Lab Course Machine Learning **Kalaiselvan Panneerselvam**

Exercise 5 **278920**

1 Data Preprocessing

– Load the data using pandas & dropping NA rows

Bank Marketing Dataset

```
Bank_data = pd.read_csv('bank-additional.csv', delimiter = ';')
Bank_data.dropna().head(5)
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate
0	30	blue-collar	married	basic.9y	no	yes	no	cellular	may	fri	...	2	999	0	nonexistent	-1.8
1	39	services	single	high.school	no	no	no	telephone	may	fri	...	4	999	0	nonexistent	1.1
2	25	services	married	high.school	no	yes	no	telephone	jun	wed	...	1	999	0	nonexistent	1.4
3	38	services	married	basic.9y	no	unknown	unknown	telephone	jun	fri	...	3	999	0	nonexistent	1.4
4	47	admin.	married	university.degree	no	yes	no	cellular	nov	mon	...	1	999	0	nonexistent	-0.1

Wine Dataset

```
#Winedata preprocessing
winewhite_data = pd.read_csv('winequality-white.csv', delimiter = ';')
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	0.307692	0.186275	0.216867	0.308282	0.106825	0.149826	0.373550	0.267785	0.254545	0.267442	0.129032	6
1	0.240385	0.215686	0.204819	0.015337	0.118694	0.041812	0.285383	0.132832	0.527273	0.313953	0.241935	6
2	0.413462	0.196078	0.240964	0.096626	0.121662	0.097561	0.204176	0.154039	0.490909	0.255814	0.338710	6
3	0.326923	0.147059	0.192771	0.121166	0.145401	0.156794	0.410673	0.163678	0.427273	0.209302	0.306452	6
4	0.326923	0.147059	0.192771	0.121166	0.145401	0.156794	0.410673	0.163678	0.427273	0.209302	0.306452	6

- Convert any non-numeric values to numeric values

Bank Marketing Dataset

```
Bank_data = pd.get_dummies(Bank_data)
Bank_data.head()
```

	pdays	previous	emp.var.rate	euribor3m	nr.employed	y	job_admin.	job_blue-collar	job_entrepreneur	job_housemaid	...	education_basic.9y	education_high.s
0	999	0	-1.8	1.313	5099.1	0	0	1	0	0	...	1	
1	999	0	1.1	4.855	5191.0	0	0	0	0	0	...	0	
2	999	0	1.4	4.962	5228.1	0	0	0	0	0	...	0	
3	999	0	1.4	4.959	5228.1	0	0	0	0	0	...	1	
4	999	0	-0.1	4.191	5195.8	0	1	0	0	0	...	0	

- Split the data into a train(80%) and test(20%)

Bank Marketing Dataset

```
x_Train, x_Test = split_DataSet(Bank_data, 0.8)


y_Train = x_Train['y']
x_Train = x_Train.drop('y', axis = 1)

y_Test = x_Test['y']
x_Test = x_Test.drop('y', axis = 1)


print("Training data size " + str(x_Train.shape))
print("Training data size "+ str(x_Test.shape))
```

```
Training data size (3283, 31)
Training data size (836, 31)
```

Wine Dataset

```
#Split wine dataset
trainSet1, testSet1 = split_DataSet(winewhite_data, 0.8)
x_Train1 = trainSet1.as_matrix(columns = ['volatile acidity', 'chlorides', 'density', 'alcohol'])
x_Test1 = testSet1.as_matrix(columns = ['volatile acidity', 'chlorides', 'density', 'alcohol'])
y_Train1 = trainSet1['quality']
y_Test1 = testSet1['quality']
```

1 Linear Classification with Gradient Descent

Exercise 1: Regularization

The objective function (also called the cost) to be minimized is the RSS plus the sum of square of the magnitude of weights. This can be depicted mathematically as:

$$Cost\ (W) = RSS(W) + \lambda * (sum\ of\ squares\ of\ weights)$$

$$= \sum_{i=1}^{N} \left\{ y_i - \sum_{j=0}^{M} w_j\, x_{ij} \right\}^2 + \lambda \sum_{j=0}^{M} w_j^2$$

In this case, the gradient would be:

$$\frac{\partial}{\partial w_j} Cost\ (W) = -2 \sum_{i=1}^{N} x_{ij} \left\{ y_i - \sum_{k=0}^{M} w_k\, x_{ik} \right\} + 2\lambda w_j$$

The Regularization factor is implemented by the below function

```
def l2Loss(lamda, weight):
    loss = 2 * lamda * weight
    loss[0] = 0
    return loss
```

K-fold Cross validation

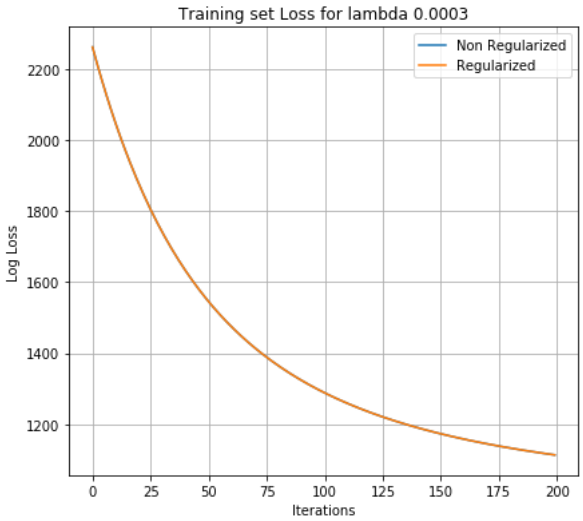
The K-fold is implemented by the below function

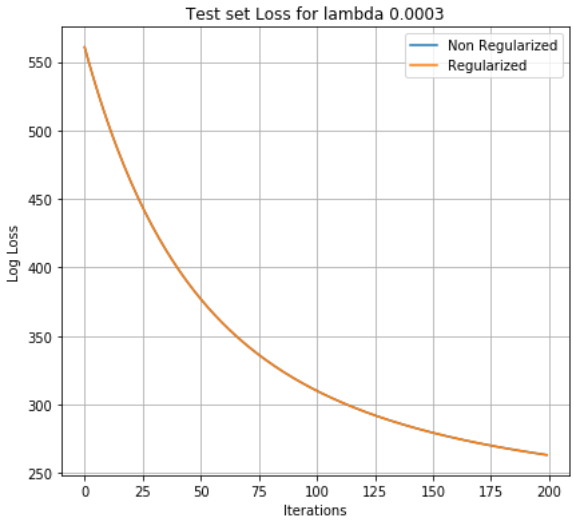
```
def kFold_Cross_Validation(xTrain, yTrain, model, modelParameters, nFolds):
    ind = np.array(range(0, len(xTrain)))
    folds = np.array_split(ind, nFolds)
    minlosslist = []
    plot_logLossTrain = []
    plot_logLossTest = []
    for i in range(0, len(folds)):
        validationSet = folds[i]
        trainSet = np.setdiff1d(ind, validationSet)
        modelParameters['xTrain'] = np.take(xTrain, trainSet, axis = 0)
        modelParameters['yTrain'] = np.take(yTrain, trainSet, axis = 0)
        modelParameters['xTest'] = np.take(xTrain, validationSet, axis = 0)
        modelParameters['yTest'] = np.take(yTrain, validationSet, axis = 0)
        modelParams, trainLoss, testLoss = model(**modelParameters)
        minlosslist.append(testLoss[-1])
        plot_logLossTrain.append(trainLoss)
        plot_logLossTest.append(testLoss)
    return modelParams, plot_logLossTrain, plot_logLossTest, minlosslist
```

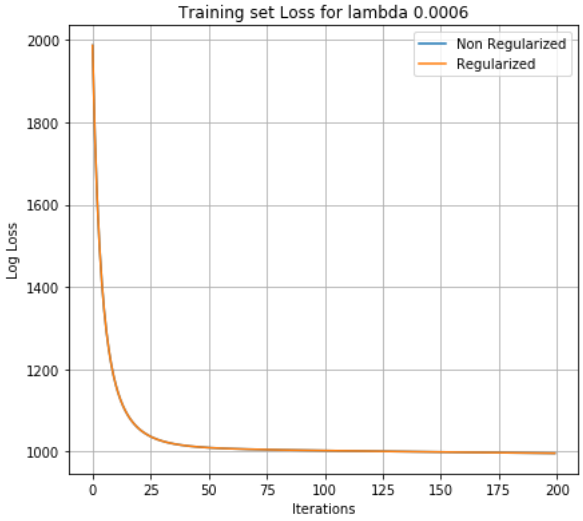
Plots of Regularization and K-fold cross validation with Hyperparameter tuning

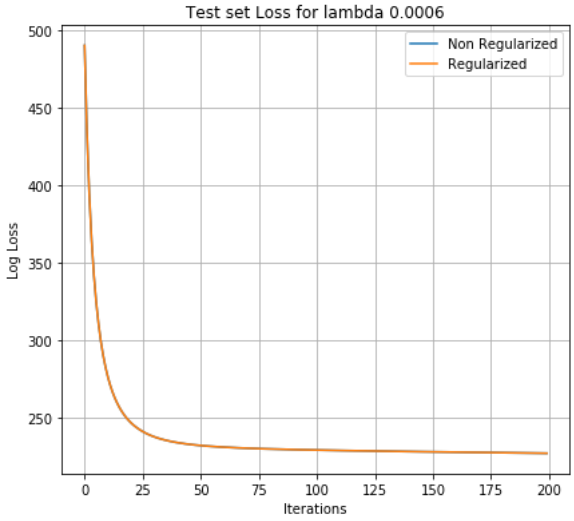
Bank Marketing Dataset

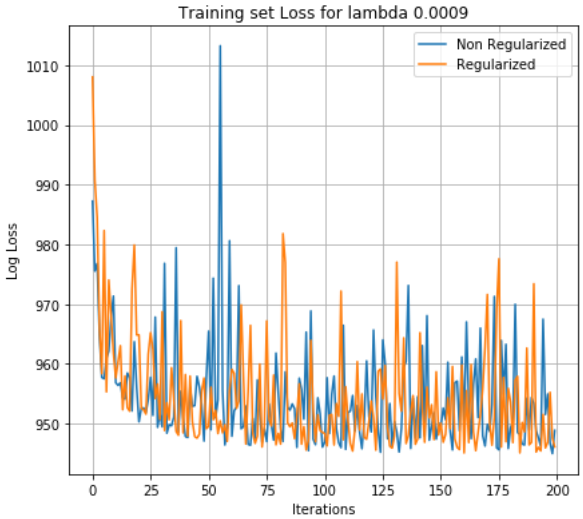
-Plots of Mini Batch Gradient Decent for Bank Dataset with regularisation

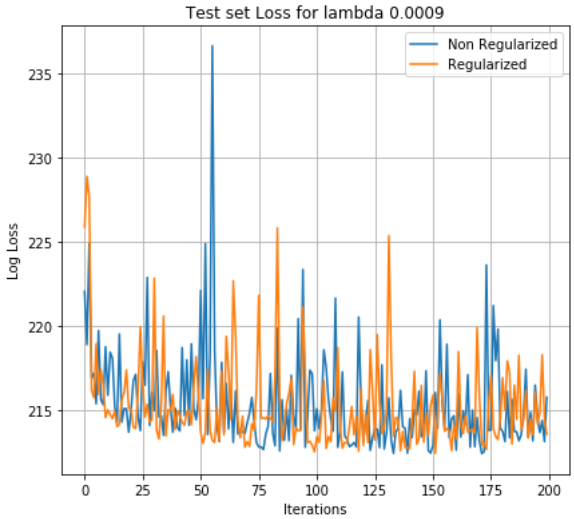






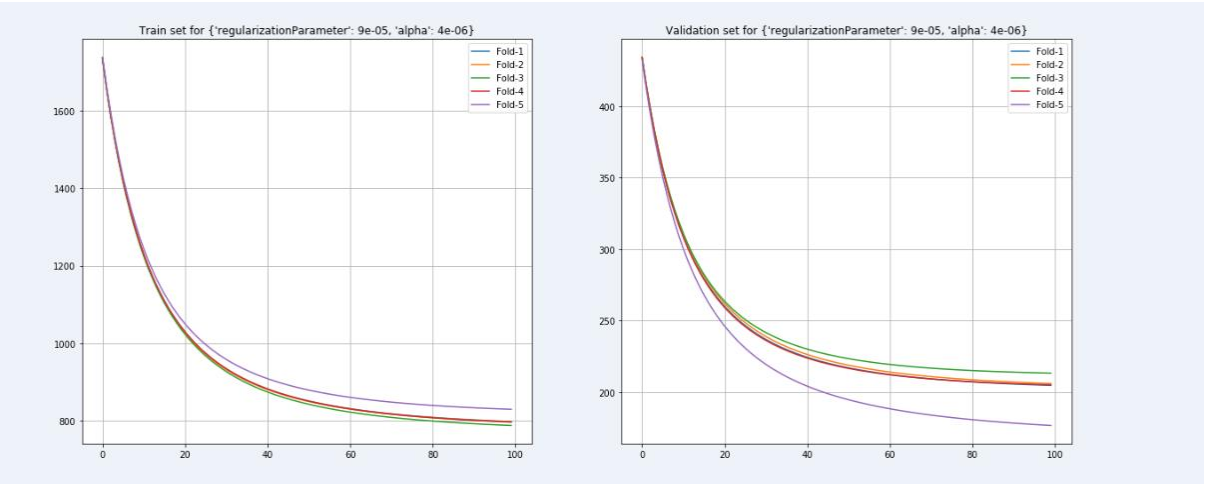


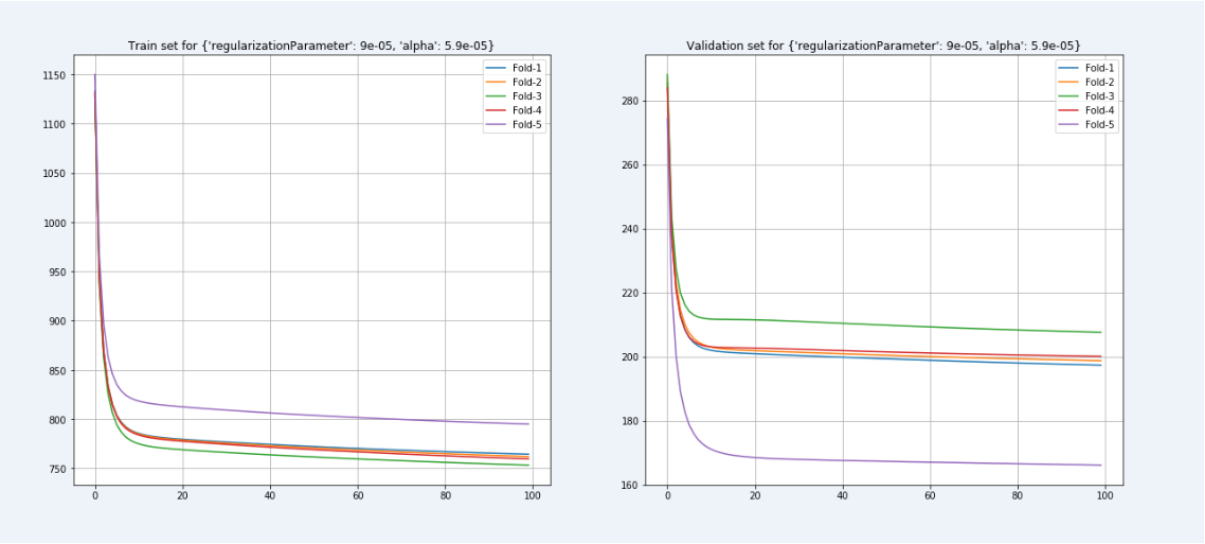


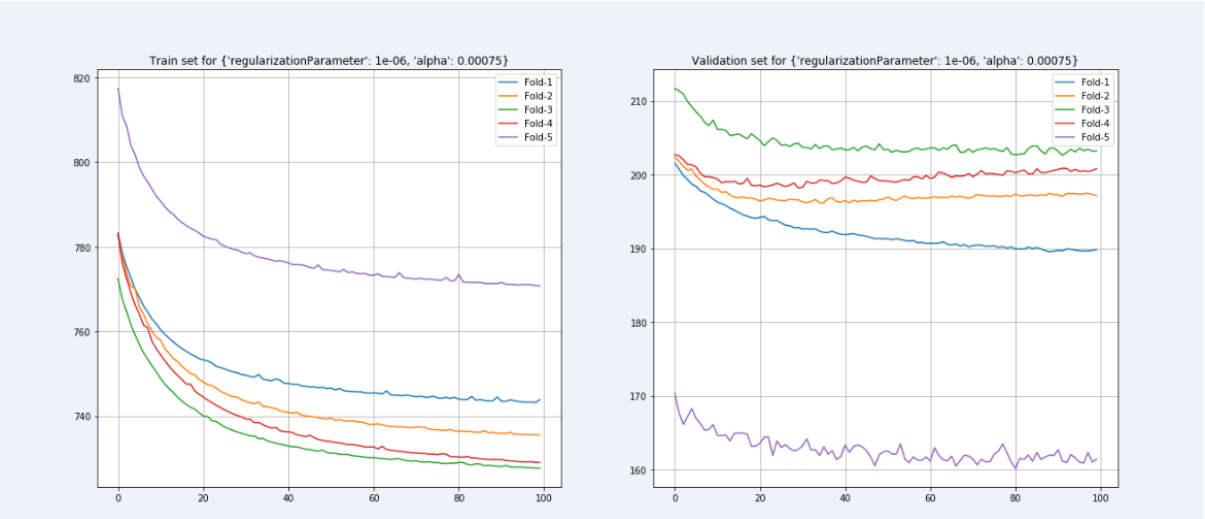


-Plots of Mini Batch Gradient Decent with K-fold cross validation for Bank Dataset with regularisation

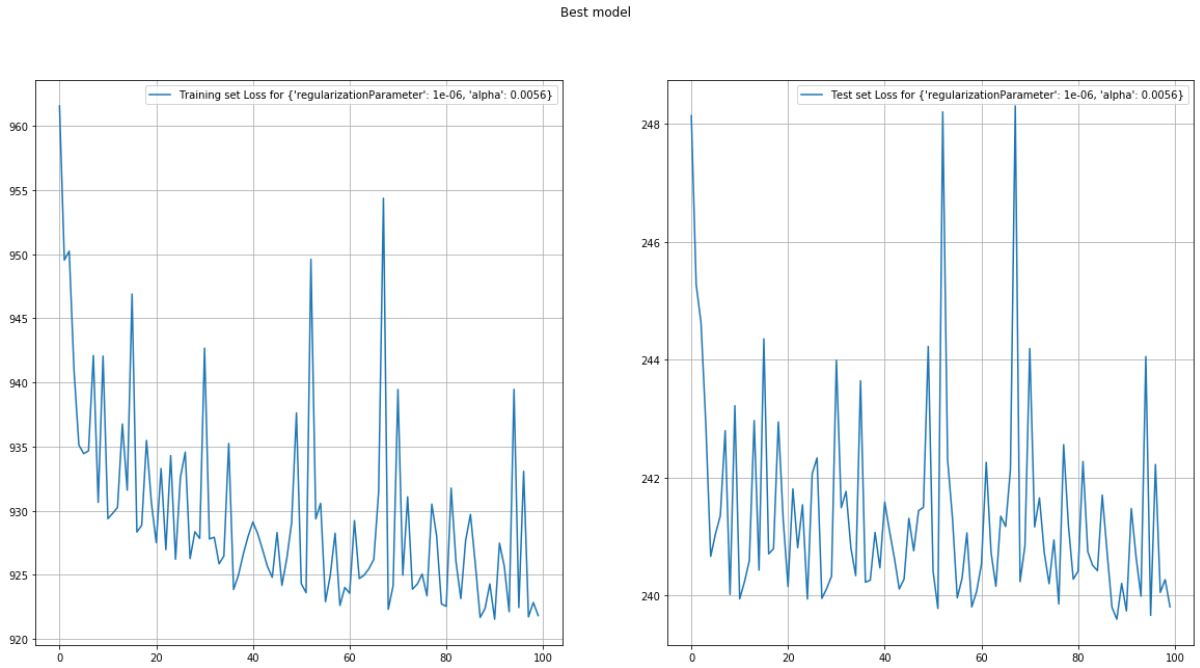
Some sample zoomed in plots:



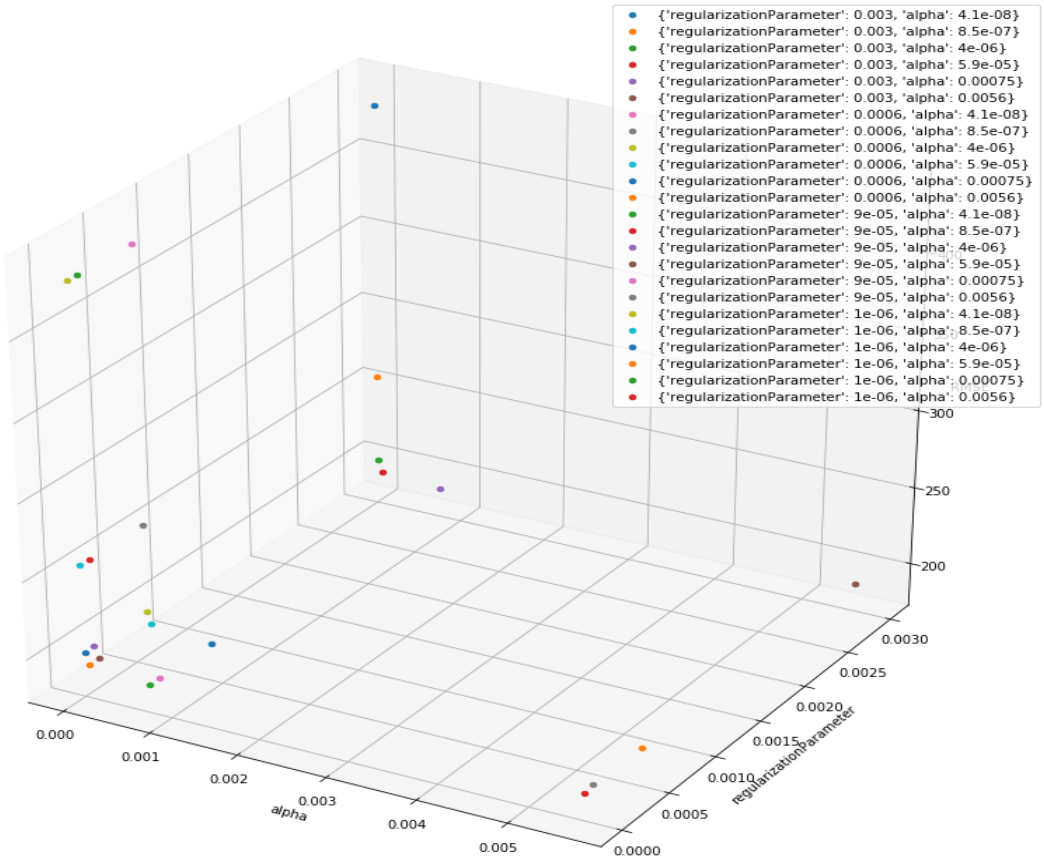




Best Model :

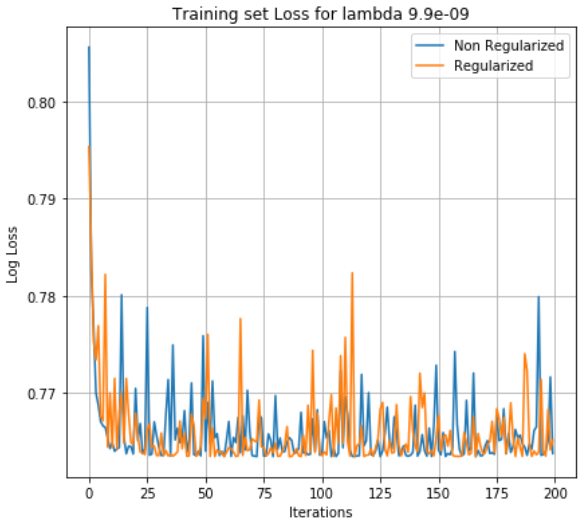


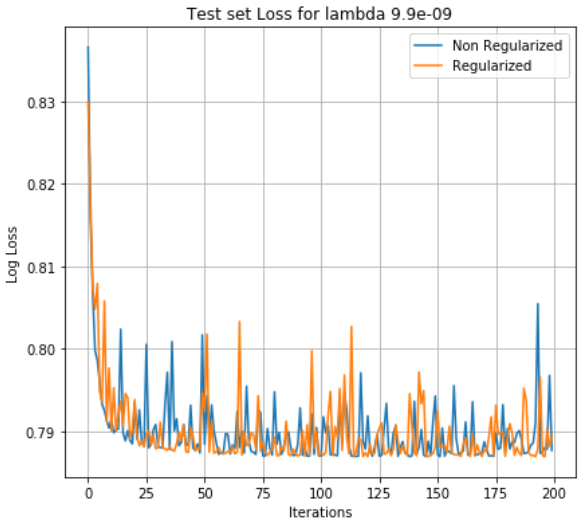
3D Plot:

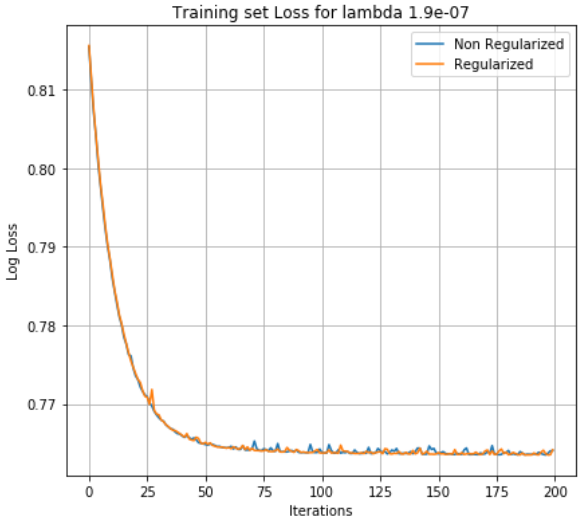


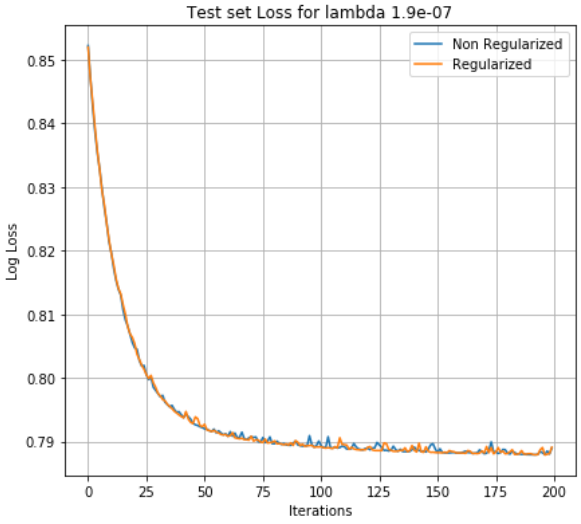
Wine Dataset

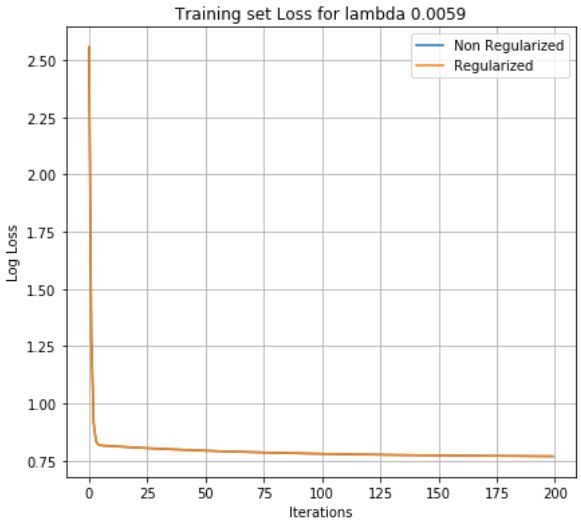
-Plots of Mini Batch Gradient Decent for Bank Dataset with regularisation

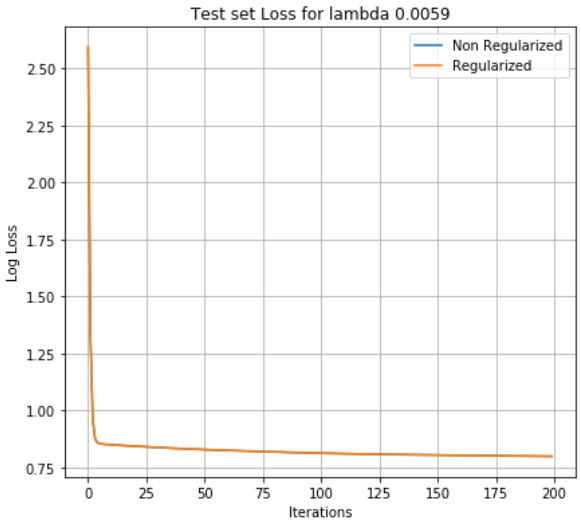




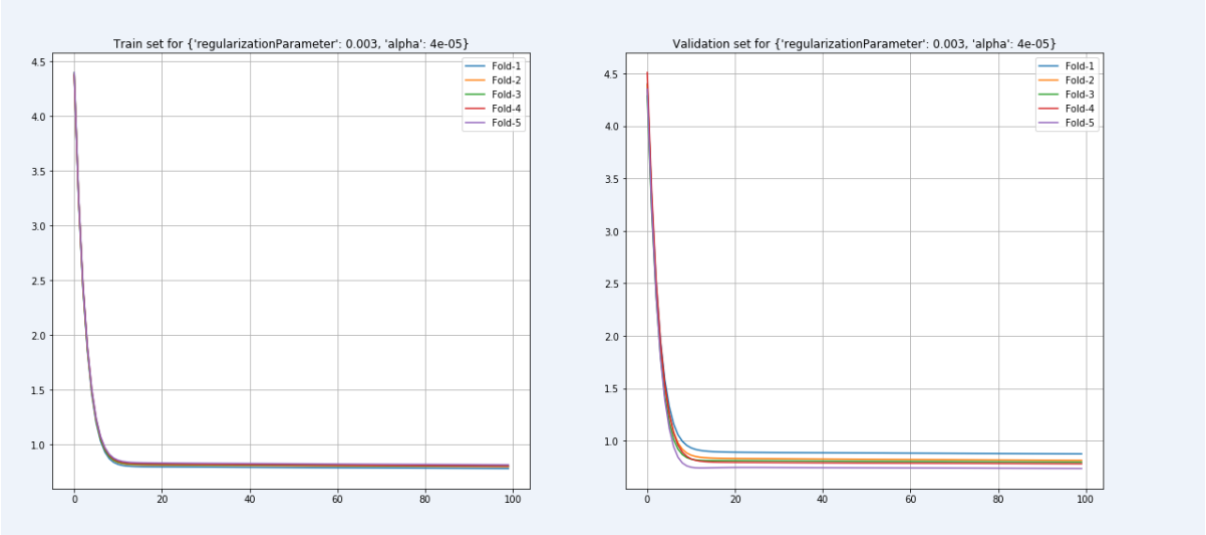


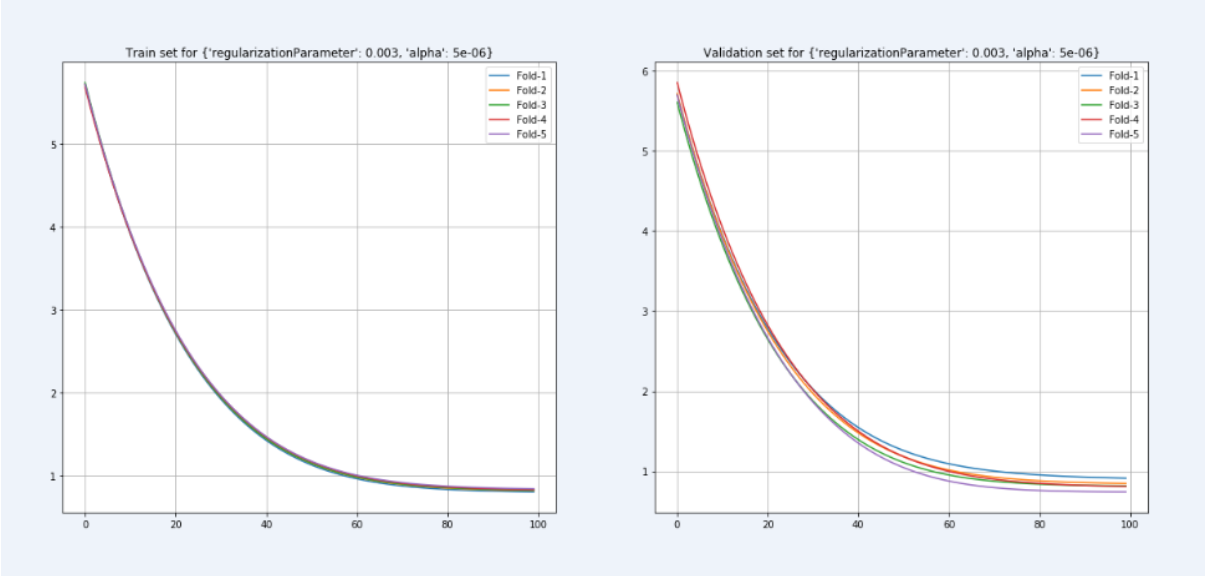


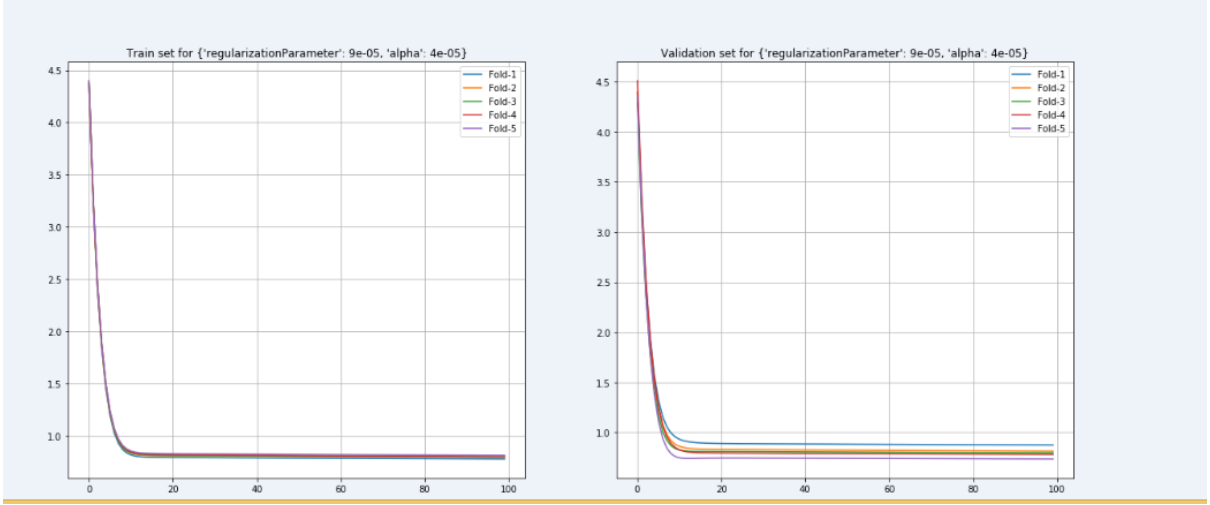




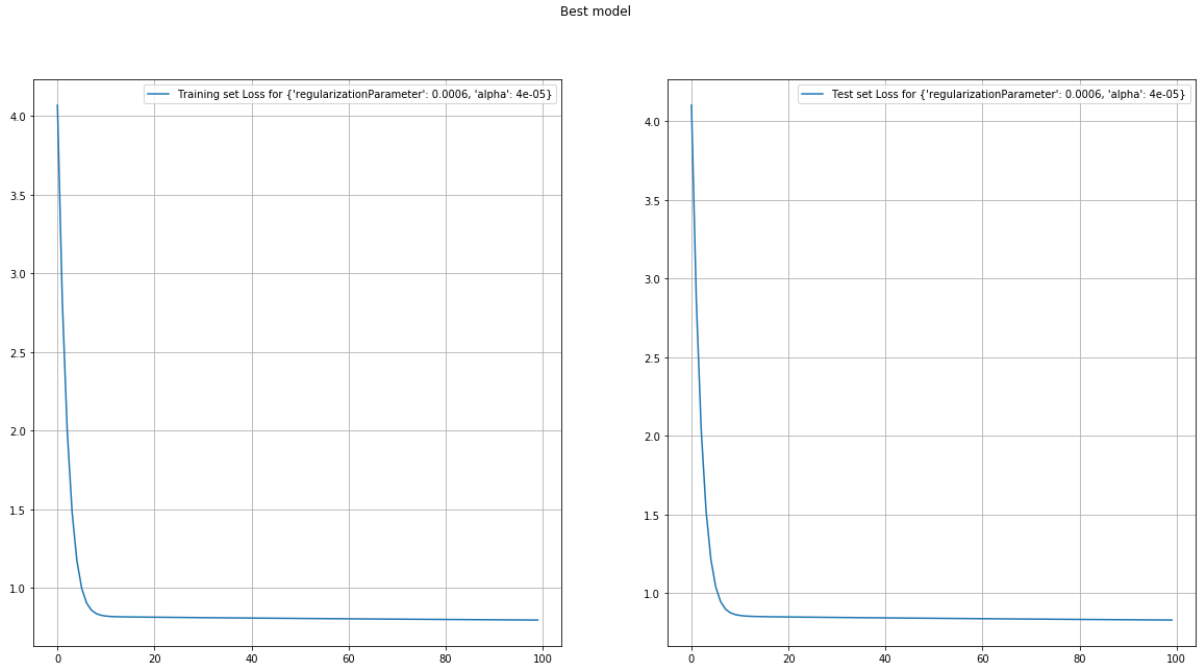
-Sample plots of Mini Batch Gradient Decent with K-fold cross validation for Wine Dataset with regularisation







Best Model :



3D Plot :