Chapter Four

Implementation

This chapter explains the methodology that has been implemented in order to compare how models perform differently if applied to the same dataset.

The models that have been tested are based on different algorithms, which are:

1- K Nearest Neighbor
2- Decision Trees
3- Artificial Neural Networks
4- Naïve Bayes

In order prepare a dataset on this 0-1 permission format from the scratch, researchers go through three stages:

- Downloading APK files, both benign and malicious from online repositories. An example of online repositories that provide malware samples in an APK form are MalGenome and Contagio.
- Using software tools to extract the used permissions from the AndroidManifest.xml file of each and every application. An example of tools that can perform this task: APKTool.
- Creating binary vectors of zeros and ones for the used permissions of each application.

The used dataset in this research was downloaded from Kaggle website. It depicts the behavior of Android applications in terms of which permissions they used. The dataset is composed of 558 rows and 331 columns. The rows represent the Android applications instances, while the columns represents the used-permission names. If the permission is used by the Android application, a value of 1 is placed, and if not, a 0 is to indicate the non-usage. The last column of the dataset is called “type”, which is the class column. It gives 0 if the application is benign and 1 if it is malicious.

The experimenting with the above 4 algorithms was conducted twice, the first time with the complete set of 331 features, and the second time with selected set of ranked features. The feature selection was done using the Chi-square algorithm. The aim of performing the experiment two times is to test whether a minimized set of features would yield a better accuracy or not, and to try to find out which number of features is the optimal for the performance of the different models.

The experiments were conducted using R programming language, version 3.4.3 (2017-11-30), on a 64-bit Windows 7 operating system, in a machine with 8 Gigabyte of RAM and Core i7 processor. The following section lists the result of each model along with some consideration. Codes and scripts used to produce the results are in the appendix section.

4.1 First: Trying models with the Complete Set of Features

4.1.1 KNN

In practice, choosing k depends on the difficulty of the concept to be learned, and the number of records in the training data. One common practice is to begin with k equal to the square root of the number of training examples. However, such rules may not always result in the single best k. An alternative approach is to test several k values on a variety of test datasets and choose the one that delivers the best classification performance.

For the k-NN algorithm, the training phase actually involves no model building; the process of training a lazy learner like k-NN simply involves storing the input data in a structured format.

[The k value that was applied here equals the square root of number of training instances, sqrt of 399 = 19.9]

	predict		
test_label	0	1	Row Total
0	77	3	80
	0.963	0.037	0.500
	0.885	0.041	
	0.481	0.019	
1	10	70	80
	0.125	0.875	0.500
	0.115	0.959	
	0.062	0.438	
Column Total	87	73	160
	0.544	0.456	

Table 4.1: KNN's confusion matrix

4.1.2 Decision Trees

Decision Trees are powerful classifiers that present their knowledge in the form of logical structures that can be understood with no statistical knowledge. They utilize a tree structure to model the relationships among features and potential outcomes.

Nevertheless, in spite of their wide applicability, it is worth noting that trees may not be an ideal fit for some scenarios. One such case might be a task where the data has a large number of nominal features with many levels or it has a large number of numeric features. These cases may result in a very large number of decisions and an overly complex tree. They may also contribute to the tendency of decision trees to overfit data. But even this weakness can be overcome by adjusting some simple parameters.

	predicted		
actual	0	1	Row Total
0	74	6	80
	0.463	0.037	
1	6	74	80
	0.037	0.463	
Column Total	80	80	160

Table 4.2: Decision tree confusion matrix

```
> permission_model

Call:
C5.0.default(x = train[-331], y = train$type)

Classification Tree
Number of samples: 398
Number of predictors: 330

Tree size: 6

Non-standard options: attempt to group attributes
```

```
> summary(permission_model)

Call:
C5.0.default(x = train[-331], y = train$type)


C5.0 [Release 2.07 GPL Edition]         Fri Dec 29 06:57:53 2017
-------------------------------

Class specified by attribute `outcome'

Read 398 cases (331 attributes) from undefined.data

Decision tree:

android.permission.READ_PHONE_STATE <= 0: 0 (184/9)
android.permission.READ_PHONE_STATE > 0:
:...android.permission.WRITE_SMS > 0: 1 (105/1)
    android.permission.WRITE_SMS <= 0:
    :...android.permission.INTERNET <= 0: 0 (7/1)
        android.permission.INTERNET > 0:
        :...android.permission.WAKE_LOCK <= 0: 1 (80/6)
            android.permission.WAKE_LOCK > 0:
            :...android.permission.ACCESS_LOCATION_EXTRA_COMMANDS <= 0: 0 (18/7)
                android.permission.ACCESS_LOCATION_EXTRA_COMMANDS > 0: 1 (4)
Evaluation on training data (398 cases):

           Decision Tree
         ----------------
         Size      Errors

            6   24( 6.0%)   <<


          (a)   (b)    <-classified as
         ----  ----
          192     7    (a): class 0
           17   182    (b): class 1


        Attribute usage:

        100.00% android.permission.READ_PHONE_STATE
         53.77% android.permission.WRITE_SMS
         27.39% android.permission.INTERNET
         25.63% android.permission.WAKE_LOCK
          5.53% android.permission.ACCESS_LOCATION_EXTRA_COMMANDS
```

Figure 4.1: printing output of decision tree model

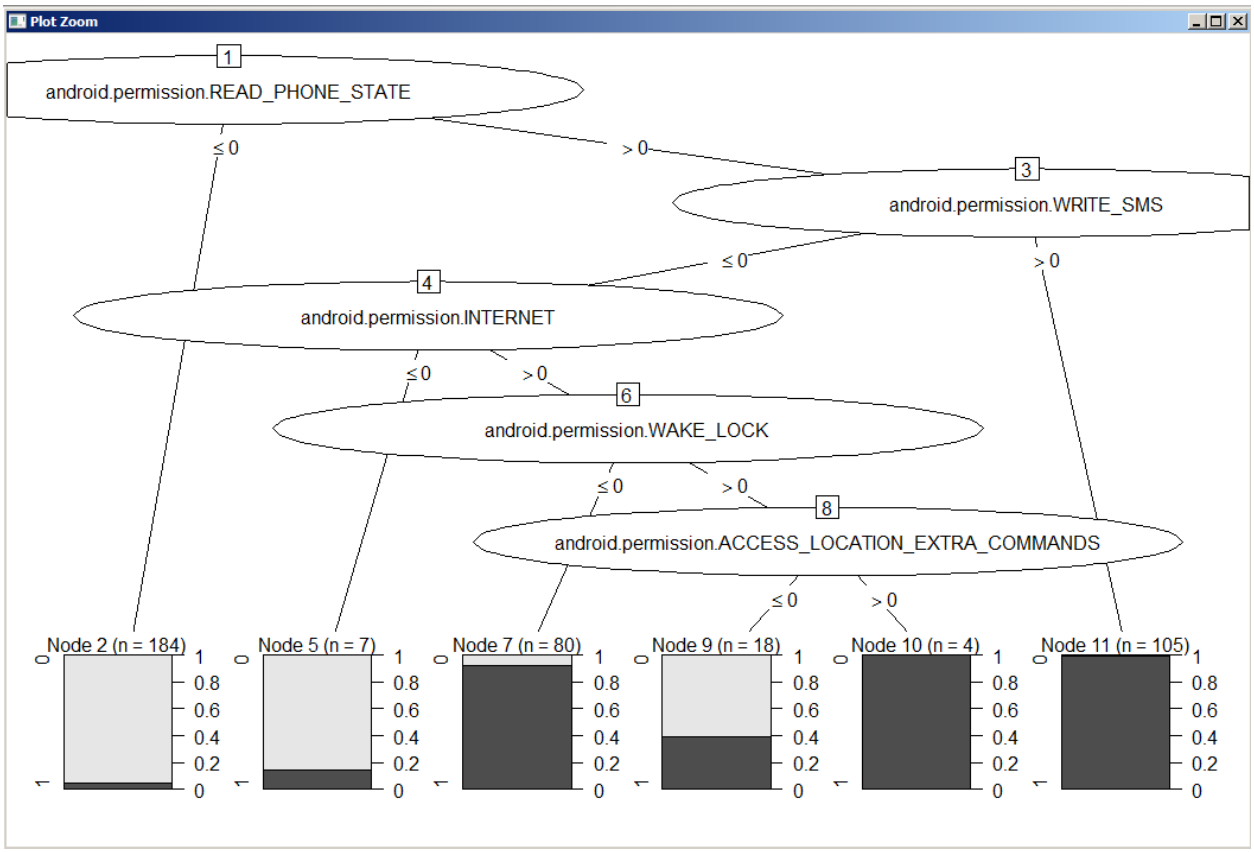

Figure 4.2: Plotting the decision tree model

4.1.3 Artificial Neural Networks

ANNs can be applied to various learning task: classification, numeric prediction, and unsupervised learning as well.

ANNs are best applied to problems where the input data and output data are well-defined or at least fairly simple, yet the process that relates the input to output is extremely complex. As a black box method, they work well for these types of black box problems.

Each neural network can be defined in terms of the following characteristics:

1. An activation function, which transforms a neuron's combined input signals into a single output signal to be broadcasted further in the network. E.g. :

2. A network topology (or architecture), which describes the number of neurons in the model as well as the number of layers and manner in which they are connected
    i. Number of layers.
    ii. Whether information in the network is allowed to travel backward, e.g.: feedforward, recurrent networks (feedback)
    iii. Number of nodes within each layer of the network.
    Unfortunately, there is no reliable rule to determine the number of neurons in the hidden layer. The appropriate number depends on the number of input nodes, the amount of training data, the amount of noisy data, and the complexity of the learning task, among many other factors.
3. The training algorithm that specifies how connection weights are set in order to inhibit or excite neurons in proportion to the input signal, for example: Backpropagation.

The neural network topology that was used for the proposed permission data is a multilayer feedforward, trained using resilient backpropagation, composed of one hidden layer, with Logistic as the activation function, and SSE as the error function.

	predicted		
actual	0	1	Row Total
0	73	7	80
	27.225	27.225	
	0.912	0.087	0.500
	0.912	0.087	
	0.456	0.044	
1	7	73	80
	27.225	27.225	
	0.087	0.912	0.500
	0.087	0.912	
	0.044	0.456	
Column Total	80	80	160
	0.500	0.500	

Table 4.3: Artificial Neural Network confusion matrix

4.1.4 Naïve Bayes

Typically, Bayesian classifiers are best applied to problems in which the information from numerous attributes should be considered simultaneously in order to estimate the overall probability of an outcome.

Naive Bayes assumes that all of the features in the dataset are equally important and independent.

While many machine learning algorithms ignore features that have weak effects, Bayesian methods utilize all the available evidence to subtly change the predictions. If large number of features have relatively minor effects, taken together, their combined impact could be quite large.

Although it is not the only machine learning method that utilizes Bayesian methods, it is the most common one. One drawback of this method is that it is not ideal for datasets with many numeric features. Nevertheless, it does well with noisy and missing data, requires relatively few examples for training, but also works well with very large numbers of examples.

Researchers who adopt the Naïve Bayes emphasize on how probabilities should be revised in the light of additional information.

	test$type		
predict_permission	0	1	Row Total
0	70	15	85
	17.794	17.794	
	0.824	0.176	0.531
	0.875	0.188	
	0.438	0.094	
1	10	65	75
	20.167	20.167	
	0.133	0.867	0.469
	0.125	0.812	
	0.062	0.406	
Column Total	80	80	160
	0.500	0.500	

Table 4.4: Naïve Bayes' confusion matrix

4.2 Second: Trying a minimized version of the dataset using Chi-square

The problem with categorical or nominal data is that numeric methods of measuring correlation among values would not work, which makes the distinguishing between the high-influence and low-influence attributes harder. Thus, analyst should think of alternative ways to test the multicollinearity problem (Independent variables are highly correlated to each other or not).

Highly-correlated columns are not useful and are excluded because we don't gain any new information by including them.

Additionally, in the case of multiple regression–type models, if two or more of the independent variables (or predictors) are correlated, then the estimates of coefficients in a regression model tend to be unstable or counter intuitive.

We need to make sure that predicators are not correlated among themselves, but are correlated with the response variable. In the case of categorical/binomial data, a Chi-square test can be used to test for independence.

Chi square test is a statistical test of independence to determine the dependency of two variables. It shares similarities with coefficient of determination (R^2).

Chi-square test is only applicable to categorical or nominal data, while coefficient is only applicable to numeric data.

We calculate Chi-square statistics between every feature variable and the target variable and observe the existence of a relationship between the variable and the target. If the target variable is independent of the feature variable, we can discard that feature variable. If they are dependent, the feature variable is very important.

To get information on correlation between two categorical variables, a comparison between the expected count of the variable and the observed count is conducted. Chi square measures how much the expected counts and observed counts derivate from each other. We aim to select the features, of which the occurrence is highly dependent on the occurrence of the class. When the two events are independent, the observed count is close to the expected count, thus, a small chi square score.

By calculate the Chi-square scores for all the features, we can rank the features by the Chi-square scores, then choose the top ranked features for model training.

Other available alternatives for ranking and feature-selection of categorical/nominal data: T-test, ANOVA.

4.3 Summary of Results

	The complete dataset	
	accuracy	Error rate
KNN	0.91875	0.08125
Decision Trees	0.925	0.075
ANN	0.9125	0.0875
Naïve Bayes	0.84375	0.15625

Table 4.5: Summary of Results

Algorithm Number of columns	KNN	DT	ANN	NB
10	0.51875	overfit	0.5	0
20	0.51875	0.5	0.5	0.5
40	0.51875	0.50	0.49375	0.5
60	0.51875	0.50	0.49375	0.5
80	0.51875	0.50	0.5	0.5
100	0.51875	overfit	0.49375	0.5

Table 4.6: Results of Applying Chi-square, showing accuracy across number of columns.