MACHINE LEARNING AND DATA MINING

HOMEWORK 1

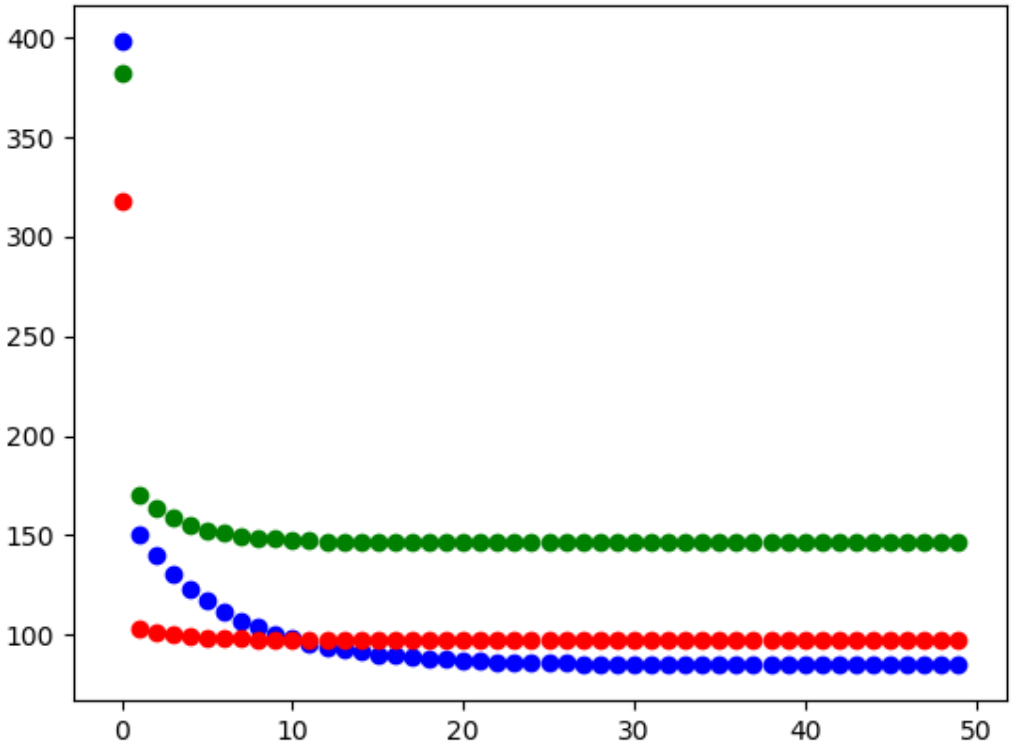
RAGHAVENDRAN KASINATHAN
z5284284

1)For House price prediction with House Age :

theta0 = 42.5407

theta1 = -10.3193

2)



3) RMSE for training set with House Age :

12.0979

4) RMSE for test set with House Age:

16.6634

5)RMSE for test set with Distance to station feature: 12.184

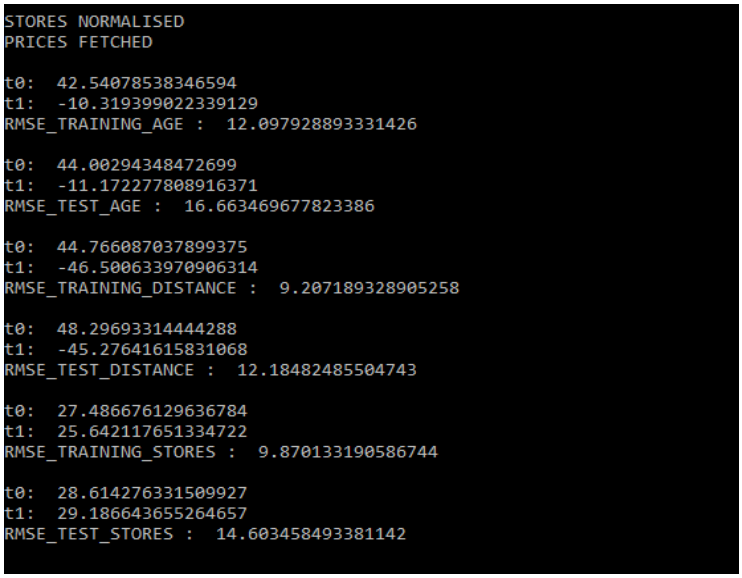
6)RMSE for test set with No. of stores: 14.6034

7)The 3 models have a pretty diverse performance that House price can be predicted more closely with Distance to Station feature than the others. And hence can be ranked based on their RMSE values during Testing.

MODEL A = House Price prediction with House Age feature
MODEL B = House Price prediction with Distance to Station feature
MODEL C = House Price prediction with number of stores feature

RANKING BASED ON PERFORMANCE WITH RMSE VALUES:

B > C > A



CODE :-

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

def create_or_test_model(input,price,m,itr):
        t0 = -1
        t1 = -0.5
        x0 = 1
        alpha = 0.01
        J = []
        for loop in range(itr):
                sqsum = 0
                for j in range(m):
                        x1 = input[j]
                        h = x0*t0 + x1*t1
                        t0 = t0 + alpha*(price[j] - h)*x0
                        t1 = t1 + alpha*(price[j] - h)*x1
                        sqsum = sqsum + (price[j] - h)*(price[j] - h)
                J.append((1/m)*(sqsum))
        return (t0,t1,J)

def printFeature(feature):
   for i in range(len(feature)):
      print(feature[i])

def normalise(feature):
   min_val = min(feature)
   max_val = max(feature)
   for i in range(len(feature)):
      feature[i] = (feature[i] - min_val) / (max_val - min_val)
   return feature

data = pd.read_csv("house_prices.csv")

age = data['house age'].values.tolist()
```

```
age = normalise(age)
print("AGES NORMALISED")
#printFeature(age)

print()

distance = data['distance to the nearest MRT station'].values.tolist()
distance = normalise(distance)
print("Distances NORMALISED")
#printFeature(distance)

print()

stores = data['number of convenience stores'].values.tolist()
stores = normalise(stores)
print("STORES NORMALISED")
#printFeature(stores)

price = data['house price of unit area'].values.tolist()
print("PRICES FETCHED")

print()



#CREATING TEST AND TRAINING SETS FOR AGE DATASET
training_set_age = age[:300]
test_set_age = age[300:400]
t0,t1,J = create_or_test_model(training_set_age,price[:300],300,50)
RMSE_TRAINING_AGE = J[49]**0.5
print("t0: ",t0,"\nt1: ",t1,"\nRMSE_TRAINING_AGE : ",RMSE_TRAINING_AGE)
print()

plt.scatter(range(50),J,c='green')

t0,t1,J = create_or_test_model(test_set_age,price[300:400],100,50)
RMSE_TEST_AGE = J[49]**0.5
print("t0: ",t0,"\nt1: ",t1,"\nRMSE_TEST_AGE : ",RMSE_TEST_AGE)
```

```
print()

#CREATING TEST AND TRAINING SETS FOR DISTANCE DATASET
training_set_distance = distance[:300]
test_set_distance = distance[300:400]
t0,t1,J = create_or_test_model(training_set_distance,price[:300],300,50)
RMSE_TRAINING_DISTANCE = J[49]**0.5
print("t0: ",t0,"\nt1: ",t1,"\nRMSE_TRAINING_DISTANCE :
",RMSE_TRAINING_DISTANCE)
print()

plt.scatter(range(50),J,c='blue')

t0,t1,J = create_or_test_model(test_set_distance,price[300:400],100,50)
RMSE_TEST_DISTANCE = J[49]**0.5
print("t0: ",t0,"\nt1: ",t1,"\nRMSE_TEST_DISTANCE : ",RMSE_TEST_DISTANCE)
print()

#CREATING TEST AND TRAINING SETS FOR STORES DATASET
training_set_stores = stores[:300]
test_set_stores = stores[300:400]
t0,t1,J = create_or_test_model(training_set_stores,price[:300],300,50)
RMSE_TRAINING_STORES = J[49]**0.5
print("t0: ",t0,"\nt1: ",t1,"\nRMSE_TRAINING_STORES :
",RMSE_TRAINING_STORES)
print()

plt.scatter(range(50),J,c='red')

t0,t1,J = create_or_test_model(test_set_stores,price[300:400],100,50)
RMSE_TEST_STORES = J[49]**0.5
print("t0: ",t0,"\nt1: ",t1,"\nRMSE_TEST_STORES : ",RMSE_TEST_STORES)
print()

plt.show()
```