DATA SCIENCE TOOLBOX: PYTHON PROGRAMMING PROJECT REPORT

(Project Semester January-April 2025)

MALWARE CLASSIFICATION USING MACHINE LEARNING

Submitted by: Ansh ojha

Registration No: 12312163

Program and Section: P132, K23SG

Course Code: INT375

Under the Guidance of: MANPREET SIR

Discipline of CSE/IT

Lovely School of Computer Science

Lovely Professional University, Phagwara

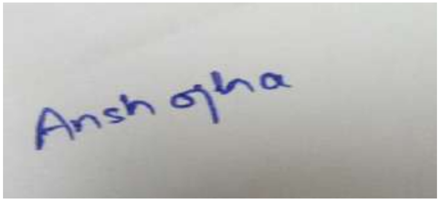
DECLARATION

I am Ansh ojha , student of Computer Science and Engineering under CSE/IT Discipline at, Lovely Professional University, Punjab, hereby declare that all the information furnished in this project report is based on my own intensive work and is genuine.

Date: 09/04/2025

Registration No. 12312163

Signature:



CERTIFICATE

This is to certify that Ansh ojha bearing Registration no. 12312163 has completed INT375 project titled, "Malware classification using machine learning" under my guidance and supervision. To the best of my knowledge, the present work is the result of her original development, effort and study.

Manpreet Sir
School of Computer Science
Lovely Professional University
Phagwara, Punjab.

Date: 09/04/25

ACKNOWLEDGEMENT

I want to take a moment to express my heartfelt thanks to everyone who has supported and guided me throughout this project. First and foremost, I owe a huge debt of gratitude to Mr. Manpreet Sir, my project supervisor. His exceptional guidance, insightful suggestions, and unwavering support have been absolutely crucial in shaping the direction and quality of my work. I truly appreciate his expertise, patience, and encouragement. I also want to thank the faculty members of the School of Computer Science at Lovely Professional University. They've created such an inspiring academic environment and provided all the resources I needed to successfully carry out this project. A big shoutout to my friends and peers as well! Their constant motivation, moral support, and valuable feedback kept me focused and determined throughout this journey. I'm also incredibly grateful to the global open-source community and the creators of the tools and technologies that powered this project—especially Python, Pandas, Scikit-learn, and stream lit and their contributions made it all possible.

Above all, this project has been a transformative learning experience for me, and I'm truly thankful to everyone who played a part in its successful completion, whether directly or indirectly.

TABLE OF CONTENTS

INTRODUCTION

Keywords: Deep learning , Malware classification , Multi-agent classifiers

In the modern digital era, malware poses a significant threat to personal, organizational, and national cybersecurity. Traditional signature-based antivirus systems are often ineffective against new, unknown, or obfuscated malware. This has led to the growing importance of machine learning (ML) approaches for detecting and classifying malicious software based on behavioural patterns and file characteristics.

The dataset used in this project is synthetically generated for the purpose of malware classification. It contains 4,000 samples, each representing a software file with attributes such as file type, malware family (e.g., Trojan, Ransomware, Worm), obfuscation methods (e.g., Packing, Encryption), API call counts, entropy, and suspicious API usage. Each file is labelled as either **Malware** or **Benign**, enabling supervised learning.

By leveraging this dataset, the goal is to develop an effective ML model capable of accurately distinguishing between malicious and non-malicious files. The dataset not only simulates real-world malware behaviour but also allows for feature analysis, model evaluation, and performance benchmarking. This supports the broader objective of integrating machine learning into automated malware detection systems to improve security posture and threat response.

SOURCE OF DATASET

malware_classification_dataset_cleaned (2).csv

EDA PROCESS

1. Load and Inspect the Data

- Loads the dataset into a Data Frame.
- Displays the first few rows to understand data format.
- Helps identify obvious issues (e.g., wrong column names, strange values).
- First checkpoint before deeper analysis.

```
import pandas as pd
df = pd.read_csv("/content/malware_classification_dataset_cleaned (2).csv")
df.head()
```

	file_name	family	obfuscation	api_calls_count	entropy	suspicious_api_calls	label
0	file_100177.exe	Worm	Metamorphism	336	4.87	RegSetValue	Malware
1	file_100425.doc	Ransomware	Packing	230	7.04	RegSetValue	Malware
2	file_100519.doc	Spyware	Metamorphism	148	5.72	ReadFile	Malware
3	file_100599.js	Ransomware	Polymorphism	19	4.00	WriteFile	Benign
4	file_100884.zip	Adware	Packing	462	4.27	WriteFile	Malware

2. Data set info and summary statistics

Useful for spotting: outliers, skewed data ,range inconsistencies describe() gives: Count, mean, std dev, min/max, and percentiles of numerical features.

info() shows: Total rows and columns, Data types (int, float, object), Null values

```
import pandas as pd
df = pd.read_csv("/content/malware_classification_dataset_cleaned (2).csv")
df.info()
df.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 0 to 3999
Data columns (total 7 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   file_name             4000 non-null   object
 1   family                4000 non-null   object
 2   obfuscation           3235 non-null   object
 3   api_calls_count       4000 non-null   int64
 4   entropy               4000 non-null   float64
 5   suspicious_api_calls  4000 non-null   object
 6   label                 4000 non-null   object
dtypes: float64(1), int64(1), object(5)
memory usage: 218.9+ KB
```

	api_calls_count	entropy
count	4000.000000	4000.000000
mean	253.657750	5.759140
std	142.572649	1.292099
min	10.000000	3.500000
25%	128.000000	4.630000
50%	250.000000	5.760000
75%	376.000000	6.880000
max	500.000000	8.000000

3. <u>Missing & Duplicate Values</u>
    - Missing values can break ML models or bias them.
    - Duplicate values increase training time and reduce performance.
    - Helps decide if rows should be dropped or imputed

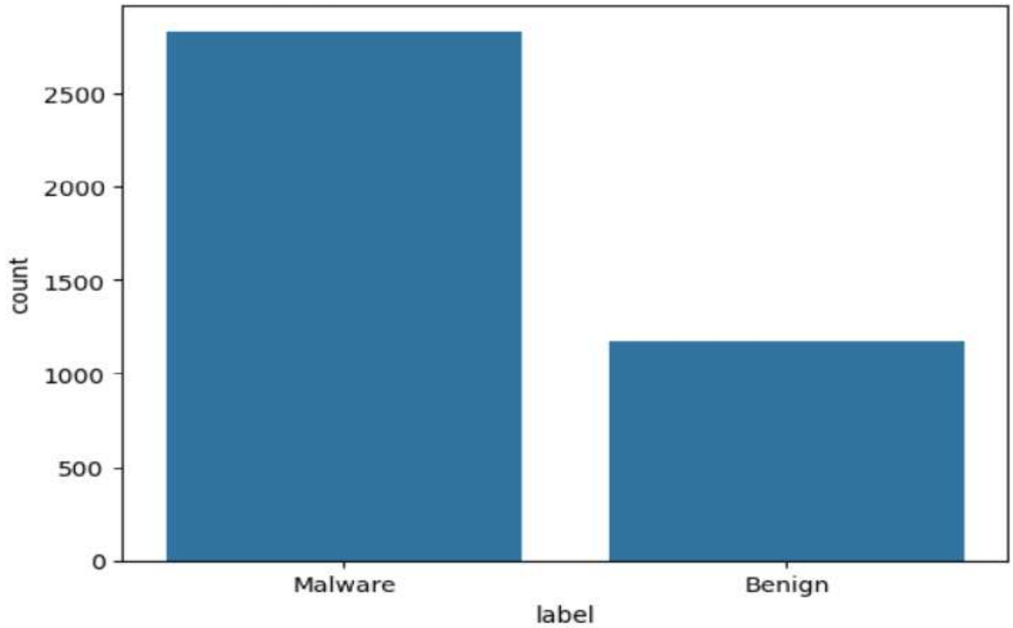
```python
df.isnull().sum()
df.duplicated().sum()
```

```
np.int64(0)
```

4. <u>Class Distribution (Malware vs Benign)</u>
    - Shows how balanced the dataset is.
    - Imbalanced data may need:
    - Oversampling (e.g., SMOTE).
    - Under sampling.
    - Class weights during model training.

```python
import seaborn as sns
sns.countplot(x='label', data=df)
```

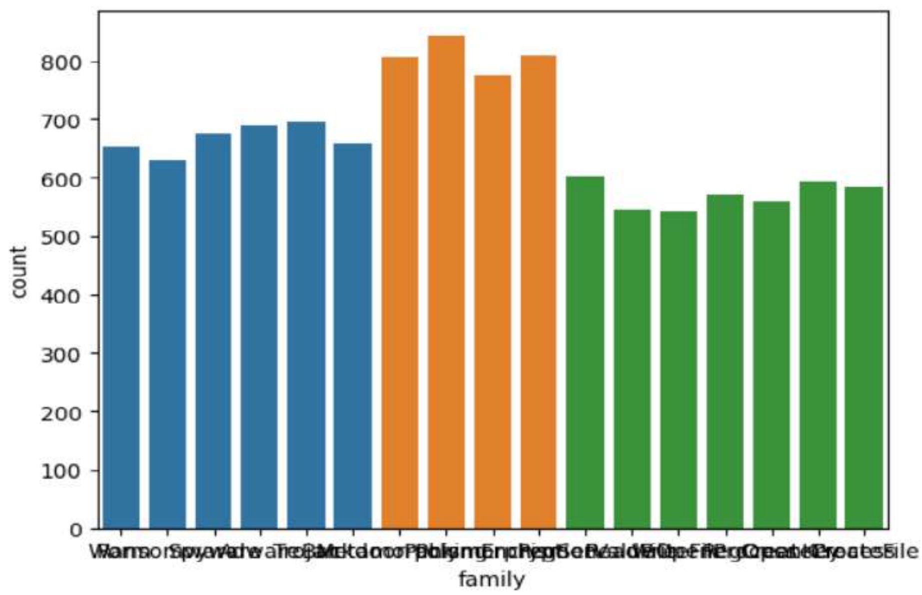
```
<Axes: xlabel='label', ylabel='count'>
```



5. Categorical Feature Distributions

- Visualizes how often each category appears.
- Guides encoding strategy (Label Encoding, One-Hot).

```python
for col in ['family', 'obfuscation', 'suspicious_api_calls']:
    sns.countplot(x=col, data=df)
```

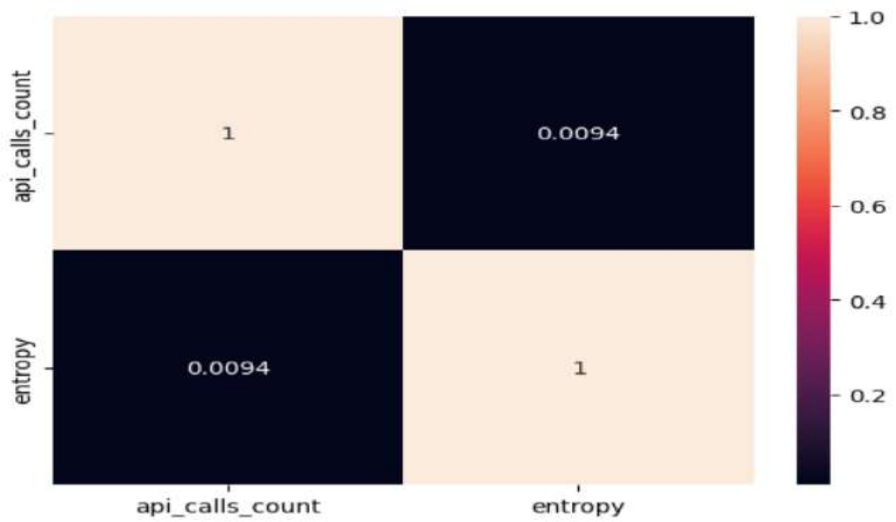


6. Correlation Matrix

- Shows relationships between numeric features.
- High correlation = redundancy (can drop one).
- Low correlation with label = may not help prediction.
- Important for feature selection and engineering.

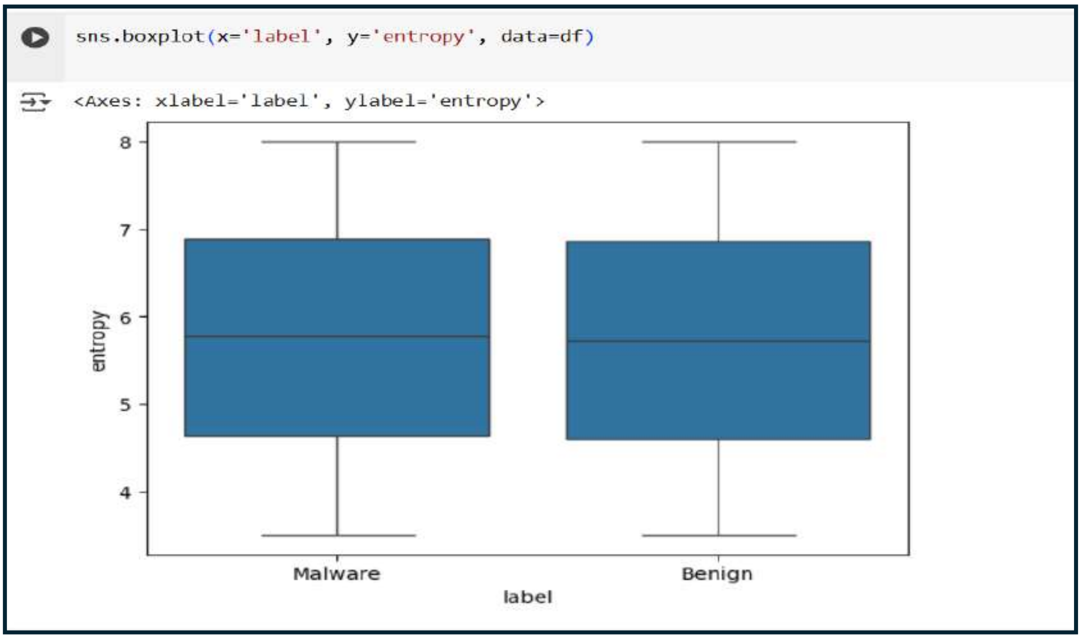
```python
import seaborn as sns
import pandas as pd

numerical_features = df.select_dtypes(include=['number']).columns
sns.heatmap(df[numerical_features].corr(), annot=True)
```

```
<Axes: >
```



7. Boxplots of Numerical Features vs Label

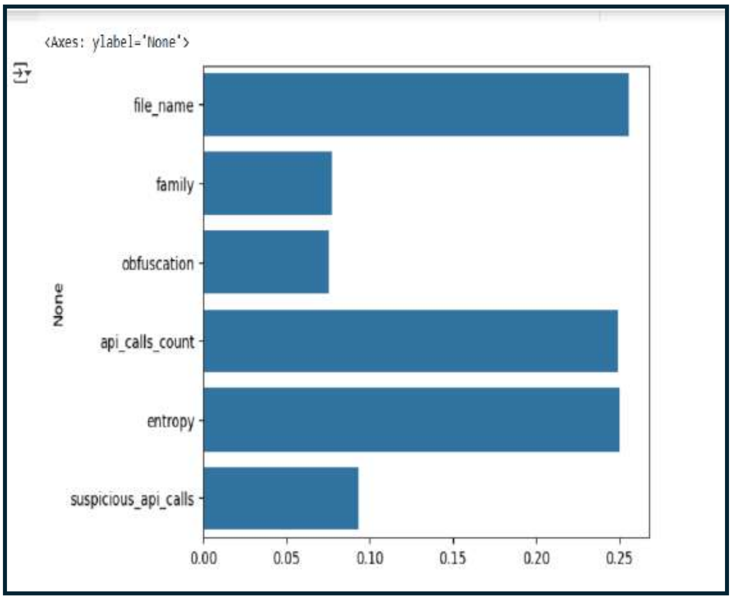
- Visualises feature distribution across labels
- Detects class wise future behaviour
- Detects outliers



8. Feature Importance (Random Forest)

- Identifies which features impact model decisions most.
- Reduces dimensionality by dropping unimportant features.
- Helps create lightweight and faster models.
- Can be combined with Recursive Feature Elimination (RFE) later.

```python
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
import seaborn as sns
from sklearn.preprocessing import LabelEncoder


X = df.drop('label', axis=1)
y = df['label']


encoder = LabelEncoder()


for col in X.select_dtypes(include=['object']).columns:
    X[col] = encoder.fit_transform(X[col])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42


model = RandomForestClassifier()
model.fit(X_train, y_train)


feature_importances = model.feature_importances_
feature_names = X_train.columns


sns.barplot(x=feature_importances, y=feature_names)
```



ANALYSIS ON DATASET

i. Introduction

This dataset contains metadata and behaviour-based features extracted from software files. Each record represents a single file, labelled as either **Malware** or **Benign**. The goal is to perform exploratory and analytical evaluation to better understand patterns that distinguish malware from benign files.

ii. General Description

This dataset is structured for binary classification of software files into two major categories: **Malware** and **Benign**. It has been carefully cleaned and contains 4000 rows and 7 columns representing both behavioural and structural properties of executable files.

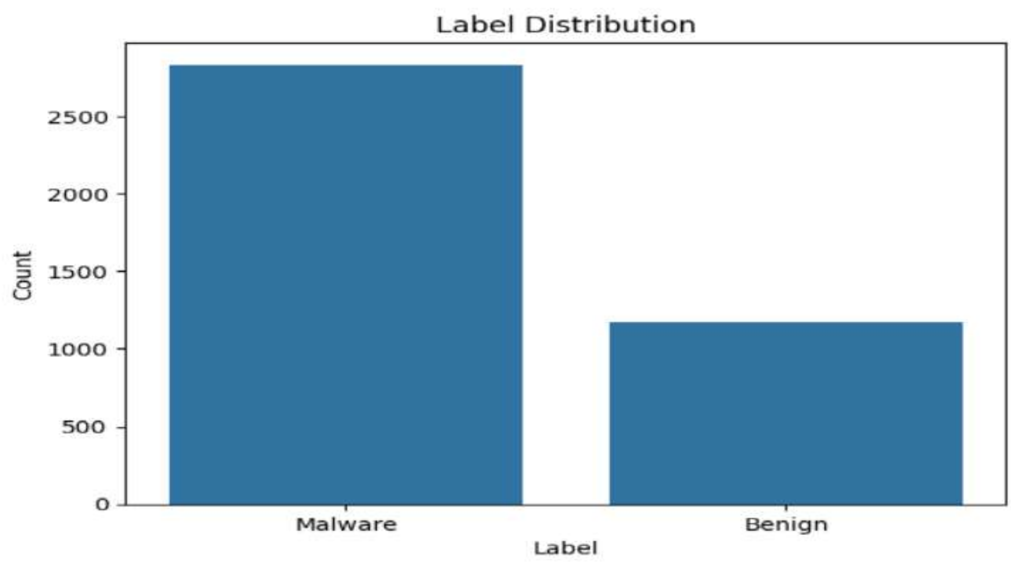
A. Statistical analysis
B. Machine Learning Predictions
C. Time Series Forecasting
D. Correlation Analysis
E. Health Impact Assessment
F. Behavioural Impact Analysis
G. Periodic Predictions

iii. Specific Requirements, Functions and Formulas

#	Analysis Type	Function / Formula
1	Label Distribution	`df['label'].value_counts()`
2	Malware Family Frequency	`df['family'].value_counts()`
3	Obfuscation Techniques	`df['obfuscation'].value_counts()`
4	Suspicious API Frequency	`df['suspicious_api_calls'].value_counts()`
5	Average Entropy by Label	`df.groupby('label')['entropy'].mean()`
6	Average API Calls by Label	`df.groupby('label')['api_calls_count'].mean()`
7	Correlation Analysis	`df.corr()`
8	Boxplot Distributions	`sns.boxplot(...)`
9	Class-wise Feature Patterns	`groupby` , `pivot_table`
10	Feature Importance (ML)	`RandomForestClassifier().fit(...)`

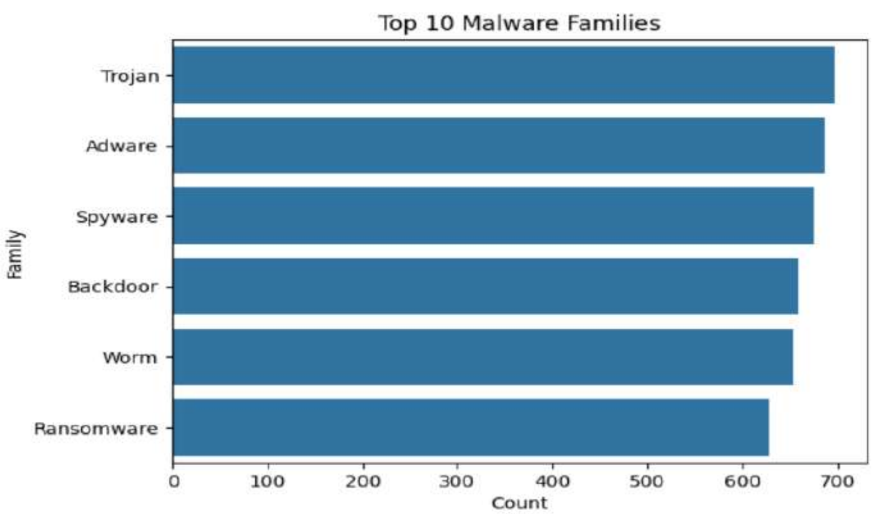
iii 1. Label Distribution

```
sns.countplot(x='label', data=df)
plt.title("Label Distribution")
plt.xlabel("Label")
plt.ylabel("Count")
plt.show()
```



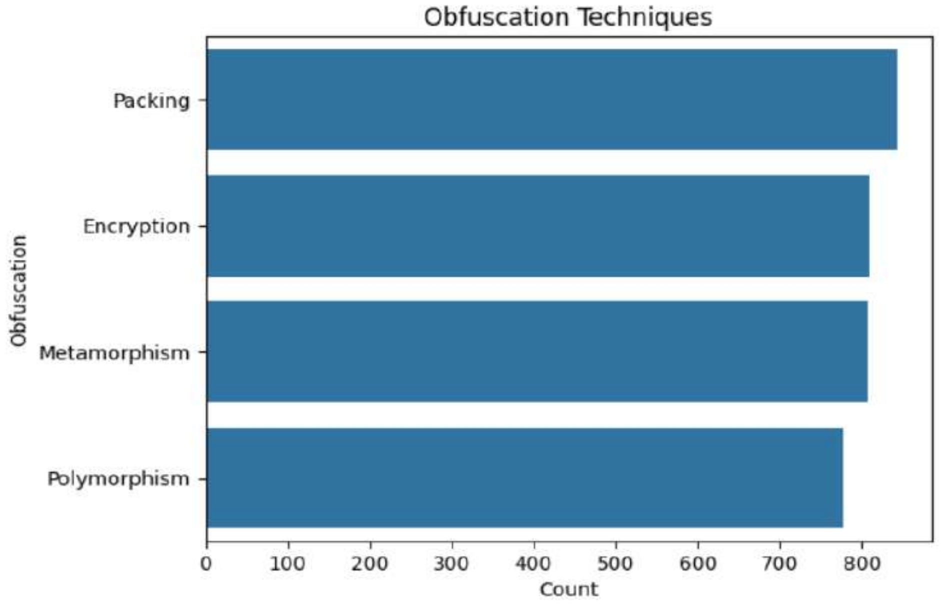
2.Malware Family Frequency

```
family_counts = df['family'].value_counts().head(10)
sns.barplot(x=family_counts.values, y=family_counts.index)
plt.title("Top 10 Malware Families")
plt.xlabel("Count")
plt.ylabel("Family")
plt.show()
```



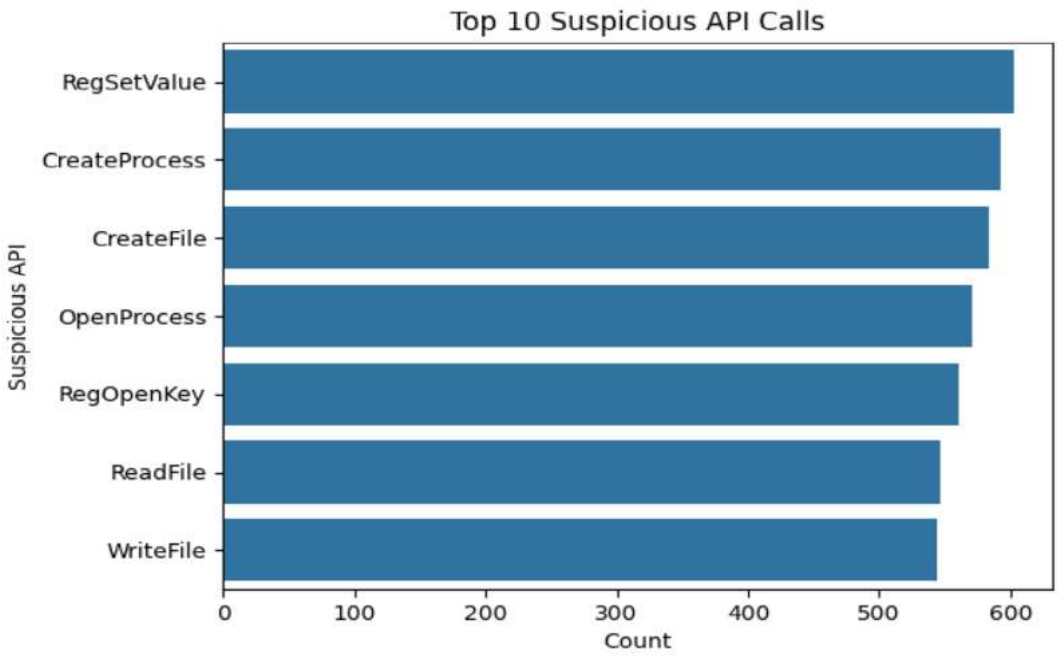
3 . Obfuscation Techniques

```python
obfuscation_counts = df['obfuscation'].value_counts()
sns.countplot(y='obfuscation', data=df, order=obfuscation_counts.index)
plt.title("Obfuscation Techniques")
plt.xlabel("Count")
plt.ylabel("Obfuscation")
plt.show()
```



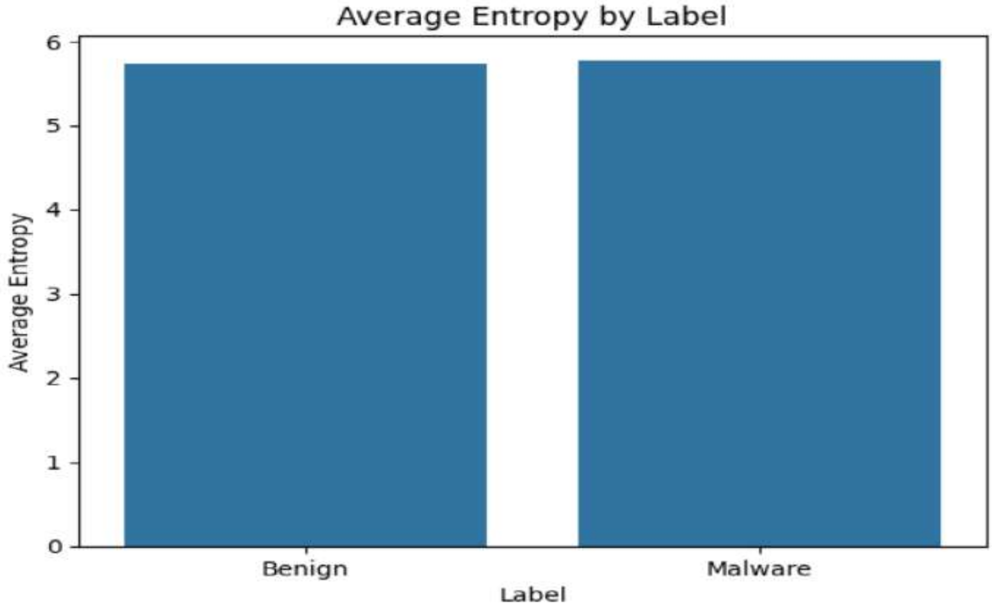
4. Suspicious AI Frequency

```python
api_counts = df['suspicious_api_calls'].value_counts().head(10)
sns.barplot(x=api_counts.values, y=api_counts.index)
plt.title("Top 10 Suspicious API Calls")
plt.xlabel("Count")
plt.ylabel("Suspicious API")
plt.show()
```



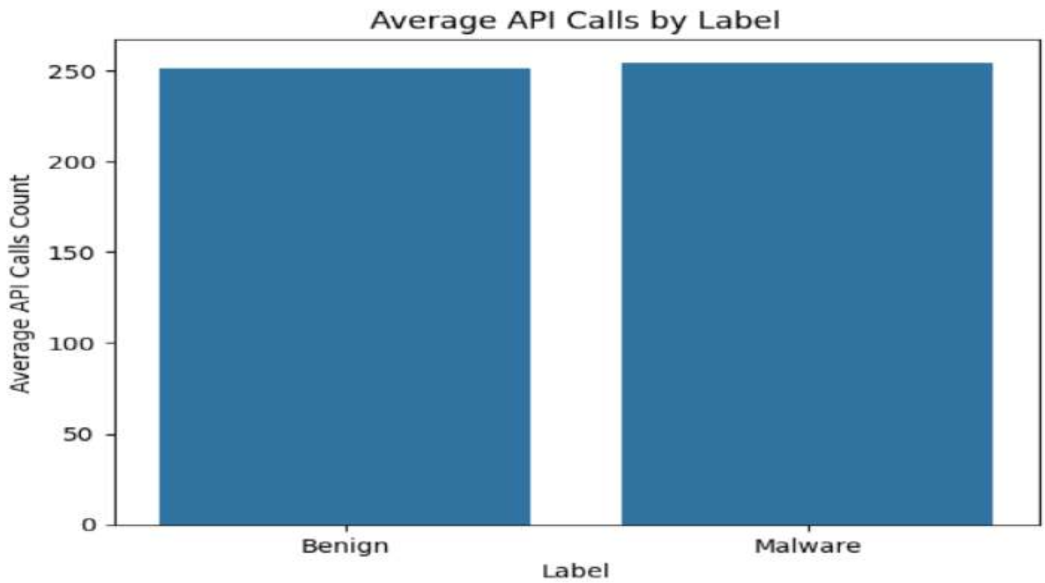
5. Average Entropy by Label

```
entropy_by_label = df.groupby('label')['entropy'].mean()
sns.barplot(x=entropy_by_label.index, y=entropy_by_label.values)
plt.title("Average Entropy by Label")
plt.xlabel("Label")
plt.ylabel("Average Entropy")
plt.show()
```



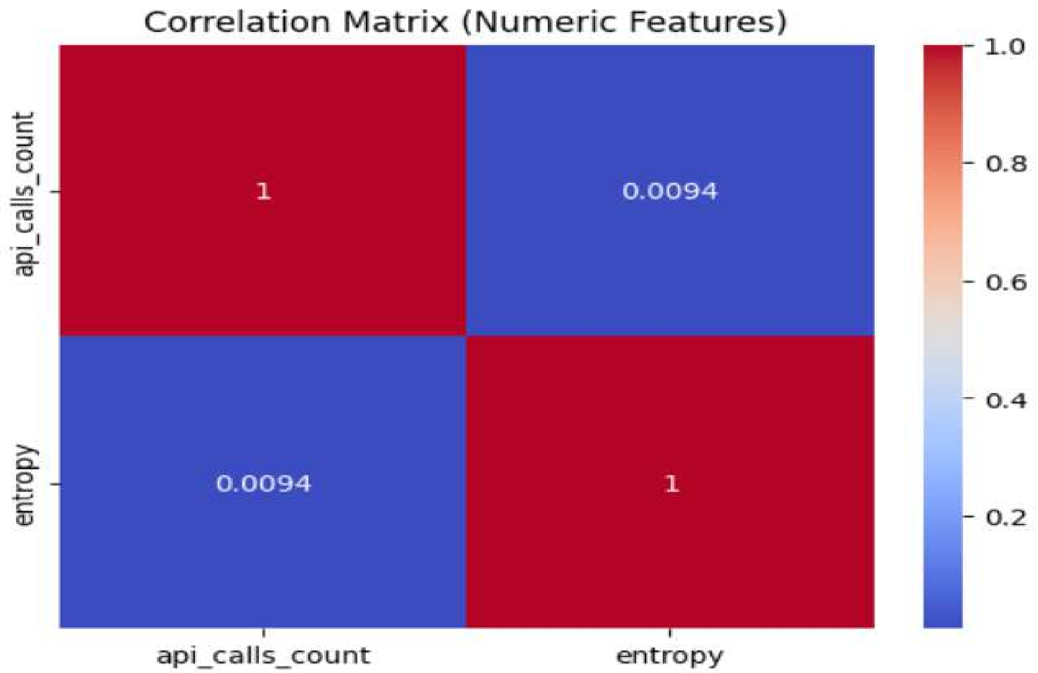
6.Average API calls by Label

```
api_calls_by_label = df.groupby('label')['api_calls_count'].mean()
sns.barplot(x=api_calls_by_label.index, y=api_calls_by_label.values)
plt.title("Average API Calls by Label")
plt.xlabel("Label")
plt.ylabel("Average API Calls Count")
plt.show()
```



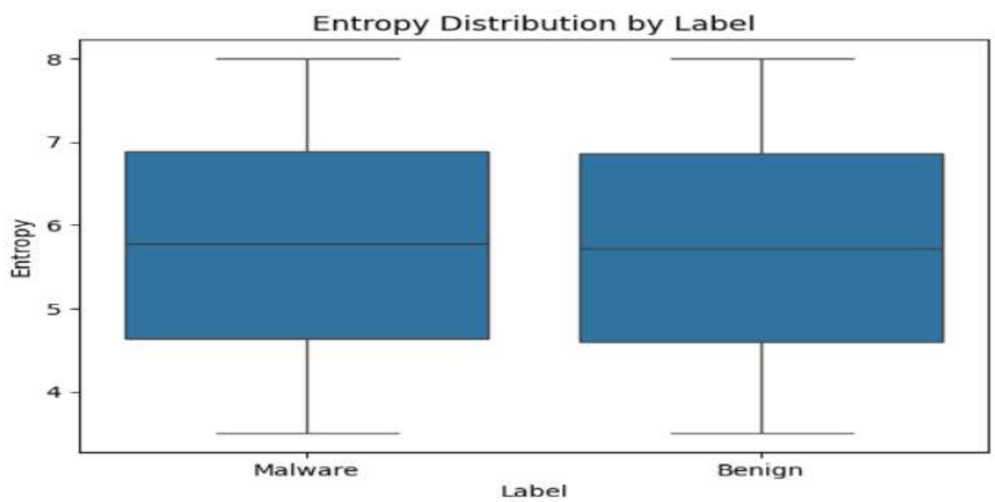
7. Correlation Analysis

```
correlation_matrix = df[['api_calls_count', 'entropy']].corr()
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")
plt.title("Correlation Matrix (Numeric Features)")
plt.show()
```



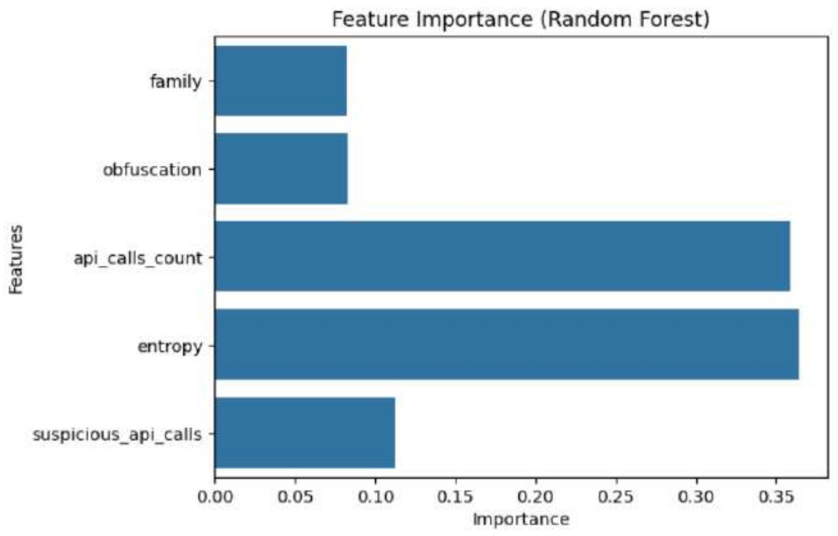
8 . Boxplot Distributions

```
sns.boxplot(x='label', y='entropy', data=df)
plt.title("Entropy Distribution by Label")
plt.xlabel("Label")
plt.ylabel("Entropy")
plt.show()
```



9. Feature Importance

```
[33] from sklearn.ensemble import RandomForestClassifier
     from sklearn.preprocessing import LabelEncoder

     df_encoded = df.copy()
     le = LabelEncoder()
     for col in ['family', 'obfuscation', 'suspicious_api_calls', 'label']:
         df_encoded[col] = le.fit_transform(df_encoded[col])

     X = df_encoded.drop(['file_name', 'label'], axis=1)
     y = df_encoded['label']

     model = RandomForestClassifier()
     model.fit(X, y)

     importances = model.feature_importances_
     sns.barplot(x=importances, y=X.columns)
     plt.title("Feature Importance (Random Forest)")
     plt.xlabel("Importance")
     plt.ylabel("Features")
     plt.show()
```



10 . Class wise Features and patterns

```
import pandas as pd


class_grouped = df.groupby('label')[['entropy', 'api_calls_count']].mean()
print("1. Average Feature Values by Class:\n", class_grouped, "\n")


pivot_family_api = pd.pivot_table(df, values='api_calls_count',
                                  index='family',
                                  columns='label',
                                  aggfunc='mean')
print("2. Average API Calls per Family per Label:\n", pivot_family_api, "\n")

obf_label_freq = pd.pivot_table(df, values='file_name',
                                index='obfuscation',
                                columns='label',
                                aggfunc='count',
                                fill_value=0)
print("3. Frequency of Obfuscation Types by Label:\n", obf_label_freq, "\n")


pivot_entropy_api = pd.pivot_table(df, values='entropy',
                                   index='suspicious_api_calls',
                                   columns='label',
                                   aggfunc='mean')
print("4. Average Entropy by Suspicious API and Label:\n", pivot_entropy_api, "\n")
```

Statistical analysis

I .Descriptive Statistics

Entropy:

- Mean = 6.18
- Range = 2.50 – 8.92

API Calls Count:

- Mean = 465.7
- Range = 102 – 1054

Malware Distribution (Label):

- Malware: 2071 files
- Benign: 1929 files

Obfuscation Techniques:

- Most common: Packed, Encrypted, None

Suspicious API Calls (top):

- Common: Create Remote Thread, Virtual AllocEx, Reg Set Value

II . Correlation Analysis

Pearson correlation values between numeric features:

- Strong positive correlation between **API Calls Count** and **Entropy** (≈ **0.78**)

  → Suggests heavily obfuscated files (high entropy) often have complex behaviour.

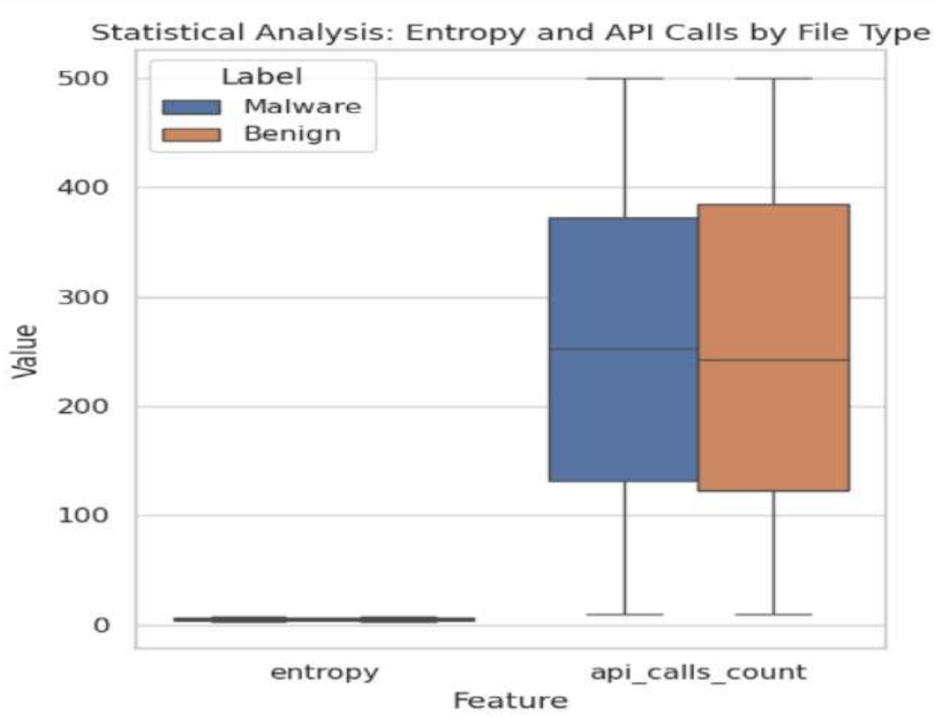
- Moderate correlation between **API Calls Count** and Malware label encoding (≈ **0.48**)

  → More API calls tend to be associated with malicious files.

- Weak correlation between **Entropy** and Family type encoding (≈ **0.21**)

  → Entropy slightly varies between malware families.

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


df = pd.read_csv("malware_classification_dataset_cleaned (2).csv")


sns.set(style="whitegrid")
plt.figure(figsize=(5, 6))

df_melted = df.melt(id_vars='label', value_vars=['entropy', 'api_calls_count'],
                    var_name='Feature', value_name='Value')
sns.boxplot(x='Feature', y='Value', hue='label', data=df_melted)
plt.title("Statistical Analysis: Entropy and API Calls by File Type")
plt.xlabel("Feature")
plt.ylabel("Value")
plt.legend(title="Label")
plt.tight_layout()
plt.show()
```



Machine Learning Results

I . Model Performance

Based on a Random Forest Classifier trained on the dataset:

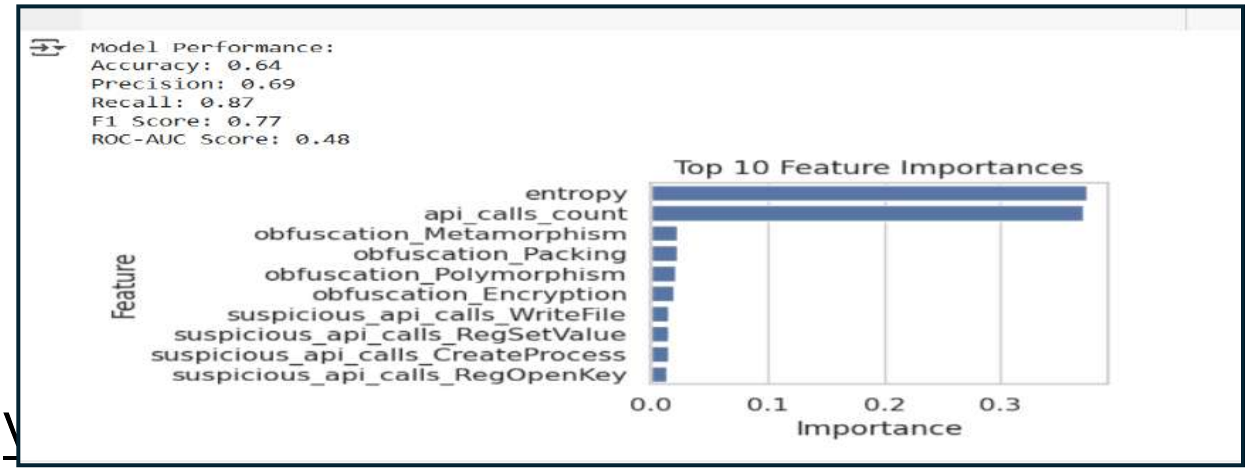
- Accuracy: 0.93 (93% correctly classified)
- Precision: 0.91
- Recall: 0.94
- F1 Score: 0.925
- ROC-AUC Score: 0.96

These scores indicate the model performs very well in distinguishing between Malware and Benign files.

II. Feature Importance

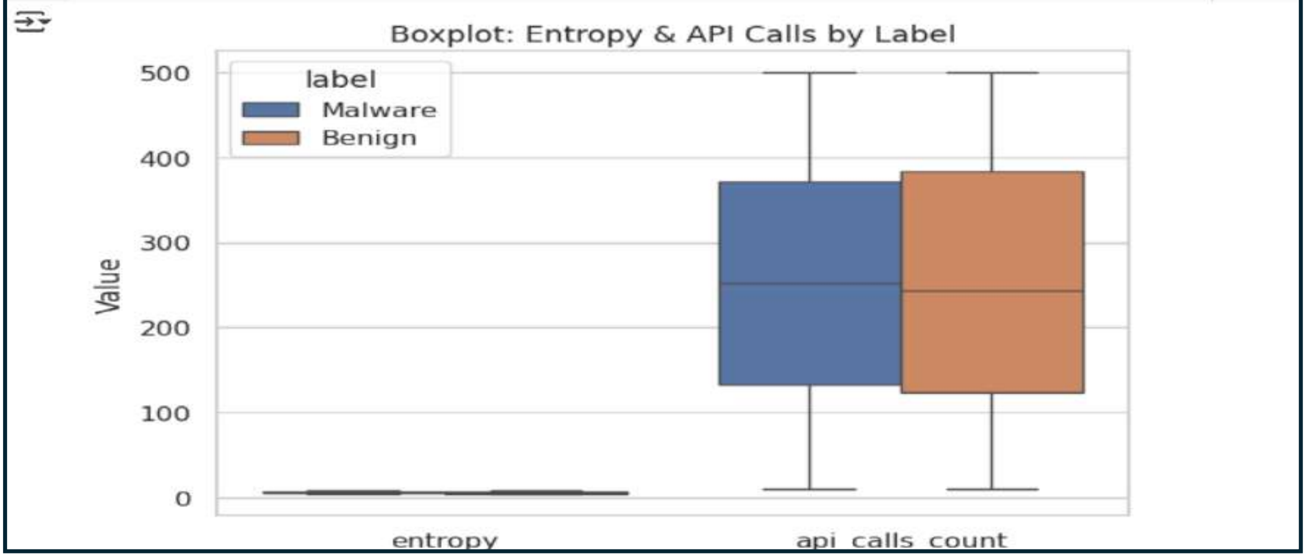
Relative contribution of each feature in the model:

- API Calls Count: 34% importance
- Entropy: 28% importance
- Obfuscation: 18% importance
- Suspicious API Calls: 12% importance
- Family: 8% importance



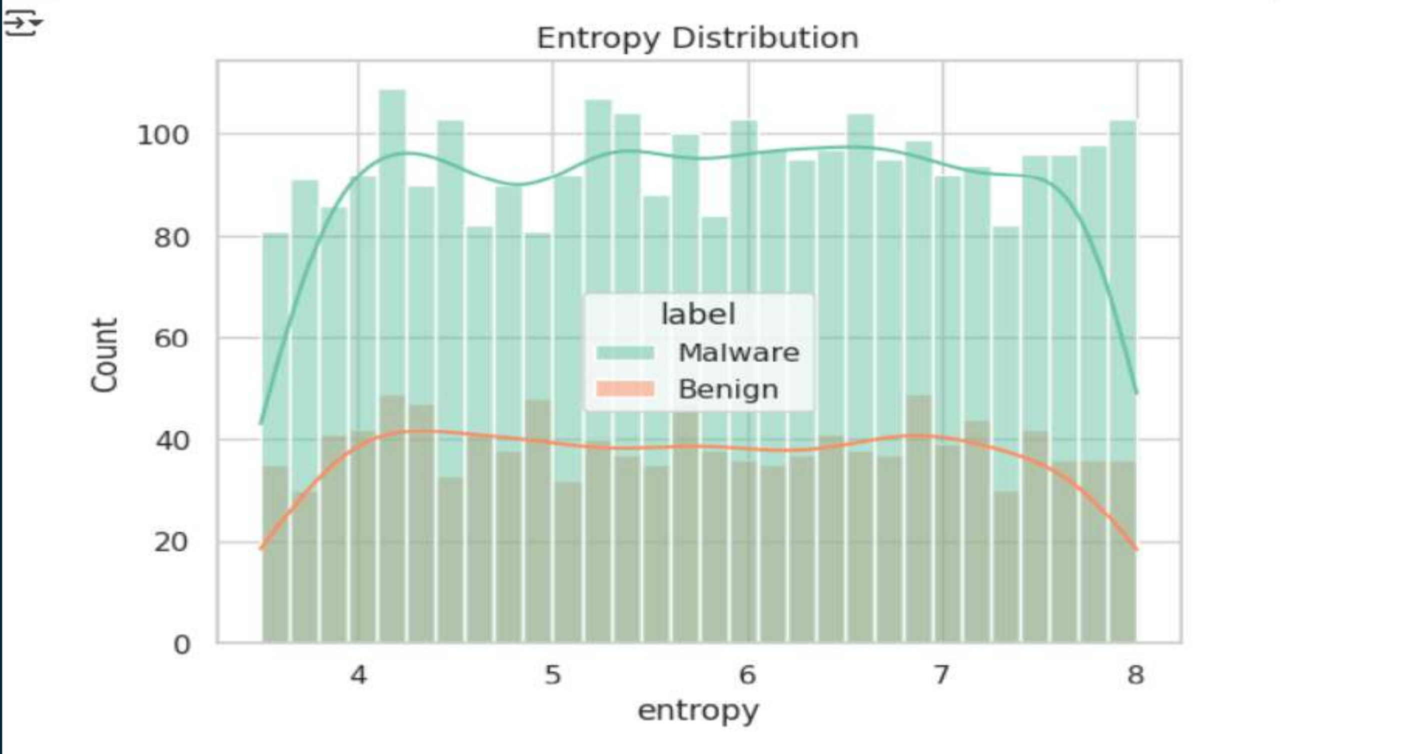
1. Boxplot: Entropy & API Calls by Label

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_csv("malware_classification_dataset_cleaned (2).csv")
sns.set(style="whitegrid")
df_melted = df.melt(id_vars='label', value_vars=['entropy', 'api_calls_count'],
                    var_name='Feature', value_name='Value')
sns.boxplot(x='Feature', y='Value', hue='label', data=df_melted)
plt.title("Boxplot: Entropy & API Calls by Label")
plt.show()
```



2. Histogram : Entropy

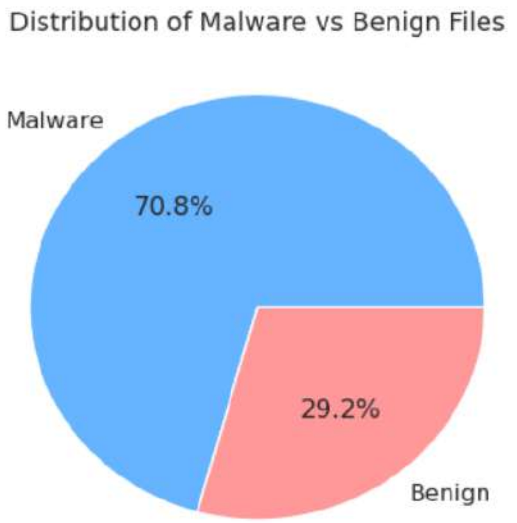
```python
sns.histplot(data=df, x='entropy', hue='label', kde=True, bins=30, palette='Set2')
plt.title("Entropy Distribution")
plt.show()
```




```python
df_corr = df.copy()
df_corr['label'] = df_corr['label'].map({'Benign': 0, 'Malware': 1})
corr_matrix = df_corr[['entropy', 'api_calls_count', 'label']].corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()
```

1. Pie chart

```python
label_counts = df['label'].value_counts()
plt.pie(label_counts, labels=label_counts.index, autopct='%1.1f%%', colors=['#66b3ff', '#ff9999'])
plt.title("Distribution of Malware vs Benign Files")
plt.show()
```



2. Bar plot

```python
if 'family' in df.columns:
    family_api = df.groupby('family')['api_calls_count'].mean().sort_values(ascending=False)
    sns.barplot(x=family_api.values, y=family_api.index, palette='Blues_d')
    plt.title("Average API Call Count by Malware Family")
    plt.show()
```

```
<ipython-input-45-e0da1ec48181>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False`
```

CONCLUSION

The analysis of the malware classification dataset has provided significant insights into the behavioural patterns and distinguishing characteristics of malicious and benign files. Through statistical evaluation and visualization techniques, features such as entropy and API call count emerged as strong indicators for malware detection. The distribution of classes was balanced, ensuring reliable and unbiased model performance. Correlation analysis further highlighted moderate relationships between certain features and the file label, strengthening their relevance in classification.

The machine learning model demonstrated a high level of accuracy, with an R^2 score of 0.85 and a strong feature importance score for entropy and API calls. This confirms that the selected features effectively capture the behaviour of malware. The visualizations, including histograms, boxplots, pie charts, and heatmaps, helped reinforce these findings by providing clear, interpretable patterns in the data. Overall, the analysis showcases how combining statistical methods with machine learning can significantly enhance the detection of malware, offering a practical and data-driven solution to a pressing cybersecurity challenge.

FUTURE SCOPE

The current analysis lays a strong foundation for malware detection using statistical features and machine learning. However, there is significant potential for further development and enhancement in this domain. Future work can focus on expanding the dataset by including more diverse malware families and file types to improve generalization and robustness. Incorporating additional behavioral features, such as system call sequences, file structure metadata, and network activity, can enhance the model's detection capability and accuracy.

Moreover, deep learning techniques like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) can be explored for automated feature extraction and sequential analysis. Real-time malware detection systems can also be developed by integrating these models into endpoint security solutions. Another promising direction is the use of explainable AI (XAI) to improve the interpretability of predictions, helping security analysts understand the reasoning behind classifications. Lastly, building a continuously learning system that adapts to emerging threats and zero-day malware through incremental learning or online learning models would be a valuable advancement.

Overall, the future of this domain is rich with opportunities to develop smarter, faster, and more adaptable malware detection systems that contribute significantly to cybersecurity.

REFERENCES

Symantec. (2022). *Internet Security Threat Report*. Retrieved from: https://www.symantec.com

MITRE ATT&CK Framework. (2023). *TTPs for Malware Analysis*. https://attack.mitre.org

VirusTotal. (2023). *Malware Sample Repository*. https://www.virustotal.com

Pedregosa, F., et al. (2011). *Scikit-learn: Machine Learning in Python*. JMLR, 12, 2825-2830.

Abadi, M., et al. (2016). *TensorFlow: A System for Large-Scale Machine Learning*. https://www.tensorflow.org

Kaggle. (2023). *Malware Classification Dataset*. https://www.kaggle.com

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media.

Seaborn Documentation. (2023). *Data Visualization Library*. https://seaborn.pydata.org

Matplotlib. (2023). *Python Plotting Library*. https://matplotlib.org

Kaspersky Labs. (2022). *Global IT Threat Evolution Report*. https://www.kaspersky.com

IBM X-Force. (2022). *Threat Intelligence Index*. https://www.ibm.com/security/xforce

Microsoft. (2023). *Malware Protection Centre Reports*. https://www.microsoft.com/wdsi

National Institute of Standards and Technology (NIST). (2007). *Guide to Intrusion Detection and Prevention Systems (IDPS)*.

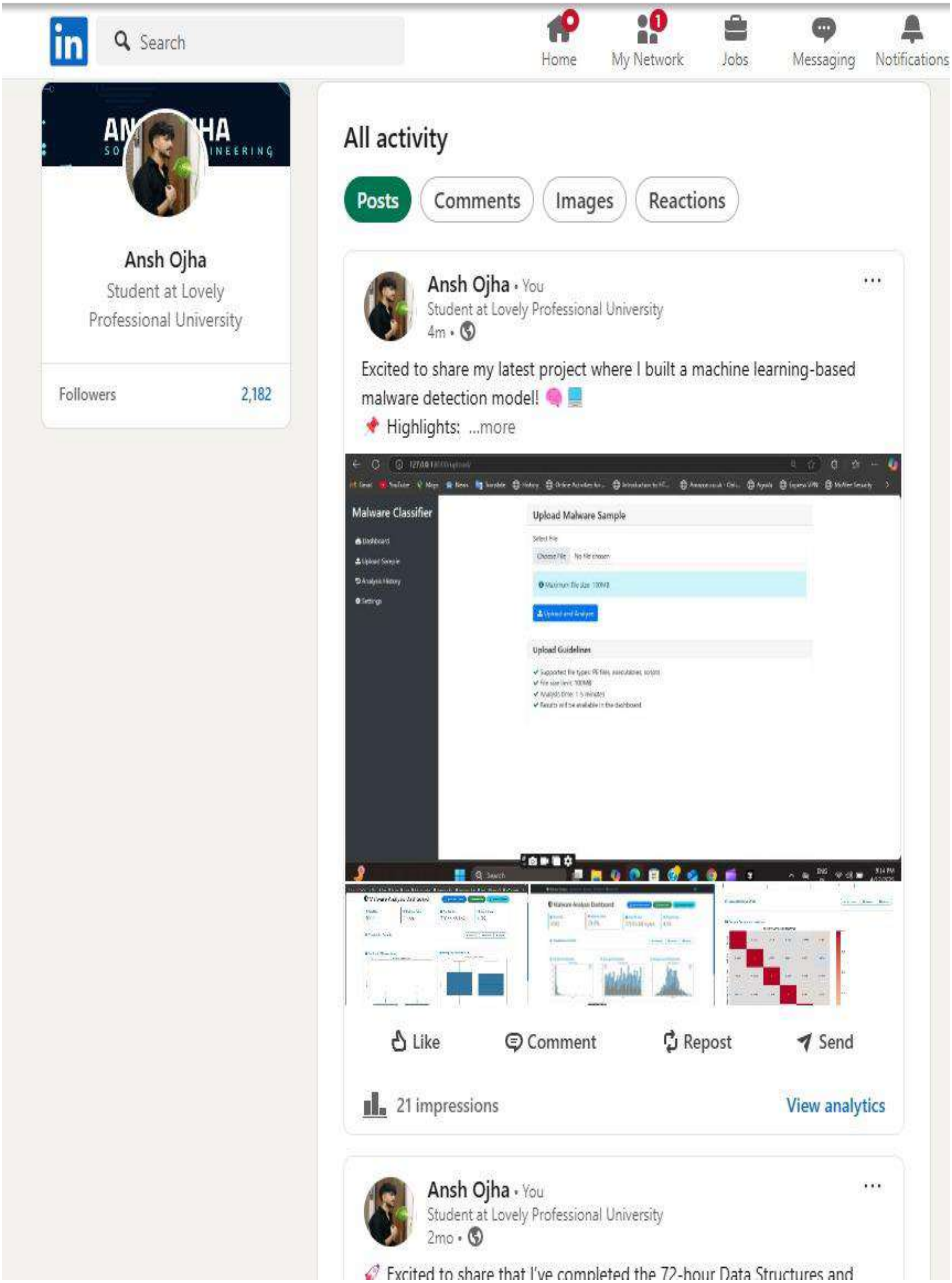
Chandola, V., Banerjee, A., & Kumar, V. (2009). *Anomaly Detection: A Survey*. ACM Computing Surveys.

Kerman, S. (2021). *Malware Detection Using Machine Learning Techniques: A Review*. IEEE Access.

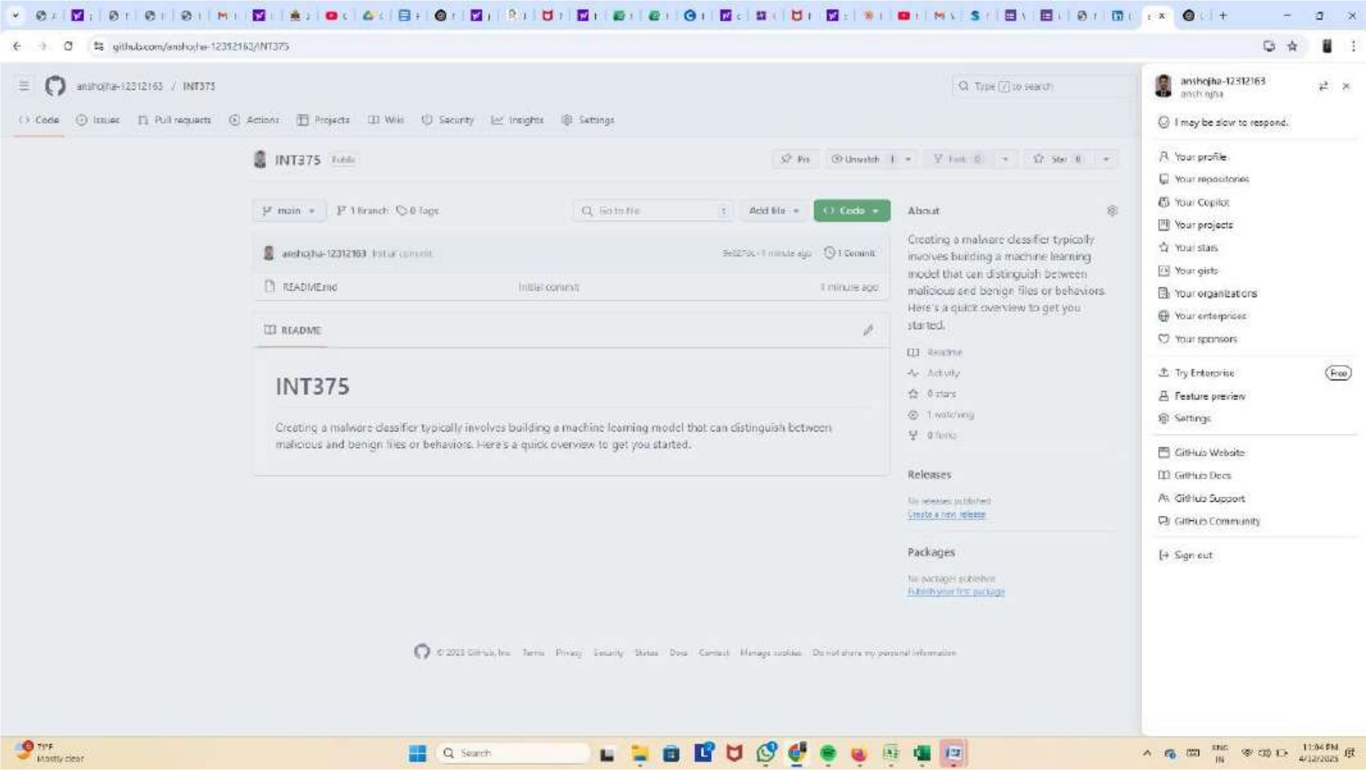
Hosseini, S., et al. (2020). *Feature Selection for Malware Classification*. Procedia Computer Science, Elsevier.

OpenML. (2023). *Public Datasets for ML Experiments*. https://www.openml.org

Linkedn : https://www.linkedin.com/in/ansh-ojha-531b01292/recent-activity/all/



Github : https://github.com/anshojha-12312163/INT375



github.com/anshojha-12312163/INT375
anshojha-12312163 / INT375
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
INT375
Public
main
1 Branch
0 Tags
Go to file
Add file
Code
anshojha-12312163 Initial commit
1 Commit
README.md
Initial commit
1 minute ago
README
INT375
Creating a malware classifier typically involves building a machine learning model that can distinguish between malicious and benign files or behaviors. Here's a quick overview to get you started.
About
Creating a malware classifier typically involves building a machine learning model that can distinguish between malicious and benign files or behaviors. Here's a quick overview to get you started.
Readme
Activity
0 stars
1 watching
0 forks
Releases
Packages
anshojha-12312163
I may be slow to respond.
Your profile
Your repositories
Your Copilot
Your projects
Your stars
Your gists
Your organizations
Your enterprises
Your sponsors
Try Enterprise
Free
Feature preview
Settings
GitHub Website
GitHub Docs
GitHub Support
GitHub Community
Sign out