21/01/2024

Telecommunication Software

Report 5

Jehanne Sarrazin

230AEM053

You will find the code from the Task 1 and 2 starting page 3. As always, all can also be found on the github repository : https://github.com/Niennaaa/TelecommunicationSoftware

Task 1 : Intrusion detection

You can find the ML algorithm used the Annex and in the PDF “TASK1&2”. The notebook containing the corresponding code is the file “TASK1&2.ipynb”.

For this task I downloaded the files in the dataset shown in class and reproduced the Machine Learning code used to analyse it. I had to tweak the code a little as some of the code didn’t work on my end.

Task 2 : Binary Search Tree

You can find the code used to build the required class in the Annex and in the PDF “TASK1&2”. The notebook containing the corresponding code is the file “TASK1&2.ipynb”.

The code works, as shown by the tests used at the end, but I believe the code used to delete a node within a tree could be improved. I chose to replace the value to remove by the next greater numerical value within the tree as, this way, the property of the tree would be conserved. However, this led me to just erase the given tree and build a new one in the new order of value calculated by my function. So, I believe this function can definitely be improved.

Task 3 : SDN Traffic classification with DT

Unfortunately, I didn’t have the time to complete this task, but I still analysed and compared the ID3 and the CART algorithm. ID3 and CART are two different decision tree algorithms that can be used for classification. They work by learning simple decision rules inferred from the data features and splitting the data into smaller and smaller subsets based on the feature values.

The ID3 algorithm only works for binary classification and base its splitting of the data on the difference of value between a node and its parent. The CART algorithm however is a Classification And Regression Tree and is not limited to binary classification. Its splitting decision is based on the goal to minimize the Gini Impurity which measures how often an element would be incorrectly labelled if labelled randomly inside the set.

I found this published article : [(PDF) Performance Evaluation Among ID3, C4.5, and CART Decision Tree Algorithm (researchgate.net)]() comparing the performances of these algorithms, and found that the CART algorithm seems to be better in every way to the ID3 algorithm : it is faster, more accurate and can handle situations such as missing values inside the set.

I plan on coding these algorithms and test their performances by myself, but I will not have the time to do it before the end of the semester. I might add my results to my github later this year, separately from this class.

Annex:

In [1]:
```
#TASK 1 : INTRUSION DETECTION
# /!\ I used the code SEEN IN THE LECTURE for this task
# /!\ This is NOT original content
# The code was tweaked a little as it wouldn't run right on my side but this is it.

#Necessary imports
!pip install --upgrade pandas --user

import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE
matplotlib.use('TkAgg')

#path = 'C:\\Users\\Jehanne\\OneDrive - De Vinci\\RTU\\TelecomSoftware\\labeled_flo
path = "C:\\Users\\Jehanne\\OneDrive - De Vinci\\RTU\\TelecomSoftware\\labeled_flow
df = pd.read_xml(path)

print(df.info)
```

```
Requirement already satisfied: pandas in c:\users\jehanne\appdata\roaming\python\pyt
hon38\site-packages (2.0.3)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\jehanne\appdata\ro
aming\python\python38\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\jehanne\anaconda3\lib\site-p
ackages (from pandas) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\jehanne\anaconda3\lib\site
-packages (from pandas) (2022.1)
Requirement already satisfied: numpy>=1.20.3 in c:\users\jehanne\appdata\roaming\pyt
hon\python38\site-packages (from pandas) (1.24.4)
Requirement already satisfied: six>=1.5 in c:\users\jehanne\anaconda3\lib\site-packa
ges (from python-dateutil>=2.8.2->pandas) (1.16.0)
WARNING: Ignoring invalid distribution -umpy (c:\users\jehanne\anaconda3\lib\site-pa
ckages)
WARNING: Ignoring invalid distribution -umpy (c:\users\jehanne\anaconda3\lib\site-pa
ckages)
```

```
<bound method DataFrame.info of                   appName  totalSourceBytes  totalDe
stinationBytes  \
0                 HTTPWeb             38007             1273547
1                 HTTPWeb             51524             1705876
2                     DNS              2845               18948
3                 HTTPWeb              4291               92920
4                 HTTPWeb              4540              113303
...                   ...               ...                 ...
142366                DNS               104                 348
142367                DNS                93                 463
142368  HTTPImageTransfer               251                  66
142369                DNS                87                 526
142370  HTTPImageTransfer               709                6264

        totalDestinationPackets  totalSourcePackets  \
0                           894                 569
1                          1192                 771
2                            52                  35
3                            71                  47
4                            83                  61
...                         ...                 ...
142366                        1                   1
142367                        1                   1
142368                        1                   3
142369                        1                   1
142370                        7                   7

                                    sourcePayloadAsBase64  \
0                                                    None
1                                                    None
2                                                    None
3                                                    None
4       R0VUIC9kZXNpZ24wNS9pbWFnZXMvMjAwOS8xMjA3L2Rhdm...
...                                                   ...
142366  lM0BAAABAAAAAAABDmQycmRmbml6ZW41YXBsCmNsb3VkZn...
142367  qx8BAAABAAAAAAABBWExNzg0AWwGYWthbWFpA25ldAAAAQ...
142368                                               None
142369  sm0BAAABAAAAAAABAmExBXR3aW1nA2NvbQAAAQABAAApEA...
142370                                               None

                                       sourcePayloadAsUTF  \
0                                                    None
1                                                    None
2                                                    None
3                                                    None
4       GET /design05/images/2009/1207/david20091207_2...
...                                                   ...
142366            ......d2rdfnizen5aplcloudfront.net..)..
142367                      ......a1784.l.akamai.net..)..
142368                                               None
142369                            .m....a1.twimg.com..)..
142370                                               None

                               destinationPayloadAsBase64  \
0                                                    None
1                                                    None
```

```
2                                                    None
3                                                    None
4                                                    None
...                                                   ...
142366  lM2BgAABAAkAAgADDmQycmRmbml6ZW41YXBsCmNsb3VkZn...
142367  qx+BgAABAAQACQAKBWExNzg0AWwGYWthbWFpA25ldAAAAQ...
142368                                               None
142369  sm2BgAABAAYACQAKAmExBXR3aW1nA2NvbQAAAQABwAwABQ...
142370                                               None

                                  destinationPayloadAsUTF direction  \
0                                                    None       L2R
1                                                    None       L2R
2                                                    None       L2L
3                                                    None       L2R
4                                                    None       L2R
...                                                   ...       ...
142366  ........d2rdfnizen5aplcloudfront.net......,..d...       L2R
142367  .......a1784.l.akamai.net........H........H..3...       L2R
142368                                               None       L2R
142369  .m.....a1.twimg.com.........a1.twimg.comedgesu...       L2R
142370                                               None       L2R

       sourceTCPFlagsDescription destinationTCPFlagsDescription  \
0                          R,P,A                            P,A
1                          R,P,A                            P,A
2                            NaN                            NaN
3                            P,A                            P,A
4                          F,P,A                          F,P,A
...                          ...                            ...
142366                       NaN                            NaN
142367                       NaN                            NaN
142368                     S,P,A                            S,A
142369                       NaN                            NaN
142370                     S,P,A                          S,P,A

               source protocolName  sourcePort    destination  \
0       192.168.1.101       tcp_ip        4646    89.234.1.43
1       192.168.1.101       tcp_ip        4722    89.234.1.43
2       192.168.2.111       udp_ip        1654  192.168.5.122
3       192.168.1.101       tcp_ip        1169  142.166.14.72
4       192.168.2.111       tcp_ip        4345  142.166.14.85
...               ...          ...         ...            ...
142366  192.168.5.122       udp_ip       60428   198.164.30.2
142367  192.168.5.122       udp_ip       59052   198.164.30.2
142368  192.168.3.115       tcp_ip        2756   203.73.24.75
142369  192.168.5.122       udp_ip        7320   198.164.30.2
142370  192.168.1.101       tcp_ip        2080   72.246.31.72

        destinationPort        startDateTime         stopDateTime     Tag
0                    80  2010-06-16T10:36:37  2010-06-16T11:02:07  Normal
1                    80  2010-06-16T10:38:51  2010-06-16T11:03:33  Normal
2                    53  2010-06-16T10:48:35  2010-06-16T11:04:24  Normal
3                    80  2010-06-16T10:51:32  2010-06-16T11:00:16  Normal
4                    80  2010-06-16T10:52:32  2010-06-16T11:03:25  Normal
...                 ...                  ...                  ...     ...
```

```
142366               53  2010-06-16T15:58:58  2010-06-16T15:58:58  Normal
142367               53  2010-06-16T15:58:59  2010-06-16T15:58:59  Normal
142368               80  2010-06-16T15:58:59  2010-06-16T15:58:59  Normal
142369               53  2010-06-16T15:58:59  2010-06-16T15:58:59  Normal
142370               80  2010-06-16T15:58:59  2010-06-16T15:58:59  Normal

[142371 rows x 20 columns]>
```

In [9]:

```python
#Building and training

AppCount = pd.value_counts(df['appName'])
AttackCount = pd.value_counts(df['Tag'])
AttackDataframe = pd.DataFrame(df.loc[df['Tag']=='Attack'])
AttackCount2 = pd.value_counts(AttackDataframe['appName'])
NormalDataframe = pd.DataFrame(df.loc[df["Tag"]=="Normal"])
NormalDataframeY = NormalDataframe[["Tag"]]
AttackDataframeY = AttackDataframe[["Tag"]]

AttackDataframe = AttackDataframe[["totalSourceBytes", "totalDestinationBytes", \
    "totalDestinationPackets", "totalSourcePackets", "sourcePort", "destinationPort

NormalDataframe = NormalDataframe[["totalSourceBytes", "totalDestinationBytes", \
    "totalDestinationPackets", "totalSourcePackets", "sourcePort", "destinationPort


NormalDataframeY = NormalDataframeY.to_numpy()
NormalDataframeY = NormalDataframeY.ravel()
labels, uniques = pd.factorize(NormalDataframeY)
NormalDataframeY = labels
NormalDataframeY = NormalDataframeY.ravel()


AttackDataframeY = AttackDataframeY.to_numpy()
AttackDataframeY = AttackDataframeY.ravel()
labelsS, uniquesS = pd.factorize(AttackDataframeY)
AttackDataframeY = labelsS
AttackDataframeY = AttackDataframeY.ravel()

indices_zero = AttackDataframeY ==0
AttackDataframeY[indices_zero] = 1

from sklearn.model_selection import train_test_split
X_train_N, X_test_N, Y_train_N, Y_test_N = train_test_split(NormalDataframe, \
    NormalDataframeY, random_state = 0, test_size = 8000)
X_train_A, X_test_A, Y_train_A, Y_test_A = train_test_split(AttackDataframe, \
    AttackDataframeY, random_state = 0, test_size = 0.3)

X_train = pd.concat([X_train_N, X_train_A])
X_train = X_train.sample(frac=1, random_state = 42)

X_test = pd.concat([X_test_N, X_test_A])
X_test = X_train.sample(frac=1, random_state = 42)

Y_train_N = pd.DataFrame(Y_train_N)
Y_train_A = pd.DataFrame(Y_train_A)
Y_train = pd.concat([Y_train_N, Y_train_A])
```

```python
Y_train = pd.DataFrame(Y_train)
Y_train = Y_train.sample(frac=1, random_state = 42)

Y_test_N = pd.DataFrame(Y_test_N)
Y_test_A = pd.DataFrame(Y_test_A)
Y_test = pd.concat([Y_test_N, Y_test_A])
Y_test= pd.DataFrame(Y_test)
Y_test = Y_train.sample(frac=1, random_state = 42)

from sklearn.model_selection import train_test_split
X_train_N, X_test_N, Y_train_N, Y_test_N = train_test_split(NormalDataframe, \
    NormalDataframeY, random_state = 0, test_size = 8000)
X_train_A, X_test_A, Y_train_A, Y_test_A = train_test_split(AttackDataframe, \
    AttackDataframeY, random_state = 0, test_size = 0.3)

X_train = pd.concat([X_train_N, X_train_A])
X_train = X_train.sample(frac=1, random_state = 42)

X_test = pd.concat([X_test_N, X_test_A])
X_test = X_train.sample(frac=1, random_state = 42)


Y_train_N = pd.Series(Y_train_N, name='Tag')
Y_train_A = pd.Series(Y_train_A, name='Tag')

Y_train = pd.concat([Y_train_N, Y_train_A])
Y_train = pd.DataFrame(Y_train)
Y_train = Y_train.sample(frac=1, random_state = 42)

Y_test_N = pd.Series(Y_test_N, name='Tag')
Y_test_A = pd.Series(Y_test_A, name='Tag')

Y_test = pd.concat([Y_test_N, Y_test_A])
Y_test= pd.DataFrame(Y_test)
Y_test = Y_train.sample(frac=1, random_state = 42)
```

In [10]:

```python
#Graph

transform = TSNE
X = X_test
trans = transform(n_components=2)
X_reduced = trans.fit_transform(X)
Y = pd.DataFrame(Y_test)
fig, ax = plt.subplots(figsize=(7,7))

ax.scatter(X_reduced[:,0],X_reduced[:,1], c = Y.iloc[:, 0].astype('category').cat.c
           cmap="jet", alpha=0.7)
#ax.scatter(X_reduced[:,0],X_reduced[:,1], c = Y[0].astype('category').cat.codes, \
#cmap="jet", alpha=0.7, aspect="equal",)

ax.set(xlabel="$X_1$", ylabel="$X_2$", title=f"{transform.__name__} visualization o
```

Out[10]:

```
[Text(0.5, 0, '$X_1$'),
 Text(0, 0.5, '$X_2$'),
 Text(0.5, 1.0, 'TSNE visualization of IDS testing dataset')]
```

In [11]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, recall_score, precision_score, \
    f1_score, classification_report
from sklearn.model_selection import cross_val_score, KFold
clf = DecisionTreeClassifier(random_state=0)
clf.fit(X_train, Y_train)
cv = KFold(n_splits=10, random_state=0, shuffle=True)
accuracy = clf.score(X_test, Y_test)
KFold10_accuracy = cross_val_score(clf, X_train, Y_train, scoring="accuracy", cv=cv
print(KFold10_accuracy.mean())
predict = clf.predict(X_test)
cm=confusion_matrix(Y_test, predict)
precision = precision_score(Y_test, predict, average = "weighted", labels=np.unique
recall = recall_score(Y_test, predict, average = "weighted", labels=np.unique(predi
f1scoreMacro = f1_score(Y_test, predict, average = "macro", labels=np.unique(predic
print(classification_report(Y_test, predict, target_names=["Normal", "Attacks"]))
```

```
0.9999851157252362
              precision    recall  f1-score   support

      Normal       1.00      1.00      1.00    134368
     Attacks       1.00      1.00      1.00         2

    accuracy                           1.00    134370
   macro avg       1.00      1.00      1.00    134370
weighted avg       1.00      1.00      1.00    134370
```

In [48]:
```
#TASK 2 : BINARY SEARCH TREE

class Node: #Node class, with a value in root and another node in left and right
    def __init__(self, root=None, left=None, right=None):
        self.root = root
        self.left = left
        self.right = right

class BinarySearchTree: #BinarySearchTree class
    def __init__(self, numbers=None): #With a root and a size. A list of numbers
        #can be given to fill the initial tree
        self.root = None
        self.size = 0
        if numbers is not None:
            for number in numbers:
                self.insert(number)
                self.size = self.size +1

    def search(self, searching, node = None): #Search of a value within the tree
        if node is None:
            return False, node
        if node.root == searching: #If root is value, we found it
            return True, node
        if node.root>searching : #Reccurence logic
            return self.search(searching, node.left)
        else :
            return self.search(searching, node.right)
```

```
    def insert(self, number, node = None): #Insert a new value
        if self.root == None: #If tree is empty we can just insert it
            self.root = Node(number)
            return
        if node==None :
            node = self.root
        alreadyExist, r= self.search(number, self.root)
        if not alreadyExist: #if the value already exist in the tree it's ignored.
            if number < node.root:
                if node.left == None: #if it fits it sits
                    self.size = self.size+1
                    node.left = Node(number)
                    return
                else : #Reccurence logic
                    self.insert(number, node.left)
            else :
                if node.right == None:
                    self.size = self.size+1
                    node.right = Node(number)
                    return
                else :
                    self.insert(number, node.right)

    def reset(self, new_numbers=None):
        # Reset the tree with new numbers
        self.root = None
        self.size = 0
        if new_numbers!=None :
            for number in new_numbers:
                self.insert(number)
                self.size += 1

    def delete(self, value, root=None): #probably not the most efficient method
        neworder = []
        if root==None:
            root = self.root.root
        exist, node = self.search(value, self.root)
        if not exist :
            print("Node doesnt exist")
            return
        #we're going to switch the node to delete with the
        #one that comes next numerically
        allValues = self.inorder(self.root)
        preorder = self.preorder(self.root)
        for i in range(len(allValues)):
            if allValues[i]==value:
                if i+1<len(allValues):
                    new = allValues[i+1]
                else :
                    new = "HighestUp"
                break
        neworder = []
        for i in preorder:#and rebuild the tree from scratch
            if i==value and new!="HighestUp":
                neworder.append(new)
            elif i==new:
```

```
                1
            else :
                neworder.append(i)
        self.reset(neworder)

    def preorder(self, node=None, res = None): #Create a list of the
        #value of the tree in preorder order
        if node == None :
            node = self.root
        if res == None :
            res = []
        res.append(node.root)
        if node.left is not None:
            res = self.preorder(node.left, res)
        if node.right is not None:
            res = self.preorder(node.right, res)
        return res

            #pb mon inorder ne prend que les trucs >root
    def inorder(self, node=None, res = None):#Create a list of the
        #value of the tree in inorder order
        if node == None :
            node = self.root
        if res == None :
            res = []
        if node.left is not None:
            res = self.inorder(node.left, res)
        res.append(node.root)
        if node.right is not None:
            res = self.preorder(node.right, res)
        return res

    def postorder(self, node=None, res = None):#Create a list of the
        #value of the tree in postorder order
        if node == None :
            node = self.root
        if res == None :
            res = []
        if node.left is not None:
            res = self.postorder(node.left, res)
        if node.right is not None:
            res = self.postorder(node.right, res)
        res.append(node.root)
        return res
```

In [50]:

```
#Entry lists
a = [49, 38, 65, 97, 60, 76, 13, 27, 5, 1]
b = [149, 38, 65, 197, 60, 176, 13, 217, 5, 11]
c = [49, 38, 65, 97, 64, 76, 13, 77, 5, 1, 55, 50, 24]

#Creation of the trees
treeA = BinarySearchTree(a)
treeB = BinarySearchTree(b)
treeC = BinarySearchTree(c)

#FUNCTION TEST
```

```python
#Orders
print("TreeA, B, C ; preorder, inorder, postorder")
print(treeA.preorder())
print(treeA.inorder())
print(treeA.postorder())

print(treeB.preorder())
print(treeB.inorder())
print(treeB.postorder())

print(treeC.preorder())
print(treeC.inorder())
print(treeC.postorder())

#Search
print("\n TreeA, B, C; searching 50 then 60")
print(treeA.search(50, treeA.root))
print(treeA.search(60, treeA.root))
print(treeB.search(50, treeB.root))
print(treeB.search(60, treeB.root))
print(treeC.search(50, treeC.root))
print(treeC.search(60, treeC.root))

#Insert
print("\n Tree A, B, C; attempting to insert 25 twice")
treeA.insert(25)
print(treeA.preorder())
treeA.insert(25)
print(treeA.preorder())

treeB.insert(25)
print(treeB.preorder())
treeB.insert(25)
print(treeB.preorder())

treeC.insert(25)
print(treeC.preorder())
treeC.insert(25)
print(treeC.preorder())


#Delete
print("\n Tree A, B, C; attempting to delete 50 then 60")
treeA.delete(50)
treeA.delete(60)
print(treeA.preorder())

treeB.delete(50)
treeB.delete(60)
print(treeB.preorder())

treeC.delete(50)
treeC.delete(60)
print(treeC.preorder())
```

```
TreeA, B, C ; preorder, inorder, postorder
[49, 38, 13, 5, 1, 27, 65, 60, 97, 76]
[1, 5, 13, 27, 38, 49, 65, 60, 97, 76]
[1, 5, 27, 13, 38, 60, 76, 97, 65, 49]
[149, 38, 13, 5, 11, 65, 60, 197, 176, 217]
[5, 11, 13, 38, 65, 60, 149, 197, 176, 217]
[11, 5, 13, 60, 65, 38, 176, 217, 197, 149]
[49, 38, 13, 5, 1, 24, 65, 64, 55, 50, 97, 76, 77]
[1, 5, 13, 24, 38, 49, 65, 64, 55, 50, 97, 76, 77]
[1, 5, 24, 13, 38, 50, 55, 64, 77, 76, 97, 65, 49]

 TreeA, B, C; searching 50 then 60
(False, None)
(True, <__main__.Node object at 0x000001DB31C5F9A0>)
(False, None)
(True, <__main__.Node object at 0x000001DB31C61A60>)
(True, <__main__.Node object at 0x000001DB31C5F670>)
(False, None)

 Tree A, B, C; attempting to insert 25 twice
[49, 38, 13, 5, 1, 27, 25, 65, 60, 97, 76]
[49, 38, 13, 5, 1, 27, 25, 65, 60, 97, 76]
[149, 38, 13, 5, 11, 25, 65, 60, 197, 176, 217]
[149, 38, 13, 5, 11, 25, 65, 60, 197, 176, 217]
[49, 38, 13, 5, 1, 24, 25, 65, 64, 55, 50, 97, 76, 77]
[49, 38, 13, 5, 1, 24, 25, 65, 64, 55, 50, 97, 76, 77]

 Tree A, B, C; attempting to delete 50 then 60
Node doesnt exist
[49, 38, 13, 5, 1, 27, 25, 65, 97, 76]
Node doesnt exist
[38, 13, 5, 11, 25, 65, 149, 197, 176, 217]
Node doesnt exist
[49, 38, 13, 5, 1, 24, 25, 65, 64, 55, 97, 76, 77]
```