CSC411: Assignment #3

Due on Monday, March 19, 2018

Zhiyan Deng, Xin Jie Lee

March 18, 2018

Prefix

The code for this project is written in Python 3

Problem 1

***Part 1**: Dataset description*

The purpose of this project is to build classifiers that can classify news headlines as 'real' or 'fake' news. The dataset for this project comprised of 1298 labelled 'fake' news headlines and 1968 'real' news headlines. Both the 'fake' news and 'real' news headlines were subsets of the the Kaggle datasets "Getting Real about Fake News" and "A Million News Headlines" respectively.

A preliminary analysis was conducted to find the keywords that might be useful for our classfication task. The keywords w_i were ranked according to their likelihood probabilities $P(w_i|class) = \frac{number\,of\,'class'\,headlines\,containing\,w_i}{total\,number\,of\,'class'\,headlines}$, where *class* was either 'real' or 'fake'. Using this approach, the top 3 keywords that might be useful in classifying 'real' news headlines were:

Word: trump, Likelihood: 0.8836382113821138
Word: donald, Likelihood: 0.42073170731707316
Word: to, Likelihood: 0.19308943089430894

As we can observe, the keyowrds 'trump', 'donald' and 'to' appeared in 88.36%, 42.07% and 19.31% of all 'real' news headlines respectively. On the other hand, the top 3 keywords that might be useful in classifying 'fake' news headlines were:

Word: trump, Likelihood: 0.987673343605547
Word: to, Likelihood: 0.28197226502311246
Word: the, Likelihood: 0.2796610169491525

Once again, 'trump' and 'to' appeared in 98.77% and 28.20% of all 'fake' news headlines respectively, while 'the' appeared in 27.97% of all 'fake' news headlines. Since the use of likelihood probabilities would result in common useful keywords for classifying both 'real' and 'fake' news, it was evident that further analysis needed to done in order to build a good news classifier.

Finally, the dataset was split into a training set containing 70% of the news headlines, a validation set and a test set that each contain 15% of the news headlines.

Problem 2

***Part 2**: Implement Naive Bayes algorithm*

A Naive Bayes classifier was implemented to classify the news headlines. The posterior probabilities for every word that appeared in the training set were computed as such: $P(class|w_i) = \frac{P(w_i|class)P(class)}{P(w_i)}$. Once again, the likelihood probability is computed as follow $P(w_i|class) = \frac{number\ of\ 'class'\ headlines\ containing\ w_i}{total\ number\ of\ 'class'\ headlines}$, while the prior is assumed to be uniform and calculated as such $P(class) = \frac{number\ of\ 'class'\ headlines}{total\ number\ of\ headlines}$. The probability of a datapoint w_i is $P(w_i) = P(w_i|class)+P(w_i|\neg class)$ where $P(w_i|\neg class) = \frac{number\ of\ '\neg class'\ headlines\ containing\ w_i}{total\ number\ of\ '\neg class'\ headlines}$. Two dictionaries of posterior probablilities were built, one for the class 'real' news, and the other for the class 'fake' news.

During the computation of the likelihood probabilities for keywords w_i, there are a few words w_i that were present in one class, but not in the other. In this situation, the likelihood probabilities of these words given a class they did not exist in would be 0, meaning $P(w_i|class) = 0$. This would lead to a problem when we computed the likelihood probability of a news headline $P(W|class)$ since $P(W|class) = P(w_i|class) \times ... \times 0 \times ...P(w_n|class) = 0$. In such a scenario, it will be hard to classify the news headline. Hence, a modification was made to the computation of the likelihood probabilities, and they were calculated as such: $P(w_i|class) = \frac{number\ of\ 'class'\ headlines\ containing\ w_i\ + m*p}{total\ number\ of\ 'class'\ headlines + m}$, where m is the number of virtual news headlines. A grid search was carried out to find the optimal m and p, which were found to be m =1 and p = 0.312.

To classify a news headline in the validation and test sets or any unseen news headline in general, we would compare the posterior probabilities $P('real'|W)$ and $P('fake'|W)$ for that headline. The posterior probabilities were calculated as follows: $P('real'|W) = P('real'|w_1) \times ... \times P('real'|w_n)$ and $P('fake'|W) = P('fake'|w_1) \times ... \times P('fake'|w_n)$. To avoid the computation of the products of many small numbers which could lead to an underflow, we calculated $P('class'|W) = exp[log(P(class|w_1)) + ... + log(P(class|w_n))]$ instead. If the posterior probability $P('real'|W)$ was greater than $P('fake'|W)$, the news would be classified as 'real' news, and vice versa. For this model, we assumed that the words were independently distributed, which was a reasonable assumption for such a small dataset.

The performance of the optimized Naive Bayes model, using m = 1 and p = 0.312, is shown below:

Training Peformance: 96.93922168780061%
Training Real News Peformance: 97.46008708272859%
Training Fake News Peformance: 96.14961496149616%

Validation Peformance: 87.73006134969326%
Validation Real News Peformance: 90.5084745762712%
Validation Fake News Peformance: 83.50515463917526%

Test Peformance: 82.85714285714286%
Test Real News Peformance: 89.83050847457628%
Test Fake News Peformance: 72.3076923076923%

Problem 3

***Part 3 (a)**: Top 10 predictive words in the Naive Bayes model*

By ranking the 10 keywords with the highest posterior probabilities $P('real'|w_i)$, we obtained the top 10 keywords whose presence strongly predicted whether a news headline was real. The likelihood probabilities were modified to include the m and p parameters as in part 2, hence the posterior probabilities incorporated these parameters as well. Shown below are the top 10 keywords and their posterior probabilities:

Top 10 keywords whose presence strongly predicted whether a news headline was real
Word: korea, Posterior: 0.5990002183881386
Word: turnbull, Posterior: 0.5974635730742806
Word: travel, Posterior: 0.5971732412890436
Word: australia, Posterior: 0.5943795936827184
Word: paris, Posterior: 0.5919327366844588
Word: refugee, Posterior: 0.5900649723900256
Word: debate, Posterior: 0.5900649723900256
Word: ban, Posterior: 0.5870696634179932
Word: asia, Posterior: 0.5855941813772634
Word: tpp, Posterior: 0.5855941813772634

The top 10 keywords whose presence strongly predicted whether a news headline was fake were obtained by finding the 10 keywords with the highest posterior probabilities $P('fake'|w_i)$. These words and their posterior probabilities are:

Top 10 keywords whose presence strongly predicted whether a news headline was fake
Word: breaking, Posterior: 0.3944552698449771
Word: u, Posterior: 0.39323678356045494
Word: 3, Posterior: 0.39301044699854987
Word: soros, Posterior: 0.39301044699854987
Word: woman, Posterior: 0.39247555853489735
Word: info, Posterior: 0.3903256370224329
Word: reason, Posterior: 0.3903256370224329
Word: steal, Posterior: 0.3903256370224329
Word: d, Posterior: 0.3903256370224329
Word: 7, Posterior: 0.3896503640313593

On the other hand, the top 10 keywords whose absence strongly predicted whether a news headline was real were obtained by finding the 10 words with the highest posterior probabilities $P('real'|\neg w_i) = \frac{P(\neg w_i|'real')P('real')}{P(\neg w_i)}$, where $P(\neg w_i|'real') = 1 - P(w_i|'real') = \frac{number\ of\ 'real'\ headlines\ that\ do\ not\ contain\ word\ w_i}{total\ number\ of\ 'real'\ headlines}$. Since in our Naive Bayes model, we modified $P(w_i|'real')$ with the parameters m=1 and p=0.312, $P(\neg w_i|'real')$ for our model will be $= 1 - modified\, P(w_i|'real') = 1 - \frac{number\ of\ 'real'\ headlines\ containing\ w_i + m*p}{total\ number\ of\ 'real'\ headlines + m}$. The probability $P(\neg w_i) = P(\neg w_i|'real') + P(\neg w_i|'fake')$, where $P(\neg w_i|'fake') = 1 - P(w_i|'fake')$ and $P(w_i|'fake') = \frac{number\ of\ 'fake'\ headlines\ containing\ w_i + m*p}{total\ number\ of\ 'fake'\ headlines + m}$. Finally $P(class) = \frac{number\ of\ 'class'\ headlines}{total\ number\ of\ headlines}$. Shown below are the top 10 keywords whose absence strongly predicted whether a news headline was real and their posterior probabilities $P('real'|\neg w_i)$:

Top 10 keywords whose absence strongly predicted whether a news headline was real
Word: trump, Posterior: 0.5427944282863112
Word: the, Posterior: 0.338062375024789
Word: to, Posterior: 0.31884038323771324
Word: hillary, Posterior: 0.3179226295134852
Word: a, Posterior: 0.31548035695437115
Word: is, Posterior: 0.3134287071729037
Word: and, Posterior: 0.31282749343219357
Word: for, Posterior: 0.31225087859894346
Word: in, Posterior: 0.3117261544024581
Word: of, Posterior: 0.31154810720706644

Lasty, the top 10 keywords whose absence strongly predicted whether a news headline was fake were obtained by finding the 10 words with the highest posterior probabilities $P('fake'|\neg w_i)$. The method to compute these posterior probabilities were similar to $P('real'|\neg w_i)$, except that the class was 'fake'. The 10 keywords and their posterior probabilities $P('fake'|\neg w_i)$ are shown below:

Top 10 keywords whose absence strongly predicted whether a news headline was fake
Word: donald, Posterior: 0.23338888479896622
Word: trumps, Posterior: 0.21010406489176298
Word: us, Posterior: 0.20809210215942636
Word: says, Posterior: 0.20453694207721537
Word: north, Posterior: 0.20283194495145626
Word: korea, Posterior: 0.202774561554109
Word: ban, Posterior: 0.2024880460065508
Word: turnbull, Posterior: 0.20152100156792543
Word: travel, Posterior: 0.20136539550935603
Word: wall, Posterior: 0.2005456591121055

***Part 3 (b)**: Top 10 predictive words excluding stopwords*

Common stopwords such as 'a', 'to' may occur frequently in both real news and fake news and hence they can be of little value in helping us to classify new headlines. To analyze the impact of keywords whose presence strongly predicted whether a news headline was real or fake and were not stopwords themselves, we removed stopwords from the lists in part 3 (a) using the list of common english stopwords from sklearn. However, we noted that stopwords were not among the top words, and hence the top 10 keywords whose presence strongly predicted whether a news headline was real remained the same:

Top 10 keywords whose presence strongly predicted whether a news headline was real
Word: korea, Posterior: 0.5990002183881386
Word: turnbull, Posterior: 0.5974635730742806
Word: travel, Posterior: 0.5971732412890436
Word: australia, Posterior: 0.5943795936827184
Word: paris, Posterior: 0.5919327366844588
Word: refugee, Posterior: 0.5900649723900256
Word: debate, Posterior: 0.5900649723900256
Word: ban, Posterior: 0.5870696634179932
Word: asia, Posterior: 0.5855941813772634
Word: tpp, Posterior: 0.5855941813772634

and the top 10 keywords whose presence strongly predicted whether a news headline was fake remained unchanged as well:

Top 10 keywords whose presence strongly predicted whether a news headline was fake
Word: breaking, Posterior: 0.3944552698449771
Word: u, Posterior: 0.39323678356045494
Word: 3, Posterior: 0.39301044699854987
Word: soros, Posterior: 0.39301044699854987
Word: woman, Posterior: 0.39247555853489735
Word: info, Posterior: 0.3903256370224329
Word: reason, Posterior: 0.3903256370224329
Word: steal, Posterior: 0.3903256370224329
Word: d, Posterior: 0.3903256370224329
Word: 7, Posterior: 0.3896503640313593

***Part 3 (c)**: Reasons for removing stopwords in the Naive Bayes model*

Stopwords are words that occur frequently and hence the frequency of their appearance should be fairly high for both real news and fake news. This would mean that both likelihood probabilities $P(w_i|'real')$ and $P(w_i|'fake')$ would be relatively high for a stopword w_i, and so would their posterior probabilities $P('real'|w_i)$ and $P('fake'|w_i)$. Hence, these words would be of little value in identifying whether a news was real or fake and thus their removal could be beneficial in building a better model.

Problem 4

***Part 4**: Training a Logistic Regression model*

The second classifier, a Logistic Regression model, was trained to classify news headlines as real news or fake news. For a single headline h, the input to the logistic regression model was a k-dimensional vector v, where v[k]=1 if the k-th keyword appears in the headline h, and v[k]=0 otherwise. The set of keywords were the words that appear in all of the headlines in the training data. The classification output would be 1 for real news and 0 for fake news. The model was implemented in pytorch, and 'L2' regularization was used to help reduce overfitting of the model. With 'L2' regularization, we would be minimizing the cost function:

$$J(\boldsymbol{\theta}) = \sum_{i=1}^{k} [-y^i log\sigma(\theta^T x^i) + (1-y^i)(-log(1-\sigma(\theta^T x^i)))] + \frac{\lambda}{2}||\mathbf{W}||^{\mathbf{2}}$$

$$\sigma(\theta^T x^i) = \frac{1}{1+e^{\theta^{\mathbf{T}}\mathbf{x}}}$$

A grid search was conducted to find the optimal λ and the final optimized model had a $\lambda = 0.0001$, a learning rate of 0.0001 and ran gradient descent for 8000 iterations. The Adam Optimizer version of gradient descent was used and the final learning curves and classification results are shown below.

Training Set Performance: 98.68823786620025%
Validation Set Performance: 86.29856850715747%
Test Set Performance: 82.6530612244898%

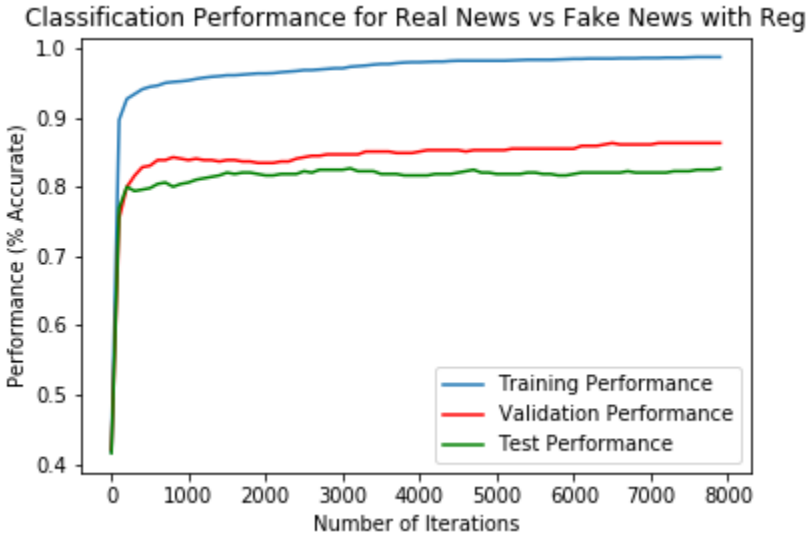

Figure 1: Classification accuracy vs learning rate with L2 Reg

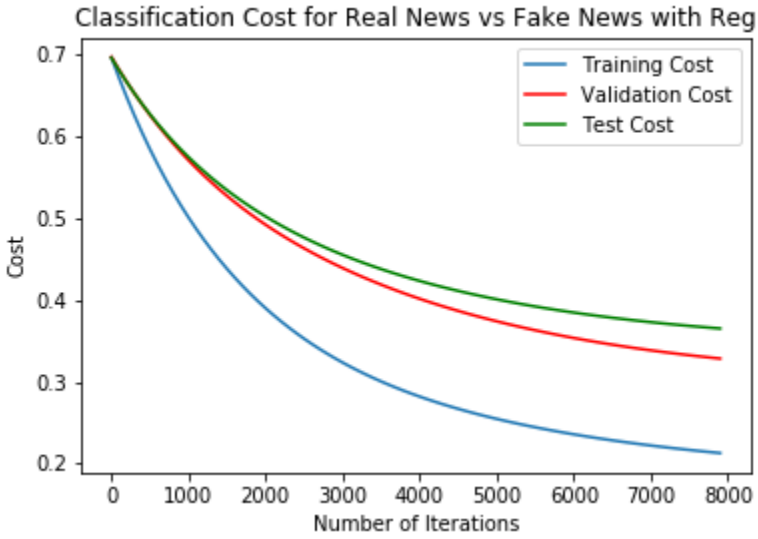

Figure 2: Classification cost vs learning rate with L2 Reg

Problem 5

***Part 5**: Representing inputs for Logistic Regression and Naive Bayes*

Every news headline input for the Logistic Regression and Naive Bayes model can be represented by the following computation:

$\theta_0 + \theta_1 I_1(x) + \theta_2 I_2(x) + ... + \theta_k I_k(x) > thr$

where $I_i(x)$ is an indicator variable for the unique keyword w_i, and $I_i(x) = 1$ if the keyword w_i exists in the news headline, and 0 otherwise. The vector $\{ w_1, w_2, ..., w_k \}$ represents the set of all uniques keywords that is obtained from the training data.

In the Naive Bayes model, θ_i is the logarithm of the posterior probability of the i^{th} keyword w_i in the training set with respect to the 'real' class minus the posterior probability of the i^{th} keyword w_i with respect to the 'fake' class, $log(P('real'|w_i) - P('fake'|w_i))$. The only exception is θ_0, which is 0 for this model. The threshold will be 0, and if $\sum_{i=0}^{k} \theta_i I_i(x) > thr = 0$, the news headline will be classified as 'real' news. If the sum is less than the threshold, the news headline will be classified as 'fake'.

On the other hand, in the Logistic Regression model, θ_i is the i^{th} parameter of the regression model that corresponds to the i^{th} unique keyword w_i in the training data, with the exception of θ_0 which is the parameter for the bias unit. The threshold is 0.5, and a news headline will be classified as 'real' news if $\sum_{i=0}^{k} \theta_i I_i(x) > thr = 0.5$, and 'fake' news if $\sum_{i=0}^{k} \theta_i I_i(x) < thr = 0.5$.

Problem 6

***Part 6 (a)**: Top 10 predictive words in the Logistic Regression model*

The top 10 words that are predictive of 'real' news in the Logistic Regression model are:

Top 10 words that are predictive of 'real' news in the Logistic Regression model
Word: 6, Weight: 0.805562734604
Word: voted, Weight: 0.763531625271
Word: keating, Weight: 0.740897536278
Word: vandalised, Weight: 0.716295778751
Word: where, Weight: 0.716208875179
Word: usas, Weight: 0.692780256271
Word: half, Weight: 0.686862587929
Word: desire, Weight: 0.680604279041
Word: capacity, Weight: 0.679793655872
Word: bitter, Weight: 0.677626788616

Contrast this list to the top 10 words for 'real' news in the Naive Bayes model:

Top 10 words that are predictive of 'real' news in the Naive Bayes model
Word: korea, Posterior: 0.5990002183881386
Word: turnbull, Posterior: 0.5974635730742806
Word: travel, Posterior: 0.5971732412890436
Word: australia, Posterior: 0.5943795936827184
Word: paris, Posterior: 0.5919327366844588
Word: refugee, Posterior: 0.5900649723900256
Word: debate, Posterior: 0.5900649723900256
Word: ban, Posterior: 0.5870696634179932
Word: asia, Posterior: 0.5855941813772634
Word: tpp, Posterior: 0.5855941813772634

There appears to be no common words between the top 10 predictive words for real news for both models. This could be the result of overfitting in either one of the two models, or that both models were regularized in a different manner.

For the top 10 words that are predictive of 'fake' news in the Logistic Regression Model:

Top 10 words that are predictive of 'fake' news in the Logistic Regression model
Word: x, Weight: -0.795315265656
Word: autistic, Weight: -0.768570184708
Word: rancher, Weight: -0.765343368053
Word: entire, Weight: -0.758085310459
Word: yearns, Weight: -0.757000148296
Word: bored, Weight: -0.746809720993
Word: hath, Weight: -0.745768547058
Word: wrought, Weight: -0.736066699028
Word: unity, Weight: -0.733608007431
Word: v, Weight: -0.731775760651

Contrast this list to the top 10 words for 'fake' news in the Naive Bayes model:

Top 10 words that are predictive of 'fake' news in the Naive Bayes model
Word: breaking, Posterior: 0.3944552698449771
Word: u, Posterior: 0.39323678356045494
Word: 3, Posterior: 0.39301044699854987
Word: soros, Posterior: 0.39301044699854987
Word: woman, Posterior: 0.39247555853489735
Word: info, Posterior: 0.3903256370224329
Word: reason, Posterior: 0.3903256370224329
Word: steal, Posterior: 0.3903256370224329
Word: d, Posterior: 0.3903256370224329
Word: 7, Posterior: 0.3896503640313593

Once again, there appears to be no common words between the top 10 predictive words for fake news for both models.

***Part 6 (b)**: Top 10 predictive words excluding stopwords in the Logistic Regression model*

After the removal of stopwords, the top 10 words that are predictive of 'real' news in the Logistic Regression Model are:

Top 10 words that are predictive of 'real' news in the Logistic Regression model
Word: 6, Weight: 0.805562734604
Word: voted, Weight: 0.763531625271
Word: keating, Weight: 0.740897536278
Word: vandalised, Weight: 0.716295778751
Word: usas, Weight: 0.692780256271
Word: half, Weight: 0.686862587929
Word: desire, Weight: 0.680604279041
Word: capacity, Weight: 0.679793655872
Word: bitter, Weight: 0.677626788616
Word: shaped, Weight: 0.673052310944

Contrast this list to the top 10 words for 'real' news in the Naive Bayes model:

Top 10 words that are predictive of 'real' news in the Naive Bayes model
Word: korea, Posterior: 0.5990002183881386
Word: turnbull, Posterior: 0.5974635730742806
Word: travel, Posterior: 0.5971732412890436
Word: australia, Posterior: 0.5943795936827184
Word: paris, Posterior: 0.5919327366844588
Word: refugee, Posterior: 0.5900649723900256
Word: debate, Posterior: 0.5900649723900256
Word: ban, Posterior: 0.5870696634179932
Word: asia, Posterior: 0.5855941813772634
Word: tpp, Posterior: 0.5855941813772634

As with part 6 (a), there appears to be no common words between both model's top 10 predictive words for real news.

After the removal of stopwords, the top 10 words that are predictive of 'fake' news in the Logistic Regression Model remained unchanged:

Top 10 words that are predictive of 'fake' news in the Logistic Regression model
Word: x, Weight: -0.795315265656
Word: autistic, Weight: -0.768570184708
Word: rancher, Weight: -0.765343368053
Word: entire, Weight: -0.758085310459
Word: yearns, Weight: -0.757000148296
Word: bored, Weight: -0.746809720993
Word: hath, Weight: -0.745768547058
Word: wrought, Weight: -0.736066699028
Word: unity, Weight: -0.733608007431
Word: v, Weight: -0.731775760651

Contrast this list to the top 10 words for 'fake' news in the Naive Bayes model:

Top 10 words that are predictive of 'fake' news in the Naive Bayes model
Word: breaking, Posterior: 0.3944552698449771
Word: u, Posterior: 0.39323678356045494
Word: 3, Posterior: 0.39301044699854987
Word: soros, Posterior: 0.39301044699854987
Word: woman, Posterior: 0.39247555853489735
Word: info, Posterior: 0.3903256370224329
Word: reason, Posterior: 0.3903256370224329
Word: steal, Posterior: 0.3903256370224329
Word: d, Posterior: 0.3903256370224329
Word: 7, Posterior: 0.3896503640313593

Once again, there appears to be no common words between the top 10 predictive words for fake news for both models.

***Part 6 (c)**: Relationship between Logistic Regression parameters and features*

In general, if the features of a Logistic Regression model are not normalized, it will be detrimental to use the magnitude of the parameters to determine the importance of the features. This is due to the fact that in general, if a particular feature x_i has a large magnitude, its corresponding parameter θ_i will have a smaller magnitude. Likewise, a feature x_j that is smaller in magnitude will tend to have larger parameters θ_j. In such a scenario, the parameters' magnitudes will not give us an accurate gauge of the importance of their corresponding features.

In our model, the features were equal in magnitude since one-hot encoding was employed. As a recap, the feature vector of each headline was represented by $\theta_0 + \theta_1 I_1(x) + \theta_2 I_2(x) + ... + \theta_k I_k(x)$, where $I_i(x)$ was an indicator variable for the unique keyword w_i, and $I_i(x) = 1$ if the keyword w_i existed in the news headline, and 0 otherwise. Hence, it was safe to assume words that appeared frequently in real news would have larger (more positive) parameters, while words that were common in fake news would have smaller (more negative) parameters.

Problem 7

***Part 7 (a)**: Training a Decision Tree classifier*

A Decision Tree classifier was trained to classify the news headlines. To help reduce overfitting of the model, the growth of the tree was stopped prematurely, and a grid search was carried out to identify the optimal depth level. Validation accuracy improved as the the maximum depth of the tree increased, up to a depth of 90. The results of the search indicated that a depth level of 90 produced the highest validation accuracy, and the final performance of the optimal model is shown below:

Training Performance: 98.1198076082%
Validation Performance: 78.118609407%
Test Performance: 76.9387755102%

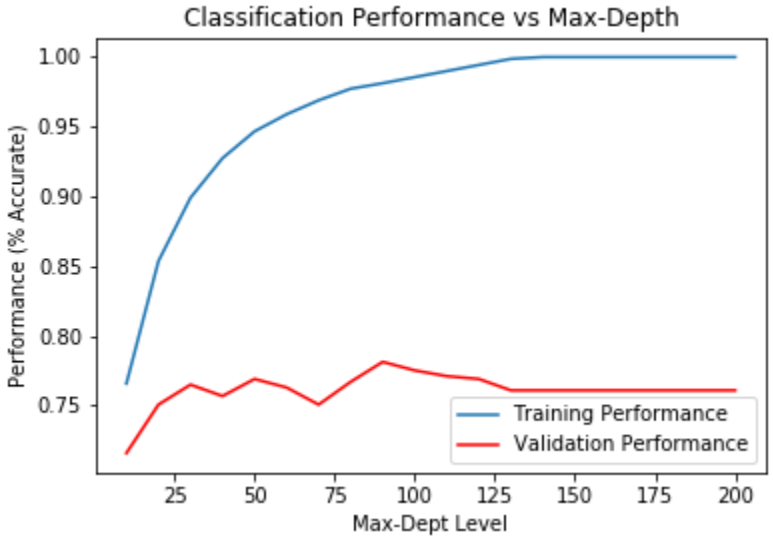

Figure 3: Classification accuracy vs max-depth level

***Part 7 (b)**: Visualization of the Decision Tree*

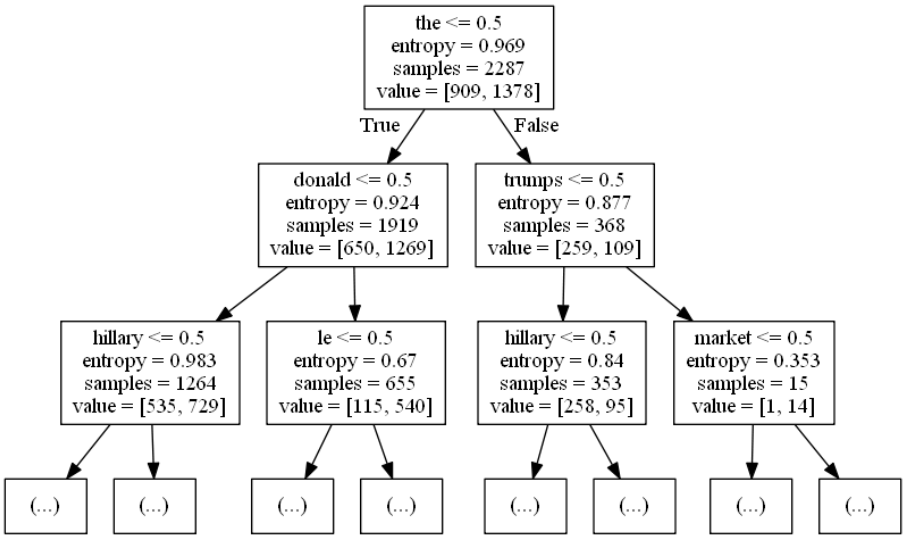

Figure 4: First two layers of the decision tree

As we can see from the from the figure above, the features selected from the Decision Tree model are similar to the ones in the Naive Bayes model, as the most important features for predicting a news to be real are among the top 10 most important features in the Naive Bayes model as well. However these keywords are again quite different from the keywords selected from the Logistic Regression model, which might be due to the regularization factor which is explained in part 6.

Please note that the image was converted from the dot file using the tool Graphviz.

***Part 7 (c)**: Summary of the three classifiers*

Of the three classifiers, Naive Bayes was the best performing classifier based on the validation and test set accuracies, followed closely by Logistic Regression. The worst performing classifier was the Decision Tree classifer. Due to the small size of the data, the assumption of words being independently distributed was relatively reasonable, hence the Naive Bayes classifier worked realtively well. On a larger dataset, we would expect the Logistic Regression model to work better since its performance would not be imapacted by the presence of correlated features as compared to the Naive Bayes model.

The Decision Tree model overfitted the most, since the drop-off from its training performance to its validation and test performance was the greatest. In our case, there was a 20% drop in performance for the Decision Tree classifier on the validation and test set even after the growth of the tree was stopped prematurely at 90 levels. Besides stopping the growth of the tree early, there are two additional ways of reducing overfitting and improving the performance of the classifier. One of which would be to 'prune' the tree by removing nodes that will drastically improve the validation performance, the other would be to use the Random Forest classifier, which would make classifications based on the averages of multiple Decison Tree classifiers.

As a recap, here are the performances of the 3 classifiers:

Naive Bayes Classifier
Training Peformance: 96.93922168780061%
Validation Peformance: 87.73006134969326%
Test Peformance: 82.85714285714286%

Logistic Regression Classifier
Training Set Performance: 98.68823786620025%
Validation Set Performance: 86.29856850715747%
Test Set Performance: 82.6530612244898%

Decision Tree Classifier
Training Performance: 98.1198076082%
Validation Performance: 78.118609407%
Test Performance: 76.9387755102%

Problem 8

***Part 8 (a)**: Compute mutual information $I(Y; x_1)$*

The mutual information $I(Y; x_1)$ quantifies how much x_1 tells us about Y, which is the class 'real' news. When picking an attribute x_j to split the tree, we pick x_j such that $I(Y; x_j)$ is as high as possible.

x_1 = 'the'

$H(Y) = \sum_y -P(Y = y) \log_2 P(Y = y)$
$= -\frac{909}{2287} \log_2 \frac{909}{2287} - \frac{1378}{2287} \log_2 \frac{1378}{2287}$
$= 0.529065889 + 0.440381853$
$= 0.969447742$

$H(Y|x_1 = 0) = -\frac{650}{1919} \log_2 \frac{650}{1919} - \frac{1269}{1919} \log_2 \frac{1269}{1919}$
$= 0.529024496 + 0.394562216$
$= 0.923586712$

$H(Y|x_1 = 1) = -\frac{259}{368} \log_2 \frac{259}{368} - \frac{109}{368} \log_2 \frac{109}{368}$
$= 0.356655435 + 0.519935222$
$= 0.876590657$

$H(Y|x_1) = P(x_1 = 0)H(Y|x_1 = 0) + P(x_1 = 1)H(Y|x_1 = 1)$
$= \frac{1919}{2287}(0.923586712) + \frac{368}{2287}(0.876590956)$
$= 0.916024649$

$I(Y; x_1) = H(Y) - H(Y|x_1)$
$= 0.969447742 - 0.916024649$
$= 0.053423093$

***Part 8 (b)**: Compute mutual information $I(Y; x_2)$*

x_2 = 'donald'

$H(Y|x_2 = 0) = -\frac{535}{1264}\log_2\frac{535}{1264} - \frac{729}{1264}\log_2\frac{729}{1264}$
$= 0.525005009 + 0.457935275$
$= 0.982940284$

$H(Y|x_2 = 1) = -\frac{115}{655}\log_2\frac{115}{655} - \frac{540}{655}\log_2\frac{540}{655}$
$= 0.440662626 + 0.22963232$
$= 0.670294946$

$H(Y|x_2) = P(x_1 = 0)H(Y|x_2 = 0) + P(x_1 = 1)H(Y|x_2 = 1)$
$= \frac{1264}{1919}(0.982940284) + \frac{655}{1919}(0.670294946)$
$= 0.87622705$

$I(Y; x_2) = H(Y) - H(Y|x_2)$
$= 0.969447742 - 0.87622705$
$= 0.093220692$

The mutual information $I(Y; x_2)$ is higher than $I(Y; x_1)$, and this suggests that we know more about Y after splitting the tree on x_2. This development is expected and further splits on other attributes x_j should help us gain more information on Y.