# Vasu Bansal

# 20BCE1072

# CSE4003

# Digital Assignment-3

# Implementation:

def extract(pe):

def get\_directory\_entries():

directory\_entries = []

# directory\_entries.append(pe.DIRECTORY\_ENTRY\_IMPORT)

# directory\_entries.append(pe.DIRECTORY\_ENTRY\_IMPORT\_size)

# directory\_entries.append(pe.DIRECTORY\_ENTRY\_EXPORT)

directory\_entries.append(

pe.OPTIONAL\_HEADER.DATA\_DIRECTORY[pefile.DIRECTORY\_ENTRY['IMAGE\_DIRECTORY\_ENTRY\_EXPORT']].VirtualAddress)

directory\_entries.append(

pe.OPTIONAL\_HEADER.DATA\_DIRECTORY[pefile.DIRECTORY\_ENTRY['IMAGE\_DIRECTORY\_ENTRY\_IMPORT']].VirtualAddress)

directory\_entries.append(

pe.OPTIONAL\_HEADER.DATA\_DIRECTORY[pefile.DIRECTORY\_ENTRY['IMAGE\_DIRECTORY\_ENTRY\_RESOURCE']].VirtualAddress)

directory\_entries.append(

pe.OPTIONAL\_HEADER.DATA\_DIRECTORY[pefile.DIRECTORY\_ENTRY['IMAGE\_DIRECTORY\_ENTRY\_EXCEPTION']].VirtualAddress)

directory\_entries.append(

pe.OPTIONAL\_HEADER.DATA\_DIRECTORY[pefile.DIRECTORY\_ENTRY['IMAGE\_DIRECTORY\_ENTRY\_SECURITY']].VirtualAddress)

return directory\_entries

def get\_section\_stats():

section\_stats = []

section\_stats.append(len(pe.sections))

section\_stats.append(float("inf"))

section\_stats.append(0)

section\_stats.append(float("inf"))

section\_stats.append(0)

section\_stats.append(float("inf"))

section\_stats.append(0)

section\_stats.append(0)

section\_stats.append(float("inf"))

section\_stats.append(0)

section\_stats.append(float("inf"))

section\_stats.append(0)

section\_stats.append(float("inf"))

section\_stats.append(0)

section\_stats.append(float("inf"))

for section in pe.sections:

entropy = section.get\_entropy()

section\_stats[1] = min(section\_stats[1], entropy)

section\_stats[2] = max(section\_stats[2], entropy)

section\_stats[3] = min(

section\_stats[3], section.SizeOfRawData)

section\_stats[4] = max(

section\_stats[4], section.SizeOfRawData)

section\_stats[5] = min(section\_stats[5], section.Misc\_VirtualSize)

section\_stats[6] = max(section\_stats[6], section.Misc\_VirtualSize)

section\_stats[7] = max(section\_stats[7], section.PointerToRawData)

section\_stats[8] = min(section\_stats[8], section.PointerToRawData)

section\_stats[9] = max(

section\_stats[9], section.VirtualAddress)

section\_stats[10] = min(

section\_stats[10], section.VirtualAddress)

section\_stats[11] = max(

section\_stats[11], section.PointerToRawData)

section\_stats[12] = min(

section\_stats[12], section.PointerToRawData)

section\_stats[13] = max(

section\_stats[13], section.Characteristics)

section\_stats[14] = min(

section\_stats[14], section.Characteristics)

return section\_stats

header = pe.FILE\_HEADER

opt\_header = pe.OPTIONAL\_HEADER

features = []

features.append(header.Machine)

features.append(header.NumberOfSections)

features.append(header.TimeDateStamp)

features.append(header.PointerToSymbolTable)

features.append(header.NumberOfSymbols)

features.append(header.SizeOfOptionalHeader)

features.append(header.Characteristics)

features.append(opt\_header.Magic)

features.append(opt\_header.MajorLinkerVersion)

features.append(opt\_header.MinorLinkerVersion)

features.append(opt\_header.SizeOfCode)

features.append(opt\_header.SizeOfInitializedData)

features.append(opt\_header.SizeOfUninitializedData)

features.append(opt\_header.AddressOfEntryPoint)

features.append(opt\_header.BaseOfCode)

features.append(opt\_header.ImageBase)

features.append(opt\_header.SectionAlignment)

features.append(opt\_header.FileAlignment)

features.append(opt\_header.MajorOperatingSystemVersion)

features.append(opt\_header.MinorOperatingSystemVersion)

features.append(opt\_header.MajorImageVersion)

features.append(opt\_header.MinorImageVersion)

features.append(opt\_header.MajorSubsystemVersion)

features.append(opt\_header.MinorSubsystemVersion)

features.append(opt\_header.SizeOfHeaders)

features.append(opt\_header.CheckSum)

features.append(opt\_header.SizeOfImage)

features.append(opt\_header.Subsystem)

features.append(opt\_header.DllCharacteristics)

features.append(opt\_header.SizeOfStackReserve)

features.append(opt\_header.SizeOfStackCommit)

features.append(opt\_header.SizeOfHeapReserve)

features.append(opt\_header.SizeOfHeapCommit)

features.append(opt\_header.LoaderFlags)

features.append(opt\_header.NumberOfRvaAndSizes)

features = features+get\_section\_stats()+get\_directory\_entries()

list\_of\_items = ["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", "SectionsLength", "SectionMinEntropy", "SectionMaxEntropy", "SectionMinRawsize", "SectionMaxRawsize", "SectionMinVirtualsize", "SectionMaxVirtualsize", "SectionMaxPhysical", "SectionMinPhysical", "SectionMaxVirtual", "SectionMinVirtual", "SectionMaxPointerData", "SectionMinPointerData", "SectionMaxChar", "SectionMainChar", "ImageDirectoryEntryExport", "ImageDirectoryEntryImport", "ImageDirectoryEntryResource", "ImageDirectoryEntryException", "ImageDirectoryEntrySecurity"]

dict = {}

for i in range(0, len(features)):

dict[list\_of\_items[i]] = features[i]

print(dict)

print("hi")

return dict

app.mount("/static", StaticFiles(directory="static"), name="static")

templates = Jinja2Templates(directory="templates")

@app.get("/")

async def read\_items(request: Request):

return templates.TemplateResponse('index.html', {"request": request})

@app.post("/predict")

async def create\_upload\_file(request: Request, file: UploadFile = File(...)):

try:

print("hi")

print(file.read())

pe = pefile.PE(data=file.file.read())

dict\_df = pd.DataFrame.from\_dict([extract(pe)])

print(dict\_df)

# print(pdf)

# print(extract(pe))

# print(model.predict(dict\_df))

# # y\_pred = model.predict(X\_test)

# # y\_pred q

# # print(confusion\_matrix(Y\_test, y\_pred))

# # print(classification\_report(Y\_test, y\_pred))

# # print(metrics.accuracy\_score(Y\_test, y\_pred))

Y\_pred = model.predict(dict\_df)

print("Ypred is " , Y\_pred)

if (Y\_pred == [0]):

return templates.TemplateResponse('pg2.html', {"request": request})

else:

return templates.TemplateResponse('yes.html', {"request": request})

except Exception as e:

print(e)

# Result:

Ensemble Model:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1 score** | **Support** |
| **0** | 0.94 | 0.96 | 0.95 | 962 |
| **1** | 0.99 | 0.98 | 0.98 | 2961 |
| **Accuracy** |  |  | 0.98 | 3923 |
| **Macro Average** | 0.97 | 0.97 | 0.97 | 3923 |
| **Weighted Average** | 0.98 | 0.98 | 0.98 | 3923 |

Adaboost Model:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1 score** | **Support** |
| **0** | 0.97 | 0.98 | 0.97 | 970 |
| **1** | 0.99 | 0.99 | 0.99 | 2953 |
| **Accuracy** |  |  | 0.99 | 3923 |
| **Macro Average** | 0.98 | 0.98 | 0.98 | 3923 |
| **Weighted Average** | 0.99 | 0.99 | 0.99 | 3923 |

Summary of All Models Used:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model Used** | **Accuracy**  **(%)** | **Precision** | **Recall** | **F1 score** |
| Gaussian Naïve Bayes | 32.24 | 0.93 | 0.32 | 0.41 |
| Random Forest | 98.72 | 0.99 | 0.99 | 0.99 |
| Adaboost | 99.29 | 0.99 | 0.99 | 0.99 |
| Ensemble | 98.39 | 0.99 | 0.99 | 0.99 |
| Random Forest with PCA | 99.41 | 0.99 | 0.99 | 0.99 |