# A SYNOPSIS ON

**TWIBOT-SPOTTER USING MACHINE LEARNING**

# Submitted in partial fulfillment of the requirements for the awards of degree of

# **BACHELOR OF ENGINEERING**

**IN**

# **(COMPUTER SCIENCE & ENGINEERING)**

****

***Submitted By***

**(Group no: 6)**

**Ritika Telang**

**Vishal Dhandare**

**Pawankumar Tummawar**

UNDER THE GUIDANCE OF

**PROF.SHAILENDRA SHENDE**

# **DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING GOVERNMENT COLLEGE OF ENGINEERING, CHANDRAPUR**

# **Year 2022-2023**

**ABSTRACT**

Twitter is a popular social media platform that is often used for communication and marketing purposes. However, the presence of Twitter bots, which are automated accounts that mimic human behavior, can pose a significant challenge to the platform's integrity. In this project, we propose a method for detecting Twitter bots using machine learning techniques. We gather data from datasets, extract relevant features, and use them to train a classifier that can differentiate between bots and genuine user accounts. Our approach involves the use of a variety of features, including temporal patterns, account metadata, and content-based features. We evaluate the performance of our method on a large dataset of Twitter accounts and show that it is effective in identifying bots with high accuracy. Our work contributes to the development of tools and methods for ensuring the integrity of social media platforms and protecting users from malicious activity. The presence of Twitter bots is a growing concern for the platform and its users. Bots can be used for a variety of purposes, including spreading false information, amplifying certain voices, and engaging in spam or other forms of malicious activity. As such, there is a need for effective methods for detecting and identifying bots in order to maintain the integrity and trustworthiness of the platform.

**INDEX**

|  |  |  |
| --- | --- | --- |
| Chapters | Content | Page No. |
| 1. | Introduction | 4 |
| 2. | Literature Survey | 6 |
| 3. | Problem Definition | 8 |
| 4. | Objectives | 10 |
| 5. | Methodology  5.1 Role of Machine Learning  5.2 How Machine Learns | 12 |
| 6. | Application  6.1 Why Not KNN  Algorithm?  6.2 Why Not Decision Tree  Algorithm?  6.3 Why Random Forest  Algorithm? | 19 |
| 7. | Software Requirements | 27 |
| 8. | Code Snippets | 29 |
| 9. | Reference |  |
| 10. | Conclusion |  |

CHAPTER-I

INTRODUCTION

**INTRODUCTION**

Social media platforms have become an integral part of our daily lives, enabling us to communicate, share information and opinions, and connect with people from around the world. However, with the increasing popularity of social media comes the growing problem of automated accounts, also known as bots that mimic human behavior and can manipulate the online discourse.

Twitter is one of the most popular social media platforms and is often used for communication, marketing, and information dissemination. However, the presence of Twitter bots poses a significant challenge to the platform's integrity and user trust. Twitter bots can be used to spread false information, amplify certain voices, and engage in other malicious activities, which can harm both individual users and the platform as a whole.

In response to this challenge, there is a need for effective methods for detecting Twitter bots. In this project, we propose a machine learning-based approach for identifying bots on Twitter, using a variety of features such as temporal patterns, account metadata, and content-based features. Our method aims to differentiate between bots and genuine user accounts with high accuracy, contributing to the development of tools and methods for ensuring the integrity of social media platforms and protecting users from malicious activity.

CHAPTER-II

LITERATURE SURVEY

**LITERATURE REVIEW ON TWITTER BOT DETECTION**

**Twitter Bot Detection with Reduced Feature Set**

**BY- Jefferson Viana Fonseca Abreu, Celia Ghedini Ralha:-** The author observed that the detection of bots for malicious activities on ONS. The author observed that the detection of bots from twitter did not involve linguistic analysis but observed not-human behaviors in parameters such as frequency and continuity of tweets, duration of activity period and the like. It is a simpler approach to not to affect user experience or to consume computational resources. This approach does not conflict with interest of apparent desire to keep bots active. However, these behavioral features can identify bot in a simple way.

**A Comprehensive Twitter Bot Detection Benchmark**

**BY- Shangbin Feng, Herun Wan, Ningnan Wang Jundong Li, Minnan Luo:-** . We collected and annotated Twitter data to present a comprehensive Twitter bot detection benchmark TwiBot20, which is representative of the diversified Twitter sphere and captures different types of bots that co-exist on major social media platforms. We make TwiBot-20 public, hoping that it would alleviate the lack of comprehensive datasets in Twitter bot detection and facilitate further research. Extensive experiments demonstrate that state-of-the-art bot detectors fail to match their previously reported performance on TwiBot-20, which shows that Twitter bot detection is still a challenging task and demands continual efforts. In the future, we plan to study novel Twitter bots and propose robust bot detectors.

# **Detecting malicious activity in Twitter using deep learning techniques**

# **BY- Loukas Llias, Loanna Roussaki:-**  A thorough analysis over the features used has been presented. In the latter case, the usage of an attention mechanism led to a substantial improvement with regards to the evaluation metrics. Via series of experiments conducted, it has been demonstrated that the proposed approaches clearly outperform the related state-of-the-art results. More specifically, in the first approach, after conducting a comparison among several feature selection and [dimensionality reduction techniques](https://www.sciencedirect.com/topics/engineering/dimensionality-reduction-technique), the techniques that lead to efficient and stable features that boost the performance of machine learning classifiers have been identified and selected.

CHAPTER – III

PROBLEM DEFINITION

**PROBLEM DEFINITION**

* The problem of detecting Twitter bots has become increasingly important in recent years due to the growing use of social media platforms for communication, marketing, and information dissemination. The **presence of bots on Twitter can pose a significant threat to the platform's integrity and user trust**, as they can be used to spread false information, manipulate public opinion, and engage in spam or other forms of malicious activity.
* The detection of Twitter bots is important for several reasons. Firstly, it can **help to protect individual users from potential harm**, such as being exposed to false information or scams. Secondly, it can help **to maintain the integrity of the platform by removing accounts that engage in malicious activity** and ensuring that users can trust the information they see on the platform. Lastly, it can help to improve the overall user experience by reducing the amount of spam and irrelevant content that users encounter on the platform.
* In light of these concerns, the development of effective methods for detecting Twitter bots is crucial. Machine learning-based approaches have shown promise in this area, and **the development of accurate and efficient Twitter bot detectors can help to ensure the continued success and sustainability of social media platforms** such as Twitter.

CHAPTER-IV

OBJECTIVES

**OBJECTIVES**

* To develop a machine learning-based approach for detecting Twitter bots that uses a combination of features including followers count, following count, verified accounts, following-to-followers ratio, and account details.
* To collect a dataset of Twitter accounts that includes both bots and genuine user accounts, and to label these accounts for use in training and testing the bot detector.
* To experiment with different machine learning algorithms and feature sets to identify the most effective approach for detecting bots on Twitter.
* To evaluate the performance of the bot detector in terms of precision, recall, and F1 score, and to compare its performance with other existing bot detection methods.
* To identify the most important features for detecting Twitter bots, and to explore how the performance of the bot detector is affected by the inclusion or exclusion of different features.
* To analyze the characteristics of detected bots and to gain insights into the types of malicious activity that bots engage in on Twitter.
* To explore the potential applications of the bot detector in identifying and mitigating the impact of bots on social media platforms, and to identify areas for future research in this field.

CHAPTER-V

METHODOLOGY

**METHODOLOGY**

5.1 Role of Machine Learning

5.2 How Machine Learns

**5.1 Role of Machine Learning**

The role of machine learning in Twitter bot detection is essential. Social media platforms like Twitter generate enormous amounts of data, which can be analysed using machine learning algorithms to identify patterns and characteristics of bot behaviour.

Machine learning algorithms can be trained on large datasets of labelled Twitter accounts, learning to recognize patterns and features that are associated with bot behaviour. These algorithms can then be used to classify new Twitter accounts as either bots or genuine users based on their characteristics and behaviours.

Machine learning algorithms can analyse a variety of features such as follower count, following count, verified accounts, following-to-follower ratios, and account details. By analysing these features, machine learning algorithms can identify patterns and correlations that may be indicative of bot behaviour. In addition, machine learning algorithms can adapt and improve over time, making them well-suited for detecting new and evolving types of bot behaviour.

Machine learning algorithms can also be used to identify bot networks and determine the extent to which they influence social media conversations. This is important because bots can be used to spread false information, manipulate public opinion, and engage in spam or other forms of malicious activity. By identifying and removing accounts that engage in malicious activity, machine learning algorithms can help maintain the integrity of the platform and ensure that users can trust the information they see on the platform.

**5.2 How Machine Learns**

In machine learning, machines learn by identifying patterns and relationships in data. The process of learning can be divided into three main stages:

**1. Data processing and preparation**: In this stage, raw data is collected and pre-processed to make it ready for analysis. This may involve tasks such as cleaning, formatting, and transforming the data into a format that can be used by machine learning algorithms.

**2**. **Model building and training**: In this stage, machine learning algorithms are applied to the pre-processed data to create models that can learn from the data. The models are trained on the data using various techniques such as supervised or unsupervised learning, reinforcement learning, and deep learning.

**3.** **Model evaluation and optimization**: In this stage, the trained models are evaluated on new data to measure their accuracy and performance. The models are then optimized by tweaking their parameters and hyper-parameters to improve their performance on new data.

The key to machine learning is **the ability of the algorithms to learn from data without being explicitly programmed**. This is achieved through the use of mathematical techniques such as optimization, regression, clustering, and classification. These algorithms are designed to identify patterns and relationships in the data, and use this information to make predictions or decisions.

Machine learning algorithms can be broadly classified into two categories: **supervised learning and unsupervised learning**. In **supervised learning**, the algorithm is **provided with a labelled dataset**, where the output variable (or target variable) is known for each data point. The algorithm learns to make predictions based on the input variables and their corresponding labels. Examples of supervised learning algorithms include linear regression, decision trees, and neural networks.

In **unsupervised learning**, the **algorithm is provided with an unlabelled dataset**, where the output variable is not known. The algorithm learns to identify patterns and relationships in the data based on the input variables alone. Examples of unsupervised learning algorithms include clustering, dimensionality reduction, and anomaly detection.

In summary, machine learning algorithms learn by identifying patterns and relationships in data through the **use of mathematical techniques such as optimization, regression, clustering, and classification**. The ability of machines to learn from data without being explicitly programmed is what makes machine learning such a powerful tool for solving complex problems.

CHAPTER-VI

APPLICATION

**APPLICATION**

6.1 Why Not KNN Algorithm?

6.2 Why Not Decision Tree

Algorithm?

6.3 Why Random Forest Algorithm?

**6.1 Why Not KNN Algorithm?**

There are several reasons why Random Forest algorithm may give higher accuracy than K-Nearest Neighbors (KNN) algorithm:

**1.** Random Forest is an ensemble learning method that builds multiple decision trees and combines their predictions to generate a final prediction. This technique reduces the risk of overfitting and improves the generalization performance of the model. On the other hand, **KNN is a lazy learning method that does not build a model**, but instead stores the entire training set and calculates distances between data points at test time. This can lead to overfitting if the training set is small or if there are many irrelevant features.

**2.** Random Forest is more robust to noisy data. **KNN is sensitive to noisy data, which means that outliers or misclassified instances can greatly affect the performance of the algorithm**. Random Forest can handle noisy data well because it builds multiple decision trees and combines their predictions, reducing the impact of individual noisy instances.

**3.** Random Forest can handle high-dimensional data. **KNN suffers from the curse of dimensionality, which means that as the number of features increases, the distance between data points becomes less meaningful.** Random Forest can handle high-dimensional data well because it builds multiple decision trees and selects only the most important features, reducing the impact of irrelevant features.

**4.** Random Forest can handle imbalanced datasets. In Twitter bot detection, the dataset is likely to be imbalanced, with fewer bots than legitimate accounts. **KNN can be biased towards the majority class, leading to poor performance on the minority class**. Random Forest can handle imbalanced datasets well because it uses techniques such as bootstrapping and feature subsampling to balance the classes.

**5.** Random Forest is less sensitive to the choice of hyper parameters. **KNN has several hyper parameters that need to be tuned, such as the number of neighbors and the distance metric**. Random Forest has fewer hyper parameters and is less sensitive to their choice, making it easier to use in practice.

When we processed our testing data on **KNN algorithm** our **accuracy** was less than the accuracy of Random Forest Algorithm which is **47.69%** in terms of **precision, recall, f1-score, support.**



**6.2 Why Not Decision Tree Algorithm?**

There are several reasons why Random Forest algorithm may give higher accuracy than Decision Tree algorithm:

**1.** **Decision Trees can often overfit the training data, meaning that they become too complex and do not generalize well to new data**. Random Forest, on the other hand, uses multiple decision trees and combines their results to produce a more accurate prediction, which reduces the risk of overfitting.

**2.** **Random Forest can generalize better than Decision Tree** because it uses multiple decision trees to make predictions, and thus it is more likely to capture the underlying patterns in the data.

**3.** **Random Forest can handle missing data, whereas Decision Tree cannot**. Random Forest can make use of the available data to make accurate predictions, even if some data is missing.

**4. Decision Tree is biased towards the features that are most important in the data**. Random Forest reduces this bias by randomly selecting a subset of features at each node.

**5. Random Forest provides a measure of feature importance, which can be useful in feature selection and feature engineering**. This allows you to identify the most important features in the data, which can lead to higher accuracy in the model.

When we processed our testing data on **Decision Tree algorithm** our **accuracy** was less than the accuracy of Random Forest Algorithm which is **68.43%** in terms of **precision, recall, f1-score, support.**



**6.3 Why Random Forest Algorithm?**

Random Forest algorithm can be a good choice for Twitter bot detection because it has several advantages over other algorithms, including:

**1. High accuracy:** Random Forest algorithm can provide high accuracy in detecting Twitter bots. This is because it uses multiple decision trees and combines their results to produce a more accurate prediction.

**2. Handles noisy data:** Twitter data can be noisy, with a lot of irrelevant information and spam. Random Forest algorithm is less sensitive to noisy data than some other algorithms, which can help improve accuracy.

**3. Handles missing data:** Twitter data can also contain missing information, such as missing follower or following counts. Random Forest algorithm can handle missing data better than some other algorithms, which can improve accuracy.

**4. Provides feature importance:** Random Forest algorithm provides a measure of feature importance, which can be useful in feature selection and feature engineering. This allows you to identify the most important features in the data, which can lead to higher accuracy in the model.

**5. Scalability:** Random Forest algorithm can handle large datasets with many features, which can be useful for Twitter bot detection, as there are many features to consider in identifying bots.

When we processed our testing data on **Random Forest Algorithm** we got the highest accuracy of **72.86%** in terms of **precision, recall, f1-score, support.**



CHAPTER-VII

SOFTWARE REQUIREMENTS

**SOFTWARE REQUIREMENTS**

* Python
* Python library- Scikit (to fetch

Algorithms)

* Python library- Pandas (to perform

mathematical operations)

* ML algorithms(KNN, Decision

Tree, Random Forest)

* HTML, CSS (frontend)
* Flask (backend)

CHAPTER-VIII

CODE SNIPPETS

HTML

<! DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<!-- <link rel="stylesheet" type="text/css" href="{{ url\_for('static', filename='style.css') }}"> -->

<title>Twitter bot Detection</title>

<style>

body {

position: relative;

margin: 0;

padding: 0;

}

body::after {

content: "";

background-image: url("https://gifdb.com/images/file/network-dynamic-installations-gkxexxrw8wls9zpb.gif");

position: absolute;

top: 0;

left: 0;

width: 100%;

height: 100%;

opacity: 0.9;

z-index: -1;

}

.main{

height: 100vh;

margin: 0 auto;

color: #ae00ff;

}

.box{

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

width: 800px;

height: 400px;

max-width: 1000px; /\* Set the maximum width of the box \*/

background-color: #fff;

box-shadow: 0 1px 3px rgba(255, 255, 255, 0.12), 0 1px 2px rgba(0,0,0,0.24);

transition: all 0.3s cubic-bezier(.25,.8,.25,1);

border-radius: 30px;

}

.box:hover{

box-shadow: 0 14px 28px rgba(255, 254, 254, 0.25), 0 10px 10px rgba(255, 255, 255, 0.22);

}

/\* style heading \*/

.heading {

padding-top: 30px;

text-align: center;

}

/\* style form \*/

.grid {

display: grid;

grid-template-columns: 1fr;

grid-row-gap: 20px;

justify-items: center;

align-items: center;

margin-top: 50px;

}

label {

font-size: 20px;

}

input[type="text"] {

font-size: 16px;

padding: 10px;

border-radius: 5px;

border: 1px solid #ccc;

}

.file\_submit {

font-size: 18px;

padding: 10px 20px;

background-color: #4CAF50;

color: #fff;

border: none;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.4s;

}

.file\_submit:hover {

background-color: #fc0000;

color: black;

box-shadow: #181818;

}

</style>

</head>

<body>

<div class="main">

<div class="box">

<div class="heading">

<h1>Twitter Bot Detection</h1>

</div>

<form class="grid" action="/result" method="POST">

<label for="bool\_input">Enter a Twitter account name: </label>

<input type="text" name="account\_name" id="account\_name" placeholder="Enter the username">

<center><input type="submit" class="file\_submit"></center>

<center>

{% if message %}

<h3 style="color:rgb(0, 0, 0)">@{{account\_name}} {{message}}</h3>

{% endif%}

</center>

</form>

</div>

</div>

</body>

</html>

CSS

body {

position: relative;

margin: 0;

padding: 0;

}

body::after {

content: "";

background-image: url("https://gifdb.com/images/high/network-dynamic-installations-gkxexxrw8wls9zpb.gif");

position: absolute;

top: 0;

left: 0;

width: 100%;

height: 100%;

opacity: 0.9;

z-index: -1;

}

.main{

height: 100vh;

margin: 0 auto;

color: #ae00ff;

}

.box{

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

width: 800px;

max-width: 1000px; /\* Set the maximum width of the box \*/

background-color: #fff;

box-shadow: 0 1px 3px rgba(255, 255, 255, 0.12), 0 1px 2px rgba(0,0,0,0.24);

transition: all 0.3s cubic-bezier(.25,.8,.25,1);

border-radius: 30px;

}

.box:hover{

box-shadow: 0 14px 28px rgba(255, 254, 254, 0.25), 0 10px 10px rgba(255, 255, 255, 0.22);

}

/\* style heading \*/

.heading {

padding-top: 30px;

text-align: center;

}

/\* style form \*/

.grid {

display: grid;

grid-template-columns: 1fr;

grid-row-gap: 20px;

justify-items: center;

align-items: center;

margin-top: 50px;

}

label {

font-size: 20px;

}

input[type="text"] {

font-size: 16px;

padding: 10px;

border-radius: 5px;

border: 1px solid #ccc;

}

.file\_submit {

font-size: 18px;

padding: 10px 20px;

background-color: #4CAF50;

color: #fff;

border: none;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.4s;

}

.file\_submit:hover {

background-color: #fc0000;

color: black;

box-shadow: #181818;

PYTHON

from flask import Flask, render\_template, request

import pandas as pd

from sklearn.ensemble import RandomForestClassifier

# from sklearn.neighbors import KNeighborsClassifier

# from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

from sklearn.feature\_extraction.text import TfidfVectorizer

app = Flask(\_name\_)

print('collecting the data')

# Step 1: Collect Data

df = pd.read\_csv('twitter\_data.csv')

df = df.drop(['id\_str', 'location', 'url', 'description', 'status', 'default\_profile', 'default\_profile\_image', 'has\_extended\_profile'], axis=1)

# performing feature engineering on followers\_count and friends\_count columns

df['followers\_count'] = df.followers\_count.apply(lambda x: 0 if pd.isnull(x) else int(x))

df['friends\_count'] = df.friends\_count.apply(lambda x: 0 if pd.isnull(x) else int(x))

df['following\_to\_followers\_ratio'] = df['friends\_count'] / df['followers\_count']

train\_df = df.copy()

# performing feature engineering on id and verified columns

# converting id to int

train\_df['id'] = train\_df.id.apply(lambda x: int(x))

# converting verified into vectors

train\_df['verified'] = train\_df.verified.apply(lambda x: 1 if ((x == True) or x == 'TRUE') else 0)

# followers are more than following --> then it is most probably a bot

condition = ((train\_df.following\_to\_followers\_ratio > 10)) # these all are bots

# converted condition datatype from boolean to int form for easy calculation

train\_df['bot'] = condition.astype(int)

# Step 3: Transform Data from text to number format

vectorizer = TfidfVectorizer(stop\_words='english')

df['screen\_name'] = df['screen\_name'].fillna('')

X = vectorizer.fit\_transform(df['screen\_name']).toarray()

y = df['bot']

print('training the data')

# Step 4: Train Model

# rf = KNeighborsClassifier(n\_neighbors=3)

# rf = DecisionTreeClassifier()

rf = RandomForestClassifier(n\_estimators=100, random\_state=42)

rf.fit(X, y)

#calculating accuracy

# accuracy = accuracy\_score(X, y)

# print('Accuracy:', accuracy)

# API route

@app.route("/")

@app.route("/checkbot", methods=['GET'])

def checkbot():

return render\_template("index.html")

@app.route("/result", methods = ['POST', 'GET'])

def result():

output = request.form.to\_dict()

account\_name = output["account\_name"]

# Preprocess and transform account description into numerical format

preprocessed\_description = vectorizer.transform([account\_name]).toarray()

# Predict label using trained model

result = rf.predict(preprocessed\_description)

if result == 1:

message = " is a bot"

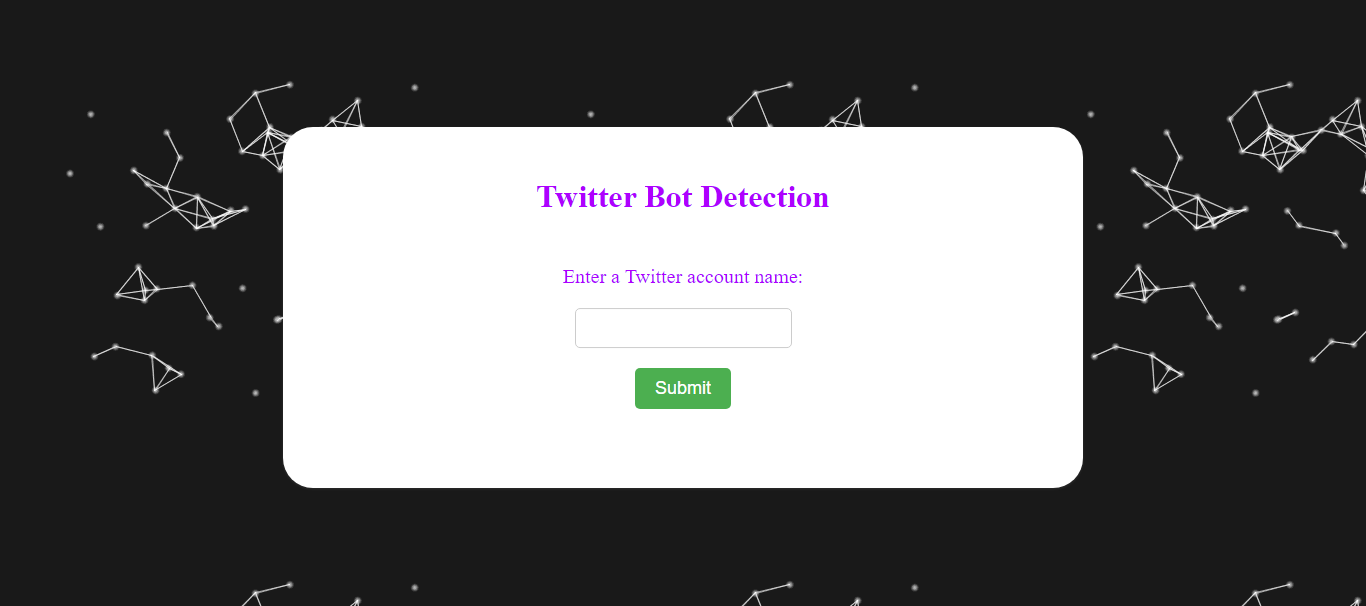
else:

message = " is not a bot"

return render\_template("index.html", message=message, account\_name=account\_name)

if \_name\_ == "\_main\_":

app.run(debug=True)



CHAPTER-IX

REFERENCE

**REFERENCE**

* Custom classification algorithm to sense the bots vs. human on social media space like <https://github.com/topics/twitter-bot-detection>
* https://github.com/jubins/MachineLearning- Detecting-Twitter-Bots
* https://www.ijsr.net/archive/v8i7/ART20199245. pdf (International Journal of Science and Research (IJSR))
* <https://www.kaggle.com/>
* <https://iopscience.iop.org/article/10.1088/1742-6596/1950/1/012006>
* <https://scholar.umw.edu/cgi/viewcontent.cgi?article=1025&context=student_research>

CHAPTER-X

CONCLUSION

**CONCLUSION**

In conclusion, Random Forest algorithm can be an effective tool for Twitter bot detection. It can provide high accuracy, handle noisy and missing data, and provide feature importance, making it useful for feature selection and engineering. Additionally, it is scalable and can handle large datasets with many features. In comparison to other algorithms such as KNN and decision tree, Random Forest has shown to outperform them in terms of accuracy. Therefore, for the task of Twitter bot detection, Random Forest algorithm can be a reliable and efficient choice.

However, further research can be done to explore other machine learning algorithms and feature engineering techniques to improve the accuracy of the bot detection system.

Another advantage of Random Forest algorithm is that it can handle imbalanced datasets, which is a common issue in bot detection. Bots are often a minority class, making it difficult to accurately predict them. Random Forest algorithm can address this issue by oversampling the minority class or using other techniques such as weighted sampling.