Robust Test - Features Groups - distributed values

we will take each columns in out data and trasfer it with randomized vaules. the values will be in the range as the original column

Import

importing pandas package for handling data objects,

using pycaret only classification.

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

Functions and Constants

In [18]:
```
# set target feature
target_label = 'tuple'
# test imfuance over rf will satesfy
learning_models = ['rf']
# define numeric features which pycaret did not recognized
num_features = ['min_packet_size', 'min_fpkt', 'min_bpkt']
# set up features groups
SSL_features = ['fSSL_session_id_len', 'fSSL_num_extensions', 'fcipher_suites'
, 'ssl_v', ]
size_features = ['size_histogram_1','size_histogram_2','size_histogram_3',
                 'size_histogram_4','size_histogram_5','size_histogram_6',
                 'size_histogram_7','size_histogram_8','size_histogram_9', 'si
ze_histogram_10']
peak_features = ['fpeak_features_1','fpeak_features_2','fpeak_features_3',
                 'fpeak_features_4','fpeak_features_5','fpeak_features_6',
                 'fpeak_features_7','fpeak_features_8','fpeak_features_9',
                 'bpeak_features_1','bpeak_features_2','bpeak_features_3',
                 'bpeak_features_4','bpeak_features_5','bpeak_features_6',
                 'bpeak_features_7','bpeak_features_8','bpeak_features_9']
TCP_features = ['SYN_tcp_scale', 'SYN_tcp_winsize']
common_features = ['packet_count', 'fpackets', 'bpackets', 'fbytes', 'bbytes',
'num_keep_alive', 'mean_fttl']
stat_features = ['min_packet_size', 'max_packet_size', 'mean_packet_size',
                 'sizevar', 'std_fiat', # 'min_fiat', 'min_biat',
                'max_fiat','max_biat','std_biat','mean_fiat','mean_biat',
                'min_fpkt','min_bpkt','max_fpkt','max_bpkt','std_fpkt','std_bp
kt','mean_fpkt','mean_bpkt']
time_features = []
forward_features = []
backward_features = []
both_features = []
```

Read Data

In [3]:
```
data = pd.read_csv(target_label+r'_dataset\new_all_features_'+target_label+'.c
sv',
                      sep='\t',
                      skiprows=[1])
```

Setup Classifier and Compare

In [4]:
```
setup(data=data,
      target=target_label,
      numeric_features=num_features,
      silent=True)
compare_models(whitelist=learning_models)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
0	Random Forest Classifier	0.9719	0.0000	0.8644	0.9707	0.9703	0.9671	0.9671	4.8559

Out[4]:
```
OneVsRestClassifier(estimator=RandomForestClassifier(bootstrap=True,
                                                     ccp_alpha=0.0,
                                                     class_weight=None,
                                                     criterion='gini',
                                                     max_depth=None,
                                                     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=10, n_jobs=
-1,
                                                     oob_score=False,
                                                     random_state=3050,
                                                     verbose=0,
                                                     warm_start=False),
                    n_jobs=-1)
```

take care of the SSL Features

In [ ]:
```
features_group = SSL_features
new_data = pd.DataFrame(columns=features_group+[target_label])
print ('current columns are : ' + str(features_group))
for i in features_group:
    print ('for columns ' + str(i))
    new_data[i] = data[i]
```

In [ ]:
```
for i in features_group:
    max_value = 0
    value = 0
    min_value = new_data[i].values[0]
    for value in new_data[i]:
        if value > max_value: max_value = value
        if value < min_value: min_value = value
    print ('new_max_value = ' + str(max_value))
    print ('new_min_value = ' + str(min_value))
    print ('values: ' + str(new_data[i].unique()))
```

Setup and Check Only SSL

In [7]:
```
new_data[target_label] = data[target_label]
setup(data=new_data,
      target=target_label,
      silent=True)
compare_models(whitelist=learning_models)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
0	Random Forest Classifier	0.6679	0.0000	0.2856	0.6244	0.5784	0.5973	0.6084	1.2558

Out[7]:
```
OneVsRestClassifier(estimator=RandomForestClassifier(bootstrap=True,
                                                     ccp_alpha=0.0,
                                                     class_weight=None,
                                                     criterion='gini',
                                                     max_depth=None,
                                                     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=10, n_jobs=
-1,
                                                     oob_score=False,
                                                     random_state=8565,
                                                     verbose=0,
                                                     warm_start=False),
                    n_jobs=-1)
```

take care of the Size Features

In [ ]:
```
features_group = size_features
new_data = pd.DataFrame(columns=features_group+[target_label])
print ('current columns are : ' + str(features_group))
for i in features_group:
    print ('for columns ' + str(i))
    new_data[i] = data[i]
```

In [ ]:
```
for i in features_group:
    max_value = 0
    value = 0
    min_value = new_data[i].values[0]
    for value in new_data[i]:
        if value > max_value: max_value = value
        if value < min_value: min_value = value
    print ('new_max_value = ' + str(max_value))
    print ('new_min_value = ' + str(min_value))
    print ('values: ' + str(new_data[i].unique()))
```

Setup and Check Only Size

In [10]:
```
new_data[target_label] = data[target_label]
setup(data=new_data,
      target=target_label,
      silent=True)
compare_models(whitelist=learning_models)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
0	Random Forest Classifier	0.8668	0.0000	0.6705	0.8647	0.8623	0.8440	0.8442	0.7384

Out[10]:
```
OneVsRestClassifier(estimator=RandomForestClassifier(bootstrap=True,
                                                     ccp_alpha=0.0,
                                                     class_weight=None,
                                                     criterion='gini',
                                                     max_depth=None,
                                                     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=10, n_jobs=
-1,
                                                     oob_score=False,
                                                     random_state=6933,
                                                     verbose=0,
                                                     warm_start=False),
                    n_jobs=-1)
```

take care of the COMMON Features

In [ ]:
```
features_group = common_features
new_data = pd.DataFrame(columns=features_group+[target_label])
print ('current columns are : ' + str(features_group))
for i in features_group:
    print ('for columns ' + str(i))
    new_data[i] = data[i]
```

In [ ]:
```
for i in features_group:
    max_value = 0
    value = 0
    min_value = new_data[i].values[0]
    for value in new_data[i]:
        if value > max_value: max_value = value
        if value < min_value: min_value = value
    print ('new_max_value = ' + str(max_value))
    print ('new_min_value = ' + str(min_value))
    print ('values: ' + str(new_data[i].unique()))
```

Setup and Check Only COMMON

In [13]:
```
new_data[target_label] = data[target_label]
setup(data=new_data,
      target=target_label,
      silent=True)
compare_models(whitelist=learning_models)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
0	Random Forest Classifier	0.9042	0.0000	0.6931	0.9005	0.8997	0.8876	0.8878	0.6437

Out[13]:
```
OneVsRestClassifier(estimator=RandomForestClassifier(bootstrap=True,
                                                     ccp_alpha=0.0,
                                                     class_weight=None,
                                                     criterion='gini',
                                                     max_depth=None,
                                                     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=10, n_jobs=
-1,
                                                     oob_score=False,
                                                     random_state=946,
                                                     verbose=0,
                                                     warm_start=False),
                    n_jobs=-1)
```

take care of the TCP Features

```
In [ ]: features_group = TCP_features
        new_data = pd.DataFrame(columns=features_group+[target_label])
        print ('current columns are : ' + str(features_group))
        for i in features_group:
            print ('for columns ' + str(i))
            new_data[i] = data[i]
```

```
In [ ]: for i in features_group:
            max_value = 0
            value = 0
            min_value = new_data[i].values[0]
            for value in new_data[i]:
                if value > max_value: max_value = value
                if value < min_value: min_value = value
            print ('new_max_value = ' + str(max_value))
            print ('new_min_value = ' + str(min_value))
            print ('values: ' + str(new_data[i].unique()))
```

Setup and Check Only TCP

In [16]:
```
new_data[target_label] = data[target_label]
setup(data=new_data,
      target=target_label,
      silent=True)
compare_models(whitelist=learning_models)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
0	Random Forest Classifier	0.5744	0.0000	0.1359	0.4051	0.4564	0.4863	0.5113	0.5247

Out[16]:
```
OneVsRestClassifier(estimator=RandomForestClassifier(bootstrap=True,
                                                     ccp_alpha=0.0,
                                                     class_weight=None,
                                                     criterion='gini',
                                                     max_depth=None,
                                                     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=10, n_jobs=
-1,
                                                     oob_score=False,
                                                     random_state=3020,
                                                     verbose=0,
                                                     warm_start=False),
                    n_jobs=-1)
```

take care of the STAT Features

In [ ]:
```
features_group = stat_features
new_data = pd.DataFrame(columns=features_group+[target_label])
print ('current columns are : ' + str(features_group))
for i in features_group:
    print ('for columns ' + str(i))
    new_data[i] = data[i]
```

In [ ]:
```
for i in features_group:
    max_value = 0
    value = 0
    min_value = new_data[i].values[0]
    for value in new_data[i]:
        if value > max_value: max_value = value
        if value < min_value: min_value = value
    print ('new_max_value = ' + str(max_value))
    print ('new_min_value = ' + str(min_value))
    print ('values: ' + str(new_data[i].unique()))
```

Setup and Check Only STAT

In [21]:
```
new_data[target_label] = data[target_label]
setup(data=new_data,
      target=target_label,
      silent=True)
compare_models(whitelist=learning_models)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
0	Random Forest Classifier	0.9538	0.0000	0.8291	0.9534	0.9514	0.9459	0.9460	4.4965

Out[21]:
```
OneVsRestClassifier(estimator=RandomForestClassifier(bootstrap=True,
                                                     ccp_alpha=0.0,
                                                     class_weight=None,
                                                     criterion='gini',
                                                     max_depth=None,
                                                     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=10, n_jobs=
-1,
                                                     oob_score=False,
                                                     random_state=6878,
                                                     verbose=0,
                                                     warm_start=False),
                    n_jobs=-1)
```

take care of the STAT+SSL Features

In [ ]:
```
features_group = stat_features+SSL_features
new_data = pd.DataFrame(columns=features_group+[target_label])
print ('current columns are : ' + str(features_group))
for i in features_group:
    print ('for columns ' + str(i))
    new_data[i] = data[i]
```

```
In [ ]: for i in features_group:
            max_value = 0
            value = 0
            min_value = new_data[i].values[0]
            for value in new_data[i]:
                if value > max_value: max_value = value
                if value < min_value: min_value = value
            print ('new_max_value = ' + str(max_value))
            print ('new_min_value = ' + str(min_value))
            print ('values: ' + str(new_data[i].unique()))
```

Setup and Check Only STAT+SSL

```
In [24]: new_data[target_label] = data[target_label]
         setup(data=new_data,
               target=target_label,
               silent=True)
         compare_models(whitelist=learning_models)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
0	Random Forest Classifier	0.9699	0.0000	0.8630	0.9697	0.9685	0.9647	0.9648	4.3556

```
Out[24]: OneVsRestClassifier(estimator=RandomForestClassifier(bootstrap=True,
                                                              ccp_alpha=0.0,
                                                              class_weight=None,
                                                              criterion='gini',
                                                              max_depth=None,
                                                              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=10, n_jobs=
         -1,
                                                              oob_score=False,
                                                              random_state=3677,
                                                              verbose=0,
                                                              warm_start=False),
                             n_jobs=-1)
```

take care of the COMMON+SSL Features

In [ ]:
```
features_group = common_features+SSL_features
new_data = pd.DataFrame(columns=features_group+[target_label])
print ('current columns are : ' + str(features_group))
for i in features_group:
    print ('for columns ' + str(i))
    new_data[i] = data[i]
```

In [ ]:
```
for i in features_group:
    max_value = 0
    value = 0
    min_value = new_data[i].values[0]
    for value in new_data[i]:
        if value > max_value: max_value = value
        if value < min_value: min_value = value
    print ('new_max_value = ' + str(max_value))
    print ('new_min_value = ' + str(min_value))
    print ('values: ' + str(new_data[i].unique()))
```

Setup and Check Only COMMON+SSL

In [27]:
```
new_data[target_label] = data[target_label]
setup(data=new_data,
      target=target_label,
      silent=True)
compare_models(whitelist=learning_models)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
0	Random Forest Classifier	0.9491	0.0000	0.8024	0.9485	0.9474	0.9404	0.9405	4.0900

Out[27]:
```
OneVsRestClassifier(estimator=RandomForestClassifier(bootstrap=True,
                                                     ccp_alpha=0.0,
                                                     class_weight=None,
                                                     criterion='gini',
                                                     max_depth=None,
                                                     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=10, n_jobs=
-1,

                                                     oob_score=False,
                                                     random_state=4138,
                                                     verbose=0,
                                                     warm_start=False),
                    n_jobs=-1)
```

take care of the STAT+COMMON+SSL Features

In [ ]:
```
features_group = stat_features+common_features+SSL_features
new_data = pd.DataFrame(columns=features_group+[target_label])
print ('current columns are : ' + str(features_group))
for i in features_group:
    print ('for columns ' + str(i))
    new_data[i] = data[i]
```

In [ ]:
```
for i in features_group:
    max_value = 0
    value = 0
    min_value = new_data[i].values[0]
    for value in new_data[i]:
        if value > max_value: max_value = value
        if value < min_value: min_value = value
    print ('new_max_value = ' + str(max_value))
    print ('new_min_value = ' + str(min_value))
    print ('values: ' + str(new_data[i].unique()))
```

Setup and Check Only STAT+COMMON+SSL

In [30]:
```
new_data[target_label] = data[target_label]
setup(data=new_data,
      target=target_label,
      silent=True)
compare_models(whitelist=learning_models)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
0	Random Forest Classifier	0.9707	0.0000	0.8658	0.9697	0.9691	0.9657	0.9658	4.3976

Out[30]:
```
OneVsRestClassifier(estimator=RandomForestClassifier(bootstrap=True,
                                                     ccp_alpha=0.0,
                                                     class_weight=None,
                                                     criterion='gini',
                                                     max_depth=None,
                                                     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=10, n_jobs=
-1,
                                                     oob_score=False,
                                                     random_state=422,
                                                     verbose=0,
                                                     warm_start=False),
                    n_jobs=-1)
```

In [ ]: