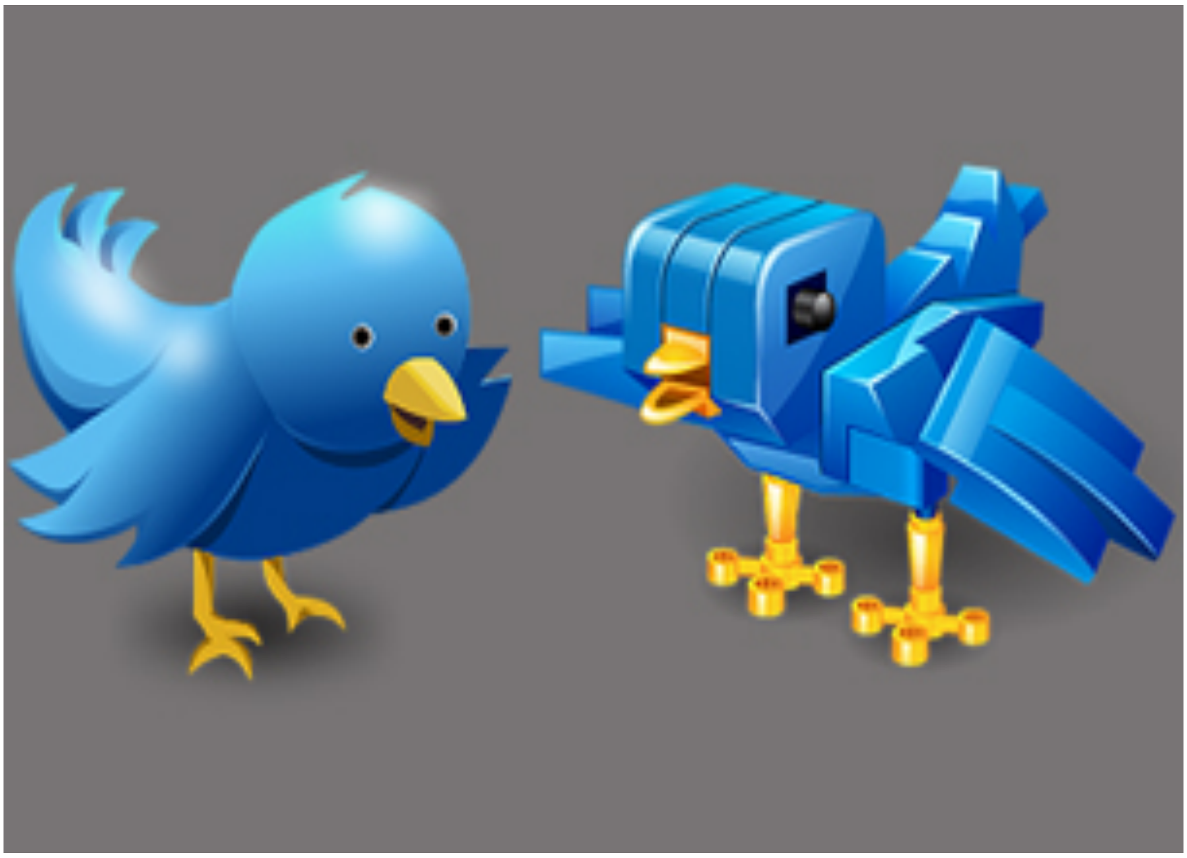
Twitter Bots Detection



(From: adweek.com)

For each member of the group:

Name and University ID:

Yuanxin Ye U5669371

Mulong Xie U6462764

Zehao Shi U6462802

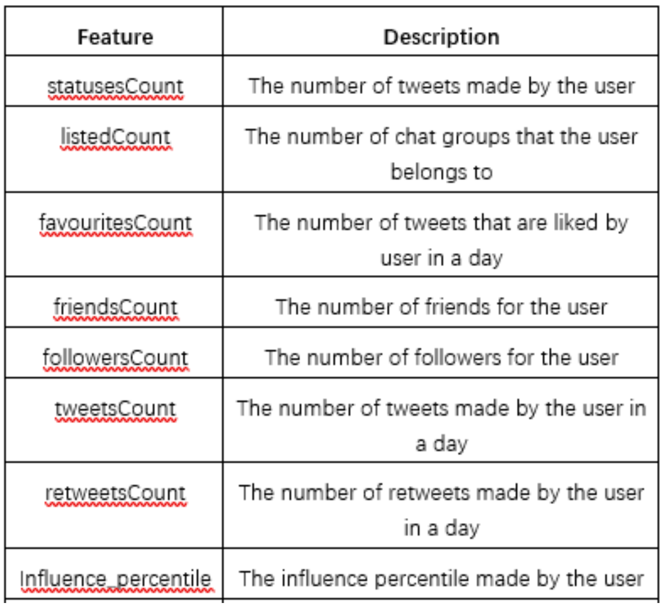
1 Introduction

To detect the bot tweet account, based on a range of selected features, we utilize various sampling techniques, machine learning algorithms and build a neural network. The assessment and comparison are given in this report.

2 Data

2.1 Feature Selection

In the original dataset, 100,000 users are given along with their personal information including 28 features. However, some features are meaningless and even have a negative impact on training accuracy. According to our training and experiment, the features shown in table below are highly correlated to the model performance.

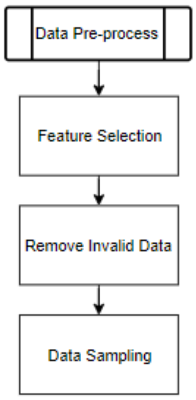


Feature	Description
statusesCount	The number of tweets made by the user
listedCount	The number of chat groups that the user belongs to
favouritesCount	The number of tweets that are liked by user in a day
friendsCount	The number of friends for the user
followersCount	The number of followers for the user
tweetsCount	The number of tweets made by the user in a day
retweetsCount	The number of retweets made by the user in a day
Influence_percentile	The influence percentile made by the user

To identify the social media bots effectively, two basic areas need to be analyzed: users' friend and activity. [1]
For user's friends, the followers and friends of a user is an effectively way to distinguish bots and real man. Normal users are usually using the social media application to socialize and interact with other users. For user's followers, they are those who are not friends of the user, which indicates the number of user's "fake" friends. For bots, there are two possibilities — they always either have few numbers of friends or have a huge number of friends[2]. A normal man is not likely to be friends with a bot because bots are assigned typical tasks in social media platform such as spreading misinformation which are not usually the same as the purpose of the normal user.
For user's activity, normal users' activity is certainly different from the bots' activity. Normal users have appropriate habits to send tweets, retweets, and like tweets while the bots may have their pre-set habits[3]. For example, normal users are not likely to send a large number of tweets in a day as this is time-consuming and annoying. However, if a user sends a hundred of tweets or retweets in a day, then the user likely to be a bot. In addition, statusesCount, listedCount, influence percentile are also essential features to differentiate human and bots because these are the features that reflects the account activity.

2.2 Data Preprocessing

In the given data set, there are lots of usable data which need to be cleaned before further process. Besides, one of the most significant challenges is the imbalanced data, which consists of 90% normal user accounts while the share of bot accounts is only around 10%. Therefore, preprocessing is indispensable in this case to perform better.

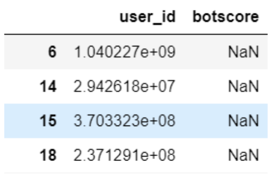


Flowchart of preprocessing

2.2.1 Remove Invalid Data

At the beginning, we convert the data type of botscore from series type to numeric type and drop the null value of botscore and decide the is_bot attribute according to the given threshold 0.5.

There are lots of unusable and invalid data as their "botscore" is NaN, which makes them untrainable. Around 80,000 data left among 100,000 training data.



	user_id	botscore
6	1.040227e+09	NaN
14	2.942618e+07	NaN
15	3.703323e+08	NaN
18	2.371291e+08	NaN

2.2.2 Extract relation between features

If we look at some feature individually, for example 'followersCount', we can notice that for both bot and non-bot account, the difference between data is not that obvious. So, we assume that the bias may exists on the number of followers and friends, and the number of tweet and retweet.

To extract this relationship, we apply a simple approach by using the formula

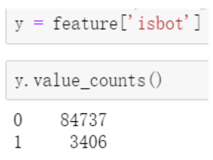
(followersCount)/ (friendsCount + 1)   (Note: plus 1 is to avoid Zero on denominator)*

(tweetsCount)/ (retweetsCount + 1)

2.2.3 Challenge: Imbalanced Dataset

As mentioned previously, one of significant challenges stems from the imbalanced dataset, which consists of the certain class set that is much larger than the other.

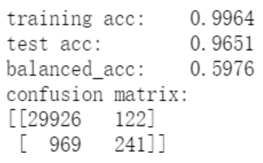
The imbalanced data would lead to a low balanced accuracy even though the overall accuracy is high. Take this assignment for example, the dominant class – "not bot" accounts for more than 90% of the whole dataset while the scale of the "bot" is much smaller.[4]



```
y = feature['isbot']

y.value_counts()

0     84737
1      3406
```



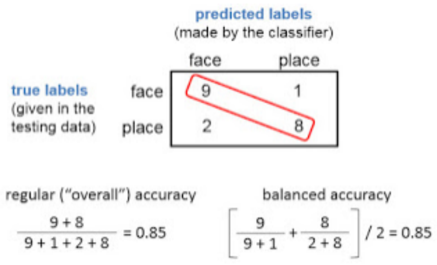
```
training acc:     0.9964
test acc:         0.9651
balanced_acc:     0.5976
confusion matrix:
[[29926    122]
 [  969    241]]
```

not_bot : bot = 84737 : 3406 in the cleaned dataset

Although the overall accuracy reaches 0.96, the balanced accuracy is much lower as the imbalanced dataset

Assessed on the basis of the confusion matrix structure given below and the formulae to calculate balanced and overall accuracy, the reason of the problem could be comprehended.

Human => Detect as human	Human => Detect as bot
Bot => Detect as human	Bot => Detect as bot


predicted labels
(made by the classifier)
face
place
true labels
(given in the testing data)
face
place
9
1
2
8
regular ("overall") accuracy
$\frac{9+8}{9+1+2+8} = 0.85$
balanced accuracy
$\left[\frac{9}{9+1} + \frac{8}{2+8}\right] / 2 = 0.85$

Formulae of accuracy[5]

2.2.4 Data Sampling

One way to conquer this issue is data sampling, a technique to reorganize the dataset by increasing or decreasing the proportion of data in certain class.

1. Oversampling: copying the minority class to boost its share
2. Undersampling: randomly remove part of majority to reduce its share
3. SMOTE (Synthetic Minority Over-sampling Technique):[4,6]
   - L: points in majority; S: points in minority
   - For each p in S, calculate k nearest neighbors of p within S
   - For the k points, draw line between each point in k and p
   - Randomly select a point r in this line and put r into S as new point of minority
4. Adjust Weight: change the weight of each class to offset the imbalance

2.2.5 Experiments

The experiment result is presented in the table, all based on basic Decision Tree model.

Preprocess	Overall Accuracy	Balanced Accuracy	Confusion Matrix
Nothing	0.9686	0.493	array([[19363, 0], [637, 0]]),
Replace nan with average	0.9603	0.495	array([[19199, 24], [776, 1]]),
Set class weight 1:25	0.8525	0.532	array([[13205, 6018], [522, 255]]),
Oversampling: copy 3 times	0.8760	0.573	array([[17064, 181], [2300, 455]]),
Oversampling: copy 5 times	0.8369	0.669	array([[15014, 1135], [2276, 1575]]),
Oversampling: copy 3 times + Undersampling * 0.3	0.8016	0.679	array([[12592, 2510], [2328, 2570]]),
SMOTE	0.9403	0.658	[[29191 857] [889 321]]

The result of experiments of different preprocess methods give us the following intuitions:

1. The balanced accuracy increases when the proportion of data in two classes gets close
2. The overall accuracy decreases at the same time as the overfit of majority class is alleviated

3 Model

Various machine learning algorithms are applied to detect the bot account, and several neural network models are built to boost the performance.

3.1 Why Python

Python is selected as the language to implement our ideas on account of several concerns.

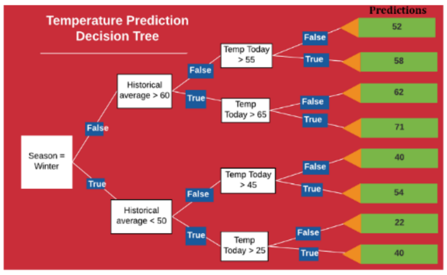
1. Powerful, abundant and convenient packages such as keras, sklearn, matplotlib and so on, which save plenty of time for us.
2. We are already familiar with this language rather than R. Since the learning focus of this assignment is not on programming language, python avoid unnecessary learning time.
3. Vigorous community, as lots of popular ML frame such as keras and Tensorflow are straightforward to use in python, attracts a mass of developers to discuss and answer question.

3.2 Machine Learning Model (Regression and Classification)

Before building and selecting the final model, several choices have been considered and tried, each of them has their pros and cons.

Decision Tree:

A simple supervised learning and decision support model. Utilized by us to serve as a testing tool.[4,7]



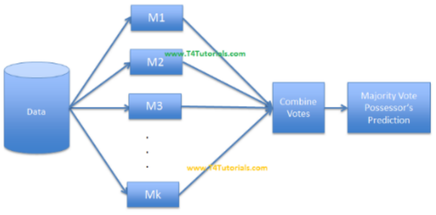
Pros: Fast (3.5s for 60,000 data); Easy to build and use.

Cons: Because of the relatively simple complexity, the overfit of majority class is not alleviated well and a low balanced accuracy is introduced.

8

Bagging (Bootstrap Aggregating):

This algorithm is composed of several independent classifiers and synthesis them after being trained.



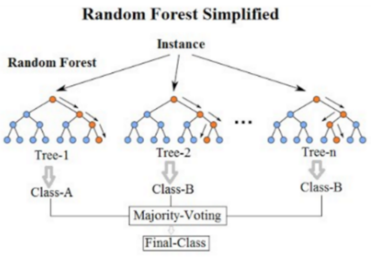
Pros: Boost the performance in terms of stability and precision; Reduce the Variance; Overcome the overfit up to a point.

Cons: Performance depends on individual classifiers; Take time to train. [4,9]

9

Random Forest:

Similar to the Bagging algorithm, each individual classifier, in this case, is a tree.



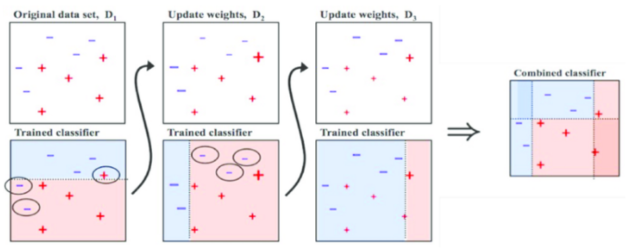
Pros: Similar to Bagging; Better performance for imbalanced dataset; It has the ability to inspect the variable interactions.

Cons: Similar to Bagging.[10]

11

Ada Boost:

Another Algorithmic Ensemble Technique, unlike Bagging, it has interaction among each sub-classifiers and calculate the residual of function through gradient descent algorithm.



Pros: Individual classifiers can communicate to others and impact the further parameters; Better adaptability and stability.

Cons: More difficult to build; Take long time to train. [12]

[13]

Support-vector machine:

SVM is a supervised learning algorithm defined by a separating hyperplane. SVM will extract the data into the higher dimensional feature space to make it possible to be separated by the plane in higher dimension.[14]

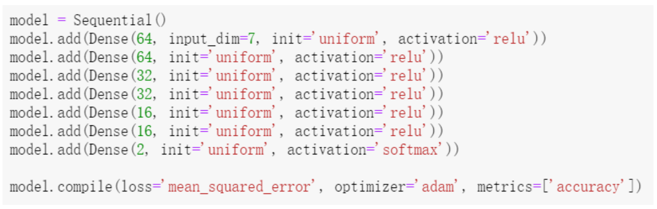
Pros: The overfitting problems can be avoided in SVM due to once the hyperplane has been found, the rest data will be redundant and will have less effect on the result.[15]

Cons: The complexity of SVM is highly dependent on the size of data set. When the size of the data set is huge, the training will take long time to complete.

3.3 Neural Network Model (Classification)

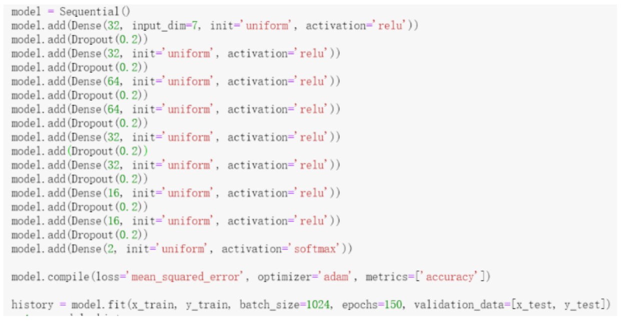
As another fashion of supervised learning, and characterized by its flexibility, the neural network may can better fit the data in situation of large scale of dataset.[16]

Several models were tried in terms of different depth and width and dropout to screen the most appropriate one.



```
model = Sequential()
model.add(Dense(64, input_dim=7, init='uniform', activation='relu'))
model.add(Dense(64, init='uniform', activation='relu'))
model.add(Dense(32, init='uniform', activation='relu'))
model.add(Dense(32, init='uniform', activation='relu'))
model.add(Dense(16, init='uniform', activation='relu'))
model.add(Dense(16, init='uniform', activation='relu'))
model.add(Dense(2, init='uniform', activation='softmax'))

model.compile(loss='mean_squared_error', optimizer='adam', metrics=['accuracy'])
```



```
model = Sequential()
model.add(Dense(32, input_dim=7, init='uniform', activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(32, init='uniform', activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(64, init='uniform', activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(64, init='uniform', activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(32, init='uniform', activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(32, init='uniform', activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(16, init='uniform', activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(16, init='uniform', activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(2, init='uniform', activation='softmax'))

model.compile(loss='mean_squared_error', optimizer='adam', metrics=['accuracy'])

history = model.fit(x_train, y_train, batch_size=1024, epochs=150, validation_data=[x_test, y_test])
```

3.4 Experiment

As the conclusion we drawn in preprocessing, the SMOTE is an effective way to sampling. Thus, all the classification experiments of various models apply SMOTE.

Classification

Model	Overall Accuracy	Balanced Accuracy	Confusion Matrix
Decision Tree	0.945	0.624	array([[32953, 968], [965, 372]]),
Bagging	0.965	0.726	confusion matrix: [[9172 444] [193 191]]
Random Forest	0.936	0.724	confusion matrix: [[9183 433] [195 189]]
Ada Boost	0.865	0.779	confusion matrix: [[26620 3484] [4618 25375]]
7layers NN	0.910	0.773	confusion matrix: [[8087 1529] [110 274]]
9layers + 0.2 Dropout NN	0.818	0.786	confusion matrix: [[8748 868] [130 254]]
9layers + 0.4 Dropout NN	0.808	0.793	confusion matrix: [[8625 991] [127 257]]
12layers + 0.4 Dropout NN	0.796	0.743	confusion matrix: [[9033 583] [174 210]]

Regression

Model	RMSE
Decision Tree	0.1060
Random Forest	0.1040
Gradient Boosting	0.1025
DNN	0.1194

The result of experiments of different preprocess methods give us the following intuitions:

1. The balanced accuracy increases followed by the increasing complexity of the model
2. The overall accuracy decreases at the same time, as the overfit of majority class is alleviated
3. In the case of basic neural network model, "deeper is better" does not hold true, because the issue of Gradient Disappear

4 Result

According to the experiments above, the optimised results for classification and regression can be visualized, with the model of 9 layers + 0.4 Dropout NN and Gradient Boosting respectively. For classification, the 9 layers + 0.4 Dropout NN model is chosen which is to ensure high balanced accuracy, so that both human and bot users can be detected at a high percentage. For Regression, the Gradient Boosting model clearly shows the lowest RMSE.

	Model	Balanced accuracy & RMSE
Classification	9 layers + 0.4 Dropout NN	0.793
Regression	Gradient Boosting	0.1025

5 Team member contribution

Mulong Xie (u6462764): classification, data preprocessing, model selection

Zehao Shi (u6462802): regression, oversampling & undersampling, SVM

Yuanxin Ye (u5669371): report writing, result analysis

6 Reference

[1] The Rise of Social Bots, Emilio Ferrara, Onur Varol, Clayton Davis, Filippo Menczer, Alessandro Flammini, July 2016
[2] https://www.distilnetworks.com/glossary/term/social-media-bots/
[3] https://en.wikipedia.org/wiki/Tweetbot
[4] https://flystarhe.github.io/docs-2014/algorithm/imbalanced-classification/readme/
[5] http://mvpa.blogspot.com/2015/12/balanced-accuracy-what-and-why.html
[6] https://www.jair.org/index.php/jair/article/view/10302
[7] https://www.jiqizhixin.com/articles/2017-07-31-3
[8] https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d
[9] https://t4tutorials.com/bagging-and-bootstrap-in-data-mining-machine-learning/
[10] https://en.wikipedia.org/wiki/Random_forest
[11] https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d
[12] https://medium.com/diogo-menezes-borges/what-is-gradient-descent-235a6c8d26b0
[13] https://medium.com/diogo-menezes-borges/boosting-with-adaboost-and-gradient-boosting-9cbab2a1af81
[14] https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72
[15] https://scikit-learn.org/stable/modules/svm.html
[16] https://natureofcode.com/book/chapter-10-neural-networks/