A Machine Learning Approach to Fake News Detection

Introduction

This project is conducted with the goal of gaining a better understanding of fake news detection. By utilizing relevant data of statements and speakers from various contexts, this project will build and optimize a supervised machine learning model to predict the truth values for statements. The detection of fake news is important because the rapid growth of fake news is becoming a cause of concern for all aspects of life, from politics to financial market and environmental protection. Election results, for example, can be manipulated by untruthful claims and deliberate lies from political candidates to change the opinions of voters. Media outlets, including traditional media like newspapers, social media like Twitter, and other platforms, have the responsibility to examine the quality of news contents and identify problematic statements before they reach wide audiences.

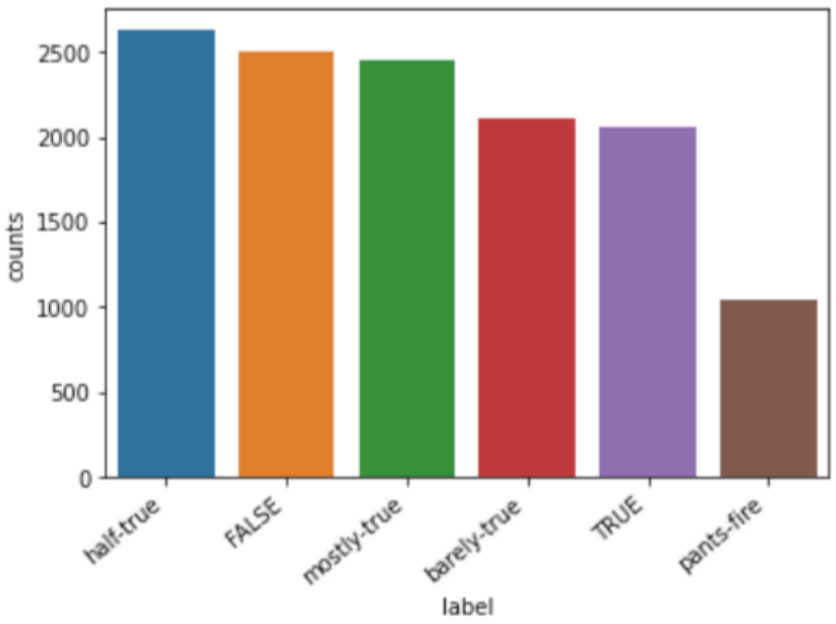
Researchers have done similar studies regarding fake news detection in recent years. Some used supervised models. The paper “Natural Language Processing based Hybrid Model for Detecting Fake News Using Content-Based Features and Social Features”, for example, built a ml model based on NLP techniques and achieved an average accuracy of 90.62% with a F1 Score of 90.33% on a standard dataset. Others used unsupervised deep learning models. The paper “Fake News Identification on Twitter with Hybrid CNN and RNN Models”, for example, used a hybrid of CNN and long-short term RNN models and achieved an 82% accuracy. Research in this area has triggered interests from social media. Twitter, for instance, acquired Fabula AI, a startup using machine learning to detect the spread of misinformation, to help identify fake news.

The intended audiences for this project include various media outlets and fact-checking websites. Media outlets, both traditional media and social media, should pay attention to the project because its machine learning model for predicting fake news may potentially increase the efficiency in detecting fake news from a large volume of statements, which could not be done manually on a large scale. In addition, fact-checking websites, such as Fact Checker from The Washington Post and FactCheck.org from the University of Pennsylvania, should also be interested in this project because the model could also enhance their ability in verifying the factual accuracy of the statements of politicians, which enables them to flag untruth statements and debunk more misinformation faster.

Dataset Description

The dataset used by this project is called “Liar: A Benchmark Dataset for Fake News Detection and was gathered by politifact.com. The dataset is allowed to be used for research purposes only, and the original sources retain the copyright of the data.

The dataset has 12791 American political statements in total and its variables include the content of a statement, the subject of a statement, the context of a statement, the name of the speaker, the speaker’s job title, the state of the speaker, and the party affiliation of the speaker. The dataset also has the total credit history counts of each speaker, including current statements, which are the counts of barely true statements, false statements, half-true statements, mostly true statements, and lies. Besides, there are six labels representing the truth values of statements, including half-true, false, mostly-true, barely-true, true, and pants-fire (lies). Their distribution are as follows.



half-true	2627
FALSE	2507
mostly-true	2454
barely-true	2103
TRUE	2053
pants-fire	1047

Figure 1. Distribution of truth value labels

A preliminary exploratory data analysis offers a basic understanding of the variables. Before that, however, I first examined the missing values in the dataset and found that two statements don't have values for all variables, so I dropped those two entries. I also found that there are some missing values for context (131) and many missing values for title and state (3567 and 2749). Since removing the missing values will greatly reduce the size of training data, I decided to not drop them and fill them with "missing".

The "statement" variable has the texts of statements. To transform this variable into a usable feature, I created a new variable "statement_length" for statements lengths. The distribution of "statement_length" shows that while its distribution closely resembles a normal distribution under 500, it has a very high range of distribution that extends to more than 3000. To avoid numerical instability, I rescaled "statement_length" to log scale (base e) and its distribution became more like normal distribution curve.

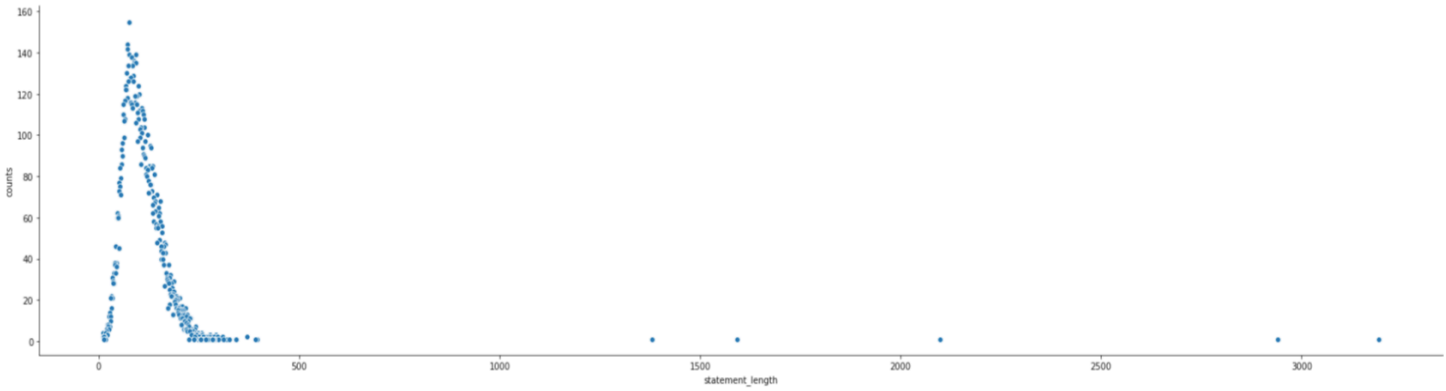


Figure 2. Distribution of statements lengths before log scale

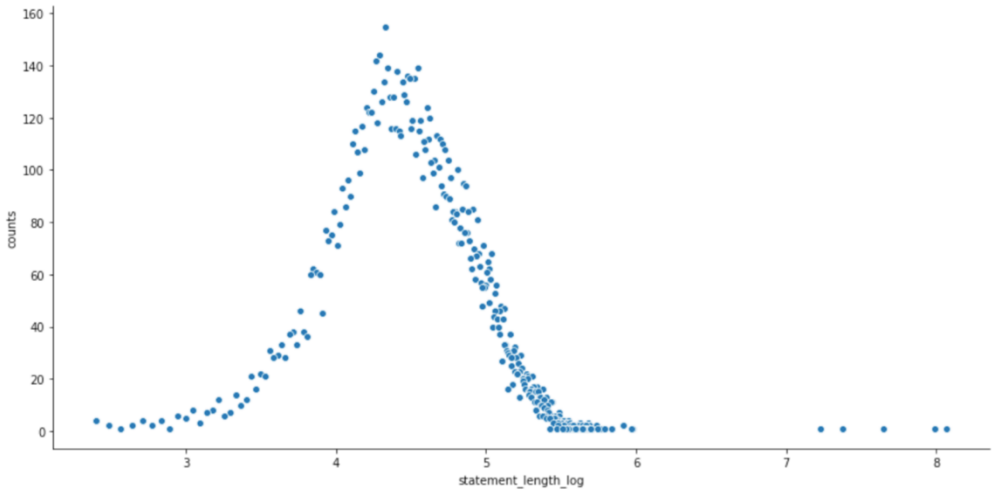


Figure 3. Distribution of statements lengths after log scale

The "subject" variable has the subjects of statements. There are 4534 different subjects and the top five most mentioned ones are health care, taxes, education, elections, and immigration. However, a lot of statements included more than one subject. To take this factor into account, I created a new variable "subject_count" for the number of subjects. The distribution of "subject_count" shows that while the great majority of the politicians referred to only 1-3 subjects in their statements, some of them referred to as many as 20 subjects. Both “subject” and “subject_count” variables will be used as features.

health-care	474
taxes	356
education	309
elections	304
immigration	303

Figure 4. Distribution of top 5 subjects

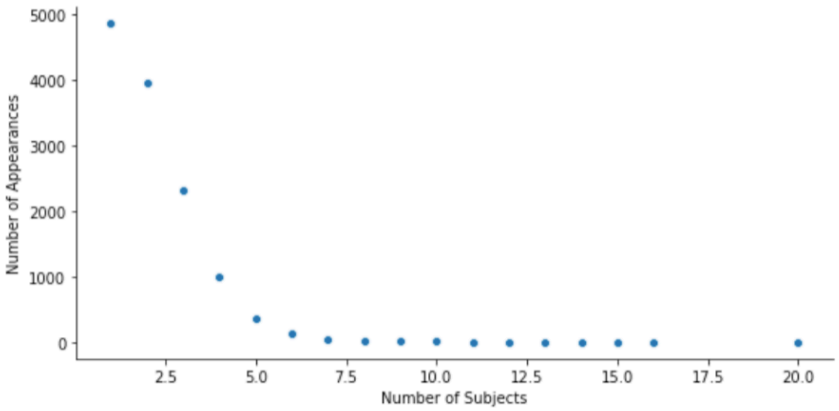


Figure 5. Distribution of number of subjects in a statement

The "speaker" variable has the names of speakers. There are in total 1355 speakers in the dataset and the top five speakers are Barack Obama, Donald Trump, Hillary Clinton, Mitt Romney, and John McCain. However, while certain speakers appear many times, the majority of speakers only appear one to two times. Therefore, I decided to not use speaker names as a feature. Moreover, the names of the existing speakers won't have much predictive power for the new speakers which will be added into the future dataset.

barack-obama	611
donald-trump	343
hillary-clinton	297
mitt-romney	212
john-mccain	189

Figure 6. Top 5 speakers

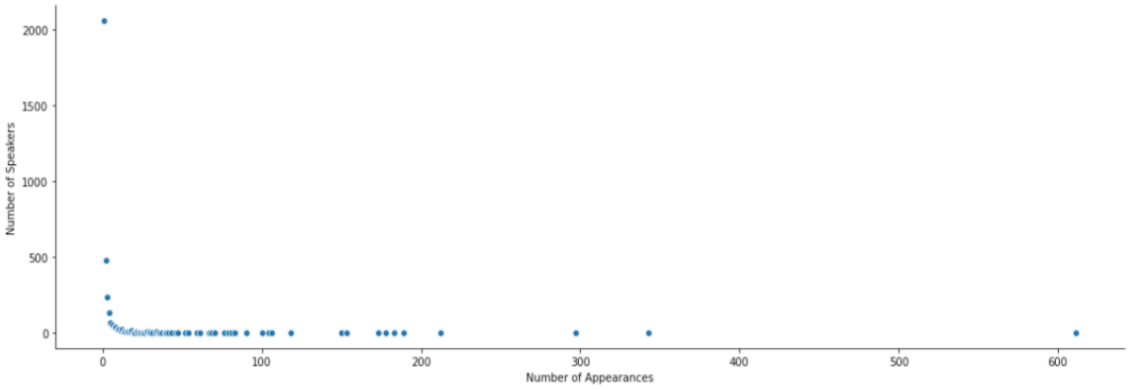


Figure 7. Distribution of number of appearances of speakers

The "title" variable has the titles of speakers. I transformed all values to lower case, as certain titles are classified as different titles just because of the differences in uppercase and lowercase, such as "U.S. Senator" and "U.S. senator". After transformation, there are 1279 different speaker titles including “missing”, and the top five are U.S. Senator, President, Governor, President-Elect, and U.S. Representative. I will use "title" as a feature because a politician's title might be related to his/her tendency to lie.

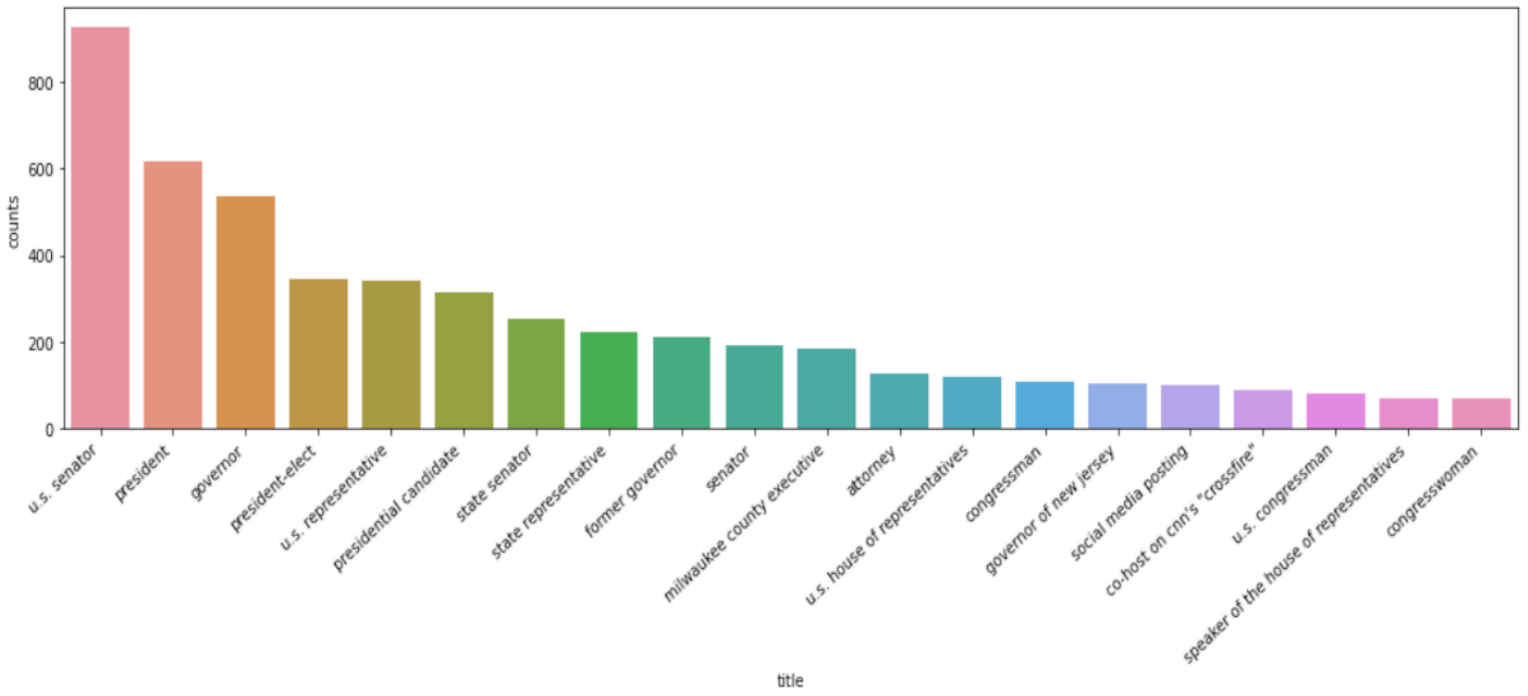


Figure 8. Distribution of top 20 speaker titles

The "state" variable has the locations of speakers. I transformed the different spellings of the same states to the same ones because some states have different spellings, like Washington, D.C. and Washington DC. There are 64 unique states after transformation. I also found several locations that don't belong to the United States, including Russia, China, United Kingdom, and Qatar. I renamed them as "other country". The distribution

of states in Figure 9 shows that Texas, Florida, Wisconsin, New York and Illinois have the greatest number of speakers in this dataset. I will use state as a feature because the location of a speaker might be related to his/her tendency to tell the truth.

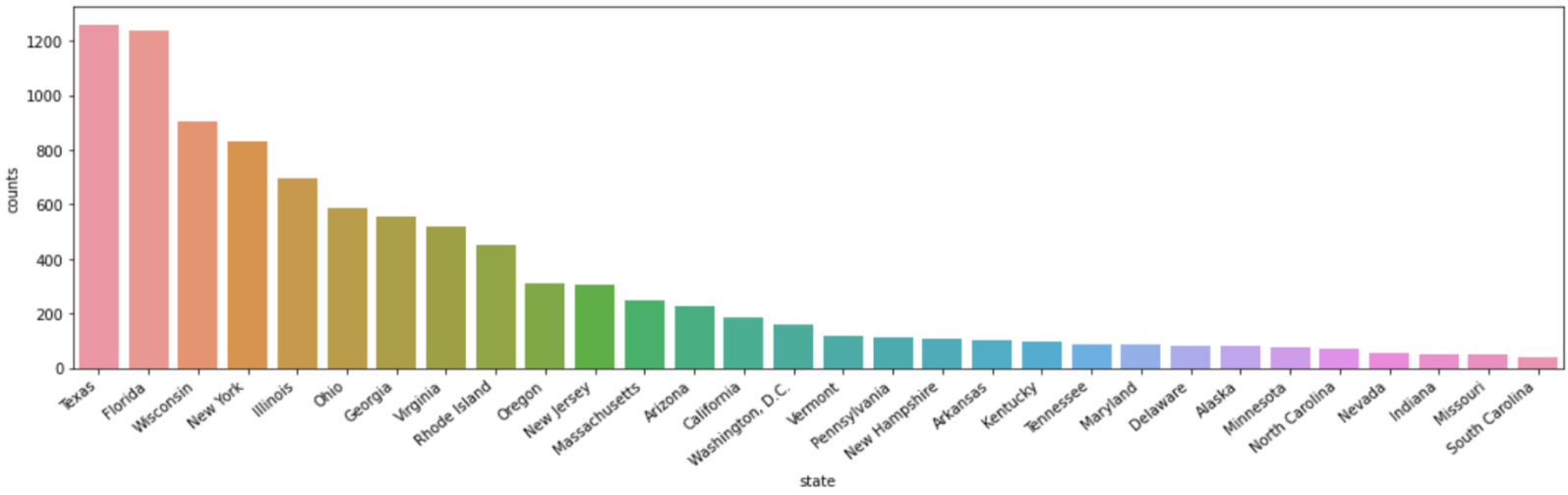


Figure 9. Distribution of top 30 states

The "party" variable has the parties that speakers belong to. While most speakers are republican or democrat, a much smaller number of speakers come from various other smaller parties like the independent party or have no party affiliation. To normalize the party feature, I renamed all parties besides republican and democrat to "other parties". Figure 10 shows that the distribution of parties became more even after the adjustment.

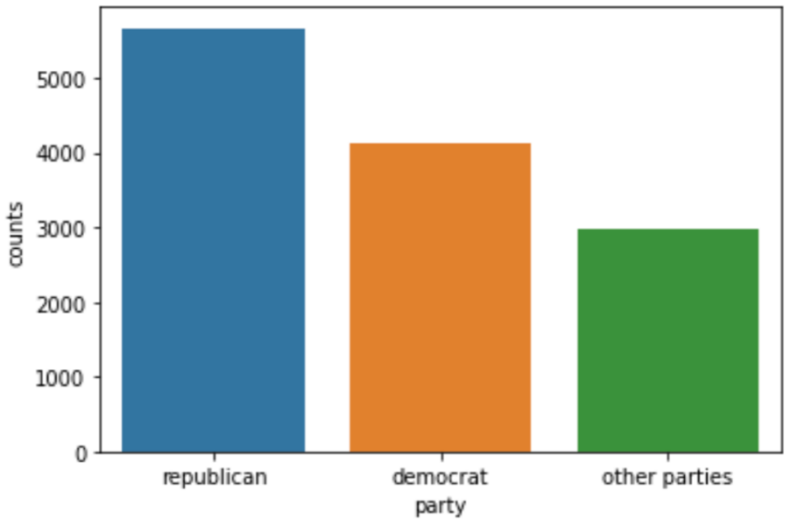


Figure 10. Distribution of parties

The "context" variable has the context of statements. While news release, interviews and press release each has around 300 statements, the majority of the contexts only have one statement. Still, I decided to use the "context" variable as a feature because the contexts of the statements may have impact on their truth values. A statement from press release, for example, might be more truthful than a statement from Tweet.

a news release	309
an interview	286
a press release	282
a speech	259
a TV ad	222

Figure 11. Top 5 contexts

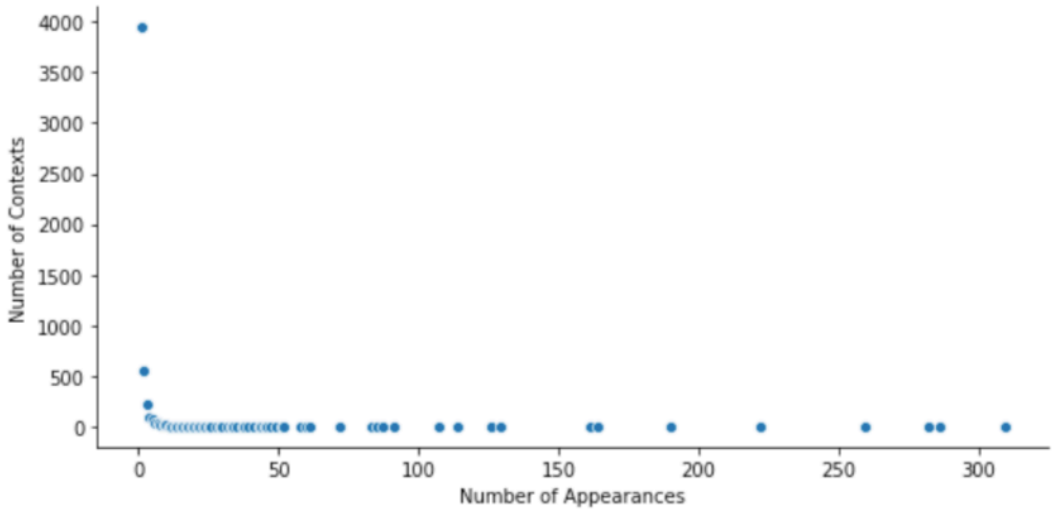


Figure 12. Distribution of number of appearances of contexts

The credit history variables record the truth values of the past statements of a speaker, including "barely_true", "not_true", "half_true", "mostly_true" and "lie". Figure 13 shows that while the distribution of these truth values is relatively even, politicians tell fewer lies but more not true (false) and half-true statements. I will use the credit history variables as features because the history of a politician telling truth and lies may have important indications regarding whether this politician will tell truth in the future statements.

Figure 14 shows the correlations among the truth values. While "barely true", "not_true", "half_true", and "mostly_true" have relatively stronger correlations with each other (0.7-

0.8), "lie" has different level of correlations with the rest. Among all truth values, "lie" is most strongly correlated with "not_true" (around 0.7) and moderately strongly correlated with "barely true" (around 0.5). It is not correlated with "half_true" and "mostly_true".

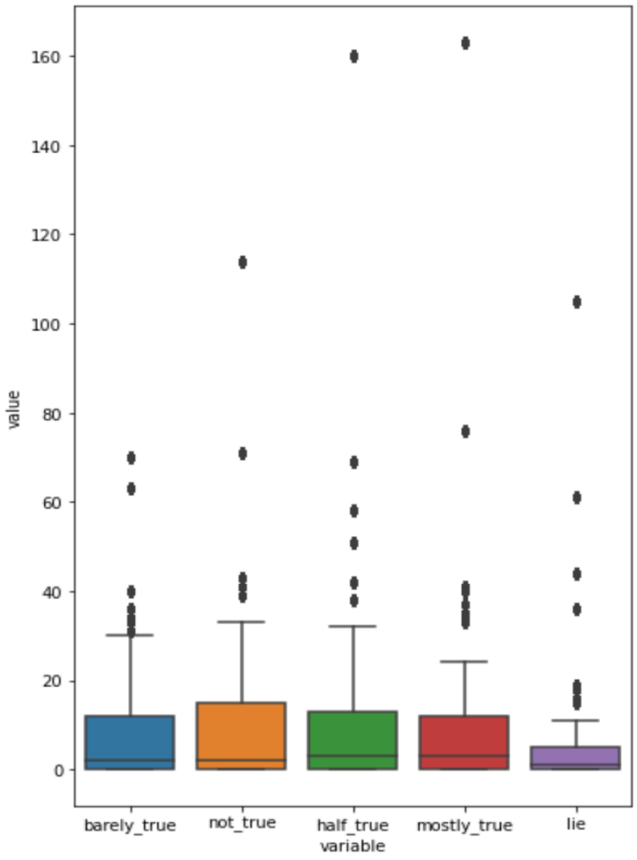


Figure 13. Distribution of truth values

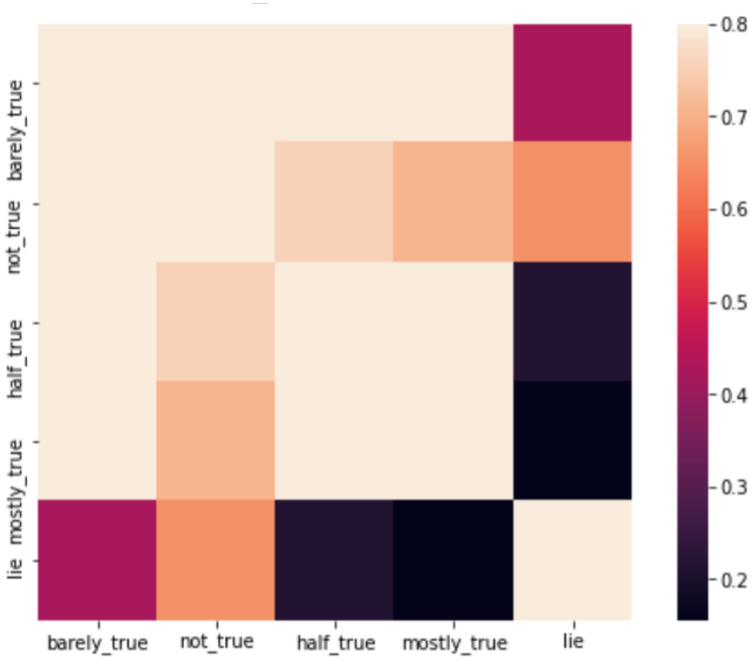


Figure 14. Correlations among truth values

To conclude, among all variables in the dataset and the new variables I created, I used “statement_length_log”, ‘subject”, “subject_count”, and “context” as features related to statements, “title”, “state”, and “party” as features related to speakers, and “barely_true”, “not_true”, “half_true”, “mostly_true” and “lie” as features related to credit history. I used the “label” variable as the labels for prediction. However, considering the difficulties of predicting six labels, I also created a new column “label 2” to classify these six labels

into two categories to observe the subsequent changes in model performance. A statement will be labeled as 1 if its label is “true”, “mostly-true”, and “half-true” because they are closer to truth, and a statement will be labeled as 0 if its label is ‘barely-true”, “false”, and “pants-fire”. Their distribution, as shown in Figure 15, is relatively even.

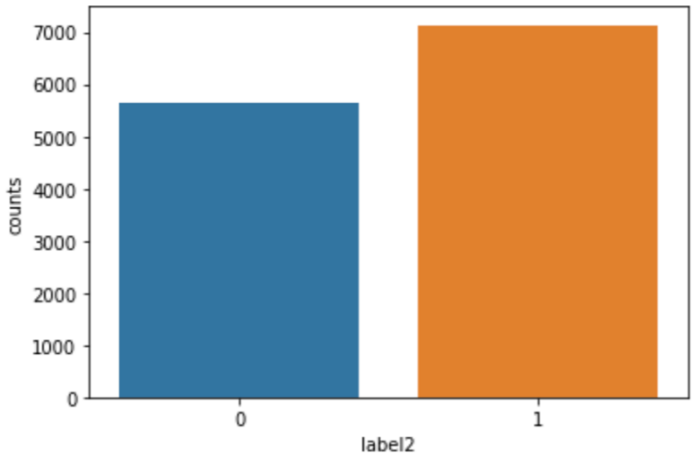


Figure 15. Distribution of label2

Primary Task

The primary task of this project includes two part. The multi-class classification part will build a machine learning model to predict six labels and the binary classification part will predict two labels. As a recapitulation, their distributions are respectively as follows.

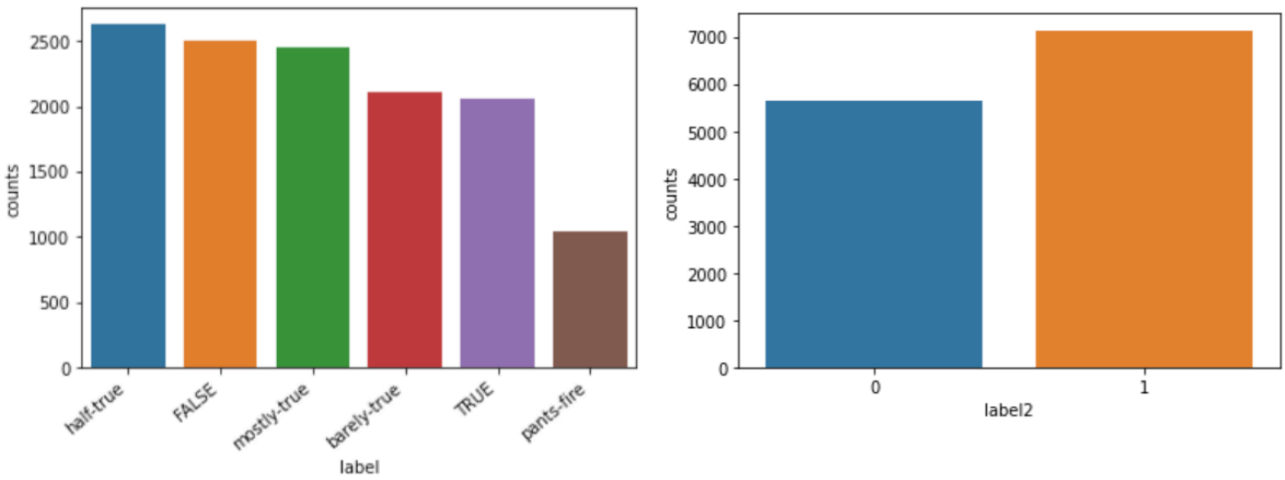


Figure 16. Distribution of multi-class classification labels and binary classification labels

Multi-Class Classification

To avoid overfitting and reporting inaccurate results, I divided my data into 80% training data and 20% testing data and used 5-fold cross-validation for hyperparameter tuning. Given the difficulties of predicting six labels with only around 8,000 entries of training data, I set the target for performance of my model to a 40% accuracy score.

For all the features I described above, I tested different subsets of features with decision tree to evaluate and select the features to use, including features related to statements, features related to speakers, as well as features related to credit history. The results show that among all the subsets, features related to statements, including "subject", "subject_count", "context", and "statement_length_log" are the most useful for model prediction, which achieved a 99.9% accuracy by themselves compared with 37.1% for speaker features and 46.8% for credit features. The accuracy score got to 100% when statement features are used together with speaker features, credit features, and when three subsets of features are used together. While statement features are able to reach a high accuracy by themselves, I still decided to use all three subsets of features jointly since they are able to achieve a 100% accuracy score with the least amount of leaves.

Given that the multi-class classification task has six labels, I tried decision tree classifier and random forest classifier for the model with 5 folds cross-validation to compare their performances. The result shows that in the absence of hyperparameter tuning, although the accuracy of decision tree is slightly better than random forest (38.60% compared with 38.47%), the performances of two classifiers are very similar. Thus, I decided to adjust hyperparameters for both and compare their performances again after tuning.

1) Decision Tree Classifier

For the decision tree classifier hyperparameter tuning, I first used Randomized-SearchCV to create a parameter grid to sample from during fitting, because it allows selecting at random to sample a wide range of hyperparameter values instead of trying every single combination. I set the number of iterations (n_iter) to 60 and the number of folds for cross validation (cv) to 5, because as much as I want to cover wider search space and reduce the chance of overfitting, I need to make sure the run time is reasonable. The hyperparameters to tune include:

- criterion: function to measure the quality of a split, including gini and entropy.
- min_impurity_decrease: a node will be split if this split induces a decrease of the impurity greater than or equal to this value.
- min_samples_split: the min number of samples required to split an internal node
- max_depth: the maximum depth of the tree.
- min_samples_leaf: the min number of samples required to be at a leaf node.

The best random search model returns a 40.54% accuracy on the test set, which has a 1.94% increase compared with the decision tree model before tuning.
The best hyperparameters are:

- criterion = "gini"
- min_impurity_decrease = 0.0001
- max_depth = 25
- min_samples_leaf = 0.01
- min_samples_split = 0.001

The result from random search allows me to narrow down the range of values for each hyperparameter. I then used GridSearchCV to specify the best values and evaluate all combinations to further improve model performance. I still set the number of folds for cross validation to 5. The best grid search model returns a 40.54% accuracy on the test set, which is the same as the last time. It seems that the performance has reached its best. The best hyperparameters are:

- criterion = "gini"
- min_impurity_decrease = 0.0001
- max_depth = 20
- min_samples_leaf = 0.01
- min_samples_split = 0.001

2) Random Forest Classifier

For the random forest classifier hyperparameter tuning, I first used Randomized-SearchCV with 80 iterations and 5 folds for cross validation. I increased the number of iterations because I want to cover a wider range of search space. The hyperparameters to tune include:

- n_estimators: the number of trees in the forest.
- max_depth: the maximum depth of the tree.
- max_features: the number of features to consider when looking for the best split
- min_samples_leaf: the min number of samples required to be at a leaf node.
- min_samples_split: the min number of samples required to split an internal node

The best random search model returns a 39.05% accuracy on the test set, which has a 0.58% increase compared with the random forest model before tuning. The best hyperparameters are:

- n_estimators = 90
- max_depth = 45
- max_features = 'sqrt'
- min_samples_split= 5
- min_samples_leaf = 1

I then used GridSearchCV with 5 folds for cross validation to further improve model performance. The best grid search model returns a 40.3% accuracy on the test set, which has a 1.25% improvement compared with the last optimization. The best hyperparameters are:

- n_estimators = 70
- max_depth = 35
- max_features = 'auto'
- min_samples_split= 2
- min_samples_leaf = 2

To conclude, after hyperparameter tuning, the accuracy scores of decision tree and random forest classifier increased by 1.94% and 1.83% respectively. The performance of decision tree classifier, with a 40.54% accuracy score, is still slightly better than random forest classifier, which has a 40.3% accuracy score.

Binary Classification

Similar to the multi-class classification part, to avoid overfitting and reporting inaccurate results, I divided my data into 80% training data and 20% testing data and used 5-fold cross-validation for hyperparameter tuning. Given that predicting two labels should be much easier than predicting six labels, I decided to increase the target for performance. However, considering the small size of training data and other limitations of the dataset, I set the target for performance of my model to a 70% accuracy score.

I also tested different subsets of features with decision tree to select the features to use. The results again show that features related to statements are the most useful among all features, which achieved a 99.9% accuracy compared with 67.2% for speaker features and 73.8% for credit features. The accuracy reached 100% when statement features are used with speaker features or credit features, and when three subsets are used together. Due to the same reason as above and for a better comparison with multi-class classification, I still decide to use all three subsets of features jointly for modeling.

I first tried decision tree classifier and random forest classifier with 5 folds cross-validation to compare their performances. The result shows that in the absence of hyperparameter tuning, the accuracy score of random forests is only slightly better than decision tree (69.44% compared with 67.78%). Thus, I had to tune hyperparameters for both and compare their performances again after adjusting. However, it is worth noticing that the accuracy of both classifiers already improved greatly compared with multi-class classification (40.5% for decision tree and 40.3% for random forest after tuning), which means reducing the number of labels for prediction can improve model performance.

1) Decision Tree Classifier

For the decision tree classifier hyperparameter tuning, I first used Randomized-SearchCV with 60 iterations and 5 folds cross validation to sample a wide range of hyperparameter values. The hyperparameters to tune are the same as the decision tree for multi-class classification. The best model from random search returns a 69.59 % accuracy score on the test set, which has a 1.81% increase compared with the decision tree before tuning. The best hyperparameters are:

- criterion = "gini"
- min_impurity_decrease = 0.001
- max_depth = 30
- min_samples_leaf = 0.001
- min_samples_split = 0.001

I then used GridSearchCV with 5 folds for cross validation to specify the best values and evaluate all combinations to further improve model performance. The best grid search model returns a 69.59% accuracy score on the test set, which is the same as the result of best random search. It seems that the performance has reached its best. The best hyperparameters are:

- criterion = "gini"
- min_impurity_decrease = 0.001
- max_depth = 25
- min_samples_leaf = 0.001
- min_samples_split = 0.001

2) Random Forest Classifier

For the random forest classifier hyperparameter tuning, I first used Randomized-SearchCV with 100 iterations and 5 folds cross validation. The hyperparameters to tune are the same with the random forest model for multi-class classification. The best random research model returns a 69.78 % accuracy score on the test set, which has a 0.34 % increase compared with the random forest model before tuning. The best hyperparameters are:

- n_estimators = 160
- max_depth = 80
- max_features = 'sqrt'
- min_samples_split= 8
- min_samples_leaf = 1

I then used GridSearchCV with 5 folds for cross validation to specify the best values and evaluate all combinations to further improve model performance. The best grid search model returns a 70.3% accuracy score on the test set, which has a 0.52% improvement compared with the result from random search.
The best hyperparameters are:

- n_estimators = 150
- max_depth = 65
- max_features = 'auto’
- min_samples_split= 8
- min_samples_leaf = 1

To conclude, after hyperparameter tuning, the accuracy scores of decision tree and random forest classifier increased by 1.81% and 0.86% respectively. The performance of random forest classifier, with a 70.3% accuracy score, is still slightly better than decision tree classifier, which has a 69.78% accuracy score.

Final Model

After tuning the hyperparameters for decision tree classifier and random forest classifier respectively for multi-class classification and binary classification, all models achieved different levels of improvement and their final results can be summarized as follows.

	Multi-class classification (six labels)	Binary classification (two labels)
Decision tree classifier	40.54%	69.59%
Random forest classifier	40.3%	70.3%

I select random forest classifier for binary classification as my final model because it has the best performance. Its tuned hyperparameters are as follows:

- n_estimators = 150
- max_depth = 65
- max_features = 'auto’
- min_samples_split= 8
- min_samples_leaf = 1

I trained the final model again using 10-fold cross validation. The model achieved a 71.3% mean accuracy score across 10 folds and a 70% accuracy on the test set.

The summary of the model performance is as follows.

```
=== Confusion Matrix ===
[[ 587  565]
 [ 194 1212]]


=== Classification Report ===
              precision    recall  f1-score   support

           0       0.75      0.51      0.61      1152
           1       0.68      0.86      0.76      1406

    accuracy                           0.70      2558
   macro avg       0.72      0.69      0.68      2558
weighted avg       0.71      0.70      0.69      2558


=== All Accuracy Scores ===
[0.71618452 0.72947615 0.71540266 0.71774824 0.69194683 0.71383894
 0.70992963 0.71149335 0.71540266 0.71048513]


=== Mean Accuracy Score ===
Mean accuracy Score - Random Forest:  0.713190811972871
```

A closer examination of the confusion matrix shows that the final model is better at predicting truth values (1) than false values (0). Most noticeably, as shown in Figure 17, the model only had around 50% accuracy when predicting false values.

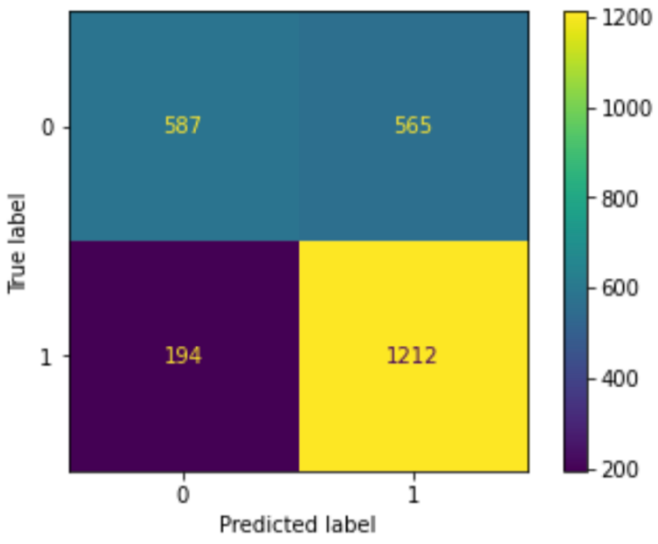


Figure 17. Confusion matrix for the final model

To have a better understanding of where my model made mistakes, I further examined the distribution of six truth values among right and wrong predictions. The result shows that among labels in the wrong predictions, there are disproportionately more barely-true and false values than other values compared with distribution in right predictions, indicating that the model is weaker at predicting these two types of statements.

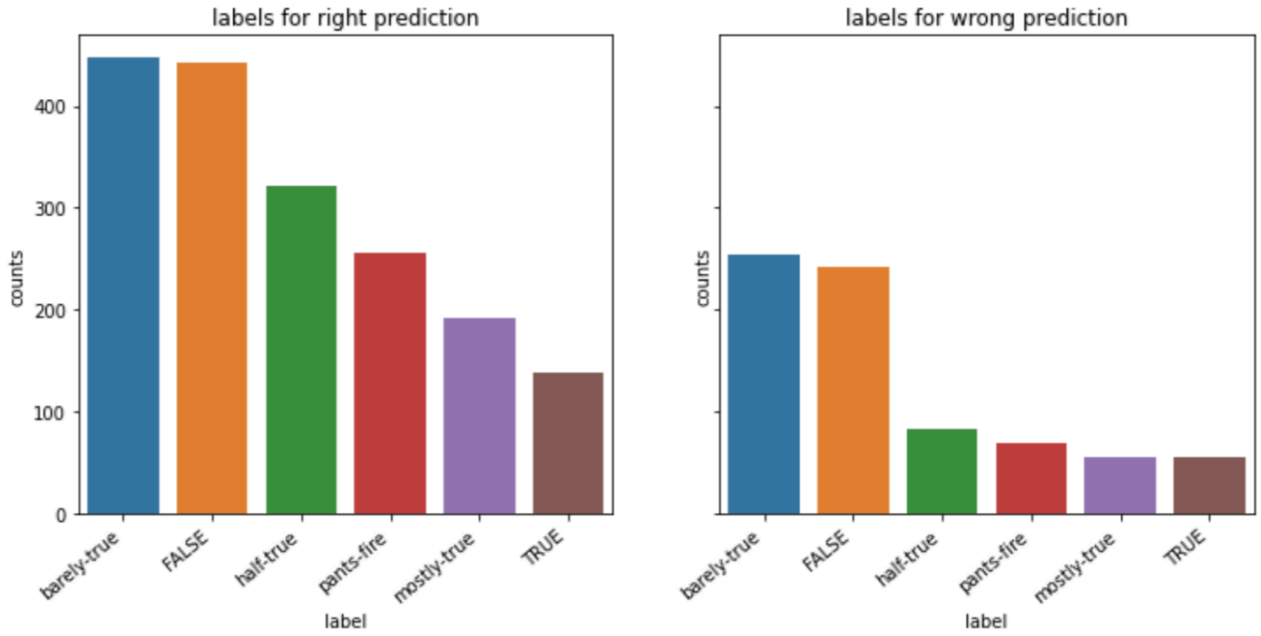


Figure 18. Distribution of six truth values among right and wrong predictions

I also examined the distribution of top 10 subjects among right and wrong predictions to see what subjects have the most errors. As shown in figure 19, subjects including health care, taxes, education, and elections have proportional distributions among the right and wrong predictions, which means that the classifier is at least equally good at predicting those subjects. However, the classifier is weaker at predicting immigration, economy, crime, and transportation because the proportions of these subjects among the wrong predictions are larger than their proportions among the right predictions.

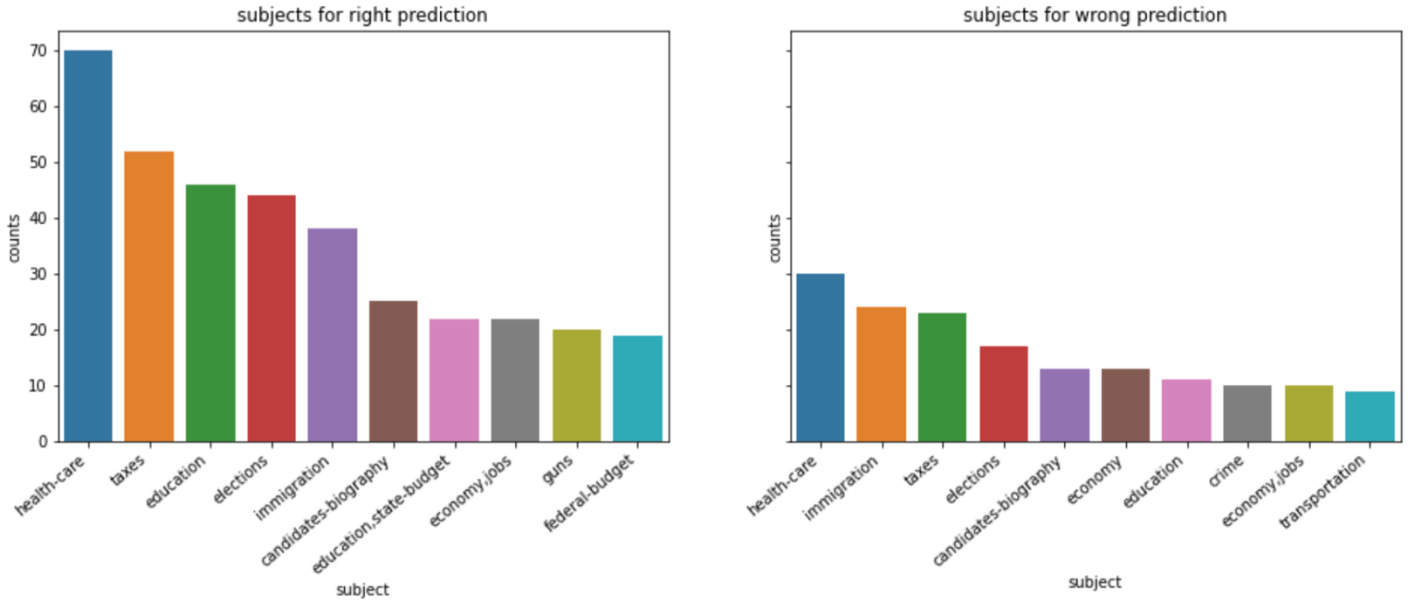


Figure 19. Distribution of subjects among right and wrong predictions

Finally, I examined the distribution of top 10 contexts among right and wrong predictions to see in what contexts my classifier is weaker at predicting. The result shows that while my classifier is good at predicting a press release, a news release, and a speech, it is worse at predicting an interview, a TV ad, a debate and a Facebook post.

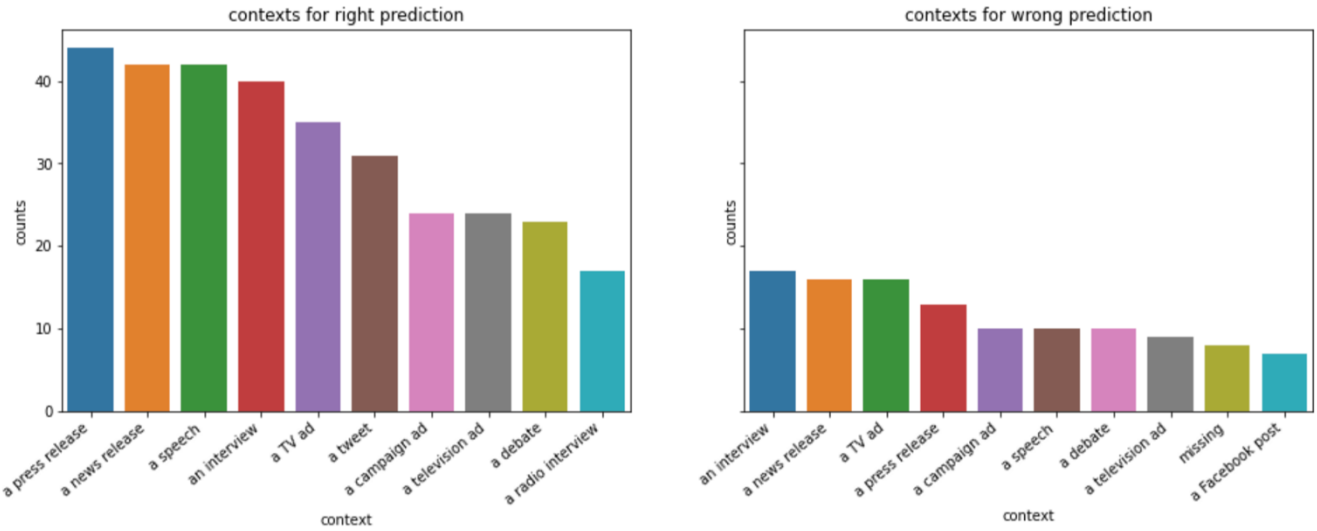


Figure 20. Distribution of subjects among right and wrong predictions

In conclusion, my final random forest model that predicts binary labels achieved a 70% accuracy after hyperparameter optimization. It is weaker at predicting false values than truth values, and it makes more errors with certain subjects and in certain contexts. The performance of the model only just reached the goal (70%) set for this project. However, given that the classifier is weaker at predicting false values, it requires more extensive manual checks from media and fact-checking to verify the truth values of news to find the fake news if the model should be put into use. The hyperparameter tuning process and model optimization are important and have a great impact because even a slight improvement in model performance can greatly reduce the amount of manual work.

Extension Task

The extension task of this project conducts a fairness audit on the classifier by separating out statements by political party to measure and describe bias of the classifier by party. This is important because it measures the disparate impact of the classification system on different parties. If the classifier is found to be biased against certain parties, for instance, by having a higher false positive rate or false negative rate, then the classifier is proved to be unfair and should not be put into professional use.

Politicians and their political parties will care about the fairness of the classifier because they don’t want to be unfairly discredited during their campaigns. Media outlets as well as fact-checking websites will also avoid using a biased classification system because they do not want to be accused of being partisan. Last but not least, social media users should also expect the classifier to be fair because they want to receive right information about issues and be correctly informed when there is fake news regardless of parties.

I only included the Republican party and the Democratic party in the fairness audit because they are the two largest parties in the United States, and they have the most data entries in the dataset. Small parties are excluded from the fairness audit because there are too many of them and they have fewer data entries. The classifier used for this extension task is the optimized random forest classifier that predicts binary outcomes.

The result of fairness audit shows that while the false positive rate of the classifier is 31.8%, the false negative rate of the classifier is only 24.8%. A closer examination of false positive rate and false negative rate by party shows that while the false positive rate for the republican party is higher than that of the democratic party by about 10%, the false negative rate for the republicans is higher than that of the democrats by a bit more than 10%. This means that the classifier has a higher tendency to not only falsely predict a statement to be true (while it is false) for republicans than for democrats, but also falsely predict a statement to be false (while it is actually true) for republicans.

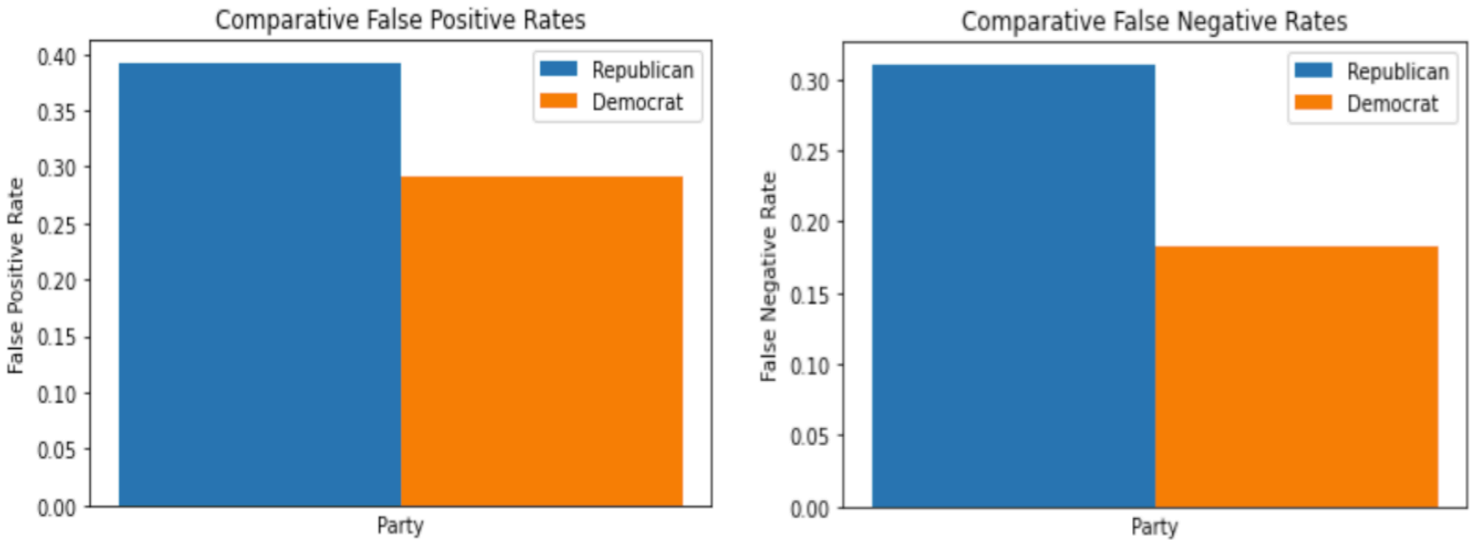


Figure 21. Comparative False Positive Rates and False Negative Rates for Republicans and Democrats

To further explore the reasons that contribute to the discrepancy in false negative rates, I examined the distribution of the two parties and the binary labels in the dataset. Figure 22 shows that while there is an equal distribution of truth and false statements for the Republican party, the number of truth statements far exceeds the number of false statements for the Democratic party. This indicates that the dataset itself is biased against the Republican party by including more false statements of republicans than democrats, which could contribute to the unfair predictive performance of the classifier.

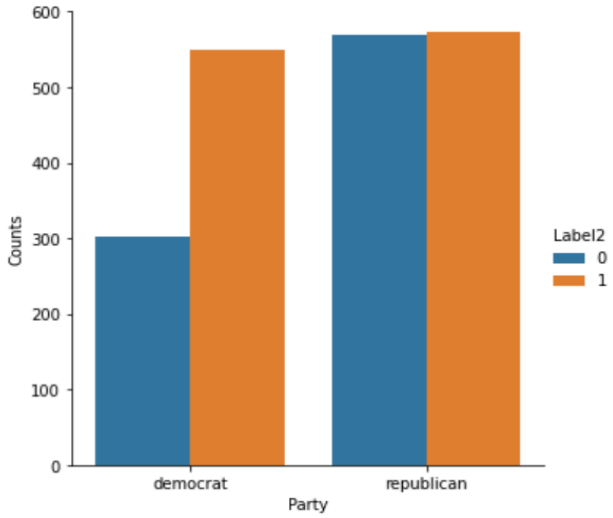


Figure 22. Distribution of parties and labels

The classifier, therefore, should not be put into professional use before its biasedness is resolved. The difference in false negative rates and false positive rates between the two parties is problematic because it makes it harder for media and fact-checking websites to distinguish true statements from false statements for republican speakers. Moreover, the classifier is also likely to unfairly discredit speakers from the Republican party even when their statements are true or giving them unfair credits while their statements are false. Therefore, the use of the biased classifier could be misleading to the public and media while contributing to unjust advantages and disadvantages during campaigns.