Unveiling Clickbait and Unmasking Fake News: A Multimodal Analysis Leveraging User Influence on Twitter

Submitted by

Anurag (13000120103)
Abhinab Paul (13000120107)
Uttkarsh Ranjan (13000120109)
Priyasu Guin (13000120117)

June, 2024

Submitted for the partial fulfillment for the degree of Bachelor of Technology in Computer Science and Engineering

TECHNO MAIN SALT LAKE
EM 4/1, SALT LAKE, SECTOR - V, KOLKATA - 700091

.

CERTIFICATE

This is to certify that the project entitled "Unveiling Clickbait and Unmasking Fake News: A Multimodal Analysis Leveraging User Influence on Twitter" prepared by Anurag (13000120103), Abhinab Paul (13000120107), Uttkarsh Ranjan (13000120109) and Priyasu Guin (13000120117) of B.Tech (Computer Science & Engineering), Final Year, has been done according to the regulations of the Degree of Bachelor of Technology in Computer Science & Engineering. The candidates have fulfilled the requirements for the submission of the project report.

It is to be understood that, the undersigned does not necessarily endorse any statement made, opinion expressed or conclusion drawn thereof, but approves the report only for the purpose for which it has been submitted.

(Signature of the Internal Guide)

(Signature of the HOD)

(Signature of the External Examiner)

.....

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

.....

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our project guide in Computer Science and Engineering department. We are extremely thankful for the keen interest he / she took in advising us, for the books and reference materials provided for the moral support extended to us.

Last but not the least we convey our gratitude to all the teachers for providing us the technical skill that will always remain as our asset and to all non-teaching staffs for the gracious hospitality they offered us.

Place: Techno India, Salt Lake
Date: June 1, 2024

.....
Anurag (13000120103)

.....
Abhinab Paul (13000120107)

.....
Uttkarsh Ranjan (13000120109)

.....
Priyasu Guin (13000120117)

Contents

Abstract

With increasing availability and reach of social media and the internet as a whole, access to information of any kind has become quite simple. However, in today's evolved era consisting of an alliance with Artificial Intelligence, the authenticity of all such information available at such ease, is questionable. Fake content is no longer limited to just social media platforms, rather it has blended with such precision that it has become almost impossible to separate it out from the actual source of truth. Thus, through this project, we aim to help the users in verifying the authenticity of the information they consume on the internet.

Our goal is to recognize and protect the users against fake news on various platforms which can in the form of an AI generated fake media (images and videos), false facts and text based articles as well as deceiving phishing links.

After extensive research on the topic, we decided to follow a multimodal approach to solve our problem. Fake content is not just limited to images or text, but to all forms of multiple forms of input formats. Thus using a multimodal approach, we would be able to detect fake content present in multiple formats. On top of that we've used multiple models in our project. Taking multiple opinions into consideration proved out to be better than relying on just one modal. We've incorporated different models as SVM, XGBoost, Random Forest, etc which together formed an ensemble model which is a key differentiating factor of our work. These models were used for textual analysis of our data, followed by which several other models were compared and then used for verification of URLs (if any) included in the data. To detect the type of fake news we incorporated the use of logic rather than training another set of models for our project, which leverages multiple factors in further classifying a piece of fake news. On the multimedia front, we tackled the problem of fake generated images, using an ensemble approach of multiple CNN models namely ResNet-50, VGG-16 and EfficientNetV2.

Despite having several fake news detectors, processing both multimedia and text are difficult. Human sentiments like satire are very common, but having the ability to accurately identify them is where research is still going on. On top of that, we are looking to classify news on the basis of the links contained in them. Thus, with the help of this project, we plan on overcoming these challenges to be able to help others in knowing the truth about the facts that they're coming across.

1 Introduction

1.1 Problem Domain

In today's tech infused world, it's safe to say that almost all of us are familiar with AI and its countless use cases that we've found till date. However, as all things which are beneficial to us, bring their own set of drawbacks, AI is no exception. And with such ease of access to social media, the power of AI can be very easily misused in the form of fake news and information to name a few. Considering this particular issue, we started this project with the goal to recognize and protect the users against fake news on social media platforms (mainly Twitter, now X) which can in the form of an AI generated fake media or text based tweets which might contain some deceiving phishing links as well.

Dealing with such different types of input data, we had to take a multimodal approach to tackle the problem in discussion. As of last semester, we dealt with the textual side of things, more precisely we were able to train models which could detect whether a particular tweet is sharing fake news or not, and whether they contain any phishing links or not. Moving in the direction of the goal that we set, in this semester, we focused on the media part of our multimodal approach. In addition to that, we also enhanced the detection of the type of fake news that a tweet might be sharing.

For handling media, we would be needing a system that would keep up, by itself, with the ever increasing amount of false content on the internet. Thus, the most probable approach in handling this scenario would be using Neural Networks, more specifically, Deep Learning.

In order to incorporate and work on all these different forms of data, we would need to train a model which could handle each type of media, and thus Multimodal Deep Learning would be the go-to approach.

1.2 Glossary

Terms	Definitions
Multimodal Learning	A type of learning where the modal is trained to understand and work with multiple forms of input data, like text, images, audio, etc.
Sentiment Analysis	The process of computationally identifying and categorizing opinions expressed in a piece of text.
Ensemble Learning	A machine learning paradigm where multiple learners (models) are trained to solve the same problem.
Deep Learning	A machine learning technique that uses multiple layers of nonlinear processing units for feature extraction and transformation, enabling the system to learn representations of data with multiple levels of abstraction.
CNN	Convolutional neural network (CNN) is a deep learning architecture that applies convolutional filters to input data, such as images, to automatically learn hierarchical representations for tasks like image classification and object detection.
Misinformation	When any false information is shared, but unintentionally or by mistake. Can be further classified into categories like "spam" or "satire"
Disinformation	When any false information is shared, and it is intentional or deliberate.
Satire	Literary technique that blends criticism with humor to expose and ridicule societal follies or vices.
Spam	Unsolicited and excessive digital messages or content sent indiscriminately.

2 Literature Review

While trying to figure out different ways to solve a problem that's bugging us, one of the most efficient and smart ways to find an efficient approach is to go through any of the existing work that is going on in our field of interest. And so, we explored the internet trying to find suitable research works that'd push us into the right direction towards extracting a meaningful solution to our problem.

User Influence: If we're working with social media platforms, there aren't many ways in which we can verify the credibility of the information, other than the source

from where we're getting this piece of data. And such similar thinking was shown light in *Measuring User Influence on Twitter* [7] where researchers have used several different ways of detecting such user characteristics on Twitter. To summarize their work, they worked with *PageRank*, *TSPR* and *HITS* models and dealt with textual and post metadata. After going through their findings we were able to familiarize ourselves with various measures to produce different rankings. However, the reason we still needed to work on a solution of our own was due to the fact that we didn't have complete access to Twitter data and heavy computational requirements amongst other factors.

Now, knowing what we need to extract was one thing, but from where should we be extracting this information? For example, if we have two people: one of them is a celebrity and the other is a common man. If both of them say something, people are more likely to give importance to the celebrity in this case. Thus, identification of such authorities was crucial, and *Identifying Topical Authorities in Microblogs* [1] where Gaussian Mixture Model and Gaussian based ranking and clustering was used, helped us in understanding the same. We came across multiple methods that could help us in knowing the rankings amongst different authorities present on some platform. *Identifying Opinion Leaders on Social Networks Through Milestones Definition* [6] also helped us in this scenario where Milestones Rank was used which was a new centrality measure based on a user's interest in a topic and exclusiveness. Although as much precise as we'd want to be, it has its own downfalls which was seen in the form of the need for an appropriate set of milestones related to the topic in this work, along with manual fixation of the various weights used here.

Existing Datasets: Since we wanted to train our models on some dataset, the best way would have been to search for any existing ones, that were also having credible sources. C*IC Truth Seeker Dataset 2023* [5] was one such dataset that ticked all most of our boxes. It was based on Twitter, so we didn't need to access the tweets ourselves, they had already worked on the tweet's metadata and used BERT and ROBERTa models to classify them as true or false.

For the image dataset, we went ahead with *CIFAKE: Image Classification and Explainable Identification of AI-Generated Synthetic Images* [8] , as it was based on a research paper where the researchers used Stable Diffusion 1.4 to generate synthetic images which would act like fake ones for our model to be trained upon. Moreover, it had plenty of images that would easily help our model to achieve desired results.

Pre Built Models: Working with human sentiments posed the greatest challenge. As it is still an ongoing field of work, we ourselves weren't able to train a model for that due to various factors like lack of datasets to name a few. Thus, we decided that we'd use a pre-trained model which could detect human semantics like positive, neutral or negative versions of sentences which we later could use as a crucial factor in determining the type of fake news that we're detecting. We used *TweetNLP Cutting-Edge Natural Language Processing For Social Media* [9] for this task as it was again backed by robust transformer based models like Tweet-Eval, TimeLMs and XLM-T. However, even though these models consistently outperformed or matched the baselines, they still had to suffer with the biases present in social media data on which these models were trained on.

Clickbaits Detection: How do we know that a particular link would be a potentially phishing one? What all factors do we need to consider for it? How do we detect such links from social media? All these questions were rattling our brains untill we came across Detecting Clickbait in Online Social Media [2]. Having used multiple models like XGBoost, Random Forest, Decision Trees, etc, we got to know what type of models could be used for such scenarios. Even though the models here were pre trained, we couldn't use them due to their lower accuracy scores and potential of overfitting on their small training set, but that didn't mean that this work didn't inspire us to move in the correct direction.

We have tabulated an overview on some of the previous approaches used in the analysis of social media content, from where we draw inspiration for our project.

Table 1: Literature Review

Title of Paper	**Model Used**	**Type of Input Data**	**Data Set**	**Results / Findings**	**Limitations**
Measuring User Influence on Twitter	PageRank, TSPR, HITS	Textual data and post metadata	Stanford Large Dataset Collection, Higgs Boson dataset 2012, Custom datasets	Different measures produce different rankings, PageRank based measures tend to correlate with real-world influence better	Restricted access to full Twitter data, Computationally intensive, Spammers can exploit simplistic measures
Identifying Topical Authorities in Microblogs	Gaussian Mixture Model, Gaussian based ranking, Clustering	Textual tweets and retweets	Twitter Firehose dataset	Achieved precision scores between 0.6-0.8 in identifying top authors, Gaussian-based ranking was more effective than list-based ranking, probabilistic clustering was important for robustness	Uses simple keyword matching to extract tweets, performing LDA on a large scale can be computationally expensive, did not fully explore the optimal weights to assign to different features
CIC Truth Seeker Dataset 2023	J48 Decision Tree, Random Forest, IBK KNN, Ada Boost, BERT, BERTweet, RoBERTa	Tweet text and metadata	Custom Twitter Dataset (TruthSeeker) labeled using verified crowdsourcing	BERTweet performed the best overall with 96.1% accuracy for binary classification, RoBERTa achieved the highest performance for 4-label classification with 68.93% accuracy	Focus solely on English language, challenges to address regarding automatically detecting fake content

Title of Paper	**Model Used**	**Type of Input Data**	**Data Set**	**Results / Findings**	**Limitations**
Detecting Clickbait in Online Social Media	XGBoost, Random Forest, AdaBoost, Decision Tree	Images, text titles, links to articles from social media and their content	Small initial training set with 2,495 posts, Large training and validation set with 19,538 posts	Best model was XGBoost classifier with 73.2% accuracy on training and 81.2% accuracy on validation set	Potential overfitting of mdoel, no analysis of performance per clickbait category, abilty to scale
Identifying Opinion Leaders on Social Networks Through Milestones Definition	Milestones Rank: A new centrality measure based on a user's interest in a topic and exclusiveness	Tweets, retweets, and replies from users related to a topic	12,910,703 tweets geolocated in the US about the nomination of Brett Kavanaugh to the Supreme Court.	MilestonesRank was able to identify opinion leaders related to the topic, good standard deviation and number of distinct values, identified anti-Trump and pro-Trump opinion leaders	Requires defining an appropriate set of milestones related to the topic, weights for milestones need to be set manually, not compared against state-of-the-art machine learning methods
CIFAKE: Image Classification and Explainable Identification of AI-Generated Synthetic Images	Stable Diffusion 1.4, CNN, Grad-CAM	CIFAR-10 dataset (60,000), Synthetic images (60,000)	Combined dataset - CIFAKE	The CNN model achieved 92.98% accuracy in classifying, Grad-CAM interpretation revealed that the model focuses on small visual imperfections in the background	the dataset may need to be updated with newer, more advanced synthetic images, the study focused on a specific dataset (CIFAR-10) and not generalized

,

Title of Paper	**Model Used**	**Type of Input Data**	**Data Set**	**Results / Findings**	**Limitations**
TweetNLP Cutting-Edge Natural Language Processing for Social Media	Transformer-based models - TweetEval, TimeLMs, and XLM-T	Textual post content	Large Twitter corpora containing millions of tweets	TweetNLP models consistently outperform or match the baselines, highlighting the importance of domain-specific pretraining and fine-tuning	Biases present in the social media data used for pre-training and fine-tuning the models.

,

3 Problem Definition and Preliminaries

3.1 Scope

- Our project focuses on identifying fake information on social media platforms, more specifically Twitter (now X).
- This project will allow users to verify the authenticity of tweets on the platform.
- The current scope of the project deals with textual and image data present in the tweet
- The type of fake news being detected will be one of the following types: [“misinformation”, “disinformation”, “spam”, “satire”].

3.2 Exclusions

This project does not involve other social media platforms except Twitter (for now).

3.3 Assumptions

- Accurate data about the alleged tweet is available.
- External sources used in the project provide correct information.

- The model will be able to generalize to new data which is not included in the training set.

3.4 Our Contribution

To deal with the given problem that we've mentioned above, we have used several ways to help the users stay safe from any kind of fake news available on the social media platforms (Twitter). We've successfully been able to achieve the following milestones:

- **Tweet Prediction** (True/False)
  We are able to accurately detect whether a particular tweet is sharing some fake or real news with the help of our Tweets Classifier. This classification merely depends upon the textual content of the tweet namely, the tweet body.

- **Link Prediction** (whether it contains a legitimate/malicious URL)
  Our Links Classifier is able to categorize a given url as a potential phishing or a real link with satisfactory results. This classification is much needed since we may encounter cases where the tweet might be sharing some real textual content but might also have links to some phishing sources at the same time.

- **Semantics Classification**
  Once we are able to classify a tweet as false, we further try to detect the type of fake news that a user was sharing, namely one of the following types: *misinformation*, *disinformation*, *spam* or *satire*.

- **Media Classification**
  As we mentioned earlier that type of fake content is not limited to just textual data, we have worked on detecting fake media as well, more precisely, fake images. We have used multiple CNN models to compare with one another so that we could end up with the best possible way to be able to distinguish between real and fake images (like Deep Fakes) that could be shared along with a tweet.

4 Dataset

4.1 Creation and Preparation

4.1.1 TweetaVerse Dataset Creation

For our project, we first searched for multiple existing datasets which would help us in detecting fake news in tweets. The datasets which we came across have been listed below:

- **ISOT Fake News Dataset**
  - It provides us access to fake and real news articles, the true ones of which were collected from a news website Reuters.com (*https://www.reuters.com/*) and the rest were collected from Politifact (*https://www.politifact.com/*).
  - The reason why we didn't proceed with this dataset is due to the lack of relations with Twitter. This dataset mainly consisted of the news article's title and its content. However, for our project, we needed a dataset which would consist of information related to tweets as well.
  - Link: *https://onlineacademiccommunity.uvic.ca/isot/2022/11/27/fake-news-detection-datasets/*

- **FakeNewsNet Dataset**
  - As the description for this dataset says, "it is an ongoing data collection project for fake news research". It consists of news articles from BuzzFeed (*https://www.buzzfeed.com/in*) and Politifact (*https://www.politifact.com/*) and their online engagement on Twitter.
  - This dataset provides access to datasets from Politifact as well as GossipCop.
  - Even though this dataset provided us with the Tweet URLs along with title and the URL for the news articles related to the content in those tweets, the problem was that most of the URLs present in this dataset were found

to be not working anymore. Moreover, due to the added restrictions of accessing the Twitter API, accessing information from so many tweet IDs was just not feasible.

- Link: *https://github.com/KaiDMML/FakeNewsNet*

- **TruthSeeker: The Largest Social Media Ground-Truth Dataset for Real/Fake Content** [5]

  - This dataset provides us with the largest ground truth fake news content related to Twitter.
  - It was prepared by crawling of tweets of real and fake news from Politifact Dataset:
    * *https://github.com/KaiDMML/FakeNewsNet/blob/master/dataset/politifact_fake.csv*
    * *https://github.com/KaiDMML/FakeNewsNet/blob/master/dataset/politifact_real.csv*

    and then taking opinions from multiple users on Amazon Mechanical Turk to classify the tweets on how closely they agree with real/fake news statement. In addition to that, TruthSeeker also provides us with metadata about the tweets which turned out to be very beneficial for our project. [5]
  - Link: *https://www.unb.ca/cic/datasets/truthseeker-2023.html*

Thus, after coming across several articles, datasets and research we figured out that we had mainly two options:

- Curate our own dataset
- Use an existing dataset which was very recent

We could not proceed with the first option as Twitter has made several changes to its API policies, thus posing multiple restrictions on the metadata of the tweets

which was easily available before. Moreover we can only access the API a limited number of times per month with the standard version.

So, keeping in mind all the factors we decided to proceed with the TruthSeeker dataset as the main foundation dataset of our model. However, TruthSeeker had many additional features which we didn't require and thus, decided to drop off the following text and lexical features:

- **unique count**: number of unique, complex words
- **total count**: total number of words
- **ORG percent**: Percent of text including spaCy ORG tags
- **NORP percent**: Percent of text including spaCy NORP tags
- **GPE percent**: Percent of text including spaCy GPE tags
- **PERSON percent**: Percent of text including spaCy PERSON tags
- **MONEY percent**: Percent of text including spaCy MONEY tags
- **DATA percent**: Percent of text including spaCy DATA tags
- **CARDINAL percent**: Percent of text including spaCy CARDINAL tags
- **PERCENT percent**: Percent of text including spaCy PERCENT tags
- **ORDINAL percent**: Percent of text including spaCy FAC tags
- **LAW percent**: Percent of text including spaCy LAW tags
- **PRODUCT percent**: Percent of text including spaCy PRODUCT tags
- **EVENT percent**: Percent of text including spaCy EVENT tags
- **TIME percent**: Percent of text including spaCy TIME tags
- **LOC percent**: Percent of text including spaCy LOC tags
- **ORG percent**: Percent of text including spaCy ORG tags
- **WORK OF ART percent**: Percent of text including spaCy WOA tags

- **QUANTITY percent**: Percent of text including spaCy QUANTITY tags
- **LANGUAGE percent**: Percent of text including spaCy LANGUAGE tags
- **Max Word**: length of the longest word in the sentence
- **Min Word**: length of the shortest word in the sentence
- **Avg Word Length**: average length of words in the sentence
- **present verb**: number of present tense verbs
- **past verb**: number of past tense verbs
- **adjectives**: number of adjectives
- **pronouns**: number of pronouns
- **TO's**: number of to usages
- **determiners**: number of determiners
- **conjunctions**: number of conjunctions
- **dots**: number of (.) used
- **exclamations**: number of (!) used
- **question**: number of (?) used
- **ampersand**: number of (&) used
- **capitals**: Number of capitalized letters
- **quotes**: number of quotation marks used
- **digits**: number of digits (0-9) used
- **long word freq**: number of long words
- **short word freq**: number of short words
- **URLs**: whether the user has a provided a url in relation to their profile

- **normalized influence**: influence score the user has, normalized

From the given dataset, we were able to procure the category of the headline as well using Zero Shot Classification as one of the following: ["politics", "world", "finance", "technology", "sports", "entertainment", "health", "education", "crime", "business", "social"]

Based on the Bot Score and Cred Score provided by TruthSeeker, we calculated them ourselves using the formulas explained in Section 6.1. We also added an additional feature called 'Exclusivity' (also explained in section 6.1). After adding all the necessary features required to calculate the Bot_Score, Cred_Score and Exclusivity, we ended up with a total of 27 features and 134194 records (*https://docs.google.com/spreadsheets/d/1q5xWxS_bz7RfsOVdDNSUaxq3Mt4xevTQ7SeI8c65bdc/edit?usp=sharing*) We tried out different splits (Section 6.2) and finally went with the 80:20 split:

- **Training Set:** 107355 records, 6 features
  Link: *https://docs.google.com/spreadsheets/d/1GDf1B2THRyy1ja3vimr-j4Sb4Z-YcPn2x7E95-0TBmEM/editgid=910528710*

- **Test Set:** 26839 records, 5 features
  Link: *https://docs.google.com/spreadsheets/d/1LKpuvTNWMgfXPUhLpVkMR-zb6_conIaxs-eZNd9KdbIM/editgid=629609330*

4.1.2 Preparing the URLGuardian Dataset

Looking at the workflow for our project, after working with the Twitter dataset which we derived from the TruthSeeker dataset, we get to know whether a particular tweet is spreading fake or real news. If the tweet is sharing fake news, we then and there declare that it is false news. But in case the news being shared is true, we proceed to check if the tweet consists of a link or not. We then check whether the particular link is a phishing link or not. Unlike the search for a Twitter dataset, phishing link datasets were readily available online and we used a combination of two of them:

- **PhishTank**

  - This service provides a set of phishing URLs in multiple formats like csv, json etc. that gets updated hourly. We took 10,000 samples from this dataset of phishing links.
  - Link: *https://www.phishtank.com/developer_info.php*
  - The dataset was obtained in the following format:
    * **phish_id:** The unique ID provided by PhishTank.
    * **phish_detail_url:** PhishTank detail url for the phish, where users can view additional data related to it.
    * **url:** The phishing URL.
    * **submission_time:** The date and time at which this phish was reported to Phishtank
    * **verified:** Whether or not this phish has been verified by PhishTank's community. All of them are 'yes' as they only provide the verified ones in their downloadable datasets
    * **verification_time:** The date and time at which the phish was verified as valid
    * **online:** Whether or not the phish is online and operational. Similar to verified, this is also always 'yes'.
    * **target:** The name of the company or brand the phish is impersonating, if it's known.

- **URL dataset (ISCX-URL2016)** [11]
  - It is a source that has a collection of benign, spam, phishing, malware & defacement URLs.
  - The number of legitimate URLs in this collection are 35,300, from which again we took 10,000 samples as legitimate links.
  - Link: *https://www.unb.ca/cic/datasets/url-2016.html*
  - The legitimate URLs dataset consisted only of the URLs and no additional attributes.

Thus, we ended up with a total of 20,000 links (10,000 each of legitimate and phishing URLs) which we'll use to train our model for detecting phishing links. Combining the two datasets, we used only the URL attribute in them and a label stating whether they're phishing (1) or legitimate (0).

After deriving the relevant features from these URLs, we finally arrived at the URLGuardian dataset having 20,000 records and 18 features (*https://docs.google.com/spreadsheets/d/1879mIZgiZWACMY6qY9tVTVF5fjH1SAy2cgUb8hP23KY/edit?usp=sharing*). We tried out different splits (Section 6.2) and finally went with the 80:20 split:

- **Training Set:** 16,000 records and 17 features (url_training)
  Link: *https://docs.google.com/spreadsheets/d/13PqxfS1MbAHSjqnzLwG_DKIGEn-120IRDkUTx1zm8zo/editgid=1533877409*

- **Testing Set:** 4000 records and 16 features (url_testing)
  Link: *https://docs.google.com/spreadsheets/d/1w6xUX5dBKbIyEfMo-0vXsWuI_S99huXD30LKuF-zRHg/editgid=640769151*

We removed only the 'domain' feature from our dataset for the splitting, as it wasn't needed for the training and testing purposes. [2]

4.1.3 Preparing Image Dataset

To train a model to detect AI generated images, we needed a dataset where we could get images that were actually real, along with images that were synthetically generated. Thus, we began searching for such datasets, where we could find sufficient quantities of images which could help us achieve satisfactory results. We came across:

- **TWIGMA: A dataset of AI-Generated Images with Metadata From Twitter:** [12]
  - This dataset consists of 800,000 AI generated images collected from Jan 2021 to March 2023 on Twitter with associated metadata (like tweet text, creation date, number of likes, etc).

  - However, there was no link provided where we could access the images used in this dataset. We even tried to get in touch with the authors of the paper themselves, but even they could not provide us with the actual dataset to work with.
  - So, even though this dataset would have provided us vast quantities of synthetic images, we'd still need to curate real images nevertheless. Thus, we decided to pursue other options.
  - Link: *https://arxiv.org/pdf/2306.08310*

- **DigiFace-1M Dataset** [3]
  - As the name suggests, this dataset consists of 1 million digitally generated face images which are originally meant for facial recognition.
  - To keep our image classifier work with a diverse set of data, we planned on collecting some facial samples from this dataset and combining it with other types of images we might find in some other datasets.
  - This was as a last resort though, since finding a complete dataset was always our preferred approach rather than trying to curate a new one, which even though provides customization, but has its own set of complications.
  - Link: *https://github.com/microsoft/DigiFace1M*

- **CIFAKE: Real and AI-Generated Synthetic Images** [8]
  - After going through a couple of datasets mentioned above, we finally decided to go with *CIFAKE*.
  - Here we got access to 1,20,000 images, consisting of both real and fake images, which solved the problem of *TWIGMA* where firstly, we did not have access to the images and secondly, lacked real images. [10] [4]
  - To make things easier for us, the dataset was already split into training and testing.
    * **Training:** 1,00,000 (50,000 each of real and fake images)

    * **Testing:** 20,000 (10,000 each of real and fake images)
  - Link to dataset: *https://www.kaggle.com/datasets/birdy654/cifake-real-and-ai-generated-synthetic-images*
  - Link to paper: *https://arxiv.org/pdf/2303.14126v1*

4.1.4 Semantic Classifier

Given the vast scope of our project, we are already working with multiple classifiers at this point. Introducing an additional model and training would complicate things unnecessarily, but as we want the best results for our users, we did not want to compromise the quality of our detection algorithms.
Thus, considering all options, we decided not to use a separate database to train our semantics classifier on, rather we will use simple logic with multiple checks to detect the type of fake news we might encounter.
There is no doubt that using a dedicated dataset would have been the better approach, but that would add additional complexity into this project (like training another model, testing it out with satisfactory results, fixing issues of overfitting on the provided dataset, etc), which at this point we could not afford. Keeping all this in mind, we decided not to use a dataset for this classifier. The working for the same, is explained in section

4.2 Data Set Annotation

4.2.1 TweetaVerse

The attribute names and their descriptions for the `TweetaVerse dataset` are as follows:

Table 2: TweetaVerse Annotation

Attribute	**Description**
ID	The unique ID of a tweet record
Target Value	The binary target label (true/false) which indicates whether the tweet body is true or false wrt. the headline present.
Headline	The headline of a news article on which a tweet is based upon.
Author	The author of the news article on which the heading is based upon.
Tweet Body	The actual content of the tweet including the text and links (if any).
User_Followers_Count (UFC)	The follower count of the user who has posted the tweet
User_Friends_Count (UFRC)	The friend count of the user who has posted the tweet. The difference between a friend(my following) and a follower(my followers) is (assuming that I'm the user we're talking about) that the former is someone who I follow, whereas the latter is someone who is following me.
User_Favourites_Count (UFVC)	TThe total number of times the tweets tweeted by a user has been marked as favorite. For example: If I've a total of 10 tweets and each of my tweets have been marked as favorites 5 times, then my UFVC = 10 x 5 = 50.
User_Tweets_Count (UTC)	The count of the total number of tweets posted by the user who has tweeted the current tweet.
User_Lists_Count (ULC)	Lists are similar to pages on Facebook. This is the total no. of pages that the user who has tweeted the current tweet, is a part of.
User_Mentions_Count (UMC)	The total number of times the user who has tweeted the current tweet has been mentioned (by tagging him/her) by other users.
User_Quotes_Count (UQC)	The total number of times the user who has tweeted the current tweet has been "quote tweeted" by other users. For example, suppose I tweeted: "Messi didn't win the world cup". Now some other user can repost my tweet adding his/her own comment/thought to it saying: People are believing "Messi didn't win the world cup". Hence the other user quoted my tweet but didn't mention me in their tweet, thus it wouldn't have been counted in UMC.

Attribute	**Description**
Tweet_Replies_Count (TRC)	The count of the replies the current tweet has received.
Tweet_Retweets_Count (TRTC)	The count of the retweets the current tweet has received.
Tweet_Favourites_Count (TFVC)	The number of times the current tweet has been marked as favorite. This is different from UFVC as that is considering the TFVC of all the tweets of that user.
Tweet_Hashtags_Count (THC)	The count of the hashtags the current tweet has.
5_Label_Target_Value (TV5)	A 5-label target value which shows how similar the current tweet is to the "Headine" that we have. The likeness to the heading has been measured as: Agree, Mostly Agree, No Majority, Mostly Disagree, Disagree.
Tweet_Category	The category to which the "Headline" belongs to from the following list: ["politics", "world", "finance", "technology", "sports", "entertainment", "health", "education", "crime", "business", "social"]
Tweet_URL_Count (TUC)	The number of URLs the tweet body contains.
Tweet_URLs_List	The list of the URLs mentioned in the tweet body which will later be used for clickbait detection.
Bot_Score	A value between 0 and 1 which indicates whether the user who has tweeted the current tweet is a bot or not, where 1 indicates the user being a bot.
Cred_Score	A value between 0 and 1 which indicates the credibility of the user as to whether the information shared by them can be treated as credible or not. More the value of cred_score being close to 1, more the user's credibility value.
User_Activity_Score (UAS)	The average activity score of a user. In simple terms it shows the average activity a particular user has on Twitter. For example: If I've posted around 10 tweets in my lifetime and my tweets have gotten an average of 15 replies, 17 favorites, 90 mentions, etc so based on such attributes I'll be able to figure out the average engagement a tweet tweeted by me would experience.

Attribute	Description
Tweet_Activity_Score (TAS)	The average activity score of a particular tweet. It is similar to UAS, only that it shows the average activity of a tweet. For example: If I've posted a new tweet and it got around 5 retweets, 8 favorites, 2 hashtags, etc so based on such attributes I'll be able to figure out the average engagement this particular tweet has experienced
UAS_Normalized	The UAS value scaled down using the Min-Max scalar algorithm.
TAS_Normalized	The TAS value scaled down using the Min-Max scalar algorithm.
Exclusivity	Based on the UAS and TAS, we say whether the current tweet's engagement is matching the user's average tweet engagement (1) or not (0).

4.2.2 URLGuardian

The attribute names and their descriptions for the `URLGuardian Dataset` are as follows [11]:

Table 3: URLGuardian Annotation

Attribute	**Description**
Domain	The domain of the URL.
Have_IP	Binary value indicating if IP address of the link is present in the URL(1) or no (0).
Have_At	Binary value for the presence of @ (1) in the link or not (0).
URL_Length	Binary value indicating whether the length of the URL is greater than or equal to 54 characters (1) or not(0).
URL_Depth	Count of the number of sub pages in the given URL based on '/'.
Redirection	Binary value for the presence of redirection to another website (1) or not (0). It uses the presence and position of '//' in the URL to arrive at the final value.
https_Domain	Presence of 'https' in the domain (1) or not (0).
Tiny_URL	If the URL is using Shortening Services, the value assigned is 1 else 0.
Prefix/Suffix	Presence of '-' in the URL (1) or not.
DNS_Record	If the DNS record for the URL is empty or not found, then the value is assigned as 1 else 0.
Web_Traffic	It measures the popularity of the website using its global rank. If the rank is more than 100000, then the website is declared as popular (1), else not so popular (0).
Domain_Age	If the age of the domain is less than 12 months, value is assigned as 1, else 0,
Domain_End	If the end period of domain is less than 6 months, value is assigned as 1, else 0.
IFrame	If the iframe is empty or response is not found then, value is assigned as 1, else 0.
Mouse_Over	If the response is empty or on mouse-over is found then, value is assigned as 1, else 0.
Web_Forwards	If the website is redirected more than 3 times, value is assigned as 1, else 0.
Right_Click	Indicates whether the Right-Click function is disabled on a web page or not (1 -disabled, 0-enabled)
Label	Target value indicating whether the URL is a phishing link (1) or not (0).

4.2.3 CIFAKE Dataset

The image dataset does not have any features as such, but a collection of a large number of images - both real and fake. When feeding this dataset to a network model, it will extract it's own features internally.

4.3 Data Set Characteristics

4.3.1 TweetaVerse Dataset

The User_Activity_Score (UAS) measures the average engagement the user's usual tweets receive. To calculate the same, we've considered factors which depend on the user's connections with other users, hence affecting their engagement. [7]

Thus, the factors affecting UAS from *TweetaVerse Dataset* are:

- **User_Followers_Count (UFC)**
  A follower on Twitter is someone who follows us: If User X follows me, then User X is my follower. The more followers a particular user has, more will be his reach and hence more people will engage with the tweets posted by them. For example, User A has 100 followers and User B has 7000 followers. Now a tweet tweeted by User B will obviously be seen by more users compared to User A as more users follow User B. Thus, UFC is a deciding factor in calculating the user's average activity score.

- **User_Friends_Count (UFRC)**
  A friend on Twitter is someone I follow, i.e. my following: If I follow User X, then User X will be my friend. If User X also follows me, then User X will also be my follower, and I'll be a friend to User X. Taking into consideration the fact that bot accounts (accounts whose main aim is to spread false information) tend to have very few following. Hence UFRC has been included as a factor affecting UAS.

- **User_Favourites_Count (UFVC)**
  Users have the ability to mark a tweet as 'favorite' for referring to it later on. It might be due to several reasons- maybe they liked the tweet very much or the information seemed suspicious, or some other factor. Thus, if a tweet is being marked as favorite it implies that it sparked some interest to a user. UFVC

is the count of the total number of times a particular user's tweet has been marked as favorite in their lifetime. This helps to take into consideration how many times any of the user's tweets is marked as 'favorite'.

- **User_Tweets_Count (UTC)**
  UTC is the total number of tweets tweeted by a particular user. In order to know about the activity of the user, we'd have to know about the total tweet count the user has in order to understand their history.

- **User_Lists_Count (ULC)**
  Lists are similar to 'Pages' on Facebook, where multiple people can post stuff. Thus, by knowing how many lists a particular user is part of, we can say how many users they're trying to reach. Statistically, bot accounts tend to be a part of multiple lists, so that they can spread misinformation as fast as possible. Hence, ULC has been used in calculating UAS.

- **User_Mentions_Count (UMC)**
  Mentions are just like tagging (using '@') someone while writing a tweet. The more mentions a particular user gets, the more engagement their other tweets receive as well. If a user has multiple mentions and the tweet posted by them is still live, it usually suggests that the information shared by them is legit. Thus, UMC is the total number of mentions the current user has received in their lifetime.

- **User_Quotes_Count (UQC)**
  Quoting is a way of reposting a tweet, but after adding our own thoughts or comments to it. For example, If I tweeted saying: *"I'm not feeling well, so I decided to skip college today."*

  Now, if some other user decides to say something about my tweet but without directly engaging with me, he can "quote tweet" me and add what's on their mind, like: *"Students these days will do anything to not go to school."* (followed by my original tweet).

  It's not the same as retweeting, as there we cannot add anything to the original tweet. So, we can see that the user is spreading the text that I wrote, but without directly mentioning me.Thus, the total number of times a user has

been "quote tweeted" is given by UQC. It is slightly different from UMC, but has similar influence on the UAS.

After gathering the factors that affect the UAS, we've considered a linear relationship amongst these factors to calculate the average activity score of the user. [1]

Thus,

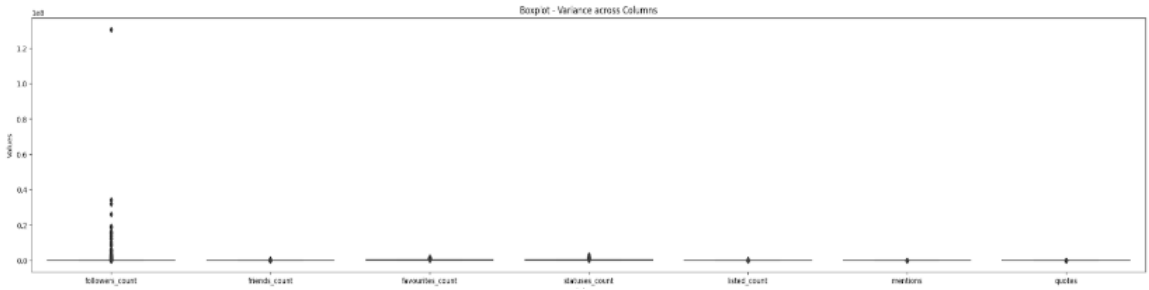
$$UAS = (w1 * UFC) + (w2 * UFRC) + (w3 * UFVC) + (w4 * ULC)$$
$$+(w5 * UTC) + (w6 * UMC) + (w7 * UQC)$$

Where, w1, w2, . . . , w7 are the weights indicating the importance for their respective factors in calculating the UAS. Now, we just need to figure out the values of these weights to find the UAS for each record in our `'TweetaVerse Dataset'`. Brainstorming on this problem, led us through the following chain of thoughts:

- We could've started off with random assignment and then used a ML model to backprop and adjust the weights, but there's a slight problem in that approach. Usually when we adjust weights like this, we already have the output variable calculated (UAS, in our case), but as we're trying to find the output variable itself, using a model to find the weights was not possible.

- Next up, we decided to examine the distribution of the features in our `Tweetaverse Dataset`. We looked at the correlation amongst these features, their frequency distribution as well as their variance.

- We focused on the variations of these factors. Variance is the amount of information a particular feature is providing. Thus, features having more variance are more important than the others. However, we needed to look at the variance graphs ignoring the outliers, as they can cause huge spikes which can alter the readings. Below is the variance plot of the above mentioned factors.



- Upon examining the variance values numerically, we got:

UFC	0.9484
UTC	0.0297
UFVC	0.0234
UFRC	0.0002
ULC	0.0000
UQC	0.0000
UCM	0.0000

  Thus, we can see that UFC would be the most influential factor in UAS as it constitutes the most information related to the user whereas ULC, UQC and UMC are on the lower end of the importance spectrum.

- Hence, considering the variance of the factors, we assigned higher weights to the more important features and lesser to the rest. We then estimated the acceptable values of these weights through a series of trial and error tests and ended up with the following weights distribution:
  - w1 = 0.8
  - w2 = 0.04
  - w3 = 0.05
  - w4 = 0.02
  - w5 = 0.05
  - w6 = 0.02
  - w7 = 0.02

The **Tweet_Activity_Score (TAS)** measures the average engagement a particular tweet is receiving. To calculate the same, similar to UAS, we've considered factors which contribute to a tweet's reach to other users on the platform, namely how many times it has been retweeted, how many replies it has got or has been marked as 'favorite', etc.

Thus, the factors affecting TAS from `TweetaVerse Dataset` are:

- **Tweet_Retweets_Count (TRTC)**
  Retweeting is just sharing a tweet which has been tweeted by someone else. As a tweet is shared, more and more people are going to read it, engage with it and maybe even share it further, thus spreading the piece of 'information' associated with it. TRTC is the total number of times a particular tweet has been retweeted in its lifetime.

- **Tweet_Favorites_Count (TFVC)**
  Unlike UFVC, TFVC is the number of times a particular tweet has been marked as 'favorite'. This helps in identifying whether the current relation with the user's average tweet activity. For example: If usually each of my tweets are marked by 10 users as 'favorite', then that measure would be available through UAS. Now suppose, I tweeted a new tweet, and this was marked as favorite by 100 users. Thus, this particular tweet's TFVC will be 100, and it'll thus show that this tweet is more popular/engaging than the rest of the tweets that I usually tweet.

- **Tweet_Replies_Count (TRC)**
  The more replies a particular tweet gets, the more the platform pushes it on the explore page and thus increases its reach. TRC is the total number of replies a particular tweet has received which will help in calculating the average engagement it has experienced.

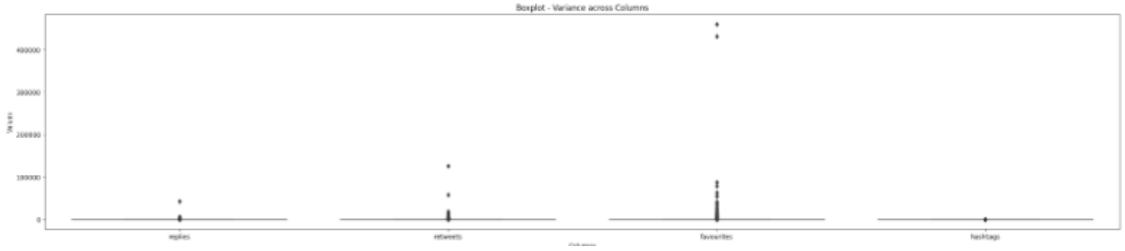
- **Tweet_Hashtags_Count (THC)**
  Hashtags are a very common way to increase the reach of a tweet. We can always see the trending hashtags on Twitter, and if users include them in their tweets, they are more likely to end up on someone's feed compared to without those hashtags. However the same can be used to easily spread false information as well. Users can include multiple hashtags so that the tweet can reach to maximum feeds possible. Thus, THC is the count of hashtags used in a particular tweet.

After gathering the factors that affect the TAS, similar to UAS, we've considered a linear relationship amongst these factors to calculate the average activity score of

a tweet. Thus,

$$TAS = (w1 * TRTC) + (w2 * TFVC) + (w3 * TRC) + (w4 * THC)$$

Where, w1, w2, w3, w4 are the weights indicating the importance for their respective factors in calculating the TAS. Similar to the calculation of UAS, we used the same approach of examining the data that we had in `TweetaVerse Dataset` to calculate the above weights.



Again taking learnings from the weights selection from UAS and seeing the above variance distribution for the Tweet related factors

- Even after ignoring the outliers present in TFVC, it still has more information compared to TRTC. Hence, TFVC would be the most influencing factor.
- TRC and THC have almost similar variations indicating they provide a similar level of information for the calculation of TAS, but significantly lower compared to TFVC and TRTC.
- Following the same method of trial and error, we ended up with the following weights distribution:
  - w1 = 0.15
  - w2 = 0.8
  - w3 = 0.049
  - w4 = 0.001
- w3 (for TRC) has been kept higher compared to w4 (for THC) as in our dataset, the average value of THC is significantly lower than TRC. Thus, to avoid overfitting, we gave comparatively higher weightage to TRC wrt THC.

Now that we've understood the meaning as well as done with the calculations of UAS and TAS, we'll look at **Exclusivity** in the following sections.

4.3.2 URLGuardian Dataset

In `URLGuardian Dataset` , we have considered these attributes to effectively classify whether the link is a phishing link or not:

- **Domain**
  The domain of a URL, which stands for Uniform Resource Locator, refers to the section of a web address that indicates the specific location of a resource on the Internet. Typically situated after the "https://" or "http://" protocol prefix, the domain is a human-readable representation of the IP address assigned to the server hosting the resource. Here, we are just extracting the domain present in the URL. This feature doesn't have much significance in the training. May even be dropped while training the model.

- **Have_IP**
  Indicates whether a URL contains an IP address or not. A URL is often associated with an IP address, URLs themselves typically don't contain the IP address directly. If an IP address is used as an alternative of the domain name in the URL, we can be sure that someone is trying to steal personal information with this URL.

  If the domain part of the URL has an IP address, the value of this feature is 1 (Phishing) or else it is 0 (Legitimate).

- **Have_At**
  Indicates whether a URL contains a '@' symbol or not. The '@' symbol causes the browser to ignore everything preceding it and the actual real address usually follows the symbol.

  The presence of the '@' symbol in the URL definitely raises suspicion over the legitimacy of the URL as the part preceding the symbol (the part which is ignored) can be a trustworthy site, say, "google.com" whereas the part following the symbol can be the address to a malicious site.

  If the domain part of the URL has the '@' symbol, the value of this feature is 1 (Phishing) or else it is 0 (Legitimate).

- **URL_Length**
  Indicates whether the URL is lengthy or not. Phishers can use very long URLs to conceal suspicious parts and thus make the URL look trustworthy

  If the length of the URL is greater than or equal to 54 then the value of this feature is 0 (Phishing), or else it is 1 (Legitimate).

- **URL_Depth**
  Indicates the depth of the URL. The term 'depth' commonly pertains to the level or hierarchy of a particular page or resource within the structure of a website. It serves as a gauge of the distance, measured in clicks or steps, that a specific page is from the website's homepage or root.

  Very deep URLs raise suspicion over legitimacy as we might get directed to an unwanted subpage from the homepage of the website.

- **Redirection**
  Indicates the presence of "//" in the URL.

  The "//" is used for redirection which means that the user will be sent to another website. This is definitely suspicious as the URL might be of a trustworthy website but on clicking the URL it might redirect to a malicious website.

  The location of the "//" in the URL is computed. We find that if the URL starts with "HTTP", that means the "//" should appear in the sixth position. However, if the URL employs "HTTPS" then the "//" should appear in seventh position.

  If the "//" is anywhere in the URL apart from after the protocol, the value of this feature is 1 (Phishing) or else 0 (Legitimate).

- **https_Domain**
  Indicates the presence of "http/https" in the domain part of the URL.

  HTTP stands for Hypertext Transfer Protocol and HTTPS stands for Hypertext Transfer Protocol Secure. Website owners need to obtain an SSL/TLS certificate in order to use HTTPS for their websites. HTTPS hence stands as an assurance for the user that the website they are trying to access is safe. Phishers try to counter this check by including https in the domain part of the

URL to trick an unsuspecting user into thinking that the website uses HTTPS protocol.

If the URL has "http/https" in the domain part, the value of this feature is 1 (Phishing) or else 0 (Legitimate).

- **TinyURL**
  Indicates the usage of URL shortening services. URL shortening is a method on the "World Wide Web" in which a URL may be made considerably smaller in length and still lead to the required webpage. This is accomplished by means of an "HTTP Redirect" on a domain name that is short, which links to the webpage that has a long URL. Phishers might use URL shortening services to completely remove suspicious parts from their URLs. If the URL is using Shortening Services, the value of this feature is 1 (Phishing) or else 0 (Legitimate).

- **Prefix/Suffix**
  Indicates the presence of '-' in the domain part of the URL. The '-' symbol is rarely used in legitimate URLs. Phishers tend to add prefixes or suffixes separated by (-) to the domain name so that users feel that they are dealing with a legitimate website, especially in cases where the website name is the same as that of the company name which has multiple words in it. Example : www.etsindia.org can faked by a phisher who could use the URL www.ets-india.org

  If the URL has '-' symbol in the domain part of it, the value of this feature is 1 (Phishing) or else 0 (Legitimate).

- **DNS_Record**
  Indicates whether the asserted website identity from the URL is present in the WHOIS database or if an appropriate DNS record for it exists.

  DNS records are information entries housed within authoritative DNS servers, furnishing details about a domain. This information encompasses particulars like the associated IP address or the mail servers responsible for handling emails for the domain.

  The WHOIS database functions as a publicly accessible repository containing details regarding registered domain names and their proprietors. It acts as a

resource for retrieving information about domain ownership, registration, and availability.

For phishing websites, either the claimed identity is not recognized by the WHOIS database or no records found for the hostname. If the DNS record is empty or not found then, the value of this feature is 1 (Phishing) or else 0 (Legitimate).

- **Web_Traffic**
  Indicates whether the URL website receives a minimum amount of traffic or not.

  This feature gauges website popularity by assessing both visitor numbers and the quantity of pages visited. Nevertheless, due to the brief existence of phishing websites, they might go unnoticed by the Alexa database (Alexa, the Web Information Company, 1996). Upon analyzing our dataset, we observe that, at times, authentic websites can even rank within the top 100,000. Moreover, a domain is categorized as "Phishing" if it either lacks traffic or is not acknowledged by the Alexa database.

  If the rank of the domain is more than 100000, the value of this feature is 1 (Phishing) else 0 (Legitimate).

- **Domain_Age**
  Indicates whether the age of the domain is long enough.

  Age here is nothing but the difference between creation and expiration time of the website. This feature can be extracted from the WHOIS database. Most phishing websites live for a short period of time thus the minimum age of the legitimate domain is considered to be 12 months for this project.

  If age of domain is less than 12 months, the value of this feature is 1 (Phishing) else 0 (Legitimate).

- **Domain_End**
  Indicates whether the period of time from current time to the domain expiration time is large enough.

This feature can be extracted from the WHOIS database. For this feature, the remaining domain time is calculated by finding the difference between expiration time & current time. Legitimate websites generally renew their domain registration before 6 months to avoid any potential interruptions in their services.

If the end period of the domain is less than 6 months, the value of this feature is 1 (Phishing) else 0 (Legitimate).

- **iFrame**
  Indicates the usage of hidden iFrames.

  IFrame is an HTML tag used to display an additional webpage into one that is currently shown. Phishers can make use of the "iframe" tag and make it invisible i.e. without frame borders. In this regard, phishers make use of the "frameBorder" attribute which causes the browser to render a visual delineation.

  If the iframe is empty or response is not found then, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

- **Mouse_Over**
  Indicates whether the Javascript event "onMouseOver" response is modified or not.

  Phishers might employ JavaScript to display a deceptive URL in the status bar, misleading users about the actual destination of a link. To identify this characteristic, we need to examine the webpage's source code, specifically focusing on the "onMouseOver" event. This event is triggered when a user hovers over a link with their mouse.

  In the process of feature extraction, we investigate whether the "onMouseOver" event in the source code results in any modifications to the status bar. If the response is empty or if the "onmouseover" event is detected, it indicates a potential attempt to manipulate the displayed URL. This analysis helps in discerning phishing attempts that utilize JavaScript to manipulate the status bar, offering a mechanism to identify and differentiate potentially deceptive URLs from legitimate ones.

  If the response is empty or onMouseOver is not found then, the value assigned to this feature is 1 (Phishing) or else 0 (Legitimate).

- **Right_Click**

  Indicates whether the Right Click function is disabled on a webpage or not.

  Phishers often employ JavaScript to deactivate the right-click function on websites, preventing users from viewing and saving the webpage source code. This behavior is analogous to the approach of "Using onMouseOver to hide the Link." However, specifically for this feature, our examination involves searching for the presence of the "event.button==2" event within the webpage's source code to determine if right-click functionality has been disabled.

  To extract this feature, we analyze the webpage's source code and inspect instances where the "event.button==2" event is utilized. This event corresponds to a user attempting to right-click on the webpage. If the source code reveals that this event is employed to hinder the right-click function, it indicates an effort to restrict users from accessing critical functionality.

  This analysis aids in identifying phishing attempts that utilize JavaScript to impede users from accessing the right-click context menu, contributing to the recognition of potentially deceptive webpages.

  If Right Click is disabled then the value assigned to this feature is 1 (Phishing) or else 0 (Legitimate).

- **Web_Forwards**

  Indicates whether the website has been redirected more than or equal to 3 times or not.

  The subtle distinction between phishing websites and legitimate ones lies in the frequency of website redirections. Upon analysis of our dataset, it is observed that legitimate websites typically undergo a maximum of one redirection. In contrast, phishing websites, characterized by this specific feature, exhibit a pattern of being redirected at least 4 times.

  This feature becomes a key discriminator in our assessment. Legitimate websites, in adhering to standard practices, tend to have minimal redirections, ensuring a straightforward and secure user experience. In contrast, the elevated count of at least four redirections in the case of phishing websites suggests a potentially malicious intent, as excessive redirections can be indicative of attempts to obfuscate the true nature of the website or engage in deceptive practices.

If the number of redirects is greater than or equal to 4 then the value assigned to this feature is 1 (Phishing) or else 0 (Legitimate).

- **Label**
  Indicates whether the URL is a phishing link or not. The dataset used is a labeled dataset and this is the label which indicates whether the URL is a phishing link or not.

  If the label has value 1 then the URL is a Phishing Link else if the value is 0 then the URL is a legitimate URL.

4.3.3 CIFAKE Dataset

Like we mentioned before, the image dataset does not have any user supplied features - the features will be internally extracted by our deep learning models. This has been elaborated in the later sections.

We have a dataset containing **60,000 each** of real and fake images. This is a direct input to our models.

4.4 Data Set Analysis

4.4.1 TweetaVerse Dataset

Each property and their respective counts are given in the table:

	TweetaVerse Dataset
Followers	1515446387
Friends	254053386
Favorites	4425917915
Tweets	4588528087
Lists	9835397
Mentions	186384
Quotes	76952
Replies	256882
Retweets	895678
Tweets marked as favorites	3700159
Hashtags	14052
Tweet URLs	79

	TweetaVerse Dataset
Labels	5 [Agree, Mostly Agree, No Majority, Mostly Disagree, Disagree]
Tweet categories	11 [”politics”, ”world”, ”finance”, ”technology”, ”sports”, ”entertainment”, ”health”, ”education”, ”crime”, ”business”, “social”]

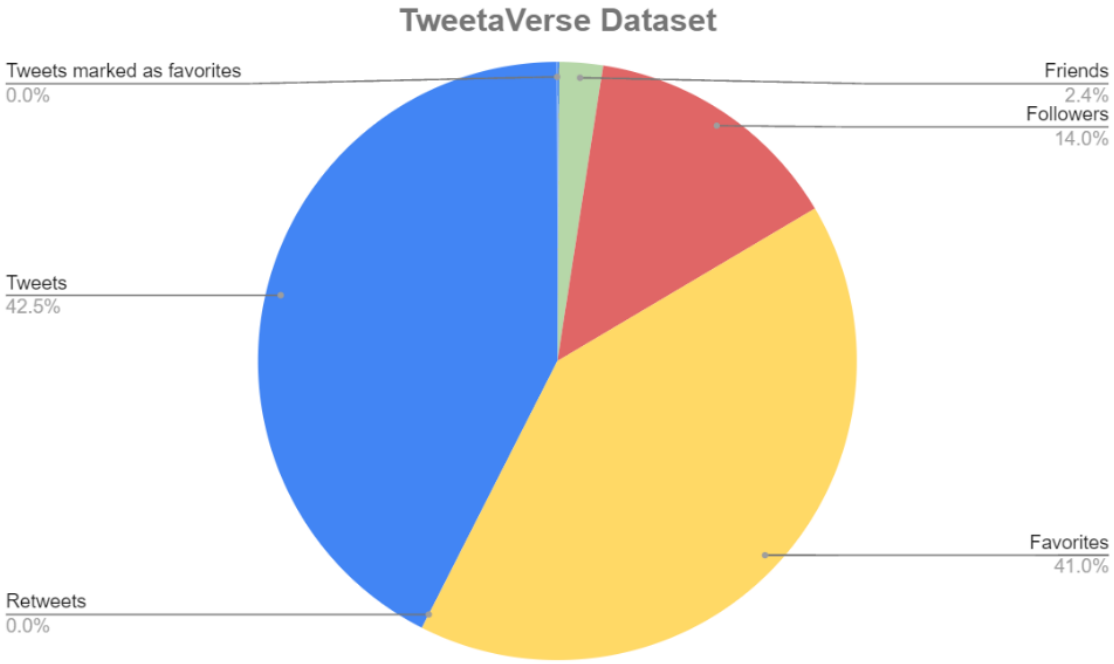


Figure 1: Analysis of TweetaVerse Dataset

4.4.2 URLGuardian Dataset

Each property and their respective counts are given in the table:

	URLGuardian Dataset
Links containing IP Address in the URL	0
Links containing '@' in the URL	176
Links exceeding the length of 54 characters	14685
Links having subpages	15173
Redirecting links	255
Links having 'https' in their domain	2
Shortened links	1071
Links containing a prefix/suffix	4292
Links with no DNS record	3781
Links experiencing large traffic	17337
Links having age greater than a year	10032
Links having an ending runway of more than 6 months	16539
Links not having an iFrame	6811
Links supporting Mouse-Over	6759
Links having right click disabled	19989
Links redirecting more than 3 times	7427
Labels	2 [1 = phishing, 0= not phishing]

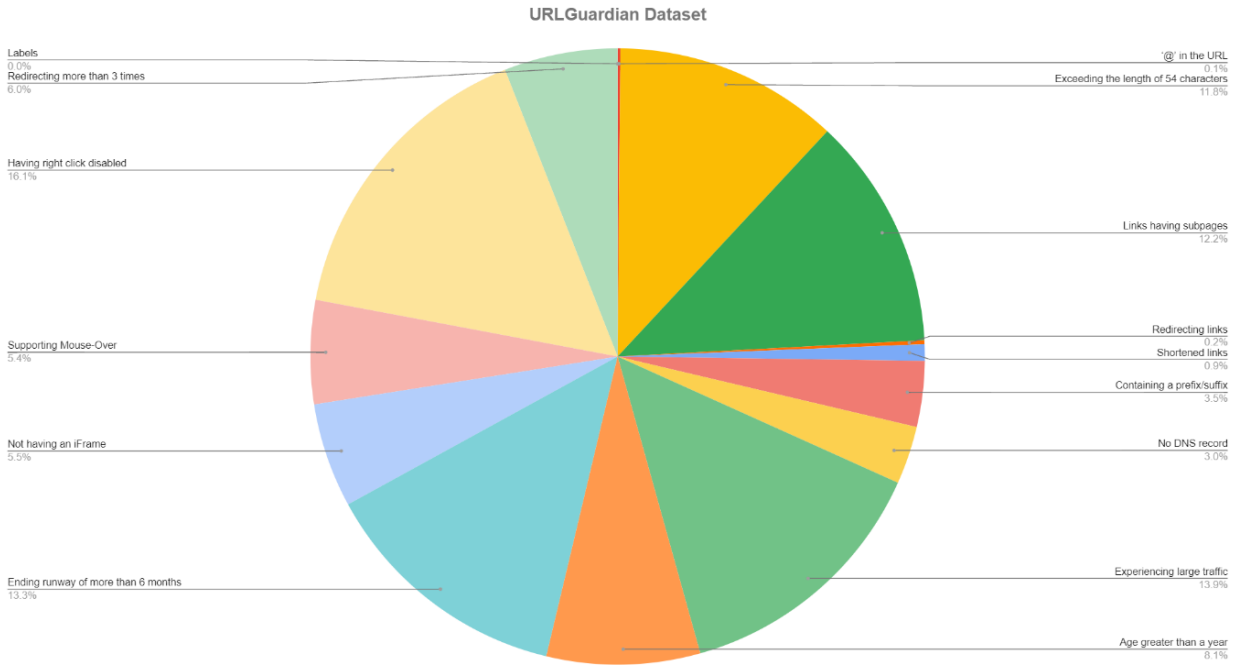


Figure 2: Analysis of URLGuardian Dataset

4.4.3 CIFAKE Dataset

Each property and their respective counts are given in the table:

	C**IFAKE Dataset**
Real images in training Set	50000
Fake images in training set	50000
Real images in testing set	10000
Fake images in testing set	10000

5 Proposed System

5.1 Data Cleaning and Preprocessing

5.1.1 TweetaVerse Dataset

- We decided to proceed with the TruthSeeker dataset as the main foundation dataset of our model. However, TruthSeeker had many additional features which we didn't require and thus, decided to drop them off. (Details in Section 4.1.1)
- From the given dataset, we were able to procure the category of the headline as well using Zero Shot Classification as one of the following: ["politics", "world", "finance", "technology", "sports", "entertainment", "health", "education", "crime", "business", "social"].
- Based on the Bot Score and Cred Score provided by TruthSeeker, we calculated them ourselves using the formulas (explained in section 6.1). We also added an additional feature called 'Exclusivity' (also explained in section 6.1). After adding all the necessary features required to calculate the Bot_Score, Cred_Score and Exclusivity, we ended up with a total of 27 features and 134194 records.
- We proceeded with a 80-20 split on the dataset for training and testing purposes.

5.1.2 URLGuardian Dataset

- We ended up with a total of 20,000 links (10,000 from PhishTank dataset and 10,000 from ISCX-URL2016 dataset)

- We used only the URL attribute in them and a label stating whether they're phishing (1) or legitimate (0).
- Then we derived the relevant features from the URL's which are explained in Section 4.2.
- During preprocessing, we dropped off the '*domain*' feature as it didn't play a part in the training process. The values were non null as that was ensured when we were extracting the features for our dataset.
- Further, we shuffled the dataset as during dataset collection the first 10,000 records were legitimate and the next 10,000 were phishing.
- We proceeded with a 80-20 split on the dataset for training and testing purposes.

5.1.3 CIFAKE Dataset

- As explained before in Section 4.1, we took the already available CIFAKE Dataset. Here we got access to 1,20,000 images, consisting of both real and fake (both equally divided).
- For our neural network models, only images were sufficient inputs, we did not require any other external features.
- The dataset was already split into training and testing datasets which added convenience.

5.2 Proposed Model

5.2.1 Tweet Classifier Model

In this classifier we have considered 5 main factors which would determine the outcome of the result in question: [*TweetBody* (vectorized), *5_Label_Target_Value*, *Bot_Score*, *Cred_Score*, *Exclusivity*]. The potential output labels were mapped against the target labels of: [*True* (1), *False* (0)] indicating whether the tweet is sharing some fake news (True) or not (False).
Since in our dataset, we had both numerical and textual data, we did the following:

- Combining all the numerical data together.

- Then, vectorizing the textual data into numerical form
- In the end combining both sets of numerical data into one, to be fed into the model for training.

As this was a classification stage, we used different models that were most suited for this job, namely: **Logistic Regression**, **Random Forest Classifier**, **SVM** and **XGBoost**.

- **Logistic Regression**
  In Logistic Regression, we plotted the inputs against the output variables in place and used the regression logic to find out the best possible regression boundary that could fit this dataset. In each iteration the model adapted much better to our dataset and the final results could be looked upon in the results comparison section. This approach was relatively simpler compared to other models we considered, and we can confirm so by looking at the results.
- **Random Forest Classifier**
  Here, we generated random decision trees on all the input parameters and analyzed each of them. We then proceeded by taking the majority voting from the combined analysis and choosing the best possible result for the classification.
- **Support Vector Machine (SVM)**
  The key component in SVM is the generation of the hyperplane, and so, to represent multiple input parameters we had to use kernel tricks to bring down our model into a lower dimension for ease of generation. The best suited hyperplane was then able to differentiate the data points as true or false.
- **XGBoost**
  This approach works on the basis of Gradient Boosting. During training, the model learns from the input features and corresponding target values. The boosting process involves sequentially adding models to correct errors made by existing models. This method combines the predictions from multiple decision trees. It builds each tree sequentially, where each new tree aims to correct the errors made by the previous ones. The trees are typically shallow, and each split is chosen to maximize a certain objective function, often related to reducing the training error. It employs gradient boosting, which means it optimizes the loss

function of the previous model using gradient descent, effectively minimizing the errors.

5.2.2 Links Classifier Model

In case of the Links Classifier, we have considered all the attributes of the URL-Guardian Dataset (Section 4.2) as the input parameters, except for the Label which would be our output parameter as: [*Phishing* (1), *Not Phishing* (0)].
This classifier deals with numerical data only and hence the training process was simpler compared to the Tweets Classifier. As this was also a classification stage, we used the following models: **Decision Tree**, **Random Forest Classifier**, **SVM**, **XGBoost** and **Multilayer Perceptron**.

- **Decision Tree**
  Using the decision tree approach, we essentially built just one tree. This single decision tree was constructed by recursively splitting the data based on the input features to minimize impurity (like Gini impurity or entropy) at each node until certain stopping criteria are met (like maximum depth of the tree, minimum number of samples per leaf node, etc). The optimization process here focuses solely on finding the best splits for this single tree.

- **Random Forest Classifier**
  Instead of relying on just a single decision tree, here we generated multiple random ones based on a random subset of the training data (sampled with replacement) and a random subset of the input features. This introduces variability and diversity among the trees. During prediction, each tree in the random forest independently makes a prediction, and then the final prediction is determined by aggregating the predictions of all trees. This ensemble approach of building multiple trees and aggregating their predictions helps to reduce overfitting and increase the robustness and generalization performance of the model compared to a single decision tree.

- **Support Vector Machine (SVM)**
  A similar process as that of the Tweets Classifier was implemented, with the input and output labels being different.

- **XGBoost**
  Once again, a comparable approach like the one in the Tweets Classifier with the set of inputs and outputs being different.

- **Multilayer Perceptron**
  This is a simple neural network consisting of the input, hidden and an output layer. The input layer took in the entire set of input parameters which were worked upon by the neurons present in the hidden layer to assign weights and then mapped them to their corresponding outputs using the softmax output layer.

Workflow:

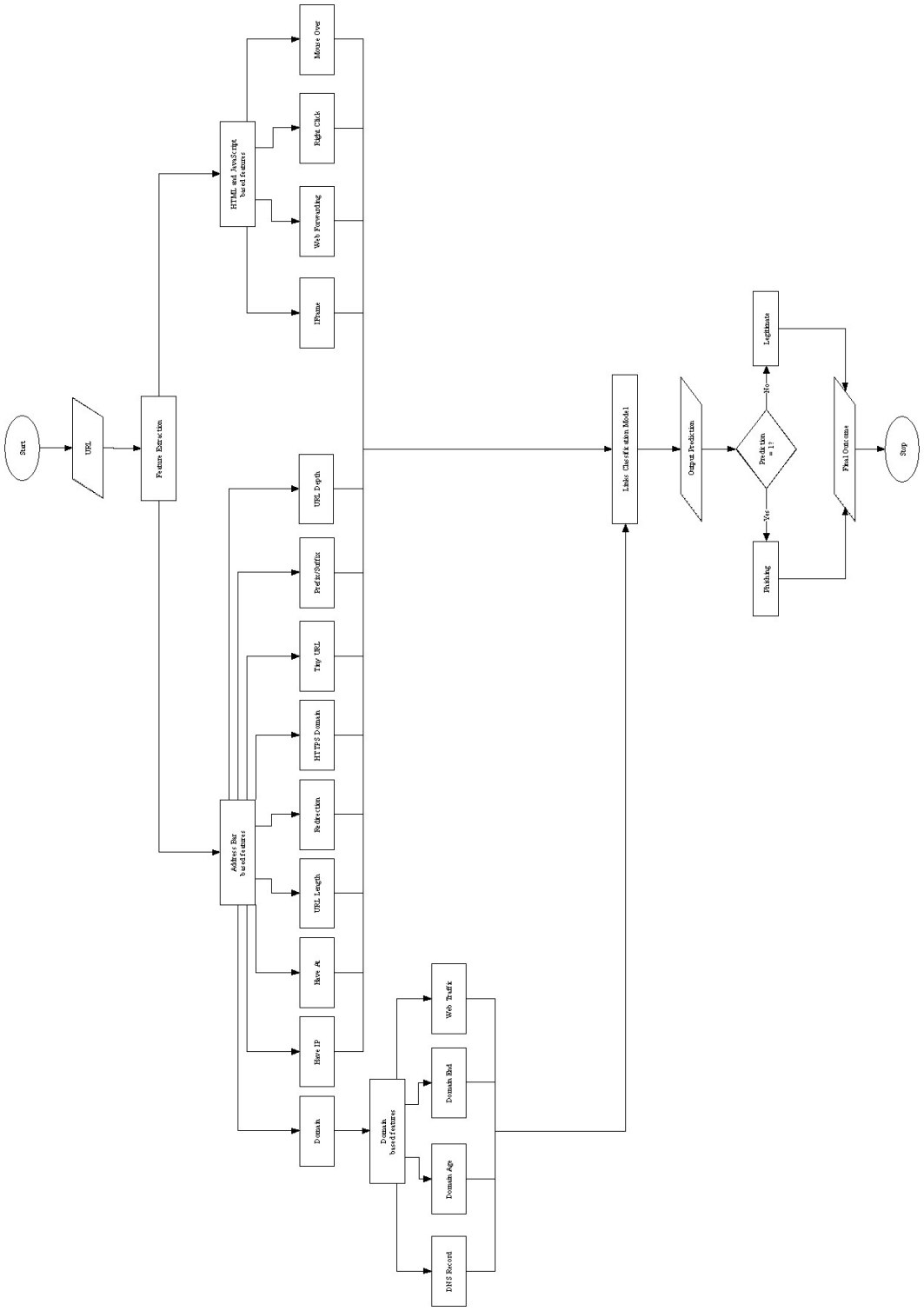


Figure 3: Link Classifier Workflow

5.2.3 Semantic Classifier Algorithm

Workflow for the semantic classifier:

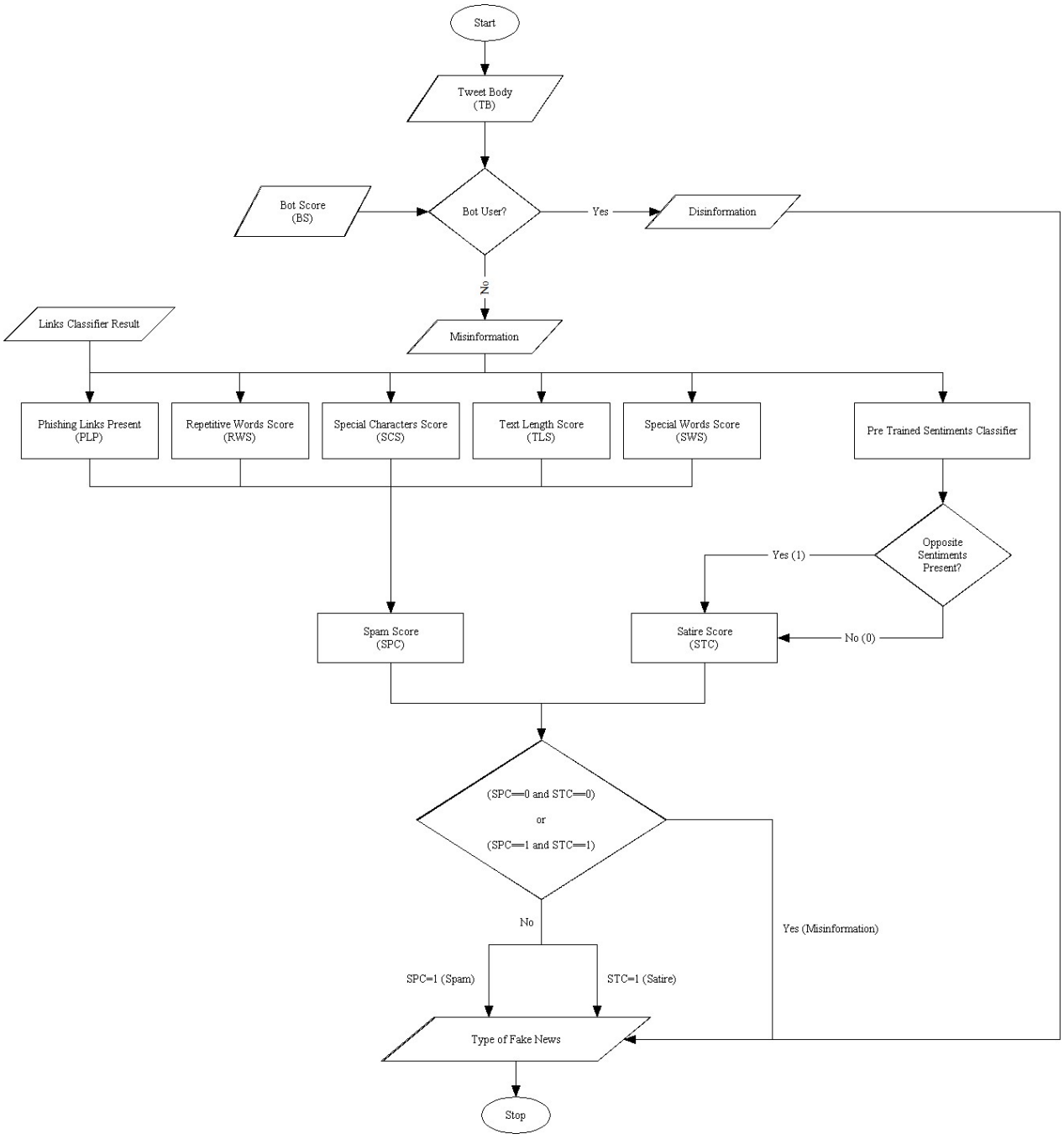


Figure 4: Semantic Classifier Workflow

The Semantics Classifier is responsible for categorizing a fake piece of tweet into one of the following types:

- **Misinformation** - if possible, we further try to classify misinformation as either "spam" or "satire".
- **Disinformation**

Thus, the final outcome out of this classifier would be one of the following: [**"misinformation"**, **"disinformation"**, **"spam"**, **"satire"**].

Looking at the workflow, we can see that the main inputs to this classifier are the Tweet Body, the Bot_Score(from the main classifier) and the output of the Links Classifier (to know if the tweet consists of any phishing links or not). Taking help of these information, we proceed in the following way:

- As per the main workflow (given at the end of section 5), we know that if a tweet body is entering the semantics classifier, it has been classified as false.
- Imagine the scenario that there is a bot account on Twitter, and that account is tweeting some information. Now, given that we know that this particular user is a bot, and is sharing some false news, would we believe that the user might've made a mistake? Of course not. Thus, we assume that in case a user is a bot, and since the user is found to be sharing false news, that would make the tweet fall under the category of "*disinformation*".
- In case the user is not found to be a bot, we atleast know that the tweet will be classified as "*misinformation*". However, we will try to see if we can further classify the tweet or not, based on the multiple factors.

Spam Detection

For detecting if any misinformation is spam or not, we consider the following factors:

- **Repetitive Words** [Repetitive Words Score (RWS)]

- Spam messages usually contain many repeating words. If we assign a threshold k, and if frequency of any word comes out to be greater than k, we classify the text as spam.
- But, we had to be careful since a real message can also have such repeating words at times.
- Taking this into consideration, we deduced a unique way to find the value of the threshold k.
- Usually when we write something long, we don't just have one or two words repeating a lot of times (ignoring stopping words), we have several of them.
- Using this idea, we firstly calculated the frequency of each unique word in the tweet body (after removal of stopping words and other pre-processing).
- Now, we find the point where the frequency of the words drop significantly in the tweet body and use that to determine how many unique words are noteworthy, with a fallback to a default value(4) if there's no clear drop-off.
- To achieve this, we first sort the words based on their frequencies in descending order. So, the most frequent word comes first, followed by less frequent words.
- Then we try to find the point where the frequency decreases sharply (where the frequency is less than half of the frequency of the most frequent word). If such a point is found, we stop and note the position of this drop-off.
- The position where the drop-off occurs is assigned to 'k'. This 'k' essentially tells us how many unique words are significant in the given sentence. If there's no significant drop-off (meaning all words have similar frequencies), it sets 'k' to a default value of 4.
- Note that in case we find a drop off point, the value of k doesn't matter, since we already know there are/are certain words that are repeating way more than the rest of the words in the sentence. Thus, k in this case merely indicates that the value of this factor will be 1.

  For example, consider the sentence: *"Hi! We are pleased to announce a discount on our latest collection. This discount is for a limited time only. Discount is valid till stocks last. Avail your discount today!!"* Here, we

know that "discount" is being used way more than the rest of the words in the sentence, Thus, in this case k would be set to 1, indicating that at least 1 word is repeated a lot of times, which would make the score of this factor as 1(indicating spam).

- On the other hand, in case we cannot find a significant drop of point, say for example the tweet: *"verify add discount verify add discount, ! @@@ verify add discount verify add discount verify add discount"*: Here we are not able to find a drop off point.
- For such cases, we have to assume a default value that we can see as too repetitive. We have considered 4, meaning if a word comes out more than 4 times, we can consider that to be repetitive. The reason for choosing this particular value is that when humans genuinely are typing something, they try not to be repetitive, but in case we find words occurring thrice or more times, it can be indicative that words are repeating a lot compared to normal human typing styles.
- This factor can either have a score 1 (indicating that there are certain words repeating a lot of times, which could mean a potential spam) or 0 (all words were within reasonable frequency limits which suggests not a potential spam).

- **Presence of Links** [Phishing Links Present (PLP)]
  - As mentioned before, the semantics classifier takes in the output from the links classifier.
  - Thus, no matter if the tweet body is true/false, if there is presence of phishing links in them, it is an attempt to mislead the user thus marking itself as spam.
  - For example: I can say that "*For your interviews register below*" and provide a fake link which takes all the private information of the user, which is spam.
  - This factor can either have a score 1 (potential spam) or 0 (not a potential spam).

- **Use of Special Character**s [Special Characters Score (SCS)]

- Excessive use of special characters like (*???*, *!!!!!!*, @@@, , etc) potentially indicate a spam message.
- We have a few assumptions for this factor. For detecting whether a sentence consists of special characters or not, we are considering two styles:
  * Continuous sequence of same special character (###, %%, &&&, etc)
  * Continuous sequence of different special characters (*, ^^%, @&&, etc)
  * We are NOT considering sequences of different characters separated by space as a single sequence.
    Example: "*& * %*" will be considered as 3 separate sequences and not a single sequence as "*&*%*"
  * This assumption is taken to ensure that non spam texts that include legitimate use of multiple special characters are not wrongly classified as spam.
- This factor also, can either have a score 1 (potential spam) or 0 (not a potential spam).

- **Text length** [Text Length Score (TLS)]
  - Spam tweets are either too short/too long compared to normal/average tweet length.
  - We researched online and figured out what is the average length of a tweet message, and used that information to accordingly mark the message as spam or not.
  - Again, some real messages can also be shorter/longer compared to the average length of a tweet.
  - As per online articles, Only 5% of tweets are longer than 190 characters, indicating that Twitter users have been for so long trained to keep their tweets short, they haven't adapted to take advantage of the extra room to write. Moreover, only 1% of tweets hit the 280-character limit.
  - We are considering the average range as [30, 140] (both inclusive).
  - In case the length is outside of this range, we can give a score of 1 (potential spam), else 0 (not a potential spam).

- We are NOT considering *"Twitter Blue"'s* 4000 character limit in this project
- Different characters have different sizes on the Twitter platform (example: emojis, etc have a character space of 2). Thus, for simplicity we assume that all characters take a space of 1 unit.
- References:
  * *https://kurtgessler.medium.com/twitter-length-study-do-longer-tweets-drive-more-engagement-and-referral-traffic-3dd0781363ff*
  * *https://techcrunch.com/2018/10/30/twitters-doubling-of-character-count-from-140-to-280-had-little-impact-on-length-of-tweets/*
  * *https://smk.co/the-average-tweet-length-is-28-characters-long-and-other-interesting-facts/: :text=The%20average%20Tweet%20length%20is%2028-%20characters%20long%2C%20and%20other%20interesting%20facts,-January%2011%2C%202012text=Earlier%20this%20week%20Twitter-%20employee,one%20million%20tweets%2C%20excluding%20retweets.*
  * *https://developer.twitter.com/en/docs/counting-characters*

- **Frequency of Special Words** [Special Words Score (SWS)]
  - Words like [*discount*, *urgent*, *hiring*, *free*, etc] are usually what spam messages consist of.
  - One way that we could've used to detect spam would've been to train a model using NLP and bag of words technique on a given dataset, but that would add additional complexity to this project.
  - Since we already are considering several other factors for detecting spam, we've simplified this approach.
  - Thus, we're using a bag of such words from multiple sources on the internet and then used that to indicate whether the tweet body is potentially spam (1) or not (0).

Thus, we have taken into consideration multiple factors to classify a piece of misinformation as "spam". However, some of the factors have more importance compared to others, and we cannot just rely on the outcome of one factor for

classification with complete accuracy. As a result, we decided to take majority voting of all the factors to classify spam with certain important factors like **repetitive_words_score**, **special_word_score** and **phishing_links_present** having more weightage in voting compared to the rest of the factors.

Satire Detection

- Now, coming to the detection of *"satire"* present in the tweet body, we could not have been able to train a model ourselves to detect the same. Moreover, no such datasets even exist that we could train our model on for such cases.
- Detecting human sentiments like "satire" is difficult for humans as well let alone teaching a machine to do that which is the reason why is is still an ongoing research in the deep learning world, but here we have taken a relatively high level and a simplistic approach to try to detect the same.
- Let's consider the sentence *"I am happy that I got late to class"*.
  Here, it is evident that the speaker is using satire in the way that he is happy (a positive sentiment) that he was late for his class (a negative sentiment) which is usually not how humans feel.
- In the sentence, *"I had the best sleep of my life after watching this new scary movie"*, the speaker uses satire to show the fact that the horror movie was so boring that they fell asleep instead of being scared.
- We could look at several other examples as well, one thing that is common in many of them is that sentences that indicate satire, sort of consist of both positive and negative sentiment in them (like happy & late, best sleep & scary movie, etc). And it is this very pattern that we have used to detect potential use of satire in the tweets.
- Thus, to speak in simple terms, we take the tweet body, we break it down into sub sentences and then try to find if it consists of both positive and negative sentiments or not. If yes, we can say that it is potentially satire (score 1) or not (score 0).

- Now, to detect the type of sentiment (positive/negative) present in sentences, we've used a pretrained model (Link: *https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment-latest?text=I+was+late*), which is a RoBERTa-base model trained on 124 million tweets from January 2018 to December 2021, and fine tuned for sentiment analysis with the TweetEval benchmark.

- Basically what this model helps us achieve is that, we feed in a sub sentence into this model, and since it has already been trained, it gives one of the three labels as output: [**"positive"**, **"negative"**, **"neutral"**] indicating the sentiment of the sentence.

- Now, we can't obviously put the entire tweet body into this model and get the desired results, since for example, *"my flight is delayed and i couldn't be happier"* overall states that the speaker is happy, and so the model would give the output as "positive". Thus, we needed to find a way to break the tweet body into sub sentences where then each sub sentence could be fed into this model.

- Thus, we used a simple technique which is inspired from the traditional NLP algorithm and can be used as a python library for our ease (Library link: *https://github.com/cardiffnlp/tweetnlp*), which can be experimented with here. This approach is part of the paper: **TweetNLP**: Cutting-Edge Natural Language Processing for Social Media (Link: *https://arxiv.org/pdf/2206.14774*) [9]

- Let's try to understand it with an example. Consider the sentence: ***"The quality of work is so amazing that it feels like it was done by a child"***

  - We parse the sentence from left to right, word by word and sort of keep track of the sentiment we have till now, like:

    * "The" - neutral
    * "The quality" - neutral
    * "The quality of.." - neutral
    * "The quality of work is so **amazing**" - **positive**

- Till this point we were having a neutral sentiment in the sentence, but now we encountered *"amazing"*, which is a positive word, and thus the overall sentiment of the sentence parsed till now changed from neutral to positive, thus, there was a break off. And this is where we split our sentence into its first sub sentence as *"The quality of work is so amazing"*.
- We continue with the remaining part of the sentence:
  - "that" - neutral
  - "that it" - neutral
  - "that it feels" - neutral
  - "that it feels **like**" - **positive**

  "Like" is a positive sentiment word, so we break it off again: "that it feels like"
- Continuing in the above fashion:
  - "it" - neutral
  - "it was" - neutral
  - "it was done" - neutral
  - "it was done by a child" - neutral

  Since this part is only left it will be the last sub sentence.
- Thus, we broke down: *"The quality of work is so amazing that it feels like it was done by a child"* into :
  "The quality of work is so amazing",
  "that it feels like",
  "it was done by a child"
- Now, we feed each of these sub sentences into the RoBERTa model and get the type of sentiment present in them. In case for a particular tweet body we come across both positive and negative ones, we are very likely to have encountered the use of satire (score 1). And if not then it's suggestive of the fact that satire was not used here (score 0).

- As we saw a very high level overview of how we are trying to detect satire in the tweet bodies it is very important to note that even though this method seems to

be effective on most of the instances that it is designed for, there will of course always be cases where it fails.

- For example, the module used to divide the tweet body into sub sentences does not recognize certain words as potentially sentimental (like *"spilled"*, *"meeting"*, etc) which are otherwise very easily recognizable to humans.

- After checking the tweet body for *"spam"* and *"satire"*, in case we get both of them absent or present, we finally categorize the tweet as misinformation only, to avoid further confusion. It is only when either of them is present that we further classify misinformation as *"spam"* or *"satire"*.

And this is how we haven't used any dataset to train this model on, and have rather relied on some useful techniques in detection of spam and satire, it is still reliable for most of them. Even though the accuracy of the model cannot be predicted accurately, we are very pleased to have been able to reliably detect "spam" and human sentiments like "satire", that too without the use of any dedicated dataset to train our classifier on. And it is for this very fact that there will be cases where the classification into these types may not be completely accurate, but chances of those cases are relatively very rare.
Link to our algorithm: *https://colab.research.google.com/drive/1VDbKCI-EaziTBpX-uXOjwiZ0jGl-B2Cwt?usp=sharing*

5.2.4 Image Classifier

The Image Classifier is responsible for classifying an image as real or fake based on the CIFAKE Dataset. In order to be able to come up with a model which can accurately perform this task, we went with a CNN model. However, just using a simple CNN model would not have sufficed. Thus, we used multiple different types of CNN's and compared how well they performed on the same given dataset (to ensure similar working conditions for all models) and compared the results we obtained. (Notebook link: *https://colab.research.google.com/drive/1XMEKn4xUahyHAl1mqb3u-HDyJEcdFDXU-?usp=sharing&authuser=1scrollTo=hFtzu8DI4QxD*)
All the models share the same architecture for a fair comparison: the input layer, followed by the base model with pre-trained weights from the *ImageNet*, then a few

dense layers, and then a unit output with a sigmoid activation function. Training is conducted with early stopping criteria monitoring the validation loss, and the best weights will be restored once the training is completed.

We used the following models for our image classification task:

- **ResNet - 50:**
  - Traditionally it's a 50-layer convolutional neural network (48 convolutional layers, one MaxPool layer, and one average pool layer).
  - It's a part of the ResNet (Residual Network) family of models, which are designed to make training deep neural networks more effective and easier.
  - The key innovation is the idea of residual learning. Instead of each layer learning a full transformation of the input, it learns the difference (or "residual") between the input and the output which is done using shortcut connections that skip one or more layers.
  - Shortcut Connections add the input of a layer directly to the output of a few layers ahead. This helps to preserve the gradient (signal) as it flows back through the network during training, making it easier to train very deep networks.
  - We used this model in the following manner:

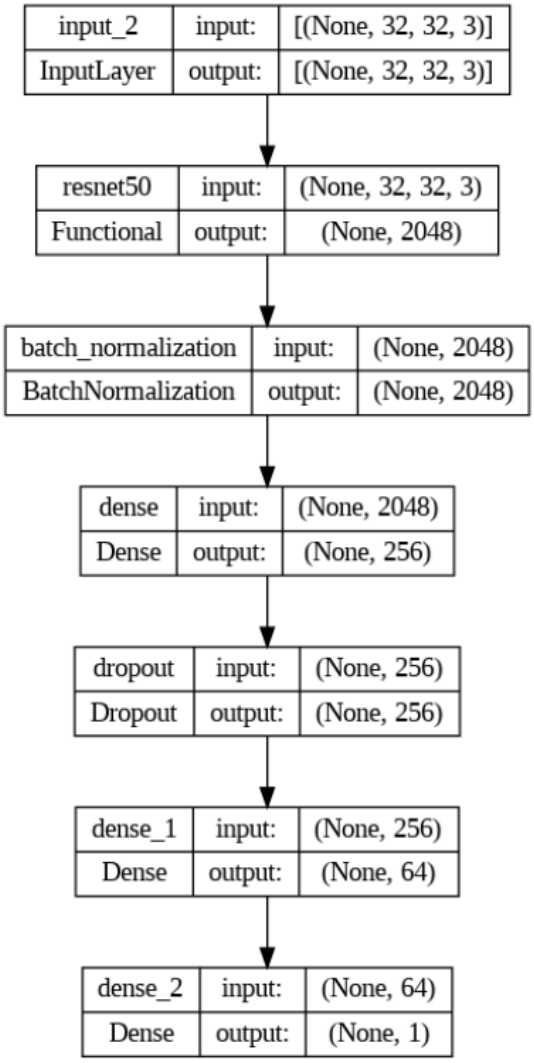


Figure 5: ResNet - 50 Working

- **VGG - 16:**
  - The VGG-16 is another CNN that consists of 16 layers and is known for its simplicity and effectiveness.
  - The "16" in VGG-16 refers to the total number of layers with learnable parameters, specifically 13 convolutional layers and 3 fully connected layers.
  - The architecture of this approach can be understood from:

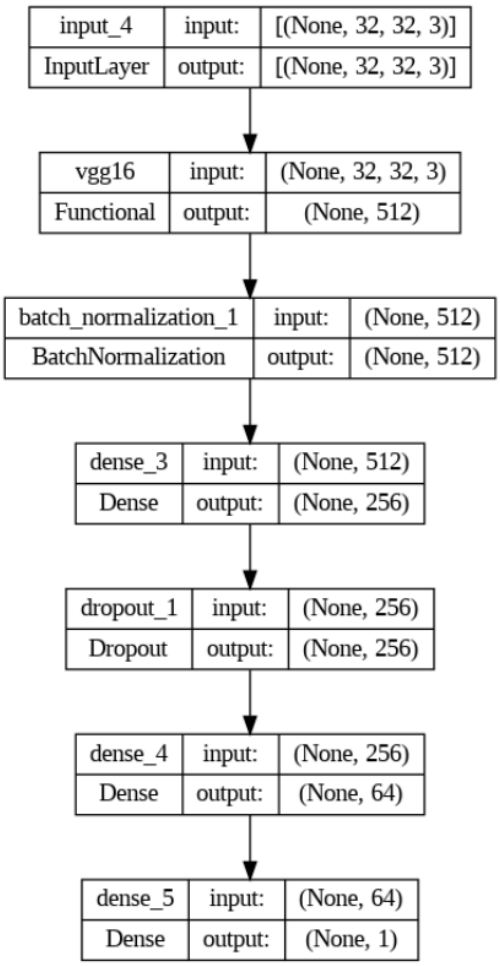


Figure 6: VGG - 16 Working

- **EfficientNetV2:**
  - EfficientNetV2 is a state-of-the-art deep learning model designed for image recognition, known for its efficiency and high performance.
  - It uses compound scaling to balance depth, width, and resolution, making it both powerful and efficient. Key innovations include efficient blocks like MBConv and Fused-MBConv, the Swish activation function, and SE layers to focus on important features.
  - Overall, EfficientNetV2 combines advanced techniques to achieve high accuracy with lower computational cost and better optimization than VGG-16, making it a popular choice for many image recognition applications.

– The architecture of this approach is given below:

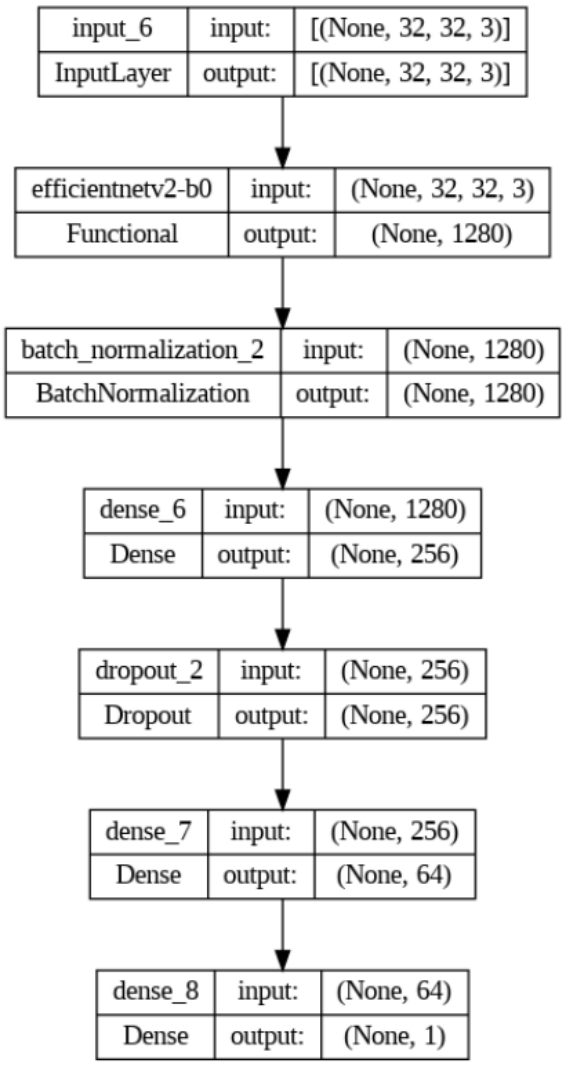


Figure 7: EfficientNetV2 working

The overall workflow of the entire model combining all these classifiers is given below:

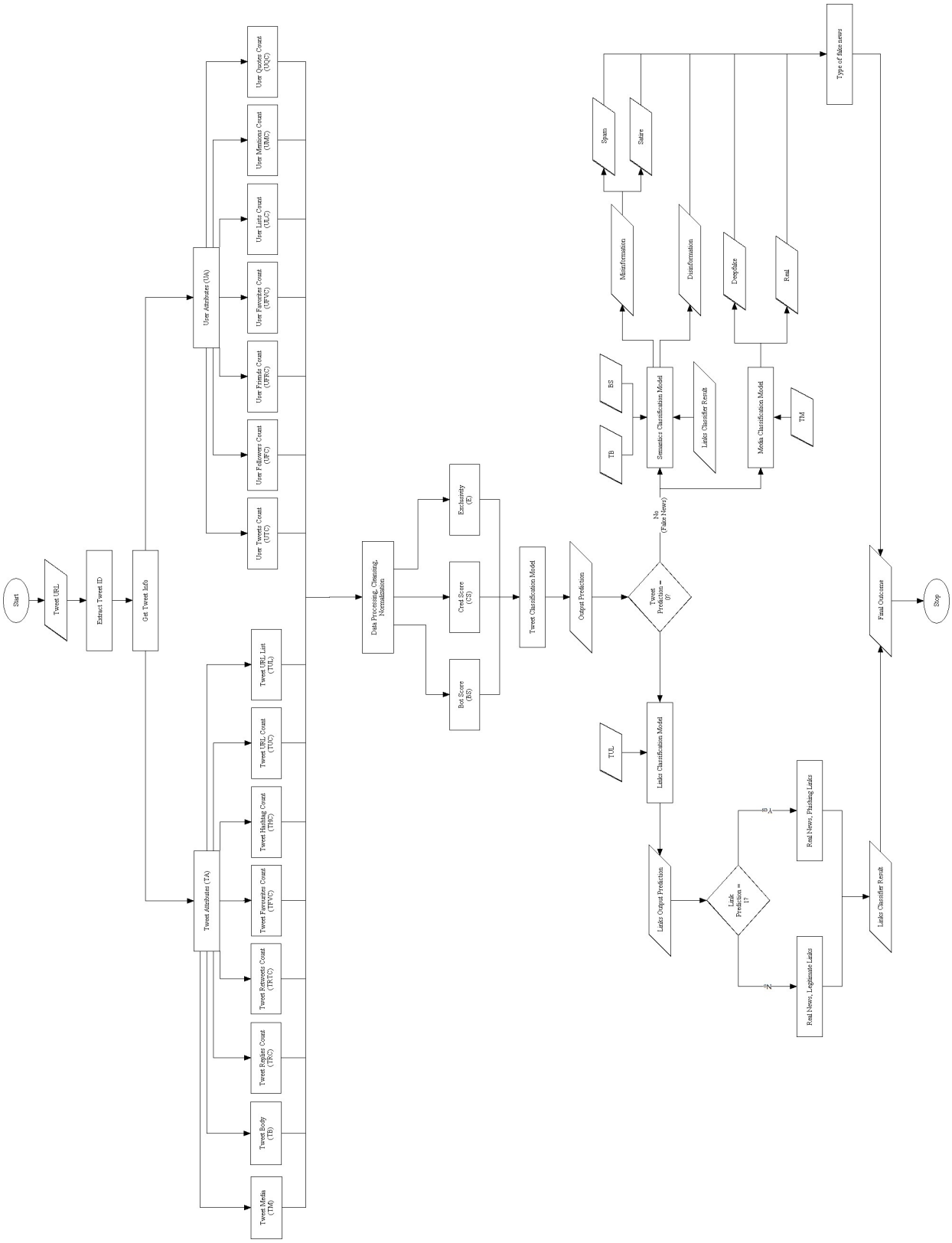


Figure 8: Overall Workflow

6 Experimental Result And Analysis

6.1 User Analysis

Now, for finding out whether a tweet is sharing a false piece of news or not, we have analyzed the user's characteristics or the user's influence. Imagine a scenario where a trusted and verified source of information is telling us something. We are more likely to believe them compared to a stranger right? Similar to this fact, we tried to incorporate the influence of the user as well while categorizing tweets as true/false. Taking much needed insights from `TruthSeeker Dataset`, we've incorporated "Bot Score" and "Cred Score" from their research. To further improve the accuracy of considering the user's influence, we've introduced an additional feature called **Exclusivity**

6.1.1 Bot Score

Based on the Bot Score in the `TruthSeeker dataset`, this is a value between 0 and 1 indicating whether a particular user is a potential bot account (1) or not (0). A value less than or equal to 0.5 indicates the user is not a bot and the opposite will label them as a bot. As mentioned in their research paper, this score was determined by a model based on 17 features using features like UFC, UFRC, UTC, ULC, TUC and the age of the account.

Unfortunately for us, we were not able to extract the age of the account as we did not have the Tweet IDs of tweets. Hence, we trained a Regression Model (Link: *https://drive.google.com/file/d/1qii8cY1aOXv28oxtT3PgrtdnjLZK15I5/view?usp=drivesdk*) to fit on the data that `TruthSeeker` provided us with, and then used this model to predict the Bot Score for our dataset processing. The idea behind using this metric is that bot accounts are commonly used to spread fake news on a platform. They are mostly short lived, meaning they have a very short lifespan, but tend to be huge in numbers. Keeping in mind the fact that calculating this lifespan was not possible, figuring out if a user is a bot or not can still provide us crucial information as to whether the news being shared by the tweet is most likely to be false or not. [5]

6.1.2 Credibility Score

Similar to the Bot Score, this metric is also based on the Cred Score in the **TruthSeeker Dataset**, which is also a value between 0 and 1 showing the credibility of

the user. The formula provided in their research work has been used to calculate the same:

$$CredScore = UFC/(UFRC + UFC)$$

We have added the edge case where if the UFC and UFRC both are 0, then Cred Score will eventually be 0 (since dividing by 0 makes no sense). The Cred Score will be 1 if the user has no friends, meaning that he is being followed by at least 1 person, thus we can say that the user is credible (though they might not have that much influence on their followers). A Cred Score of 0 will indicate that either the user has no followers or no followers and friends, and thus we can say that no other user on the platform "trusts" / finds this user as a credible one.

6.1.3 Exclusivity

We already have UAS and TAS with us from before. We define Exclusivity as *"a binary metric which signifies whether a particular tweet is exclusive (not a part of) a user's previous other tweets based on their activity scores namely, UAS and TAS"*.

Imagine a scenario where a particular User X gets around 300 likes on his tweets. All of a sudden X tweets something which booms up and receives around 4000 likes. Thus, only after knowing the fact that X usually got around 300 likes, we were able to say that this new tweet doesn't fit the usual pattern of X. So, with the help of UAS (user's average activity 300 likes) and TAS (tweet's activity 4000 likes) the current tweet is said to be Exclusive(1) of the tweets posted by user X.

We needed to come up with an inequality, to calculate the values for Exclusivity. We knew that the TAS has to be greater than UAS, i.e, the tweet's activity has to be more than the user's activity to be declared as Exclusive (1), else not (0). Thus, If

$$TAS > k * UAS,$$

$$Exclusivity = 1, else\, 0$$

Where, k is a threshold, i.e. as to how much the tweet's activity needs to be more than the user's activity to be classified as Exclusive. Taking the value of k as **1.5** states that the current tweet has 50% more engagement compared to the user's usual engagement. [6]

6.2 Overall Classification Performance

We have created tables consolidating the results for different classifiers for each category and for different splits:

Tweets Classifier

Category	Dataset Split	Algorithm	Accuracy	Precision	Recall	F1-Score	AUC
Tweets Classifier	70-30%	XGBoost	0.947	0.939	0.832	0.882	0.973
		Random Forest	0.944	0.932	0.827	0.877	0.972
		Logistic Regression	0.897	0.859	0.679	0.758	0.915
		SVM	0.924	0.904	0.763	0.828	0.955
	75-25%	XGBoost	0.947	0.935	0.837	0.883	0.973
		Random Forest	0.942	0.93	0.82	0.871	0.971
		Logistic Regression	0.897	0.86	0.68	0.759	0.914
		SVM	0.926	0.907	0.766	0.831	0.956
	80-20%	XGBoost	0.947	0.941	0.831	0.882	0.972
		Random Forest	0.944	0.931	0.826	0.876	0.972
		Logistic Regression	0.897	0.86	0.681	0.76	0.913
		SVM	0.926	0.908	0.769	0.833	0.922

We proceeded to go with the 80-20% split as we got good results across all the different performance metrics for each classifier compared to others, and also this ratio is a commonly used standard ratio for training majority of models.

Links Classifier

Category	Dataset Split	Algorithm	Accuracy	Precision	Recall	F1-Score	AUC
Links Classifier	70-30%	Decision Tree	0.928	0.979	0.875	0.924	0.953
		Random Forest	0.899	0.991	0.805	0.888	0.962
		MLP	0.936	0.986	0.883	0.932	0.97
		XGBoost	0.934	0.983	0.882	0.93	0.97
		SVM	0.89	0.994	0.784	0.876	0.906
	75-25%	Decision Tree	0.935	0.982	0.887	0.932	0.958
		Random Forest	0.907	0.996	0.82	0.9	0.965
		MLP	0.94	0.988	0.892	0.938	0.976
		XGBoost	0.939	0.988	0.892	0.937	0.976
		SVM	0.897	0.996	0.801	0.887	0.941
	80-20%	Decision Tree	0.901	0.946	0.852	0.897	0.939
		Random Forest	0.895	0.995	0.796	0.884	0.957
		MLP	0.934	0.97	0.897	0.932	0.976
		XGBoost	0.931	0.966	0.896	0.929	0.976
		SVM	0.886	0.786	0.786	0.874	0.933

We proceeded to go with the 80-20% split as we got good results across all the different performance metrics for each classifier compared to others, and also this ratio is a commonly used standard ratio for training majority of models.

Image Classifier

Category	Dataset Split	Algorithm	Accuracy	Precision	Recall	F1-Score
Media Classifier	83-17%	ResNet-50	0.9604	0.9692	0.951	0.96
		VGG-16	0.9629	0.9611	0.9649	0.963
		EfficientNet V2	**0.9715**	**0.9811**	**0.9615**	**0.9712**

We went with 83-17% split as it was already prepared and provided in the CIFAKE dataset with perfectly shuffled samples in both training and testing set, and was already best suited for our purpose of training a classification model.

6.3 Comparison of Evaluation Metrics

- **Tweets Classifier**

Plotting the performance chart for these models, we get:

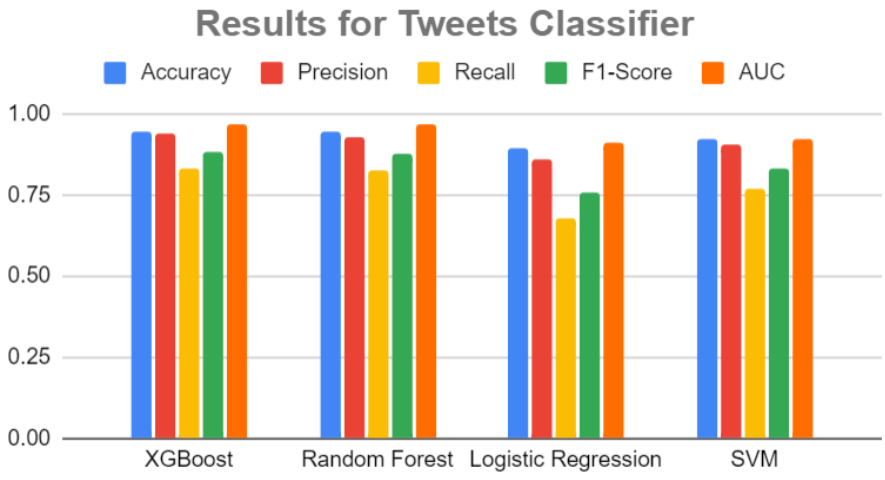


Figure 9: Tweet Classifier Result

As we can see from the above results, that XGBoost has the most accuracy compared to the rest. Thus, we proceeded with the **XGBoost** model for our Tweets Classifier.

- **Links Classifier**
  We plot the performance chart for the models:

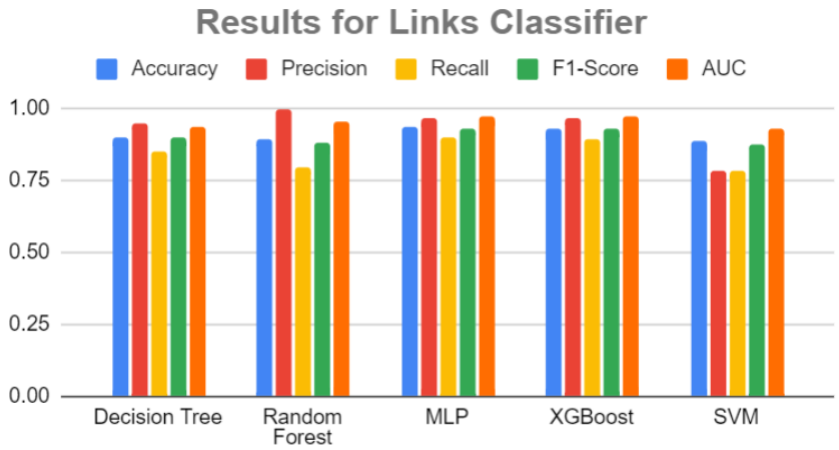


Figure 10: Link Classifier Result

As we can see from the above results, like in the previous case, XGBoost has the most accuracy compared to the rest. Thus, we proceeded with **XGBoost** model for our Links Classifier.

- **Image Classifier**
  Plotting the performance chart for these models, we get:

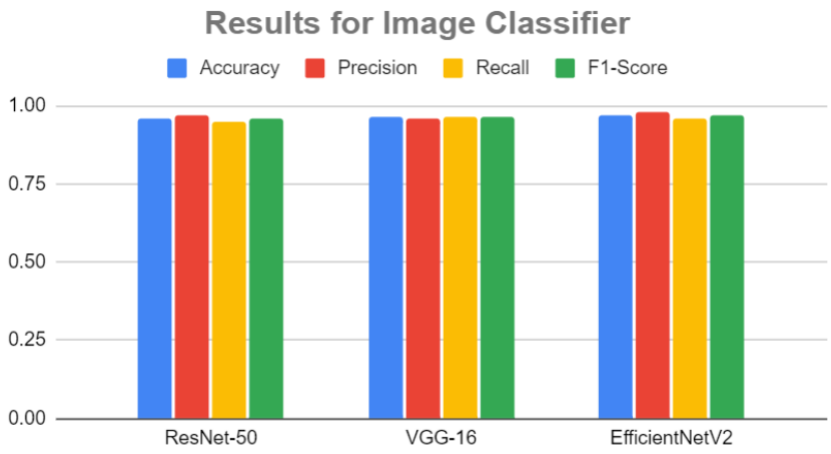


Figure 11: Image Classifier Result

Comparing the three models, we decided to go with **EfficientNetV2** since it has the maximum accuracy of about 97% along with higher precision scores amongst all the models.

7 Conclusion

Working on this project helped us gain valuable insights on working with highly specific datasets related to social media platforms, namely Twitter(now X). We had to manipulate many features from the datasets we found on the internet as well as add some of our own to get the results which we were looking for. We've successfully been able to incorporate multiple models which detect fake contents in tweets, followed by checking the legitimacy of URLs (if any) present in the tweet and further categorizing them as disinformation, misinformation, satire or spam in case they are found to be false. Adding on the textual inputs we were also able to handle classification of images based on whether they're real or fake.

7.1 Project Benefits

As discussed before, we studied multiple models that try to address the problem of detecting fake content online. Some limitations we encountered with them were limited platform availability, manual fact checking, dealing only with a particular type of media, having to leave the current work session and many more. In today's

fast paced world, people want to make their every second count. Considering all these factors, our project aims at providing the following benefits:

- Detection of human sentiments like satire, spam etc
- Detection of multiple media types
- Browser extension support, along with a dedicated platform
- Continuous learning and updates
- Real-Time Analysis and User Feedback

8 Future Scope & Work

Taking on from last semester we are able to cater inputs of multiple types, namely text and images. As mentioned before we've aimed to work with multimodal data, a part of which has already been achieved. However, we also plan on being able to extend this classification to videos as well, so that we can cover all types of media on social media platforms (Twitter). Moreover, we also plan on using other models and datasets for our semantics classifier for a better classification. Additionally, in order to make our models easily accessible for the people, we plan to add web extension support so that they can seamlessly fact check information without having to leave their pages. Currently we are focused only on a particular social media platform like Twitter (now X), however fake content is not just limited to this platform. Keeping that in mind, we also plan on being able to detect fake content on other social media platforms as well.

References

[1] Scott Counts Aditya Pal. Identifying topical authorities in microblogs. 2011.

[2] Jorge Bendahan Rami Puzis Aviad Elyashar. Detecting clickbait in online social media: You won't believe how we did it. 2022.

[3] Gwangbin Bae, Martin de La Gorce, Tadas Baltrušaitis, Charlie Hewitt, Dong Chen, Julien Valentin, Roberto Cipolla, and Jingjing Shen. Digiface-1m: 1 million digital face images for face recognition. In *2023 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2023.

[4] Jordan J. Bird and Ahmad Lotfi. Cifake: Image classification and explainable identification of ai-generated synthetic images, 2023.

[5] Sajjad Dadkhah, Xichen Zhang, Alexander Gerald Weismann, Amir Firouzi, and Ali A Ghorbani. Truthseeker: The largest social media ground-truth dataset for real/fake content. 2023.

[6] David Hans Rodolfo Villarroel Roberto Munoz Fabián Riquelme, Pablo Gonzalez-Cantergiani. Identifying opinion leaders on social networks through milestones definition. 2019.

[7] Pablo González-Cantergiani Fabián Riquelme. Measuring user influence on twitter: A survey. 2016.

[8] Ahmad Lotfi Jordan J. Bird. Cifake: Image classification and explainable identification of ai-generated synthetic images. 2024.

[9] Talayeh Riahi Asahi Ushio Daniel Loureiro Dimosthenis Antypas Joanne Boisson Luis Espinosa-Anke Fangyu Liu Eugenio Martínez-Cámara Gonzalo Medina Thomas Buhrmann Leonardo Neves Francesco Barbieri Jose Camacho-Collados, Kiamehr Rezaee1. Tweetnlp: Cutting-edge natural language processing for social media. 2022.

[10] Alex Krizhevsky. Learning multiple layers of features from tiny images. pages 32–33, 2009.

[11] Mohammad Saiful Islam Mamun, Mohammad Ahmad Rathore, Arash Habibi Lashkari, Natalia Stakhanova, and Ali A Ghorbani. Detecting malicious urls using lexical analysis. In *Network and System Security: 10th International Conference, NSS 2016, Taipei, Taiwan, September 28-30, 2016, Proceedings 10*, pages 467–482. Springer, 2016.

[12] James Zou Yiqun T. Chen. Twigma: A dataset of ai-generated images with metadata from twitter. 2023.