John Bellamy

Battelle Data Science Challenge Problem

Goal: 1) Through exploratory analysis and *feature selection* identify the traits that appear to be correlated with virus infection based on the Microsoft Malware Prediction data, and how the traits affect risk for infection. **2)** Build and test a model based on features found in feature selection. **3a)** Explain how I would approach this problem with more time and if it was encountered "in the wild". **3b)** Build a model using *feature extraction.*

Challenges: We have a lot of columns and we have missing values. We need to treat categorical and discrete data differently than we would continuous data. This is always challenging when we aren't familiar with the data. This is doubly-true when time is short. The data set is large so it takes quite a bit of time to run on one machine. I had to sample the data so make it work.

Data: Selecting option 2 there are a couple problems that needed to be taken care of immediately. I only have 16 GB of memory on my machine, so I took a random sample of the full training set (6.5%) to give me more memory space for modeling, and the ability to get a model trained faster. I also increased swap on my machine. The prior is scripted in the data_prep folder. I also used Dask to speed up some processes.

The data is somewhat balanced between the two classes in "HasDetections", which I thought was surprising. Observations for 0 were 290032 and observations for 1 were 289864.

Feature Engineering Approach: We need to see what sort of data we are dealing with (Kaggle page helps too) and plan to deal with the data types in the set. Straight off we are able to identify categorical data (these are represented by text), in 19 columns with some data being nominal (like 'ProductName').

The rest of the columns at first glance appear numeric, but after looking at data and Kaggle we can see many of these like OsBuild, OsSuite, and etc. are nominals and columns like 'Census_IsWIMBootEnabled'. The only four continuous variables I identified are: 'Census_InternalBatteryNumberOfCharges', 'Census_SystemVolumeTotalCapacity' 'Census_TotalPhysicalRAM' and 'Census_PrimaryDiskTotalCapacity'.

There could be some argument made that Physical Ram could be treated as nominal, (or maybe even turned into ordinal with binning) but I would argue these variables are continuous within an interval. *Note: It is possible that in my rush I looked over something. I looked over* 'DefaultBrowsersIdentifier' and had to remove it later.

After identifying the types of data we are dealing with I filled all null values through imputating. For continuous variables we impute the mean (I prefer mean imputation for continuous variables) and for the categorical/nominal variables we impute with most frequent occurring value. ***See Data Preprocessing notebook in data_prep folder.***

Goal 1 Outcome: Using Dask to speed up process, data saved in the cleaning phase was loaded into a pandas dataframe. The categorical variables were encoded with LabelEncoder. The first 81 features (MachineIdentifier not included) were then stored as a Dask array. Likewise, our target variable was stored in a Dask array. 40 features that explained ~91% of variance was chosen through a RandomForest. ***See Exploratory and Feature Selection in exploratory folder.***

There needs to be some population adjustments made for many of the features, since some objects are common in datasets where we divide the positive and negative class. For example, we see

country that was coded 42 appearing at the top of both datasets. Actually, top five are in both sets. Countries with codes 88, 213, 200 appear to be more likely to not have a detection. 59 and 206 appear more likely to have a detection.

However, it looks like the SmartScreen that is coded 3 is more likely to have a detection. Screens coded 5 and 9 are also slightly more likely to have detections.

Looking at pairplots for each class we see that secure boot is not enabled for *most* samples. However, in the the sample with no detections the numbers are closer to even. We also see a correlation between 'Census_InternalBatteryNumberOfCharges', ram and storage.

Goal 2 Outome: Usually, one would try several different models (see last section: *Out in the wild*) but for our purposes (to demonstrate how well feature engineering and selection went) we chose one algorithm GradientBoostingClassifier. The features found in Goal 1 were used as the data from which a testing and training set were defined.

After testing the model we showed an roc_auc score of 70%. However, we can certainly do more work to improve this model. Cross-validation and more time spent on feature selection/engineering. For example, some add more features than the 40 chosen and double-check there weren't any 'bad' columns left in like the BrowserIdentifier mentioned before.

Out in the wild (Goal 3a): Generally, when I model (let's assume we are dealing with a supervised learning problem) I follow the steps below.

1) Look at summary statistics and plan how I will engineer the data. Where are there nulls? What kind of data is there? Are there any extreme values in continuous variables? This step was skipped in this challenge.

2) I engineer based on the finding of 1. Remove nulls, as most algorithms cannot use nulls. Save cleaned/engineered data as needed.

3) I then explore the engineered data. However, if we are dealing with highly dimensional data, it's probably best to explore only those features that seem to be correlated to the target variable. That is the approach I took with this problem. If we have data with only a few dimensions, we might well explore all columns, though smaller dimensional data does not mean we don't have to worry about feature selection/extraction.

4) Model data. If a model is the goal of the analysis, then I build at least four models and pick the model that performs best on the data. With highly dimensional data I might also try feature extraction in addition to feature selection techniques in combination with various ML algorithms. For example, using PCA and a classification algorithm to choose only the components that add to the model. I would then use gridsearch and k-fold cross-validation on at least the algorithm that performs best when tuning the final model.

3b Outcome) Here we built a model using the same GradientBoostingClassifier, but this time used SVD to reduce dimensionality. Using 40 components the model had an roc_auc score of just below 70%. Both models could use more work, obviously and it is likely I missed something in my haste :)

Conclusion: Although I took more time to complete this than I planned, it was fun and challenging. I wish I would've spent more time studying the variables, but with the time crunch I was able to achieve the goals I established to some degree or another. With some more feature engineering and GridSearching for the best parameters there me be a model here yet. Furthermore, spending more

doing exploratory analysis with the features that show a correlation to the target would likely yield some m ore interesting insights.