C3879C - Capstone Project

AY2021 Term 3

Coursework Final Submission

Personal Details	
Name	**Benjamin Tan Khoon Siong**
Student No.	**20067332**

Compliance Statement	
Plagiarism I declare that this report is my original work. I understand that if I am suspected of plagiarism, my enrolment in the programme may be terminated.	✓
Retention of Backup Copy I declare that I have a back-up electronic copy of this report for immediate submission.	✓
Signature	
IMPORTANT: Non-compliance to these clauses will result in unconditional rejection of your submission	

School of Infocomm (SOI)

Republic Polytechnic

TWITTER HUMAN/BOT CLASSIFICATION WITH MACHINE LEARNING TECHNIQUES

Date of Submission: 03-Oct-2021

Submitted By:

20067332 Benjamin Tan Khoon Siong

School of Infocomm (SOI)

Republic Polytechnic

Abstract

This capstone project attempts to design a binary classifier trained via ML techniques to predict whether a twitter account, based off a couple of twitter account parameters, is a bot or not a bot. The project used pre-labelled humans and bots dataset from the cresci-2017 dataset hosted on Bot-O-Meter[1], a joint project of the Observatory on Social Media and the Network Science Institute at Indiana University to detect twitter bots. The features used for training are nine account-based features such as the age of the account, whether there are description and profile banner as well as normalised over the account age, metrics such as follower counts, favourites count etc. The hypothesis is that bots being mass produced, are likely to not have description or banners and that they are likely to have low follower, friends, and favourites count. K-Fold cross validation training and testing were conducted on two sets of data: One based on dumb follower bot accounts and the other on traditional spambots. Four ML models were evaluated, namely (1) Log Regression, (2) Random Forest, (3) Support Vector Machine and (4) KNN. All four models have high levels of precision, recall and F-1 Score with Random Forest having a near 99% in all three metrics. Hyper parameters tuning via RandomSearchCV was conducted on all four models with very slight increase in performance. A separate python script was created to allow users to input the ID of a twitter user. This script uses the Random Forest model for prediction on whether the account is a bot or not. Unfortunately, the model performs poorly when deployed in the real environment with its inability to predict and detect bots. This is likely due to the overly simplistic nature of bots from the training dataset which was in 2017. Newer studies in 2020 and 2021 highlighted that twitter bots have grown far more sophisticated and simple user metrics are no longer diagnostic for bot classifiers. Nevertheless, this project highlights the need to move beyond simplistic features for bot classifier and study into analysing the network of followers/friends/favourites beyond the account itself.

[1] https://botometer.osome.iu.edu/bot-repository/datasets.html

Acknowledgement

I would like to give thanks to the lecturers for the Pattern Recognition model for the useful reference codes provided in the weekly lessons and my mentor, Mr. David Leong for the comments. My only regret is that given my extremely heavy work and personal commitment over the past two months, I did not have much capacity to engage with my Mentor.

Background

It is generally acknowledged that state and non-state actors are actively waging misinformation campaigns and information operations to subtly shape the social media narrative in the target population. Given the expanding volume of social media data and the more complex information landscape, the development of such a detector would enable an analyst to detect signs of such operations and recommend counter measures if required. If the detector is simple to use and easy to understand, this tool can be improved through open-source development and deployed for easy usage by the population.

This project limits the social media landscape to Twitter and will utilise pre-curated datasets from Bot-O-Meter to get labelled datasets for both human and bot accounts. Twitter is a good place to start as they have well-documented API for researchers to extract required data from public Twitter accounts. Initially, this project seeks to extract relevant features using the Twitter API from both human and bot twitter accounts based on the ID provided in more recent datasets such as the 2020 astroturf dataset[2] on Bot-O-Meter but it was found that most of the bot accounts were already suspended by Twitter. Hence this project used the cresci-2017 dataset[3] instead, which already had the account features pre-extracted.

This paper used four ML techniques which will be explained in the following sections to train models based on the dataset and evaluates the models on a hold-out set. The project also provides a python script for users to input either the Twitter User ID or the ID number and the script using the trained model, outputs the prediction whether the account is a bot or not.

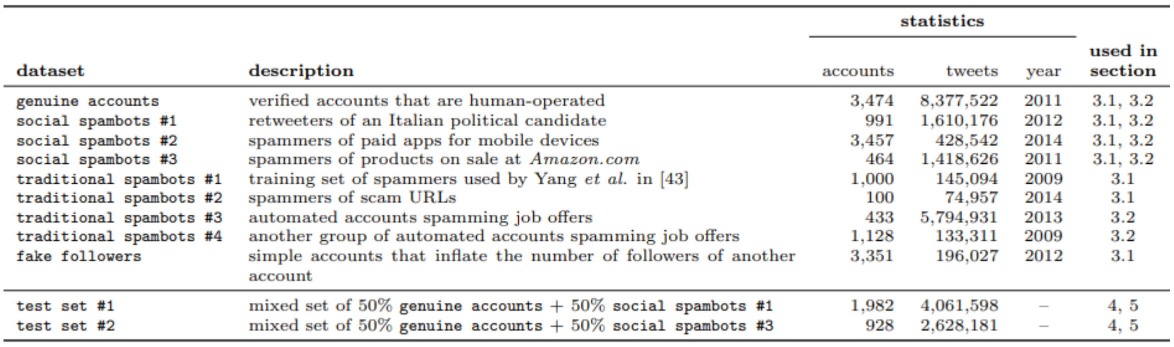
[2] https://botometer.osome.iu.edu/bot-repository/datasets/astroturf/astroturf.tar.gz
[3] https://botometer.osome.iu.edu/bot-repository/datasets/cresci-2017/cresci-2017.csv.zip

Methodology and Design

DATASETS

As described in the background, we used the pre-labelled cresci-2017 dataset. This dataset was used in a 2017 study to analyse if Twitter bot behaviours have evolved (Cresc, Di Pietro, & Petrocchi, 2017). Referring from this paper, the datasets provided are as such (see **Figure 1**):

Figure 1: Cresci-2017 Datasets



		statistics			
dataset	description	accounts	tweets	year	used in section
genuine accounts	verified accounts that are human-operated	3,474	8,377,522	2011	3.1, 3.2
social spambots #1	retweeters of an Italian political candidate	991	1,610,176	2012	3.1, 3.2
social spambots #2	spammers of paid apps for mobile devices	3,457	428,542	2014	3.1, 3.2
social spambots #3	spammers of products on sale at *Amazon.com*	464	1,418,626	2011	3.1, 3.2
traditional spambots #1	training set of spammers used by Yang *et al.* in [43]	1,000	145,094	2009	3.1
traditional spambots #2	spammers of scam URLs	100	74,957	2014	3.1
traditional spambots #3	automated accounts spamming job offers	433	5,794,931	2013	3.2
traditional spambots #4	another group of automated accounts spamming job offers	1,128	133,311	2009	3.2
fake followers	simple accounts that inflate the number of followers of another account	3,351	196,027	2012	3.1
test set #1	mixed set of 50% genuine accounts + 50% social spambots #1	1,982	4,061,598	–	4, 5
test set #2	mixed set of 50% genuine accounts + 50% social spambots #3	928	2,628,181	–	4, 5

With the considerations of having a large enough dataset and ensuring that both classes (humans and bots) are sufficiently represented, this project used the *genuine accounts* and *fake followers* dataset which has about 3000 accounts each.

This project's hypothesis is that Twitter Bot accounts are likely to be lacking in personalised details and have low levels of connections with other accounts as their purpose is to spam likes or retweet certain narratives. To this effect, this project is looking for two types of data to be used as features for training through ML models.

(1) Personalised Data: Whether the account is geo-enabled, has profile description and whether the account uploaded an account banner

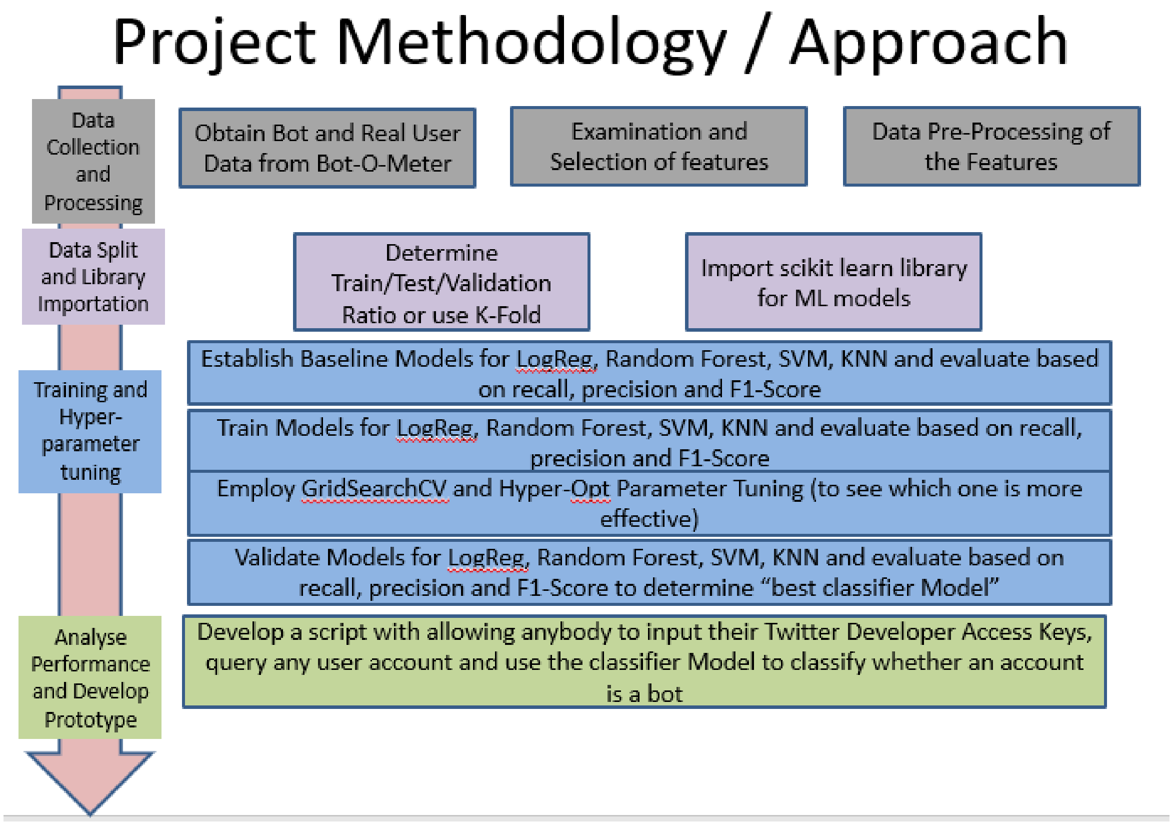
(2) Connection Metrics: how many followers, how many tweets, how many friends and how many following

(3) Time: In order to provide standardised metrics for connection, there is a need to know when the account was created so that the connection metrics can be normalised by time

METHODOLOGY

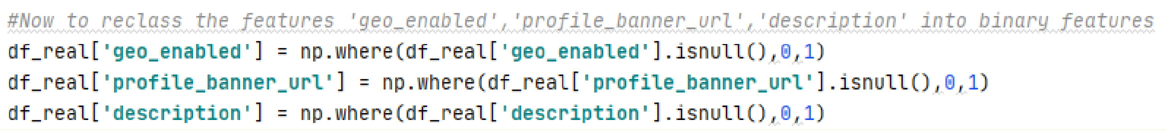
Figure 2 shows the general methodology adopted for this project.

Figure 2: General Methodology


Project Methodology / Approach
Data Collection and Processing
Obtain Bot and Real User Data from Bot-O-Meter
Examination and Selection of features
Data Pre-Processing of the Features
Data Split and Library Importation
Determine Train/Test/Validation Ratio or use K-Fold
Import scikit learn library for ML models
Training and Hyper-parameter tuning
Establish Baseline Models for LogReg, Random Forest, SVM, KNN and evaluate based on recall, precision and F1-Score
Train Models for LogReg, Random Forest, SVM, KNN and evaluate based on recall, precision and F1-Score
Employ GridSearchCV and Hyper-Opt Parameter Tuning (to see which one is more effective)
Validate Models for LogReg, Random Forest, SVM, KNN and evaluate based on recall, precision and F1-Score to determine "best classifier Model"
Analyse Performance and Develop Prototype
Develop a script with allowing anybody to input their Twitter Developer Access Keys, query any user account and use the classifier Model to classify whether an account is a bot

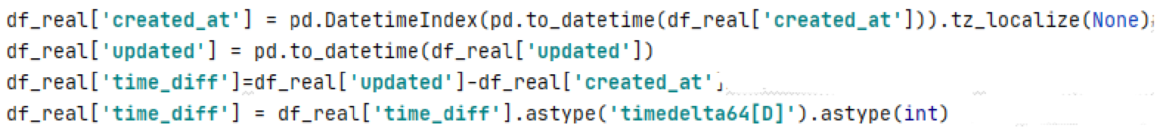
Data-Preprocessing. Both the *genuine accounts* and *fake followers* dataset have the above required features but they require some form of data manipulation in order to use them for training purposes. First off, the personalised data consists of text strings for the 'profile description field' and URLs for the 'banner field'. These will be recast into 1 and 0 to present the existence of description and banners for 1 and the lack of for 0. Second, there is a need to transform the time fields into suitable datatime format for subtraction to obtain the age of the account by days. Lastly, the connection metrics need to be normalised by dividing by account age and scaled to between 0 to 1 so that these features do not overwhelm the personalised data (see **Figure 3-5**). [*Refer to capstone_data_pre_processing.py*]

Figure 3: Recasting to 0 and 1s



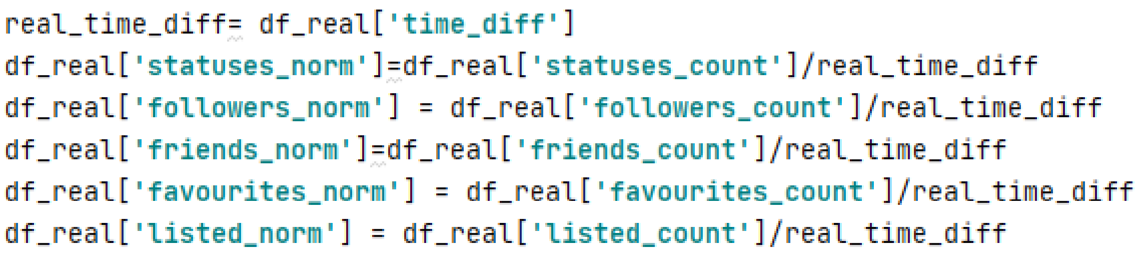
```
#Now to reclass the features 'geo_enabled','profile_banner_url','description' into binary features
df_real['geo_enabled'] = np.where(df_real['geo_enabled'].isnull(),0,1)
df_real['profile_banner_url'] = np.where(df_real['profile_banner_url'].isnull(),0,1)
df_real['description'] = np.where(df_real['description'].isnull(),0,1)
```

Figure 4: Getting Age of Account by Days

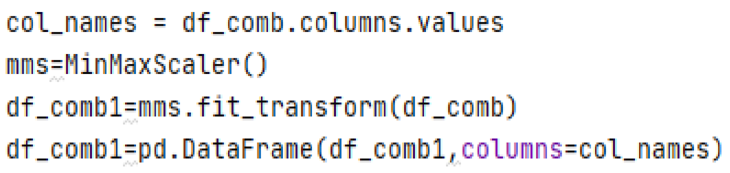


```
df_real['created_at'] = pd.DatetimeIndex(pd.to_datetime(df_real['created_at'])).tz_localize(None)
df_real['updated'] = pd.to_datetime(df_real['updated'])
df_real['time_diff']=df_real['updated']-df_real['created_at'
df_real['time_diff'] = df_real['time_diff'].astype('timedelta64[D]').astype(int)
```

Figure 5: Normalising and Scaling of Features



```
real_time_diff= df_real['time_diff']
df_real['statuses_norm']=df_real['statuses_count']/real_time_diff
df_real['followers_norm'] = df_real['followers_count']/real_time_diff
df_real['friends_norm']=df_real['friends_count']/real_time_diff
df_real['favourites_norm'] = df_real['favourites_count']/real_time_diff
df_real['listed_norm'] = df_real['listed_count']/real_time_diff
```



```
col_names = df_comb.columns.values
mms=MinMaxScaler()
df_comb1=mms.fit_transform(df_comb)
df_comb1=pd.DataFrame(df_comb1,columns=col_names)
```

CROSS-VALIDATION

Given the manageable size of the data at around 7000 data points with 9 features each, this project used K-fold cross validation where k=4 for the training of the data. To reduce the risk of overfitting and creating bias during evaluation, I split 20% of the data to be a hold-out set. K-Fold was then executed on the 80% training data for all four models to ensure maximum usage of all the data in training. [*Refer to capstone_crossValidation.py*]

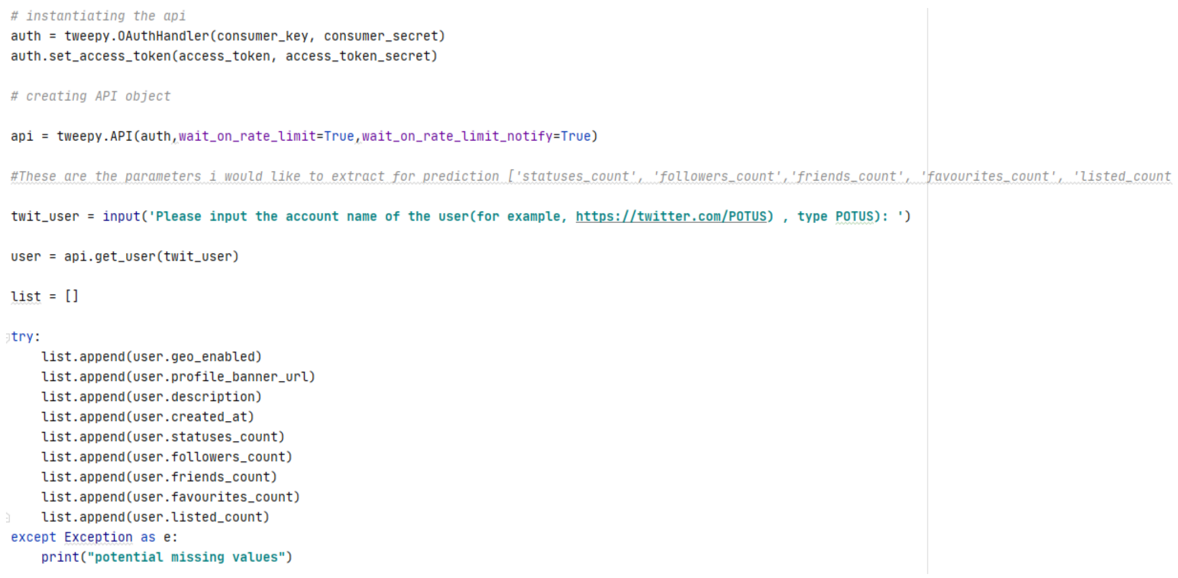
MODELS TRAINING AND OPTIMISATION

This project made extensive use of the sklearn libraries for ready made models for logistic regression, random forest, SVM and KNN. For hyperparameter tuning, this project used GridSearchCV from the sklearn libraries.

PROTOTYPE

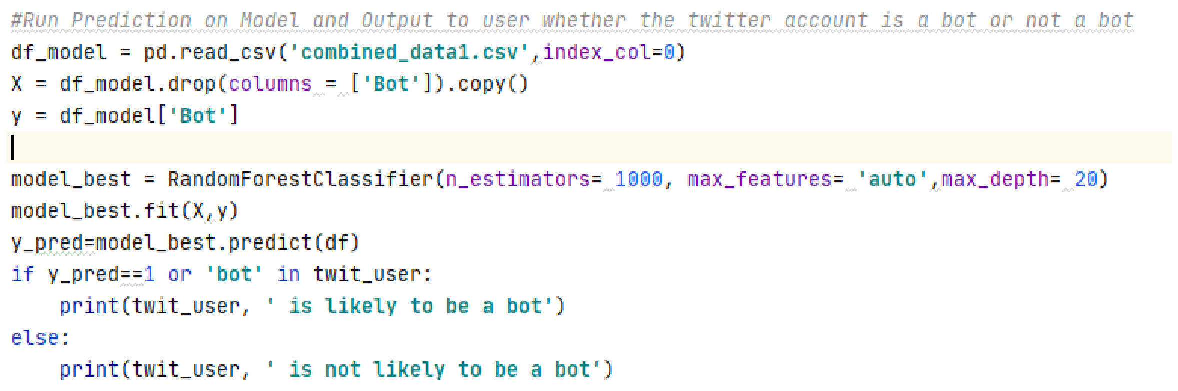
After selecting the best performing model based on the evaluation metrics of ‘recall’,’precision, and ‘F1-Score, this model will be included as part of a function that will be called to predict whether a user’s provided account name is a bot or not. To obtain the features from the account required to run the model’s prediction, this project has obtained a Twitter developer API account to allow for a python script to extract the required features via the API. Upon the user’s input of an account name, the python script will call on the Twitter API through the tweepy library, pull out the necessary information, apply the data processing steps required to recast, normalise and scale the input data and then output its prediction (see **Figure 6**).

Figure 6: Twitter API Call for Required Feature Extraction



```
# instantiating the api
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)

# creating API object

api = tweepy.API(auth,wait_on_rate_limit=True,wait_on_rate_limit_notify=True)

#These are the parameters i would like to extract for prediction ['statuses_count', 'followers_count','friends_count', 'favourites_count', 'listed_count

twit_user = input('Please input the account name of the user(for example, https://twitter.com/POTUS) , type POTUS): ')

user = api.get_user(twit_user)

list = []

try:
    list.append(user.geo_enabled)
    list.append(user.profile_banner_url)
    list.append(user.description)
    list.append(user.created_at)
    list.append(user.statuses_count)
    list.append(user.followers_count)
    list.append(user.friends_count)
    list.append(user.favourites_count)
    list.append(user.listed_count)
except Exception as e:
    print("potential missing values")
```

Figure 7: Output



```
#Run Prediction on Model and Output to user whether the twitter account is a bot or not a bot
df_model = pd.read_csv('combined_data1.csv',index_col=0)
X = df_model.drop(columns = ['Bot']).copy()
y = df_model['Bot']

model_best = RandomForestClassifier(n_estimators= 1000, max_features= 'auto',max_depth= 20)
model_best.fit(X,y)
y_pred=model_best.predict(df)
if y_pred==1 or 'bot' in twit_user:
    print(twit_user, ' is likely to be a bot')
else:
    print(twit_user, ' is not likely to be a bot')
```

Findings

The baseline models for all four ML techniques performed well, with all four exceeding 95% in the three-evaluation metrics (precision, recall, F1-Score) for both a simple train-test split, and a K-Fold spilt with testing done on the hold-out set (see Table 1). The best performing model is the Random Forest Classifier model which achieved 99%. [*Refer to LogReg.py, random_forest.py, SVM.py and KNN.py*]

Table 1: Consolidation of Results(K-Fold=4)

Model (Default parameters)	**mPrecision Score**	**mRecall Score**	**mF1 Score**
Logistic Regression	0.963	0.977	0.970
Random Forest	0.996	0.990	0.993
SVM	0.960	0.978	0.969
KNN	0.970	0.979	0.975

While the models are already performing well, for the sake of completeness, the project executed tuning via RandomisedSearchCV with parameters n_iter=20 and cv=4, giving 80 random combinations of parameters. The best parameters for the models are:

(1) Random Forest- 'n_estimators': 1000, 'max_features': 'auto', 'max_depth': 20
(2) SVM- 'kernel': 'poly', 'gamma': 1, 'C': 100
(3) KNN- 'weights': 'distance', 'n_neighbors': 5, 'leaf_size': 40, 'algorithm': 'auto'

Table 2: Consolidation of Results(After Optimisation)

Model (Default parameters)	**mPrecision Score**	**mRecall Score**	**mF1 Score**
Logistic Regression	N/A		
Random Forest	0.996	0.991	0.993
SVM	0.969	0.981	0.975
KNN	0.973	0.982	0.977

As the best performing model is the Random Forest Classifier, this model was chosen to be the final predictor model. [*Refer to new_Twitter_prediction_Script.py and test_prediction_bot_dataset.py*]

Evaluation and Analysis

To simulate the deployment of this dataset on new operational data, I used another dataset, botwiki-2019 from Bot-O-Meter to evaluate the performance of the predictor on unseen data. Unfortunately, the predictor completely fails to predict a single bot out of a random sample of 50 from the data (4 accounts were suspended). Thinking that this could be an artifact of the model training on the more simplistic *fake_followers* dataset, I used the *traditional spambots #1* dataset and retrain the Random Forest model. This did not cause any change to the prediction model. As explained in (Cresc, Di Pietro, & Petrocchi, 2017), by the time the study released, then state of the art bot detectors such as Bot-or-Not?, and various academical models with supervised and unsupervised training all had terrible performances against the 2017 bots, having misidentified most of them to be genuine. By 2019, bots are likely to be more sophisticated and may have used randomly generated phrases using NLP for their description, the uploading of a random banner, using NLP to generate status updates (which may be nonsensical) that simple models like this project would not be able to pick up on. Existing bot detectors also face issues with scalability given Twitter API call limit caps and the dynamic nature of bot designs mean that designers are constantly trying to outwit the latest bot detection models.

Conclusion

To sum out, the project set off to create a binary classifier for humans or bot twitter account. The ML models applied to the training dataset were effective in creating a well performing classifier for data from 2017 but when applied to newer generations of known twitter bot accounts, the model completely fails. This highlights the difficulty in designing an effective bot detector as even then state of the art bot detectors also failed miserably when applied against newer bots. It just simply means that the diagnostic features in 2017 is no longer applicable to modern bots.

Recommendations

Moving ahead, the results of this study showed that any Twitter Bot detector needs to re-examine what behaviours that only bots can do and humans cant and extract those behaviours for ML training. A bot detector also needs to solve the generalisation problem and yet at the same time, comply with the strict Twitter API limitations in the number of calls per day. To expand on this study, a modern bot detector would need to tap onto analysis of the tweets itself using NLP methods such as whether the tweet makes logical sense, whether the tweet are always perfectly punctuated or spelt or if the tweet makes sense in context.

Appendices

This is a short description of the dataset file names and their purpose:

S/N	**File name**	**Purpose**
1	real_users.csv	Raw data from *genuine accounts* dataset
2	fake_users.csv	Raw data from *fake followers* dataset
3	spam_users.csv	Raw data from *traditional spambots #1* dataset
4	botwiki-2019.tar	Raw data from bot-O-meter botwiki 2019 dataset
5	capstone_data_pre_processing.py	Data pre processing script
6	capstone_crossValidation.py	Cross validation script
7	LogReg.py	Logistic Regression Script
8	random_forest.py	Random forest classifier script
9	SVM.py	SVM script
10	KNN.py	KNN script
11	new_Twitter_prediction_Script.py	Function defining prediction model
12	test_prediction_bot_dataset.py	Twitter API call on user input and then run prediction model on input

References

Bot-O-Meter. (n.d.). *Bot Repository*. Retrieved from Bot-O-Meter: https://botometer.osome.iu.edu/bot-repository/datasets.html

Cresc, S., Di Pietro, R., & Petrocchi, M. (2017, Jan 11). *The paradigm-shift of social spambots.* Retrieved from arxiv: https://arxiv.org/pdf/1701.03017.pdf