Ensemble Bagging Models

Import

In [1]:
```
import pandas as pd
from pycaret.classification import *
import time
from DB_scripts import rnd_bln_split_CSV as shf
```

Settings

In [2]:
```
# set constants
target_label = 'tuple'
learning_model = ['rf','et','lightgbm','xgboost']
num_features = ['min_packet_size', 'min_fpkt', 'min_bpkt']
file_name = "all_features_"
path = target_label + "_dataset/"
ensemble_method='Bagging'
```

In [3]:
```
# function for making model-prediction over the data set and measure the run t
ime
def timed_prediction(in_data,in_model):
    t = time.process_time()
    predicted = predict_model(in_model, data=in_data)
    elapsed_time = time.process_time() - t
    print("prediction took: " + str(elapsed_time))
    return predicted
```

In [4]:
```
# function for checkign the correction of the model-prediction over the data
def check_correction(in_predicted):
    count=0
    index = in_predicted.index
    number_of_rows = len(index)
    for i in range(0,number_of_rows):
        if str(int(in_predicted.iloc[i][target_label])) != str(int(in_predicte
d.iloc[i]['Label'])):
            #print("prediction not matched in line " + str(i) + " as " + str(i
n_predicted.iloc[i]['app']) + "!=" + str(in_predicted.iloc[i]['Label']))
            count=count+1
    print("number of error: " + str(count) + " from " + str(number_of_rows) +
" test samples \n which is " + str(count/number_of_rows) + " precent of erro
r.")
```

In [5]:
```
# compare answers and labeled test
def compare_prediction_with_answers(in_predicted, in_answers):
    count=0
    index = in_predicted.index
    number_of_rows = len(index)
    for i in range(0,number_of_rows):
        if str(in_answers.iloc[i]) != str(int(in_predicted.iloc[i]['Label'])):
count=count+1
        # print the unmatched answers
        #print("answer os and test label are not matched in line " + str(i) +
 " as " + str(answers.iloc[i]['os']) + "!=" + str(predict_test.iloc[i]['Labe
l']))
    print("number of error: " + str(count) + " from " + str(number_of_rows) +
" test samples \n which is " + str(count/number_of_rows) + " precent of erro
r.")
```

In [6]:
```
# activating balanced random data shuffling
shf.split_CSV_randomly_balanced(target_label,file_name)
```

Read Data

In [7]:
```
data = pd.read_csv(path+file_name+target_label+'_train.csv',
                        sep='\t',
                        skiprows=[1])
```

In [8]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14442 entries, 0 to 14441
Data columns (total 70 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   fSSL_session_id_len             14442 non-null  float64
 1   fSSL_num_extensions             14442 non-null  float64
 2   fSSL_num_compression_methods    14442 non-null  float64
 3   SYN_tcp_scale                   14442 non-null  float64
 4   SYN_MSS                         14442 non-null  float64
 5   SYN_tcp_winsize                 14442 non-null  float64
 6   fcipher_suites_1                14442 non-null  float64
 7   fcipher_suites_2                14442 non-null  float64
 8   fcipher_suites_3                14442 non-null  float64
 9   fSSLv_1                         14442 non-null  float64
 10  fSSLv_2                         14442 non-null  float64
 11  fSSLv_3                         14442 non-null  float64
 12  fSSLv_4                         14442 non-null  float64
 13  size_histogram_1                14442 non-null  float64
 14  size_histogram_2                14442 non-null  float64
 15  size_histogram_3                14442 non-null  float64
 16  size_histogram_4                14442 non-null  float64
 17  size_histogram_5                14442 non-null  float64
 18  size_histogram_6                14442 non-null  float64
 19  size_histogram_7                14442 non-null  float64
 20  size_histogram_8                14442 non-null  float64
 21  size_histogram_9                14442 non-null  float64
 22  size_histogram_10               14442 non-null  float64
 23  fpeak_features_1                14442 non-null  float64
 24  fpeak_features_2                14442 non-null  float64
 25  fpeak_features_3                14442 non-null  float64
 26  fpeak_features_4                14442 non-null  float64
 27  fpeak_features_5                14442 non-null  float64
 28  fpeak_features_6                14442 non-null  float64
 29  fpeak_features_7                14442 non-null  float64
 30  fpeak_features_8                14442 non-null  float64
 31  fpeak_features_9                14442 non-null  float64
 32  bpeak_features_1                14442 non-null  float64
 33  bpeak_features_2                14442 non-null  float64
 34  bpeak_features_3                14442 non-null  float64
 35  bpeak_features_4                14442 non-null  float64
 36  bpeak_features_5                14442 non-null  float64
 37  bpeak_features_6                14442 non-null  float64
 38  bpeak_features_7                14442 non-null  float64
 39  bpeak_features_8                14442 non-null  float64
 40  bpeak_features_9                14442 non-null  float64
 41  packet_count                    14442 non-null  float64
 42  min_packet_size                 14442 non-null  float64
 43  max_packet_size                 14442 non-null  float64
 44  mean_packet_size                14442 non-null  float64
 45  sizevar                         14442 non-null  float64
 46  std_fiat                        14442 non-null  float64
 47  fpackets                        14442 non-null  float64
 48  bpackets                        14442 non-null  float64
 49  fbytes                          14442 non-null  float64
 50  bbytes                          14442 non-null  float64
 51  min_fiat                        14442 non-null  float64
```

```
 52  min_biat                    14442 non-null  float64
 53  max_fiat                    14442 non-null  float64
 54  max_biat                    14442 non-null  float64
 55  std_biat                    14442 non-null  float64
 56  mean_fiat                   14442 non-null  float64
 57  mean_biat                   14442 non-null  float64
 58  min_fpkt                    14442 non-null  float64
 59  min_bpkt                    14442 non-null  float64
 60  max_fpkt                    14442 non-null  float64
 61  max_bpkt                    14442 non-null  float64
 62  std_fpkt                    14442 non-null  float64
 63  std_bpkt                    14442 non-null  float64
 64  mean_fpkt                   14442 non-null  float64
 65  mean_bpkt                   14442 non-null  float64
 66  mean_fttl_1                 14442 non-null  float64
 67  mean_fttl_2                 14442 non-null  float64
 68  num_keep_alive              14442 non-null  float64
 69  tuple                       14442 non-null  int64
dtypes: float64(69), int64(1)
memory usage: 7.7 MB
```

In [9]:
```
data[target_label].unique()
```

Out[9]:
```
array([16102, 14602, 15602, 18102, 13102, 11602, 16201, 13201, 18201,
       16101, 13101, 18101, 16202, 18202, 13202, 16302, 18302, 13302,
       13403, 18403, 17403, 14601, 17101, 12101, 17201, 12201, 16403,
       16103, 13103, 18103], dtype=int64)
```

Read Unseen Test

In [10]:
```
unseen_data = pd.read_csv(path+file_name+target_label+'_test.csv',
                          sep='\t',
                          skiprows=[1])
```

In [11]:
```
# check for target column values (we got all wanted values...)
unseen_data[target_label].unique()
```

Out[11]:
```
array([16102, 14602, 15602, 18102, 13102, 11602, 16201, 13201, 18201,
       16101, 13101, 18101, 16202, 18202, 13202, 16302, 18302, 13302,
       13403, 18403, 17403, 14601, 17101, 12101, 17201, 12201, 16403,
       16103, 13103, 18103], dtype=int64)
```

In [12]:
```
# saving the target column
answers = unseen_data[target_label]
```

In [13]:
```
# dropping traget column from test.
unseen_data = unseen_data.drop(columns=[target_label])
```

In [14]:
```
unseen_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6189 entries, 0 to 6188
Data columns (total 69 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   fSSL_session_id_len             6189 non-null   float64
 1   fSSL_num_extensions             6189 non-null   float64
 2   fSSL_num_compression_methods    6189 non-null   float64
 3   SYN_tcp_scale                   6189 non-null   float64
 4   SYN_MSS                         6189 non-null   float64
 5   SYN_tcp_winsize                 6189 non-null   float64
 6   fcipher_suites_1                6189 non-null   float64
 7   fcipher_suites_2                6189 non-null   float64
 8   fcipher_suites_3                6189 non-null   float64
 9   fSSLv_1                         6189 non-null   float64
 10  fSSLv_2                         6189 non-null   float64
 11  fSSLv_3                         6189 non-null   float64
 12  fSSLv_4                         6189 non-null   float64
 13  size_histogram_1                6189 non-null   float64
 14  size_histogram_2                6189 non-null   float64
 15  size_histogram_3                6189 non-null   float64
 16  size_histogram_4                6189 non-null   float64
 17  size_histogram_5                6189 non-null   float64
 18  size_histogram_6                6189 non-null   float64
 19  size_histogram_7                6189 non-null   float64
 20  size_histogram_8                6189 non-null   float64
 21  size_histogram_9                6189 non-null   float64
 22  size_histogram_10               6189 non-null   float64
 23  fpeak_features_1                6189 non-null   float64
 24  fpeak_features_2                6189 non-null   float64
 25  fpeak_features_3                6189 non-null   float64
 26  fpeak_features_4                6189 non-null   float64
 27  fpeak_features_5                6189 non-null   float64
 28  fpeak_features_6                6189 non-null   float64
 29  fpeak_features_7                6189 non-null   float64
 30  fpeak_features_8                6189 non-null   float64
 31  fpeak_features_9                6189 non-null   float64
 32  bpeak_features_1                6189 non-null   float64
 33  bpeak_features_2                6189 non-null   float64
 34  bpeak_features_3                6189 non-null   float64
 35  bpeak_features_4                6189 non-null   float64
 36  bpeak_features_5                6189 non-null   float64
 37  bpeak_features_6                6189 non-null   float64
 38  bpeak_features_7                6189 non-null   float64
 39  bpeak_features_8                6189 non-null   float64
 40  bpeak_features_9                6189 non-null   float64
 41  packet_count                    6189 non-null   float64
 42  min_packet_size                 6189 non-null   float64
 43  max_packet_size                 6189 non-null   float64
 44  mean_packet_size                6189 non-null   float64
 45  sizevar                         6189 non-null   float64
 46  std_fiat                        6189 non-null   float64
 47  fpackets                        6189 non-null   float64
 48  bpackets                        6189 non-null   float64
 49  fbytes                          6189 non-null   float64
 50  bbytes                          6189 non-null   float64
 51  min_fiat                        6189 non-null   float64
```

```
 52  min_biat                6189 non-null   float64
 53  max_fiat                6189 non-null   float64
 54  max_biat                6189 non-null   float64
 55  std_biat                6189 non-null   float64
 56  mean_fiat               6189 non-null   float64
 57  mean_biat               6189 non-null   float64
 58  min_fpkt                6189 non-null   float64
 59  min_bpkt                6189 non-null   float64
 60  max_fpkt                6189 non-null   float64
 61  max_bpkt                6189 non-null   float64
 62  std_fpkt                6189 non-null   float64
 63  std_bpkt                6189 non-null   float64
 64  mean_fpkt               6189 non-null   float64
 65  mean_bpkt               6189 non-null   float64
 66  mean_fttl_1             6189 non-null   float64
 67  mean_fttl_2             6189 non-null   float64
 68  num_keep_alive          6189 non-null   float64
dtypes: float64(69)
memory usage: 3.3 MB
```

Setup Classifier

which splits the data into train and test for model building

In [ ]:
```
setup(data=data,
      target=target_label,
      numeric_features=num_features,
      silent=True)
# train_size=0.99
```

Compare Models

In [17]:
```
compare_models(whitelist=learning_model)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
0	Extreme Gradient Boosting	0.9760	0.0000	0.8723	0.9747	0.9745	0.9719	0.9719	13.1820
1	Extra Trees Classifier	0.9747	0.0000	0.8929	0.9743	0.9735	0.9704	0.9704	7.1094
2	Random Forest Classifier	0.9679	0.0000	0.8419	0.9665	0.9656	0.9623	0.9624	4.3846
3	Light Gradient Boosting Machine	0.9385	0.0000	0.8171	0.9738	0.9530	0.9291	0.9311	6.5520

Out[17]:
```
OneVsRestClassifier(estimator=XGBClassifier(base_score=None, booster=None,
                                            colsample_bylevel=None,
                                            colsample_bynode=None,
                                            colsample_bytree=None, gamma=Non
e,
                                            gpu_id=None, importance_type='gai
n',
                                            interaction_constraints=None,
                                            learning_rate=None,
                                            max_delta_step=None, max_depth=No
ne,
                                            min_child_weight=None, missing=na
n,
                                            monotone_constraints=None,
                                            n_estimators=100, n_jobs=-1,
                                            num_parallel_tree=None,
                                            objective='binary:logistic',
                                            random_state=7694, reg_alpha=Non
e,
                                            reg_lambda=None,
                                            scale_pos_weight=None,
                                            subsample=None, tree_method=None,
                                            validate_parameters=None,
                                            verbosity=0),
                    n_jobs=-1)
```

Creating Learning Model - Bagging RF

In [18]:
```
model = create_model(learning_model[0], ensemble=True, method=ensemble_method)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9782	0.0000	0.8389	0.9752	0.9758	0.9745	0.9746
1	0.9594	0.0000	0.7741	0.9530	0.9545	0.9525	0.9526
2	0.9614	0.0000	0.8142	0.9605	0.9579	0.9548	0.9550
3	0.9654	0.0000	0.8184	0.9643	0.9626	0.9594	0.9595
4	0.9743	0.0000	0.8296	0.9713	0.9720	0.9699	0.9699
5	0.9674	0.0000	0.8547	0.9654	0.9650	0.9618	0.9619
6	0.9753	0.0000	0.7571	0.9729	0.9722	0.9710	0.9711
7	0.9703	0.0000	0.8419	0.9684	0.9679	0.9653	0.9653
8	0.9733	0.0000	0.8359	0.9734	0.9712	0.9687	0.9688
9	0.9614	0.0000	0.7918	0.9581	0.9592	0.9548	0.9548
Mean	0.9686	0.0000	0.8157	0.9663	0.9658	0.9633	0.9633
SD	0.0063	0.0000	0.0302	0.0070	0.0068	0.0074	0.0073

Prediction

In [19]:
```
predicted = timed_prediction(data,model)
```

```
prediction took: 2.984375
```

In [20]:
```
check_correction(predicted)
```

```
number of error: 181 from 14442 test samples
 which is 0.012532890181415316 precent of error.
```

Make Unseen Test

In [21]:
```
predicted = predict_model(model, data=unseen_data)
```

In [22]:
```
predicted['Label'].unique()
```

Out[22]:
```
array([16102, 13102, 14602, 16302, 18102, 15602, 18101, 13202, 11602,
       16201, 18201, 13201, 12201, 16101, 17101, 13101, 12101, 16202,
       18202, 13302, 18302, 13403, 17403, 18403, 16403, 14601, 17201,
       16103, 13103])
```

check prediction correction

In [23]:
```
answers.unique()
```

Out[23]:
```
array([16102, 14602, 15602, 18102, 13102, 11602, 16201, 13201, 18201,
       16101, 13101, 18101, 16202, 18202, 13202, 16302, 18302, 13302,
       13403, 18403, 17403, 14601, 17101, 12101, 17201, 12201, 16403,
       16103, 13103, 18103], dtype=int64)
```

In [24]:
```
compare_prediction_with_answers(predicted,answers)
```

```
number of error: 213 from 6189 test samples
 which is 0.034415899175957346 precent of error.
```

Creating Learning Model - Bagging ET

In [25]:
```
model = create_model(learning_model[1], ensemble=True, method=ensemble_method)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9842	0.0000	0.8921	0.9817	0.9822	0.9815	0.9815
1	0.9683	0.0000	0.8502	0.9663	0.9663	0.9629	0.9630
2	0.9654	0.0000	0.8374	0.9631	0.9624	0.9595	0.9596
3	0.9743	0.0000	0.8746	0.9735	0.9721	0.9699	0.9700
4	0.9832	0.0000	0.9327	0.9840	0.9832	0.9803	0.9803
5	0.9664	0.0000	0.8651	0.9653	0.9650	0.9606	0.9607
6	0.9802	0.0000	0.8502	0.9773	0.9784	0.9768	0.9769
7	0.9713	0.0000	0.8517	0.9708	0.9699	0.9664	0.9665
8	0.9782	0.0000	0.8847	0.9788	0.9766	0.9745	0.9746
9	0.9624	0.0000	0.8459	0.9632	0.9617	0.9560	0.9560
Mean	0.9734	0.0000	0.8685	0.9724	0.9718	0.9688	0.9689
SD	0.0074	0.0000	0.0272	0.0074	0.0076	0.0087	0.0087

Prediction

In [26]:
```
predicted = timed_prediction(data,model)
```

```
prediction took: 18.484375
```

In [27]:
```
check_correction(predicted)
```

```
number of error: 123 from 14442 test samples
 which is 0.008516825924387204 precent of error.
```

Make Unseen Test

In [28]:
```
predicted = predict_model(model, data=unseen_data)
```

In [29]:
```
predicted['Label'].unique()
```

Out[29]:
```
array([16102, 14602, 15602, 18102, 13102, 13202, 16302, 11602, 16201,
       18201, 13201, 13101, 18101, 12201, 16101, 17101, 14601, 12101,
       16202, 18202, 13302, 18302, 13403, 18403, 16103, 16403, 17403,
       17201, 13103])
```

check prediction correction

In [30]:
```
answers.unique()
```

Out[30]:
```
array([16102, 14602, 15602, 18102, 13102, 11602, 16201, 13201, 18201,
       16101, 13101, 18101, 16202, 18202, 13202, 16302, 18302, 13302,
       13403, 18403, 17403, 14601, 17101, 12101, 17201, 12201, 16403,
       16103, 13103, 18103], dtype=int64)
```

In [31]:
```
compare_prediction_with_answers(predicted,answers)
```

```
number of error: 177 from 6189 test samples
 which is 0.028599127484246242 precent of error.
```

Creating Learning Model - Bagging lightgbm

In [32]:
```
model = create_model(learning_model[2], ensemble=True, method=ensemble_method)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9822	0.0000	0.8550	0.9787	0.9800	0.9792	0.9792
1	0.9703	0.0000	0.8296	0.9690	0.9678	0.9653	0.9653
2	0.9693	0.0000	0.8270	0.9701	0.9682	0.9641	0.9642
3	0.9753	0.0000	0.8425	0.9741	0.9731	0.9710	0.9711
4	0.9802	0.0000	0.9213	0.9793	0.9792	0.9769	0.9769
5	0.9683	0.0000	0.8711	0.9664	0.9668	0.9630	0.9630
6	0.9822	0.0000	0.8599	0.9786	0.9801	0.9792	0.9792
7	0.9753	0.0000	0.8570	0.9751	0.9744	0.9711	0.9711
8	0.9723	0.0000	0.8851	0.9745	0.9720	0.9676	0.9676
9	0.9772	0.0000	0.8843	0.9774	0.9766	0.9733	0.9734
Mean	0.9753	0.0000	0.8633	0.9743	0.9738	0.9711	0.9711
SD	0.0049	0.0000	0.0271	0.0042	0.0049	0.0057	0.0057

Prediction

In [33]:
```
predicted = timed_prediction(data,model)
```

```
prediction took: 145.921875
```

In [34]:
```
check_correction(predicted)
```

```
number of error: 125 from 14442 test samples
 which is 0.008655310898767483 precent of error.
```

Make Unseen Test

In [35]:
```
predicted = predict_model(model, data=unseen_data)
```

In [36]:
```
predicted['Label'].unique()
```

Out[36]:
```
array([16102, 14602, 18103, 18102, 18302, 15602, 13102, 13302, 16302,
       11602, 16201, 18201, 12201, 13201, 16101, 13103, 18101, 17101,
       13101, 12101, 16202, 18202, 13202, 13403, 16103, 18403, 17403,
       14601, 17201, 16403])
```

check prediction correction

In [37]:
```
answers.unique()
```

Out[37]:
```
array([16102, 14602, 15602, 18102, 13102, 11602, 16201, 13201, 18201,
       16101, 13101, 18101, 16202, 18202, 13202, 16302, 18302, 13302,
       13403, 18403, 17403, 14601, 17101, 12101, 17201, 12201, 16403,
       16103, 13103, 18103], dtype=int64)
```

In [38]:
```
compare_prediction_with_answers(predicted,answers)
```

```
number of error: 166 from 6189 test samples
 which is 0.026821780578445628 precent of error.
```

Creating Learning Model - Bagging xgboost

In [39]:
```
model = create_model(learning_model[3], ensemble=True, method=ensemble_method)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9812	0.0000	0.8545	0.9777	0.9790	0.9780	0.9780
1	0.9693	0.0000	0.8482	0.9690	0.9679	0.9641	0.9642
2	0.9693	0.0000	0.8441	0.9687	0.9668	0.9641	0.9642
3	0.9763	0.0000	0.8531	0.9733	0.9743	0.9722	0.9722
4	0.9773	0.0000	0.9138	0.9762	0.9762	0.9734	0.9734
5	0.9723	0.0000	0.8755	0.9698	0.9706	0.9676	0.9676
6	0.9773	0.0000	0.8132	0.9740	0.9752	0.9734	0.9734
7	0.9763	0.0000	0.8584	0.9745	0.9742	0.9722	0.9723
8	0.9733	0.0000	0.8378	0.9734	0.9720	0.9687	0.9688
9	0.9713	0.0000	0.8719	0.9694	0.9696	0.9664	0.9664
Mean	0.9744	0.0000	0.8570	0.9726	0.9726	0.9700	0.9700
SD	0.0037	0.0000	0.0252	0.0030	0.0037	0.0043	0.0043

Prediction

In [40]:
```
predicted = timed_prediction(data,model)
```

```
prediction took: 2.390625
```

In [41]:
```
check_correction(predicted)
```

```
number of error: 128 from 14442 test samples
 which is 0.008863038360337904 precent of error.
```

Make Unseen Test

In [42]:
```
predicted = predict_model(model, data=unseen_data)
```

In [43]:
```
predicted['Label'].unique()
```

Out[43]:
```
array([16102, 14602, 16302, 18102, 15602, 13102, 13202, 11602, 16201,
       18201, 13201, 16101, 12201, 18101, 13101, 12101, 16202, 18202,
       13302, 18302, 13403, 17403, 16103, 18403, 16403, 14601, 17101,
       17201, 13103])
```

check prediction correction

In [44]:
```
answers.unique()
```

Out[44]:
```
array([16102, 14602, 15602, 18102, 13102, 11602, 16201, 13201, 18201,
       16101, 13101, 18101, 16202, 18202, 13202, 16302, 18302, 13302,
       13403, 18403, 17403, 14601, 17101, 12101, 17201, 12201, 16403,
       16103, 13103, 18103], dtype=int64)
```

In [45]:
```
compare_prediction_with_answers(predicted,answers)
```

```
number of error: 160 from 6189 test samples
 which is 0.025852318629827113 precent of error.
```

In [ ]: