HOMEWORK 1

TEAM: ARATRIK CHANDRA 23M0786
SAYANTAN BISWAS 23M0806

TITLE : ANALYZING THE EFFECTS OF HYPERPARAMETERS ON LOSS AND ACCURACY FOR BINARY AND MULTICLASS CLASSIFICATION MODELS.

SECTION 1: BINARY CLASSIFICATION(LOGISTIC REGRESSION)

Q1. Mathematical derivation of the gradient term for logistic regression which you must have used while implementation of logistic regression.

Loss function for logistic regression is:

$$L(P_W(x^{(i)}), y^{(i)}) = \begin{cases} -\log(P_W(x^{(i)})) & ; \text{ if } y^{(i)}=1 \\ -\log(1-P_W(x^{(i)})) & ; \text{ if } y^{(i)}=0 \end{cases}$$

combining these two equations we get

$$L(P_W(x^{(i)}), y^{(i)}) = -y^{(i)} * \log(P_W(x^{(i)})) - (1-y^{(i)}) * \log(1-P_W(x^{(i)}))$$

The cost function is:

$$J(W) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)} * \log(P_W(x^{(i)})) + (1-y^{(i)}) * \log(1-P_W(x^{(i)}))\right]$$

Now, ~~Gradient~~ ~~of~~ Gradient of $J(W)$ is : $\begin{bmatrix} \frac{\partial J(W)}{\partial W_1} \\ \frac{\partial J(W)}{\partial W_2} \\ \vdots \\ \frac{\partial J(W)}{\partial W_n} \end{bmatrix}$

Predicted value of $y^{(i)} = P_W(x^{(i)}) = \sigma(z)$

where $\sigma(z) = \frac{1}{1+e^{-z}}$ and $z = W^T x$

now we will compute derivative of $\sigma(z)$ w.r.t z as we will need it in future.

$$\frac{d\sigma(z)}{dz} = \frac{0.(1+e^{-z}) - (1).(e^{-z} * (-1))}{(1+e^{-z})^2}$$

or, $$\frac{d\sigma(z)}{dz} = \frac{e^{-z}}{(1+e^{-z})^2} = \frac{1-1+(e^{-z})}{(1+e^{-z})^2} = \frac{1+e^{-z}}{(1+e^{-z})^2} - \frac{1}{(1+e^{-z})^2}$$

or, $$\frac{d\sigma(z)}{dz} = \frac{1}{1+e^{-z}} * \left(1 - \frac{1}{1+e^{-z}}\right) = \sigma(z).(1-\sigma(z))$$

Now, we compute partial derivatives (Applying chain Rules) for $j = 1$ to n,

$$\frac{\partial(J(w))}{\partial w_j} = -\frac{1}{m} * \left(\sum_{i=1}^{m} \left[y^{(i)} * \frac{1}{P_w(x^{(i)})} * \sigma(z) * (1-\sigma(z)) * \frac{\partial(w^T x)}{\partial w_j} \right] + \sum_{i=1}^{m} \left[(1-y^{(i)}) * \frac{1}{1-P_w(x^{(i)})} * (-\sigma(z)(1-\sigma(z))) * \frac{\partial(w^T x)}{\partial w_j} \right] \right)$$

$$\frac{\partial(J(w))}{\partial w_j} = -\frac{1}{m} * \left(\sum_{i=1}^{m} \left[y^{(i)} \frac{1}{P_w(x^{(i)})} P_w(x^{(i)}) (1-P_w(x^{(i)})) * x_j^{(i)} \right] + \sum_{i=1}^{m} \left[(1-y^{(i)}) * \frac{1}{1-P_w(x^{(i)})} * (-P_w(x^{(i)}))(1-P_w(x^{(i)})) * x_j^{(i)} \right] \right)$$

$$= -\frac{1}{m} \left(\sum_{i=1}^{m} \left[y^{(i)} * (1-P_w(x^{(i)})) * x_j^{(i)} - (1-y^{(i)}) * P_w(x^{(i)}) x_j^{(i)} \right] \right)$$

$$= -\frac{1}{m} * \left(\sum_{i=1}^{m} \left[y^{(i)} - y^{(i)} * P_w(x^{(i)}) - P_w(x^{(i)}) + y^{(i)} * P_w(x^{(i)}) \right] * x_j^{(i)} \right)$$

$$= -\frac{1}{m} * \left(\sum_{i=1}^{m} \left[y^{(i)} - P_w(x^{(i)}) \right] * x_j^{(i)} \right)$$

Now, if write it in Matrix form then we will (for the gradient with respect to all the weights including bias term) get,

$$\frac{\partial J(w)}{\partial w} = \frac{1}{m} x^T \left[P_w(x) - y \right]$$

RESULTS:

LEARNING RATE: The Learning Rate of the Gradient Descent determines the rate of learning by limiting the amount of change of weights on each parameter such that the loss function decreases neither slowly or rapidly avoiding Exploding Gradient and Vanishing Gradient Problem. The Learning Rates we have used in out experiments are 10^{-1}, 10^{-2}, 10^{-4} and 10^{-6}.

Sl. No.	Learning Rate	Training Loss	Validation Loss	Training Accuracy	Validation Accuracy
1.	10^{-1}	1.44	1.32	0.49	0.53
2.	10^{-2}	**0.68**	**0.68**	**0.63**	**0.57**
3.	10^{-4}	0.69	0.69	0.51	0.47
4.	10^{-6}	0.69	0.69	0.51	0.47

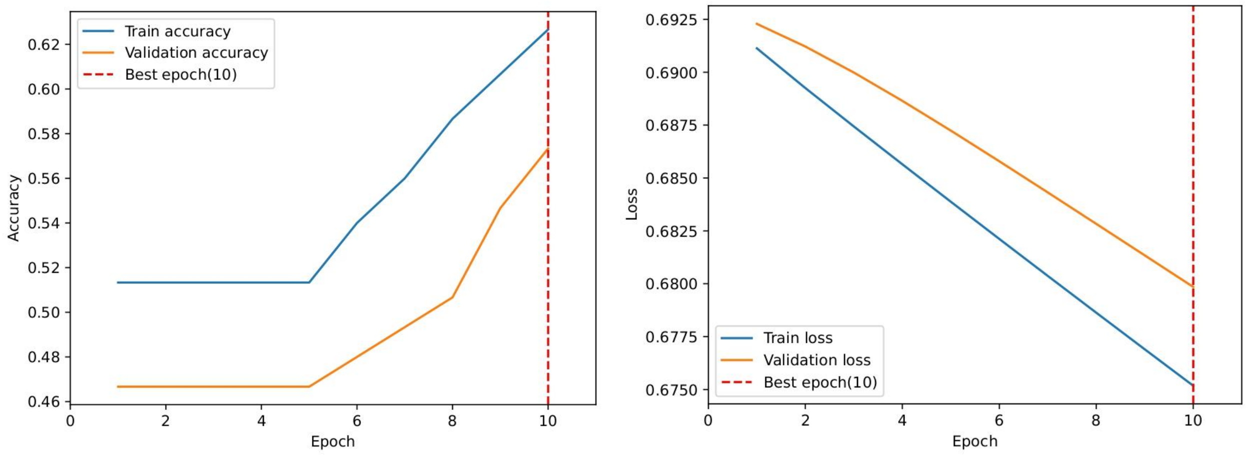


Fig: Accuracy and Loss wrt Training and Validation for Learning Rate=10^{-2}

With low learning rates we can see the improvements is linear.But with very low learning rate Gradient Descent converges very slowly.With high learning rates it starts to look more exponential.If we use Higher learning rates then loss will decay much faster, but sometime it may overshoot the global minima as happens with the learning rate 0.1, we have higher loss and lower accuracy compared to learning rate 0.01.So we choose **0.01** as the optimal Learning rate.

NUMBER OF EPOCHS: The number of Epochs determines the number of iterations in Gradient Descent i.e. how many times the weights on the corresponding parameters are updated in order to find the best fit model. The number of epochs must not be too small to result in poor training of the model and also very little change in accuracy of a very large number of epochs after a satisfactory accuracy is achieved can be ignored. In this report, for the best learning rate found, we have experimented with 100, 250, 500 and 1000 epochs for both the models in order to get the best epoch.

Sl. No.	Best Learning Rate	Number of Epochs	Training Loss	Validation Loss	Training Accuracy	Validation Accuracy
5.	10^{-2}	100	0.55	0.57	0.95	0.84
6.		250	0.43	0.47	0.96	0.85
7.		**500**	**0.33**	**0.40**	**0.97**	**0.87**
8.		1000	0.25	0.35	0.97	0.87

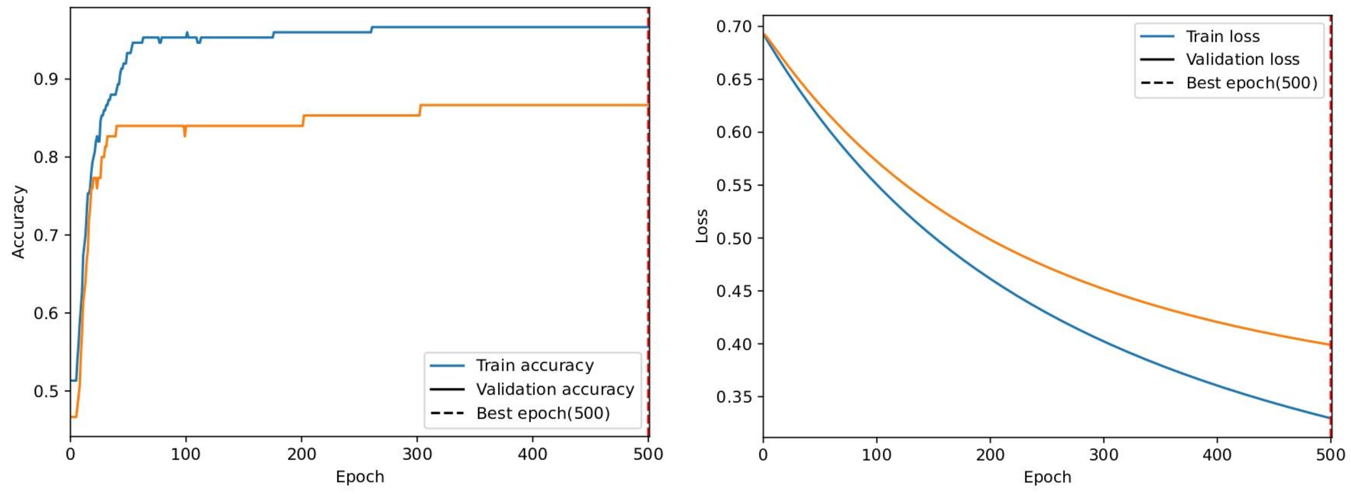
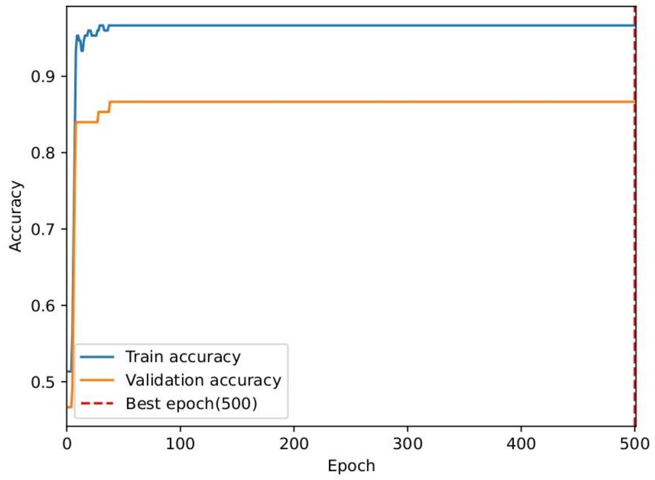


Fig: Accuracy and Loss wrt Training and Validation for Learning Rate=10^{-2} and Epochs = 500

As we increase number of epochs, the model gets chance to be trained for a longer period of time and has more opportunities to learn patterns from the training data. This result in an increase in validation accuracy and a decrease in validation loss. However,if we train the model for too many epochs, it may start to overfit to the training data, which can result in a decrease in validation accuracy and an increase in validation loss.From the graphs it is clear that after 300 epochs the Validation accuracy remains almost constant though loss decreases. So we choose **500** as the optimal number of epoch.

MOMENTUM: Momentum is the hyperparameter used in iterative algorithms that helps in accelerating the convergence of the optimization process. In this report, for the best learning rate and number of epochs found, we have experimented with momentum 0 and 0.9 to obtain the hyperparameters best found for the model.

Sl. No.	Best Learning Rate	Best Epoch	Momentum	Training Loss	Validation Loss	Training Accuracy	Validation Accuracy
9.	10^{-2}	**500**	0	0.33	0.40	0.97	0.87
10.			**0.9**	**0.14**	**0.37**	**0.97**	**0.87**



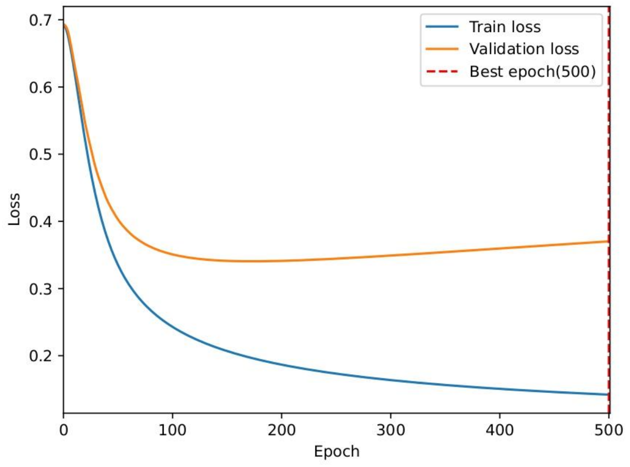


Fig: Accuracy and Loss wrt Training and Validation for Learning Rate=10^{-2}, Epochs = 500 and momentum = 0.9

By using momentum, the optimization algorithm moves faster towards the direction of the global minima of the loss function, which results in a decrease in validation loss though the accuracy is same incase of momentum=0.9 compared to without momentum**.**so we choose **0.9** as momentum value.

METHODOLOGY:
We implemented Logistic Regression on the binary dataset from scratch without using any builtin packages. This report is mainly based on Logistic Regression using Gradient Descent.
We have initialized all the weights and bias to zero before starting the training and calculated the gradient of the log loss function in order to update the weights at every epoch. For logistic regression, sigmoid function is used and for every predicted output lesser 0.5 is assumed to be in class 0 and the rest in class 1. More about this method of obtaining the gradient of the log loss function is mentioned at the end of this report.

CONCLUSION:

The main objective of this report is to bring forward how tuning the hyperparameters results in the change in loss and accuracy for a Binary classifier using Logistic Regression. Along with this, the aim is to find the best fit hyperparameters for this model on the given dataset so that the model gives the best prediction on the validation dataset.
On performing this experiment, we have obtained that the Binary Classification Model using Logistic Regression on the binary dataset is performing most efficiently for **Learning Rate=** 10^{-2}**, Number of Epochs= 500** and **Momentum= 0.9.**
It gives astonishing results of a **Validation Accuracy = 87%** and **Validation Loss = 0.37.**

SECTION 2: LINEAR CLASSIFIER(LOGISTIC REGRESSION)

Q2. Study on iris. How to adopt logistic regression to multi-class setting.

Logistic Regression is a very popular classification Algorithm.

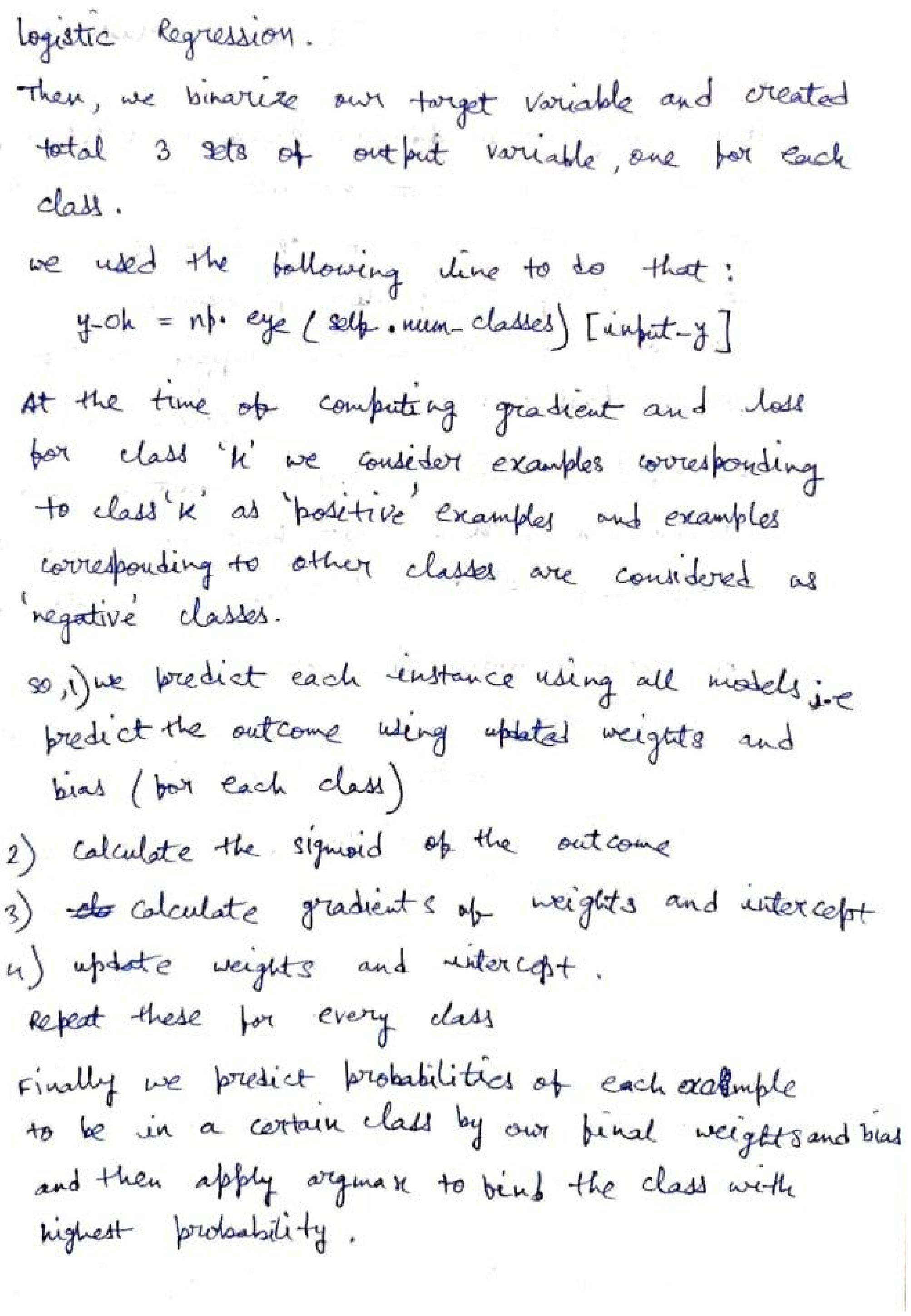
It is mainly used in datasets having numerical input variables and a target variable that has two possible values or classes. This type of problems are referred to as binary classification problems.

Generally, class labels are mapped to 1 for the positive class and 0 for the negative class.

By defaut, logistic Regression cannot be used for classification problems that have more than two class labels (called as Multiclass classification problem)

one popular modification for adapting logistic regression to a multiclass classification problem is to split the multiclass classification problem into multiple binary classification problems and fit a standard logistic regression model on each of the subproblems.

In our 'iris' dataset for the target variable we have total three classes namely 0, 1, 2. Here we will build our multiclass linear classifier with 3 classes as 3 logistic regression classifiers with "1 vs all" strategy. For this, we created 3 sets of weights each corresponding to one



logistic Regression.

Then, we binarize our target variable and created total 3 sets of output variable, one for each class.

we used the following line to do that:

```
y_oh = np.eye(self.num_classes)[input_y]
```

At the time of computing gradient and loss for class 'k' we consider examples corresponding to class 'k' as 'positive' examples and examples corresponding to other classes are considered as 'negative' classes.

so, 1) we predict each instance using all models i.e predict the outcome using updated weights and bias (for each class)

2) calculate the sigmoid of the outcome

3) ~~cl~~ calculate gradients of weights and intercept

4) update weights and intercept.

Repeat these for every class

Finally we predict probabilities of each example to be in a certain class by our final weights and bias and then apply argmax to find the class with highest probability.

RESULTS:

LEARNING RATE: The Learning Rate of the Gradient Descent determines the rate of learning by limiting the amount of change of weights on each parameter such that the loss function decreases neither slowly or rapidly avoiding Exploding Gradient and Vanishing

Gradient Problem. The Learning Rates we have used in out experiments are 10^{-1} , 10^{-2}, 10^{-4} and 10^{-6}.

Sl. No.	Learning Rate	Training Loss	Validation Loss	Training Accuracy	Validation Accuracy
1.	10^{-1}	**1.45**	**1.43**	**0.76**	**0.80**
2.	10^{-2}	1.84	1.83	0.32	0.40
3.	10^{-4}	2.07	2.07	0.31	0.40
4.	10^{-6}	2.08	2.08	0.40	0.31

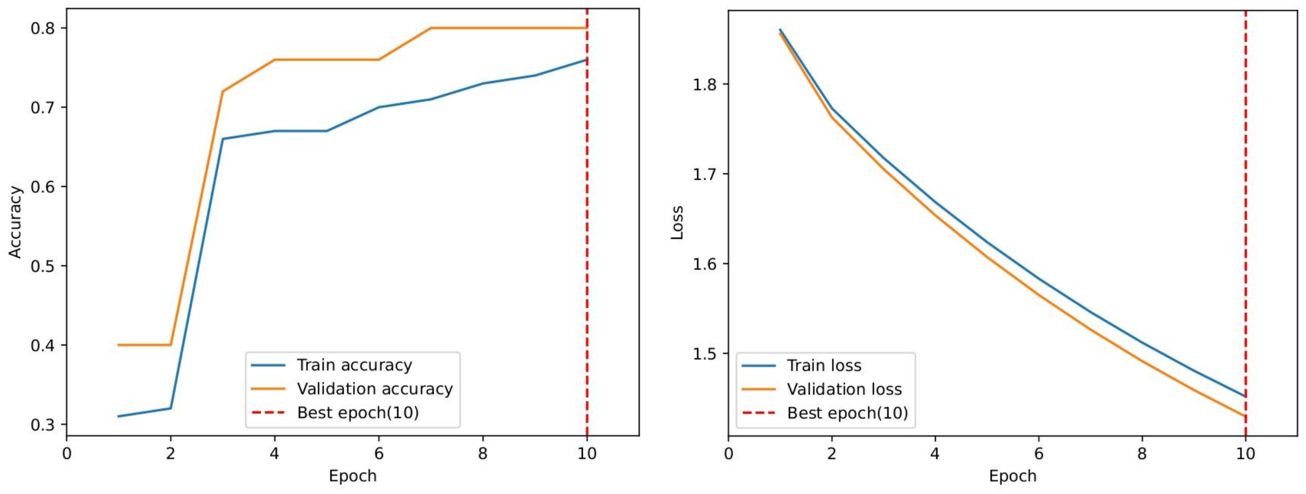


Fig: Accuracy and Loss wrt Training and Validation for Learning Rate=10^{-1}

With low learning rates we can see the improvements is linear.But with very low learning rate Gradient Descent converges very slowly.With high learning rates it starts to look more exponential.In our model with the learning rate 0.1, we have much lower validation loss and much higher accuracy compared to learning rate 0.01 and less than that.So we choose **0.1** as the optimal Learning rate.

NUMBER OF EPOCHS: The number of Epochs determines the number of iterations in Gradient Descent i.e. how many times the weights on the corresponding parameters are updated in order to find the best fit model. The number of epochs must not be too small to result in poor training of the model and also very little change in accuracy of a very large number of epochs after a satisfactory accuracy is achieved can be ignored. In this report, for the best learning rate found, we have experimented with 100, 250, 500 and 1000 epochs for both the models in order to get the best epoch.

Sl. No.	Best Learning Rate	Number of Epochs	Training Loss	Validation Loss	Training Accuracy	Validation Accuracy
5.	10^{-1}	100	0.93	0.91	0.96	0.96
6.		**250**	**0.80**	**0.77**	**0.97**	**1.00**
7.		500	0.71	0.69	0.96	1.00
8.		1000	0.65	0.62	0.96	1.00

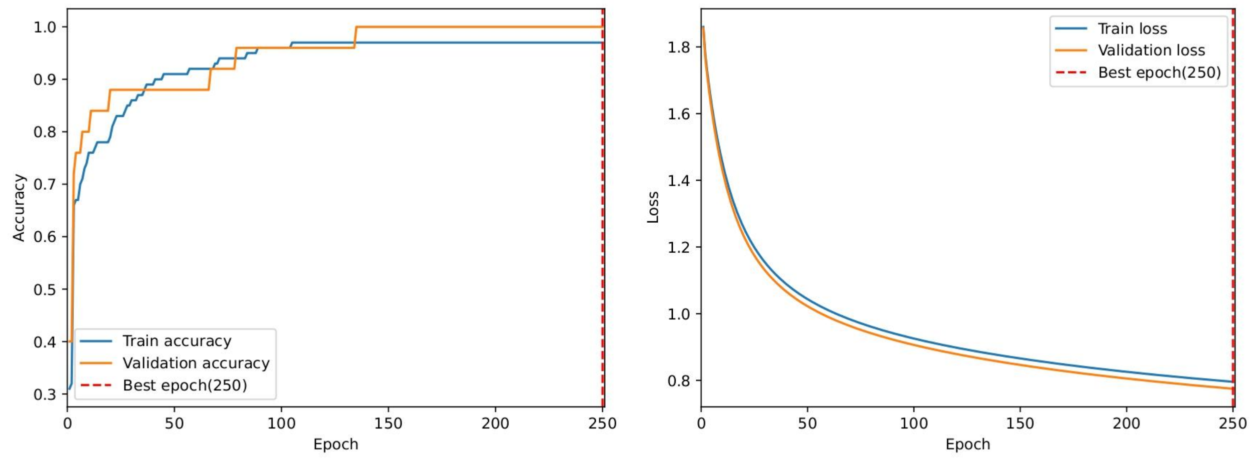
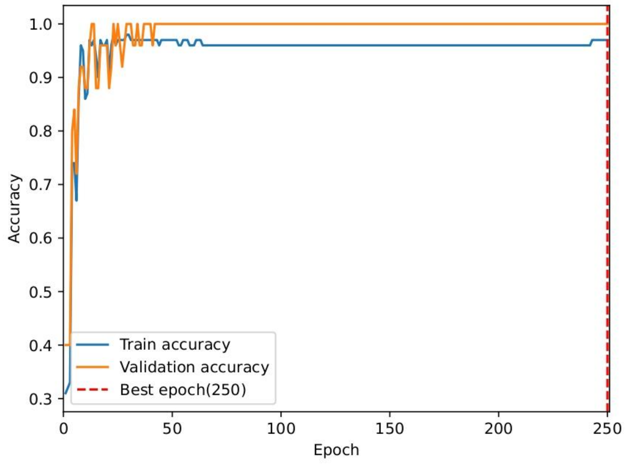


Fig: Accuracy and Loss wrt Training and Validation for Learning Rate=10^{-1} and Epochs = 250

As we increase number of epochs, the model gets chance to be trained for a longer period of time and has more opportunities to learn patterns from the training data. This result in an increase in validation accuracy and a decrease in validation loss.We can see after 250 iterations validation accuracy reaches 100%.After that for 500 and 1000 iterations validation loss decreases but not that significantly compared to the increase in number of iterations.So we take **250** as the optimal number of epoch.

MOMENTUM: Momentum is the hyperparameter used in iterative algorithms that helps in accelerating the convergence of the optimization process. In this report, for the best learning rate and number of epochs found, we have experimented with momentum 0 and 0.9 to obtain the hyperparameters best found for the model.

Sl. No.	Best Learning Rate	Best Epoch	Momentum	Training Loss	Validation Loss	Training Accuracy	Validation Accuracy
9.	10^{-1}	250	0	0.80	0.77	0.97	1.00
10.			**0.9**	**0.59**	**0.56**	**0.97**	**1.00**



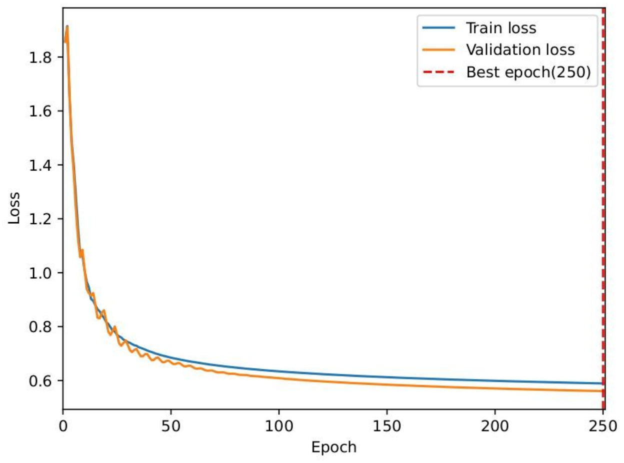


Fig: Accuracy and Loss wrt Training and Validation for Learning Rate=10^{-1}, Epochs = 250 and Momentum=0.9

By using momentum, the optimization algorithm moves faster towards the direction of the global minima of the loss function, which results in a large decrease(from 0.77 to 0.56) in validation loss though the accuracy is same incase of momentum=0.9 compared to without momentum.so we choose **0.9** as momentum value.

METHODOLOGY:

This portion of the methodology behind implementing Multiclass Classification using Logistic Regression is mentioned at the beginning of as the answer of Q2.

CONCLUSION:

The main objective of this report is to bring forward how tuning the hyperparameters results in the change in loss and accuracy for a Binary classifier using Logistic Regression. Along with this, the aim is to find the best fit hyperparameters for this model on the given dataset so that the model gives the best prediction on the validation dataset.
On performing this experiment, we have obtained that the Multiclass Classification Model using Logistic Regression on the iris dataset is performing most efficiently for **Learning Rate=** 10^{-1}**, Number of Epochs= 250** and **Momentum= 0.9.**
It gives astonishing results of a **Validation Accuracy = 100%** and **Validation Loss = 0.56.**

In machine learning, achieving 100% validation accuracy is usually not a very good thing to have.It could indicate that there may be some issues with the model or the data, such as data leakage or an imbalanced validation set.
However, in this case, the problem is very simple and the number of training example(100) is very very small compared to machine learning standards.
Also it can happen when the training data is more difficult to classify than the validation data or it may be the case that the validation data is closely related to the training data.

So in such cases the model is able to learn the relationship and pattern between the input features and the target variable very well.In this way a model may achieve 100% validation accuracy.