1. a. The *train_valid* function splits the dataset into a training set and a validation set. Even though the performance of a model is always assessed by its performance of the final test set (and neither the training nor the validation sets), having a validation set is recommended to tune the different hyperparameters in the model and to avoid overfitting.

i. The performance of different models with varying hyperparameter combinations is assessed on the validation set. And the combination which gives the best performance, is finally selected.

ii. And, overfitting is avoided by using the validation set and not the training set to evaluate the performance of different hyperparameter combinations. This is so because the performance of the models is assessed over a data set other than the training set.

b. Before testing, it is correct to re-train the model on the entire training set (training + validation). It is because by doing so:

i. A larger sample size is available to train the model with the best hyperparameter combination, than when only the training set (after the *train_valid* split) is used for training.

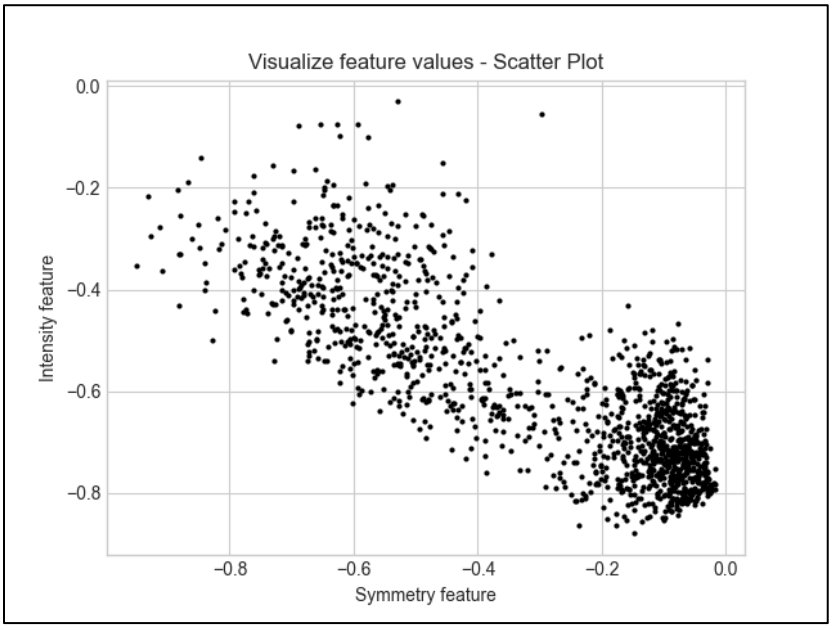
ii. And, the all the samples which were included in determining the best hyperparameter combination are included in final training.

c. Please find it in the submitted code.

d. The third feature is the bias we introduce in the model. It is the log odds when all the dependent variables equate to 0. If this is not included in the system, when the input space is 0, it would indicate that the probability of belonging to either class (in binary classification) would be same (0.5), which can never be the case.

e. Please find it in the submitted code.

f. **Fig:** Visualization of features.



2. a. Loss function $E(w)$ for one training sample (x, y):

$$E(w) = \ln(1 + e^{-y.w^T x})$$

b. Gradient $\nabla E(w)$:

$$\nabla E(w) = \frac{\partial E}{\partial w} = \frac{1}{(1 + e^{-y.w^T x})} (0 + e^{-y.w^T x}.(-y).x$$

$$or, \quad \nabla E(w) = \frac{e^{-y.w^T x}(-y).x}{(1 + e^{-y.w^T x})}$$

Since, y in constant:
Therefore,

$$\nabla E(w) = \frac{(-y)e^{-y.w^T x}.x}{(1 + e^{-y.w^T x})}$$

For optimal w, we equate:

$$\nabla E(w) = 0$$

Since, *y* and *x* cannot be 0
Therefore,

$$\frac{-e^{-y.w^T x}}{(1 + e^{-y.w^T x})} = 0$$

$$or, \quad \frac{-1}{(e^{y.w^T x} + 1)} = 0$$

$$or, \quad \frac{-1}{(e^{y.w^T x} + 1)} = 0$$

$$or, \quad e^{y.w^T x} = -\infty$$

$$or, \quad y.w^T x = 0$$

Since, y cannot be 0
Therefore,

$$w^T x = 0$$

c. Having an optimal w for Logistic Regression problems would be efficient if the input feature space is only 2-dimensional. Having a 2-dimensional input space makes the equation ($w^T x = 0$) quadratic, for which we have a solution. However, if we have input feature space which is greater than 2-dimensional, we do not have a solution for the equation ($w^T x = 0$). Hence, solving based on a optimal w is not the most efficient way.

The sigmoid function is used in the prediction when we base the decision boundary on the probability. Depending upon the data set, the probability threshold for classifying the data samples, can be ascertained. And, in order to use this probability threshold for predicting, the sigmoid function is used, which converts the linear summation into a (0,1) range (signifying probability).

d. The sigmoid function is essentially a monotonically increasing function. This means that for each value of $\theta(z)$, we will have one and only one *z*. Hence, even if the threshold for $\theta(z)$ is

shifted from 0.5 to 0.9, we will have only one *z*. And having only one z means the decision boundary is still linear.

e. The essential property of Logistic Regression that results in a linear decision boundary is the monotonically increasing nature of the sigmoid function and that Logistic Regression doesn't have any hidden layers (unlike Neural Networks).

3. a. Please find it in the submitted code.

b. Please find it in the submitted code.

c. Please find it in the submitted code.

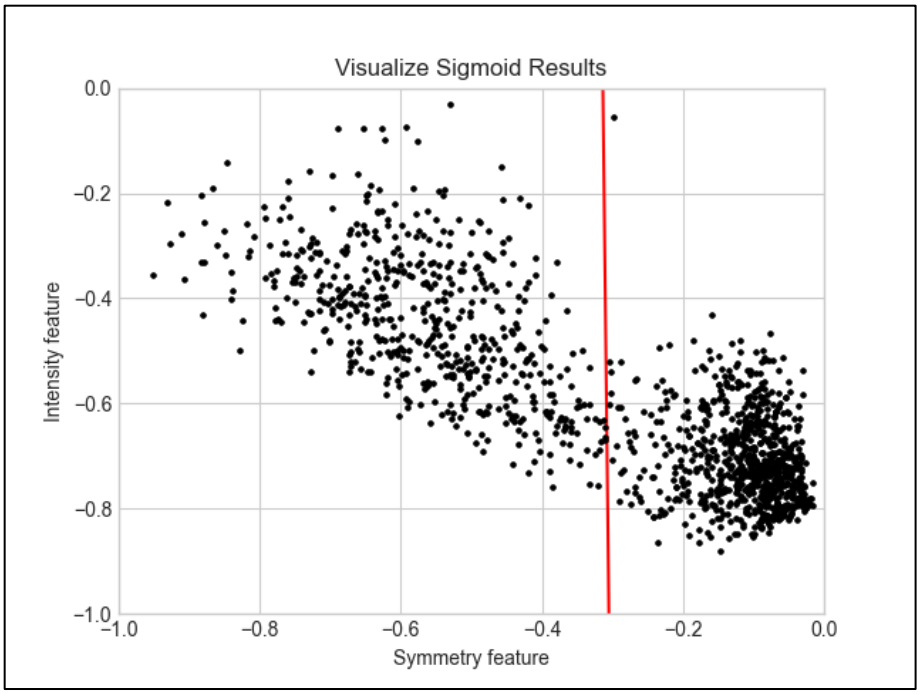
d. Best Hyperparameter combination for Binary Logistic Regression

i. learning_rate = 0.2

ii. max_iter = 500

iii. batch_size = 5

Fig: Visualization of Sigmoid Results



e. Test Accuracy = 93.5065%

4. a. Please find it in the submitted code.

b. Please find it in the submitted code.

c. Please find it in the submitted code.

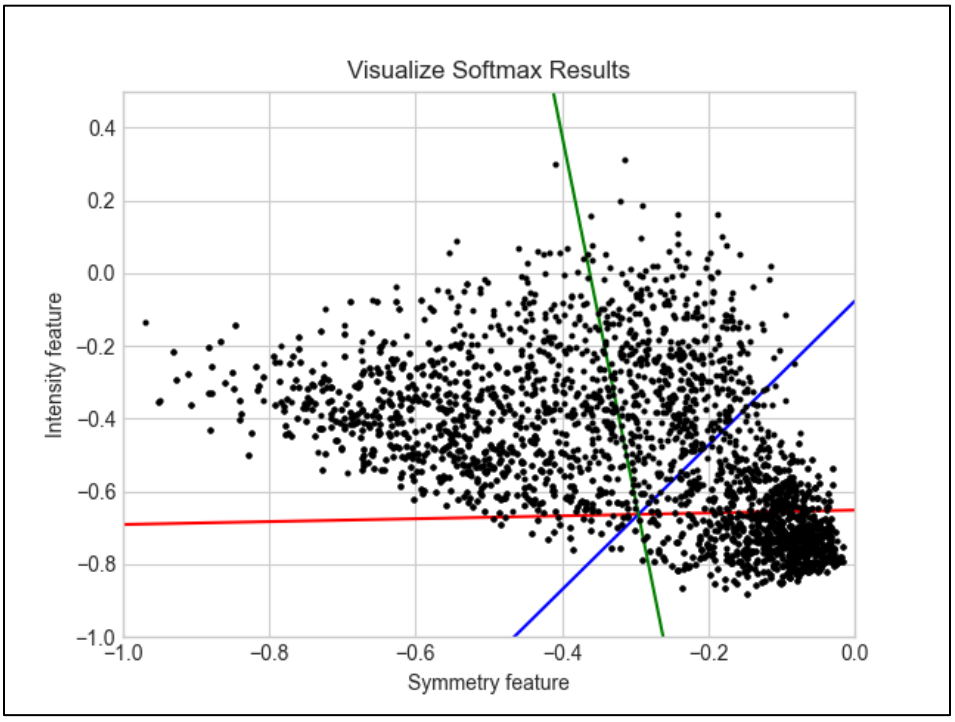
d. Best Hyperparameter combination for Multiple Logistic Regression

i. learning_rate = 0.2

ii. max_iter = 100

iii. batch_size = 20

Fig: Visualization of Softmax Results



e. Test Accuracy = 86.8453%

5. a. Accuracy when training using softmax = 94. 1558%

Accuracy when training using sigmoid = 94.1558%

Both the sigmoid and softmax approaches yield the same accuracy. This means that for a binary classification problem, sigmoid and softmax essentially perform the same calculations.

Insight: Sigmoid training takes slightly more time than softmax.