Project Title

Text Classification of News

Team: t-rex
Akash Sindhu

Motivation and Background

The motivation behind me choosing this project is to understand can our machine learning models solve a news classification problem. This problem is dynamic in nature and relies heavily on external factors that change a lot. If we thought of the news dataset as the time series data just like stocks where new events define the outcome, we will be on the same page. But having said that, I can still make a model on past news datasets to understand the dataset and solve some data science questions like how well our model can perform on new news, for two similar topics, can my model have the same results. I can also check if the model can perform well on the same topic across time.

I care about fake news as it can change people's sentiments for a certain period. We saw that many times in the past during the presidential election in the US during the time of Trump and Clinton. Besides this, fake news can also decrease trust and manipulate people's decision-making power on certain crucial topics in society and life. Incorrect personal health or misleading medical information can also adversely affect the social well-being of the target area. Fake news not only affects individuals on a small scale but on a larger scale as well.

There have been many studies that showed that fake news is not good for society and can affect badly in many ways one cannot imagine.

Real-life stories

- Death sparks probe into China's Baidu. (2016, May 3). *BBC News*.
  Fake news advertisements promoted by Baidu, China's biggest search engine, may have contributed to the death of a 21-year-old student. Wei Zexi, who was diagnosed with a rare form of cancer, pursued an experimental treatment that was featured at the top of his search results.
- General effects of biased and incomplete information on public health
  - Speed, E., & Mannion, R. (2017). The Rise of Post-truth Populism in Pluralist Liberal Democracies: Challenges for Health Policy. *International Journal of Health Policy and Management, 6*(5), 249–251. (U-M Library Access; PubMed Central)

Problem Statement

Keeping in mind that solving questions or a problem in data science is not possible without having a pre-requisite machine learning model/idea. So, I will first spend some time talking about machine learning approaches I used to get a required model that can be trusted to some extent. Afterward, I will try to solve some data science questions related to what our model can solve or should solve.

Some of the questions I would like to solve in this project are:

Can I join two random news text datasets?

Does bias and variance matter in text datasets?

Can I develop state of an art model on a test dataset?

Can I get state-of-the-art accuracy on new news data?

Do we need to train the model on topic-specific news?

How will my model perform on two similar topics of present news considering the train data has those topics?

Data Science Pipeline and Methodology

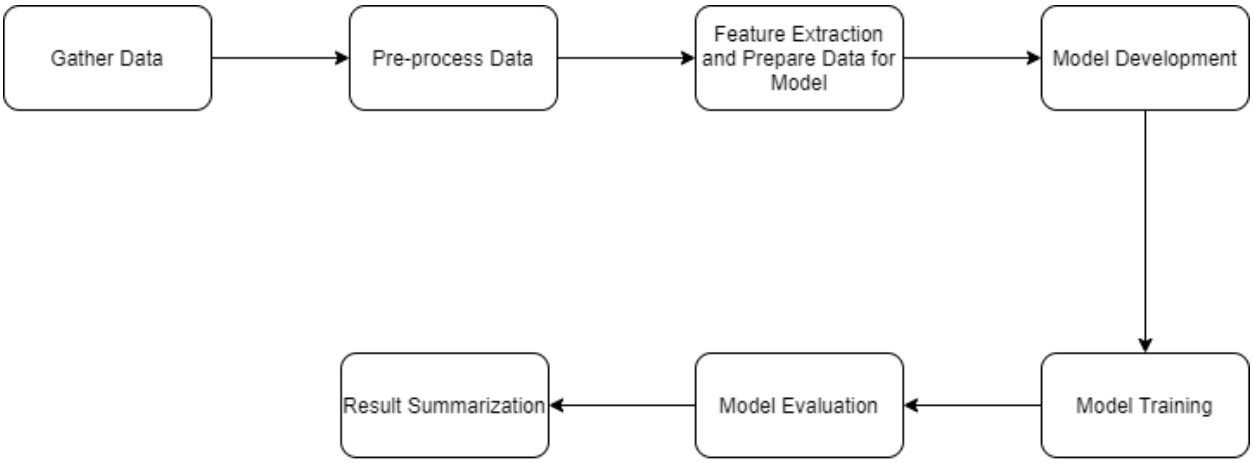

Figure 1: Workflow of Project

This will be the pipeline of the development process. We first will gather data from different sources, pre-process the data, extract some features and tokenize pad sequence and split the data into 80:20 ratio of train and test respectively and ingest in the model, develop the model architecture, train the model and evaluate the model.

Data Gathering:

I downloaded 2 datasets from Kaggle and 1 dataset from some random website. I then combined these together to make a balanced large dataset.

Besides this, I am using AYLIEN news API to extract the news from = 'NOW-40DAYS' until = 'NOW-20DAYS'. This data is not labelled but I will compare the results of the models and check if similar performing models perform similar on this data. To extract the news from this API, I use the topic that is topmost in the top 500 rows of the dataset. With the help of Spacy, I am able to extract this information.

Pre-process Data

Text dataset requires a lot of cleaning to reduce the noise and increase the presence of meaningful words. The dataset we downloaded in the raw format was not cleaned and could not be feed into machine learning model. So, I cleaned the dataset and removed these:

1. Lowercase all words.
2. Filter out punctuations.
3. Remove words that are not alphabetic.
4. Remove stop words.
5. Remove Stem words.
6. Remove words that are of length 1.

Feature extraction and prepare dataset for model

Feature extraction in text dataset consists of tokenizing the text based on word count, word frequencies, TF-IDF, and hashing. We are going to use keras.tokenizer and texts_to_sequences() to integer encode the words. After that, we will pad the sequences to the length of 400 so that all the text news sent to the model has the same length.

Distribution of x_train, y_train, x_test, y_test.

The dataset is shuffled to ensure that each data point creates an "independent" change on the model, without being biased by the same points before them. Data is split into 80% train data and 20% test data.

Model development

Baseline Model is used for all the datasets and tests.

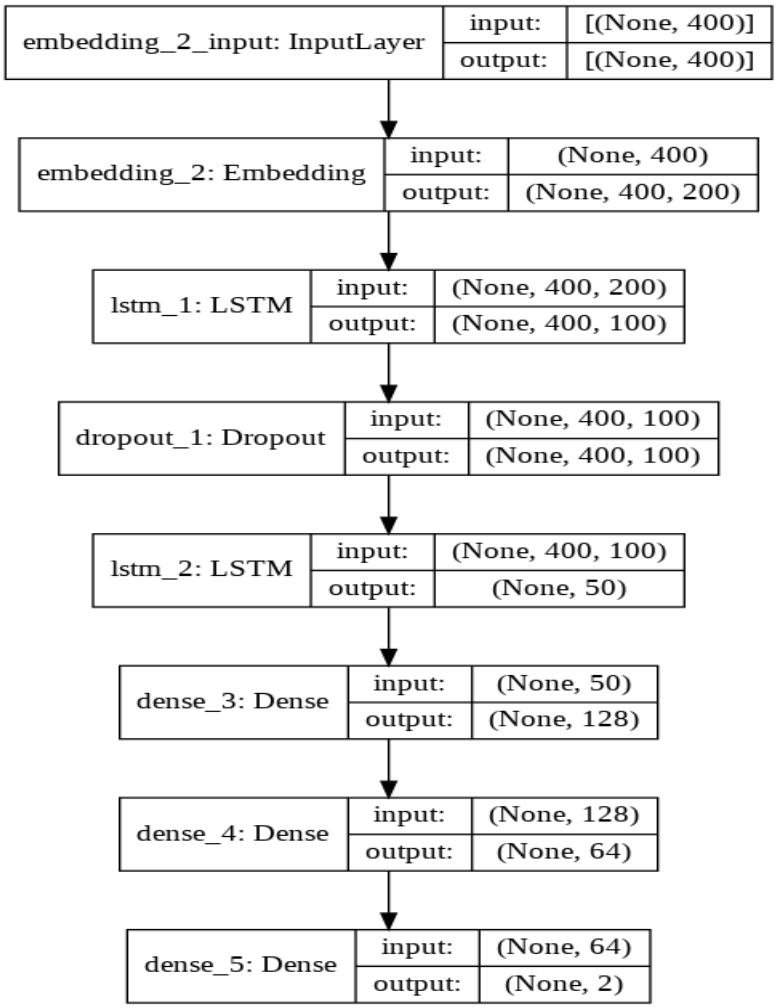

Figure 2: Baseline Model Structure Plot

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 400, 200)          10836800
_________________________________________________________________
lstm_1 (LSTM)                (None, 400, 100)          120400
_________________________________________________________________
dropout_1 (Dropout)          (None, 400, 100)          0
_________________________________________________________________
lstm_2 (LSTM)                (None, 50)                30200
_________________________________________________________________
dense_3 (Dense)              (None, 128)               6528
_________________________________________________________________
dense_4 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_5 (Dense)              (None, 2)                 130
=================================================================
Total params: 11,002,314
Trainable params: 11,002,314
Non-trainable params: 0
```

Figure 3: Model Parameters and Structure

I am using the Embedding layer with weights from the pre-trained word embedding model link . This model was trained on a Twitter dataset of 2B tweets, 27B tokens, 1.2M vocab, uncased, 200d vectors, 1.42 GB in size. I am only using weights for the words that are present in the train data so that it will be fast and a little bit of a challenge for our model to predict on test data if some new work comes up. After the embedding layer, I am using LSTM with 100 cells to learn the sequential context in the text. After that, I am forcing the model to forget some information by using a dropout layer with 50 percent of neurons to stay alive at the given time. This is a very strong killing off neurons and I think will boost the accuracy. LSTM layer is again used to learn more sequential relations in the text and is followed by dense layers and then output layer with softmax function.

Model training

Same Baseline model is trained on all the datasets and has showed great results.

Evaluation

I am using 4 different datasets from different sources. But the point is they are all about fake or real news. Here first I will try to establish a relationship between bias and variance by showing the results. To allow the machine learning model to understand deep relations between the data points, we need to have a good trade-off between bias and variance. We will check if combining the random news dataset will get us good results or not. To check this, we will first run the base model with each dataset and then run it on the combined dataset. This will allow us to compare

the accuracy and will let us know if we can combine all datasets or not. We will visualize the word embeddings of each dataset as it allows us to see bias, variance, and how far the data points are from each other.

Test 1 on Dataset 1:

Dataset consists of 3171 Real and 3164 Fake news articles. We will not care what is the source of the dataset as this is what we are trying to find out.

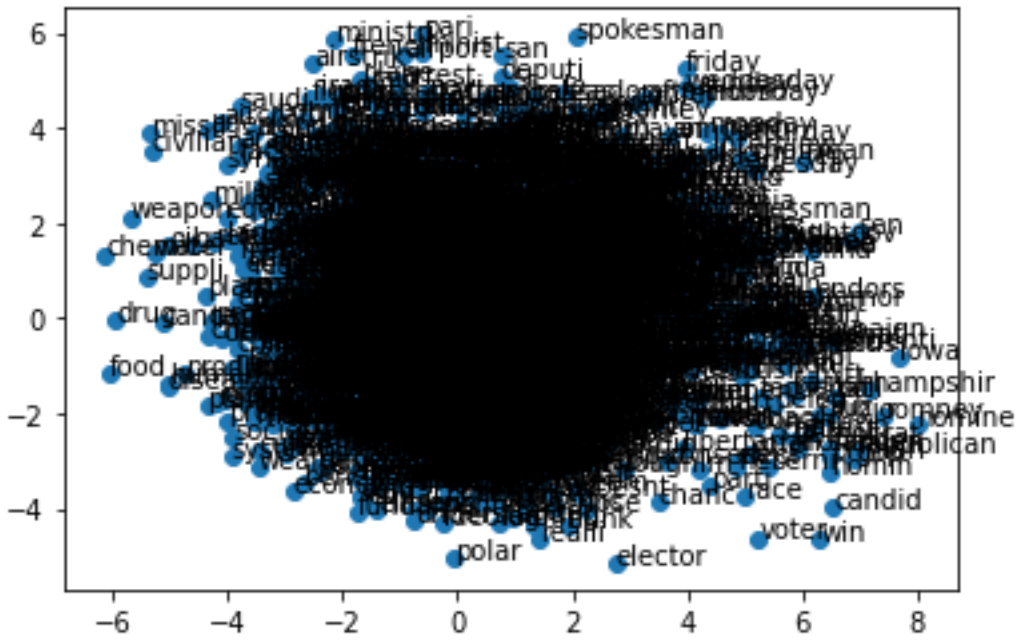

Figure 4: Word Embeddings of using Word2Vec algorithm.

The figure shows that our dataset is centered and has most probably low bias and definitely low variance. So, this dataset is ideal and should perform well.

```
Epoch 1/15
159/159 [==============================] - 18s 94ms/step - loss: 0.6449 - accuracy:
0.6504
Epoch 2/15
159/159 [==============================] - 15s 92ms/step - loss: 0.5130 - accuracy:
0.7409
Epoch 3/15
159/159 [==============================] - 14s 91ms/step - loss: 0.4220 - accuracy:
0.7941
Epoch 4/15
159/159 [==============================] - 14s 90ms/step - loss: 0.3249 - accuracy:
0.8659
Epoch 5/15
```

```
159/159 [==============================] - 14s 91ms/step - loss: 0.4640 - accuracy: 0.7302
Epoch 6/15
159/159 [==============================] - 14s 90ms/step - loss: 0.3873 - accuracy: 0.8090
Epoch 7/15
159/159 [==============================] - 14s 90ms/step - loss: 0.4086 - accuracy: 0.7741
Epoch 8/15
159/159 [==============================] - 14s 91ms/step - loss: 0.3712 - accuracy: 0.8275
Epoch 9/15
159/159 [==============================] - 14s 91ms/step - loss: 0.4083 - accuracy: 0.8100
Epoch 10/15
159/159 [==============================] - 14s 91ms/step - loss: 0.4654 - accuracy: 0.7490
Epoch 11/15
159/159 [==============================] - 14s 90ms/step - loss: 0.2815 - accuracy: 0.8777
Epoch 12/15
159/159 [==============================] - 14s 91ms/step - loss: 0.2356 - accuracy: 0.9128
Epoch 13/15
159/159 [==============================] - 14s 89ms/step - loss: 0.1625 - accuracy: 0.9410
Epoch 14/15
159/159 [==============================] - 14s 91ms/step - loss: 0.1134 - accuracy: 0.9671
Epoch 15/15
159/159 [==============================] - 14s 91ms/step - loss: 0.0681 - accuracy: 0.9790
```

We can see that our model is able to perform well on approaching 99.73% accuracy during training.

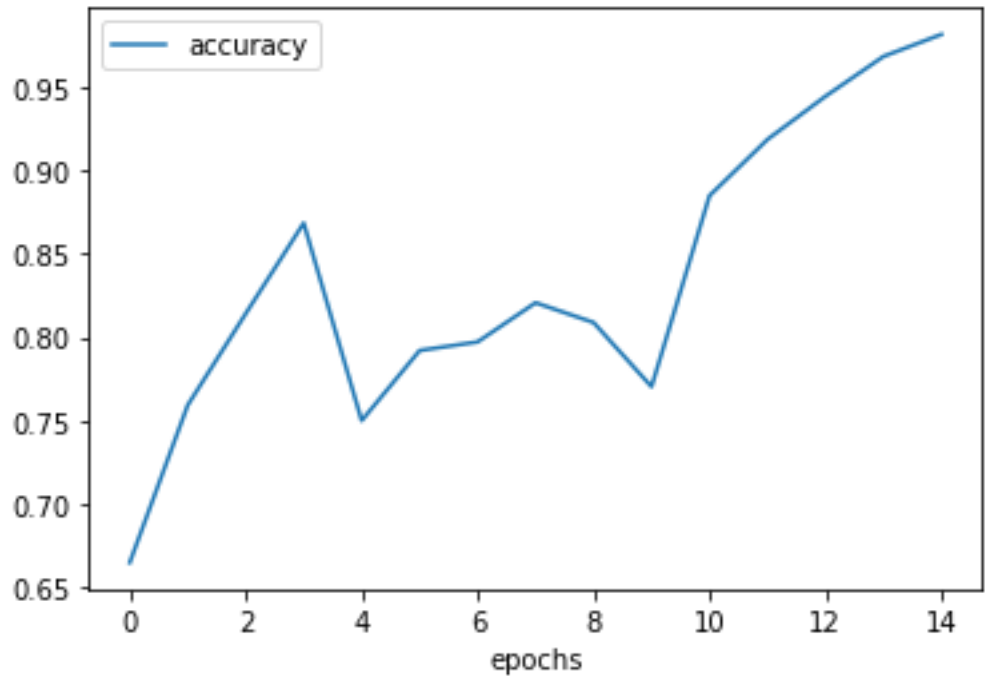

Figure 5: Accuracy with number of epochs. We can see huge increase in the accuracy from 9 to 15 epochs.

Model performance on Test Data:

LOSS: 0.4599151611328125

ACC: 0.8942383527755737

confusion matrix

[[533 95]

[39 600]]

Confusion matrix shows really good results here and most of the predicted values are correct.

	precision	recall	f1-score	support
0	0.93	0.85	0.89	628
1	0.86	0.94	0.90	639
accuracy			0.89	1267
macro avg	0.90	0.89	0.89	1267
weighted avg	0.90	0.89	0.89	1267

Topmost common word in the top 500 lines are:

('Trump', 1256), ('Clinton', 1144), ('Republican', 404), ('Donald Trump', 366), ('Hillary Clinton', 353), ('Republicans', 320), ('American', 287), ('Democrats', 280), ('Obama', 280), ('Americans', 264), ('FBI', 246), ('Democratic', 240), ('ISIS', 239), ('Sanders', 233), ('Congress', 222), ('GOP', 211), ('Hillary', 193), ('Senate', 180), ('Cruz', 174), ('Lesley Stahl', 144), ('Bush', 135), ('House', 134), ('CNN', 123), ('Rubio', 95),

Number of News extracted: 20

Prediction by the model for topic "Trump": array([0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1])

Model do predict some of the news as fake and real.

Test 2 with dataset 2:

Dataset consists of 2973 Real and 2014 Fake news articles.

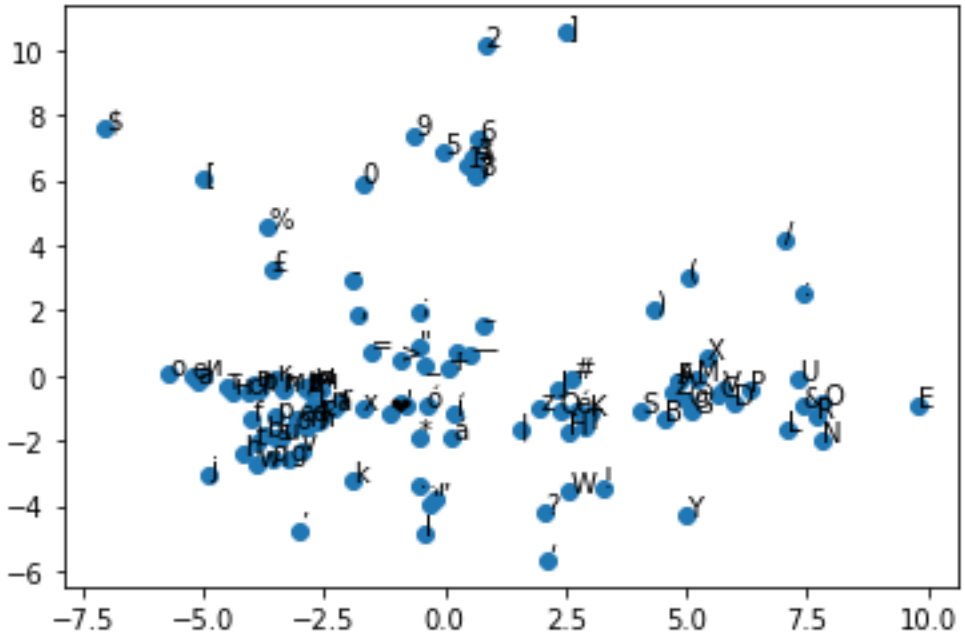

Figure 6: Word Embeddings of using Word2Vec algorithm.

The figure shows that our dataset is centered and has most probably high bias and definitely higher variance than previous dataset. Let's see if our model is able to learn this.

```
Epoch 1/30
125/125 [==============================] - 16s 96ms/step - loss: 0.6788 - accuracy: 0.5787
Epoch 2/30
125/125 [==============================] - 12s 95ms/step - loss: 0.6722 - accuracy: 0.5929
Epoch 3/30
```

```
125/125 [==============================] - 12s 95ms/step - loss: 0.6222 - accuracy:
0.6399
Epoch 4/30
125/125 [==============================] - 12s 95ms/step - loss: 0.5808 - accuracy:
0.6520
Epoch 5/30
125/125 [==============================] - 12s 93ms/step - loss: 0.5659 - accuracy:
0.6511
Epoch 6/30
125/125 [==============================] - 12s 94ms/step - loss: 0.5527 - accuracy:
0.6799
Epoch 7/30
125/125 [==============================] - 12s 94ms/step - loss: 0.5503 - accuracy:
0.6778
Epoch 8/30
125/125 [==============================] - 12s 94ms/step - loss: 0.5505 - accuracy:
0.6661
Epoch 9/30
125/125 [==============================] - 12s 94ms/step - loss: 0.5797 - accuracy:
0.6725
Epoch 10/30
125/125 [==============================] - 12s 94ms/step - loss: 0.5458 - accuracy:
0.6719
Epoch 11/30
125/125 [==============================] - 12s 94ms/step - loss: 0.5507 - accuracy:
0.6717
Epoch 12/30
125/125 [==============================] - 12s 93ms/step - loss: 0.5521 - accuracy:
0.6752
Epoch 13/30
125/125 [==============================] - 12s 93ms/step - loss: 0.5364 - accuracy:
0.6817
Epoch 14/30
125/125 [==============================] - 12s 93ms/step - loss: 0.5377 - accuracy:
0.6855
Epoch 15/30
125/125 [==============================] - 12s 93ms/step - loss: 0.5344 - accuracy:
0.6950
Epoch 16/30
125/125 [==============================] - 12s 93ms/step - loss: 0.5529 - accuracy:
0.6811
Epoch 17/30
125/125 [==============================] - 12s 92ms/step - loss: 0.5520 - accuracy:
0.6596
Epoch 18/30
```

```
125/125 [==============================] - 12s 93ms/step - loss: 0.5444 - accuracy: 0.6794
Epoch 19/30
125/125 [==============================] - 11s 92ms/step - loss: 0.5480 - accuracy: 0.6705
Epoch 20/30
125/125 [==============================] - 12s 92ms/step - loss: 0.5413 - accuracy: 0.6722
Epoch 21/30
125/125 [==============================] - 12s 92ms/step - loss: 0.5337 - accuracy: 0.6925
Epoch 22/30
125/125 [==============================] - 12s 93ms/step - loss: 0.5500 - accuracy: 0.6741
Epoch 23/30
125/125 [==============================] - 11s 91ms/step - loss: 0.5481 - accuracy: 0.6696
Epoch 24/30
125/125 [==============================] - 12s 92ms/step - loss: 0.5503 - accuracy: 0.6691
Epoch 25/30
125/125 [==============================] - 12s 93ms/step - loss: 0.5426 - accuracy: 0.6812
Epoch 26/30
125/125 [==============================] - 12s 92ms/step - loss: 0.5363 - accuracy: 0.6823
Epoch 27/30
125/125 [==============================] - 12s 92ms/step - loss: 0.5426 - accuracy: 0.6754
Epoch 28/30
125/125 [==============================] - 12s 92ms/step - loss: 0.5316 - accuracy: 0.6930
Epoch 29/30
125/125 [==============================] - 11s 92ms/step - loss: 0.5378 - accuracy: 0.6759
Epoch 30/30
125/125 [==============================] - 12s 93ms/step - loss: 0.5485 - accuracy: 0.6796
```

We can see that our model is having hard time to learn because of high variance. It will be difficult for the model to predict accurately.

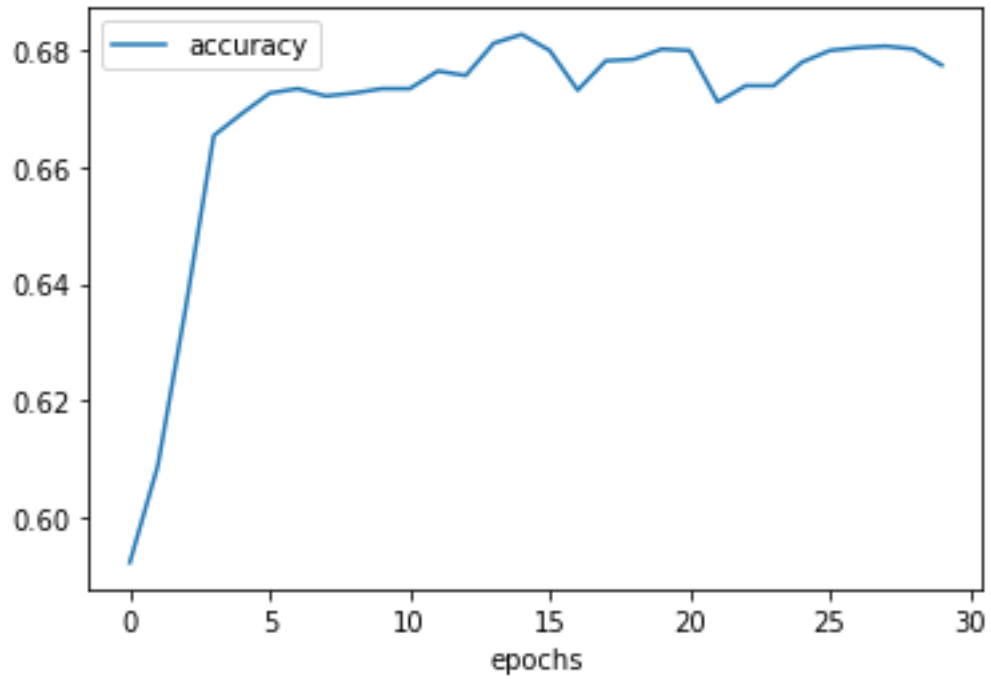

Figure 7: We can see our accuracy increased from 0 to 4 epochs but stayed almost same from 4 to 30 epochs. I think the reason is local minima in the gradient decent.

Model performance on Test Data:

LOSS: 1.033440113067627

ACC: 0.5661322474479675

As discussed earlier, the dataset has higher bias and variance, so it performed poor on the test dataset. By looking at the Figure 4, I can pretty much say that our model is not overfitted and this is the not the reason for not performing well on test dataset.

confusion matrix

[[45 371]

[62 520]]

	precision	recall	f1-score	support
0	0.42	0.11	0.17	416
1	0.58	0.89	0.71	582
accuracy			0.57	998

macro avg	0.50	0.50	0.44	998
weighted avg	0.52	0.57	0.48	998

Most Common words:

('Trump', 276), ('Instagram', 275), ('Cornell', 256), ('Lopez', 239), ('Kim', 222), ('Meghan', 209), ('American', 202), ('Justin', 197), ('Harry', 172), ('Aguilera', 166), ('Jolie', 128), ('Kate', 126), ('Diana', 125), ('British', 124), ('Jennifer', 117), ('Brad', 116), ('Prince Harry', 116), ('Weinstein', 111), ('Kardashian', 110)

Number of News extracted: 20

Prediction of Model for topic "Trump": array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])

All the predictions are marked as Real News. This is not accurate, we will get to know in the next model.

Test 3 with Dataset 3:

Dataset consists of Real 10387 and Fake 10374 news.

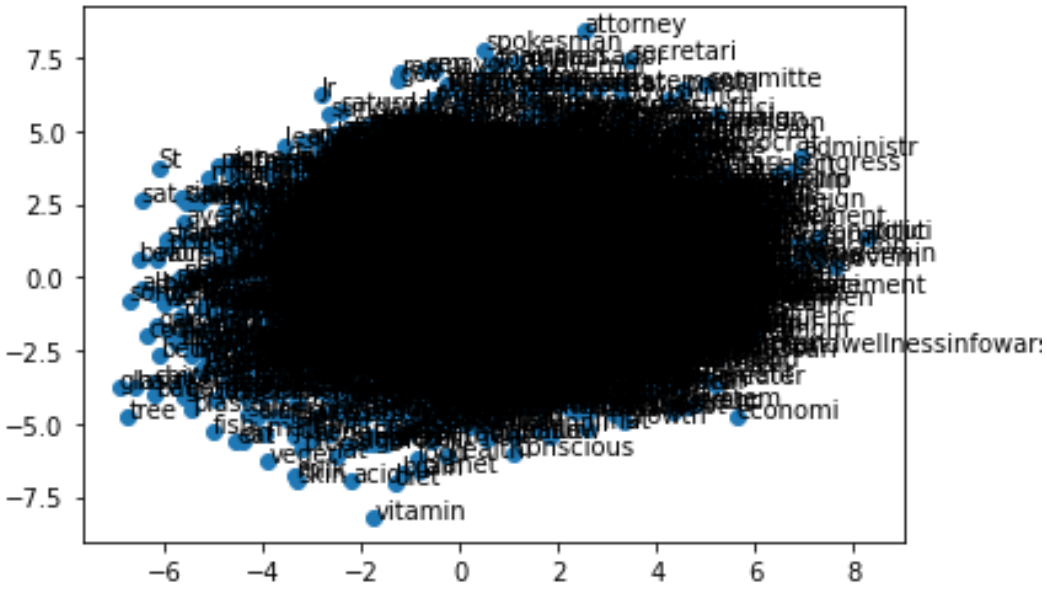

Figure 8: Word Embeddings of data using Word2Vec algorithm.

From the figure I can say that our dataset is pretty much low bias and low variance.

```
Epoch 1/15
519/519 [==============================] - 97s 179ms/step - loss: 0.6379 -
accuracy: 0.6345
Epoch 2/15
```

```
519/519 [==============================] - 92s 177ms/step - loss: 0.4626 -
accuracy: 0.7656
Epoch 3/15
519/519 [==============================] - 91s 175ms/step - loss: 0.3818 -
accuracy: 0.8101
Epoch 4/15
519/519 [==============================] - 90s 174ms/step - loss: 0.1876 -
accuracy: 0.9279
Epoch 5/15
519/519 [==============================] - 90s 174ms/step - loss: 0.1114 -
accuracy: 0.9635
Epoch 6/15
519/519 [==============================] - 90s 173ms/step - loss: 0.0536 -
accuracy: 0.9841
Epoch 7/15
519/519 [==============================] - 89s 172ms/step - loss: 0.0306 -
accuracy: 0.9922
Epoch 8/15
519/519 [==============================] - 89s 172ms/step - loss: 0.0266 -
accuracy: 0.9933
Epoch 9/15
519/519 [==============================] - 89s 172ms/step - loss: 0.0143 -
accuracy: 0.9958
Epoch 10/15
519/519 [==============================] - 89s 171ms/step - loss: 0.0152 -
accuracy: 0.9962
Epoch 11/15
519/519 [==============================] - 89s 171ms/step - loss: 0.0134 -
accuracy: 0.9967
Epoch 12/15
519/519 [==============================] - 89s 171ms/step - loss: 0.0062 -
accuracy: 0.9984
Epoch 13/15
519/519 [==============================] - 88s 170ms/step - loss: 0.0015 -
accuracy: 0.9998
Epoch 14/15
519/519 [==============================] - 89s 171ms/step - loss: 0.0052 -
accuracy: 0.9992
Epoch 15/15
519/519 [==============================] - 89s 171ms/step - loss: 0.0056 -
accuracy: 0.9990
```

As the dataset had low bias and variance, model reached accuracy of 99.29% in just 10 epochs. It makes sense as classifier is having easier time to learn the underline relationship. In the previous test, we had to run for 30 epochs to achieve the same accuracy as it had high variance.

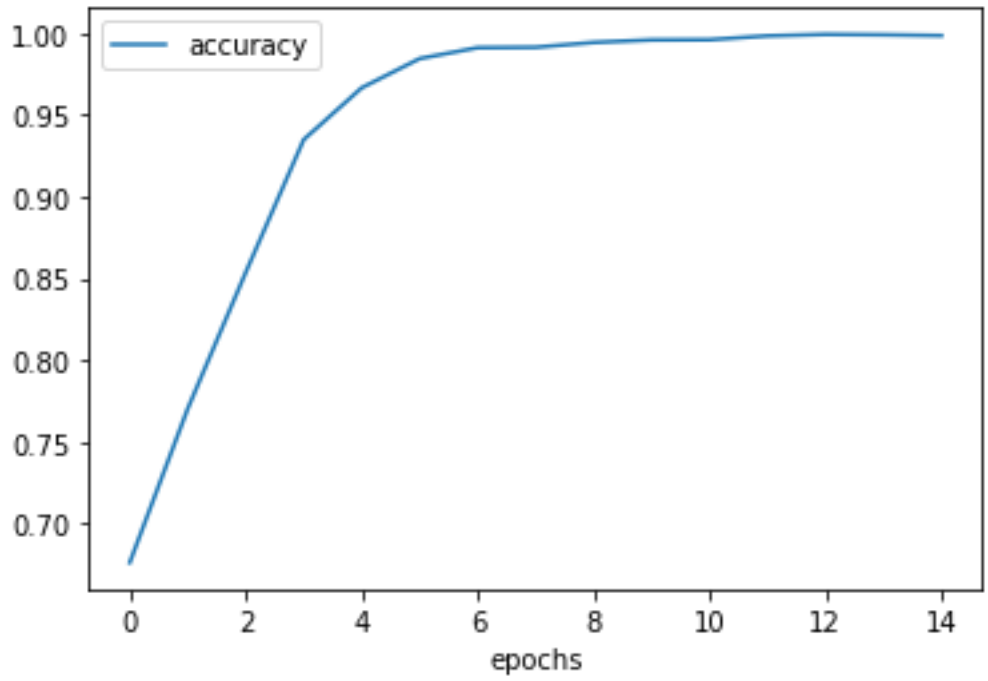

Figure 9: There is increase in accuracy just so perfectly.

Model performance on Test Data.

LOSS: 0.25699344277381897

ACC: 0.9453406929969788

Model performed well on the test dataset.

confusion matrix

[[1953 121]

 [106 1973]]

Model performed really well as there are very less false positive and false negative.

	precision	recall	f1-score	support
0	0.95	0.94	0.95	2074
1	0.94	0.95	0.95	2079
accuracy			0.95	4153
macro avg	0.95	0.95	0.95	4153

weighted avg 0.95 0.95 0.95 4153

Topmost common words in the top 500 rows.

('Trump', 1209), ('Clinton', 460), ('American', 302), ('Obama', 220), ('Hillary Clinton', 206), ('Donald Trump', 186), ('Republican', 183), ('Jewish', 165), ('Democrats', 155), ('Arab', 142), ('Americans', 136), ('Arabs', 135), ('Hillary', 127), ('Russian', 126), ('FBI', 121), ('Republicans', 121), ('Morris', 109), ('Congress', 101), ('ISIS', 99), ('Democratic', 96), ('Chinese', 91)

Number of News extracted: 20

Prediction of model for topic “Trump”: array([0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1])

Prediction of model for topic “Clinton”: array([0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])

Combine the datasets 2, 3 and 4 and compare results.

Dataset consists of Real 16531 and Fake 15552 news.

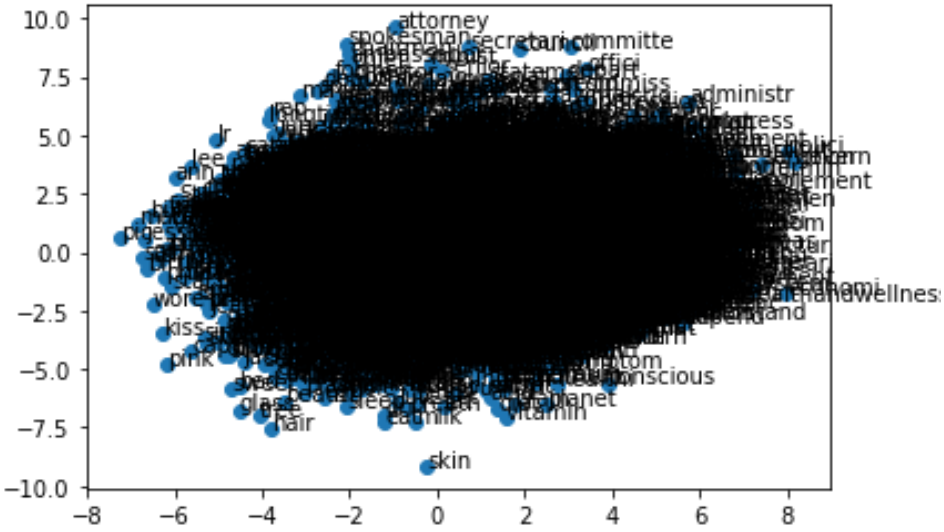

Figure 10: Word Embeddings of data using Word2Vec algorithm.

```
Epoch 1/12
803/803 [==============================] - 173s 211ms/step - loss: 0.6096 -
accuracy: 0.6367
Epoch 2/12
803/803 [==============================] - 168s 209ms/step - loss: 0.4475 -
accuracy: 0.7610
Epoch 3/12
803/803 [==============================] - 169s 210ms/step - loss: 0.4842 -
accuracy: 0.7553
Epoch 4/12
803/803 [==============================] - 167s 208ms/step - loss: 0.2005 -
accuracy: 0.9219
```

```
Epoch 5/12
803/803 [==============================] - 167s 209ms/step - loss: 0.0988 -
accuracy: 0.9660
Epoch 6/12
803/803 [==============================] - 168s 209ms/step - loss: 0.0586 -
accuracy: 0.9818
Epoch 7/12
803/803 [==============================] - 166s 206ms/step - loss: 0.0322 -
accuracy: 0.9904
Epoch 8/12
803/803 [==============================] - 168s 210ms/step - loss: 0.0215 -
accuracy: 0.9937
Epoch 9/12
803/803 [==============================] - 157s 196ms/step - loss: 0.0159 -
accuracy: 0.9965
Epoch 10/12
803/803 [==============================] - 155s 193ms/step - loss: 0.0139 -
accuracy: 0.9960
Epoch 11/12
803/803 [==============================] - 157s 196ms/step - loss: 0.0344 -
accuracy: 0.9904
Epoch 12/12
803/803 [==============================] - 155s 193ms/step - loss: 0.0131 -
accuracy: 0.9964
```

The model performed really well with 99.64% accuracy on the training data.

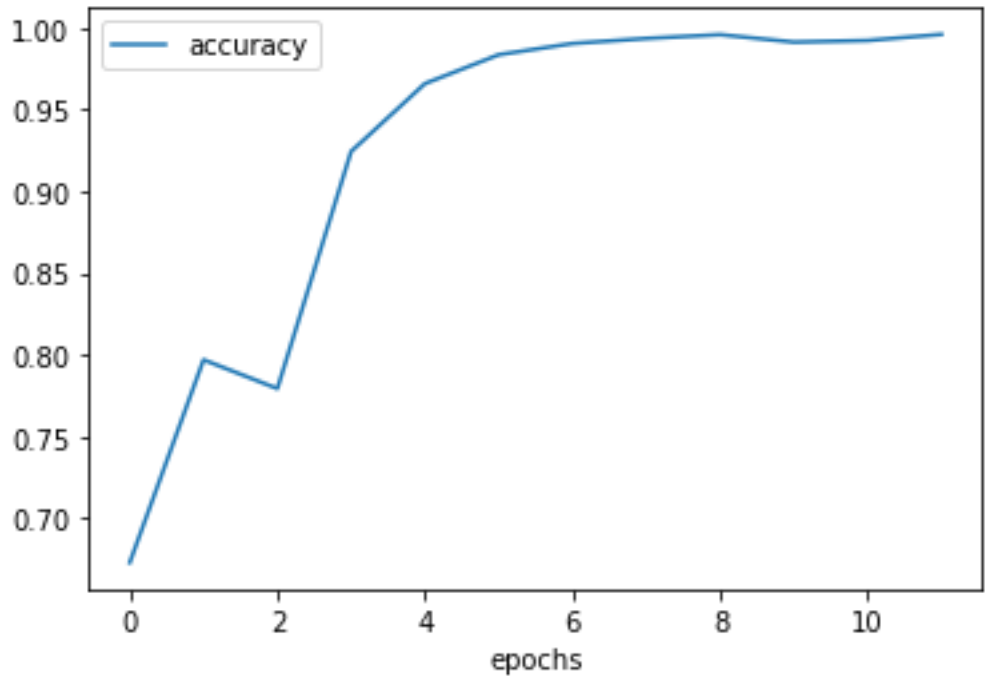

Figure 11: Train Accuracy wrt epochs

Model performance on Test Data.

LOSS: 0.4406101703643799

ACC: 0.9043166637420654

confusion matrix

[[2746 357]

 [257 3057]]

	precision	recall	f1-score	support
0	0.91	0.88	0.90	3103
1	0.90	0.92	0.91	3314
accuracy			0.90	6417
macro avg	0.90	0.90	0.90	6417
weighted avg	0.90	0.90	0.90	6417

Topmost common words in top 500 rows: ('Trump', 1256), ('Clinton', 1144), ('Republican', 404), ('Donald Trump', 366), ('Hillary Clinton', 353), ('Republicans', 320), ('American', 287), 'Democrats', 280), ('Obama', 280), ('Americans', 264), ('FBI', 246), ('Democratic', 240), ('ISIS', 239), ('Sanders', 233), ('Congress', 222), ('GOP', 211)

Number of News extracted: 20

Pridiction of model for topic “Trump”: array([0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0])

Prediction of model for topic “Clinton”: array([0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0])

Data Product

I don’t have a specific data product. To run my application, one must just follow the Jupiter notebook and store the required files in the google bucket.

Lessons Learnt

Some of the lessons I learned during this project are:

- The topmost common topics are similar in train data and “Trump” being on the top 5 in each of them.

- The model also performed well on two similar topics given the train and test accuracy are state of the art.
- By looking at the topmost common words in the top 500 rows, the datasets are from a specific area that is politics.
- Same news articles were extracted for each of the tests “since = 'NOW-40DAYS' until = 'NOW-20DAYS'” and got similar results for models that performed well on train and test data.
- Bias and Variance tradeoffs are very important to look for while generating the dataset.
- Fake news datasets from different sources can or cannot be combined based on the variance levels in the datasets. If two news datasets are of different topics then, they cannot be combined.
- Predictions of present news data of the models that performed well on train and test are good as compared to models that performed poorly on train and test data.
- Models that perform poorly on test data have either all predictions as fake or real.

Summary

In this project, I tried to solve questions regarding the general machine learning and data science related to fake news. I can now say that we do need to prepare fake news data just like stocks dataset where we have only one company dataset in time series. Having more than 1 topic like politics and science can not go along with each other and make the classifier confuse by increasing the variance in the dataset. We saw that happening in one of our datasets where variance was high, and we got poor results. We also saw that given the same area as politics, our model does perform well on similar topics within that area.

To make this project further if I have time, I would like to develop an automated system that will keep track of up-to-date news but in a specific area only. Then it will be accessed from the website for someone who wants to classify the relevancy of the news he just read.