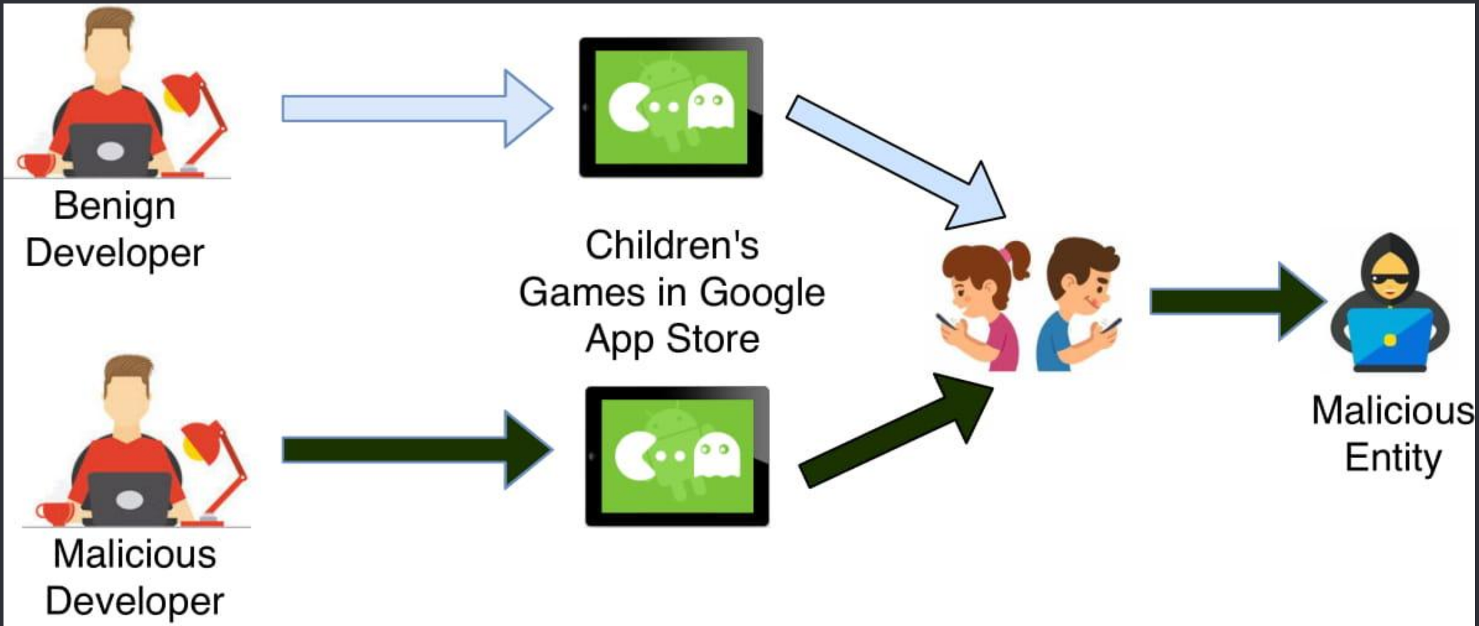
Detecting Privacy Violations in Children's Apps Using HPCs

Suha Hussain
Queens High School for the Sciences
Grade 12

Background Information



Background Information

CHILDREN'S ONLINE PRIVACY PROTECTION ACT

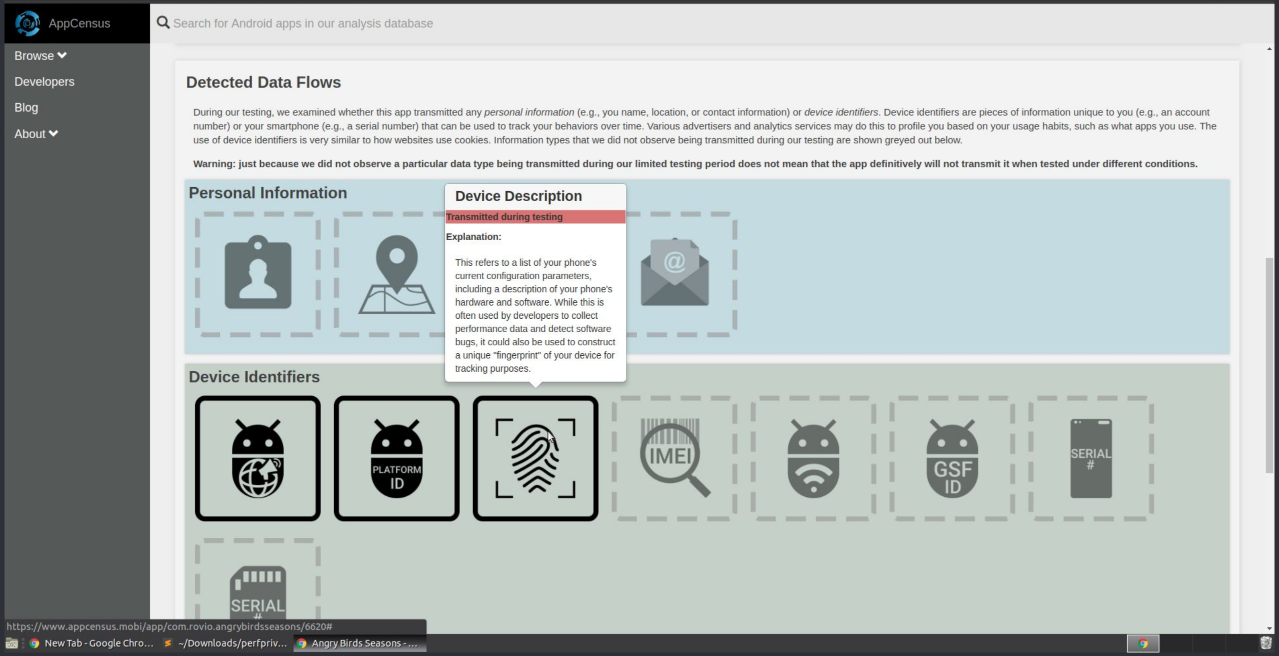
iKeepSafe

COPPA

- Protected Resources
- Location
- Email
- Advertising ID
- Device Description
- Google Services ID
- SIM Serial Number
- Name
- Phone Number
- IMEI
- Android ID
- Serial Number

Background Information

- Reyes et al., 2018



Detected Data Flows

During our testing, we examined whether this app transmitted any *personal information* (e.g., you name, location, or contact information) or *device identifiers*. Device identifiers are pieces of information unique to you (e.g., an account number) or your smartphone (e.g., a serial number) that can be used to track your behaviors over time. Various advertisers and analytics services may do this to profile you based on your usage habits, such as what apps you use. The use of device identifiers is very similar to how websites use cookies. Information types that we did not observe being transmitted during our testing are shown greyed out below.

Warning: just because we did not observe a particular data type being transmitted during our limited testing period does not mean that the app definitively will not transmit it when tested under different conditions.

Personal Information

Device Description

Transmitted during testing

Explanation:

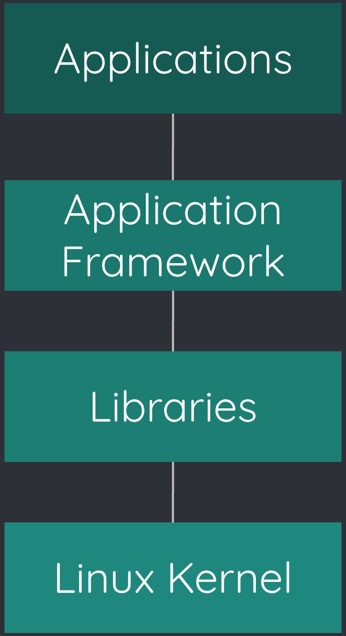
This refers to a list of your phone's current configuration parameters, including a description of your phone's hardware and software. While this is often used by developers to collect performance data and detect software bugs, it could also be used to construct a unique "fingerprint" of your device for tracking purposes.

Device Identifiers

PLATFORM ID
IMEI
GSF ID
SERIAL #
SERIAL #
https://www.appcensus.mobi/app/com.rovio.angrybirdsseasons/6620#
New Tab - Google Chro...
~/Downloads/perfpriv...
Angry Birds Seasons - ...

Background Information

- Weaver et al., 2013
- Chen et al., 2010
- Zhou et al., 2012
- Lu et al., 2014
- Singh et al., 2017
- Gulmegozlu et al., 2017



Applications

Application Framework

Libraries

Linux Kernel

Objective

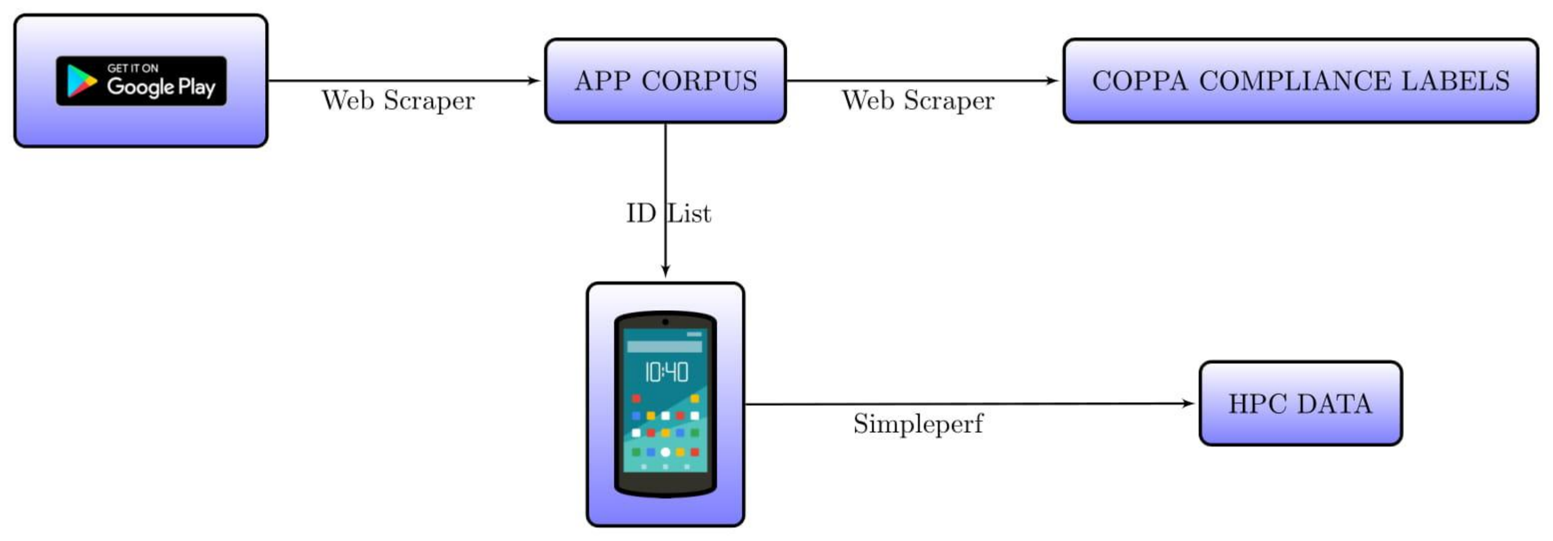
- Develop a machine learning classifier that can detect the COPPA compliance status of an Android app from HPC data

Overall Methodology

1. Collect data.
2. Prepare and establish dataset.
3. Develop and test general violation detectors.
4. Develop and test specialized violation detectors.

Data Collection Methodology

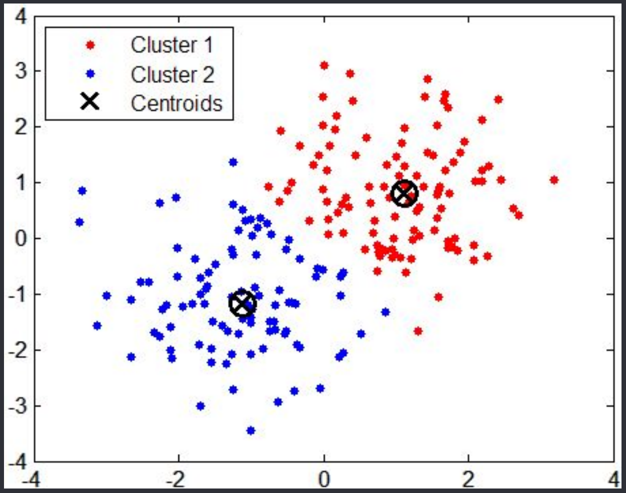
Materials: Python, Google Play Store, Sublime Text, App Census Data, Moto-G Smartphone, Android 8.0, Monkey, Simpleperf

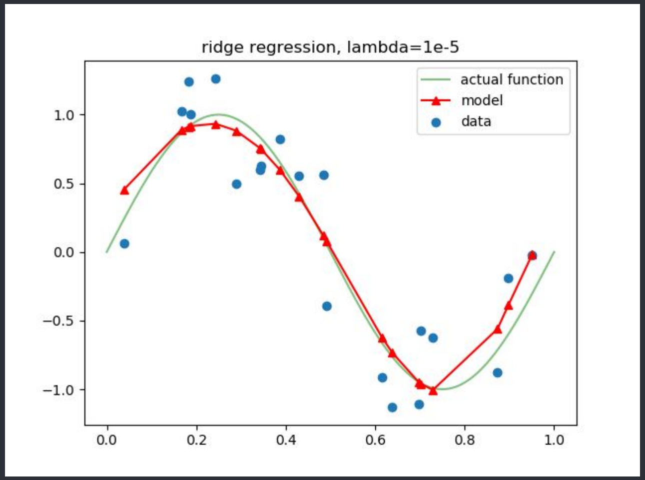


Data Preparation Methodology

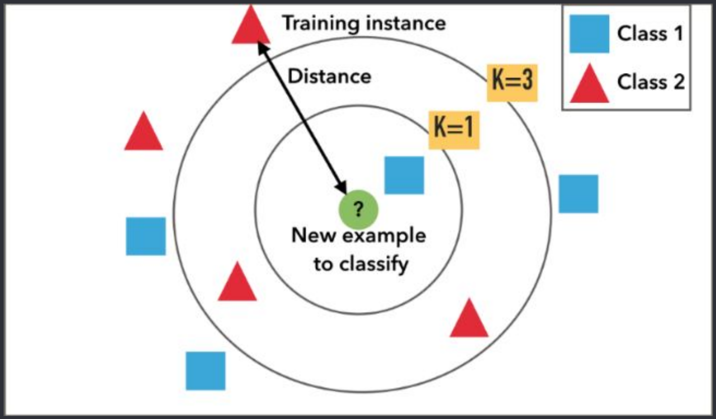
Materials: Dataset, Jupyter Notebooks, Python 3, Scikit-Learn, Slurm Cluster (GPUs)

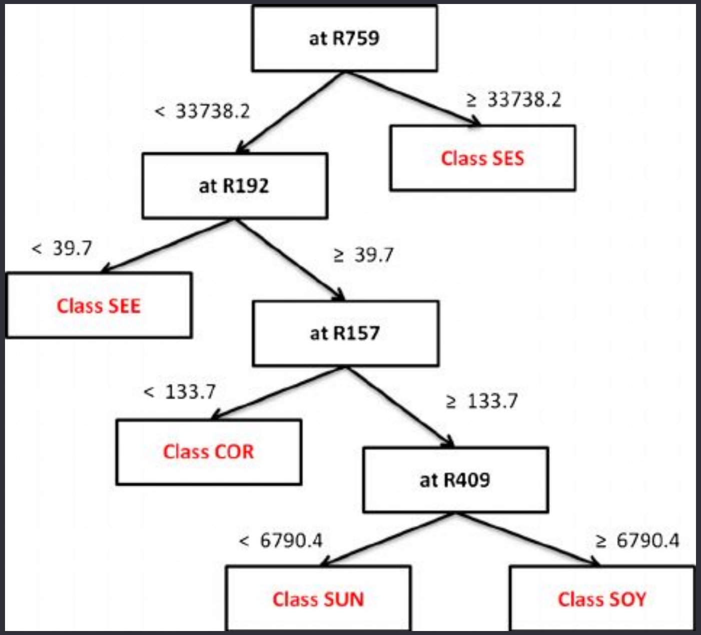
1. Apply k-means clustering to the HPC data.
2. Apply Ridge or LASSO regression to a copy of the dataset as a form of feature reduction.

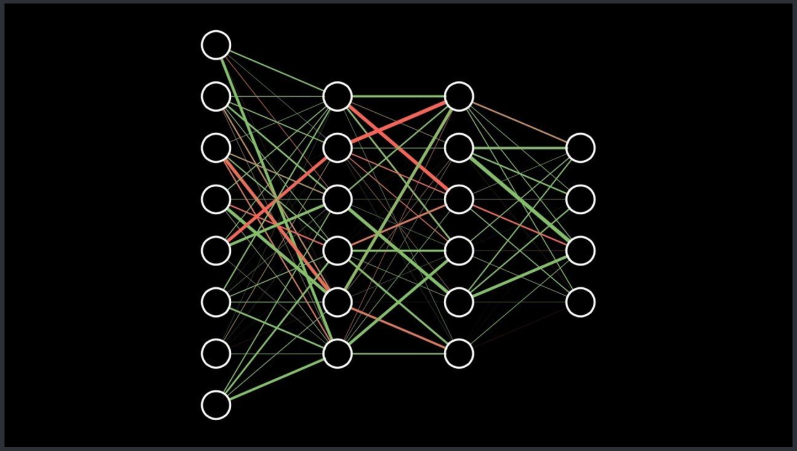




Supervised Learning Techniques

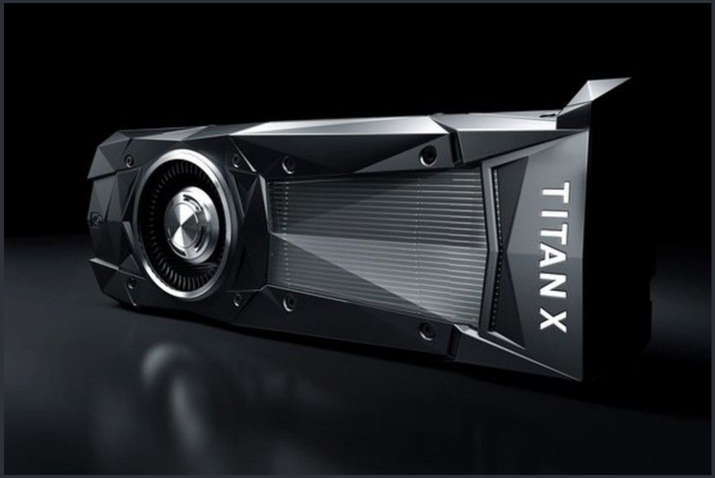






Detection Materials

- Python 3
- Scikit-Learn
- Keras with Tensorflow backend
- Slurm cluster (GPUs)
- Jupyter Notebooks



General Detection Procedure

1. Develop and test k-nearest neighbors (KNN), decision tree (DT), random forest (RF), and a multilayer perceptron neural network (NN) to the dataset that has not undergone feature reduction with general labels.
2. Test the same algorithms on the dataset with general labels that has undergone feature reduction.

General Detection Results

Method	TPR	FPR	Precision	Accuracy
KNN	94.36	0.000035	99.99	99.94
RF	92.11	0.000034	99.99	99.91
DT	86.96	0.000320	99.99	99.84
NN	95.81	0.000370	99.99	99.92

Data without feature reduction

General Detection Results

Method	TPR	FPR	Precision	Accuracy
KNN	95.65	0.000086	99.99	99.94
RF	91.70	0.000051	99.99	99.91
DT	87.40	0.000200	99.99	99.84
NN	92.40	0.000100	99.99	99.91

Data with feature reduction

Specialized Detection Procedure

1. Split the COPPA compliance labels of the dataset by feature.
2. Apply feature reduction.
3. Develop and test k-nearest neighbors (KNN) to each subdataset.

Specialized Detection Results

COPPA Violation	HPC Parameters	Accuracy
Serial	instructions raw-l1-dcache raw-load-retired branch-stores	99.06
Advertising ID	L1-icache-load-miss raw-l1-icache raw-bus-access raw-bus-cycles	99.10
Device Description	L1-icache-load-misse dTLB-store-misses branch-load-misses branch-misses	99.01

Limitations

- The machine learning classifiers were not incorporated within the operating system of an Android phone.
- The classifiers were not optimized for robustness or resource consumption.
- The Monkey tool has a sub-optimal execution path, compromising the veracity of the data.
- A specialized detector was not developed for every COPPA feature.

Suggestions

- Generate HPC data from a tool that performs the function of Monkey that makes use of reinforcement learning.
- Utilize HPCs for GDPR violation detection.
- Incorporate the software as part of a two-phase detection system.
- Incorporate the software within the layers of the Android operating system.

Conclusion

- Utilizing HPCs for COPPA violation detection yields high accuracies and low misclassification rates in addition to being adaptable and efficient.

References

- (2017). 2017 Internet Crime Report. Federal Bureau of Investigation: Internet Crime Complaint Center. pages 4
- Broadhurst, R. (2017). Cybercrime: Thieves, swindlers, bandits and privateers in cyberspace. Social Science Research Network. pages 4
- Chen, D., Vachharajani, N., Hundt, R., Li, X., Eranian, S., Chen, W., and Zheng, W. (2013). Taming hardware event samples for precise and versatile feedback directed optimizations. IEEE Transactions on Computers, 62(2):376–389. pages 5
- Chen, D., Vachharajani, N., Hundt, R., Liao, S.-w., Ramasamy, V., Yuan, P., Chen, W., and Zheng, W. (2010). Taming hardware event samples for fdo compilation. In Proceedings of the 8th annual IEEE/ACM international symposium on Code generation and optimization, pages 42–52. ACM. pages 5
- Commission, F. T. https://www.ftc.gov/enforcement/rules/rulemaking-regulatory-reform-proceedings/childrens-online-privacy-protection-rule.
- Das, S., Werner, J., Antonakakis, M., Polychronakis, M., and Monrose, F. Sok: The challenges, pitfalls, and perils of using hardware performance counters for security. In SoK: The Challenges, Pitfalls, and Perils of Using Hardware Performance Counters for Security. pages 6.
- Demme, J., Maycock, M., Schmitz, J., Tang, A., Waksman, A., Sethumadhavan, S., and Stolfo, S. (2013). On the feasibility of online malware detection with performance counters. In ACM SIGARCH Computer Architecture News, volume 41, pages 559–570. ACM. pages 5
- Egelman, S. (2018). ”our children’s apps aren’t directed at children.”. pages 4

References

- Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep Learning. MIT Press. http://www.deeplearningbook.org. pages 11
- Gulmezoglu, B., Zankl, A., Eisenbarth, T., and Sunar, B. (2017). Perfweb: How to violate web privacy with hardware performance events. In European Symposium on Research in Computer Security, pages 80–97. Springer.
- Gupta, U., Babu, M., Ayoub, R., Kishinevsky, M., Paterna, F., and Ogras, U. Y. (2018). Staff: online learning with stabilized adaptive forgetting factor and feature selection algorithm. In Proceedings of the 55th Annual Design Automation Conference,page 177. ACM. pages 5
- Hans, C. (2009). Bayesian lasso regression. Biometrika, 96(4):835–845. pages 9
- Lu, K., Zhou, X., Bergan, T., and Wang, X. (2014). Efficient deterministic multithreading without global barriers. In ACM SIGPLAN Notices, volume 49.
- Lu, K., Zhou, X., Wang, X.-P., Bergan, T., and Chen, C. (2015). An efficient and flexible deterministic framework for multithreaded programs. Journal of Computer Science and Technology, 30(1):42–56. pages 5
- Mushtaq, H., Al-Ars, Z., and Bertels, K. (2012). Detlock: portable and efficient deterministic execution for shared memory multicore systems. In High Performance Computing, Networking, Storage and Analysis (SCC).
- Ragan, S. (2016). uknowkids.com responds to data breach, says proprietary ip also exposed. pages 4
- Reyes, I. (2018). Won’t somebody think of the children. page 6383. pages 4, 6, 7
- Reyes, I., Wijesekera, P., Reardon, J., Elazari Bar On, A., Razaghpanah, A.,Vallina-Rodriguez, N., and Egelman, S. (2018). Examining coppa compliance at scale.

References

- Singh, B., Evtyushkin, D., Elwell, J., Riley, R., and Cervesato, I. (2017). On the detection of kernel-level rootkits using hardware performance counters. In Proceedings of the ACM Asia Conference on Computer and Communications.
- Wang, X. and Karri, R. (2013). Numchecker: Detecting kernel control-flow modifying rootkits by using hardware performance counters. In Design Automation Conference (DAC), 2013 50th ACM/EDAC/IEEE, pages 1–7. IEEE. pages 5
- Weaver, V. M. (2013). Linux perf event features and overhead. In The 2nd International Workshop on Performance Analysis of Workload Optimized Systems, FastPath, volume 13. pages 5
- Zhou, X., Lu, K., Wang, X., and Li, X. (2012). Exploiting parallelism in deterministic shared memory multiprocessing. Journal of Parallel and Distributed Computing, 72(5):716–727. pages 5

Acknowledgements

I greatly appreciate the mentorship and assistance of:

- Professor Ramesh Karri at NYU Tandon
- Professor Kanad Basu at NYU Tandon
- Dr. Ujjwal Gupta of Intel
- Mr. Jose Mondestin at QHSS@YC

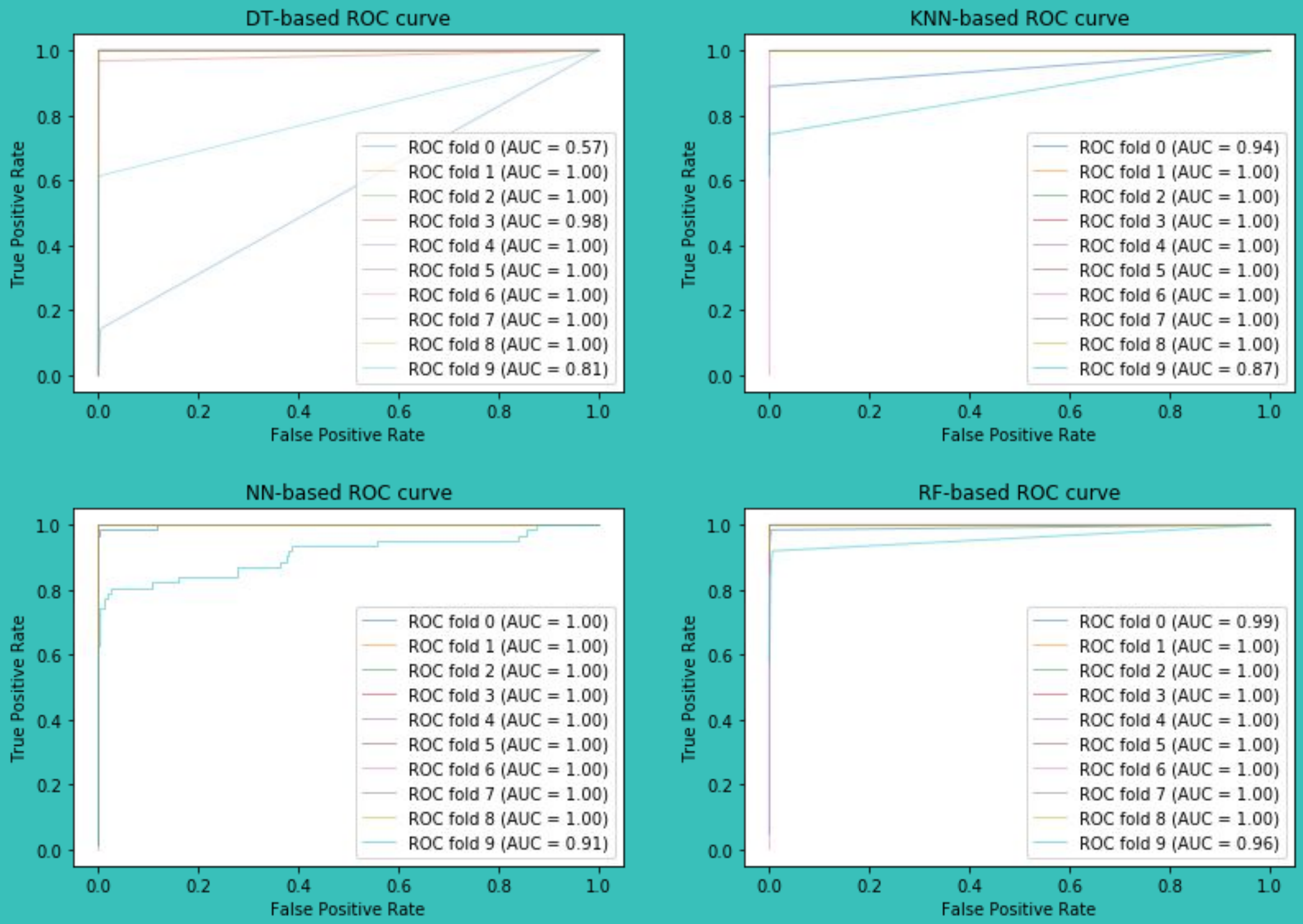
Thank you for your time.

ARE THERE ANY QUESTIONS?

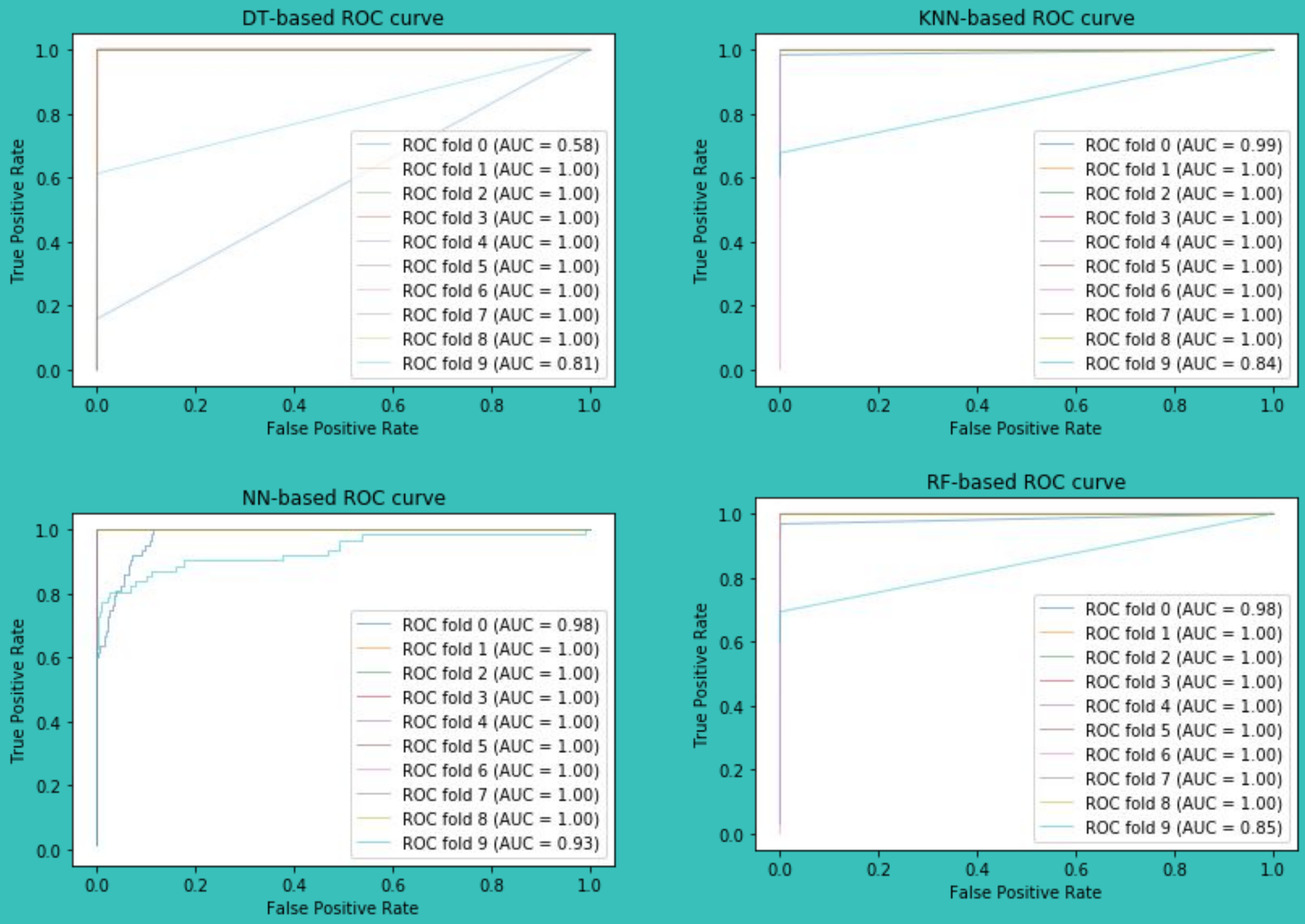
More information can be found in the Appendix of this presentation or at sshussain.me.

APPENDIX

General Detection Results (Without FR)



General Detection Results (With FR)


DT-based ROC curve
True Positive Rate
False Positive Rate
ROC fold 0 (AUC = 0.58)
ROC fold 1 (AUC = 1.00)
ROC fold 2 (AUC = 1.00)
ROC fold 3 (AUC = 1.00)
ROC fold 4 (AUC = 1.00)
ROC fold 5 (AUC = 1.00)
ROC fold 6 (AUC = 1.00)
ROC fold 7 (AUC = 1.00)
ROC fold 8 (AUC = 1.00)
ROC fold 9 (AUC = 0.81)
KNN-based ROC curve
True Positive Rate
False Positive Rate
ROC fold 0 (AUC = 0.99)
ROC fold 1 (AUC = 1.00)
ROC fold 2 (AUC = 1.00)
ROC fold 3 (AUC = 1.00)
ROC fold 4 (AUC = 1.00)
ROC fold 5 (AUC = 1.00)
ROC fold 6 (AUC = 1.00)
ROC fold 7 (AUC = 1.00)
ROC fold 8 (AUC = 1.00)
ROC fold 9 (AUC = 0.84)
NN-based ROC curve
True Positive Rate
False Positive Rate
ROC fold 0 (AUC = 0.98)
ROC fold 1 (AUC = 1.00)
ROC fold 2 (AUC = 1.00)
ROC fold 3 (AUC = 1.00)
ROC fold 4 (AUC = 1.00)
ROC fold 5 (AUC = 1.00)
ROC fold 6 (AUC = 1.00)
ROC fold 7 (AUC = 1.00)
ROC fold 8 (AUC = 1.00)
ROC fold 9 (AUC = 0.93)
RF-based ROC curve
True Positive Rate
False Positive Rate
ROC fold 0 (AUC = 0.98)
ROC fold 1 (AUC = 1.00)
ROC fold 2 (AUC = 1.00)
ROC fold 3 (AUC = 1.00)
ROC fold 4 (AUC = 1.00)
ROC fold 5 (AUC = 1.00)
ROC fold 6 (AUC = 1.00)
ROC fold 7 (AUC = 1.00)
ROC fold 8 (AUC = 1.00)
ROC fold 9 (AUC = 0.85)