Compare Classification Setups For All Algorithms

Import

In [26]:
```
import pandas as pd
from pycaret.classification import *
import time
```

Read Data

using all of the data to get teh sense of what learning algorithm should we use

In [46]:
```
# set target feature
target_label = 'tuple'
# set learning models
learning_models = ['rf','svm','et']
# set numeric features which pycaret takes as category
num_features = ['min_packet_size', 'min_fpkt', 'min_bpkt']
```

In [15]:
```
data = pd.read_csv(target_label+r'_dataset\new_all_features_'+target_label+'_train.csv',
                        sep='\t',
                        skiprows=[1])
```

Clear Setup And Compare

In [17]:
```
# setup(data = data,
#         target=target_label,
#         silent=True,
#         numeric_features=num_features)
# compare_models()
```

Advance Normlized Setup And Compare

In [ ]:
```
setup(data = data,
        target=target_label,
        silent=True,
        numeric_features=num_features,
        normalize=True,
        transformation=True,
        transformation_method='quantile')
```

In [47]:
```
model = compare_models(whitelist=learning_models, n_select=3)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
0	Extra Trees Classifier	0.9780	0.0000	0.9057	0.9779	0.9771	0.9743	0.9743	7.0925
1	Random Forest Classifier	0.9675	0.0000	0.8397	0.9663	0.9652	0.9619	0.9620	4.3111
2	SVM - Linear Kernel	0.9240	0.0000	0.7665	0.9274	0.9209	0.9110	0.9113	1.1964

read test

In [48]:
```
unseen_data = pd.read_csv(target_label+'_dataset/new_all_features_'+target_lab
el+'_test.csv',
                          sep='\t',
                          skiprows=[1])
```

In [49]:
```
# saving the target column
answers = unseen_data[target_label]
```

In [50]:
```
# dropping 'app' column from test.
unseen_data = unseen_data.drop(columns=[target_label])
```

Blend Models

In [53]:
```
blended_model = blend_models(estimator_list = model, method = 'hard')
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9753	0.0000	0.8778	0.9754	0.9739	0.9711	0.9711
1	0.9713	0.0000	0.8580	0.9695	0.9692	0.9664	0.9664
2	0.9753	0.0000	0.9106	0.9746	0.9741	0.9711	0.9711
3	0.9693	0.0000	0.8387	0.9663	0.9665	0.9641	0.9641
4	0.9733	0.0000	0.8631	0.9740	0.9728	0.9687	0.9688
5	0.9792	0.0000	0.8926	0.9788	0.9781	0.9757	0.9757
6	0.9812	0.0000	0.9508	0.9816	0.9809	0.9780	0.9780
7	0.9644	0.0000	0.7975	0.9626	0.9610	0.9583	0.9584
8	0.9773	0.0000	0.8753	0.9763	0.9757	0.9734	0.9734
9	0.9772	0.0000	0.8881	0.9747	0.9756	0.9733	0.9734
Mean	0.9744	0.0000	0.8752	0.9734	0.9728	0.9700	0.9701
SD	0.0047	0.0000	0.0391	0.0054	0.0055	0.0055	0.0055

Check

In [60]:
```
t = time.process_time()
predicted = predict_model(model, data=unseen_data)
elapsed_time = time.process_time() - t
print("prediction took: " + str(elapsed_time))
```

```
prediction took: 3.9375
```

In [61]:
```
# 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 [62]:
```
compare_prediction_with_answers(predicted,answers)
```

```
number of error: 148 from 6189 test samples
 which is 0.02391339473259008 precent of error.
```

In [ ]: