Detecting Malware in Android

Professor Fu
Hani Alshahrani
Harrison Mansour
Seaver Thorn

Outline

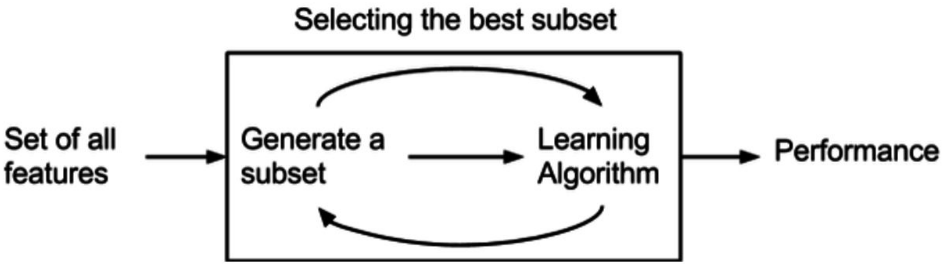
- Key Terms
- Fisher Score
- Chi Square
- Cross Validation
- Python Source Code
- Last Week
- Features Correlation
- Preliminary Results
- Timeline
- This week

Keywords

Feature selection

- An algorithm used to reduce the number of predictor variables in machine learning. Often used to simplify models, shorten training time, and avoid overfitting.

.



Keywords (cont)

		Condition (as determined by "Gold standard")		
		Condition Positive	Condition Negative	
Test Outcome	Test Outcome Positive	**True Positive**	**False Positive** (Type I error)	**Positive predictive value** = Σ True Positive / Σ Test Outcome Positive
	Test Outcome Negative	**False Negative** (Type II error)	**True Negative**	Negative predictive value = Σ True Negative / Σ Test Outcome Negative
		Sensitivity = Σ True Positive / Σ Condition Positive	Specificity = Σ True Negative / Σ Condition Negative	

Accuracy

$$ACC = (TP + TN)/(TP + FP + FN + TN)$$

Data Standardization

- Modify the dataset to achieve a mean of 0 and std deviation of 1 on all variables. This is done to avoid potential problems with some features returning negative numbers. Doing this technique increased accuracy by 25% in SVM.

Fisher Score (Feature Selection Metric)

- For a feature f, the higher the F-score, the more discriminative and important f is for classification accuracy.
- Calculated on feature vectors $\vec{x}_k$, k = 1 … m
- n_+ and n_-. are the number of positive (malware) and negative (benign) samples
- $\bar{x}_i, \bar{x}_i^+, \bar{x}_i^-$ Are the average of the i-th feature for the whole, positive, and negative sets

$$F(i) \equiv \frac{\left(\bar{x}_i^{(+)} - \bar{x}_i\right)^2 + \left(\bar{x}_i^{(-)} - \bar{x}_i\right)^2}{\frac{1}{n_+ - 1}\sum_{k=1}^{n_+}\left(x_{k,i}^{(+)} - \bar{x}_i^{(+)}\right)^2 + \frac{1}{n_- - 1}\sum_{k=1}^{n_-}\left(x_{k,i}^{(-)} - \bar{x}_i^{(-)}\right)^2}$$

Chi-Square (Feature Selection Metric)

- Statistical method that can understand the relation between observed variables and the expected results
- Used in Python to determine the optimal number of features from our dataset.

The value of the test-statistic is

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} = N \sum_{i=1}^{n} \frac{(O_i/N - p_i)^2}{p_i}$$

where

χ^2 = Pearson's cumulative test statistic, which asymptotically approaches a χ^2 distribution.

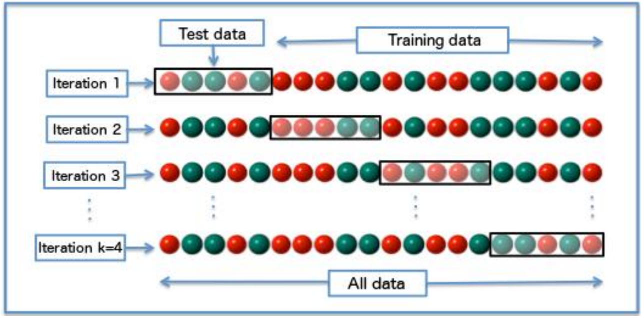
O_i = the number of observations of type *i*.

N = total number of observations

$E_i = Np_i$ = the expected (theoretical) frequency of type *i*, asserted by the null hypothesis that the fraction of type *i* in the population is p_i

n = the number of cells in the table.

Cross-validation



- The data is split into k equal size samples.
- K-1 samples are used for training the machine learning model.
- 1 set is used for testing the machine learning model based on the training data.
- Data is typically split so there is the same proportion of true/false values in each sample.
- In our experiments, we have split the data only into 2 sets, one training and one test set. We have found that a 70% training and 30% testing set has given us good results.

Python source code

chisquare.py

```python
# Keep track of what is the best accuracy with the optimal number of features.
best_acc = 0
optimal_features=0
while test_percent <= 0.5:

    # Split data into training and test set.
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_percent, random_state=40)

    # Use our feature selection method to select the best number of features.
    kbest = SelectKBest(feature_selection_method, "all")
    for i in range (1,size):
        # Loop through all features and use the best i features.
        num_fea = i

        # Train with cross validation
        pipeline = Pipeline([('kbest', kbest), (classifier_name, classifier )])
        grid_search = GridSearchCV(pipeline, param_grid={"kbest__k": range(1, num_fea+1), 'lr__C': np.logspace(-10, 10, 5)}, cv=nfolds)
        grid_search.fit(X_train, y_train)

        # Predict new results.
        y_predict = grid_search.predict(X_test)
        acc = accuracy_score(y_test, y_predict)

        # Keep track of X and Y vals for a graph
        xvals = np.append(xvals, num_fea)
        yvals = np.append(yvals, acc)

        # Print results as we go.
        print "{} accuracy with {} features using {}% training set: {}% \n".format(classifier_name,i,1-test_percent,acc*100)

        # Update our best accuracy if necessary
        if acc>best_acc:
            opt_test_percent = test_percent
            best_acc=acc
            optimal_features=i

    # Increase the test set by 10%
    test_percent = test_percent + 0.1
```

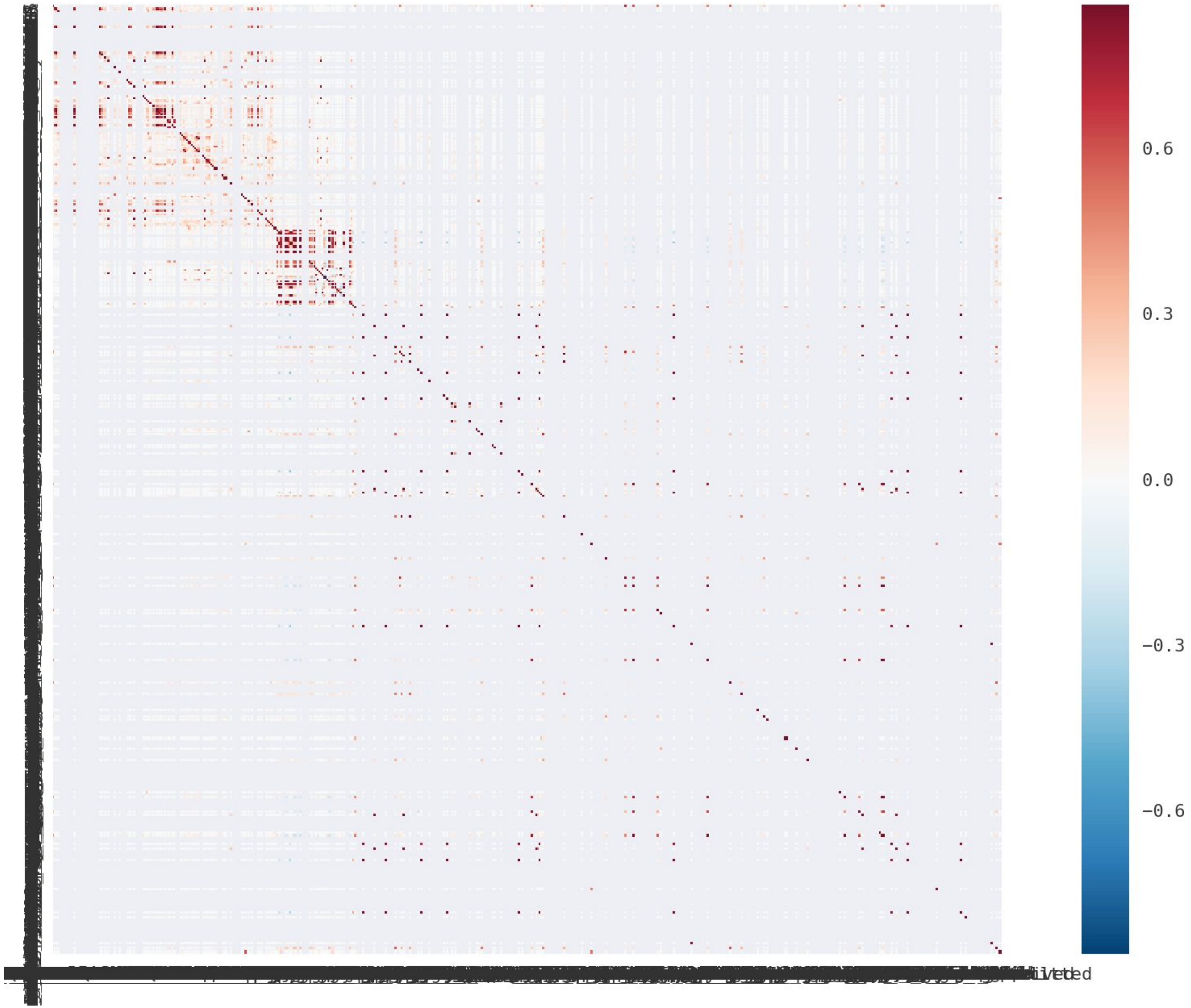
Last week

- Day 1: Use F-Score to compare the best feature selection with RFE. Work on Python scripts.
- Day 2: Attempt to calculate best feature size. Many algorithms give different results. Used chi-square, and F-Score as our metric.
- Day 3: Graph and record results of feature selection. Attempt first machine learning algorithm which achieves 96.5% accuracy. (Random Forest)
- Day 4: Modify python scripts to work with many different feature selection, and machine learning algorithms.
- Day 5: Run this script over the weekend to test for the best feature selection algorithm and machine learning algorithm.

Features correlation

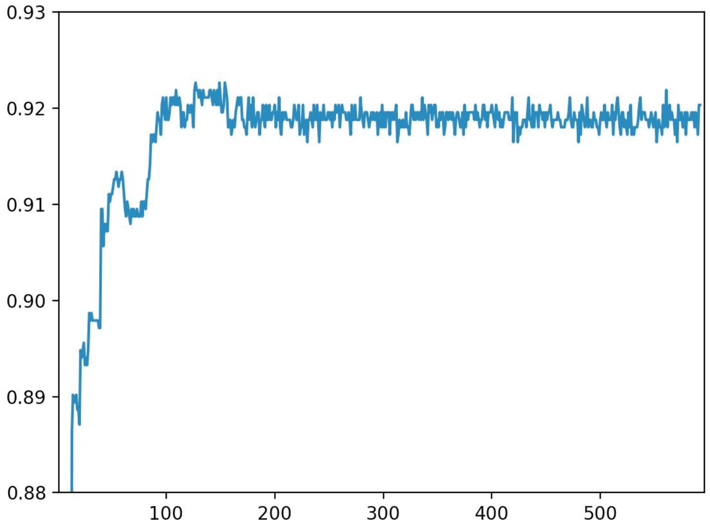
- ACCESS_CHECKIN_PROPERTIES and ACCOUNT_MANAGER have a correlation coefficient of 0.707025
- ACCESS_CHECKIN_PROPERTIES and BIND_INPUT_METHOD have a correlation coefficient of 0.707025 also (coincidence?)
- ACCESS_CHECKIN_PROPERTIES and BROADCAST_PACKAGE_REMOVED have a correlation coefficient of 0.707025

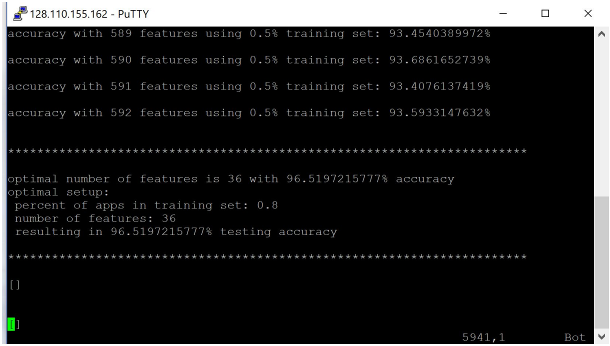
- Correlation Matrix of features
- Largest set of correlated features is in top left
    - This area represents permissions.
- Most features have correlation very close to zero



Preliminary Results

- Running the feature selection algorithm using F-Score and SVM with a linear kernel.
- Automated script is running to determine best machine learning algorithm.
    - Determining best features by F-Score and chi2, running each machine learning algorithm twice using different feature selection algorithms.
    - Testing Logistic Regression, Gaussian Naive Bayes, Random Forests, and SVM with linear and rbf kernel.





Timeline

Week 1

- Project Introduction
- Reading about Android OS and general security practices

Week 2

- Reading about Machine Learning
- Hands on work with Python and Machine Learning

Week 3

- Analyze features of malicious/benign applications on Android

Week 4

- Categorize features of malware applications

Week 5

- Propose a detection technique
- Implement detection technique with Machine Learning.

Week 6

- Compare results of using different Machine Learning algorithms.

Week 7

- Improve detection rates based on the results.

Week 8

- Start writing first draft of final report/publishable results

Week 9

- Revise draft

Week 10

- Finalize paper and submit deliverables

This week

- Analyze results from last week
- Implement the best machine learning algorithm based on these results.
- Fine-tune

Questions?

References

- Weisstein, Eric W. "Chi-Squared Test". *MathWorld*.
- Amirudin, D. (2014, April 14). Gini, ROC, AUC (and Accuracy). Retrieved June 9, 2017, from https://staesthetic.wordpress.com/2014/04/14/gini-roc-auc-and-accuracy/
- KAUSHIK, SAURAV. (2014, December 1). Introduction to Feature Selection methods with an example (or how to select the right variables?). Retrieved June 9, 2017, from https://www.analyticsvidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables/
- Lindorfer, M., Neugschwandtner, M., & Platzer, C. (2015, July). Marvin: Efficient and comprehensive mobile app classification through static and dynamic analysis. In *Computer Software and Applications Conference (COMPSAC), 2015 IEEE 39th Annual* (Vol. 2, pp. 422-433). IEEE.