**Machine Learning Assignment 2019**

Group:

Simon Rosen- 1391995

Christopher Pillay-1362077

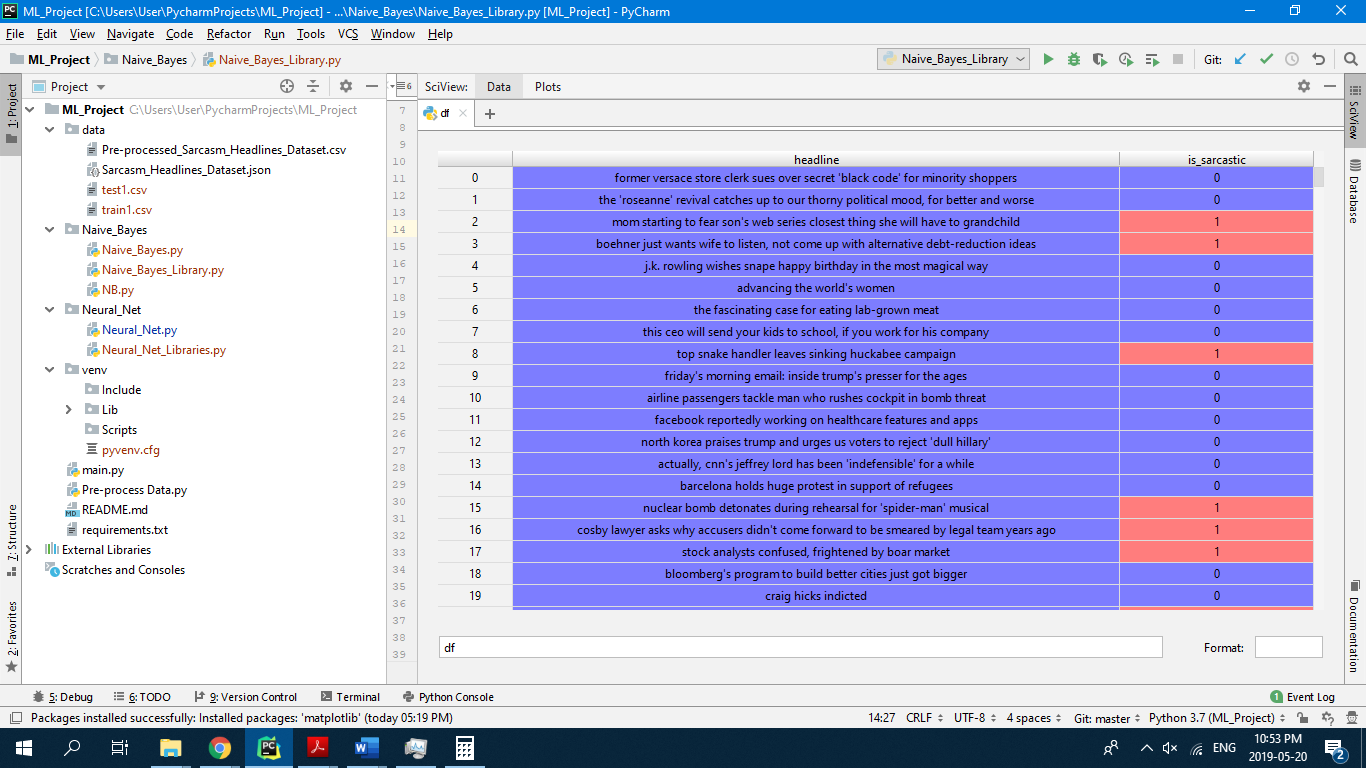
Robel Abraham-1348744

Kimay Ramnarain-1370763

**Question 1**

The dataset we choose is a dataset containing news headlines that are either sarcastic or not sarcastic. This dataset has two attributes, headline and article\_link but we decided to drop article\_link as all the sarcastic headlines were from “The Onion” website and all the non-sarcastic headlines were from the “Huffpost” website so using this feature would most definitely have skewed the accuracy of our models. The headline attribute will later be transformed into a list of features for each unique word present for each of the training sets used. The targets are either sarcastic (represented as 1) or non-sarcastic (represented as 0). There are a total of 26 709 datapoints in this dataset with 14 985 (56.1%) being non-sarcastic and 11 724 (43.9%) being sarcastic.

Some examples of the dataset are as follows:



We are trying to predict sarcasm in new headlines.

**Question 2**

**Logistic Regression**

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’.

**Naive Bayes**

In order to split the data into training data, testing data and validation data, the 'test\_train\_split' function was used to split the data into training and testing data(80/20 split, respectively). In order to pre-process the data regular expressions were used to remove punctuation from the headlines so that we could analyse the words accurately.

**Neural Network**

Exact same as for logistic regression with the exception that TF\_IDF vectorizer was not used as this appeared to decrease the accuracy in this case. We used CountVectorizer to change each headline from a String to a vector of integer values specifying the no of occurrences of each word in the bag of words, i.e. {0, 0, 2, 0, …}. Each element of these vectors was then used as the features for the model.

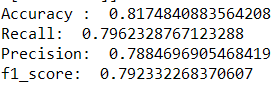
**Question 3**

**Logistic Regression**

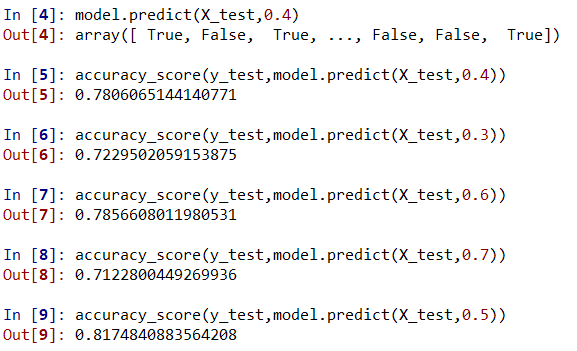
For Logistic Regression we created a class with all the required functions such as the add intercept function which set up our feature matrix by adding a column of 1’s to the front of the matrix, the sigmoid function which we made use of to generate our probabilities, the loss function which helps us 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)



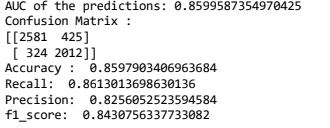
Threshold Choice:



**Naive Bayes**

For Naive Bayes we created several functions to help make predictions. The first being the get\_text function, which gathered all headlines with the same classifications (0 for not sarcastic and 1 for sarcastic headlines). We then implemented the count\_text function, which returns the counter for the amount of times a word occurs. The get\_y\_count function was implemented in order to obtain the number of items in each class(0 or 1). These functions were implemented in order to assist with making predictions and calculating probabilities, which are needed when implementing Naive Bayes.

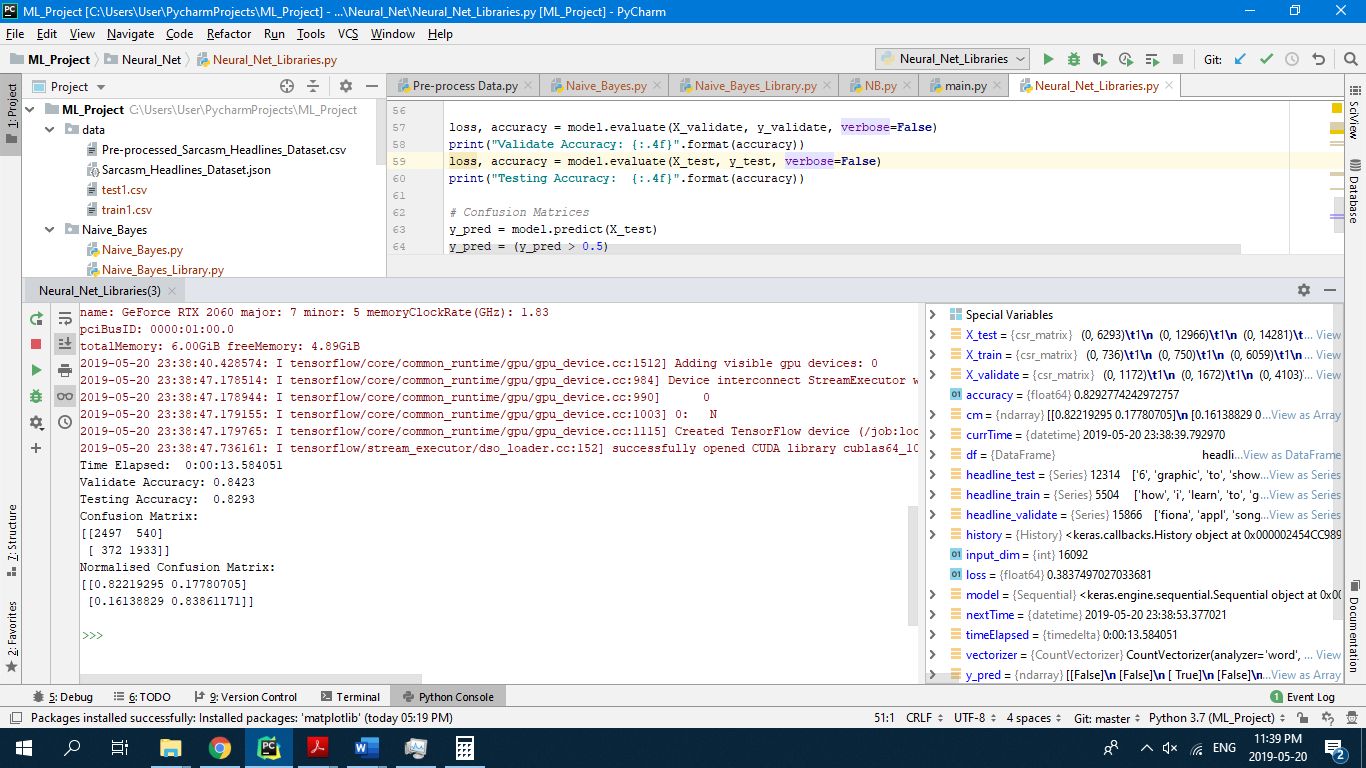
The main functions involved in classifying the headlines were the make\_decison and make\_class\_preditctions. The former of which was used to classify each word in each headline as either sarcastic or not sarcastic, we also applied smoothing in this function in order to multiplying/dividing by a zero when performing calculations. The latter of the two functions made the final decision on the whole headline based on which there were more sarcastic or non-sarcastic words in the headline. We set our alpha value for smoothing to 1 as this yielded the greatest accuracy, as seen in the results below:



**Neural Network**

We decided to use the module Keras (running off the GPU variant of TensorFlow) to create a neural network as this module allows for rather large neural networks with many features to be trained relatively quickly. We choose to create a neural network with two hidden layers, with all the layers except the final layer having ‘relu’ activation functions. The final layer used a sigmoid activation function as the output of this neural net is binary.

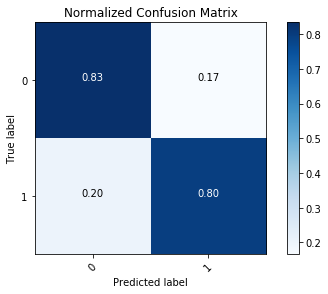
The 2 hidden layers had 20 neurons and 10 neurons respectively and this was chosen somewhat arbitrarily but by trial and error it appeared to give good accuracy. We only used 2 hidden layers as adding more layers did not seem to drastically increase the accuracy but merely made the algorithm more computationally expensive. The two hyperparameters used as arguments to fit this model were number of epochs and batch size with batch size being the number of training examples used in one iteration of the algorithm and the number of epochs being the number of iterations over the entire dataset. We selected a batch size of 32 as using a batch size of 1 i.e. using stochastic gradient descent would be too computationally expensive and using a batch size the size of the training set i.e. using batch gradient descent would have been too inaccurate, therefore a good compromise was to use batch gradient descent which meant using a batch size between 1 and the size of the training set. Through research we found that 64 was a popular batch size, hence we decided to use that value. We set the number of epochs to 1 as it appeared that the greater the number of epochs the lower the accuracy which is most likely due to overfitting. Also an added benefit of setting the number of epochs to 1 was that this was the least computationally expensive parameter that could be chosen.

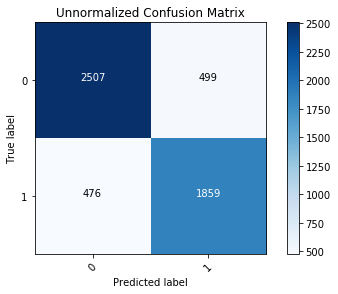


**Question 4**

**Logistic Regression**

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.





**Naive Bayes**

Our Naive Bayes algorithm yielded an accuracy of 0.859, the algorithm that we implemented predicted the headlines correctly 85% of the times. From our recall value of 0.86, we see that 86% of the headlines that were sarcastic were correctly classified as sarcastic. Our precision score was 0.82 which implies that 82% of all headlines that were predicted to be sarcastic, were in fact sarcastic. These high statistics tell us that a high percentage of the headlines were correctly classified. Our f1 score was 0.84 which implies that our Naive Bayes model is a good classifier. We recommend calculating probabilities that are needed for this function, such as standard deviation, mean, etc…for each class of words in separate functions as it allows calculations to be performed with a bit more ease and makes error tracing easier.

**Neural Network**

This could have been improved by increasing the number of epochs and implementing drop out to reduce the likelihood of overfitting which can be common when so many features are used.

**General**

Naïve Bayes worked the best of the three algorithms both in accuracy and in the time needed to train. The reason Naïve Bayes is so much faster is because with this algorithm there is no cost function being optimised and as such it does not require more than one iteration while logistic regression and neural networks require many iterations due to the fact that they need to optimize their cost functions which quickly becomes very computationally expensive the more features are present. The reason we feel Naïve Bayes was the most accurate of all our algorithms is because due to computational limitations we might not have run the logistic regression algorithm close enough to convergence as this would have taken too many iterations to be feasible and similarly for the neural network it might have been run on too few epochs to optimise the weights of the neural network to the required degree.

It is hard to tell what the best possible performance is that we can achieve on the dataset due to the complexity of human language but in terms of efficiency Naïve Bayes definitely performs the best due to the reasons stated above.