# Project Proposal

Pawan Medidi, Anna Zhao, Simon Luong, Jacob Zeigler, Suhel Keswani

## Introduction/Background

Over the last decade as social media has become increasingly developed, so has the world of making bots within them. They have become increasingly complex, varied, and abundant, and their volume and influence has sparked international conversation. So, our team thought it would be interesting to classify whether a comment on a social media platform is a human or a bot.

## Dataset Description

TwiBot-20, developed at Washington University, includes verified human users and various bots, comprising approximately 230K accounts. This dataset encompasses complete post histories and following relationships. For each user, it contains profile information, the 200 most recent tweets, 20 random neighbors/followers, and a bot status indicator. Currently, traditional bot detection methods have proven inconclusive in this dataset.

Dataset link: TwiBot-20 (We have the full data)

## Literature Review

As machine learning advances, numerous studies have emerged on bot detection, each with unique methodologies. Prior reviews highlight that models based on random forests, support vector machines, and convolutional neural networks excel in detecting Twitter bots. Feed Forward Neural Networks, support vector machines, gradient boosting, and multinomial naive Bayes are effective against spam bots. [2] Ongoing research aims to enhance deep learning methods by integrating the best features of existing models to improve text feature utilization and anomaly detection. [3] One study using similar methods to our projected ones looks at how detecting twitter bots can be done more effectively with reduced features finding that only 5 features (number of tweets, followers, mutuals, likes, and lists included) was effective for detecting bots compared to a full feature larger set.[1]

## Problem Definition

**Problem:** Social media platforms are increasingly filled with bots generating vast amounts of content. These bots range from simple spambots to sophisticated conversational bots that engage in harmful activities. Their ability to influence public opinion, spread misinformation, and undermine genuine user interactions poses significant risks.

**Motivation:** Identifying and categorizing whether a comment is a bot can mitigate the spread of false information, preserve the integrity of social media conversations, and protect public opinion and democratic processes from manipulation.

**Solution:** We propose a Bot Comment Classification Analysis system using machine learning and natural language processing to identify and classify whether a comment is a human or bot based on their commenting styles. This system will analyze linguistic patterns, posting behaviors, and engagement metrics to distinguish humans from bots, and provide real-time detection to mitigate bot activity. This approach aims to enhance the understanding of bot behavior and contribute to a more trustworthy social media environment.

## Methods

We will utilize Scikit-Learn and TensorFlow libraries. Our preprocessing methods include lemmatization, tokenization, removing non-alphanumeric characters, and Word2Vec for feature extraction. For supervised learning, we will employ XGBoost, Random Forest, Decision Trees, and K-Nearest Neighbors.

We aim to use the following metrics:

**Accuracy:** (Correct Classifications) / (Total Classifications)

**Confusion Matrix:** contains true positives, false positives, true negatives, and false negatives for each class.

**Precision:** (True positives) / (Total predicted positives)

**Recall:** (True positives) / (All actual positives)

**F1 Score:** 2 * (Precision * Recall) / (Precision + Recall)

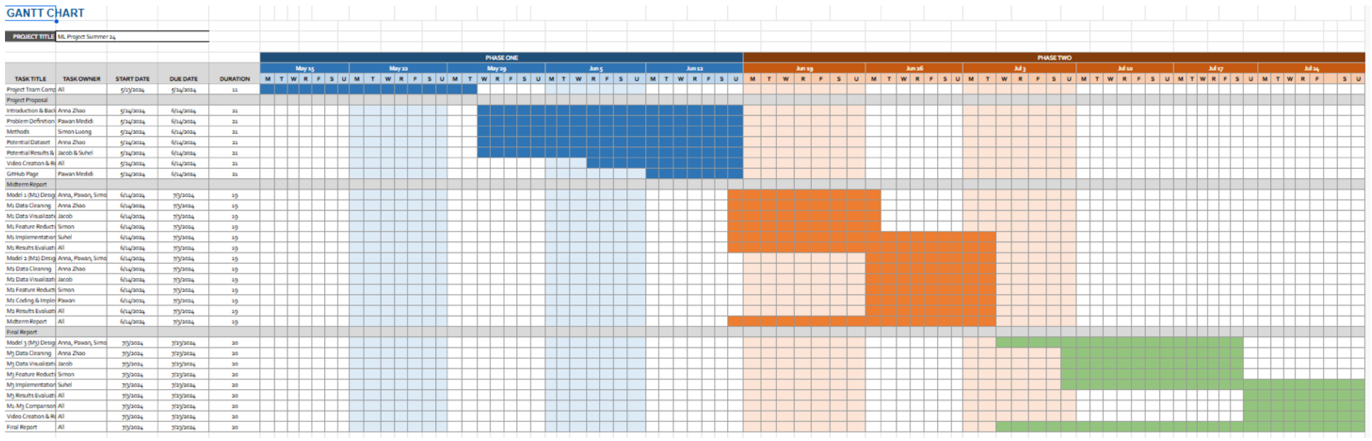
We aim to achieve 75-85% accuracy, precision, and recall, balancing these metrics to optimize model performance.

## Potential Results

We plan to start predicting whether a comment is human or bot based on historical data. Once we can do this reliably, we will gather more current data from this year, similar to our reference dataset, and predict the results for this year's comments. This will enable us to validate our model's accuracy with recent data and ensure its relevance in detecting contemporary bot behaviors.

## Gantt Chart

List each member’s planned responsibilities for the entirety of the project.



GANTT CHART

PROJECT TITLE ML Project Summer 24

| TASK TITLE | TASK OWNER | START DATE | DUE DATE | DURATION |
|---|---|---|---|---|
| Project Team Comp | All | 5/13/2024 | 5/24/2024 | 11 |
| Project Proposal | | | | |
| Introduction & Back | Anna Zhao | 5/24/2024 | 6/14/2024 | 21 |
| Problem Definition | Pawan Medidi | 5/24/2024 | 6/14/2024 | 21 |
| Methods | Simon Luong | 5/24/2024 | 6/14/2024 | 21 |
| Potential Dataset | Anna Zhao | 5/24/2024 | 6/14/2024 | 21 |
| Potential Results & | Jacob & Suhel | 5/24/2024 | 6/14/2024 | 21 |
| Video Creation & Re | All | 5/24/2024 | 6/14/2024 | 21 |
| GitHub Page | Pawan Medidi | 5/24/2024 | 6/14/2024 | 21 |
| Midterm Report | | | | |
| Model 1 (M1) Desig | Anna, Pawan, Simo | 6/14/2024 | 7/3/2024 | 19 |
| M1 Data Cleaning | Anna Zhao | 6/14/2024 | 7/3/2024 | 19 |
| M1 Data Visualizati | Jacob | 6/14/2024 | 7/3/2024 | 19 |
| M1 Feature Reducti | Simon | 6/14/2024 | 7/3/2024 | 19 |
| M1 Implementation | Suhel | 6/14/2024 | 7/3/2024 | 19 |
| M1 Results Evaluati | All | 6/14/2024 | 7/3/2024 | 19 |
| Model 2 (M2) Desig | Anna, Pawan, Simo | 6/14/2024 | 7/3/2024 | 19 |
| M2 Data Cleaning | Anna Zhao | 6/14/2024 | 7/3/2024 | 19 |
| M2 Data Visualizati | Jacob | 6/14/2024 | 7/3/2024 | 19 |
| M2 Feature Reducti | Simon | 6/14/2024 | 7/3/2024 | 19 |
| M2 Coding & Imple | Pawan | 6/14/2024 | 7/3/2024 | 19 |
| M2 Results Evaluati | All | 6/14/2024 | 7/3/2024 | 19 |
| Midterm Report | All | 6/14/2024 | 7/3/2024 | 19 |
| Final Report | | | | |
| Model 3 (M3) Desig | Anna, Pawan, Simo | 7/3/2024 | 7/23/2024 | 20 |
| M3 Data Cleaning | Anna Zhao | 7/3/2024 | 7/23/2024 | 20 |
| M3 Data Visualizati | Jacob | 7/3/2024 | 7/23/2024 | 20 |
| M3 Feature Reducti | Simon | 7/3/2024 | 7/23/2024 | 20 |
| M3 Implementation | Suhel | 7/3/2024 | 7/23/2024 | 20 |
| M3 Results Evaluati | All | 7/3/2024 | 7/23/2024 | 20 |
| M1-M3 Comparison | All | 7/3/2024 | 7/23/2024 | 20 |
| Video Creation & Re | All | 7/3/2024 | 7/23/2024 | 20 |
| Final Report | All | 7/3/2024 | 7/23/2024 | 20 |

## Contribution Table

| Name | Proposal Contributions |
|---|---|
| Anna | Introduction/Background, Literature Review, Dataset Description |
| Pawan | Problem Definition, Methods |
| Simon | Methods |
| Suhel | Potential Results and Discussion |
| Jacob | Potential Results and Discussion, Gantt Chart, GitHub Repository |

## References

[1] J. V. Fonseca Abreu, C. Ghedini Ralha and J. J. Costa Gondim, "Twitter Bot Detection with Reduced Feature Set," 2020 IEEE International Conference on Intelligence and Security Informatics (ISI), Arlington, VA, USA, 2020, pp. 1-6, doi: 10.1109/ISI49825.2020.9280525.

[2] M. Aljabri, R. Zagrouba, A. Shaahid, F. Alnasser, A. Saleh, and D. M. Alomari, “Machine learning-based social media bot detection: a comprehensive literature review,” Social Network Analysis and Mining, vol. 13, no. 1, Jan. 2023, doi: https://doi.org/10.1007/s13278-022-01020-5.

[3] K. Hayawi, S. Saha, M. M. Masud, S. S. Mathew, and M. Kaosar, “Social media bot detection with deep learning methods: a systematic review,” Neural Computing and Applications, vol. 35, Mar. 2023, doi: https://doi.org/10.1007/s00521-023-08352-z.

# Midterm Checkpoint

## Results and Discussion

### MobileBERT Embeddings and Data Processing

We processed the text data using MobileBERT, extracting embeddings and normalizing them. The final DataFrame (`final_df`) contains 17161 rows and 515 columns, with each row representing a tweet and its associated embeddings. We use 511 different features extracted from the text content of each tweet. An important thing to note is that right now we truncate each tweet at 127 characters; using all of the text could improve results.

### XGBoost Classifier

**Error Encountered:** We faced an issue loading the XGBoost library due to the missing OpenMP runtime on the system.

**Next Steps:** Resolve the issue by installing the OpenMP library and reattempt running the XGBoost model. This step is crucial for leveraging the power of gradient boosting for classification.

### KNN Classifier

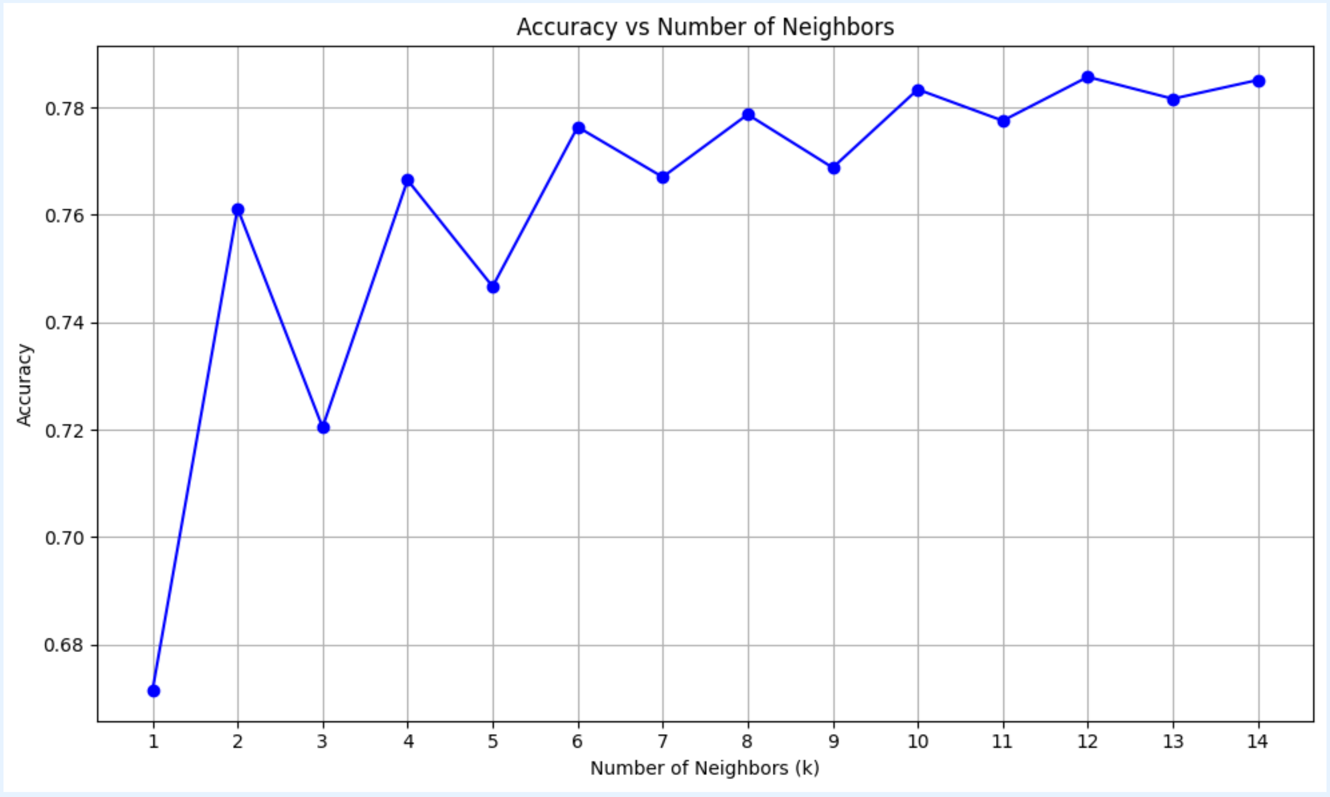
**Results:**

**Accuracy:** 0.785

**Precision:** 0.100

**Recall:** 0.003

**F1 Score:** 0.005



**Visualization:** The accuracy vs. number of neighbors plot indicates how model performance changes with different values of k. The best accuracy was achieved with 14 neighbors.

## Discussion

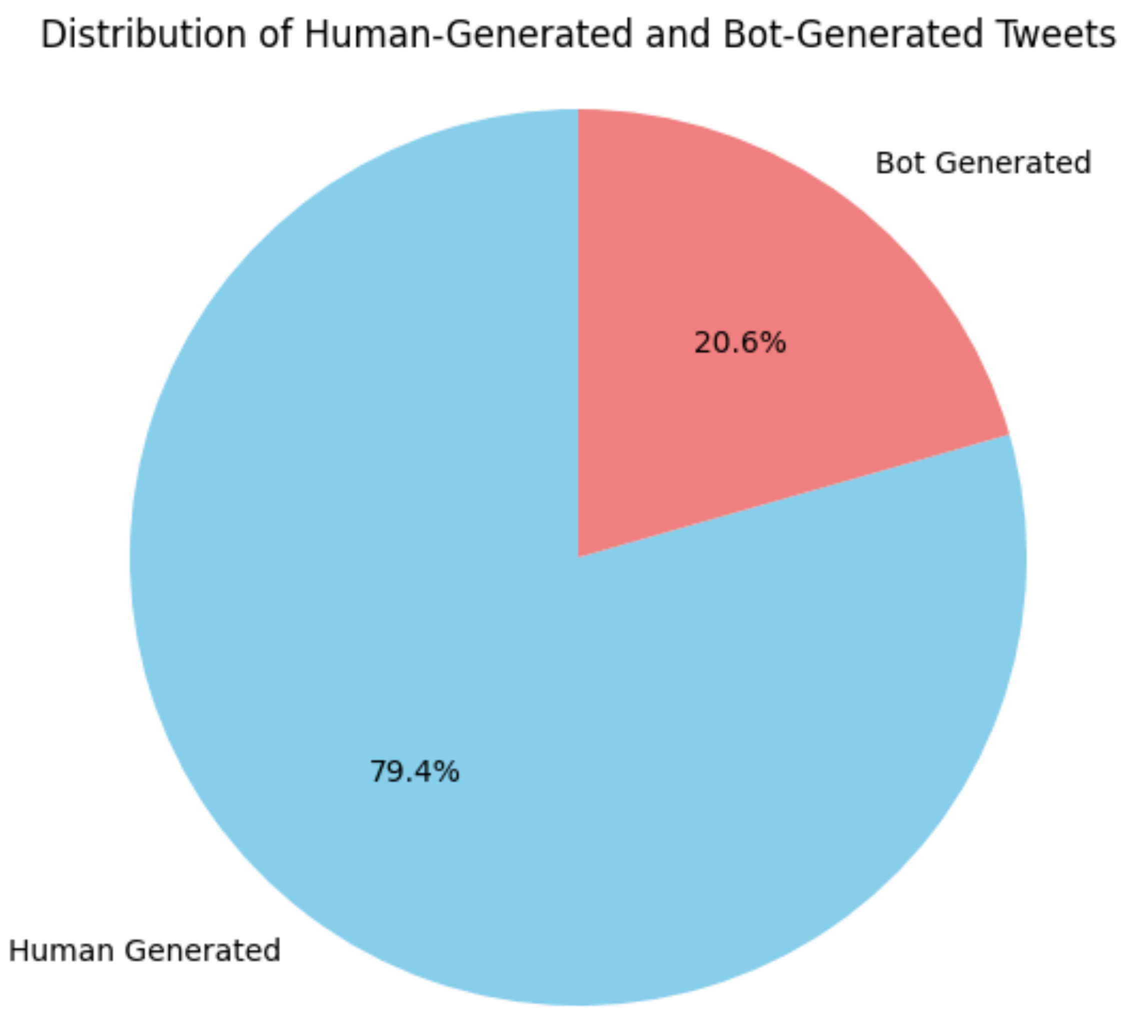
**Justification of Results:**

High accuracy and low precision; low F1 score can be attributed to class imbalance, with a high majority of human tweets. The KNN model cannot capture the complexity of text data with high sentiment and is likely overfitting the data.

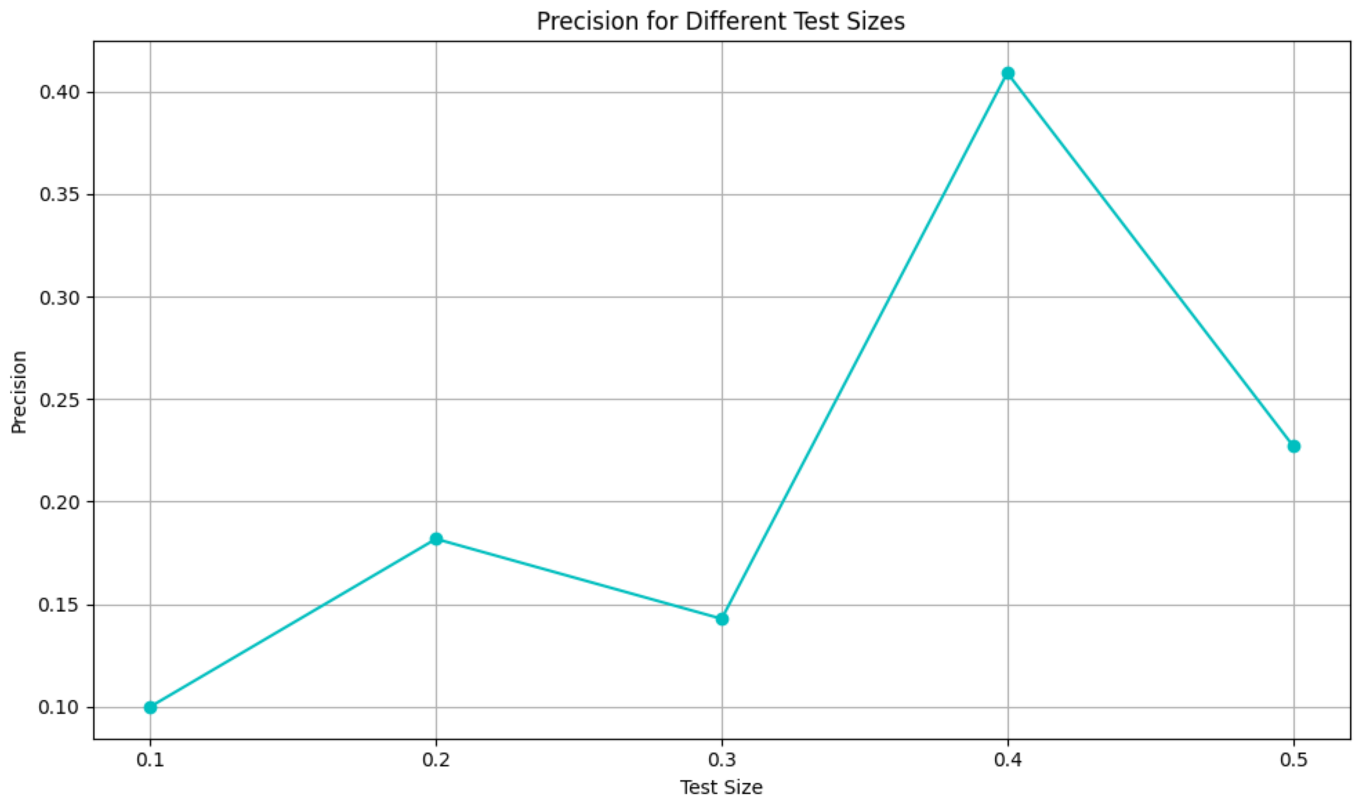
**Performance Analysis:** The KNN model achieved an accuracy of 78.5%, which is fairly good. However, the precision, recall, and F1 scores are very low. This suggests that while the model correctly classifies a significant number of tweets overall, it struggles to correctly identify and distinguish between the two classes (human vs. bot).

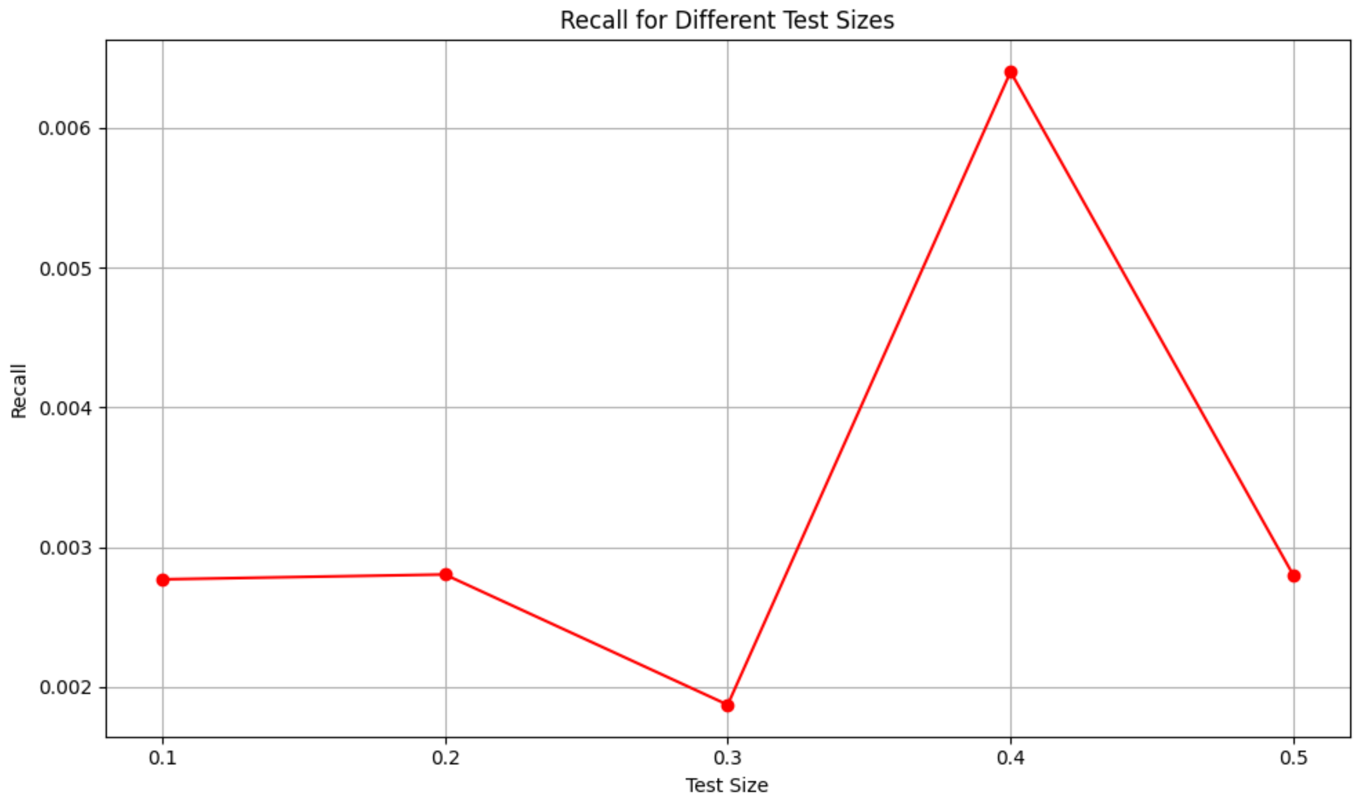
**Reasoning:** The low precision and recall indicate that the model is not effectively capturing the nuances in the data that differentiate bots from humans. KNN might not be the best fit for this task due to the high dimensionality of the embeddings and the nature of the data.

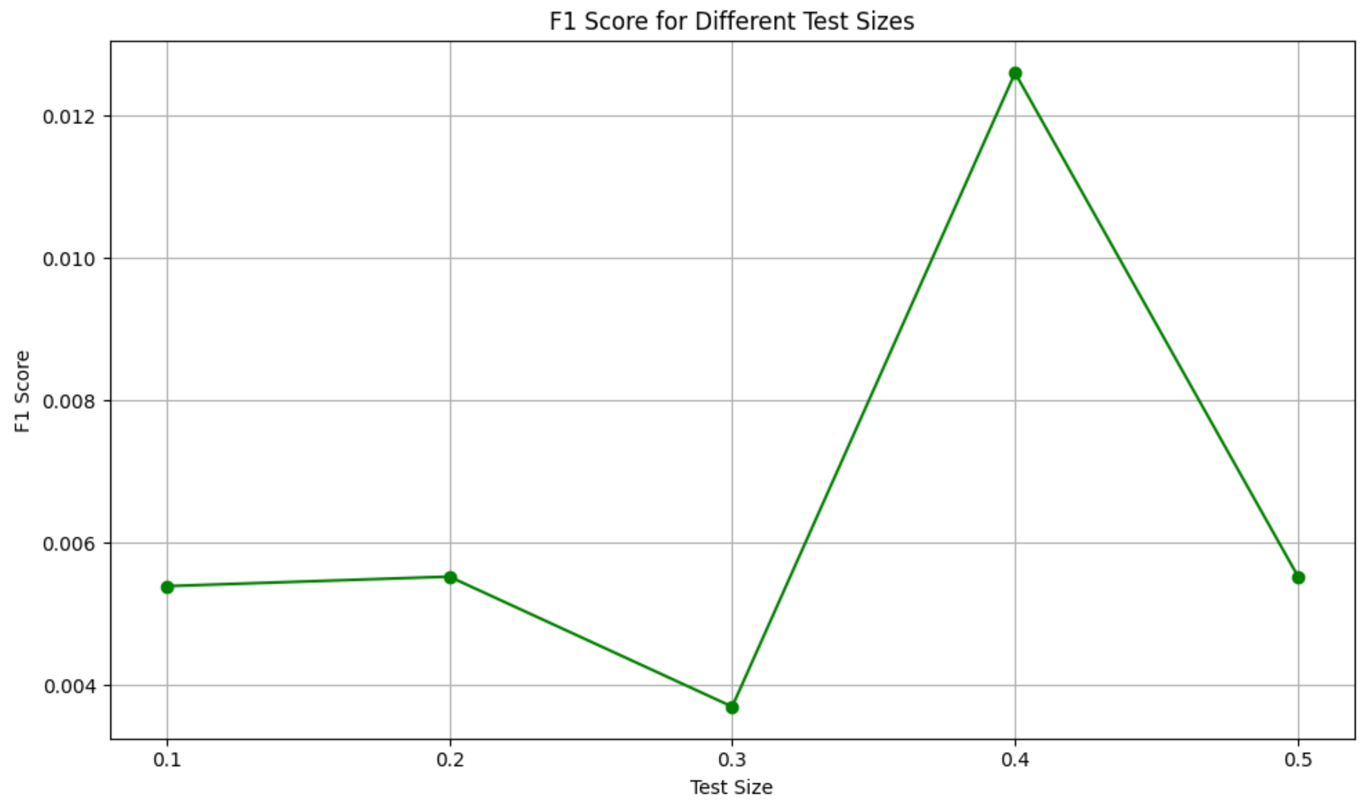
## Visuals

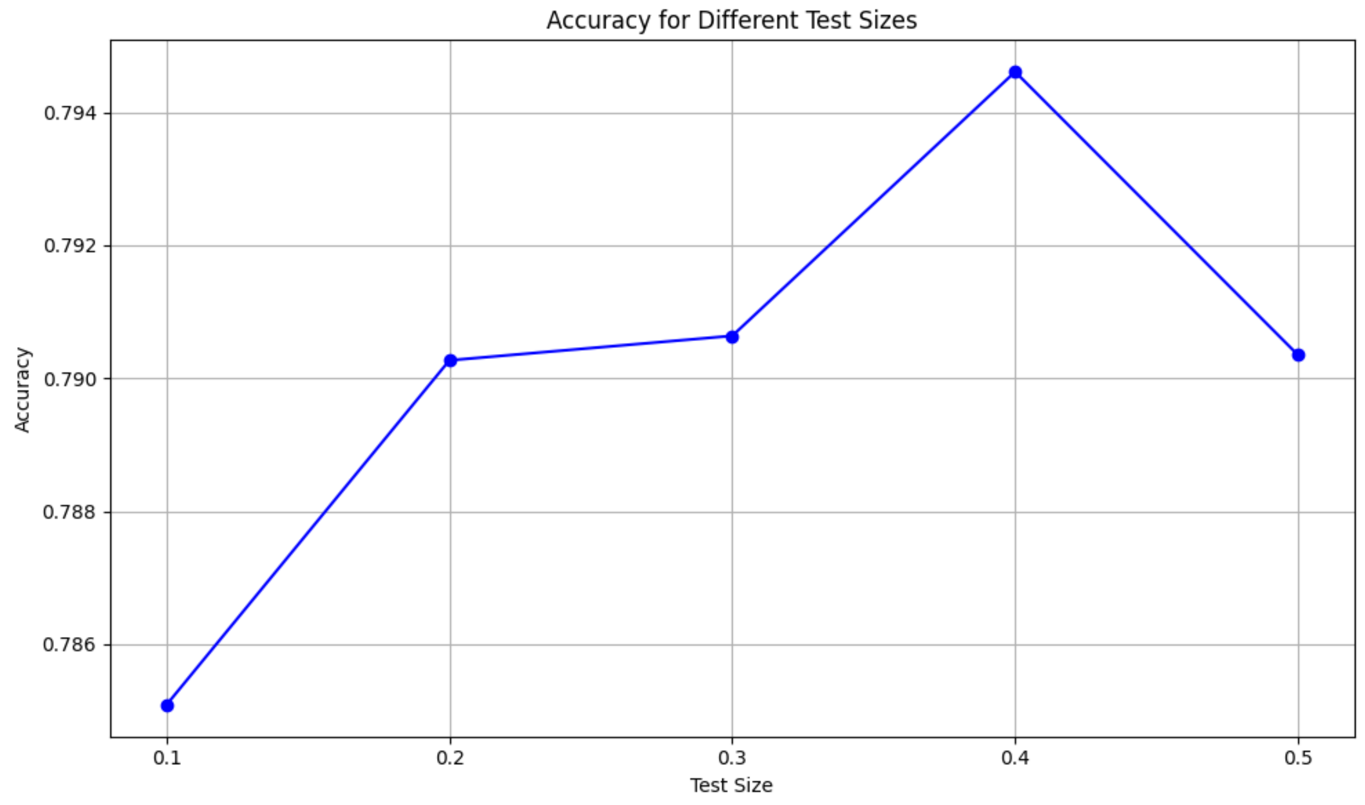


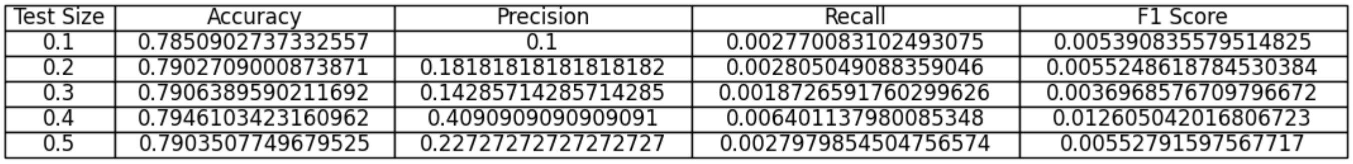
13630 human data points and 3531 bot data points.


Precision for Different Test Sizes
Precision
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.1
0.2
0.3
0.4
0.5
Test Size


Recall for Different Test Sizes
Recall
0.006
0.005
0.004
0.003
0.002
0.1
0.2
0.3
0.4
0.5
Test Size


F1 Score for Different Test Sizes
F1 Score
0.012
0.010
0.008
0.006
0.004
0.1
0.2
0.3
0.4
0.5
Test Size


Accuracy for Different Test Sizes
Accuracy
0.794
0.792
0.790
0.788
0.786
0.1
0.2
0.3
0.4
0.5
Test Size



| Test Size | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| 0.1 | 0.7850902737332557 | 0.1 | 0.002770083102493075 | 0.005390835579514825 |
| 0.2 | 0.7902709000873871 | 0.18181818181818182 | 0.002805049088359046 | 0.0055248618784530384 |
| 0.3 | 0.7906389590211692 | 0.14285714285714285 | 0.0018726591760299626 | 0.0036968576709796672 |
| 0.4 | 0.7946103423160962 | 0.4090909090909091 | 0.006401137980085348 | 0.012605042016806723 |
| 0.5 | 0.7903507749679525 | 0.22727272727272727 | 0.0027979854504756574 | 0.00552791597567717 |

So in reality, we have accuracies where our lowest is 78.5% and our highest is 79.46%.

## Next Steps

Model Optimization: Continue tuning hyperparameters for KNN to see if performance can be improved.

Alternative Models: Explore other models such as SVM, Random Forest, or neural networks which might handle the complexity of the data better.

Feature Engineering: Perform further feature engineering to better capture the distinguishing features between human and bot tweets.

## Conclusion

**KNN Classifier:** While achieving a decent accuracy, the KNN model needs improvement in precision and recall.

**XGBoost Classifier:** Resolving the dependency issue is crucial to evaluating its performance.

**Neural Network:** Further evaluation and fine-tuning are necessary to validate the model's effectiveness for energy prediction.

## Overall Next Steps

- Resolve XGBoost Issues: Install the necessary libraries and re-evaluate the model.
- Optimize KNN and Explore Alternatives: Continue hyperparameter tuning and explore other classification models.
- Fine-tune Neural Network: Ensure all evaluation metrics are captured and improve model performance.

These steps will help in achieving a more accurate and reliable classification of tweets and better energy consumption predictions.

## Why We Use These Models

### XGBoost

- **Handles Imbalanced Data:** XGBoost can manage class imbalance effectively through weight adjustments and custom loss functions, ensuring better performance in identifying minority classes.
- **Captures Complex Relationships:** With its boosting technique, XGBoost excels at capturing complex, non-linear relationships in high-dimensional data, making it suitable for nuanced differences in text embeddings.

### KNN (K-Nearest Neighbors)

- **Simple and Intuitive:** KNN is straightforward to implement and understand, making it a great starting point for initial model development and benchmarking.
- **Effective Local Pattern Recognition:** By classifying based on the closest neighbors, KNN can effectively identify local patterns and clusters within the data, which can be useful for distinguishing similar comments.

## Midterm Contributions

| Name | Midterm Contributions |
|---|---|
| Pawan Medidi | Coded the GitHub Pages, created the ReadMe and added all info, and met with the TA to finalize and discuss changes neeeded. |
| Anna Zhao | Initial proposal information, data preprocessing, modeling. |
| Simon Luong | Cleaning the dataset, analyzing it, getting features, setting up the KNN, and running that model on our data. |
| Suhel Keswani | Making interpreting our results, and making/getting all the visuals/statistics for the data we used and the results we got from that. |
| Jacob Zeigler | Helped with proposal. |

# Final Report

Pawan Medidi, Anna Zhao, Simon Luong, Jacob Zeigler, Suhel Keswani

## Introduction/Background

Over the last decade as social media has become increasingly developed, so has the world of making bots within them. They have become increasingly complex, varied, and abundant, and their volume and influence have sparked a lot of conversation. Our team aims to classify whether a comment on a social media platform is a human or a bot.

### Literature Review

As machine learning advances, numerous studies have emerged on bot detection, each with unique methodologies. Prior reviews highlight that models based on random forests, support vector machines, and convolutional neural networks excel in detecting Twitter bots. Feed Forward Neural Networks, support vector machines, gradient boosting, and multinomial naive Bayes are effective against spam bots. Ongoing research aims to enhance deep learning methods by integrating the best features of existing models to improve text feature utilization and anomaly detection. One study using similar methods to our projected ones looks at how detecting twitter bots can be done more effectively with reduced features finding that only 5 features (number of tweets, followers, mutuals, likes, and lists included) was effective for detecting bots compared to a full feature larger set.

### Dataset Description

We utilized the TwiBot-20 dataset, developed at Washington University, which includes verified human users and various bots, comprising approximately 230K accounts. This dataset encompasses complete post histories and following relationships. For each user, it contains profile information, the 200 most recent tweets, 20 random neighbors/followers, and a bot status indicator. Currently, traditional bot detection methods have proven inconclusive in this dataset.

Dataset link: TwiBot-20 (We have the full data)

## Problem Definition

**Problem:** Social media platforms are increasingly filled with bots generating vast amounts of content. These bots range from simple spambots to sophisticated conversational bots that engage in harmful activities. Their ability to influence public opinion, spread misinformation, and undermine genuine user interactions poses significant risks.

**Motivation:** Identifying and categorizing whether a comment is a bot can mitigate the spread of false information, preserve the integrity of social media conversations, and protect public opinion and democratic processes from manipulation.

**Solution:** We propose a Bot Comment Classification Analysis system using machine learning and natural language processing to identify and classify whether a comment is a human or bot based on their commenting styles. This system will analyze linguistic patterns, posting behaviors, and engagement metrics to distinguish humans from bots, and provide real-time detection to mitigate bot activity. This approach aims to enhance the understanding of bot behavior and contribute to a more trustworthy social media environment.

## Methods

We utilized the TwiBot-20 dataset, developed at Washington University, which includes verified human users and various bots, comprising approximately 230K accounts. For this project, we focused on the following machine learning algorithms: K-Nearest Neighbors (KNN), Random Forest, and a Neural Network model.

Our preprocessing methods included:

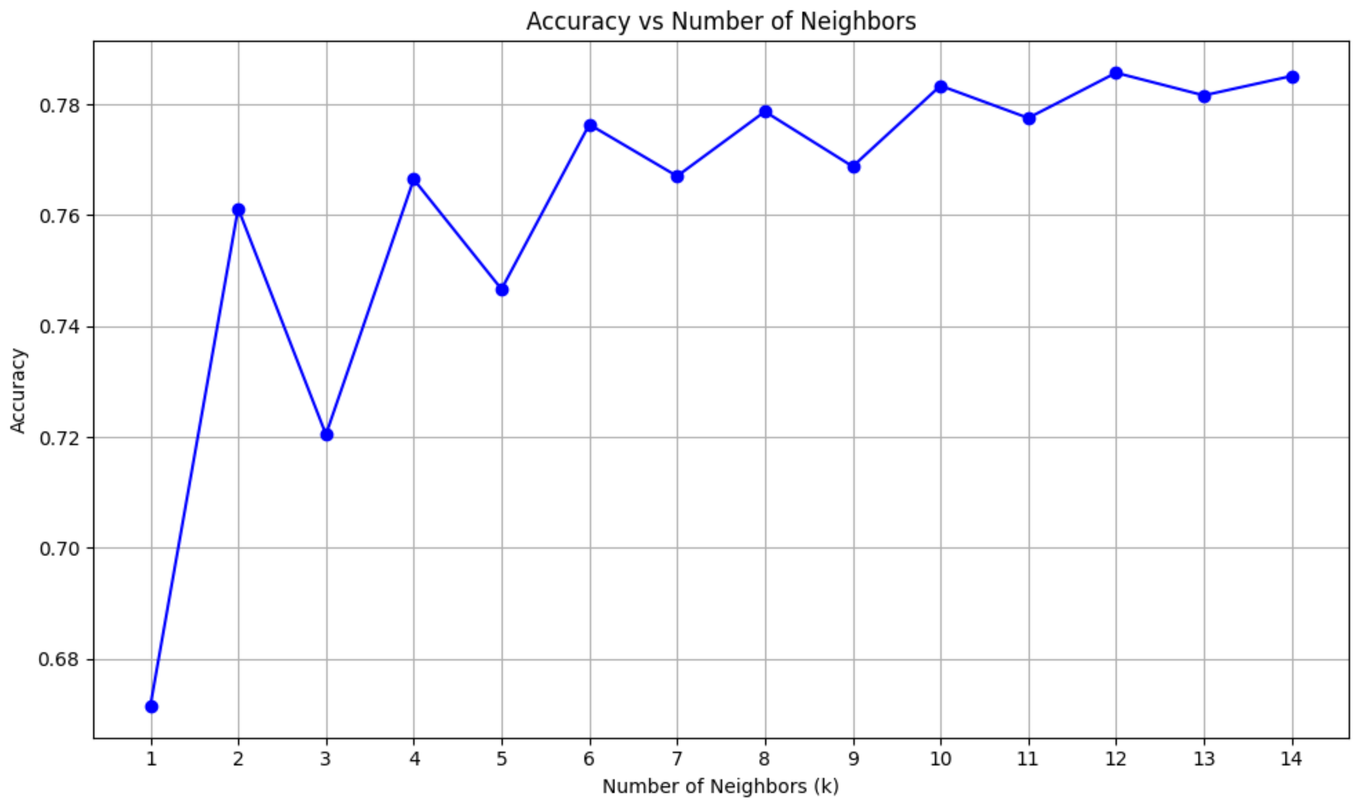
- Tokenization
- Lemmatization
- Removing non-alphanumeric characters
- Truncating text to a fixed length
- Using MobileBERT for feature extraction

We utilized Scikit-Learn and TensorFlow libraries for model implementation. Our evaluation metrics included accuracy, precision, recall, and F1 score.

## Results and Discussion

### K-Nearest Neighbors (KNN) Classifier

- **Accuracy:** 0.785
- **Precision:** 0.100
- **Recall:** 0.003
- **F1 Score:** 0.005



The KNN model achieved an accuracy of 78.5%, which is fairly good. However, the precision, recall, and F1 scores are very low. This suggests that while the model correctly classifies a significant number of tweets overall, it struggles to correctly identify and distinguish between the two classes (human vs. bot).

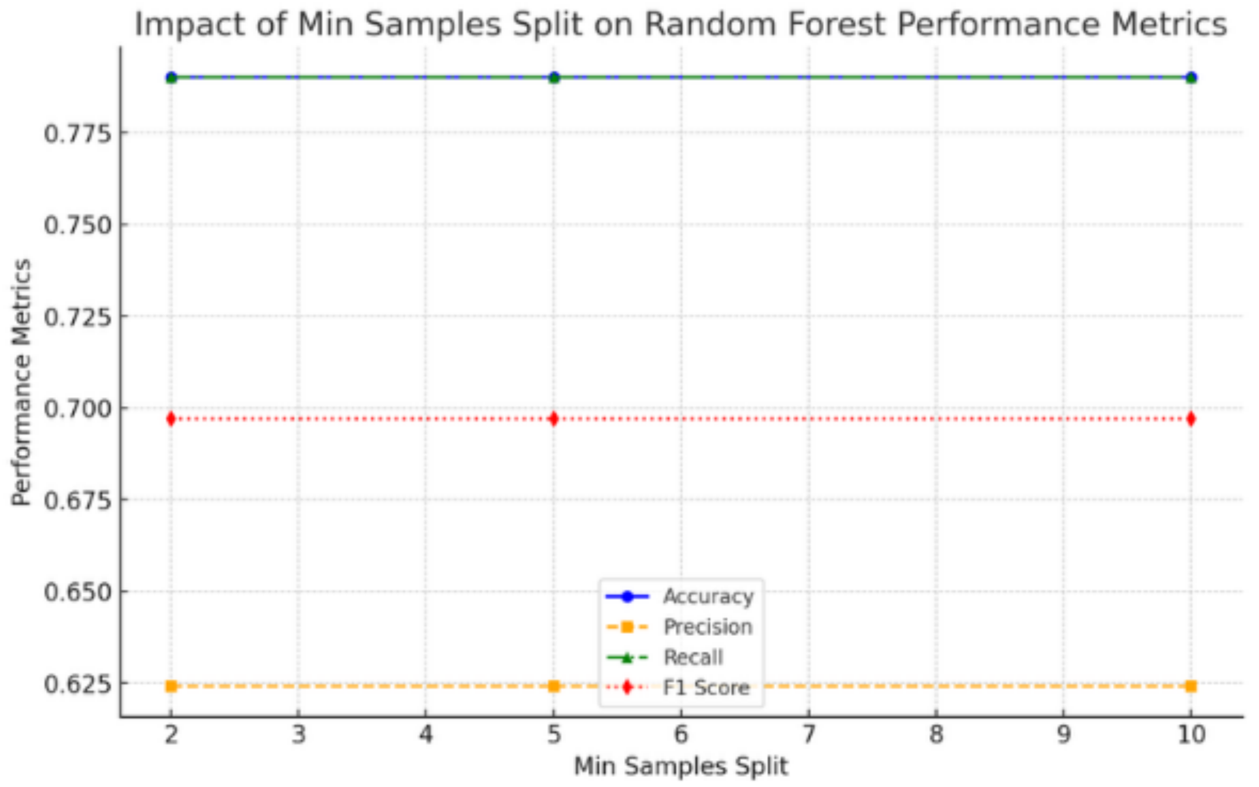
### Random Forest Classifier

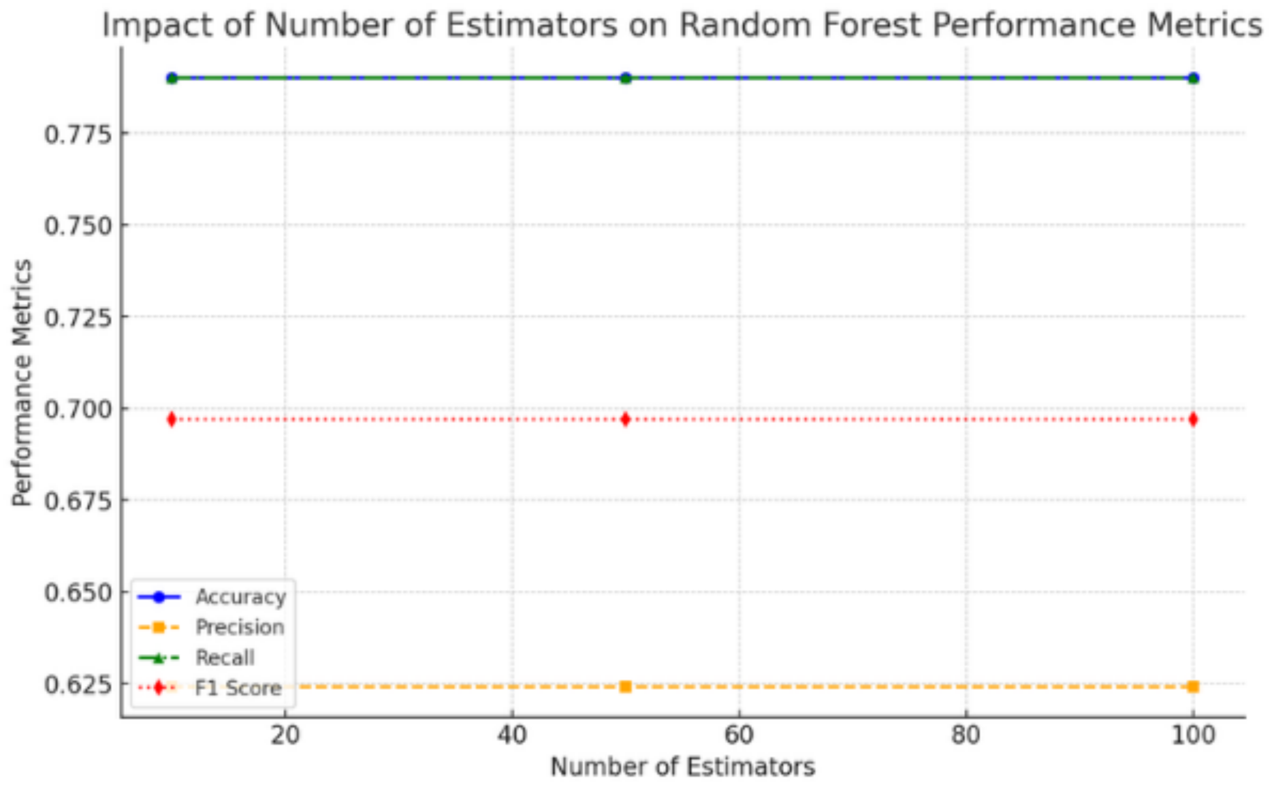
**Accuracy:** 0.790

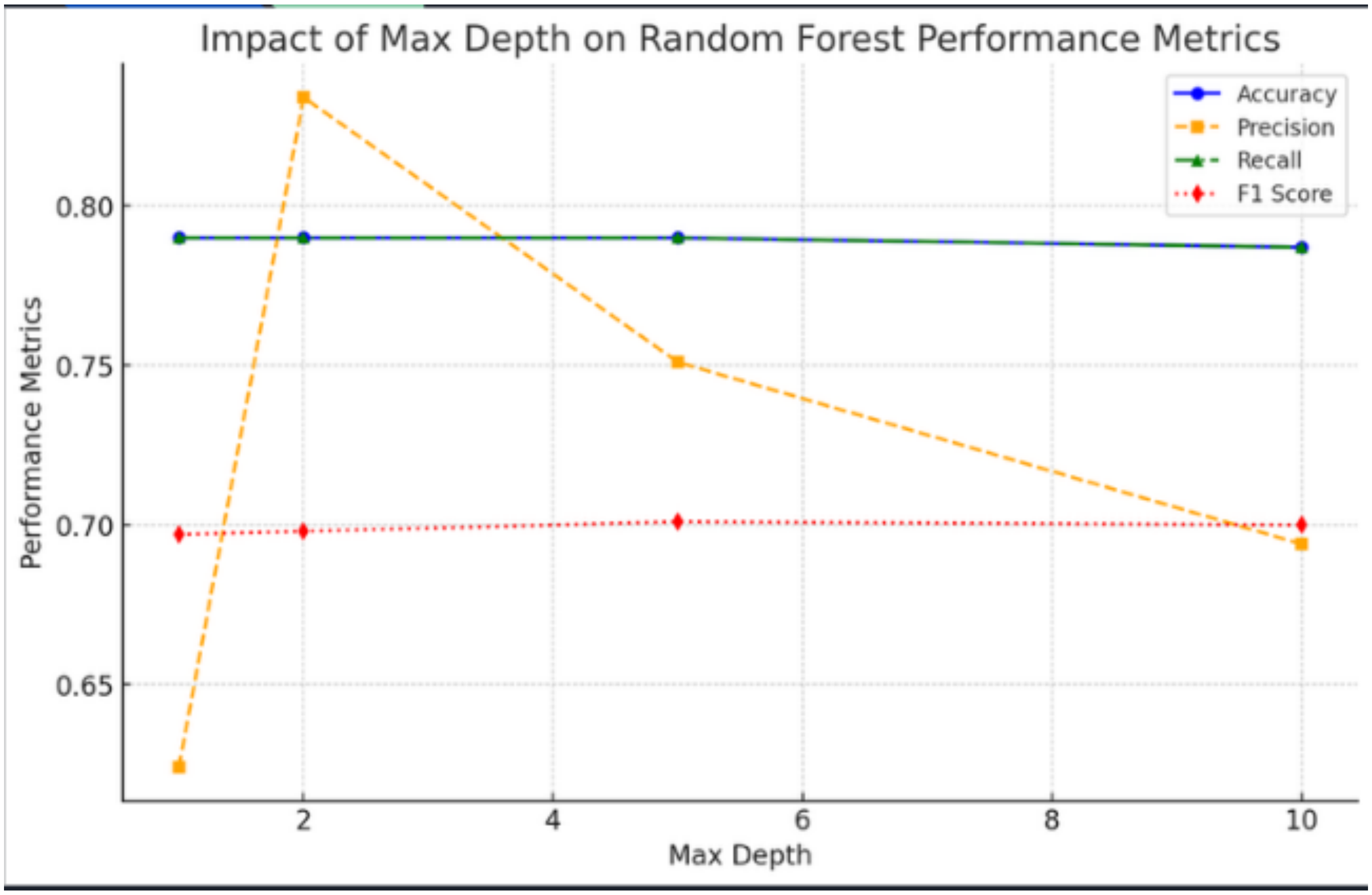
**Precision:** 0.624

**Recall:** 0.790

**F1 Score:** 0.697


Impact of Min Samples Split on Random Forest Performance Metrics
Performance Metrics
0.775
0.750
0.725
0.700
0.675
0.650
0.625
Accuracy
Precision
Recall
F1 Score
2
3
4
5
6
7
8
9
10
Min Samples Split


Impact of Number of Estimators on Random Forest Performance Metrics
Performance Metrics
0.775
0.750
0.725
0.700
0.675
0.650
0.625
Accuracy
Precision
Recall
F1 Score
20
40
60
80
100
Number of Estimators



The Random Forest model showed improved performance with an accuracy of 79.0%, precision of 62.4%, recall of 79.0%, and an F1 score of 69.7%. This indicates that the model is better at distinguishing between humans and bots compared to KNN.

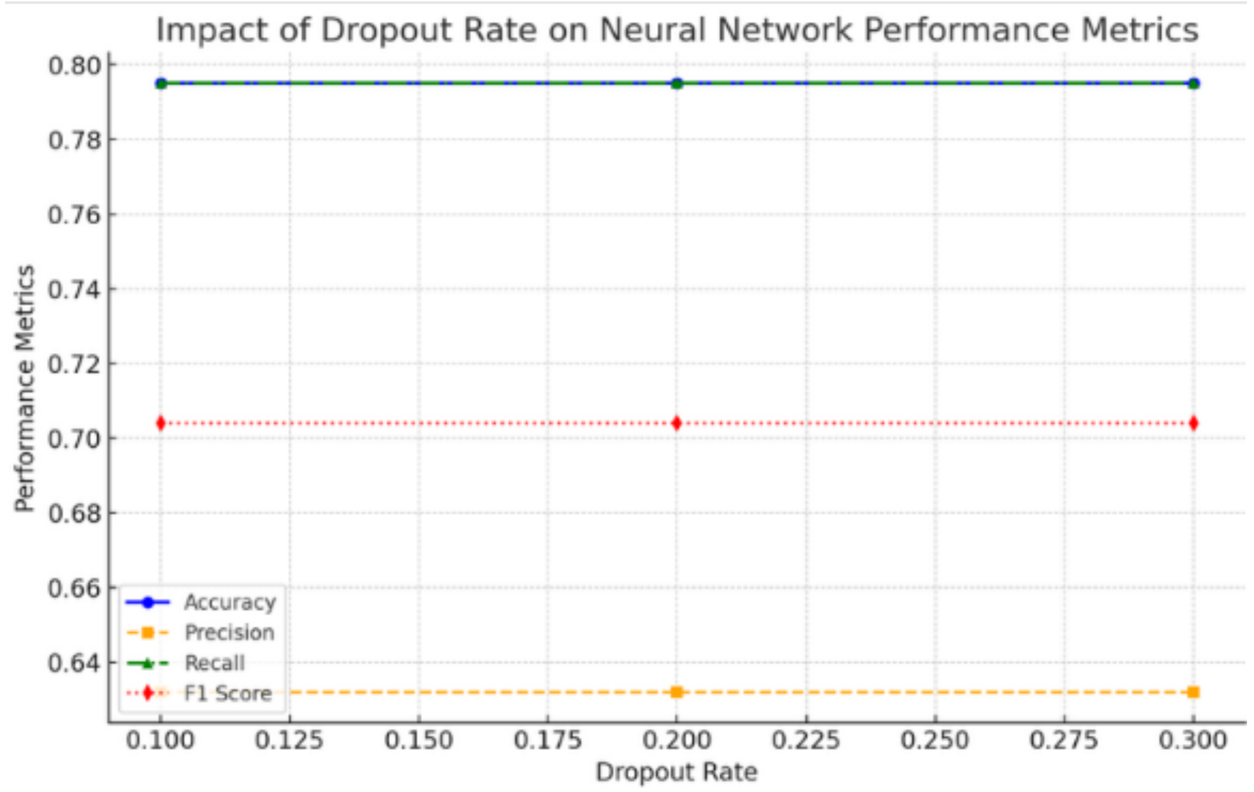
### Neural Network Model

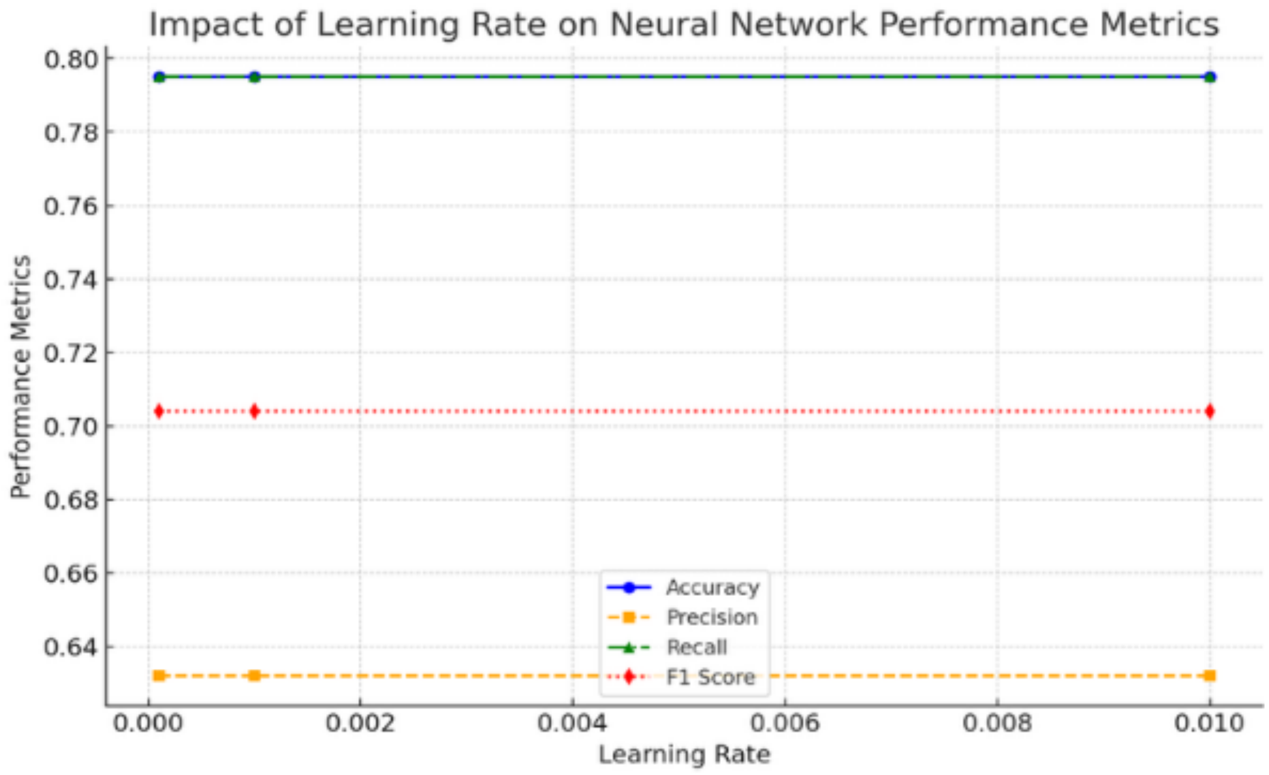
**Accuracy:** 0.795

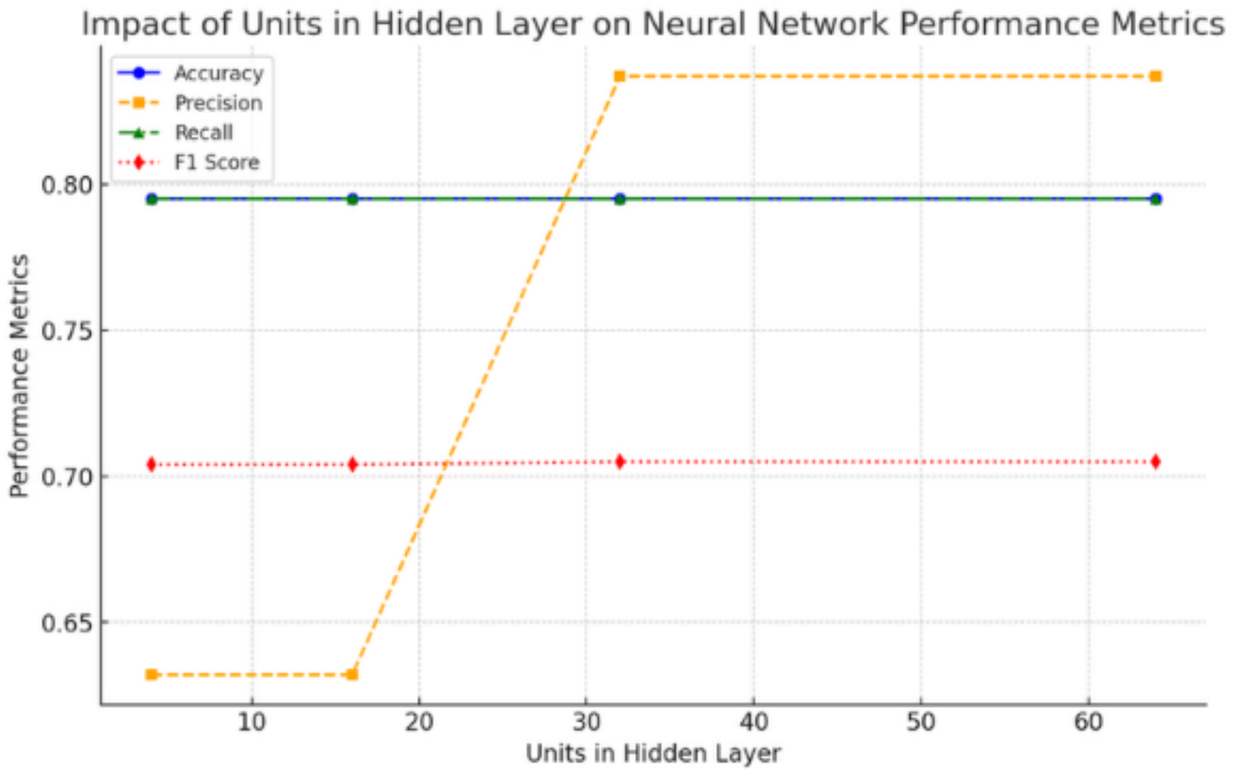
**Precision:** 0.632

**Recall:** 0.795

**F1 Score:** 0.704

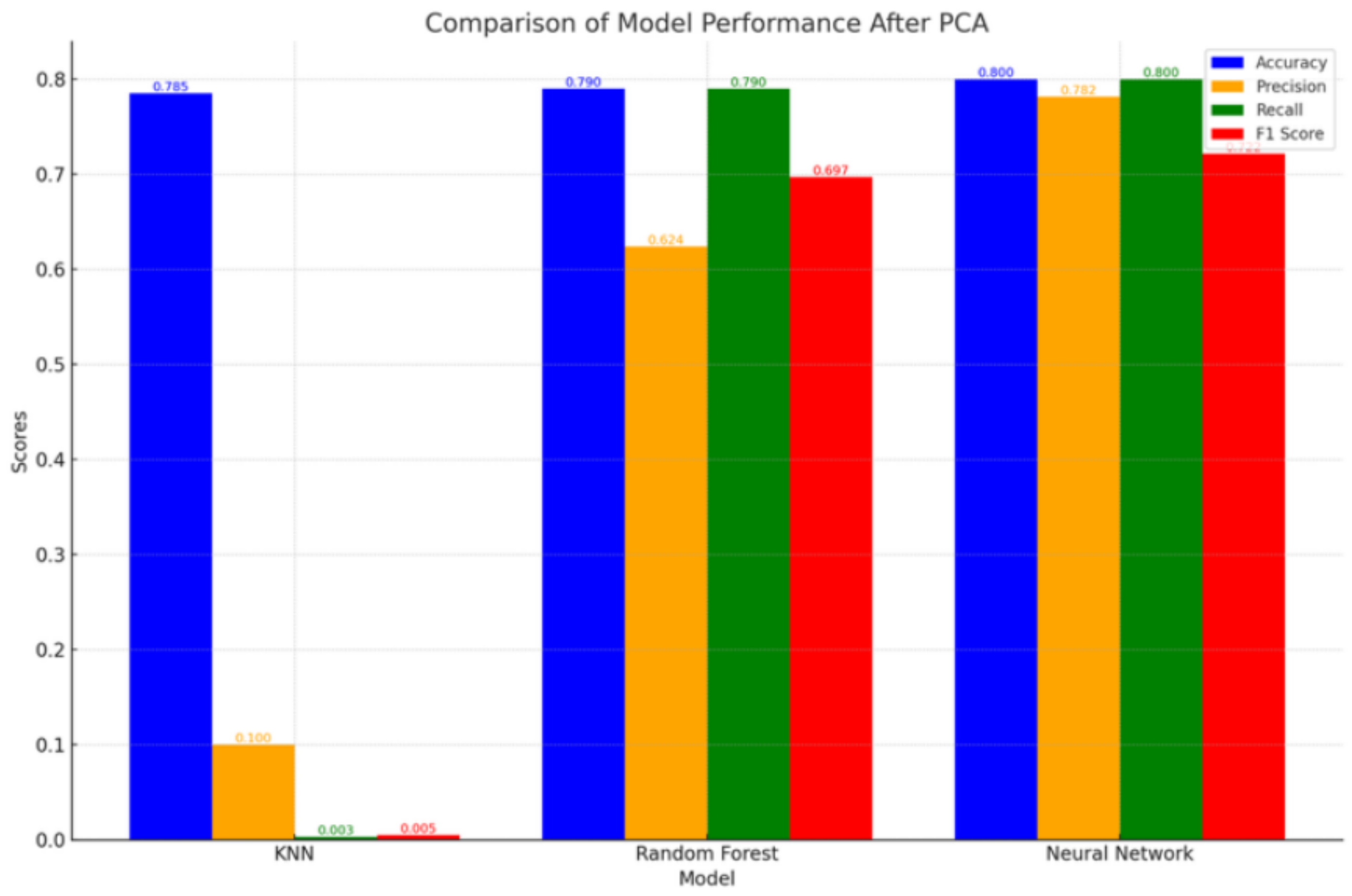

Impact of Dropout Rate on Neural Network Performance Metrics
Performance Metrics
Dropout Rate
0.80
0.78
0.76
0.74
0.72
0.70
0.68
0.66
0.64
0.100
0.125
0.150
0.175
0.200
0.225
0.250
0.275
0.300
Accuracy
Precision
Recall
F1 Score


Impact of Learning Rate on Neural Network Performance Metrics
Performance Metrics
Learning Rate
0.80
0.78
0.76
0.74
0.72
0.70
0.68
0.66
0.64
0.000
0.002
0.004
0.006
0.008
0.010
Accuracy
Precision
Recall
F1 Score



The Neural Network model achieved the highest accuracy at 79.5%, with a precision of 63.2%, recall of 79.5%, and an F1 score of 70.4%. This indicates that neural networks, with their capacity to learn complex patterns, performed the best among the models we tested.

## Comparison of Models



| Model | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| KNN | 0.785 | 0.100 | 0.003 | 0.005 |
| Random Forest | 0.790 | 0.624 | 0.790 | 0.697 |
| Neural Network | 0.795 | 0.632 | 0.795 | 0.704 |

From our analysis, the Neural Network model showed the best performance, followed by the Random Forest model. The KNN model, while simple and intuitive, did not perform well for this classification task.

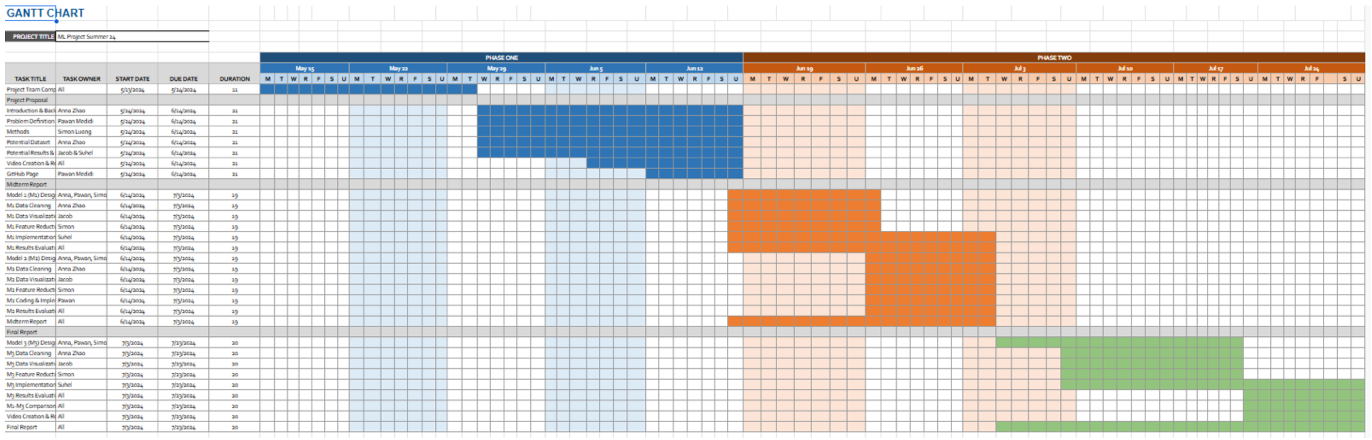
## Next Steps

Further optimize hyperparameters for the Random Forest and Neural Network models to potentially improve their performance.

Explore other advanced models like Transformers which might offer better performance in handling text data.

Consider using more sophisticated feature engineering techniques to enhance the model's ability to distinguish between human and bot comments.

## Gantt Chart



GANTT CHART

PROJECT TITLE: ML Project Summer 24

| TASK TITLE | TASK OWNER | START DATE | DUE DATE | DURATION |
|---|---|---|---|---|
| Project Team Comp | All | 5/13/2024 | 5/24/2024 | 11 |
| Project Proposal | | | | |
| Introduction & Back | Anna Zhao | 5/24/2024 | 6/14/2024 | 21 |
| Problem Definition | Pawan Medidi | 5/24/2024 | 6/14/2024 | 21 |
| Methods | Simon Luong | 5/24/2024 | 6/14/2024 | 21 |
| Potential Dataset | Anna Zhao | 5/24/2024 | 6/14/2024 | 21 |
| Potential Results & | Jacob & Suhel | 5/24/2024 | 6/14/2024 | 21 |
| Video Creation & Re | All | 5/24/2024 | 6/14/2024 | 21 |
| GitHub Page | Pawan Medidi | 5/24/2024 | 6/14/2024 | 21 |
| Midterm Report | | | | |
| Model 1 (M1) Desig | Anna, Pawan, Simo | 6/14/2024 | 7/3/2024 | 19 |
| M1 Data Cleaning | Anna Zhao | 6/14/2024 | 7/3/2024 | 19 |
| M1 Data Visualizati | Jacob | 6/14/2024 | 7/3/2024 | 19 |
| M1 Feature Reducti | Simon | 6/14/2024 | 7/3/2024 | 19 |
| M1 Implementation | Suhel | 6/14/2024 | 7/3/2024 | 19 |
| M1 Results Evaluati | All | 6/14/2024 | 7/3/2024 | 19 |
| Model 2 (M2) Desig | Anna, Pawan, Simo | 6/14/2024 | 7/3/2024 | 19 |
| M2 Data Cleaning | Anna Zhao | 6/14/2024 | 7/3/2024 | 19 |
| M2 Data Visualizati | Jacob | 6/14/2024 | 7/3/2024 | 19 |
| M2 Feature Reducti | Simon | 6/14/2024 | 7/3/2024 | 19 |
| M2 Coding & Imple | Pawan | 6/14/2024 | 7/3/2024 | 19 |
| M2 Results Evaluati | All | 6/14/2024 | 7/3/2024 | 19 |
| Midterm Report | All | 6/14/2024 | 7/3/2024 | 19 |
| Final Report | | | | |
| Model 3 (M3) Desig | Anna, Pawan, Simo | 7/3/2024 | 7/23/2024 | 20 |
| M3 Data Cleaning | Anna Zhao | 7/3/2024 | 7/23/2024 | 20 |
| M3 Data Visualizati | Jacob | 7/3/2024 | 7/23/2024 | 20 |
| M3 Feature Reducti | Simon | 7/3/2024 | 7/23/2024 | 20 |
| M3 Implementation | Suhel | 7/3/2024 | 7/23/2024 | 20 |
| M3 Results Evaluati | All | 7/3/2024 | 7/23/2024 | 20 |
| M1-M3 Comparison | All | 7/3/2024 | 7/23/2024 | 20 |
| Video Creation & Re | All | 7/3/2024 | 7/23/2024 | 20 |
| Final Report | All | 7/3/2024 | 7/23/2024 | 20 |

## Contribution Table

| Name | Final Contributions |
|---|---|
| Anna Zhao | Introduction/Background, Dataset Description, Data Preprocessing |
| Pawan Medidi | Problem Definition, Methods, GitHub Repository |
| Simon Luong | Model Implementation, Data Analysis, Data preprocessing |
| Suhel Keswani | Results and Discussion, Visualizations |
| Jacob Zeigler | Slides setup and Visualizations |

## References

[1] J. V. Fonseca Abreu, C. Ghedini Ralha and J. J. Costa Gondim, "Twitter Bot Detection with Reduced Feature Set," 2020 IEEE International Conference on Intelligence and Security Informatics (ISI), Arlington, VA, USA, 2020, pp. 1-6, doi: 10.1109/ISI49825.2020.9280525.

[2] M. Aljabri, R. Zagrouba, A. Shaahid, F. Alnasser, A. Saleh, and D. M. Alomari, “Machine learning-based social media bot detection: a comprehensive literature review,” Social Network Analysis and Mining, vol. 13, no. 1, Jan. 2023, doi: https://doi.org/10.1007/s13278-022-01020-5.

[3] K. Hayawi, S. Saha, M. M. Masud, S. S. Mathew, and M. Kaosar, “Social media bot detection with deep learning methods: a systematic review,” Neural Computing and Applications, vol. 35, Mar. 2023, doi: https://doi.org/10.1007/s00521-023-08352-z.