FakeFinder: Social Media Fake Account Detection Using Machine Learning

PROJECT REPORT

Submitted by

GEO JOSEPH (KTE21MCA-2019)

Department of Computer Applications
RAJIV GANDHI INSTITUTE OF TECHNOLOGY
KOTTAYAM - 686 501

May 2023

DECLARATION

I undersigned hereby declare that the project report **Social Media Fake Account Detection Using Machine Learning**, submitted for partial fulfilment of the requirements for the award of degree of Master of Computer Applications of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by me under supervision of **Dr. Vineetha S**. This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Kottayam

May 2023 Geo Joseph

ACKNOWLEDGMENT

I wish to thank the multitude of people who have helped me during the course of the MCA. First of all, I thank God almighty for His grace and blessings for without his unforeseen guidance this would have remained in dreams.

I express my sincere gratitude to **Dr. Jalaja M. J**, Principal, Rajiv Gandhi Institute of Technology, Kottayam, for providing the ambiance for carrying out the work of this project.

I deeply indebted to **Dr. Reena Murali**, HOD and Professor, Department of Computer Applications, for providing permission and availing all required facilities for undertaking the project in a systematic way.

I feel deeply honoured in expressing my sincere thanks to my project guide **Dr. Vineetha S**, Associate Professor, Department of Computer Applications for the constructive suggestions and inspirations that helped me during the preparation of this project.

I would like to express thanks to our project coordinator **Dr. Vineetha S / Dr. Sangeetha Jose**, Associate Professor, Department of Computer Applications, for his constant support and encouragement. I am also grateful to the Rajiv Gandhi Institute of Technology, Kottayam for allowing me to do this project and providing all the required facilities.

I take this opportunity to thank all teaching and non-teaching staffs of Department of Computer Applications for their help.

GEO JOSEPH

ABSTRACT

In the present era, social media platforms are widely used for conversations, sharing information, gaining knowledge, etc. As the number of new users on these platforms increases, the number of fake accounts also increases. This increasing number of fake accounts has become a significant challenge to the credibility and reliability of social media networks. Verifying these accounts manually is difficult and time-consuming.

The proposed system FakeFinder is a web app that uses machine learning algorithms to detect if an account is fake or not. Users can verify the genuinity of an account by providing the link of suspicious account to the application. FakeFinder fetches the details of the account using web scrapping, converts the data and passes it to machine learning models. Ensemble learning method is utilized where outputs from Random Forest, LSTM and KNN models are averaged. A dataset containing details of both real and fake accounts is used to train and validate the models. The language used is Python and Django framework is used as the backend.

Based on the results generated by FakeFinder appropriate actions such as account reporting or blocking can be initiated by users. The results will contribute to enhancing the credibility and trustworthiness of social media platforms and prevents fake accounts from spreading misinformation and spam. The proposed system will also help social media companies to identify and remove fake accounts quickly, improving the overall user experience and trust in their platforms.

Keywords : Social Media, Fake Accounts, Random Forest, LSTM, KNN

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

A large number of people are victims of attacks from fake social media accounts. The absence of a proper system for detecting and verifying fake accounts for the public has led to an increase in these accounts and attacks.

1.1 GENERAL BACKGROUND

The popularity of social media platforms has increased drastically over time. People now rely heavily on these platforms for communication, information sharing and gathering, entertainment, and more. While these platforms are highly useful, fake accounts can create negative effects. Fake accounts can be created for a variety of reasons, including spreading misinformation, conducting social engineering attacks, influencing political opinions, and engaging in fraudulent activities. Such accounts can also compromise the privacy and security of legitimate users by spreading malware or collecting personal information.

Manually verifying an account can be difficult and time consuming. There are also machine learning based systems but they produce only an average accuracy in detecting fake accounts and are also limited to social media platforms such as Twitter and Facebook. Less researches are being done on platforms such as Instagram.

Social media platforms may work internally to remove fake accounts based on user reports, but end-users lack methods for verifying the authenticity of suspicious accounts. The are no publicly available systems to check the authenticity of an account.

1.2 OBJECTIVE

Providing a public platform for verifying the authenticity of an account and obtaining better prediction results through machine learning models are the primary objectives of the proposed system, FakeFinder.

The need for verifying the authenticity of an account can be instantaneous in most of the situations. A public platform for verifying fake accounts will enable users to quickly know the state of the suspicious account. It will also enable them to take actions such as account reporting and blocking.

An ensemble machine learning model can improve the accuracy of the final result. Outputs from machine learning models such as Random Forest, LSTM and KNN are fed to ensemble model for finding the most accurate result.

Enabling users the ability to verify an account helps to increase security and thereby reduce the risk of fraudulent activities such as identity theft or scams. It will also contribute in improving the credibility and trustworthiness of social media platforms.

1.3 MOTIVATION

As the current systems does not provide any public platforms for the verification of fake accounts, FakeFinder can be a crucial tool for users to protect themselves and others from potential scams and frauds. With the increasing prevalence of fake accounts on social media, there is an urgent need for a reliable and accessible solution to detect and flag them. FakeFinder aims to fill this gap by providing a user-friendly and accurate platform for identifying suspicious profiles, contributing to a safer and more trustworthy online community.

FakeFinder utilizes ensemble learning models which includes Random Forest,

LSTM, and KNN, to achieve a high level of accuracy in detecting fake social media accounts. Thus, FakeFinder can provide users with a reliable and effective means of identifying suspicious profiles. Moreover, the use of machine learning enables FakeFinder to continually learn and improve its performance over time, ensuring that it remains an effective tool for the ongoing fight against fake accounts on social media platforms.

CHAPTER 2

LITERATURE REVIEW

Manually verifying an account's authenticity is difficult and time consuming. Also the authenticity cannot be determined by quick inspection as there are several aspects that relates to the genuineness of an account. Social media platforms internally conducts verification schemes based on the reports done by end users. But this does not enable users to manually verify any specific account. Machine learning methods use decision trees, SVM and neural networks which produce an average accuracy of 90% Other methods such as graph analysis, classification and data mining are also used. These systems are mainly focused on social media platforms such as Facebook and Twitter.

2.1 RELATED WORKS

Related works focus on machine learning techniques as well as graph analysis.

2.1.1 Identifying Fake Facebook Profiles Using Data Mining Techniques

This study provides methodology for detecting fake Facebook profiles using data mining techniques such as decision trees and random forests to identify suspicious profiles based on various features, such as profile information, activity patterns, and network structure. A dataset of over 2,500 real and fake Facebook profiles is used. The methodology uses various features to identify suspicious profiles, such as the number of friends, profile picture quality, and activity patterns. However, not all of these features may be available for all Facebook profiles, which could limit the overall effectiveness.

2.1.2 Identifying Fake Accounts on Social Networks Based on Graph Analysis and Classification Algorithms

This proposed approach utilizes both graph analysis and classification algorithms. The use of graph analysis can capture the structural patterns and relationships among users, which is helpful in identifying suspicious accounts. The study did not provide detailed information on the specific classification algorithms used, which can make it difficult to replicate or compare the results. The performance of the proposed approach may be affected by the quality and quantity of the available data, which can be a limitation.

CHAPTER 3

METHODOLOGY

FakeFinder is a web application that uses machine learning to detect fake accounts. Users can verify the genuineness of an account by providing the link of suspicious account to the application. FakeFinder fetches the details of the account using web scrapping, converts the data and passes it to machine learning models. Ensemble learning method is utilized where outputs from Random Forest, LSTM and KNN are averaged. A dataset containing details of both real and fake accounts is used to train and validate the models.

3.1 ANALYSIS OF DATASET

A publicly available dataset from Kaggle is used for training the models. The dataset contains details of about 700 fake and original accounts on Instagram. The attributes are :

1. Profile pic : User has profile picture or not
2. Nums/length username : Ratio of number of numerical chars in username to its length
3. Fullname words : Full name in word tokens
4. Nums/length fullname : Ratio of number of numerical characters in full name to its length
5. Name == username : Are username and full name literally the same

6. Description length : Bio length in characters
7. External URL : Has external URL or not
8. Private : Private account or not
9. Posts : Number of posts
10. Followers : Number of followers
11. Follows : Number of follows
12. Fake : Class (0 genuine, 1 spammer)

3.2 MACHINE LEARNING MODELS

FakeFinder uses ensemble learning technique where the output is the average of three machine learning models; ranndom forest, LSTM and KNN.

3.2.1 Random Forest

Random Forest is a machine learning algorithm that is commonly used for classification, regression, and other tasks. It is an ensemble method that combines multiple decision trees to improve accuracy and reduce overfitting.

The basic idea behind Random Forest is to build a large number of decision trees, each using a random subset of the training data and a random subset of the available features. During training, each tree is trained independently on a different subset of the data and features, and then the predictions of all the trees are combined to produce a final output. This process of combining multiple models is called ”bagging” and it helps to reduce the variance of the model.

In Random Forest, the decision trees are trained using a process called ”boot-

strapping". This involves randomly selecting subsets of the training data with replacement, so that each tree is trained on a slightly different set of data. The trees are also designed to be relatively uncorrelated, so that the errors made by one tree are not strongly correlated with the errors made by other trees.

To make predictions, new data is passed down each of the decision trees, and the predicted output of each tree is combined to form a final prediction. The final output can be a probability estimate or a class label, depending on the task.

Random Forest has several advantages over other machine learning algorithms. For example, it is robust to overfitting, works well with large datasets, and can handle both categorical and continuous features. It is also relatively easy to tune the hyper-parameters of a Random Forest model, which can help to improve its performance. Overall, Random Forest is a powerful and versatile machine learning algorithm that is widely used in many different domains.

3.2.2 LSTM (Long Short-Term Memory)

LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN) that is commonly used for sequence modeling and prediction tasks, such as speech recognition, natural language processing, and time series analysis. The main advantage of LSTM over traditional RNNs is its ability to learn long-term dependencies in sequential data. This is achieved through the use of memory cells and gating mechanisms that allow the network to selectively remember or forget information at each time step.

In an LSTM network, each cell has three main components: an input gate, a forget gate, and an output gate. These gates control the flow of information into and out of the cell, allowing it to selectively store or discard information as needed. At each

time step, the input gate decides which new information to let into the cell, based on the current input and the previous hidden state. The forget gate decides which information to discard from the cell, based on the current input and the previous hidden state. And the output gate determines which information to output from the cell, based on the current input and the current hidden state.

Through the use of these gates and memory cells, LSTM networks can effectively learn long-term dependencies in sequential data, which is often difficult or impossible for traditional RNNs. This makes them well-suited for a wide range of tasks, such as predicting the next word in a sentence, recognizing speech, or forecasting future values in a time series. LSTM has become a popular and widely used neural network architecture, and it has been applied successfully to many different domains, such as natural language processing, speech recognition, and image captioning. Its effectiveness in modeling sequential data has made it a key tool in many deep learning applications.

3.2.3 KNN (K-Nearest Neighbors)

KNN (K-Nearest Neighbors) is a simple and effective machine learning algorithm used for classification and regression tasks. It is a type of instance-based learning, where the model learns by comparing new data points to the labeled training examples and classifying them based on the nearest neighbors.

In the case of classification, given a new input data point, KNN looks for the k-nearest neighbors to that point in the training dataset. The value of k is a hyperparameter that is typically set based on empirical evaluation. The output class label for the new data point is determined by a majority vote of the k-nearest neighbors. In the case of regression, KNN works by averaging the values of the k-nearest neighbors to the new data point. One advantage of KNN is its simplicity and ease of implementation, and it

can work well on datasets with a small number of features. However, it may not perform as well on high-dimensional datasets with many features, as the distance metric used to calculate nearest neighbors can become less informative in higher dimensions.

3.3 SOFTWARE ENVIRONMENT

Python programming language is used along with Django as backend framework. Python libraries such as Keras, scikit-learn, beautifulsoup, pickle are also used.

3.3.1 Django

Django is a high-level web framework for building web applications quickly and efficiently. It is written in Python and follows the Model-View-Controller (MVC) architectural pattern. Django provides a lot of built-in functionality for common web development tasks, such as URL routing, form handling, authentication, and database management. It also supports various database systems, including PostgreSQL, MySQL, and SQLite.

One of the main advantages of Django is its focus on code reusability and modularity. Django applications are organized into reusable components called ”apps”, which can be easily integrated into other Django projects. This makes it easy to build complex web applications by combining pre-existing apps with custom code. Django also provides a powerful object-relational mapper (ORM) that allows developers to work with databases using Python objects rather than raw SQL. This makes it easier to work with databases and reduces the risk of SQL injection vulnerabilities.

Django also has a large and active community, with many third-party packages and plugins available to extend its functionality. Overall, Django is a powerful and

flexible web framework that makes it easy to build complex web applications quickly and efficiently.

3.3.2 Keras

Keras is a high-level neural network API written in Python, which is capable of running on top of multiple backends such as TensorFlow, CNTK, or Theano. It was developed with the goal of making deep learning accessible and easy to use for researchers and practitioners. Keras provides a user-friendly, modular, and extensible interface for defining and training various types of deep learning models, such as feedforward neural networks, convolutional neural networks (CNNs), recurrent neural networks (RNNs), and more. It supports both sequential and functional model architectures, and offers a range of layers, activations, loss functions, and optimizers to customize the model. Keras also provides a range of utilities and tools to aid in the preprocessing, evaluation, and visualization of data, and includes support for GPU acceleration to speed up training. One of the key benefits of Keras is its ease of use and flexibility. It provides a simple API that enables quick experimentation and prototyping of deep learning models, allowing researchers and practitioners to focus on developing and testing new ideas, rather than getting bogged down in the implementation details.

3.3.3 Scikit-learn

Scikit-learn is a popular choice for machine learning tasks due to its ease of use, flexibility, and comprehensive set of tools. With its intuitive and user-friendly API, users can easily experiment with a range of machine learning algorithms, perform data preprocessing and feature selection, and evaluate the performance of their models. It also provides a variety of utility functions for cross-validation, hyperparameter tuning, and model evaluation, making it easier to optimize and improve the performance of

machine learning models. Additionally, the library is built on top of NumPy, SciPy, and matplotlib, making it easy to integrate with other Python libraries and data analysis tools.

One of the key advantages of scikit-learn is its speed and scalability. The library is designed to be efficient and scalable, making it possible to work with large datasets and perform complex modeling tasks. Scikit-learn implements many algorithms in C, Cython, and Fortran, which provide fast and efficient computation. It also includes support for parallel processing, allowing users to take advantage of multi-core processors and distributed computing frameworks. With its emphasis on speed, scalability, and ease of use, Scikit-learn has become a go-to tool for machine learning practitioners in both research and industry.

3.3.4 BeautifulSoup

Beautiful Soup is a Python library used for web scraping purposes to extract data from HTML and XML files. It allows for parsing and navigating HTML and XML documents in a way that is easy to use and understand. Beautiful Soup can be used to extract information from web pages by locating specific tags and their attributes, and then extracting the content inside those tags. It can also be used to navigate the document tree to find specific elements and their relationships to other elements.

Beautiful Soup provides a number of functions for parsing and navigating HTML and XML files. For example, it can be used to search for specific tags, extract attributes and values, and navigate between elements. It also provides methods for modifying and manipulating the document tree, such as adding or removing elements, modifying attributes, and reformatting the document. Beautiful Soup is built on top of popular Python libraries such as lxml and html5lib, which provide a fast and reliable parsing engine. Beautiful Soup is a powerful and flexible library that allows devel-

opers to extract and manipulate data from HTML and XML files. Its ease of use, rich feature set, and robust parsing engine make it a popular choice for web scraping and data extraction tasks.

3.3.5 Pickle

Pickle is a Python module used for serializing and deserializing Python objects. Serialization is the process of converting objects into a format that can be stored or transmitted, while deserialization is the process of converting the serialized data back into objects. Pickle allows for the efficient storage and retrieval of Python objects, making it useful for a variety of tasks such as data persistence, caching, and inter-process communication.

Pickle works by converting Python objects into a stream of bytes, which can then be written to a file or transmitted over a network. The module provides two main methods for serialization and deserialization: dump() and load(). The dump() method writes the serialized data to a file or file-like object, while the load() method reads the serialized data and reconstructs the original Python objects.

Pickle supports a wide range of Python objects, including built-in types such as integers, strings, and lists, as well as custom classes and objects. However, it should be noted that not all Python objects can be pickled, such as functions and file objects that are currently open. Additionally, the pickle format is specific to Python, so objects serialized with pickle cannot be easily read by other programming languages.

3.4 PROPOSED METHODOLOGY

FakFinder takes the Instagram username or profile link as input. The scrap() function takes this input, finds the corresponding profile and collects the web data. All the data fetched by the scrap() function is publicly available data and no other data is illegally collected.

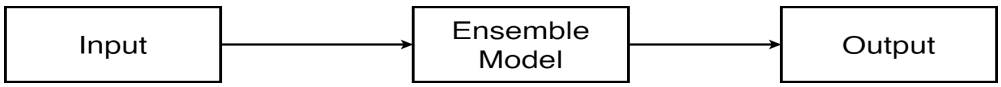

Fig. 3.1. DFD Level 0

The data required for testing is collected from the scraped data. It is then converted into a list of numeric variables. This list is passed to each of the machine learning model, ie, random forest, LSTM and kNN respectively. Random forest takes test size as 30% while LSTM and KNN takes 20% as test size. The number of neighbours in KNN is taken as 3. All the models are set a random state of 42.

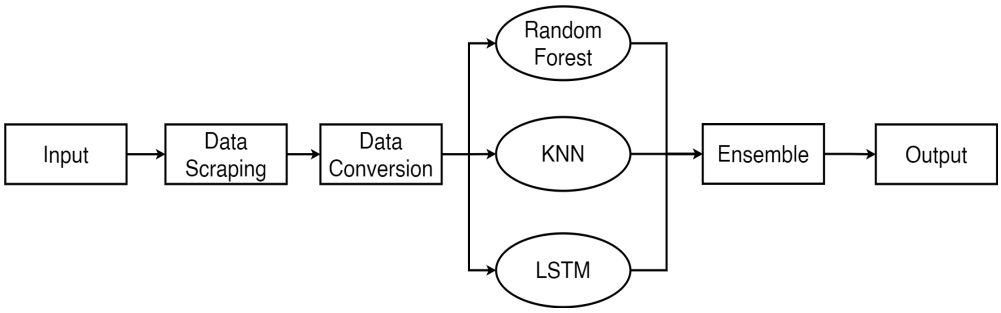

Fig. 3.2. DFD Level 1

The binary outputs from each models is passed to a function ensemble-output() where the average of three outputs is calculated. This result is passed to the user as final prediction.

CHAPTER 4

RESULTS AND DISCUSSIONS

The results of the FakeFinder system were evaluated using a dataset of both real and fake accounts. The Random Forest model obtained highest accuracy of 94% while LSTM and KNN models achieved accuracies of 92% and 90% respectively. The results obtained from FakeFinder can help users in determining the authenticity of a suspicious account more precisely. FakeFinder has the potential to help social media companies to identify and remove fake accounts quickly, improving the overall user experience and trust in their platforms. Users can also use the system to verify the genuinity of an account before engaging with it, which can help to prevent the spread of misinformation and spam on social media.

4.1 SYSTEM TESTING

System testing is a crucial phase in the testing process, as it validates the functionality and performance of a fully integrated software product.The main objective of this testing phases to evaluate the system specifications from end-to-end. To conduct system testing, it is necessary to use all the components that have successfully passed integration testing as input. Table 4.5. indicates the test cases in FakeFinder

Table 4.1. Test Cases Of Web Application

Input	**Response**
Empty	Home Page
Username	Output Generated
Profile link	Output Generated

CHAPTER 5

CONCLUSION

FakeFinder is a web application that uses machine learning algorithms and web scraping techniques to detect fake social media accounts. The platform is publicly available which enables verification of fake accounts easy for the users. The system achieved an average accuracy of 92%, with the ensemble learning method improving the performance of the system. The results demonstrate the potential of the system to help users to identify and remove fake accounts quickly, which can enhance the credibility and trustworthiness of social media platforms.

5.1 SCOPE FOR FUTURE WORK

Future work can include improving the system's performance by incorporating additional features and training data. Additionally, the system can be extended to include other types of social media platforms, such as LinkedIn. FakeFinder can be integrated with social media companies to facilitate quick identification and removal of fake accounts, thereby enhancing the overall user experience.

APPENDIX

UI DESIGN

.1 Home Page

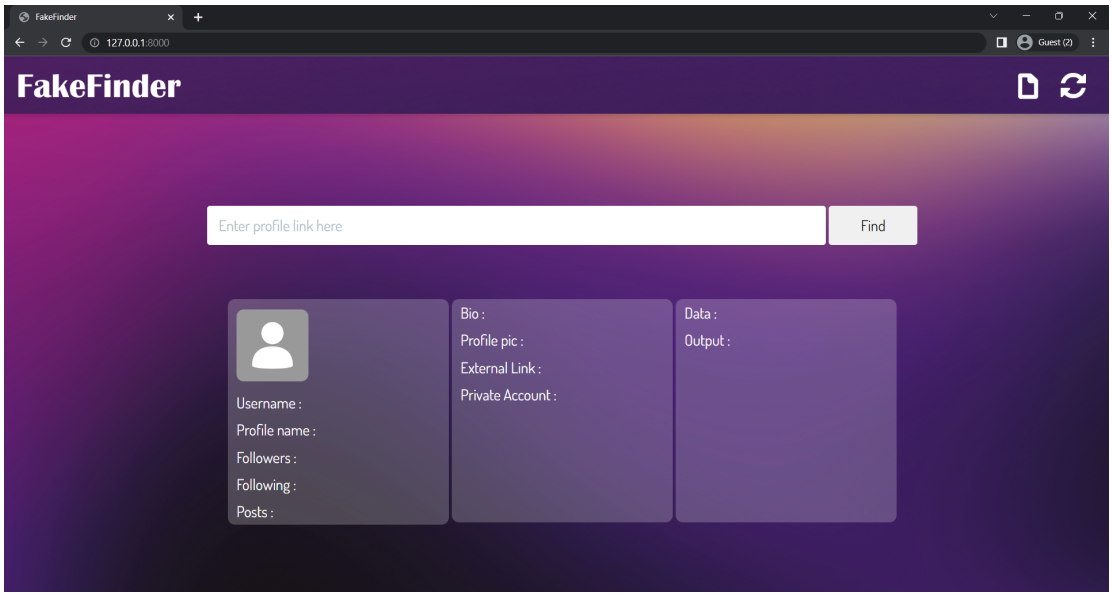

Fig. .1. Home Page

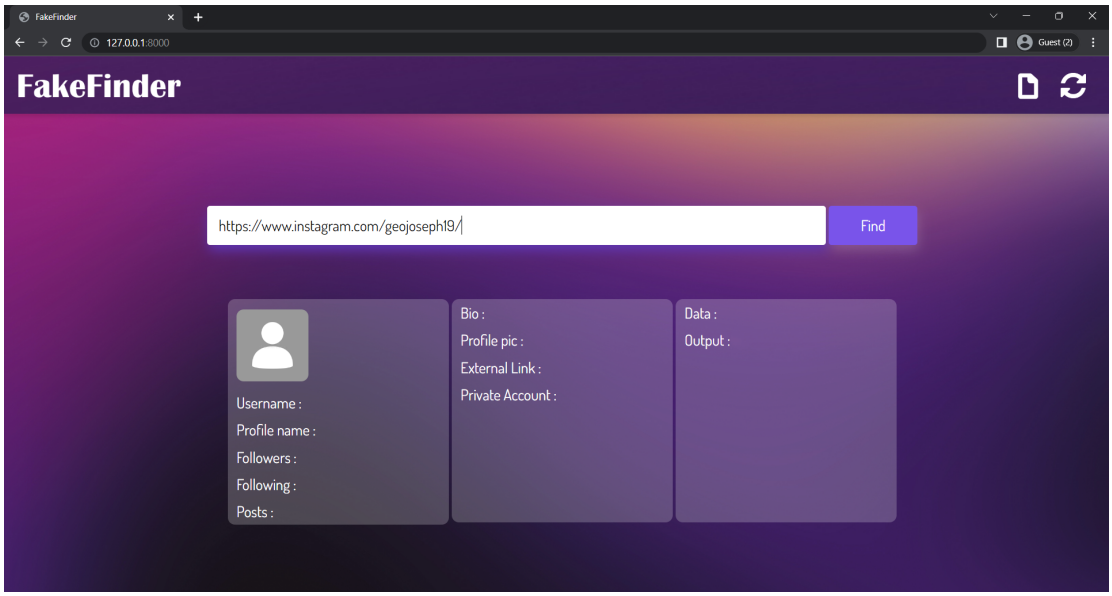

Fig. .2. Profile Link Pasted

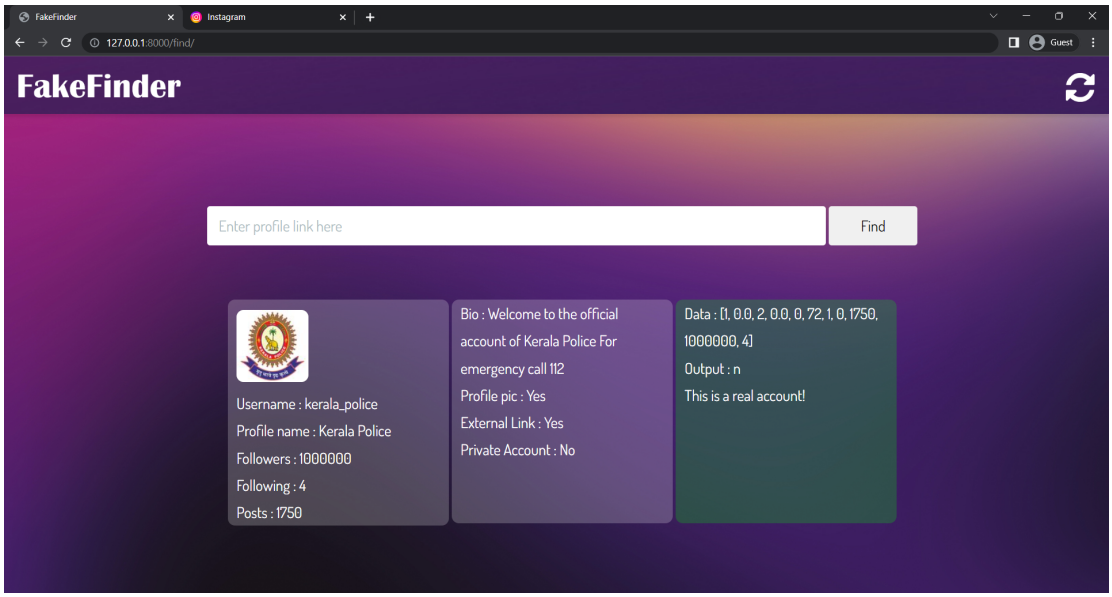

Fig. .3. Detecting an Original Account

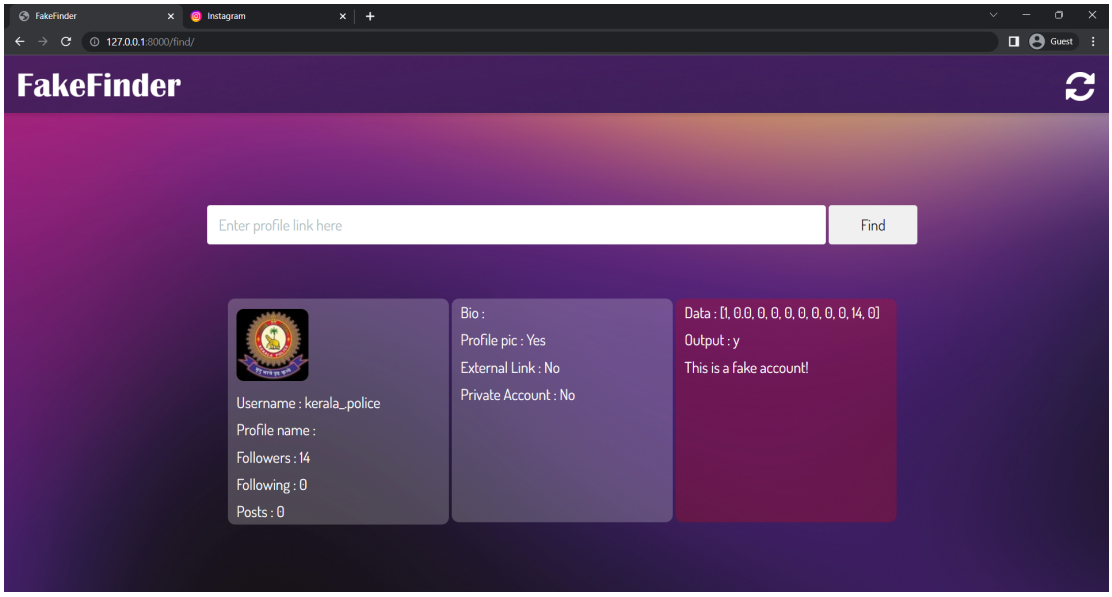

Fig. .4. Detecting a Fake Account

APPENDIX

GIT HISTORY

Geo Joseph / MCAS4 / FakeFinder

Commits

Search commits | All branches

Author	Commit	Message		Date
Geo Joseph	b629188	Added knn model and routed the final output as average of three models	master	2 days ago
Geo Joseph	bba7c1a	Tested, recorded and successfully verified details of 10 original and 10 fake accounts	master	2 days ago
Geo Joseph	8f086f0	Fixed issue in converting posts count abbrevations	master	4 days ago
Geo Joseph	b06d20f	Created pickle file and connected inputs and outputs	master	4 days ago
Geo Joseph	3d2c91b	Input data is processed and converted to fit for testing	master	4 days ago
Geo Joseph	b4e9712	Set up connection between form input and django.	master	5 days ago
Geo Joseph	20c47f7	Added text box and submit button with responsive css	master	5 days ago
Geo Joseph	e97a6ee	created fakefinder homepage with background image and headings	master	5 days ago
Geo Joseph	0dfe22d	Added requirements file	master	2023-04-22
Geo Joseph	5e726b5	Created and initialized django project and app files	master	2023-04-22
Geo Joseph	9635c0e	Fixed issues with fetching data	master	2023-04-21
Geo Joseph	295ab31	Successfully fetched account status(private/public)	master	2023-04-21
Geo Joseph	5bf82ea	Check account status(private/public) using webdriver	master	2023-04-17
Geo Joseph	773b8ed	Updated bs_scrap to fetch bio and profile picture availability	master	2023-04-11
Geo Joseph	20fff20	Updated algorithm to fetch profile name, flwrs, flwing and posts count to variables	master	2023-04-11
Geo Joseph	7128f87	Added code check if an account has profile picture or not	master	2023-04-03
Geo Joseph	16a6d7d	BeautifulSoup Code added. Scraps <meta> tag containing mixed data	master	2023-04-03
Geo Joseph	1500116	Selenium scrapping updated	master	2023-03-30
Geo Joseph	f1c15b6	Web scrapping codes added	master	2023-03-30
Geo Joseph	d143c95	lstm modifications	master	2023-03-29
Geo Joseph	3d83cef	fp_lstm input processing updated	master	2023-03-28
Geo Joseph	cbd90ef	fp_lstm updates	master	2023-03-28
Geo Joseph	ec1c7a9	lstm ipynb file added from Google Colab	master	2023-03-28
Geo Joseph	25db202	Modifications on fp_rf	master	2023-03-27
Geo Joseph	edd7a69	Random forest ipnyb file added from Google Colab	master	2023-03-27

Prev Next

Fig. .1. GIT History Part 1

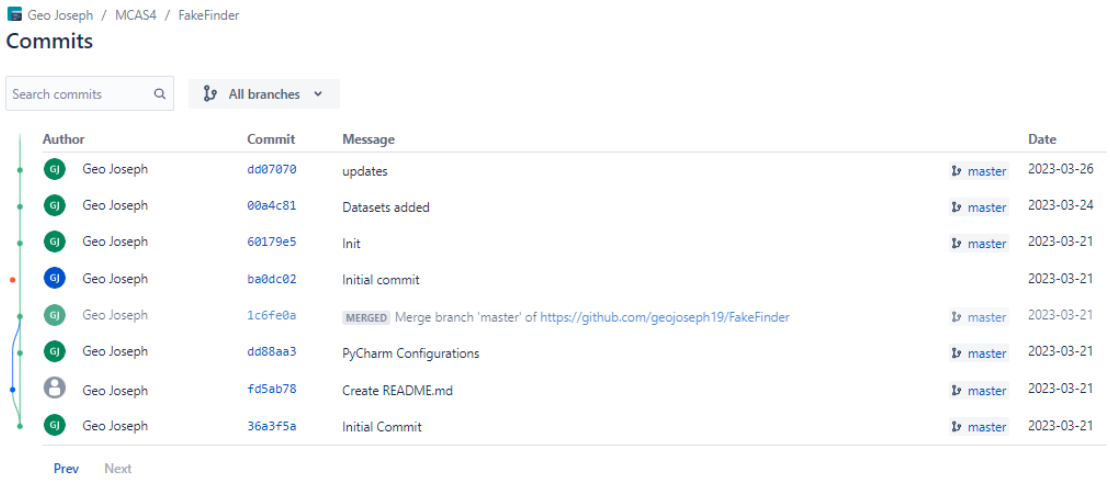

Fig. .2. GIT History Part 2

REFERENCES

[1] AlbayatiMohammed ,Altamimi Ahmad, "Identifying Fake Facebook Profiles Using Data Mining Techniques", Journal of ICT Research and Applications, Vol.13,2019.

[2] Mohammadreza Mohammadrezaei, Mohammad Ebrahim Shiri and Amir Masoud Rahmani1, "Identifying Fake Accounts on Social Networks Based on Graph Analysis and Classification Algorithms",Security and Communication Networks, Vol.2018, 2018.

[3] Instagram Fake Spammer Genuine Accounts Dataset, https://www.kaggle.com/datasets/free4ever1/instagram-fake-spammer-genuine-accounts

[4] Stackoverflow, https://stackoverflow.com/

[5] Django Documentation, https://docs.djangoproject.com/en/4.2/