Fake New Detections

Presented by: Zane Alderfer, James Burrows, Monica Fam, Sandy Leung

Introduction

With rapid technological advancement, fake news appears in many forms, including **fabricated stories**, **misleading headlines**, and **manipulated media**. It spreads quickly on social media, exploiting trust and influencing public opinion. **Fake news detection involves analyzing news content to determine its truthfulness by classifying news as either real or fake**

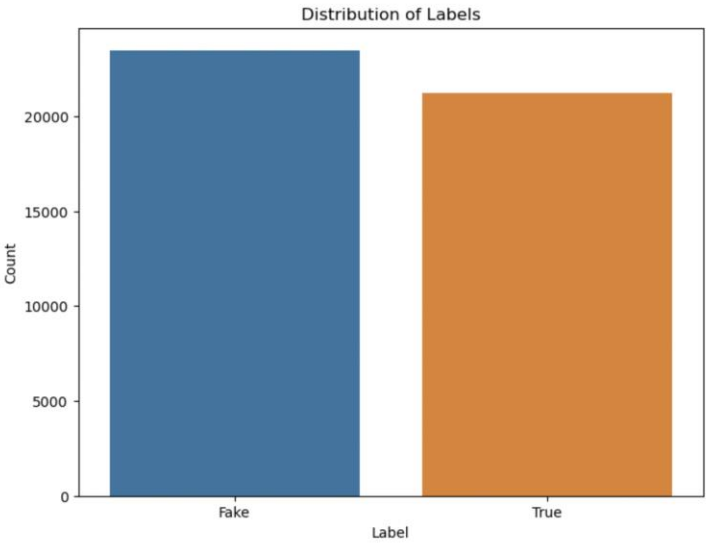
Dataset Overview

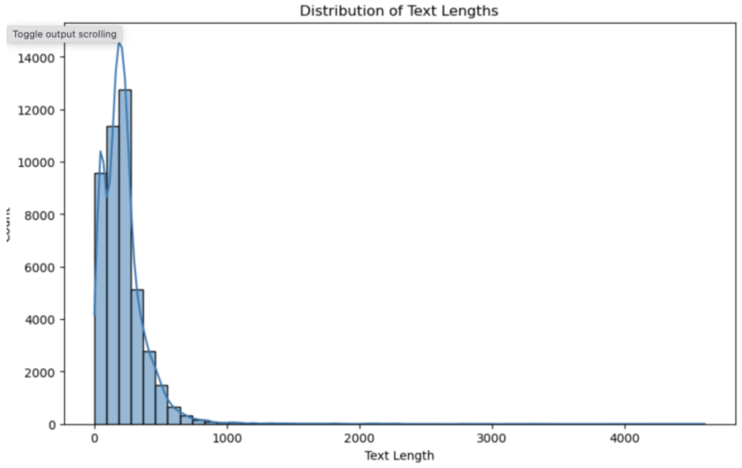
- Two datasets that are split between true news and fake news
- Each dataset contain five columns
  - Index: The index of the data frame
  - Title: The title of the news article
  - Text: The text content of the news article
  - Subject: The subject category of the news article
  - Date: The date the news article was published

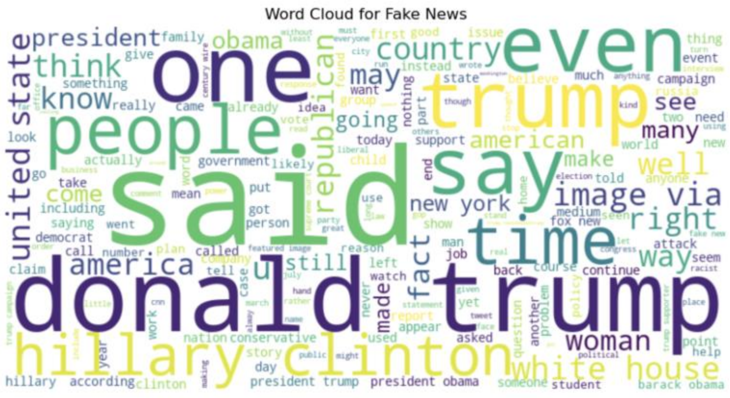
Exploratory Data Analysis (EDA)

- Add 'label' column to identify true and fake news
  - True = 0, False = 1
- Concatenate the true and fake news datasets into one dataset
- Check for any missing values in the dataset
- Drop any duplicate rows
- Tokenize and convert input text to lower case for uniformity
- Filter out tokens that are not alphabetic (e.g., removes numbers, punctuation, etc.)
- **Tools**: NLTK, pandas, sklearn

Visualizations



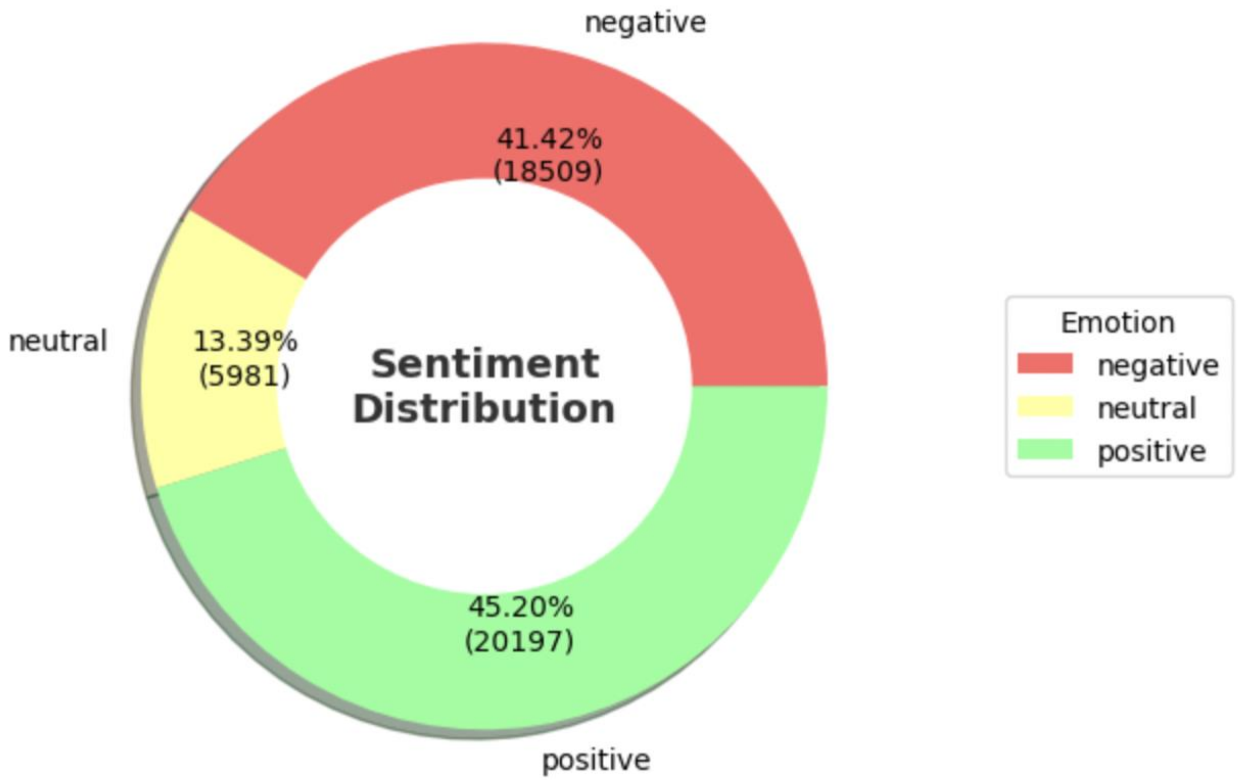




Sentiment Analysis

- Fake new contribute to negative sentiment through sensationalism and emotional manipulation to engage consumers
- True news report on serious or negative events that naturally evoke negative sentiments

- Fake news contribute to positive sentiment through potential propaganda and manipulate emotions
- True new contribute to positive sentiment through uplifting stories, success stories, or positive developments



Convolutional Neural Networks (CNN) Model

- <u>Dataset</u>: fake.csv and true.csv from Kaggle

Steps:

- Merging and labeling datasets
- Preprocessing: Tokenization, stopword removal, lemmatization
- Splitting: Training (64%), Validation (16%), Test (20%)
- **Embedding**: Converts text to dense vectors
- **Conv1D**: 128 filters, kernel size of 5, activation='relu'
- **GlobalMaxPooling1D**: Reduces dimensionality
- **Dense**: 64 units, activation='relu'
- **Dropout**: 0.5 for regularization
- **Output**: 1 unit, activation='sigmoid' for binary classification

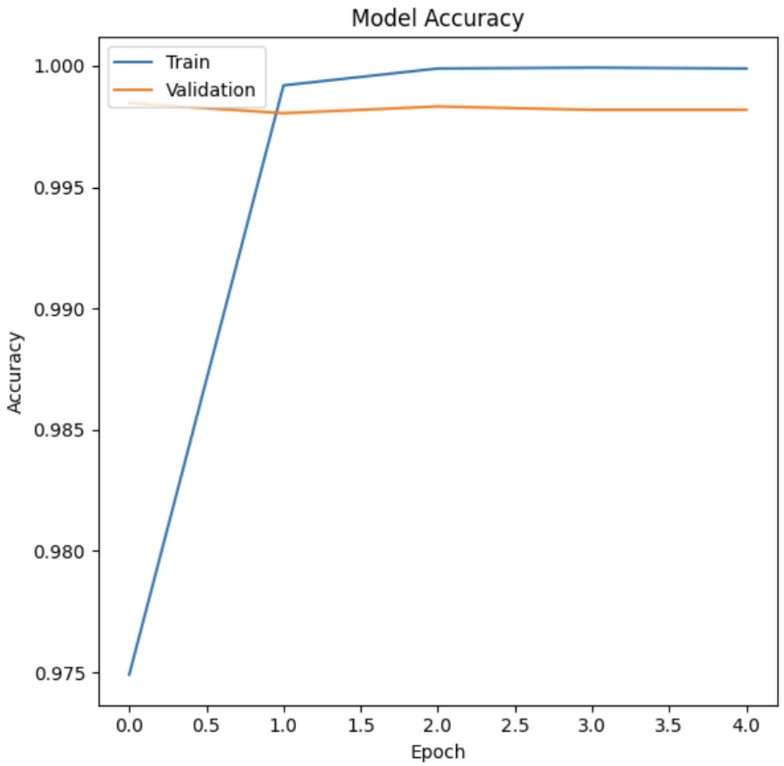
CNN Training & Evaluation

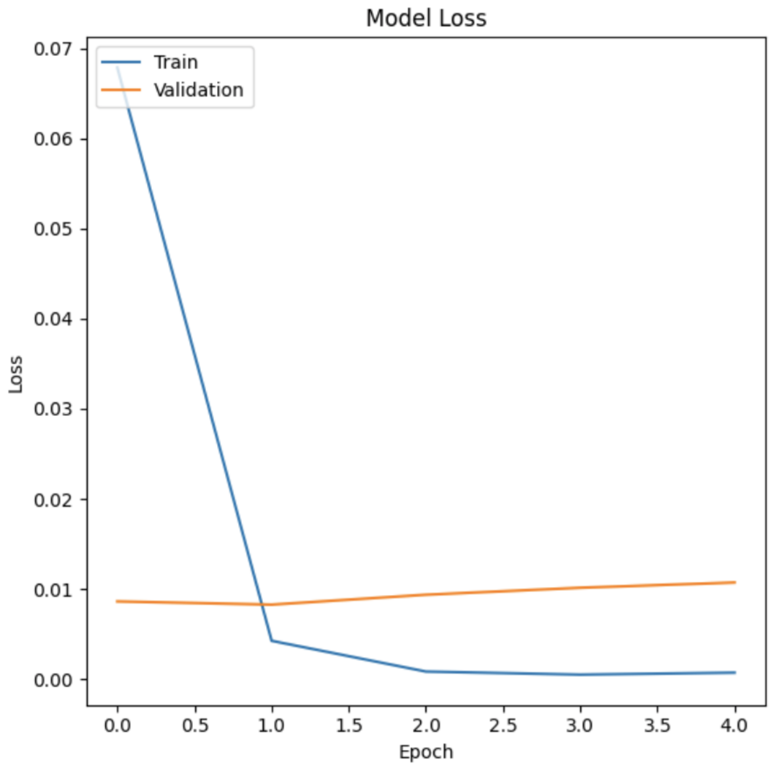
Parameters:

- vocab_size: 20,000
- embedding_dim: 100
- max_length: 200
- epochs: 5

Results:

- **Training Accuracy**: ~99.84%
- **Validation Accuracy**: ~99.85%
- **Test Accuracy**: ~99.84%
- **Observation**: High accuracy indicates effective training, but potential overfitting





CNN Epochs

```
Epoch 1/5
894/894 - 81s - loss: 0.0725 - accuracy: 0.9728 - val_loss: 0.0100 - val_accuracy: 0.9972 - 81s/epoch - 90ms/step
Epoch 2/5
894/894 - 81s - loss: 0.0043 - accuracy: 0.9990 - val_loss: 0.0102 - val_accuracy: 0.9972 - 81s/epoch - 90ms/step
Epoch 3/5
894/894 - 80s - loss: 0.0017 - accuracy: 0.9995 - val_loss: 0.0229 - val_accuracy: 0.9964 - 80s/epoch - 90ms/step
Epoch 4/5
894/894 - 77s - loss: 9.3790e-04 - accuracy: 0.9998 - val_loss: 0.0214 - val_accuracy: 0.9962 - 77s/epoch - 86ms/step
Epoch 5/5
894/894 - 78s - loss: 5.2400e-04 - accuracy: 0.9999 - val_loss: 0.0148 - val_accuracy: 0.9966 - 78s/epoch - 87ms/step
280/280 - 8s - loss: 0.0172 - accuracy: 0.9978 - 8s/epoch - 29ms/step

Test Accuracy: 0.9977623820304871
```

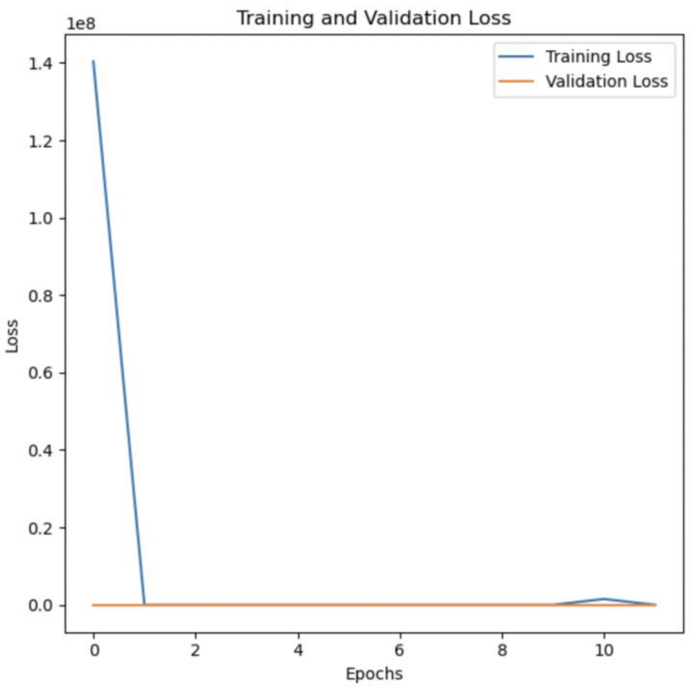
Recurrent Neural Network (RNN) Model

- Splitting: Training (64%), Validation (16%), Test (20%)
- Utilized an RNN architecture for **sequential** data processing
- **Embedding**: Converts text to dense vectors
- **SimpleRNN** layer: 64 units, activation='relu', dropout=0.2, recurrent_dropout=0.2
- **Dense**: 1 unit, activation='sigmoid'
- **Dropout**: 0.5 for regularization
- **Output**: 1 unit, activation='sigmoid' for binary classification
- Incorporated early stopping with a patience of 2 epochs to prevent overfitting

RNN Training & Evaluation

Result

- **Test Loss:** ~18.97%
- **Test Accuracy**: ~92.46%
- The model indicates strong performance, with high accuracy and relatively low loss on the test data



RNN Epochs

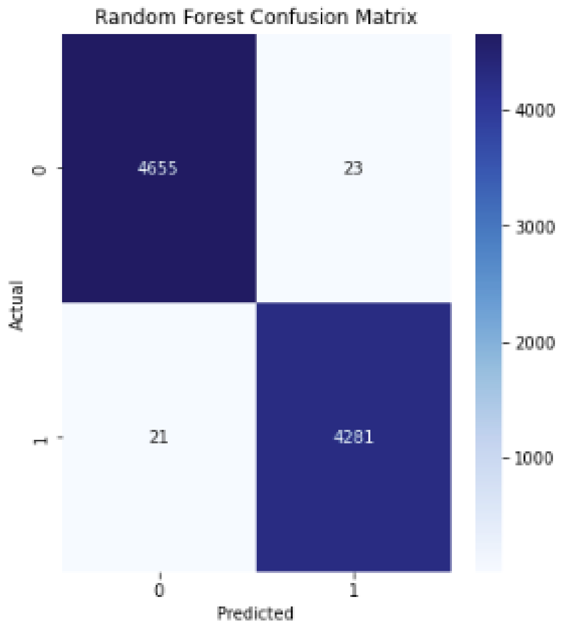
The training set depicts **accuracy improves** and **loss decreases**, indicating that the model is learning from the test data. Early stopping activates by the **11th epoch**, **guarding against overfitting** and **warranting optimal performance** on the validation set.

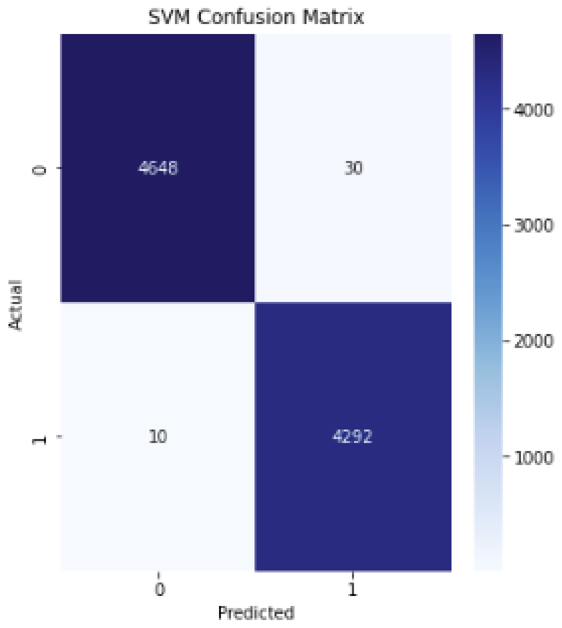
```
Epoch 1/20
1174/1174 ━━━━━━━━━━━━━━━━━━━━ 102s 86ms/step - accuracy: 0.8230 - loss: 199455312.0000 - val_accuracy: 0.8945 - val_loss: 0.2946
Epoch 2/20
1174/1174 ━━━━━━━━━━━━━━━━━━━━ 100s 85ms/step - accuracy: 0.8986 - loss: 1.7498 - val_accuracy: 0.9026 - val_loss: 0.2545
Epoch 3/20
1174/1174 ━━━━━━━━━━━━━━━━━━━━ 103s 88ms/step - accuracy: 0.9137 - loss: 0.2331 - val_accuracy: 0.9116 - val_loss: 0.2322
Epoch 4/20
1174/1174 ━━━━━━━━━━━━━━━━━━━━ 101s 86ms/step - accuracy: 0.9281 - loss: 0.2296 - val_accuracy: 0.9191 - val_loss: 0.2180
Epoch 5/20
1174/1174 ━━━━━━━━━━━━━━━━━━━━ 102s 87ms/step - accuracy: 0.9387 - loss: 0.2129 - val_accuracy: 0.9212 - val_loss: 0.2094
Epoch 6/20
1174/1174 ━━━━━━━━━━━━━━━━━━━━ 102s 87ms/step - accuracy: 0.9431 - loss: 0.1669 - val_accuracy: 0.9232 - val_loss: 0.2022
Epoch 7/20
1174/1174 ━━━━━━━━━━━━━━━━━━━━ 101s 86ms/step - accuracy: 0.9474 - loss: 0.1534 - val_accuracy: 0.9245 - val_loss: 0.1972
Epoch 8/20
1174/1174 ━━━━━━━━━━━━━━━━━━━━ 101s 86ms/step - accuracy: 0.9515 - loss: 0.1414 - val_accuracy: 0.9238 - val_loss: 0.1937
Epoch 9/20
1174/1174 ━━━━━━━━━━━━━━━━━━━━ 101s 86ms/step - accuracy: 0.9548 - loss: 0.1310 - val_accuracy: 0.9255 - val_loss: 0.1900
Epoch 10/20
1174/1174 ━━━━━━━━━━━━━━━━━━━━ 103s 88ms/step - accuracy: 0.9566 - loss: 0.4421 - val_accuracy: 0.9246 - val_loss: 0.1897
Epoch 11/20
1174/1174 ━━━━━━━━━━━━━━━━━━━━ 102s 87ms/step - accuracy: 0.9577 - loss: 1174058.0000 - val_accuracy: 0.9236 - val_loss: 0.1971
Epoch 12/20
1174/1174 ━━━━━━━━━━━━━━━━━━━━ 102s 87ms/step - accuracy: 0.9561 - loss: 0.1249 - val_accuracy: 0.9241 - val_loss: 0.1921
```

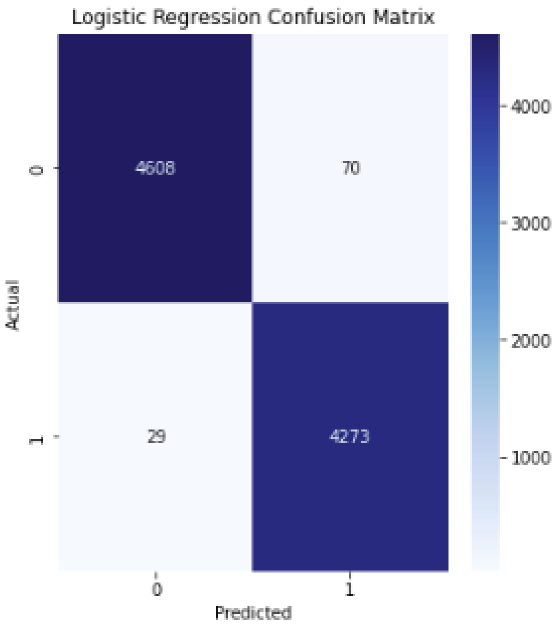
Logistic Regression, Random Forest, SVM

- 3 predictor models were made to establish true vs fake news articles
- All 3 had very high accuracy with SVM having the highest

```
Logistic Regression Accuracy: 0.9889755011135858
Random Forest Accuracy: 0.9951002227171493
SVM Accuracy: 0.9955456570155902

Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      4678
           1       0.98      0.99      0.99      4302

    accuracy                           0.99      8980
   macro avg       0.99      0.99      0.99      8980
weighted avg       0.99      0.99      0.99      8980

Random Forest Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      4678
           1       0.99      1.00      0.99      4302

    accuracy                           1.00      8980
   macro avg       1.00      1.00      1.00      8980
weighted avg       1.00      1.00      1.00      8980

SVM Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.99      1.00      4678
           1       0.99      1.00      1.00      4302

    accuracy                           1.00      8980
   macro avg       1.00      1.00      1.00      8980
weighted avg       1.00      1.00      1.00      8980
```



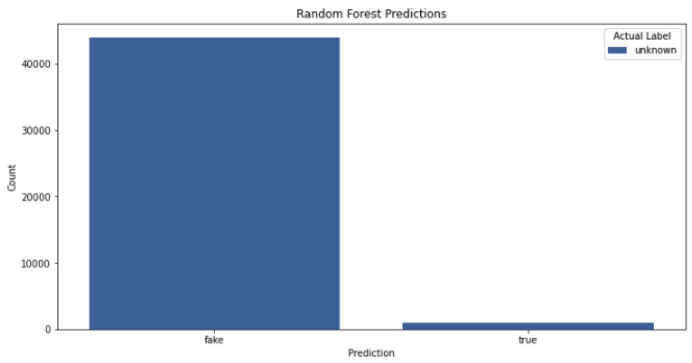


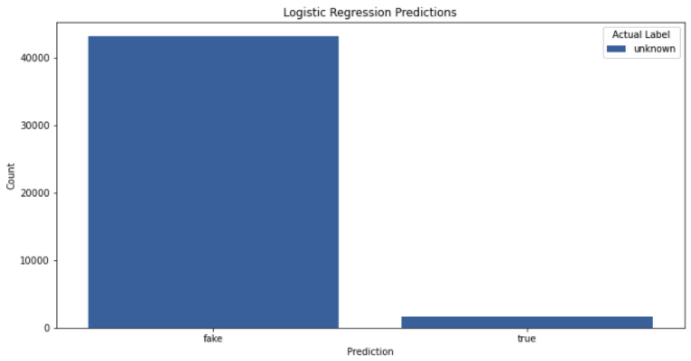


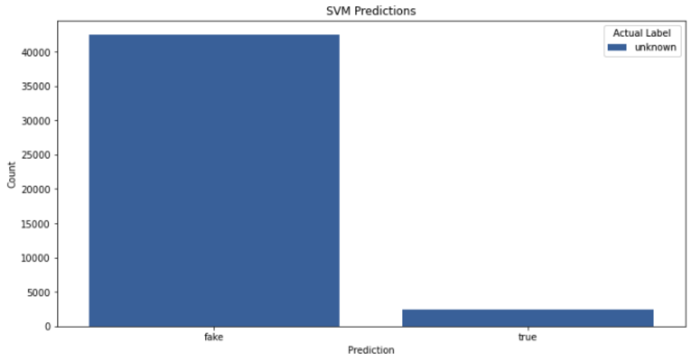
News Domains Ran Against Predictor Models

- 5 Different domain-specific news outlets were ran against the predictor models
- Entertainment, Sports, Science & Technology, Health, and Business
- All news articles were joined together and all 3 predictor models heavily favored identifying the articles as "fake"

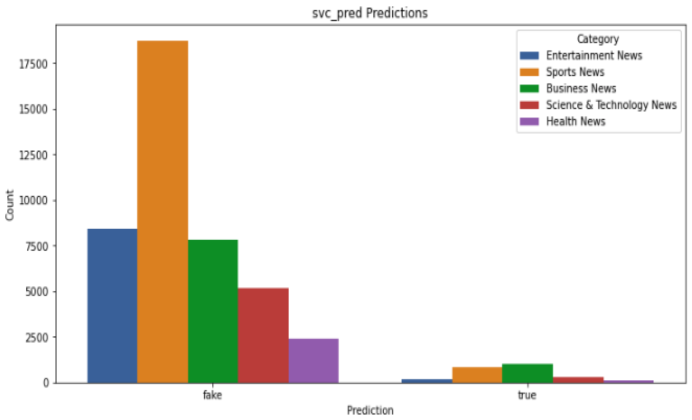
Source: https://newsdata.io/datasets

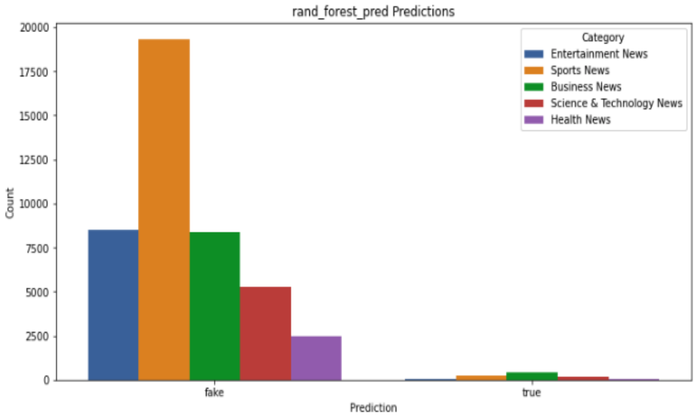


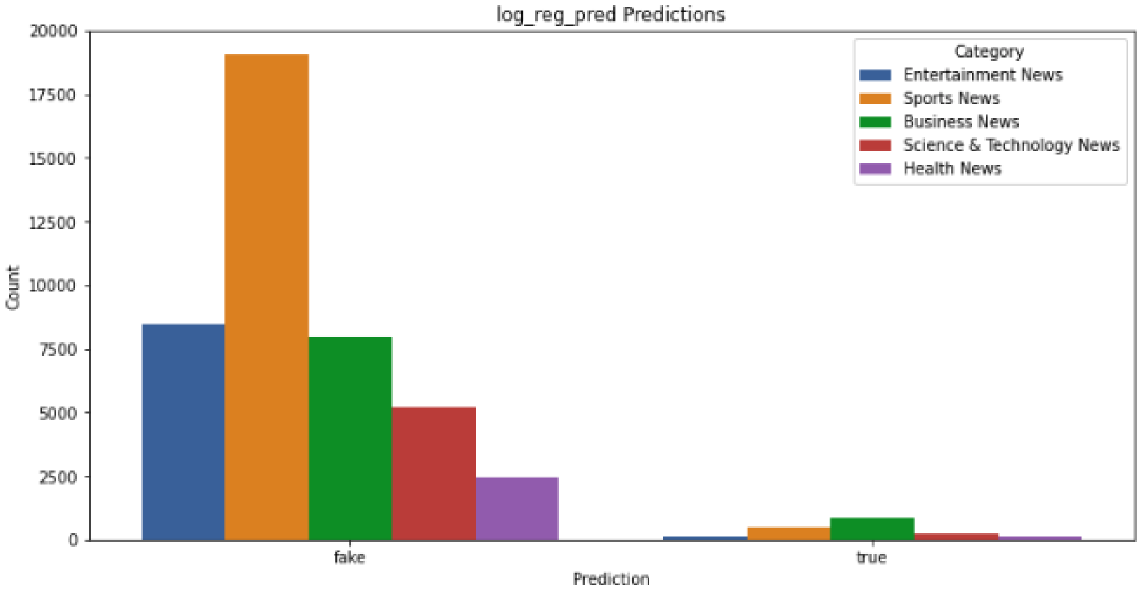




Breakdown by News Domain







DistilBERT: Distillation Bidirectional Encoder Representations from Transformers

- A smaller and faster version of BERT that only uses 66 million parameters instead of BERT's 340 million
- Had to run on GPU due to computational intensity
- Splitting: Training (64%), Validation (16%), Test (20%)
- Tokenizer: DistilBertTokenizerFast
- Code is inspired by Kajal Kamari at Step-by-Step BERT Implementation Guide - Analytics Vidhya
- Used Pytorch instead of TensorFlow
- Model Architecture
  - DistilBERT
    - Uses DistilBERT as the foundation of the model to take advantage of BERT embeddings
  - Dense Layer with ReLU
  - Dropout
    - Used to prevent overfitting
  - Sigmoid
    - Used for Binary Classification

DistilBERT: Smaller, Faster, and Lighter BERT Model (with Python Examples) | PythonProg

DistilBERT Results

- Test Accuracy: 90.83
- Validation Accuracy: 98.51
- Training Loss: 0.54
- Validation Loss: 0.52
- Model was very effective but there are some concerns about it being at risk for overfitting

	precision	recall	f1-score	support
0	0.97	0.98	0.97	1319
1	0.99	0.99	0.99	3181
accuracy			0.98	4500
macro avg	0.98	0.98	0.98	4500
weighted avg	0.98	0.98	0.98	4500

Conclusions

- All three Deep Learning models had high accuracies and low losses
  - There could be potential concerns of overfitting
- Machine Learning tended to heavily classify that all news is Fake News
- Using a transformer is a computationally expensive method and should be considered very carefully