```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
#importing all libraries
import os
import six
import sys
import pickle
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objects as go
import plotly.express as px
import plotly.figure_factory as ff
from sklearn import preprocessing
from sklearn import ensemble
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
sys.modules['sklearn.externals.six'] = six
from mlxtend.classifier import StackingClassifier
from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier, GradientBoostingC
from sklearn.metrics import confusion_matrix, classification_report , accuracy_score
from sklearn.feature_selection import mutual_info_classif, SelectFromModel, SelectPercent
import warnings
warnings.filterwarnings("ignore")
```

```
#loading dataset file
df = pd.read_csv('/content/drive/MyDrive/pe_malware_classification/Data/dataset_malwares.
df.head()
```

	Name	e_magic	e_cblp	e_cp	e_crlc	e_cparh
0	VirusShare_a878ba26000edaac5c98eff4432723b3	23117	144	3	0	
1	VirusShare_ef9130570fddc174b312b2047f5f4cf0	23117	144	3	0	
2	VirusShare_ef84cdeba22be72a69b198213dada81a	23117	144	3	0	
3	VirusShare_6bf3608e60ebc16cbcff6ed5467d469e	23117	144	3	0	
4	VirusShare_2cc94d952b2efb13c7d6bbe0dd59d3fb	23117	144	3	0	

5 rows × 79 columns

```
#only taking below columns for training(removed unnecessary columns)
df = df[['e_magic', 'e_cblp', 'e_cp', 'e_crlc', 'e_cparhdr', 'e_minalloc',
       'e_maxalloc', 'e_ss', 'e_sp', 'e_csum', 'e_ip', 'e_cs', 'e_lfarlc',
       'e_ovno', 'e_oemid', 'e_oeminfo', 'e_lfanew', 'Machine',
       'NumberOfSections', 'TimeDateStamp', 'PointerToSymbolTable',
       'NumberOfSymbols', 'SizeOfOptionalHeader', 'Characteristics', 'Magic',
       'MajorLinkerVersion', 'MinorLinkerVersion', 'SizeOfCode',
       'SizeOfInitializedData', 'SizeOfUninitializedData',
       'AddressOfEntryPoint', 'BaseOfCode', 'ImageBase', 'SectionAlignment',
       'FileAlignment', 'MajorOperatingSystemVersion',
       'MinorOperatingSystemVersion', 'MajorImageVersion', 'MinorImageVersion',
       'MajorSubsystemVersion', 'MinorSubsystemVersion', 'SizeOfHeaders',
       'CheckSum', 'SizeOfImage', 'Subsystem', 'DllCharacteristics',
       'SizeOfStackReserve', 'SizeOfStackCommit', 'SizeOfHeapReserve',
       'SizeOfHeapCommit', 'LoaderFlags', 'NumberOfRvaAndSizes', 'Malware']]
df.head()
```

	e_magic	e_cblp	e_cp	e_crlc	e_cparhdr	e_minalloc	e_maxalloc	e_ss	e_sp	e_csu
0	23117	144	3	0	4	0	65535	0	184	
1	23117	144	3	0	4	0	65535	0	184	
2	23117	144	3	0	4	0	65535	0	184	
3	23117	144	3	0	4	0	65535	0	184	
4	23117	144	3	0	4	0	65535	0	184	

5 rows × 53 columns

```
#shape of dataset
df.shape
```

```
(19611, 53)
```

```
#information of datatype
df.info()
```

```
Data columns (total 53 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   e_magic                      19611 non-null  int64
 1   e_cblp                       19611 non-null  int64
 2   e_cp                         19611 non-null  int64
 3   e_crlc                       19611 non-null  int64
 4   e_cparhdr                    19611 non-null  int64
 5   e_minalloc                   19611 non-null  int64
 6   e_maxalloc                   19611 non-null  int64
 7   e_ss                         19611 non-null  int64
 8   e_sp                         19611 non-null  int64
 9   e_csum                       19611 non-null  int64
 10  e_ip                         19611 non-null  int64
```

```
 13  e_ovno                     19611 non-null  int64
 14  e_oemid                    19611 non-null  int64
 15  e_oeminfo                  19611 non-null  int64
 16  e_lfanew                   19611 non-null  int64
 17  Machine                    19611 non-null  int64
 18  NumberOfSections           19611 non-null  int64
 19  TimeDateStamp              19611 non-null  int64
 20  PointerToSymbolTable       19611 non-null  int64
 21  NumberOfSymbols            19611 non-null  int64
 22  SizeOfOptionalHeader       19611 non-null  int64
 23  Characteristics            19611 non-null  int64
 24  Magic                      19611 non-null  int64
 25  MajorLinkerVersion         19611 non-null  int64
 26  MinorLinkerVersion         19611 non-null  int64
 27  SizeOfCode                 19611 non-null  int64
 28  SizeOfInitializedData      19611 non-null  int64
 29  SizeOfUninitializedData    19611 non-null  int64
 30  AddressOfEntryPoint        19611 non-null  int64
 31  BaseOfCode                 19611 non-null  int64
 32  ImageBase                  19611 non-null  int64
 33  SectionAlignment           19611 non-null  int64
 34  FileAlignment              19611 non-null  int64
 35  MajorOperatingSystemVersion  19611 non-null  int64
 36  MinorOperatingSystemVersion  19611 non-null  int64
 37  MajorImageVersion          19611 non-null  int64
 38  MinorImageVersion          19611 non-null  int64
 39  MajorSubsystemVersion      19611 non-null  int64
 40  MinorSubsystemVersion      19611 non-null  int64
 41  SizeOfHeaders              19611 non-null  int64
 42  CheckSum                   19611 non-null  int64
 43  SizeOfImage                19611 non-null  int64
 44  Subsystem                  19611 non-null  int64
 45  DllCharacteristics         19611 non-null  int64
 46  SizeOfStackReserve         19611 non-null  int64
 47  SizeOfStackCommit          19611 non-null  int64
 48  SizeOfHeapReserve          19611 non-null  int64
 49  SizeOfHeapCommit           19611 non-null  int64
 50  LoaderFlags                19611 non-null  int64
 51  NumberOfRvaAndSizes        19611 non-null  int64
 52  Malware                    19611 non-null  int64
dtypes: int64(53)
memory usage: 7.9 MB
```

```
#statistic of dataset
df.describe()
```

	e_magic	e_cblp	e_cp	e_crlc	e_cparhdr	e_minalloc	
count	19611.0	19611.000000	19611.000000	19611.000000	19611.000000	19611.000000	1
mean	23117.0	178.615726	71.660752	49.146958	37.370710	37.032635	6
std	0.0	987.200729	1445.192977	1212.201919	864.515405	915.833139	
min	23117.0	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	23117.0	144.000000	3.000000	0.000000	4.000000	0.000000	6
50%	23117.0	144.000000	3.000000	0.000000	4.000000	0.000000	6
75%	23117.0	144.000000	3.000000	0.000000	4.000000	0.000000	6
max	23117.0	59448.000000	63200.000000	64613.000000	43690.000000	43690.000000	6

8 rows × 53 columns

```
#checking null values in the dataset
df.isna().sum()
```

```
e_magic                        0
e_cblp                         0
e_cp                           0
e_crlc                         0
e_cparhdr                      0
e_minalloc                     0
e_maxalloc                     0
e_ss                           0
e_sp                           0
e_csum                         0
e_ip                           0
e_cs                           0
e_lfarlc                       0
e_ovno                         0
e_oemid                        0
e_oeminfo                      0
e_lfanew                       0
Machine                        0
NumberOfSections               0
TimeDateStamp                  0
PointerToSymbolTable           0
NumberOfSymbols                0
SizeOfOptionalHeader           0
Characteristics                0
Magic                          0
MajorLinkerVersion             0
MinorLinkerVersion             0
SizeOfCode                     0
SizeOfInitializedData          0
SizeOfUninitializedData        0
AddressOfEntryPoint            0
BaseOfCode                     0
ImageBase                      0
SectionAlignment               0
FileAlignment                  0
```

```
MajorOperatingSystemVersion    0
MinorOperatingSystemVersion    0
MajorImageVersion              0
MinorImageVersion              0
MajorSubsystemVersion          0
MinorSubsystemVersion          0
SizeOfHeaders                  0
CheckSum                       0
SizeOfImage                    0
Subsystem                      0
DllCharacteristics             0
SizeOfStackReserve             0
SizeOfStackCommit              0
SizeOfHeapReserve              0
SizeOfHeapCommit               0
LoaderFlags                    0
NumberOfRvaAndSizes            0
Malware                        0
dtype: int64
```

```
df.columns
```

```
Index(['e_magic', 'e_cblp', 'e_cp', 'e_crlc', 'e_cparhdr', 'e_minalloc',
       'e_maxalloc', 'e_ss', 'e_sp', 'e_csum', 'e_ip', 'e_cs', 'e_lfarlc',
       'e_ovno', 'e_oemid', 'e_oeminfo', 'e_lfanew', 'Machine',
       'NumberOfSections', 'TimeDateStamp', 'PointerToSymbolTable',
       'NumberOfSymbols', 'SizeOfOptionalHeader', 'Characteristics', 'Magic',
       'MajorLinkerVersion', 'MinorLinkerVersion', 'SizeOfCode',
       'SizeOfInitializedData', 'SizeOfUninitializedData',
       'AddressOfEntryPoint', 'BaseOfCode', 'ImageBase', 'SectionAlignment',
       'FileAlignment', 'MajorOperatingSystemVersion',
       'MinorOperatingSystemVersion', 'MajorImageVersion', 'MinorImageVersion',
       'MajorSubsystemVersion', 'MinorSubsystemVersion', 'SizeOfHeaders',
       'CheckSum', 'SizeOfImage', 'Subsystem', 'DllCharacteristics',
       'SizeOfStackReserve', 'SizeOfStackCommit', 'SizeOfHeapReserve',
       'SizeOfHeapCommit', 'LoaderFlags', 'NumberOfRvaAndSizes', 'Malware'],
      dtype='object')
```

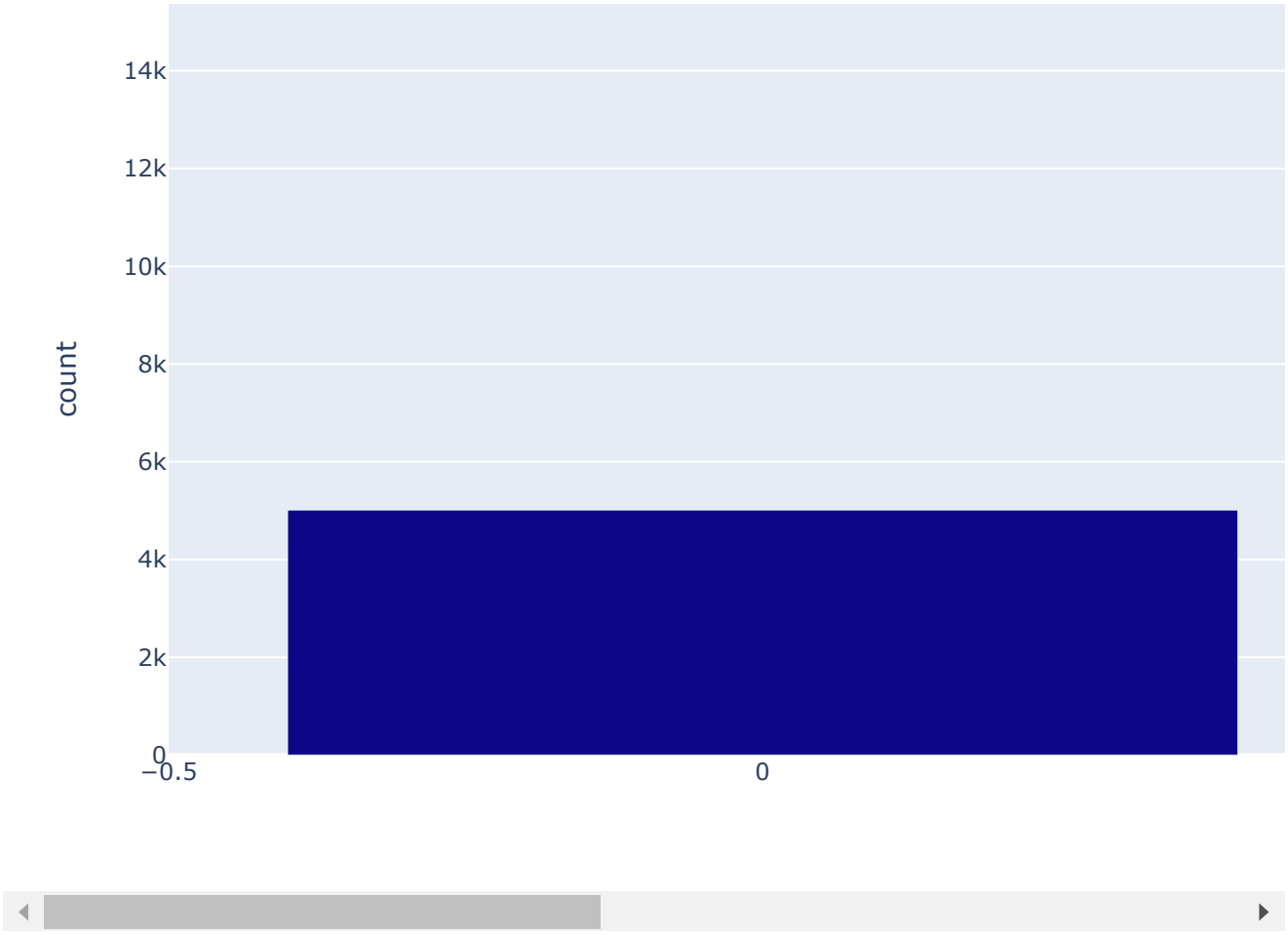
Data Visualization

```
#number of unique value in each columns
for i in df.columns:
  print(i,df[i].unique().size)
```

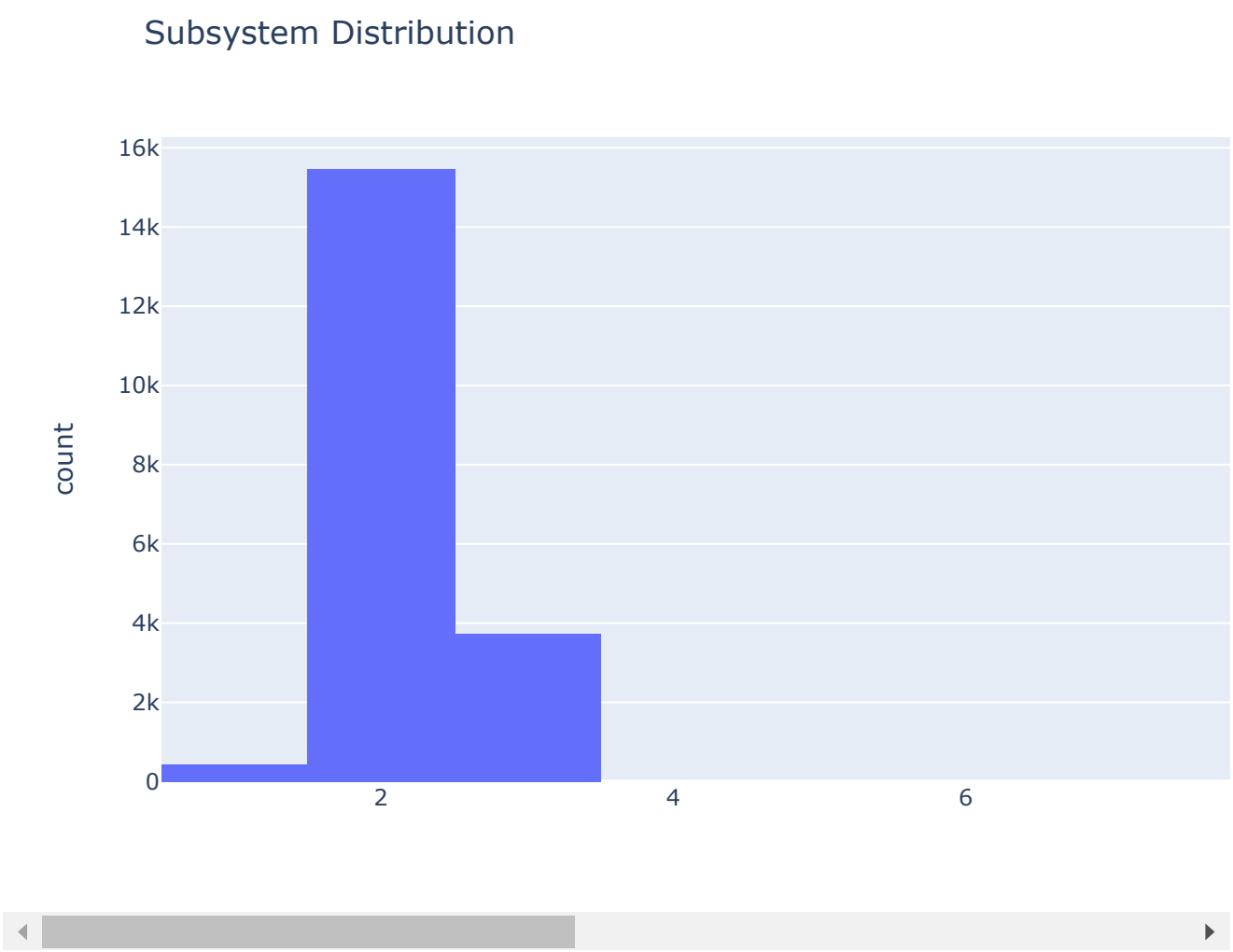
```
e_magic 1
e_cblp 39
e_cp 43
e_crlc 26
e_cparhdr 22
e_minalloc 20
e_maxalloc 17
e_ss 13
e_sp 26
e_csum 16
e_ip 28
e_cs 23
e_lfarlc 16
```

```
e_ovno 14
e_oemid 87
e_oeminfo 96
e_lfanew 63
Machine 4
NumberOfSections 29
TimeDateStamp 11145
PointerToSymbolTable 70
NumberOfSymbols 83
SizeOfOptionalHeader 5
Characteristics 103
Magic 2
MajorLinkerVersion 52
MinorLinkerVersion 65
SizeOfCode 1894
SizeOfInitializedData 1962
SizeOfUninitializedData 328
AddressOfEntryPoint 9481
BaseOfCode 300
ImageBase 900
SectionAlignment 9
FileAlignment 8
MajorOperatingSystemVersion 22
MinorOperatingSystemVersion 26
MajorImageVersion 53
MinorImageVersion 114
MajorSubsystemVersion 7
MinorSubsystemVersion 9
SizeOfHeaders 33
CheckSum 12236
SizeOfImage 1269
Subsystem 6
DllCharacteristics 75
SizeOfStackReserve 34
SizeOfStackCommit 27
SizeOfHeapReserve 22
SizeOfHeapCommit 22
LoaderFlags 35
NumberOfRvaAndSizes 19
Malware 2
```

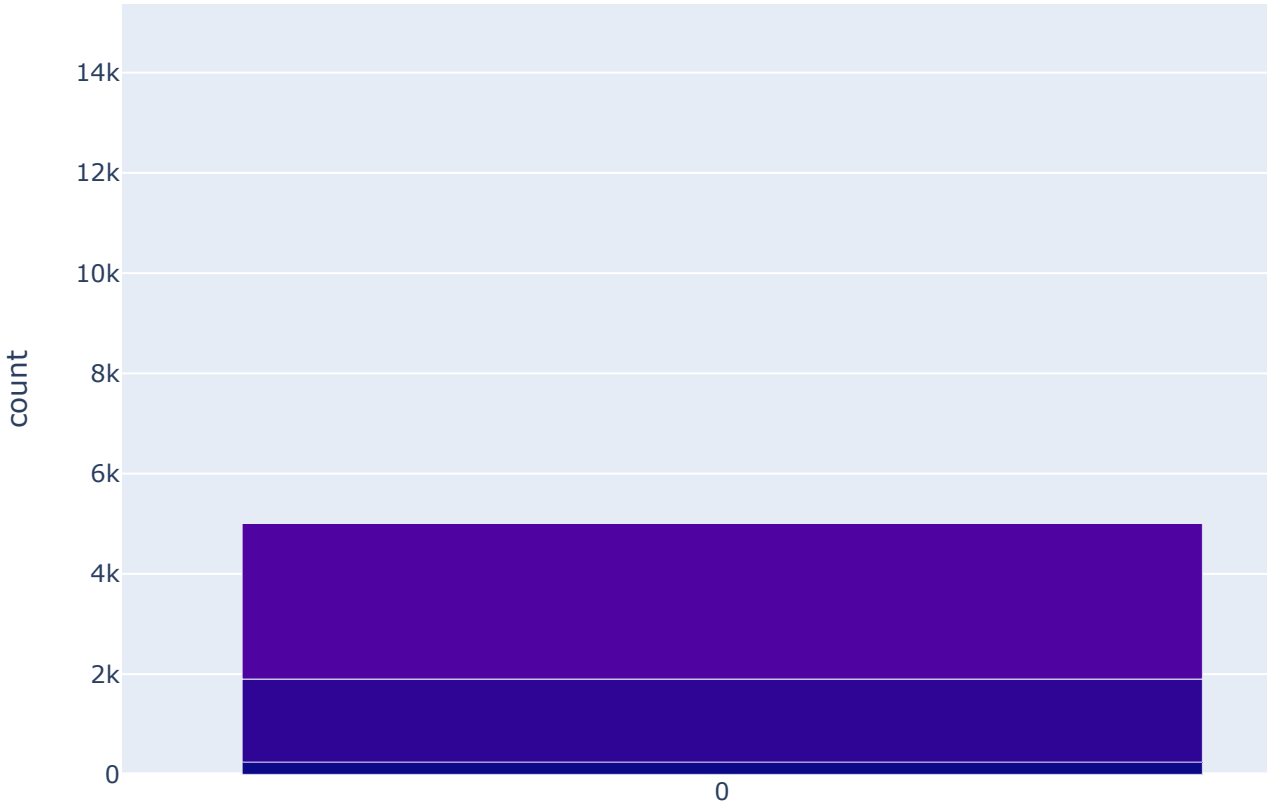
```
#bar chart showing value count of target class(data is imblanced beacuse piechart is not
df1 = df['Malware'].value_counts().reset_index().rename(columns={'index':'Malware','Malwa
fig = px.bar(df1, y='count', x='Malware', color='Malware')
fig.show()
```



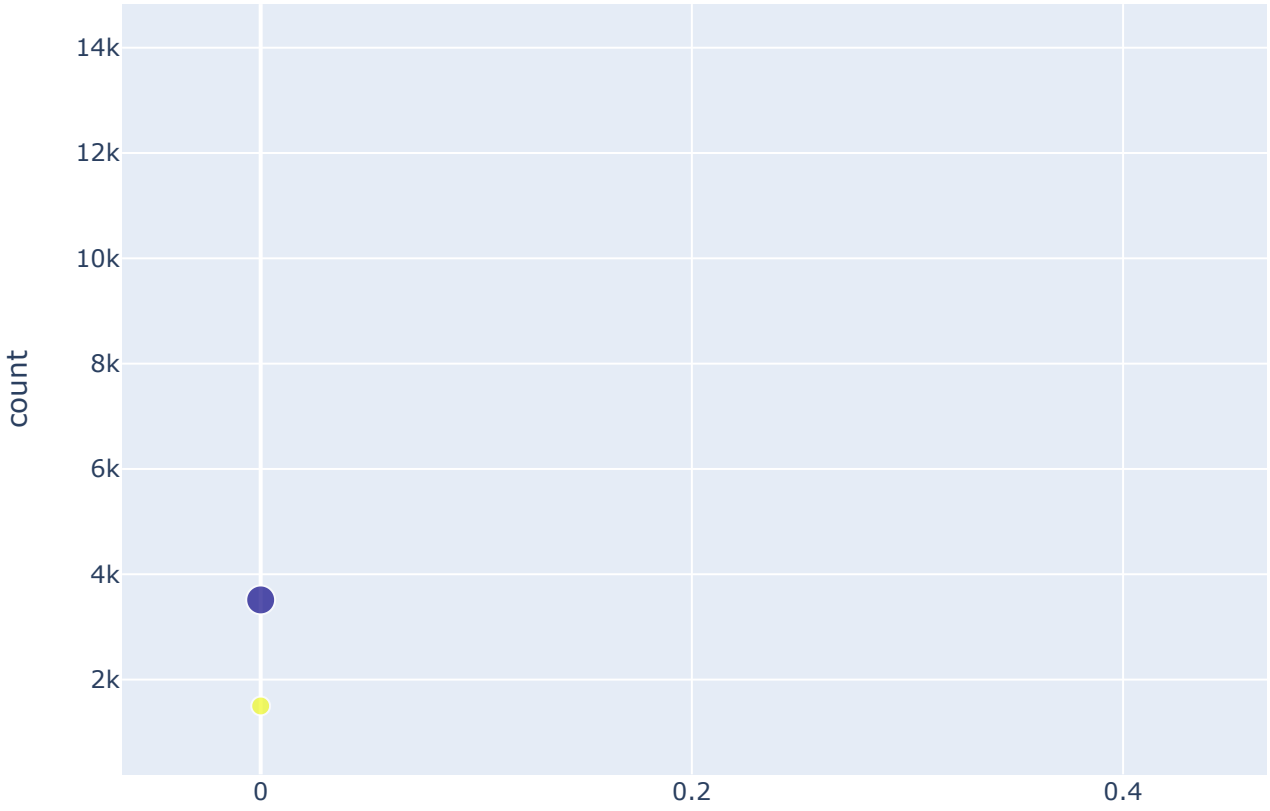

```
#Data distribution of Subsystem
fig = px.histogram(df, x="Subsystem" , title="Subsystem Distribution")
fig.show()
```




```
#Subsystem wise Malware count
df3 = df[['Subsystem','Malware']].groupby(['Subsystem','Malware']).value_counts()
df3 = df3.reset_index()
df3.columns = ['Subsystem','Malware','count']
fig = px.bar(df3, x='Malware', y='count',color='Subsystem',barmode="group")
fig.show()
```

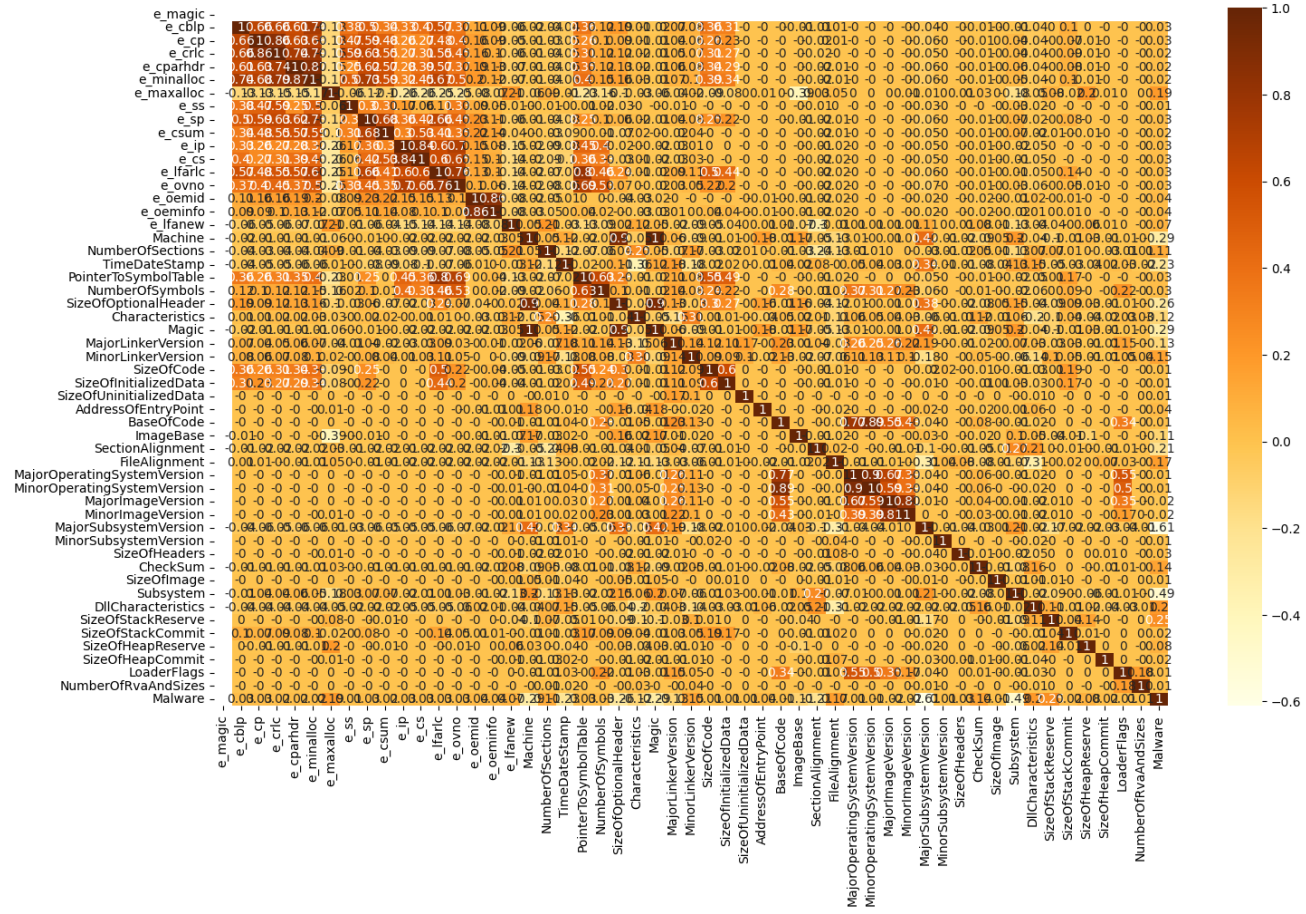



```
#Magic wise Malware count
df3 = df[['Malware','Magic']].groupby(['Malware','Magic']).value_counts()
df3 = df3.reset_index()
df3.columns = ['Malware','Magic','count']
fig = px.scatter(df3, x='Malware', y='count',color='Magic',size="count")
fig.show()
```




```
#correlation matrix
correlation = df.corr().round(2)
plt.figure(figsize = (17,10))
sns.heatmap(correlation, annot = True, cmap = 'YlOrBr')
```

<Axes: >


e_magic
e_cblp
e_cp
e_crlc
e_cparhdr
e_minalloc
e_maxalloc
e_ss
e_sp
e_csum
e_ip
e_cs
e_lfarlc
e_ovno
e_oemid
e_oeminfo
e_lfanew
Machine
NumberOfSections
TimeDateStamp
PointerToSymbolTable
NumberOfSymbols
SizeOfOptionalHeader
Characteristics
Magic
MajorLinkerVersion
MinorLinkerVersion
SizeOfCode
SizeOfInitializedData
SizeOfUninitializedData
AddressOfEntryPoint
BaseOfCode
ImageBase
SectionAlignment
FileAlignment
MajorOperatingSystemVersion
MinorOperatingSystemVersion
MajorImageVersion
MinorImageVersion
MajorSubsystemVersion
MinorSubsystemVersion
SizeOfHeaders
CheckSum
SizeOfImage
Subsystem
DllCharacteristics
SizeOfStackReserve
SizeOfStackCommit
SizeOfHeapReserve
SizeOfHeapCommit
LoaderFlags
NumberOfRvaAndSizes
Malware
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

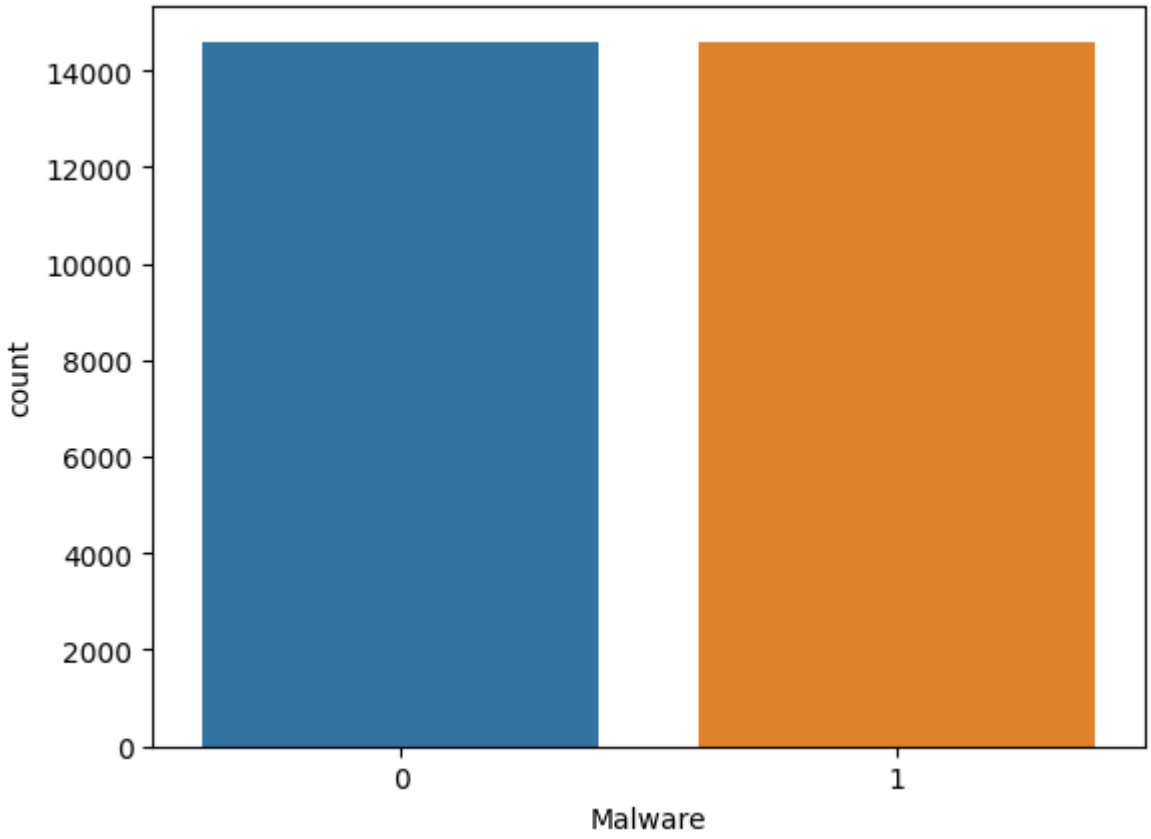
Data Preprocessing

```
#splitting data into x and y
X = df.drop('Malware',axis=1)
y = df.Malware
```

```
#data balancing
oversample = SMOTE()
X, y = oversample.fit_resample(X,y)
```

```
sns.countplot(x=y)
```

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

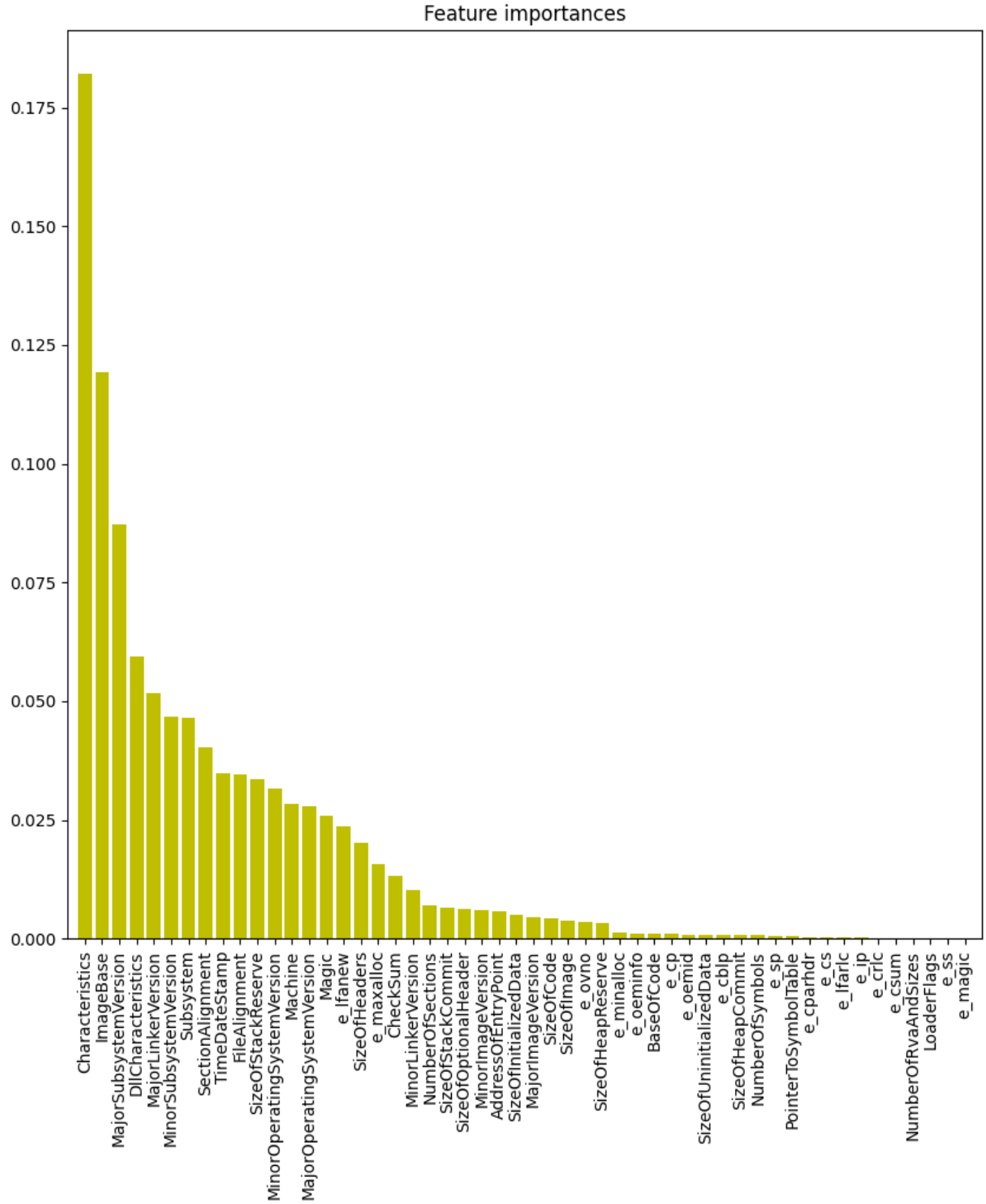



```
#splitting data into train and test with ratio of 80:20
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,stratify=y)
```

```
X_train.shape
```

```
(23358, 52)
```

```
#model feature importance(for visualization only)
fearture_name = X_train.columns.values
model = ensemble.ExtraTreesRegressor(n_estimators=25, max_depth=30, max_features=0.3, n_j
model.fit(X_train,y_train)
#plot imp
importance = model.feature_importances_
std = np.std([tree.feature_importances_ for tree in model.estimators_],axis=0)
indices = np.argsort(importance)[::-1][:52]
plt.figure(figsize=(10,10))
plt.title("Feature importances")
plt.bar(range(len(indices)), importance[indices], color="y")
plt.xticks(range(len(indices)), fearture_name[indices], rotation='vertical')
plt.xlim([-1, len(indices)])
plt.show()
```

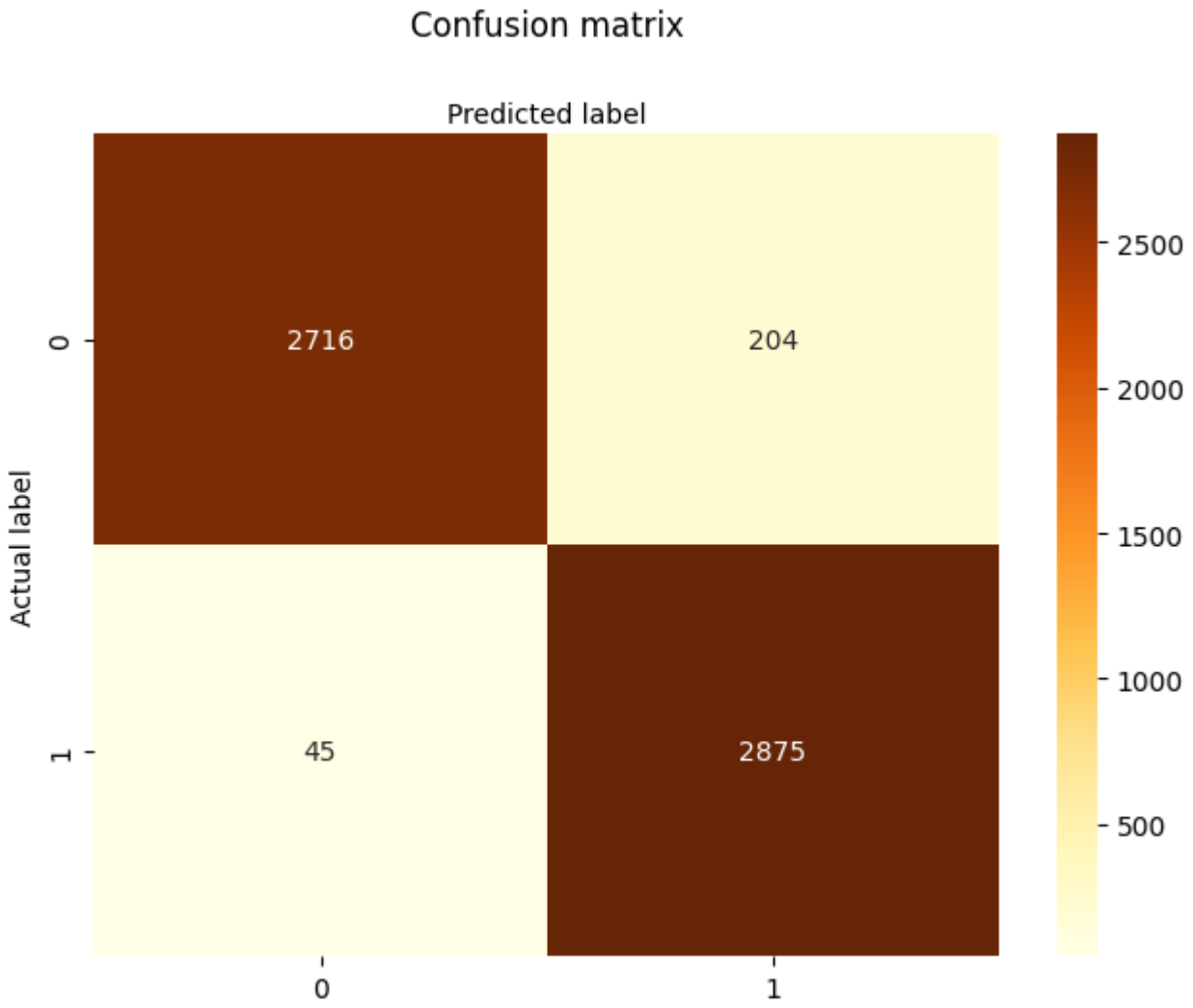

Feature importances
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
Characteristics
ImageBase
MajorSubsystemVersion
DllCharacteristics
MajorLinkerVersion
MinorSubsystemVersion
Subsystem
SectionAlignment
TimeDateStamp
FileAlignment
SizeOfStackReserve
MinorOperatingSystemVersion
Machine
MajorOperatingSystemVersion
Magic
e_lfanew
SizeOfHeaders
e_maxalloc
CheckSum
MinorLinkerVersion
NumberOfSections
SizeOfStackCommit
SizeOfOptionalHeader
MinorImageVersion
AddressOfEntryPoint
SizeOfInitializedData
MajorImageVersion
SizeOfCode
SizeOfImage
e_ovno
SizeOfHeapReserve
e_minalloc
e_oeminfo
BaseOfCode
e_cp
e_oemid
SizeOfUninitializedData
e_cblp
SizeOfHeapCommit
NumberOfSymbols
e_sp
PointerToSymbolTable
e_cparhdr
e_cs
e_lfarlc
e_ip
e_crlc
e_csum
NumberOfRvaAndSizes
LoaderFlags
e_ss
e_magic

Machine Learning Models

AdaBoost Classifier

```
adb_model = AdaBoostClassifier(n_estimators=4)
adbclf = adb_model
adbclf.fit(X_train,y_train)
y_pred = adbclf.predict(X_test)
#accuracy score
print("Accuracy Score: ",accuracy_score(y_test,y_pred))
#confusion Matrix
matrix =confusion_matrix(y_test, y_pred)
class_names=[0,1]
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
sns.heatmap(pd.DataFrame(matrix), annot=True, cmap="YlOrBr" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
#Classification Report
print(classification_report(y_test, y_pred))
```

```
Accuracy Score:  0.9573630136986301
```




```
              precision    recall  f1-score   support

           0       0.98      0.93      0.96      2920
           1       0.93      0.98      0.96      2920

    accuracy                           0.96      5840
   macro avg       0.96      0.96      0.96      5840
weighted avg       0.96      0.96      0.96      5840
```

Gradient Boosting Classifier

```
gbc_model = GradientBoostingClassifier(n_estimators=8)
gbc = gbc_model
gbc.fit(X_train,y_train)
y_pred = gbc.predict(X_test)
#accuracy score
print("Accuracy Score: ",accuracy_score(y_test,y_pred))
#confusion Matrix
matrix =confusion_matrix(y_test, y_pred)
class_names=[0,1]
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
sns.heatmap(pd.DataFrame(matrix), annot=True, cmap="YlOrBr" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
#Classification Report
print(classification_report(y_test, y_pred))
```

Random Forest Classifier

```
#RandomForest Classifier
rf_clf = RandomForestClassifier(n_estimators=2, criterion='gini', max_depth=3,)
rf = rf_clf
rf.fit(X_train,y_train)
y_pred = rf.predict(X_test)
#accuracy score
print("Accuracy Score: ",accuracy_score(y_test,y_pred))
#confusion Matrix
matrix =confusion_matrix(y_test, y_pred)
class_names=[0,1]
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
sns.heatmap(pd.DataFrame(matrix), annot=True, cmap="YlOrBr" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
#Classification Report
print(classification_report(y_test, y_pred))
```

```
Accuracy Score:  0.9214041095890411
```