Fake News Detection Project Report

LI YIXI 20828874

1. Introduction

According to Media Statistics in 2021, 52% of Americans say they regularly encounter fake news online, and media trust worldwide has dropped by 8% between 2020 and 2021. It is critical to judge the news quality in such an information exploration era. In this project, we aim to build an efficient model to detect the fake news, using the data from the period of US presidential election in 2016. We are interested in what features fake news should have and which model will detect it most accurately. Our result will not only help readers or editors to verify the news quality but also help the social media websites to regulate the news contents.

2. Data Understanding and Preparation

2.1 Data source

Fake or Real News | Kaggle

2.2 Brief Summary

Dimension	Description		
Number of records	6335		
Existing attributes	index	int	a unique random number ID of each record
	title	string	title of news
	text	string	content of news
Newly created attributes	title_length	int	number of words in each news title
	text_length	int	number of words in each news text

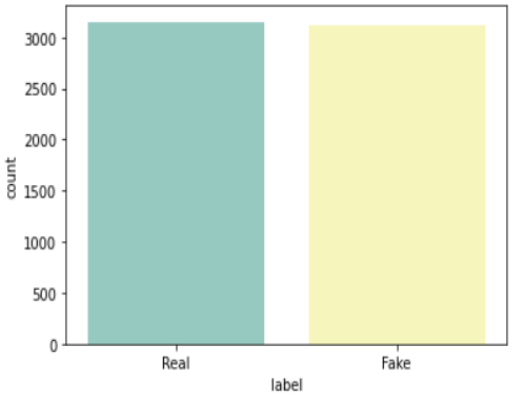
2.3 Data Cleaning

Problem	Description	Handling
Duplicates	29 duplicate records	Drop them directly to avoid noise
Missing values	36(0.57%) values in text are missing	Drop them directly due to low proportion
Nonstandard form of text	stop words, different morphology of the same word, useless punctuations, mixture of capital case lower case, words with extreme frequency	Apply count_vectorizer to handle these problems

2.4 Data Visualization

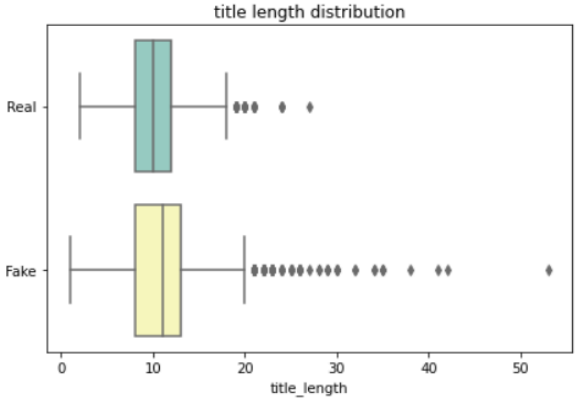
2.4.1 Class Balance

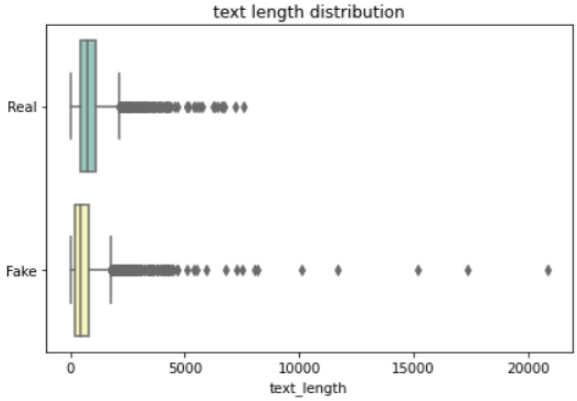
According to the bar chart, we have a balanced dataset.



2.4.2 Length Distribution

According to the boxplot, Fake news includes more extreme values in both title_length and text_length. It is interesting to notice that the title length of fake news is longer on average while text length of fake news is shorter on average. We consider fake news probably requires longer titles to catch eyes. As for the text content, fake news is made up and probably more likely to be lacking in detail, so it is shorter than real ones.



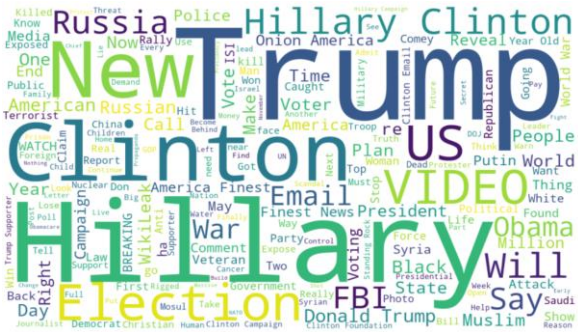


	text_length			title_length		
Label	mean	min	max	mean	min	max
Fake	686.8	1.0	20,891.0	11.1	1.0	53.0
Real	873.7	7.0	7,602.0	9.86	2.0	27.0

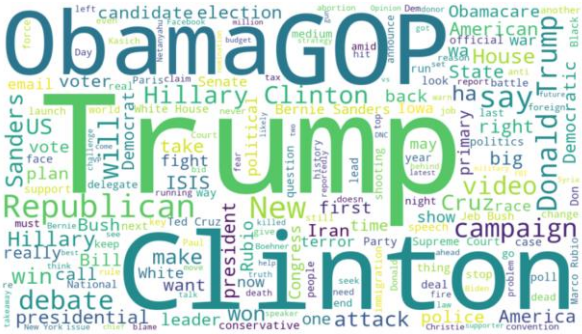
2.4.3 Hot words in news title

According to the word cloud plots, hot words are similar in fake news and real news, mainly related to the name of the president candidates.

• Fake News Title



• Real News Title



3. Model Building

At the beginning, we divided the data set into training(49%), validation(21%), and test datasets(30%), then applied eight data mining techniques. In each technique, we fitted the models with title data, text data and combined whole text data in the training set separately. The fitted models were evaluated by prediction performance in the validation set.

Furthermore, to improve the performance of models, we applied different feature selection methods or hyper-parameter tuning under each data mining technique. We used AUC to determine which model performs best in model building stage.

3.1 Logistic Regression

Logistic Regression is simple and very efficient on binary classification problems. It converts log-odds to the probability of a case being classified with a specific class label.

As logistic regression is good at handling numeric variables, we first tried *title_length* and *text_length* to train a model. The result was not satisfying(AUC=65%), which was reasonable because it is hard to judge fake news without any text information.

Next, we fitted the model with *title data*, *text data* and *whole data*, respectively. The model with *title data* had limited performance on accuracy (64.77%) and AUC score (70.95%), while the other two models had great accuracy over 90% and AUC performance over 95%.

To better interpret the result, we explored the best feature selection for the latter two models, seeking a chance to enhance their performance. We applied two feature selection methods, filter method with the chi-squared test and embedded method to find the optimal features k in the validation set. However, the selection of features was not significant, with very close AUC performance as before. Finally, we chose the model with 1550 features from *text data* suggested by the embedded method as the best logistic regression model (AUC=95.79%).

3.2 Naive Bayes Classifier

Naive Bayes classifier is a classification algorithm based on the Bayes Theorems. The reason why it is called "Naive" is because it assumes conditional independence given the class. Although it cannot describe the correlations between attributes, Naïve Bayes classifier is simple and efficient, and always performs well for classification in text mining.

At first, we fitted three multinomial Naïve Bayes classifier models with *title data*, *text data* and *whole data*. We left the model with *title data* at this stage because it performs much worse than others,with a 20% lower AUC.

To save the computational cost, we applied the filter method with the chi-squared test and embedded method with logistic regression to select features. Similar to logistic regression models, we evaluated prediction performance of each model on the validation set to select the number of features. Finally, we selected the one with 110 features using *whole data* (AUC=92.88%) as the best model under Naive Bayes classifier.

3.3 Decision Tree

Decision Tree applies recursive partitioning on training examples to build a tree. At each step, it chooses a splitting attribute which maximizes the information gain based on entropy or Gini impurity. Without any pruning, we achieved AUC 70.66% based on the *title data*, 80.41% based on the *text data*, and 80.72% based on the *whole data* respectively[1].

To avoid overfitting problems in test set, we tried two methods to select the best combination of parameters, GridSearchCV and manual pre-pruning. After comparing the AUC score[2], we found that manual pre-pruning based on the *whole data*, with criterion='entropy' and max depth=6, can reach the highest AUC 93.96%.

In the manual pre-pruning process, we used the validation data to tune the max_depth and find an optimal model. The reason why we only use max_depth is because the improvement of performance is limited when turning other parameters such as min_split, as shown in *figure 3.3.2*.

---

[1] *See Appendix 6.2.1*

[2] *See Appendix 6.2.5*

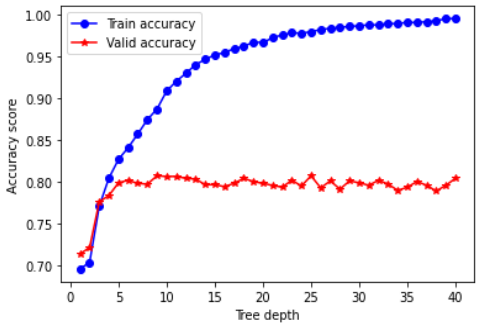


figure 3.3.1

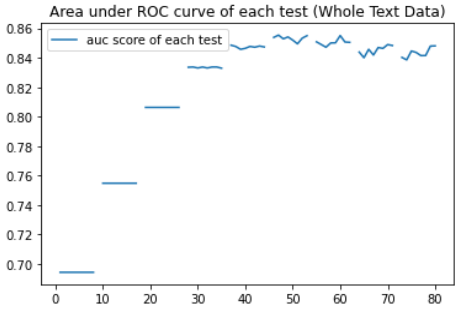


figure 3.3.2

In addition, we generated tree graphs for each model and put the tree graph of the best model in the appendix file[3].

3.4 Random Forest

Random Forest is a typical bagging method, which applies majority voting of multiple decision trees to improve the stability and prediction accuracy.

Unsurprisingly, title data has a poor performance again(AUC=71.47%). Without any hyperparameter tuning, the AUC using *text data* is as high as 96.53%, and the improvement on AUC (96.65%) after hyperparameter tuning (set number of trees = 650 using Grid_Search_CV) is limited. What's more, there is no significant difference on AUC when using *text data* or *whole data*. Considering the time efficiency, we chose *text data*(without tuning) as the best model under Random Forest.

3.5 SVM

SVM is a popular classification algorithm, in which we plot each data item as a point in n-dimensional space with the value of each feature being the value of a particular coordinate. Similarly, we applied SVM to the *title data*, *text data*, and *whole data* separately. Without any hyperparameter tuning[4], the model with *title data* has a low AUC with 67.91% on validation set, and model with the *text data* has a similar AUC (92.84%) compared with the model with the *whole data* (AUC=93.56%), which is quite satisfactory.

To improve the model performance, we used Grid_Search_CV to do hyperparameter tuning on *kennel*, *C* (strength of the regularization), and *gamma* (how far the influence of a single training example reaches). We tried four different kernels(linear, rbf, sigmoid, poly), five different *C* ([0.01,0.05,0.1,0.5,1]), four different *gamma* ([0.05,0.1,0.5,1,5]) and observe the validation set performance. We didn't set a range in *C* and *gamma* due to time efficiency. Finally, we found that the combination {'C': 0.01, 'gamma': 0.05, 'kernel': 'linear'} could achieve the highest AUC (95.8%) on the validation set, which is the best model under SVM.

As we learned in class, SVM is sensitive to different kernels. It is interesting to notice the linear kernel generates the best performance, and this is always true in text classification problems. The reason might be the number of features is usually larger than the number of samples in text classification scenarios, thus it is likely to cause overfitting problems when using nonlinear kernels and lead to a poor performance in validation set.

Kernel	Linear	RBF	Sigmond	Poly
AUC in validation set	95.8%	73.72%	82.89%	86.84%

figure 3.5.1 AUC on different kernels

[3] *See Appendix File Tree_best.pdf*

[4] *default parameter setting: { 'kernel': 'rbf', 'C': 1, 'gamma': 'scale'}*

3.6 KNN

KNN algorithm is a lazy learner, and it classifies a new case by finding majority labels among its K nearest neighbors. Traditionally, euclidean distance is used to find the nearest neighbor. Without any hyperparameter tuning, we achieved AUC 62.55% based on the title data, 80.55% based on the *text data*, and 81.01% based on the *whole data*.

Considering KNN doesn't perform well in high dimensional cases, and we had over 2000 features, we tried to select key features using filter and embedded methods. We tried to find the optimal K with highest AUC using the validation set. Within expectation, AUC improved 3.6% and 5.2% after tuning based on the *text data* and *whole data*, respectively. Finally, we chose the model fit on 190 text features using the filter method and k=4 under KNN, which had the highest AUC(89.27%).

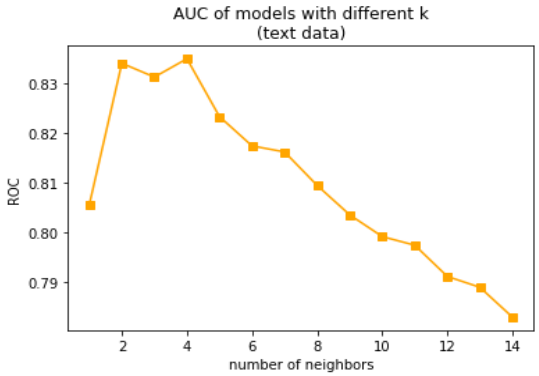


figure 3.6.1

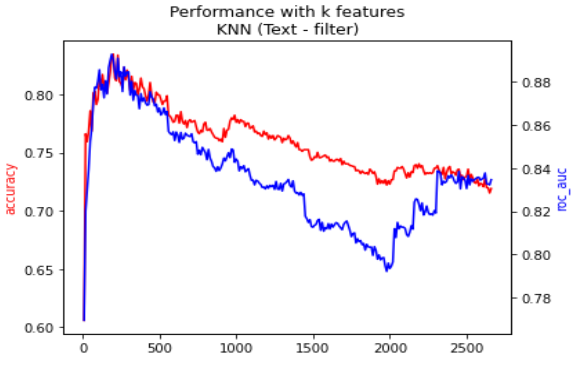


figure 3.6.2

3.7 Ensemble Learning

To improve the performance of unsatisfactory models, we applied simple averaging ensemble learning, which applies different learning algorithms fitted on the same data and then combines their predictions as the prediction of the model.

The key idea we choose which model to be combined is that they are not complex and not perform well individually, i.e., AUC is below 95%.

For *title* data, we built the model with an ensemble of all above algorithms. However, the performance did not improve obviously (AUC=70.23%). For *text data* and *whole data*[5], we combined Naive Bayes Classifier, decision tree and KNN as the new model. The AUC fitted with text data and *whole data* is 93.77% and 94.63% respectively. So, we chose the model fitted with *whole data* as the best one under ensemble learning.

3.8 Neural Network

To explore the potential improvement, we also implemented neural network embedding, whose framework contains one embedding layer, one flat layer and one dense layer.

However, the previous preprocessing tool(CountVectorizer) would discard lots of information about word order and grammar, which was not suitable anymore. Neural networks such as CNN, RNN can infer meaning from the order of words in a sample. Therefore, in the data preprocessing stage, we represented the text as a sequence of tokens, preserving order, and used padding to ensure that all sequences in a list have the same length. In the training process, we used the early stopping method to monitor the validation loss in

[5] *For detailed performance data, please refer to Appendix 6.3*

each epoch in order to avoid the overfitting problem (stop training if the loss suddenly becomes larger than the previous epoch).

Finally, modeling on *text data* with only 6 epochs reached the best performance (AUC=93.23% on validation set). Although AUC score was not as high as expected, it was reasonable because there exists lots of space to do further modification, such as deeper and more complex layers and framework, smaller batch size (32,64 or 128), try other optimizer and activation methods etc. We believed those modifications might change the performance significantly if we do the further exploration. At least by far, the neural network has proved its powerful learning ability with even simple networks.

3.9 Summary of model building

- *Title data* had a poor prediction performance under each algorithm, which indicates it is hard to distinguish fake news just through titles.
- More information doesn't guarantee better prediction performance[6].
- Feature selection could improve time efficiency and prevent overfitting problems.

Algorithm	Data used	Method	Best Performance		
			AUC	Total time(s)	Time saved(s)
Logistic Regression	*Text*	Embedded feature selection	95.79%	0.2473	0.0451
Naive Bayes	*Whole*	Embedded feature selection	92.88%	0.0068	0.0067
Decision Tree	*Whole*	Embedded feature selection	93.96%	0.3631	0.5764
Random Forest	*Text*	NA	96.52%	5.1874	-2.1614
SVM	*Whole*	Hyperparameter tuning	95.80%	49.2555	-25.6408
KNN	*Text*	Filter method and hyperparameter tuning	89.27%	1.2244	-0.4008
Ensamble Learning	*Whole*	Naive Bayesian +KNN+DecisionTree	94.61%	2.1449	-
Neural Network	*Text*	Callback to optimize epoch numbers	93.30%	6.0558	-

Time saved = Total Time with tuning - Total Time without tuning

4. Performance Evaluation

4.1 Brief Overview

In the model building stage, we selected the best model for each algorithm based on AUC. To compare the performance among these models, we re-trained these eight best models on the train_all dataset (train+validation) using parameters and features selected and made predictions based on the *test dataset*. The table below showed the test set performance of eight models across different measures.

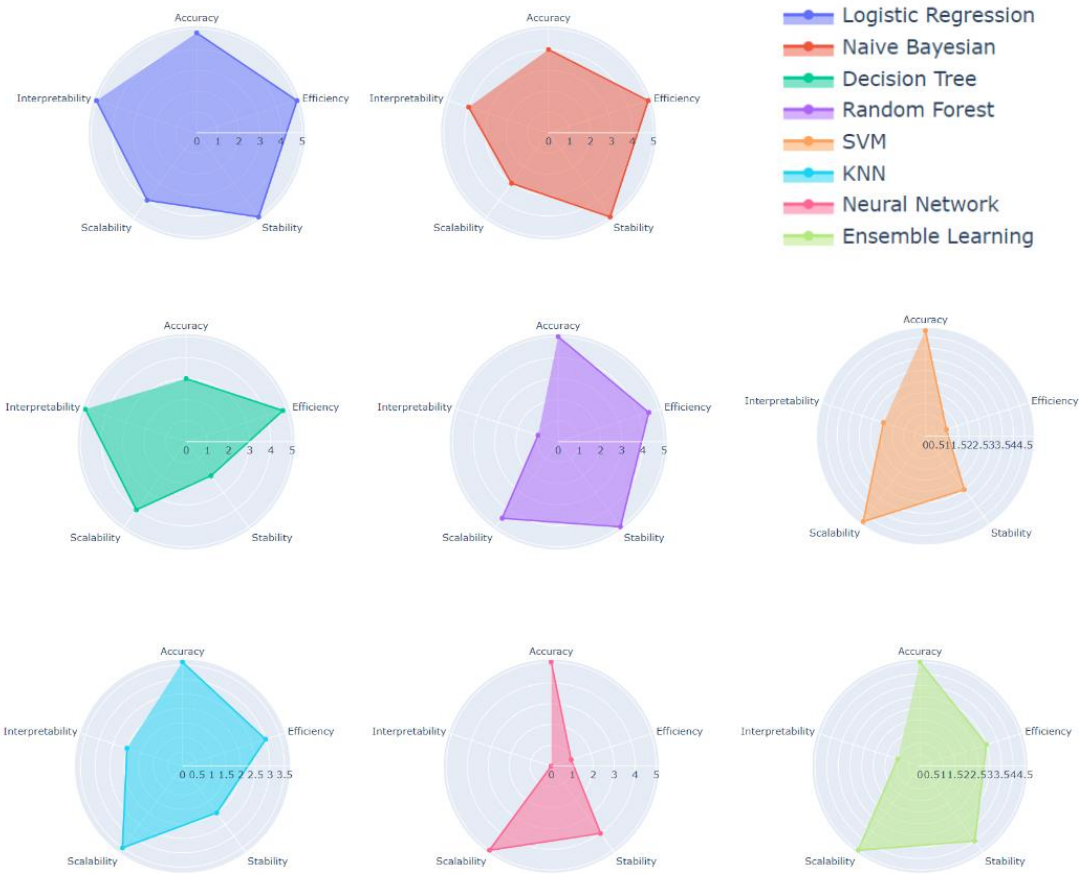
Besides, we set a benchmark model as baseline, where the prediction of our target variable is the majority labels in the train_all set. Its accuracy is only 48.91%, which is almost a random guess. It is obvious that all the other models are more accurate than the benchmark one.

[6] *Result table without feature selection/hyperparameter tuning could be found in Appendix 6.4*

Algorithm	Test Set Performance					
	Accuracy	TPR	AUC	Fit Time(s)	Prediction Time(s)	Total Time(s)
Logistic Regression	90.64%	91.47%	95.80%	0.2451	0.0022	0.2473
Naive Bayes	84.42%	87.20%	91.93%	0.0050	**0.0018**	**0.0068**
Decision Tree	81.50%	84.08%	86.90%	0.3578	0.0053	0.3631
Random Forest	**91.49%**	90.53%	**97.36%**	4.9665	0.2209	5.1874
SVM	90.64%	**92.92%**	96.56%	42.8383	6.4172	49.2555
KNN	82.40%	83.04%	89.08%	**0.0016**	1.2228	1.2244
Ensemble Learning	88.20%	90.22%	94.88%	0.3749	1.7700	2.1449
Neural Network	87.25%	-	94.94%	4.1518	1.9040	6.0558

4.2 Evaluation through Radar Map

Radar Map is a great tool to show strength and weakness of each algorithm. Five dimensions are chosen to evaluate the performance, which are *Efficiency* (whether it is time consuming), *Accuracy* (whether the prediction is precise), *Interpretability* (whether the model is simple and easy to interpret), *Scalability* (whether it has assumptions or could be applied in high dimension data) and *Stability* (whether the algorithm is sensitive to hyperparameter tuning). From 0 to 5, a higher score indicates a better performance on the corresponding perspective. Therefore, an ideal algorithm we expect should have a large and balanced area on radar map.



Below are the evaluation details of each data mining technique.

4.3 Evaluation of each technique

4.3.1 Logistic Regression

The Logistic Regression model performed well with high accuracy and efficiency. And there was no overfitting problem as we can see from the accuracy improvement. Given its power on generating binary class probabilities without requiring linearity between inputs and output variables, this model would be very helpful in such cases handling text.

4.3.2 Naive Bayesian

The Naive Bayesian model spent the least time among these five algorithms while maintaining a relatively good accuracy performance, which indicates one strength of Naive Bayesian model - it is very simple and can perform efficiently in both storage space and computation time. But it also has one limitation as it requires the conditional independence assumption. Obviously, here not all attributes are independent of each other, since words on the same topic tend to appear together.

4.3.3 Decision Tree

The Decision Tree model spent a relatively short time and had a general performance. The performance is below expectation, which is because the text data may contain some noise. Another reason for that performance of the decision tree method is not so good is perhaps the sample size is not big enough. The decision tree usually performs well in a huge dataset.

Compared to the performance based on train and validation dataset, the accuracy and AUC of decision tree based on train_all and test dataset have some changes. This is because Tree structure is not very stable, which is sensitive to change of training data.

4.3.4 Random Forest

Random forest achieved the best performance among eight models and was highly time efficient and easy to find the feature importance.

4.3.5 SVM

The SVM model reached high AUC and TPR, but it was very time-consuming. As a very effective model in high dimensional spaces, handling data with massive features like this case would be easy for it. But it doesn't provide probability estimates directly, which requires highly time-consuming calculation using 5-fold cross validation.

4.3.6 KNN

The KNN model had the worst performance on accuracy. It is reasonable as separate words in our data cannot provide sufficient information for the model to learn. Meanwhile it takes longer time for prediction, which is not user-friendly especially when there's massive data.

4.3.7 Ensemble Learning

The Ensemble Learning model had good prediction accuracy by combining weak classifiers to a stronger one but is difficult for interpretation.

4.3.8 Neural Network

Neural Network had good prediction performance, but it is impossible to interpret the results due to black box issue and costed us lots of time to build the model.

4.3.9 Comparison on Top Informative Words among Algorithms

After checking the top 20 informative words for fake news detection under three available

algorithms, we found the most informative words between different models are partly different, but there are still some words in common. Specifically, 2016 appears as one of the top 3 informative words for all the algorithms. Logistic Regression and Decision Tree have more words in common - article, october, november_2016, print, hillary clinton, ago, trump win, obama, etc. These words or their combinations can help us detect fake news with considerable accuracy levels, the details are in the conclusion.

5. Conclusion

5.1 Which model is most accurate in predicting fake news?

All models are wrong, but some are useful.

For people not interested in models, we will provide some common sense. It is hard to identify fake news by title only, so please have a look at the text. We recommend you observe the length of the text, the news is more likely to be fake if it is too short or extremely long. Besides, be more careful when seeing negative information because it is more likely to be made up to draw your attention.

For people who seek for a balance between accuracy and efficiency, we will highly recommend the logistics regression with feature selection, which is simple, fast, and always has a good performance.

For people who seek for accuracy, we will recommend you try Random Forest, which is time efficient and always has a good performance.

For people interested in fancy algorithms, deep learning such as CNN is an ideal choice.

5.2 Which words are more informative in predicting fake news?

Logistic regression, Naive Bayesian and Decision tree could return the feature importance, and a higher score indicates this word is more informative in prediction. The top 20 key words under each algorithm could be found in figure 5.

- **Nouns and verbs are more useful than adverbs and adjectives?**

According to the word class, 65% of the keywords are nouns and 21.67% of them are verbs.

Word Class	Count	Percentage
noun	39	65.00%
verb	13	21.67%
adj	5	8.33%
adverb	3	5.00%
Grand Total	**60**	**100.00%**

When taking part of the nouns, it is interesting to notice that date plays an important role in prediction. It is not surprising that “2016” has a high rank in each algorithm, because the president election was held in 2016. “October” and “November” are also important, which is probably due to the fact that US election day is on November 8th and October is the most exciting phase for last propaganda. For example, on October 28 FBI said it would be investigating newly discovered emails pertinent to its previous investigation of Hillary Clinton's private server. This news is a tipping point on discussion across America and led to heavy drops of Hillary's polls within days. It is likely to be utilized by social media to create fake news and interfere with voting.

Similarly, “Obama”, “Hillary”, “Trump” are the important names in prediction because they are the key figures in this election. Based on this information, we inferred that our dataset was collected in the late phase(after July) during the election, as Trump and Hillary were nominated to be the candidate by their party in late July.

Subclass	Count	Percentage
others	16	41.03%
date	11	28.21%
name	7	17.95%
place	3	7.69%
party	2	5.13%
Grand Total	**39**	**100.00%**

However, we consider that there might be a popularity bias in these keywords. According to the association rule learned in the lecture, high *P(Beer|Diaper)* perhaps just a reflection of correlation, because "Diaper" occurred too frequently. Similarly, almost every news contains date like "2016" and candidate names like "Trump" thus log P(Fake|Key words) is high.

- **Fake news is more negative?**

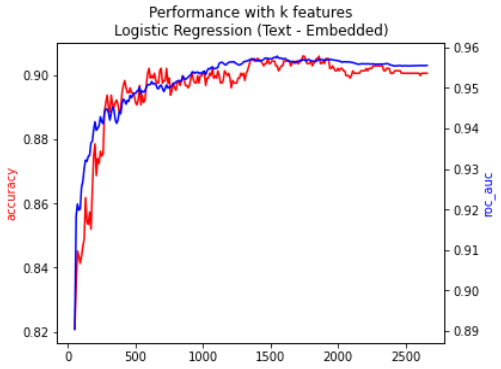
Five important adjectives aroused our interest too, particularly two of them("illegal", "violent") are negative and the rest are neutral. In reality, people are more easily attracted by bad news than good news. We infer that people who write fake news know this logic well, so they add more negative words to catch public eyes. Definitely, we will be more excited in hearing Hillary's email scandal compared with her contribution to charity.

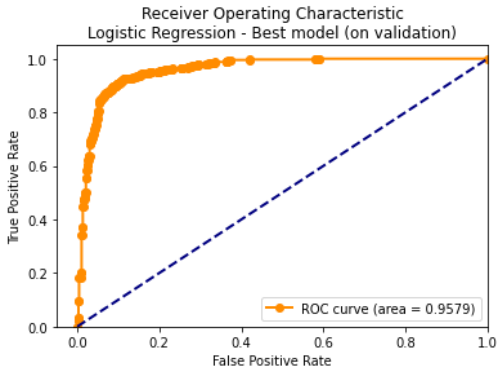
5.3 What are the potential problems to be noticed?

The cost matrix in this problem is interesting to be explored in the future. The decision threshold is set to be 0.5 among all the algorithms in our project. However, False Negative cost might be different from False Positive cost in reality. In this case, FN means a news is real but misclassified as fake, while FP means the news is fake but misclassified as real one. Besides, it is hard to quantify the cost. For example, if readers believe in fake news, they may change the votes from Hillary to Trump, but what is the opportunity cost here?

6. Appendix

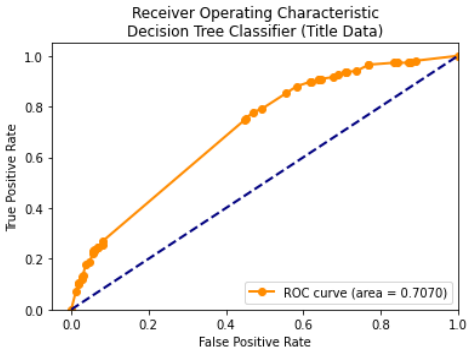
6.1 Demonstration on Feature Selection (Logistic Regression)

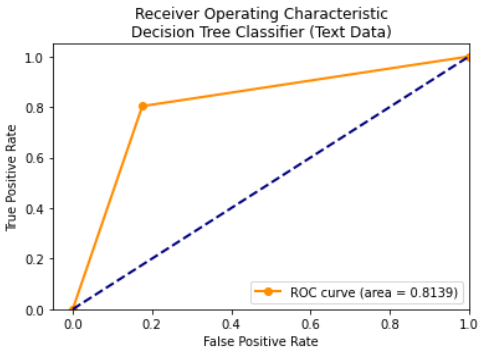


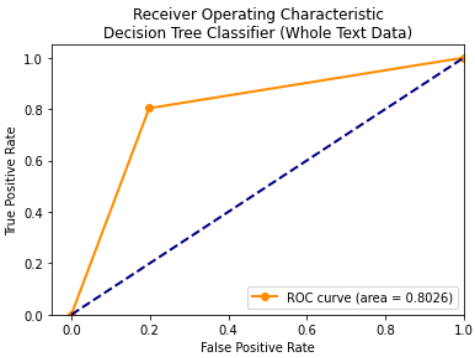


6.2 Decision Tree

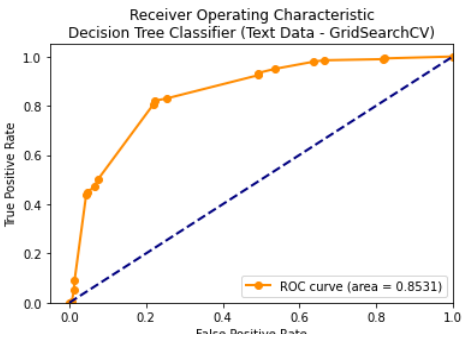
6.2.1 Performance of default models based on title data, text data and whole data

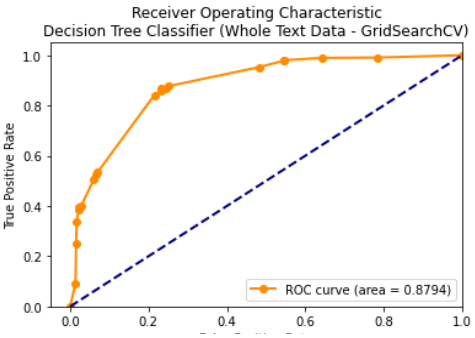


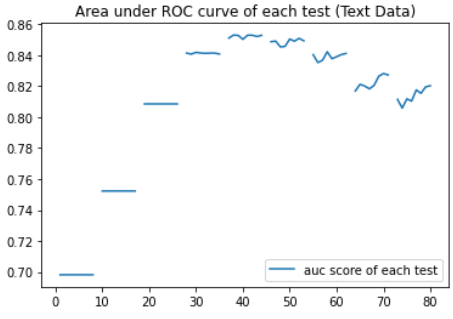


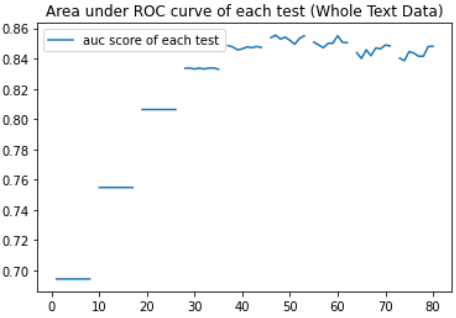


6.2.2 Performance of GridSearchCV based on text data and whole data respectively



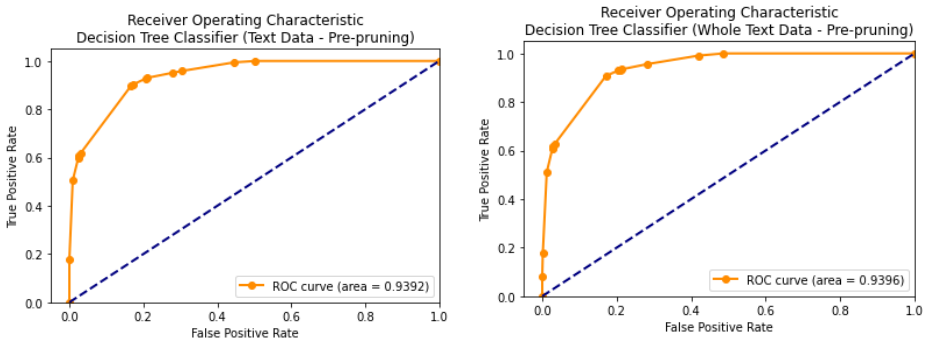




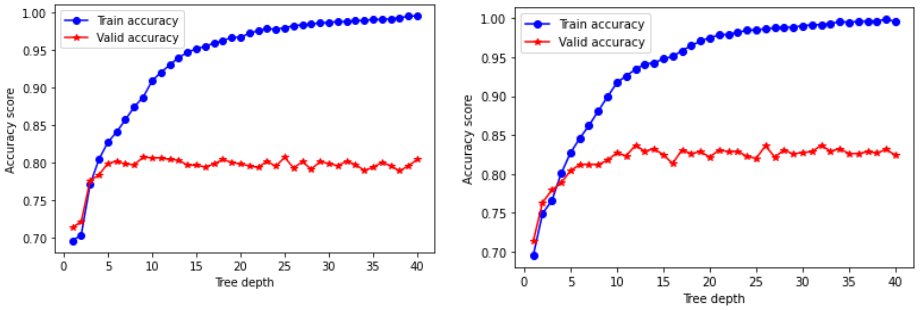


(A single blue line represents a combination of a single max_depth value with different min_split values from 1 to 10.)

6.2.3 Performance of Pre-Pruning based on text data and whole data respectively



6.2.4 Pre-pruning process curves



6.2.5 Comparisons on AUC performances of different pre-pruning methods on different datasets

Data	Pre-pruning Method	AUC
Title	default	0.7070
Text	default	0.8139
	GridSearchCV	0.8531
	Pre-pruning	0.9392
Whole	default	0.8026
	GridSearchCV	0.8794
	Pre-pruning	0.9396

6.3 Summary of all models built without feature selection/hyperparameter tuning

Algorithm	Data used	Without feature selection/hyperparameter tuning			
		AUC	Fit Time(s)	Prediction Time(s)	Total Time(s)
Logistic Regression	Length	0.65072552	0.02696418	0.00230312	0.02926731
	Title	0.70949411	0.02677536	0.00053477	0.02731013
	Text	0.95554707	0.29041004	0.00202131	0.29243135
	Whole	0.95546174	0.27897715	0.00227165	0.28124880
Naive Bayes	Title	0.69922697	0.006409645	0.00114846	0.00755810

	Text	0.90695223	0.00689244	0.00629830	0.01319075
	Whole	0.90851929	0.00960993	0.00389552	0.01350545
Decision Tree	Title	0.70655027	0.00642157	0.00130963	0.00773120
	Text	0.80411470	0.81785798	0.00407362	0.82193160
	Whole	0.80715772	0.93553734	0.00391436	0.93945169
Random Forest	Title	0.71471186	0.26239467	0.05952978	0.32192445
	Text	0.96526304	2.86606025	0.15991163	3.02597188
	Whole	0.96345615	2.86038851	0.17120981	3.03159833
SVM	Title	0.67914468	0.75113583	0.08097768	0.83211350
	Text	0.92843668	20.0444863	2.50348449	22.5479708
	Whole	0.93560085	20.93454218	2.68020606	23.6147482
KNN	Tite	0.62550506	0.00296712	0.20567131	0.20863843
	Text	0.80551802	0.01213073	0.85136414	0.86349487
	Whole	0.81008542	0.00454521	0.81903505	0.82358027
Neural Network	Tite	0.87634104	5.07632184	0.15237951	5.22870136
	Text	0.93229371	4.55905676	0.12110066	4.68015742
	Whole	0.93191451	4.05947471	0.12069631	4.18017101

6.4 Top 20 informative words comparison

Rank	Logistic	Naïve Bayesian	Decision Tree
1	2016	washington	say
2	comments	2016	gop
3	article	column	2016
4	establishment	offer	october
5	october	canada	article
6	28	speaking	hillary
7	source	violent	obama
8	uk	short	nomination
9	share	obama	republican

10	corporate	arrest	didn
11	democratic	addition	rubio
12	november 2016	wind	november
13	researcher	september	com
14	oct	hear	trump
15	print	ago	print
16	illegal	wealth	email
17	hillary clinton	indicate	ted
18	ago	conced	nominee
19	comments	et	2012
20	trump win	survey	house

orange indicates the word appears three times in three algorithms, green indicates appears twice