WEST UNIVERSITY OF TIMIŞOARA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
BACHELOR STUDY PROGRAM: Computer Science in English

BACHELOR THESIS

SUPERVISORS:
Lect. Dr. Cosmin Bonchiş
Drd. Alin Brinduşescu

GRADUATE:
Emanuel Covaci

TIMIŞOARA
2020

WEST UNIVERSITY OF TIMIŞOARA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
BACHELOR STUDY PROGRAM: Computer Science in English

Detecting Fake Accounts on Social Media

SUPERVISORS:
Lect. Dr. Cosmin Bonchiş
Drd. Alin Brinduşescu

GRADUATE:
Emanuel Covaci

TIMIŞOARA
2020

Abstract

Nowadays, online social networks have changed the way we communicate. People are more likely to use the social network to keep in touch with family, friends and college. The massive growth of online social media creates a perfect environment for hackers. They use a fake account on the social media platform, such as Facebook, Twitter, Instagram to gain access to personal data. The purpose of my research is to create an application that identifies if an account is fake or not and thus to protect people from fake accounts.

Contents

Chapter 1

Introduction

Online Social Networks represent one of the main parts of our lives. Most of us wake up in the morning and the first thing we do is to check our smartphone. We want to know if anyone looked for us, we want to read the long list of notification, after that we check what is new and later we decide it is time to wake up from the bed and start a new day.

We must admit the truth, we live in two different worlds, we have a real identity and we have, at the same time, an online identity. Online social networks know a lot of things about us: when we wake-up, when we left our house, who our close friends are, they basically know everything and that is a big problem; our sensitive and personal data are available on the internet. Online social networks have become a target for the hackers lately. They create fake accounts to carry out abuse.

First, the hackers create a new account on a different social network, such as Facebook, Twitter or Instagram,and try to make as many connections as possible with real people.

Then, when becoming friends in the social world they try to steal sensitive information and sell this on the black market.

In other cases, they use fake identities to share fake news to manipulate the people in the process of decision making.

Non-profit organizations have spent on sponsoring political ads on Facebook between 2014 and 2018 up to 2.53 million US dollars. Many political ads were totally fake or missed an important piece of information in order to manipulate people.

Facebook is one of the most popular online social networks, created in 2004 by Mark Zuckerberg. From then on, this platform has grown every year and now they have "over 2.6 billion monthly active users as of the first quarter of 2020" [1]

Facebook predicted correctly the future of the social networks so they decided to extend their presence on other online platforms.

On 9 April 2012 Facebook announced the acquisition of Instagram. At that time, Instagram was more a photo filter application, but had a big potential to become one of the biggest rivals for Facebook, so they decided to spend 1 billion

US dollars for this acquisition.
In 2014 they decided to buy WhatsApp, an online messenger application for 19 billion US dollars.
In 2013 Facebook also tried to buy Snapchat because more and more young users decided to use it instead of Facebook, so they sensed danger when seeing the potential of this new application. They wanted to spend up to 3 billion US dollars for this acquisition, but Snapchat repeatedly refused. So Facebook decided to focus all their attention on Instragam in order to attract young users.

Facebook estimated in 2015 that up to 14 million of its monthly active users are in fact fake, therefore hackers decide to activate new accounts on their social platform violating in this way the website's terms of services.

A new online social platform named TikTok reported in May 2020 that they had reached 800 million active users, but the most important part was represented by the fact that 41 per cent of all TikTok accounts are owned by users aged between 16 and 24.

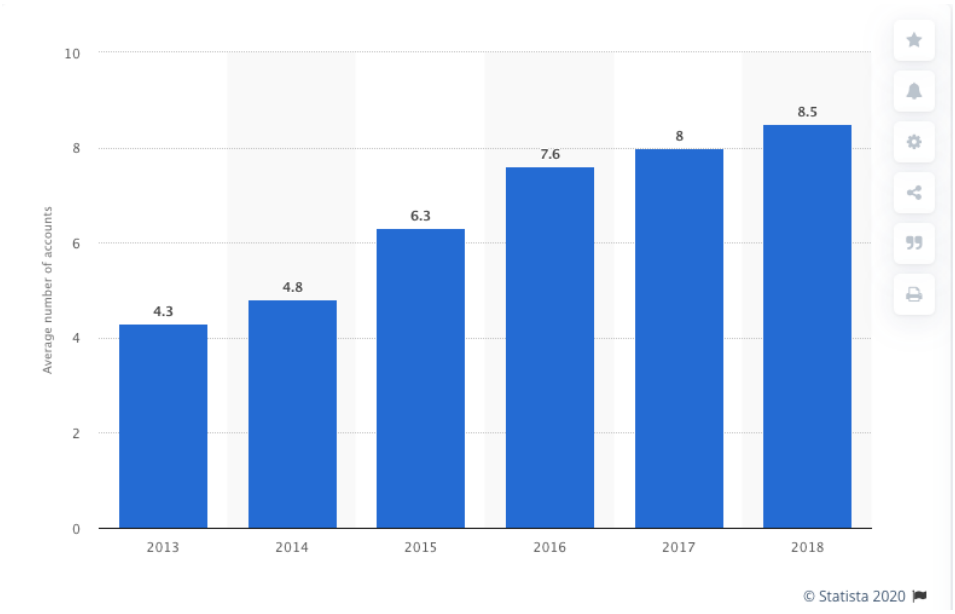

Figure 1.1: Average number of social media accounts per internet user from 2013 to 2018 [2]

We can see very clearly the importance and growth of social media. This growth makes our lives easier by giving us so many ways to communicate, but at the same time, we entrust all our personal data to the different online platforms. In these times, information means power, so many entities try to have all the information in order to have control over people's actions.

Chapter 2

Problem Description

2.1 What are Fake Profiles ?

A fake profile is the description of a personality, institution or corporation that does not truly exist, on social media. In other cases, they use the real information of these entities and create a fake profile that looks real and credible for a normal user. Hackers, as often as possible use real names, personalities that look genuine. The profile pictures they use, are normally adjusted variants of pictures taken from real individuals or associations. Most fake accounts are newly opened and have few friends or connection on their profile, and the majority of their friends do not have anything in common.

These accounts are used to abuse sign-up bonuses, to generate fake news and spam content and eventually spread malware. Sometimes these fake users try to influence the results of reviews or try to manipulate public opinion in voting processes.

2.2 Who creates Fake accounts and why?

The first time when fake accounts appeared on social media was when individuals wanted intimacy, wanted to remain anonymous on the internet, therefore avoiding to use their real name, real profile pictures and even avoid posting anything at all on social media. They only wanted to have an account to see what was happening on social media.

Then, some individuals seized the opportunity of using different user profiles to gain bonus offers. For example, in the gaming industry, many users decide to open a new account to receive sign-up bonuses, referral bonuses. In other cases, fake accounts are used to influence the result of surveys.

Finally, some hackers want to collect sensitive and private information in order to blackmail people or companies and they demand large sums of money in order not to publish the stolen information.

2.3 The anatomy of a fake account

- **Identity source data** – Before running the user registration processes, bad actors will obtain identity source data, either stolen data, fabricated data, or a combination of the two, and use the bulk data as inputs for their next attack phase [3]
- **Registration/User enrollment processes** – Threat actors use APIs to sidestep new account registration forms and generate them behind the scenes in large quantities.[3]
- **Accounts Created** – Once the fake accounts are created, cybercriminals can use them for the various purposes listed above [3]

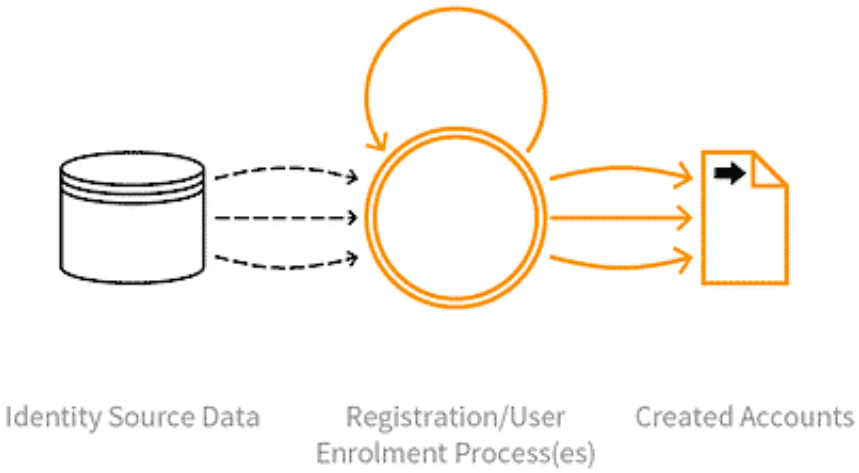

Figure 2.1: The anatomy of a fake account creation attack [3]

Chapter 3

Related Work

3.1 Facebook removes 3.2 billion fake accounts, millions of child abuse posts

> "(Reuters) - Facebook Inc (FB.O) removed 3.2 billion fake accounts between April and September this year (2019), along with millions of posts depicting child abuse and suicide, according to its latest content moderation report released on Wednesday. That more than doubles the number of fake accounts taken down during the same period last year, when 1.55 billion accounts were removed, [4] according to the report [5]"

3.2 Facebook AI

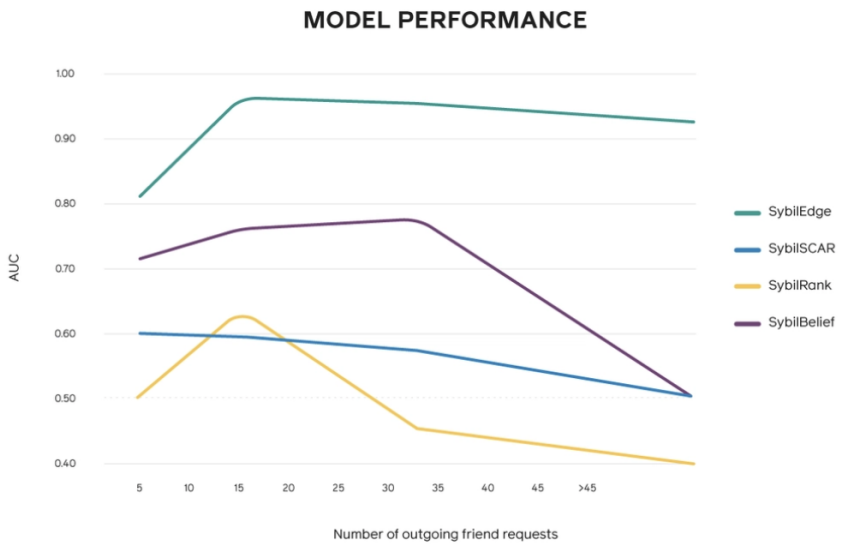

Figure 3.1: SybilEdge Model [6]

Facebook engineers Roee Eilat and Udi Weinsberg created a new algorithm in order to help Facebook identify more quickly the fake account on the platform. This algorithm received the name SybilEdge. Their approach analyzes the way

new users make friends on the platform and can detect if an account is fake with only 20 friend requests sent to other users. An internal study revealed that fake accounts and genuine accounts have a totally different behaviour. A fake user receives a rejection from the real accounts more often than in case of a genuine profile.

3.3 Twitter and fake accounts

After the U.S.A. presidential campaign in 2016, Twitter took a lot of action to remove all fake accounts that were used in the campaign to change the people's opinion and gain votes. From Virginia Commonwealth University, Professor Christopher Whyte said that companies like Facebook, Twitter, Instagram try not to make all their research and conclusions from the process of detecting fake accounts on their platform public because in this way they can have control over time. If they publicise all these information, then hackers can find a way to avoid this restriction. So with this approach, they can gain time in the battle against fake accounts. It is all about time, how time passes from the moment a fake account is created and the moment the algorithm detects and deletes the fake account. They try to keep their response time as short as possible.

Chapter 4

Solution

Detection of fake users is also a challenge for the real users in order to protect themselves from different cybernetic attacks.

4.1 How can I detect if a profile is fake?

You must track the following characteristics of fake account:

1. **The profile photo does not look realistic**
    - Fake accounts try to look real as much as possible so in some cases they steal different images from real users and create a new identity. In general, they have the perfect image, the one that looks very professional and inspires credibility, but after a very detailed analysis we can observe that only a few pictures are of high quality, the rest being just normal or very low quality and another red flag is that professional images do not have the photographer's credit mentioned in their description. By doing this, they try to bypass a report reclamation from the original author of photography. In other cases, they use a profile picture that is unusual, something that a real person would most likely try to avoid.
    - You can use reverse google images, instead of searching for a photo by keywords you can search by an image and see the results. If the image is stolen from the internet, in the majority of cases google will find the original site.
2. **The profile was created recently**
    - Most of us created a Facebook account between 2007 - 2010, as compared to very young users who created an account in the past year or two. So a fake user will always be created in the last period.
3. **Strange posts**
    - Something that is universal for all fake accounts is that they have very strange posts in their feed – posts that no real person would upload.
4. **Number of followers and the reaction received from the followers**

- This is probably the most common red flag. Fake accounts can have a lot of followers, but when they post something on the internet they receive few reactions compared with the number of followers

These are just a few steps that you can follow and make a good prediction about the authenticity of the profile.

4.2 My solution

It is really difficult and time-consuming to follow all these steps from the last section to identify whether an account is fake or not.

My solution is an online application that uses artificial intelligence to detect if a social media account is fake.

1. The user will provide the public link access to the user account and the the online application will start to analyze the profile.
2. By using Machine Learning I will create a model able to predict if the user account is fake or not.
3. This model will be based on some criteria like: the number of posts, the number of reaction received on each post reported to the total number of followers (on Instagram) or friends (on Facebook) etc.
4. The Machine Learning model will analyze the public profile and elaborate a report with the prediction.
5. The user will have the option to give feedback for prediction. If the user is really sure that a profile is fake, but the model predicts that it is real, the user can send feedback and that information will be used to improve the accuracy of the model. In this manner, the model becomes more and more precise with every use.

Chapter 5

Application Architecture

I choose to develop an application that detects if a Twitter profile is fake or real.

The main reason behind this is that Twiter offers simple and advanced API integration.

After the Cambridge Analytica case, Facebook restricted all integration with his platform in order to increase the level of security. This decision also impacted Instagram, so from my investigation I concluded that the best decision is to integrate my machine learning algorithm with Twitter.

5.1 Application name and logo

The name of the application is **Twittop**.

Figure 5.1: Twittop Logo

5.2 High level architecture design

The system is composed of three main layers:

1. **Machine Learning Model**
2. **Front-end**
3. **Back-end**

Each layer is represented by an independent custom application, each application has its own responsibility and features in order to fulfill the systems requirements.

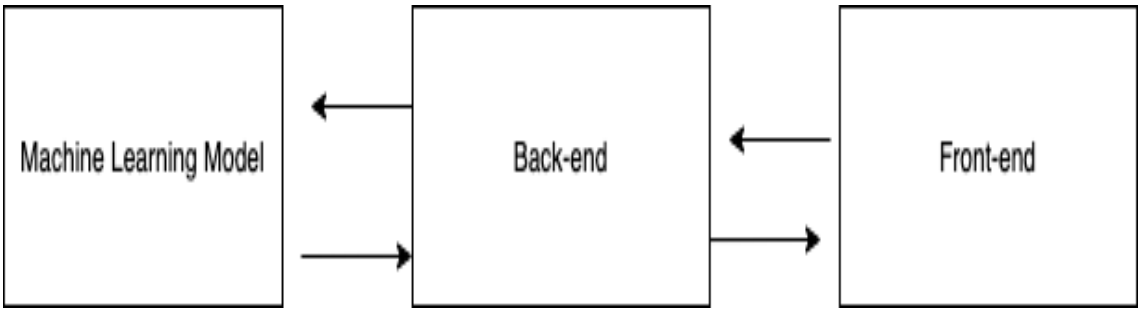

Figure 5.2: High level architecture design

1. **Machine Learning Model** is responsible for the detection of fake Twitter profiles.
2. **Front-end** represents the web application that makes the connection between the user and our system
3. **Back-end** is responsible for connecting the web application with the machine learning model

5.3 Machine Learning

5.3.1 Machine Learning Model

"Machine Learning is the science of getting computers to learn and act like humans do, and improve their learning over time in autonomous fashion, by feeding them data and information in the form of observations and real-world interactions."[7]

"The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers to learn automatically, without human intervention or assistance, and adjust actions accordingly." [8]

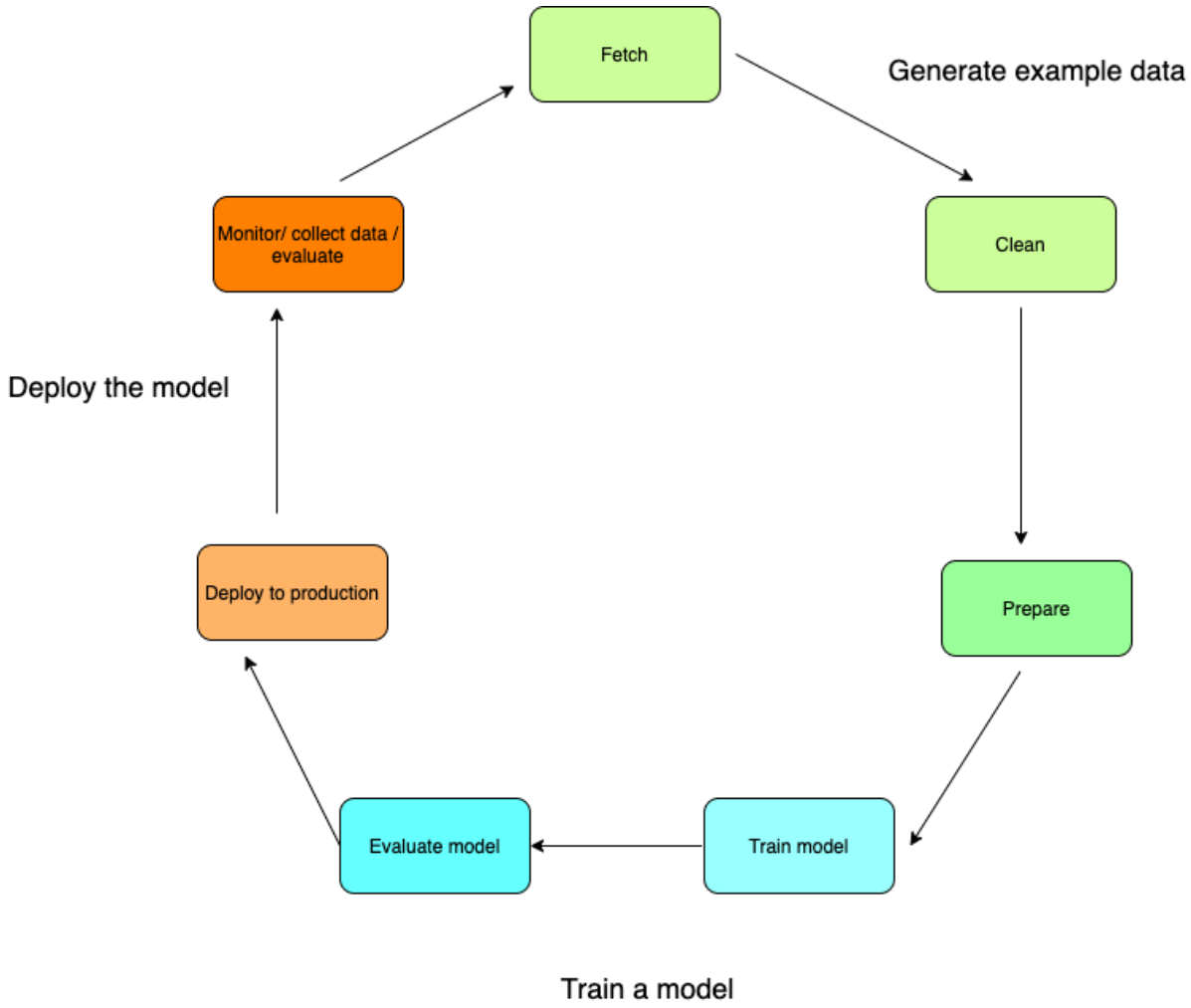

Figure 5.3: The workflow of machine learning

For this project, my preference was to use Keras Machine Learning Framework in order to train and deploy a model able to make the correct prediction.

5.3.2 Keras

"Keras is an API designed for human beings, not machines. Keras follows the best practices for reducing cognitive load: it offers consistent and simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear and actionable error messages. It also has extensive documentation and developer guides."[9]

Figure 5.4: Keras: Deep learning for humans

5.3.3 Fetch, clean and prepare

In this section, I will describe the way I adjusted the data that is passed into the model to learn from.

For training, I used a public data set that can be found here `https://github.com/emanuelcovaci/twittop/tree/master/machine_learning/Dataset` .

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from keras.models import Sequential
from keras.optimizers import SGD
from keras.layers import Dense, BatchNormalization, Activation,
    Dropout
from keras.callbacks import EarlyStopping, Callback
from matplotlib import pyplot as plt
import matplotlib.pyplot as plt
from keras.models import load_model
from IPython.display import clear_output

# Loading data
user_object = dict()
user_object["fake"] = pd.read_csv("fake_twitter_accounts.csv")
user_object["legit"] = pd.read_csv("real_twitter_accounts.csv")

# Removing unnecessary columns
```

```
user_object["legit"] = user_object["legit"].drop(
    ["id", "name", "screen_name", "created_at", "lang", "location",
    "default_profile", "default_profile_image",
     "geo_enabled", "profile_image_url", "profile_banner_url", "
   profile_use_background_image",
     "profile_background_image_url_https", "profile_text_color", "
   profile_image_url_https",
     "profile_sidebar_border_color", "profile_background_tile", "
   profile_sidebar_fill_color",
     "profile_background_image_url", "profile_background_color", "
   profile_link_color", "utc_offset", "protected",
     "verified", "dataset", "updated", "description"], axis=1)

user_object["fake"] = user_object["fake"].drop(
    ["id", "name", "screen_name", "created_at", "lang", "location",
    "default_profile", "default_profile_image",
     "geo_enabled", "profile_image_url", "profile_banner_url", "
   profile_use_background_image",
     "profile_background_image_url_https", "profile_text_color", "
   profile_image_url_https",
     "profile_sidebar_border_color", "profile_background_tile", "
   profile_sidebar_fill_color",
     "profile_background_image_url", "profile_background_color", "
   profile_link_color", "utc_offset", "protected",
     "verified", "dataset", "updated", "description"], axis=1)

#Converting DataFrame to Numpy Array
user_object["legit"] = user_object["legit"].values
user_object["fake"] = user_object["fake"].values

# Checking Availability of 'URL' and 'Time Zone'
for index in range(len(user_object["legit"])):
    if type(user_object["legit"][index][5]) == str:
        user_object["legit"][index][5] = 1

    if type(user_object["legit"][index][6]) == str:
        user_object["legit"][index][6] = 1

for index in range(len(user_object["fake"])):
    if type(user_object["fake"][index][5]) == str:
        user_object["fake"][index][5] = 1

    if type(user_object["fake"][index][6]) == str:
        user_object["fake"][index][6] = 1

# Changing DataType of data to 'float64'
user_object["legit"] = user_object["legit"].astype(np.float64)
user_object["fake"] = user_object["fake"].astype(np.float64)

# Changing all 'NaN' to '0'

where_nans = np.isnan(user_object["legit"])
user_object["legit"][where_nans] = 0

where_nans = np.isnan(user_object["fake"])
user_object["fake"][where_nans] = 0

# Creating merged dataset from legit and fake profile dataset.
```

```
X = np.zeros((len(user_object["fake"]) + len(user_object["legit"]), 7))
Y = np.zeros(len(user_object["fake"]) + len(user_object["legit"]))

for index in range(len(user_object["legit"])):
    X[index] = user_object["legit"][index] / max(user_object["legit"][index])
    Y[index] = -1

for index in range(len(user_object["fake"])):
    bound = max(user_object["fake"][index])
    if bound == 0:
        bound = 1

    X[len(user_object["legit"]) + index] = user_object["fake"][index] / bound  # Normalizing Data [0 <--> 1]
    Y[len(user_object["legit"]) + index] = 1

# Splitting Dataset into Train and Test
X_train_data, X_test_data, y_train_data, y_test_data = train_test_split(X, Y,

        test_size=0.24, random_state=42)


#Creating Accuracy Loss Graph callback
class PlotLearning(Callback):
    def on_train_begin(self, logs={}):
        self.i = 0
        self.x = []
        self.losses = []
        self.val_losses = []
        self.acc = []
        self.val_acc = []
        self.fig = plt.figure()

        self.logs = []

    def on_epoch_end(self, epoch, logs={}):
        self.logs.append(logs)
        self.x.append(self.i)
        self.losses.append(logs.get('loss'))
        self.val_losses.append(logs.get('val_loss'))
        self.acc.append(logs.get('acc'))
        self.val_acc.append(logs.get('val_acc'))
        self.i += 1
        f, (ax1, ax2) = plt.subplots(1, 2, sharex=True)

        clear_output(wait=True)

        ax1.set_yscale('Log')
        ax1.plot(self.x, self.losses, label="loss")
        ax1.plot(self.x, self.val_losses, label="val_loss")
        ax1.legend()

        ax2.plot(self.x, self.acc, label="accuracy")
        ax2.plot(self.x, self.val_acc, label="validation accuracy")
        ax2.legend()
```

```
121
122         plt.show();
123
124
125 plot = PlotLearning()
```

5.3.4 Building Predictive Model

```
1  model = Sequential([
2      BatchNormalization(),
3
4      Dense(16, activation="relu", kernel_regularizer="l2"),
5      BatchNormalization(),
6      Dense(8, activation="relu", kernel_regularizer="l2"),
7      BatchNormalization(),
8      Dense(1, activation="tanh"),
9  ])
10
11 model.build((None, X.shape[1]))
12 model.summary()
13 model.compile(
14     optimizer="adadelta",
15     loss="binary_crossentropy",
16     metrics=["acc"]
17 )
```

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
batch_normalization_13 (Batc (None, 7)                 28
_________________________________________________________________
dense_13 (Dense)             (None, 16)                128
_________________________________________________________________
batch_normalization_14 (Batc (None, 16)                64
_________________________________________________________________
dense_14 (Dense)             (None, 8)                 136
_________________________________________________________________
batch_normalization_15 (Batc (None, 8)                 32
_________________________________________________________________
dense_15 (Dense)             (None, 1)                 9
=================================================================
Total params: 397
Trainable params: 335
Non-trainable params: 62
_________________________________________________________________
```

Figure 5.5: Model Summary

5.3.5 Train model

```
1
2 model.fit(X_train_data, y_train_data, epochs=2000, validation_data
     =(X_test_data, y_test_data), shuffle=True,
3           batch_size=100,
4           callbacks=[early_stopping, plot])
```

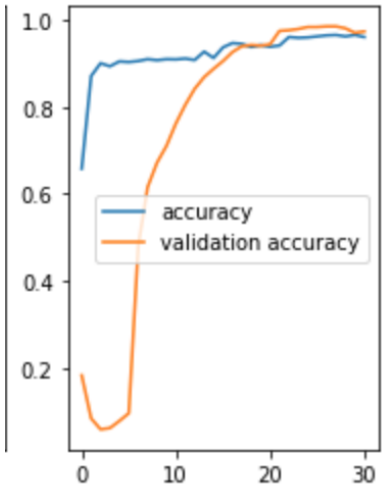

Figure 5.6: Train model: accuracy and validation accuracy

5.3.6 Evaluate and make prediction

```
prediction = model.predict(X_test_data).T[0]

for index in range(len(prediction)):
    prediction[index] = -1 if prediction[index] < 0 else 1


def plot_confusion_matrix(cm, title='CONFUSION MATRIX', cmap=plt.cm
    .Reds):
    target_names = ['Fake', 'Real']
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(target_names))
    plt.xticks(tick_marks, target_names, rotation=45)
    plt.yticks(tick_marks, target_names)
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')


mat = confusion_matrix(y_test_data, prediction)
print(mat)

plot_confusion_matrix(mat)
```

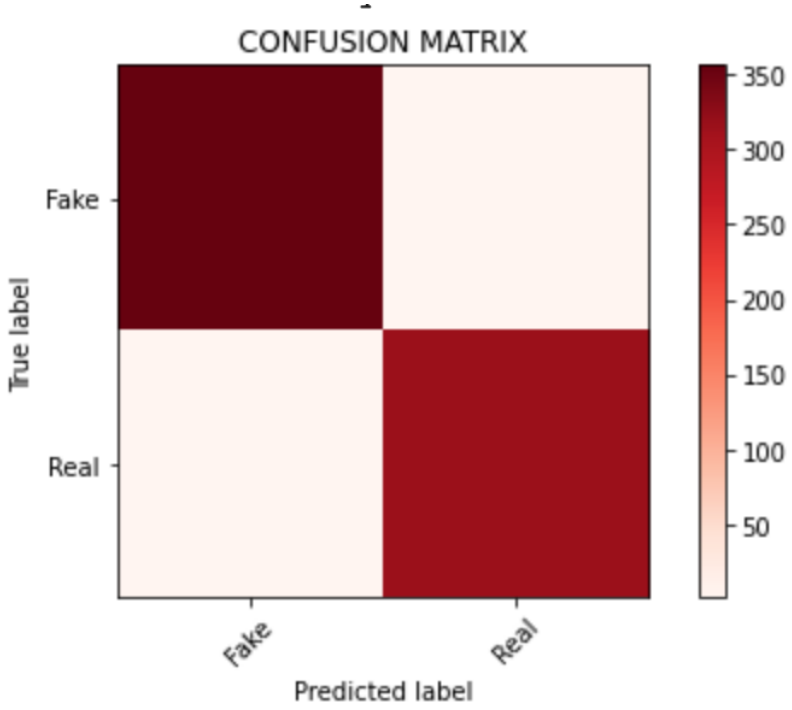

Figure 5.7: CONFUSION MATRIX

5.3.7 Validation Accuracy

```
_, train_accuracy = model.evaluate(X_train_data, y_train_data)
_, validation_accuracy = model.evaluate(X_test_data, y_test_data)
print("Train Accuracy:", train_accuracy)
print("Validation Accuracy:", validation_accuracy)

# Outputs:
# 2141/2141 [==============================] - 0s 110us/step
# 677/677 [==============================] - 0s 126us/step
# Train Accuracy: 0.9789817929267883
# Validation Accuracy: 0.9719350337982178
```

5.4 Back-end

For the back-end part, I prefer to use Flask Framework.

> "Flask is a lightweight WSGI web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around Werkzeug and Jinja and has become one of the most popular Python web application frameworks.
>
> Flask offers suggestions, but doesn't enforce any dependencies or project layout. It is up to the developer to choose the tools and libraries they want to use. There are many extensions provided by the community that make adding new functionality easy."[10]

Figure 5.8: Flask framework logo

Back-end component makes the connection between the Web application and machine learning model.
For that, I need the two end-points.

- **analyze profile** : receive from the front-end a user object that contains the important attributes that are used in the prediction part. Object data is transformed to fulfil de input data format in the machine learning model. After execution this end-point return to front-end the result of the prediction..

```
@app.route('/analyze_profile', methods=['GET', 'POST'])
def analyze_profile():
#Request parameters
    req_data = request.get_json()
    user_profile = req_data['userProfile']
    print (user_profile)
#Get data from request object
    user_prediction = {
        'statuses_count': user_profile['statuses_count'],
        'followers_count': user_profile['followers_count'],
        'friends_count': user_profile['friends_count'],
        'favourites_count': user_profile['favourites_count'],
        'listed_count': user_profile['listed_count'],
        'url': user_profile['url'],
        'time_zone': user_profile['time_zone']
    }
#Prepare data for prediction
    df_user_prediction = pd.DataFrame(user_prediction, columns
   =['statuses_count', 'followers_count', 'friends_count',
```

```
19
      'favourites_count', 'listed_count', 'url', 'time_zone'],
20                                        index=[0])
21 # ...
22
23 #Load model
24     model = load_model('../machine_learning/Dataset/
       model_twitter.hdf5')
25     model.summary()
26 #Make Prediction
27     prediction = model.predict(df_user_prediction)
28 # ....
29 # send the model prediction to Web Application
30     return data_response
```

- **send feedback**: this end-point it created to receive from the front-end the user feedback regarding a prediction. This function saves the object data in the directory destined for user feedback as a JSON file. This JSON file later cand be loaded in Machine Learning Model a retain to improve prediction performance.After the JSON file is saved on the disk the end-point response to front-end with a success status code.

```
1 @app.route('/send_feedback', methods=['GET', 'POST'])
2 def send_feedback():
3     req_data = request.get_json()
4     profile = req_data['profile']
5     profile_data = profile['profile_data']
6     username = profile_data['screen_name']
7     data_response = {
8         'message': 'Your feedback was recorded!',
9     }
10    filename = username + '.json'
11    with open('user_feedback/' + filename, 'w') as outfile:
12        json.dump(req_data, outfile)
13    return data_response
```

5.5 Front-end

For the front-end part, I prefer to use Angular.
Front-end has a direct connection with the user. The user can analyze a twitter profile by using a web browser and accessing the addresses of the application.

Angular is an application design framework and development platform for creating efficient and sophisticated single-page apps.

Figure 5.9: Angular logo

For the front-end part, I implemented the following components to respond to requirements

- **SearchComponent**: is responsible to render the search page. When the user accesses the Web Application this component is called. The main purpose of this component is to offer a simple page with a search bar where the user can type the Twiter username and when the user clicks on the GO button or presses enter the component sends the username to the next layer.
- **TwitterProfileComponent**:is responsible for showing the prediction, if the Twitter account is fake or not. To achieve that, this component calls the Twitter API to receive all profile information. When the component receives the information from Twitter this component calls the back-end end-point to analyze the profile. That was detailed in the previous section. The function waits for the response from the back-end and then shows the report result in the page.

5.5.1 Angular services

To be able to connect the TwitterProfileComponent with the back-end I implemented the following services:

- **analyze**:

```
analyze(userProfile: UserProfile) {
    return this.http.post<UserProfile>(this.ip + '/
   analyze_profile', {
      userProfile
    }).toPromise().then(response => {
      return response;
    }).catch(err => {
      return err;
    });
  }
```

- **sendFeedback**:

```
sendFeedback(profile: any) {
   return this.http.post(this.ip + '/send_feedback', {
     profile
   }).toPromise().then(response => {
     return response;
   }).catch(err => {
     return err;
   });
 }
```

- **TwitterService**

```
  <!-- This function makes an API call to Twitter to obtain
   profile information-->
getUserProfile(username: string) {
    return new Promise(resolve => {
      this.http.get(`http://localhost:7890/1.1/users/show.json
   ?screen_name=${username}`).subscribe(data => {
        resolve((data));
      });
    });
  }
```

```
  <!-- This function makes an API call to Twitter to obtain
   all tweets posted by the user-->
  return new Promise(resolve => {
      this.http.get(`http://localhost:7890/1.1/statuses/
   user_timeline.json?count=${count}&screen_name=${username}`)
   .subscribe(data => {
        resolve((data as Array<any>));
      });
    });
  }
```

5.5.2 Node Server

I created a simple node server that acts as a proxy. This proxy server authorised my request to twitter API. In the server configuration file, I added the credential access received from the Twitter API.

5.5.3 Twitter API

"The Tweet object will also contain User objects of the Users involved within a Tweet. In case of Retweets and Quoted Tweets, the top-level user object represents what account took that action, and the JSON payload will include a second user within the retweeted status for the account that created the original Tweet. User objects can be retrieved using the id or screen name. In general these user metadata values are relatively constant. Some fields never change, such as the user's id provided as a string id str and when the account was created. Other metadata can occasionally change, such as the screen name, displayname, description, location, and other profile details. Some

metadata frequently changes, such as the number of Tweets the account has posted statuses count and its number of followers followers count" [11]

Figure 5.10: Twitter user


```
{
  "id": 6253282,
  "id_str": "6253282",
  "name": "Twitter API",
  "screen_name": "TwitterAPI",
  "location": "San Francisco, CA",
  "profile_location": null,
  "description": "The Real Twitter API. Tweets about API
   changes, service issues and our Developer Platform. Don't
   get an answer? It's on my website.",
  "url": "https:\/\/t.co\/8IkCzCDr19",
  "entities": {
    "url": {
      "urls": [{
        "url": "https:\/\/t.co\/8IkCzCDr19",
        "expanded_url": "https:\/\/developer.twitter.com",
        "display_url": "developer.twitter.com",
        "indices": [
          0,
          23
```

```
        ]
      }]
    },
    "description": {
      "urls": []
    }
  },
  "protected": false,
  "followers_count": 6133636,
  "friends_count": 12,
  "listed_count": 12936,
  "created_at": "Wed May 23 06:01:13 +0000 2007",
  "favourites_count": 31,
  "utc_offset": null,
  "time_zone": null,
  "geo_enabled": null,
  "verified": true,
  "statuses_count": 3656,
  "lang": null,
  "contributors_enabled": null,
  "is_translator": null,
  "is_translation_enabled": null,
  "profile_background_color": null,
  "profile_background_image_url": null,
  "profile_background_image_url_https": null,
  "profile_background_tile": null,
  "profile_image_url": null,
  "profile_image_url_https": "https:\/\/pbs.twimg.com\/
   profile_images\/942858479592554497\/BbazLO9L_normal.jpg",
  "profile_banner_url": null,
  "profile_link_color": null,
  "profile_sidebar_border_color": null,
  "profile_sidebar_fill_color": null,
  "profile_text_color": null,
  "profile_use_background_image": null,
  "has_extended_profile": null,
  "default_profile": false,
  "default_profile_image": false,
  "following": null,
  "follow_request_sent": null,
  "notifications": null,
  "translator_type": null
}
```


Listing 5.1: Example user object

5.6 Sequence diagram of the application layer

This section will briefly describe the interaction between the user and our system. Below you have a detailed sequence diagram which shows the way our system behaves whenever a user issues a request to a particular web page. This high-level diagram contains every single layer of our system along with the messages which are sent between them.

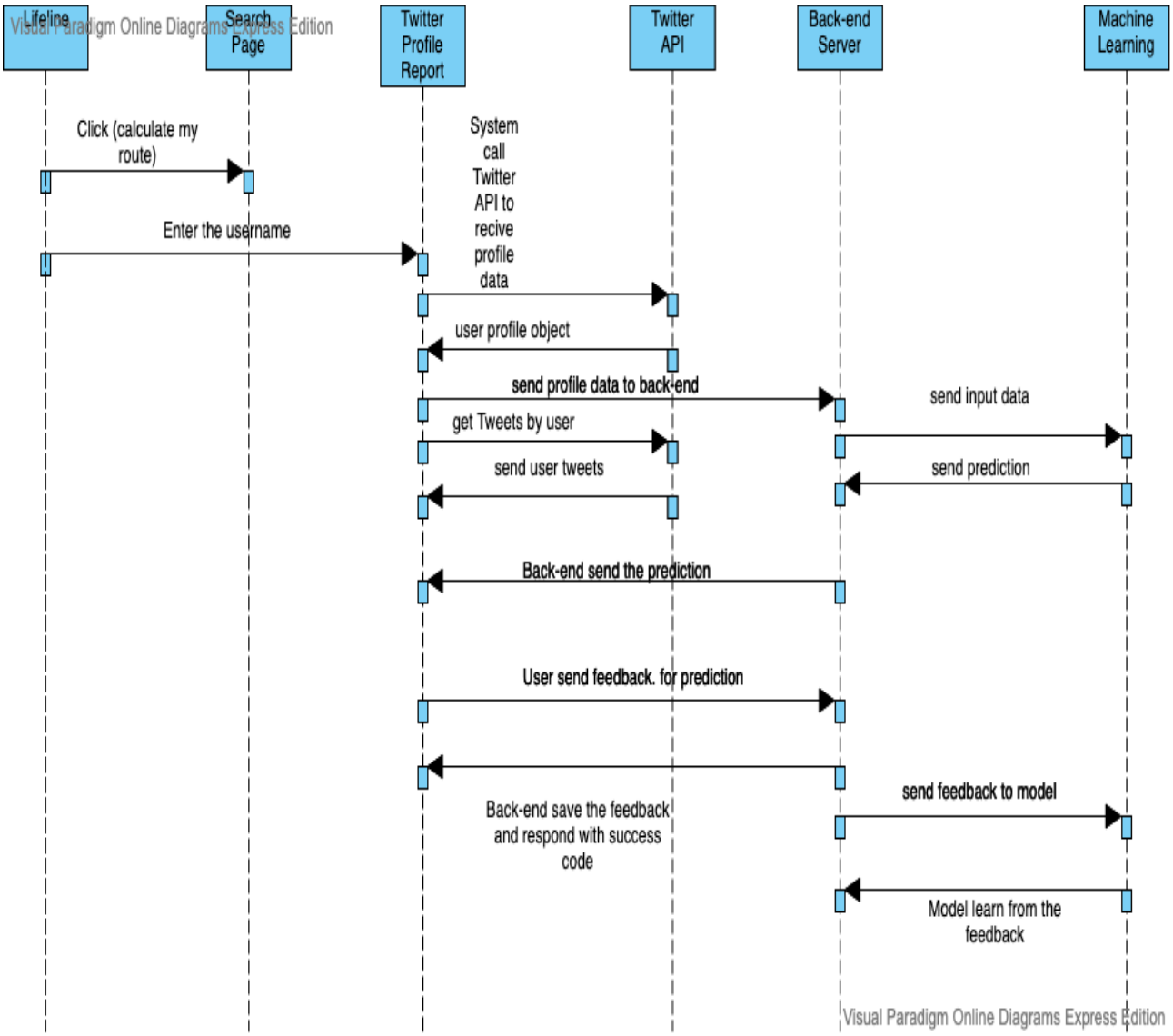

Figure 5.11: Sequence diagram of the application layer.

Chapter 6

Application Functionalities

6.1 User Guide

The Twittop Application helps the user to identify if a Twitter user is fake or not using the power of a Machine Learning algorithm.

To receive profile analysis the user must provide a valid username.

6.1.1 Search page

When the user opens the application that will show the search page, here the user can type in the search box the username of the Twitter account and then click on the GO button or press the enter key from the keyboard.

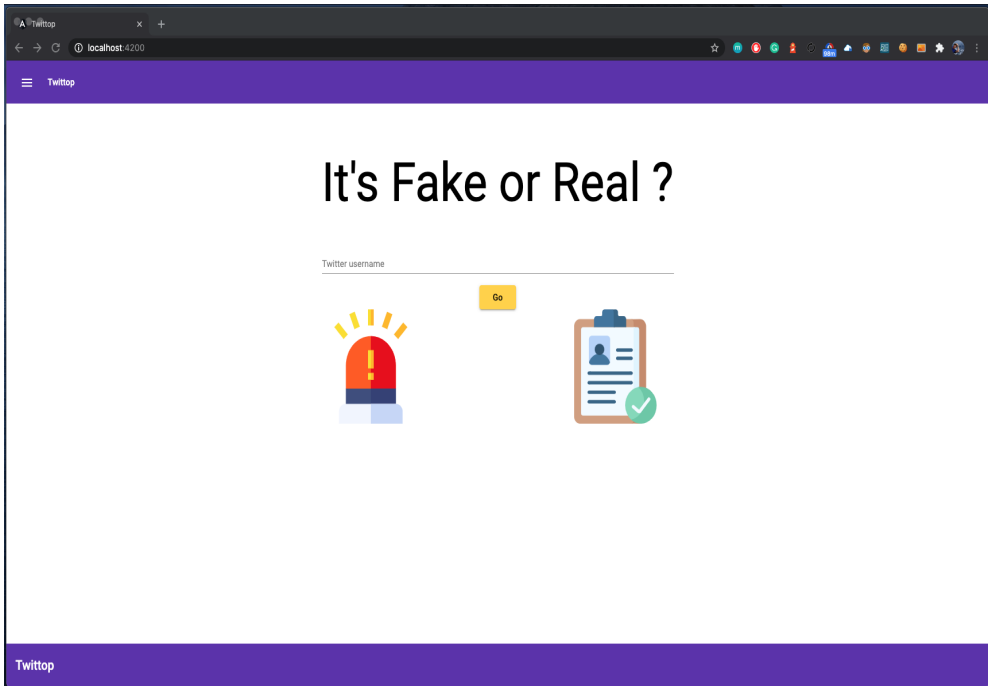

Figure 6.1: Search page

6.1.2 User types the username of the Twitter account

The User types the username of the Twitter account and then presses enter from keyboard or clicks on the GO button.

Figure 6.2: User type the username of the Twitter account

6.1.3 Show report results

Using the power of the trained machine learning model, the application made a prediction. The report will show if the account is genuine or fake.

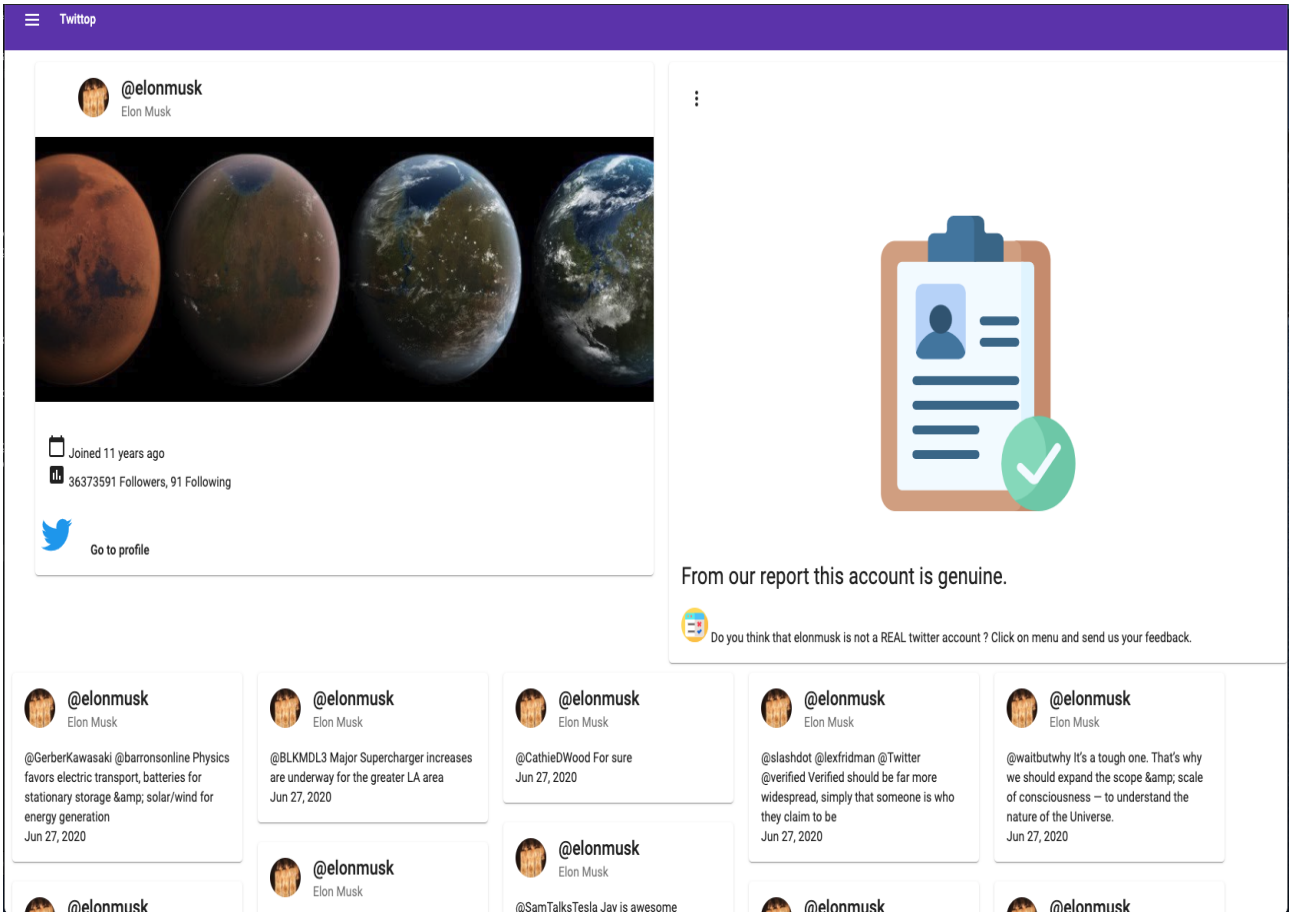

Figure 6.3: Show report results: account is genuine

If the account is fake the application will show the following page report.

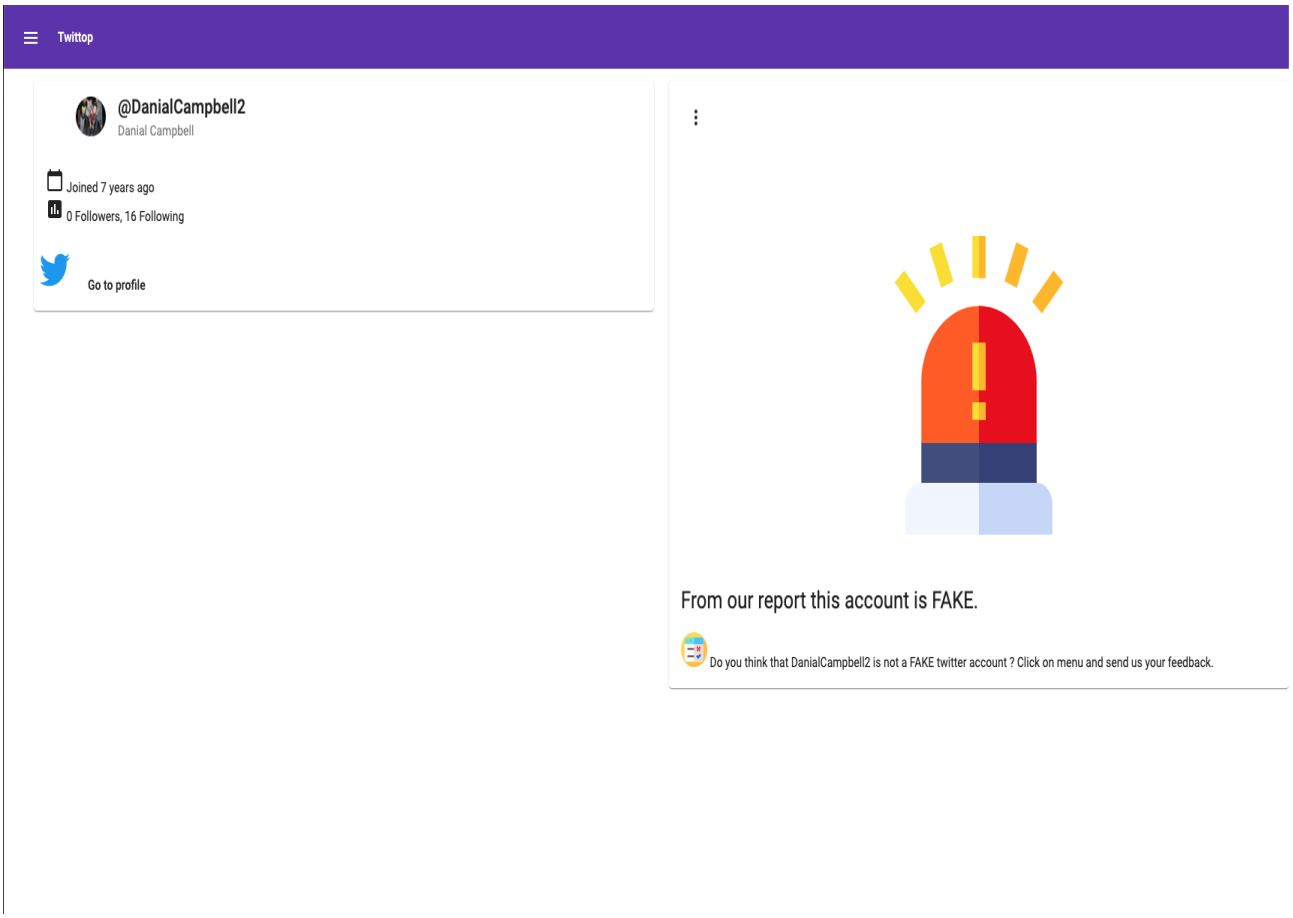

Figure 6.4: Show report results: account is FAKE

6.1.4 User send feedback

If the model predicts a wrong classification for the profile and the user is sure of that, he/she has the option to send feedback to the system.
This feedback will be used to retrain the model so in this way the model becomes more precise with every use.

For that, the user must select the menu button from the report card and then click on the Send Feedback button.

The user clicks on the send feedback button

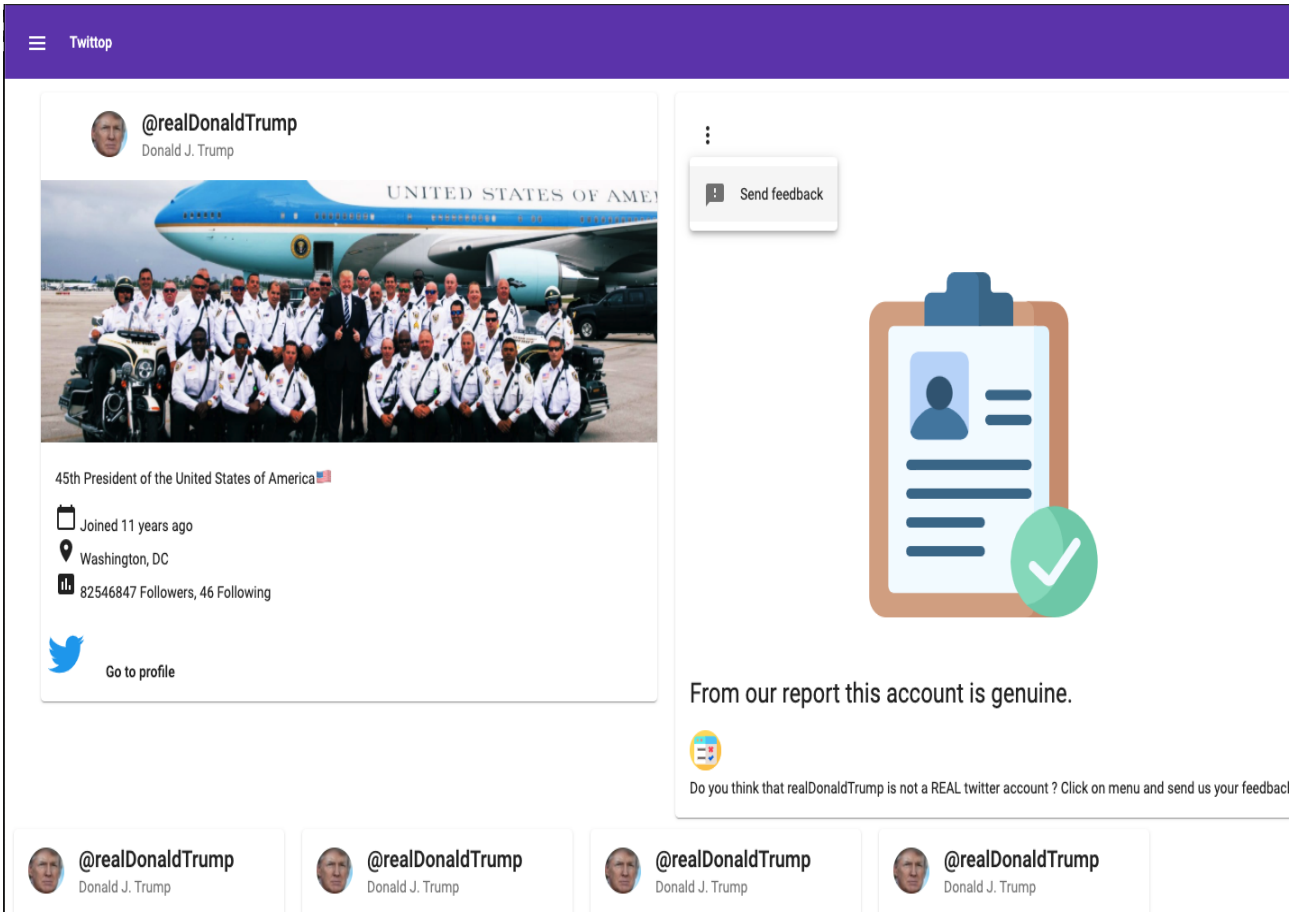

Figure 6.5: User send feedback

A confirmation dialogue will be showed to prevent accidental clicks.

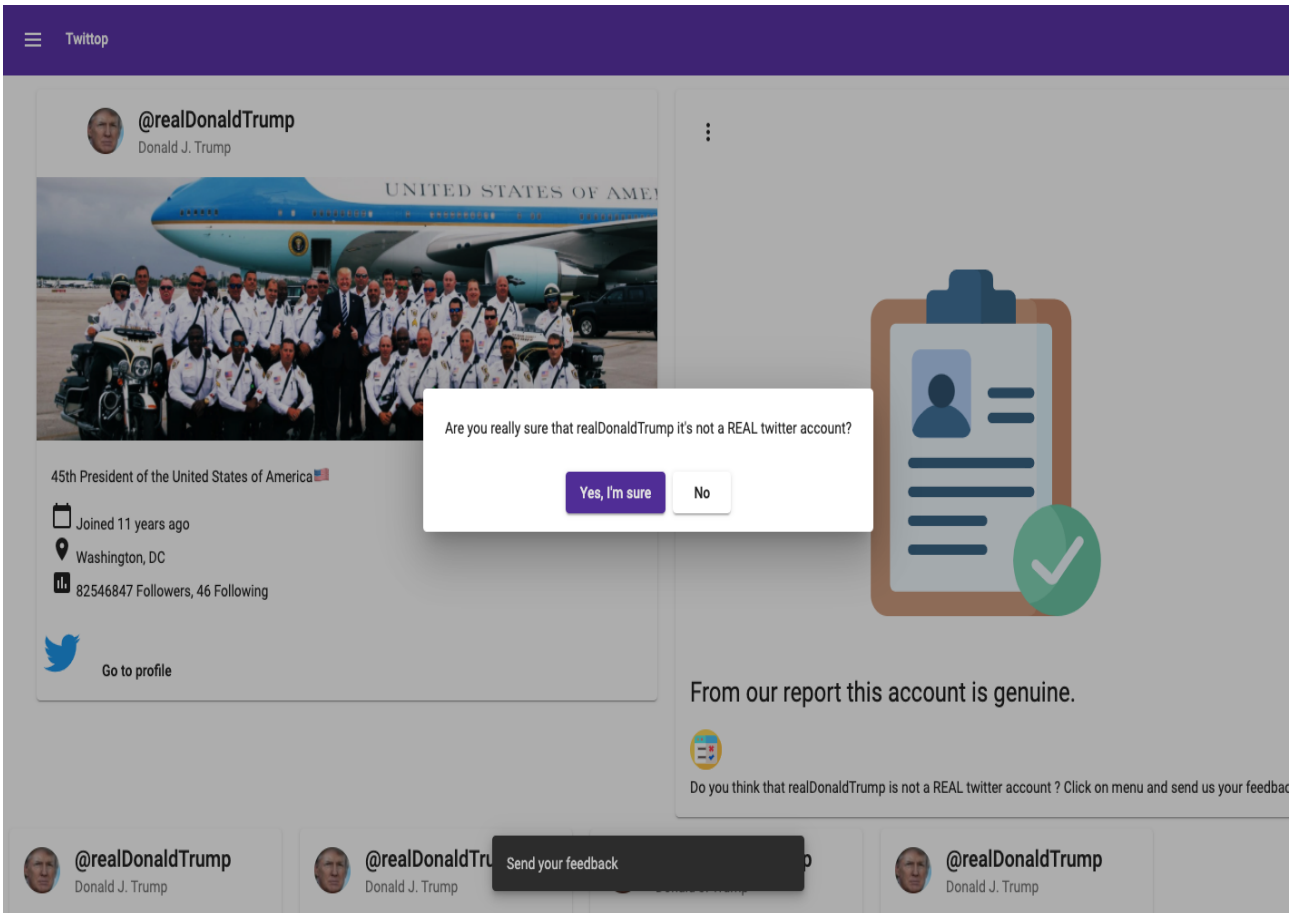

Figure 6.6: User send feedback

The user clicks on the confirmation button.

On the bottom of the page will be shown a confirmation message when the feedback is received on the server.

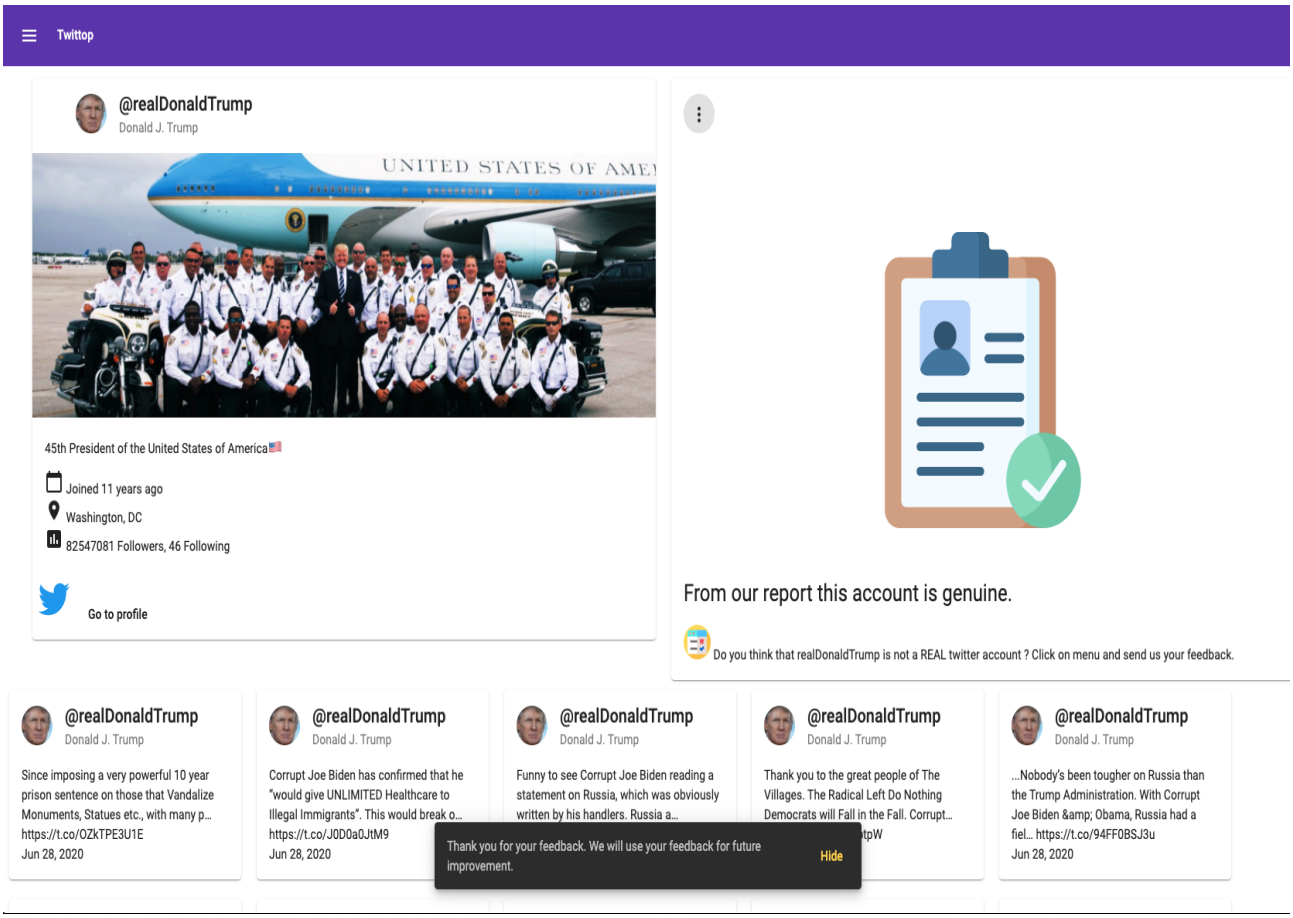

Figure 6.7: User send feedback

Chapter 7

Developer Guide

In this chapter, I will explain how the application can be used in the local environment.

Hardware requirements to run the application in the development mode

- **Processor:** 2 gigahertz (GHz) or faster processor
- **RAM**: 8 GB

Software requirements:

- **Operating system:** Linux, MacOS or Windows
- **NodeJS, Angular=9.0**
- **NPM**
- **Python=3.6.9**

The Source code can be found here `https://github.com/emanuelcovaci/twittop`

7.1 Installing angular dependencies

Clone the repository. For that, you can easily use the git command

```
#!/bin/bash
git clone https://github.com/emanuelcovaci/twittop
```

Listing 7.1: Clone the repository

And then navigate to twittop directory

```
#!/bin/bash
cd twittop
```

Listing 7.2: Change directory

And run

```
#!/bin/bash
npm install
```

Listing 7.3: Install npm packages

After the installation process is done you must to run the angular server

```
#!/bin/bash
ng serve
```

Listing 7.4: Install npm packages

Then you must open a new terminal and run the node server proxy. For that you must sign in to your account on the Twitter developer platform and from there you will receive key access.

When Twitter approved your aplication you must to create a file in directory twittop named twitter-server.json

```
  <!-- Example of twitter-server.json file-->
{
  "consumerKey": "your-consumerKey",
  "consumerSecret": "your-consumerSecret",
  "accessToken": "your-accessToken",
  "accessTokenSecret": "your-accessTokenSecret",
  "port": "7890"
}
```

After that, you must run the server proxy

```
#!/bin/bash
twitter-proxy twitter-server.json
```

Listing 7.5: Run the server proxy

Then you must to open a new terminal and run the flask server.

```
#!/bin/bash
#From the root directory
cd flask-server
export FLASK_ENV=development
export FLASK_APP=app.py
flask run
```

Listing 7.6: Run the flask server

Now that the application is up and running, access the following link in browser to use the application `http://localhost:4200/`

Chapter 8

Future work and Improvements

In the future, I want to integrate machine learning with other online social media platforms such as Facebook, Instagram.

I want to create a general solution able to detect if an account is fake or not regardless of the platform.

Another functionality I want to develop is represented by integration with chatbot. I want to create a chatbot that is connected to the server end-point.

Also, I want to create a web browser extension that can be installed on all internet browsers and when a user accesses a profile it will automatically check if the account is fake or genuine without any other interaction needed form the user.

Bibliography

[1] Number of monthly active facebook users worldwide as of 1st quarter 2020. *www.statista.com*, 2020.

[2] Average number of social media accounts per internet user from 2013 to 2018. *www.statista.com*, 2020.

[3] How to detect and prevent fake account creation on your websites and apps. *DATADOME*.

[4] Facebook removes 3.2 billion fake accounts, millions of child abuse posts. *Reuters*, 2019.

[5] Guy Rosen. Community standards enforcement report, november 2019 edition. *Facebook*, 2019.

[6] Udi Weinsberg Adam Breuer, Roee Eilat. Detecting fake accounts on social networks with sybiledge. *Facebook Research*, 2020.

[7] Daniel Faggella. What is machine learning? *merj.com*, 2020.

[8] What is machine learning? a definition. *expertsystem*, 2020.

[9] Keras. *Keras*, 2020.

[10] Flask. *The Pallets Projects*.

[11] Tweet objects. *Twitter Developer*.