# MonkeyPox_Misinformation_P15

**Arun Kumar Ramesh** Unity ID: arames25
**Ashok Kumar Selvam** Unity ID: aselvam
**Justin Pahl** Unity ID: jppahl
**Sutapa Dey Tithi** Unity ID: stithi

## Background

Misinformation in media has always been a concern. In the past, access to media with large reach was controlled by major publishing or broadcast companies and as such the creation and mass dissemination of misinformation was more centralized and controlled.

With the development of the internet, anyone could become a publisher (i.e. by creating a website or blog) and as such, had the ability to create credible looking misinformation. Disseminating misinformation across the internet has the potential to sow division and hatred, and create widespread hysteria or paranoia. Therefore, it is crucial to verify the accuracy of the news or information.

In the beginning misinformation could be controlled through human content moderation. However, as more and more people become comfortable with using these platforms, this approach is quickly becoming unsustainable and fully automated content moderation seems the mostly feasible solution. It is equally important that these auto classifiers are perceived as trustworthy, leading to complex trade-offs that must be considered when developing the algorithms. (3; 4)

## Introduction

Our project uses a dataset from Kaggle, "Monkeypox misinformation: Twitter dataset"(1). This dataset contains 5787 Monkeypox-related tweets and their metadata, along with the ground truth classification. The metadata of our dataset are but not restricted to: 'the date of the tweet', 'user account creation date', 'user description', 'user is verified', 'user has url', 'reply count', 'retweet count', 'like count', 'quote count', 'followers count', 'users total tweets count', 'listed count', 'user location'.

In this project, we utilize various Natural Language Processing techniques to pre-process the tweets and other attributes in the dataset. We then use this pre-processed data to develop machine learning models to classify misinformation within a selection of Tweets relating to the recent Monkeypox outbreak. For the midway report, we focused on our EDA efforts, performing preprocessing and training simple Machine Learning Classification Algorithms. For the final report, we have completed training multiple machine learning models and evaluated various ways of combining the features and models to generate accurate results.

## Method

### Exploratory Data Analysis (EDA)

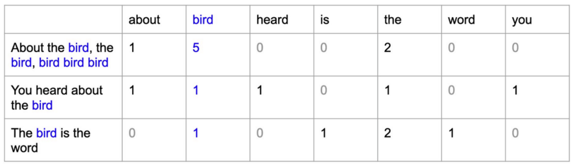
Firstly, we performed EDA to gain insights into the dataset and help us understand the data better. As part of the initial EDA, we created an attribute for the account age in months as well as a temporary binary attribute flagging if a user lists an alternate, right-leaning social media account in their user bio.

## Preprocessing

In addition to normalizing the various numeric features, preprocessing was performed on the text fields (tweets and user description) as well. First, for the tweets, we split out the hashtags to create a new text feature, then for all three text fields (tweet, hashtags, and user description) we removed the URLs, and other non-alpha-numeric characters present in the tweet. All characters were converted to lowercase, had stopwords removed and then tokenized. To evaluate different NLP techniques on the tokenized features, four methods were performed: Stemming w/ Count Vector (i.e., Bag of Words), Stemming w/ TF-IDF, Lemmatization w/ Count Vector, and Lemmatization w/ TF-IDF. The various methods are explained below.

Table 1: NLP Processing Techniques Used

| | |
|---|---|
| **Stemming** | An approach that removes the suffixes of words to get the root stem. For instance (e.g., writing becomes write). Performed using NLTK's PorterStemmer. |
| **Lemmatization** | An approach that gets the root word or 'lemma' from a given word (e.g., bought becomes buy). Performed using Spacy. |
| **TF-IDF** | A measure of the relevancy of a particular word (or token) present in the dataset. Generated using Sklearn's TfidfVectorizer. |
| **Count Vector** | A column of unique words (or tokens) occurring in the dataset where each row represents the frequency of occurrence of the words (or tokens) in the datapoint. Generated using Sklearn's CountVectorizer |



| | about | bird | heard | is | the | word | you |
|---|---|---|---|---|---|---|---|
| About the bird, the bird, bird bird bird | 1 | 5 | 0 | 0 | 2 | 0 | 0 |
| You heard about the bird | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| The bird is the word | 0 | 1 | 0 | 1 | 2 | 1 | 0 |

Example of Count Vectorization (2)

## Principal Component Analysis (PCA)

After preprocessing, we performed PCA on various splits to the dataset. By performing PCA, we reduced the number of features in our dataset that would be used for training the various classification models while retaining enough components to maintain 96% of the explained variance.

## Model Generation & Evaluation

We then split the dataset and used 80% for training and 20% for testing. We trained multiple Machine Learning Classification algorithms using combinations of different features, including KNN, SVM, Decision Tree, Random Forest, and Naïve Bayes.

We evaluated our various trained models against multiple metrics, including accuracy, precision, recall, and F1-score. Confusion matrices were also generated to evaluate the various model performances.

In our view, the most important factor in a classifier for misinformation is that it's considered authoritative by users. Maximizing the amount of actual misinformation correctly labeled (i.e., maximize Recall) and minimizing the number of Tweets incorrectly labeled as misinformation (i.e., maximize Precision) would best achieve this goal. Therefore, F1-Score was chosen as our target metric to optimize.

Model tuning was accomplished by performing a Grid Search using 10-fold cross-validation and multiple possible tuning parameters targeting our chosen metric of F1-Score.

As F1-Score was used as our model optimization metric, for the remainder of the report, this metric is referenced when evaluating the models and showing results. Similarly, for NLP encoding, Stemming w/ TF-IDF encoded strings contributed to the highest-performing models and will be the encoding method used unless stated otherwise. However, the full set of metrics is generated as part of the code available in our repository and the various NLP encoding methods.

## Experiment

**Hypothesis:** Using the Tweets and their accompanying metadata, is it possible to train a classifier to identify misinformation with high accuracy? If yes, what are the attributes or (combination of attributes) that can be used to identify the misinformation?

Through our various experiment setups described in the following section, we have evaluated different models and feature subsets to try and answer these questions.

### Dataset Down-sample

Table 2 shows the class distribution in the monkeypox.csv dataset. It contains 5787 tweets relating to the recent Monkeypox outbreak with class information as shown below.

Table 2: Class Information in the Original Dataset

| Category | Misinformation (1) | Not Misinformation (0) |
|---|---|---|
| Binary Class | 1069 | 4718 |

We noticed a heavy class imbalance in the original dataset. There were 1069 data points belonging to the Misinformation class and 4718 data points belonging to the Not-Misinformation class. The number of Not-Misinformation classes was more than four times the number of Misinformation classes, introducing bias into our initial models. To correct this, we decided to down-sample the Not-Misinformation class. The resultant dataset consisted of 1069 Not-Misinformation classes and 1069 Misinformation classes (maintaining a 1:1 class ratio).

Table 3: Class Information in the Down-Sampled Dataset

| Category | Misinformation (1) | Not Misinformation (0) |
|---|---|---|
| Binary Class | 1069 | 1069 |

### Experiment Design

Using the down-sampled dataset, we developed various Setups to train the model. This was done to identify the attribute combination that provides the best results. We trained Machine Learning Algorithms on each setup, and the results obtained are mentioned in the next section.

In addition, the applicable text fields for each of the Setups below were processed using both Lemmatization and Stemming to obtain the root words and then encoded using both a Bag of Words and TF-IDF processes. This resulted in four separate datasets for each text field that were then tested individually on each setup.

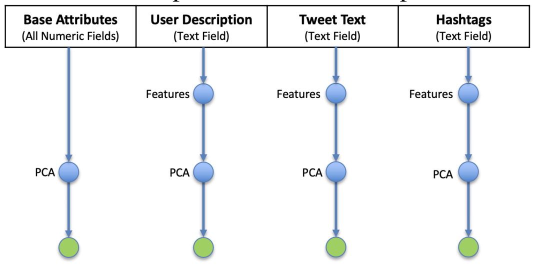
**Setup 1:** This Setup involves the Base Attributes as features. The Base Attributes include 'retweet count', 'like count', 'reply count', 'followers count', 'quote count', 'following count', 'tweet count', 'user is verified', and 'account age at time of tweet'. The Base Attributes being numerical data, are normalized. Feature reduction is performed using PCA, keeping a variance of 98% and the number of reduced components to 3.

**Setup 2:** In this setup, we only take the User Description attribute into consideration. The textual data is cleaned as described previously (i.e. non-ASCII characters, URLs, stop words removed, etc.) and encoded as described above. The pre-processed data now contains 2000 features. Feature reduction is performed with PCA, keeping the variance at around 96% and the number of reduced components around 1000.

**Setup 3:** In this setup, we will only consider the Tweets. Note that the hashtags are removed from the tweet texts for this setup. Similar to Setup 2, we preprocess the textual data to generate 2000 features. We then perform PCA, keeping the variance at around 96% and the number of reduced components around 1300.

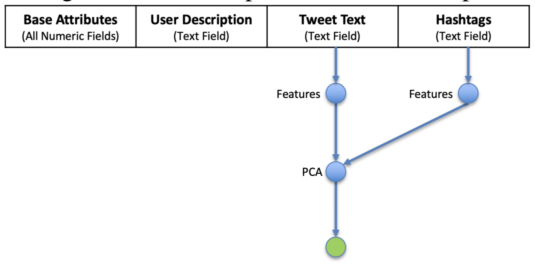
**Setup 4:** We extracted all the Hashtags from individual tweets and considered them as a separate feature for this setup. The Hashtags were preprocessed to create 2000 features in a similar fashion as the other textual data. We then perform PCA, keeping the variance at around 96% and the number of reduced components around 300.

Figure 1: Feature Split and Models, Setups 1, 2, 3, and 4



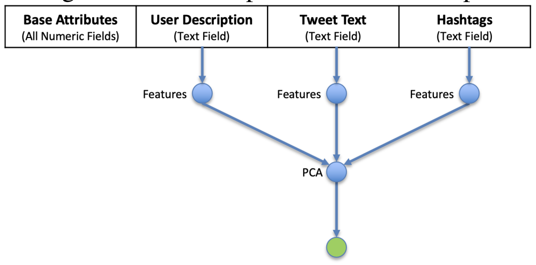
**Setup 5:** For these next experiments, we wanted to test if considering various text features together resulted in a better result. For this setup, we combined the tweets and the extracted hashtags. We preprocessed the Tweets and Hashtags separately by generating a Bag of Words and a TF-IDF model on the stemmed and lemmatized data with stop words removed. We then combined the generated features and performed PCA reduction keeping the variance at around 96% and the number of reduced components around 1000.

Figure 2: Feature Split and Model, Setup 5



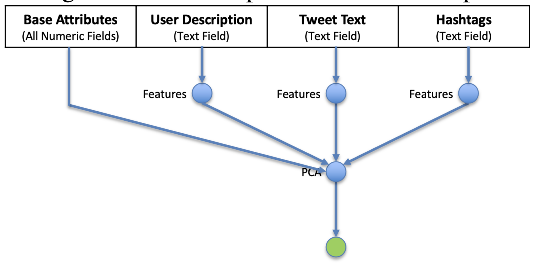
**Setup 6:** In this Setup, we included the User Description additionally to Setup 5. We performed similar preprocessing steps as the previous setup and combined the generated features. We then performed PCA reduction keeping the variance at around 96% and the number of reduced components around 2000.

Figure 3: Feature Split and Model, Setup 6



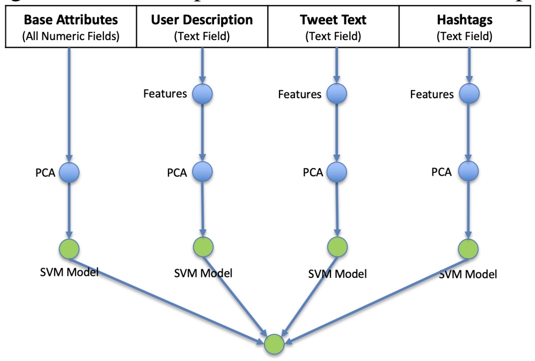
**Setup 7:** This Setup involves training on combined features of Tweets, Hashtags, User Description, and Base Attributes. The Tweets, Hashtags and User Description were preprocessed using the same process described previously. The Base Attributes being numerical data, were Normalized. The features are then concatenated, and PCA is performed for feature reduction. The variance is kept at around 96%, and the number of components after the reduction was around 2000.

Figure 4: Feature Split and Model, Setup 7



**Setup 8:** This Setup involves ensembling the results obtained from Setup 1, Setup 2, Setup 3, and Setup 4 with the aim to get better results. We then chose the best models from each setup and performed soft voting to obtain the classification output.

Figure 5: Feature Split and Ensemble Model Setup 8



## Results

### Exploratory Data Analysis (EDA)

After analyzing the various features in the dataset, the following results were found to be the most significant.

First, the user's verification status (i.e. blue check mark). Although the majority of users are not verified (i.e., verified users only account for 6.95% of the Tweets in the dataset), they do not show up in the same proportion in Tweets labeled as misinformation, accounting for only 0.65% of Tweets belonging to this class. Although it is important to note that with the changes to the system verifying users (i.e., moving from an invite system to a paid system), this relation may not be true in the future.

Second, a derived attribute was generated for the purposes of the EDA, signaling if the alternate accounts for right-leaning social media sites of the Tweet authors were listed in their bio (i.e., Truth Social, Parler, Gettr, or Gab). The results show that although users with this attribute make up only 0.71% of the Tweets in the dataset, they account for 1.68% of the Tweets labeled as 'Misinformation'. Interestingly, if you further analyze only the tweets made by these users with this attribute, 44% of their tweets are classified as 'Misinformation'. This shows that keywords in the user description should also be useful for classification in the final model.

Third, we considered another derived feature, "account age", which is the age of a user account when that user posted that corresponding tweet. (see Table 4). Tweets from accounts created more recently are more likely to produce misinformation.

Table 4: Class Distribution in Different Age (Account) Groups

| Age (Account) Groups | Percentage of Misinformation Tweets out of total tweets by that group |
|---|---:|
| [0, 1) | 33.576% |
| [1, 5) | 32.882% |
| [5, 10) | 26.470% |
| [10, 25) | 23.088% |
| [25, 50) | 18.960% |
| [50, 100) | 17.378% |
| [100, 200) | 11.818% |

Finally, the number of followers (see Table 5). Tweets from accounts with fewer followers showed an increase in the amount of misinformation.

Table 5: Class Distribution in Different Follower Counts Groups

| Followers Count Groups | Percentage of Misinformation Tweets out of total tweets by that group |
|---|---:|
| [0, 25) | 28.628% |
| [25, 50) | 20.895% |
| [50, 100) | 20.493% |
| [100, 1000) | 20.184% |
| [1000-) | 13.123% |

**Final Classification Models**

After training multiple models, using a Grid Search technique with 10-fold cross-validation to tune the parameters optimizing for F1-Score, we were able to achieve high-performing models for many of the experimental setups. Shown below are the final F1-Scores for the various setups, with the highest performing tuned model in shaded, bold text.

Table 6: Setup 1 (Base Attributes)

| Preprocessing | KNN | SVM | RF | DT | NB |
|---|---:|---:|---:|---:|---:|
| Normalization | 72.300 | **74.72** | 72.68 | 68.08 | 73.80 |

Table 7: Setup 2 (User Description)

| Text Preprocessing | KNN | SVM | RF | DT | NB |
|---|---:|---:|---:|---:|---:|
| Lemmatization (CV) | 73.52 | 77.23 | 77.15 | 61.97 | 59.76 |
| Lemmatization (TF-IDF) | 69.28 | 80.24 | 75.57 | 62.52 | 53.71 |
| Stemming (CV) | 74.65 | 79.51 | 79.16 | 68.49 | 55.74 |
| Stemming (TF-IDF) | 69.47 | **82.80** | 77.54 | 66.213 | 55.73 |

In Setup 1, almost all the models gave similar results. The best one was SVM with 74.72% F1-score.

In Setup 2, we collected results with all 4 combinations of NLP techniques. SVM with Stemmed TF-IDF worked the best with 82.8% F1-score.

Table 8: Setup 3 (Tweets)

| Text Preprocessing | KNN | SVM | RF | DT | NB |
|---|---:|---:|---:|---:|---:|
| Lemmatization (CV) | 84.57 | 91.50 | 86.20 | 65.64 | 59.85 |
| Lemmatization (TF-IDF) | 92.55 | 93.92 | 87.20 | 71.96 | 66.97 |
| Stemming (CV) | 89.55 | 91.90 | 87.91 | 71.96 | 62.18 |
| Stemming (TF-IDF) | 91.46 | **94.58** | 88.44 | 73.94 | 66.06 |

Table 9: Setup 4 (Hashtags)

| Text Preprocessing | KNN | SVM | RF | DT | NB |
|---|---:|---:|---:|---:|---:|
| Lemmatization (CV) | 67.69 | 68.04 | 68.37 | 67.67 | 67.86 |
| Lemmatization (TF-IDF) | 68.46 | 68.04 | 68.71 | 68.36 | 67.86 |
| Stemming (CV) | 68.04 | 68.03 | 68.26 | 68.12 | 67.32 |
| Stemming (TF-IDF) | 68.25 | **68.71** | 68.04 | 68.36 | 67.64 |

In Setup 3, all the models except Decision Tree and Naive Bayes gave a good result. SVM with Stemmed TF-IDF worked the best with 94.58% F1-score.

In Setup 4, all the models performed poorer than our other setups. The best of these models was SVM with stemmed TF-IDF with F1-score of 68.71%. Although it is worth noting that by simply training on hashtags, a classifier was able to perform better than a 'coin flip' (i.e., > 50%), so it was included in Setup 8 (i.e., the Ensemble model)

Table 10: Setup 5 (Tweets and Hashtags)

| Text Preprocessing | KNN | SVM | RF | DT | NB |
|---|---|---|---|---|---|
| Lemmatization (CV) | 82.46 | 92.45 | 86.63 | 69.90 | 60.29 |
| Lemmatization (TF-IDF) | 92.77 | 93.86 | 88.74 | 70.47 | 66.97 |
| Stemming (CV) | 88.88 | 92.12 | 88.053 | 72.03 | 62.92 |
| Stemming (TF-IDF) | 91.25 | **94.33** | 90.04 | 76.81 | 66.21 |

Table 11: Setup 6 (User Desc., Tweets, HT's)

| Text Preprocessing | KNN | SVM | RF | DT | NB |
|---|---|---|---|---|---|
| Lemmatization (CV) | 86.18 | 93.17 | 86.40 | 65.20 | 59.13 |
| Lemmatization (TF-IDF) | 85.52 | 92.48 | 88.98 | 61.57 | 69.10 |
| Stemming (CV) | 88.99 | 93.33 | 88.14 | 64.91 | 61.82 |
| Stemming (TF-IDF) | 86.04 | **94.61** | 90.20 | 66.18 | 68.33 |

In Setup 5, all the models except Decision Tree and Naive Bayes performed well. The best one was SVM with stemmed TF-IDF with F1-score of 94.33%.

In Setup 6, similar to the previous setup, all the models except Decision Tree and Naive Bayes performed well. The best one was SVM with stemmed TF-IDF with F1-score of 94.61%.

Table 12: Setup 7 (Base Attributes, User Description, Tweets, and Hashtags)

| Text Preprocessing | KNN | SVM | RF | DT | NB |
|---|---|---|---|---|---|
| Lemmatization (CV) | 82.49 | 91.33 | 88.83 | 64.18 | 63.25 |
| Lemmatization (TF-IDF) | 87.75 | 92.52 | 88.73 | 67.79 | 70.18 |
| Stemming (CV) | 84.21 | 89.41 | 88.94 | 72.22 | 64.89 |
| Stemming (TF-IDF) | 82.47 | **93.89** | 89.83 | 70.32 | 72.81 |

In Setup 7, similar to the previous setups, all the models except Decision Tree and Naive Bayes performed well. The best one was SVM with stemmed TF-IDF with F1-score of 93.89%.

As we combine the datasets, the performance of our KNN model is reduced significantly. Random Forest was consistent across all the combinations from setup 3. SVM was also steady, with a peak in Setup 6.

Table 13: Setup 8 Ensemble (Base Attributes, User Description, Tweets, and Hashtags)

| Text Preprocessing | Tweets | Hashtags | User Description | Base Attributes | Ensemble |
|---|---|---|---|---|---|
| Stemming (TF-IDF) | 94.58 | 68.71 | 82.80 | 74.72 | **87.63** |

In Setup 8, we tried an ensemble model with soft scoring. This performed well but did not exceed the performance of the other setups. The F1-score of this model was 87.63%.

## Conclusion

Across all experiments, our best performing model (and final selection) was an SVM model (Kernel: Gaussian) trained on all the text fields (i.e., Setup 6: Tweet, Hashtags and User Description, see Table 11) using Stemmed TF-IDF to preprocess the textual data.

SVM tends to perform well for datasets with a large number of components. Since we had 2000+ components, SVM proved to perform best with this dataset. TF-IDF also performed better when compared to Count Vectorization as it considers the importance of a word in the dataset. Usually, Lemmatization performs better than Stemming. Lemmatization considers the context of the word in the sentence. Since tweets do not necessarily follow proper grammatical structure, we suspect the words' lemma was not able to be identified correctly. Contrastingly, stemming works on individual words or tokens, and it performed marginally better in our experiments (i.e., 0.2% increase in F1-Score).

In addition, we feel that given more experimentation and time, the Ensemble model could be improved (i.e., evaluating different weights applied to the soft voting algorithm) and possibly provide superior results. (see Table 13).

Our main findings showed that just the Tweet text and a well-optimized model were able to achieve excellent results. Adding in other textual data (such as user descriptions and hashtags improved the F1-Score marginally). Further, including the other numeric meta-data (such as likes, retweets, etc.) did not improve the results for this dataset.

From available studies, (3; 4) and early findings from our EDA, we had expected that numerical features would have played a larger role in our final models but were instead dominated by the text features. We believe this is mostly due to the dataset's size, quality, and maturity. From the published Monkeypox twitter research (3), there was shown to be a strong correlation between numerical features related to engagement (i.e., likes, retweets, etc.) and misinformation which was not as strong in our dataset. A lesson learned during this project would be to select a more mature and well-studied dataset (i.e., A public COVID-19 dataset) and then apply the learning from those models to an emerging problem like Monkeypox. However, even without the assistance of numerical data, we were able to achieve an excellent result relying primarily on the processed textual data (i.e., Tweet, Hashtags, and User Description).

## References

[1] Stephen Krone, "Monkeypox misinformation: Twitter dataset", `https://www.kaggle.com/datasets/stephencrone/monkeypox`

[2] Ajitesh Kumar, "Text Classification using Bag-of-words Model", `https://vitalflux.com/text-classification-bag-of-words-model-python-sklearn/`, August 2021

[3] Yeimer Ortiz-Martínez., Jheinner Sarmiento., D Katterine Bonilla-Aldana., Alfonso J Rodríguez-Morales (2022), *Monkeypox goes viral: measuring the misinformation outbreak on Twitter*, `https://jidc.org/index.php/journal/article/view/35905027/2876`, J Infect Dev Ctries 2022; 16(7):1218-1220.

[4] Yeimer Ortiz-Martínez., Jheinner Sarmiento., Chayakrit Krittanawong, MD., Bharat Narasimhan, MD., Hafeez Ul Hassan Virk, MD., Harish Narasimhan, MHS., Joshua Hahn, MD., Zhen Wang, PhD., W.H. Wilson Tang, MD., *(2020) Misinformation Dissemination in Twitter in the COVID-19 Era*, `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7426698/`, The American Journal of Medicine Volume 133 Issue 12

## Repo Location

`https://github.ncsu.edu/aselvam/engr-ALDA-Fall2022-P15.git`