CS 7641 Project Final Report

Group 14

Fake News

Introduction/Background

Literature Review

The spread of misinformation and fake news has been accelerated by the internet and social media. This surge in deceptive information has had a profound impact on societal values, influenced public opinion on critical issues, and even distorted the truth [1]. To address this, researchers are using Machine Learning (ML) and Natural Language Processing (NLP) models to automate fake news detection and determine the authenticity of information more efficiently than manual methods.

Dataset Description

- Kaggle Fake News Dataset: Includes a dataset of news articles labeled as fake or real based on the sources and content.
- FakeNewsNet: Contains data from two fact-checking websites: PolitiFact and GossipCop, with metadata and social context information.
- LIAR Dataset: Includes over 12,000 manually labeled short statements from political fact-checking website PolitiFact. Each statement is classified into six categories from "pants on fire" to "true".

Problem Definition

Problem

The purpose of this project is to build a system that can detect fake news accurately. Fake news is harmful to society [2], and detecting it is a challenging task since a huge amount of news is generated every second. Our goal is to judge whether news is true or false based on its content and metadata.

Motivation

Fake news has real-world significance. It has a crucial impact on public opinion, political stability, and social harmony. As a classic NLP problem, it provides us with a lot of opportunities to explore various machine learning techniques, such as unsupervised learning for feature extraction and supervised learning for classification. In addition, the availability of diverse and well-documented datasets makes this problem a viable subject for experimentation.

Methods

We plan to clean and standardize the data using techniques such as removing punctuation, stop words, and HTML tags, as well as performing tokenization, lemmatization, or stemming. For feature extraction, we will explore Word2Vec, Term Frequency (TF), and TF-IDF [3]. For unsupervised learning, we will focus on K-Means clustering. For supervised learning, we will prioritize Support Vector Machines (SVM), Decision Trees, and Random Forest.

Explanation of Model Choices

- **K-Means** is a widely used clustering algorithm for unsupervised learning, known for its simplicity and effectiveness in grouping data into clusters.
- **Support Vector Machines (SVM)** are chosen for their ability to handle high-dimensional spaces and perform well in text classification tasks.
- **Decision Trees and Random Forest** are included for their interpretability and robustness, with Random Forest offering an ensemble approach that improves prediction accuracy by reducing overfitting.

- **Neural Network** is included with 2 hidden layers of size 100 and 20 resspectively. We explored this option as the power of having many parameters can yield potentially great results, after we have attempted simplier models and evaluated their effectiveness.

Midterm Progress Check

We chose Support Vector Machines (SVM) to do fake news detection, mainly because of two reasons:

- We need to handle a large amount of text data in fake news detection, which typically introduces a high-dimensional feature space after preprocessing. SVMs excel in managing high-dimensional data by identifying the optimal separating hyperplane within this space.
- SVMs offer a variety of kernel functions (linear, polynomial, RBF, etc.), allowing us to experiment with different kernels to find the best fit for our data.

We preprocess the data by cleaning and standardizing it, which includes removing empty entries and eliminating stop words. For feature extraction, we use Term Frequency (TF) [3] to quantify the occurrence of each word in the text.

1. Data Preparation

- We utilized **Pandas** to load the dataset.
- Rows with missing values were **dropped** to ensure data quality.

2. Text Preprocessing

- We downloaded **NLTK stop words** and filtered them out from the text data.
  - Stop words are common words that don’t carry much meaning (e.g., "a", "an", "the").

3. Feature Transformation

- We applied **CounteVectorizer** to transform our text data with a maximum of 50 features for efficiency reasons.
  - This approach replaces the naive **Bag of Words** model with richer vector representations.

4. Model Fitting

- We fitted a **Linear Support Vector Classifier** with:
  - **10000 iterations** in the training for a reasonable training time.
  - A fixed `random_state` for **reproducibility**.

Next Steps

- Hyperparameter tuning is yet to be performed.

Final Report Addition

For our final report, we have additionally implemented a **Neural Network** model with 2 hidden layers of size 100 and 20 respectively. We also implemented the Random Forest model to compare performance with the SVM model. Finally, we consolidated our results from applying K-means clustering to the dataset with 2 clusters, choosing the cluster with the most fake news label as the fake news cluster and the other cluster the true news label.

1. and 2. Data Preparation and Text Preprocessing

- Same as midterm progress check.

3. Feature Transformation

- We applied **CounteVectorizer** to transform our text data with a maximum of 500 features for neural networks.
- Because we are already using neural networks which have a fairly large amount of parameters relative to simpler models, we decide to expand on the vocabulary size to 500 to allow for more features to be considered.
- We applied **Word2Vec** to transform our text data with Random Forest, as we explored other ways to represent text, and **word2vec** is a type of embedding that helps us transform words in higher dimensional space as dense vectors. That way, we don't have to arbitrarily throw out words with a vocab size limit like our previous approach.

- We applied **PCA** to reduce our components to 2 for K-means clustering for better visualization, as well as other number of components to explore the effect.

4. Model Fitting

- We fitted a **2-layer Neural Network** with:
  - **100 and 20 neurons in hidden layers** in the training for a reasonable training time.
  - A fixed `random_state` for **reproducibility**.
- We fitted a **Random Forest** model with:
  - **100 trees** in the forest for a reasonable training time.
  - A fixed `random_state` for **reproducibility**.
- We trained **K-means** clustering with 2 and 10 components to explore the effect of different number of clusters on the data.
- Even though K-means is an unsupervised learning method, we were curious and evaluated its performance by comparing the clustering results with the true labels. We assigned labels according to the majority of the true labels in each cluster.

Results and Discussion

Quantitative Metrics

For unsupervised learning, we can use total distance and Silhouette coefficient to gauge how well clusters are formed. We can plot the total distance function for each point to its cluster for K-means and compute the Silhouette coefficient for all the clustering methods outlined. BetaCV and Normalized Cut can be computed.
For supervised learning, we can compute the confusion matrix showing Type I/II error as well as correct classification. This also helps compute accuracy (correct prediction), precision (true correct out of all predicted correct), recall (true correct out of actual correct), and F1 score (balance of recall and precision).

Project Goals & Expected Results

Our ultimate goal is to create a functional fake news detector. Given a news article in clean text, we hope to achieve > 70% accuracy in detecting fake news. State of art models mentioned in [4] achieve 92% with TF, TF-IDF, SVM combined.

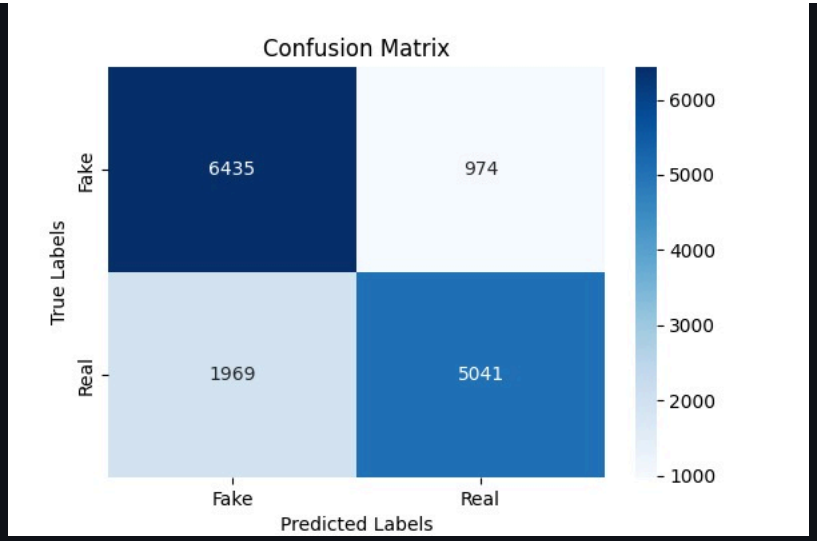
Midterm Model Implementation and Metrics

For our midterm progress check, we have completed our data processing as mentioned above and fitted a **Linear SVC model** using an **50 feature count vectorizer** on our data as training/validation set.

Below is the confusion matrix along with performance measures of the model: **accuracy, precision, recall, F1 score**, and the **confusion matrix** visualized.

Midterm Model Performance

Metric Interpretation



Accuracy: 0.7958943061238644

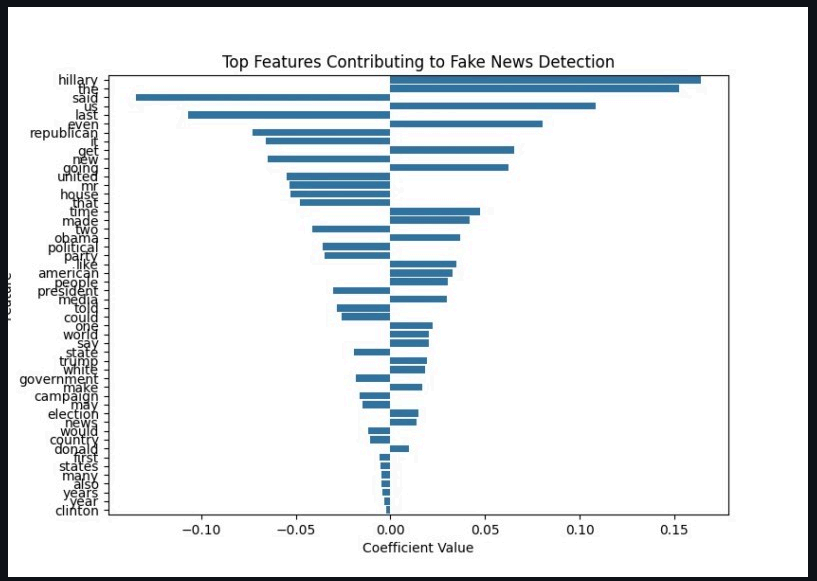
Precision: 0.7657068062827225

Recall: 0.8685382642731813

F1 Score: 0.8138873079112123

Our model is a **binary classifier**. A random guess would be **50%**, and given our relatively simple model and approach, an **accuracy greater than 80%** seems to be performing quite well. Our **lower precision** is dragging our **F1 score** down as well, so if we could focus on **avoiding classifying fake news as real**, we should expect improvements.

Feature Importance



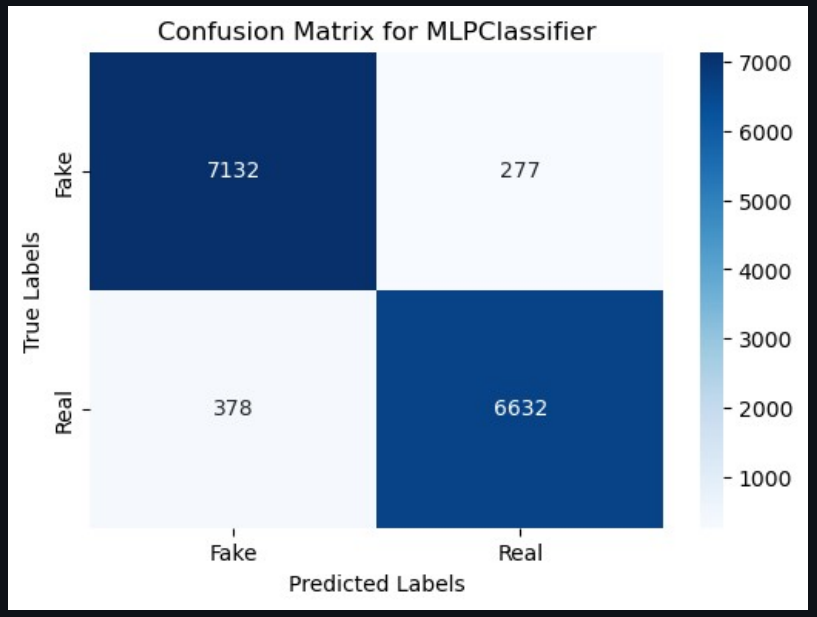
Here we provide a chart for the features which are frequently appearing words and its contribution to the classification.

The coefficients determine the weight in which it contributes to the classification. A large positive coefficient means the word is more likely to be associated with fake news, and a large negative coefficient means the word is more likely to be associated with real news.

In our data exploration, we found that 0 is assigned to be real news and 1 is assigned to be fake news, although the original documentation sometimes conflicts this definition when addressing accuracy etc.

Final Model Performance

Neural Network Performance



Our 2-layer Neural Network achieved:

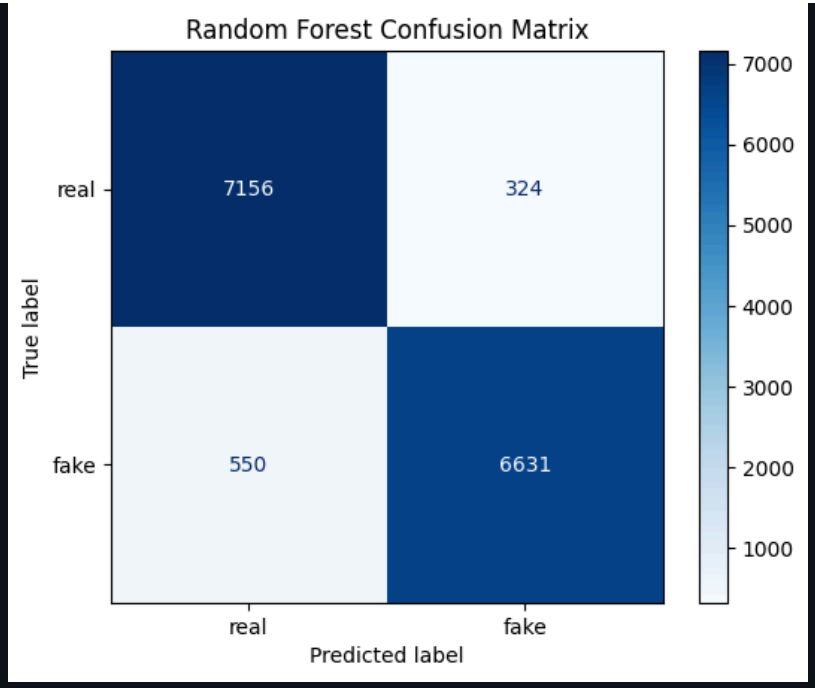
Accuracy: 0.9545738262015396 Precision: 0.9496671105193076 Recall: 0.9626130381967877 F1 Score: 0.9560962531000737

The Neural Network showed the best performance among all models, demonstrating:

- Better feature learning capability with its hidden layers
- Improved handling of complex patterns in text data
- Higher capacity to capture semantic relationships

Given that these are figured obtained from our validation set, we believe that overfitting is not the biggest concern here

Random Forest Model Performance



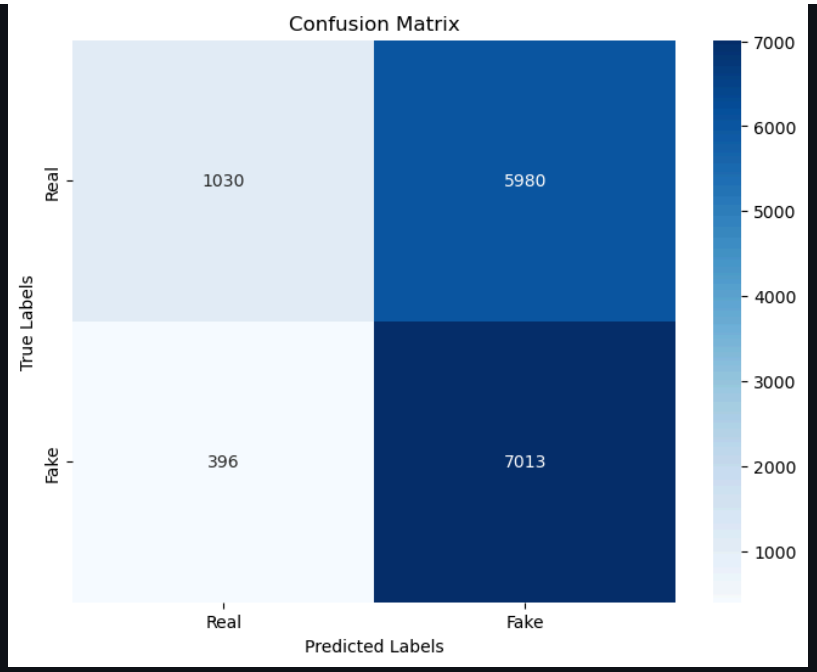
Our Random Forest model achieved the following metrics when using Word2Vec embeddings:

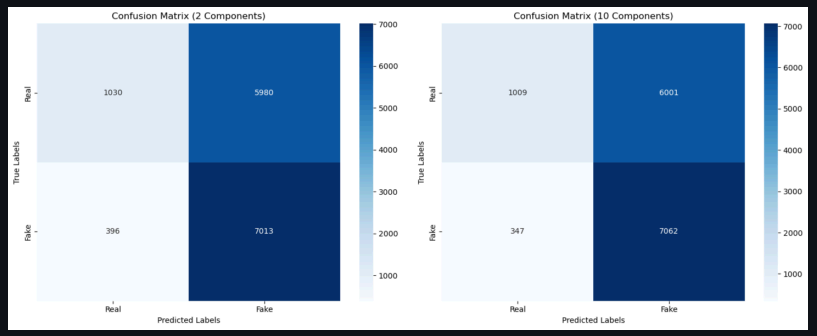
Accuracy: 0.9407953072778119 Precision: 0.9297971918876755 Recall: 0.9561497326203209 F1 Score: 0.9427893488004219

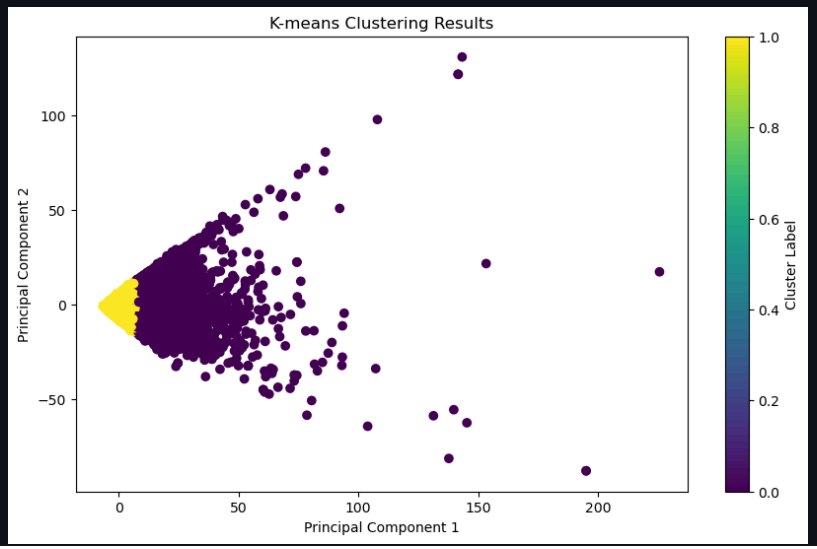
The Random Forest model performed a lot better than our SVC baseline, likely due to:

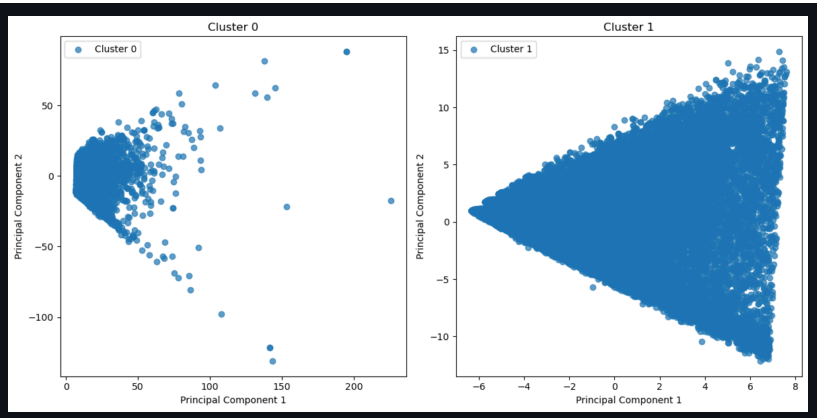
- Using Word2Vec embeddings instead of simple count vectorization
- The ensemble nature of Random Forest helping to reduce overfitting
- Better handling of non-linear relationships in the data

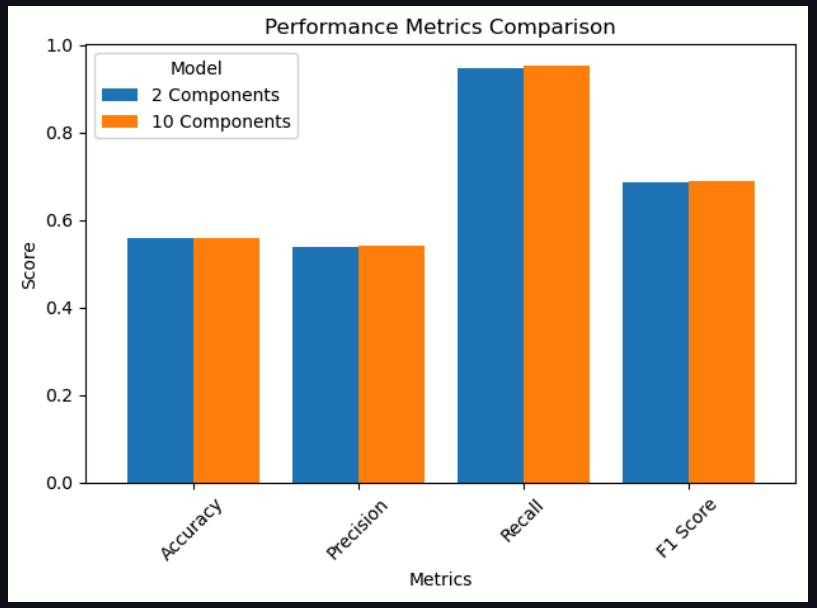
K-means Clustering Results


Confusion Matrix
1030
5980
396
7013
Real
Fake
True Labels
Real
Fake
Predicted Labels
7000
6000
5000
4000
3000
2000
1000


Confusion Matrix (2 Components)
1030
5980
396
7013
Confusion Matrix (10 Components)
1009
6001
347
7062
Real
Fake
True Labels
Predicted Labels


K-means Clustering Results
Principal Component 2
Principal Component 1
Cluster Label
100
50
0
−50
0
50
100
150
200
1.0
0.8
0.6
0.4
0.2
0.0





Our unsupervised approach using K-means achieved:

2-component model performance: Accuracy: 0.558 Precision: 0.540 Recall: 0.947 F1 Score: 0.687

10-component model performance: Accuracy: 0.560 Precision: 0.541 Recall: 0.953 F1 Score: 0.690

The lower performance of K-means is expected as it:

- Doesn't use label information during training
- Relies solely on feature similarity for clustering
- Serves as a baseline for comparison with supervised methods

Interestingly, even though we used PCA to reduce the components to 2 for visualization, the clustering results were not good at all, and using more components or even the full number of features from our top vocabulary size of 50 yields only marginally better results. This suggests that unsupervised clustering might not be a good tool at all for this task.

Model Comparison

Overall Model Performance Comparison

1. **Neural Network (Best Performance)**

- Highest accuracy at 95.5%
- Most balanced precision and recall
- Complex model capable of learning intricate patterns
- Requires more computational resources

2. **Random Forest**
   - Strong performance with 94.1% accuracy
   - Effective with Word2Vec embeddings
   - Good balance of complexity and interpretability
   - More robust than single decision trees, less prone to overfitting
3. **Support Vector Machine**
   - Good baseline performance with 79.6% accuracy
   - Simple and interpretable
   - Lower recall indicates missed fake news
   - Fast training and prediction
4. **K-means Clustering (Lowest Performance)**
   - ~56% accuracy shows limitations of unsupervised approach
   - High recall but poor precision
   - Useful for exploratory analysis
   - Not recommended for production use
   - A possible improvement is to use Word2Vect as input instead of CounteVectorizer

Next Steps

1. **Model Improvements**
   - Experiment with simpler neural network architectures
   - Try emsemble models with other types of classifiers, experiement with Adaboost ec
   - Implement external validation with other text data to evaluate real world performance
2. **Feature Engineering**
   - Explore additional text preprocessing techniques
   - Test different word embedding methods
   - Consider metadata features beyond text content

References

1. Olan, F., Jayawickrama, U., Arakpogun, E.O. et al. Fake news on Social Media: the Impact on Society. Inf Syst Front 26, 443-458 (2024). Link
2. Shu, Kai, et al. "Fake news detection on social media: A data mining perspective." ACM SIGKDD explorations newsletter 19.1 (2017): 22-36. Link
3. Ahmed, Hadeer, Issa Traore, and Sherif Saad. "Detection of online fake news using n-gram analysis and machine learning techniques." Springer International Publishing, 2017. Link
4. Alghamdi, Jawaher, Suhuai Luo, and Yuqing Lin. "A comprehensive survey on machine learning approaches for fake news detection." Multimedia Tools and Applications 83.17 (2024): 51009-51067. Link

---

Contribution Table:

Name	Final Contribution
Ian	Model Selection, Expected Results, Discussion, Progress Write Up
Hui	Problem Statement, Motivation, Data Processing
Feng	Methods, Code Implementation, Report Writing
Dongzhao	Introduction, Streamlit, Coding

Name	Final Contribution
Sarah	Dropped the Class
All	Literature Review, Team Meetings, Code Review, Slides

Gantt Chart: