



# INFO 4290 S10

## Final Report

**April 8th, 2024**

---

Team Member	Student Number
Le Thien Nhan Nguyen	100407640
Arin Banerjee	100351961
Harmen Rai	100335755
Karanjot Sandhu	100351539

## Tables of Content

<b>Background .....</b>	<b>3</b>
<b>Functional Requirement.....</b>	<b>3</b>
Data Train Collection .....	3
Extraction .....	3
Data Labelling .....	3
Machine Learning Model .....	3
Training and Testing Data Model .....	3
Real-time data collecting .....	3
Reporting .....	4
<b>Non-Functional Requirement .....</b>	<b>4</b>
Usability .....	4
Performance.....	4
Error Handling .....	4
Compliance.....	4
Reliability .....	4
Scalability .....	4
<b>Project scope.....</b>	<b>5</b>
<b>Project budget .....</b>	<b>5</b>
<b>Code/train model .....</b>	<b>8</b>
Training Parameters: .....	8
Model and Accuracy: .....	9
Random Forest Classifier: .....	9
Logistic Regression:.....	10
Gradient Boosting:.....	10
Conclusion of between Model .....	10
<b>Dataset: .....</b>	<b>11</b>
<b>React page/UI .....</b>	<b>12</b>
<b>Data Collection .....</b>	<b>12</b>
Challenges .....	13
<b>Conclusion .....</b>	<b>13</b>
<b>REFERENCES.....</b>	<b>13</b>

## Background

**Abstract:** We're creating a tool to find Twitter bots, those automated accounts that can spread fake news and spam. Using machine learning, we will analyse a user's followers and those they follow to spot potential bot accounts. This helps make online interactions more genuine and reliable.

**Problem Statement:** Twitter is swarmed with automated bot accounts that harm the quality of user experiences by spreading fake information, generating fake engagement and spam. To tackle this problem, we're building a tool that can tell apart real users from bots.

## Functional Requirement

### Data Train Collection

- Establish a guideline to collect a data train and testing set to training the machine algorithm
- The Data should include both genuine users and bot account to help the algorithm to
- understand the difference

### Extraction

- Extract and collect data from user's profile and tweet behaviour through Twitter API, and process then export it for the Json file into a csv file

### Data Labelling

- Filter down the user's data collected, through a predetermined criteria as they may not provided efficient data
- Create a labelled dataset with a ground truth of genuine and bot accounts to train and validate the model.

### Machine Learning Model

- Develop a machine learning model for bot detection based on the research that was conducted.

### Training and Testing Data Model

- Train the model on the labelled dataset and validate its performance using appropriate metrics, such as precision, recall, F1-score, and accuracy

### Real-time data collecting

- Establish a system that collect select user's follower data in-real time through Twitter API model
- Filter down redundant data before inserting it in the machine learning model

## **Reporting**

- The system represents the data in an easy to understand format for the user. Complete with actual bot's follower of the account, the percentage, etc

## **Non-Functional Requirement**

### **Usability**

- User is able easily understand and utilize the UI within 5 minutes
- Provide clear and concise documentation or user guides to help users make the most of the system

### **Performance**

- Data filtering should be performed efficiently to avoid bottlenecks in data processing. This is especially important when dealing with a large dataset.

### **Error Handling**

- Implement error-handling mechanisms to address any issues that may arise during data filtering, ensuring the robustness of the system

### **Compliance**

- Ensure that the system complies with relevant data privacy and security regulations, respecting user rights and privacy
- Adhere to ethical guidelines and Twitter's terms of service to ensure that data collection and analysis are conducted responsibly and legally

### **Reliability**

- System should be available and functional with minimal downtime or service interruptions

### **Scalability**

- System should be able to handle an increase of users and data without a decrease in performance
- 

## **Introduction**

With the continuous growth and innovation of social media, there are more people using it now than ever before. With the increase of authentic users also come the inauthentic, or bot accounts. These accounts mostly bring unwanted traffic towards others in ways that at times can even hinder an user's experience on social media. Bots that spam advertisement, crypto or political messages can be used to manipulate the algorithm and cause constant spam and leads to issues with investors. Without knowing how much of a social media user base is

authentic, investors will have a tough time knowing how active a social media platform is and how much of the growth is authentic. A good example which is currently relevant is Twitter/X. Recently there has been an increase in bots that spam the replies of users which is causing some to feel irritated with their experience on the app. With this also comes engagement that is not authentic, and can change the algorithm of popular tweets. If a tweet gets more engagement it will show up on a users timeline, but with the engagement possibly being bottled it can show how easily manipulated information can become. With the creation of a bot detection program, incorporated with a machine learning algorithm we can determine and analyse the impact of crypto bots on the website and social media application.

## Project scope

The overall scope and key needs that will be included are to train/test an algorithm model to get data and categorise twitter accounts by specific categories like username, follower count, following count, keywords related to the bots, average tweet time, hashtag count and so forth.

## Project budget

The fees that are to be considered is the plan for access to twitter api itself.

## Find the right access for you

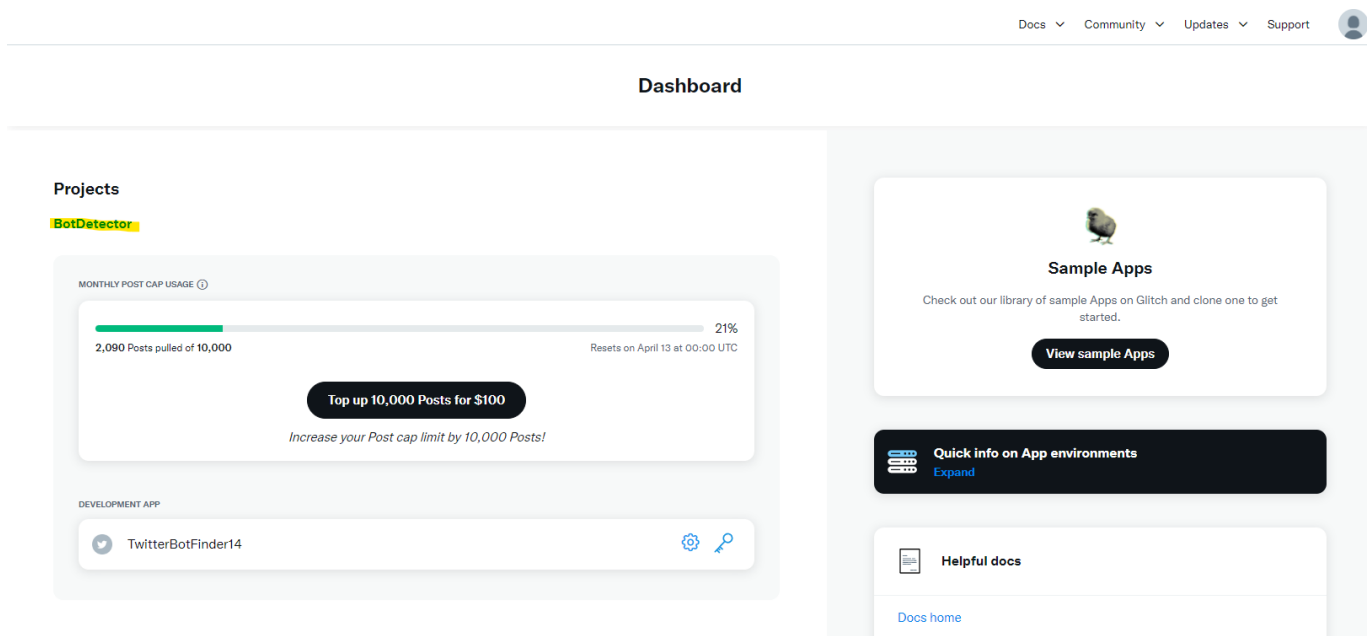
Free	Basic	Pro	Enterprise
For write-only use cases and testing the X API	For hobbyists or prototypes	For startups scaling their business	For businesses and scaled commercial projects
<ul style="list-style-type: none"> <li>Rate limited access to v2 post posting and media upload endpoints</li> <li>1,500 Posts per month - posting limit at the app level</li> <li>1 app ID</li> <li>Login with X</li> <li>Free</li> </ul>	<ul style="list-style-type: none"> <li>Rate limited access to suite of v2 endpoints</li> <li>3,000 Posts per month - posting limit at the user level</li> <li>50,000 Posts per month - posting limit at the app level</li> <li>10,000 Posts per month - read-limit rate cap</li> <li>2 app IDs</li> <li>Login with X</li> <li>\$100 per month</li> </ul>	<ul style="list-style-type: none"> <li>Rate-limited access to suite of v2 endpoints, including search and filtered stream</li> <li>1,000,000 Posts per month - GET at the app level</li> <li>300,000 Posts per month - posting limit at the app level</li> <li>3 app IDs</li> <li>Login with X</li> <li>\$5,000 per month</li> </ul>	<ul style="list-style-type: none"> <li>Commercial-level access that meets your and your customer's specific needs</li> <li>Managed services by a dedicated account team</li> <li>Complete streams: replay, engagement metrics, backfill, and more features</li> <li>Monthly subscription tiers</li> </ul>
Get started	Subscribe now	Subscribe now	Apply now

With the free tier not providing enough data to collect in terms of posts and a smaller limit, the team was forced to purchase the basic tier for approximately \$100 USD (or around \$130 Canadian) for access. The pro version is much too expensive with the number of tweets able to pull from the endpoint being considerably more than the basic tier, however with the price point it was not worth it and the basic tier was seen as more suitable for the needed outcome.

	Free	Basic	Pro
Getting access	<a href="#">Get Started</a>	<a href="#">Get Started</a>	<a href="#">Get Started</a>
Price	Free	\$100/month	\$5000/month
Access to X API v2	✓ (Only post creation)	✓	✓
Access to standard v1.1	✓ (Only Media Upload, Help, Rate Limit, and Login with X)	✓ (Only Media Upload, Help, Rate Limit, and Login with X)	✓ (Only Media Upload, Help, Rate Limit, and Login with X)
Project limits	1 Project	1 Project	1 Project
App limits	1 App per Project	2 Apps per Project	3 Apps per Project
post caps - Post	1,500	3,000	300,000
post caps - Pull	✗	10,000	1,000,000
Filteres stream API	✗	✗	✓

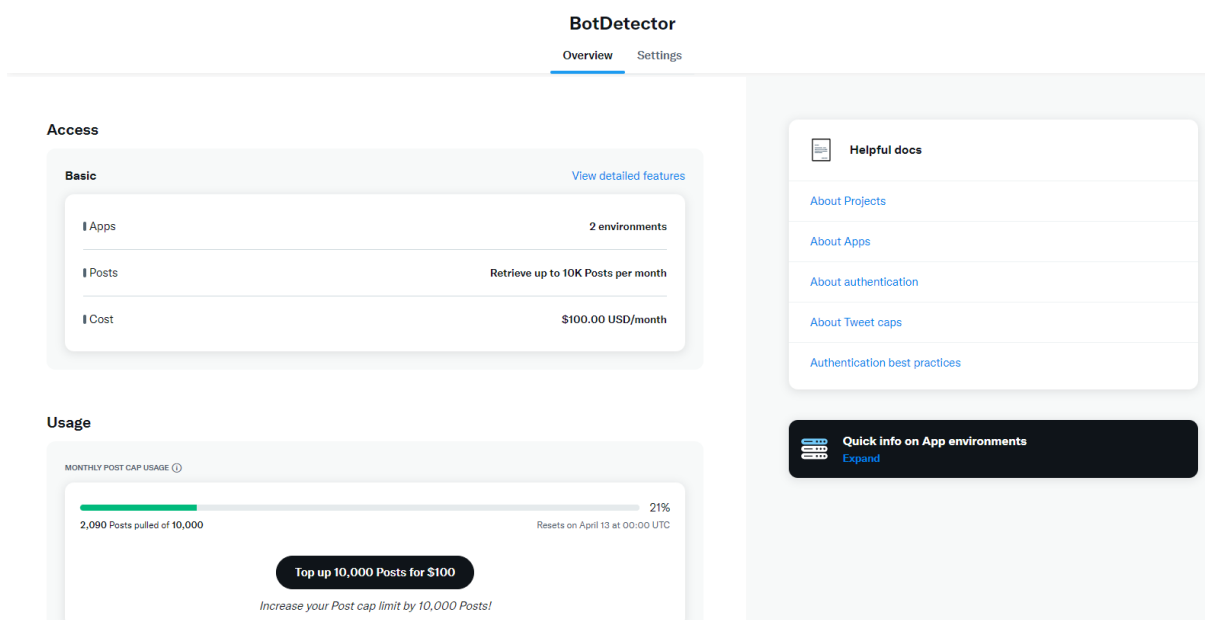
## Twitter Developer Account

The first part before being able to use twitter API for a project is to make a twitter developer account. This account is a self user interface within developer.x.com where developers can manage their API access through projects and applications. Projects create the permissions for allowing Apps to connect to X API v2 endpoints. Only Apps within Projects can access X API v2. Projects each have a X volume consumption limit for any connected Apps within the Project on a monthly basis Here, we can see our existing projects being worked on with this case being the bot detector.




A Project allows a user to organise their work with the X Developer Platform’s APIs, and are required when making requests to the X API v2 endpoints. You are also limited to being able to retrieve a certain number of Posts per month from specific endpoints at the Project-level.



Here we can see the number of apps, amount of posts we are able to retrieve depending on the api plan with the usage, costs.



For user and app authentication, you can edit the App icon, App name, App description, your website URL, callback URLs/redirect URIs, terms of service URL, privacy policy URL, organisation name, organisation URL, and purpose or use case of the App.

DEVELOPMENT APP

 TwitterBotFinder14

BOTDETECTOR

TwitterBotFinder14

SettingsKeys and tokens


App details

Edit

NAME

TwitterBotFinder14

APP ICON



APP ID

28482402

DESCRIPTION

This information will be visible to people who've authorized your App

This app was created to use the X API.

ENVIRONMENT

Development

User authentication settings

User authentication not set up

Authentication allows users to log in to your App with Twitter. It also allows your App to make specific requests for authenticated users.

Set up

## Code/train model

The objective of this report is to present the results of our efforts in developing and comparing multiple models for Twitter bot detection. Our aim was to assess the accuracy of each model and gain insights into their predictive capabilities to ultimately identify the most effective approach.

### Training Parameters:

In our pursuit of distinguishing between human users and bots on Twitter, we meticulously selected several parameters extracted from the Twitter API. These parameters, while informative, required manual curation to ensure relevance and reliability in training our models. We incorporated the following key parameters:



- **Average Tweet Time:** We hypothesised that human users exhibit a characteristic reaction time in responding to tweets, whereas automated bots may respond more swiftly. Therefore, we included average tweet time as a distinguishing feature, with lower tweet times potentially indicating bot activity.
- **Contains Keyword:** By identifying specific financial-related keywords within tweets, we aimed to detect content associated with financial advertising, a common characteristic of bot-generated content.
- **Percentage Difference:** Bots often exhibit a skewed follower-following ratio, typically boasting a higher follower count than following count. This disparity, captured through percentage difference calculations, served as an additional discriminatory feature.

## Model and Accuracy:

We employed a variety of machine learning models to accommodate the diverse nature of our dataset and to explore different predictive algorithms. The dataset initially comprised 90 entries, with subsequent expansions aimed at improving model accuracy. The dataset was partitioned into 80% training data and 20% testing data for model evaluation. Below are the accuracies achieved by each model:

## Random Forest Classifier:

- **Description:** Random Forest is an ensemble learning method that constructs a multitude of decision trees during training and outputs the mode of the classes (classification) or the mean prediction (regression) of the individual trees.
- **Strengths:** Effective for handling high-dimensional datasets with a large number of features, robust against overfitting, and capable of capturing complex interactions and non-linear relationships between features.
- **Accuracy:** 62.5%

```
Model performance: Random Forest Classifier
Accuracy: 0.625

Classification Report:
              precision    recall  f1-score   support

   False      0.57      0.73      0.64        11
    True      0.70      0.54      0.61        13

   accuracy                0.62        24
  macro avg      0.64      0.63      0.62        24
 weighted avg      0.64      0.62      0.62        24

Confusion Matrix:
[[8 3]
 [6 7]]
```

## Logistic Regression:

- Description: Logistic Regression is a linear classification algorithm that models the probability of a binary outcome based on one or more predictor variables.
- Strengths: Simple and interpretable, efficient to train, and suitable for large-scale datasets.
- Accuracy: 58.3%

```
Model performance: Logistic Regression
Accuracy: 0.5833333333333334

Classification Report:
              precision    recall  f1-score   support

   False         0.54         0.64         0.58         11
    True         0.64         0.54         0.58         13

 accuracy          0.58         0.58         0.58         24
  macro avg         0.59         0.59         0.58         24
weighted avg         0.59         0.58         0.58         24

Confusion Matrix:
[[7 4]
 [6 7]]
```

## Gradient Boosting:

- Description: Gradient Boosting is an ensemble technique that builds a series of decision trees sequentially, focusing on minimising the loss function during training.
- Strengths: Capable of achieving high predictive accuracy, robust to overfitting, and effective for handling heterogeneous data types.
- Accuracy: 62.5%

```
Model performance: Gradient Boosting
Accuracy: 0.625

Classification Report:
              precision    recall  f1-score   support

   False         0.57         0.73         0.64         11
    True         0.70         0.54         0.61         13

 accuracy          0.62         0.62         0.62         24
  macro avg         0.64         0.63         0.62         24
weighted avg         0.64         0.62         0.62         24

Confusion Matrix:
[[8 3]
 [6 7]]
```

## Conclusion of between Model

In conclusion, our comparative analysis of various machine learning models for Twitter bot detection revealed promising results. While Random Forest Classifier and Gradient Boosting

exhibited comparable accuracies at 62.5%, Logistic Regression slightly trailed behind at 58.3%. Further refinements and optimizations are warranted to enhance the robustness and generalizability of our bot detection system. Additionally, ongoing dataset expansion and feature engineering efforts will be pivotal in improving model performance and ensuring its effectiveness in real-world scenarios.

## Dataset:

The dataset comprises several fields of data input for each Twitter account, including Username, FollowerCount, FollowingCount, ContainsKeywords, AverageTweetTime, HashtagUsage, PercentDifference, and a boolean classification label for Bot identification.

Username	FollowerCount	FollowingCount	ContainsKeywords	AverageTweetTime	HashtagUsage	PercentDifference	Bot
AM730Trai	80134	188	FALSE	-302.556	11	-99.7654	FALSE
RichQuack	342147	721	FALSE	-395.222	0	-99.7893	TRUE
BostonGro	656	138	TRUE	-105.667	0	-78.9634	TRUE
DAZN_CA	42437	734	FALSE	-9637.44	10	-98.2704	FALSE
Cryptoniar	5354	369	TRUE	-1667.56	0	-93.108	TRUE
GrokovichI	356	157	TRUE	-840.222	1	-55.8989	TRUE
ghulamhar	115	1061	TRUE	-27.5	0	822.6087	FALSE
CryptoStoc	1170	273	FALSE	0	0	-76.6667	FALSE
corruptedc	157	211	FALSE	0	0	34.3949	FALSE
andysuethi	0	0	TRUE	0	0	0	TRUE
haf_bot	72	4	FALSE	0	0	-94.4444	TRUE
pjstock28	76939	1075	TRUE	-43147.9	8	-98.6028	FALSE
Wealthsim	88305	11	TRUE	-12538.9	0	-99.9875	FALSE
JustinEnge	239	274	FALSE	0	0	14.64435	FALSE
cmScanne	1151	18	FALSE	-1792.33	0	-98.4361	TRUE
_Bitify	539	1	FALSE	-3600	0	-99.8145	TRUE
sasa_ghad	22824	3160	FALSE	-407.444	0	-86.1549	FALSE
kjosandhu	12	95	FALSE	0	0	691.6667	FALSE
Jashen619	45	182	FALSE	0	0	304.4444	FALSE
jhon7073	57	1	TRUE	-48.5556	6	-98.2456	TRUE
054_Reyha	37	230	TRUE	-340.556	2	521.6216	TRUE
muel808	40	1	TRUE	-27.1111	8	-97.5	TRUE
0cryptoB6	7	375	TRUE	-27.8889	2	5257.143	TRUE
CRYPTO_FI	18164	9	TRUE	-43817.4	44	-99.9505	TRUE
FT	5884278	1047	FALSE	-3569.56	0	-99.9822	FALSE
BoardIQ	292	47	TRUE	-86398	0	-83.9041	FALSE
100000000	5011	282	FALSE	-31500.0	0	-99.5375	TRUE

**Username:** Unique identifier for each Twitter account.

**FollowerCount:** Number of followers for each account.

**FollowingCount:** Number of accounts being followed by each user.

**ContainsKeywords:** Boolean value indicating whether the account's tweets contain specific keywords associated with bot activity. In this case keywords such as 'stocks' and 'crypto'.

**AverageTweetTime:** Average time gap between tweets for each account.

**HashtagUsage:** Frequency and relevance of hashtags used in tweets.

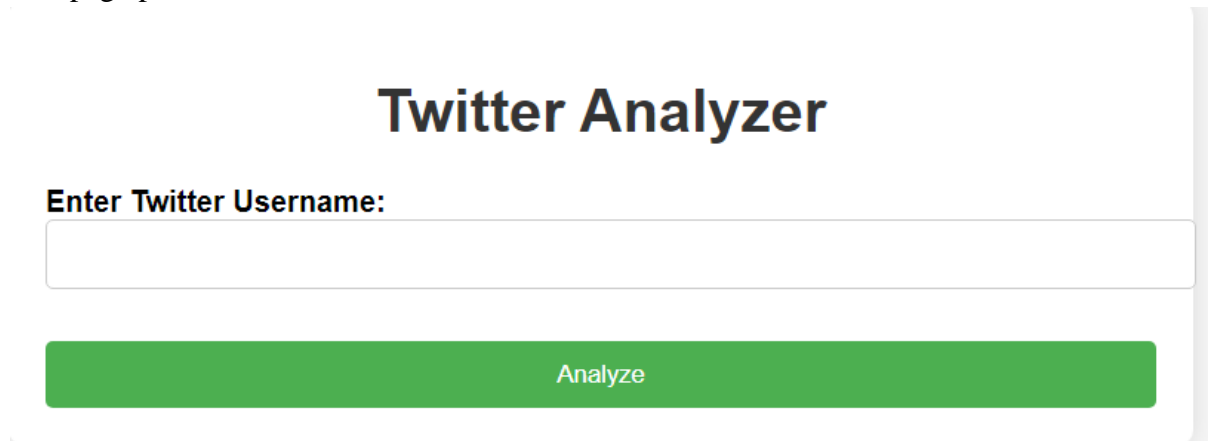
**PercentDifference:** Percentage difference between followers and following counts.

**Bot:** Boolean classification indicating whether the account is classified as a bot.(This require manual input before input into our models)

## React page/UI

For the UI of our project we used html to create a webpage that allows the user to enter a twitter user's username and click a button to fire off the data collection and machine learning code and display the results of what was retrieved and whether the account is likely a bot or not. We used react to host it and fire our code and display the results.

Webpage portal:



The image shows a web portal titled "Twitter Analyzer". Below the title, there is a label "Enter Twitter Username:" followed by a text input field. Below the input field is a green button with the text "Analyze".

## Data Collection

As part of this project, we developed a Python script to analyse data from Twitter. The aim was to gain insights into user behaviour and interests based on their tweets. By leveraging the Twitter API, we implemented various functionalities to retrieve, process, and analyse tweet data.

Functionality:

- Retrieving Tweets: The `get_user_tweets(username)` function allows us to fetch recent tweets from a specified user. This function utilises the Twitter API endpoint to access tweet data efficiently.
- Keyword Analysis: With the `contains_keywords(username)` function, we can determine if a user's tweets contain keywords related to finance. This analysis provides valuable insights into the user's interests and topics of discussion.
- Tweet Timing Analysis: The `get_average_tweet_time(username)` function calculates the average time between consecutive tweets of a user. This metric offers an understanding of the user's tweeting frequency and engagement patterns.
- User Metrics Retrieval: Using the `get_user_data(username)` function, we retrieve public metrics data of a user, such as follower and following counts. These metrics help in understanding the user's social influence and network size.

- **API Usage:**  
The project heavily relies on the Twitter API for accessing tweet data and user metrics. The use of API keys (BEARER\_TOKEN) ensures secure authentication and access to Twitter's resources, enabling seamless data retrieval and analysis.
- **Finance Keyword Detection:** By analysing the presence of finance-related keywords in a user's tweets, we gain insights into their financial interests and discussions. To ensure robustness and reliability, the code incorporates error handling mechanisms. This includes checking for successful API responses and handling exceptions gracefully, thereby enhancing the overall stability of the application.

In conclusion, this project demonstrates the capability of leveraging the Twitter API to analyse user behaviour and interests on the platform. By combining data retrieval, processing, and analysis techniques, **valueTweet Timing Insights:** Calculating the average time between tweets allows us to understand the user's tweeting habits and engagement frequency on the platform.

## **Challenges**

Our biggest challenge was figuring out what data we could retrieve from the X API. It was hard to find reliable research on what we can collect because twitter turned into X and lots of features were removed and not correctly documented on. There were certain data points we wanted to include such as whether the user was verified, however the api only returns the old verified checkmark data which is no longer used, and the new ones are not included in it yet. So the change of companies caused issues in what was available and what documentation was available.

## **Conclusion**

Overall, this project showcased a variety of factors like machine learning, model training, twitter api and the importance of api in application data analyses. Also with the familiarity of React and how to build user interfaces. With the main topic that was focused on bots on social media, primarily related to financial and crypto and how rampant they can be on twitter/X. We got to see first hand how bots can be influenced to change and control the algorithm to boost certain topics by fake engagement. More companies, like meta and tiktok are cracking down on bots and the quicker twitter/X deals with the issue the better it will be for the consumer experience.

## **REFERENCES**

<https://developer.twitter.com/en/docs/twitter-api>  
<https://developer.twitter.com/en/support/x-api/developer-account1>