CmpE 462 : Machine Learning

Assignment 2

Name & Surname : Emre Girgin
ID : 2016400099

Part 1

In this part of the assignment I've trained 6 different Logistic Regression models. (2 batch sizes x 3 Learning rates) Only van and saab classes are taken. The features of the samples are normalized based on **MinMaxNormalization**. For stopping the training, I've checked the training loss of the last two epochs. If their difference is lower than a threshold, training stops. As the learning rate I've tested with 0.01, 0.5 and 1.0. Note that the x-axis of the figures is the iterations, not the epochs.

If you want to see additional figures and the training details, make sure that you have made *debug* variable from the code is **True.**

```
Iteration: 470 | FoldID: 1 | LearningRate: 0.01 | Batch Size: 332 | Train Loss: 0.5982 | Val Loss: 0.7037 | Accuracy: 0.59 | Diff: 0.00010
Iteration: 471 | FoldID: 1 | LearningRate: 0.01 | Batch Size: 332 | Train Loss: 0.5981 | Val Loss: 0.7036 | Accuracy: 0.59 | Diff: 0.00010
Iteration: 472 | FoldID: 1 | LearningRate: 0.01 | Batch Size: 332 | Train Loss: 0.5980 | Val Loss: 0.7035 | Accuracy: 0.59 | Diff: 0.00010
Iteration: 473 | FoldID: 1 | LearningRate: 0.01 | Batch Size: 332 | Train Loss: 0.5979 | Val Loss: 0.7034 | Accuracy: 0.59 | Diff: 0.00010
Iteration: 474 | FoldID: 1 | LearningRate: 0.01 | Batch Size: 332 | Train Loss: 0.5978 | Val Loss: 0.7033 | Accuracy: 0.59 | Diff: 0.00010
Iteration: 475 | FoldID: 1 | LearningRate: 0.01 | Batch Size: 332 | Train Loss: 0.5977 | Val Loss: 0.7032 | Accuracy: 0.59 | Diff: 0.00010
Iteration: 476 | FoldID: 1 | LearningRate: 0.01 | Batch Size: 332 | Train Loss: 0.5976 | Val Loss: 0.7031 | Accuracy: 0.59 | Diff: 0.00010
Iteration: 477 | FoldID: 1 | LearningRate: 0.01 | Batch Size: 332 | Train Loss: 0.5975 | Val Loss: 0.7030 | Accuracy: 0.59 | Diff: 0.00010
Iteration: 478 | FoldID: 1 | LearningRate: 0.01 | Batch Size: 332 | Train Loss: 0.5974 | Val Loss: 0.7030 | Accuracy: 0.59 | Diff: 0.00010
Iteration: 479 | FoldID: 1 | LearningRate: 0.01 | Batch Size: 332 | Train Loss: 0.5973 | Val Loss: 0.7029 | Accuracy: 0.59 | Diff: 0.00010
Iteration: 480 | FoldID: 1 | LearningRate: 0.01 | Batch Size: 332 | Train Loss: 0.5972 | Val Loss: 0.7028 | Accuracy: 0.59 | Diff: 0.00010
Iteration: 481 | FoldID: 1 | LearningRate: 0.01 | Batch Size: 332 | Train Loss: 0.5971 | Val Loss: 0.7027 | Accuracy: 0.59 | Diff: 0.00010
Iteration: 482 | FoldID: 1 | LearningRate: 0.01 | Batch Size: 332 | Train Loss: 0.5970 | Val Loss: 0.7026 | Accuracy: 0.59 | Diff: 0.00010
Iteration: 483 | FoldID: 1 | LearningRate: 0.01 | Batch Size: 332 | Train Loss: 0.5969 | Val Loss: 0.7025 | Accuracy: 0.59 | Diff: 0.00010
Iteration: 484 | FoldID: 1 | LearningRate: 0.01 | Batch Size: 332 | Train Loss: 0.5968 | Val Loss: 0.7024 | Accuracy: 0.59 | Diff: 0.00010
Iteration: 485 | FoldID: 1 | LearningRate: 0.01 | Batch Size: 332 | Train Loss: 0.5967 | Val Loss: 0.7023 | Accuracy: 0.59 | Diff: 0.00010
Iteration: 486 | FoldID: 1 | LearningRate: 0.01 | Batch Size: 332 | Train Loss: 0.5966 | Val Loss: 0.7022 | Accuracy: 0.59 | Diff: 0.00010
Iteration: 487 | FoldID: 1 | LearningRate: 0.01 | Batch Size: 332 | Train Loss: 0.5965 | Val Loss: 0.7021 | Accuracy: 0.59 | Diff: 0.00010
Iteration: 488 | FoldID: 1 | LearningRate: 0.01 | Batch Size: 332 | Train Loss: 0.5964 | Val Loss: 0.7020 | Accuracy: 0.59 | Diff: 0.00010
Iteration: 489 | FoldID: 1 | LearningRate: 0.01 | Batch Size: 332 | Train Loss: 0.5963 | Val Loss: 0.7019 | Accuracy: 0.59 | Diff: 0.00010
Iteration: 490 | FoldID: 1 | LearningRate: 0.01 | Batch Size: 332 | Train Loss: 0.5962 | Val Loss: 0.7018 | Accuracy: 0.60 | Diff: 0.00010
Iteration: 491 | FoldID: 1 | LearningRate: 0.01 | Batch Size: 332 | Train Loss: 0.5961 | Val Loss: 0.7017 | Accuracy: 0.60 | Diff: 0.00010
Iteration: 492 | FoldID: 1 | LearningRate: 0.01 | Batch Size: 332 | Train Loss: 0.5960 | Val Loss: 0.7016 | Accuracy: 0.60 | Diff: 0.00010
Iteration: 493 | FoldID: 1 | LearningRate: 0.01 | Batch Size: 332 | Train Loss: 0.5959 | Val Loss: 0.7015 | Accuracy: 0.60 | Diff: 0.00010
Iteration: 494 | FoldID: 1 | LearningRate: 0.01 | Batch Size: 332 | Train Loss: 0.5958 | Val Loss: 0.7014 | Accuracy: 0.60 | Diff: 0.00010
Iteration: 495 | FoldID: 1 | LearningRate: 0.01 | Batch Size: 332 | Train Loss: 0.5957 | Val Loss: 0.7013 | Accuracy: 0.60 | Diff: 0.00010
Iteration: 496 | FoldID: 1 | LearningRate: 0.01 | Batch Size: 332 | Train Loss: 0.5956 | Val Loss: 0.7012 | Accuracy: 0.60 | Diff: 0.00010
```

Training details when debug is True.

Step 1

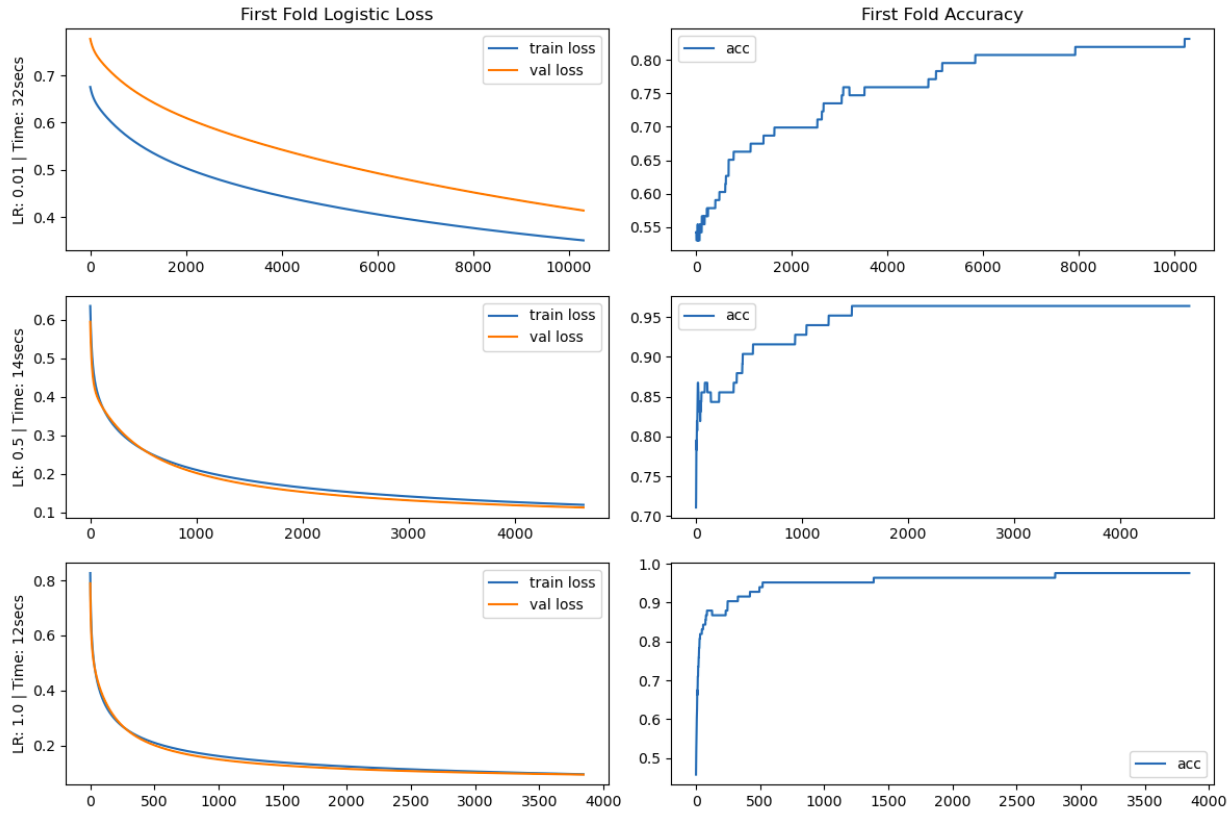

Figure1 Loss and Accuracy values of the first fold when full gradient batch is used.

Increasing the learning rate makes convergence faster in terms of both time and number of iterations. The model did not diverge when LR is 1.0 since we normalize the data.

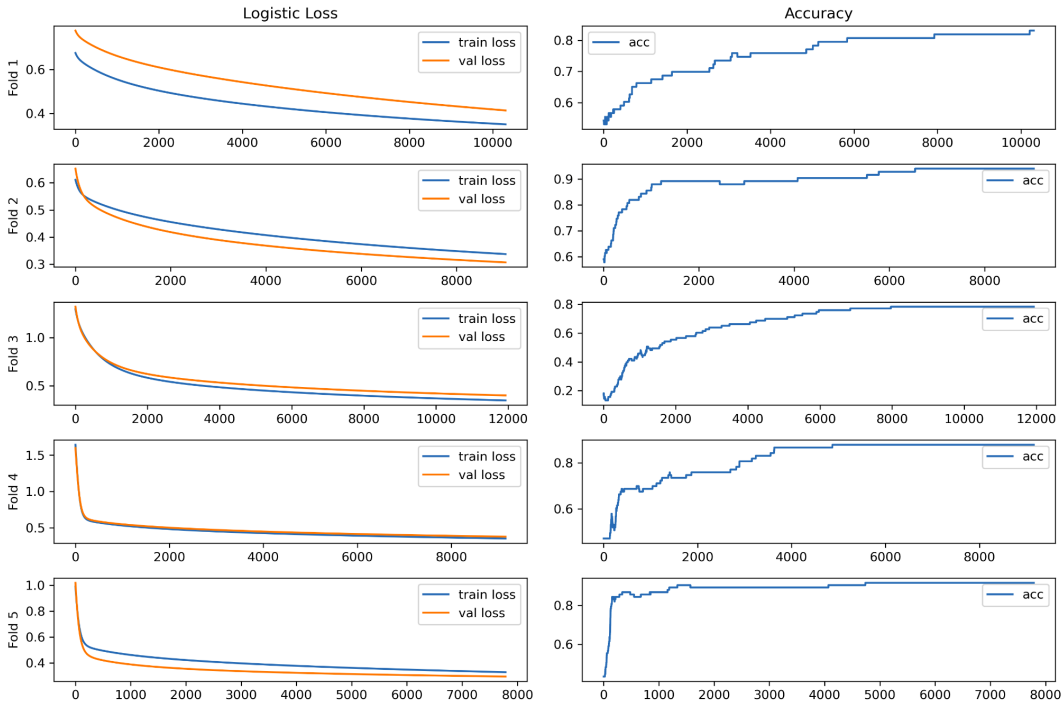

Figure2 Full Gradient Batch & LR:0.01

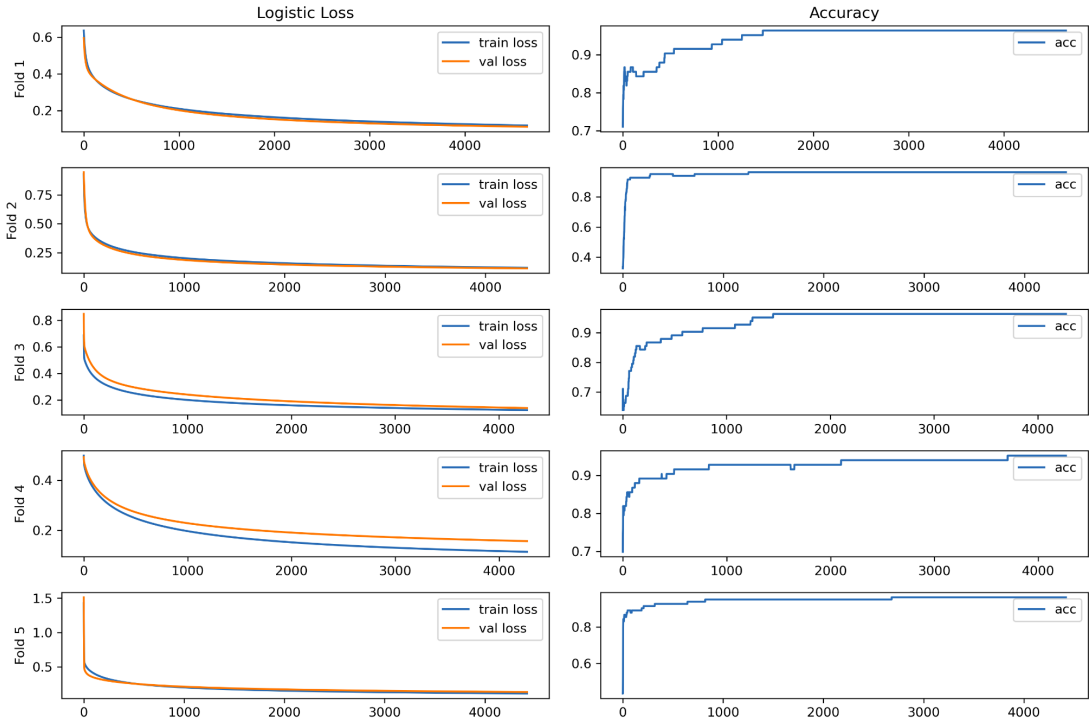

Figure3 Full Gradient Batch & LR:0.5

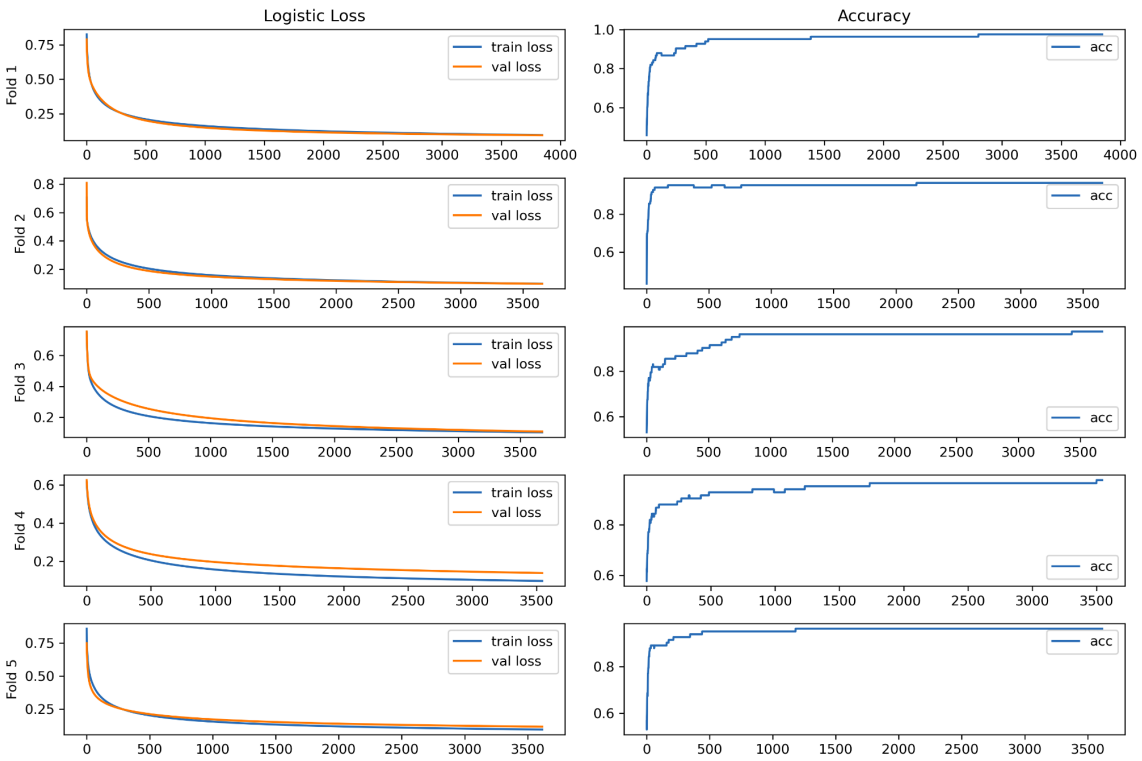

Figure4 Full Gradient Batch & LR:1.0

By default only the **Figure 1** is created after the run. In order to obtain **Figure2&3&4** please make the debug variable **True** from the code.

The acceleration effect of the larger learning rate can also be seen from the **Figure2&3&4.**

Step 2

For the step2 I've used **64 as the mini batch size**.

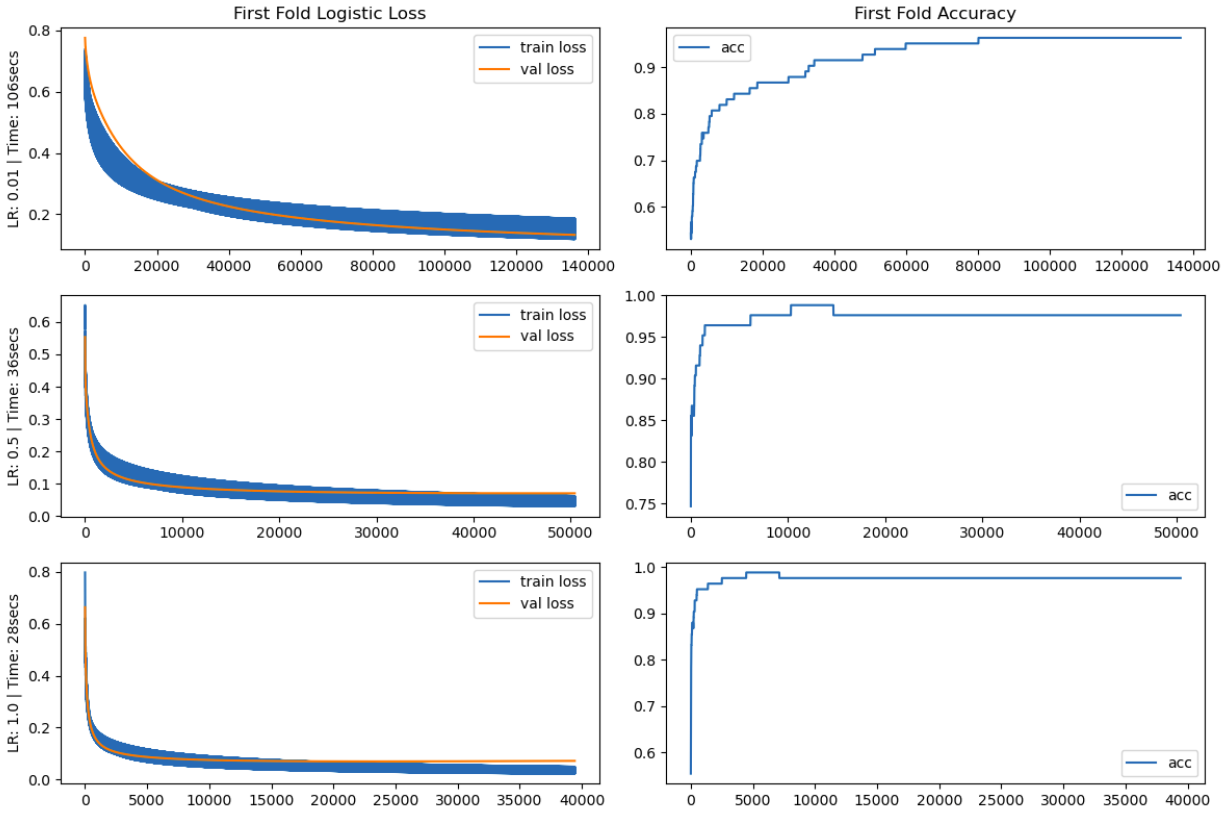

Figure 5 The logistic loss and the accuracy graphs when the Mini Batch Gradient is used.

The training loss value so fluctuates that its graph looks like a region. This fluctuation makes the convergence harder but it reaches a very high accuracy and a very low loss value immediately. However, for a long time, it fluctuates around the optimal weights. Thus, the convergence criteria is met later.

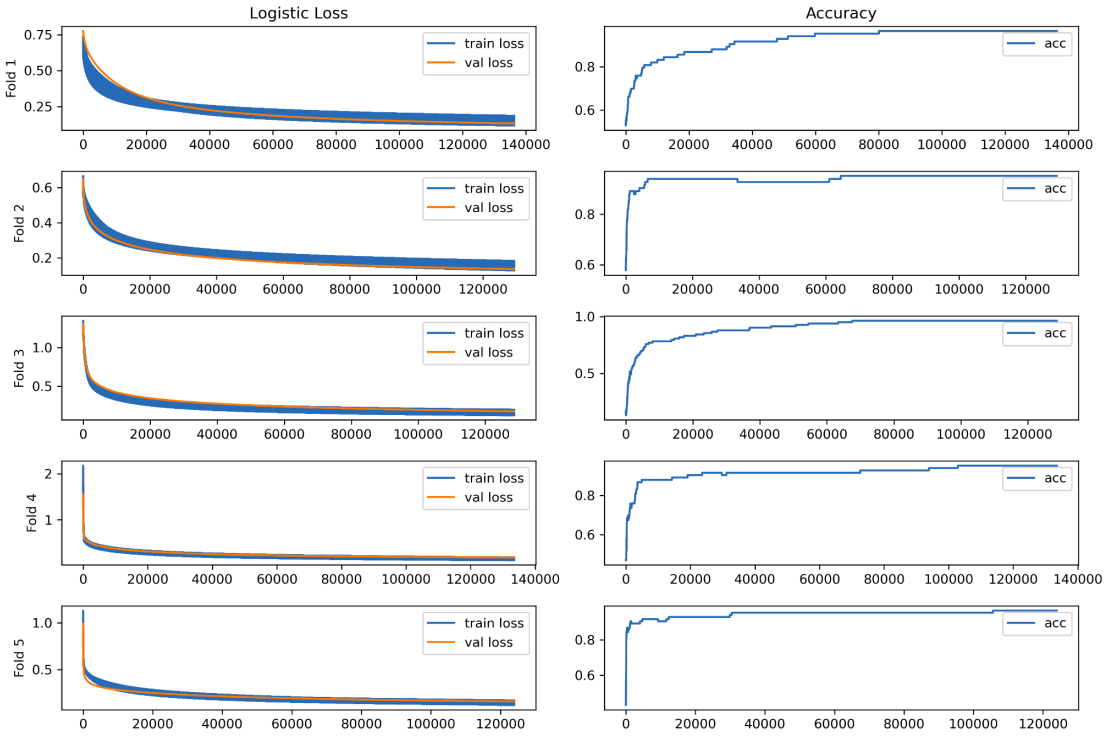

Figure 6 Mini Batch Gradient Descent & LR:0.01

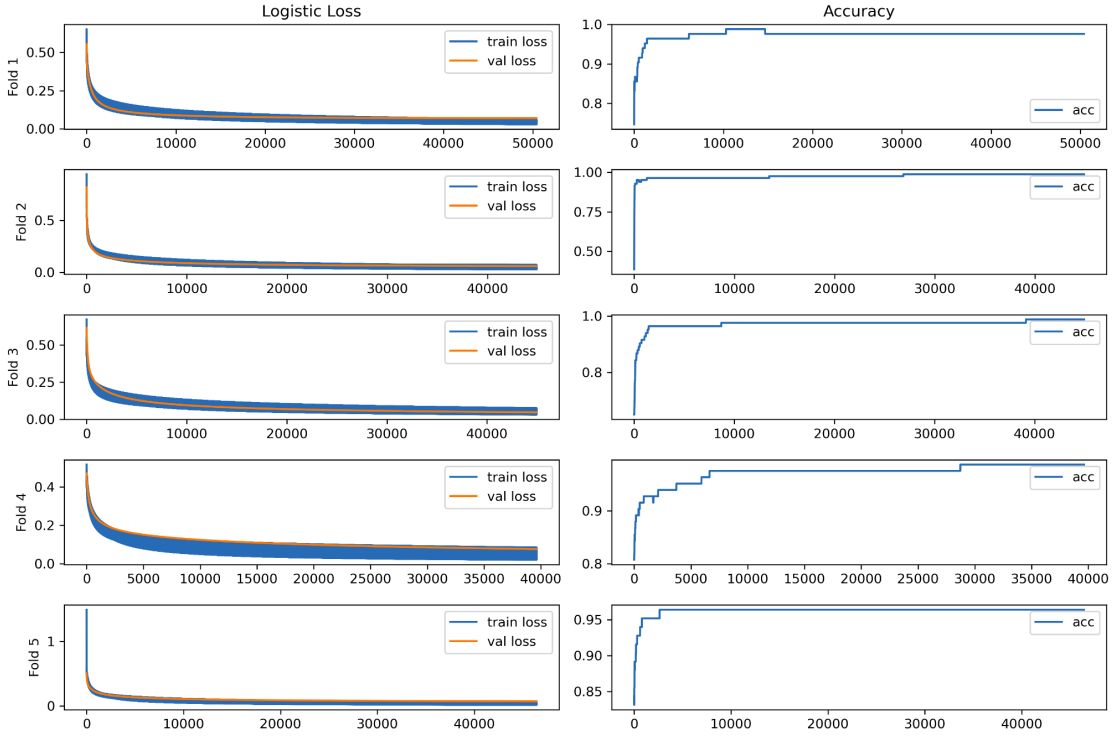

Figure 7 Mini Batch Gradient Descent & LR:0.5

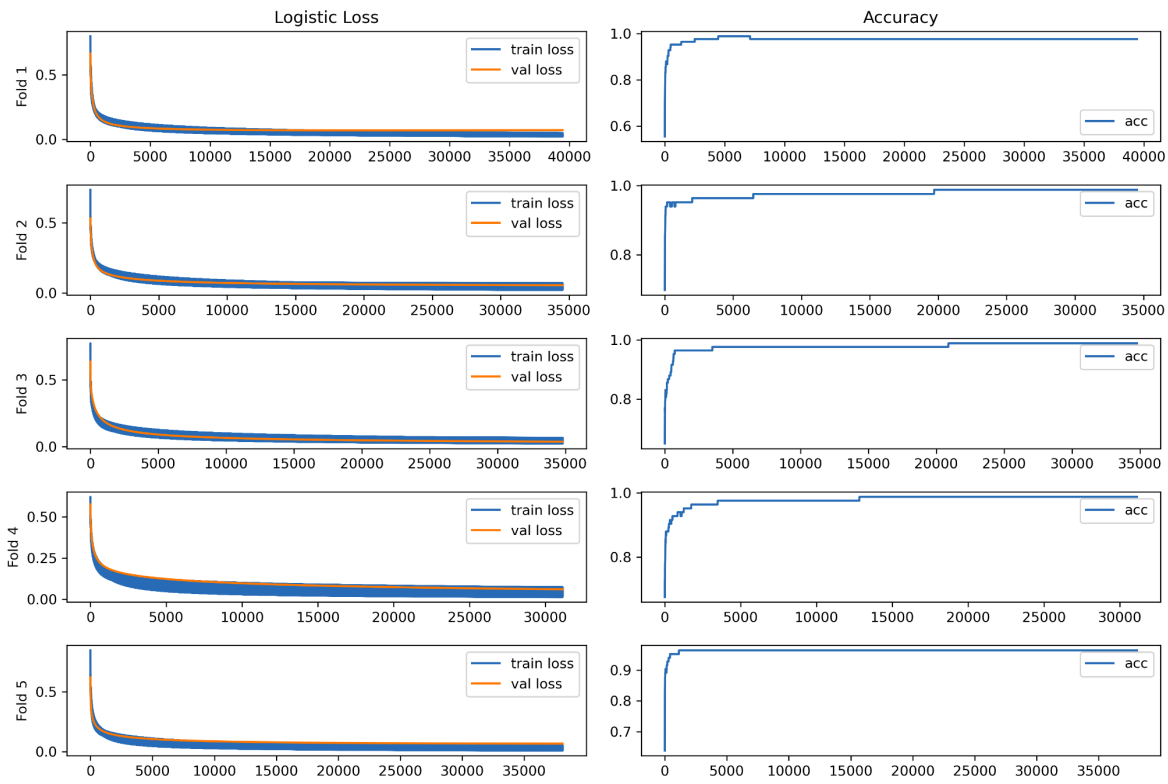

Figure 8 Mini Batch Gradient Descent & LR:1.0

By default only the **Figure 5** is created after the run. In order to obtain **Figure6&7&8** please make the debug variable **True** from the code.

Part 2

In naive bayes, we try to estimate and compare the probability of a sample X belonging to a class y_k. $P(Y = y_k|X))$ In order to obtain the probability of the sample X belongs to each class, we use the statistics we collected from the training set. Our prediction y^* for a sample X can be formulated like this:

$$y^* = argmax_{y_k} p(Y = y_k|X)$$

If we apply the Bayes Rule we obtain:

$$y^* = argmax_{y_k} \frac{p(X|Y = y_k)p(Y = y_k)}{p(X)}$$

If we treat every feature as independent and identically distributed, we can rewrite the formula like this:

$$y^* = argmax_{y_k} \prod_j p_j(x^j|Y = y_k)p(Y = y_k)$$

This assumption makes out Bayes classifier ‘Naive’.

For the problem, the probability of being ‘mammal’ of our test sample is:

$$p(Class = mammals|X = test) = p(GiveBirth = yes|Class = mammals)* p(CanFly = no|Class = mammals)* p(LiveInWater = yes|Class = mammals)* p(HaveLegs = no|Class = mammals)* p(Class = mammals)$$

$$p(Class = mammals|X = test) = \frac{6}{7} * \frac{6}{7} * \frac{2}{7} * \frac{2}{7} * \frac{7}{20}$$
$$p(Class = mammals|X = test) = \frac{1008}{48020}$$

The probability of being 'non-mammal' is this:

$$p(Class = non - mammals|X = test) = p(GiveBirth = yes|Class = non - mammals)* p(CanFly = no|Class = non - mammals)* p(LiveInWater = yes|Class = non - mammals)* p(HaveLegs = no|Class = non - mammals)* p(Class = non - mammals)$$

$$p(Class = non - mammals|X = test) = \frac{1}{13} * \frac{10}{13} * \frac{3}{13} * \frac{4}{13} * \frac{13}{20}$$
$$p(Class = non - mammals|X = test) = \frac{1560}{571220}$$

Since $P(Class = mammals|X = test)$ is greater than $P(Class = non - mammals|X = test)$, we make our prediction as the test sample X belong to the class 'mammal'.