CSC411: Assignment 3

Due on Monday, March 19, 2018

Naireen Hussain, Najah Hassan

March 20, 2018

Problem 1

Analyzing the Dataset
The datasets all seem to be from news centered around politics. Nearly all of them refer to Trump.
Here is a table summarizing how often selected keywords show up in the real and fake news

Word	Occurrence in Real News	Occurrence in Fake News
trump	1744	1328
obama	34	60
hillary	24	150
immigration	22	3
muslim	6	6
syria	18	9
finance	5	0
war	14	29
polls	4	15

The above table represents raw counts of words in the real and fake news. However, the data is a bit unbalanced, since there is about 1900 real headlines, and about 1300 fake news headlines. Here is the same table updated to shows occurrences of that word per headline in each section. For values above one, it means that there are on average more than just one occurrence of that word in the headline.

Word	Occurrence in Real News	Occurrence in Fake News
trump	0.8857	1.022
obama	0.01727	0.04619
hillary	0.01219	0.1155
immigration	0.01117	0.002309
muslim	0.003047	0.004619
syria	0.009142	0.006928
finance	0.002539	0.0
war	0.00711	0.02232
polls	0.002031	0.01155

Problem 2

Implementing a Naive Bayes Classifier
We implemented a Naive Bayes classifier to determine if the headline was fake or real. For features, we used the probability of a particular word showing up in the headline. For words that did not exist, we used a method called "delta smoothing" to add virtual examples into the training set.
So for features, we determined the following:

$$P(word|class) = \frac{count(word, class) + m\hat{p}}{count(class) + m} \tag{1}$$

The added prior of $m\hat{p}$ is to account for the case if the count is zero. This is undesirable, because then you'd think the probability of any word order never seen before is zero.
Now the probabilities of individual words given the class has been determined. Now, we want to use that to determine the probability of a headline. This can be done by simply multiplying the probabilities of each of the individual words in the entire test set

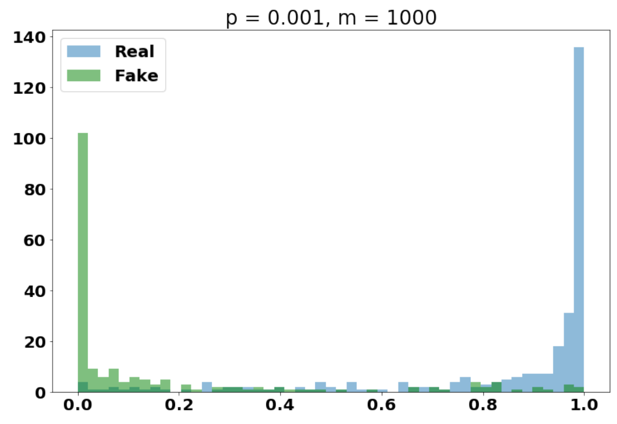
$$P(real|headline) = \frac{P(headline|real) * P(real)}{P(headline|real) * P(real) + P(headline|fake) * P(fake)} \tag{2}$$

$$P(real|headline) = \frac{\Pi_{i=1}^{k} p(word|real) * p(real)}{\Pi_{i=1}^{k} p(word|real) * p(real) + \Pi_{i=1}^{k} p(word|fake) * p(fake)} \tag{3}$$

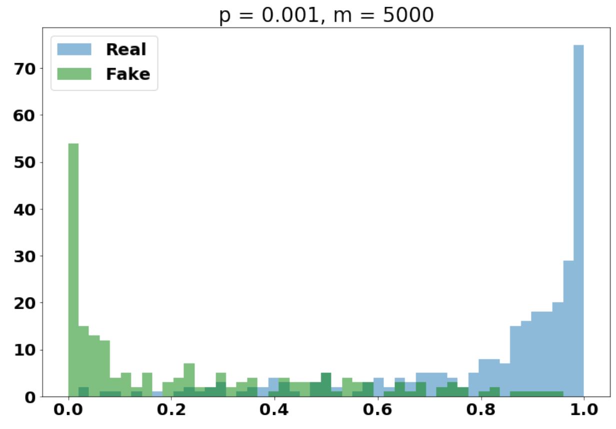
where $k =$ the number of distinct words in the corpus
Since there is multiplication of several very small numbers, its possible to run into arithmetic underflow. One way around this is to do the following substitution.

$$\Pi_{i=1}^{k} P(word|fake) = exp(log(\Pi_{i=1}^{k} P(word|fake))) = exp(\Sigma_{i=1}^{k} log(P(word|fake))) \tag{4}$$

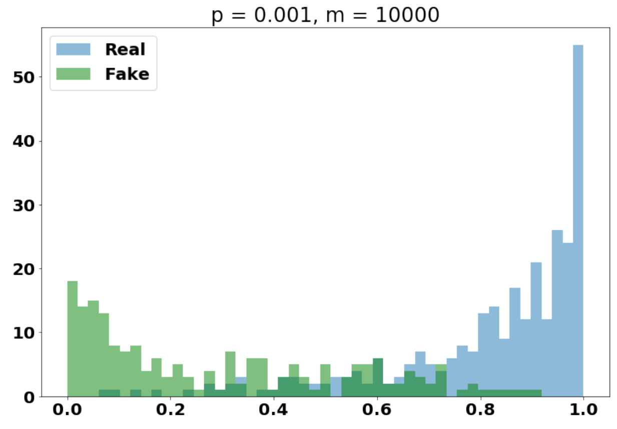
With some tuning, we were able to get a score of about 86.7%. This was done with a $\hat{p} = 0.001$, and $m = 1000$.
We tuned first by finding an optimal value of m. Using this optimal value of m, we then try to determine an optimal value of $\hat{p}$. All results generated during tuning was done by testing the model with the validation set. This assumes the effects of each parameter is independent. While this may be be entirely true, the approximation still gave good results, and allowed for a quicker optimization process.
Here are graphical visualizations showing how the distribution of scores move around with varying parameters



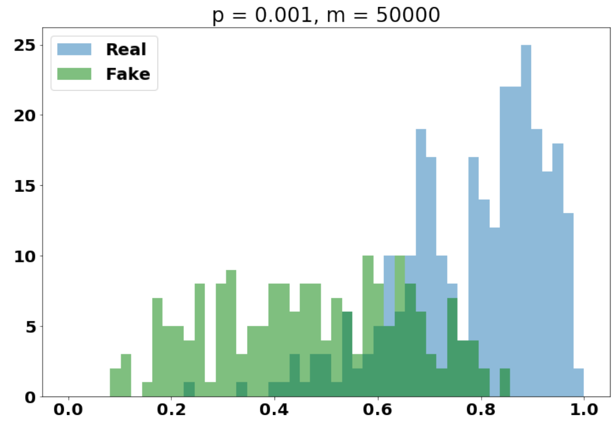
(a) Train : 95.1% Validation : 86.735%



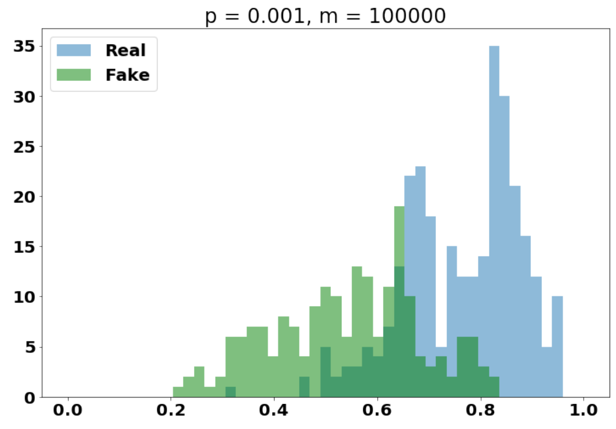
(b) Train : 92. 0 % Validation : 85.714%



(c) Train : 88.9% Validation : 85.102%

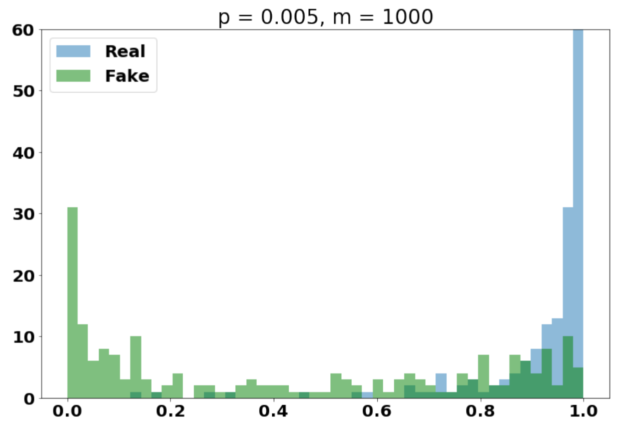


(d) Train : 81.3 % Validation : 79.183%

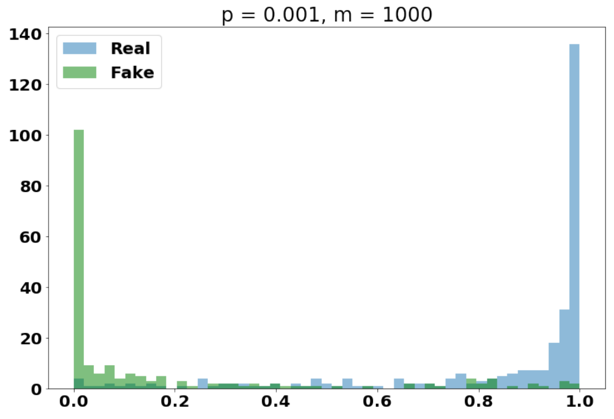


(e) Train : 75.5% Validation : 73.673 %

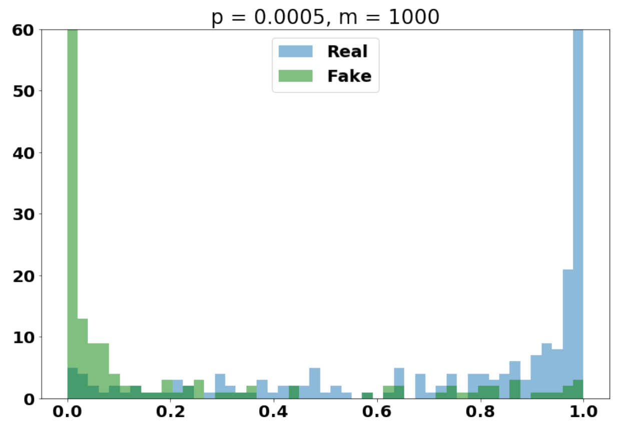
Figure 1: With a fixed $\hat{p}$ of 0.001, we varied the number of virtual examples added, and recorded the changing accuracy scores, which can be seen under each graph. Adding 1000 virtual samples gave us the best results.



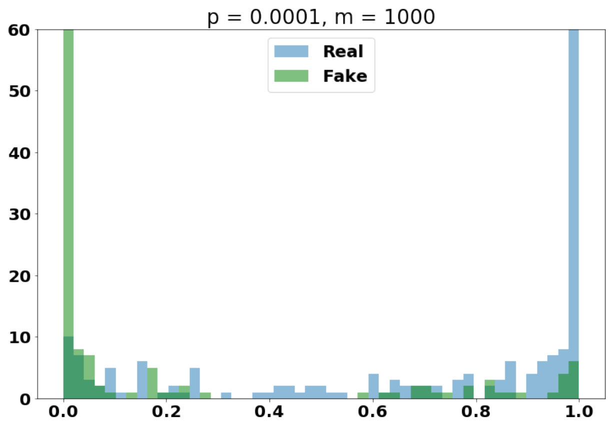
(a) Train : 87.5 % Validation : 81.224%



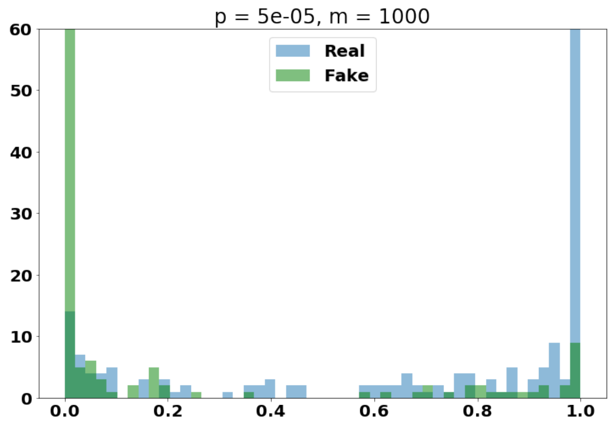
(b) 86.735%



(c) Train : 95.5% Validation : 84.898%



(d) Train : 96.1 % Validation : 83.265%



(e) Train : 97.3 % Validation : 83.061%

Figure 2: After selecting an optimal value of m, we varied the prior for the missing words, and recorded the changing accuracy scores, which can be seen under each graph. In conclusion, adding 1000 virtual samples with a prior of 0.001 gave us the best results, with an accuracy of 86.735%

After all the tuning, we tested the test dataset.

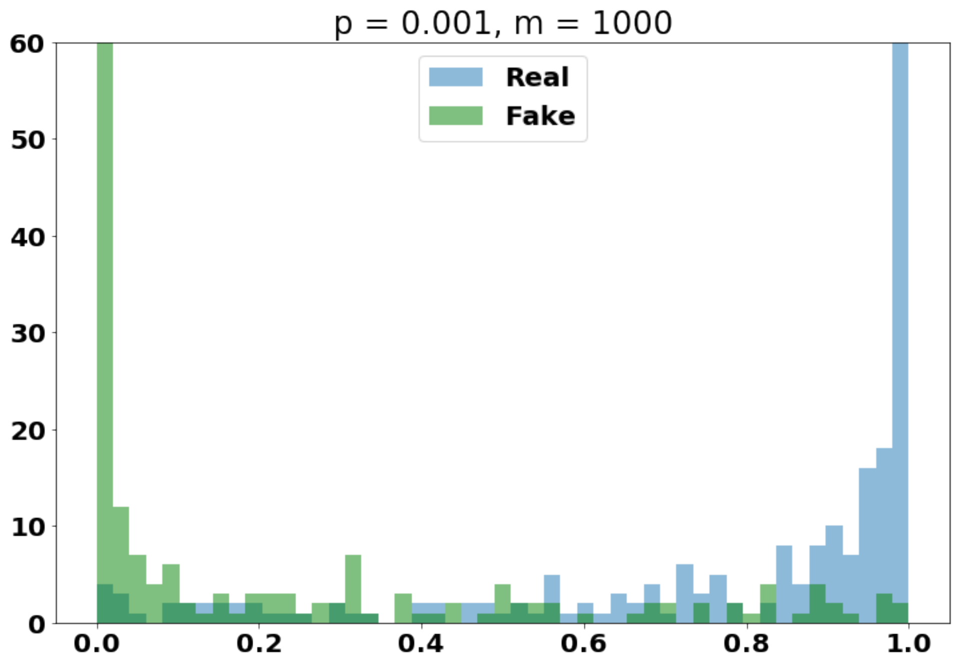


Figure 3: Final Test Performance: Accuracy Score of 85.3%

Problem 3

A

Interpreting the Features
As mentioned before, for each word, we solve for the probability of it appearing given that the headline is either real or fake, giving us the equations:

$$P(word|real) = \frac{count(word, real)}{count(real)} \; P(word|fake) = \frac{count(word, fake)}{count(fake)} \tag{5}$$

To determine which 10 words predicted the headline with its presence, we took 10 words who had the highest $P(word|real)$, and another 10 for $P(word|fake)$.
To determine which words highly indicated a class label with its absence, we took the ten words who had the highest value for $P(real|\neg word)$, and another ten that had the highest value for $P(fake|\neg word)$
These probabilities can be determined as follows with Bayes rules:

$$P(class|\neg word) = \frac{P(\neg word|class) * P(class)}{P(\neg word)} \tag{6}$$

$$P(\neg word|class) = 1 - P(word|class) \tag{7}$$

$$P(\neg word) = \frac{\text{headlines without word}}{\text{total number of headlines}} \tag{8}$$

The results are as follows, in order of most importance to least:

A	trump	donald	to	us	in	trumps	on	of	says	for
B	trunp	the	to	donald	in	of	for	a	and	on
C	trump	the	to	donald	in	for	of	a	and	is
D	trump	donald	to	trumps	us	on	in	of	says	for

Table 1: Word Table

A	Words whose presence indicated the news was likely real
B	Words whose absence indicated the news was likely real
C	Words whose presence indicated the news was likely fake
D	Words whose absence indicated the news was likely fake

Table 2: Legend

B

Here is an updated table with stop words removed

A	trump	donald	trumps	says	election	clinton	north	korea	ban	wall
B	trump	donald	hillary	clinton	election	obama	just	new	president	says
C	trump	donald	hillary	clinton	election	just	obama	new	president	america
D	trump	donald	trump	says	election	korea	north	ban	clinton	wall

Table 3: Word Table with same legend as above, but with stop words removed

C

Its makes sense to remove stop words, because you want to see if the words have any contextual meaning to it. For example, the word ”trump” indicates the sentence has something to do with a president, but the words such as ”us” and ”in” don’t really provide us with the same insight. However, it may be worthwhile to include stop words, since the model seems to think that their absence in the headline tends to indicate something, as they mostly occur in groups B and D, but not in A and C. Moreover, for more complicated models that are trying to learn context rather than pick up key words, the stop words are important, because it can sometimes change the entire meaning of the sentence. For example, the sklearn list of stop words includes the words can and $can't$. The sentence ”Trump can lead a country” vs ”Trump cant lead a county” have entirely different meanings. However, this is only really useful with more complicated models, such as Google’s word2vec model, which aims to encapsulate context.

Problem 4

Performance with Logistic Regression
Here are figures showing the performance with logistic regression with both L1 and L2 regularization.

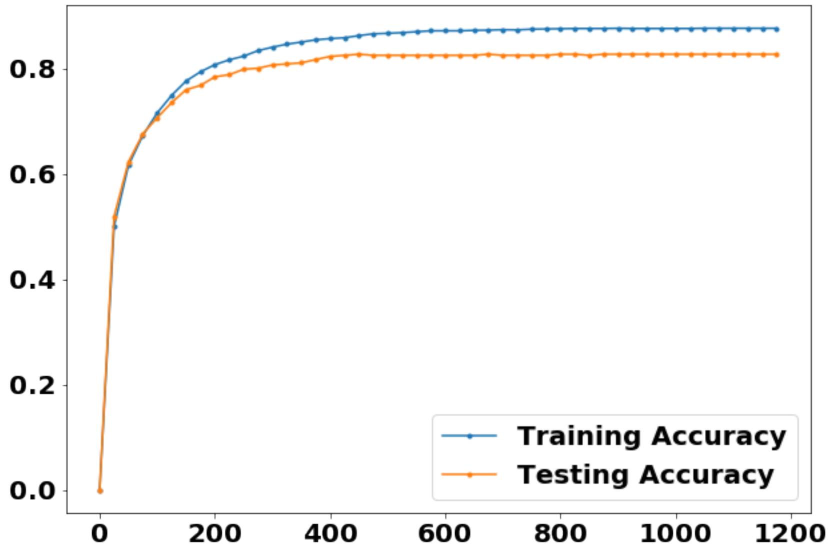


Figure 4: L2 Regularization - Max Score: 82.6%

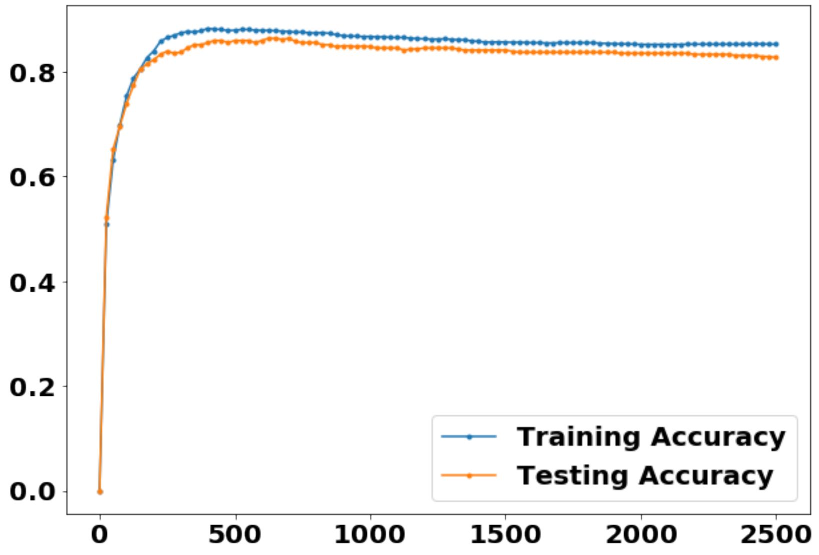


Figure 5: L1 Regularization - Max Score: 86.5%

The regularization parameter for both is 5. This was determined experimentally. We started testing values from 0.5 Smaller values gave poor result, but increasing it steadily increased score until about 5, where it began to decrease score. This is when it began penalizing the weights to the point where the model was no longer complex enough to capture the differences between the two classes of news. As can be seen from the plots. L2 regularization performs slightly better than L1. For both, a learning rate of 0.001 was used. This is because anything less than this took too long to converge, making the model inefficient to train, and for values larger, you would begin to run into arithmetic overflow errors.

Problem 5

In the example of the Naive Bayes classifier, the θs represent the probability of a sample with the feature I(x) being in the selected category. The I(x)s represent each word that appears in the test phrase.

In the case of logistic regression, the θs represent the weights of each feature and the I(x) represent the presence of each feature word in the test sample.

Problem 6

A

The table, and the corresponding legend below, show the top ten postive and negative weights and their corresponding words.

A	1.55	1.56	1.62	1.66	1.69	1.75	1.84	1.88	1.93	2.29
B	scales	traders	australian	trumpton	nose	korean	debates	donaldbreaux	turnbulls	usa
C	-1.87	-1.67	-1.57	-1.57	-1.39	-1.33	-1.32	-1.29	-1.16	-1.10
D	trumpcare	him	their	0	today	ab	c	ignorant	young	head

A	Top 10 positive θs
B	Words corresponding to positive θs
C	Top 10 negative θs
D	Words corresponding to negative θs

Table 4: Legend

In comparison to the lists from part 3(a), there seems to be a difference. There still seems to be a general theme of Trump in the words. However, the actual words seem to be significantly different. The word 'usa' corresponds to the highest positive theta in this list while it doesn't show up at all in the previous list. The same thing happens with the negative theta, the most negative weight, 'trumpcare' doesn't show up in the original list either.

B

The table below shows the lists with the stop words removed. The legend in table 4 still applies.

A	2.29	1.93	1.88	1.84	1.75	1.69	1.66	1.62	1.56	1.55
B	usa	turnbulls	donaldbreaux	debates	korean	nose	trumpton	australian	traders	scales
C	-1.87	-1.57	-1.39	-1.33	-1.32	-1.29	-1.16	-1.10	-1.09	-1.09
D	trumpcare	0	today	ab	c	ignorant	young	head	justice	newcastle

The same thing observed in part A can be observed here. The words themselves seem very different, even though the theme of Trump seems to be common in List A. It's interesting to note that the words in D for Problem 6 contains a number, and what appears to be letter fragments such as 'ab' and 'c', which was not seen in Problem 3.

C

This may be a bad idea because the parameters would be affected by the number of times a certain word occurs in each classified set. This would lead to inaccurate results if the chosen word was a common stop word. It is reasonable to use the magnitude in this problem because the occurrence of each word is counted only once in each headline and hence its magnitude doesn't have extreme values, its either 0 or 1. This means all present words contribute equally.

Problem 7

A

The graph below shows the relationship between the maximum depth of the decision tree and the performance accuracy of the training and validation set. The orange line represents the training set and the blue line represents the validation set. There doesn't seem to be as much improvement in the validation set accuracy as there is in the training set accuracy as the depth increases.

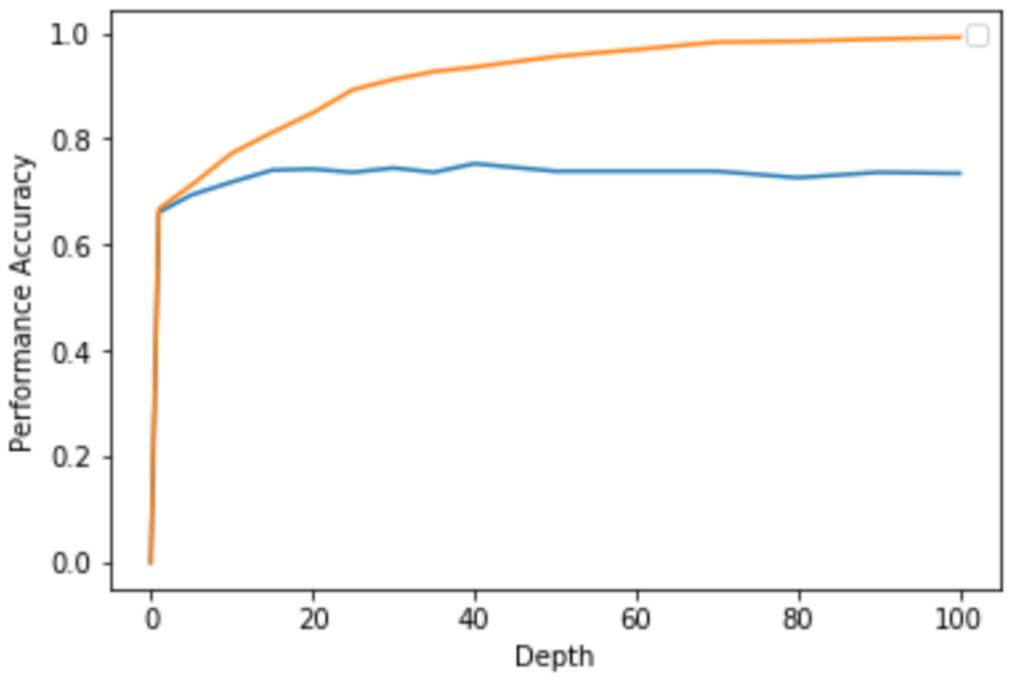


Figure 6: Graph of Depth of Decision Tree vs. Performance Accuracy

The best-performing depth was decided to be 25 because that was the point after which training set accuracy did not change by a lot.

The criterion of the decision tree model was changed to "entropy" in order to observe a difference. However, there was not a significant change made to the performance accuracy in this case so it was returned to the default 'gini' value.

B

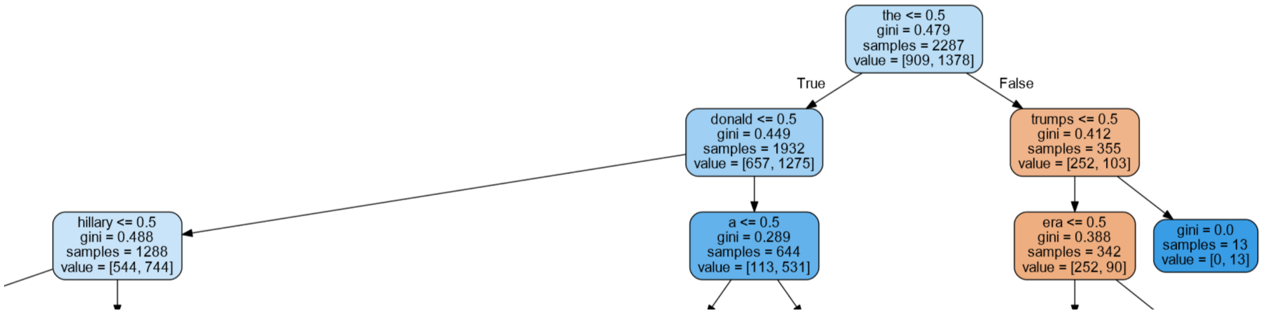


Figure 7: Visualization of Decision Tree

The above image shows the first two levels of the chosen decision tree classifier. The main feature words are all compared below from each classifier. Stop words are included.

Naive Bayes	trump the to donald in for of a and is
Logistic Regression	usa turnbulls donaldbreaux debates korean nose trumpton australian traders scales
Decision Tree	the donald trumps hilary a era

There is some similarity between the words whose presence indicate fake news (from 3a) using Naive Bayes and the feature words in the decision tree. The words 'donald', 'the' and 'a' can be seen in both lists. The words are significantly different from those chosen in the logistic regression classifier.

C

The Naive Bayes classifier gave us an accuracy of 86.73 % on the validation set under optimal tuning parameters. The logistic regression classifier gave us values of 86.5 % on training set accuracy and 84.2 % on validation set accuracy. The decision tree gave us a 92.70 % accuracy on training set and a 73.67 % accuracy on the validation set.
The decision tree classifier overfits the most. This is because the training set accuracy is much higher than the validation set accuracy. The tree has been modelled to fit the training set but does not perform as well on validation and test sets.

Problem 8

The mutual information on a split can be calculated by the formula below.

$$I(Y, x) = H(Y) - \sum_x P(X = x)H(Y|X = x)$$

In the case above, Y represents final classification, which is whether a given sample is real or fake news. x represents the chosen feature word based on which we are doing the split. The function H() represents the uncertainty and is calculated using the formula below.

$$H(V) = \sum_v P(V = v)log_2 P(V = v)$$

A

The first split occured at the word 'the' and this was used to calculate mutual information.

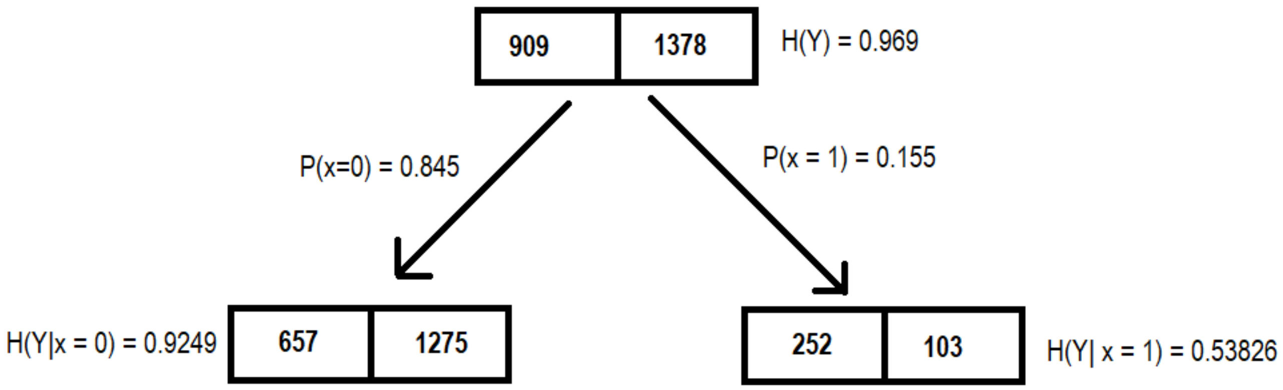


Figure 8: Mutual Information when x = 'the'

The numbers in the right hand side of each box represent how many samples fall into one category in our classification and the ones on the left represent the ones that fall into the other. The total number of samples for each split word is the sum of the numbers in the two halves.

$$H(Y) = \frac{-909}{2287} log_2 \frac{909}{2287} + \frac{-1378}{2287} log_2 \frac{1378}{2287}$$
$$= 0.969$$

$$P(x = 0) = \frac{657 + 1275}{2287} = \frac{1932}{2287} = 0.84477$$
$$P(x = 1) = 1 - P(x = 0) = 0.155$$
$$P(Y = 0|x = 0) = \frac{657}{2287} / \frac{1932}{2287} = 0.34$$
$$P(Y = 1|x = 0) = \frac{1275}{2287} / \frac{1932}{2287} = 0.66$$

$$H(Y|x=0) = 0.9249$$

In a similar manner,

$$H(Y|x=1) = 0.538$$

Therefore,

$$\begin{aligned} I(Y,x) &= 0.969 - 0.845(0.9249) - 0.155(0.53826) \\ &= 0.10424 \end{aligned}$$

B

We repeat the above calculations using a different word. This time we use the word 'donald'.

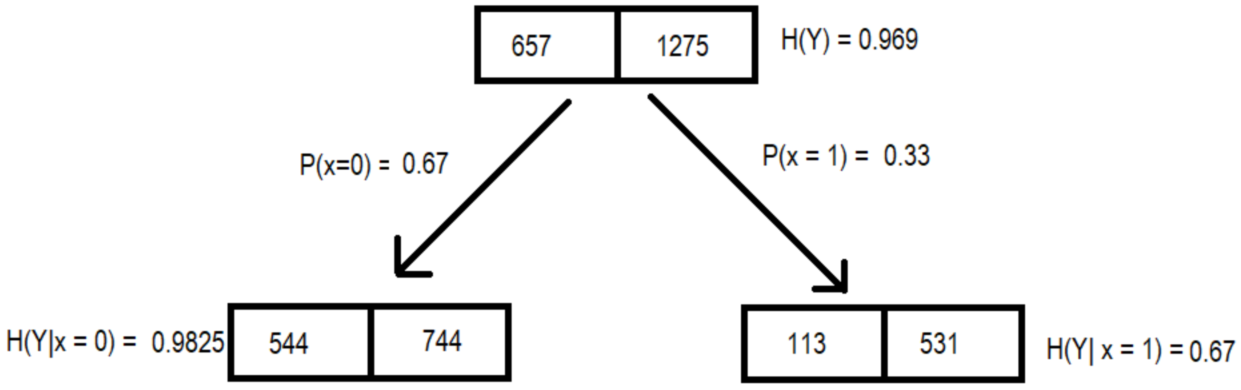


Figure 9: Mutual Information when x = 'donald'

$$\begin{aligned} H(Y) &= \frac{-657}{1932} log_2 \frac{657}{1932} + \frac{-1275}{1932} log_2 \frac{1275}{1932} \\ &= 0.925 \end{aligned}$$

$$\begin{gathered} P(x=0) = \frac{544+744}{1932} = \frac{1288}{1932} = 0.67 \\ P(x=1) = 1 - P(x=0) = 0.33 \\ P(Y=0|x=0) = \frac{544}{1932} / \frac{1288}{1932} = 0.422 \\ P(Y=1|x=0) = \frac{744}{1932} / \frac{1288}{1932} = 0.578 \\ H(Y|x=0) = 0.9825 \end{gathered}$$

Similarly, H(Y—x=1) = 0.67.
Therefore,

$$\begin{aligned} I(Y,x) &= 0.92488 - 0.67(0.9825) - 0.33(0.67) \\ &= 0.0455 \end{aligned}$$

The mutual information in the second calculation is lower thant the first one. This means that when a split occurs at this point, it is more certain whether a sample is real or fake as opposed to when it was split at the first point.