lication-of-supervised-learning-1

August 2, 2023

In this project we have used Machine Learning Algorithm (Inspired from AI) such as Supervised Machine Learning- Regression algorithm for obtaining a best fit curve for an input dataset of Thermoemf and temperature. Here, we have inputed the dataset as training data for machine learning and thereafter saved it as a dataframe using Pandas.Thereafter , we have constructed a LinearRegression model using Machine Learning algorithm ,especially SciKitLearn library even though it is a quadratic fit because LinearRegression is the fundamental module for any regression fit.Hence, the quadratic fitting is actually derived from the linearRegression model. Thereafter ,once the best fitting curve is obtained, we determine the parameters for the best fit polynomial using machine learning libraries. Henceforth,we generated an array of temperature with equispaced partitions of 10 degree Celsius. We have plotted the array with the best fit polynomial function generated from the plot for thermoemf and obtained the neutral temperature from the array itself by using the first and second derivative test.

[1]:
```
import numpy as np
from sklearn.preprocessing import PolynomialFeatures
import pandas as pd
#x=np.arange(20,420,10)
X_train=np.
  ↪array([29,45,50,55,61,64,70,77,79,84,89,94,98,105,111,116,121,126,130,135,140,143,148,155,1
Y_train=[0.113,0.150,0.189,0.227,0.272,0.299,0.330,0.377,0.396,0.427,0.461,0.
  ↪488,0.516,0.558,0.589,0.618,0.643,0.670,0.688,0.711,0.736,0.749,0.761,0.
  ↪799,0.818,0.832,0.850,0.870,0.888,0.905,0.921,0.934,0.945,0.954,0.962,0.
  ↪968,0.972,0.976,0.980,0.981,0.981,0.982,0.983,0.983,0.983,0.983,0.982,0.
  ↪980,0.979,0.976,0.971,0.968,0.960,0.953,0.944,0.936,0.927,0.917,0.901,0.
  ↪876,0.859,0.823,0.804,0.784,0.758,0.721,0.688,0.667,0.631,0.581,0.539,0.
  ↪496,0.454,0.410]
dict={'temperature':X_train,'Thermoemf':Y_train}
df=pd.DataFrame(dict)
df.to_csv('thermo.csv',sep=' ',index=False,header=True)
df_1=pd.read_csv('thermo.csv')
print(df)
```

```
    temperature  Thermoemf
0            29      0.113
1            45      0.150
2            50      0.189
3            55      0.227
4            61      0.272
```

```
..         …          …
69         390       0.581
70         397       0.539
71         404       0.496
72         411       0.454
73         418       0.410

[74 rows x 2 columns]
```

[86]:
```
poly=PolynomialFeatures(degree=4,include_bias=False)
poly_features=poly.fit_transform(X_train.reshape(-1,1))
print(poly_features)
```

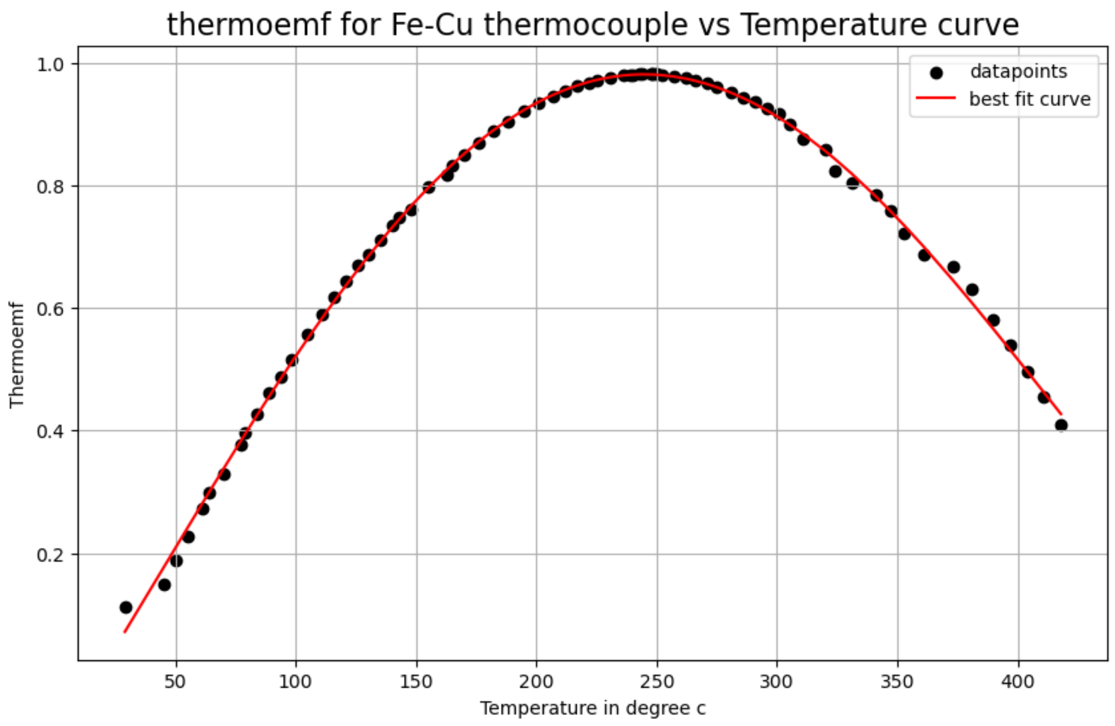
```
[[2.90000000e+01 8.41000000e+02 2.43890000e+04 7.07281000e+05]
 [4.50000000e+01 2.02500000e+03 9.11250000e+04 4.10062500e+06]
 [5.00000000e+01 2.50000000e+03 1.25000000e+05 6.25000000e+06]
 [5.50000000e+01 3.02500000e+03 1.66375000e+05 9.15062500e+06]
 [6.10000000e+01 3.72100000e+03 2.26981000e+05 1.38458410e+07]
 [6.40000000e+01 4.09600000e+03 2.62144000e+05 1.67772160e+07]
 [7.00000000e+01 4.90000000e+03 3.43000000e+05 2.40100000e+07]
 [7.70000000e+01 5.92900000e+03 4.56533000e+05 3.51530410e+07]
 [7.90000000e+01 6.24100000e+03 4.93039000e+05 3.89500810e+07]
 [8.40000000e+01 7.05600000e+03 5.92704000e+05 4.97871360e+07]
 [8.90000000e+01 7.92100000e+03 7.04969000e+05 6.27422410e+07]
 [9.40000000e+01 8.83600000e+03 8.30584000e+05 7.80748960e+07]
 [9.80000000e+01 9.60400000e+03 9.41192000e+05 9.22368160e+07]
 [1.05000000e+02 1.10250000e+04 1.15762500e+06 1.21550625e+08]
 [1.11000000e+02 1.23210000e+04 1.36763100e+06 1.51807041e+08]
 [1.16000000e+02 1.34560000e+04 1.56089600e+06 1.81063936e+08]
 [1.21000000e+02 1.46410000e+04 1.77156100e+06 2.14358881e+08]
 [1.26000000e+02 1.58760000e+04 2.00037600e+06 2.52047376e+08]
 [1.30000000e+02 1.69000000e+04 2.19700000e+06 2.85610000e+08]
 [1.35000000e+02 1.82250000e+04 2.46037500e+06 3.32150625e+08]
 [1.40000000e+02 1.96000000e+04 2.74400000e+06 3.84160000e+08]
 [1.43000000e+02 2.04490000e+04 2.92420700e+06 4.18161601e+08]
 [1.48000000e+02 2.19040000e+04 3.24179200e+06 4.79785216e+08]
 [1.55000000e+02 2.40250000e+04 3.72387500e+06 5.77200625e+08]
 [1.63000000e+02 2.65690000e+04 4.33074700e+06 7.05911761e+08]
 [1.65000000e+02 2.72250000e+04 4.49212500e+06 7.41200625e+08]
 [1.70000000e+02 2.89000000e+04 4.91300000e+06 8.35210000e+08]
 [1.76000000e+02 3.09760000e+04 5.45177600e+06 9.59512576e+08]
 [1.82000000e+02 3.31240000e+04 6.02856800e+06 1.09719938e+09]
 [1.88000000e+02 3.53440000e+04 6.64467200e+06 1.24919834e+09]
 [1.95000000e+02 3.80250000e+04 7.41487500e+06 1.44590062e+09]
 [2.01000000e+02 4.04010000e+04 8.12060100e+06 1.63224080e+09]
 [2.07000000e+02 4.28490000e+04 8.86974300e+06 1.83603680e+09]
 [2.12000000e+02 4.49440000e+04 9.52812800e+06 2.01996314e+09]
 [2.17000000e+02 4.70890000e+04 1.02183130e+07 2.21737392e+09]
 [2.22000000e+02 4.92840000e+04 1.09410480e+07 2.42891266e+09]
 [2.25000000e+02 5.06250000e+04 1.13906250e+07 2.56289062e+09]
 [2.31000000e+02 5.33610000e+04 1.23263910e+07 2.84739632e+09]
 [2.36000000e+02 5.56960000e+04 1.31442560e+07 3.10204442e+09]
 [2.39000000e+02 5.71210000e+04 1.36519190e+07 3.26280864e+09]
 [2.40000000e+02 5.76000000e+04 1.38240000e+07 3.31776000e+09]
 [2.43000000e+02 5.90490000e+04 1.43489070e+07 3.48678440e+09]
 [2.44000000e+02 5.95360000e+04 1.45267840e+07 3.54453530e+09]
 [2.48000000e+02 6.15040000e+04 1.52529920e+07 3.78274202e+09]
 [2.48000000e+02 6.15040000e+04 1.52529920e+07 3.78274202e+09]
 [2.49000000e+02 6.20010000e+04 1.54382490e+07 3.84412400e+09]
 [2.50000000e+02 6.25000000e+04 1.56250000e+07 3.90625000e+09]
 [2.52000000e+02 6.35040000e+04 1.60030080e+07 4.03275802e+09]
 [2.57000000e+02 6.60490000e+04 1.69745930e+07 4.36247040e+09]
 [2.62000000e+02 6.86440000e+04 1.79847280e+07 4.71199874e+09]
 [2.66000000e+02 7.07560000e+04 1.88210960e+07 5.00641154e+09]
 [2.71000000e+02 7.34410000e+04 1.99025110e+07 5.39358048e+09]
 [2.75000000e+02 7.56250000e+04 2.07968750e+07 5.71914062e+09]
 [2.81000000e+02 7.89610000e+04 2.21880410e+07 6.23483952e+09]
 [2.86000000e+02 8.17960000e+04 2.33936560e+07 6.69058562e+09]
 [2.91000000e+02 8.46810000e+04 2.46421710e+07 7.17087176e+09]
 [2.96000000e+02 8.76160000e+04 2.59343360e+07 7.67656346e+09]
 [3.01000000e+02 9.06010000e+04 2.72709010e+07 8.20854120e+09]
 [3.05000000e+02 9.30250000e+04 2.83726250e+07 8.65365062e+09]
 [3.11000000e+02 9.67210000e+04 3.00802310e+07 9.35495184e+09]
 [3.20000000e+02 1.02400000e+05 3.27680000e+07 1.04857600e+10]
 [3.24000000e+02 1.04976000e+05 3.40122240e+07 1.10199606e+10]
 [3.31000000e+02 1.09561000e+05 3.62646910e+07 1.20036127e+10]
 [3.41000000e+02 1.16281000e+05 3.96518210e+07 1.35212710e+10]
 [3.47000000e+02 1.20409000e+05 4.17819230e+07 1.44983273e+10]
 [3.53000000e+02 1.24609000e+05 4.39869770e+07 1.55274029e+10]
 [3.61000000e+02 1.30321000e+05 4.70458810e+07 1.69835630e+10]
 [3.73000000e+02 1.39129000e+05 5.18951170e+07 1.93568786e+10]
 [3.81000000e+02 1.45161000e+05 5.53063410e+07 2.10717159e+10]
 [3.90000000e+02 1.52100000e+05 5.93190000e+07 2.31344100e+10]
 [3.97000000e+02 1.57609000e+05 6.25707730e+07 2.48405969e+10]
 [4.04000000e+02 1.63216000e+05 6.59392640e+07 2.66394627e+10]
 [4.11000000e+02 1.68921000e+05 6.94265310e+07 2.85343042e+10]
 [4.18000000e+02 1.74724000e+05 7.30346320e+07 3.05284762e+10]]
```

[87]:
```
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
```

[88]:
```
poly_reg_model=LinearRegression()
poly_reg_model.fit(poly_features,Y_train)
Y_predicted=poly_reg_model.predict(poly_features)
print(Y_predicted)
```

```
[0.07148059 0.17570823 0.20835805 0.24093505 0.27984563 0.29919671
 0.33762835 0.38190155 0.3944201  0.4254277  0.4559823  0.48603477
 0.50968328 0.55015163 0.58383294 0.61113622 0.63770202 0.66349229
 0.68354168 0.70784573 0.731277   0.74490369 0.76687388 0.79602857
 0.8269732  0.83430246 0.85189779 0.87161608 0.8897803  0.90636104
 0.92366962 0.93673568 0.94814883 0.95638664 0.96346007 0.96936443
 0.97234459 0.97703786 0.97965858 0.98066869 0.98091181 0.98136089
 0.98141727 0.98117743 0.98117743 0.98100138 0.980779   0.9801955
 0.97793051 0.97452132 0.97097744 0.96553738 0.96038595 0.95134532
 0.94262661 0.9328512  0.92204009 0.91021599 0.90004374 0.88362842
 0.85649871 0.84351527 0.81949    0.7824555  0.758813   0.73418844
 0.69995502 0.64598025 0.60853546 0.5653168  0.53109759 0.49652136
 0.4617524  0.42696167]
```

[89]:
```
plt.figure(figsize=(10,6))
plt.title('thermoemf for Fe-Cu thermocouple vs Temperature curve',size=16)
plt.scatter(X_train,Y_train,c='black',label='datapoints')
plt.plot(X_train,Y_predicted,c='red',label='best fit curve')
plt.xlabel('Temperature in degree c')
plt.ylabel('Thermoemf')
plt.legend()
plt.grid()
plt.show()
```

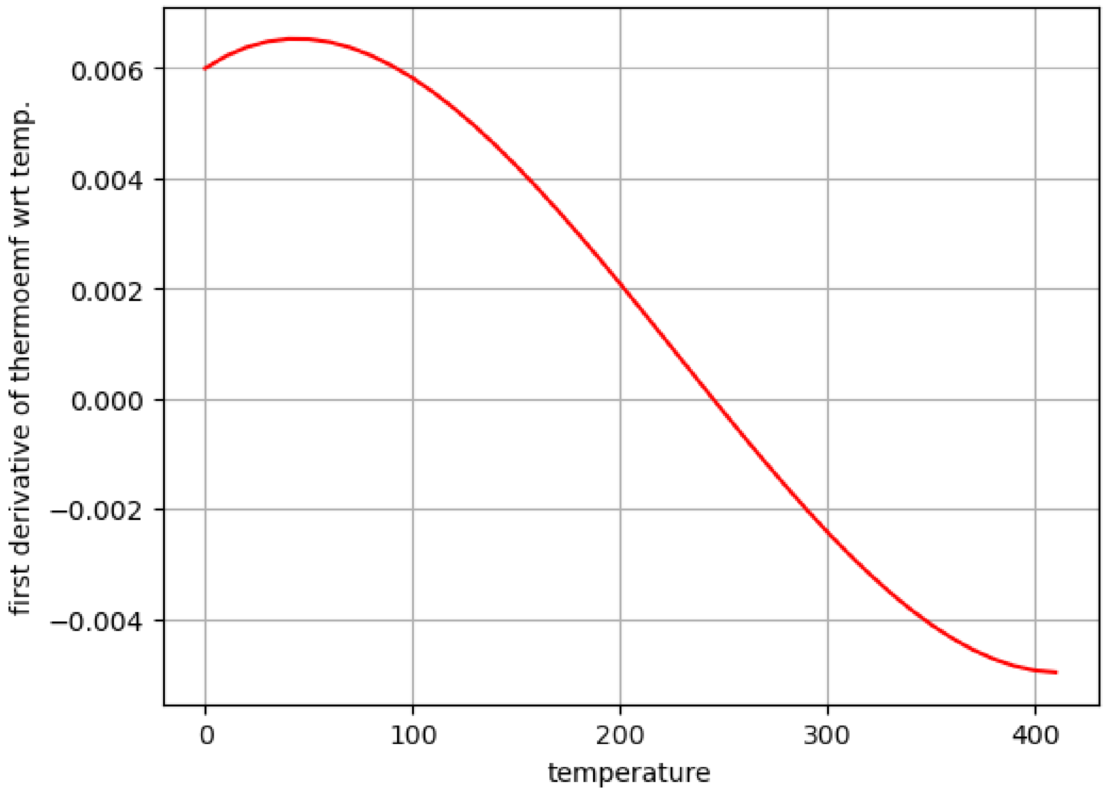


```
[90]: print('The best fit polynomial coefficients are:',poly_reg_model.coef_) #␣
       ↪Output of parameters of polynomial
```

```
The best fit polynomial coefficients are: [ 5.99472686e-03  1.26383327e-05
-1.05557025e-07  1.15720386e-10]
```

```
[138]: "Now we are generating a temperature array from 29 degree C to 420 degree C,␣
        ↪with equispaced partitions of 10 degree C and plotting it with the␣
        ↪polynomial function"
       x=np.arange(0,420,10)
       y=lambda u:5.99472686e-03*u+1.26383327e-05*u**2-1.05557025e-07*u**3+1.
        ↪15720386e-10*u**4
       Y=[]
       for i in x:
           Y.append(y(i))
```

```
[139]: ''''n=len(x)
       h=0.01
       Y,Y2=[],[]
       for i in x:
           dff2=(y(i+2*h)-2*y(i+h)+y(i))/h**2
           dff1=(y(i+h)-y(i))/h
           Y.append(dff2)
           Y2.append(dff1)
       print(Y2)
       plt.plot(x,Y,c='red',label='second derivative function')
       plt.xlabel('Temperature')
       plt.ylabel('Thermoemf')
       plt.legend()
       plt.grid()
       plt.show()
       '''
       Y1,Y2=[],[]
       from scipy.misc import derivative
       for i in x:
           dff1=derivative(y,i,0.05,1)
           dff2=derivative(y,i,0.05,2)
           Y1.append(dff1)
           Y2.append(dff2)
       plt.plot(x,Y1,c='red')
       plt.grid()
       plt.xlabel('temperature')
       plt.ylabel('first derivative of thermoemf wrt temp.')
       plt.show()
```



[140]:
```
Y1_new=np.round(Y1,decimals=3)
print('the first derivatives wrt temp\n')
print(Y1_new)
```

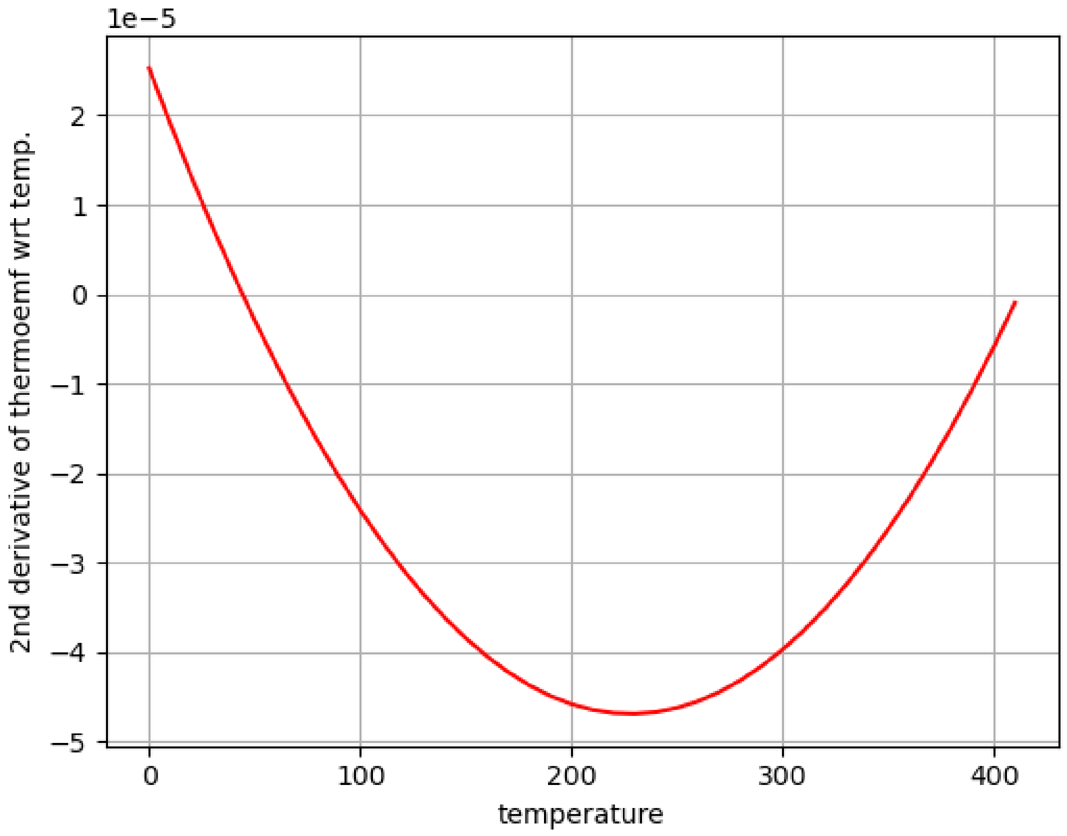
```
the first derivatives wrt temp

[ 0.006  0.006  0.006  0.006  0.007  0.007  0.006  0.006  0.006  0.006
  0.006  0.006  0.005  0.005  0.005  0.004  0.004  0.003  0.003  0.003
  0.002  0.002  0.001  0.001  0.    -0.    -0.001 -0.001 -0.002 -0.002
 -0.002 -0.003 -0.003 -0.004 -0.004 -0.004 -0.004 -0.005 -0.005 -0.005
 -0.005 -0.005]
```

[151]:
```
X=np.linspace(x[25],x[27],100,dtype=None,retstep=False)
Y3,Y4=[],[]
for i in X:
    Y3.append(np.round(derivative(y,i,0.05,1),3))
    Y4.append(derivative(y,i,0.05,2))
for i in range(len(X)):
    if Y3[i]==0:
        if Y4[i]<=0:
            print('the neutral temperature in degree Celsius is= %0.0f'%X[i])
            print('the second derivative of Neutral temperature at this point␣
↪is=%f'%Y4[i])
            break
```

```
the neutral temperature in degree Celsius is= 250
the second derivative of Neutral temperature at this point is=-0.000046
```

[152]:
```
plt.plot(x,Y2,c='red')
plt.xlabel('temperature')
plt.ylabel('2nd derivative of thermoemf wrt temp.')
plt.grid()
plt.show()
```



[ ]:
```
```