**Group #1**

**Balaji Arcot Kulasekaran**

**Bhavna Suri**

**Harsh Kantilal Sharma**

**Rudrashis Poddar**

**Rabie Ahmed M Alsaihati**

**University of Cincinnati**

**College of Applied Science and Engineering**

**Machine learning**

**Professor Anca Ralescu**

**Homework Two**

* **General Steps for all the algorithms before modelling the data:**

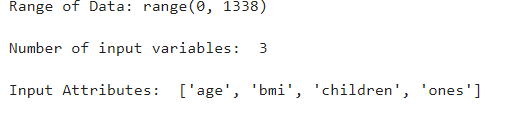
1. Analyze the features and record count for the data set.

Code Snippet:

data = np.genfromtxt("/content/insurance-2.csv", delimiter=',')

X = data[1:, 0:3]#The three columns are in X

T = range(X.shape[0])



1. Compute the mean and standard deviation for each attribute.

Code Snippet:

muX = []

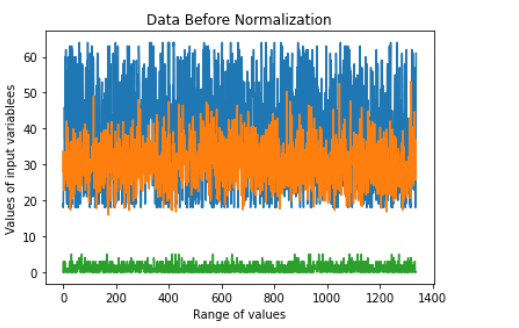
stdX = []

for i in range(X.shape[1]):

plt.plot(T,X[:,i])

muX.append(mean(X[:,i]))

stdX.append(stdev(X[:,i]))



1. Perform data preprocessing to standardize the data and segregate them into train and test set respectively.

Code Snippet:

#data standarization

repmuX=numpy.matlib.repmat(muX,X.shape[0],1)

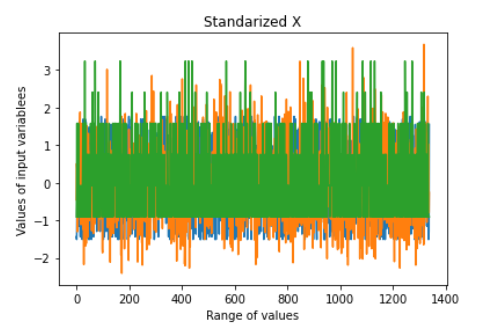
repstdX=numpy.matlib.repmat(stdX,X.shape[0],1)

standardizedX=(X-repmuX)/repstdX

X=standardizedX

for i in range(X.shape[1]):

plt.plot(T,standardizedX[:,i])



# Split data into train and test

y=data[1:,3]

X = np.concatenate([X, np.ones(len(X))[:, np.newaxis]], axis=1)

m=int(0.5\*len(X))

Xtrain = X[0:m,:]#Training data set

ytrain = y[0:m]#Training data set

Xtest = X[m:,:]#test data set

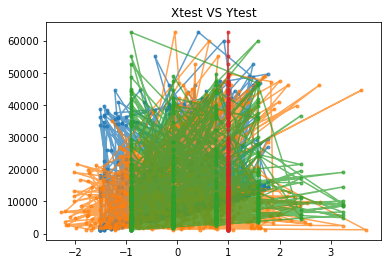
ytest = y[m:]#test data set

#Plotting Xtest VS Ytest

plt.plot(Xtest,ytest, alpha=0.7, marker=".")

plt.title("Xtest VS Ytest")

plt.show()

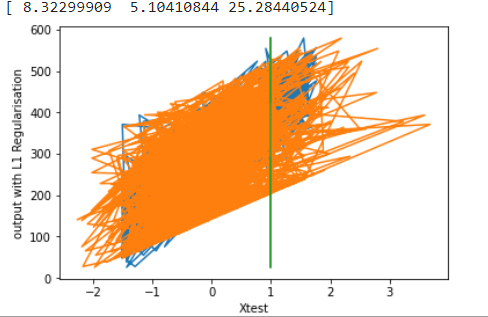


* **For each of the algorithms we:**

1. Train the model
2. Compute MSE for the predictions made
3. Plot regression lines for each attribute based on which the model is trained.

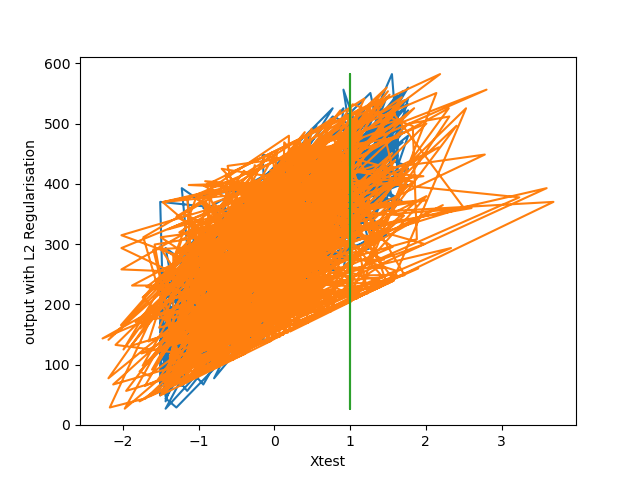
* **Batch Gradient Descent**
* **With L1 Regularization:**

Plot for L1 regularization



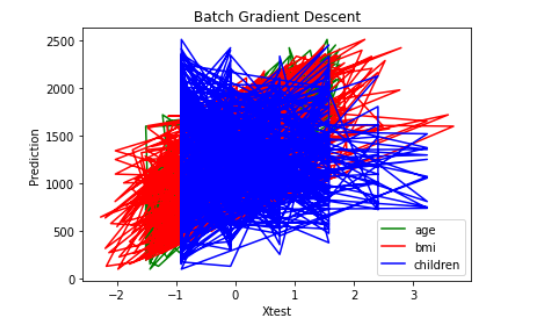
* **With L2 Regularization**

Plot for L2 regularization



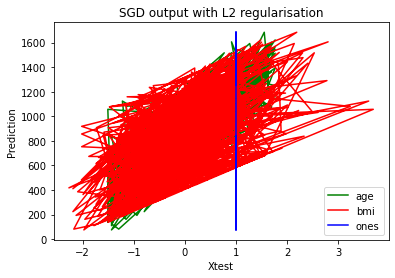
* **Without L2 Regularization**

Plot for without L2 regularization



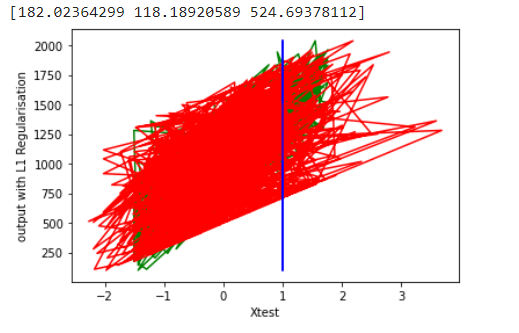
* **Stochastic Gradient Descent**
* **With L2 Regularization**

Plot for L2 regularization



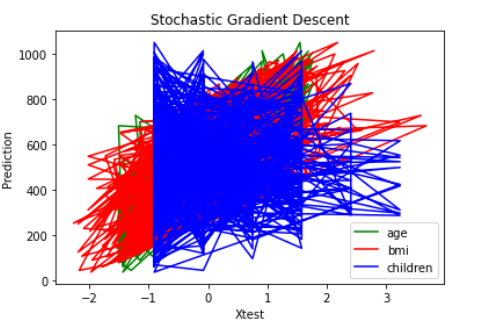
* **With L1 Regularization**

Plot for L1 regularization

****

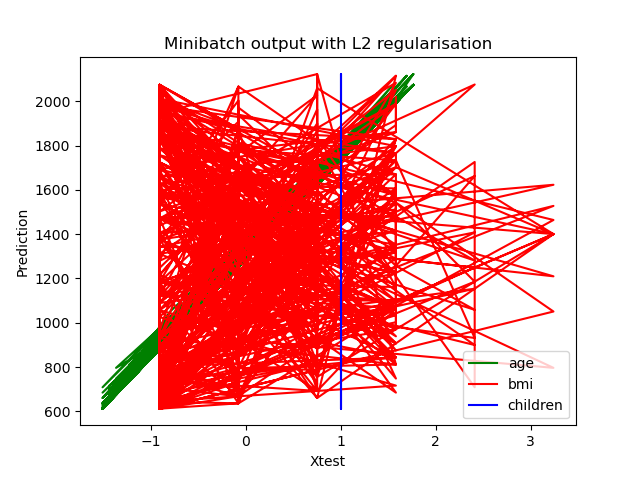
* **Without L2 Regularization**

Plot for without L2 regularization



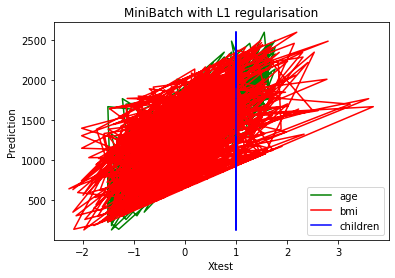
* **Mini Batch**
* **With L2 Regularization**

Plot for L2 regularization



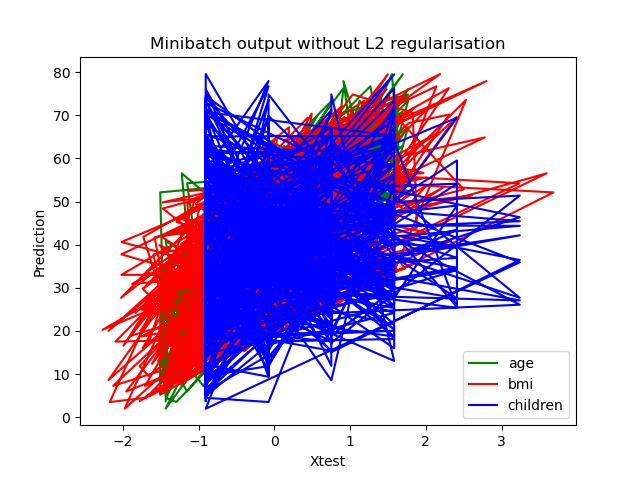
* **With L1 Regularization**

Plot for L1 regularization



* **Without L2 Regularization**

Plot for without L2 regularization



* **Analysis:**
* The stochastic gradient descent and the mini batch gradient descent are fast algorithms in comparison with the batch gradient descent.
* For SGD with L1 regularization the weights were [178.76, 111.36,528] and for L2 [182.02,118.189,524.69]
  + The lowest maximum predicted output was for SGD is the one without regularization as seen in the table below. In contrast, the highest was for SGD with L1 Regularization.
* From the results we obtained, it could be seen that stochastic gradient descent are good but not optimal
* For BGD with L1 regularization the weights were [8.32, 5.1,25.1] and for L2 [9.39,6.48,26.723]
* Batch Gradient Descent can be used for smoother curves.
  + L1 and L2 regularization were similar. The highest was for BGD without Regularization which was 2500.
* All the three gradients descent variants have their advantages and disadvantages
* The weight vectors change slightly in each variant of gradient descent algorithms
* Finally, for Mini-batch algorithms with no regularization the output was 80. For Minibatch with L2 regularization, the output was 2000.

This Table Above summarizes the predicted output VS Xtest for the algorithms