Observations

General

We obtained really good results. The individual models performed very well themselves which is clear from the results of the previous section. To ensure high confidence and robustness we used the ensemble of these three models for final predictions.

We chose an ensemble approach, as opposed to taking a big feature vector having features from strings, structure and dynamic data because these sets of features are expected to be independent and we get the general benefits of ensemble techniques such as avoiding overfitting.

Our decision to train multiclass classifiers was motivated by the reasoning that within malware classes the class distinctions will be sharper and therefore easier for the model to learn. This intuition was confirmed by training binary classifiers as well. The results for multi-class classifier were consistently superior.

We settled on RandomForestClassifier (an ensemble technique in itself) after trying multiple different classifiers for its quality and efficiency.

String analysis

During our manual analysis of the `String.txt` files of various classes we made the following observations which proved helpful in designing our model:

1. In `String.txt` file we observed that we can divide all the keywords into 2 types: **artefacts** (such as special characters and low length strings) and **useful** keywords such as english words. We concluded that artefacts are not useful for the task of classification therefore they can be filtered. For this we filtered out all special character containing strings and strings of length below 5 after a bit of experimentation.
2. Using the filter described in 1, we obtained frequency table of keywords in different `String.txt` files, and manually analyzed those. We observed that certain keywords such as `GetProcAddress` and `LoadLibraryA` have relatively better ranking in malware files as compared to benign. This suggests a pattern to malware classification based on keyword frequencies.
3. We also observed that overall benign files had a larger number of useful keywords. One possible reason can be that malware are more likely to be packed which also explains the observation 2. This again suggests that the frequency of keyword can be a useful feature for this task.

However the drawback with such an approach was large size of feature vector. Thus, we adopted the feature hashing trick to do the feature reduction. Using a feature hasher it is possible to transform a arbitrary size frequency table into a fixed size feature vector. We experimented with the size of this feature vector to maximize the accuracy while keeping training times optimal.

Structure analysis

PE file is a rich source of features, so it is important to get all the right features without polluting the model with too many useless features. The observations which guided the modelling decisions for structure analysis are as follows:

1. Some features don't vary and therefore don't bear on the classification. VarianceThreshold removes such features.
2. A lot of things are repeated across sections. We have used derived features count, mean, min and max to capture the distribution of repeated features. An important feature of this kind is entropy which is mentioned for different sections. Entropy is important because we know that malwares use special packing and encryption for obfuscation, which directly affects the entropy.
3. The nesting structure of the sections captures the directory tree structure of the resource directory and we intuitively felt that this directory tree structure would be different for malwares and benign softwares, as it is used for resources such as icons, etc. which malwares may not care about.
4. A very important information in the PE file is the .dll files which are imported. Malwares will import a characteric group of .dll files for system or network related functions, a pattern which may well distinguish it from benign softwares. We have laid emphasis on this, making each .dll file imported a feature in itself.
5. Apart from this, we observed the presence of some flags in certain sections. Each flag is made a feature as malwares and benign softwares are likely to have different requirements of flags.

Dynamic analysis

Manually skimming through the given JSON files for different classes and consulting the internet we are able to draw following significant observations :

1. Some signatures are already present in the given JSON files which indicate the results of some preliminary tests performed by the software itself (or the data collector). In these signatures we apparently found some very indicative ones like "Antivirus test results" and "packed malware found". Each of the signature was further already associated with a severity level indicative of its likelihood of being malware. Thus, this became on of our features in the model. In particular we used maximum of these severities as one component in our feature vector.
2. We realized that certain important activity dumps primarily network activity and duration of run which otherwise could not be obtained from static analysis should definitely be included as features. Network activity is known to be a good discriminator.
3. Although API calls made are often analyzed in static analysis itself,we observed that some packed malwares manage to sniff through it. Therefore API calls traced from dynamic analysis report is more reliable and thus one of our features.
4. The report also included a summary of actions, such as number of files read, opened, deleted. Number of services created, registry entries modified, deleted etc. These summaries proved to be useful discriminators as well.