Allen kombasseril
z5232188

COMP9417 HW1

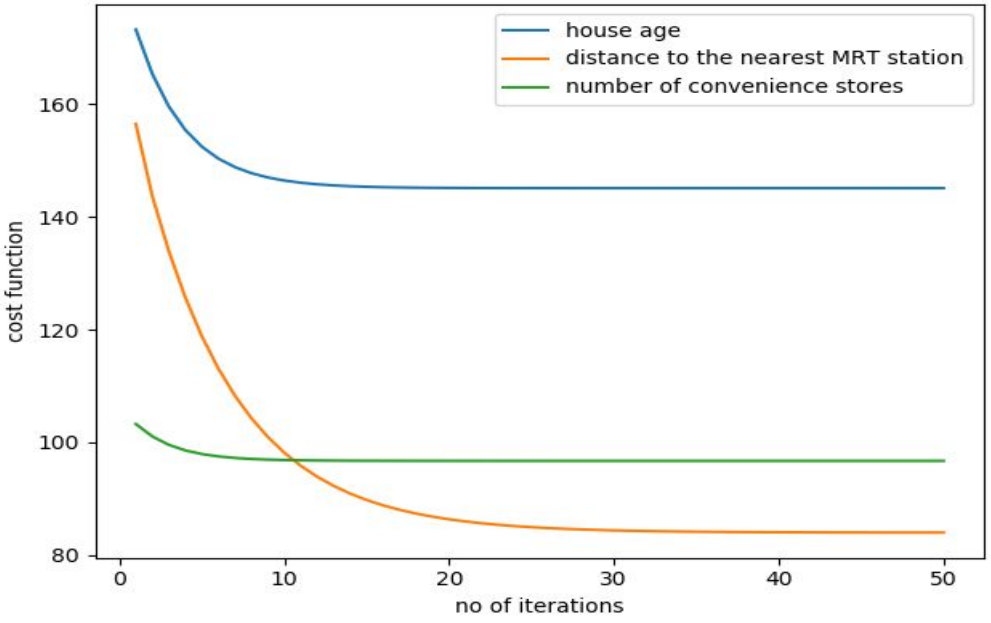
1. The θ parameters (θ_0 , θ_1) from step 3 when you are using house age feature. (2 marks)

Here, $\theta_0 =$ 42.54078538,
$\theta_1 =$ -10.31939902

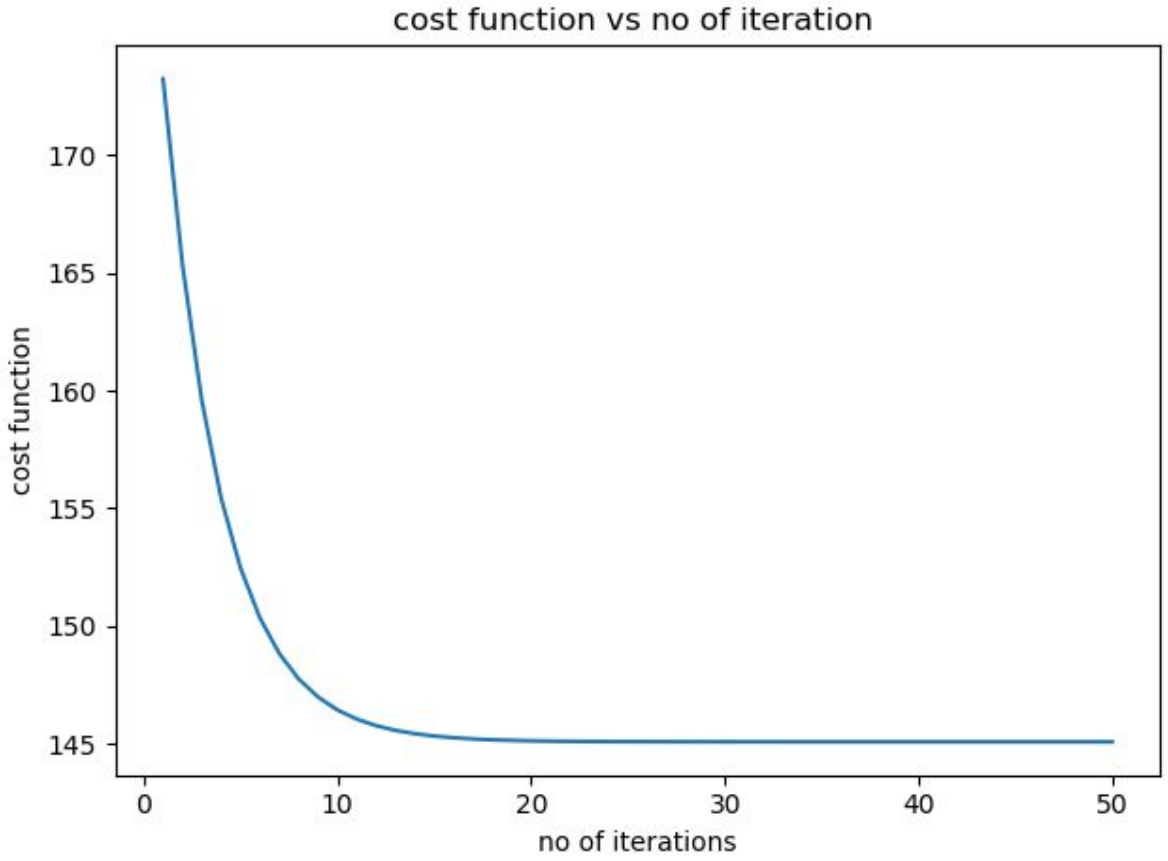
2. A plot, which visualises the change in cost function $J(\theta)$ at each iteration.

The below figure shows the cost function vs number of iteration graph for the three different models using the following features.

- House age
- Distance of the nearest MRT station
- Number of convenience store



Individual graph for the house age feature alone is as shown below:



3. RMSE for your training set when you use house age feature.

RMES =12.04551030591235

4. RMSE for test set, when you use house age feature.

RMES = 16.587314503400513

5. RMSE for test set, when you use distance to the station feature.

RMES = 12.652088009723935

6. RMSE for test set, when you use number of stores feature.

RMES= 14.731993508206783

7. Compare the performance of your three models and rank them accordingly.

To compare different models, you just need to compare an appropriate evaluation metric like RMSE on the test data.

RMSE on test set for distance to the station feature<RMSE on test set when you use number of stores feature<RMSE on test set for house age feature.

$$RMSE_{distance\ to\ the\ station} < RMSE_{number\ of\ stores} < RMSE_{house\ age}$$

Lower the RMSE the better the model performs on our test set..

$$model_{distance\ to\ the\ station} > model_{number\ of\ stores} > model_{house\ age}$$

So our best model is the one we trained with distance to the station and our worst model is the one we trained with house age.

My code is as shown below ::

```
import numpy as np
import matplotlib.pyplot as plt
import math

def preprocessing(csvfile): #function for preprocessing

    df=pd.read_csv(csvfile)         #reading csv file
    df=df.set_index("No")                #setting column as index
    df_y=df["house price of unit area"]  #separating price
    df = (df - df.min()) / (df.max() - df.min())  #normalizing
    df['house price of unit area']=df_y  #updating price
    return df

def split_rows(dataframe):         #function to split dataframe
    return dataframe.iloc[0:300],dataframe.iloc[300:400]

def train_test_split(df):              #function to splitting dataset
    X_train, X_test=split_rows(df)        # seperating to test and train
    y_train= X_train.pop('house price of unit area')
    y_test= X_test.pop('house price of unit area')
    y_train= y_train.to_frame()
    y_test= y_test.to_frame()

    return X_train,X_test,y_train,y_test

def cal_cost(theta,X,y): # function to calculate cost
    y=y.to_numpy()
    m=len(y)
    predictions=X.dot(theta.T)
    sub=predictions-y
    sub=np.square(sub)
    subsum=np.sum(sub, axis=0)
    cost=(1/m)*subsum.item()
    return cost
```

```
def gradient_descent(X,y,theta,learning_rate=0.01,iterations=50):  # function to perform gradietn descent
    m=len(y)

    cost_history=np.zeros(iterations)
    theta_history=np.zeros((iterations,2))

    for it in range(iterations):  # looping iterations

        for i in range(0,len(X)):  # looping samples

            X_row = X[i]
            prediction=np.dot(X_row,theta.T)
            y_row=y.iloc[i]
            y_row=y_row.to_numpy()
            sub=y_row-prediction
            sub_alpha=sub*learning_rate
            theta_sub=X_row *sub_alpha
            theta=np.add(theta,theta_sub)   #updating theta


        theta_history[it,:]=theta             # collecting theta history
        cost_history[it]= cal_cost(theta,X,y)# costs in every iteration stored

    return theta, cost_history,theta_history
```

```
def evaluation(X_test,y_test,theta):  #function to evaluate RMSE
    y_test = y_test.to_numpy()
    m = len(y_test)
    predictions = X_test.dot(theta.T)
    sub = predictions - y_test
    sub = np.square(sub)
    subsum = np.sum(sub, axis=0)
    cost = (1 / m) * subsum.item()
    RMSE=math.sqrt(cost)

    return RMSE


x=preprocessing('house_prices.csv')  # preprocessing the file

X_train,X_test,y_train,y_test=train_test_split(x)  #splitting the dataset
```

```
x=preprocessing('house_prices.csv')  # preprocessing the file

X_train,X_test,y_train,y_test=train_test_split(x)  #splitting the dataset

######### running model with house age feature

print("\n\n\n\nmodel1")

X_one_train=X_train['house age']
X_one_test =X_test['house age']
theta= np.array([[-1,-0.5]])
X_one_b_train=np.c_[np.ones((len(X_one_train),1)),X_one_train]
X_one_b_test= np.c_[np.ones((len(X_one_test),1)),X_one_test]
np.savetxt("X_one_b.csv", X_one_b_train, delimiter=",")
theta,cost_history,theta_history= gradient_descent(X_one_b_train,y_train,theta,0.01)  #calculating gradient descent

size=len(cost_history)

iterationlist=list(range(1,size+1)) # making a list of iteratons for ploting graph
print("iteration ",iterationlist)
plt.figure()
plt.plot(iterationlist,cost_history) # plotting graph between iteration and cost
#plt.show()
RMSE=evaluation(X_one_b_test,y_test,theta) #evaluating RMSE
RMSE_training=evaluation(X_one_b_train,y_train,theta) #evaluating RMSE on train
print("theta final",theta)
print("RMSE ERROR:",RMSE)
print("RMSE_training",RMSE_training)
print('cost history',cost_history)
#print('cost history shape',cost_history.shape)
print('theta_history',theta_history)
```

```
#### running model with distance to nearest station

print("\n\n\n model 2")


X_one_train=X_train['distance to the nearest MRT station']
X_one_test =X_test['distance to the nearest MRT station']
theta= np.array([[-1 ,-0.5]])
X_one_b_train=np.c_[np.ones((len(X_one_train),1)),X_one_train]
X_one_b_test= np.c_[np.ones((len(X_one_test),1)),X_one_test]
#np.savetxt("X_one_b.csv", X_one_b_train, delimiter=",")
theta,cost_history,theta_history= gradient_descent(X_one_b_train,y_train,theta,0.01)
print("theta updated",theta)
print("cost history shape",len(cost_history))
size=len(cost_history)
#iterationlist=list(range(1,size+1))
print("iteration ",iterationlist)
#plt.figure()
#plt.plot(iterationlist,cost_history,label="distance to the nearest MRT station")
#plt.show()
RMSE=evaluation(X_one_b_test,y_test,theta)
print("RMSE ERROR:",RMSE)
print('cost history',cost_history)
```

```
#### running model with number of convenience stores

print("\n\n\n model 3")

X_one_train=X_train['number of convenience stores']
X_one_test =X_test['number of convenience stores']
theta= np.array([[-1,-0.5]])
X_one_b_train=np.c_[np.ones((len(X_one_train),1)),X_one_train]
X_one_b_test= np.c_[np.ones((len(X_one_test),1)),X_one_test]

np.savetxt("X_one_b.csv", X_one_b_train, delimiter=",")
theta,cost_history,theta_history= gradient_descent(X_one_b_train,y_train,theta,0.01)

size=len(cost_history)
iterationlist=list(range(1,size+1))
#plt.figure()
#plt.plot(iterationlist,cost_history,label="number of convenience stores")
#plt.legend()
plt.xlabel('no of iterations')
plt.ylabel('cost function')
plt.title("cost function vs no of iteration ")
plt.tight_layout()
plt.savefig("graph")
RMES=evaluation(X_one_b_test,y_test,theta)
print("RMSE ERROR:",RMES)
print('cost history',cost_history)
```