In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
```

Dataset analysis

Benign and malicious PE Files Dataset for malware detection

In [2]:
```
df = pd.read_csv('./dataset/dataset_malwares.csv')
df.head()
```

Out[2]:

	Name	e_magic	e_cblp	e_cp	e_crlc	e_cparhdr	e_minalloc	e_maxalloc	e_ss	e_sp	.
0	VirusShare_a878ba26000edaac5c98eff4432723b3	23117	144	3	0	4	0	65535	0	184	.
1	VirusShare_ef9130570fddc174b312b2047f5f4cf0	23117	144	3	0	4	0	65535	0	184	.
2	VirusShare_ef84cdeba22be72a69b198213dada81a	23117	144	3	0	4	0	65535	0	184	.
3	VirusShare_6bf3608e60ebc16cbcff6ed5467d469e	23117	144	3	0	4	0	65535	0	184	.
4	VirusShare_2cc94d952b2efb13c7d6bbe0dd59d3fb	23117	144	3	0	4	0	65535	0	184	.

5 rows × 79 columns

In [3]:
```
df.shape
```

Out[3]:
```
(19611, 79)
```

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19611 entries, 0 to 19610
Data columns (total 79 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   Name                          19611 non-null  object
 1   e_magic                       19611 non-null  int64
 2   e_cblp                        19611 non-null  int64
 3   e_cp                          19611 non-null  int64
 4   e_crlc                        19611 non-null  int64
 5   e_cparhdr                     19611 non-null  int64
 6   e_minalloc                    19611 non-null  int64
 7   e_maxalloc                    19611 non-null  int64
 8   e_ss                          19611 non-null  int64
 9   e_sp                          19611 non-null  int64
 10  e_csum                        19611 non-null  int64
 11  e_ip                          19611 non-null  int64
 12  e_cs                          19611 non-null  int64
 13  e_lfarlc                      19611 non-null  int64
 14  e_ovno                        19611 non-null  int64
 15  e_oemid                       19611 non-null  int64
 16  e_oeminfo                     19611 non-null  int64
 17  e_lfanew                      19611 non-null  int64
 18  Machine                       19611 non-null  int64
 19  NumberOfSections              19611 non-null  int64
 20  TimeDateStamp                 19611 non-null  int64
 21  PointerToSymbolTable          19611 non-null  int64
 22  NumberOfSymbols               19611 non-null  int64
 23  SizeOfOptionalHeader          19611 non-null  int64
 24  Characteristics               19611 non-null  int64
 25  Magic                         19611 non-null  int64
 26  MajorLinkerVersion            19611 non-null  int64
 27  MinorLinkerVersion            19611 non-null  int64
 28  SizeOfCode                    19611 non-null  int64
 29  SizeOfInitializedData         19611 non-null  int64
 30  SizeOfUninitializedData       19611 non-null  int64
 31  AddressOfEntryPoint           19611 non-null  int64
 32  BaseOfCode                    19611 non-null  int64
 33  ImageBase                     19611 non-null  int64
 34  SectionAlignment              19611 non-null  int64
 35  FileAlignment                 19611 non-null  int64
 36  MajorOperatingSystemVersion   19611 non-null  int64
 37  MinorOperatingSystemVersion   19611 non-null  int64
 38  MajorImageVersion             19611 non-null  int64
```

```
 39  MinorImageVersion             19611 non-null  int64
 40  MajorSubsystemVersion         19611 non-null  int64
 41  MinorSubsystemVersion         19611 non-null  int64
 42  SizeOfHeaders                 19611 non-null  int64
 43  CheckSum                      19611 non-null  int64
 44  SizeOfImage                   19611 non-null  int64
 45  Subsystem                     19611 non-null  int64
 46  DllCharacteristics            19611 non-null  int64
 47  SizeOfStackReserve            19611 non-null  int64
 48  SizeOfStackCommit             19611 non-null  int64
 49  SizeOfHeapReserve             19611 non-null  int64
 50  SizeOfHeapCommit              19611 non-null  int64
 51  LoaderFlags                   19611 non-null  int64
 52  NumberOfRvaAndSizes           19611 non-null  int64
 53  Malware                       19611 non-null  int64
 54  SuspiciousImportFunctions     19611 non-null  int64
 55  SuspiciousNameSection         19611 non-null  int64
 56  SectionsLength                19611 non-null  int64
 57  SectionMinEntropy             19611 non-null  float64
 58  SectionMaxEntropy             19611 non-null  int64
 59  SectionMinRawsize             19611 non-null  int64
 60  SectionMaxRawsize             19611 non-null  int64
 61  SectionMinVirtualsize         19611 non-null  int64
 62  SectionMaxVirtualsize         19611 non-null  int64
 63  SectionMaxPhysical            19611 non-null  int64
 64  SectionMinPhysical            19611 non-null  int64
 65  SectionMaxVirtual             19611 non-null  int64
 66  SectionMinVirtual             19611 non-null  int64
 67  SectionMaxPointerData         19611 non-null  int64
 68  SectionMinPointerData         19611 non-null  int64
 69  SectionMaxChar                19611 non-null  int64
 70  SectionMainChar               19611 non-null  int64
 71  DirectoryEntryImport          19611 non-null  int64
 72  DirectoryEntryImportSize      19611 non-null  int64
 73  DirectoryEntryExport          19611 non-null  int64
 74  ImageDirectoryEntryExport     19611 non-null  int64
 75  ImageDirectoryEntryImport     19611 non-null  int64
 76  ImageDirectoryEntryResource   19611 non-null  int64
 77  ImageDirectoryEntryException  19611 non-null  int64
 78  ImageDirectoryEntrySecurity   19611 non-null  int64
dtypes: float64(1), int64(77), object(1)
memory usage: 11.8+ MB
```

In [5]:
```
df.describe()
```

Out[5]:

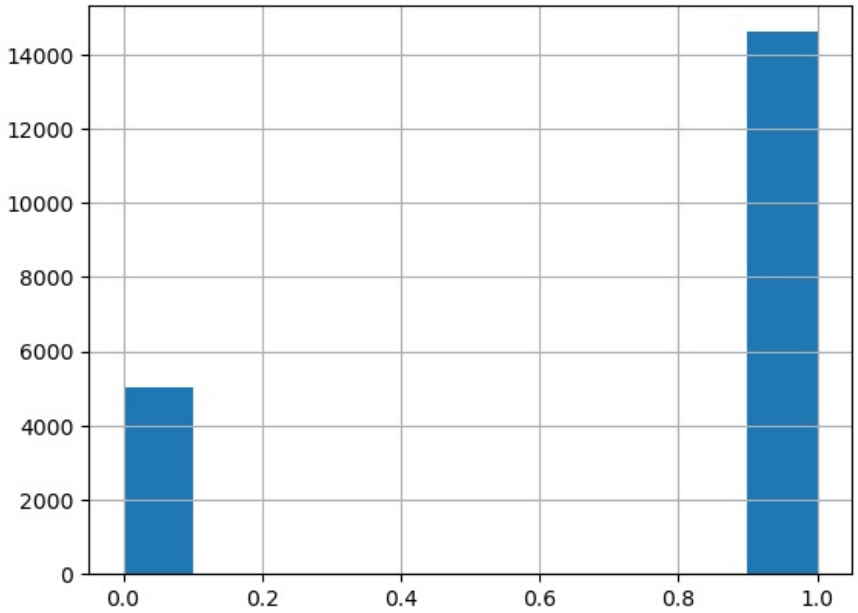
	e_magic	e_cblp	e_cp	e_crlc	e_cparhdr	e_minalloc	e_maxalloc	e_ss	e_sp
count	19611.0	19611.000000	19611.000000	19611.000000	19611.000000	19611.000000	19611.000000	19611.000000	19611.00000
mean	23117.0	178.615726	71.660752	49.146958	37.370710	37.032635	64178.739687	10.418490	226.46530
std	0.0	987.200729	1445.192977	1212.201919	864.515405	915.833139	9110.755873	637.116265	1249.68033
min	23117.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
25%	23117.0	144.000000	3.000000	0.000000	4.000000	0.000000	65535.000000	0.000000	184.00000
50%	23117.0	144.000000	3.000000	0.000000	4.000000	0.000000	65535.000000	0.000000	184.00000
75%	23117.0	144.000000	3.000000	0.000000	4.000000	0.000000	65535.000000	0.000000	184.00000
max	23117.0	59448.000000	63200.000000	64613.000000	43690.000000	43690.000000	65535.000000	61436.000000	65464.00000

8 rows × 78 columns

In [6]:
```
df.drop(['Name', 'Machine', 'TimeDateStamp', 'SectionMainChar'], axis=1, inplace=True)
```

In [7]:
```
df['Malware'].hist()
```

Out[7]:
```
<Axes: >
```



In [8]:
```python
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X = df.drop("Malware", axis=1)
y = df["Malware"]

X_scale = scaler.fit_transform(X)
```

In [9]:
```python
# for imbalance datset stratify (will make sure the distibution is balanced in y train and test)
X_train, X_test, y_train, y_test = train_test_split(X_scale, y, test_size=0.2, stratify=y, random_state=42)
```

Training Multiple ML Algorithms

In [10]:
```python
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score

def train_and_evaluate(X_train, y_train, X_test, y_test):
    models = {
        'Logistic Regression': LogisticRegression(solver='liblinear'),
        'Random Forest': RandomForestClassifier(),
        'K-Nearest Neighbors': KNeighborsClassifier()
    }

    results = {}

    for name, model in models.items():
        model.fit(X_train, y_train)
        predictions = model.predict(X_test)

        accuracy = accuracy_score(y_test, predictions)
        precision = precision_score(y_test, predictions, average='weighted')
        recall = recall_score(y_test, predictions, average='weighted')

        results[name] = {
            'Accuracy': accuracy,
            'Precision': precision,
            'Recall': recall
        }

        print(f"{name}:\n Accuracy: {accuracy:.2f}\n Precision: {precision:.2f}\n Recall: {recall:.2f}\n")
```

```
    return None

train_and_evaluate(X_train, y_train, X_test, y_test)
```

```
Logistic Regression:
 Accuracy: 0.95
 Precision: 0.95
 Recall: 0.95

Random Forest:
 Accuracy: 0.99
 Precision: 0.99
 Recall: 0.99

K-Nearest Neighbors:
 Accuracy: 0.97
 Precision: 0.97
 Recall: 0.97
```