in the training set as we found several abused apps are developed by biggest technology companies in China.

5. Testing

5.1. Performance Metrics

For every machine learning problem, the ultimate goals are:

1. High precision and recall for training set.
2. Accurate prediction in out-of-sample data.

Usually item 1 is reasonable to reach if the training model is good enough. But item 2 is relatively difficult due to the lack of out-of-sample data. Sometimes, even if we do have out-of-sample data, evaluation is still impossible as, for supervised learning, those data are not labeled. In this specific training problem, our goals are:

1. while pushing precision and recall, precision is supposed to be better than recall. Precision is defined as

$$precision = \frac{\sum true\_positive}{\sum test\_outcome\_positive}. \quad (5)$$

In this problem, precision indicates the confidence on classified abused apps. Recall is defined as

$$recall = \frac{\sum true\_positive}{\sum total\_positive}. \quad (6)$$

In this problem, recall indicates coverage of our results. Usually, higher precision will lead to lower recall, and vice versa [29]. For abused app detection, we emphasize on higher precision since we want to make sure we minimize the probability of labeling benign apps as abused apps and we may tolerate missing several abused apps.

2. narrowing down the scope of abused app detection instead of declaring abused apps. For app market providers, labeling an abused app is very serious and sensitive and it always involves business level decisions. We hope that our machine learning results could help app market providers narrow down the scope of investigation – only have to inspect abused apps we classify.

5.2. Results

Figure 8 depicts that the precision, recall and f-score vary from top 10 features to top 55 features (all features) in iTunes US app store. We have the following observations:

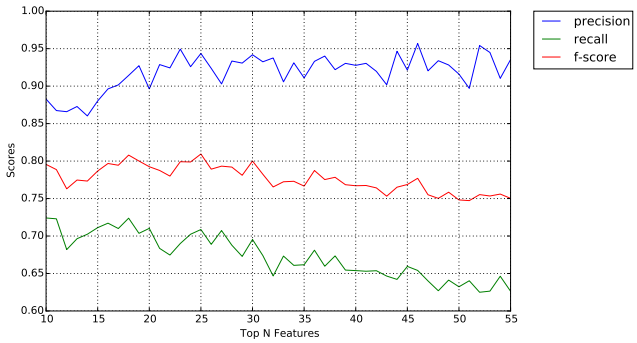

Figure 8. iTunes US App Store: Evaluation for Top n Features

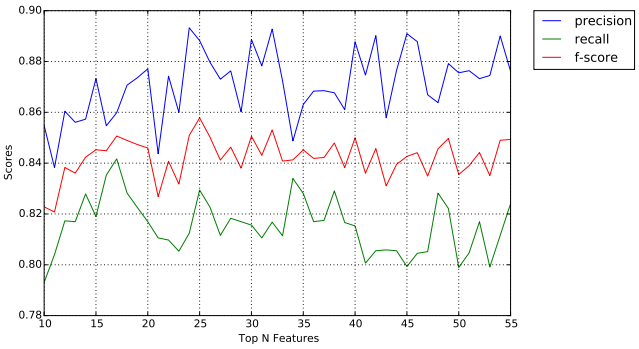

Figure 9. iTunes China App Store: Evaluation for Top n Features

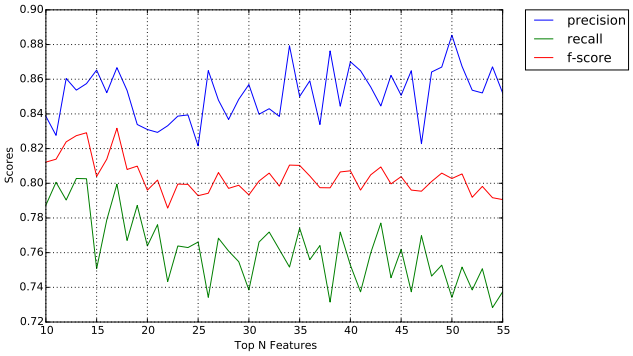

Figure 10. iTunes UK App Store: Evaluation for Top n Features

1. Generally, as we increase the number of top features, precision increases slowly while recall and f-score decrease relatively fast.
2. Precision is always better than recall, which meets our requirement.
3. When $n = 18$, both f-score and recall reach their peaks.

Figure 9 and Figure 10 are our evaluations for iTunes China app store and iTunes UK app store, respectively.

Even though the plot is very spiky for iTunes China app store, we still have the same observations for both UK and China app stores as for US app store. For Figure 10, even though f-score($n = 10$) is slightly better than f-score($n = 15$), we are still willing to choose the latter one as we refer high precision to high recall. Therefore, from both figures, we choose top 25 features for iTunes China app store and top 14 features for iTunes UK app store.

Table 3 summarizes our results.

After we get our final training models h for each app store, we can use these training model to classify apps in **residual data**. We call the newly found 'abused app' as *suspicious app* as we have no evidence to prove those apps are abused ones. Table 4 summarizes our results. From the table we can see that our coverage in iTunes UK is relatively larger than other app stores with 9.51% apps labelled as abused app (suspicious) while the other two stores have around 5% suspicious apps. These observations are however specific to the datasets we use.

Classification Result Validation. The question may be asked after we label some abused(suspicious) apps in **residual data** is that how to validate those apps or how suspicious those apps could be. Though direct validation is difficult as we have no evidence to show if those apps are abused or not, here, we present two additional features to evaluate our results.

Consecutive Reviewer IDs: In Section 2.3, we point out that some abused apps may contain consecutive IDs. Here we define **consecutive user IDs** as if two IDs rate one app in the same day and the difference between IDs is less than 1000.

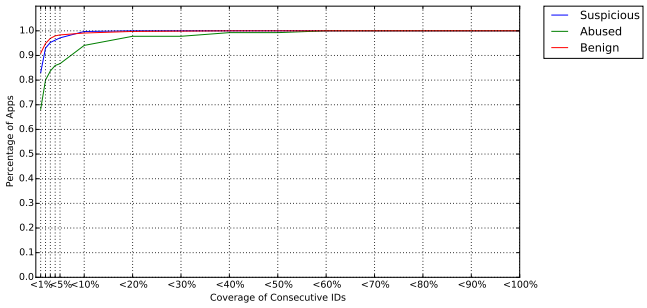

Figure 11. Consecutive User IDs in iTunes US App Store

Figure 11, 12 and 13 depict the situations of existence of consecutive IDs in iTunes US, iTunes China and iTunes UK app stores, respectively. In each figure, x-axis represents the coverage of consecutive IDs, which is defined as percentage of consecutive user IDs over number of users who leave comments for this app. For instance, '<10%' means that coverage of consecutive IDs is less than 10%. Y-axis represents the percentage

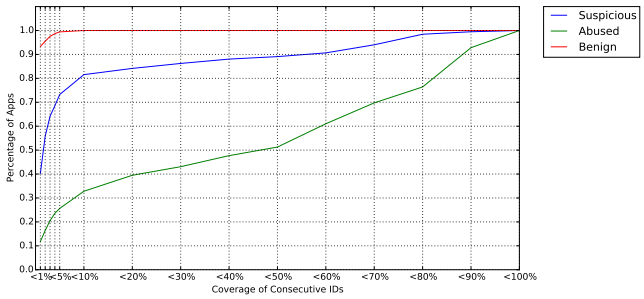

Figure 12. Consecutive User IDs in iTunes China App Store

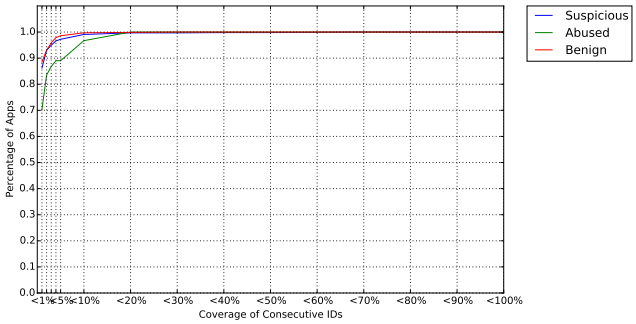

Figure 13. Consecutive User IDs in iTunes UK App Store

of those apps over the number of apps in **residual data**. In the legend, 'suspicious' refers to the abused apps we classified by our training model; 'abused' refers to the abused app in our training set; 'benign' means the normal apps in our training data.

We can easily see that some apps contain more consecutive user IDs than normal apps do. This is because when we are picking abused apps for our training set, the algorithm we use will explicitly find collusion groups that have attacked more than one apps. It is conceivable that attackers will use same fake user IDs to launch another attack, which is much easier and more feasible than finding the same group of real users to attack another app. A group of fake users often have consecutive IDs, so those abused apps attacked by fake users can be more easily detected by our algorithm and put them into our training set and those abused apps attacked by real users remain undetected and discovered by our training models later.

Put these three figures together, we discover that, in iTunes US and UK, most abused apps are attacked by **indirect formation** groups, while in iTunes China, we are safe to draw conclusion that more than 50% of abused apps are attacked by **direct formation** groups (defined in Section 2.3).

Review Density: Another feature we use to evaluate our classification results is **review density**, which is the percentage of review area. Figure 14 is the number of

Table 3. Training Model Evaluation Table

App Store	Precision	Recall	F-score	Top n Features
iTunes US	91.43%	72.38%	80.52%	18
iTunes China	88.82%	82.94%	85.58%	25
iTunes UK	85.37%	80.28%	82.27%	14

Table 4. Classified Suspicious Apps In Each App Store

App Store	# Suspicious Apps	# Residual Apps	Coverage(%)
iTunes US	1103	22703	4.86%
iTunes China	1123	21056	5.33%
iTunes UK	930	9777	9.51%

reviews vs. week, where blue area is defined as review area and total area is represented by the square.

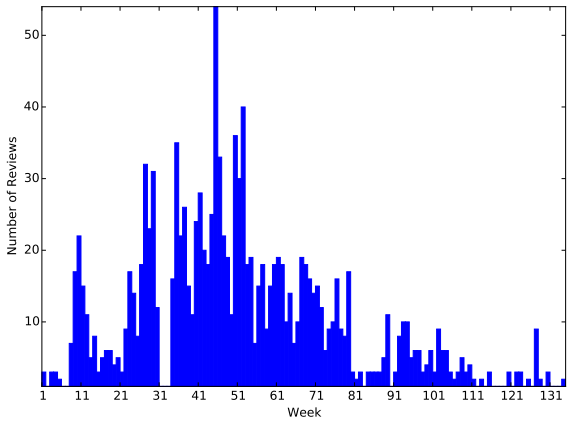

Figure 14. Review Density Example

Usually, abused apps should have low review density. However, we notice that most apps, no matter they are benign or abused, will gain lots of reviews in the first several weeks. That is because usually app stores will put new apps in a specific section in the front page so that users may be able to notice them. To mitigate this issue, we remove the first 5 weeks data for each app.

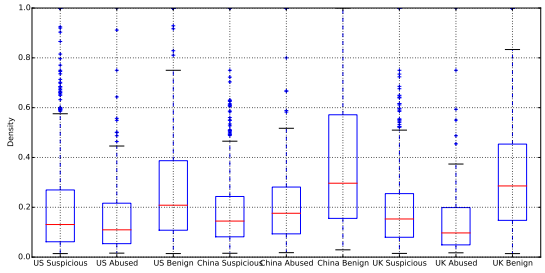

Figure 15. Review Density for iTunes US, China, UK App Store

Figure 15 depicts box plots of review density for iTunes US, China and UK app stores. We can see that, compared with benign apps, abused apps usually have lower review density. The suspicious apps that we classified have similar review density distribution to that of abused apps. Review densities of benign apps in China and UK are higher than those in US, which means US benign apps' review distributions are less spiky, in general.

There are some outliers in both suspicious and abused apps, whose review densities are relatively large. Nevertheless, it is superficial to say these outliers are false positive. As we cut out the first 5-week data for each app, there might be attacking behavior hidden in the first 5 weeks, which will greatly reduce the review density if we put those data back. Another possible case is, if abused apps are successfully promoted, users will easily find these apps in app store, which leads to many reviews.

We also randomly picked some suspicious apps we found and manually investigated their rating distributions. Even theoretically, it is very difficult for human being to process those large amount of data and find useful information from it, we did notice some skeptical data and figure 16 depicts one of them.

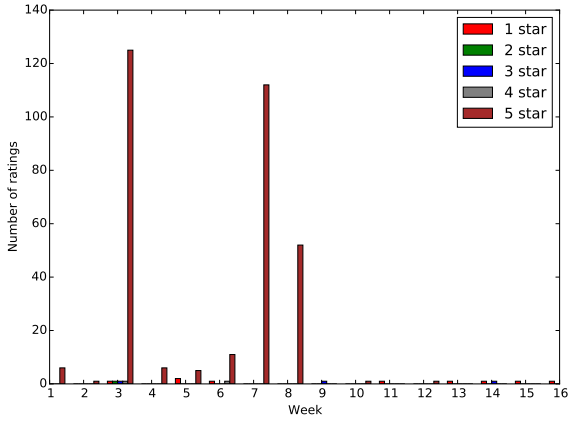

Figure 16. Suspicious App (app ID: 593313544)

6. Data Analysis

We have done some data analysis during our training and result evaluation, but it is worth analyzing the data we have from different perspectives.

Categories of Abused App. Figure 17 depicts the percentage of apps of each category in three app stores. Note that abused apps here include abused apps in our training set and suspicious apps classified by our training models.

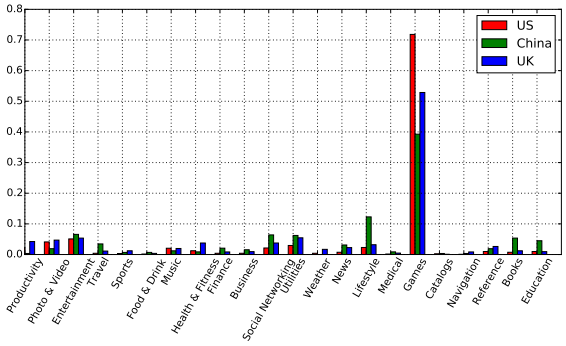

Figure 17. iTunes Abused Apps Categories

Though we use different training sets to train our models, in different countries, game apps are more likely to be abused than any other categories. In iTunes US and UK app store, game apps take up more than 70% and 50%, respectively. Either because there are more games than other apps or games in iTunes app store always draw more attention than any other categories simply because games have more varieties and many users would like to explore new games and spend money on them. In this case, promoting a game app could attract more users' attention than promoting a sport app. In China, more (in terms of percentage) lifestyle apps are abused. It might be because many coupon companies and e-commerce companies are competing for market shares.

6.1. Review Content Analysis

If collusion groups focus on manipulating average scores, to reduce the cost, malicious users usually leave simple and short reviews. Abused apps here include both abused apps in the training set and suspicious apps we classified with our training model. Benign apps are the rest of apps. Figure 18 depicts the average length(number of tokens) of review contents for abused apps and benign apps in iTunes US, UK and China app stores. The method used to calculate review length is to tokenize each review content and title without punctuation. For English, number of tokens is number of words, while for Chinese, it is the number of characters. Some comments contain several languages,

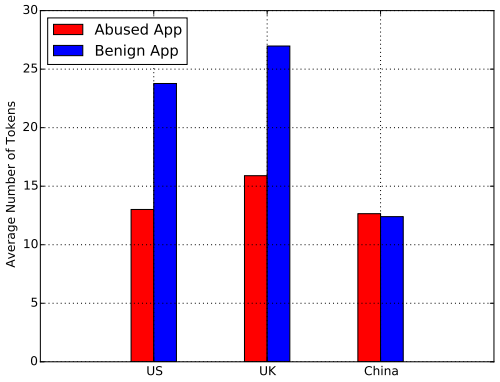

Figure 18. Average Review Content Lenght in iTunes US, UK and China App Store

but with Lucene's[30] help, our tool is still able to count the number of tokens correctly.

We have the following three takeaways from this figure:

1. US and UK users tend to write longer reviews than users in China. Even English and Chinese are quite different, by manually inspecting review contents, we found that US and UK users tend to describe the apps they purchased in great detail and share more personal experience. But China users tend to give a brief judgment and usually do not share user experience.

2. In iTunes US and UK app stores, compared with benign apps, the average length (number of tokens) of comments for abused apps is nearly half of that for benign apps, which means malicious users do tend to leave simple and short comments.

3. In iTunes China app store, we can see that the average lengths are same for both of abused and benign apps. That's because even benign users in China leave short and simple reviews.

We also calculate the average review lengths of each star in these three app stores. By inspecting Figures 19, 20 and 21, we have the following observations:

1. The average review lengths of higher rating levels (4 and 5 stars) are lower than those of lower rating levels(1 and 2 stars). The higher rating level, the shorter its reviews are. This is because the higher rating apps get, the less bugs they have and users usually use more words to complain and report bugs in lower level reviews.

2. For iTunes US and UK app stores, average review lengths for abused apps are always lower than

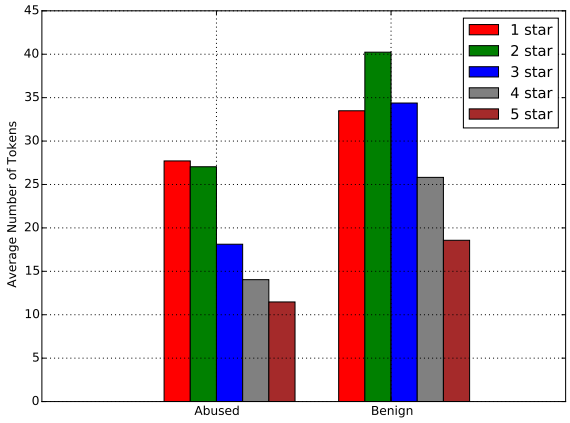

Figure 19. Average Review Content Lengths in iTunes US App Store

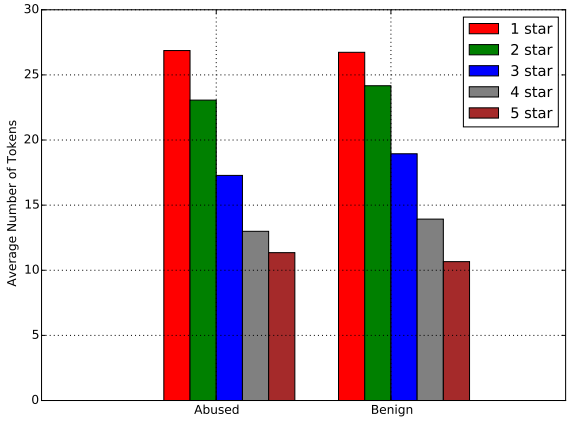

Figure 20. Average Review Content Lengths in iTunes China App Store

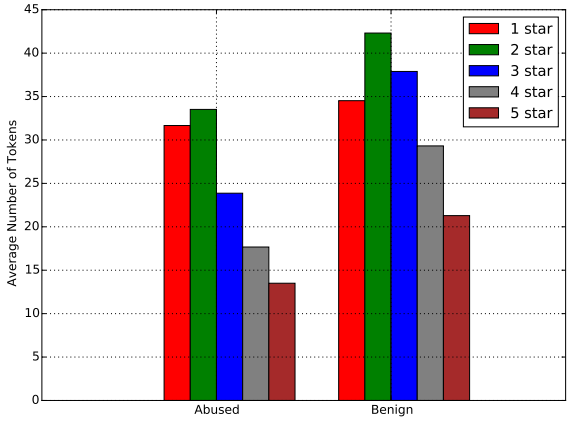

Figure 21. Average Review Content Lengths in iTunes UK App Store

benign apps. Our explanation is that there are some app-promoting websites[2] and apps[3] are made for recruiting users to review apps and pay them. Intuitively, if users are paid to review an app, their reviews' quality usually is low, which means review contents are short, simple, sometimes meaningless. Those websites do not require users to leave high ratings, therefore, we can see that the average lengths of each rating level for abused apps are always lower. However, this observation does not fit iTunes China app store. First of all, we have not found any similar app-promoting websites in China. Secondly, those websites use Paypal to refund users, but Paypal is not popular in China. Therefore, we think users in China may find difficulty to get paid from those websites.

7. Related Work

Fake reviews or review spams in traditional online stores like Amazon, Yelp have been studied a lot. Jindal et al. [31] investigated fake reviews based on three typical ones: untruthful opinions which aim at misleading other users deliberately, reviews on brand only instead of on product quality and non-reviews including advertisement, questions, etc. Later, they further proposed an approach [32] to identify atypical review patterns by way of finding unexpected rules and rule group. Li et al. [33] used machine learning techniques and semi-supervised method, on the basis of manually labeled fake reviews, to detect unlabeled fake reviews. Ott et al. [34] emphasized psycholinguistic methods and text analysis. Their approach takes standard word and part of speech as the training data for supervised learning. These researches focus on discovering individual fake reviews in traditional market.

However, in app store, since most reviews are short, it is difficult to differentiate whether a single review is faked or not. Moreover, individual attack to an app will not work considering the large number of reviews toward a single app.

Xie and Zhu [35] studied the underground market of trading mobile app reviews and proposed an abused app tracer, which starts from known abused app set and detects related unknown abused apps by following commonly reviewed apps and shared reviewers in an iterative way. In [36], the authors proposed an approach to identifying attackers of collusive promotion groups in an app store. Different from our work, their idea is to exploit the abnormal change of app rankings to identify promoted apps, and measures their pairwise similarity to finally identifies the collusive group members,

[2] https://promodispenser.com/, https://giftmeapps.com/, etc.
[3] https://itunes.apple.com/app/id688637547?mt=8

assuming multiple promoted apps have similar ranking change patterns. Our dataset does not have app daily ranking change information.

In [13], Xie and Zhu proposed a collusion group detection algorithm named GroupTie, which measures the tie strength (similarity) of raters based on their common rating behavior. The more apps two raters have commonly rated and the more similar their ratings are, the more likely they belong to the same collusion group. They build a relation graph named tie graph and detect collusion groups by applying graph clustering. In [37], the authors proposed a system called PADetective to detect miscreants who are likely to be conducting promotional attacks. PADetective adopts supervised learning to characterize promotion attackers with 15 features (e.g., day intervals, semantic similarity), and then applies the trained model to detect other attackers. The differences between our work and PADetective are two-fold. First, PADetective is designed to detect attackers whereas our approach is to detect promoted apps. Second, PADetective uses more user features whereas our approach uses more app features.

8. Conclusion And Future Work

In this paper, we proposed a machine learning based approach to detecting app rating manipulation. We extracted 55 features from our data and use random forest algorithm to rank their importance, so that we are able to figure out which features are critical in abused app detection. It turned out that the feature rankings we get match our definitions and characteristics of abused app and collusion group, which means our features can be used to separate abused apps from benign apps well. To improve training performance in terms of speed and accuracy, we selected top n features for each app store and applied our trained models to the data that we crawled but are not used during the learning procedure, in order to discover more abused apps. Even though we lack evidences to prove that average ratings of the newly discovered abused apps have been manipulated, by using two more features(consecutive user IDs and review density), it is reasonable to believe that our training models have practical merits.

Our future work includes the following.

1. Use unsupervised learning[38] to detect collusion groups. Since we found that app rating manipulation is always done by large number of malicious users in a relatively short period of time, unsupervised learning is able to divide all reviews into different clusters. Therefore, we may find useful information from those clusters.

2. Nature language processing could be introduced to analyze the review contents. We could use sentimental analysis to judge if reviewer's comment matches its rating score, which might be a good way to classify review as malicious one or benign one. Also, by removing stop words, we will be able to analyze important word frequencies and other stuffs. Although some researchers have done similar researches before[34][39][31], analyzing review contents along is less convincing.

Acknowledgement: The work of Zhu and Wang were supported by NSF CNS-1618684.

References

[1] https://www.statista.com/statistics/271644/worldwide-free-and-paid-mobile-app-store-downloads/.

[2] https://9to5mac.com/2018/10/11/app-store-massive-revenue-growth-over-google-play/.

[3] How rating affects mobile app ranking. http://www.insidemobileapps.com/2012/06/29/how-rating-affects-ranking-in-search-results-and-top-charts-across-platforms/.

[4] https://medium.com/@incipiagabe/aso-case-study-google-play-app-ranking-manipulation-652407eb85b2.

[5] Yao, Y., Viswanath, B., Cryan, J., Zheng, H. and Zhao, B.Y. (2017) Automated crowdturfing attacks and defenses in online review systems. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*: 1143–1158.

[6] Xie, Z., Zhu, S., Li, Q. and Wang, W. (2016) You can promote, but you can't hide: large-scale abused app detection in mobile app stores. In *Proceedings of the 32nd Annual Conference on Computer Security Applications, ACSAC 2016, Los Angeles, CA, USA, December 5-9, 2016*: 374–385.

[7] Hoeffding, W. (1963) Probability inequalities for sums of bounded random variables. *Journal of the American statistical association* **58**(301): 13–30.

[8] Allahbakhsh, M., Ignjatovic, A., Benatallah, B., Bertino, E., Foo, N. *et al.* (2013) Collusion detection in online rating systems. In *Web Technologies and Applications* (Springer), 196–207.

[9] Lehmann, S., Schwartz, M. and Hansen, L.K. (2008) Biclique communities. *Physical Review E* **78**(1): 016108.

[10] Abu-Mostafa, Y.S., Magdon-Ismail, M. and Lin, H.T. (2012) *Learning From Data* (AMLbook.com), chap. The Learning Problem, 3–5.

[11] Kohavi, R. (1995) A study of cross-validation and bootstrap for accuracy estimation and model selection (Morgan Kaufmann): 1137–1143.

[12] Abu-Mostafa, Y.S., Magdon-Ismail, M. and Lin, H.T. (2012) *Learning From Data* (AMLbook.com), chap. Overfitting, 123–126.

[13] Xie, Z. and Zhu, S. (2014) Grouptie: toward hidden collusion group discovery in app stores. In *7th ACM Conference on Security & Privacy in Wireless and Mobile Networks, WiSec'14, Oxford, United Kingdom, July 23-25, 2014*: 153–164.

[14] Liu, Y., Yang, Y. and Sun, Y.L. (2008) Detection of collusion behaviors in online reputation systems. In *Signals, Systems and Computers, 2008 42nd Asilomar Conference on* (IEEE): 1368–1372.

[15] Bansal, T., Chen, B. and Sinha, P. Fastprobe: Malicious user detection in cognitive radio networks through active transmissions .

[16] Breiman, L. (2001) Random forests. *Machine learning* **45**(1): 5–32.

[17] Dietterichl, T.G. (2002) Ensemble learning. *The handbook of brain theory and neural networks* : 405–408.

[18] Safavian, S.R. and Landgrebe, D. (1991) A survey of decision tree classifier methodology. *Systems, Man and Cybernetics, IEEE Transactions on* **21**(3): 660–674.

[19] Breiman, L. (1993) *Classification and regression trees* (CRC press).

[20] Oshiro, T.M., Perez, P.S. and Baranauskas, J.A. (2012) How many trees in a random forest? In *Machine Learning and Data Mining in Pattern Recognition* (Springer), 154–168.

[21] Cortes, C. and Vapnik, V. (1995) Support vector machine. *Machine learning* **20**(3): 273–297.

[22] Hagan, M.T., Demuth, H.B., Beale, M.H. *et al.* (1996) *Neural network design* (Pws Pub. Boston).

[23] Bengio, Y., Courville, A. and Vincent, P. (2014) Representation learning: A review and new perspectives .

[24] Dietterich, T.G. (2000) An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine learning* **40**(2): 139–157.

[25] Guyon, I. and Elisseeff, A. (2003) An introduction to variable and feature selection. *The Journal of Machine Learning Research* **3**: 1157–1182.

[26] Vapnik, V., Levin, E. and Le Cun, Y. (1994) Measuring the vc-dimension of a learning machine. *Neural Computation* **6**(5): 851–876.

[27] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M. *et al.* (2011) Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research* **12**: 2825–2830.

[28] Strobl, C., Boulesteix, A.L., Zeileis, A. and Hothorn, T. (2007) Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC bioinformatics* **8**(1): 25.

[29] Buckland, M.K. and Gey, F.C. (1994) The relationship between recall and precision. *JASIS* **45**(1): 12–19.

[30] Hatcher, E., Gospodnetic, O. and McCandless, M. (2004), Lucene in action.

[31] Jindal, N. and Liu, B. (2008) Opinion spam and analysis. In *Proceedings of the 2008 International Conference on Web Search and Data Mining* (ACM): 219–230.

[32] Jindal, N., Liu, B. and Lim, E. (2010) Finding unusual review patterns using unexpected rules. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM 2010, Toronto, Ontario, Canada, October 26-30, 2010*: 1549–1552.

[33] Li, F., Huang, M., Yang, Y. and Zhu, X. (2011) Learning to identify review spam. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*: 2488–2493.

[34] Ott, M., Choi, Y., Cardie, C. and Hancock, J.T. (2011) Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1* (Association for Computational Linguistics): 309–319.

[35] Xie, Z. and Zhu, S. (2015) Appwatcher: unveiling the underground market of trading mobile app reviews. In *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks, New York, NY, USA, June 22-26, 2015*: 10:1–10:11.

[36] Chen, H., He, D., Zhu, S. and Yang, J. (2017) Toward detecting collusive ranking manipulation attackers in mobile app markets. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2017, Abu Dhabi, United Arab Emirates, April 2-6, 2017*: 58–70.

[37] Sun, B., Luo, X., Akiyama, M., Watanabe, T. and Mori, T. (2017) Characterizing promotional attacks in mobile app store. In *Applications and Techniques in Information Security - 8th International Conference, ATIS 2017, Auckland, New Zealand, July 6-7, 2017, Proceedings*: 113–127.

[38] Barlow, H.B. (1989) Unsupervised learning. *Neural computation* **1**(3): 295–311.

[39] Fu, B., Lin, J., Li, L., Faloutsos, C., Hong, J. and Sadeh, N. (2013) Why people hate your app: making sense of user feedback in a mobile app store. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (ACM): 1276–1284.