FRANKFURT UNIVERSIY OF APPLIED SCIENCES, MARCH 2019, AI/ML SOFTWARE ENGINEERING SUPERVISED BY DAMIR DOBRIC

Batch implementation for Logistic Regression Algorithm

Vasuki Muthusamy – Matrikel Number - 1264337

Abstract — Logistic regression is a predictive analysis which is appropriate regression analysis for binary classification.

The prediction procedure is quite similar to multiple linear regressions, with the exception that the response variable is binomial. The multivariable methods explore a relation between two or more predictor (independent) variables and one outcome (dependent) variable. The model describing the relationship expresses the predicted value of the outcome variable as a sum of products, each product formed by multiplying the value and coefficient of the independent variable. The coefficients are obtained as the best mathematical fit for the specified model. A coefficient indicates the impact of each independent variable on the outcome variable adjusting for all other independent variables. In the current model whole training set is used to compute the coefficients (weights). This can be infeasible for large data sets so batch may be used instead, in computing the loss function over subset of the data. During the learning and optimization process, instead of loading all the training dataset at once, they are loaded in sets of specific size. These sets are called batches. Normalization is an important pre-processing step before data is fed into the logistic regression model. Data preprocessor is updated accordingly where the normalization happens per batch.

I. INTRODUCTION

Logistic Regression is one of the most popular ways to fit models for categorical data, especially for binary response data in Data Modeling. The goal of logistic regression is to find the best fitting (yet reasonable) model to describe the relationship between the dichotomous characteristic of interest (dependent variable = response or outcome variable) and a set of independent (predictor or explanatory) variables. Logistic regression generates the coefficients (and its standard errors and significance levels) of a formula to predict a logit transformation of the probability of presence.

The project is divided into four main parts: Introduction, Project description, Implementation, Test and Result, and Discussion.

Introduction section explains Logistic Regression algorithm. Project Description provides a detailed view of Logistic Regression Model. After that, Implementation, Test and Result section help in understanding the code changes performed, test conducted to verify the changes and the corresponding results. The document ends with a conclusion, future works and references used.

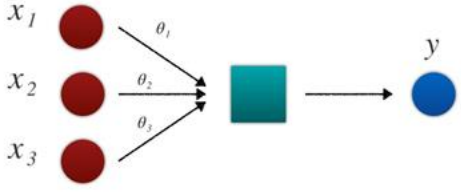
II. PROJECT DESCRIPTION

A. Logistic Regression Model

Logistic regression is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). It is used to predict a binary outcome (1 / 0, Yes / No, True / False) given a set of independent variables. Generally, logistic regression is well suited for describing and testing hypotheses about relationships between a categorical outcome variable and one or more categorical or continuous predictor variables.

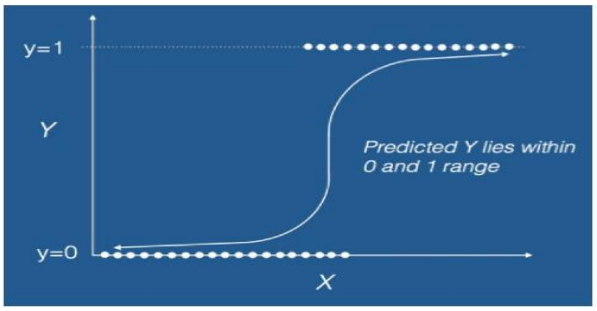
Logistic regression model is shown in Fig. 1.

Fig. 1. Logistic Regression Model



A sample sigmoid curve is shown in Fig. 2.

Fig. 2. Logistic Regression Sigmoid Curve



B. Logistic Regression Calculation

Logistic regression is named for the function used at the core of the method, the logistic function. It is also called the sigmoid function and was developed by statisticians. It's an S-shaped curve (shown in Fig. 2.) that can take any real-valued number and map it into a value between 0 and 1.

Logistic function is given by:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Where f(x) - Standard logistic function

e – The base of the natural logarithms

(Euler's number or the EXP ())

x - Small range of real numbers

(We plug into the function for which logistic value has to be arrived)

The logistic function finds applications in a range of fields, including artificial neural networks, biology (especially ecology), chemistry, demography, economics, geoscience, mathematical psychology, probability, sociology, political science, linguistics, and statistics. Logistic functions are used in logistic regression to model how the probability p of an event may be affected by one or more explanatory variables.

Illustration of Logistic function

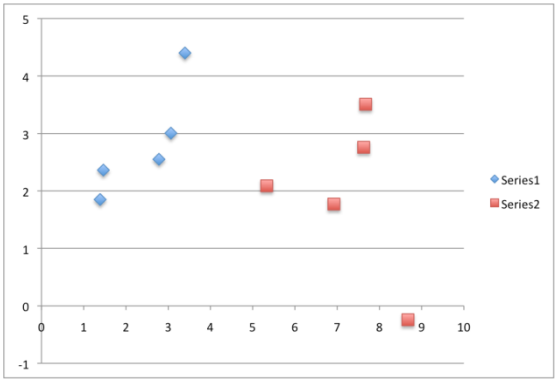
Dataset in Table. 1 has two input variables (X1 and X2) and one output variable (Y). In input variables are real-valued random numbers drawn from a Gaussian distribution. The output variable has two values, making the problem a binary classification problem.

Below is a plot of the dataset. You can see that it is completely contrived and that we can easily draw a line to separate the classes.

Table. 1. Raw Dataset

Independent Variables		Outcome
X1	X2	Y
2.7810836	2.550537003	0
1.465489372	2.362125076	0
3.396561688	4.400293529	0
1.38807019	1.850220317	0
3.06407232	3.005305973	0
7.627531214	2.759262235	1
5.332441248	2.088626775	1
6.922596716	1.77106367	1
8.675418651	-0.242068655	1
7.673756466	3.508563011	1

Fig. 3. Logistic Function Plot



C. Representation Used for Logistic Regression

The Logistic regression model takes real-valued inputs and makes a prediction. Logistic regression uses an equation as the representation, very much like linear regression. Input values are combined linearly using weights or coefficient values to predict an output value. A key difference from linear regression is that the output value being modeled is a binary value (0 or 1) rather than a numeric value.

Model consists of a vector β in d-dimensional feature space. For a point x in feature space, project it onto β to convert it into a real number z in the range - ∞ to + ∞.

Below is an example logistic regression equation:

$$z = \alpha + \boldsymbol{\beta} \cdot \mathbf{x} = \alpha + \beta_1 x_1 + \cdots + \beta_d x_d$$

Where $\mathbf{Z}$ – Predicted Output

$\mathbf{X}$ – d dimensional feature space

Training a logistic regression model actually means to optimize β so the model gives the best possible reproduction of training set labels.

For dataset given in Table. 1, the logistic regression has three coefficients, for example:

$$y = b0 + b1*x1 + b2*x2$$

Where y – Predicted output

b0 – bias or intercept term

b1 – coefficient for the input value x1

b2 – coefficient for the input value x2

Each column in your input data has an associated b coefficient (a constant real value) that must be learned from your training data. The job of the learning algorithm will be to discover the best values for the coefficients (b0, b1 and b2) based on the training data.

D. **Gradient Descent**

Gradient Descent is the process of minimizing a function by following the gradients of the cost function. This involves knowing the form of the cost as well as the derivative so that from a given point you know the gradient and can move in that direction, towards the minimum value.

In machine learning, we can use a technique that evaluates and updates the coefficients for every iteration called gradient descent to minimize the error of a model on our training data.

The way this optimization algorithm works is that each training instance is shown to the model one at a time. The model makes a prediction for a training instance, the error is calculated and the model is updated in order to reduce the error for the next prediction. This procedure can be used to find the set of coefficients in a model that result in the smallest error for the model on the training data.

Given each training instance:

1. Calculate a prediction using the current values of the coefficients.

2. Calculate new coefficient values based on the error in the prediction.

The process is repeated until the model is accurate enough (e.g. error drops to some desirable level) or for a fixed number iterations. You continue to update the model for training instances and correcting errors until the model is accurate enough or cannot be made any more accurate. It is often a good idea to randomize the order of the training instances shown to the model to mix up the corrections made.

E**. Making Prediction**

Making predictions with a logistic regression model is as simple as plugging in numbers into the logistic regression equation and calculating a result. All we do is to plug the variables into the logistic regression equation specified by the regression coefficients estimates that we get from the coefficient table output.

III. IMPLEMENTATION, TEST AND RESULT

A. Logistic Regression – Batch Implementation

Logistic regression module is already available as a part of LearningAPI. So our objective here is to implement batched processing for existing algorithm in the Learning API.

Logistic Regression pipeline modules are:

1) Data Mapping
2) Normalization of data
3) Training the regression model and arriving at coefficients
4) Prediction of outcome using the learned values
5) Validation of prediction

Since it is an addition of functionality to an existing algorithm it necessitates certain modification to the LearningAPI modules.

LearningAPI modules impacted for the batch implementation are as follows:

1) Normalization of data

In batch processing only a subset of data is processed at a given time to in order to normalize the data the minimum and maximum values are continuously updated for each batch. The normalization is performed based on the overall minimum and maximum values of the batch data processed inclusive of the data in the current batch.

2) Training is performed by running the process in batch

For each batch of data the coefficients are calculated and the learned data is used in the next batch. In each batch the model will have learning. So, the final model will have the learning accumulated thought the process.

Given below are the flow charts for logistic regression in normal mode and batch mode.

Fig. 4. Logistic Regression Normal Mode – Flow chart

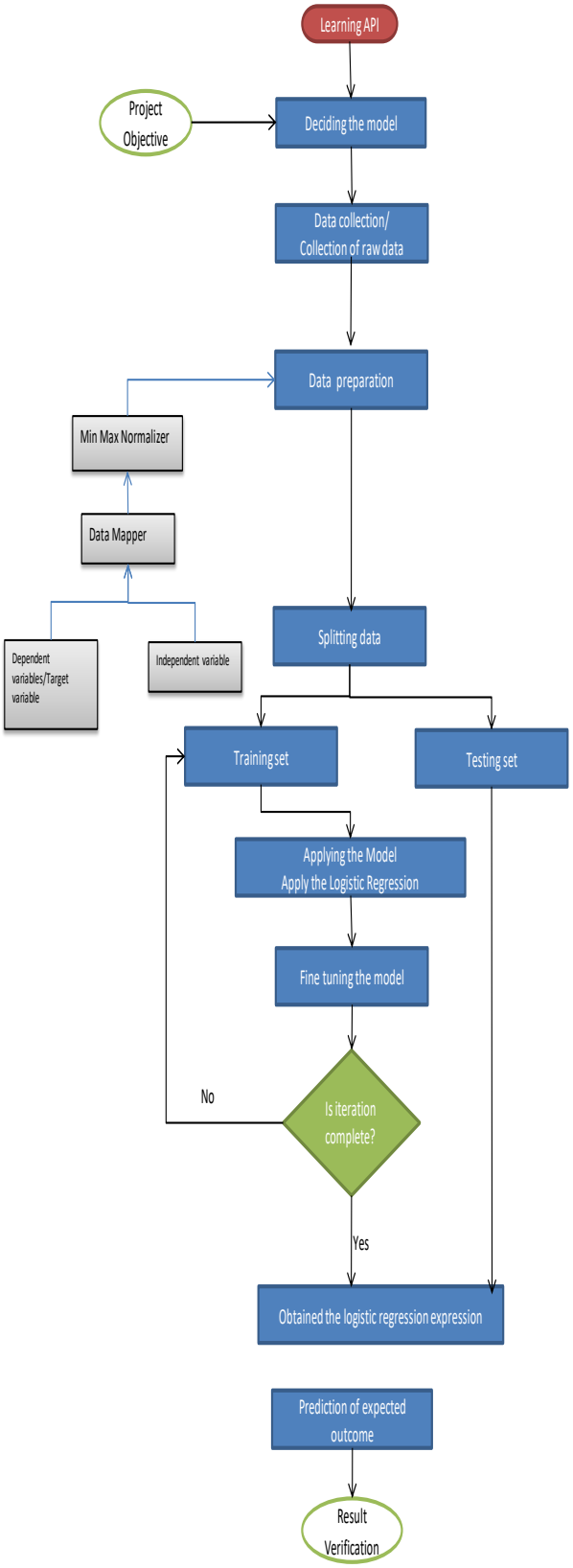
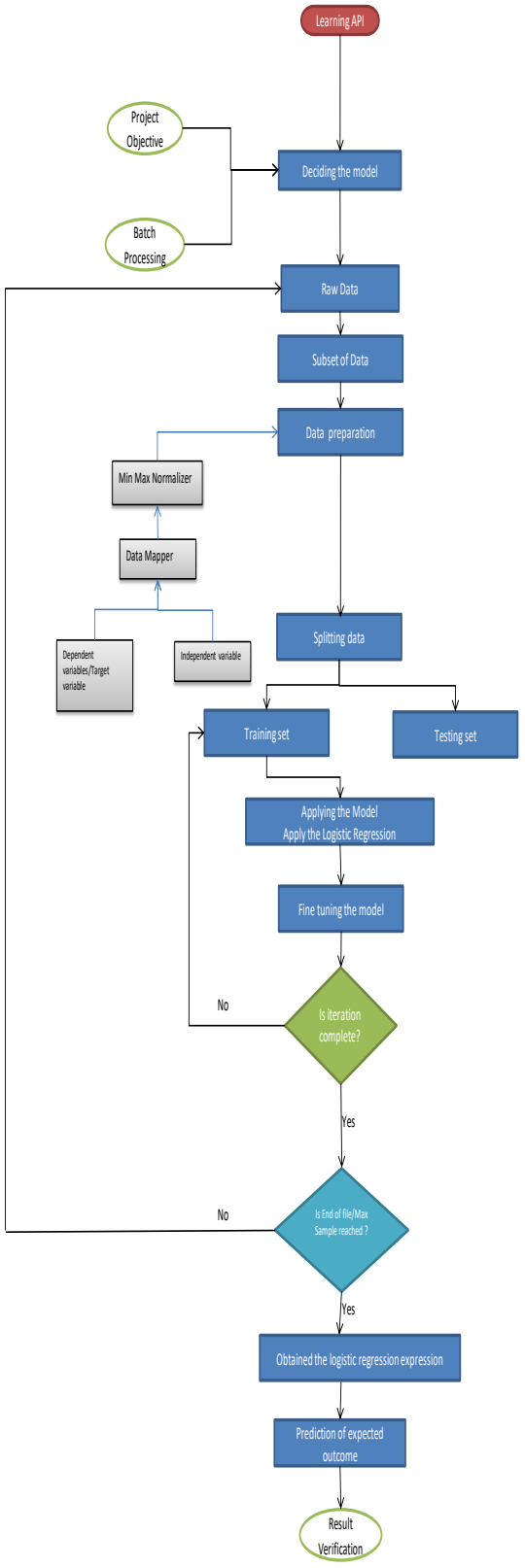


Fig. 5. Logistic Regression Batch Mode – Flow chart



Function description about each pipeline module:

1) LearningAPI

LearningAPI is the API containing Machine Learning algorithms, which can run in the pipeline of modules compatible to each other. LearningAPI contains useful libraries fully implemented in .NET Core. It provides a platform to build, modify and execute machine learning algorithms.

2) Batching raw data

Batch refers to how training samples are used while computing coefficients. Batch size implies the number of training examples in one pass. The higher the batch size, the more memory space needed for the run. Thought the “batch" and "online" approaches are similar they can produce very different results. In spite of the superiority of online training, batch training is important in processing large batches of training points in machine learning. In LearningAPI batching technique depends on the mode of input data. E.g. Inline data, CSV file, text file.

a) For Inline data below is the code snippet used to obtain data in batches. Batch size and maximum samples are supplied as input values. Batch size defines the number if records to be processed in a given pass.

```
if (currentBatch < maxSamples / batchSize)
{
    List<object[]> batch = new List<object[]>();

    batch.AddRange(data.Skip(currentBatch * batchSize).Take(batchSize));

    ctx.IsMoreDataAvailable = true;

    currentBatch++;

    return batch.ToArray();
}
else
{
    ctx.IsMoreDataAvailable = false;
    return null;
}
```

b) For CSV file a new class CsvDataProviderBatch () is added to the module CsvDataProvider. Batch size is provided as a parameter to the class. Batch size defines the number if records to be processed in a given pass.

```
/// Reading CSV file for fixed batch size
/// </summary>
public object[][] Run(object data, IContext ctx)
{
    List<object[]> rawData = new List<object[]>();

    using (StreamReader reader = File.OpenText(m_FileName))
    {
        int linenum = 0;
        int linecount = 0;
        foreach (string line in readLineFromFile(reader))
        {
            //split line in to column
            var strCols = line.Split(m_Delimiter);

            //skip first ... rows
            var headerLine = isHeaderIncluded ? 1 : 0;
            if (linenum < m_SkipRows + headerLine + m_LinesRead)
            {
                linenum++;
                continue;
            }

            //Transform data from row->col in to col->row
            var singleRow = new object[strCols.Length];

            //define columns
            for (int i = 0; i < strCols.Length; i++)
            {
                singleRow[i] = strCols[i];
            }

            rawData.Add(singleRow);

            linecount++;
            if (linecount == m_BatchSize)
            {
                m_LinesRead = m_LinesRead + m_BatchSize;
                ctx.IsMoreDataAvailable = true;
                return rawData.ToArray();

            }
        }
        if (reader.EndOfStream == true)
            ctx.IsMoreDataAvailable = false;
        return rawData.ToArray();
    }
}
```

3) DefaultDataMapper

DefaultDataMapper performs the preprocessing of raw data. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues. For regression data preprocessing includes transforming the categorical data to indicator data which is performed by the DataMapper.

4) MinMaxNormalizer

Minmax normalization is a normalization strategy which linearly transforms x to y= (x-min)/ (max-min), where min and max are the minimum and maximum values in X, where X is the set of observed values of x.

5) LogisticRegression

Learning rate of 0.13 and iteration of 10 were identified as appropriate values to perform LogisticRegression. First batch is very much similar to the current online regression model. From the second batch run we can observe that instead of starting the training with initialized weights (coefficients) values from last run were utilized. This repeats till the end of maximum sample or end of file.

6) RunBatch

RunBatch module is used in place of Run to perform batch processing.

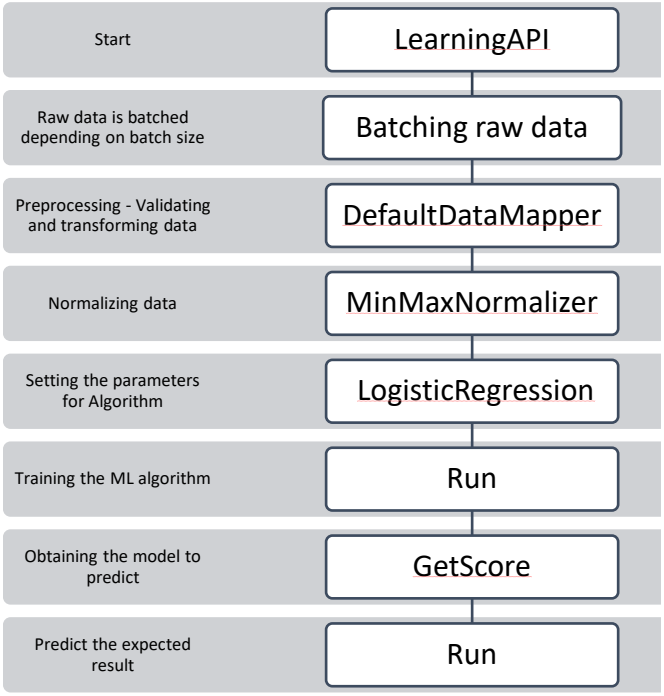
7) GetScore

GetScore help is obtaining the coefficient values needed to use in the Logistic Regression model and the error calculated during each iteration for reference purpose.

6) Run

The final run is to perform the prediction of expected results which hells in evaluating the learning of algorithm.

Fig. 6. Logistic Regression Pipeline Modules – Flow chart



B. **Unit test and result**

Execution steps of batch process in LearningAPI.

Let's take unit test "LogisticRegressionBatchwithCSV" as an example. In this unit test the input data file "admit_binary.csv" is a CSV file and has about 395 records. The input file has 4 columns out of which three are predictors GRE,GPA ,rank and one output admit.

Step 1: To identify a batch size: The process needs an input value called batch size. In this unit test the batch size is set as '50' as shown below:

```
public void LogisticRegression_Batch_Real_Example()
{
    string m_binary_data_path = @"SampleData\binary\admit_binary.csv";

    var binary_path = System.IO.Path.Combine(Directory.GetCurrentDirectory(), m_binary_data_path);

    LearningApi api = new LearningApi(loadMetaData1());
    api.UseBatchCsvDataProvider(binary_path, ',', false, 1, 50);
```

Step 2: Batch run for training: Once the initial set up is done the logistic regression performed using the 'RunBatch' process. Here, the run process will repeat itself automatically after every '50' records till end of file is reached without any manual intervention. After every execution 'weights' of the predictors are stored and used in the subsequent runs to obtain final score. Below is the variable that stores the final score:

```
LogisticRegressionScore score = api.GetScore() as LogisticRegressionScore;
```

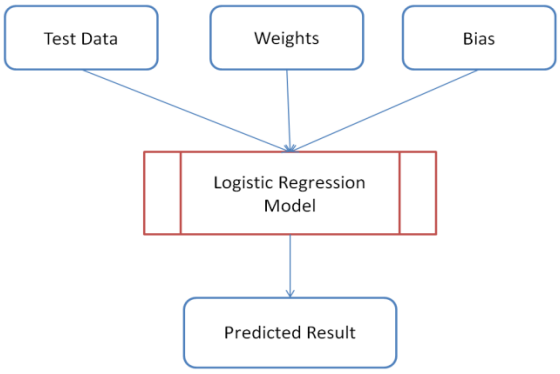
Step 3: Prediction of result: Once the scores for the predictors are available prediction can be performed. In this unit test the prediction is performed and results are obtained as below:

```
var result = api.Algorithm.Predict(testData as double[][], api.Context) as LogisticRegressionResult;
```

By using the above steps logistic regression model can be trained with files having large volume of input records and results can be predicted.

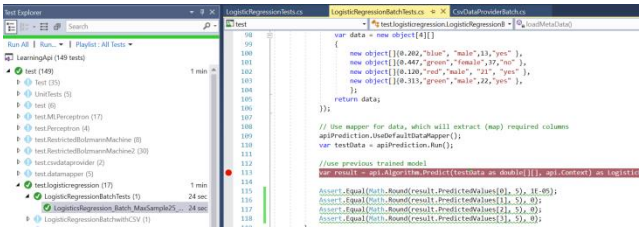
The flowchart of test steps is described below.

Fig. 7. Testing Process Flow Chart – Flow chart



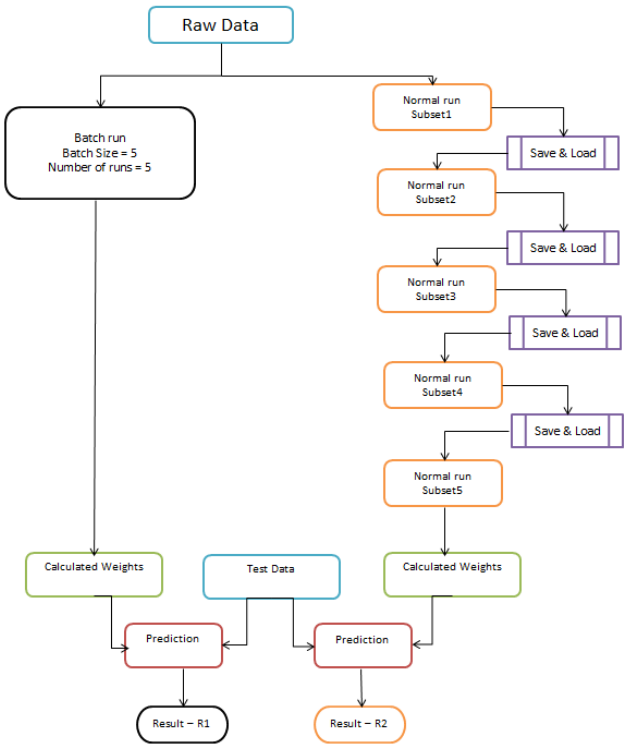
1) LogisticRegressionBatchTests

This unit test performs the training of the algorithm by supplying data in batch. Iteration is set to 10 and learning rate is 0.15.Maximum data sample used is 25 and the batch size is provided as 5.

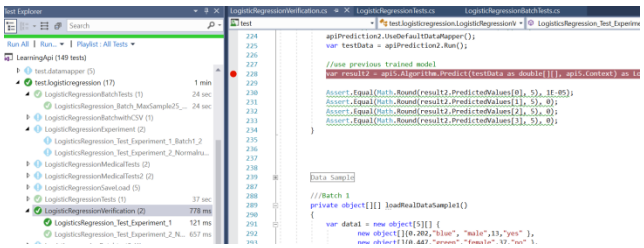


2) LogisticRegressionVerification

In the unit test ((1) LogisticRegressionBatchTests) though the predicted values are similar to the values obtained in normal run the values for weight differed. So to confirm the batch modifications are performed correct we adopted the below experiment.

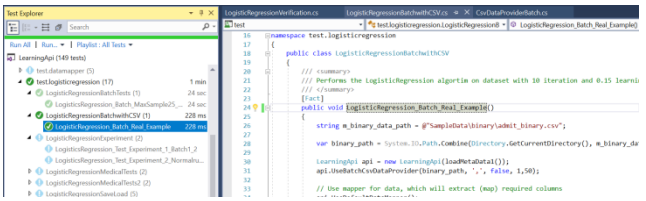


Here after obtaining results R1 and R2 are compared and the experiment resulted in R1 = R2. With this the batch run is validated.



3) LogisticRegressionBatchwithCSV

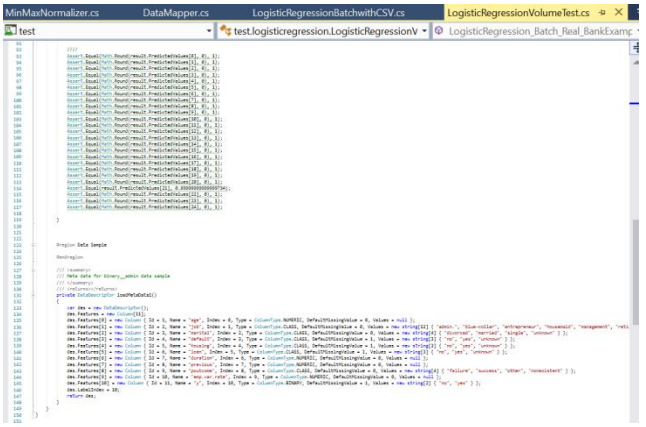
To implement batch process with input data coming from CSV file there is a new class added to the module called CSVDataProviderBatch.cs. This uses the input file called admit_binary.csv. The CSV file is processed with a batch size of 50. Unit test was successful and the predicted result matched the result from normal run.



4) LogisticRegressionVolumeTest

To perform the volume test with lots of data. I have selected a dataset https://archive.ics.uci.edu/ml/datasets/Bank+Marketing

Which had the bank data and based on certain feature the algorithm decides whether a customer will subscribe for a term deposit or not.



IV. DISCUSSION

A. Conclusion

In this project an approach for batch implementation for logistic regression algorithm has been presented. Given a large dataset this method can be adopted and a predicted result can be obtained

The whole program is run in C# .NET Core.

B. Future Challenge

Adoption of batch processing for the other machine learning algorithms in the LearningAPI.

V. REFERENCE

1. https://github.com/UniversityOfAppliedSciencesFrankfurt/LearningApi

2. https://github.com/UniversityOfAppliedSciencesFrankfurt/LearningApi/blob/master/WorkingWithBatches.md
3. https://machinelearningmastery.com/logistic-regression-tutorial-for-machine-learning/

4. https://christophm.github.io/interpretable-ml-book/logistic.html

5. https://medium.com/greyatom/logistic-regression-89e496433063