NSL_Kdd

May 18, 2021

0.1 Importer les package necessaire

[2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

[3]:
```
columns = (['duration'
,'protocol_type'
,'service'
,'flag'
,'src_bytes'
,'dst_bytes'
,'land'
,'wrong_fragment'
,'urgent'
,'hot'
,'num_failed_logins'
,'logged_in'
,'num_compromised'
,'root_shell'
,'su_attempted'
,'num_root'
,'num_file_creations'
,'num_shells'
,'num_access_files'
,'num_outbound_cmds'
,'is_host_login'
,'is_guest_login'
,'count'
,'srv_count'
,'serror_rate'
,'srv_serror_rate'
,'rerror_rate'
,'srv_rerror_rate'
,'same_srv_rate'
```

```
,'diff_srv_rate'
,'srv_diff_host_rate'
,'dst_host_count'
,'dst_host_srv_count'
,'dst_host_same_srv_rate'
,'dst_host_diff_srv_rate'
,'dst_host_same_src_port_rate'
,'dst_host_srv_diff_host_rate'
,'dst_host_serror_rate'
,'dst_host_srv_serror_rate'
,'dst_host_rerror_rate'
,'dst_host_srv_rerror_rate'
,'attack'
,'level'])
```

[4]:
```
df_train = pd.read_csv("KDDTrain+.csv",header=None, names = columns)
df_test = pd.read_csv("KDDTest+.csv",header=None, names = columns)
```

0.2 vue simple des dataframes

[5]:
```
df_train.head(10)
```

[5]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	\
0	0	tcp	ftp_data	SF	491	0	0	
1	0	udp	other	SF	146	0	0	
2	0	tcp	private	S0	0	0	0	
3	0	tcp	http	SF	232	8153	0	
4	0	tcp	http	SF	199	420	0	
5	0	tcp	private	REJ	0	0	0	
6	0	tcp	private	S0	0	0	0	
7	0	tcp	private	S0	0	0	0	
8	0	tcp	remote_job	S0	0	0	0	
9	0	tcp	private	S0	0	0	0	

	wrong_fragment	urgent	hot	…	dst_host_same_srv_rate	\
0	0	0	0	…	0.17	
1	0	0	0	…	0.00	
2	0	0	0	…	0.10	
3	0	0	0	…	1.00	
4	0	0	0	…	1.00	
5	0	0	0	…	0.07	
6	0	0	0	…	0.04	
7	0	0	0	…	0.06	
8	0	0	0	…	0.09	
9	0	0	0	…	0.05	

	dst_host_diff_srv_rate	dst_host_same_src_port_rate	\

```
0                    0.03                         0.17
1                    0.60                         0.88
2                    0.05                         0.00
3                    0.00                         0.03
4                    0.00                         0.00
5                    0.07                         0.00
6                    0.05                         0.00
7                    0.07                         0.00
8                    0.05                         0.00
9                    0.06                         0.00

   dst_host_srv_diff_host_rate  dst_host_serror_rate  \
0                         0.00                  0.00
1                         0.00                  0.00
2                         0.00                  1.00
3                         0.04                  0.03
4                         0.00                  0.00
5                         0.00                  0.00
6                         0.00                  1.00
7                         0.00                  1.00
8                         0.00                  1.00
9                         0.00                  1.00

   dst_host_srv_serror_rate  dst_host_rerror_rate  dst_host_srv_rerror_rate  \
0                      0.00                  0.05                      0.00
1                      0.00                  0.00                      0.00
2                      1.00                  0.00                      0.00
3                      0.01                  0.00                      0.01
4                      0.00                  0.00                      0.00
5                      0.00                  1.00                      1.00
6                      1.00                  0.00                      0.00
7                      1.00                  0.00                      0.00
8                      1.00                  0.00                      0.00
9                      1.00                  0.00                      0.00

    attack  level
0   normal     20
1   normal     15
2  neptune     19
3   normal     21
4   normal     21
5  neptune     21
6  neptune     21
7  neptune     21
8  neptune     21
9  neptune     21
```

[10 rows x 43 columns]

[6]:
```
df_test.head(10)
```

[6]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	…
0	0	tcp	private	REJ	0	0	0	
1	0	tcp	private	REJ	0	0	0	
2	2	tcp	ftp_data	SF	12983	0	0	
3	0	icmp	eco_i	SF	20	0	0	
4	1	tcp	telnet	RSTO	0	15	0	
5	0	tcp	http	SF	267	14515	0	
6	0	tcp	smtp	SF	1022	387	0	
7	0	tcp	telnet	SF	129	174	0	
8	0	tcp	http	SF	327	467	0	
9	0	tcp	ftp	SF	26	157	0	

	wrong_fragment	urgent	hot	…	dst_host_same_srv_rate	…
0	0	0	0	…	0.04	
1	0	0	0	…	0.00	
2	0	0	0	…	0.61	
3	0	0	0	…	1.00	
4	0	0	0	…	0.31	
5	0	0	0	…	1.00	
6	0	0	0	…	0.11	
7	0	0	0	…	1.00	
8	0	0	0	…	1.00	
9	0	0	0	…	0.50	

	dst_host_diff_srv_rate	dst_host_same_src_port_rate	…
0	0.06	0.00	
1	0.06	0.00	
2	0.04	0.61	
3	0.00	1.00	
4	0.17	0.03	
5	0.00	0.01	
6	0.72	0.00	
7	0.00	0.00	
8	0.00	0.01	
9	0.08	0.02	

	dst_host_srv_diff_host_rate	dst_host_serror_rate	…
0	0.00	0.00	
1	0.00	0.00	
2	0.02	0.00	
3	0.28	0.00	
4	0.02	0.00	
5	0.03	0.01	

```
6                         0.00                  0.00
7                         0.00                  0.01
8                         0.03                  0.00
9                         0.00                  0.00

   dst_host_srv_serror_rate  dst_host_rerror_rate  dst_host_srv_rerror_rate  \
0                      0.00                  1.00                      1.00
1                      0.00                  1.00                      1.00
2                      0.00                  0.00                      0.00
3                      0.00                  0.00                      0.00
4                      0.00                  0.83                      0.71
5                      0.00                  0.00                      0.00
6                      0.00                  0.72                      0.04
7                      0.01                  0.02                      0.02
8                      0.00                  0.00                      0.00
9                      0.00                  0.00                      0.00

         attack  level
0       neptune     21
1       neptune     21
2        normal     21
3         saint     15
4         mscan     11
5        normal     21
6        normal     21
7  guess_passwd     15
8        normal     21
9  guess_passwd      7

[10 rows x 43 columns]
```

0.3 Verification des dataframes

(verifier les duplications, et les valeurs null)

[7]:
```
def verification(df,name):
    d = df[df.duplicated()]
    Nan = df[df.isna().any(axis=1)]
    print('----------',name,' dataframe --------')
    print('Il y a {} lignes est repete'.format(len(d)))
    print('Il y a {} valeurs null'.format(len(Nan)))
    print()
    return d, Nan
```

[8]:
```
d_train, Nan_train = verification(df_train, 'training')
d_test, Nan_test = verification(df_test, 'testing')
```

```
---------- training  dataframe --------
```

```
Il y a 0 lignes est repete
Il y a 0 valeurs null

---------- testing  dataframe --------
Il y a 0 lignes est repete
Il y a 0 valeurs null
```

0.4 comparaison entre les dataframes (testing et training)

0.4.1 1. Les dimensions

[9]:
```
print("Train dataframe : ", df_train.shape)
print("Test dataframe : ", df_test.shape)
```

```
Train dataframe :  (125973, 43)
Test dataframe :  (22544, 43)
```

0.4.2 2. les colonnes avec des valeurs qualitative

[10]:
```
def QualitativeColonnes(df, name):
    print('----------',name,' dataframe --------')
    for col in df.columns:
        if df[col].dtype == 'object':
            nbCat = len(df[col].value_counts())
            print('Colonne : {} a {} categories'.format(col, nbCat))
    print('\n')
```

[11]:
```
QualitativeColonnes(df_train,'training')
QualitativeColonnes(df_test,'testing')
```

```
---------- training  dataframe --------
Colonne : protocol_type a 3 categories
Colonne : service a 70 categories
Colonne : flag a 11 categories
Colonne : attack a 23 categories


---------- testing  dataframe --------
Colonne : protocol_type a 3 categories
Colonne : service a 64 categories
Colonne : flag a 11 categories
Colonne : attack a 38 categories
```

0.4.3 Remarque:

On peut remarquer une difference entre les deux dataframe dans les deux colonnes (service e

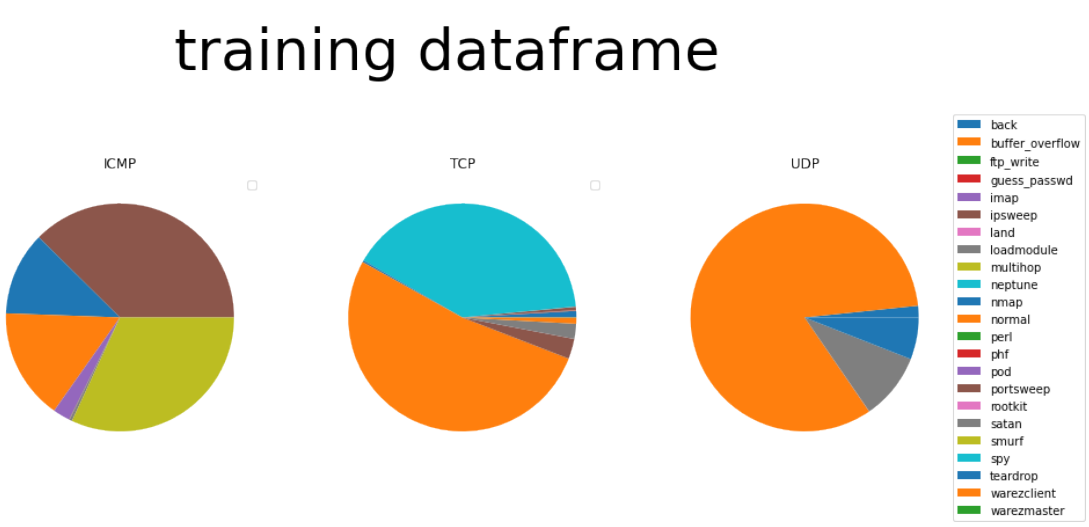
---

0.4.4 3. Visualisation des donnees de colonne attack

[12]:
```
def DiagCirculaire(df,name):
    ax = pd.crosstab(df.attack, df.protocol_type)
    ax.columns = [''] * len(ax.columns)
    ax.
↪plot(kind='pie',labels=None,subplots=True,figsize=(15,10),title=['ICMP','TCP',␣
↪'UDP'])
    plt.legend(loc='center left',labels=ax.index,bbox_to_anchor=(1, 0.5))
    plt.figtext(.5,.8,name, fontsize=50, ha='center')
```

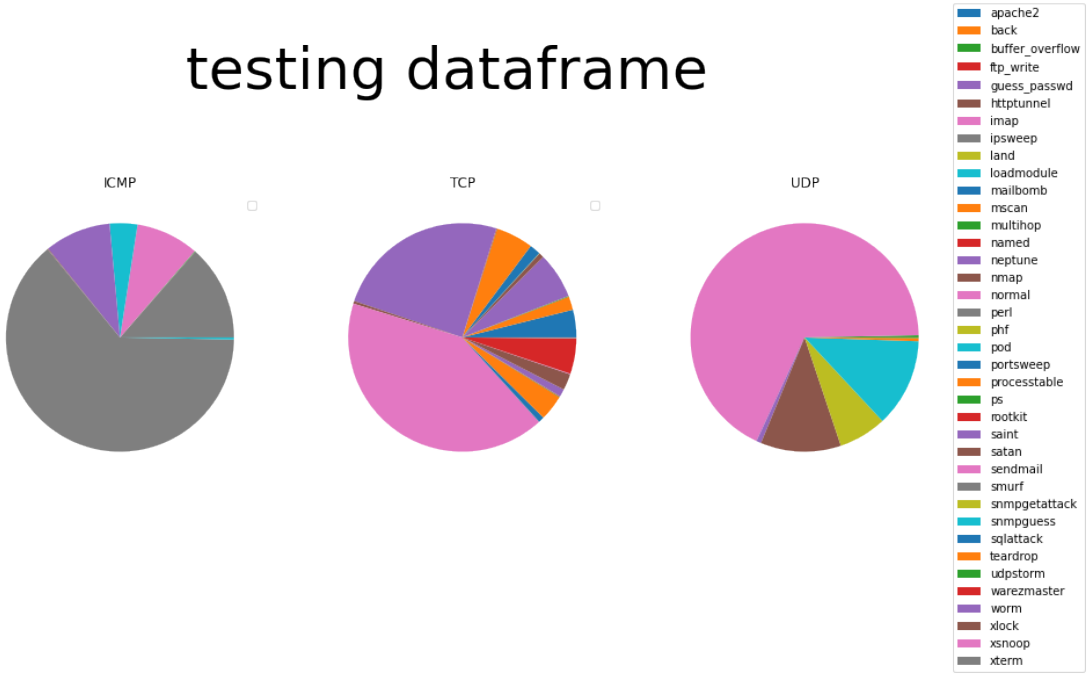
[13]:
```
DiagCirculaire(df_train,'training dataframe')
```

```
No handles with labels found to put in legend.
No handles with labels found to put in legend.
No handles with labels found to put in legend.
```



[14]:
```
DiagCirculaire(df_test,'testing dataframe')
```

```
No handles with labels found to put in legend.
No handles with labels found to put in legend.
No handles with labels found to put in legend.
```



1 Data Preprocessing

1.1 Numérisation

(conversion des données symboliques vers numériques)

1.1.1 1. Determiner les caracteristiques symbolique

(protocol_type, service, flag)

[15]:
```
Qualitative_col = ['protocol_type','service','flag']
df_train_symb = df_train[Qualitative_col]
df_test_symb = df_test[Qualitative_col]
```

[16]:
```
print('training data :')
df_train_symb.head(10)
```

```
training data :
```

[16]:

	protocol_type	service	flag
0	tcp	ftp_data	SF
1	udp	other	SF
2	tcp	private	S0
3	tcp	http	SF
4	tcp	http	SF
5	tcp	private	REJ

```
6           tcp     private   S0
7           tcp     private   S0
8           tcp  remote_job   S0
9           tcp     private   S0
```

[17]:
```
print('testing data :')
df_test_symb.head(10)
```

```
testing data :
```

[17]:
```
  protocol_type   service  flag
0           tcp   private   REJ
1           tcp   private   REJ
2           tcp  ftp_data    SF
3          icmp     eco_i    SF
4           tcp    telnet  RSTO
5           tcp      http    SF
6           tcp      smtp    SF
7           tcp    telnet    SF
8           tcp      http    SF
9           tcp       ftp    SF
```

[18]:
```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
def applyLabelEncoder(df):
    return df.apply(LabelEncoder().fit_transform)
```

[19]:
```
def dummyColonnesHeader(df):
    str = ['Protocol_','service_','flag_']
    protocol_type=sorted(df.protocol_type.unique())
    protocol_list=[str[0] + x for x in protocol_type]
    service=sorted(df.service.unique())
    service_list=[str[1] + x for x in service]
    flag=sorted(df.flag.unique())
    flag_list=[str[2] + x for x in flag]
    return protocol_list + service_list + flag_list


def applyOneHotEncoder(df_LbEn,df_symb):
    enc = OneHotEncoder()
    data_transform = enc.fit_transform(df_LbEn)
    return pd.DataFrame(data_transform.
 ↪toarray(),columns=dummyColonnesHeader(df_symb))
```

1.1.2 2. Application LabelEncoder

[20]:
```
df_train_LbEn = applyLabelEncoder(df_train_symb)
df_test_LbEn = applyLabelEncoder(df_test_symb)
```

[21]:
```
print('Training dataframe apres application de LabelEncoder :')
df_train_LbEn.head(10)
```

```
Training dataframe apres application de LabelEncoder :
```

[21]:

	protocol_type	service	flag
0	1	20	9
1	2	44	9
2	1	49	5
3	1	24	9
4	1	24	9
5	1	49	1
6	1	49	5
7	1	49	5
8	1	51	5
9	1	49	5

[22]:
```
print('testing dataframe apres application de LabelEncoder :')
df_test_LbEn.head(10)
```

```
testing dataframe apres application de LabelEncoder :
```

[22]:

	protocol_type	service	flag
0	1	45	1
1	1	45	1
2	1	19	9
3	0	13	9
4	1	55	2
5	1	22	9
6	1	49	9
7	1	55	9
8	1	22	9
9	1	18	9

1.2 Normalisation

(rendre des données dans l intervalle [0,1])

1.2.1 1. Application OneHotEncoder

[23]:
```
df_train_OneEnc = applyOneHotEncoder(df_train_LbEn,df_train_symb)
df_test_OneEnc = applyOneHotEncoder(df_test_LbEn,df_test_symb)
```

[24]:
```
print('Training dataframe apres application de OneHotEncoder :')
df_train_OneEnc.head(10)
```

```
Training dataframe apres application de OneHotEncoder :
```

[24]:

	Protocol_icmp	Protocol_tcp	Protocol_udp	service_IRC	service_X11	\
0	0.0	1.0	0.0	0.0	0.0	
1	0.0	0.0	1.0	0.0	0.0	
2	0.0	1.0	0.0	0.0	0.0	
3	0.0	1.0	0.0	0.0	0.0	
4	0.0	1.0	0.0	0.0	0.0	
5	0.0	1.0	0.0	0.0	0.0	
6	0.0	1.0	0.0	0.0	0.0	
7	0.0	1.0	0.0	0.0	0.0	
8	0.0	1.0	0.0	0.0	0.0	
9	0.0	1.0	0.0	0.0	0.0	

	service_Z39_50	service_aol	service_auth	service_bgp	service_courier	\
0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	
5	0.0	0.0	0.0	0.0	0.0	
6	0.0	0.0	0.0	0.0	0.0	
7	0.0	0.0	0.0	0.0	0.0	
8	0.0	0.0	0.0	0.0	0.0	
9	0.0	0.0	0.0	0.0	0.0	

	…	flag_REJ	flag_RSTO	flag_RSTOS0	flag_RSTR	flag_S0	flag_S1	\
0	…	0.0	0.0	0.0	0.0	0.0	0.0	
1	…	0.0	0.0	0.0	0.0	0.0	0.0	
2	…	0.0	0.0	0.0	0.0	1.0	0.0	
3	…	0.0	0.0	0.0	0.0	0.0	0.0	
4	…	0.0	0.0	0.0	0.0	0.0	0.0	
5	…	1.0	0.0	0.0	0.0	0.0	0.0	
6	…	0.0	0.0	0.0	0.0	1.0	0.0	
7	…	0.0	0.0	0.0	0.0	1.0	0.0	
8	…	0.0	0.0	0.0	0.0	1.0	0.0	
9	…	0.0	0.0	0.0	0.0	1.0	0.0	

	flag_S2	flag_S3	flag_SF	flag_SH
0	0.0	0.0	1.0	0.0
1	0.0	0.0	1.0	0.0
2	0.0	0.0	0.0	0.0
3	0.0	0.0	1.0	0.0
4	0.0	0.0	1.0	0.0
5	0.0	0.0	0.0	0.0

```
6      0.0      0.0      0.0      0.0
7      0.0      0.0      0.0      0.0
8      0.0      0.0      0.0      0.0
9      0.0      0.0      0.0      0.0

[10 rows x 84 columns]
```

[25]:
```
print('testing dataframe apres application de OneHotEncoder :')
df_test_OneEnc.head(10)
```

```
testing dataframe apres application de OneHotEncoder :
```

[25]:
```
   Protocol_icmp  Protocol_tcp  Protocol_udp  service_IRC  service_X11  \
0            0.0           1.0           0.0          0.0          0.0
1            0.0           1.0           0.0          0.0          0.0
2            0.0           1.0           0.0          0.0          0.0
3            1.0           0.0           0.0          0.0          0.0
4            0.0           1.0           0.0          0.0          0.0
5            0.0           1.0           0.0          0.0          0.0
6            0.0           1.0           0.0          0.0          0.0
7            0.0           1.0           0.0          0.0          0.0
8            0.0           1.0           0.0          0.0          0.0
9            0.0           1.0           0.0          0.0          0.0

   service_Z39_50  service_auth  service_bgp  service_courier  \
0             0.0           0.0          0.0              0.0
1             0.0           0.0          0.0              0.0
2             0.0           0.0          0.0              0.0
3             0.0           0.0          0.0              0.0
4             0.0           0.0          0.0              0.0
5             0.0           0.0          0.0              0.0
6             0.0           0.0          0.0              0.0
7             0.0           0.0          0.0              0.0
8             0.0           0.0          0.0              0.0
9             0.0           0.0          0.0              0.0

   service_csnet_ns  …  flag_REJ  flag_RSTO  flag_RSTOS0  flag_RSTR  \
0               0.0  …       1.0        0.0          0.0        0.0
1               0.0  …       1.0        0.0          0.0        0.0
2               0.0  …       0.0        0.0          0.0        0.0
3               0.0  …       0.0        0.0          0.0        0.0
4               0.0  …       0.0        1.0          0.0        0.0
5               0.0  …       0.0        0.0          0.0        0.0
6               0.0  …       0.0        0.0          0.0        0.0
7               0.0  …       0.0        0.0          0.0        0.0
8               0.0  …       0.0        0.0          0.0        0.0
9               0.0  …       0.0        0.0          0.0        0.0
```

	flag_S0	flag_S1	flag_S2	flag_S3	flag_SF	flag_SH
0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	1.0	0.0
3	0.0	0.0	0.0	0.0	1.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	1.0	0.0
6	0.0	0.0	0.0	0.0	1.0	0.0
7	0.0	0.0	0.0	0.0	1.0	0.0
8	0.0	0.0	0.0	0.0	1.0	0.0
9	0.0	0.0	0.0	0.0	1.0	0.0

[10 rows x 78 columns]

1.2.2 remarque:

d'apres la visualisation des donnees on peut remarque qu'il y un manquant de quelque valeur entre les deux dataframe

---

1.3 Equilibrage des dataframes

(ajouter les colonnes manquant dans les deux databases)

[30]:
```
col_train_OneEnc = df_train_OneEnc.columns.tolist()
col_test_OneEnc = df_test_OneEnc.columns.tolist()
```

[49]:
```
diff_list = set(col_train_OneEnc).difference(set(col_test_OneEnc))
diff_list #la list des difference entre les 2 Database pour ce qui est df de␣
 ↪one hot encodre
```

[49]:
```
{'service_aol',
 'service_harvest',
 'service_http_2784',
 'service_http_8001',
 'service_red_i',
 'service_urh_i'}
```

[38]:
```
for col in diff_list:
    df_test_OneEnc[col] = 0
```

[39]:
```
df_test_OneEnc.head(10)
```

[39]:

	Protocol_icmp	Protocol_tcp	Protocol_udp	service_IRC	service_X11	\
0	0.0	1.0	0.0	0.0	0.0	
1	0.0	1.0	0.0	0.0	0.0	

```
2            0.0           1.0           0.0          0.0          0.0
3            1.0           0.0           0.0          0.0          0.0
4            0.0           1.0           0.0          0.0          0.0
5            0.0           1.0           0.0          0.0          0.0
6            0.0           1.0           0.0          0.0          0.0
7            0.0           1.0           0.0          0.0          0.0
8            0.0           1.0           0.0          0.0          0.0
9            0.0           1.0           0.0          0.0          0.0

   service_Z39_50  service_auth  service_bgp  service_courier  \
0             0.0           0.0          0.0              0.0
1             0.0           0.0          0.0              0.0
2             0.0           0.0          0.0              0.0
3             0.0           0.0          0.0              0.0
4             0.0           0.0          0.0              0.0
5             0.0           0.0          0.0              0.0
6             0.0           0.0          0.0              0.0
7             0.0           0.0          0.0              0.0
8             0.0           0.0          0.0              0.0
9             0.0           0.0          0.0              0.0

   service_csnet_ns  …  flag_S2  flag_S3  flag_SF  flag_SH  service_red_i  \
0               0.0  …      0.0      0.0      0.0      0.0              0
1               0.0  …      0.0      0.0      0.0      0.0              0
2               0.0  …      0.0      0.0      1.0      0.0              0
3               0.0  …      0.0      0.0      1.0      0.0              0
4               0.0  …      0.0      0.0      0.0      0.0              0
5               0.0  …      0.0      0.0      1.0      0.0              0
6               0.0  …      0.0      0.0      1.0      0.0              0
7               0.0  …      0.0      0.0      1.0      0.0              0
8               0.0  …      0.0      0.0      1.0      0.0              0
9               0.0  …      0.0      0.0      1.0      0.0              0

   service_urh_i  service_harvest  service_aol  service_http_8001  \
0              0                0            0                  0
1              0                0            0                  0
2              0                0            0                  0
3              0                0            0                  0
4              0                0            0                  0
5              0                0            0                  0
6              0                0            0                  0
7              0                0            0                  0
8              0                0            0                  0
9              0                0            0                  0

   service_http_2784
0                  0
```

```
1                   0
2                   0
3                   0
4                   0
5                   0
6                   0
7                   0
8                   0
9                   0

[10 rows x 84 columns]
```

1.4 jointure des donnees

[99]:
```
df_train_new = df_train.join(df_train_OneEnc)
df_test_new = df_test.join(df_test_OneEnc)
```

[100]:
```
print("Training data : ", df_train_new.shape)
print("testing data : ", df_test_new.shape)
```

```
Training data :  (125973, 127)
testing data :  (22544, 127)
```

1.5 Nettoyage

(Selection des meilleurs attributs)

[101]:
```
def suppression(df):
    df.drop('protocol_type', axis=1, inplace=True)
    df.drop('service', axis=1, inplace=True)
    df.drop('flag', axis=1, inplace=True)
```

[102]:
```
suppression(df_train_new)
suppression(df_test_new)
```

[103]:
```
print("Training data : ", df_train_new.shape)
print("testing data : ", df_test_new.shape)
```

```
Training data :  (125973, 124)
testing data :  (22544, 124)
```

1.6 traitement à propos colonnes des attacks

[104]:
```
def Attacks(attack):
    Dos =␣
↪['neptune','land','pod','smurf','teardrop','back','worm','udpstorm','processtable','apache2
    Probe = ['ipsweep','satan','nmap','portsweep','mscan','saint']
```

```
    R2l =␣
↪['ftp_write','guess_passwd','imap','multihop','phf','spy','warezclient','warezmaster','snmp
    U2r =␣
↪['buffer_overflow','loadmodule','perl','rootkit','ps','xterm','sqlattack']
    if attack in Dos:
        atk = 1
    elif attack in Probe:
        atk = 2
    elif attack in R2l:
        atk = 3
    elif attack in U2r:
        atk = 4
    else:
        atk = 0
    return atk
```

[105]:
```
train_attack_types = df_train_new.attack.apply(Attacks)
test_attack_types = df_test_new.attack.apply(Attacks)
```

[107]:
```
df_train_new['attack'] = train_attack_types
df_test_new['attack'] = test_attack_types
```

[110]:
```
print("Training data : ")
print(df_train_new.attack.value_counts())
print("\ntesting data : ")
print(df_test_new.attack.value_counts())
```

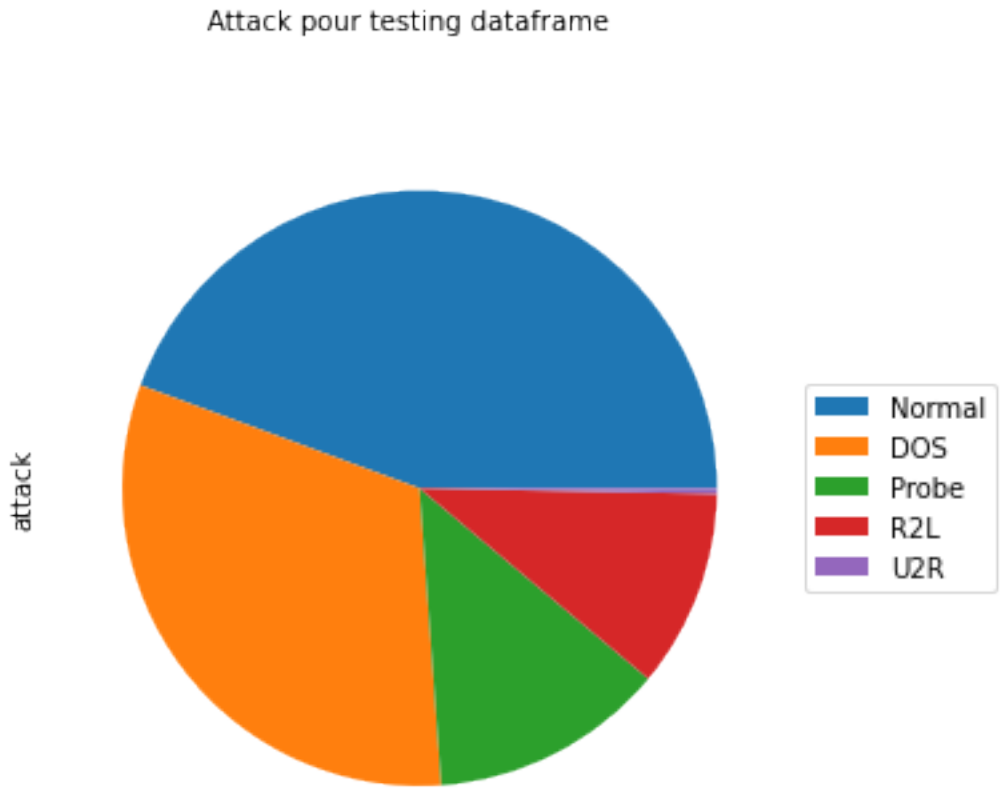
```
Training data :
0    67343
1    45927
2    11656
3      995
4       52
Name: attack, dtype: int64

testing data :
0    10004
1     7167
3     2885
2     2421
4       67
Name: attack, dtype: int64
```

[133]:
```
def DiagAttack(df, name):
    ax = df.attack.value_counts()
    ax.plot(kind='pie',labels=None,subplots=True,figsize=(5,8))
```

```
    plt.legend(loc='center␣
↪left',labels=['Normal','DOS','Probe','R2L','U2R'],bbox_to_anchor=(1, 0.5))
    plt.figtext(.5,.8,'Attack pour '+str(name)+' dataframe',fontsize=10,␣
↪ha='center')
```

[134]:
```
DiagAttack(df_train_new,'training')
```

[135]:
```
DiagAttack(df_test_new,'testing')
```


Attack pour testing dataframe
attack
Normal
DOS
Probe
R2L
U2R

[ ]: