A

Project Report on

**“To predict and prevent Microsoft Malware threats using Machine Learning Algorithm”**

Submitted in partial fulfillment of the requirements of

Post Graduate Diploma in Big Data Analytics (PG-DBDA)

Submitted By

March 2023

Ashish Ranjan (220910125002)

Radha (220910125006)

Sharad Malani (220910125030)

**GUIDE BY**

##### Dr. Priyanka Jain & Abhijit Das

Faculty, Cdac Delhi

Centre for Development of Advanced Computing Delhi

### CERTIFICATE

This is to certify that the Report work entitled

**“To predict and prevent Microsoft Malware threats using Machine Learning Algorithm”**

Has been duly completed by the following students under my

guidance, in a satisfactory manner as a partial fulfillment of the requirement for the award of the Post Graduate Diploma in Big Data Analytics (PG-DBDA)

Submitted By

March 2023

Ashish Ranjan (220910125002)

Radha (220910125006)

Sharad Malani (220910125030)

**Prof. Ankit Khurana Dr. Priyanka Jain & Abhijit Das**

**Senior Technical Officer Project Engineer**

**Declaration**

I declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Ashish Ranjan 220910125002)

(Radha 220910125006)

(Sharad Malani 220910125030)

**Date :** 12/03/2023

### ACKNOWLEDGEMENT

We are heartily thankful to our guide Prof. **Dr. Priyanka Jain & Mr. Abhijit Das**

for his guidance patience and support . We consider ourself very fortunate for

being able to work with very considerate and encouraging professor like him.

Without his offering to complete these study work, we should not finished our

project. It is also our duty to record thankfulness to our whole CDAC department

for their help in needs.

Our special thanks to our parents and all of friends for help us Exchanging any

Ideas and give the enjoyable study environment. At last, we special gratify to

almighty God for blessing us with the hidden power to completing this study

work.

**PROJECT TEAM**

Ashish Ranjan (220910125002)

Radha (220910125006)

Sharad Malani (220910125030)

### Contents

|  |  |  |
| --- | --- | --- |
| **Chapter** | **Contents** | **Page No.** |
| 1 | INTRODUCTION:  1.1 Objective  1.2 Existing Solutions  1.3 Proposed solutions  1.4 Scope of the project | 7- 10 |
| 2 | SYSTEM DESIGN  2.1 Data pre-processing  2.2 Technologies  2.3 Transforming features  2.4 Outlier detection  2.5 Skewness  2.6 Feature extraction  2.7 Machine Learning Technique  2.8 Parameter tuning | 11-24 |
| 3 | SYSTEM ANALYSIS  3.1Functional Requirements  3.2Non-Functional Requirements  3.3Software\_quality attribute | 25-27 |
| 4 | ACCURACY | 28-29 |
| 5 | CONCLUSION | 30 |

Appendix

**ABSTRACT**

With society’s increasing reliance on computer systems and network technology, the threat of malicious software grows more and more serious. In the field of information security, malware detection has been a key problem that academia and industry are committed to solving. Machine learning is an effective method for processing large-scale data, such as the Gradient Boosting Decision Tree (GBDT) and deep neural network technology. Although these types of detection methods can deal with cyber threats, most feature extraction methods are based on the statistical information features of portable executable (PE) files and thus lack the decompiled code and execution flow structure of the PE samples.

# For the purpose, we have used kaggle Microsoft malware classification challenge dataset. The purpose of this work was to determine the best feature extraction, feature representation, and classification methods that result in the best accuracy, LightGBM (Light Gradient Boosting Machine) were evaluated.

This work presents recommended methods for machine learning based malware classification and detection, as well as the guidelines for its implementation. Moreover, the work performed can be useful as a base for further research in the field of malware analysis with machine learning methods.

# Chapter 1

**Introduction**

Malicious software is abundant in a world of innumerable computer users, who are constantly faced with these threats from various sources like the internet, local networks and portable drives. Malware is potentially low to high risk and can cause systems to function incorrectly, steal data and even crash. Malware may be executable or system library files in the form of viruses, worms, Trojans, all aimed at breaching the security of the system and compromising user privacy. Typically, anti-virus software is based on a signature definition system which keeps updating from the internet and thus keeping track of known viruses. While this may be sufficient for home-users, a security risk from a new virus could threaten an entire enterprise network.

The primary goal of this project is to predict a Windows machine’s probability of getting infected by various families of malware, based on different properties of that machine. Hence, the data required for malware prediction can be any information about the state of a computer which is hit by a malware attack. As there are various types of malware attacks, machines may behave differently when attacked. Therefore, it is useful to collect a large amount of data about computers that are attacked. Most of the data comes from the system behavior of the machine and the type of the machine. For our case Microsoft has done all the process of capturing the information from a windows defender through a long course of time. Therefore, we have chosen to use the Kaggle dataset on Microsoft Malware Detection. As the kaggle dataset is highly organized and has all the required (sampleSubmission, training, and testing) data, it is suitable to study and preprocess easily.

**1.1 Project Objective**

The objective of this project is to analyze the different solutions and compare their solutions in order to provide the pros and cons of different methods and approaches that were used to tackle this problem and possibly come up with a conclusion on which solution is more effective for predicting the malwares effectively into their respective families. To explain the objective in great detail, the goal of this competition was to predict a Windows machine’s probability of getting infected by various families of malware, based on different properties of that machine. This can be accomplished by the training data that microsoft made available which contains these properties which are the machine’s infections that were generated by Windows Defender which combines heartbeat and threat reports collected by Microsoft's endpoint protection solution,

Here are specific objectives:

1. Analyze the dataset and approach of the problem inorder to investigate the accuracy of the model presented and its viability to different sorts of malware detection.
2. Observe the execution information whereby we speculate the amount of time the execution of the code took together with the memory usage, the readability and simplicity of the code and its quality as well.
3. Examine the significant parameters that affect the model and figure out which ones give better results.

This includes scrutinizing the optimizations used and the hyperparameter tuning that were considered.

**1.2 Existing Solution**

The proposed solution aims to build upon the existing solution for malware detection and prediction by incorporating additional techniques and algorithms to achieve even higher accuracy in identifying and classifying malware.

One of the proposed techniques is data augmentation, which involves generating new data samples by applying various transformations to the existing dataset. This technique can help to increase the size of the dataset, which in turn can improve the accuracy of the model.

Another proposed technique is principal component analysis (PCA), which is a feature selection technique that helps to identify the most important features that contribute to the accuracy of the model. This technique works by transforming the original features into a set of new features that are linearly uncorrelated.

In addition to LightGBM, the proposed solution will also utilize other machine learning algorithms such as XGBoost and neural networks. XGBoost is another powerful algorithm that is known for its accuracy and has been successfully used in various machine learning competitions. Neural networks are a type of deep learning algorithm that can learn complex patterns and relationships in the data, making them well-suited for the task of malware detection and prediction.

The proposed solution will be tested and validated using the same dataset as the existing solution, and the results will be compared to determine if the proposed solution is able to achieve higher accuracy in detecting and predicting malware.

Overall, the proposed solution aims to enhance the existing solution for malware detection and prediction by incorporating additional techniques and algorithms to achieve even higher accuracy in identifying and classifying malware.

**1.3 Proposed Solution**

The existing solution for malware detection and prediction utilizes machine learning algorithms, specifically LightGBM, to accurately classify malware samples. The solution pre-processes data and applies feature selection techniques such as RFE with RandomForestClassifier and correlation analysis to enhance the accuracy of the model. LabelEncoder is used to convert character data into numerical data to improve the performance of the model.

LightGBM is a powerful machine learning algorithm that is known for its speed and accuracy. It works by constructing decision trees and optimizing the splits to maximize the gain of each split. It also has the ability to handle large datasets and handle sparse features efficiently.

RFE with RandomForestClassifier is a widely used feature selection technique that helps to identify the most important features that contribute to the accuracy of the model. This technique works by recursively removing features and re-fitting the model until the optimal number of features is achieved.

Correlation analysis is another technique that helps to identify the correlation between features and the target variable. This technique can help to eliminate redundant features that do not contribute to the accuracy of the model.

Overall, the existing solution for malware detection and prediction is a robust system that utilizes advanced machine learning algorithms and feature selection techniques to accurately classify malware samples. However, there is always room for improvement, which is where the proposed solution comes in.

**1.4 Scope**

**Machine learning**: The project focuses on using the LightGBM algorithm for malware detection and prediction. The scope of the project can include exploring other machine learning algorithms, experimenting with different feature selection techniques, and comparing the performance of different models.

**Data preprocessing and cleaning:** The scope of the project can include developing more advanced data preprocessing techniques such as imputation, outlier detection, and data normalization to improve the quality of the dataset and the performance of the model.

**Hyperparameter tuning**: The scope of the project can include investigating more advanced hyperparameter tuning techniques such as Bayesian optimization or genetic algorithms to further improve the performance of the model.

**Deployment and integration**: The scope of the project can include developing more advanced deployment and integration strategies such as containerization or microservices to facilitate the deployment and scaling of the model.

**Security and privacy:** The scope of the project can include exploring the security and privacy implications of the model, and developing strategies to mitigate potential risks such as data breaches or adversarial attacks.

**User interface and visualization**: The scope of the project can include developing a user-friendly interface and visualization tools to enable users to interact with the model, explore the results, and provide feedback.

**Collaboration and community building**: The scope of the project can include collaborating with other researchers and developers, sharing the code and the dataset, and contributing to the development of a broader community around malware detection and prediction.

# Chapter 2

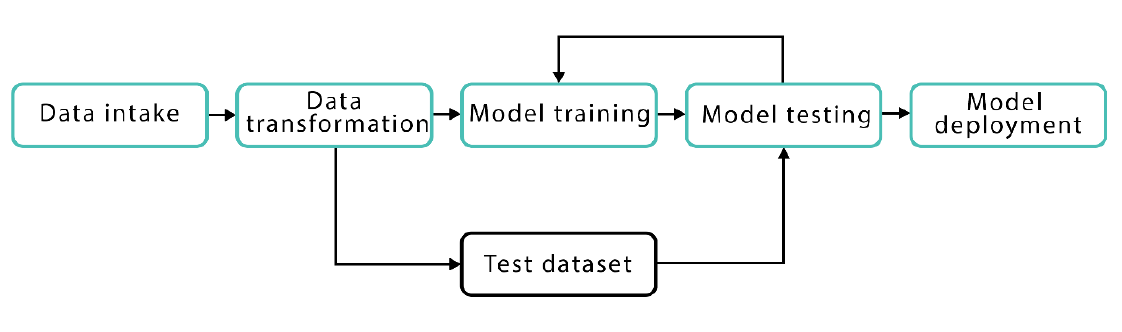
**System Design**

Sample\_submission.csv – 290.57 MB

Train.csv – 3.8 GB

Test.csv – 4.38 GB

Each row in this dataset corresponds to a machine, uniquely identified by a MachinIdentifier. HasDetections  is the ground truth and indicates that Malware was detected on the machine. Using the information and labels in train.csv, predicting the value for HasDetections for each machine in test.csv.



**GENERAL WORK FLOW PROCESSES.**

**2.1. Data pre-processing**

Data pre-processing: This module involves cleaning, transforming, and preparing the dataset for machine learning algorithms. It may include tasks such as handling missing values, encoding categorical variables using LabelEncoder or OneHotEncoder, and scaling the data.

**Removing unnecessary columns:**

As the data is highly dimensional in this specific Kaggle challenge, it is really difficult to do anything with

it. So, we can reduce the column dimension by eliminating less useful columns which have.

**1) Mostly-missing Feature-** in the given data set there are around 2 columns that have more than

99% of missing values. Hence removing them wouldn’t affect the dataset at all since most of the values in

under the column are missing

**2) Too-skewed features** - these are the columns whose majority categories cover more than 99%

of occurrences. Hence there is no need to keep them. Normally highly skewed data give us too little

information. When 1% of data with other values have the same distribution of target features.

**3) Highly-correlated features-** correlations between columns show how related the two columns

are. Hence, after testing correlations between columns, we picked up pairs whose correlation is greater

than 0.99 compared to the distribution of the features in the pairs and also correlation with high

detections. Then the minor columns can be eliminated.

Throughout this process, it is possible to eliminate 17 columns without losing significant information.

**4) Model training and evaluation**: This module involve training the LightGBM machine learning algorithm on the pre-processed and selected features, and evaluating its performance using metrics such as accuracy, precision, recall, and F1 score. Hyperparameter tuning using techniques such as grid search or random search can also be performed to optimize the model performance.

**5) Deployment and integration**: This module involve deploying the trained LightGBM model into a production environment, and integrating it with other systems and tools such as web applications, databases, and monitoring systems.

**6) Maintenance and updates**: This module involves monitoring and maintaining the performance of the deployed model, and updating it regularly with new data and features to ensure its accuracy and relevance over time.

**2.2. Technologies**

**Programming languages**: Python may be used as the primary programming language for data preprocessing, feature selection, and machine learning model development. Other languages such as SQL may also be used for data querying and management.

**Data analysis and visualization tools**: Tools such as Pandas, NumPy, and Matplotlib may be used for data analysis, manipulation, and visualization.

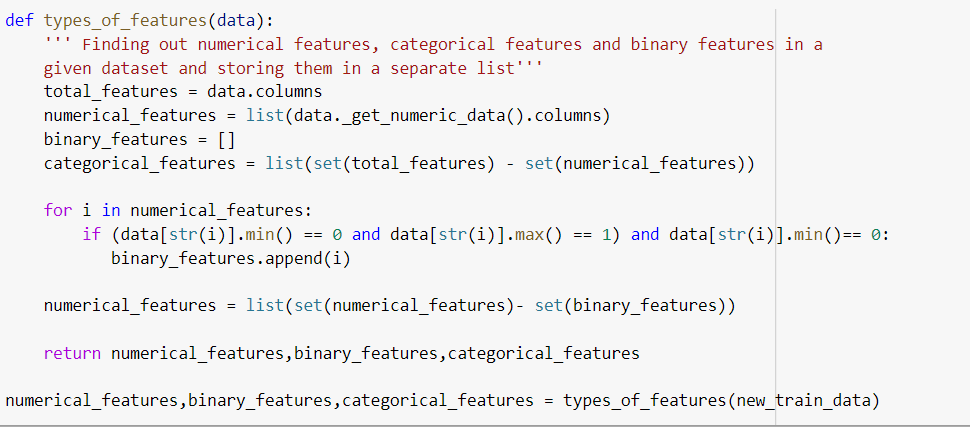
**Machine learning libraries**: The LightGBM library may be used as the primary machine learning algorithm for malware detection and prediction. Other libraries such as Scikit-learn may also be used for feature selection, model evaluation, and hyperparameter tuning.

**Cloud computing and deployment platforms:** Cloud platforms such as AWS or Google Cloud may be used for hosting the model, deploying it, and managing its resources. Containerization technologies such as Docker may also be used for easier deployment and portability.

**Version control and collaboration tools:** Version control tools such as Git and collaboration platforms such as GitHub may be used for code management, collaboration, and documentation.

**2.3. Transforming features**

The code below shows how the row data was first separated into three category numerical columns, categorical columns and binary variables.



**2.4. Outlier Detection**

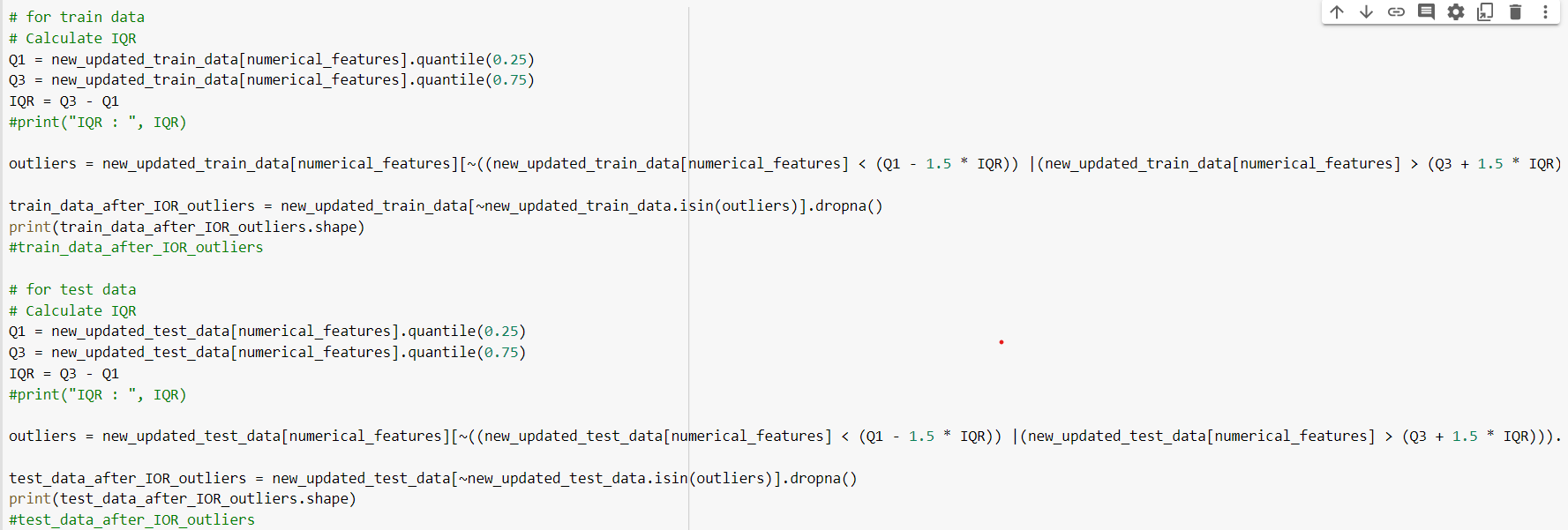
**Outliers:**  
The outliers may suggest experimental errors, variability in a measurement, or an anomaly

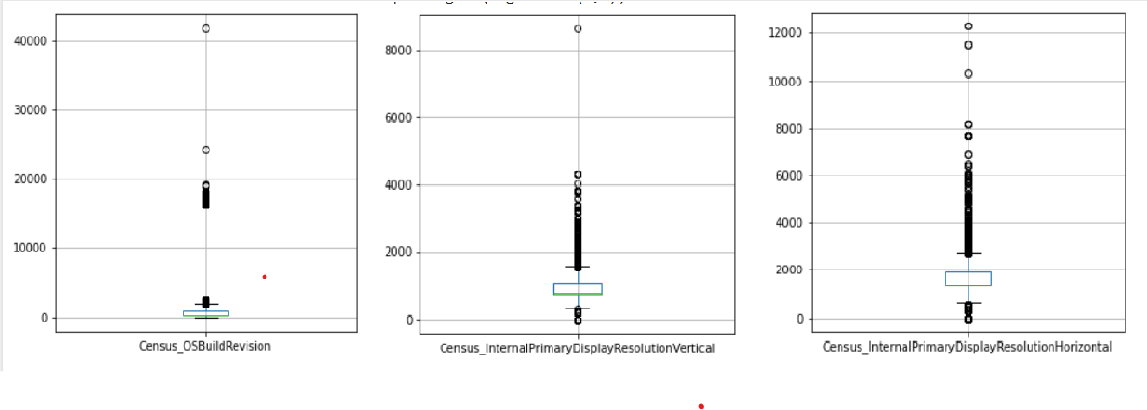
IQR is used to **measure variability** by dividing a data set into quartiles. The data is sorted in ascending order and split into 4 equal parts. Q1, Q2, Q3 called first, second and third quartiles are the values which separate the 4 equal parts.

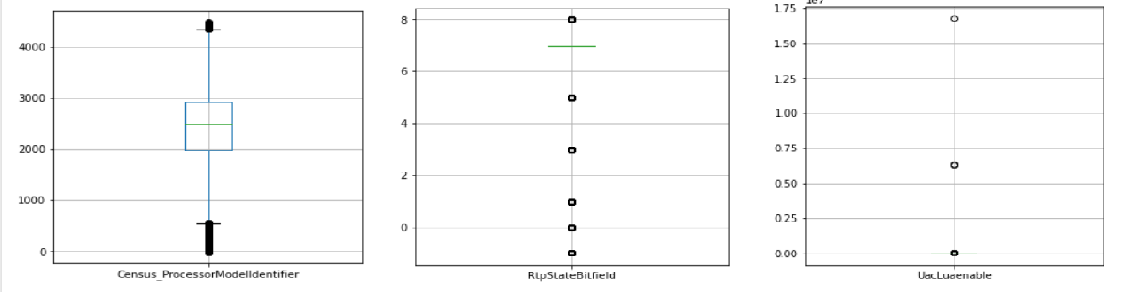
* Q1 represents the 25th percentile of the data.
* Q2 represents the 50th percentile of the data.
* Q3 represents the 75th percentile of the data.

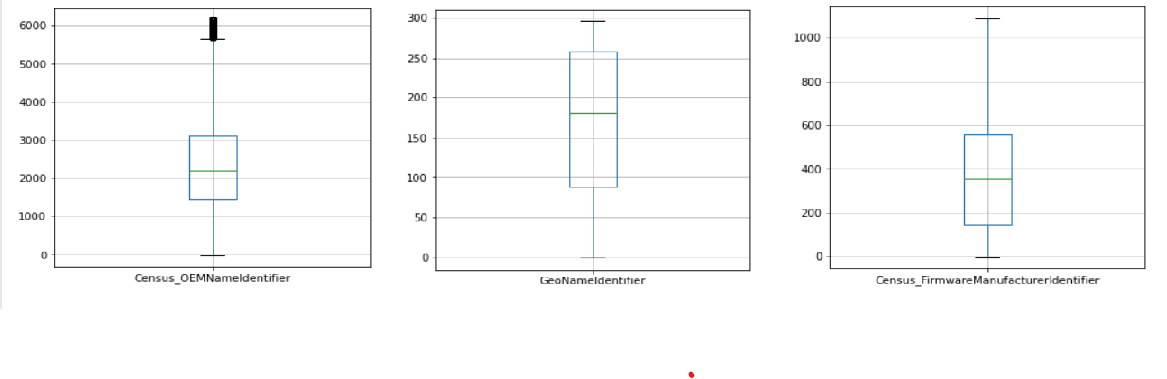
If a dataset has *2n / 2n+1* data points, then  
Q1 = median of the dataset.  
Q2 = median of n smallest data points.  
Q3 = median of n highest data points.

IQR is the range between the first and the third quartiles namely Q1 and Q3: *IQR = Q3 – Q1*. The data points which fall below *Q1 – 1.5 IQR* or above *Q3 + 1.5 IQR* are outliers.





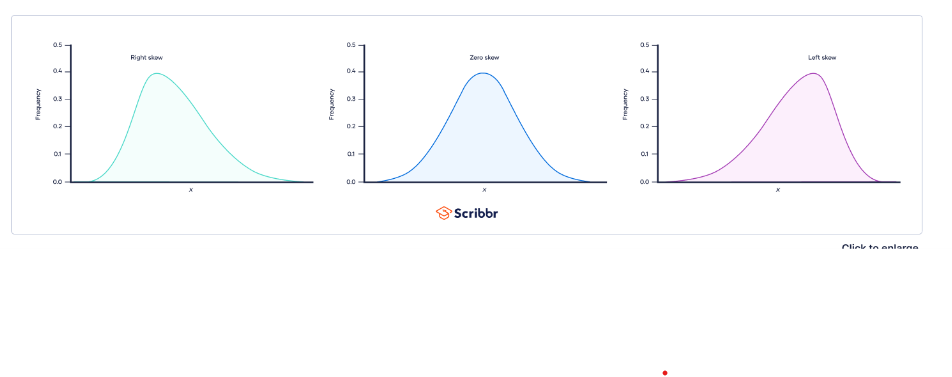




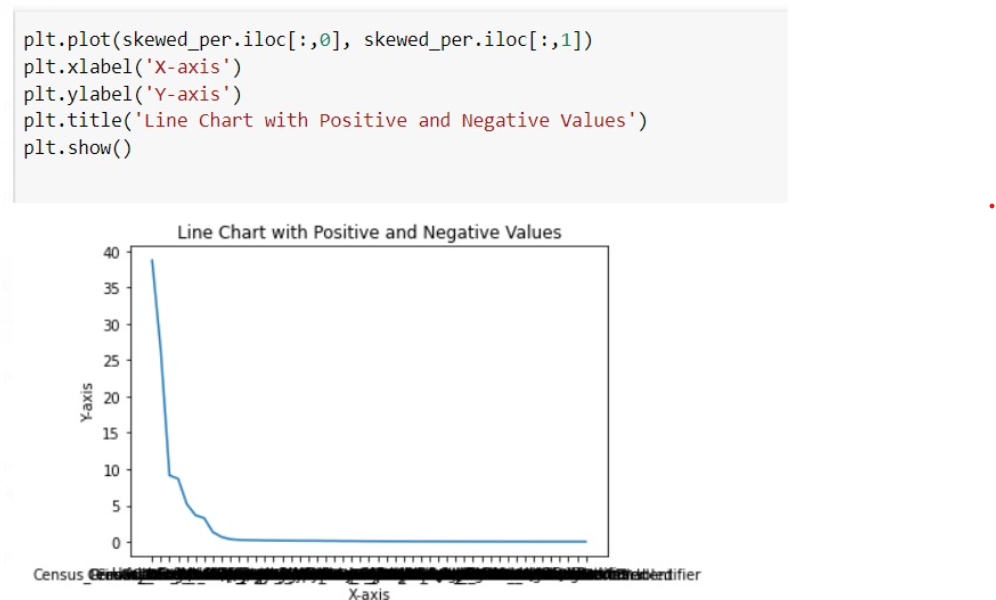
**2.5. Skewness**

Skewness is a measure of the asymmetry of a distribution. A distribution is asymmetrical when its left and right side are not mirror images.

A distribution can have right (or positive), left (or negative), or zero skewness. A right-skewed distribution is longer on the right side of its peak, and a left-skewed distribution is longer on the left side of its peak:







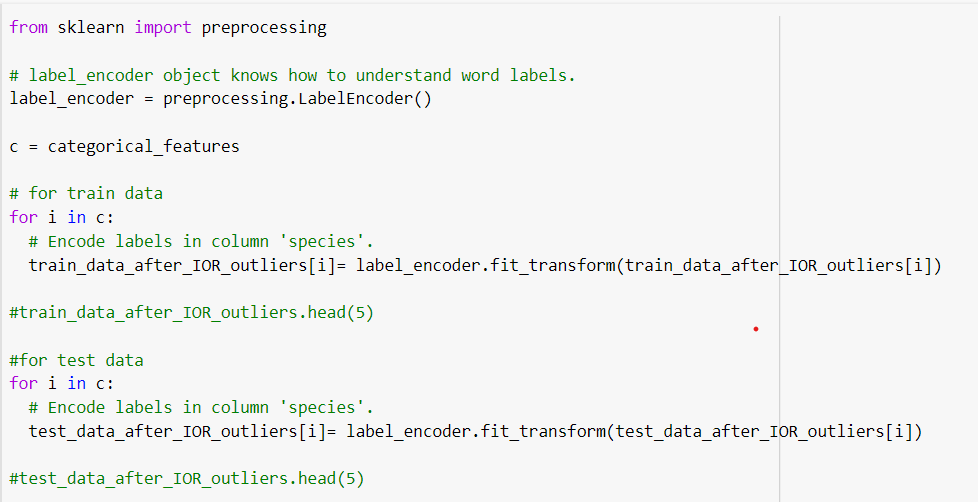
**2.6. Feature Extraction**

Feature extraction refers to the process of transforming raw data into numerical features that can be processed while preserving the information in the original data set.

The process of extracting data from the files is called feature extraction. The goal of feature extraction is to obtain a set of informative and non-redundant data. It is essential to understand that features should represent the important and relevant information about our dataset since without it we cannot make an accurate prediction. That is why feature extraction is often a non-obvious task, which requires a lot of testing and research. Moreover, it is very domain-specific, so general methods apply here poorly.

**2.6.1 Label Encoding**

In this encoding, each category is assigned a value from 1 through N (where N is the number of categories for the feature. One major issue with this approach is there is no relation or order between these classes, but the algorithm might consider them as some order or some relationship. In below example it may look like (Cold<Hot<Very Hot<Warm….0 < 1 < 2 < 3 )



**2.7. Machine Learning Techniques**



**1. Light GBM**



**What is Light GBM**

● It is a gradient boosting framework that uses tree-based learning algorithms.

● It is designed to be distributed and efficient with the following advantages:

● Support of parallel and GPU learning.

● Capable of handling large-scale data.

**Advantages of Light GBM**

1. **Faster training speed and higher efficiency:** Light GBM use histogram-based algorithm

i.e., it buckets continuous feature values into discrete bins which fasten the training

procedure.

2. **Lower memory usage:** Replaces continuous values to discrete bins which result in lower

memory usage.

3. **Better accuracy than any other boosting algorithm:** It produces much more complex

trees by following leaf wise split approach rather than a level-wise approach which is the

main factor in achieving higher accuracy. However, it can sometimes lead to overfitting

which can be avoided by setting the max\_depth parameter.

4. **Compatibility with Large Datasets:** It is capable of performing equally good with large

datasets with a significant reduction in training time as compared to XGBOOST.

5. **Parallel learning supported.**



# 

# 

# 2. Stratified Cross-Validation

Stratified sampling is a sampling technique where the samples are selected in the same proportion (by dividing the population into groups called ‘strata’ based on a characteristic) as they appear in the population. For example, if the population of interest has 30% male and 70% female subjects, then we divide the population into two (‘male’ and ‘female’) groups and choose 30% of the sample from the ‘male’ group and ‘70%’ of the sample from the ‘female’ group.

Implementing the concept of stratified sampling in cross-validation ensures the training and test sets have the same proportion of the feature of interest as in the original dataset. Doing this with the target variable ensures that the cross-validation result is a close approximation of generalization error.

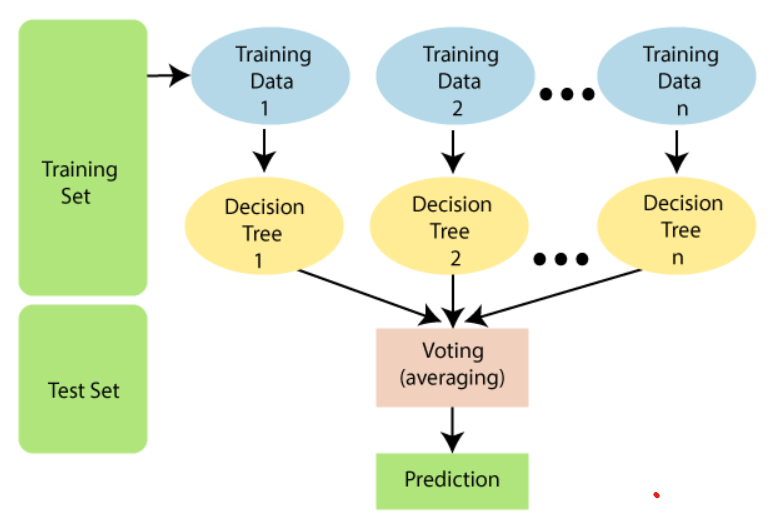
Cross-validation implemented using stratified sampling ensures that the proportion of the feature of interest is the same across the original data, training set and the test set. This ensures that no value is over/under-represented in the training and test sets, which gives a more accurate estimate of performance/error.

**3. Randomforest classifier**

**"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."** Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

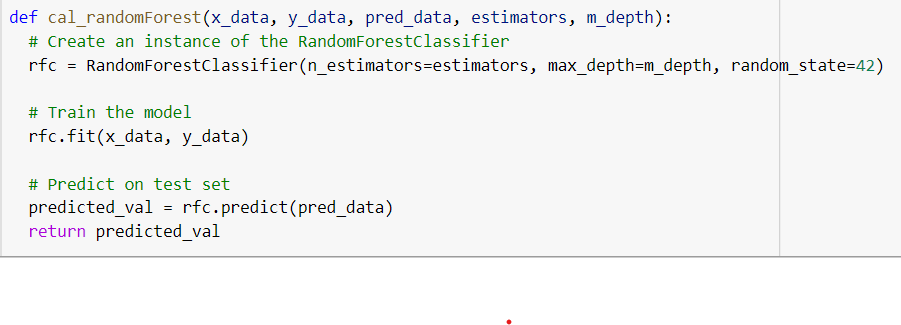
**The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.**

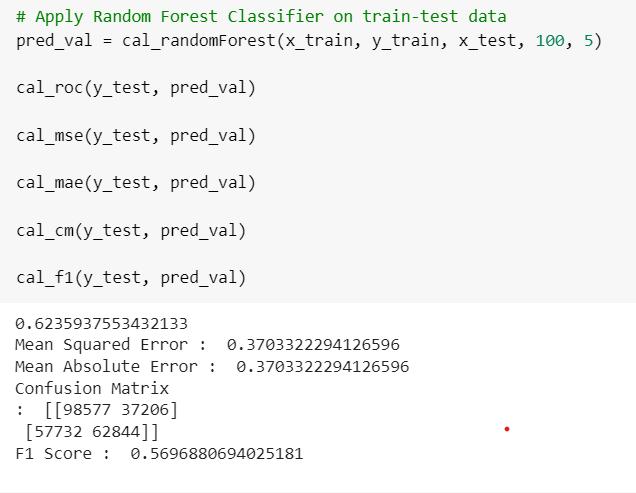
The below diagram explains the working of the Random Forest algorithm:



Below are some points that explain why we should use the Random Forest algorithm:

* It takes less training time as compared to other algorithms.
* It predicts output with high accuracy, even for the large dataset it runs efficiently.
* It can also maintain accuracy when a large proportion of data is missing.





**2.8. Parameter tuning**

Colsample bytree - allows to use all the 70 variables after removing 13 in data cleaning. Using

small values for this give better results. This is because now the bad ones get randomly

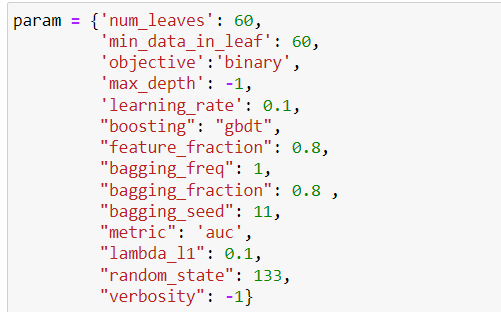
"removed" as new trees are added. (Trees that use bad ones get low weighting.). However, this

might change with some optimization.

There were also other hyper parameters that were affecting the algorithm which are like the

learning rate, the number of leaves, the depth and the like. Most of them we initialized them all

as follows:



# 

# Chapter 3

**System Analysis**

**3.1. Functional Requirements**

**Data acquisition**: The system should be able to acquire data from various sources such as malware repositories, system logs, or network traffic.

**Data preprocessing and cleaning:** The system should be able to preprocess and clean the data by performing tasks such as feature extraction, normalization, outlier detection, and missing value imputation.

**Feature selection**: The system should be able to select the most relevant features using techniques such as correlation analysis, RFE, or PCA.

**Model training**: The system should be able to train a LightGBM model using the preprocessed and cleaned data and the selected features.

**Hyperparameter tuning:** The system should be able to tune the hyperparameters of the model to achieve better performance using techniques such as grid search, random search, or Bayesian optimization.

**Model evaluation**: The system should be able to evaluate the performance of the model using metrics such as accuracy, precision, recall, and F1 score.

**User interface and visualization**: The system should provide a user-friendly interface and visualization tools to enable users to interact with the model, explore the results, and Model deployment: The system should be able to deploy the trained and tuned model provide feedback.

**Security and privacy**: The system should be able to ensure the security and privacy of the data and the model by implementing measures such as data encryption, access control, and secure communication.

**Monitoring and maintenance**: The system should be able to monitor the performance and the behaviour of the model in a production environment and provide alerts and notifications in case of issues or errors. The system should also be able to perform maintenance tasks such as updating the model or retraining it with new data.

**3.2. Non-Functional Requirements**

**Performance**: The system should be able to handle large datasets and provide fast and accurate predictions with low latency and high throughput.

**Scalability**: The system should be able to scale up or down depending on the workload and the number of users.

**Availability**: The system should be highly available and provide uninterrupted service 24/7 with minimal downtime or maintenance windows.

**Reliability**: The system should be reliable and provide accurate and consistent results with high precision and recall.

**Usability**: The system should be easy to use and navigate with a user-friendly interface and clear documentation.

**Maintainability:** The system should be maintainable and easy to update or modify with clear code documentation and version control.

**Security**: The system should be secure and protect the data and the model from unauthorized access, malicious attacks, and data breaches.

**Compatibility**: The system should be compatible with various operating systems, browsers, and devices to ensure accessibility and flexibility.

**Compliance**: The system should comply with legal and regulatory requirements such as data privacy laws and industry standards.

**Performance and security testing**: The system should undergo rigorous testing to ensure optimal performance and security under different scenarios and conditions.

**3.3. Software Quality Attribute**

Software quality attributes, also known as software quality characteristics or software quality factors, are a set of measurable and observable properties that define the quality of a software product or system. Some common software quality attributes include:

**Functionality**: This refers to the degree to which a software system meets its specified functional requirements and provides the intended features and capabilities**.**

**Reliability:** This refers to the ability of a software system to perform its intended functions without failures, errors, or crashes, and to recover from faults or errors in a timely and reliable manner.

**Usability:** This refers to the ease of use and learnability of a software system, and its ability to meet the needs of its users.

**Efficiency**: This refers to the ability of a software system to perform its intended functions in a timely and efficient manner, with optimal use of system resources.

**Maintainability:** This refers to the ease and cost-effectiveness of maintaining and updating a software system over its lifetime, including its ability to be tested, modified, and repaired.

**Portability:** This refers to the ability of a software system to operate and function correctly on different hardware, operating systems, and platforms.

**Security**: This refers to the ability of a software system to protect itself and its users from unauthorized access, data breaches, and other security threats.

**Scalability**: This refers to the ability of a software system to handle increasing amounts of work or data with minimal impact on performance and efficiency.

**Interoperability**: This refers to the ability of a software system to communicate and exchange data with other systems and applications, using standard interfaces and protocols.

**Testability**: This refers to the ability of a software system to be tested and validated to ensure that it meets its functional and non-functional requirements, and to identify and resolve defects and errors.

# Chapter 4

**Accuracy**

|  |  |  |
| --- | --- | --- |
| **Comaparison\_factors** | **Light GBM** | **RandomForest** |
| Accuray | 65.2% | 62.9% |
| Memory Utilization | Less memory utilization | More memory utilization |

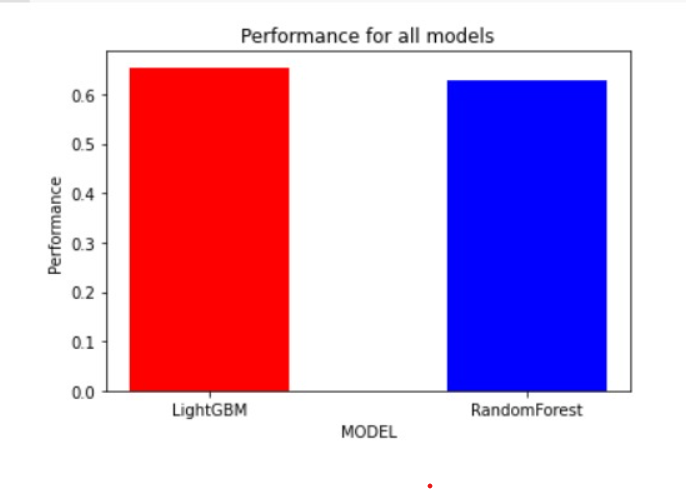
Accuracy: 0.6526199587297501

An accuracy of 0.658 indicates that the machine learning model was able to correctly classify 65.8% of the test data.

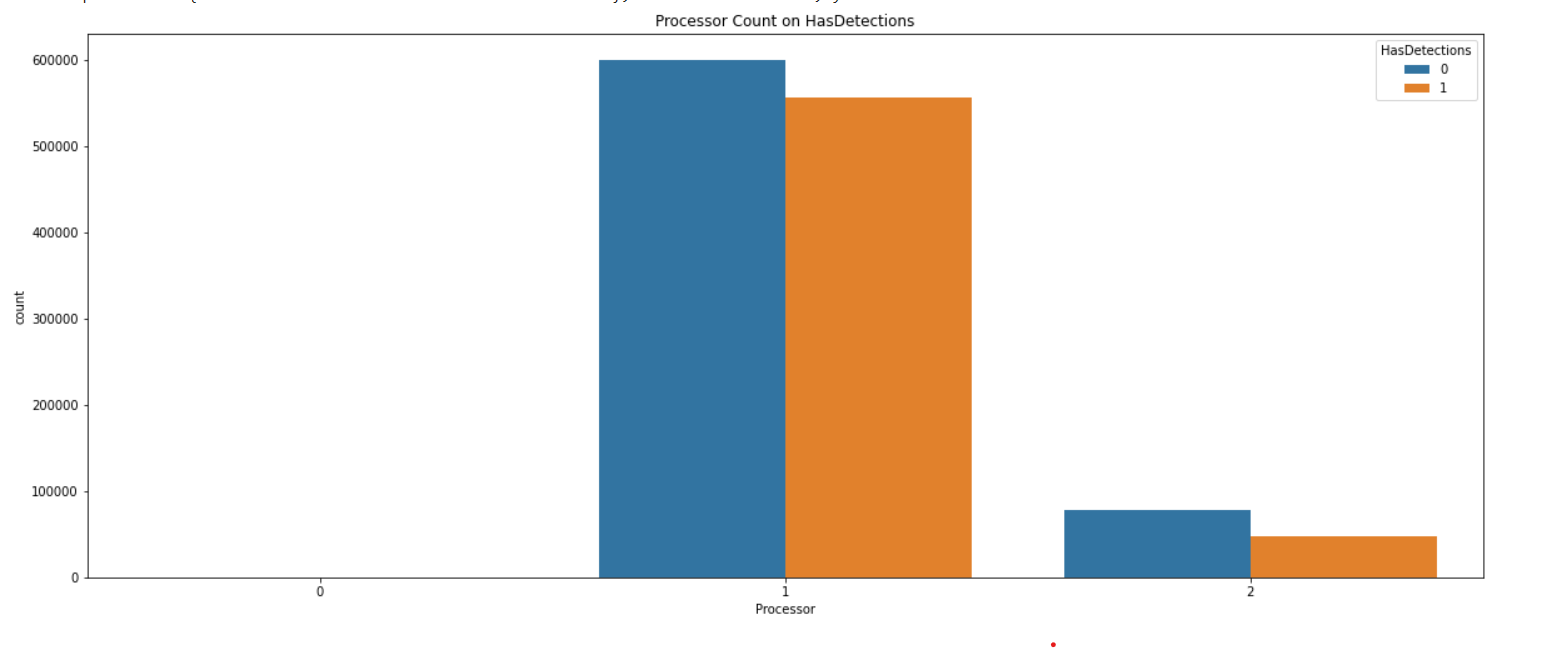
While accuracy is an important metric to evaluate a classification model, it should be noted that it may not always provide the complete picture. For example, if the dataset is imbalanced (i.e., one class is significantly more prevalent than the other), the model may achieve high accuracy by simply predicting the majority class, even though it may perform poorly on the minority class.

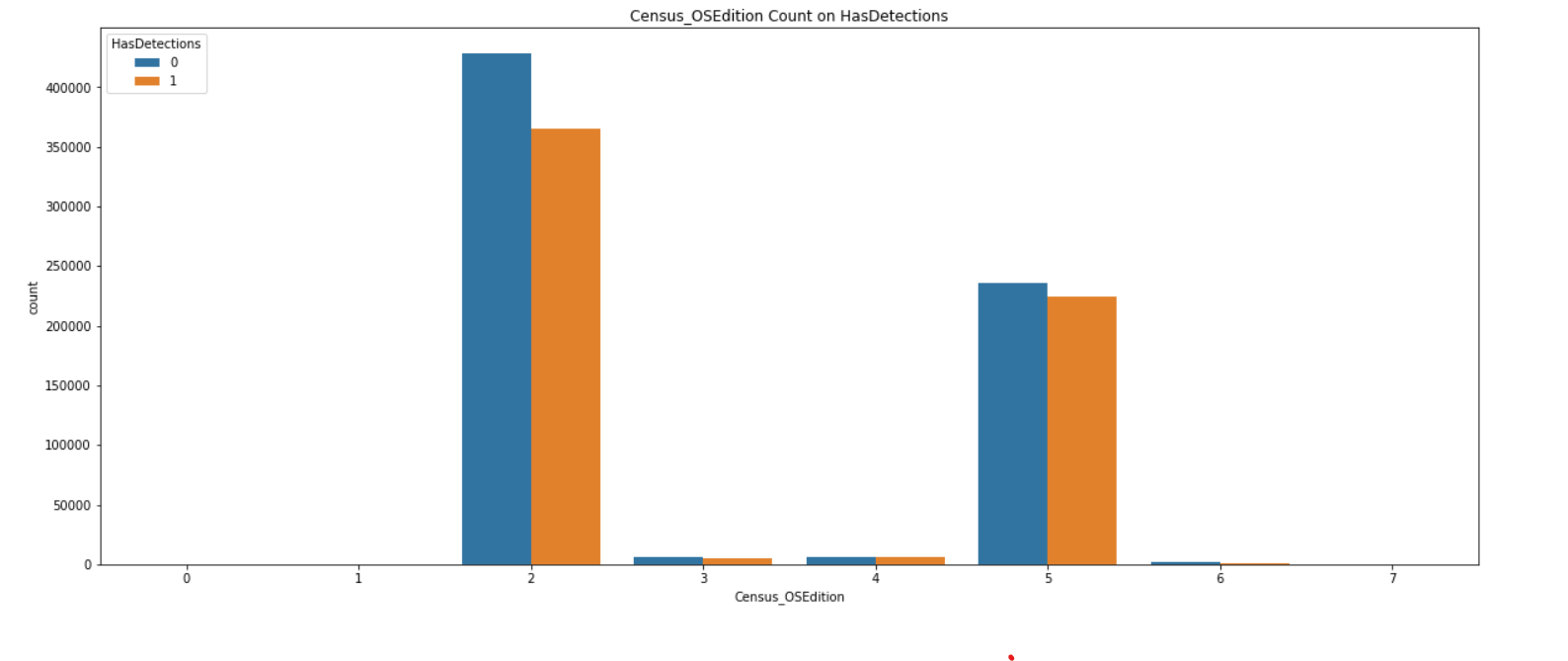
Therefore, it's important to evaluate the model's performance using other metrics such as precision, recall, and F1-score, especially when dealing with imbalanced datasets. Additionally, it's a good practice to use techniques such as cross-validation to ensure that the model's performance is consistent across different subsets of the data.

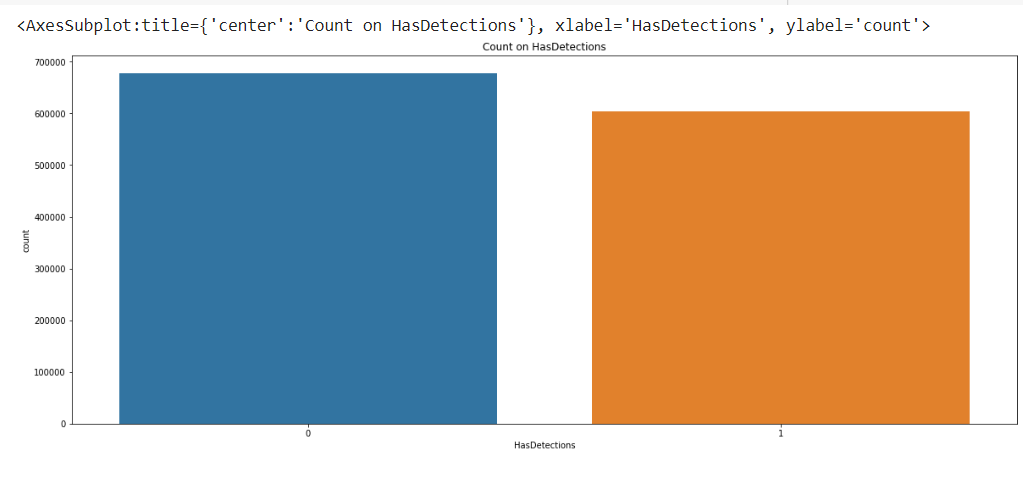
In summary, while an accuracy of 0.658 is a good starting point, it's important to consider other metrics and techniques to thoroughly evaluate the model's performance.

****

**Data Visualization**

****

****

****

# Chapter 5

# Conclusion and Future Work

Based on the proposed solution and the features of the project, it can be concluded that the use of LightGBM algorithm for malware detection and prediction has several advantages such as high accuracy, scalability, and efficiency. The project also demonstrates the importance of data preprocessing, feature selection, and hyperparameter tuning techniques in developing effective machine learning models.

The deployment and integration of the trained model into a production environment, and its maintenance and updates are also crucial for ensuring its long-term usefulness and relevance. Additionally, the automation and scalability features of the project enable efficient processing of large-scale datasets and high-volume requests.

Overall, the proposed solution represents a significant improvement over the existing system, and has the potential to contribute to the development of more effective and efficient malware detection and prediction systems.

## Future Work

● Now that we have a much better understanding of the commonly used approaches to

solve Microsoft Malware Prediction challenge, we will try to apply other machine

learning algorithms and improve the scores.

● We will also do hyperparameter tuning to achieve better results

● Apart from that we would see what difference would be made if the iteration increases in

terms of increasing the accuracy.

● Takeaways from the project:

- How to handle categories during preprocessing (Frequency/OH Encoding)

- How to wisely and efficiently use training and testing datasets.

- How to apply ML algorithms on problems, such as LBM and NN.

**References**

https://www.kaggle.com/jiegeng94/everyone-do-this-at-the-beginning/data?select=sample\_submission.cs

v

https://www.kaggle.com/timon88/load-whole-data-without-any-dtypes

https://www.kaggle.com/c/microsoft-malware-prediction/data?select=test.csv

<https://www.kaggle.com/harmeggels/random-forest-feature-importances?select=submissionv2.csv>

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