Malware Prediction

***Abstract*—Malware refers to malicious software designed to infect and harm computer systems by exploiting vulnerabilities in their software and hardware. Malware detection is of crucial interest to operating system manufacturers aiming to prevent their consumers and enterprises from befalling harm. With the increase in malware types and their rate of spread, manual heuristic malware analysis is no longer effective. Manufacturers are instead using data science techniques to predict whether a system will be hit with malware in the near future. Prediction, followed by taking preventative action in endangered systems can be a very effective measure of protection from malware. This paper takes a look at different machine learning techniques that can be used to predict a system's probability of getting hit by various families of malware, based on different properties of that system. Given a dataset of these properties and the machine infections, the proposed solution is to use a gradient boosting framework, namely LightGBM, to build a model that predicts whether a system will soon be hit with malware.**

***Index Terms*—malware, malware detection, machine learning**

I. Introduction

With rapidly evolving malware and newer methods of malware dissemination via the internet, rudimentary methods of malware detection involving static analysis of executables is no longer sufficient to prevent modern systems from malware attacks. Operating system manufacturers who have a vested interest in protecting their users from malware attacks are constantly searching for more effective malware detection systems to stay one step ahead of malware perpetrators. Early detection of at risk systems to secure them before malware attack is an interesting strategy that exploits newly developed machine learning techniques.

Certain software and hardware properties such as a system's antivirus software, operating system version etc. can help us detect at-risk systems. For example, a system running on an outdated operating system version which doesn't reflect recent bug fixes might be more vulnerable to malicious attacks than an up-to-date system. Similarly systems with a weak or no antivirus software are in general more prone to malware infections.Most operating systems manufacturers collect telemetry data of systems running on their software. This data would include properties of interest to our application, namely, system configurations and malware infection reports. Given such a dataset consisting of machine properties and whether the machine was hit by malware, a model to predict whether a new machine given its properties will be hit by malware soon, can be built. There are various machine learning techniques that can be used to model the patterns associated with malware attacks in computer systems. The ones explored in this paper are: LightGBM and LSTM. Taking into consideration the given dataset, LGBM is settled on as the best approach to the proposed problem statement.

II. Literature Survey

A number of papers have been written on predicting the risk of a system being hit with malware. While most implementations involve analysis of executables within a system, there do exist quite a few studies that delve into using predicting malware risk using machine learning techniques. One such study by Fanny Lalonde Lévesque, José M. Fernandez and Anil Somayaji [1] presents a neural network model predicting risk of malware victimization based on a user's demographic and their online behavior. Another study by Chanhyun Kang, Noseong Park, B. Aditya Prakash, Edoardo Serra, V. S. Subrahmanian [2] looks into predicting the number of malware infections in a country given a history of detected malware attacks on systems registered to that country using ensemble models.

III. Problem Statement

The dataset for the proposed work chosen was the Microsoft Malware Prediction dataset from Kaggle which consists of system properties and machine infections of systems running on Windows [3].

Some of the properties are antivirus version, operating system version, engine version and processor architecture. The dataset was compiled by Microsoft using heartbeat and threat reports collected by Windows Defender. Windows Defender serves as Windows operating systems' endpoint protection against malware. The problem which we tried to solve was to predict if a machine will soon be hit with malware. There are totally around 9 million rows and 84 columns in the training dataset and there are another 9 million rows and columns 83 in the test dataset. Each row in the test and train dataset corresponds to a system, uniquely identified by a 'MachineIdentifier'. 'HasDetections' is the target and indicates whether Malware was detected on the system. 'HasDetections' is missing in the test dataset and must be predicted using the train dataset.

Along with the aforementioned dataset we are also using the Antivirus Signature vs Timestamp dataset [4]. It consists of mappings from antivirus signature versions to timestamps. Antivirus signature version ('AvSigVersion') is updated approximately every 2 hours. 95% of user antiviruses regularly update their antivirus signature version making them a trustworthy timestamp for each dataset observation. This means that the antivirus signature version of a system when it was sampled can be mapped to the time at which the system was sampled. The timestamps from this dataset are provided by Microsoft. Microsoft has derived these timestamps in the manner explained above, by approximating sampling time from 'AvSigVersion' of the observation.

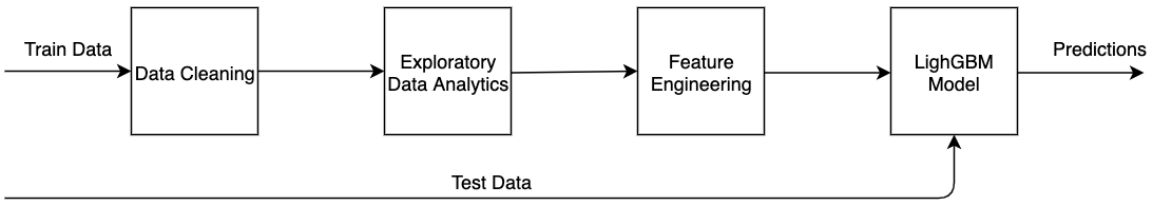

Fig. 1. System Architecture of Proposed Solution

This kind of malware risk detection is in essence a time series problem, with the sampling date of each data point greatly influencing the some of the system's properties. The given data set is split into test and train in such a way that a majority of entries in the train data are from August and September 2018 while the testing data is mostly from October and November 2018.But the problems posed to the traditional time series approach by this dataset are the following:

- New systems are added to the dataset with time.
- There are systems that occasionally go offline for variable durations of time. No data from these systems are recorded in this period.
- Systems receive OS patches, bug fixes and OS upgrades over time thereby changing their properties. [1]

This analysis is intuitive as newer versions of operating systems and antivirus software are introduced to combat ever-improving malware.Given the shortcomings of having a plain time series perspective of the problem, it is best to have a final model that is not strictly a time-series approach to malware prediction, but can accommodate features that are indicative of time. Based on this there were two approaches we pursued: LSTM or Gradient boosting Decision Trees. To capture the time series aspect of the problem we engineer new features by making use of the Antivirus Signature vs Timestamp dataset.

IV. Proposed Solution

The proposed solution is a Gradient Boosting Decision Tree (LightGBM) to predict the probability of a machine being hit by a malware soon.

A. Data Cleaning

Before the model was trained, some data cleaning had to be done. There were 5 columns in the dataset which had above 70% empty values. They were dropped.

The columns were :

- PuaMode
- CensusProcessorClass
- DefaultBrowsersIdentifier
- CensusIsFlightingInternal
- CensusInternalBatteryType

A column named SmartScreen had some inconsistent values. There were values like 'promt', 'promprt', etc for the value 'prompt','of' in place of 'off', 'enabled' for 'on','00000' for '0' and so on. Those inconsistent values too had to be replaced by the proper values.We also encountered many NaN values for many rows. So we replaced the NaN values with the median of the column to which it belonged to. We chose the median because Median is robust to outliers. Also looking at the test dataset, many category values for certain categorical attributes that were present in the training data were missing in test data. Hence they will not be of much use in predicting for test data. Such category values were replaced with a uniform dummy value prior to encoding. For all categorical variables sorted frequency encoding was done.

B. Exploratory Data Analysis

Dataset consists of about 8 million rows and 83 columns describing several attributes of Microsoft devices under consideration.

Target variable is whether the malware is detected or not.

Figure 4 shows that data is balanced with respect to target attribute hence no need of up sampling any class.

Figure 5 shows as expected Microsoft has many more computers that touch devices. The rate of infections is also lower for touch devices.

Analysis of attributes is divided into mainly following categories [7]:

- Variables with high amount of unique values
  - Engine Version
  - App Version
- AntiVirus(Av) Attributes
  - AvSigVersion
  - AVProductStatesIdentifier
  - AVProductsInstalled
- OS Attributes
  - OsPlatformSubRelease
  - OsBuildLab
- Processor Attributes
  - CensusProcessorCoreCount
  - CensusProcessorModelIdentifier
- Other important attributes
  - CensusPrimaryDiskTotalCapacity
  - CensusTotalPhysicalRAM

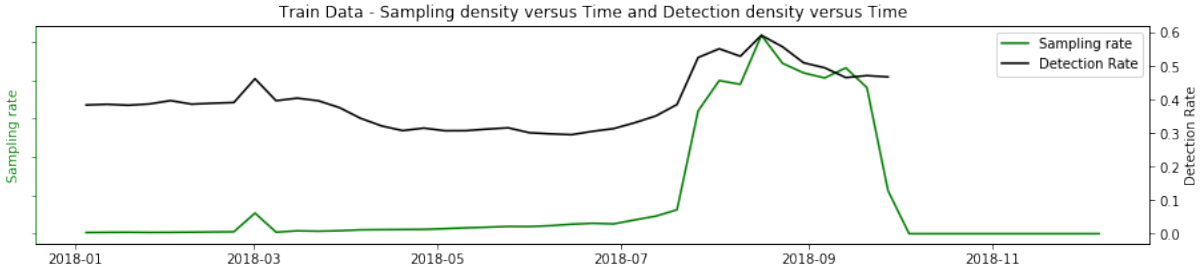

Fig. 2. Train Data - Sampling density versus Time and Detection density versus Time

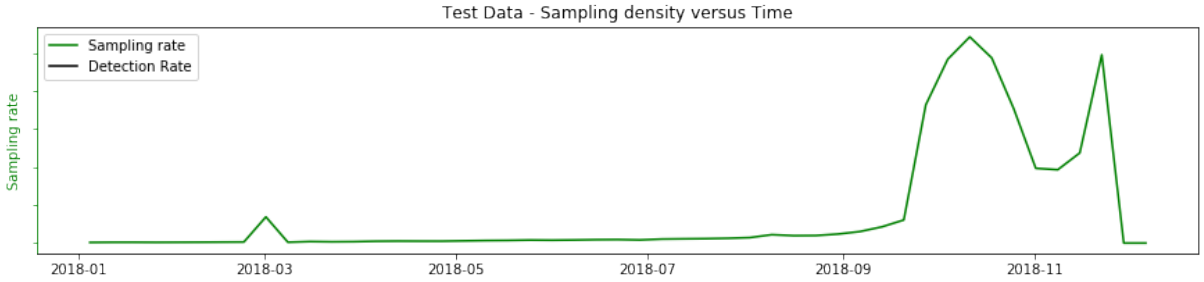

Fig. 3. Test Data - Sampling density versus Time

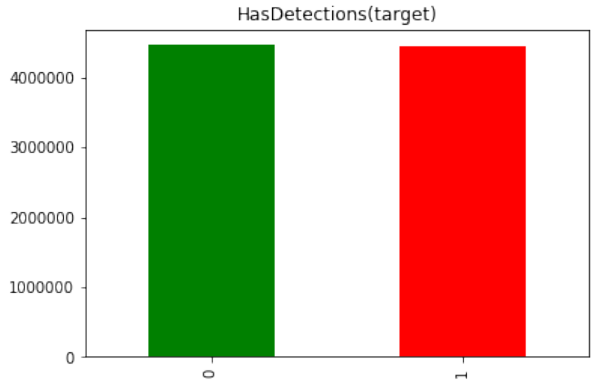

Fig. 4. Distribution of Target attribute across the dataset

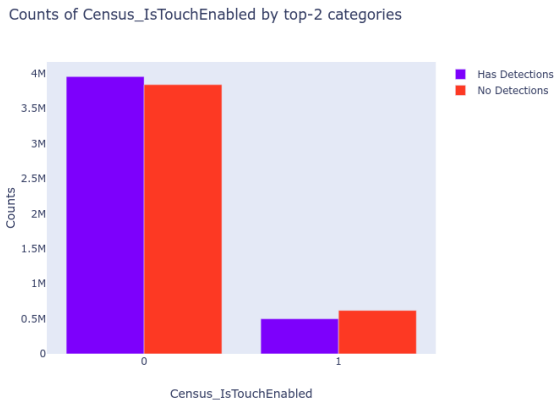

Fig. 5. Distribution of devices in touch and non-touch categories

- CensusInternalPrimaryDiagonalDisplaySizeInInches
- CensusInternalBatteryNumberOfCharges
- CensusChassisTypeName
- CensusMDC2FormFactor

A deep dive on some of the above mentioned attributes:

1) AVProductsInstalled: If a computer has an antivirus, it is less likely to be infected. But Figure 6 shows that having two antiviruses has an opposite effect.

2) CensusMDC2FormFactor: Figure 7 shows that Detachable devices usually have SSD, desktop PC have HDD and SDD and Notebooks usually have HDD.

An exploration of the time series aspect of the problem - As mentioned previously, malware detection could be viewed as a time series problem. To visualize this, plots sampling density over time were drawn for both train and test data as seen in Fig 2. and Fig 3. respectively. The green line depicts sampling density over time and the black line represents the malware detection rate (only for training data) over time. The following observations can be made:

- A majority of the train data is from August to October 2018.
- A majority of the test data is from October to December 2018.
- It is evident that there exists a relation between detection rate and sampling rate as the detection rate roughly increases and decreases with sampling rate for the training data.

The last point indicates that it would be useful to engineer a feature that can capture the sampling rate at the time the

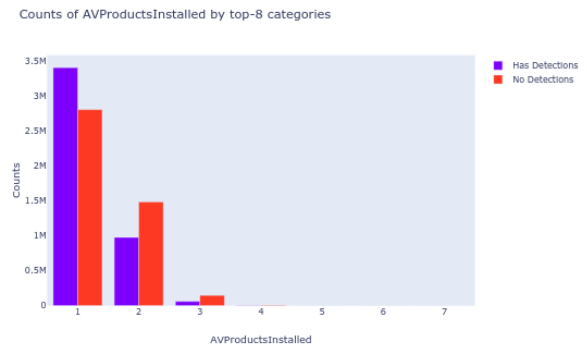

Fig. 6. Counts of AvProducts Installed

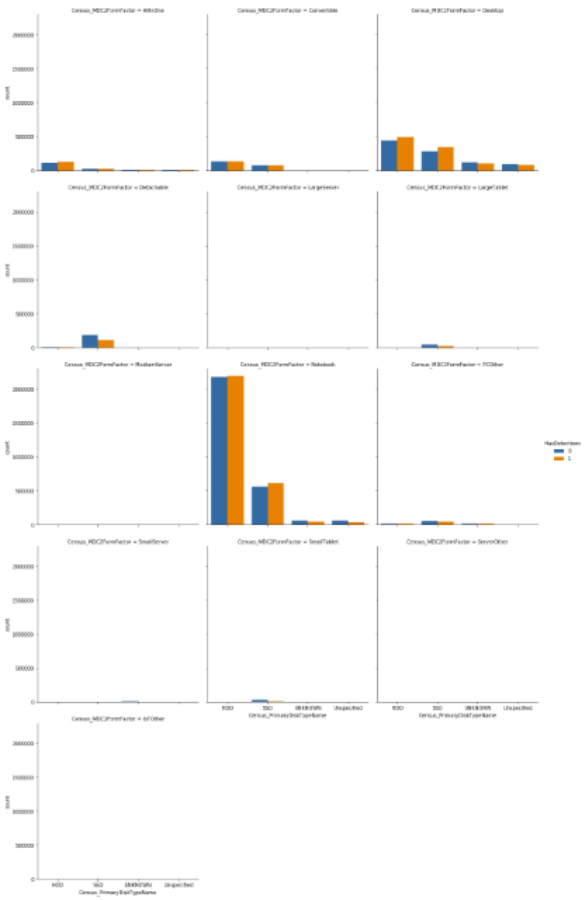

Fig. 7. Correlation between CensusPrimaryDiskTypeName and CensusMDC2FormFactor

observation was sampled, as a machine is more likely to have malware if it is sampled from a period with a high sampling rate.

C. Feature Engineering

To capture the sampling rate, we engineer a new feature 'WeekNumber' which is the number of weeks that had passed since 1st Jan 2018 when the observation was sampled. If an observation was sampled before 1st Jan 2018, this attribute will have a negative value.

The newly engineered feature is now representative the sampling rate at the time at which the observation was sampled. This new feature is important for predicting whether the observation is of a system with a malware detection. An observation with 'WeekNumber' value equal to the most frequent value of 'WeekNumber' will have been sampled from the time period when sampling rate was very high. When the sampling rate is high detection rate is high as seen in Fig. 5. Thus an observation with a common 'WeekNumber' value is more likely to have malware detection.

We later apply sorted frequency encoding on the attribute as the sorted frequency of 'WeekNumber' is essentially sampling rate, and sampling rate directly correlates with the target which we are trying to predict (Malware detection rate).

D. Light Gradient Boosting Machine

Boosting involves converting an ensemble of weak learners, usually decision trees into a strong learner. LightGBM is a gradient boosting framework that uses a tree based learning algorithm.

The advantages of using LGBM for the given problem and dataset is as follows:

- As stated before, the given dataset consists of several categorical variables with high cardinalities. LGBM internally handles categorical variables by applying integer encoding. The disadvantage of having to develop an embedding system is forgone.
- The given dataset has missing values for several attributes. LGBM is better equipped than neural networks to handle these missing values. This is done internally by ignoring missing value while splitting and later allocating it to the side reduces loss the most [5].
- LGBM is better at memory optimization for large data sets and takes lower memory to run. It also supports parallel and GPU learning speeding up the learning process [6].

To summarize, the best model for handling missing values, handling categorical features is the Light Gradient Boosting Machine.

V. Evaluation Metrics

Since the test data provided didn't have target values,the train data was itself further split into train and test data for model training.

The evaluation metric used was Area Under the Curve.

VI. Experiments

A. LightGBM

We trained an LGBM model with the following parameters:

- number of leaves = 250
- learning rate = 0.02
- number of boosted trees to fit = 1800
- boosting type = Traditional Gradient Boosting Decision Tree

The model gave an AUC of about 0.67

Fig 8. depicts the feature importance of 20 of the most important features of the train data for the model. Note that newly engineered feature 'WeekNo' is one of the most important attributes for predicting malware.

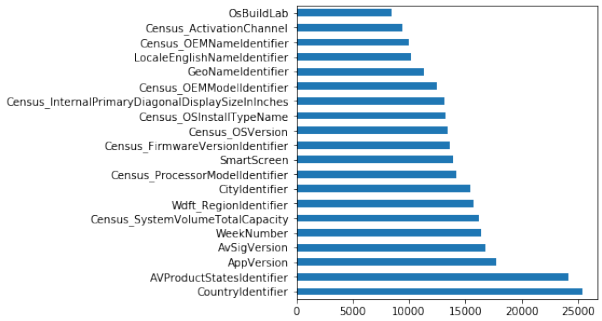

Fig. 8. Feature importance of the 20 most important features- sourced from the trained LGBM model

B. LSTM

Before trying LGBM, we experimented with LSTM.

With the recent breakthroughs that have been happening in data science, it is found that for almost all of sequence prediction problems, Long short Term Memory networks, a.k.a LSTMs have been observed as the most effective solution.

LSTMs have an edge over conventional feed-forward neural networks and RNN in many ways [8]. Infact, the Long Short-Term Memory network or LSTM network is a type of recurrent neural network used in deep learning because very large architectures can be successfully trained. Hence we trained a LSTM model with 1 hidden layer consisting of 256 memory units with sigmoid as activation function, adam optimizer and binaryCrossentropy as loss function. This model gave an AUC score of 0.55.LSTM-CNN was also tried. But similar results were obtained.

VII. Results

The LightGBM model had an AUC score of 0.67.

The LSTM model had an AUC score of 0.55.

The LightGBM outperformed the LSTM as expected.

VIII. Conclusion

Given the relatively good performance of our model we can conclude that using machine learning techniques to predict whether a system will soon be hit with malware could potentially be an excellent preventative measure in addition to existing security tools.

References

[1] Lévesque, F. L., Fernandez, J. M., Somayaji, A. (2014). Risk prediction of malware victimization based on user behavior. In 2014 9th international conference on malicious and unwanted software: The Americas (MALWARE) (pp. 128–134). IEEE.

[2] Chanhyun Kang, Noseong Park, B. Aditya Prakash, Edoardo Serra, V. S. Subrahmanian, Ensemble Models for Data-driven Prediction of Malware Infections, Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, February 22-25, 2016, San Francisco, California, USA.

[3] Microsoft, Microsoft Malware Prediction Data Description, Kaggle. Accessed on: Oct. 11, 2019. [Online]. Available: https://www.kaggle.com/c/microsoft-malware-prediction/data

[4] Chris Deotte, Malware Timestamps, Kaggle. Accessed on: Oct. 11, 2019. [Online]. Available: https://www.kaggle.com/cdeotte/malware-timestamps

[5] keitakurita, LightGBM and XGBoost Explained, mlexplained. Accessed on: Oct. 11, 2019. [Online]. Available: http://mlexplained.com/2018/01/05/lightgbm-and-xgboost-explained/ Section 3.2.

[6] Shahini, Maryam; Farhanian, Ramin; and Ellis, Marcus (2019) "Machine Learning to Predict the Likelihood of a Personal Computer to Be Infected with Malware," SMU Data Science Review: Vol. 2 : No. 2 , Article 9. Available: https://scholar.smu.edu/datasciencereview/vol2/iss2/9

[7] Andrew lukyanenko. (2019). Is this Malware? [EDA, FE and lgb][updated]. Retrieved 21 November, 2019, from https://www.kaggle.com/artgor/is-this-malware-eda-fe-and-lgb-updated

[8] Pranjal srivastava. (2017). Essentials of Deep Learning : Introduction to Long Short Term Memory. Retrieved 21 November, 2019, from https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/

IX. Contributions

A. Rachana Jayaram

- Data Cleaning
- Feature Engineering
- LightGBM Model
- Literature Survey and Report
- Code integration, documentation and Video Integration

B. Skanda

- Data Cleaning
- Feature Engineering
- LSTM and LSTM-CNN
- Writing about the problem statement, data cleaning and proposed solution
- explanation about problem statement, data cleaning in video

C. Mehul Thakral

- Exploratory Data Analysis
- Data Visualizaton
- Channel creation and Video upload
- Explaining EDA, experiments and their results in video
- Writing about EDA, experiments and their results in report