Minimizing Features Set

Import

importing pandas package for handling data objects,

using pycaret only classification.

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

Functions

In [2]:
```
# set target feature
target_label = 'tuple'
learning_models = ['rf','xgboost']
num_features = ['min_packet_size', 'min_fpkt', 'min_bpkt']
```

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.")
```

Read Data

the data set "../app_dataset/all_features_app.csv" is the main resource and has beed splited to train and test.

In [6]:
```
data = pd.read_csv(target_label+r'_dataset\all_features_'+target_label+'_trai
n.csv',
                      sep='\t',
                      skiprows=[1])
```

previews of the top and the bottom train data set.

Setup and Compare

setting up the train data set for Classification. targeting the prediction value to the 'app' column. https://pycaret.org/classification/ (https://pycaret.org/classification/)

NOTE:this will split the data to train the test by default setting, when the test part will be used in prediction.

In [7]:
```
# clean data setup
setup(data=data,
      target=target_label,
      numeric_features=num_features,
      silent=True)
model = compare_models(whitelist=learning_models)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
0	Extreme Gradient Boosting	0.9751	0.0000	0.8610	0.9733	0.9734	0.9708	0.9709	13.8211
1	Random Forest Classifier	0.9680	0.0000	0.8500	0.9670	0.9661	0.9626	0.9626	4.3359

Prediction

make a prediction procces over the trained data, see validation results.

In [8]:
```
predicted = timed_prediction(data,model)
```

```
prediction took: 28.75
```

In [9]:
```
check_correction(predicted)
```

```
number of error: 111 from 14442 test samples
 which is 0.007685916078105526 precent of error.
```

Tune and Finalize

tune model hyperparameters for better performance and quality and finilazing model for testing over unseen data set.

thos two doing nothing...

In [10]:
```
# tuned = tune_model(model)
```

In [11]:
```
# fin_tun_mod = finalize_model(tuned)
```

In [12]:
```
# evaluate_model(fin_tun_mod)
```

Read Test

Read the unseen test data set, with the basic data information

In [13]:
```
unseen_data = pd.read_csv(target_label+'_dataset/all_features_'+target_label+
'_test.csv',
                      sep='\t',
                      skiprows=[1])
```

In [14]:
```
# saving the target column
answers = unseen_data[target_label]
```

In [15]:
```
# dropping targer column from test.
unseen_data = unseen_data.drop(columns=[target_label])
```

Independent Prediction

make a prediction of the 'app' label of the model on the unseen test data set.

In [16]:
```
predicted = timed_prediction(unseen_data,model)
```

Check Test Correction

read a already prepared answers set of the unseen test data set and make comparison between the answers and the predicted 'app' label

In [17]:
```
compare_prediction_with_answers(predicted,answers)
```

```
number of error: 141 from 6189 test samples
 which is 0.022782355792535142 precent of error.
```

Minimized Dataset

droping low variance features, and grouping binary features groups

with extra python script, can see in ...

Read New Data

the data set "../app_dataset/new_all_features_app.csv" is the main resource and has beed splited to train and test.

In [18]:
```
data = pd.read_csv(target_label+r'_dataset\new_all_features_'+target_label+'_t
rain.csv',
                      sep='\t',
                      skiprows=[1])
```

Setup and Compare

```
In [20]: # clean data setup
         setup(data=data,
               target=target_label,
               numeric_features=num_features,
               silent=True)
         model = compare_models(whitelist=learning_models)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
0	Extreme Gradient Boosting	0.9766	0.0000	0.8755	0.9759	0.9751	0.9726	0.9726	13.1376
1	Random Forest Classifier	0.9691	0.0000	0.8555	0.9673	0.9669	0.9639	0.9639	4.3328

Prediction

make a prediction procces over the trained data, see validation results.

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

```
prediction took: 23.109375
```

```
In [22]: check_correction(predicted)
```

```
number of error: 99 from 14442 test samples
 which is 0.006855006231823847 precent of error.
```

Tune and Finalize

tune model hyperparameters for better performance and quality and finilazing model for testing over unseen data set.

thos two doing nothing...

```
In [23]: # tuned = tune_model(model)
```

```
In [24]: # fin_tun_mod = finalize_model(tuned)
```

```
In [25]: # evaluate_model(fin_tun_mod)
```

Read Test

Read the unseen test data set, with the basic data information

In [26]:
```
unseen_data = pd.read_csv(target_label+'_dataset/new_all_features_'+target_lab
el+'_test.csv',
                          sep='\t',
                          skiprows=[1])
```

In [27]:
```
# saving the target column
answers = unseen_data[target_label]
```

In [28]:
```
# dropping targer column from test.
unseen_data = unseen_data.drop(columns=[target_label])
```

Independent Prediction

make a prediction of the 'app' label of the model on the unseen test data set.

In [31]:
```
predicted = timed_prediction(unseen_data,model)
```

```
prediction took: 11.3125
```

Check Test Correction

read a already prepared answers set of the unseen test data set and make comparison between the answers and the predicted 'app' label

In [30]:
```
compare_prediction_with_answers(predicted,answers)
```

```
number of error: 152 from 6189 test samples
 which is 0.024559702698335757 precent of error.
```