In [23]:

```
#Necessary import files
import pandas as pd
import seaborn as sb
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LogisticRegression

import pdb;

from IPython.core.debugger import set_trace
import random
import matplotlib.pyplot as plt
```

In [24]:

```
#Read csv
electricity_data = pd.read_csv("energydata_complete.csv")
electricity_data_appliance_rand10 =  electricity_data
```

In [25]:

```
#input variables
x_appliances = electricity_data_appliance_rand10.drop(lab
els = ['date','lights','Appliances','T6','RH_6','T7','RH_
7','T8','RH_8',

'T9','RH_9','rv1','rv2','T5','RH_5','T4','RH_4','Tdewpoin
t',

'RH_out'],axis = 1)
```

In [26]:

```
# output varaible
y_appliances = electricity_data_appliance_rand10[['Appliances']]
```

In [27]:

```
random.seed(1)
```

In [28]:

```
#Bringing to a common scale
scaler = MinMaxScaler()
x_appliances_scaled = x_appliances
y_appliances_scaled = y_appliances

x_appliances_scaled = scaler.fit_transform(x_appliances)
y_appliances_scaled = scaler.fit_transform(y_appliances)
```

In [29]:

```
x_appliances_scaled = np.array(x_appliances_scaled)
```

In [32]:

```
#70% - training, 30% - validation
X_Train, X_Test, Y_Train, Y_Test = train_test_split(x_appliances_scaled, y_appliances_scaled, test_size=0.3,random_state=1)
```

In [33]:

```python
#Default function - Used only for cross verification with my gradient descend results
from sklearn import datasets, linear_model
reg = linear_model.LinearRegression()
reg.fit(X_Train, Y_Train)

coef = reg.coef_
intercept = reg.intercept_
print('Coefficients: \n', coef.round(3))
print('Intercept: \n', intercept.round(3))
```

```
Coefficients: 
 [[ 0.065  0.704 -0.315 -0.673  0.178 -0.027
-0.021  0.001  0.035  0.008]]
Intercept: 
 [0.197]
```

In [35]:

```
main_beta = np.array(np.repeat(0.1, 10)).reshape(1,10)
beta_0 = 0.1
totalRows = len(X_Train) #Total number of rows in the dat
afram
learningRate = 0.9
CostValues_9_validation = []

for i in range(0,30000) :
    main_beta = main_beta - (learningRate/totalRows) * np
.sum(X_Test * (X_Test @ main_beta.T - Y_Test + beta_0), a
xis=0)
    beta_0 = beta_0 - (learningRate/totalRows) * np.sum((
X_Test @ main_beta.T - Y_Test + beta_0), axis=0)

cost = (0.5 * (1/totalRows))* (np.sum(X_Test @ main_beta.
T + beta_0 - Y_Test) ** 2)
CostValues_9_validation.append(cost)
print(beta_0)
print(main_beta)

print(CostValues_9_validation)
```

```
[0.20566666]
[[ 8.31523735e-02  7.49619135e-01 -3.3746688
5e-01 -7.12517576e-01
   1.59534639e-01 -2.70251870e-02 -5.1418745
0e-03 -9.28546788e-05
   3.25213015e-02  6.94199924e-03]]
[1.1350815347796348e-10]
```

In [34]:

```
main_beta = np.array(np.repeat(0.1, 10)).reshape(1,10)
beta_0 = 0.1
totalRows = len(X_Train) #Total number of rows in the dat
afram
learningRate = 0.9
CostValues_9_training = []


for i in range(0,30000) :
    main_beta = main_beta - (learningRate/totalRows) * np
.sum(X_Train * (X_Train @ main_beta.T - Y_Train + beta_0
), axis=0)
    beta_0 = beta_0 - (learningRate/totalRows) * np.sum((
X_Train @ main_beta.T - Y_Train + beta_0), axis=0)

cost = (0.5 * (1/totalRows))* (np.sum(X_Train @ main_beta
.T + beta_0 - Y_Train) ** 2)
print("Cost = ",cost)
CostValues_9_training.append(cost)
print(beta_0)
print(main_beta)
print(CostValues_9_training)
```

```
Cost =  3.1269641753475197e-16
[0.19731856]
[[ 0.06477214  0.70433847 -0.31515031 -0.673
4302   0.17833086 -0.026704
  -0.02116264  0.00137992  0.03469382  0.008
2824 ]]
[3.1269641753475197e-16]
```

In [36]:

```
beta_0_1 =beta_0
beta_coefficients_1 = main_beta

mse1 = (1/len(X_Train)) * np.sum( (X_Test * beta_coefficients_1 + beta_0_1 ) -  Y_Test) ** 2
print(mse1)
```

```
3087.6658567015143
```

In [14]:

```
electricity_data_appliance_best_10 =  electricity_data
```

In [37]:

```
x_appliances = electricity_data_appliance_best_10[['T1','RH_1','T3','RH_3','T4','RH_4','T8','RH_8','T9', 'RH_9']]
y_appliances = electricity_data_appliance_best_10[['Appliances']]
```

In [38]:

```
scaler = MinMaxScaler()
x_appliances_scaled = x_appliances
y_appliances_scaled = y_appliances

x_appliances_scaled = scaler.fit_transform(x_appliances)
y_appliances_scaled = scaler.fit_transform(y_appliances)
```

In [39]:

```
x_appliances_scaled = np.array(x_appliances_scaled)
```

In [40]:

```
X_Train, X_Test, Y_Train, Y_Test = train_test_split(x_appliances_scaled, y_appliances_scaled, test_size=0.3, random_state=1)
```

In [41]:

```
from sklearn import datasets, linear_model
reg = linear_model.LinearRegression()
reg.fit(X_Train, Y_Train)

coef = reg.coef_
intercept = reg.intercept_
print('Coefficients: \n', coef.round(3))
print('Intercept: \n', intercept.round(3))
```

```
Coefficients: 
 [[-0.055  0.241  0.286  0.068  0.033  0.011 
0.051 -0.206 -0.251 -0.05 ]]
Intercept: 
 [0.059]
```

In [43]:

```
#Validation
main_beta = np.array(np.repeat(0.1, 10)).reshape(1,10)
beta_0 = 0.1
totalRows = len(X_Train) #Total number of rows in the datafram
learningRate = 0.9
CostValues_9_validation = []

for i in range(0,30000) :
    main_beta = main_beta - (learningRate/totalRows) * np.sum(X_Test * (X_Test @ main_beta.T - Y_Test + beta_0), axis=0)
    beta_0 = beta_0 - (learningRate/totalRows) * np.sum((X_Test @ main_beta.T - Y_Test + beta_0), axis=0)

cost = (0.5 * (1/totalRows))* (np.sum(X_Test @ main_beta.T + beta_0 - Y_Test) ** 2)
CostValues_9_validation.append(cost)
print(beta_0)
print(main_beta)

print(CostValues_9_validation)
```

```
[0.05048942]
[[-0.08932893  0.22851333  0.26500806  0.08186398  0.04250923  0.04271795
   0.09838177 -0.23556285 -0.25330342 -0.04184032]]
[1.096475981992043e-20]
```

In [44]:

```
#Training
main_beta = np.array(np.repeat(0.1, 10)).reshape(1,10)
beta_0 = 0.1
totalRows = len(X_Train) #Total number of rows in the dat
afram
learningRate = 0.9
CostValues_9_training = []


for i in range(0,30000) :
    main_beta = main_beta - (learningRate/totalRows) * np
.sum(X_Train * (X_Train @ main_beta.T - Y_Train + beta_0
), axis=0)
    beta_0 = beta_0 - (learningRate/totalRows) * np.sum((
X_Train @ main_beta.T - Y_Train + beta_0), axis=0)

cost = (0.5 * (1/totalRows))* (np.sum(X_Train @ main_beta
.T + beta_0 - Y_Train) ** 2)
print("Cost = ",cost)
CostValues_9_training.append(cost)
print(beta_0)
print(main_beta)
print(CostValues_9_training)
```

```
Cost =  4.015258607090805 5e-32
[0.05932439]
[[-0.05507639  0.24055236  0.28591529  0.067
755    0.03284957  0.01098882
   0.05142826 -0.20612976 -0.25149364 -0.049
65594]]
[4.0152586070908055e-32]
```

In [45]:

```
beta_0_1 =beta_0
beta_coefficients_1 = main_beta

mse1 = (1/len(X_Train)) * np.sum( (X_Test * beta_coefficients_1 + beta_0_1 ) -  Y_Test) ** 2
print(mse1)
```

```
118.34527426668004
```

In [ ]: