**Readme**

**Files Used:-**

1. regression.py
2. logregression.py

There are three questions in the assignment, and the first python files contain a solution to question 1, while the second python files contain a solution to question 2 and 3.

**About regression.py :**

1. regression.py contains the class named Regression.
2. Regression contains the following member functions with the functionalities :

* **NfoldSplit(self,Data,n):** Splits the Data into n folds.
* **MSE(self,ytrue,ypred):** Calculates the Mean Squared Error using ytrue and ypred values.
* **Load\_DataSet(self, path):** Loads the dataset from the specified path.
* **fit(self):** Fits the training data using LinearRegression() of sklearn and save the model.
* **Normal(self):** Fits the training data using Normal Equation and save the model.
* **SKlearn(self):** Fits the training data using LinearRegression() of sklearn and save the model.
* **Load(self):** Loads the model saved by the fit(self) and then calculate and print the Table stats.
* **LoadNormal(self):** Loads the model saved by the Normal(self) and then calculate and print the Table stats.
* **LoadSKlearn(self):** Loads the model saved by the SKlearn(self) and then calculate and print the Table stats using sklearn predict function.

**About logregression.py :**

1. logregression.py contains the class named LogRegression.py.
2. LogRegression contains the following member functions with the functionalities :

* **accuracy(self, actual, predicted):** Calculates the accuracy using actual class labels and predicted class labels.
* **Classwise\_Accuracy(self, Ypred, Yact):** Calculates each class's classwise accuracy using actual class labels and predicted class labels.
* **NfoldSplit(self,Data,n):** Splits the Data into n folds.
* **Load\_DataSet(self, arg):** Loads the dataset from the specified path.
* **Loss(self, Ypred, Yact):** Calculates the cross-entropy loss using the actual class labels and predicted class probabilities.
* **plot(self, Res\_Main):** Plots the training and accuracy curves for all the epochs using data stored in Res\_Main.
* **Hypothesis(self,x,w,b):** Calculates the hypothsis values(sigmoid) form x(data points), w(weights) and b(bias).
* **DerivateFactor(self,Data,w,b,D):** Calculates the derivative factor used in update equation.
* **predict(self, X\_test):** Predicts the class probabilities of the data points X\_test.
* **Scatter\_plot(self,arg):** Creates the scatter plot for the whose path is provided in the arg list.
* **fit(self,arg,reg = None ,method = None,plot=False,print\_table=False,print\_class=False,ret=False):**Fits the training data for each fold with the following method :
* method = None (simple logistic regression)
* method = Regularization (logistic regression with l2 regulization)
* method = One vs One classifier for multiclass classification
* method = One vs Rest classifier for multiclass classification

Weights saving is also done with the fitting of data.

All the methods are coded from scratch.

* **Regularization(self, arg):** Performs the gird search over different values of regularization constants by calling the fit() method with appropriate parameters.
* **Predefined(self, arg,method=None, lamb = None):** Performs the data fitting with the following methods :
* method = None (simple logistic regression)
* method = Regularization (logistic regression with l2 regularization)
* method = One vs. One classifier for multiclass classification
* method = One vs. Rest classifier for multiclass classification

Model saving is also done with the fitting of data.

All the methods are implemented using the sklearn library.

* **loadsaved(self,arg,reg = None ,method = None,plot=False,print\_table=False,print\_class=False,ret = False):** Loads the weights saved by fit() method and then perform theses operations :
* Plots the training and accuracy curves for all the epochs using data stored in Res\_Main.
* Print the table stats.
* Print the classwise accuracies whenever required.
* **LoadReg(self, arg):** Loads the weights saved by Regularization() and print the optimal regularization constant with the table stats.
* **loadpredefined(self,arg,method=None,lamb = None):** Loads the weights saved by Predefined() method and then perform these operations :
* Print the table stats.
* Print the classwise accuracies whenever required.

**Steps to Run the code :**

1. Unzip the submitted files.
2. Store the files in the google drive.
3. To run the regression.py :
   * 1. Store the data set in the drive
     2. Open the regression.py in the google colab.
     3. Mount the drive.
     4. Set the right path argument which contains the dataset.
     5. Compile all the cells to get the required output.
4. To run the regression.py :
   * 1. Store the data set in the drive
     2. Open the logregression.py in the google colab.
     3. Mount the drive.
     4. Set the right path argument which contains the dataset.
     5. Compile all the cells to get the required output.