Linear Regression how best fit the data set(31-05-2023)

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 [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
```

In [2]:

```
#step 2: Reading the Dataset
df = pd.read_csv(r"C:\Users\smb06\Downloads\bottle.csv.zip")
df
```

```
C:\Users\smb06\AppData\Local\Temp\ipykernel_1368\1570419062.py:2: DtypeWar
ning: Columns (47,73) have mixed types. Specify dtype option on import or
set low_memory=False.
  df = pd.read_csv(r"C:\Users\smb06\Downloads\bottle.csv.zip")
```

Out[2]:

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta
0	1	1	054.0 056.0	19-4903CR-HY-060-0930-05400560-0000A-3	0	10.500	33.4400	NaN	25.64900
1	1	2	054.0 056.0	19-4903CR-HY-060-0930-05400560-0008A-3	8	10.460	33.4400	NaN	25.65600
2	1	3	054.0 056.0	19-4903CR-HY-060-0930-05400560-0010A-7	10	10.460	33.4370	NaN	25.65400
3	1	4	054.0 056.0	19-4903CR-HY-060-0930-05400560-0019A-3	19	10.450	33.4200	NaN	25.64300
4	1	5	054.0 056.0	19-4903CR-HY-060-0930-05400560-0020A-7	20	10.450	33.4210	NaN	25.64300
...	...	...	...	...	...	...	...	...	...
864858	34404	864859	093.4 026.4	20-1611SR-MX-310-2239-09340264-0000A-7	0	18.744	33.4083	5.805	23.87055
864859	34404	864860	093.4 026.4	20-1611SR-MX-310-2239-09340264-0002A-3	2	18.744	33.4083	5.805	23.87072
864860	34404	864861	093.4 026.4	20-1611SR-MX-310-2239-09340264-0005A-3	5	18.692	33.4150	5.796	23.88911
864861	34404	864862	093.4 026.4	20-1611SR-MX-310-2239-09340264-0010A-3	10	18.161	33.4062	5.816	24.01426

In [3]:

```
df = df[['Salnty','T_degC']]
#Taking only selected two attributes from dataset
df.columns = ['sal','Temp']
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
Cell In[3], line 1
----> 1 df = df[['Salnty','T_degC']]
      2 #Taking only selected two attributes from dataset
      3 df.columns = ['sal','Temp']

NameError: name 'df' is not defined
```

In [4]:

```
df.head()
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
Cell In[4], line 1
----> 1 df.head()

NameError: name 'df' is not defined
```

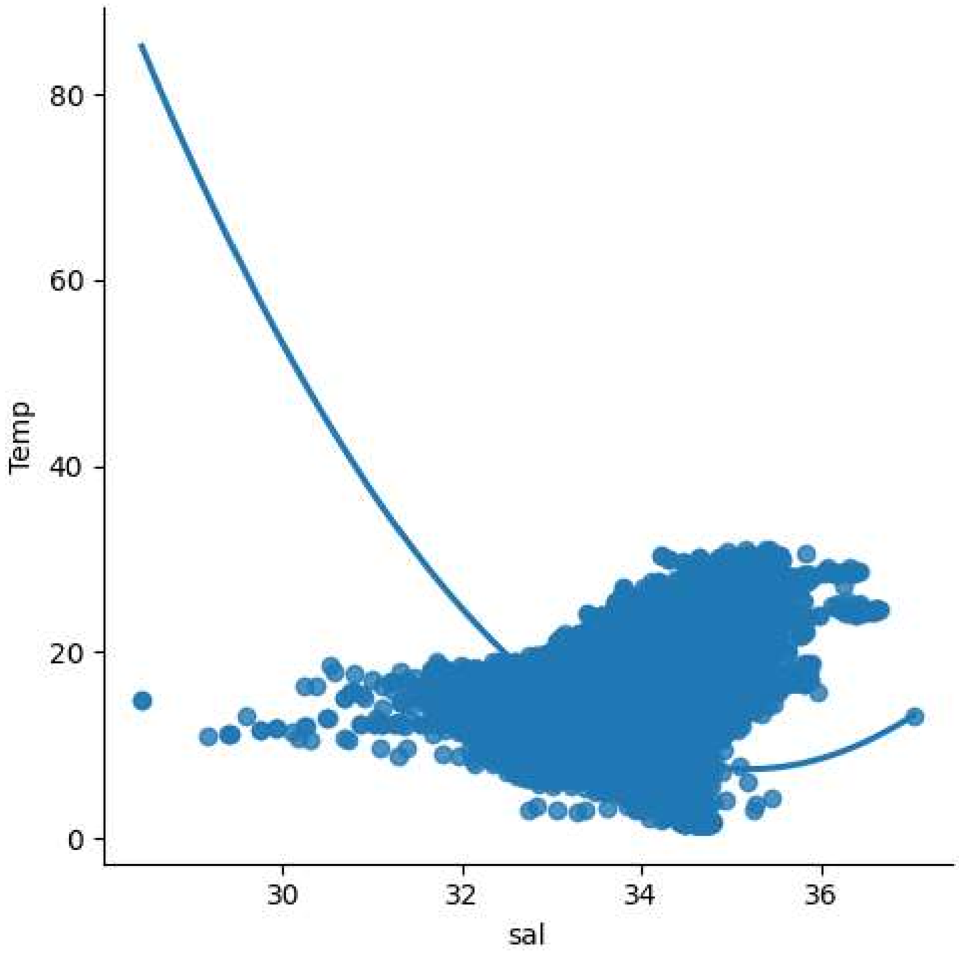
	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta
864862	34404	864863	093.4 026.4	20- 1611SR- MX-310- 2239- 09340264- 0015A-3	15	17.533	33.3880	5.774	24.15297

In [12]:

```
#Step-3: Exploring the Data Scatter - plotting the data scatter
sns.lmplot(x="sal",y="Temp",data=df,order=2,ci=None)
```

Out[12]:

```
<seaborn.axisgrid.FacetGrid at 0x18d1dc66650>
```



In [13]:

```
#step 4: Dta cleaning - eliminating NON and missing input numbers
df.fillna(method='ffill',inplace=True)
```

```
C:\Users\smb06\AppData\Local\Temp\ipykernel_1368\477090421.py:2: SettingWi
thCopyWarning:
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-doc
s/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(method='ffill',inplace=True)
```

In [22]:

```
#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 convert
#how each dataframe contains only one coloumn
```

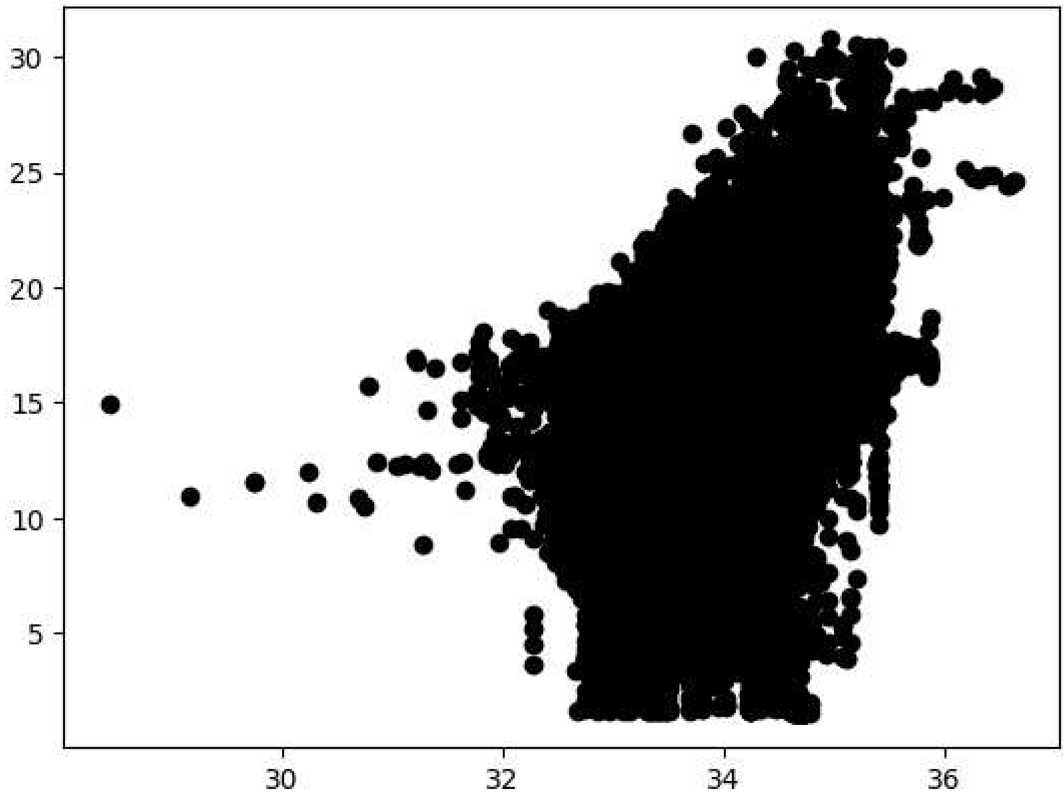
In [23]:

```
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))
```

```
0.20391771457966557
```

In [27]:

```
#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
```

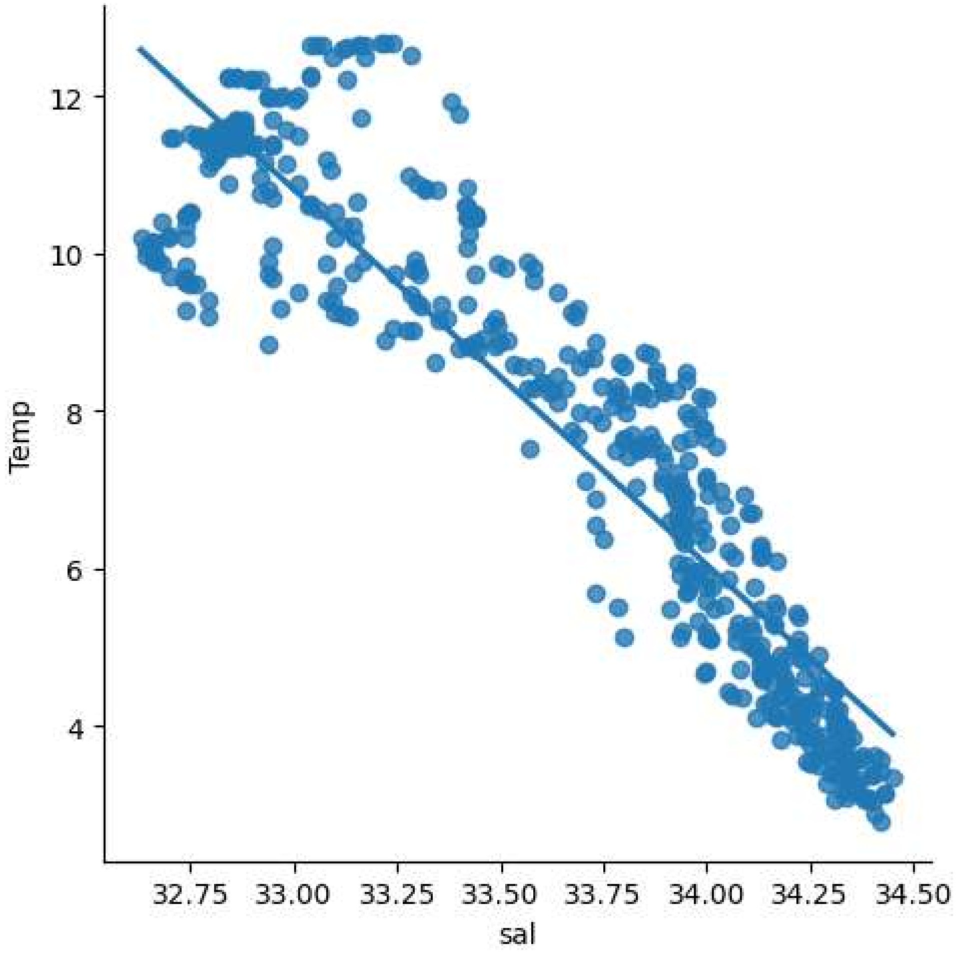


In [35]:

```
# 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[35]:

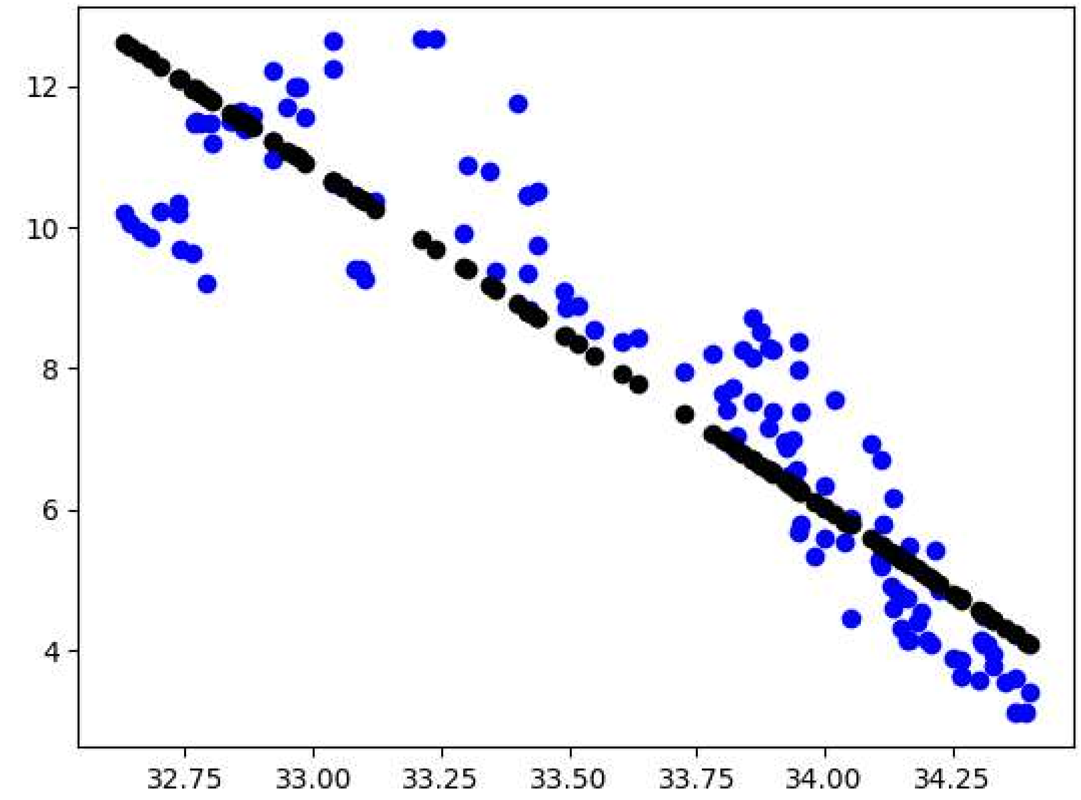
```
<seaborn.axisgrid.FacetGrid at 0x18d1e045610>
```



In [39]:

```
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: 0.8272272839837145
```



In [40]:

```
#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: 0.8272272839837145
```

```
#step9:conclusion
Dataset we have taken is poor for Linear Model,but with the smaller data works well
with Linear Model.
```