Predicting Future Malware Attacks on Cloud Systems using Machine Learning

CAP 5771: Data Mining

Sai Kalyan Tarun Tiruchirapally (ST22Q), Shivam Agnihotri (SA22BB), Sparsh Khare (SK22BO)

1. Abstract

Data mining and Machine learning have applications in a myriad of fields, one such fastest-growing field is cybersecurity. Often, we come across situations where our systems are vulnerable to attacks by malware. We all have antivirus and defenders installed on our systems, but have we imagined why we still have these threats on our systems and if we could detect if our system is vulnerable to this malware and take necessary action before any considerable damage is done to the system? As a part of this project, we aim to implement different machine learning techniques that could precisely detect if the system is prone to a malware attack. To achieve this goal, we have decided to train the dataset with various classification algorithms such as lgbm, xgboost, and various other methodologies and evaluate the optimal techniques to forecast the risks.

2. Introduction

Data security is always an issue in the modern world. We can never say our data is safe from attackers or no one will try to access our data without authorization. The attackers can use a plethora of means to attack our system. One of these ways is a Malware attack which is extensively discussed in our project. Our project deals with the prediction of these Malware Attacks on Cloud Systems using Machine Learning techniques, where we train machine learning models in such a way that they can counter the malware attack on our Cloud System. Our project's objective is to implement techniques and identify subtle patterns which can forecast the probability of attacks that can be used to devise prevention techniques. We used the real-world dataset accumulated by Microsoft with large volumes and features and tried to identify useful features and patterns and trained the pruned data on various classification techniques. Our further goal is to work on optimization algorithms considering the accuracy factor and adding more System features and advanced approaches to protect systems more effectively. The main agenda of this problem is to produce an ML (Machine Learning) technique that can be able to predict the probability of a machine being infected by malware. Nowadays, a substantial number of methods are being proposed to tackle the problems associated with malware attacks. GBM (Light Gradient Boosted machine) is the most popular method for overseeing this situation. Since the data in the current world is ridiculously huge therefore it is also difficult for us to work on this amount of data efficiently.

3. Literature Survey

We have been using antivirus software as a solution to detect malware. But the antivirus solutions use the signature to detect the malware. This means the malware must be known to the software. If the malware is not previously known, then it will not be detected. That calls for a new mechanism to detect the malware. The research suggests integrating signature and machine learning models for higher accuracy

As per previous research on malware prediction, using algorithms such as gradient-boosting decision trees, it is possible to predict malware with high accuracy even in the case of a large dataset that could be highly sparse. The research suggests that the LightGBM model gives faster results using less memory compared to the XGBoost algorithm. The reason for LGBM being fast is that it follows leaf-wise execution for training the dataset while the XGBoost algorithm follows the level-wise execution. LightGBM is more accurate and faster when it comes to working with large datasets.

Some research suggests that algorithms such as Random Forest and decision trees can also be used for malware prediction with good accuracy, but the random forest can take much more time in learning than the other 2 algorithms.

As per the experiments and implementation, it was seen that LightGBM has a very less false negative rate as compared to the other 2 models which makes it a better choice.

4. Methodology

4.1 Decision tree: The decision tree is basically an extension of a graph and is also based on the branching technique. It is an ideal way to represent an algorithm that comprises several conditional statements. It contains 3 nodes; it shows a different output for every input node. It is a clever way to represent conditional control-containing algorithms.

4.2 Random Forest: Random Forest is based on the prediction approach. It takes subsets of data and makes a prediction based on it. It constructs many decision trees at the training time and is ideal for regression and classification.

4.3 XGBOOST: Xgboost is an exceedingly popular technique used to solve both classification and regression problems. It is a supervised algorithm that uses sequentially built shallow decision trees to provide accurate results and a highly scalable training method that avoids overfitting.

4.4 LGBM: LGBM is short for light gradient-boosting machine. It is a community-distributed framework used extensively for machine learning that uses the concept of gradient boosting. It is based on decision tree algorithms and is highly efficient for Machine Learning tasks.

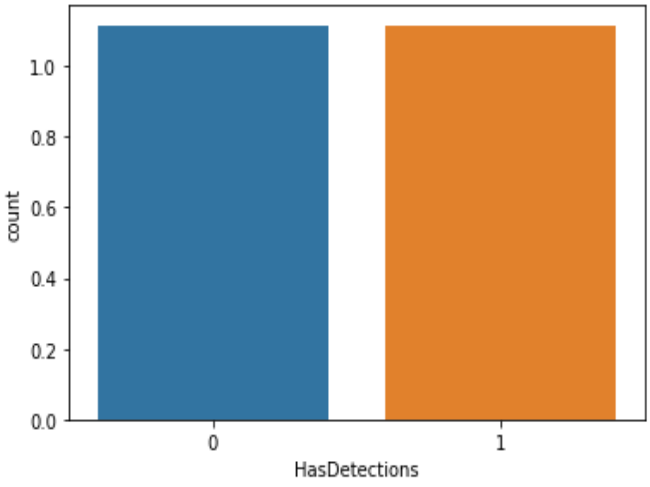
4.5 Neural network: A neural network can be visualized as a network of neurons that are organized in layers. The bottom layer is formed by the inputs and the top layers consist of the outputs. A neural network might also have intermediate layers. We have used the MLCP (Multi-layer perceptron classifier) in this project. MLCP classifier relies on a Neural network to do classification.

5. Dataset

5.1 Data Acquisition: Microsoft team was accumulating data for a long time and published it online for research. We used this dataset since it is detailed and contains a high volume. The dataset contains 4.04 GB of data which is about 83 columns with categorical, numerical, and binary features. The dataset is taken from the following source:

https://www.kaggle.com/c/microsoft-malware-prediction/data

We used the training data for testing as well by splitting beforehand since the testing data available is unlabeled.



5.2 Data Preprocessing: The dataset has 83 features, and all the data might not contain useful information. Along with increasing the computational load, using all the features to train the model might mislead the classifiers. Hence, we performed various pre-processing techniques like removing highly missing, sparse, skewed data and inputting the

missing values, and encoding categorical features. This will save computational load as well as enhance the model's performance. The data is balanced so we did not need to solve for class imbalance

6. Implementation

We divided our experiment into different modules at each step different processes are applied:

1. **Data Analysis**: Various steps are performed to understand the breadth and depth of the dataset.
2. **Preprocessing:** We performed various manipulations on the data so that I can be trained on the models.
3. **Training & testing:** We trained the models and predicted the evaluation metrics like AUC and ROC and accuracy scores.

These are the different step-by-step mechanisms that we applied in our experiment:

6.1 Test train data creation: We used the available train data and split it into a 70:30 ratio and used it for training and testing.

6.2 Class Imbalance: We checked for class imbalance and found out that the dataset is already balanced.

6.3 Outliers handling: An outlier is vastly different from the remaining values on the set. It is either vastly smaller or vastly larger than the given dataset. We have used the z score to filter out the outliers from the original dataset.

6.4 Missing Value: If no data is stored in an observation for a variable, then we can say that there is a missing value. Each column is checked for the type (numerical, categorical, binary) and in case there is a missing value present in that cell it is updated with an appropriate substitution depending on the column type.

6.5 Feature selection: while training the models we observed that some of the columns have highly skewed categories which are not a good fit for the training data. Hence, we removed such columns from the dataset.

6.6 Models: We had supervised learning data; we conducted some research of different supervised learning classification models which have their unique advantages and finalized on the following 5 classification models:

- Decision tree
- Random forest
- Xgboost
- ANN (Artificial Neural Network)
- LGBM
- Ensemble method: We tried to use a version of bagging where we combined the results of the aforementioned models and took a vote to determine the final binary class.

6.7 Challenges faced:

1. Data volume: The dataset was vast with 83 dimensions. So, we had to perform a lot of data visualizations to understand the intricacies of the data and recognize patterns. We also needed to perform a very extensive preprocessing to eliminate redundant and trivial features.

2. Computational load: Since the data is huge even after a detailed preprocessing performing each step took a significant load on the machines and the time incurred to run is also high.

6.8 Steps taken to handle challenges:

- We only loaded a ***subset*** of the data to train and test the models.
- We performed regressive ***preprocessing*** and engineered the data in an effective manner.
- We serialized and saved the model states and object states which incur a huge computational load into ***pickle*** files.

7. Evaluation Metrics

AUC Score

For this problem, model performance was evaluated based on the area under the ROC i.e., the score between the predicted probability and true label. The area under the ROC tells us how good a model is in distinguishing between true positives and false positives. We can use the curve to prove the overall efficiency of our model. If a model is highly efficient, then the predicted points and actual points should be the same. The overall strike rate of a model is calculated by comparing both these quantities and finding a percentage.

Accuracy

Accuracy will be defined as the instances that we classified correctly divided by the total number of instances. In short, the accuracy value tells us how many instances we classified correctly.

$$ACC = TP+TN /(TP+TN+FP+FN)$$

Precision

Precision is the ratio value of TP to TP+FP. In other words, this value tells us how many computers that we classified as malware attacks were actually attacked.

$$P= TP/(TP+FP)$$

Recall

A recall is defined as the ratio of the value of the no. of malware found to the total number of malwares in the set.

$$R= TP/(TP+FN)$$

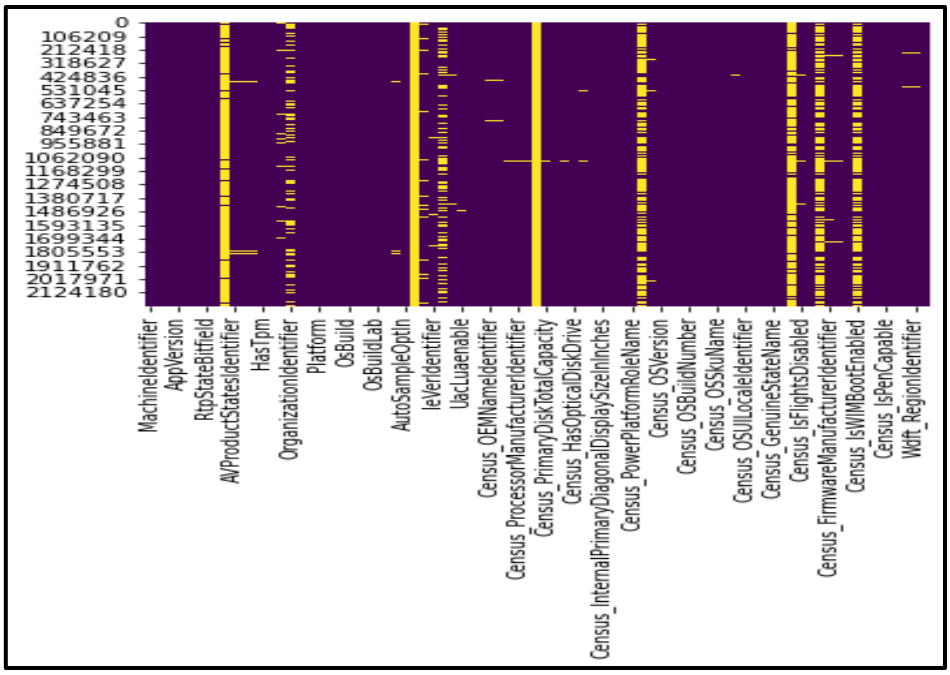
F1 Score

The F1 score is one of the most important results to be calculated. Its value depends on both recall and precision. In the formula, the F1 Score is equal to the weighted harmonic mean of the precision value and recall value.

$$F1= 2. P. R /(P+R) = 2.TP/(2.TP+FP+FN)$$

8. Experimental Results:

8.1 Visualizing missing values (null values) using a heat map (EDA):



8.2 Correlation with target variable (EDA):

AVProductStatesIdentifier	0.117074
IsProtected	0.057895
Census_TotalPhysicalRAM	0.055781
Wdft_IsGamer	0.054065
Census_ProcessorCoreCount	0.053625
RtpStateBitfield	0.041211

Census_IsTouchEnabled	-0.040603
AVProductsEnabled	-0.042131
Census_IsVirtualDevice	-0.051897
Census_IsAlwaysOnAlways ConnectedCapable	-0.063203
AVProductsInstalled	-0.149741
Census_IsWIMBootEnabled	NaN

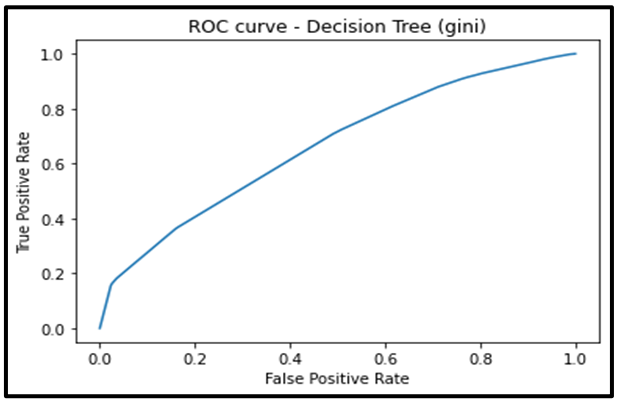
8.3 Decision Tree metrics:

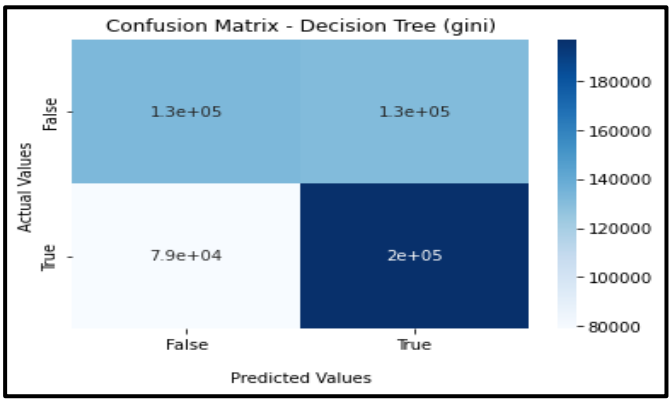
Gini:

	Precision	Recall	f-1 Score	support
0	0.63	0.50	0.56	264335
1	0.60	0.71	0.65	276139

accuracy			0.61	540474
macro avg	0.61	0.61	0.61	540474
weighted	0.61	0.61	0.61	540474

Under fitting and over fitting	Training set score	0.6117
	Test set score	0.6107
AUC score		0.6641
Model accuracy score		0.6107





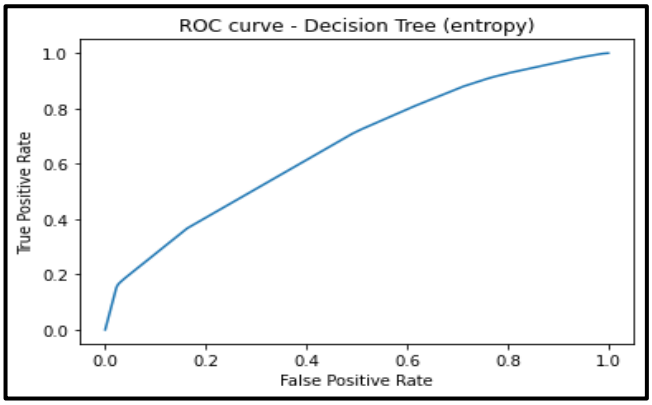
Entropy:

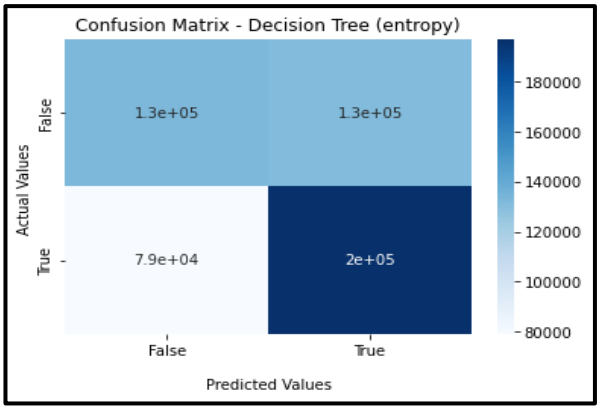
Gini:

	Precision	Recall	f-1 Score	support
0	0.63	0.50	0.56	264335
1	0.60	0.71	0.65	276139

accuracy			0.61	540474
macro avg	0.61	0.61	0.61	540474
weighted	0.61	0.61	0.61	540474

Underfitting and overfitting	Training set score	0.6117
	Test set score	0.6106
AUC score		0.6639
Model accuracy score		0.6106



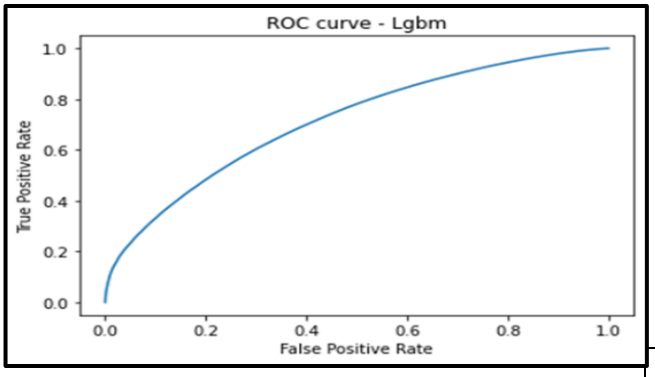


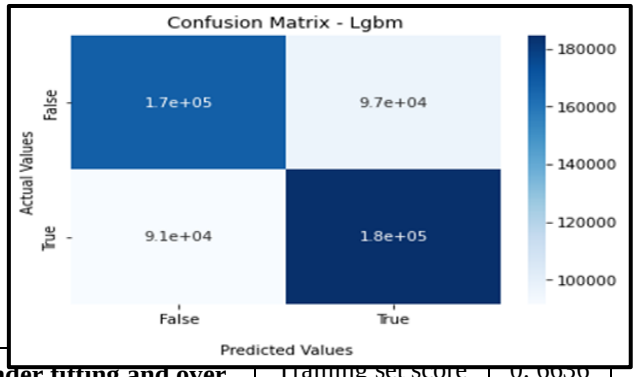
8.4 Lgbm metrics:

accuracy			0.66	540474
macro avg	0.66	0.66	0.66	540474
weighted	0.66	0.66	0.66	540474
macro avg	0.65	0.65	0.65	540474
weighted	0.65	0.65	0.65	540474

Under fitting and over fitting	Training set score	0.6534
	Test set score	0.6519
AUC score		0.7138
Model accuracy score		0.6519

	Precision	**Recall**	**f-1 Score**	**support**
0	0.65	0.63	0.64	264335
1	0.66	0.67	0.6	276139

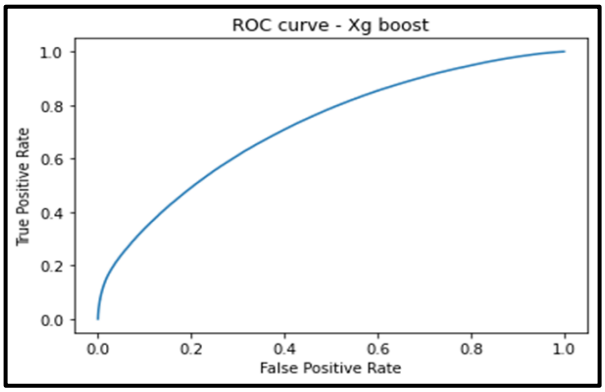


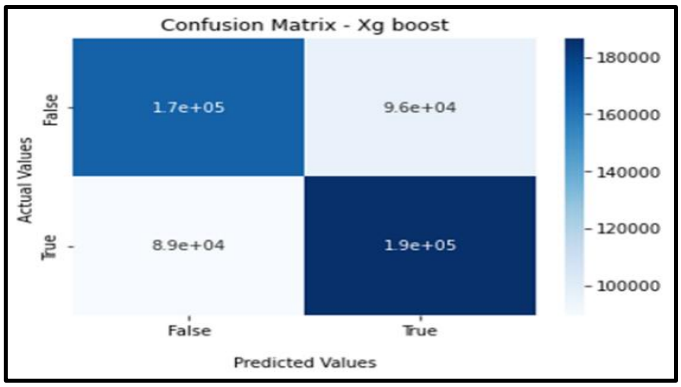


8.5 XG boost:

	Precision	**Recall**	**f-1 Score**	**support**
0	0.65	0.64	0.65	264335
1	0.66	0.68	0.67	276139

Under fitting and over fitting	Training set score	0. 6636
	Test set score	0. 6562
AUC score		0. 7198
Model accuracy score		0. 6562



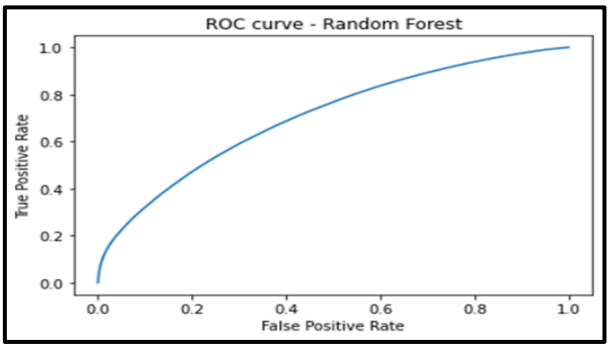


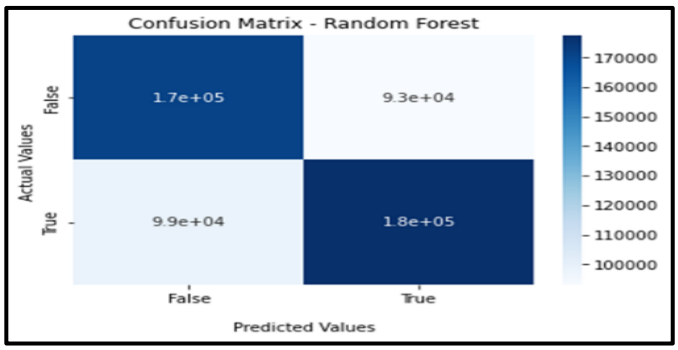
8.6 Random Forest:

	Precision	Recall	f-1 Score	support
0	0.63	0.65	0.64	264335
1	0.66	0.64	0.65	276139

accuracy			0.64	540474
macro avg	0.64	0.65	0.64	540474
weighted	0.65	0.64	0.64	540474

Under fitting and over fitting	Training set score	1.0000
	Test set score	0. 6449
AUC score		0.7046
Model accuracy score		0.6449





8.7 Neural networks:

Neural network model accuracy score: **0.5109**

8.8 Ensemble techniques:

Accuracy score of ensemble technique: **0.647**

9. Hardware used:

We had a huge dataset to train. We used both a local IDE called Jupyter notebook and Google Collab to run our algorithms. The model was implemented on the following 2 machines:

- Windows 10 - 16GB RAM, 8 vCPUs, intel i7 9th Generation processor.
- Mac – macOS Monterey, 8 GB RAM, Apple M1.

10. Conclusion

The dataset was trained on 5 models. It is very evident that LGBM gives a higher accuracy score of around 65% as compared to the other models and we can claim that LGBM is the best not only by the accuracy but also the AUC score which is a more suitable metric for our problem. However, an AUC score greater than 85% was not achieved due to the high amount of null data and less relevant features in the dataset. We also only used a subset of the data and due to computational capabilities, with more training on the data, the models might have performed more efficiently. We also tried to use the ensemble technique which uses voting as the criteria to classify the feature vector and it gave an accuracy of 65%. The LGBM model has an AUC score of 71% which is significantly better than that of the decision tree and slightly better than the random forest, Xgboost, and random forest. This makes LGBM a better choice for this problem.

11. Future Work:

- In future systems, the work on cloud systems is going to be quite extensive and the market growth of cloud computing is going to rise exponentially. Hence, this project can be proven as a project with many implementations in the future.
- Malware is used as one of the most common forms of attack. If the model can successfully predict a malware attack, then its overall efficiency increases phenomenally thereby making it highly efficient.
- In lightgbm, there is one parameter of a device where we had kept default i.e., CPU means CPU is used for training, this can be extended to GPU as future work to reduce the training time and if have sufficient computational power.
- As a part of future work, one can try another machine learning algorithm that uses gradient boosting on decision trees known as catboost and can evaluate the performance of the model.

12. References:

1. https://medium.com/swlh/microsoft-malware-prediction-using-classical-machine-learning-algorithms-5ade962ca73a
2. https://www.kaggle.com/c/microsoft-malware-prediction
3. https://ieeexplore.ieee.org/document/9358835
4. https://ieeexplore.ieee.org/document/9230624
5. https://ieeexplore.ieee.org/document/9058240
6. https://ieeexplore.ieee.org/document/9848045