
3	33.420	10.45
4	33.421	10.45
5	33.431	10.45
6	33.440	10.45
7	33.424	10.24
8	33.420	10.06
9	33.494	9.86

In [5]:
```
# Step 3: Exploring the Data Scatter - Plotting the data scatter

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

Out[5]: <seaborn.axisgrid.FacetGrid at 0x28680004a90>

Out[5]: <seaborn.axisgrid.FacetGrid at 0x28680004a90>




80
60
40
20
0
Temp
30
32
34
36
Sal

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

Out[6]:

	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

Out[6]:

	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 [7]:
```
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
```

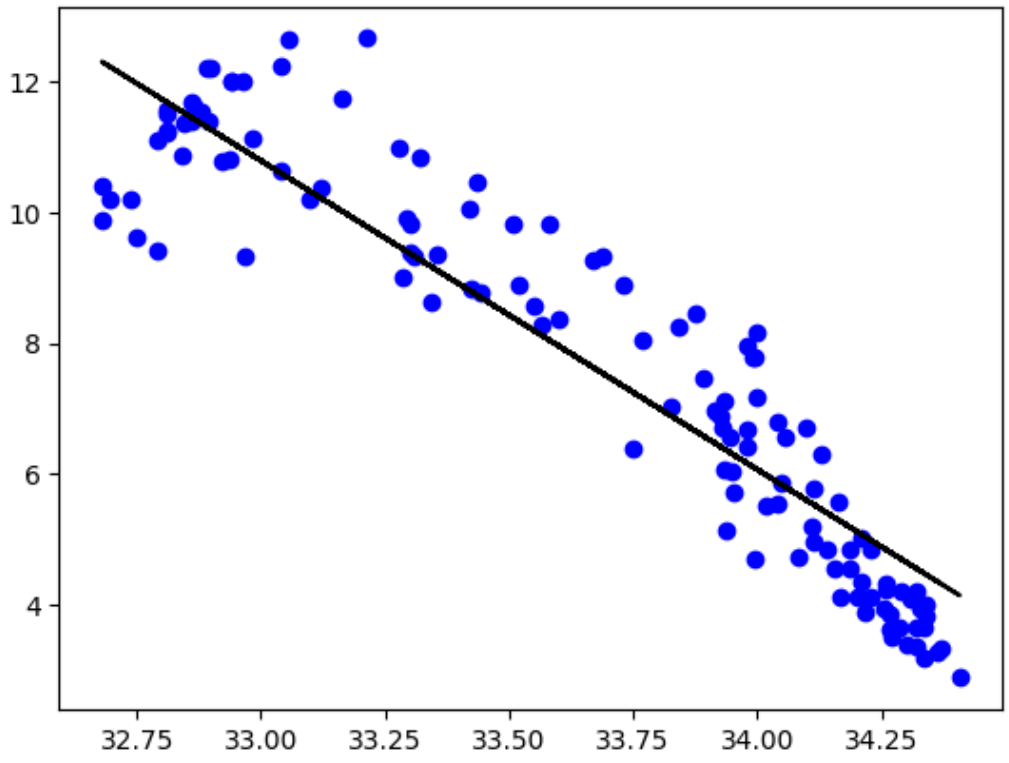
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 864863 entries, 0 to 864862
Data columns (total 2 columns):
 #   Column Non-Null Count   Dtype
--- ------ --------------   -----
 0   Sal   817509 non-null float64
 1   Temp   853900 non-null float64
dtypes: float64(2)
memory usage: 13.2 MB

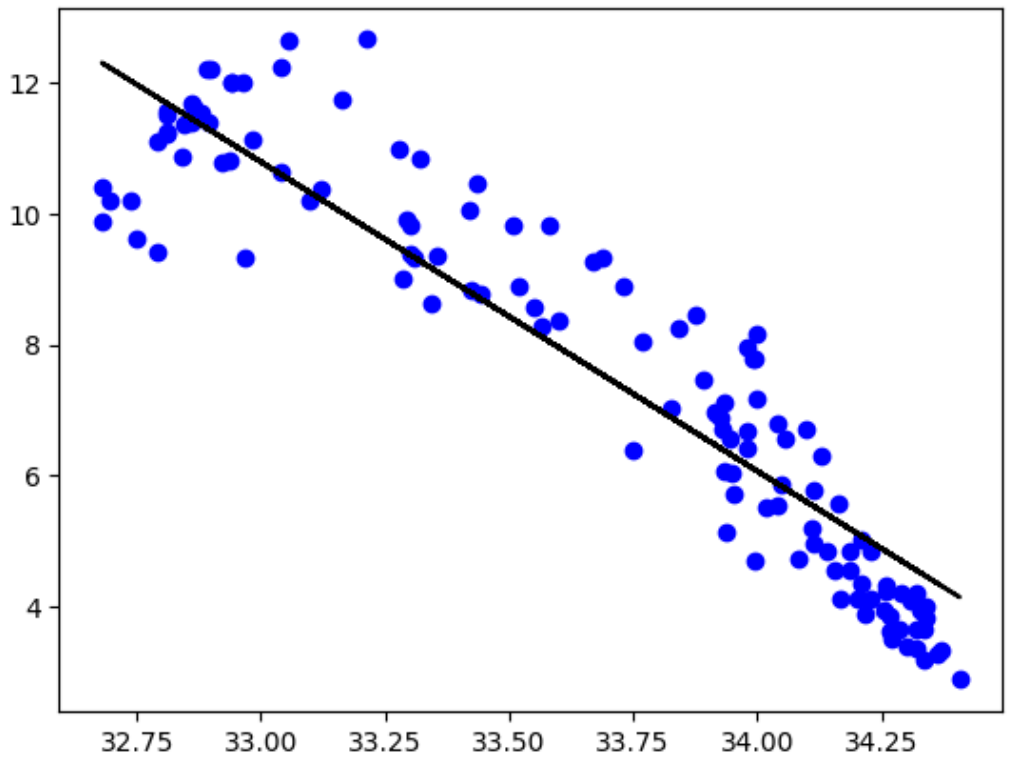
In [10]:
```
#Step 4: Data Cleaning - Eliminating NaN or missing input numbers

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

```
C:\Users\butyl\AppData\Local\Temp\ipykernel_21356\4132056527.py:3: SettingWithCop
yWarning:
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/stabl
e/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(method ='ffill', inplace = True)
C:\Users\butyl\AppData\Local\Temp\ipykernel_21356\4132056527.py:3: SettingWithCop
yWarning:
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/stabl
e/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(method ='ffill', inplace = True)
```

In [18]:
```
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:  0.8695899703952981
Regression:  0.8695899703952981
```





In [11]:
```
# Step 5: Training Our Model

X = np.array(df['Sal']).reshape(-1, 1)

y = np.array(df['Temp']).reshape(-1, 1)

# Separating the data into independent and dependent variables and Converting each

# Now each dataframe contains only one column
```

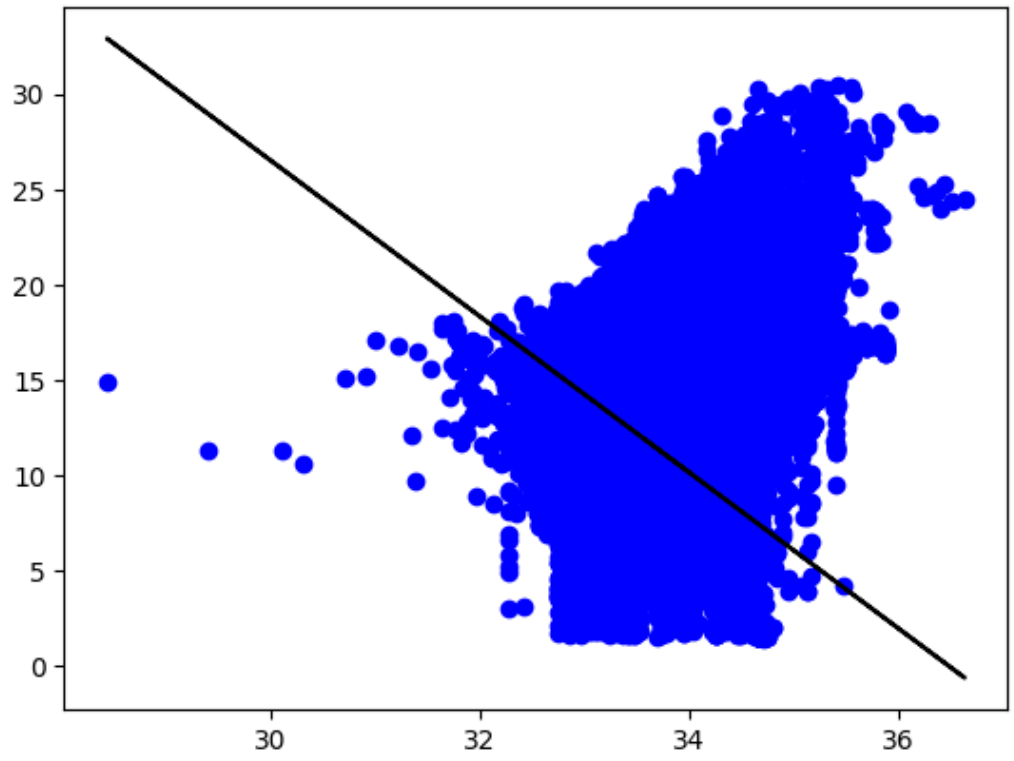
In [13]:
```
df.dropna(inplace = True)

# Dropping any rows with Nan values
```

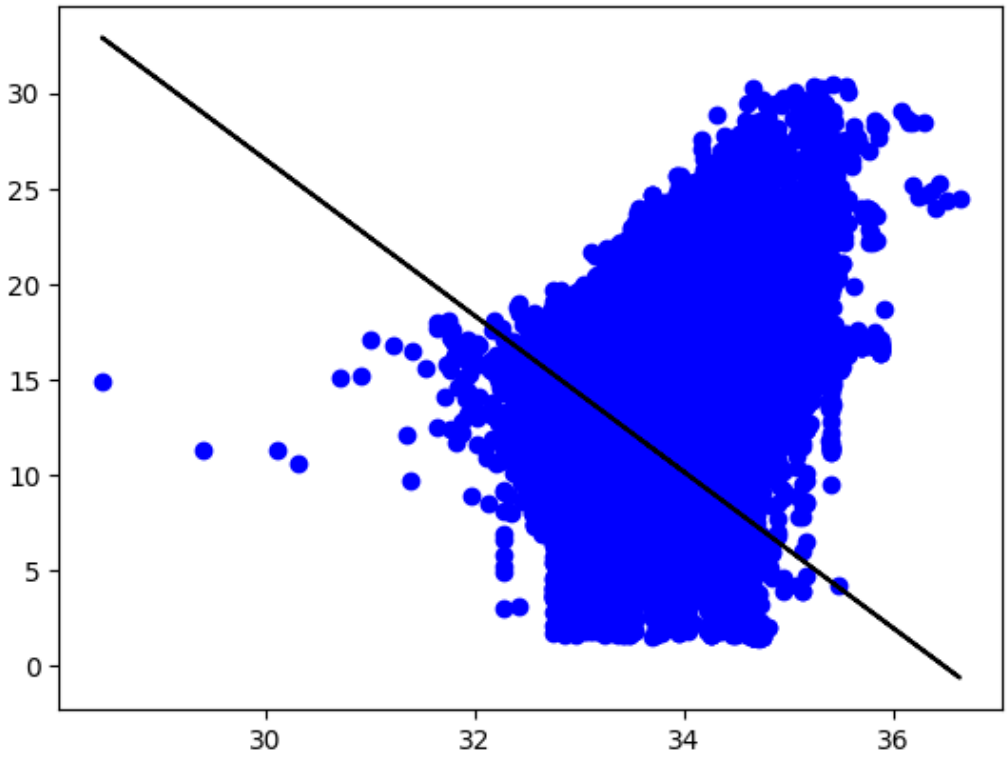
```
C:\Users\butyl\AppData\Local\Temp\ipykernel_21356\1365071386.py:1: SettingWithCop
yWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/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)
C:\Users\butyl\AppData\Local\Temp\ipykernel_21356\1365071386.py:1: SettingWithCop
yWarning:
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/stabl
e/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)

# Splitting the data into training and testing data

regr = LinearRegression()

regr.fit(X_train, y_train)

print(regr.score(X_test, y_test))
```

```
0.20859332007551334
0.20859332007551334
```

In [15]:
```
# Step 6: Exploring Our 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()

# Data scatter of predicted values
```

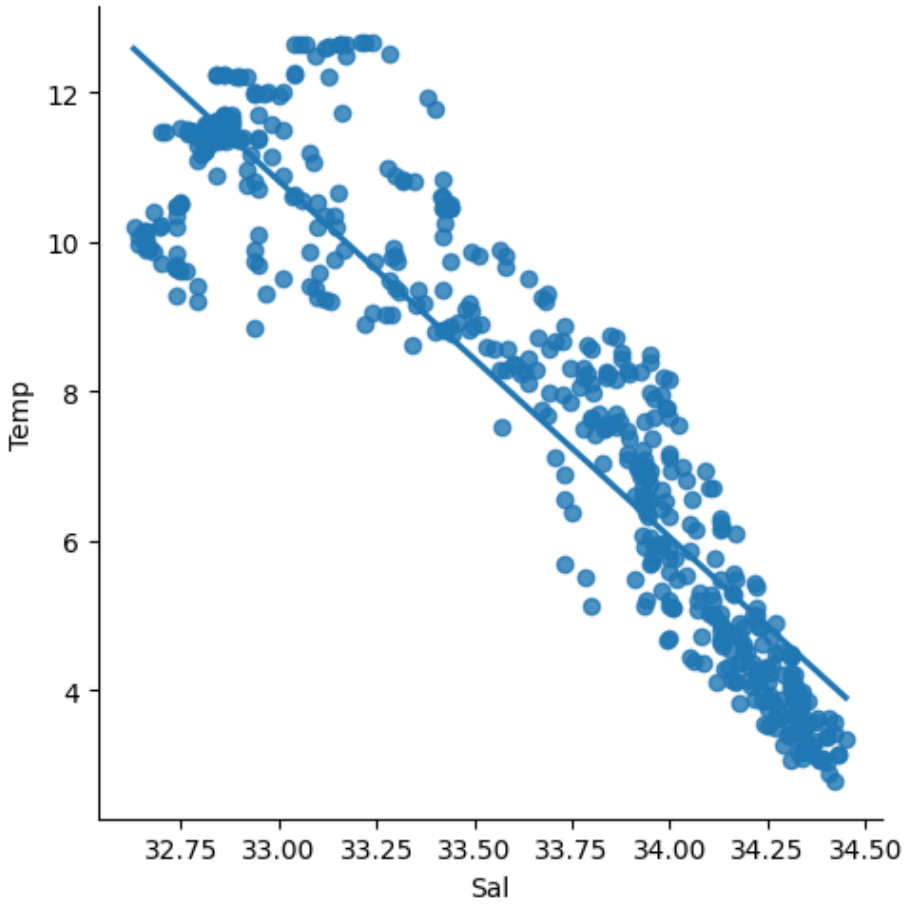


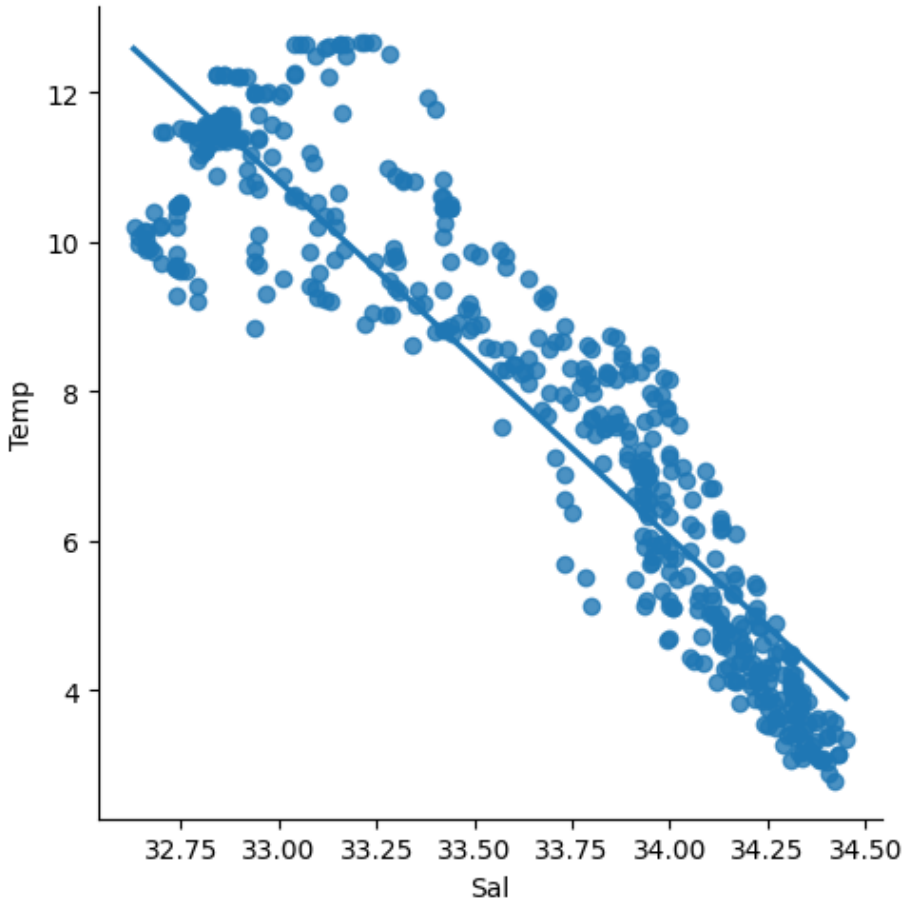

30
25
20
15
10
5
0
30
32
34
36

In [17]:
```
# 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[17]: <seaborn.axisgrid.FacetGrid at 0x28686cc9bd0>

Out[17]: <seaborn.axisgrid.FacetGrid at 0x28686cc9bd0>





In [19]:
```
#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)

# Evaluate 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: 0.8695899703952981
R2 score: 0.8695899703952981
```

Step 9: Conclusion:

Dataset we have taken is poor for Linear Model but with the smaller data works well with linear model