PROBLEM STATEMENT:How best fit the Dataset?

In [ ]:

```

```

##step 1:importing all required libraries

In [2]:

```
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
```

##step 2: Reading the dataset

In [3]:

```
df=pd.read_csv(r"C:\Users\MY HOME\Downloads\bottle.csv.zip")
df
```

```
C:\Users\MY HOME\AppData\Local\Temp\ipykernel_16540\76021945.py:1: DtypeWarning: Columns (47,73) have mixed typ
es. Specify dtype option on import or set low_memory=False.
  df=pd.read_csv(r"C:\Users\MY HOME\Downloads\bottle.csv.zip")
```

Out[3]:

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta	O2Sat	...	R_PHAEO	R_PRES	R_SAMP	DIC1
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	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	8	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	10	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	19	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	20	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	0	NaN	NaN

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta	O2Sat	...	R_PHAEO	R_PRES	R_SAMP	DIC1
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	4.0	NaN
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	3.0	NaN
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	2.0	NaN
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	1.0	NaN

864863 rows × 74 columns

In [4]:

```
#taking selected columns from dataset
df=df[['Salnty','T_degC']]
df
```

Out[4]:

```
         Salnty  T_degC
      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 [5]:

```
#Renamin columns for easier process(OPTIONAL)
df.columns=['sal','temp']
df
```

Out[5]:

```
             sal    temp
      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 [6]:

```
df.head(10)
```

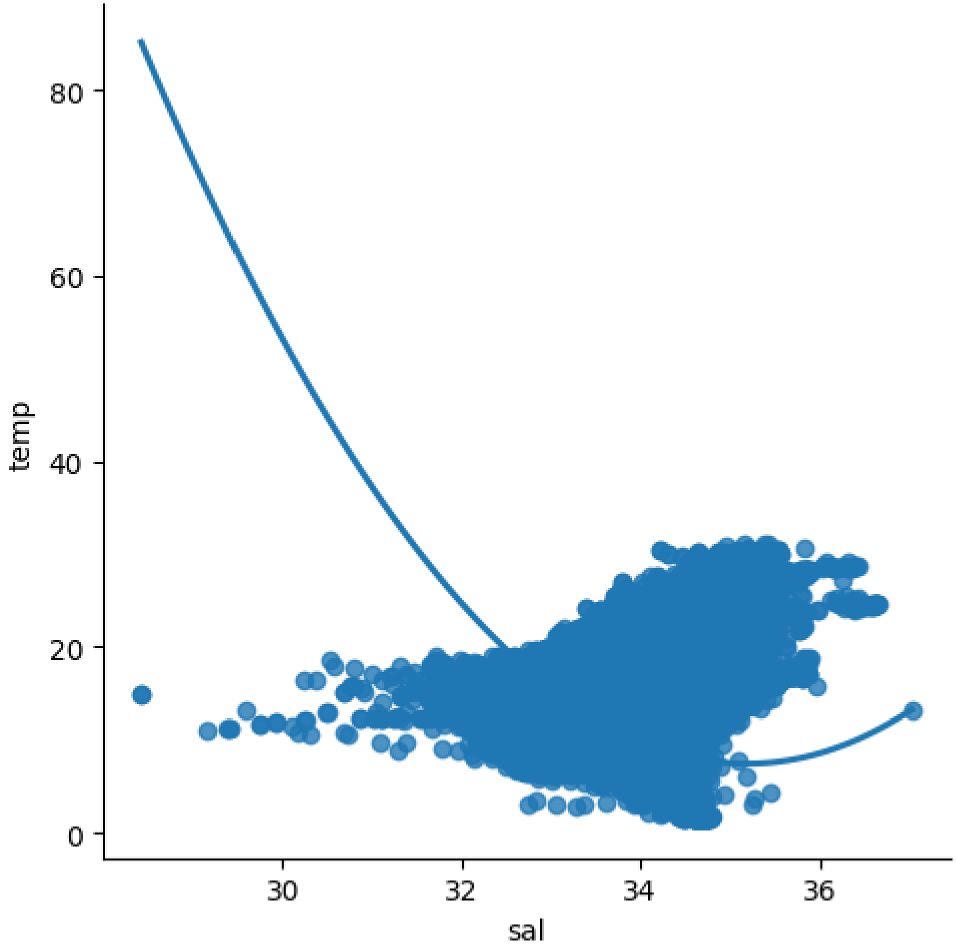
Out[6]:

```
      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
```

###step 3: Exploring to data scatter-pltting the data

In [7]:

```
sns.lmplot(x="sal",y="temp",order=2,data=df,ci=None)
plt.show()
```



In [13]:

```
df.describe()
```

Out[13]:

	sal	temp
count	814247.000000	814247.000000
mean	33.841337	10.860287
std	0.461636	4.224930
min	28.431000	1.440000
25%	33.489000	7.750000
50%	33.866000	10.110000
75%	34.197000	13.930000
max	37.034000	31.140000

In [14]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 814247 entries, 0 to 864862
Data columns (total 2 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   sal     814247 non-null  float64
 1   temp    814247 non-null  float64
dtypes: float64(2)
memory usage: 18.6 MB
```

###step 4:Data cleaning-Eliminating Nan/missing values

In [15]:

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

```
C:\Users\MY HOME\AppData\Local\Temp\ipykernel_16540\2729279820.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(method="ffill",inplace=True)
```

Out[15]:

	sal	temp
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

814247 rows × 2 columns

###step 5:Training our model

In [16]:

```
#Separating data into independent & dependent variables
#Now each dataframe contains only one coloumn
x=np.array(df['sal']).reshape(-1,1)
y=np.array(df['temp']).reshape(-1,1)
#Dropping any rows with Nan values
df.dropna(inplace=True)
df
```

```
C:\Users\MY HOME\AppData\Local\Temp\ipykernel_16540\1980608945.py:6: 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)
```

Out[16]:

	sal	temp
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

814247 rows × 2 columns

In [17]:

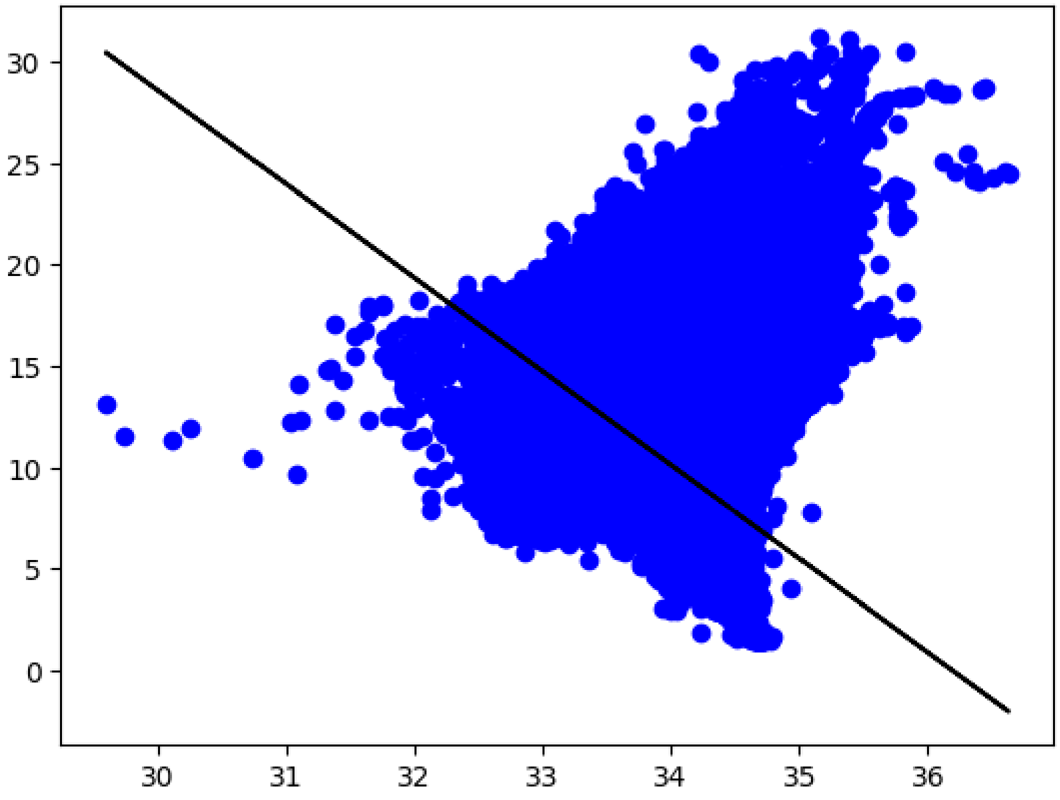
```
#Splitting the data into training and testing data
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

```
0.25789802084759594
```

###step 6:Exploring our results

In [18]:

```
#Data scatter to predict the values
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



###step 7:Working with the smaller dataset

In [19]:

```
#selecting the first 500 rows
df500=df[:][:500]
df500
```

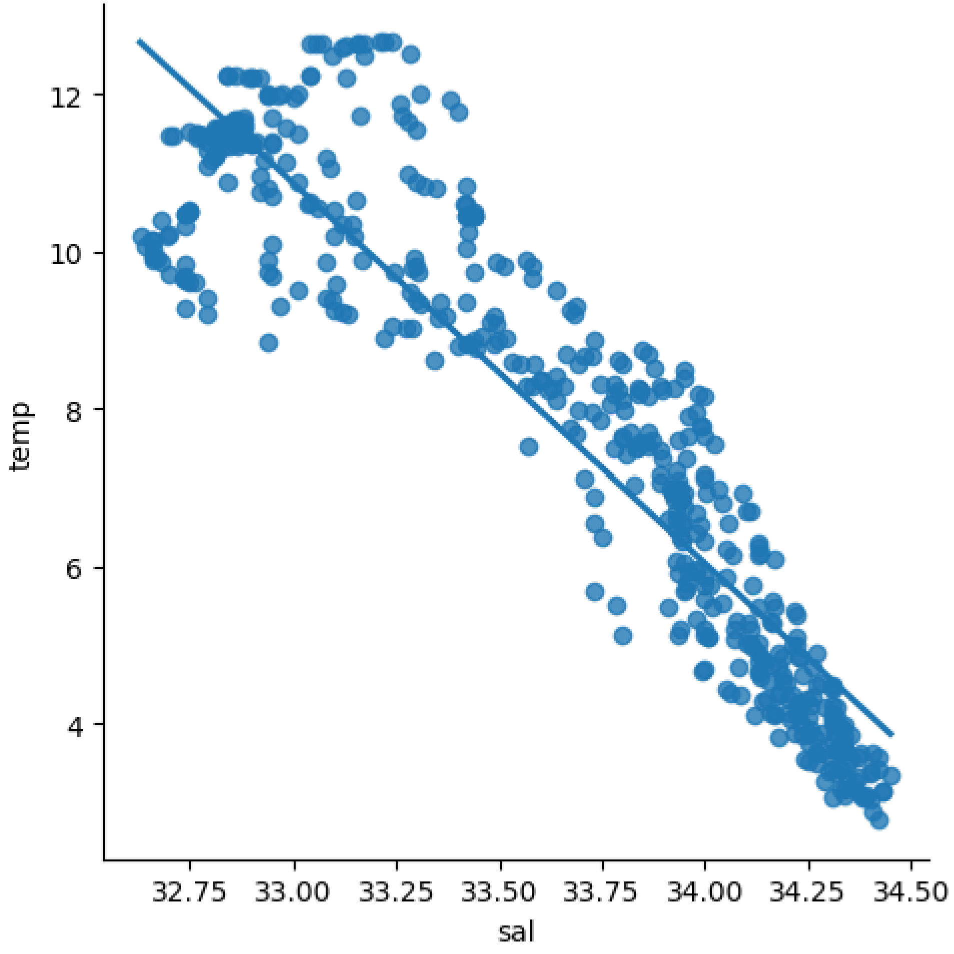
Out[19]:

```
        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
...      ...     ...
502   33.310   12.00
503   33.260   11.88
504   33.265   11.74
505   33.280   11.66
506   33.296   11.55
```

500 rows × 2 columns

In [20]:

```
sns.lmplot(x="sal",y="temp",data=df500,order=1,ci=None)
plt.show()
```



In [21]:

```
df500.fillna(method='ffill',inplace=True)
df500
```

Out[21]:

	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
...	...	...
502	33.310	12.00
503	33.260	11.88
504	33.265	11.74
505	33.280	11.66
506	33.296	11.55

500 rows × 2 columns

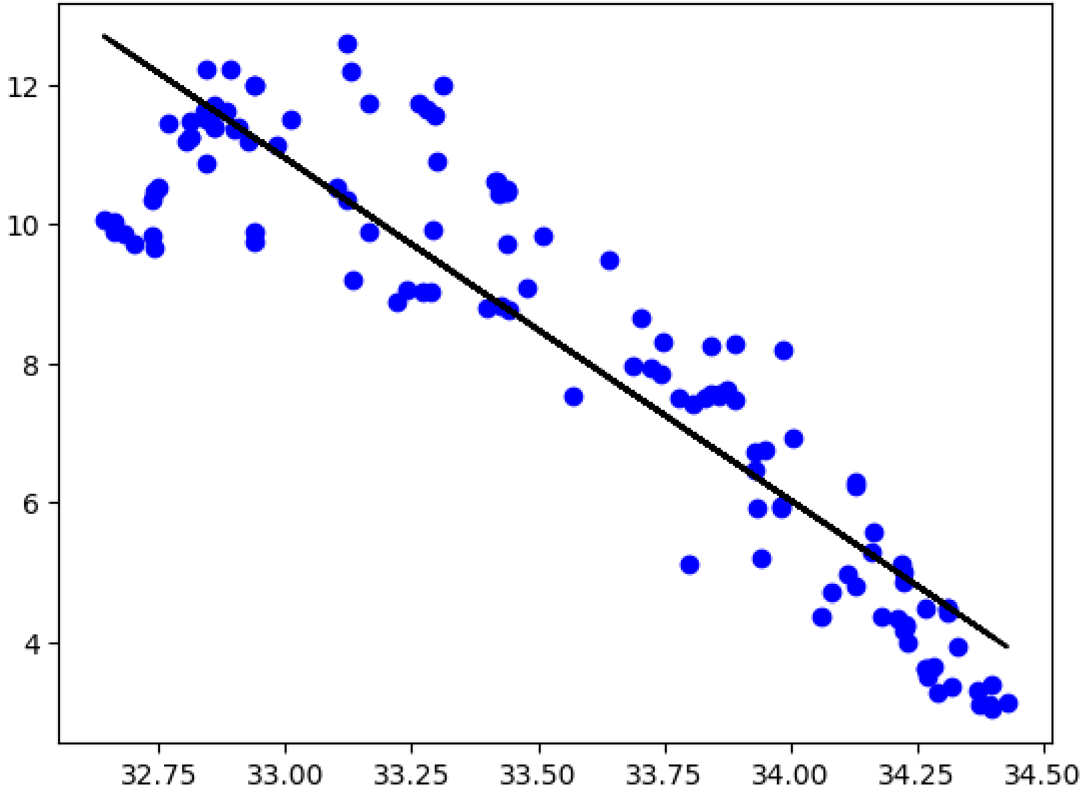
In [22]:

```
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))
```

```
regression: 0.831953305245337
```

In [23]:

```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



###step 8:Evaluation of model

In [24]:

```
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: 0.8319533305245337
```

conclusion:

```
 The dataset we have taken is not best fit.but,if we take small amount of data in this dataset it maybe fit
.
```