Machine Learning Coursework 1 - Linear Regression with Stochastic Gradient Descent (SGD)

Step 1 - Exploratory Data Analysis (EDA)

In [ ]:
```
#importing all required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import random
```

In [3]:
```
data = r"C:\Users\amiru\Downloads\ML_CW\window_heat.csv"
df = pd.read_csv(data)
```

In [4]:
```
# to see the first few rows of the dataframe
df.head()
```

Out[4]:

	dT[C]	Qdot[W]
0	21.178681	11401.184490
1	14.291487	7685.340740
2	4.461636	2008.096958
3	12.111569	5101.150536
4	10.510689	6033.044369

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

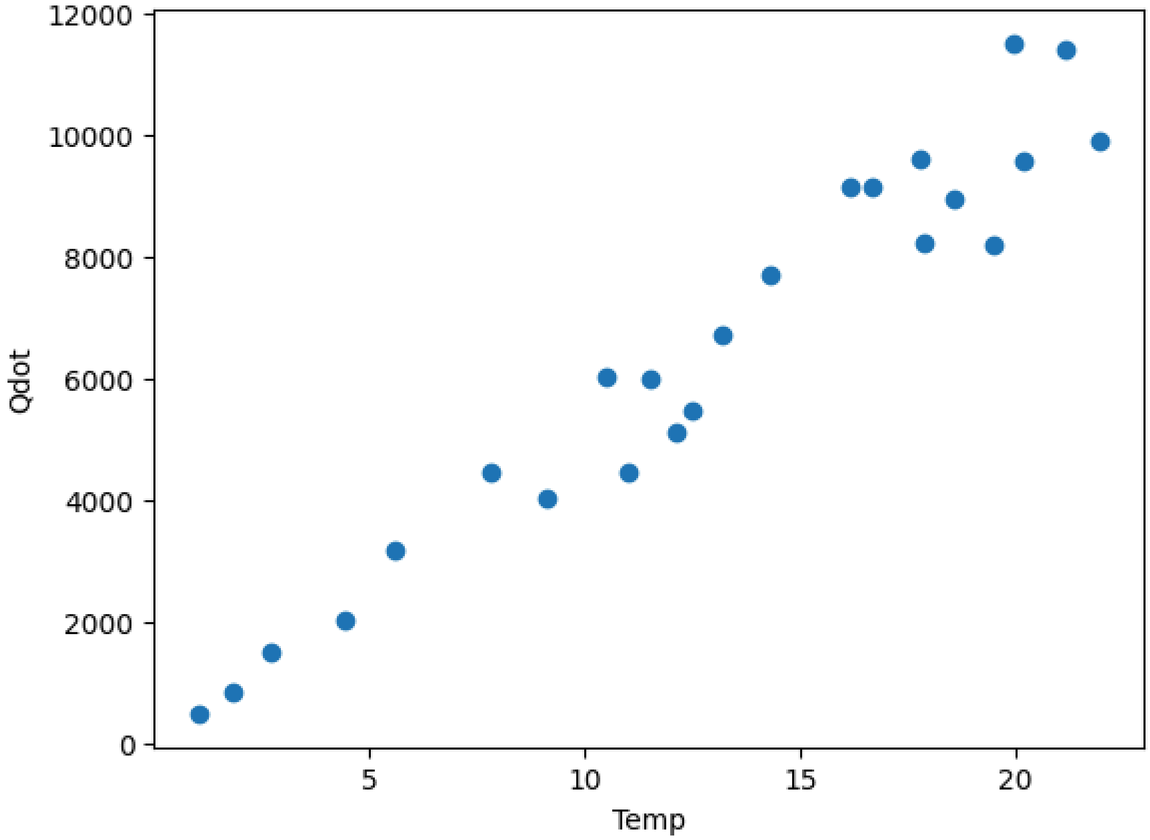
Out[5]:

	dT[C]	Qdot[W]
count	24.000000	24.000000
mean	12.817510	6398.294138
std	6.413851	3290.322723
min	1.054535	482.653133
25%	8.813390	4336.959436
50%	12.858729	6378.707791
75%	18.042443	9143.042818
max	21.930115	11507.788760

In [6]:
```
#check to see if there are missing values
print(df.isnull().sum())
```

```
dT[C]      0
Qdot[W]    0
dtype: int64
```

In [7]:
```
plt.scatter(df['dT[C]'], df['Qdot[W]'])
plt.xlabel('Temp')
plt.ylabel('Qdot')
plt.show()
```



After the initial EDA, we could see that all datas are available, and there are no missing data.
The data is also distributed normally hence there would not be any data transformation required

Step 2 - Define the loss and model function

In [ ]:
```
# normalise the value of x and y training data
x= df["dT[C]"]
y=df["Qdot[W]"]
x = (x - np.min(x)) / (np.max(x) - np.min(x))
y = (y - np.min(y)) / (np.max(y) - np.min(y))
points = len(x)
print(x)
```

```
0     0.964004
1     0.634088
2     0.163210
3     0.529664
4     0.452977
5     0.037599
6     0.722763
7     0.477402
8     0.502432
9     0.582549
10    0.839764
11    0.805104
12    0.000000
13    0.387519
14    0.548360
15    0.800288
16    0.747985
17    0.905421
18    0.218149
19    0.324130
20    0.883355
21    0.916948
22    1.000000
23    0.079813
Name: dT[C], dtype: float64
```

In [ ]:

```
# loss using mean squared error
def lossmse(w, b, x, y, points):
    """
    w: weight
    b: bias
    x: feature
    y: label
    points: length of dataset

    """
    total_loss = 0
    for i in range(points):
        total_loss += (y[i]-w*x[i]+b)**2

    total_loss = total_loss / float(points)

    return total_loss
```

In [25]:

```
def lossmae(w, b, x, y, points):
    """
    w: weight
    b: bias
    x: feature
    y: label
    points: length of dataset

    """
    total_loss = 0
    for i in range(points):
        total_loss += abs(y[i] - (w * x[i] + b))  # Using absolute error instead

    total_loss = total_loss / float(points)  # Averaging over the number of poin
```

```
    return total_loss
```

In [ ]:
```
def sgd_linear_regression(x, y, w, b, alpha, points, epoch):
    """
    x: feature
    y: label
    w: initial / current weight
    b: initial / current bias
    alpha: learning rate
    points: length of dataset
    epoch: number of training iterations
    """

    cost_list = []
    epoch_list = []

    for i in range(epoch):
        # Select a random data point for stochastic gradient descent
        random_index = random.randint(0, points - 1)
        x_i = x[random_index]
        y_i = y[random_index]

        # Predict the output
        y_predicted = w * x_i + b

        # Compute gradients for this data point
        dl_dw = -2.0 * x_i * (y_i - y_predicted)
        dl_db = -2.0 * (y_i - y_predicted)

        # Update weights and bias
        w = w - alpha * dl_dw
        b = b - alpha * dl_db

        # Compute the loss
        loss = lossmse(w, b, x, y, points)
        #loss = (y_i - y_predicted) ** 2

        # Log loss every 10 epochs
        if i % 100 == 0:
            cost_list.append(loss)
            epoch_list.append(i)
            print(f'Epoch:{i}, Loss:{loss}')

    return w, b, loss, cost_list, epoch_list
```

In [26]:
```
def sgd_linear_regression_mae(x, y, w, b, alpha, points, epoch):
    """
    x: feature
    y: label
    w: initial / current weight
    b: initial / current bias
    alpha: learning rate
    points: length of dataset
    epoch: number of training iterations
    """

    cost_list = []
    epoch_list = []
```

```
    for i in range(epoch):
        # Select a random data point for stochastic gradient descent
        random_index = random.randint(0, points - 1)
        x_i = x[random_index]
        y_i = y[random_index]

        # Predict the output
        y_predicted = w * x_i + b

        # Compute gradients for this data point
        dl_dw = -2.0 * x_i * (y_i - y_predicted)
        dl_db = -2.0 * (y_i - y_predicted)

        # Update weights and bias
        w = w - alpha * dl_dw
        b = b - alpha * dl_db

        # Compute the loss
        loss = lossmae(w, b, x, y, points)
        #loss = (y_i - y_predicted) ** 2

        # Log loss every 10 epochs
        if i % 100 == 0:
            cost_list.append(loss)
            epoch_list.append(i)
            print(f'Epoch:{i}, Loss:{loss}')

    return w, b, loss, cost_list, epoch_list
```

Step 3 - Training the model to obtain w and b

In [31]:
```
w,b,loss,cost_list,epoch_list = sgd_linear_regression(x,y,1,1,0.01,points=points
```

```
Epoch:0, Loss:0.9142032463773924
Epoch:100, Loss:0.18232940537975073
Epoch:200, Loss:0.12076484755868089
Epoch:300, Loss:0.08936255840050879
Epoch:400, Loss:0.06907617984874094
Epoch:500, Loss:0.05765941562426511
Epoch:600, Loss:0.04392810376581508
Epoch:700, Loss:0.03688233591229283
Epoch:800, Loss:0.03006816358598716
Epoch:900, Loss:0.025143891300813435
Epoch:1000, Loss:0.02231456213823731
Epoch:1100, Loss:0.017380848900090718
Epoch:1200, Loss:0.014354435895617303
Epoch:1300, Loss:0.013237195727298147
Epoch:1400, Loss:0.01138303380953621
Epoch:1500, Loss:0.00996623202116813
Epoch:1600, Loss:0.008959473435638115
Epoch:1700, Loss:0.008491547852305695
Epoch:1800, Loss:0.007309480158141618
Epoch:1900, Loss:0.007311149567117034
Epoch:2000, Loss:0.006742393314855824
Epoch:2100, Loss:0.00626663319279254
Epoch:2200, Loss:0.006659512245282829
Epoch:2300, Loss:0.00625150054211773
Epoch:2400, Loss:0.005585758017009338
Epoch:2500, Loss:0.005533652131361034
Epoch:2600, Loss:0.006036959036923158
Epoch:2700, Loss:0.0059604452233674166
Epoch:2800, Loss:0.005529859033649249
Epoch:2900, Loss:0.005649772188877462
Epoch:3000, Loss:0.00555895965703479
Epoch:3100, Loss:0.005293157909426614
Epoch:3200, Loss:0.005530230613161032
Epoch:3300, Loss:0.005395793489135176
Epoch:3400, Loss:0.005328426221705287
Epoch:3500, Loss:0.005359488417806025
Epoch:3600, Loss:0.005070553243721035
Epoch:3700, Loss:0.005157860270024979
Epoch:3800, Loss:0.0049829709070267295
Epoch:3900, Loss:0.005173230992281767
Epoch:4000, Loss:0.005139539055217015
Epoch:4100, Loss:0.005247786152913227
Epoch:4200, Loss:0.00525825085754338
Epoch:4300, Loss:0.005154200526668959
Epoch:4400, Loss:0.005259407372918009
Epoch:4500, Loss:0.0049897291236071645
Epoch:4600, Loss:0.004967677877295513
Epoch:4700, Loss:0.004981417190320479
Epoch:4800, Loss:0.0049213403315044825
Epoch:4900, Loss:0.004927738471756626
Epoch:5000, Loss:0.00495957728067717
Epoch:5100, Loss:0.004956533036150579
Epoch:5200, Loss:0.005085600045288941
Epoch:5300, Loss:0.00494009589686509
Epoch:5400, Loss:0.004889880342773915
Epoch:5500, Loss:0.004895010783922054
Epoch:5600, Loss:0.004917278053598575
Epoch:5700, Loss:0.004915770104412141
Epoch:5800, Loss:0.004997033705944031
Epoch:5900, Loss:0.004999250038234014
```

```
Epoch:6000, Loss:0.004893595286044617
Epoch:6100, Loss:0.0049194640176356465
Epoch:6200, Loss:0.004936470665478371
Epoch:6300, Loss:0.004901617041362429
Epoch:6400, Loss:0.004909434734291252
Epoch:6500, Loss:0.005001998999184453
Epoch:6600, Loss:0.004913439113921597
Epoch:6700, Loss:0.0049303988567498775
Epoch:6800, Loss:0.004940730178401899
Epoch:6900, Loss:0.004916430370346726
Epoch:7000, Loss:0.004921593968160458
Epoch:7100, Loss:0.005049662634013364
Epoch:7200, Loss:0.005081838902932674
Epoch:7300, Loss:0.005086742451803246
Epoch:7400, Loss:0.004966382052178603
Epoch:7500, Loss:0.005056567244465523
Epoch:7600, Loss:0.0051006324960123145
Epoch:7700, Loss:0.005233870225171908
Epoch:7800, Loss:0.00536164106715413
Epoch:7900, Loss:0.005115011840200192
Epoch:8000, Loss:0.005258808950830542
Epoch:8100, Loss:0.005285449440479155
Epoch:8200, Loss:0.0052808764673541704
Epoch:8300, Loss:0.0052547237215916755
Epoch:8400, Loss:0.005091800907235281
Epoch:8500, Loss:0.005115772398640843
Epoch:8600, Loss:0.005185288582246807
Epoch:8700, Loss:0.005107538319339793
Epoch:8800, Loss:0.004937823999140849
Epoch:8900, Loss:0.004935874120668447
Epoch:9000, Loss:0.004932281941743902
Epoch:9100, Loss:0.005033375621661487
Epoch:9200, Loss:0.004935832523483749
Epoch:9300, Loss:0.004892766732924446
Epoch:9400, Loss:0.004918732867211546
Epoch:9500, Loss:0.004906655431285604
Epoch:9600, Loss:0.005129656843852441
Epoch:9700, Loss:0.005190931511763228
Epoch:9800, Loss:0.005054592888791702
Epoch:9900, Loss:0.004912080929067559
```

In [ ]:

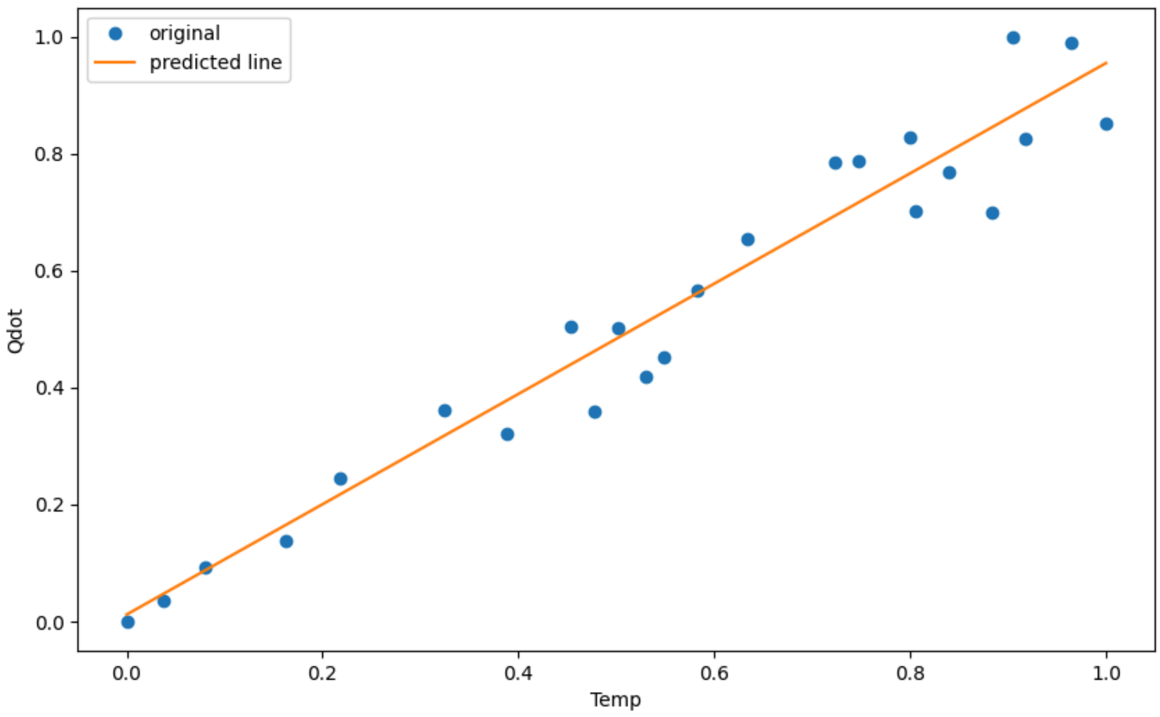
```
w
```

Out[ ]: 0.7464134505815038

In [23]:

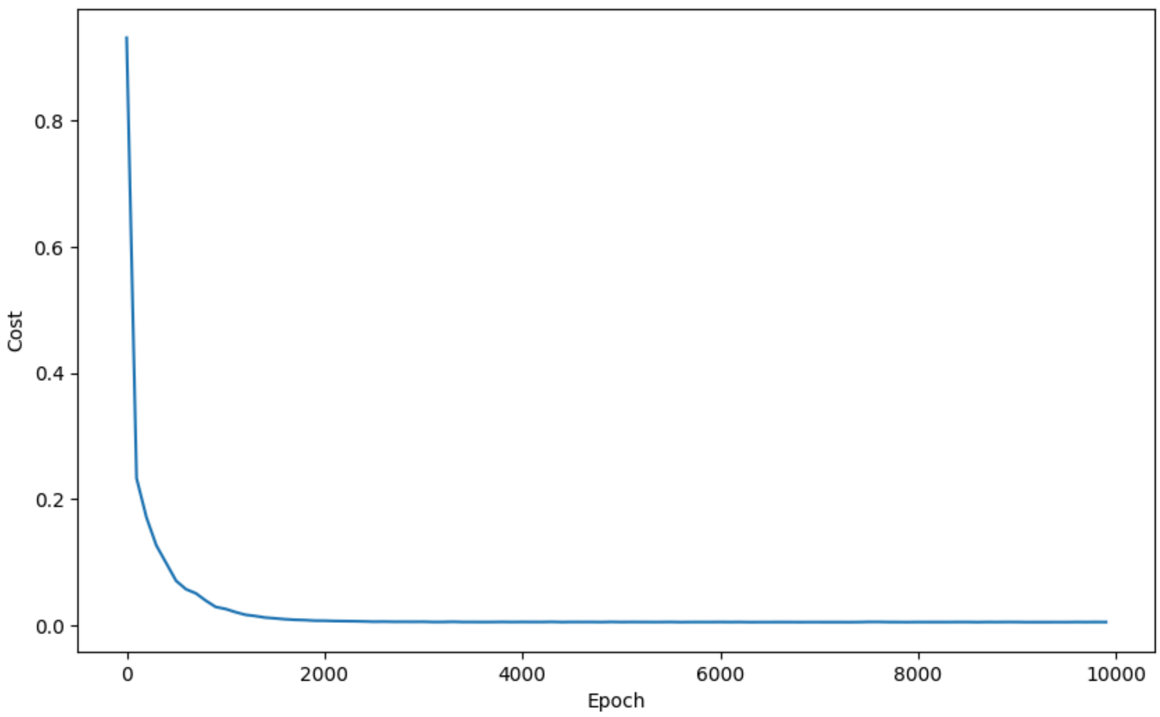
```
plt.figure(figsize=(10, 6))
plt.plot(x, y, marker = 'o', ls='', label = 'original')
plt.plot([0,1],[b,w*1+b],label = 'predicted line')
plt.xlabel('Temp')
plt.ylabel('Qdot')
plt.legend()
```

Out[23]: <matplotlib.legend.Legend at 0x28d361bda00>



In [24]:
```
plt.figure(figsize=(10, 6))
plt.plot(epoch_list,cost_list)
plt.xlabel('Epoch')
plt.ylabel('Cost')
```

Out[24]: Text(0, 0.5, 'Cost')



In [29]:
```
w2,b2,loss2,cost_list2,epoch_list2 = sgd_linear_regression_mae(x,y,1,1,0.01,poin
```

```
Epoch:0, Loss:0.9980752811075777
Epoch:100, Loss:0.11841569375760358
Epoch:200, Loss:0.09234728418004111
Epoch:300, Loss:0.08939309493351244
Epoch:400, Loss:0.07968049416655351
Epoch:500, Loss:0.08150617577876501
Epoch:600, Loss:0.0717290239500146
Epoch:700, Loss:0.06913413128699444
Epoch:800, Loss:0.06898496656441443
Epoch:900, Loss:0.06520318220536296
Epoch:1000, Loss:0.06448466112851252
Epoch:1100, Loss:0.06348754997267204
Epoch:1200, Loss:0.06331668649468374
Epoch:1300, Loss:0.06229326488551242
Epoch:1400, Loss:0.061456560124229044
Epoch:1500, Loss:0.06044018165336615
Epoch:1600, Loss:0.06149362876132894
Epoch:1700, Loss:0.0601668258531757
Epoch:1800, Loss:0.06063829590202597
Epoch:1900, Loss:0.0594760359358207
Epoch:2000, Loss:0.05933829039156396
Epoch:2100, Loss:0.059755617895090186
Epoch:2200, Loss:0.05961860412082245
Epoch:2300, Loss:0.05940345892457985
Epoch:2400, Loss:0.05904104046653614
Epoch:2500, Loss:0.058986135481611905
Epoch:2600, Loss:0.060108134985205404
Epoch:2700, Loss:0.05912073175338068
Epoch:2800, Loss:0.05895337150755476
Epoch:2900, Loss:0.059218441830977964
Epoch:3000, Loss:0.05894536713924146
Epoch:3100, Loss:0.05895033558932957
Epoch:3200, Loss:0.058940349317809754
Epoch:3300, Loss:0.05913808966972575
Epoch:3400, Loss:0.05903278049005165
Epoch:3500, Loss:0.058846593821340214
Epoch:3600, Loss:0.05886641469895901
Epoch:3700, Loss:0.058902888446958235
Epoch:3800, Loss:0.058952724956680407
Epoch:3900, Loss:0.05891692515564286
Epoch:4000, Loss:0.058929458348820414
Epoch:4100, Loss:0.0589308667752886
Epoch:4200, Loss:0.05891779317340809
Epoch:4300, Loss:0.05894611326115656
Epoch:4400, Loss:0.05894471326167796
Epoch:4500, Loss:0.05895981618511089
Epoch:4600, Loss:0.058975107705019986
Epoch:4700, Loss:0.05897726821362259
Epoch:4800, Loss:0.05903929964162132
Epoch:4900, Loss:0.05963900013928619
Epoch:5000, Loss:0.05895564061359818
Epoch:5100, Loss:0.05895496339238735
Epoch:5200, Loss:0.059187514697060246
Epoch:5300, Loss:0.05890232632455733
Epoch:5400, Loss:0.05891851308478164
Epoch:5500, Loss:0.05957494808418654
Epoch:5600, Loss:0.05893470611556181
Epoch:5700, Loss:0.05913595619915066
Epoch:5800, Loss:0.059137545593338926
Epoch:5900, Loss:0.05893857818148404
```

```
Epoch:6000, Loss:0.058963444427804663
Epoch:6100, Loss:0.05901163326819734
Epoch:6200, Loss:0.059011976831900535
Epoch:6300, Loss:0.05893560071361694
Epoch:6400, Loss:0.05889257769386491
Epoch:6500, Loss:0.05887634801218972
Epoch:6600, Loss:0.059861732341285716
Epoch:6700, Loss:0.058842381516514136
Epoch:6800, Loss:0.05887667167179902
Epoch:6900, Loss:0.05893436685262942
Epoch:7000, Loss:0.058916459736159686
Epoch:7100, Loss:0.05889606541035681
Epoch:7200, Loss:0.05880484140088335
Epoch:7300, Loss:0.059546529660444825
Epoch:7400, Loss:0.059340175283189904
Epoch:7500, Loss:0.058907488030054106
Epoch:7600, Loss:0.05895319563593871
Epoch:7700, Loss:0.0589049559588793
Epoch:7800, Loss:0.05891334223745732
Epoch:7900, Loss:0.058840331491431086
Epoch:8000, Loss:0.05963676222449372
Epoch:8100, Loss:0.058927584868905876
Epoch:8200, Loss:0.05885917586261493
Epoch:8300, Loss:0.05883116286922055
Epoch:8400, Loss:0.05881922673344824
Epoch:8500, Loss:0.058858741398050196
Epoch:8600, Loss:0.05884346244249181
Epoch:8700, Loss:0.05888938292821256
Epoch:8800, Loss:0.05880504572033576
Epoch:8900, Loss:0.06013699816498145
Epoch:9000, Loss:0.058875720773142626
Epoch:9100, Loss:0.05896979087494192
Epoch:9200, Loss:0.058987796240512785
Epoch:9300, Loss:0.05897194670638629
Epoch:9400, Loss:0.05909259429514955
Epoch:9500, Loss:0.05901362539589617
Epoch:9600, Loss:0.05891841576857102
Epoch:9700, Loss:0.05902874961297796
Epoch:9800, Loss:0.05913796319335568
Epoch:9900, Loss:0.05930331675732392
```

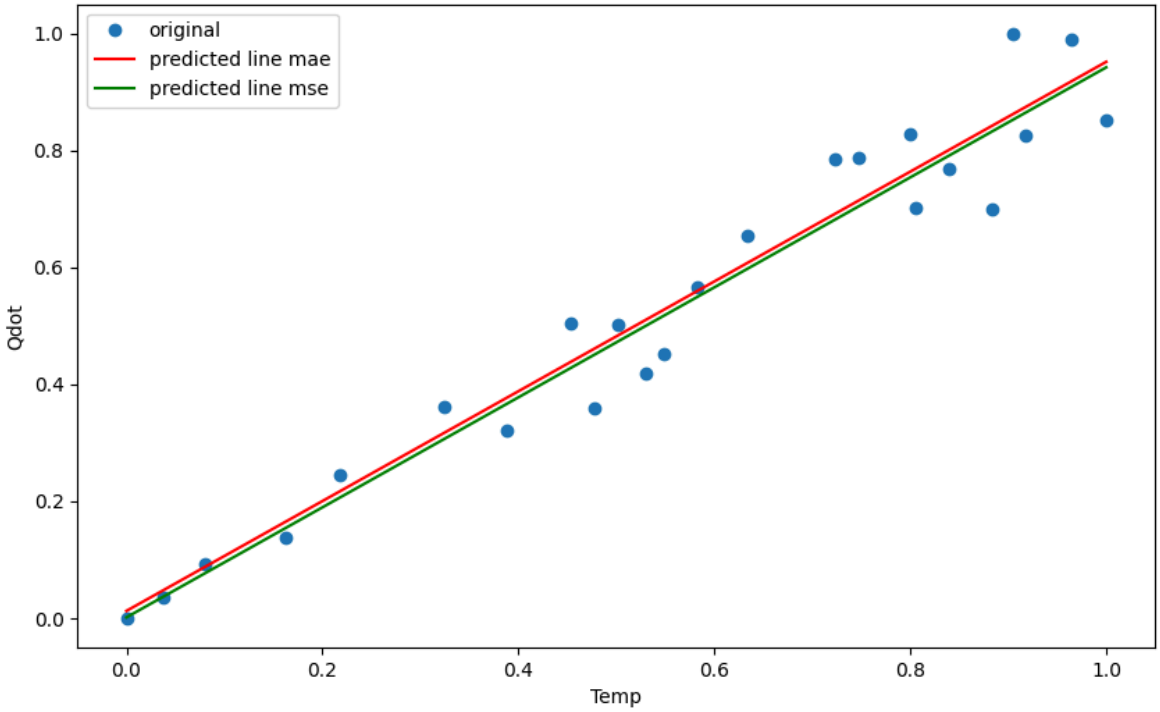
In [32]:

```
plt.figure(figsize=(10, 6))
plt.plot(x, y, marker = 'o', ls='', label = 'original')
plt.plot([0,1],[b2,w2*1+b2],label = 'predicted line mae', color = 'red')
plt.plot([0,1],[b,w*1+b],label = 'predicted line mse', color = 'green')

plt.xlabel('Temp')
plt.ylabel('Qdot')
plt.legend()
```

Out[32]:

```
<matplotlib.legend.Legend at 0x28d36b2c5b0>
```


original
predicted line mae
predicted line mse
Qdot
Temp
0.0
0.2
0.4
0.6
0.8
1.0

In [ ]: