Machine learning for detection and classification of malware

MASTER OF TECHNOLOGY
IN
INFORMATION TECHNOLOGY

SCHOOL OF ENGINEERING
IN
TEZPUR UNIVERSITY

By

Gobinda Malakar
Roll No:CSI22006

Under the Supervision of

Dr. Shobhanjana Kalita
Assistant Professor
Department of Computer Science & Engineering
Tezpur University

Department of Computer Science and Engineering
TEZPUR UNIVERSITY

DECLARATION

I, Gobinda Malakar, hereby declare that this dissertation entitled “Machine learning for detection and classification of malware” is submitted to the Department of Computer Science & Engineering, Tezpur University, Tezpur, Assam, India.

(Gobinda Malakar)
Department of CSE
Roll No. CSI22006

Department of Computer Science and Engineering
TEZPUR UNIVERSITY

Certificate

This is to certify that the dissertation entitled "“Machine learning for detection and classification of malware " , submitted to the Department of Computer Science and Engineering, Tezpur University, in partial fulfillment for the award of the degree of Master of Technology in Computer Science and Engineering, is a record of Bonafede work carried out by Gobinda Malakar, Roll No. CSI22006 under my supervision and guidance. All help received by her from various sources have been duly acknowledged .

Dr. Shobhanjana Kalita
Assistant Professor
Department of CSE
Tezpur University

Department of Computer Science and Engineering
TEZPUR UNIVERSITY
Certificate

This is to certify that the dissertation entitled ““Machine learning for detection and classification of malware ” , submitted to the Department of Computer Science and Engineering, Tezpur University, in partial fulfillment for the award of the degree of Master of Technology in Computer Science and Engineering, is a record of Bonafede work carried out by Gobinda Malakar, Roll No. CSI22006. He has worked under the supervision of Dr. Shobhanjana Kalita.

Dr. Sarat Saharia
Professor & HOD
Department of CSE
Tezpur University

Contents

1. *Introduction*
2. *Problem Definition*
3. *Motivation*
4. *Literature review*
5. *Proposed Plan*
6. *Malware analysis*

   6.1 Static analysis

   6.2 Dynamic analysis
7. *Background Study*

   7.1 Static analysis

   7.2 Dynamic analysis

   7.3 Pearson correlation

   7.4 MiniMaxScaler

   7.5 KNN algorithm
8. *Implementation and results*
9. *Conclusion*
10. *References*

Abstract

Malware detection is a critical task in ensuring the security of computer systems, and machine learning techniques have proven to be effective in this domain. This report presents a study and implementation on detecting malware using machine learning techniques, focusing on feature selection and an exploration of malware analysis. The objective of this report was to develop an effective approach for identifying and classifying malware instances based on selected features and leveraging the KNN algorithm. The research began by collecting a dataset comprising both benign files and malware samples. The dataset was preprocessed to ensure data quality and suitability for machine learning analysis. Feature selection techniques were applied to identify the most relevant attributes that contribute to distinguishing malware from benign files. The chosen features encompassed a combination of behavioral, static, and dynamic characteristics commonly associated with malware. Overall, this report contributes to the field of malware detection by showcasing the effectiveness of feature selection and the KNN algorithm. It emphasizes the significance of studying malware analysis to inform feature selection and enhance the performance of machine learning models for malware detection.

1 Introduction

The fast advancement of technology has created both enormous benefits and major challenges. Because of the interconnection of devices, networks, and systems, there is always the potential of malware, which is malicious software designed to compromise, disrupt, or obtain unauthorized access to computer systems. Malware manifests in various forms and employs diverse techniques to infect and propagate across systems. It can be distributed through infected email attachments, malicious websites, compromised software, or drive-by downloads. Once inside a system, malware may execute unauthorized actions, such as modifying files, stealing sensitive information, launching denial-of-service attacks, or creating backdoors for remote control by attackers. Malware features encompass a wide range of indicators that help in identifying and classifying malicious software. These features can be broadly categorized into static and dynamic attributes. Static features include properties that can be determined without executing the malware. Dynamic features, on the other hand, capture the behavior and activities of malware during execution.

Traditional signature-based detection methods are becoming less successful as malware sophistication and complexity increase. As a result, using machine learning approaches to address this threat has emerged as a promising solution. Machine learning has emerged as a powerful tool in the field of malware detection, offering the potential to effectively identify and classify malware based on patterns and characteristics learned from vast amounts of data. By leveraging the capabilities of machine learning algorithms, security experts and researchers can develop intelligent systems that adapt to new threats and enhance the overall cybersecurity landscape. The involvement of machine learning in malware detection brings several advantages. Firstly, machine learning algorithms can process large datasets efficiently, allowing for comprehensive analysis of features and behaviors that distinguish malware. Secondly, these techniques can adapt and learn from new samples, enabling the system to stay up-to-date with emerging malware strains. Thirdly, machine learning can capture complex relationships and non-linear patterns, enhancing the detection accuracy compared to traditional rule-based or signature-based methods. This report contributes the specific application of feature selection, the KNN algorithm, and the study of malware analysis to develop comprehensive and effective malware detection models.

2 Problem Definition

We have been given a Malware dataset where we have to perform some feature selection techniques to extract the proper features of the malware dataset, classify the dataset and apply any algorithm of machine learning to detect the malware efficiently.

3 Motivation

The motivation behind conducting this work on detection and classification of malware using machine learning techniques and malware analysis stems from the pressing need to combat the ever-growing threat of malicious software. Malware poses significant risks to individuals, organizations, and the entire digital ecosystem, causing financial losses, data breaches, and compromising system integrity. Traditional methods of malware detection often fall short in effectively identifying and mitigating new and evolving malware variants. Machine learning offers a promising solution to enhance the detection and classification of malware. By leveraging its ability to learn from data, machine learning techniques can capture intricate patterns, adapt to new threats, and provide proactive defense mechanisms. Through this research, we aim to harness the potential of machine learning to develop robust and efficient malware detection.

The overall motivation of this study is to contribute to the field of malware detection by exploring the potential of machine learning, feature selection, and malware analysis. By developing effective malware detection, we aim to improve cybersecurity measures, protect individuals and organizations from malicious attacks, and mitigate the financial and operational risks associated with malware infections.

4 Literature Review

Numerous techniques and algorithms were put forward in the field of malware detection and classification. Some of them are discussed below.
Shabtai et al. [1] provide a taxonomy for malware detection using machine learning algorithms by reporting some feature types and feature selection techniques used in the literature. They mainly focus on the feature selection techniques (Gain ratio, Fisher score, document frequency, and hierarchical feature selection) and classification algorithms (Artificial Neural Networks, Bayesian Networks, Naïve Bayes, K-Nearest Neighbor, etc). In addition, they review how ensemble algorithms can be used to combine a set of classifiers. Bazrafshan et al. [2] identify three main methods for detecting malicious software: (1) signature based methods, (2) heuristic-based methods and behavior-based methods. In addition, they investigate some features for malware detection and discuss concealment techniques used by malware to evade detection. Nonetheless, the aforementioned research does not consider either dynamic or hybrid approaches.
Daniel Gibert ⁎, Carles Mateu, Jordi Planes [3] Conducting a detailed review on the rise of machine learning for detection and classification of malware involves exploring the research developments, trends, and challenges associated with the application of machine learning in the detection and classification of malware. The review aims to provide a comprehensive understanding of the advancements in this domain and shed light on the key challenges that researchers and practitioners face. It provides a detailed description of the methods and features in a traditional machine learning workflow, from the feature extraction, selection and reduction steps to classification. The traditional approaches are classified into three main categories: (1) static based and (2) dynamic-based approaches and (3) hybrid approaches. On the one hand, static-based approaches extract features derived from a piece of program without involving its execution. On the other hand, dynamic-based approaches include those approaches that extract features from the execution of malware during runtime. Lastly, hybrid approaches are those that combine static and dynamic analysis to extract features. Akshit Kamboj, Priyanshu Kumar, Amit Kumar Bairwa ⇑ , Sandeep Joshi [4] they used a dataset for training several different supervised and unsupervised machine learning models. Some of the main supervised machine learning models that were used include Logistic Regression, Decision Tree, and Random Forest Model. On the other hand,

Principal Component Analysis and K-Means clustering are some of the unsupervised machine learning models that were used for this project. Several dataset balancing operations were also performed on the obtained dataset. Some of these techniques include Oversampling, Undersampling, SMOTE, and Balanced Bagging Classifier. Out of all these models, the Random Forest Model was found to be the most accurate. Its accuracy went as high as 99.99% for the test dataset.

5 Proposed Plan

First, the dataset needs to be preprocessed by removing any missing values and irrelevant columns. Then classify the dataset to identify the bening and the malware files. Next, the data needs to be transformed to a numeric format suitable for machine learning algorithms. Once the data is in the correct format, feature selection techniques can be applied to identify the most relevant features for malware detection. Finally, one machine learning algorithm can be applied to the selected features to classify the malware samples accurately. This solution will help in identifying and detecting malware by extracting essential features that can distinguish malicious and benign programs.

6 Malware Analysis

1. Static Analysis

Static analysis of malware refers to the examination and analysis of malicious software without executing or running it. It involves inspecting the characteristics and properties of malware at rest, such as its file structure, code snippets, metadata, and other static attributes. Static analysis is a fundamental technique used in cybersecurity to understand the behavior, functionality, and potential risks associated with malware samples.

Static analysis features

Static analysis features are a set of characteristics extracted from the code or binary file of a program, which can be used to detect malware using machine learning techniques. Some common static analysis features used for malware detection include:

1. Opcode Frequencies: The frequency of use of different CPU instructions or opcodes in the code of a program can be a useful feature for detecting malware.

2. API Calls: The types and frequency of API calls made by a program can also provide valuable information about its behavior and potential malicious intent.

3. Control Flow Graphs: The control flow graph represents the program's structure and sequence of operations. Malware may have different control flow structures than benign software.

4. String Analysis: The analysis of strings within the code can reveal information about the behavior of the program, such as the use of encryption, obfuscation, or command-and-control (C2) servers.

5. Metadata: Information such as file size, creation date, and file type can provide additional clues about the potential malicious intent of the program.

2. Dynamic Analysis

Dynamic analysis of malware refers to the examination and analysis of malicious software by executing or running it in a controlled environment. Unlike static analysis, which focuses on inspecting the characteristics of malware at rest, dynamic analysis involves observing the behavior and actions of malware during runtime. This technique provides valuable insights into the actual execution flow, runtime interactions, network communications, and potential malicious activities performed by the malware.

Dynamic analysis features

Dynamic analysis features are a set of characteristics extracted from the behavior of software while it is executing.Some common dynamic analysis features used for malware detection include:

1. System Calls: The types and frequency of system calls made by a program can provide valuable information about its behavior and potential malicious intent.

2. Network Traffic: Analysis of the network traffic generated by a program can reveal information about its communication with external servers or devices, which can be indicative of malicious activity.

3. File Activity: Monitoring the creation, modification, or deletion of files by a program can provide clues about its behavior and potential malicious intent.

4. Registry Activity: The analysis of changes to the Windows Registry, which contains important system configuration information, can provide additional information about the behavior of a program.

5. **Process Behavior:** Observing the creation, termination, and behavior of processes spawned by a program can provide valuable information about its activity and potential malicious intent.

7 Background Study

7.1 Malware

Malware, short for malicious software, refers to any software or program specifically designed to harm, exploit, or compromise computer systems, networks, or devices without the knowledge or consent of the user. Malware is created with malicious intent and can cause various types of harm, such as data theft, unauthorized access, system disruption, financial loss, or privacy violations.

There are numerous types and categories of malware, each with its own characteristics and objectives. Some common types of malware include:

1. Viruses
2. Worms
3. Trojans
4. Ransomware
5. Spyware
6. Adware

7.2 Pearson correlation

Pearson correlation, also known as Pearson's correlation coefficient, is a statistical measure that quantifies the linear relationship between two continuous variables. It is named after Karl Pearson, who introduced this measure in statistics. The Pearson correlation coefficient is denoted by the symbol "r" and takes values between -1 and +1.

The formula for calculating the Pearson correlation coefficient (r) between two variables, X and Y, can be expressed as follows:

$r = (\Sigma((X - \mu X) * (Y - \mu Y))) / (\text{sqrt}(\Sigma(X - \mu X)^2) * \text{sqrt}(\Sigma(Y - \mu Y)^2))$

where:

Σ represents the summation symbol, indicating that the following operation is performed for each data point.
X and Y are the variables being analyzed.
μX and μY are the means (averages) of X and Y, respectively.
sqrt denotes the square root function.

7.2 MinMaxScaler

MinMaxScaler is a data preprocessing technique commonly used in machine learning to scale numerical features within a specific range. It is specifically designed to transform the feature values into a given range, typically between 0 and 1.

The MinMaxScaler formula is as follows:

$X\_scaled = (X - X\_min) / (X\_max - X\_min)$

where:

X is the original feature value.
X_min is the minimum value of the feature in the dataset.
X_max is the maximum value of the feature in the dataset.
X_scaled is the scaled value of X.

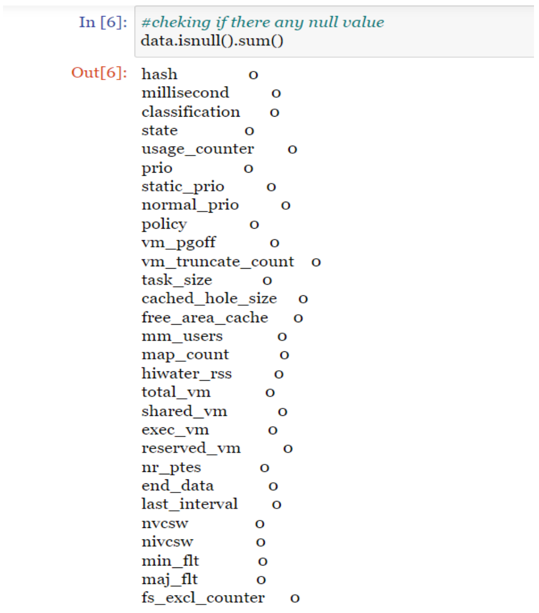
7.3 KNN algorithm(K-nearest neighbors)

The k-nearest neighbors (KNN) algorithm is a simple yet powerful machine learning algorithm used for both classification and regression tasks. It is a non-parametric method that makes predictions based on the similarity between the input data and its k nearest neighbors.

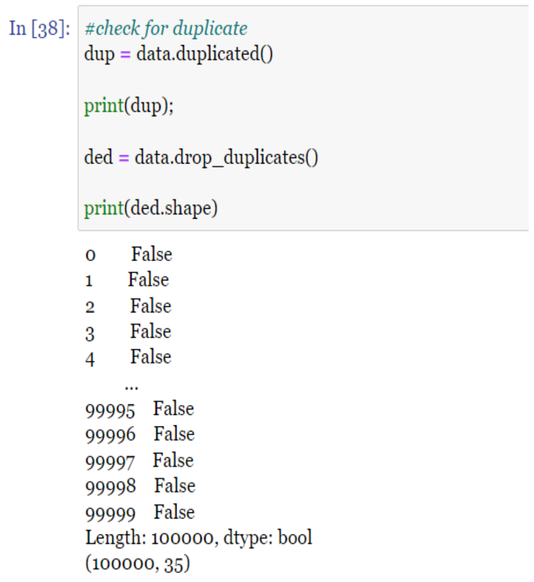
8 Implementation and Results

There are several implementation steps that are discussed below:

1. First, Perform data cleaning to address any missing values, or any null values and duplicate values in the dataset.

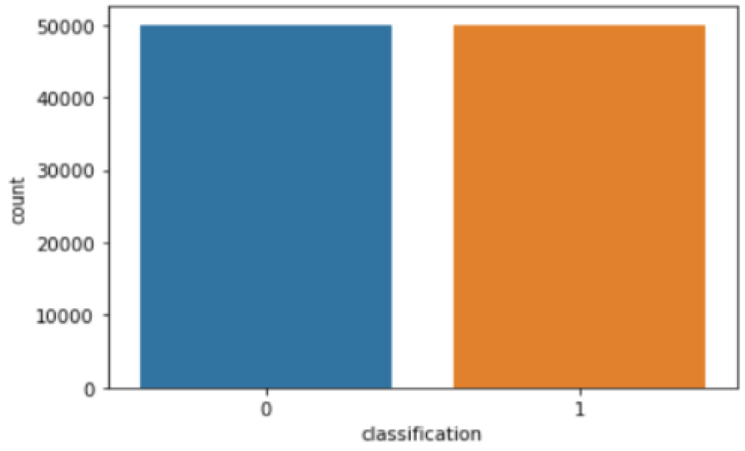


```
In [6]:  #cheking if there any null value
         data.isnull().sum()

Out[6]:  hash               0
         millisecond        0
         classification     0
         state              0
         usage_counter      0
         prio               0
         static_prio        0
         normal_prio        0
         policy             0
         vm_pgoff           0
         vm_truncate_count  0
         task_size          0
         cached_hole_size   0
         free_area_cache    0
         mm_users           0
         map_count          0
         hiwater_rss        0
         total_vm           0
         shared_vm          0
         exec_vm            0
         reserved_vm        0
         nr_ptes            0
         end_data           0
         last_interval      0
         nvcsw              0
         nivcsw             0
         min_flt            0
         maj_flt            0
         fs_excl_counter    0
```

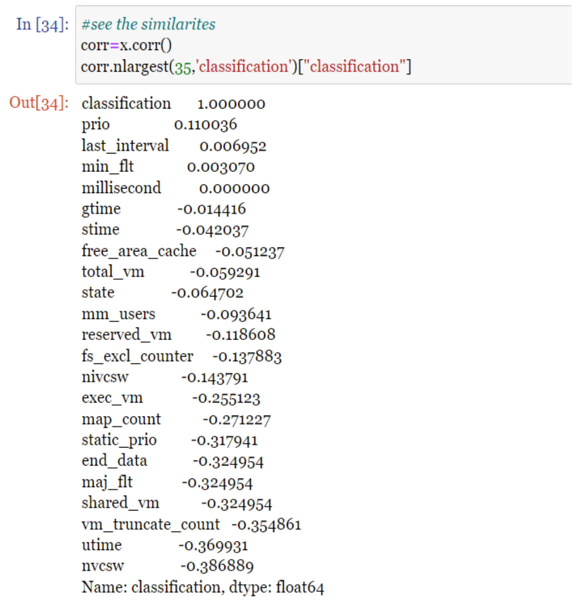


```
In [38]:  #check for duplicate
          dup = data.duplicated()

          print(dup);

          ded = data.drop_duplicates()

          print(ded.shape)

          0      False
          1      False
          2      False
          3      False
          4      False
                 ...
          99995  False
          99996  False
          99997  False
          99998  False
          99999  False
          Length: 100000, dtype: bool
          (100000, 35)
```

2. Classify the dataset to identify how many bening and the malware files are in the dataset. Analyze the distribution of the classes (benign and malware) in the dataset. Determine the number of samples belonging to each class.

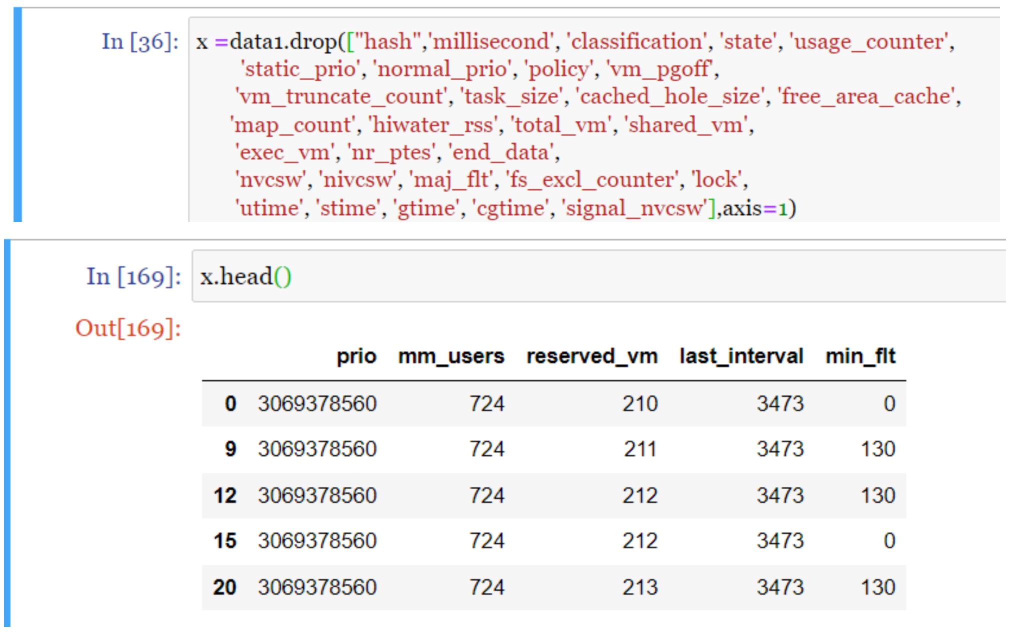


3. Feature selection based on correlation(Pearson's correlation coefficient) to check which is highly correlated with the target variable or have strong inter-correlations among themselves.



```
In [34]: #see the similarites
         corr=x.corr()
         corr.nlargest(35,'classification')["classification"]

Out[34]: classification      1.000000
         prio              0.110036
         last_interval      0.006952
         min_flt          0.003070
         millisecond        0.000000
         gtime            -0.014416
         stime            -0.042037
         free_area_cache   -0.051237
         total_vm          -0.059291
         state           -0.064702
         mm_users          -0.093641
         reserved_vm       -0.118608
         fs_excl_counter   -0.137883
         nivcsw           -0.143791
         exec_vm          -0.255123
         map_count         -0.271227
         static_prio       -0.317941
         end_data          -0.324954
         maj_flt          -0.324954
         shared_vm         -0.324954
         vm_truncate_count  -0.354861
         utime            -0.369931
         nvcsw            -0.386889
         Name: classification, dtype: float64
```

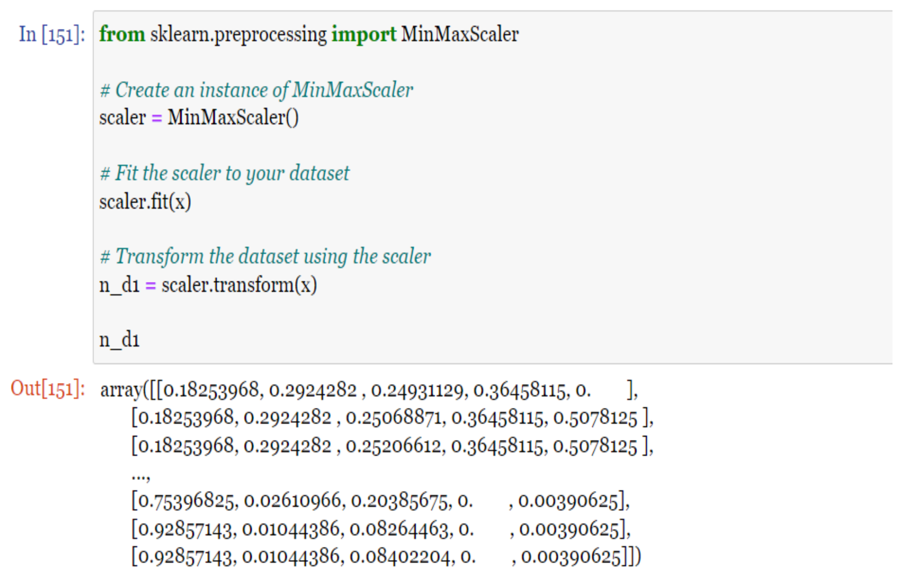
4. Drop some features based on correlation.



```
In [36]: x =data1.drop(["hash",'millisecond', 'classification', 'state', 'usage_counter',
               'static_prio', 'normal_prio', 'policy', 'vm_pgoff',
              'vm_truncate_count', 'task_size', 'cached_hole_size', 'free_area_cache',
             'map_count', 'hiwater_rss', 'total_vm', 'shared_vm',
              'exec_vm', 'nr_ptes', 'end_data',
              'nvcsw', 'nivcsw', 'maj_flt', 'fs_excl_counter', 'lock',
              'utime', 'stime', 'gtime', 'cgtime', 'signal_nvcsw'],axis=1)

In [169]: x.head()

Out[169]:
```

	prio	mm_users	reserved_vm	last_interval	min_flt
0	3069378560	724	210	3473	0
9	3069378560	724	211	3473	130
12	3069378560	724	212	3473	130
15	3069378560	724	212	3473	0
20	3069378560	724	213	3473	130

5. Data converted into correct format for the model using MiniMaxScaler.



```
In [151]: from sklearn.preprocessing import MinMaxScaler

          # Create an instance of MinMaxScaler
          scaler = MinMaxScaler()

          # Fit the scaler to your dataset
          scaler.fit(x)

          # Transform the dataset using the scaler
          n_d1 = scaler.transform(x)

          n_d1

Out[151]: array([[0.18253968, 0.2924282 , 0.24931129, 0.36458115, 0.        ],
                 [0.18253968, 0.2924282 , 0.25068871, 0.36458115, 0.5078125 ],
                 [0.18253968, 0.2924282 , 0.25206612, 0.36458115, 0.5078125 ],
                 ...,
                 [0.75396825, 0.02610966, 0.20385675, 0.        , 0.00390625],
                 [0.92857143, 0.01044386, 0.08264463, 0.        , 0.00390625],
                 [0.92857143, 0.01044386, 0.08402204, 0.        , 0.00390625]])
```

6. Apply KNN algorithm on the dataset.

   Steps involved in KNN -

   1. Initialize the value of K (the number of nearest neighbors).

   2. For each instance in the test dataset:

      2.1. Calculate the distance between the test instance and all the training instances using a distance metric (e.g. Euclidean distance).

      2.2. Sort the distances in ascending order and select the K nearest neighbors.

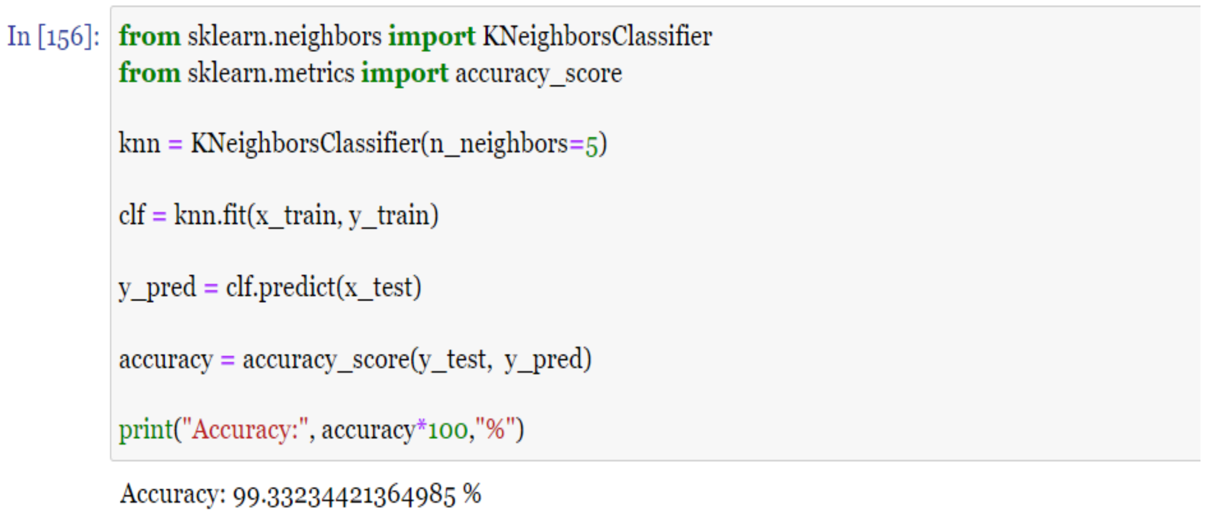
      2.3. Retrieve the labels of the K nearest neighbors.

      2.4. Determine the class with the majority vote among the K nearest neighbors.

      2.5. Assign the determined class as the predicted class for the test instances.

   3. Repeat step 2 for all instances in the test dataset.

   4. Evaluate the performance of the KNN algorithm



```
In [156]: from sklearn.neighbors import KNeighborsClassifier
          from sklearn.metrics import accuracy_score

          knn = KNeighborsClassifier(n_neighbors=5)

          clf = knn.fit(x_train, y_train)

          y_pred = clf.predict(x_test)

          accuracy = accuracy_score(y_test,  y_pred)

          print("Accuracy:", accuracy*100,"%")

          Accuracy: 99.33234421364985 %
```

Achieving an accuracy of 99% in the K-nearest neighbors (KNN) algorithm is a significant achievement and indicates the effectiveness of the model on the given dataset.

9 Conclusion

Machine learning has revolutionized the field of malware detection, providing powerful tools to combat increasingly sophisticated cyber threats. Machine learning algorithms contribute to enhancing security measures and staying one step ahead of malware creators. The detection of malware using machine learning techniques, specifically through the use of features selection and the K-nearest neighbors (KNN) algorithm, along with a study of malware analysis, has shown promising results in identifying and mitigating the threats posed by malicious software. However, continuous research, collaboration, and advancements in this domain are essential to effectively address the ever-changing landscape of malware and ensure the security of digital systems and networks.

10 References

1. The rise of machine learning for detection and classification of malware: Research developments, trends and challenges.
https://www.sciencedirect.com/science/article/pii/S1084804519303868

2. Detection of malware in downloaded files using various machine learning models.
https://www.sciencedirect.com/science/article/pii/S111086652200072X