SNS_ASS5

April 8, 2021

[1]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from numpy import e, log
import seaborn as sns
from itertools import combinations
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import OrdinalEncoder
from sklearn.linear_model import LinearRegression,LogisticRegression
from sklearn.metrics import classification_report
from sklearn.metrics import precision_score,recall_score,f1_score,accuracy_score

from sklearn import svm


# import warnings
import warnings
# filter warnings
warnings.filterwarnings('ignore')
```

[2]:
```
from google.colab import drive
drive.mount("/content/drive/")
```

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

[3]:
```
df = pd.read_csv("drive/My Drive/full.csv")

print("Number of data points:",df.shape[0])
```

```
Number of data points: 4898430
```

[4]:
```
df.head()
```

[4]:

	0	tcp	http	SF	215	...	0.00.10	0.00.11	0.00.12	0.00.13	normal.
0	0	tcp	http	SF	162	...	0.0	0.0	0.0	0.0	normal.
1	0	tcp	http	SF	236	...	0.0	0.0	0.0	0.0	normal.
2	0	tcp	http	SF	233	...	0.0	0.0	0.0	0.0	normal.
3	0	tcp	http	SF	239	...	0.0	0.0	0.0	0.0	normal.
4	0	tcp	http	SF	238	...	0.0	0.0	0.0	0.0	normal.

[5 rows x 42 columns]

[5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4898430 entries, 0 to 4898429
Data columns (total 42 columns):
 #   Column   Dtype
---  ------   -----
 0   0        int64
 1   tcp      object
 2   http     object
 3   SF       object
 4   215      int64
 5   45076    int64
 6   0.1      int64
 7   0.2      int64
 8   0.3      int64
 9   0.4      int64
 10  0.5      int64
 11  1        int64
 12  0.6      int64
 13  0.7      int64
 14  0.8      int64
 15  0.9      int64
 16  0.10     int64
 17  0.11     int64
 18  0.12     int64
 19  0.13     int64
 20  0.14     int64
 21  0.15     int64
 22  1.1      int64
 23  1.2      int64
 24  0.00     float64
 25  0.00.1   float64
 26  0.00.2   float64
 27  0.00.3   float64
 28  1.00     float64
 29  0.00.4   float64
 30  0.00.5   float64
 31  0.16     int64
```

```
 32  0.17     int64
 33  0.00.6   float64
 34  0.00.7   float64
 35  0.00.8   float64
 36  0.00.9   float64
 37  0.00.10  float64
 38  0.00.11  float64
 39  0.00.12  float64
 40  0.00.13  float64
 41  normal.  object
dtypes: float64(15), int64(23), object(4)
memory usage: 1.5+ GB
```

0.0.1 Object data types:

It is clear from the above observation that there are four columns whose data types are object type i.e, categorical features and we need to encode them to numerical type. So, we will use some library to encode them to numerical.

[6]:
```
obj_df = df.select_dtypes(include=['object']).copy()
obj_df.head()
```

[6]:
```
   tcp  http  SF  normal.
0  tcp  http  SF  normal.
1  tcp  http  SF  normal.
2  tcp  http  SF  normal.
3  tcp  http  SF  normal.
4  tcp  http  SF  normal.
```

[7]:
```
obj_df[obj_df.isnull().any(axis=1)]
```

[7]:
```
Empty DataFrame
Columns: [tcp, http, SF, normal.]
Index: []
```

[8]:
```
obj_df["tcp"].value_counts()
```

[8]:
```
icmp    2833545
tcp     1870597
udp      194288
Name: tcp, dtype: int64
```

[9]:
```
obj_df["http"].value_counts()
```

[9]:
```
ecr_i        2811660
private      1100831
http          623090
smtp           96554
other          72653
               ...
tftp_u             3
aol                2
```

```
harvest            2
http_8001          2
http_2784          1
Name: http, Length: 70, dtype: int64
```

[10]:
```
obj_df["SF"].value_counts()
```

[10]:
```
SF        3744327
S0         869829
REJ        268874
RSTR         8094
RSTO         5344
SH           1040
S1            532
S2            161
RSTOS0        122
OTH            57
S3             50
Name: SF, dtype: int64
```

[11]:
```
obj_df["normal."].value_counts()
```

[11]:
```
smurf.             2807886
neptune.           1072017
normal.             972780
satan.               15892
ipsweep.             12481
portsweep.           10413
nmap.                 2316
back.                 2203
warezclient.          1020
teardrop.              979
pod.                   264
guess_passwd.           53
buffer_overflow.        30
land.                   21
warezmaster.            20
imap.                   12
rootkit.                10
loadmodule.              9
ftp_write.               8
multihop.                7
phf.                     4
perl.                    3
spy.                     2
Name: normal., dtype: int64
```

[12]:
```
ord_enc = OrdinalEncoder()
df["make_tcp"] = ord_enc.fit_transform(df[["tcp"]])
```

```
df[["tcp", "make_tcp"]].head(11)
```

[12]:

	tcp	make_tcp
0	tcp	1.0
1	tcp	1.0
2	tcp	1.0
3	tcp	1.0
4	tcp	1.0
5	tcp	1.0
6	tcp	1.0
7	tcp	1.0
8	tcp	1.0
9	tcp	1.0
10	tcp	1.0

[13]:
```
ord_enc = OrdinalEncoder()
df["make_http"] = ord_enc.fit_transform(df[["http"]])
df[["http", "make_http"]].head(11)
```

[13]:

	http	make_http
0	http	24.0
1	http	24.0
2	http	24.0
3	http	24.0
4	http	24.0
5	http	24.0
6	http	24.0
7	http	24.0
8	http	24.0
9	http	24.0
10	http	24.0

[14]:
```
ord_enc = OrdinalEncoder()
df["make_SF"] = ord_enc.fit_transform(df[["SF"]])
df[["SF", "make_SF"]].head(11)
```

[14]:

	SF	make_SF
0	SF	9.0
1	SF	9.0
2	SF	9.0
3	SF	9.0
4	SF	9.0
5	SF	9.0
6	SF	9.0
7	SF	9.0
8	SF	9.0
9	SF	9.0
10	SF	9.0

[15]:
```
#ord_enc = OrdinalEncoder()
#df["make_normal"] = ord_enc.fit_transform(df[["normal."]])

#df[["normal.", "make_normal"]].head(11)
```

[16]:
```
df.head(10)
```

[16]:

	0	tcp	http	SF	215	...	0.00.13	normal.	make_tcp	make_http	make_SF
0	0	tcp	http	SF	162	...	0.0	normal.	1.0	24.0	9.0
1	0	tcp	http	SF	236	...	0.0	normal.	1.0	24.0	9.0
2	0	tcp	http	SF	233	...	0.0	normal.	1.0	24.0	9.0
3	0	tcp	http	SF	239	...	0.0	normal.	1.0	24.0	9.0
4	0	tcp	http	SF	238	...	0.0	normal.	1.0	24.0	9.0
5	0	tcp	http	SF	235	...	0.0	normal.	1.0	24.0	9.0
6	0	tcp	http	SF	234	...	0.0	normal.	1.0	24.0	9.0
7	0	tcp	http	SF	239	...	0.0	normal.	1.0	24.0	9.0
8	0	tcp	http	SF	181	...	0.0	normal.	1.0	24.0	9.0
9	0	tcp	http	SF	184	...	0.0	normal.	1.0	24.0	9.0

[10 rows x 45 columns]

[17]:
```
target=df['normal.'].values
df.drop(['tcp', 'http','SF','normal.'], axis=1, inplace=True)

df.head(10)
```

[17]:

	0	215	45076	0.1	0.2	...	0.00.12	0.00.13	make_tcp	make_http	make_SF
0	0	162	4528	0	0	...	0.0	0.0	1.0	24.0	9.0
1	0	236	1228	0	0	...	0.0	0.0	1.0	24.0	9.0
2	0	233	2032	0	0	...	0.0	0.0	1.0	24.0	9.0
3	0	239	486	0	0	...	0.0	0.0	1.0	24.0	9.0
4	0	238	1282	0	0	...	0.0	0.0	1.0	24.0	9.0
5	0	235	1337	0	0	...	0.0	0.0	1.0	24.0	9.0
6	0	234	1364	0	0	...	0.0	0.0	1.0	24.0	9.0
7	0	239	1295	0	0	...	0.0	0.0	1.0	24.0	9.0
8	0	181	5450	0	0	...	0.0	0.0	1.0	24.0	9.0
9	0	184	124	0	0	...	0.0	0.0	1.0	24.0	9.0

[10 rows x 41 columns]

[18]:
```
target
```

[18]:
```
array(['normal.', 'normal.', 'normal.', ..., 'normal.', 'normal.',
       'normal.'], dtype=object)
```

[19]:
```
Y=target

X=df.values

print(X[0])
```

```
[0.000e+00 1.620e+02 4.528e+03 0.000e+00 0.000e+00 0.000e+00 0.000e+00
 0.000e+00 1.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 2.000e+00 2.000e+00
 0.000e+00 0.000e+00 0.000e+00 0.000e+00 1.000e+00 0.000e+00 0.000e+00
 1.000e+00 1.000e+00 1.000e+00 0.000e+00 1.000e+00 0.000e+00 0.000e+00
 0.000e+00 0.000e+00 0.000e+00 1.000e+00 2.400e+01 9.000e+00]
```

[20]:
```
Y[:10]
```

[20]:
```
array(['normal.', 'normal.', 'normal.', 'normal.', 'normal.', 'normal.',
       'normal.', 'normal.', 'normal.', 'normal.'], dtype=object)
```

Limiting Data points:

[ ]:
```
#X=X[:1500000]
#Y=Y[:1500000]
```

[ ]:
```
from sklearn.decomposition import PCA
```

[ ]:
```
#pca = PCA(n_components=10)
#X=pca.fit_transform(X)
```

[21]:
```
X_train, X_val, y_train, y_val = train_test_split(X, Y, test_size=0.33,␣
↪random_state=42)
```

[22]:
```
print(X_train.shape)
print(y_train.shape)

print(X_val.shape)
print(y_val.shape)
```

```
(3281948, 41)
(3281948,)
(1616482, 41)
(1616482,)
```

0.1 Testing related data

[23]:
```
df2 = pd.read_csv("drive/My Drive/test.csv",header=None)

print("Number of data points:",df2.shape[0])
```

```
Number of data points: 311029
```

[24]:
```
df2.head(10)
```

[24]:

	0	1	2	3	4	...	37	38	39	40	41
0	NaN	0	udp	private	SF	...	0.00.7	0.00.8	0.00.9	0.00.10	0.00.11
1	0.0	0	udp	private	SF	...	0.0	0.0	0.0	0.0	0.0
2	1.0	0	udp	private	SF	...	0.0	0.0	0.0	0.0	0.0

```
3  2.0   0  udp   private  SF  ...     0.0     0.0     0.0      0.0      0.0
4  3.0   0  udp   private  SF  ...     0.0     0.0     0.0      0.0      0.0
5  4.0   0  udp   private  SF  ...     0.0     0.0     0.0      0.0      0.0
6  5.0   0  udp  domain_u  SF  ...     0.0     0.0     0.0      0.0      0.0
7  6.0   0  udp   private  SF  ...     0.0     0.0     0.0      0.0      0.0
8  7.0   0  udp   private  SF  ...     0.0     0.0     0.0      0.0      0.0
9  8.0   0  tcp      http  SF  ...    0.01     0.0     0.0      0.0      0.0

[10 rows x 42 columns]
```

[25]:
```
### Dropping the first column as it is an index

df2.drop(df2.columns[[0]], axis=1, inplace=True)
```

[26]:
```
df2.head(10)
```

[26]:
```
   1    2         3   4    5   ...      37      38      39       40       41
0  0  udp   private  SF  105  ...  0.00.7  0.00.8  0.00.9  0.00.10  0.00.11
1  0  udp   private  SF  105  ...     0.0     0.0     0.0      0.0      0.0
2  0  udp   private  SF  105  ...     0.0     0.0     0.0      0.0      0.0
3  0  udp   private  SF  105  ...     0.0     0.0     0.0      0.0      0.0
4  0  udp   private  SF  105  ...     0.0     0.0     0.0      0.0      0.0
5  0  udp   private  SF  105  ...     0.0     0.0     0.0      0.0      0.0
6  0  udp  domain_u  SF   29  ...     0.0     0.0     0.0      0.0      0.0
7  0  udp   private  SF  105  ...     0.0     0.0     0.0      0.0      0.0
8  0  udp   private  SF  105  ...     0.0     0.0     0.0      0.0      0.0
9  0  tcp      http  SF  223  ...    0.01     0.0     0.0      0.0      0.0

[10 rows x 41 columns]
```

[27]:
```
ord_enc = OrdinalEncoder()
df2["make_tcp"] = ord_enc.fit_transform(df2[[2]])
df2[[2, "make_tcp"]].head(11)
```

[27]:
```
      2  make_tcp
0   udp       2.0
1   udp       2.0
2   udp       2.0
3   udp       2.0
4   udp       2.0
5   udp       2.0
6   udp       2.0
7   udp       2.0
8   udp       2.0
9   tcp       1.0
10  udp       2.0
```

[28]:
```
ord_enc = OrdinalEncoder()
df2["make_http"] = ord_enc.fit_transform(df2[[3]])
df2[[3, "make_http"]].head(11)
```

[28]:
```
            3  make_http
0    private       46.0
1    private       46.0
2    private       46.0
3    private       46.0
4    private       46.0
5    private       46.0
6   domain_u       11.0
7    private       46.0
8    private       46.0
9       http       22.0
10   private       46.0
```

[29]:
```
ord_enc = OrdinalEncoder()
df2["make_SF"] = ord_enc.fit_transform(df2[[4]])
df2[[4, "make_SF"]].head(11)
```

[29]:
```
     4  make_SF
0   SF      9.0
1   SF      9.0
2   SF      9.0
3   SF      9.0
4   SF      9.0
5   SF      9.0
6   SF      9.0
7   SF      9.0
8   SF      9.0
9   SF      9.0
10  SF      9.0
```

[30]:
```
df2.drop([2, 3,4], axis=1, inplace=True)
df2.head(10)
```

[30]:
```
   1    5    6    7    8  ...        40        41  make_tcp  make_http  make_SF
0  0  105  146  0.1  0.2  ...  0.00.10  0.00.11       2.0       46.0      9.0
1  0  105  146  0.0  0.0  ...       0.0       0.0       2.0       46.0      9.0
2  0  105  146  0.0  0.0  ...       0.0       0.0       2.0       46.0      9.0
3  0  105  146  0.0  0.0  ...       0.0       0.0       2.0       46.0      9.0
4  0  105  146  0.0  0.0  ...       0.0       0.0       2.0       46.0      9.0
5  0  105  146  0.0  0.0  ...       0.0       0.0       2.0       46.0      9.0
6  0   29    0  0.0  0.0  ...       0.0       0.0       2.0       11.0      9.0
7  0  105  146  0.0  0.0  ...       0.0       0.0       2.0       46.0      9.0
8  0  105  146  0.0  0.0  ...       0.0       0.0       2.0       46.0      9.0
9  0  223  185  0.0  0.0  ...       0.0       0.0       1.0       22.0      9.0

[10 rows x 41 columns]
```

[31]:
```
df2 = df2.iloc[0:]
l=[]
for i in df2.iloc[0]:
```

```
  if len(str(i))>3:
    l.append(str(i)[0:4])
  else:
    l.append(i)
df2.iloc[0]=np.array(l)
```

[32]:
```
df2.head(10)
```

[32]:

	1	5	6	7	8	9	...	39	40	41	make_tcp	make_http	make_SF
0	0	105	146	0.1	0.2	0.3	...	0.00	0.00	0.00	2.0	46.0	9.0
1	0	105	146	0	0	0	...	0.0	0.0	0.0	2	46	9
2	0	105	146	0	0	0	...	0.0	0.0	0.0	2	46	9
3	0	105	146	0	0	0	...	0.0	0.0	0.0	2	46	9
4	0	105	146	0	0	0	...	0.0	0.0	0.0	2	46	9
5	0	105	146	0	0	0	...	0.0	0.0	0.0	2	46	9
6	0	29	0	0	0	0	...	0.0	0.0	0.0	2	11	9
7	0	105	146	0	0	0	...	0.0	0.0	0.0	2	46	9
8	0	105	146	0	0	0	...	0.0	0.0	0.0	2	46	9
9	0	223	185	0	0	0	...	0.0	0.0	0.0	1	22	9

[10 rows x 41 columns]

[33]:
```
X_test=df2.values
print(len(X_test))
```

311029

[ ]:
```
#pca = PCA(n_components=10)
#X_test=pca.fit_transform(X_test)
```

0.2 Applying Logistic regression Model

[ ]:
```
clf = LogisticRegression(penalty='l2',random_state=0,multi_class='auto').
↪fit(X_train, y_train)
```

[ ]:
```
y_predval = clf.predict(X_val)
```

[ ]:
```
precision_score(y_val, y_predval, average='weighted')
```

[ ]: 0.9616212081493103

[ ]:
```
recall_score(y_val, y_predval, average='weighted')
```

[ ]: 0.9678134368338157

[ ]:
```
f1_score(y_val, y_predval, average='weighted')
```

[ ]: 0.9631995133442403

[ ]:
```
from sklearn.metrics import accuracy_score

accuracy_score(y_val, y_predval)
```

[ ]:
```
0.9678134368338157
```

[ ]:
```
y_pred = clf.predict(X_test)
```

[ ]:
```
prediction = pd.DataFrame(y_pred).to_csv('testLabelLog.csv',header=["target"])
```

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

[ ]:
```
y_pred
```

[ ]:
```
array(['normal.', 'normal.', 'normal.', ..., 'normal.', 'normal.',
       'normal.'], dtype=object)
```

0.3 Applying the SVM Model

[ ]:
```
ylin_clf = svm.LinearSVC()
lin_clf.fit(X_train, y_train)
```

[ ]:
```
LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
```

[ ]:
```
y_predval = lin_clf.predict(X_val)
```

[ ]:
```
precision_score(y_val, y_predval, average='weighted')
```

[ ]:
```
0.9974327233474799
```

[ ]:
```
recall_score(y_val, y_predval, average='weighted')
```

[ ]:
```
0.9939386890791236
```

[ ]:
```
f1_score(y_val, y_predval, average='weighted')
```

[ ]:
```
0.9956167275161313
```

[ ]:
```
from sklearn.metrics import accuracy_score

accuracy_score(y_val, y_predval)
```

[ ]:
```
0.9939386890791236
```

[ ]:
```
y_pred = lin_clf.predict(X_test)
```

[ ]:
```
prediction = pd.DataFrame(y_pred).to_csv('testLabelSVM.csv',header=['target'])
```

[ ]:
```
y_pred
```

[ ]:
```
array(['normal.', 'normal.', 'normal.', ..., 'normal.', 'normal.',
       'normal.'], dtype=object)
```

[ ]:
```
# Play an audio beep. Any audio URL will do.
from google.colab import output
```

```
output.eval_js('new Audio("https://upload.wikimedia.org/wikipedia/commons/0/05/
↪Beep-09.ogg").play()')
```

0.4 Applying NaiveBayes

[ ]:
```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB().fit(X_train, y_train)
```

[ ]:
```
y_predval = gnb.predict(X_val)
```

[ ]:
```
precision_score(y_val, y_predval, average='weighted')
```

[ ]:
```
0.9904090357675394
```

[ ]:
```
recall_score(y_val, y_predval, average='weighted')
```

[ ]:
```
0.9484435954127544
```

[ ]:
```
f1_score(y_val, y_predval, average='weighted')
```

[ ]:
```
0.9658755438779745
```

[ ]:
```
from sklearn.metrics import accuracy_score

accuracy_score(y_val, y_predval)
```

[ ]:
```
0.9484435954127544
```

[ ]:
```
y_pred = gnb.predict(X_test)
```

[ ]:
```
prediction = pd.DataFrame(y_pred).to_csv('testLabelGNB.csv',header=["target"])
```

[ ]:
```
y_pred
```

[ ]:
```
array(['normal.', 'normal.', 'normal.', ..., 'normal.', 'normal.',
       'normal.'], dtype='<U16')
```

[ ]:
```
# Play an audio beep. Any audio URL will do.
from google.colab import output
output.eval_js('new Audio("https://upload.wikimedia.org/wikipedia/commons/0/05/
↪Beep-09.ogg").play()')
```

0.5 Applying Random forests:

[34]:
```
from sklearn.ensemble import RandomForestClassifier
print(len(X))
```

```
4898430
```

[35]:
```
clf = RandomForestClassifier(n_estimators=500,max_depth=15, random_state=0)
clf.fit(X_train,y_train)
```

[35]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=15, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=500,
                       n_jobs=None, oob_score=False, random_state=0, verbose=0,
                       warm_start=False)
```

[37]:
```
y_predval = clf.predict(X_val)
```

[38]:
```
precision_score(y_val, y_predval, average='weighted')
```

[38]:
```
0.9999359544600412
```

[39]:
```
recall_score(y_val, y_predval, average='weighted')
```

[39]:
```
0.9999387558908791
```

[40]:
```
f1_score(y_val, y_predval, average='weighted')
```

[40]:
```
0.999936184951831
```

[41]:
```
from sklearn.metrics import accuracy_score

accuracy_score(y_val, y_predval)
```

[41]:
```
0.9999387558908791
```

0.5.1 Random forest produced the best results so far.So training with it to produce final test-Label.csv

[ ]:
```
clf = RandomForestClassifier(n_estimators=500,max_depth=15, random_state=0)
clf.fit(X,Y)
```

[ ]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=15, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=500,
                       n_jobs=None, oob_score=False, random_state=0, verbose=0,
                       warm_start=False)
```

[43]:
```
y_pred = clf.predict(X_test)
print(len(y_pred))
print(len(X_test))
```

```
311029
311029
```

[45]:
```
prediction = pd.DataFrame(y_pred).
 ↪to_csv('submission_12_2020201012_2020201087_2020201095_2020202020.
 ↪csv',header=['target'])
```

[44]:
```
y_pred
```

[44]:
```
array(['normal.', 'normal.', 'normal.', ..., 'normal.', 'normal.',
       'normal.'], dtype=object)
```

[42]:
```
# Play an audio beep. Any audio URL will do.
from google.colab import output
output.eval_js('new Audio("https://upload.wikimedia.org/wikipedia/commons/0/05/
 ↪Beep-09.ogg").play()')
```

1 Applying Decision Trees

[ ]:
```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(random_state=0)
clf.fit(X_train,y_train)
```

[ ]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=0, splitter='best')
```

[ ]:
```
y_predval = clf.predict(X_val)
```

[ ]:
```
precision_score(y_val, y_predval, average='weighted')
```

[ ]:
```
0.9999953298485896
```

[ ]:
```
recall_score(y_val, y_predval, average='weighted')
```

[ ]:
```
0.9999950509810811
```

[ ]:
```
f1_score(y_val, y_predval, average='weighted')
```

[ ]:
```
0.9999951262046003
```

[ ]:
```
from sklearn.metrics import accuracy_score

accuracy_score(y_val, y_predval)
```

[ ]:
```
0.9999950509810811
```

[ ]:
```
y_pred = clf.predict(X_test)
```

[ ]:
```
prediction = pd.DataFrame(y_pred).
 ↪to_csv('submission_12_2020201012_2020201087_2020201095_2020202020_1.
 ↪csv',header=['target'])
```

[ ]:
```
y_pred
```

[ ]:
```
array(['normal.', 'normal.', 'normal.', ..., 'normal.', 'normal.',
       'normal.'], dtype=object)
```

[47]:
```
%%capture
!wget -nc https://raw.githubusercontent.com/brpy/colab-pdf/master/colab_pdf.py
from colab_pdf import colab_pdf
colab_pdf('SNS_ASS5.ipynb')
```

[ ]:
```

```