**Logistic Regression**

1.

2. To split the data into the training, validation and test sets we made use of the ‘’train\_test\_split’’ function and applied it twice to the dataset, we produced the training data and test data first then further split the original training data into another set of training data and validation data, in both cases our test size was 20% for the splits. We then proceeded to clean the input data of our headlines by making use of regular expressions to remove all punctuation and convert the headlines to lowercase. To normalize and create our bag of words we used the TF-IDF vectorizer with its analyzer set to ‘word’.

3. For Logistic Regression we created a class with all the required functions such as the add intercept function which set up our feature matrix by adding a column of 1’s to the front of the matrix, the sigmoid function which we made use of to generate our probabilities, the loss function which helps us find the best thetas for our model by measuring how well the algorithm performed with those thetas, a fit function to perform gradient descent and finally a predict and predict probability function which make use of the sigmoid function and an input threshold to classify our results.

When we fitted the data to the model we used a learning rate of 0.5 since lower learning rates yielded lower rates in accuracy, our number of iterations was set at 10000 since low iterations yielded lower rates in accuracy and a higher number of iterations became computationally expensive. Our intercept was set to True since we wanted the constant present in our feature matrix. We set our threshold (determines whether a probability of a predicted value belongs to class 0 or 1) to a value of 0.5 as this gave us the highest accuracy when we tested values such as 0.3, 0.4, 0.6, 0.7 (See image ThresholdChoice)

4. By looking at our unnormalized confusion matrix and accuracy we can see that we have a score of 0.817, this means that in our test data set, roughly 82% of the headlines were correctly predicted, which is quite a high score for our model. Our recall score was 0.796 which implies that approximately 80% our positive data was classified correctly, and our precision score was 0.788 implies that approximately 79% of predicted positives were actually positive. Therefore, a high recall and precision implies that the model will return a high percentage of results that are correctly classified. Our f1 score which is also known as our harmonic mean of the recall and precision score is 0.79, which implies that our model is a reasonable classifier. These scores reflect the best performance of our model for logistic regression which were achieved by testing and changing our hyperparameters. Although we recommend that stochastic gradient descent be used instead of our regular implementation of gradient descent as it will speed up computation and this would allow more iterations when the model is trained to improve the results of the classifier.