Fake News Detection

Anvith Maddipoti, Jessica Nguyen, Caitlyn Pratt, and Lucy Schilling

School of Information, University of Texas at Austin

I 320D: Applied Machine Learning with Python

Professor Abhijit Mishra

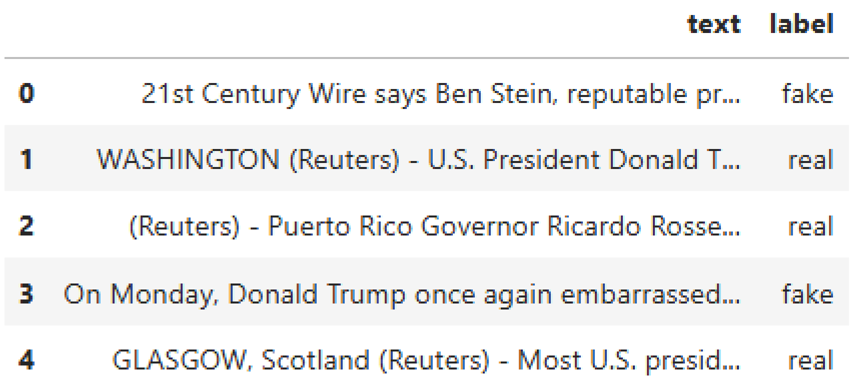
December 11, 2025

Fake News Detection

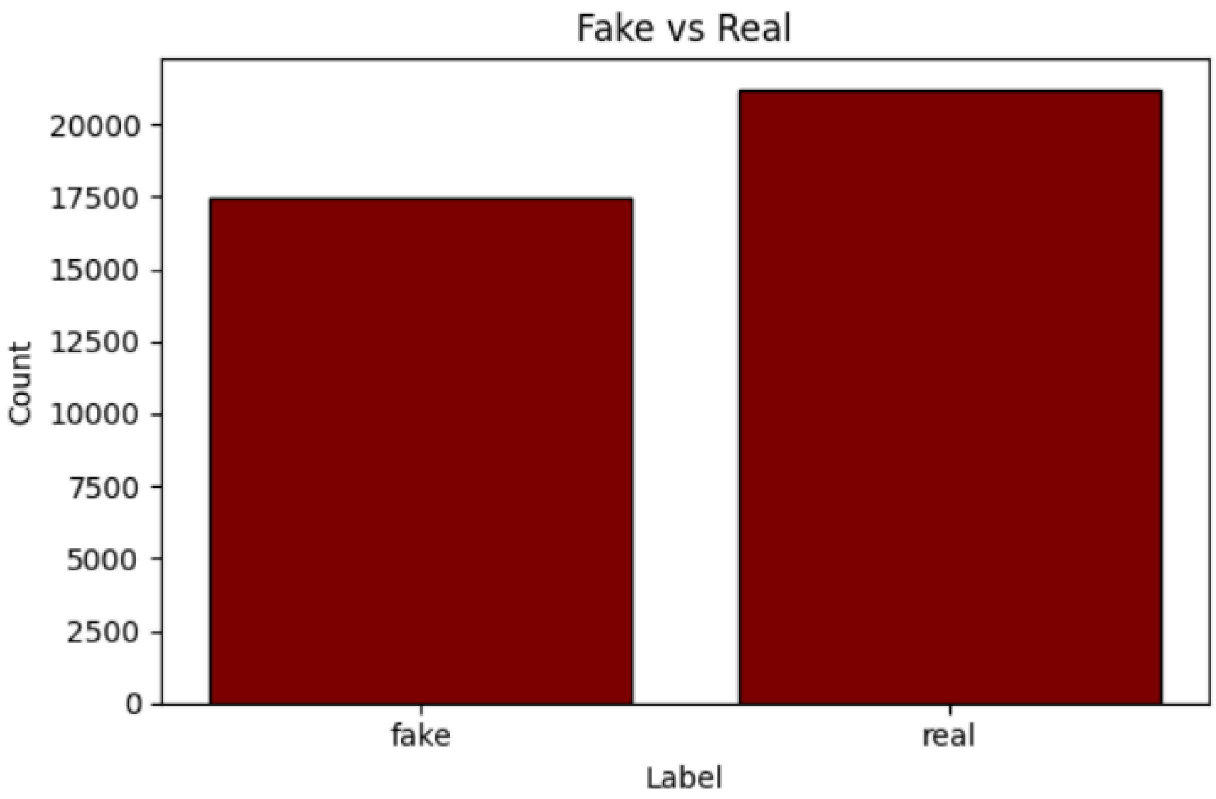
With the increasing rise of sophisticated AI-generated media such as images, videos, and news articles, misinformation and disinformation have become increasingly widespread. The rapid spread of false information has made it difficult for an average person to distinguish between reliable and unreliable news sources, inherently serving as a catalyst for the continuous cycle of misinformation distribution. As a result, this issue stresses the demand for developing tools that can assess the credibility of content in an unreliable media landscape. This project aims to address that by informing people on what to trust and slowing the spread of misinformation through its ability to accurately determine whether a news article is genuine or fabricated. The overall objective of this project was to create a machine learning model that can identify if a written article is considered real or fake. A dataset consisting of files containing real and fake news was utilized to train and evaluate the accuracy of the text-based model. By applying machine learning techniques such as preprocessing, sentiment analysis, building a long short-term memory (LSTM) model, creating a receiver operating characteristic (ROC) curve, and using a local interpretable model-agnostic explanations (LIME) package, we were able to build a model that predicts based on semantic patterns.

Data Description

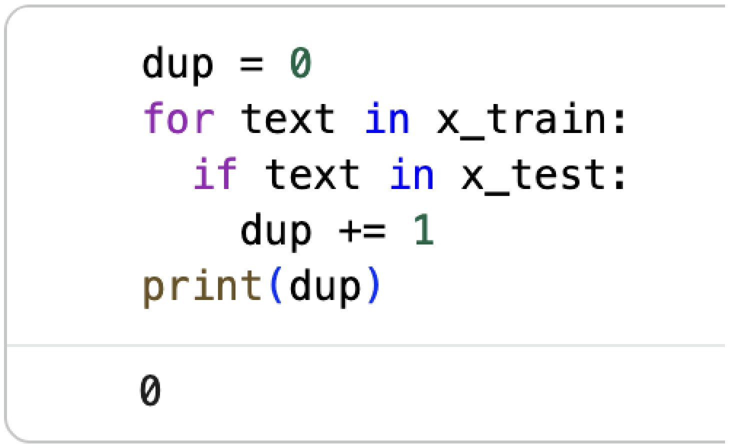
The project utilized a dataset from Kaggle, Clementibasillon (2024), containing two files including real and fake news articles. To prepare the dataset for our model, the two data frames were labeled with their respective classification, consolidated into one data frame, and shuffled to remove any ordering bias prior to splitting and training. The title, subject, and date columns as well as any NaN and duplicate values within the dataframe were dropped. After pre-processing, we were left with the text and label column consisting of 21,191 real articles and 17,455 fake articles, totaling 38,646 rows.



	text	label
0	21st Century Wire says Ben Stein, reputable pr...	fake
1	WASHINGTON (Reuters) - U.S. President Donald T...	real
2	(Reuters) - Puerto Rico Governor Ricardo Rosse...	real
3	On Monday, Donald Trump once again embarrassed...	fake
4	GLASGOW, Scotland (Reuters) - Most U.S. presid...	real



Since the label column is categorical, LabelEncoder from sklearn.preprocessing was applied to convert the categories into 0 and 1, representing fake and real articles respectively. The tokenizer from tensorflow.keras.preprocessing.text package processed the text data and transformed words into integers, and built a vocabulary that included the top 5,000 most frequent words. To ensure the sequences were the same length, all sequences were padded to a maximum length of 1,000. Lastly, the final data frame was split into 80% training data, 10% validation data, and 10% test data for effective model development. To check for data leakage, a for loop was used to test if any texts in the training dataset appeared in the testing dataset. No texts appeared to be the same.

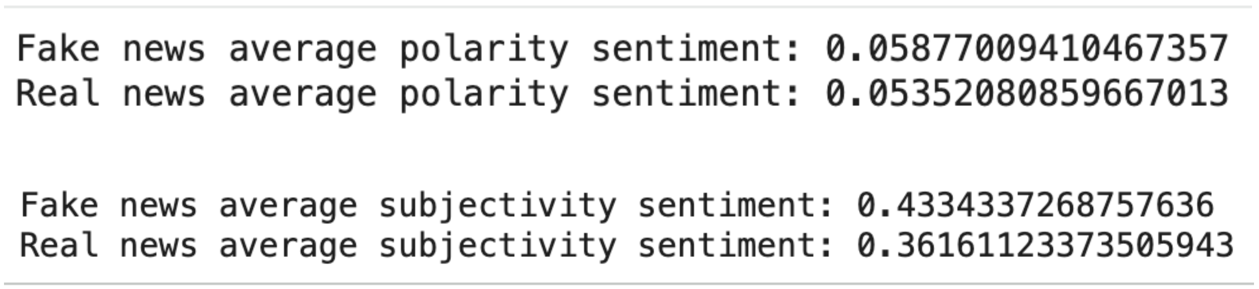


```
dup = 0
for text in x_train:
  if text in x_test:
    dup += 1
print(dup)
```

```
0
```

Sentiment Analysis

Sentiment analysis from the TextBlob package was used to calculate the emotional tone of text data. TextBlob was built on the shoulders of the Natural Language Toolkit (nltk) for understanding text data. The original dataframe was copied into a new variable to prepare the text for sentiment analysis without altering the original dataset for later usage. Characters other than letters, numbers, and individual spaces were removed. Additionally, a defined list of stopwords were removed. Sentiment Analysis from TextBlob can calculate polarity and subjectivity scores. The TextBlob package uses an API and a lexicon based approach to assign polarity and subjectivity scores to each word. Polarity is scored from [-1,1] representing the range between negative sentiment and positive sentiment respectively, and subjectivity scores range between [0,1] representing the range between objectivity and subjectivity respectively. The scores are aggregated across sentences, handling cases for negation (flip the meaning of the proceeding word) and intensifiers (intensify the meaning of the next word). The dataframe was separated into real and fake datasets based on the label column, and new columns were created for each text row's subjectivity score and polarity score. The polarity and subjectivity scores were averaged for both datasets. The polarity scores for both real and fake texts were neutral. However, the subjectivity score for fake texts was slightly higher than the subjectivity score for real texts, showing that fake texts are slightly more subjective than real texts.



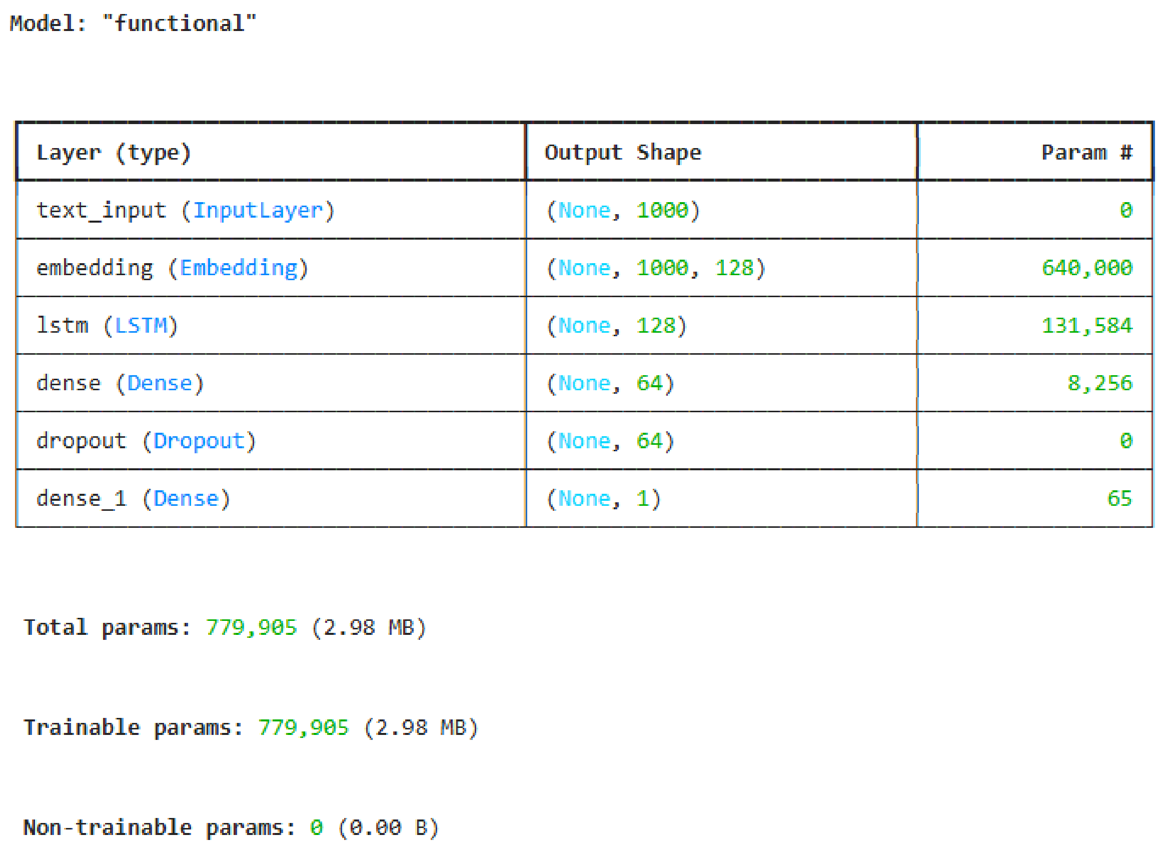
```
Fake news average polarity sentiment: 0.05877009410467357
Real news average polarity sentiment: 0.05352080859667013

Fake news average subjectivity sentiment: 0.4334337268757636
Real news average subjectivity sentiment: 0.36161123373505943
```

Methodology

Long Short Term Memory Model

The training data was fitted to a custom LSTM model using the tensorflow.keras.layers package. LSTM models are a type of recurrent neural network that takes long term memory in the cell state and short term memory in the hidden state into account to detect the semantic patterns in text data. The first layer of our model consisted of an input layer that takes in the tokenized words, represented as a sequence of integers. The next layer was an embedding layer that converted the word indexes into a 128 dimensional vector. Afterwards, the LSTM layer reviewed tokens one at a time, considering what data to store in the cell state and hidden state. The dense layer takes in the findings from the LSTM layer, finds the non linear patterns, and outputs a 64 dimensional layer. The dropout layer prevents overfitting by deactivating 50% of nodes to make the model more robust. The dropout for this project was high because there were concerns for the model overfitting to the data. Lastly, the output layer uses sigmoid activation for binary classification. The model was compiled with a binary_crossentropy loss to penalize the model for incorrect classification for a binary outcome, an adam optimizer to update the models weights, and accuracy as the metric. Additional dense and dropout layers were added to the model in an attempt to prevent overfitting, but each change produced similar results. The model was trained with 32 samples at a time in the training dataset, as defined by the batch size. The model was trained over 10 iterations through the entire dataset, as defined by the number of epochs. After each epoch, the model was evaluated on the validation dataset.



```
Model: "functional"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ text_input (InputLayer)         │ (None, 1000)           │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ embedding (Embedding)           │ (None, 1000, 128)      │       640,000 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ lstm (LSTM)                     │ (None, 128)            │       131,584 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ (None, 64)             │         8,256 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout (Dropout)               │ (None, 64)             │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                 │ (None, 1)              │            65 │
└─────────────────────────────────┴────────────────────────┴───────────────┘

 Total params: 779,905 (2.98 MB)

 Trainable params: 779,905 (2.98 MB)

 Non-trainable params: 0 (0.00 B)
```

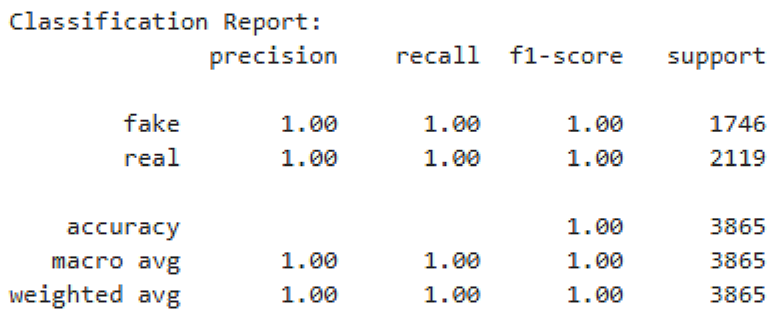
RoBERTa Model

We initially implemented a RoBERTa model as our approach to the fake-news classification task, as it is a widely used transformer model known for its strong performance on natural language processing (NLP) tasks. It was trained on the same cleaned dataset that we later used for the LSTM model in order to maintain consistency. However, the RoBERTa also achieved high accuracy on the dataset, which posed similar concerns of overfitting. Since the RoBERTa's accuracy was extremely high, we transitioned into creating and using the LSTM model to investigate whether a different architecture would perform differently.

Results

Model Performance

Our LSTM model performed exceptionally well, scoring a test accuracy of 99.84%. The classification report from the sklearn.metrics package showed perfect precision, recall, and F1-scores of 1.00 for both the fake and real news categories. These results show that our model correctly classified nearly all the test samples correctly, with minimal errors.



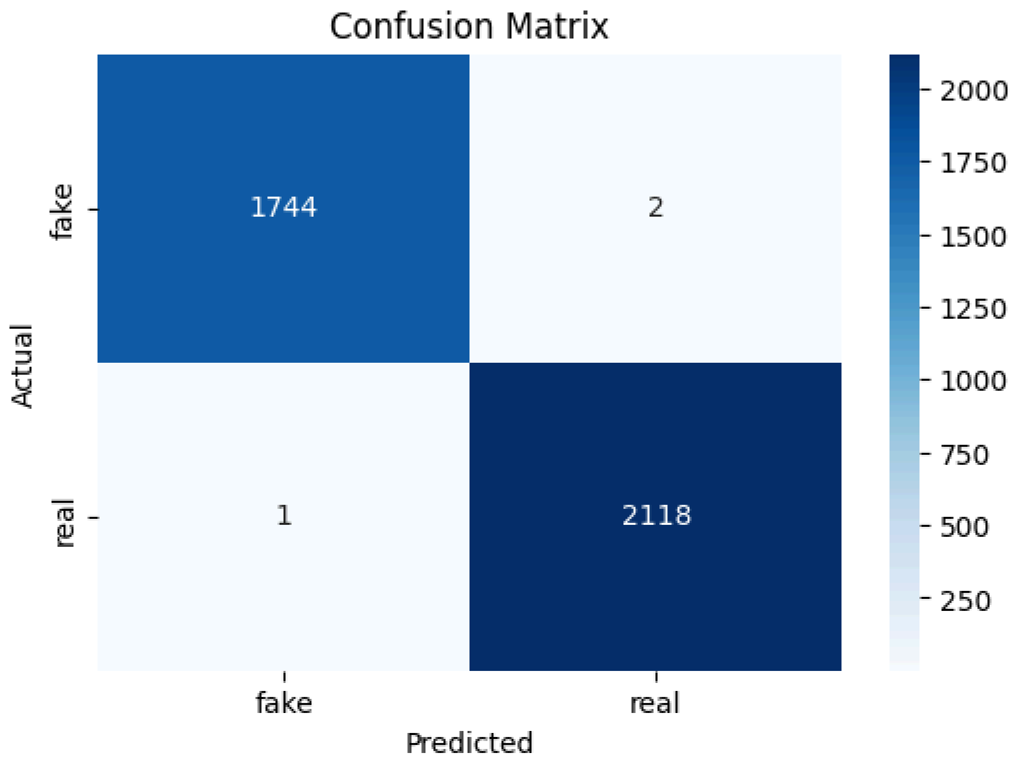
```
Classification Report:
              precision    recall  f1-score   support

        fake       1.00      1.00      1.00      1746
        real       1.00      1.00      1.00      2119

    accuracy                           1.00      3865
   macro avg       1.00      1.00      1.00      3865
weighted avg       1.00      1.00      1.00      3865
```

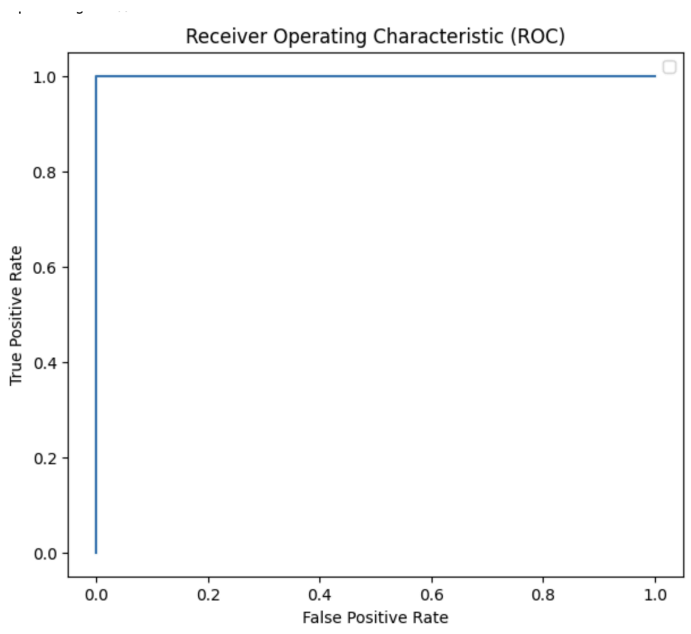
Error Analysis

Despite our near perfect accuracy, the confusion matrix diagram reveals a minute number of misclassifications in the test set of 3,862 articles. In specifics, the model produced 1 false positives, meaning 1 real articles were incorrectly classified as fake, and 2 false negatives, where fake news articles were incorrectly classified as real. The model correctly identified 1,744 fake articles and 2,118 real articles, suggesting that the model either learned highly distinguished patterns between the two classifications, there is a potential data leakage - although we could not identify any, or the inherent separability of the dataset.



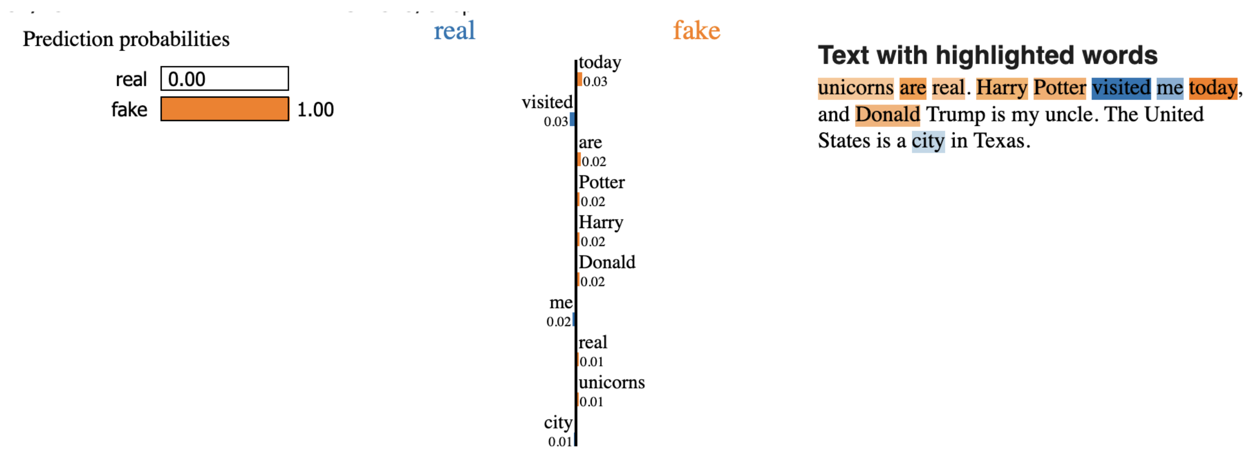
ROC Curve Analysis

The Receiver Operating Characteristic (ROC) curve visualizes the LSTM model's strong classification ability on the test data. The ROC curve compares the false positive rate, or the rate at which real articles are classified as fake, against the true positive rate, which compares the rate at which fake articles were correctly identified. It exhibits a sharp rise to the upper left corner, achieving a true positive rate of near 1.0 with minimal false positives. This shape, most closely resembled by a 90º angle, indicates that the test data is easily classified by the model with minimal confusion between both outcomes. Furthermore, the curve just further shows the model's high confidence predictive ability.



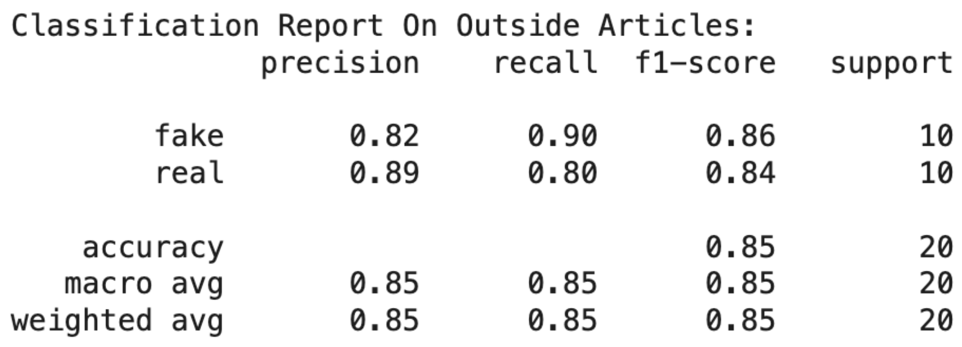
LIME Explainer

A text explainer from the LimeTextExplainer package was imported to further evaluate our LSTM model. A function was created to take in a string, pad and tokenize the text, run the text through the model, and return the probabilities that the text belongs in either class. The explainer was defined with the class names real and fake. The explainer takes in a sample text, runs it through the model prediction function, and shows how the top 10 words contributed to the model output. However, the explainer does not show the semantic patterns that also contribute to the overall prediction of texts. The explainer was less intuitive on the texts from the dataset, so an extremely fake sentence was created to show how different words affect the prediction of the model.



Real World Testing

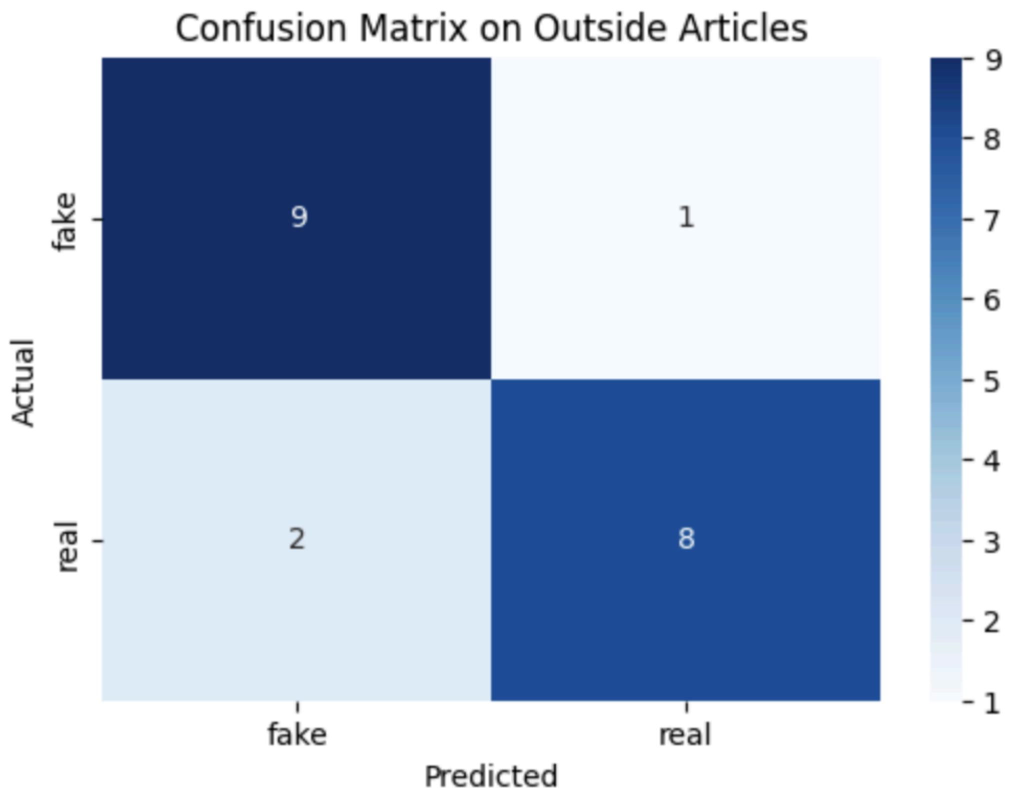
To evaluate the LSTM model's ability in the real world, the primary use case for the model, we tested it on 10 current articles from a reputable news source, The British Broadcasting Corporation (BBC), and 10 more on articles from a satirical magazine, The Onion. The URLs for each article were appended to their respective list, one for fake news and the other for real news. The texts were extracted from these articles using the BeautifulSoup package and appended to lists that were converted into two dataframes for fake news and real news. A column was created to assign the fake and real label to each dataframe. The dataframes were concatenated and shuffled, to create an outside source testing dataframe of 20 samples. The new dataframe was treated similarly to the original datasets test data; the text data was padded and tokenized, the label column was encoded into 0's and 1's, and the model's accuracy was evaluated on the preprocessed data. A classification report and confusion matrix was created from the results.



```
Classification Report On Outside Articles:
              precision    recall  f1-score   support

        fake       0.82      0.90      0.86        10
        real       0.89      0.80      0.84        10

    accuracy                           0.85        20
   macro avg       0.85      0.85      0.85        20
weighted avg       0.85      0.85      0.85        20
```



Further Discussion of Results

Although the model performs incredibly well in testing, it warrants further consideration. The near perfect accuracy and the shape of the ROC curve implies that either the data set consists of highly differentiable characteristics between real and fake news or there may be unintended data leakage. The model's performance on real world applications suggests that the model may have overfitted to the dataset, even with the implementation of overfitting prevention methods like dropout. Our model was somewhat able to determine whether or not an article was fake based on the outside articles, but our product needs more robust training before deployment.

Conclusion

Key Findings

The LSTM model we created to differentiate real and fake news articles achieved a near-perfect accuracy and ROC curve on training and test datasets. The results indicate that the model is able to learn the textual patterns that differentiate between fake and real articles, such as the word choice, context, and writing style. Both the embedding and LSTM layers allowed the model to capture the semantic information in the articles effectively and showed through in the model's training and testing. The model's performance was also reflected in the precision, recall, and f1-scores, where they were all 1.0. Overall, the findings indicate that the model is able to effectively distinguish the textual features in the articles to predict real and fake news.

Limitations

Despite the high accuracy in both the training and testing, the model showed clear weaknesses when evaluated on articles outside the dataset. The most significant limitation of the model is that it often fails to classify real-world articles correctly, sometimes labeling them as fake, which suggests that the model is not able to generalize well beyond the dataset. Since the dataset only contained the text and labels, the high accuracy may be due to the writing style or differences in topics rather than the semantic distinctions of the articles. The differences between the classes may have made them easier to separate, suggesting that the data may have been extremely differentiable. As a result, the model's near-perfect performance on the training and test data might not translate well into real-world fake-news detection.

Roadbumps

Mashayekhi (2025) was the first dataset we adopted for this project, which was from Kaggle as well. The dataset was preprocessed and split similarly to the previously described dataset. A custom LSTM and RoBERTa model were trained on this dataset with similar methods. However, both models resulted in extremely low accuracy of 50%. After deeply inspecting the dataset, the datacard mentioned the dataset was synthetically generated. The texts contained one randomly generated word after another. The sentences had no meaning or patterns, explaining why both models were unable to detect semantic patterns. This dataset was discarded, and the previously mentioned dataset was adopted for this project.

Future Work and Applications

Given the limitations observed in how the model performed on real articles, future work should focus on improving the model's ability to generalize beyond the dataset it was trained on. One approach is to use a more diverse dataset that includes more topics, time periods, and a wider variety of language to expose the model to different writing styles. Increasing the maximum text length during tokenization could also help the model capture more context from longer articles, and adjusting the architecture or fine-tuning existing layers may help improve the model's capacity to learn complex patterns. Tuning the batch size, learning rate, or dropout rate could also improve the model's performance. We also did not explore RoBERTa's performance into the second dataset due to time constraints, however, future work should also include observing RoBERTa's ability to classify articles in comparison to our current LSTM.

Code and Replicability

To view the code and replicate our results, please refer to the GitHub link below:

https://github.com/aefyb/AML-finalProject

References

Alpert, H. (2022, March 5). *How to use Beautiful Soup to parse text for NLP projects*. Medium. https://halpert3.medium.com/how-to-use-beautiful-soup-to-parse-text-for-nlp-projects-48acc9145f89

Clmentbisaillon (2024, April 19) *fake-and-real-news-dataset* [Data set] Kaggle. https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset/data

Devêci, B. (2025, May). Fake-News Detection with TF-IDF & BERT [Notebook]. Kaggle. https://www.kaggle.com/code/busradeveci/fake-news-detection-with-tf-idf-bert#Fake-News-Detection-Project:-Summary-and-Evaluation

Marcotcr. (2021). LIME: Explaining the predictions of any machine learning classifier [Code repository]. GitHub. https://github.com/marcotcr/lime

Mashayekhi, M. (2025, April 27). *Fake News Detection Dataset* [Data set]. Kaggle. https://www.kaggle.com/datasets/mahdimashayekhi/fake-news-detection-dataset

Rivaldo, R. (2020, December 29). Sentiment-Analysis [Code repository]. GitHub. https://github.com/RichardRivaldo/Sentiment-Analysis?tab=readme-ov-file

Sentdex. (2018, February 27). Sentiment Analysis in Python with TextBlob and VADER Sentiment (also Dash p.6) [Video]. YouTube. https://www.youtube.com/watch?v=qTyj2R-wcks