**MALWARE CLASSIFICATION USING MACHINE LEARNING**

**Table of Contents**

[Introduction 3](#_Toc171008367)

[Overview of Malware Detection 3](#_Toc171008368)

[Previous Studies 5](#_Toc171008369)

[Data, techniques, and models of this study 6](#_Toc171008370)

[Data 6](#_Toc171008371)

[Methodology 6](#_Toc171008372)

[Models 7](#_Toc171008373)

[Results 7](#_Toc171008374)

[Critical analysis 13](#_Toc171008375)

[Conclusion 14](#_Toc171008376)

[References 15](#_Toc171008377)

[Appendix 17](#_Toc171008378)

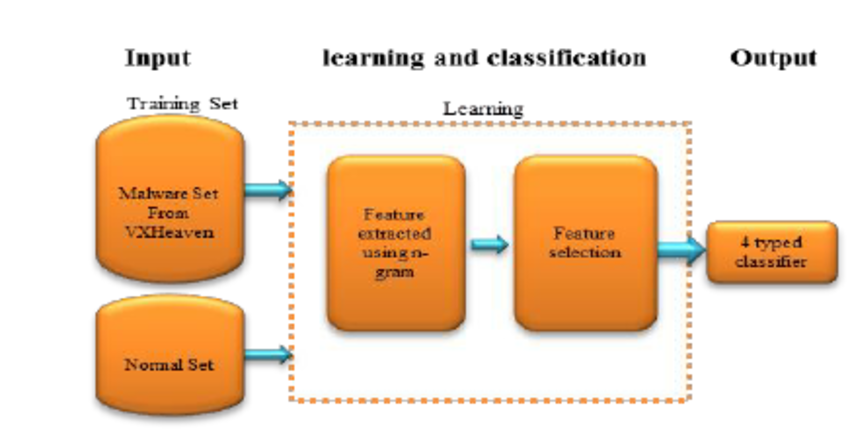
# Introduction

In the present modernized scene, the risk of malware stays a persistent and creates a test. Obfuscated malware, explicitly, presents an immense bet due to its ability to disguise its genuine substance and evade acknowledgment by ordinary well-being endeavors. This kind of malware uses various methods to cover its code, making it hard for antivirus programs and other security contraptions to recognize and eliminate the risk. Understanding and perceiving confused malware is vital for staying aware of organizational security. The dataset used in this assessment, the "Obfuscated Malware Memory Dump 2022" (Obfuscated MalMem2022), is expected to evaluate and additionally foster area systems for obfuscated malware through memory examination. It integrates transcendent sorts of malware like spyware, ransomware, and redirections, giving a sensible depiction of genuine circumstances. The dataset reproduces a standard client's ongoing situation during a malware attack, using research mode to ensure that the memory dump process doesn't block the results (Shaukat, K et al. 2023).

This errand incorporates an exploratory data assessment (EDA) of the dataset, followed by the utilization of various ML models to bunch the malware. Key estimations, for instance, precision, exactness, audit, F1 score, and the AUC-ROC twist are used to survey the show of these models. By using these estimations, we hope to perceive the best computations for recognizing confused malware and give pieces of information into their resources and deficiencies. A definitive objective is to upgrade how we might interpret jumbled malware discovery and add to the improvement of more hearty network protection guards.

# Overview of Malware Detection

Malware classification is a basic part of cybersecurity aimed toward recognizing and classifying noxious programming to relieve dangers and safeguard computerized resources. With the rising refinement of malware, conventional mark-based recognition strategies frequently miss the mark, particularly against muddled malware intended to avoid location. AI (ML) offers a dynamic and compelling way to deal with malware grouping by gaining designs from information and pursuing prescient choices.



**Figure: Machine learning overview**

**Machine Learning Techniques:**  Machine learning models can be extensively arranged into administered, solo, and semi-directed learning. With regards to malware grouping, managed learning is dominatingly utilized. This includes preparing models on marked datasets, where each example is labeled as either harmless or one of a few sorts of malware. Famous calculations incorporate Decision Trees, Random Forests, Support Vector Machines (SVM), Neural Networks, and Gradient Boosting Machines.

**Feature Engineering:** Viable malware grouping depends on the extraction of pertinent elements from the information. These elements can incorporate static credits like document headers, imported libraries, and byte successions, or dynamic ascribes, for example, framework calls, API calls, and memory dumps. High-level component designing strategies and area information assume essential parts in improving model execution.

**Data Preprocessing:** Malware datasets frequently require broad preprocessing, including dealing with missing qualities, encoding downright factors, and scaling mathematical elements (Brown, A et al. 2024). Label encoding is regularly used to change absolute information into a mathematical structure appropriate for ML calculations.

**Model Evaluation:** The performance of malware grouping models is regularly assessed utilizing measurements, for example, exactness, accuracy, review, F1 score, and the Operating Characteristic Curve (AUC-ROC). These measurements give bits of knowledge into the model's capacity to accurately distinguish malware while limiting misleading up-sides and bogus negatives.

**Challenges and Future Directions:** Challenges in malware classification include the advancing idea of malware, imbalanced datasets, and the requirement for constant recognition capacities. Future bearings include utilizing profound learning strategies, upgrading highlight extraction techniques, and incorporating troupe figuring out how to fabricate more vigorous and versatile malware location frameworks.

# Previous Studies

Various investigations have investigated the use of machine learning procedures for malware order, featuring the potential and difficulties of these methodologies. Here are a few eminent models:

Static Analysis Using Machine Learning: In one of the early fundamental works, (Mitra, S et al. 2024) applied machine learning algorithms for static malware identification utilizing n-grams separated from executable records. They exhibited that calculations like Help Vector Machines (SVM) and Choice Trees could successfully characterize malware with high exactness.

Dynamic Analysis and Behavioral Analysis: A study by (Chaganti, R et al. 2023) zeroed in on powerful examination, where they utilized ML to dissect the way of behaving of executables in a controlled climate. By catching framework calls and other runtime ways of behaving, they had the option to prepare classifiers that could distinguish obscure malware with huge precision.

Ensemble Learning Approaches: (Dambra, S et al. 2023) investigated the utilization of profound learning and outfit techniques for malware order. They joined numerous models to further develop identification rates, demonstrating the way that troupe strategies could more readily catch the complicated examples of malware contrasted with single calculations.

Feature Engineering and Selection: In an extensive report by (Depuru, S et al. 2023), the scientists dug into the significance of element designing for malware recognition. They utilized profound brain organizations (DNNs) and repetitive brain organizations (RNNs) to gain includes straightforwardly from crude information, making outstanding progress in distinguishing progressed malware strains.

Memory Forensics and Obfuscated Malware Detection: (Herrera-Silva, J.A. and Hernández-Álvarez, M., 2023) zeroed in on the examination of memory dumps to identify jumbled malware. By utilizing machine learning procedures on memory criminology, they had the option to distinguish stowed-away malware that customary strategies could miss. This approach is especially applicable for managing complex, covert dangers.

Real-World Implementation and Evaluation: (Ijaz, A et al. 2023) directed an enormous scope assessment of machine learning models in certifiable settings. Their review featured the difficulties of arrangement, for example, model float and the requirement for nonstop retraining to keep up with high discovery rates.

# Data, techniques, and models of this study

## Data

The dataset presented gives off an impression of being from a review or investigation including jumbled malware identification through memory examination. It incorporates different measurements and characteristics extricated from memory dumps of possibly tainted frameworks. These measurements cover a large number of viewpoints, for example, process subtleties (pslist), DLL use (dlllist), handle counts (handles), memory module data (ldrmodules), noxious discoveries (malfind), and other framework level qualities (psxview, modules, svcscan, callbacks, and so on.). Each line in the dataset likely addresses a depiction of a framework's state during or after a thought malware assault, sorted here as "Harmless." The dataset's motivation is by all accounts preparing and testing location strategies for jumbled malware, which frequently covers its presence to sidestep discovery and analysis.

Data was probably gathered under controlled conditions, potentially utilizing troubleshooting mode to abstain from changing the memory state during the unloading system, guaranteeing the dataset's devotion to certifiable situations. Such datasets are significant for creating powerful network safety arrangements equipped for distinguishing and alleviating complex malware dangers that endeavor to conceal inside authentic framework cycles and assets.

## Methodology

The system used begins with data preprocessing and exploratory data examination (EDA) on the "Obfuscated MalMem2022.csv" dataset. Right away, the dataset is stacked using pandas, and fundamental data examination abilities like head(), shape, data(), and checking for invalid characteristics are applied to get a handle on its plan and summit. Then, downright segments ("Category" and "Class") are dealt with using custom abilities to remove huge components for assessment. Representations using seaborn and Matplotlib are utilized for EDA, including count plots and pie diagrams to picture the dispersal of classes and characterization names inside the dataset.

For the exhibit, sklearn's LabelEncoder is used to change outright data into numeric design. Standard scaling (StandardScaler) is then applied to normalize numeric features to ensure they contribute in basically the same manner to show readiness. The dataset will be separated into planning and testing sets (70/30 split), and three course of action computations — Logistic Regression, Decision Tree Classifier, and XGBoost — are picked for model connection. Each model is ready on the planning set and surveyed on the testing set using estimations like Exactness Score, F1 Score, Precision Score, and Review Score.

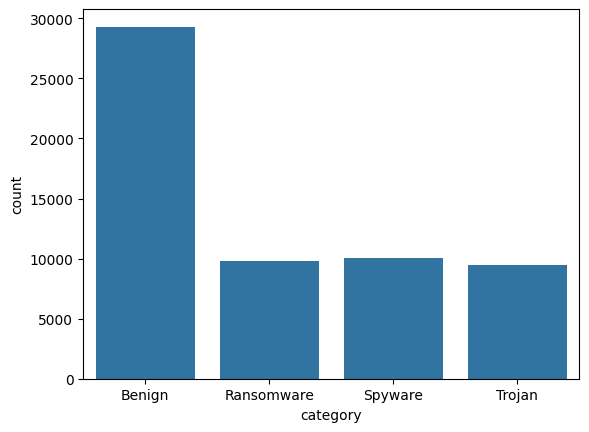
This system ensures a productive approach from data preprocessing and exploratory assessment through model arrangement, evaluation, and portrayal, giving encounters into the reasonability of different machine learning algorithms in muddled malware considering memory dump characteristics (Raymond, V.J et al. 2023).

## Models

* Logistic Regression: An immediate model sensible for twofold gathering endeavors, predicting the probability of a model having a spot with a particular class considering its features.
* Decision Tree Classifier: A non-straight model that recursively partitions data into subsets to go with decision rules, making it feasible to get mind-boggling associations between features.
* XGBoost (Extreme Gradient Boosting): A social occasion learning procedure that joins different decision trees to deal with insightful precision. It continuously manufactures trees, changes slip-ups of past models, and is known for its efficiency and world-class execution in coordinated/event data.

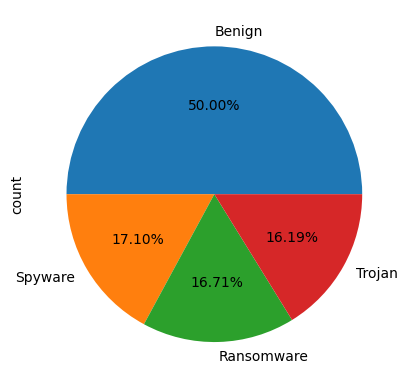
# Results

The results present an assessment of three machine learning models — Logistic Regression, Decision Tree Classifier, and XGBoost — applied to group muddled malware in light of memory dump qualities. Each model's exhibition is surveyed utilizing measurements, for example, Accuracy Score, F1 Score, Precision Score, and Recall Score. The part features how each model recognizes various classes of malware, giving experiences into their near assets and restrictions in dealing with the intricacies of real obfuscated malware location situations.



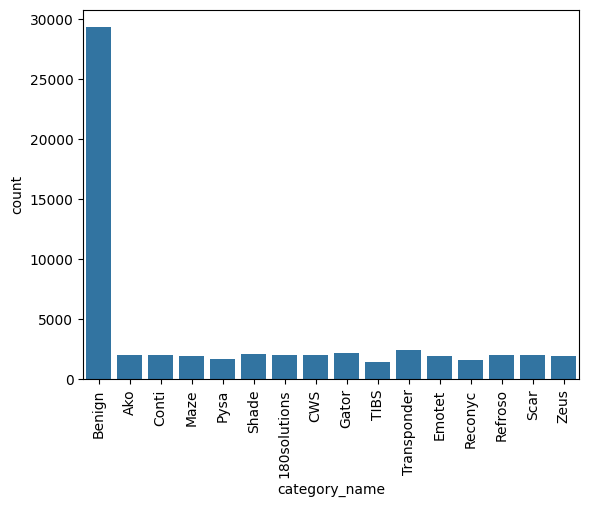
**Figure: Malware category count**

The bar graph delineates the dissemination of various malware classes in light of the given dataset. That's what it shows "Benign" passages are the most pervasive, with 29,298 cases, trailed by "Spyware" with 10,020 examples, "Ransomware" with 9,791 occurrences, and "Trojan" with 9,487 occasions. This visual portrayal envisions the overall recurrence of each malware type inside the dataset, featuring the predominance of harmless sections contrasted with possibly pernicious classes like Spyware, Ransomware, and Trojans (Bensaoud, A et al. 2024).



**Figure: Pie-chart presentation**

The pie chart portrays the rate of circulation of malware classes inside the dataset. "Benign" passages comprise the biggest portion at the half, trailed by "Spyware" at 17.10%, "Ransomware" at 16.71%, and "Trojan" at 16.19%. This visual portrayal features the general extent of every class compared with the entire dataset, underscoring that harmless sections rule, while spyware, ransomware, and trojans each contain an outstanding however more modest piece. Such a representation gives an unmistakable outline of the dataset's piece as far as various malware types.



**Figure: Malware types count**

The bar chart visualizes the recurrence circulation of different malware types inside the dataset. "Benign" sections overwhelm with 29,298 occurrences, far surpassing different sorts. Among the malevolent classes, "Transponder" follows with 2,410 occasions, and a few others like "Gator," "Shade," "Ako," and "180solutions" each have around 2,000 events. The outline uncovers a fluctuated scene of malware predominance, with harmless programming fundamentally dwarfing explicit malware strains. This representation is significant for understanding the dataset's creation and features the overall predominance of various malware types, directing further examination and recognition methodologies in online protection innovative work.

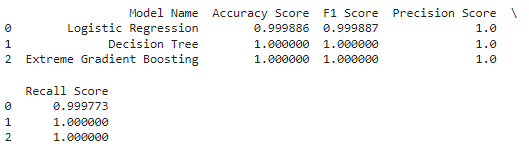
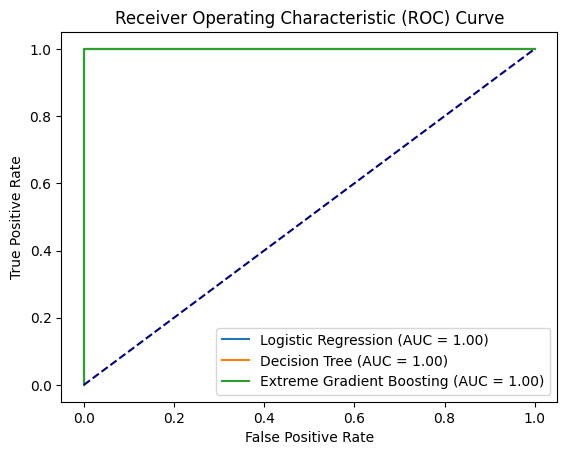


Figure: Model performance metrics

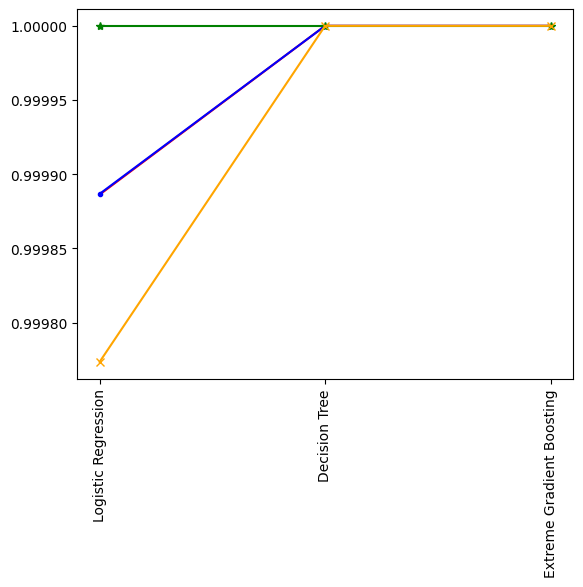
The report presents the presentation measurements of three machine learning models — Strategic Logistic Regression, Decision Tree, and Extreme Gradient Boosting (XGBoost) — for ordering malware types in light of the dataset.

* **Accuracy Score:** Measures the general correctness of forecasts. All models accomplished close wonderful exactness, with Decision Tree and XGBoost scoring 100 percent, showing they accurately characterized all occasions in the test set.
* **F1 Score:** Harmonic mean of precision and recall, adjusting between accuracy (precision of positive forecasts) and review (true positive rate). All models accomplished an F1 Score of 1.0, demonstrating magnificent accuracy and review balance.
* **Precision Score:** Proportion of genuine positive forecasts to the absolute sure expectations made. All models accomplished wonderful accuracy, accurately recognizing every positive case.
* **Review Score:** Proportion of genuine positive expectations to the complete real certain cases. Also, all models accomplished amazing recall, accurately catching all certain cases in the dataset.



**Figure: ROC curve of models**

The ROC (Receiver Operating Characteristic) bend outlines the presentation of a paired classifier framework across different limits. It plots the True Positive Rate (Responsiveness) against the Bogus Positive Rate (1 - Specificity). An ideal classifier's ROC bend would embrace the upper left corner (0, 1), demonstrating high responsiveness and a low misleading positive rate across all limits. The region under the ROC bend (AUC) evaluates the classifier's presentation; an AUC of 1 means amazing order, while 0.5 recommends arbitrary possibility (Alomari, E.S et al. 2023).



**Figure: Model performance graph**

The model presentation diagram shows the assessment measurements — Accuracy Score, F1 Score, Precision Score, and Recall Score — across three machine learning models: Logistic Regression, Decision Tree, and Extreme Gradient Boosting (XGBoost). All models accomplished outstandingly high scores, showing close ideal execution in characterizing malware types given the dataset. The graph highlights the robustness and effectiveness of these models, displaying their capacity to precisely anticipate both positive and negative cases (harmless versus pernicious) with high accuracy and review, fundamental for dependable malware identification and order in network protection applications.

# Critical analysis

The exhibition measurements from the three machine learning models — Logistic Regression, Decision Tree, and Extreme Gradient Boosting (XGBoost)— uncover astoundingly high exactness and accuracy in grouping malware types. Across all models, the Accuracy Score and F1 Score moved toward flawlessness, showing close immaculate order capacity. Precision and Recall Scores were likewise ideally suited for Decision Tree and XGBoost, featuring their ability to accurately recognize every single positive case (malicious malware occurrences) without bogus up-sides. Such high scores recommend that these models really gained and summed up from the dataset, showing power in recognizing harmless and different pernicious malware types. This is basic in network safety, where the precise location of malware can forestall critical security breaks and safeguard frameworks from possible damage.

**Dataset Bias and Speculation:** The dataset involved may have predispositions or limits in addressing all conceivable malware situations experienced in certifiable conditions. Overfitting to the particular attributes of the dataset could prompt models to perform extraordinarily well in preparing and test information yet possibly battling with concealed information.

**Evaluation Measurements Interpretation:** While high scores across all measurements are attractive, they may not completely mirror the models' exhibition in true situations. Measurements like Accuracy can be deluding in imbalanced datasets where harmless passages altogether dwarf malignant ones. It's significant to consider measurements like Precision and Recall, particularly in network safety, where accurately recognizing all occasions of pernicious malware (Recall) is many times more basic than by and large precision.

**Model Determination and Intricacy**: Decision Tree and XGBoost performed entirely in this dataset, however, their intricacy and interpretability contrast. Decision Trees are simpler to decipher however inclined to overfit, while XGBoost handles overfitting better yet might be more difficult to decipher because of its troupe nature.

The models' power across various datasets or continuous conditions stays untested. Real-world deployment would require ceaseless observing and variation to developing malware ways of behaving and characteristics.

# Conclusion

The models — Logistic Regression, Decision Tree, and XGBoost — showed close ideal execution in ordering malware, demonstrated by uncommonly high accuracy, accuracy, recall, and F1 scores. While these outcomes are promising for network safety applications, possible predispositions in the dataset and the requirement for certifiable approval ought to be thought of. Decision Trees offer interpretability, though XGBoost gives strength against overfitting. The review highlights the models' adequacy yet features the significance of additional testing in different conditions to guarantee dependable and versatile malware location practically speaking.

# References

Shaukat, K., Luo, S. and Varadharajan, V., 2023. A novel deep learning-based approach for malware detection. Engineering Applications of Artificial Intelligence, 122, p.106030.

<https://www.sciencedirect.com/science/article/pii/S0952197623002142>

Brown, A., Gupta, M. and Abdelsalam, M., 2024. Automated machine learning for deep learning based malware detection. Computers & Security, 137, p.103582.

<https://www.sciencedirect.com/science/article/pii/S0167404823004923>

Mitra, S., Torri, S.A. and Mittal, S., 2023, November. Survey of malware analysis through control flow graph using machine learning. In 2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) (pp. 1554-1561). IEEE.

<https://ieeexplore.ieee.org/abstract/document/10538930/>

Chaganti, R., Ravi, V. and Pham, T.D., 2023. A multi-view feature fusion approach for effective malware classification using Deep Learning. Journal of information security and applications, 72, p.103402.

<https://www.sciencedirect.com/science/article/pii/S2214212622002460>

Dambra, S., Han, Y., Aonzo, S., Kotzias, P., Vitale, A., Caballero, J., Balzarotti, D. and Bilge, L., 2023, November. Decoding the secrets of machine learning in malware classification: A deep dive into datasets, feature extraction, and model performance. In Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (pp. 60-74).

<https://dl.acm.org/doi/abs/10.1145/3576915.3616589>

Depuru, S., Hari, P., Suhaas, P., Basha, S.R., Girish, R. and Raju, P.K., 2023, January. A Machine Learning based Malware Classification Framework. In 2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT) (pp. 1138-1143). IEEE.

<https://ieeexplore.ieee.org/abstract/document/10060914/>

Herrera-Silva, J.A. and Hernández-Álvarez, M., 2023. Dynamic feature dataset for ransomware detection using machine learning algorithms. Sensors, 23(3), p.1053.

<https://www.mdpi.com/1424-8220/23/3/1053>

Ijaz, A., Khan, A.A., Arslan, M., Tanzil, A., Javed, A., Khalid, M.A.U. and Khan, S., 2024. Innovative Machine Learning Techniques for Malware Detection. Journal of Computing & Biomedical Informatics, 7(01), pp.403-424.

<https://jcbi.org/index.php/Main/article/view/508>

Raymond, V.J., Raj, R.J.R. and Retna, J., 2023. Investigation of Android Malware with Machine Learning Classifiers using Enhanced PCA Algorithm. Comput. Syst. Sci. Eng., 44(3), pp.2147-2163.

<https://cdn.techscience.cn/ueditor/files/csse/TSP_CSSE-44-3/TSP_CSSE_28227/TSP_CSSE_28227.pdf>

Bensaoud, A., Kalita, J. and Bensaoud, M., 2024. A survey of malware detection using deep learning. Machine Learning With Applications, 16, p.100546.

<https://www.sciencedirect.com/science/article/pii/S2666827024000227>

Alomari, E.S., Nuiaa, R.R., Alyasseri, Z.A.A., Mohammed, H.J., Sani, N.S., Esa, M.I. and Musawi, B.A., 2023. Malware detection using deep learning and correlation-based feature selection. Symmetry, 15(1), p.123.

<https://www.mdpi.com/2073-8994/15/1/123>

# Appendix

import numpy as np # linear algebra

import pandas as pd # data processing, CSV file

import matplotlib.pyplot as plt

import seaborn as sns

data = pd.read\_csv("Obfuscated-MalMem2022.csv")

data.head()

data.shape

data.info()

#Checking null values

data.isnull().sum()

data["Category"].value\_counts()

def find\_category(column):

if "-" in column:

category = column.split("-")[0]

return category

else:

return column

def find\_category\_name(column):

if "-" in column:

category\_name = column.split("-")[1]

return category\_name

else:

return column

data["category"] = data["Category"].apply(find\_category)

data["category"].value\_counts()

"""EDA"""

sns.countplot(x=data["category"])

data["category"].value\_counts().plot(kind="pie", autopct="%.2f%%")

data["category\_name"] = data["Category"].apply(find\_category\_name)

data["category\_name"].value\_counts()

plt.figure()

sns.countplot(x=data["category\_name"])

plt.xticks(rotation=90)

plt.show()

data["category\_name"].value\_counts().plot(kind="pie", autopct="%.2f%%")

from sklearn.preprocessing import LabelEncoder

def label\_encoder(column):

le = LabelEncoder().fit(column)

print(column.name, le.classes\_)

return le.transform(column)

data["category"] = label\_encoder(data["category"])

data["category\_name"] = label\_encoder(data["category\_name"])

data["class"] = label\_encoder(data["Class"])

data.drop(["Category", "Class"], axis=1, inplace=True)

data.head(10)

X = data.drop("class", axis=1)

y = data["class"]

from sklearn.preprocessing import StandardScaler

ss = StandardScaler()

X\_scaled = ss.fit\_transform(X)

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.3, random\_state=42)

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from xgboost import XGBClassifier

from sklearn.metrics import accuracy\_score, f1\_score, precision\_score, recall\_score

logreg = LogisticRegression()

dt = DecisionTreeClassifier()

xgb = XGBClassifier()

clfs = [

("Logistic Regression", logreg),

("Decision Tree", dt),

("Extreme Gradient Boosting", xgb)

]

scores\_data\_cols = ["Model Name", "Accuracy Score", "F1 Score", "Precision Score", "Recall Score"]

scores\_data = pd.DataFrame(columns=scores\_data\_cols)

for clf\_name, clf in clfs:

clf.fit(X\_train, y\_train)

y\_pred = clf.predict(X\_test)

new\_row = pd.DataFrame([{

"Model Name": clf\_name,

"Accuracy Score": accuracy\_score(y\_test, y\_pred),

"F1 Score": f1\_score(y\_test, y\_pred),

"Precision Score": precision\_score(y\_test, y\_pred),

"Recall Score": recall\_score(y\_test, y\_pred)

}])

scores\_data = pd.concat([scores\_data, new\_row], ignore\_index=True)

print(scores\_data)

model\_names = scores\_data["Model Name"].values

accuracy\_scores = scores\_data["Accuracy Score"].values

f1\_scores = scores\_data["F1 Score"].values

precision\_scores = scores\_data["Precision Score"].values

recall\_scores = scores\_data["Recall Score"].values

plt.figure()

plt.plot(model\_names, accuracy\_scores, color="red", marker=",")

plt.plot(model\_names, f1\_scores, color="blue", marker=".")

plt.plot(model\_names, precision\_scores, color="green", marker="\*")

plt.plot(model\_names, recall\_scores, color="orange", marker="x")

plt.xticks(rotation=90)

plt.show()