Microsoft's Malware Prediction Challenge

Shivesh Ganju† and Ravi Shankar‡

†Department of Computer Science, Courant Institute of Mathematical Sciences, New York University

An Important Problem

The malware industry continues to be a well-organized, well-funded market dedicated to evading traditional security measures. Once a computer is infected by malware, criminals can hurt consumers and enterprises in many ways. This project aims in classifying if a malware can/can't infect a window's machine based on different properties of the machine. The model classifies if a machine is getting infected by malware by performing Machine learning algorithms such as Support Vector Machines, Random Forests and LightGBM and then compares the performance of the models using some performance metrics such as Area Under Curve and confusion matrix.

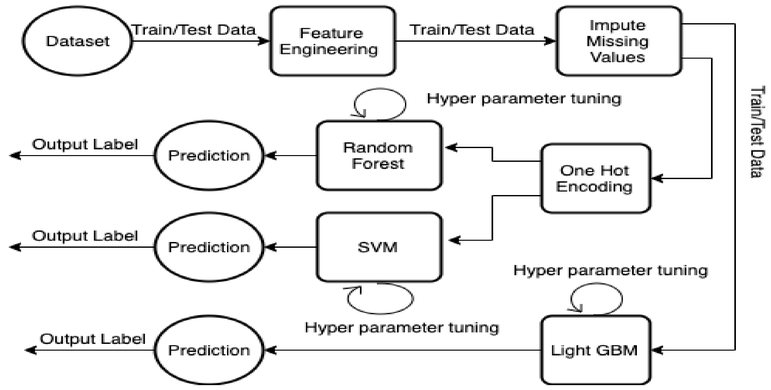
Motivation

This problem aims to solve a plaguing real world problem in malware industry where there are more than a *billion* customers using the enterprise products and softwares.The classification problem also helps us in learning some of the basic feature engineering techniques such as categorical feature handling using one hot encoder, handling large data-sets and deriving new features from existing ones which are essential for any machine learning engineer. Moreover this problem helps us get a hands on experience on applying algorithms learnt during the course such as Support Vector Machines using Linear kernel, Random Forests on a large data-set as well as exploring into the world of competitive Machine learning where we would be testing algorithms such as LightGBM which is a tree based ensemble model.

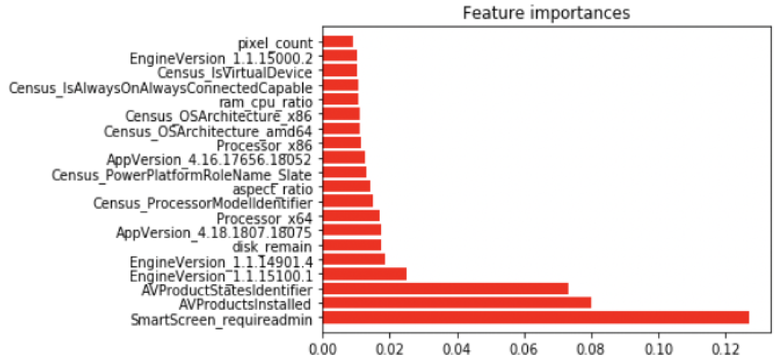
Methodology

• ***Splitting data into training and validation set***
Training data was split into training and validation data in a 75/25 ratio.
• ***Replacing missing data***
Imputation of missing values (NAN) of numerical features were done with the median value for the feature. For categorical features missing values were replaced with mode.
• ***Feature Engineering***
Numerically engineered features were created from the original features.
• ***Categorical Feature Handling***
Almost 30 features were categorical features which were one-hot encoded before deploying the machine learning model.
• ***Dimensionality Reduction***
Feature selection using feature importance from Random Forest.
• ***Machine Learning Models Used:***
∗ **Random Forest**: Used sklearn's $RandomForestClassifier$ to train our model.
∗ **Linear SVM**: Implementation of $LinearSVC$ was done in two ways. One was using linear kernel on the entire dataset and once after applying dimensionality reduction on the training data set to measure the performance improvement.
∗ **LGBMClassifier**: LGBMClassifier uses internal frequency coding for handling categorical features hence one-hot encoding was not needed.
• ***Validation Testing:***
Validation testing was performed using **K-Fold** validation technique where **5** fold validation was used since the data was be large.
• ***Performance Testing:***
Classification accuracy was measured through a **AccuracyScore** of sklearn, plotting confusion matrix for checking for false positives and false negatives and **AUC** was used to check for false positives against true positives.

Algorithm



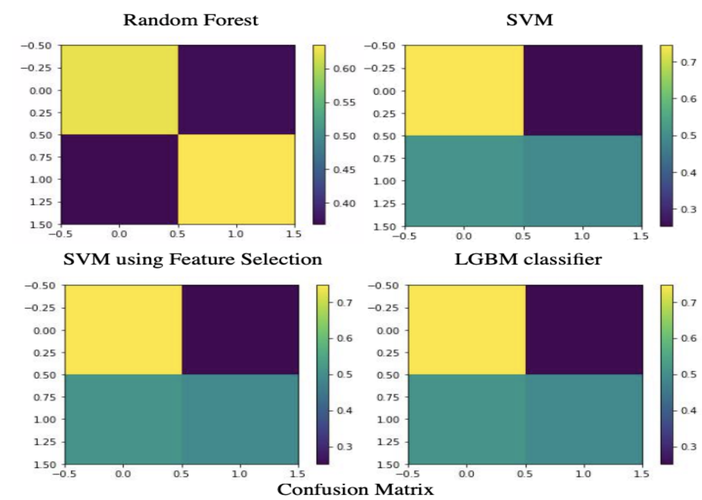
Results



Classifier	Test Error	AUC Score
Random Forest	36.9%	0.630
Linear SVM	38.6%	0.618
Linear SVM + feature selection	38.1%	0.619
LGBM Classifier	37.5%	0.625

SVM verified that our hypothesis was correct and indeed after discarding unimportant features there was an improvement in performance.
From the above we can make 3 claims
• In this problem Tree based ensemble methods were suitably better for handling categorical features and large data set than SVM in terms of computational performance and accuracy.
• Dimensionality reduction improved performance for SVM the reason for the same might be because unimportant features affect the support vectors such that it degrades the accuracy.

Comparison



According to our findings Random Forest's performance was better than SVM and LightGBM.From the confusion matrix we can see we have a higher amount of false positives than false negatives. This is still good because in our problem it is better to classify a machine as a malware than not classifying a malware machine as one. From the business perspective, false positive cases helps them to put additional checks on that machine so that it does not get affected by malware. The highest AUC score also implies that this model has the best measure of seperability than the other classifiers

References

1.https://www.kaggle.com/c/microsoft-malware-prediction/
2.https://medium.com/hugo-ferreiras-blog/dealing-with-categorical-features-in-machine-learning-1bb70f07262d
3.https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc
4.https://lightgbm.readthedocs.io/en/latest/