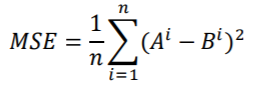
**Machine Learning and Data Mining Lab**

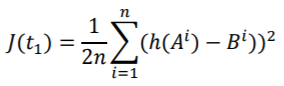
**Lab sheet Linear Regression**

PART A : Prerequisite for linear regression implementation

1. Create an array x = [1, 1, 2, 3, 4, 3, 4, 6, 4] using numpy. Calculate a function h(x)=t0+t1\*x, where t0=1.2 and t1=0.5, for all values of x and plot a graph with x on one axis and h(x)on another axis.
2. Create two arrays A and B with the following values using numpy array. Let (Ai,Bi) represent a data point with i th element of A and B. A = [1, 1, 2, 3, 4, 3, 4, 6, 4] B = [2, 1, 0.5, 1, 3, 3, 2, 5, 4] Find out the dot product of the vectors. [Hint use numpy np.dot(a,b)]
3. Plot a graph marking the data points (Ai,Bi) with A on the X-axis and B on the Y-axis.
4. Calculate Mean Square Error (MSE) of A and B with the formulae where n is the no: of sample data points.



1. Modify the above equation with the following cost function. Implement as a function with prototype def compute\_cost\_function(n,t1,A,B):



Take h(x) =t1\*x and t1= 0.5 Modify the above code iterating for different values of t1 and calculate 𝐽(𝑡1).Try with t1 =0.1,0.3,0.5,0.7,0.8. Plot a graph with t1 on X-axis and 𝐽(𝑡1) on Y-axis. [hint sum\_squared\_error = np.square(np.dot(features, theta) - values).sum() cost = sum\_squared\_error / (2\*m)]

PART B : Linear Regression Implementation

1. Linear regression with one variable.
2. Generate a new data set from student scores with one feature studytime and output variable average grade = (G1+G2+G3)/3
3. Load the new data set
4. Plot data
5. Implement linear regression using inbuilt package python Scikit

from sklearn.linear\_model import LinearRegression

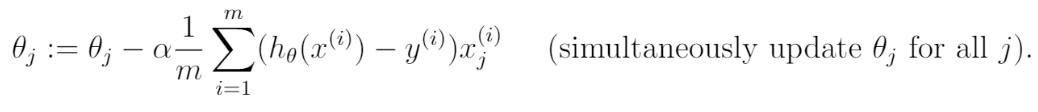
regressor = LinearRegression()

regressor.fit(X\_train, y\_train)

y\_pred = regressor.predict(X\_test) ]

1. Implement gradient descent algorithm with the function prototype def gradient\_descent(alpha, x, y, max\_iter=1500): where alpha is the learning rate, x is the input feature vector. y is the target. Subject the feature vector to normalisation step if needed. Convergence criteria: when no: of iterations exceed max\_iter.

[hint sum\_squared\_error = np.square(np.dot(features, theta) - values).sum() cost = sum\_squared\_error / (2\*m)]



1. Vary learning rate from 0.1 to 0.9 and observe the learned parameter.
2. Draw the contour plot of cost function and simulate the steps of gradient descent.

Example contour for a function

xmesh, ymesh = np.mgrid[-2:2:50j,-2:2:50j]

fmesh = f(np.array([xmesh, ymesh]))

plt.contour(xmesh, ymesh, fmesh)

def f(x):

return 0.5\*x[0]\*\*2 + 2.5\*x[1]\*\*2

1. Do simple k-fold and repeated k-fold. Compute error metrics ME, MAE, MSE, RMSE and compare.

PART B : Extra Credit

1. Implement gradient descend for multivariate linear regression to fit data in full data set.
2. Analyze impact of each input variable on the output variable average grade(g1+g2+g3/3). Plot each input variable vs average grade and analyse the relationship.