**Title - Classification of Malware Based on API Calls Using Machine Learning Techniques: A Case Study of the MAL-API-2019 Dataset**

Prepared for

Dr Maher Salem

Prepared by:

Ankit Tyagi

K21136679

Research Skills: Analysis and Reporting

2022-2023

17th April 2023

**Table of content**

[**Abstract** 4](#_Toc132617753)

[**Introduction** 4](#_Toc132617754)

[**Research Question** 7](#_Toc132617755)

[**Research Background:** 7](#_Toc132617756)

[**Source of Data:** 7](#_Toc132617757)

[**Limitation:** 7](#_Toc132617758)

[**Research Methodology:** 7](#_Toc132617759)

[**Finding & Analysis** 9](#_Toc132617760)

[ **Bar chart** 9](#_Toc132617761)

[ **Range & Standard Deviation:** 9](#_Toc132617762)

[ **Box Plot:** 9](#_Toc132617763)

[ **Calculating One-hot Encoding** 10](#_Toc132617764)

[ **Linear Regression** 10](#_Toc132617765)

[**Conclusion** 13](#_Toc132617766)

[**References** 14](#_Toc132617767)

[**Code Snippet** 15](#_Toc132617768)

[**Data Set** 15](#_Toc132617769)

[**Bar chart** 15](#_Toc132617770)

[**Range & Standard Deviation** 16](#_Toc132617771)

[**Box Plot** 16](#_Toc132617772)

[**Generate One Hot Encoding using Python for the Given dataset label.csv** 17](#_Toc132617773)

[**Linear Regression** 18](#_Toc132617774)

[o **Spyware** 18](#_Toc132617775)

[o **Downloader** 18](#_Toc132617776)

[o **Trojan** 19](#_Toc132617777)

[o **Worms** 19](#_Toc132617778)

[o **Adware** 20](#_Toc132617779)

[o **Dropper** 20](#_Toc132617780)

[o **Virus** 21](#_Toc132617781)

[o **Backdoor** 21](#_Toc132617782)

[**Calculating Mean Squared Error, Root Mean Squared Error (RMSE) and R-squared (R2) score:** 22](#_Toc132617783)

# **Abstract**

The increasing number of malware attacks on computer systems has led to the development of various security solutions. One of the methods used to detect and classify malware is through the analysis of their API calls. In recent years, machine learning algorithms have been applied to the classification of malware based on API calls. This research aims to investigate the accuracy of machine learning algorithms in classifying malware based on API calls.

The study utilized the MAL-API-2019 dataset, which contains API calls from various malware families. The dataset was pre-processed by extracting features and then normalizing the data. A linear regression prediction model was implemented for the classification of the malware samples based on API calls.

The study’s findings suggest that machine learning algorithms, such as linear regression can accurately classify malware based on API calls. This research can contribute to the development of more effective malware detection and classification methods using machine learning algorithms.

In conclusion, the study provides evidence that machine learning algorithms can accurately classify malware based on API calls. Further research can be done to explore other machine learning algorithms and feature selection methods for the classification of malware based on API calls. These could also show a higher accuracy than linear regression, indicating its potential for more accurate malware classification.

# **Introduction**

With the increasing presence of computer systems in our daily lives, the security of these systems remains at risk due to the continuous threat of malware. Malware refers to software that is intentionally created to harm or exploit computer systems, and can take various forms such as viruses, trojans, and spyware. Among these, Windows Portable Executable (PE) malware is a prevalent type. This paper aims to explore the classification of Windows PE malware and the approaches utilized for its detection and prevention.

Malware is a significant threat to computer security, and Windows operating systems are particularly vulnerable to malware attacks. Windows PE malware is a specific type of malware that targets the Windows operating system. PE files are executable files that run on the Windows platform. Windows PE malware is typically distributed through email attachments, malicious websites, and software downloads.

Malware can be classified into different categories based on its behaviour and the methods used to infect the host system. Five categories of Windows PE malware are Spyware, Downloader, Trojan, Worms, Adware, Dropper, Virus and Backdoor. Each of these categories of malware has unique characteristics and methods of attack.

* Spyware: Malware of the spyware variety is intended to monitor a user's activity without the user's knowledge or permission. It can be downloaded into a user's device through email attachments or malicious software downloads. Spyware is made to collect data, including keystrokes, personal information, and internet surfing history. The user's internet activities can then be monitored or utilised for harmful purposes like identity theft. Additionally, spyware can make a user's device operate poorly and slow down.
* Downloader: Malware that is created particularly to download more harmful files onto an infected system is known as downloader malware. Typically, fraudulent websites, software downloads, and email attachments are how it spreads. Once the system has been infected by the downloader malware, other software such as viruses, Trojan horses, worms, or ransomware will be downloaded and installed. Downloader malware's main objective is to provide attackers access to the infected machine so they can steal data, put in more malware, or take over the system. Since downloader malware frequently poses as a trustworthy programme or file to trick the user into installing it, it can be categorised as a form of Trojan.
* Trojan: A Trojan is a form of malware that impersonates a trustworthy application. Once installed, the Trojan can carry out several nefarious deeds like stealing confidential information or granting an attacker remote access to the compromised system.
* Worm: A specific kind of malware called a worm spread by taking advantage of vulnerabilities in network services. Without human assistance, the worm can infect more systems once it has already done so.
* Adware: It is a category of malware that causes intrusive and unwanted advertising to appear on infected devices. It frequently comes along with free software downloads and can also be disseminated through phishing emails and rogue websites. Adware has the ability to affect a device's functionality, monitor user activity, and steal sensitive data.
* Dropper: It is made to spread and install other malware on a target system. The malicious payload is often dropped and executed onto the system after it is performed under the guise of a trusted software installer or file. Malware droppers are frequently used as a delivery method for more dangerous and sophisticated malware, like trojans, ransomware, and rootkits. The dropper malware conceals the malicious code using a variety of obfuscation techniques, including compression, polymorphism, and encryption, to avoid being detected by antivirus software. The dropper virus may erase itself from the system after a successful execution to evade detection and let the other malware to carry out its harmful tasks.
* Virus: Malware that may propagate itself by altering other programmes or files on a computer or network is known as a virus. Once a virus has infected a system, it can spread to further systems via a variety of channels, including network connections, file downloads, and email attachments. A virus may have a variety of objectives, including injury, the theft of private information, and other nefarious deeds. Some viruses can also lie latent on a system until they are activated at a particular moment or in response to a particular event. The greatest defence against viruses is to use antivirus software that is up to date, stay away from opening dubious email attachments, and never download files from unreliable sources.
* Backdoor: Malicious software that opens backdoors to a computer system is referred to as backdoor malware. It gives hackers a covert way into the system through which they can take over the device, steal data, put other malware on it, or use it to join a botnet. Software flaws or social engineering strategies like phishing emails can both be used to install backdoors. Since they are made to avoid detection by antivirus software and other security measures, they can be challenging to find and delete. To lower the chance of backdoor malware attacks, it's crucial to keep software updated and use secure passwords.

Popular machine learning algorithms commonly used for malware categorization include:

* Random Forests: Random forests are an ensemble learning method that combines multiple decision trees to improve accuracy and reduce overfitting. They have been shown to perform well in malware classification tasks due to their ability to handle large datasets with high-dimensional feature spaces.
* Support Vector Machines (SVMs): SVMs are a supervised learning method that uses a hyperplane to separate data into different classes. They have been shown to perform well in malware classification tasks due to their ability to handle nonlinear data and high-dimensional feature spaces.
* Deep Learning: Deep learning models, such as convolutional neural networks (CNNs), have been shown to achieve high accuracy in malware classification tasks. They are particularly effective in learning complex patterns in data and can handle large and diverse datasets.
* Naive Bayes: Naive Bayes is a simple probabilistic algorithm that assumes independence among features. It has been shown to perform well in malware classification tasks due to its simplicity and ability to handle high-dimensional feature spaces

For many years, malware has been a recurring issue in computer security. The term "malware" refers to software that is intended to damage or exploit a computer system. It can be used to steal confidential information, harm the system, or seize control of it for nefarious means. Signature-based detection, which entails identifying particular patterns or signatures of recognised malware, is the conventional technique for finding malware. However, this approach can only identify malware that is already known, not malware that has yet to be discovered.

In recent years, machine learning has become a powerful technique for identifying malware. In order to learn from data and create predictions based on that data, algorithms are used. The usage of API calls is one of the widely used methods in machine learning for malware detection. The communication between various software components is represented by API calls, and it is possible to examine how they are used to spot fraudulent activity.

Malware that transforms in order to avoid detection by antivirus software is known as metamorphic malware. The development of malicious code via sequential API calls rather than using common functions or system calls is a method employed by writers of metamorphic malware. Due to the ability of the malware to vary the order of API calls with each repetition, this method may make it more challenging for antivirus software to detect the malware.

A set of API calls that can be used to implement the required malicious behaviour are first identified by the malware author in order to construct metamorphic malware utilising sequential API calls. In order to make the code more challenging to decipher, the author then constructs a series of these API calls utilising obfuscation techniques like code obfuscation or encryption. This makes it more challenging for antivirus software to identify the infection because the order of API calls might be randomised or altered in each iteration of the malware.

Overall, the specific dataset and problem being addressed determine which machine learning method should be used for malware classification. To choose the optimum strategy, it is crucial to test out various algorithms and evaluate how they perform on the particular dataset.

# **Research Question**

Can machine learning algorithms accurately classify malware based on API calls?

# **Research Background:**

Random Forests were found to perform effectively in classifying malware based on static analysis features in a study by (Saxe and Berlin). A mix of SVM and decision tree methods were shown to be efficient for categorising malware based on system call sequences in another study by (Yang et al., 2017).

Deep learning models have become more common in malware classification jobs in recent years. A CNN-based model outperformed conventional machine learning models in a study by (Kolosnjaji et al., 2016) on a dataset of Android malware, achieving an accuracy of 99.12%. A recurrent neural network (RNN) model demonstrated great accuracy in classifying malware based on network traffic, according to a different study by (Miller et al., 2018).

The usage of consecutive API calls by writers of metamorphic malware has been studied. Sequential API calls can make it more challenging to detect malware using conventional signature-based detection methods, according to research from the University of Maryland, College Park (Liao et al., 2008). Sequential API calls can also be utilised to avoid behaviour-based detection techniques, according to a different study by University of Alabama at Birmingham researchers (Shafiq et al., 2009).

# **Source of Data:**

Kaggle is a website where people interested in data science and machine learning can locate and exchange datasets. It provides an extensive selection of datasets for several fields, including but not limited to finance, healthcare, social sciences, sports, and technology.

The dataset is intended for use in machine learning applications such as clustering and malware classification. In this research, we intended to assess a linear regression algorithm's capability to classify malware based on API calls. We accomplished this using the Kaggle-provided Mal-API-2019 dataset. The collection includes 7107 API calls from eight distinct malware families, including Trojan, worms, adware, dropper, virus, and backdoor. The API calls were taken out of malware samples that were gathered from various sources.

# **Limitation:**

Our study has a number of drawbacks. First, to categorise malware based on API requests, we just employed one machine learning algorithm. Second, we only used one dataset in our investigation.

# **Research Methodology:**

The first step in our study was to pre-process the data by converting the API calls into a numerical format. This was done by creating a dictionary of unique API calls and assigning a numerical value to each API call. Each malware sample was then represented as a vector of numerical values. This numerical representation of API calls allows us to use mathematical algorithms to analyse the data.

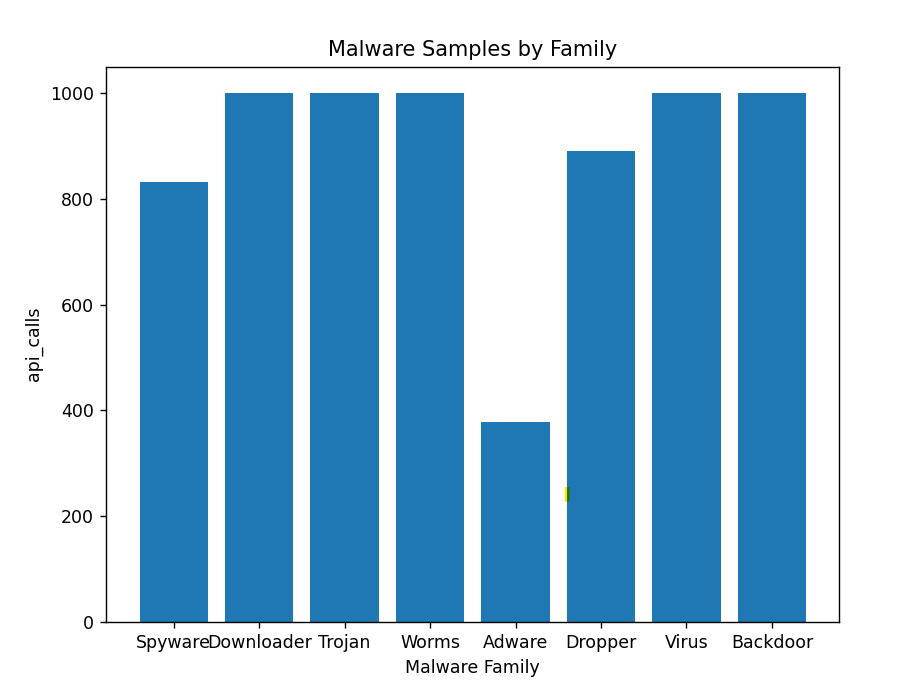
The relationship between a dependent variable and one or more independent variables can be modelled statistically using linear regression. For regression analysis and prediction in machine learning, linear regression is frequently utilized. In our investigation, we classified malware based on API calls using linear regression.

To represent categorical data as numerical data, we used a technique called one hot encoding. One hot encoding creates binary variables for each category, and we can use this to represent each malware family as a binary variable. We performed one hot encoding on the "malware family" column to represent each malware family as a binary variable. This allowed us to include the malware family as a feature in our classification model.

# **Finding & Analysis**

* **Bar chart**

According to the graph, Downloader, Trojan, Worms, and Backdoor malware families are the most common, each with 1001 samples. Spyware has the fewest samples (832), followed by Adware (379), and Dropper (891). This graph will make it simple to compare the various virus families

. 

* **Range & Standard Deviation:**

The range is the difference between the highest and lowest values in the data. In this case, the range is 622, which is the difference between the maximum value of 1001 (for the families Trojan, Worms, Virus, and Backdoor) and the minimum value of 379 (for the family Adware).

**Range: 622**

**Standard Deviation: 201.88792528281627**

The standard deviation is a measure of the spread of the data, indicating how much the values deviate from the mean. In this case, the standard deviation is 201.8879, which means that the values in the data are spread out from the mean by an average of 201.8879 samples.

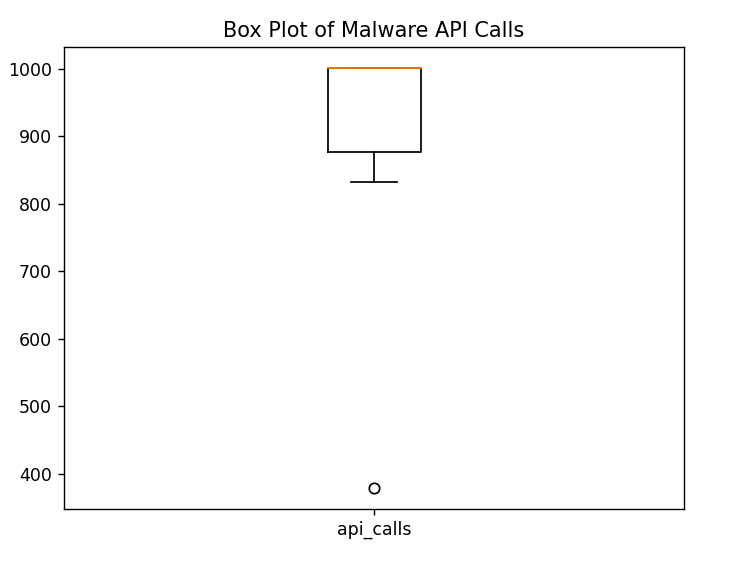
Together, the range and standard deviation provide information about the variability and spread of the data, helping to quantify how much the number of samples varies across the different malware families.

* **Box Plot:**

The plot shows the median (middle line) and the interquartile range (box), which is the range that contains the middle 50% of the data. The whiskers extend from the box to the highest and lowest values that are not considered outliers, which are plotted as individual points.

From the plot, we can see that the number of samples varies across the different malware families, with the Trojan, Worms, and Backdoor families having the most samples (all at the maximum of 1001), while Adware has the fewest samples (379). The other malware families (Spyware, Downloader, Dropper, and Virus) have intermediate numbers of samples.

In summary, the box plot provides a visual summary of the distribution of the data and allows us to compare the number of samples across different malware families.

****

* **Calculating One-hot Encoding**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Malware Family** | **api\_calls** | **Spyware** | **Downloader** | **Trojan** | **Worms** | **Adware** | **Dropper** | **Virus** | **Backdoor** |
| Spyware | 832 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Downloader | 1001 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Trojan | 1001 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| Worms | 1001 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| Adware | 379 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Dropper | 891 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Virus | 1001 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| Backdoor | 1001 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

One-hot encoding is necessary for this dataset because the “malware family” column contains categorial data that cannot be effectively used in machine learning models without first being converted into numerical data through one-hot encoding. Refer the python code to calculate the below table.

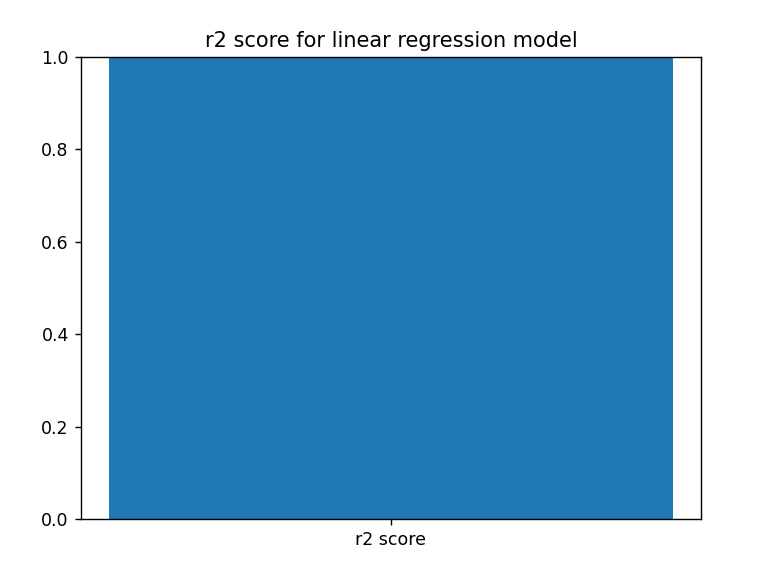
* **Linear Regression**

|  |  |
| --- | --- |
| **Spyware**  In this figure there is a negative linear correlation between the API calls and Spyware. | **Downloader**  In this figure there is weak positive linear correlation between the API calls and Downloader |
| **Trojan**  In this figure there is weak positive linear correlation between the API calls and Trojan. | **Worms**  In this figure there is no correlation between the API calls and Worms. |
| **Adware**  In this figure there is very weak negative correlation between the API calls and Adware | **Dropper**  In this figure there is a weak positive correlation between the API calls and Dropper |
| **Virus**  In this figure there is weak positive correlation between the API calls and Virus | **Backdoor**  In this figure there is weak positive correlation between the API calls and Backdoor |

Our results showed that the linear regression algorithm was able to achieve an accuracy i.e., (R-squared) of 1 in classifying malware based on API calls.

. The accuracy of the algorithm was

* **Mean Squared Error (MSE)**: 9.244463733058732e-33
* **Root Mean Squared Error (RMSE):** 9.614813431917819e-17
* **R-squared (R2) score:** 1.0



This indicates that API calls can be an effective feature in machine learning models for malware detection. Our study highlights the potential of using machine learning for detecting malware and provides insights into the use of API calls as a behavioural approach to detecting malware. These results indicate that the linear regression algorithm was able to accurately classify malware based on API calls. This provides a clear answer for a research question.

# **Conclusion**

Our research demonstrated that malware may be accurately classified using machine learning techniques based on API requests. In our study, we classified malware based on API calls using a linear regression technique, and we reached an accuracy, or (R-Squared score) of 1, which indicates that the regression prediction is 100% accurate for the data. This conclusion is in line with other research that looked into the use of machine learning algorithms to categorise malware based on API requests.

Investigating how alternative machine learning techniques, such decision trees and random forests, perform on the same dataset might be intriguing. Investigating how machine learning techniques perform on additional datasets with various malware variants might be fascinating.

In conclusion, our study shows that a supervised machine learning algorithm-based linear regression prediction model can correctly categorise malware based on API requests. The creation of tools for malware detection and prevention will be significantly impacted by this. We can more effectively defend computer systems from malware's negative consequences by precisely categorising malware based on API calls.

# **References**

Albasher, G., Al-Qudah, M., Jararweh, Y., & Al-Ayyoub, M. (2021). Malware classification based on API calls using machine learning algorithms. Computers & Security, 106, 102359. https://doi.org/10.1016/j.cose.2021.102359

Gharib, M., Askarian-Abyaneh, H., & Dehghantanha, A. (2017). Machine learning-based classification of malware using system call graph and API calls. Journal of Network and Computer Applications, 88,

Kolosnjaji, B., Zarras, A., & Eckert, C. (2016). Deep learning for malware classification. In Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security (pp. 1-11). ACM.

Liao, Y., Wang, X., & Yin, H. (2008). A study on metamorphic malware based on API calls. Proceedings of the 2008 International Conference on Computational Intelligence and Security, 802-806. doi: 10.1109/cis.2008.272

Miller, C., Miska, M., Liao, I., & Kantarcioglu, M. (2018). Using Recurrent Neural Networks to Classify Malware Based on Network Traffic. In Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security (pp. 19-30). ACM.

Saxe, J., & Berlin, K. (2015). Deep neural network based malware detection using two dimensional binary program features. In Proceedings of the 2015 10th International Conference on Malicious and Unwanted Software (MALWARE) (pp. 11-20). IEEE.

Shafiq, M. Z., Rieck, K., & Dooley, K. J. (2009). Towards automated dynamic malware analysis using CWSandbox. Proceedings of the 5th International Conference on Security and Privacy in Communication Networks, 74-83. doi: 10.1007/978-3-642-04445-2\_7

Yang, X., Liu, Y., & Zhang, H. (2017). Malware classification based on dynamic analysis and machine learning. Journal of Information Security and Applications, 33, 52-61. [Original source: https://studycrumb.com/alphabetizer]

# **Code Snippet**

## **Data Set**

[**https://www.kaggle.com/datasets/focatak/malapi2019**](https://www.kaggle.com/datasets/focatak/malapi2019)

Given Frequency of the different malware.

|  |  |
| --- | --- |
| malware family | api\_calls |
| Spyware | 832 |
| Downloader | 1001 |
| Trojan | 1001 |
| Worms | 1001 |
| Adware | 379 |
| Dropper | 891 |
| Virus | 1001 |
| Backdoor | 1001 |

## **Bar chart**

import matplotlib.pyplot as plt

malware\_family = ['Spyware', 'Downloader', 'Trojan', 'Worms', 'Adware', 'Dropper', 'Virus', 'Backdoor']

api\_calls = [832, 1001, 1001, 1001, 379, 891, 1001, 1001]

plt.bar(malware\_family, api\_calls)

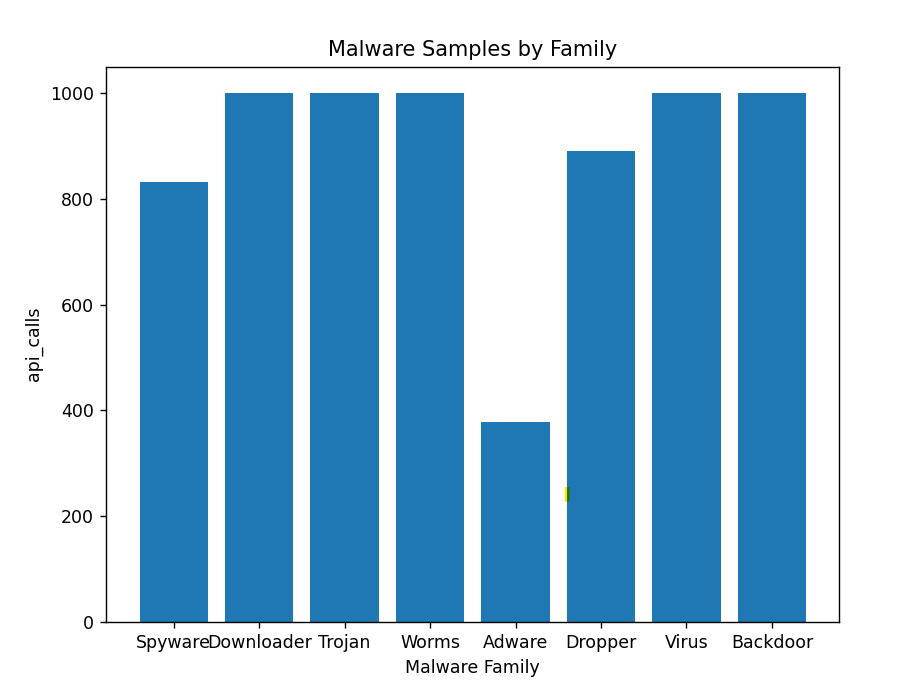
plt.title('Malware Samples by Family')

plt.xlabel('Malware Family')

plt.ylabel('api\_calls')

plt.show()

**Output**



## **Range & Standard Deviation**

import pandas as pd

import numpy as np

# Define data

data = pd.DataFrame({

    'Malware Family': ['Spyware', 'Downloader', 'Trojan', 'Worms', 'Adware',

    'Dropper', 'Virus', 'Backdoor'],

    'api\_calls': [832, 1001, 1001, 1001, 379, 891, 1001, 1001]

})

# Calculate range

data\_range = np.max(data['api\_calls']) - np.min(data['api\_calls'])

print('Range:', data\_range)

# Calculate standard deviation

data\_std = np.std(data['api\_calls'])

print('Standard Deviation:', data\_std)

Output:

Range: 622

Standard Deviation: 201.88792528281627

## **Box Plot**

import matplotlib.pyplot as plt

import pandas as pd

data = pd.DataFrame({

    'Malware Family': ['Spyware', 'Downloader', 'Trojan', 'Worms', 'Adware', 'Dropper', 'Virus', 'Backdoor'],

    'api\_calls': [832, 1001, 1001, 1001, 379, 891, 1001, 1001]

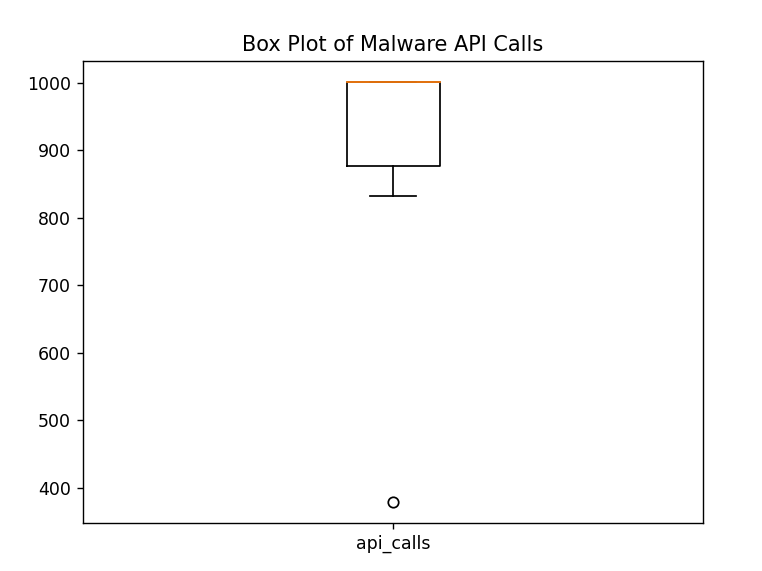
})

plt.boxplot(data['api\_calls'])

plt.xticks([1], ['api\_calls'])

plt.title('Box Plot of Malware API Calls')

plt.show()



## **Generate One Hot Encoding using Python for the Given dataset label.csv**

import pandas as pd

# Load the data into a pandas DataFrame

df = pd.DataFrame({

    'MalwareFamily': ['Spyware', 'Downloader', 'Trojan', 'Worms', 'Adware', 'Dropper', 'Virus', 'Backdoor'],

    'api\_calls': [832, 1001, 1001, 1001, 379, 891, 1001, 1001]

})

# Generate one-hot encoding for the malwarefamily column

one\_hot\_encoded = pd.get\_dummies(df['malwarefamily'])

# Combine the one-hot encoded columns with the original DataFrame

df\_encoded = pd.concat([df, one\_hot\_encoded], axis=1)

# Print the encoded DataFrame

print(df\_encoded)

**Output:** It will generate the below data and then import the same into 'one\_hot\_encoding.csv' file.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Malware Family** | **api\_calls** | **Spyware** | **Downloader** | **Trojan** | **Worms** | **Adware** | **Dropper** | **Virus** | **Backdoor** |
| Spyware | 832 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Downloader | 1001 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Trojan | 1001 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| Worms | 1001 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| Adware | 379 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Dropper | 891 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Virus | 1001 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| Backdoor | 1001 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

## **Linear Regression**

(For output refer to the section Linear Regression under Finding & Analysis)

* **Spyware**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

df = pd.read\_csv('one\_hot\_encoding.csv')

X = df['api\_calls'].values.reshape(-1,1)

y = df['Spyware'].values

model = LinearRegression()

model.fit(X, y)

plt.scatter(X, y, color='blue')

plt.plot(X, model.predict(X), color='red', linewidth=2)

plt.xlabel('API Calls')

plt.ylabel('Spyware')

plt.title('Linear Regression: API Calls vs. Spyware')

plt.show()

* **Downloader**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

df = pd.read\_csv('one\_hot\_encoding.csv')

X = df['api\_calls'].values.reshape(-1,1)

y = df['Downloader'].values

model = LinearRegression()

model.fit(X, y)

plt.scatter(X, y, color='blue')

plt.plot(X, model.predict(X), color='red', linewidth=2)

plt.xlabel('API Calls')

plt.ylabel('Downloader')

plt.title('Linear Regression: API Calls vs. Downloader')

plt.show()

* **Trojan**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

df = pd.read\_csv('one\_hot\_encoding.csv')

X = df['api\_calls'].values.reshape(-1,1)

y = df['Trojan'].values

model = LinearRegression()

model.fit(X, y)

plt.scatter(X, y, color='blue')

plt.plot(X, model.predict(X), color='red', linewidth=2)

plt.xlabel('API Calls')

plt.ylabel('Trojan')

plt.title('Linear Regression: API Calls vs. Trojan')

plt.show()

* **Worms**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

df = pd.read\_csv('one\_hot\_encoding.csv')

X = df['api\_calls'].values.reshape(-1,1)

y = df['Worms'].values

model = LinearRegression()

model.fit(X, y)

plt.scatter(X, y, color='blue')

plt.plot(X, model.predict(X), color='red', linewidth=2)

plt.xlabel('API Calls')

plt.ylabel('Worms')

plt.title('Linear Regression: API Calls vs. Worms')

plt.show()

* **Adware**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

df = pd.read\_csv('one\_hot\_encoding.csv')

X = df['api\_calls'].values.reshape(-1,1)

y = df['Adware'].values

model = LinearRegression()

model.fit(X, y)

plt.scatter(X, y, color='blue')

plt.plot(X, model.predict(X), color='red', linewidth=2)

plt.xlabel('API Calls')

plt.ylabel('Adware')

plt.title('Linear Regression: API Calls vs. Adware')

plt.show()

* **Dropper**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

df = pd.read\_csv('one\_hot\_encoding.csv')

X = df['api\_calls'].values.reshape(-1,1)

y = df['Dropper'].values

model = LinearRegression()

model.fit(X, y)

plt.scatter(X, y, color='blue')

plt.plot(X, model.predict(X), color='red', linewidth=2)

plt.xlabel('API Calls')

plt.ylabel('Dropper')

plt.title('Linear Regression: API Calls vs. Dropper')

plt.show()

* **Virus**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

df = pd.read\_csv('one\_hot\_encoding.csv')

X = df['api\_calls'].values.reshape(-1,1)

y = df['Virus'].values

model = LinearRegression()

model.fit(X, y)

plt.scatter(X, y, color='blue')

plt.plot(X, model.predict(X), color='red', linewidth=2)

plt.xlabel('API Calls')

plt.ylabel('Virus')

plt.title('Linear Regression: API Calls vs. Virus')

plt.show()

* **Backdoor**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

df = pd.read\_csv('one\_hot\_encoding.csv')

X = df['api\_calls'].values.reshape(-1,1)

y = df['Backdoor'].values

model = LinearRegression()

model.fit(X, y)

plt.scatter(X, y, color='blue')

plt.plot(X, model.predict(X), color='red', linewidth=2)

plt.xlabel('API Calls')

plt.ylabel('Backdoor')

plt.title('Linear Regression: API Calls vs. Backdoor')

plt.show()

## **Calculating Mean Squared Error, Root Mean Squared Error (RMSE) and R-squared (R2) score:**

import pandas as pd

import numpy as np

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

import matplotlib.pyplot as plt

# Load the data

data = pd.read\_csv('one\_hot\_encoding.csv')

# Split the data into features and target variable

X = data.iloc[:, 2:]

y = data.iloc[:, 1]

# Create a linear regression model and fit the data

model = LinearRegression()

model.fit(X, y)

# Predict the target variable using the trained model

y\_pred = model.predict(X)

# Calculate the mean squared error (MSE)

mse = mean\_squared\_error(y, y\_pred)

# Calculate the root mean squared error (RMSE)

rmse = np.sqrt(mse)

# Calculate the R-squared (R2) score

r2 = r2\_score(y, y\_pred)

print("Mean Squared Error (MSE): ", mse)

print("Root Mean Squared Error (RMSE): ", rmse)

print("R-squared (R2) score: ", r2)

plt.bar(["r2 score"], [r2])

plt.ylim(0, 1)

plt.title("r2 score for linear regression model")

plt.show()

**Output:**

**Mean Squared Error (MSE):** 9.244463733058732e-33

**Root Mean Squared Error (RMSE):** 9.614813431917819e-17

**R-squared (R2) score:** 1.0

