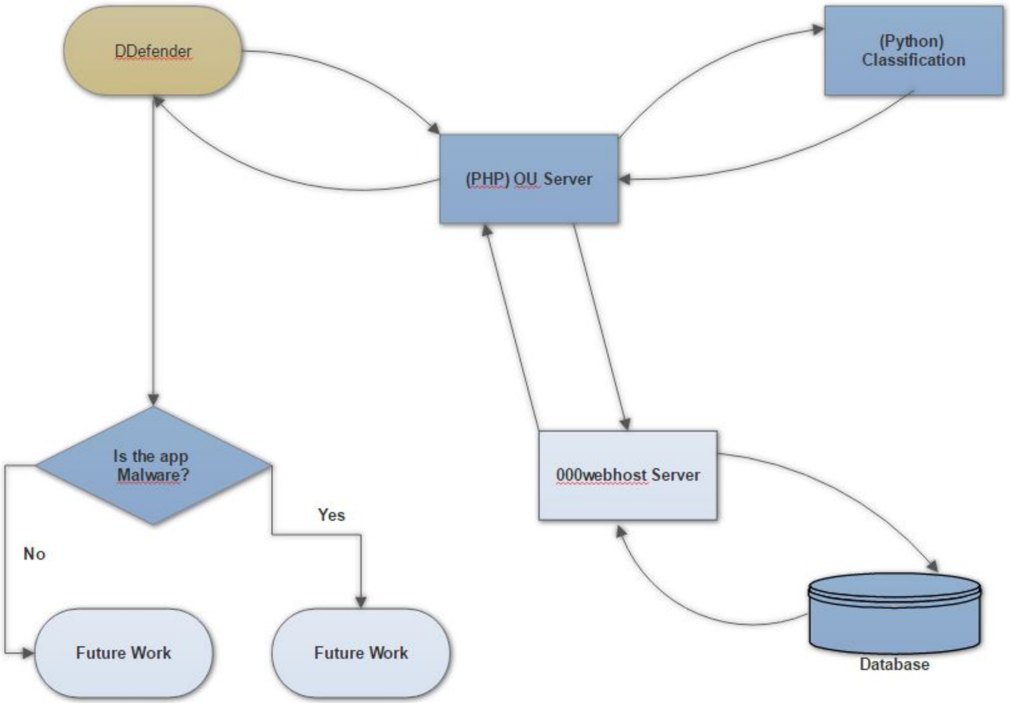
Detecting Malware in Android

Professor Fu
Hani Alshahrani
Harrison Mansour
Seaver Thorn

Outline

- DDefender (Our App changes)
- Unsupervised Learning
- Confusion matrices
- Grid Search
- Deep Learning
- Features related to accuracy
- This week
- Questions

DDefender

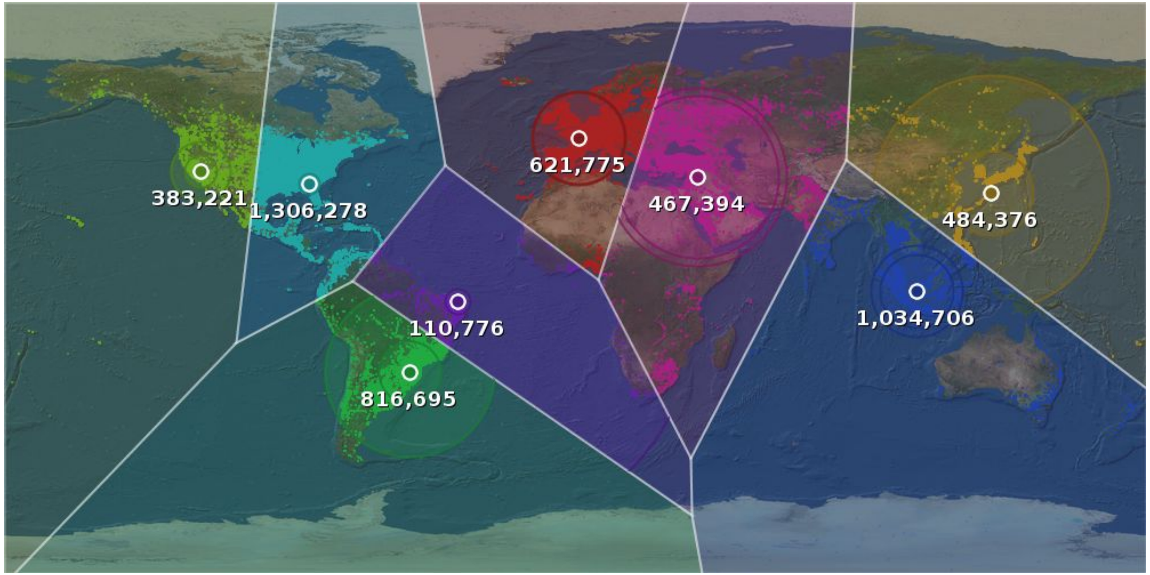


DDefender
(Python) Classification
(PHP) OU Server
Is the app Malware?
No
Yes
000webhost Server
Future Work
Future Work
Database

Unsupervised Learning Algorithms

KMeans (clustering):

- Classify n samples into k groups. Attempts to find similar samples and groups them together.



How does a confusion matrix work?

- By definition a confusion matrix C is such that C_{ij} is equal to the number of observations known to be in group i but predicted to be in group j.

```
p={set of predicted values}
a={set of actual values}

p[i],a[i] element of {True,False}

for i in range (p):
    #this is a true positive
    if p[i] == True and a[i] == True:
        C[True][True] += 1
    #this is a false positive
    elif p[i] == True and a[i] == False:
        C[True][False] += 1
    #this is a false negative
    elif p[i] == False and a[i] == True:
        C[False][True] += 1
    #this is a true negative
    elif p[i] == False and a[i] == False:
        C[False][False] += 1
```

Intersections

- Definition: $A \cap B = \{x | x \in A \wedge x \in B\}$
- Prediction ∩ True Value = # Of True Positives.
- Prediction ∩ False Values = # Of True Negatives.
- ¬Prediction ∩ True Values = # Of False Positives
- ¬Prediction ∩ False Values = # Of False Negatives

A Case Study

Imagine a study evaluating a new test that screens people for a disease.

- True positive: Sick people correctly identified as sick.
- False positive: Healthy people incorrectly identified as sick.
- True negative: Healthy people correctly identified as healthy.
- False negative: Sick people incorrectly identified as healthy.

Confusion Matrices

K-NN:

3266	6
58	977

Gaussian Naive Bayes:

3094	178
331	704

		Condition (as determined by "Gold standard")		
		Condition positive	Condition negative	
Test outcome	Test outcome positive	**True positive**	**False positive** (Type I error)	Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Test outcome positive}}$
	Test outcome negative	**False negative** (Type II error)	**True negative**	Negative predictive value = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Test outcome negative}}$
		Sensitivity = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	Specificity = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Accuracy

Grid Search

- Solves the problem of model selection by finding the optimal hyper-parameters of the model.
- Exhaustive search through a set of defined parameters guided by cross-validation accuracy to find the optimal parameters for the model.
- Example: SVM Requires at least two parameters that need to be tuned to have optimal results. C and Gamma:
  - In the grid search we set:
    - C = {0.1, 1, 10, 100, 1000} (C's are best chosen by increasing by x10 ("RBF SVM parameters", 2016)).
    - Gamma = $\{10^{-3}, \ldots, 10^{3}\}$ (These ranges usually contain the most optimal values ("RBF SVM parameters", 2016)).

Deep Learning Parameters

- We started by using the parameters defined in ANASTASIA, and then went on to optimize for our particular test case.
- Initial parameters(Fereidooni,2016):
  - optimizer=sgd(lr=0.1,decay=1e-6,momentum=0.9)
  - Hidden layers=6
    - layer_size=[3200,1600,800,400,200,100]
    - epochs=600, batch_size=1
- Final parameters:
  - optimizer=adadelta with default parameters
  - Hidden layers=3
    - layer_size=[128,Dropout(0.05),1]
    - epochs=500,batch_size=500

Deep Learning Results

- Accuracy: 95.01%
- False Positive Rate: 67.4% of all errors were false positives
- False Negative Rate: 32.6% of all errors were false negatives

How some features are related to accuracy

- Tested model with the optimal number of parameters, then removed the most useful ones. 96.2% with the 36 most optimal features.
- Numberofpermissions - 95.8% without including numbersofpermissions.
- Top 5 features removed - 91.0%
- Top 10 features removed - 88.7%
- Top 50 features removed - 83.1%
- Model that predicts every sample as benign - 75%

This Week

- Add more functionality to our app to obtain more features.
    - Network information - Including IP address and Web URLs
    - Android App Intents
    - Android App Components
    - Suspicious API calls (if possible)
- Get more training data with these new features
- Return to our machine learning algorithms and see if our accuracy can be improved. (Currently, our most accurate model is Random Forest or Neural Networks with about 95% accuracy).

References

- Trevino, A. (2016). “Introduction to K-Means Clustering”. Datascience.com. Retrieved from https://www.datascience.com/blog/introduction-to-k-means-clustering-algorithm-learn-data-science-tutorials.
- Fereidooni, H. et al. ANASTASIA: ANdroid mAlware detection using STAtic analySIs of Applications, 2016 IEEE.

References (cont)

- Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E. (2011). “Scikit-learn: Machine Learning in Python”. Journal of Machine Learning Research 12. 2825-2830.

Questions