Malware detection with machine learning and Deep learning || data science project

My LinkedIn : Ismail Ameskour

- Build ML aand DL models to detect malware (we will make three algorithms : random forest , logistic regression ,neural network (DL))
- Make comparison of all model and select the best model

in datasets :

- 41,323(original file) binaries (exe,dll) - legitimate
- 96,724 malware files

In [166…
```
import pandas as pd
import sklearn
from sklearn.feature_selection import SelectFromModel
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_validate
import matplotlib.pyplot as plt
```

In [167…
```
MalwareDataset = pd.read_csv('MalwareData.csv', sep='|', low_memory =True)
```

In [168…
```
MalwareDataset.head()
```

Out[168…

	Name	md5	Machine	SizeOfOptionalHeader	Characteristics	MajorLinkerVersion	MinorLinkerVersion	SizeOfCode	SizeOfInitializedData	SizeOfUninitializedI
0	memtest.exe	631ea355665f28d4707448e442fbf5b8	332	224	258	9	0	361984	115712	
1	ose.exe	9d10f99a6712e28f8acd5641e3a7ea6b	332	224	3330	9	0	130560	19968	
2	setup.exe	4d92f518527353c0db88a70fddcfd390	332	224	3330	9	0	517120	621568	
3	DW20.EXE	a41e524f8d45f0074fd07805ff0c9b12	332	224	258	9	0	585728	369152	
4	dwtrig20.exe	c87e561258f2f8650cef999bf643a731	332	224	258	9	0	294912	247296	

5 rows × 57 columns

In [169…
```
MalwareDataset.shape
```

Out[169… (138047, 57)

In [170…
```
MalwareDataset.describe()
```

Out[170…

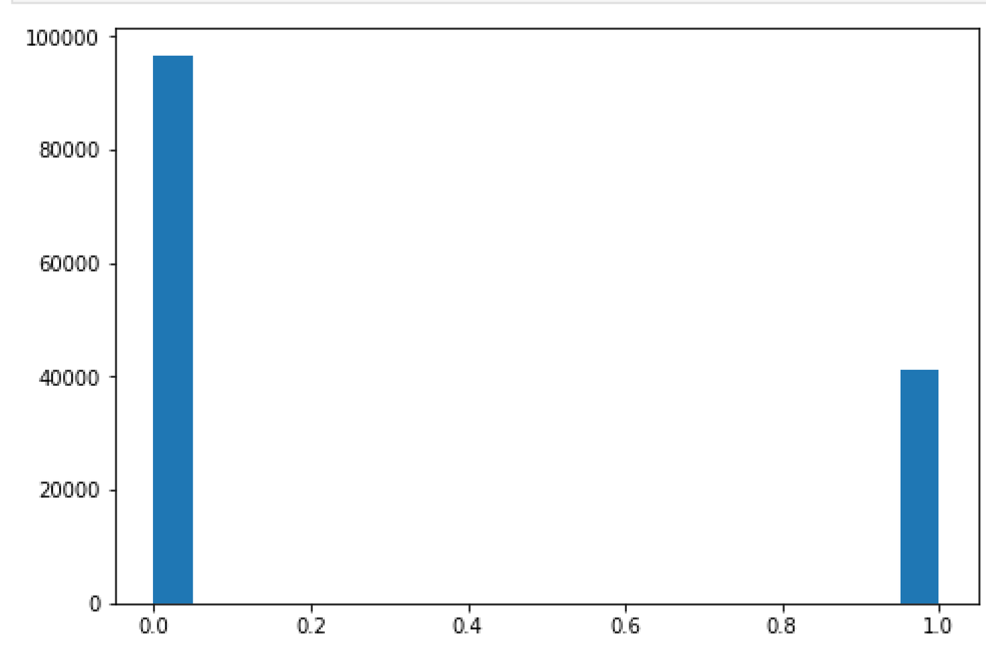
	Machine	SizeOfOptionalHeader	Characteristics	MajorLinkerVersion	MinorLinkerVersion	SizeOfCode	SizeOfInitializedData	SizeOfUninitializedData	AddressOfEntryPoint	BaseOfC
count	138047.000000	138047.000000	138047.000000	138047.000000	138047.000000	1.380470e+05	1.380470e+05	1.380470e+05	1.380470e+05	1.380470e
mean	4259.069274	225.845632	4444.145994	8.619774	3.819286	2.425956e+05	4.504867e+05	1.009525e+05	1.719561e+05	5.779845e
std	10880.347245	5.121399	8186.782524	4.088757	11.862675	5.754485e+06	2.101599e+07	1.635288e+07	3.430553e+06	5.527658e
min	332.000000	224.000000	2.000000	0.000000	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e
25%	332.000000	224.000000	258.000000	8.000000	0.000000	3.020800e+04	2.457600e+04	0.000000e+00	1.272100e+04	4.096000e
50%	332.000000	224.000000	258.000000	9.000000	0.000000	1.136640e+05	2.631680e+05	0.000000e+00	5.288300e+04	4.096000e
75%	332.000000	224.000000	8226.000000	10.000000	0.000000	1.203200e+05	3.850240e+05	0.000000e+00	6.157800e+04	4.096000e
max	34404.000000	352.000000	49551.000000	255.000000	255.000000	1.818587e+09	4.294966e+09	4.294941e+09	1.074484e+09	2.028711e

8 rows × 55 columns

In [171…
```
Legit = MalwareDataset[0:41323].drop(['legitimate'], axis=1)
Malware = MalwareDataset[41323::].drop(['legitimate'], axis=1)
print("The shape of the legit dataset is : %s samples, %s features"%(Legit.shape[0], Legit.shape[1]))
print("The shape of the mal dataset is : %s samples, %s features"%(Malware.shape[0], Malware.shape[1]))
```

```
The shape of the legit dataset is : 41323 samples, 56 features
The shape of the mal dataset is : 96724 samples, 56 features
```

In [172…
```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.hist(MalwareDataset['legitimate'],20)
plt.show()
# there are 2 number of legitimate in datasets is 1 and 0 , for 1 is original file while 0 is malware
# so we can see output that  0 which were that malware files these are 96 000 something , and then 1 which were  original
#files somthing 40000
```



Data cleaning

In [173…
```
y = MalwareDataset['legitimate']
MalwareDataset=MalwareDataset.drop(['legitimate'],axis=1)
```

Data preprocessing

In [174…
```
MalwareDataset=MalwareDataset.drop(['Name'],axis=1)
MalwareDataset=MalwareDataset.drop(['md5'],axis=1)
print("the Name and md5 variables are removed successfully")
```

```
the Name and md5 variables are removed successfully
```

Spliting the dataset into test and train

In [175…
```
#from sklearn.model_selection import train_test_split
#X_train , X_test ,y_train , y_test = train_test_split(MalwareDataset, test_size=0.2 , random_state=42)

X_train, X_test, y_train, y_test = train_test_split(MalwareDataset, y,test_size=0.2, random_state=42)
```

In [176…
```
X_train.shape
```

Out[176… (110437, 54)

Model Building

1- Random Forest

In [177…
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification
clf = RandomForestClassifier(max_depth=2 , random_state=0)
randomModel = clf.fit(X_train , y_train)
```

Random forest Evaluation on test data

In [178…
```
from sklearn.metrics import f1_score , accuracy_score, plot_confusion_matrix , auc , confusion_matrix
```

In [179…
```
#accuracy on the train dataset
train_pred = randomModel.predict(X_train)
accuracy_score(y_train,train_pred)
```

Out[179… 0.9828318407780001

In [180…
```
#accuracy on the test dataset
prediction=randomModel.predict(X_test)
accuracy_score(y_test,prediction)
```

Out[180… 0.9838102136906918

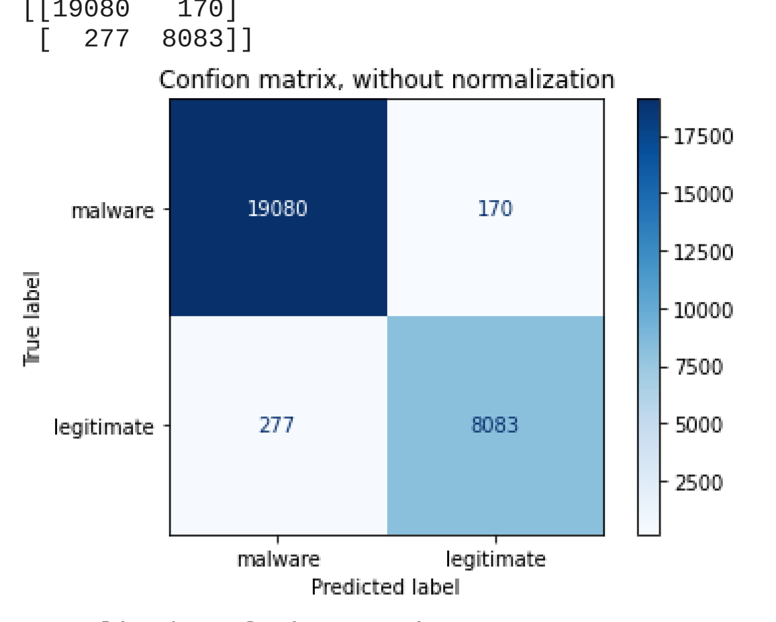
In [181…
```
f1_score(y_test,prediction)
```

Out[181… 0.9730933606212002

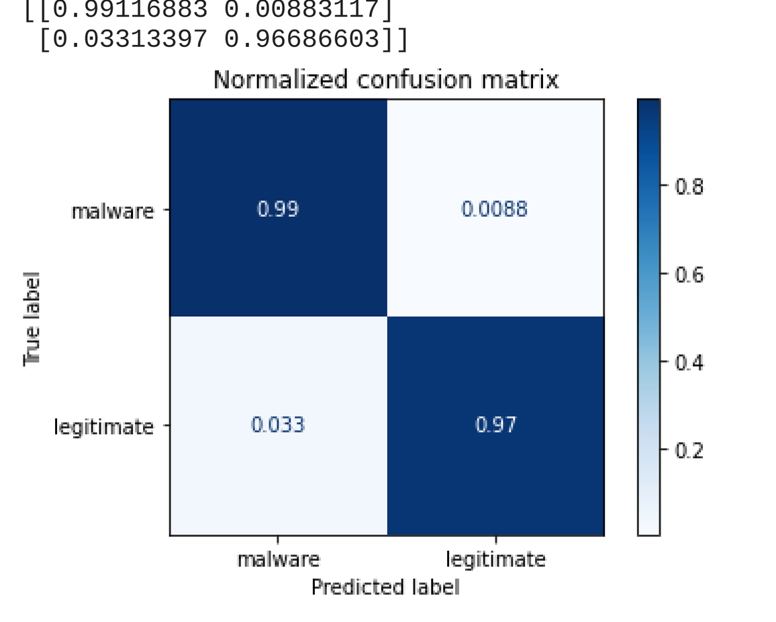
Confusion matrix

In [182…
```
titles_options = [("Confion matrix, without normalization", None),
                  ("Normalized confusion matrix",'true')]

for title , normalize in titles_options :
    disp = plot_confusion_matrix(randomModel, X_test, y_test,
                                 display_labels=['malware', 'legitimate'],
                                 cmap=plt.cm.Blues,
                                 normalize=normalize)

    disp.ax_.set_title(title)

    print(title)
    print(disp.confusion_matrix)

    plt.show()
```

```
Confion matrix, without normalization
[[19080   170]
 [  277  8083]]
```




```
Normalized confusion matrix
[[0.99116883 0.00883117]
 [0.03313397 0.96686603]]
```



2- Logistic regression

In [183…
```
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression(random_state=0)
logModel=clf.fit(X_train, y_train)
```

```
c:\users\ismail\appdata\local\programs\python\python39\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to con
verge (status=2):
ABNORMAL_TERMINATION_IN_LNSRCH.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Mode Evaluaiton

In [184…
```
# Accuracy on the dataset
train_log=logModel.predict(X_train)
accuracy_score(y_train,train_log)
```

Out[184… 0.7015221347917817

In [185…
```
#Accuracy on the dataset
pred=logModel.predict(X_test)
accuracy_score(y_test,pred)
```

Out[185… 0.6972111553784861

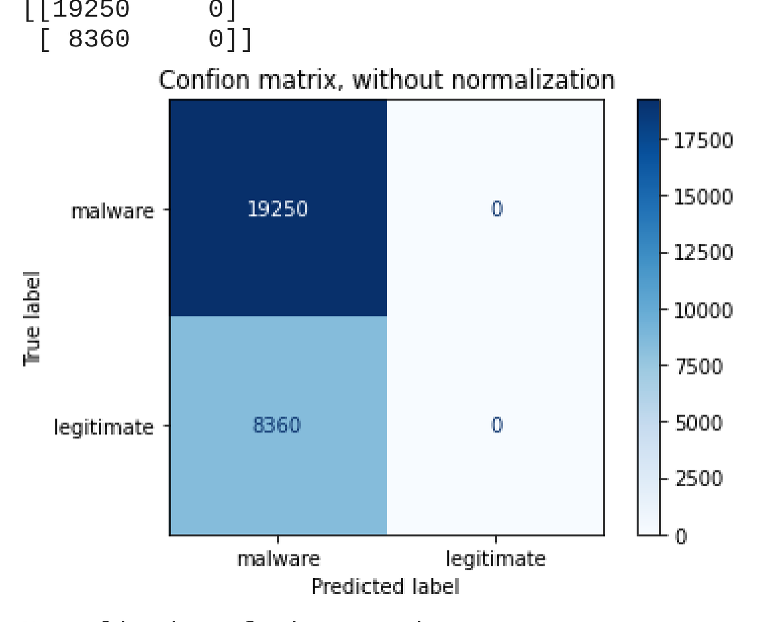
In [186…
```
f1_score(y_test,pred)
```

Out[186… 0.0

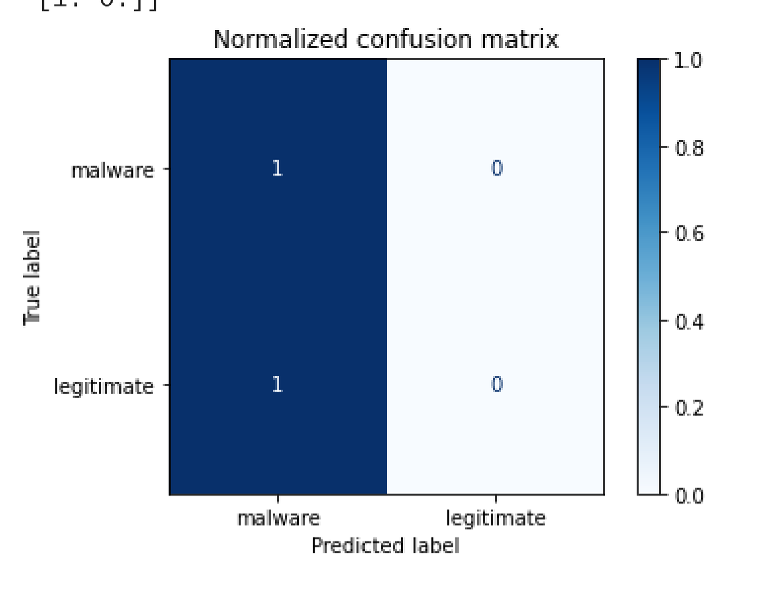
Confusion matrix

In [187…
```
titles_options = [("Confion matrix, without normalization", None),
                  ("Normalized confusion matrix",'true')]

for title , normalize in titles_options :
    disp = plot_confusion_matrix(logModel, X_test, y_test,
                                 display_labels=['malware', 'legitimate'],
                                 cmap=plt.cm.Blues,
                                 normalize=normalize)

    disp.ax_.set_title(title)

    print(title)
    print(disp.confusion_matrix)

    plt.show()
```

```
Confion matrix, without normalization
[[19250     0]
 [ 8360     0]]
```




```
Normalized confusion matrix
[[1. 0.]
 [1. 0.]]
```



3- Neural Netework

In [188…
```
import tensorflow as tf

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

In [189…
```
#Define model
model=Sequential()
model.add(Dense(16,input_dim=54,activation="relu"))
model.add(Dense(8,activation="relu"))
model.add(Dense(4,activation="relu"))
model.add(Dense(1,activation="sigmoid"))
model.summary() # Print model summary
```

```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_13 (Dense)            (None, 16)                880

 dense_14 (Dense)            (None, 8)                 136

 dense_15 (Dense)            (None, 4)                 36

 dense_16 (Dense)            (None, 1)                 5

=================================================================
Total params: 1,057
Trainable params: 1,057
Non-trainable params: 0
_________________________________________________________________
```

In [190…
```
#Compile model
model.compile(loss="binary_crossentropy",optimizer="rmsprop",metrics=["accuracy"])
```

In [191…
```
#fit Modeel
model.fit(X_train,y_train,epochs=5,batch_size=33)
```

```
Epoch 1/5
3347/3347 [==============================] - 11s 3ms/step - loss: 5569130.5000 - accuracy: 0.9536
Epoch 2/5
3347/3347 [==============================] - 11s 3ms/step - loss: 4071678.5000 - accuracy: 0.9507
Epoch 3/5
3347/3347 [==============================] - 10s 3ms/step - loss: 6033113.5000 - accuracy: 0.9501
Epoch 4/5
3347/3347 [==============================] - 11s 3ms/step - loss: 2847725.2500 - accuracy: 0.9523
Epoch 5/5
3347/3347 [==============================] - 11s 3ms/step - loss: 2141470.7500 - accuracy: 0.9512
```

Out[191…
```
<keras.callbacks.History at 0x12bb60195e0>
```

Model evaluation

In [192…
```
#Accuracy on the test dataset
y_prediction=model.predict(X_test)
y_prediction=[1 if y>=0.5 else 0 for y in y_prediction]
accuracy_score(y_test,y_prediction)
```

```
863/863 [==============================] - 3s 3ms/step
```

Out[192… 0.9590365809489315

In [193…
```
confusion_matrix(y_test,y_prediction)
```

Out[193…
```
array([[19002,   248],
       [  883,  7477]], dtype=int64)
```

In [194…
```
f1_score(y_test,y_prediction)
```

Out[194… 0.9296860428971092