Project Report

Problem

There are a huge number of malwares circulating the internet and malwares can come in many forms due to polymorphism using obfuscation and packing among other techniques. Malware classification can aid in malware analysis and can even be applied to new files to aid in detection.

Dataset

For the dataset, each sample has 2 files, the asm file which contacts the information extracted from the IDA disassembler and the byte code file. The dataset is split into 75% training and 25% testing with the ratio of the classes kept consistent across both sets. This is due to the Simda class which only had a total of 42 samples. The features used were all extracted from the asm file of each sample. 2 sets of features were extracted out, the frequency of the opcodes that occurred in the sample and 2 gram opcode. An issue that occurred was finding opcodes in certain files, due to obfuscation and packing, there were a few files that did not have the .text section in the asm file which is where the code of the file should be. Therefore, only samples that had the .text section was used in this dataset, this excluded a total of 605 sample.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Class | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Number of samples excluded |  |  |  |  |  |  |  |  |  |

The total number of features extracted was around 300k. To reduce the number of features, the total number of occurrences for a particular feature across all the samples were calculated. By setting a minimum number of occurrences the number of features can be greatly decreased. The value of 40 was eventually decided because for a particular feature to be relevant, it should occur in each of the sample at least once. Since the smallest number of samples in a particular class was 42, the value 40 made sense and was able to reduce the number of features to 9478.

|  |  |
| --- | --- |
| Total # of Occurrences of Feature | Number of features |
| 0 or greater | 357,006 features |
| Greater than 10 | 15537 features |
| Greater than 20 | 12239 features |
| Greater than 40 | 9478 features |

Approach

A total of 3 different models were used, Random Forest, SVM and Logistic Regression. Before training the model, feature normalization was done on the training set and the mean and variance of the training set was then applied to the test set. For Random Forest, no hyper parameter tuning was done. However, further feature selection was done, the initial method was to use the wrapper method to find the best combination of features, however the processing time was too long. Instead, information gain was calculated for each feature and a threshold was set to further reduce the number of features. The threshold of above 0.6 was set and it reduced the number of features to 267. The reason 0.6 was chosen is because through the use of cross validation, increments of 0.1 was used to calculate the Accuracy and F1 score for each increment and the results showed that the values below 0.6 had the same Accuracy and F1 score as 0.6. Therefore it was ideal to choose 0.6 to reduce the number of features used yet still maintaining the same score.

|  |  |  |
| --- | --- | --- |
| Random Forest | Accuracy | F1 Score |
| Information Gain > 0.1 | 0.954315305 | 0.937249777 |
| Information Gain > 0.2 | 0.954315305 | 0.937249777 |
| Information Gain > 0.3 | 0.954315305 | 0.937249777 |
| Information Gain > 0.4 | 0.954315305 | 0.937249777 |
| Information Gain > 0.5 | 0.954315305 | 0.937249777 |
| Information Gain > 0.6 | 0.954315305 | 0.937249777 |
| Information Gain > 0.7 | 0.954315305 | 0.937245833 |

The model trained using the training data was then applied to the testing data.

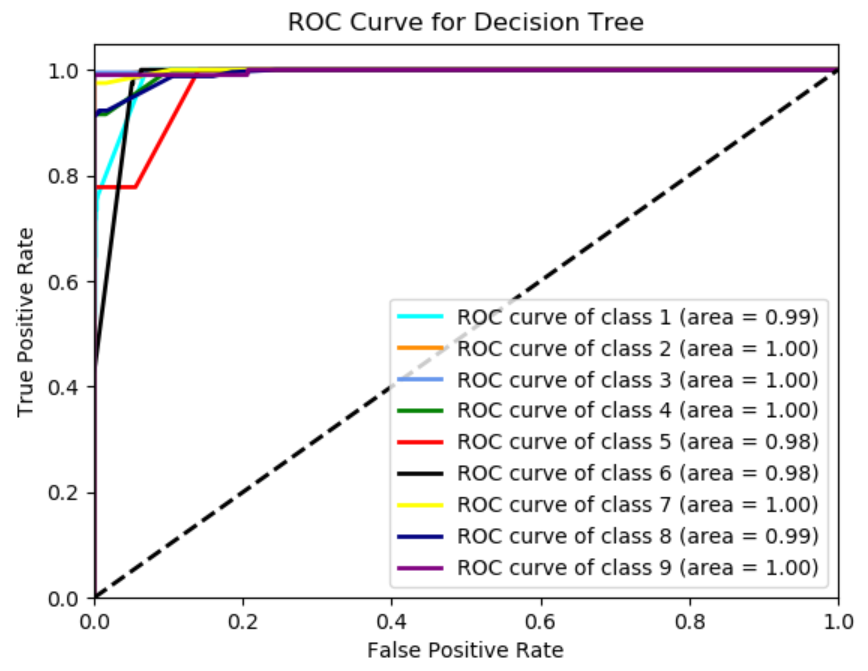
|  |  |  |
| --- | --- | --- |
| Classification | Test Accuracy | Test F1 Score |
| Random Forest | 0.9444 | 0.9231 |

Individual Class Precision and Recall

|  |  |  |
| --- | --- | --- |
| Class | Precision | Recall |
| 1 | 0.96 | 0.76 |
| 2 | 1 | 0.99 |
| 3 | 1 | 0.99 |
| 4 | 0.96 | 0.92 |
| 5 | 1 | 0.78 |
| 6 | 0.59 | 0.97 |
| 7 | 0.99 | 0.97 |
| 8 | 0.99 | 0.91 |
| 9 | 1 | 0.99 |

Confusion Matrix

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | Actual Class | | | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Predicted Class | 1 | 235 | 0 | 0 | 4 | 0 | 70 | 0 | 0 | 0 |
| 2 | 2 | 491 | 0 | 0 | 0 | 2 | 0 | 1 | 0 |
| 3 | 0 | 0 | 586 | 0 | 0 | 3 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 87 | 0 | 8 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 7 | 2 | 0 | 0 | 0 |
| 6 | 2 | 0 | 0 | 0 | 0 | 147 | 0 | 2 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 2 | 78 | 0 | 0 |
| 8 | 5 | 0 | 0 | 0 | 0 | 16 | 0 | 225 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 201 |



For SVM, 2 kernels were used and the hyper parameter of C was tuned using cross validation of k = 5 to determine the best value for the hyper parameter. The 2 kernels were sigmoid and rbf, polynomial was used at the start however, the accuracy was much lower and so it was put aside. The F1 score and accuracy was used in determining the value of the hyper parameter.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | RBF Kernel | | | | Sigmoid Kernel | | | |
| C Value | Train Accuracy | TrainF1 | Test Accuracy | TestF1 | Train Accuracy | TrainF1 | Test Accuracy | TestF1 |
| 0.1 | 0.75164 | 0.531512 | 0.744532 | 0.517119 | 0.755495 | 0.562351 | 0.754089 | 0.560926 |
| 1 | 0.929574 | 0.891355 | 0.908515 | 0.869538 | 0.897555 | 0.855121 | 0.889876 | 0.842913 |
| 2 | 0.941743 | 0.905 | 0.921977 | 0.884229 | 0.908428 | 0.87487 | 0.899541 | 0.860429 |
| 3 | 0.947065 | 0.91097 | 0.92819 | 0.891055 | 0.911334 | 0.880055 | 0.901035 | 0.864496 |
| 4 | 0.948331 | 0.91258 | 0.930951 | 0.894733 | 0.912687 | 0.882827 | 0.901841 | 0.866022 |
| 5 | 0.948935 | 0.913256 | 0.931411 | 0.895132 | 0.915132 | 0.886029 | 0.903801 | 0.870625 |
| 100 | 0.953481 | 0.927102 | 0.933253 | 0.892961 | 0.924225 | 0.898963 | 0.897691 | 0.854812 |
| 150 | 0.953653 | 0.92892 | 0.933023 | 0.892258 | 0.923995 | 0.897694 | 0.893549 | 0.849404 |
| 200 | 0.953768 | 0.930223 | 0.933714 | 0.897711 | 0.923507 | 0.896332 | 0.889292 | 0.839368 |
| 300 | 0.953941 | 0.933558 | 0.933829 | 0.896511 | 0.924456 | 0.896709 | 0.890789 | 0.837209 |
| 325 | 0.954085 | 0.936338 | 0.933829 | 0.898094 | 0.925146 | 0.897621 | 0.890674 | 0.836716 |
| 350 | 0.954085 | 0.936338 | 0.933829 | 0.898094 | 0.925779 | 0.899289 | 0.891826 | 0.842243 |
| 375 | 0.954085 | 0.936338 | 0.933829 | 0.897135 | 0.925693 | 0.898643 | 0.890674 | 0.8401 |
| 400 | 0.954171 | 0.936493 | 0.933829 | 0.897135 | 0.925837 | 0.898692 | 0.89148 | 0.841275 |
| 425 | 0.954171 | 0.936493 | 0.933829 | 0.897135 | 0.926182 | 0.899034 | 0.890327 | 0.839615 |
| 450 | 0.9542 | 0.936545 | 0.933714 | 0.895802 | 0.926412 | 0.90034 | 0.890442 | 0.84026 |
| 475 | 0.9542 | 0.936545 | 0.933714 | 0.895802 | 0.926383 | 0.900489 | 0.889981 | 0.839456 |
| 500 | 0.9542 | 0.936545 | 0.933829 | 0.896027 | 0.925174 | 0.900615 | 0.890096 | 0.839738 |
| 600 | 0.9542 | 0.936545 | 0.933829 | 0.894819 | 0.925174 | 0.900046 | 0.889178 | 0.837347 |
| 700 | 0.9542 | 0.936545 | 0.933714 | 0.893133 | 0.925059 | 0.902321 | 0.888258 | 0.836799 |

As can be seen in the table above, the sigmoid kernel did consistently worse than the SVM model, therefore the sigmoid kernel was not used against the testing set. After trying different possible values for C. The value of C with the highest F1 score and accuracy was 325, therefore the decision was to use 325 as the value for the hyper parameter.

The model trained using the training data was then applied to the testing data.

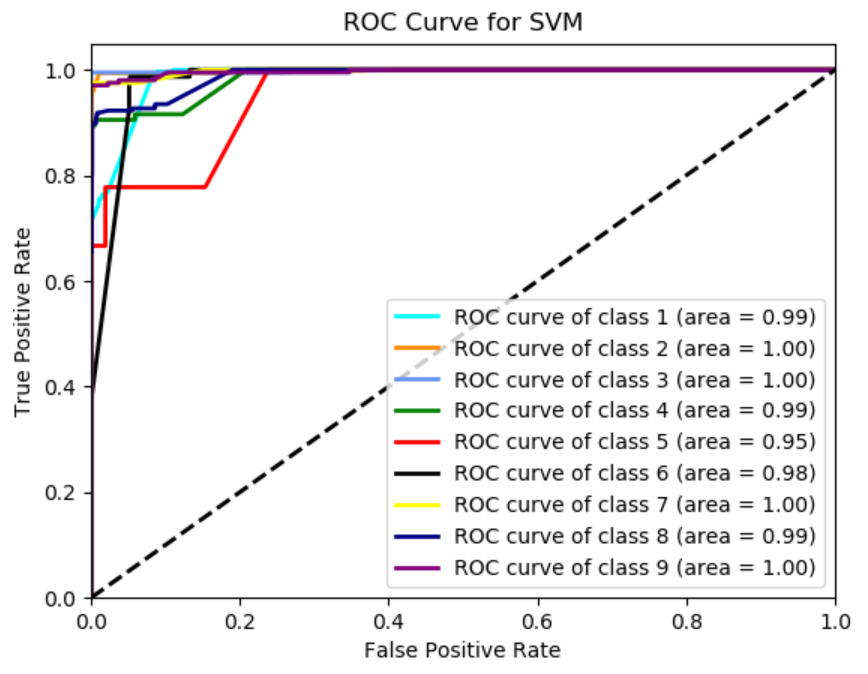
|  |  |  |
| --- | --- | --- |
| Classification | Test Accuracy | Test F1 Score |
| SVM (rbf kernel) | 0.92791 | 0.89735 |

Individual Class Precision and Recall

|  |  |  |
| --- | --- | --- |
| Class | Precision | Recall |
| 1 | 0.85 | 0.77 |
| 2 | 1 | 0.94 |
| 3 | 1 | 0.99 |
| 4 | 0.99 | 0.87 |
| 5 | 0.86 | 0.67 |
| 6 | 0.58 | 0.99 |
| 7 | 0.97 | 0.97 |
| 8 | 0.99 | 0.89 |
| 9 | 1 | 0.97 |

Confusion Matrix

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | Predicted Class | | | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Actual Class | 1 | 238 | 0 | 0 | 0 | 0 | 70 | 1 | 0 | 0 |
| 2 | 27 | 466 | 0 | 0 | 0 | 2 | 0 | 1 | 0 |
| 3 | 0 | 0 | 586 | 0 | 0 | 3 | 0 | 0 | 0 |
| 4 | 2 | 0 | 0 | 83 | 0 | 10 | 0 | 0 | 0 |
| 5 | 1 | 0 | 0 | 0 | 6 | 2 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 149 | 0 | 2 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 2 | 78 | 0 | 0 |
| 8 | 9 | 0 | 0 | 1 | 1 | 16 | 1 | 218 | 0 |
| 9 | 4 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 197 |



For Logistic Regression, the hyper parameter that required tuning was the regularization, which helps with overfitting and can also be considered as an embedded feature selection method. Again, cross validation was used with the F1 score and accuracy metric to determine the value of the hyper parameter.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Regularization | Train Accuracy | TrainF1 | Test Accuracy | TestF1 |
| 1 | 0.954229 | 0.937048 | 0.946258 | 0.91316 |
| 0.9 | 0.954229 | 0.937048 | 0.946027 | 0.912833 |
| 0.8 | 0.954229 | 0.937048 | 0.946258 | 0.915433 |
| 0.7 | 0.954229 | 0.937048 | 0.946372 | 0.915543 |
| 0.65 | 0.954229 | 0.937048 | 0.946488 | 0.917188 |
| 0.64 | 0.954229 | 0.937048 | 0.946488 | 0.917188 |
| 0.63 | 0.9542 | 0.936545 | 0.946488 | 0.917188 |
| 0.62 | 0.9542 | 0.936545 | 0.946488 | 0.917188 |
| 0.61 | 0.9542 | 0.936545 | 0.946488 | 0.917188 |
| 0.6 | 0.9542 | 0.936545 | 0.946488 | 0.917188 |
| 0.55 | 0.9542 | 0.936545 | 0.946257 | 0.914032 |
| 0.5 | 0.9542 | 0.936545 | 0.946257 | 0.914032 |
| 0.4 | 0.9542 | 0.936545 | 0.946142 | 0.913926 |
| 0.3 | 0.954171 | 0.935997 | 0.946027 | 0.912328 |
| 0.25 | 0.95397 | 0.93212 | 0.945912 | 0.91211 |
| 0.2 | 0.953797 | 0.930794 | 0.945797 | 0.913307 |
| 0.15 | 0.953768 | 0.930223 | 0.945913 | 0.913395 |
| 0.1 | 0.95374 | 0.930165 | 0.945336 | 0.912372 |
| 0.05 | 0.953509 | 0.925395 | 0.944991 | 0.907639 |
| 0.01 | 0.951985 | 0.917868 | 0.945221 | 0.908737 |

As can be seen from the above table, the range from 0.65 to 0.6 have the best F1 score and accuracy on the cross validation set. 0.6 was picked because that would increase the amount of regularization therefore reduce the amount of overfitting.

The model trained using the training data was then applied to the testing data.

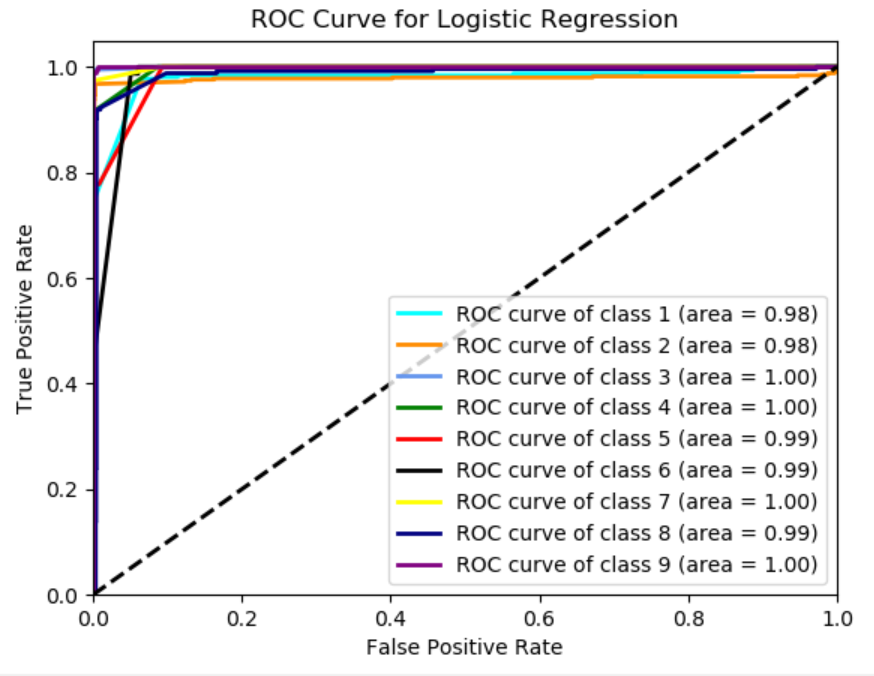
|  |  |  |
| --- | --- | --- |
| Classification | Test Accuracy | Test F1 Score |
| Logistic Regression | 0.93709 | 0.89618 |

Individual Class Precision and Recall

|  |  |  |
| --- | --- | --- |
| Class | Precision | Recall |
| 1 | 0.99 | 0.74 |
| 2 | 0.99 | 0.97 |
| 3 | 1 | 0.99 |
| 4 | 0.99 | 0.88 |
| 5 | 0.58 | 0.78 |
| 6 | 0.58 | 0.99 |
| 7 | 1 | 0.97 |
| 8 | 0.93 | 0.92 |
| 9 | 1 | 0.99 |

Confusion Matrix

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | Predicted Class | | | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Actual Class | 1 | 230 | 4 | 1 | 0 | 0 | 70 | 0 | 4 | 0 |
| 2 | 0 | 480 | 0 | 1 | 3 | 2 | 0 | 9 | 1 |
| 3 | 0 | 0 | 586 | 0 | 0 | 3 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 84 | 0 | 10 | 0 | 1 | 0 |
| 5 | 0 | 0 | 0 | 0 | 7 | 2 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 149 | 0 | 2 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 2 | 78 | 0 | 0 |
| 8 | 2 | 1 | 0 | 0 | 1 | 16 | 0 | 226 | 0 |
| 9 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 201 |



Conclusion

As can be seen, random forest had both the best F1 score and accuracy which makes random forest the best model to use for this classification problem. A thing to note is that for most of the misclassification, the misclassification was classified as the Tracur class. This could be due to the capability of Tracur. It can download and run files, including other malware and give a hacker control of your PC. Since the other malwares were backdoor, trojan and worms, it can be quite similar to Tracus which could be why the other classes misclassified it as Tracus.

Further Improvement

For further improvement, I would like to do a few things, first I believe that SVM can have better results with better tuning. There is another parameter Gamma that I did not tune. Gamma tunes the number of samples that have an impact on the margin and with that, the results may improve and perhaps be better than Decision Tree. Another things I would like to look at is 3 Gram Opcode, allowing for more specific combinations of opcode as features may improve the precision and prevent overfitting. Another things I think could be beneficial is the technique brought up by another student to increase the number of samples to reduce class inbalance which is SMOTE, this can help to increase the number of samples for the 5th class which has only 42 samples. The last thing I would like to try is N Gram Byte Code which is not looking at opcode but the raw binary data of the file itself, however this would probably require at least a 4 Gram which increases the feature space tremendously.