Machine Learning Engineering Capstone Project

Amrita Sharma
May 30th, 2020

Definition

Project Overview

Given these uncertain times of a global pandemic, recession and growing unemployment we are exposed to number of news sources on various means such as TV, internet and phone. However, the validity of news sources is a growing concern as the uncertainty of pandemic and recession grows every day. The project focuses on to create a Machine Learning model to determine if a news article is fake news or true news. Previous research [13] [14] has already been done on this domain which are referenced for this project.

My inspiration to create an algorithm to identify true news articles is very important for people to make important business and personal decisions. Because fake news can lead to bad decisions and false rumors which might not be good for the economy and country.

Dataset is chosen from Kaggle dataset [1] to determine if a news article is true or fake news. An important aspect to note about the problem is that the news articles are categorized as Fake or True news based on news article content. Thus, the project will focus on news article primarily for feature engineering and model creation.

Problem Statement

The problem is that given a news articles we need to determine if it is true or fake news. There are two datasets given, Fake.csv and True.csv which consists of fake news and true news respectively. When combined both together with corresponding labels we create a set of labeled news articles thus defining the problem as a supervised machine learning problem.

The attributes in the dataset are such as news titles, news subjects and news dates, are used majorly for exploratory data analysis purposes. Since the news labels of true or fake are based on news articles, we will solve the given supervised learning problem using news articles text content.

Metrics

Our aim is to classify as many true news as possible thus, we will use Accuracy, Precision and Recall as the key metrics to evaluate model performance.

Model metrics are defined as follows:

$$\text{Accuracy} = \frac{\mathrm{TP+TN}}{\mathrm{TP+TN+FP+FN}}$$

$$\text{Precision} = \frac{\mathrm{TP}}{\mathrm{TP+FP}}$$

$$\text{Recall} = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$$

Term definitions:

- TP – True Positives (Model correctly predicts that the news article is true)
- FP – False Positives (Model incorrectly predicts that the news article is true)
- TN – True Negatives (Model correctly predicts that the news article is fake)
- FN – False Negatives (Model incorrectly predicts that the news article is fake)

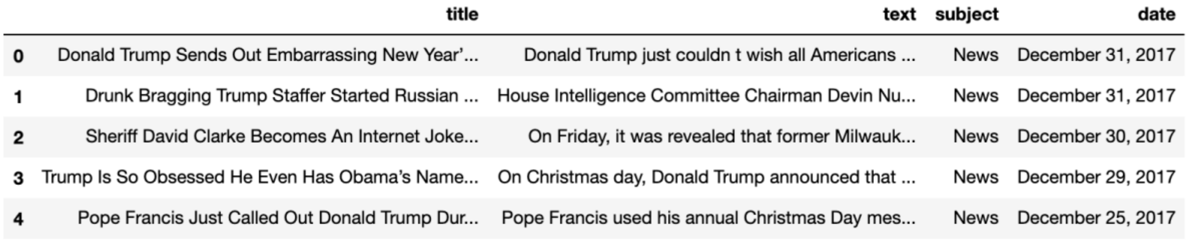
Based on the above, we will focus on a high accuracy i.e. model correctly predicts as many true news articles and fake news articles. We will also focus on high precision since we want to correctly predict as many true news as possible.

<u>Analysis</u>

Data Exploration

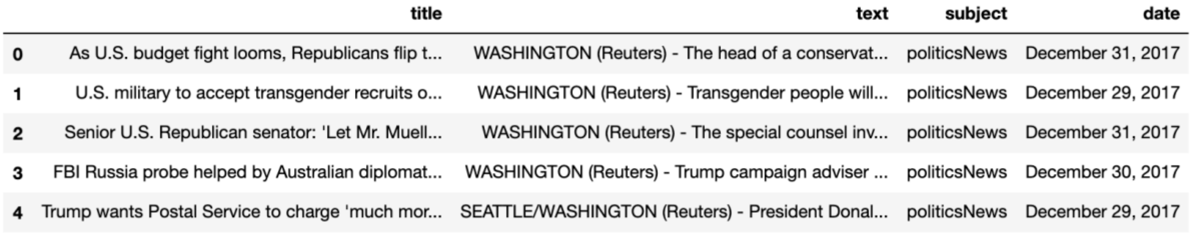
Our dataset consists of 23481 fake news articles and 21417 true news articles thus having almost equal number of articles thus our dataset is almost balanced. Hence, dataset is almost perfectly balanced, and we do not need to perform any data balancing methods. There were no missing values in the dataset hence we did not need to drop any records from real and fake news.

Sample fake news data:



	title	text	subject	date
0	Donald Trump Sends Out Embarrassing New Year’...	Donald Trump just couldn t wish all Americans ...	News	December 31, 2017
1	Drunk Bragging Trump Staffer Started Russian ...	House Intelligence Committee Chairman Devin Nu...	News	December 31, 2017
2	Sheriff David Clarke Becomes An Internet Joke...	On Friday, it was revealed that former Milwauk...	News	December 30, 2017
3	Trump Is So Obsessed He Even Has Obama’s Name...	On Christmas day, Donald Trump announced that ...	News	December 29, 2017
4	Pope Francis Just Called Out Donald Trump Dur...	Pope Francis used his annual Christmas Day mes...	News	December 25, 2017

Sample real news data:



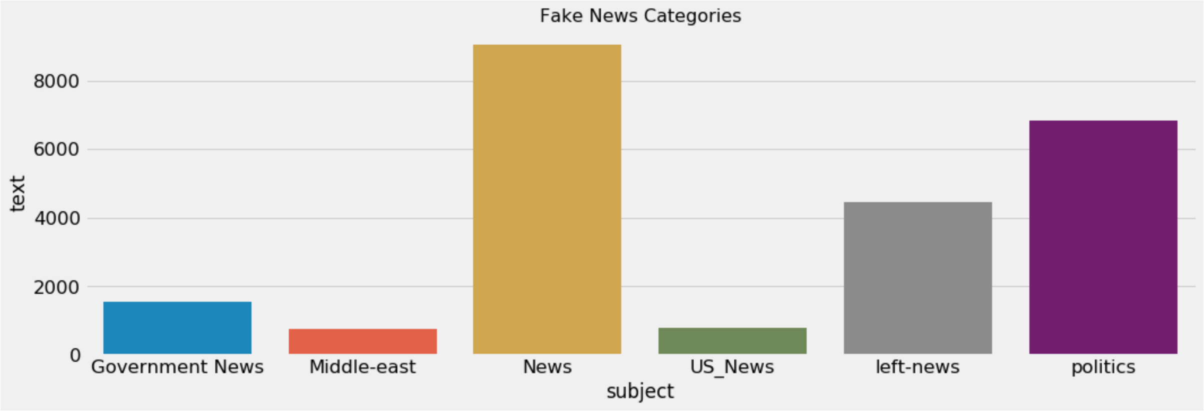
	title	text	subject	date
0	As U.S. budget fight looms, Republicans flip t...	WASHINGTON (Reuters) - The head of a conservat...	politicsNews	December 31, 2017
1	U.S. military to accept transgender recruits o...	WASHINGTON (Reuters) - Transgender people will...	politicsNews	December 29, 2017
2	Senior U.S. Republican senator: 'Let Mr. Muell...	WASHINGTON (Reuters) - The special counsel inv...	politicsNews	December 31, 2017
3	FBI Russia probe helped by Australian diplomat...	WASHINGTON (Reuters) - Trump campaign adviser ...	politicsNews	December 30, 2017
4	Trump wants Postal Service to charge 'much mor...	SEATTLE/WASHINGTON (Reuters) - President Donal...	politicsNews	December 29, 2017

Based on the exploratory data analysis we can see from the sample data that we have attributes such as news titles, news categories, news date and news articles. We have different type of news categories for fake and real news.

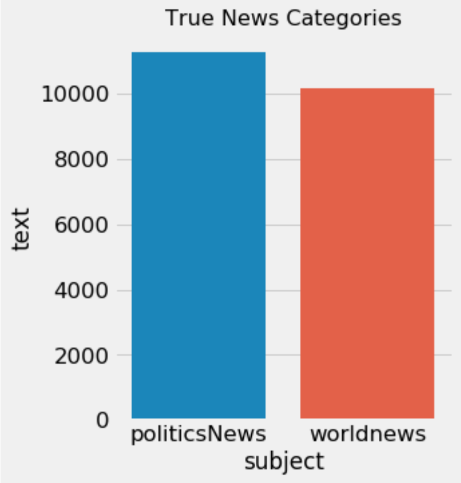
For instance, fake news subject categories are ‘News', 'politics', 'Government News', 'left-news', 'US_News' and 'Middle-east'. Real news categories are 'politicsNews', 'worldnews'.

As we can see for fake news, the news categories are world news separated into different categories such as government news, politics and middle east. However true news has only two broad categories which are politics new and world news.

Exploratory Visualization



Based on the above chart, we can see that “News” category has the highest number of fake news articles. And the second highest category is “politics” for the fake news articles.

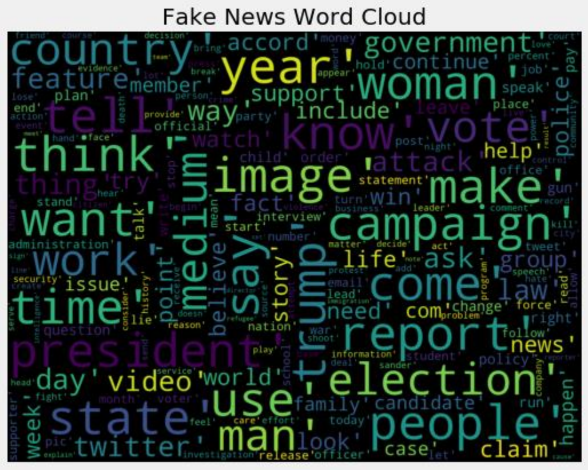


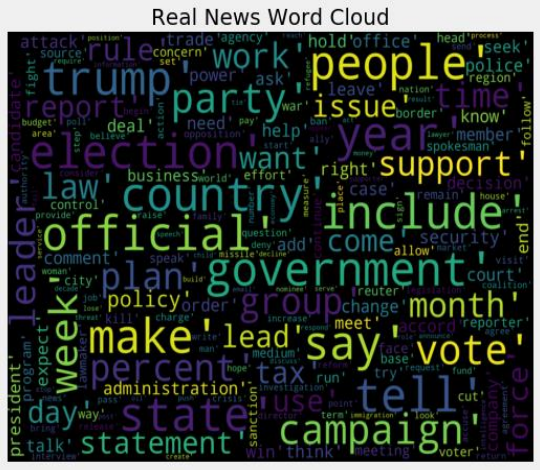
Based on the above chart, we can see that “politics News” category has the highest number of real news articles

To understand news content in-depth in fake and true news we created the word clouds.

For creating the word cloud, news articles were pre-processed using the following steps:

1. Convert news articles to text list
2. Convert text list to text words
3. Convert text words to lemmatized texts – Allowed only nouns and verbs to convert words into root form


Fake News Word Cloud


Real News Word Cloud

Above word clouds are created using top 200 words using fake news and true news articles words respectively.

Based on the above chart, we can wee that fake news word cloud contains words such as trump, election, president, twitter, government, woman, attack, think, country. These words are mostly related to creating opinions about election, campaign and current affairs.

Based on the above chart, we can see that true news word cloud contains words such as people, include, campaign, vote, make, trump, support, right, official, country which indicates that true news is related about current affairs and is more informative in nature.

Above word clouds gives an intuition about what type of words are there in fake vs true news. Although words seem similar however there are contextual information which distinguishes the two types of articles.

Algorithms and Techniques

For the final model, I have tested different machine learning algorithms such as Multinomial-Naïve Bayes, Support-Vector Classifiers and Logistic Regression. I have used default algorithm parameters and implemented techniques such as Count Vectorizer, TF-IDF Vectorization as feature inputs in the above-mentioned machine learning algorithms. I have also tested Fully Connected Feed-Forward Models using sigmoid activation function in the last layer to get binary output to determine if the news article is true of fake. The best model was chosen using different permutations using evaluation metrics.

Multinomial Naïve-Bayes algorithm is a specific instance of Naïve-Bayes algorithm which refers to conditional independence of each feature. However, the multinomial Naïve-Bayes assumes a multinomial distribution of each feature and applies Bayes' theorem which is specifically designed for text documents.

Support Vector Machines is used to find a hyper-plane in an n-dimensional space (when n is number of features) that distinctly classify the data points. Since our objective is to classify the data points, we maximize the distance between the data points of 2 classes i.e. here true news and fake news features. Using the support-vectors we maximize the distance between the data points of these 2 classes thus maximize the margin of the classifier.

Logistic Regression model is a statistical model which uses a logistic function to model a binary dependent variable using a set of attributes as independent variables. For the current problem, we predict the class of news article 1-true news and 0-fake news using news article content as vector features.

Fully connected Feed-forward model is the simplest neural network model in which the information only moves in forward direction, from input to multiple hidden nodes and used sigmoid activation function to get binary output from the model. Furthermore, we mapped the predicted probabilities using a threshold to calculate the test accuracy on the dataset.

Benchmark

As mentioned above in the Data Exploration section, we do not need to perform under-sampling or over-sampling since the dataset is balanced with 23481 fake news articles and 21417 true news articles.

For the benchmark model, we wanted to capture the following features:

- Sentiment Score

  This feature represents the sentiment score of news title. For this feature we use the news title since news title is usually the summary of the news article or defines what the news article is all about. The sentiment score captures the attitude or the emotion of the writer i.e. whether it conveys a positive or negative sentiment. The sentiment function of TextBlob [2] returns two properties polarity and subjectivity.

  The sentiment score calculates polarity of each news title which is a float value which lies within the range [-1,1], with -1 being negative and 1 being positive sentiment [3]. Thus, the idea was that negative sentiment might be related to fake news hence we used this feature in the benchmark model.

- Readability Score

  This feature represents the readability score of the news article. This is calculated using Flesch Reading Ease Score. Following table is helpful to access the ease of readability in a document [4].

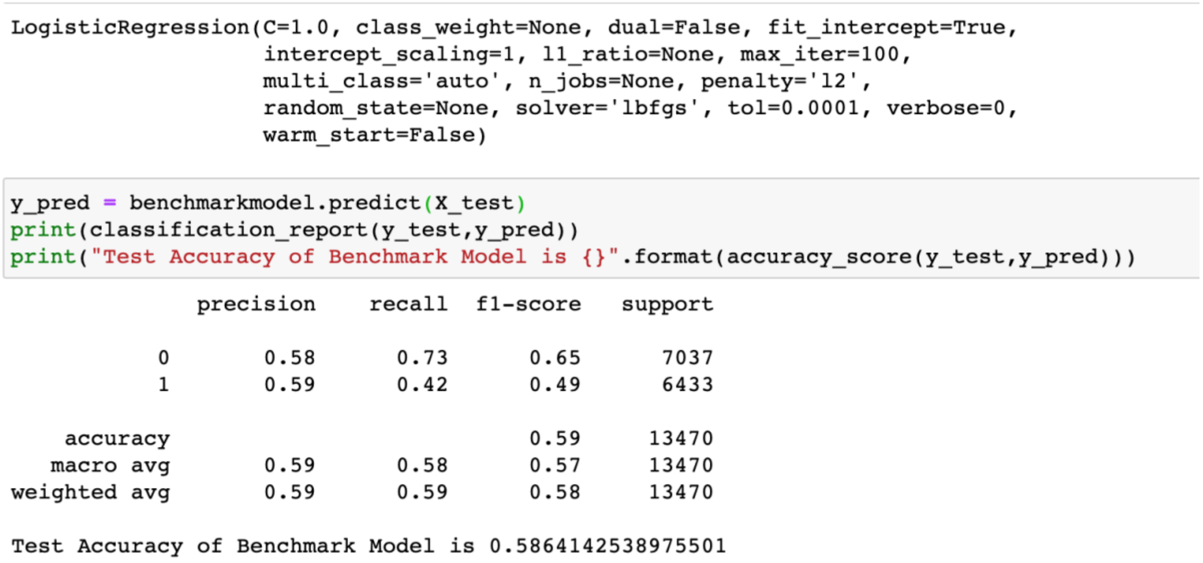
  a) 90-100: Very Easy
  b) 80-89: Easy
  c) 70-79: Fairly Easy
  d) 60-69: Standard
  e) 50-59: Fairly Difficult
  f) 30-49: Difficult
  g) 0-29: Very Confusing

  Hence, the idea was that difficult or very easy articles might be related to fake articles. Similarly, standard scores might be correlated to true news articles and therefore, readability score was used in the benchmark model.

- Word Count

  This feature represents the number of words in the news article. The idea was that fake might be related to higher word count and hence was used in the benchmark model.

Using the above features dataset was split into 70:30 train and test. Label was 1 – true news and 0 – fake news a logistic regression model was created using default parameters. Following is the result of the logistic regression model.



```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
y_pred = benchmarkmodel.predict(X_test)
print(classification_report(y_test,y_pred))
print("Test Accuracy of Benchmark Model is {}".format(accuracy_score(y_test,y_pred)))
```

```
              precision    recall  f1-score   support

           0       0.58      0.73      0.65      7037
           1       0.59      0.42      0.49      6433

    accuracy                           0.59     13470
   macro avg       0.59      0.58      0.57     13470
weighted avg       0.59      0.59      0.58     13470

Test Accuracy of Benchmark Model is 0.5864142538975501
```

Benchmark model has a Test Accuracy of 58.64% Accuracy, Precision of 58% and Recall of 59%. Thus, this is our benchmark model which we will be referred as we further develop Machine Learning models. The benchmark model used above do not use contextual information from the news article which explains the low accuracy, precision and recall values.

Methodology

Data Preprocessing

In the benchmark models we did not capture the contextual information, however in the models ahead we want to capture the contextual information represented in the news article content and following are the data-preprocessing steps.

- **Count Vectorizer**

  This converts the text documents into a matrix of token counts. That means, the news articles are converted into a Term-Frequency (TF) matrix. However, since news articles contains sentences following cleaning steps are taken before converting the news article text into a term-frequency matrix:

  a) Lowercase the text
  b) Remove special characters, white spaces and patterns
  c) Filter words with length > 3
  d) Remove English stop words
  e) Considered a default minimum frequency of 10 for TF matrix

- **TF-IDF Transformer**

  This pre-processing step transforms a Term-Frequency (TF) matrix to a Term Frequency – Inverse Document Frequency Matrix. This transformation is used to scale down the occurrence of high frequency words with less empirical value as compared to words with high empirical value. This information will provide the necessary contextual information required for text classification.

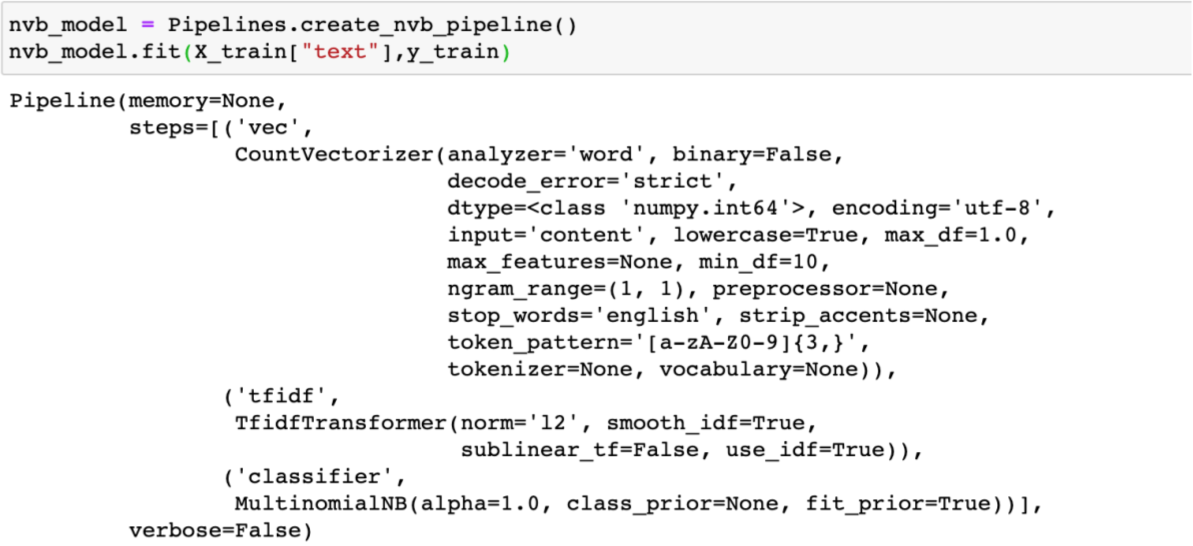
Implementation

The pre-processing steps such as Count Vectorizer was implemented using sci-kit learn library [5] with necessary steps such as stop words removal, punctuation and other text cleaning steps as parameters. TF-IDF Transformer was implemented using sci-kit learn library [6] using default parameters.

As explained above Logistic Regression benchmark model was created using default parameters using sci-kit learn library [7] and results were obtained as above. For implementing other classifiers, we used sci-kit learn library pipeline [8] which was used to sequentially apply the list of pre-processing transformations and classifiers. The specific pipelines were created in python module ml_pipelines.py which consists of 2 machine learning pipelines using classifiers such as Multinomial Naïve-Bayes and Linear SVC respectively. Multinomial Naïve-Bayes was implemented using sci-kit learn library [9] and Linear SVC was also implemented using sci-kit learn library [10] with default parameters.

The advantage of using python modules and sci-kit learn library pipeline [8] is that the final Machine learning workflow only needs to call fit transformation on training data and predict method to test data. The intermediate steps are taken care of in the pipelines thus avoiding unnecessary steps and redundant efforts.

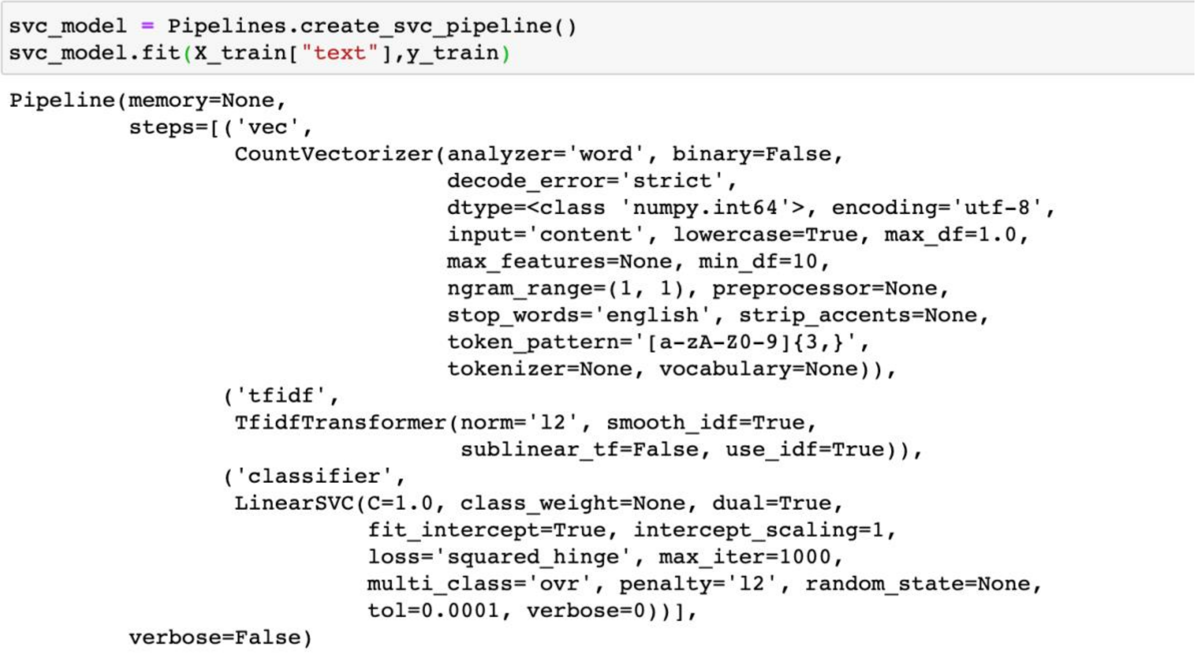
Multinomial Naïve-Bayes Pipeline implementation



```
nvb_model = Pipelines.create_nvb_pipeline()
nvb_model.fit(X_train["text"],y_train)
```

```
Pipeline(memory=None,
         steps=[('vec',
                 CountVectorizer(analyzer='word', binary=False,
                                 decode_error='strict',
                                 dtype=<class 'numpy.int64'>, encoding='utf-8',
                                 input='content', lowercase=True, max_df=1.0,
                                 max_features=None, min_df=10,
                                 ngram_range=(1, 1), preprocessor=None,
                                 stop_words='english', strip_accents=None,
                                 token_pattern='[a-zA-Z0-9]{3,}',
                                 tokenizer=None, vocabulary=None)),
                ('tfidf',
                 TfidfTransformer(norm='l2', smooth_idf=True,
                                  sublinear_tf=False, use_idf=True)),
                ('classifier',
                 MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True))],
         verbose=False)
```

Linear SVC Pipeline implementation

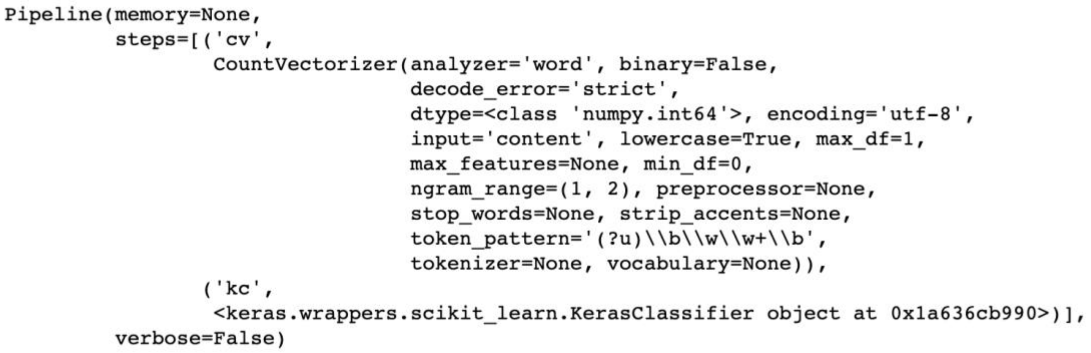


```
svc_model = Pipelines.create_svc_pipeline()
svc_model.fit(X_train["text"],y_train)
```

```
Pipeline(memory=None,
         steps=[('vec',
                 CountVectorizer(analyzer='word', binary=False,
                                 decode_error='strict',
                                 dtype=<class 'numpy.int64'>, encoding='utf-8',
                                 input='content', lowercase=True, max_df=1.0,
                                 max_features=None, min_df=10,
                                 ngram_range=(1, 1), preprocessor=None,
                                 stop_words='english', strip_accents=None,
                                 token_pattern='[a-zA-Z0-9]{3,}',
                                 tokenizer=None, vocabulary=None)),
                ('tfidf',
                 TfidfTransformer(norm='l2', smooth_idf=True,
                                  sublinear_tf=False, use_idf=True)),
                ('classifier',
                 LinearSVC(C=1.0, class_weight=None, dual=True,
                           fit_intercept=True, intercept_scaling=1,
                           loss='squared_hinge', max_iter=1000,
                           multi_class='ovr', penalty='l2', random_state=None,
                           tol=0.0001, verbose=0))],
         verbose=False)
```

Fully connected feed-forward model was implemented using Keras Sequential Model library [11]. Default hyper-parameters were added to the feed-forward model and used the sci-kit learn pipeline library to sequential add the pre-processing steps.

Fully Connected feed-forward model implementation



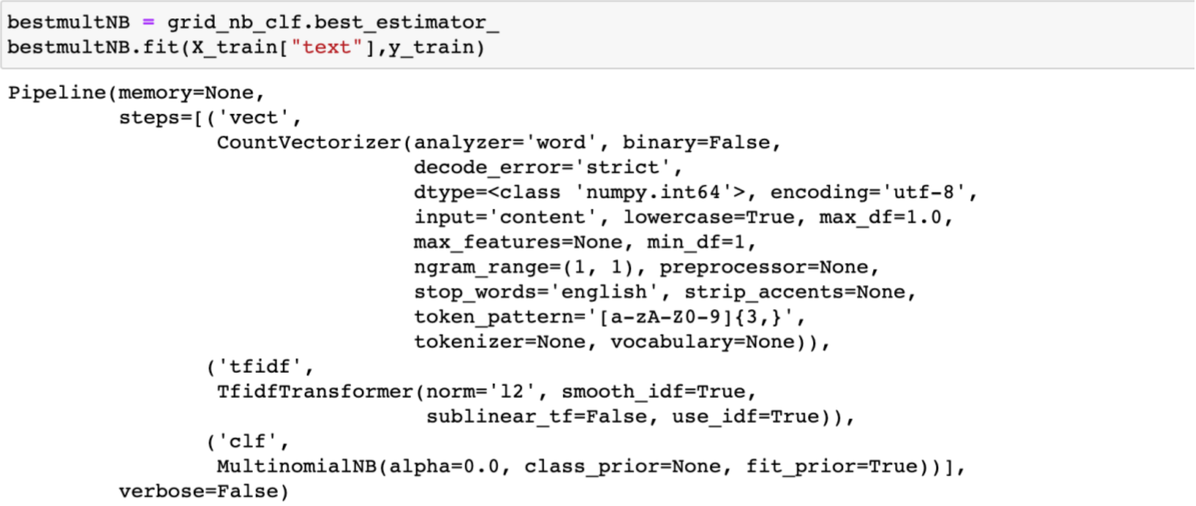
```
Pipeline(memory=None,
         steps=[('cv',
                 CountVectorizer(analyzer='word', binary=False,
                                 decode_error='strict',
                                 dtype=<class 'numpy.int64'>, encoding='utf-8',
                                 input='content', lowercase=True, max_df=1,
                                 max_features=None, min_df=0,
                                 ngram_range=(1, 2), preprocessor=None,
                                 stop_words=None, strip_accents=None,
                                 token_pattern='(?u)\\b\\w\\w+\\b',
                                 tokenizer=None, vocabulary=None)),
                ('kc',
                 <keras.wrappers.scikit_learn.KerasClassifier object at 0x1a636cb990>)],
         verbose=False)
```

Refinement

Hyper-parameter tuning was performed for the above machine learning models. For multinomial Naïve-Bayes model, best alpha value = 0 was obtained from Grid Search with cross-validation using the following best model parameters.

Grid Search was implemented using sci-kit learn library [12]. The grid search generates an exhaustive list of parameters provided in grid parameters and generates the best parameters. Since the default parameters performs well, we tuned the model only to find the best alpha value. The out-of-sample accuracy is 93.58%. Detailed model results are shown in the Results section.

Multinomial Naïve-Bayes Best estimator model parameters

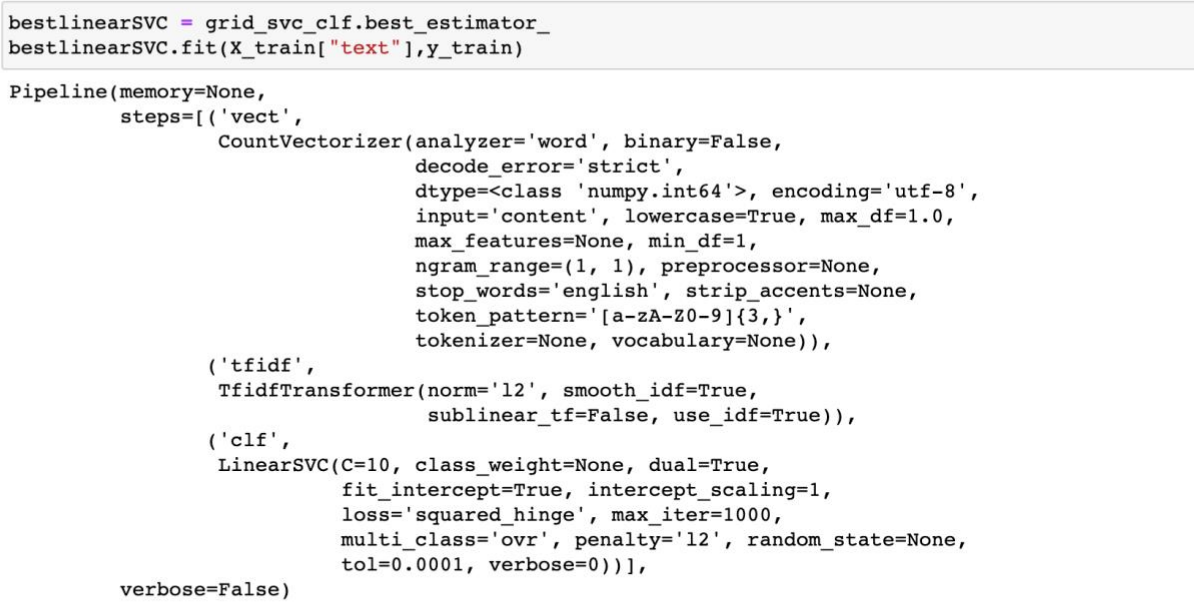


```
bestmultNB = grid_nb_clf.best_estimator_
bestmultNB.fit(X_train["text"],y_train)
```

```
Pipeline(memory=None,
         steps=[('vect',
                 CountVectorizer(analyzer='word', binary=False,
                                 decode_error='strict',
                                 dtype=<class 'numpy.int64'>, encoding='utf-8',
                                 input='content', lowercase=True, max_df=1.0,
                                 max_features=None, min_df=1,
                                 ngram_range=(1, 1), preprocessor=None,
                                 stop_words='english', strip_accents=None,
                                 token_pattern='[a-zA-Z0-9]{3,}',
                                 tokenizer=None, vocabulary=None)),
                ('tfidf',
                 TfidfTransformer(norm='l2', smooth_idf=True,
                                  sublinear_tf=False, use_idf=True)),
                ('clf',
                 MultinomialNB(alpha=0.0, class_prior=None, fit_prior=True))],
         verbose=False)
```

For Linear SVC model, best C value = 10 was obtained from Grid Search with cross-validation using the following best model parameters. Since the default parameters performs well, we tuned the model only to find the best C value. The out-of-sample accuracy is 99.28%. Detailed model results are shown in the Results section.

Multinomial Naïve-Bayes Best estimator model parameters



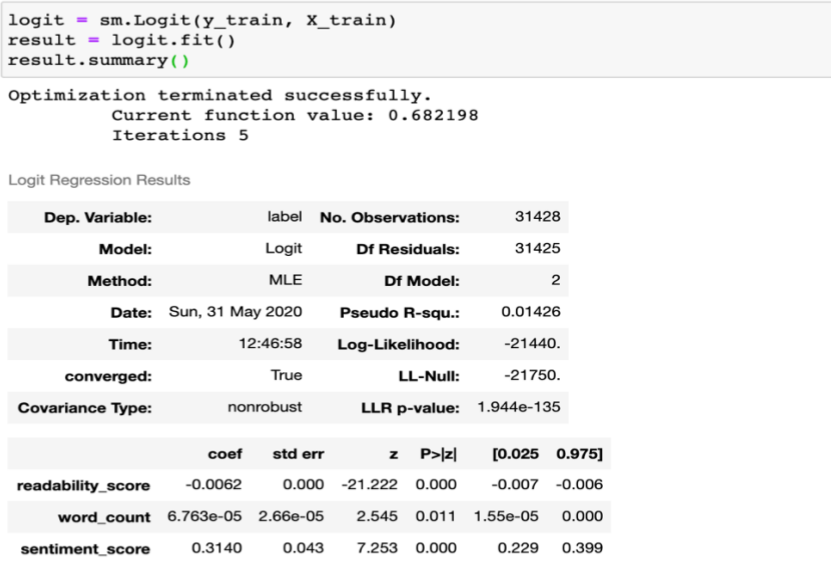
```
bestlinearSVC = grid_svc_clf.best_estimator_
bestlinearSVC.fit(X_train["text"],y_train)
```

```
Pipeline(memory=None,
         steps=[('vect',
                 CountVectorizer(analyzer='word', binary=False,
                                 decode_error='strict',
                                 dtype=<class 'numpy.int64'>, encoding='utf-8',
                                 input='content', lowercase=True, max_df=1.0,
                                 max_features=None, min_df=1,
                                 ngram_range=(1, 1), preprocessor=None,
                                 stop_words='english', strip_accents=None,
                                 token_pattern='[a-zA-Z0-9]{3,}',
                                 tokenizer=None, vocabulary=None)),
                ('tfidf',
                 TfidfTransformer(norm='l2', smooth_idf=True,
                                  sublinear_tf=False, use_idf=True)),
                ('clf',
                 LinearSVC(C=10, class_weight=None, dual=True,
                           fit_intercept=True, intercept_scaling=1,
                           loss='squared_hinge', max_iter=1000,
                           multi_class='ovr', penalty='l2', random_state=None,
                           tol=0.0001, verbose=0))],
         verbose=False)
```

Other models' hyper-parameter tuning was not required since we already obtained good out-of-sample accuracy, precision and recall values from the above two models.

Results

Benchmark Logistic Regression model results



```
logit = sm.Logit(y_train, X_train)
result = logit.fit()
result.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.682198
         Iterations 5
```

Logit Regression Results

Dep. Variable:	label	**No. Observations:**	31428
Model:	Logit	**Df Residuals:**	31425
Method:	MLE	**Df Model:**	2
Date:	Sun, 31 May 2020	**Pseudo R-squ.:**	0.01426
Time:	12:46:58	**Log-Likelihood:**	-21440.
converged:	True	**LL-Null:**	-21750.
Covariance Type:	nonrobust	**LLR p-value:**	1.944e-135

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
readability_score	-0.0062	0.000	-21.222	0.000	-0.007	-0.006
word_count	6.763e-05	2.66e-05	2.545	0.011	1.55e-05	0.000
sentiment_score	0.3140	0.043	7.253	0.000	0.229	0.399

Based on the above results we see that readability score and sentiment score are correlated to the true news article because of significant coefficient values. The coefficient interpretations are not considered for interpretations because of less accuracy, precision and recall values. However, the benchmark model gives us an intuition about types of features to be considered in the model development.

Following are the results of the model tested on out-of-sample data during the machine learning model development to classify fake or true news articles.

Model	Features	Accuracy	Precision	Recall
Benchmark Model (Logistic Regression)	Readability Score, Sentiment Score, Word Count	58.64%	58%	59%
Multinomial Naïve-Bayes	Count Vectorizer, TF-IDF	93.12%	93%	93%
Linear SVC	Count Vectorizer, TF-IDF	99%	99%	99%
Fully Connected Feed-Forward Model (1-hidden layer – 50 nodes, dropout – 0.02, Adam optimizer)	Count Vectorizer	87.3%	-	-
Fully Connected Feed-Forward Model (1-hidden layer – 20 nodes, dropout – 0.02, Adam optimizer)	Count Vectorizer, TF-IDF	88.3%	-	-
Tuned Multinomial Naïve-Bayes	Count Vectorizer, TF-IDF	93.58%	93%	94%
Tuned Linear SVC	**Count Vectorizer, TF-IDF**	**99.28%**	**99%**	**99%**

Model Evaluation and Validation

Based on the above model results we choose the tuned linear SVC model with Count Vectorizer and TF-IDF as feature vectors of news text article. The SVC model has the highest accuracy of 99.28% with a recall and precision value of 99%.

The SVC model results performs considerably better than our benchmark logistic regression model due to use of Count-Vectorizer and TF-IDF features thus, capturing the appropriate contextual information to be fed in the SVC model algorithms.

Justification

Linear SVC model has a drawback of over-fitting the data however we obtained an out-of-sample accuracy of 99% using hyper-parameter tuning with cross-validation. Since the linear SVC model already performed well without hyper-parameter, we tuned the C-value of the model, which indicates the regularization strength in the model.

Regularization is used to avoid over-fitting the model thus, given an already high accuracy, precision and recall we wanted to make sure that we are not over-fitting the model, hence only tuned the C-value of the model using Grid Search.

Hence obtained a C-value=10 was obtained using Grid Search using cross-validation. Cross-validation is used to train the model with taking a portion of our training data as out-of-sample to avoid overfitting our model. Therefore, we can see that with the best pre-processed text features, SVC classifier and hyper-parameter tuning using Grid Search with cross-validation our model performed the best amongst all the other models with 99% out-of-sample accuracy, precision and recall.

Future Improvements

Although we were able to create a Machine Learning algorithm with out-of-sample accuracy, precision and recall of 99%, however, for deploying the above model in order to determine we need to have a semi-supervised learning approach. The reason is that news articles change every day because of current affairs hence the text features is ever-changing. Thus, we need to regularly update our Machine Learning model in order to capture the latest text features and context to have better accuracy, precision and recall. Because Machine Learning models based on past text features may not be relevant to the latest textual features thus increasing the misclassification rate.

References

1. https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset
2. https://textblob.readthedocs.io/en/dev/api_reference.html#textblob.blob.TextBlob
3. https://textblob.readthedocs.io/en/dev/quickstart.html#sentiment-analysis
4. https://pypi.org/project/textstat/0.2/
5. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
6. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html
7. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
8. https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html
9. https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html

10. https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html
11. https://keras.io/guides/sequential_model/
12. https://scikit-learn.org/stable/modules/grid_search.html
13. Ahmed H, Traore I, Saad S. “Detecting opinion spams and fake news using text classification”, Journal of Security and Privacy, Volume 1, Issue 1, Wiley, January/February 2018.
14. Ahmed H, Traore I, Saad S. (2017) “Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques. In: Traore I., Woungang I., Awad A. (eds) Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments. ISDDC 2017. Lecture Notes in Computer Science, vol 10618. Springer, Cham (pp. 127-138).
15. https://python-graph-gallery.com/wordcloud/
16. https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC
17. https://scikit-learn.org/stable/modules/svm.html
18. https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html
19. Stuart J. Russell and Peter Norvig. 2003. Artificial Intelligence: A Modern Approach (2 ed.). Pearson Education. See p. 499 for reference to "idiot Bayes" as well as the general definition of the Naive Bayes model and its independence assumptions