Abhinav Vittal

Justin Ross

QTM 347 Machine Learning

December 13, 2023

**Detecting Malware in Microsoft Machines**

**Abstract**

This project is an attempt at the [Microsoft Malware Prediction Kaggle](https://www.kaggle.com/competitions/microsoft-malware-prediction/overview) competition from 2018. Using a train and test dataset composed of binary, categorical, and numeric variables, competitors must predict whether a given computer has malware. Our analysis began with feature selection. We applied Principal Component Analysis (PCA) for numeric and binary features and employed the Least Absolute Shrinkage and Selection Operator (LASSO) for categorical ones. With these methods, a comprehensive list of variables was created. These features were tested in the following models: logistic regression, random forest, KNN, XGBoost, and gradient boost. The success of the model was determined by the Area Under the Receiver Operating Characteristic (ROC) curve and to a lesser extent the F1 score.. The Random Forest offered the best performance in both metrics. We then tested the chosen model on a previously unseen dataset. The model's accuracy dipped from 69% on the cross-validated test data to 62% on the independent test set. This study emphasizes the importance of robust model performance evaluation across diverse datasets for reliable predictions in practical scenarios.

**Introduction**

The illegal malware and hacking industry poses a significant and lucrative threat. With the advent of technology, corporations such as Microsoft find themselves entrusted with safeguarding the security of their over *one billion* strong customer base. The consequences of hacking extend far beyond financial losses, delving into personal implications that can be far-reaching and long-lasting. Given the high stakes involved, the timely detection of malware emerges as a pivotal defense in mitigating cyberattacks. The practical applications of such a problem are easily seen; with over $13 billion lost annually due to malware, the industry is undoubtedly lucrative. There are a plethora of approaches to solving this problem, however, within class constraints, it appears considering this a classification problem is best; either a machine has malware or does not. To that end, a variety of decision making models must be tested. At first glance, the following models should be tested: logistic regression, random forest classifier, gradient boosting classifier, and KNN classifier. Due to Kaggle’s privacy policy, we are unable to view the exact code behind other similarly successful models. However, considering the comparable quality of our results and Kaggle’s best performers along with existing online materials, we believe our approach is the most robust. Importantly, model hyperparameters were selected using either a bayesian search (to minimize the computational power needed – a seriously limiting factor) and the model was cross-validated to ensure no overfitting occurred. The key components include the target variable “HasDetections” which is either a 1 or 0 and the 52 chosen features. The largest limitation was the size of the data. With a 3.5GB train data frame and a 4.1GB test data frame comprising 7 million and 9 million rows respectively, the entire data could not be used. A stratified random sample of size 50,000 was selected from the train data based on the population proportion of 1s to 0s.

**Setup**

Considering the extensive set of 52 features, providing a detailed description of each one becomes impractical. Before selecting our features, the data was appropriately scaled and separated based on data type, as previously outlined. We then one-hot encoded our categorical variables so that our models could process them. Following the implementation of PCA and a LASSO regression, the resulting dataset was categorized into 46 categorical variables and 6 numerical and binary variables. The “HasDetections” column in the training data was split almost evenly, with half of the computers having malware. Notably, the numerical variables exhibited a more pronounced impact on the target column. Given the complexity arising from the multitude of features, discerning the precise nature of their relationships with the target—whether linear, logarithmic, etc.—poses a challenge.

All models employed in our analysis revolved around classification. Logistic regression yielded log-odds, KNN clustered data points, and the random forest utilized decision trees for classification. Recognizing the pivotal role of hyperparameters in model performance, a Bayesian search was leveraged for optimization. Given constraints related to data size and computational resources, the optimization process, along with hyperparameter testing, was executed through cross-validation to ensure robustness and reliability in model tuning.

**Results**

The Random Forest model exhibited superior performance in our cross-validated results on the training dataset. It boasts an impressive Area Under ROC of 77.15% and an F1 score of 0.6976. These metrics surpassed those of all other models tested. The cross-validated accuracy on the training data was 69%. However, when the model was assessed on the test data, the accuracy dropped to 62%, indicating a potential challenge in generalizing the model's performance beyond the training set.

In testing alternative models, several challenges were encountered. A Support Vector Classifier (SVC) model was abandoned due to excessively long compute times required for hyperparameter optimization. XGBoost emerged as the second-best performer among the algorithms, aligning with the expectation given its tree-based, ensemble nature adept at handling non-linearity. This finding also implies that the underlying data might possess non-linear characteristics, posing challenges for simpler linear models. The distinctive feature of XGBoost lies in its sequential error correction approach, which contrasts with the averaging mechanism of random forest. The comparable performance of logistic regression, KNN, and gradient boost classifier models suggests their limitations in outperforming each other significantly. This analysis underscores the importance of model selection and the nuanced interpretation of results, offering valuable insights into the dataset's characteristics and the efficacy of different machine learning approaches.

|  |  |  |
| --- | --- | --- |
| **Model** | **Area Under ROC** | **F1 Score** |
| Random Forest | 0.7715 | 0.6976 |
| XG Boost | 0.7194 | 0.6468 |
| GradientBoostingClassifier | 0.6975 | 0.6343 |
| Logistic Regression | 0.6908 | 0.6273 |
| KNN Classified | 0.6906 | 0.6911 |

**Discussion**

*Results*

Initially, the results might have seemed less than encouraging – a 70% would be  a terrible score on a test. However, a closer examination of alternative models provided valuable insights into the complexity of the dataset. Notably, the performance of our models closely paralleled widely accepted solutions. With ML resources ubiquitous online, implementing models are all but standardized. The differences between approaches lies in feature selection. The number of principal components used and the alpha parameter in our LASSO regression played a crucial role in shaping the model's performance. This underscores the significance of the preprocessing steps in determining the effectiveness of our models. By leveraging techniques such as PCA and LASSO, we strategically curated our feature set, influencing the subsequent model outcomes. Despite the challenges posed by the drop in accuracy on the test dataset, our results stand on a fairly solid foundation.

*Using Big Data*

This project was quite eye-opening on the changes in how we build an end to end ML model using large datasets. With over 7GB in total data, we definitely traversed into the realm of ‘Big Data’. Due to processing and RAM limitations, we had to think of creative ways to work around these limitations in order to successfully build our model. To begin with, we upgraded to Google Colab Pro, which gave us access to more RAM and different processors such as GPUs. This provided our runtime session with enough memory to avoid a kernel crash. We also learned the importance of cloud computing in Machine Learning and hope to move away from Google Colab and towards AWS, as it utilizes a ‘pay as you go’ pricing option which can be a cost effective option when building a ML model with a lot of data. We also implemented the use of Pyspark, which is the Python API for Apache Spark. Pyspark has the ability to parallelize computations across multiple workers (cores) and thus can perform distributed computations on dataframes significantly faster than Pandas. As we move forward in our data science career and work with larger datasets, we might consider moving away from the traditional Pandas library and use Polars or Pyspark which are better suited for ‘big data’.

**Conclusion**

In conclusion, this project sought to address the critical challenge of detecting malware on Microsoft machines, a task of immense importance given the severe financial and personal consequences associated with malware attacks. Our methodology involved a systematic approach to data cleaning and model evaluation. To ensure a representative dataset, we initiated the process by applying a random stratified subset technique. Subsequent data cleaning steps included the exclusion of columns with null values exceeding 50%, and the appropriate handling of other missing values. This meticulous preparation set the stage for a comprehensive exploration of various machine learning models. A diverse set of models underwent rigorous testing, with hyperparameters optimized using a Bayesian search and models cross-validated for robustness. Our key evaluation metrics, the Area Under the ROC Curve and the F1 score, guided the selection of the most effective model. Notably, the random forest emerged as the top-performing model with a cross-validated accuracy of 69% on the training dataset.

However, the transition to the test dataset revealed a slight decrease in performance, emphasizing the challenges of generalization. Our model yielded an ROC AUC score of 0.677 when validated against our validation dataset, obtained through a stratified random subset. We consider this result to be an overall success, as it would place our model amongst the top submissions for the Microsoft Malware Competition. The insights gained from this project provide a solid foundation for future efforts to enhance malware detection capabilities. In essence, the project not only contributes to the advancement of cybersecurity but also underscores the intricate nature of real-world data science applications.