Real vs Fake News Classification

Asad, Hussein, Sudha, Jacob

Fake news in a big picture

- **Fake news** is no more a buzzword.

  A fully **fabricated claim** created with an intention to deceive, often for a secondary gain.

- Fake news is definitely an issue. Let us understand **WHY**

  People tend to believe what they see/ read.

  It **hinders** the mindset of common people with **misleading information**.

  Spread of fake news has the potential for extremely **negative impact** on society.

  It affects not only common people but it can **dictate fate of the entire nation**.

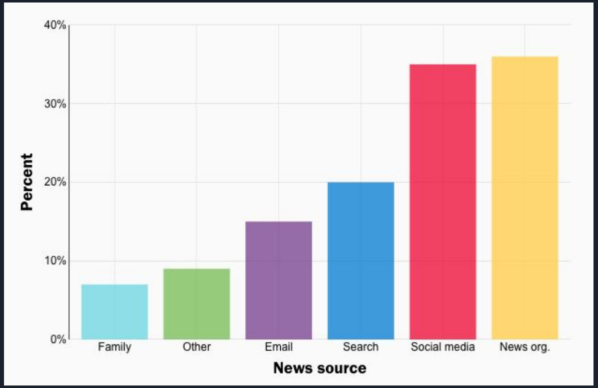
- It's an **ongoing** problem,

  Came across a very apt headline(CNN) to 'stress' the importance of identifying fake news,
  "**No matter who wins the US election, the world's 'fake news' problem is here to stay**"

  So, the job of identifying fake news accurately and delisting it is going to be there always!

Our Solution

- Prevalence of Fake News in media and online social media platforms are skyrocketing,



- It is therefore, need of the hour to check the authenticity of the news before it spread across the society.

- This project emphasises on providing solutions to the community by providing a **reliable platform** to check the Authenticity of the news.

- Our proposed solution is a **Robust Text Classification System using advanced data mining techniques supported by machine learning models.**

- Our goal is to develop a model that classifies a given article as **'Fake' or 'Fact(Real)'**

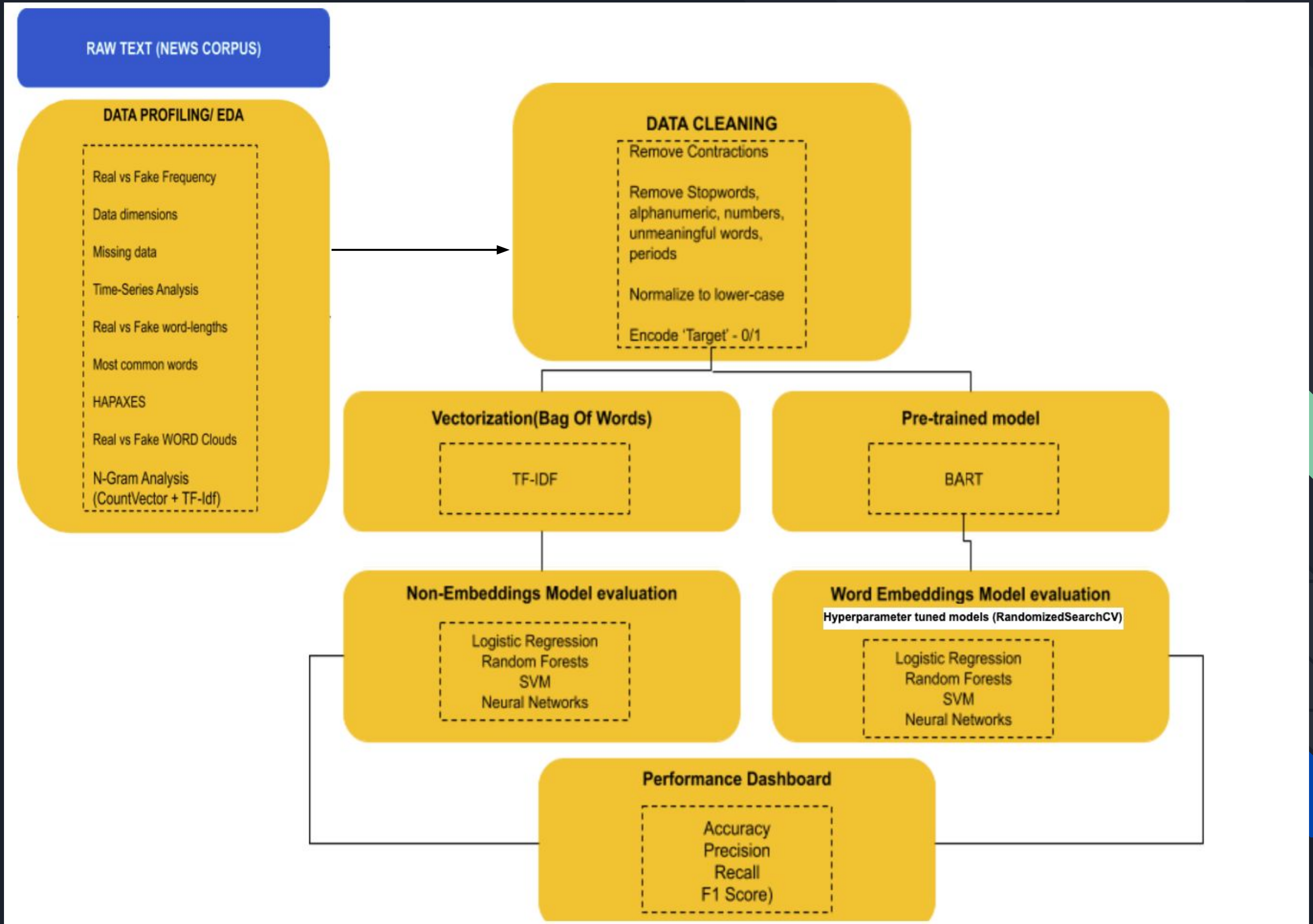
Dataset

- Kaggle Fake News Dataset (https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset)
  - 17,903 Fake News
  - 20,826 True News
- ML Models were overfitting to this dataset.
  - 99% Classification Accuracy
  - 50% Accuracy on different dataset
- To fix this issue, we added additional data from the

  Fake News Corpus (https://github.com/several27/FakeNewsCorpus)
  - 22.5k Fake News
  - 22.5k True News
- Kaggle/ Corpus data is already labeled as 'Fake' or 'Fact(Real)' to be used for comparing our model predictions.

title	text	subject	date
Donald Trump Sends Out Embarrassing New Year's Eve Message; This is Disturbing	Donald Trump just couldn t wish all Americans a Happy New Year and leave it at that. Instead, he had...	News	December 31, 2017
Drunk Bragging Trump Staffer Started Russian Collusion Investigation	House Intelligence Committee Chairman Devin Nunes is going to have a bad day. He s been under the as...	News	December 31, 2017

High-level implementation



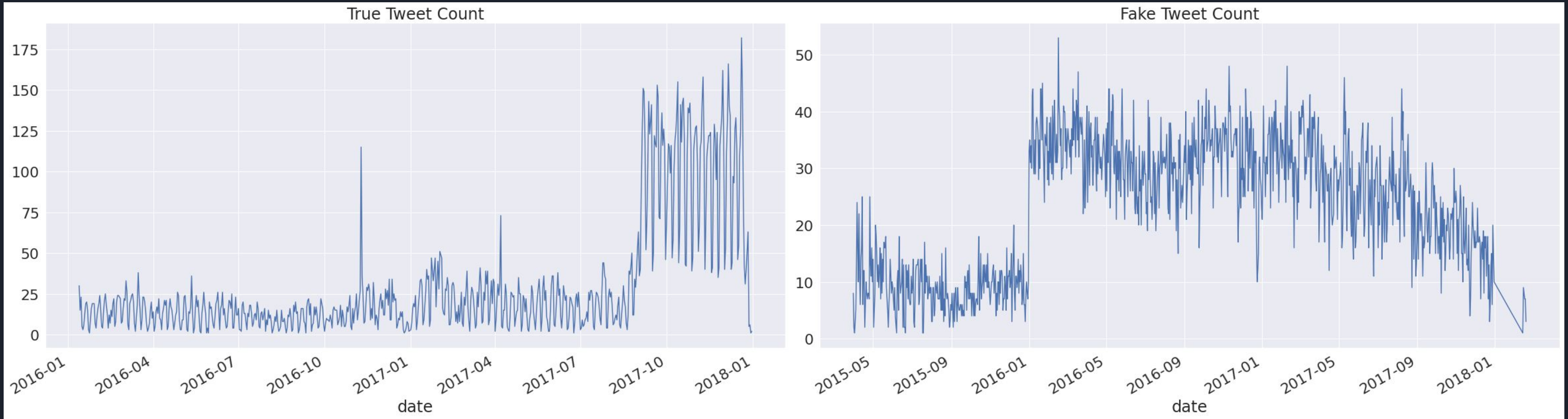
EDA

Extensive EDA on the dataset allowed us to make key decisions while preprocessing our data and training our models.

Primary motivation here is to get a sense of what our data is actually representing.

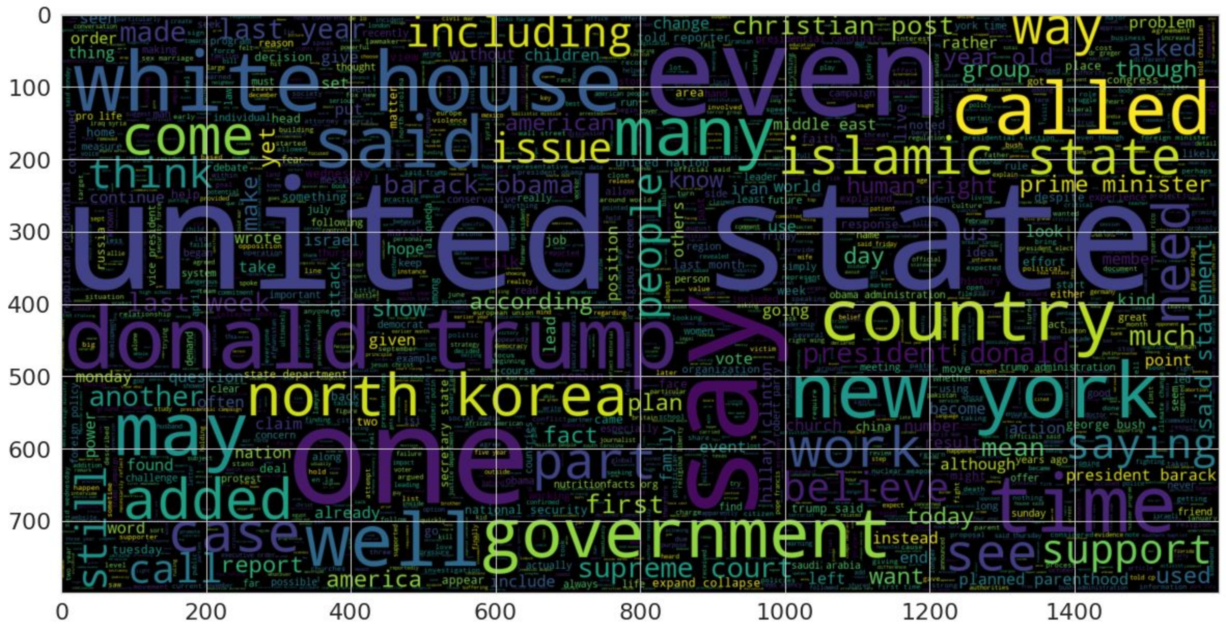
- Time Series Analysis
- Word Frequency Analysis: (‘Reuters’, ‘Washington’)
- Word Clouds
- N-gram analysis

EDA #1 Time Series Analysis



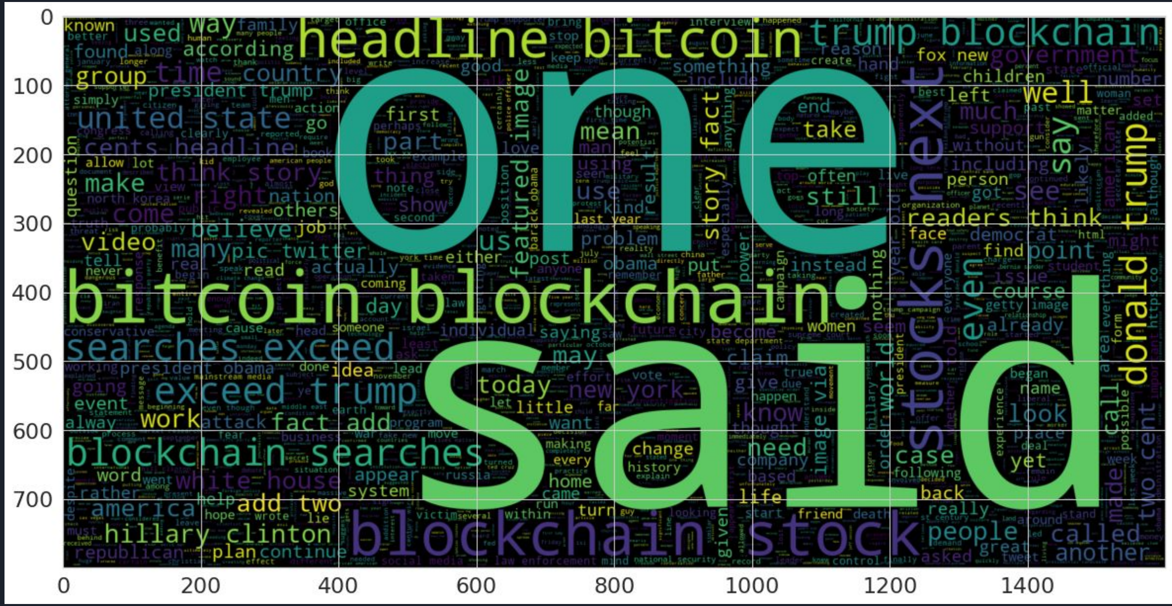
- Charting the frequency of Tweets for both Real and Fake news
- Frequency of both are the same for most of the timeline except for jump at the end for Real news
- Possible explanations could be the midterm elections (Real news) and Trump's inauguration (Fake news)

EDA #2 True News Word Cloud



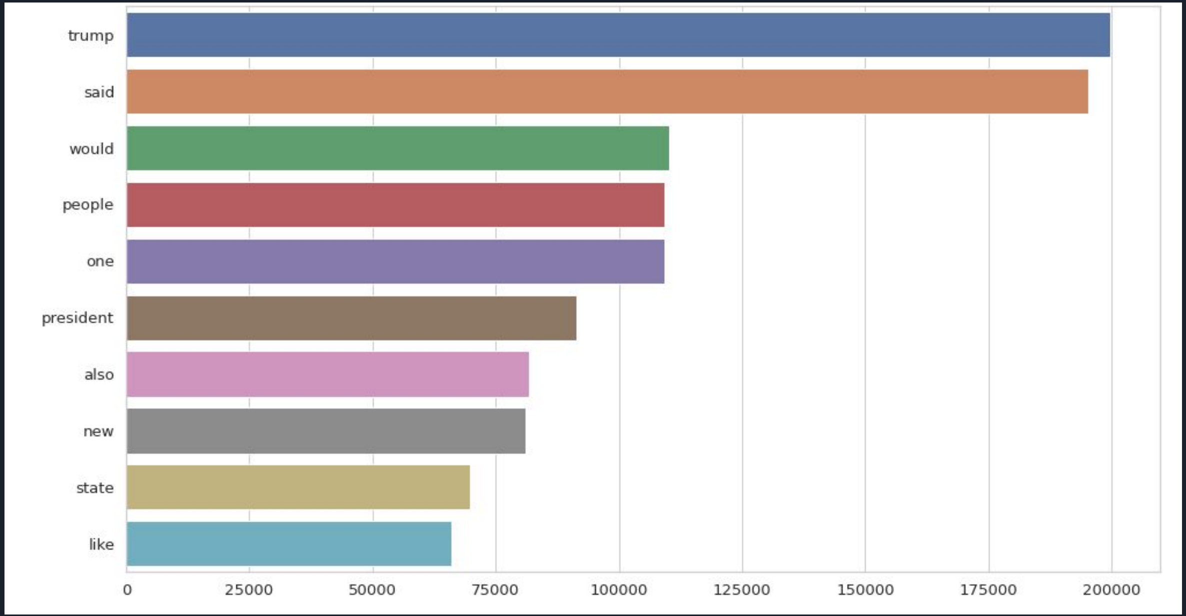
- Constructed with the *wordcloud* library in python
- Allows for high level view of word frequency at a glance
- Stop words and key indicator words removed (e.g. *reuters*, *washington*)

EDA #3 Fake News Word Cloud



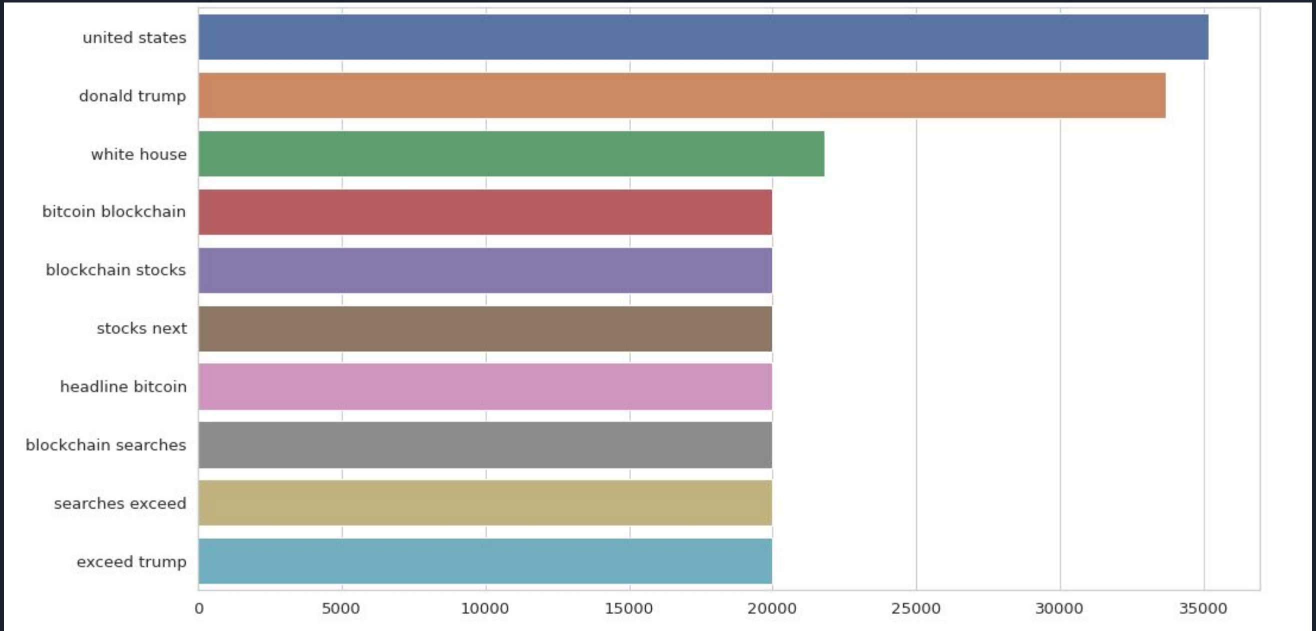
- At the time of sampling, bitcoin was nearing its all-time-high of around $19,800

EDA #4 Unigram Analysis



- Taken across all news (Real and Fake)
- Stop words removed as well

EDA #5 Bigram Analysis



- Reveals more insight to the relatedness of the documents
- Again, see the bitcoin topic appear more heavily

Preprocessing

Goal

- Reduce dimensions and complexity of ML models.
- Generalize the model to avoid overfitting.

General Preprocessing

1. Lower case text
2. Convert contractions to their full form
3. Removed stop words, punctuations and numbers.
4. Lemmatized words.

Stop Words

These words include:

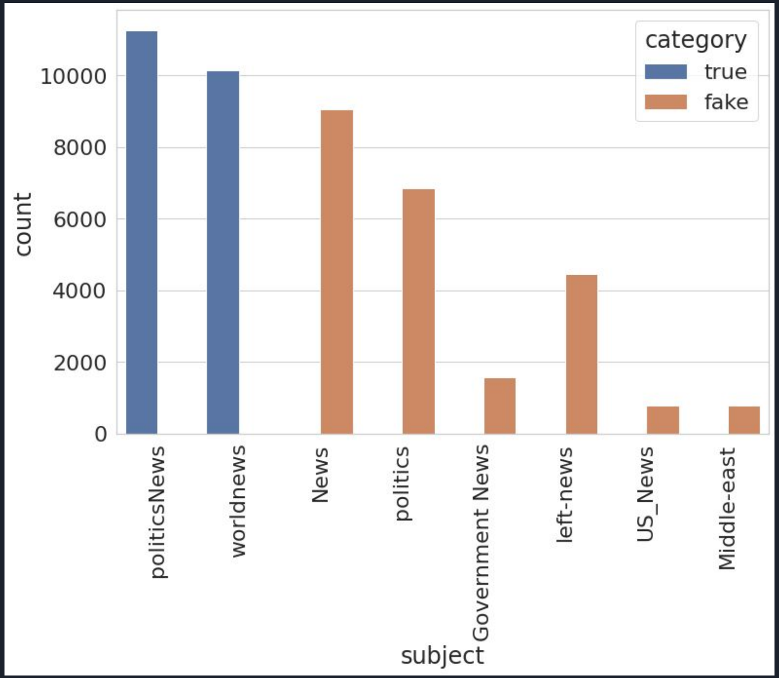
- a
- I
- the
- in
- of
- for
- at
- to
- on
- with
- from

This IS JuSt an Example	—>	this is just an example
Aren’t, isn’t, we’ll	—>	are not, is not, we will
This text is with stop words	—>	~~This~~ text ~~is with~~ stop words
playing, played, player	—>	play~~ing~~, play~~ed~~, play~~er~~

Preprocessing

Dataset-specific Preprocessing

- We found that classification can be done with subject field alone.
- We also found that the words “reuters” and “washington” were causing class imbalance in the true news..
- Target column is in text format as fake/true.



Text Representation

- ML models work with numerical data
- Convert text to numbers.
    - Bag of Words (non-embeddings)
    - Word Embeddings

	Text	Target
Document 1	nsa actually protect readers think story fact ...	0
Document 2	turmeric curcumin pancreatic cancer carcinogen...	1
Document 3	obama greatest criminal history say trump joe ...	0
Document 4	ankara turkey urged united states monday revie...	1

	F1	F2	F3	F4	....	Fn
Document 1	1.0	1.1	0.2	-0.5	....	-0.1
Document 2	0.9	1.3	0.3	0.2	....	-0.8
Document 3	1.3	0.3	0.1	-0.9	....	0.5
Document 4	0.5	1.2	0.5	-0.5	....	0.0

Text Representation

Bag of Words Model

- Represents text as a count of words.
- Example: “I like learning and I like sports and cars.”
- Representation:

Words	I	like	learning	and	sports	cars.
Count	2	2	1	2	1	1

Models

- Count Vectors
- **TF-IDF Vectors**
    - Count vectors but contains scores of how well a word represent a document instead of count.

Text Representation

	Food	Airplane	Fruit	House	Pie
Apple	0.9	0.01	0.99	0.1	0.85

Embeddings

- Semantic *understanding* of language.
- There are too many embeddings models.
  - Used Logistic Regression to score classification accuracy of different embeddings.
- **Use BART word embeddings.**

	BERT	**USE**	**Elmo**	**GloVe**	**Word2vec**	**Albert**	**BART**	**T5**
Train Accuracy	98.4%	96.3%	99%	96.7%	95%	98%	**99.9%**	99.6%
Test Accuracy	97.8%	95.5%	98%	96.9%	94%	97%	**99.4%**	99.0%
Model	bert_base_uncased	use_dan	elmo_bi_lm	wiki_300	google_news_300	albert_base	**Bart-large**	t5-large

Text Representation

Compare classification performance of TF-IDF vectors and Word Embeddings

BART Embeddings	TF-IDF Vectors
89898 x 1024	89898 x 221412
Not sparse	Sparse vectors
Semantic understanding	Word scoring scheme

Neural Networks

```
model = Sequential()

model.add(layers.InputLayer(input_shape=X_train.shape[1]))
model.add(Dense(64, activation='relu'))
model.add(layers.BatchNormalization()) # Batch normalization
model.add(Dense(64, activation='relu'))
model.add(layers.BatchNormalization()) # Batch normalization
model.add(Dense(64, activation='relu'))
model.add(layers.BatchNormalization()) # Batch normalization
model.add(Dense(64, activation='relu'))
model.add(layers.BatchNormalization()) # Batch normalization
model.add(Dense(64, activation='relu'))
model.add(layers.BatchNormalization()) # Batch normalization
model.add(Dense(64, activation='relu')) # 4 dense layers
model.add(Dense(1, activation = 'sigmoid'))

model.compile(loss ="binary_crossentropy", optimizer='adam', metrics = ['accuracy', keras.metrics.Precision(), keras.metrics.Recall()])
model.summary()
```

- Two independent models (embedding & non-embedding data)
- Primary issue was to deal with overfitting through regularization methods
- Training and testing proved to be challenging for the non-embedding model
  - Potentially the reason why it scored lower than other models

Results

- Non-embeddings (TF-IDF vectors) performed better.
    - Presence/absence of words plays larger role than meaning and positioning.
- Neural Networks trained on BART embeddings had the best performance.
- Logistic Regression and SVM had highest average accuracy across both embeddings and non-embeddings.

	Logistic Regression	Random Forests	SVM	Neural Networks
Non-Embeddings	98.5%	96.8%	98.1%	93.2%
Embeddings	94.8%	93.3%	95.4%	98.8%

Model Takeaways

Dataset:

- Nature of dataset presented a relatively clean and linearly separable set of classes.
- TF-IDF vectors are sparse due to large number of dimensions.

SVM

- Works well with sparse data
- Linear kernel suited to maximizing binary classification results

Logistic Regression

- Performs well due to the separability of the classes.
- Works well with high dimensional data and binary classification

Random Forests

- Tree-based model, hence a simpler model than SVM and Logistic Regression.
- Additional Feature Engineering can help => Less Overfitting

Key Findings

- Accuracy vs. Speed (Logistic Regression vs. SVM)
- Although TF-IDF vectors performed better, it might not be the best solution to deploy.
  - Inferring new unseen documents requires re-training all the documents.
  - 200k dimensions (mostly sparse) means inefficient.
  - Countvectorizer might be a better option.
- Word embeddings do not always improve the performance of a model. (Keep it Simple)
- Dimensionality reduction through PCA can significantly reduce training time in the future.
  - Tradeoffs with accuracy
- Neural Networks did not provide significant improvement.
- Future Works - Need to train on much larger corpus of data to avoid overfitting and to make model more robust.

Questions?