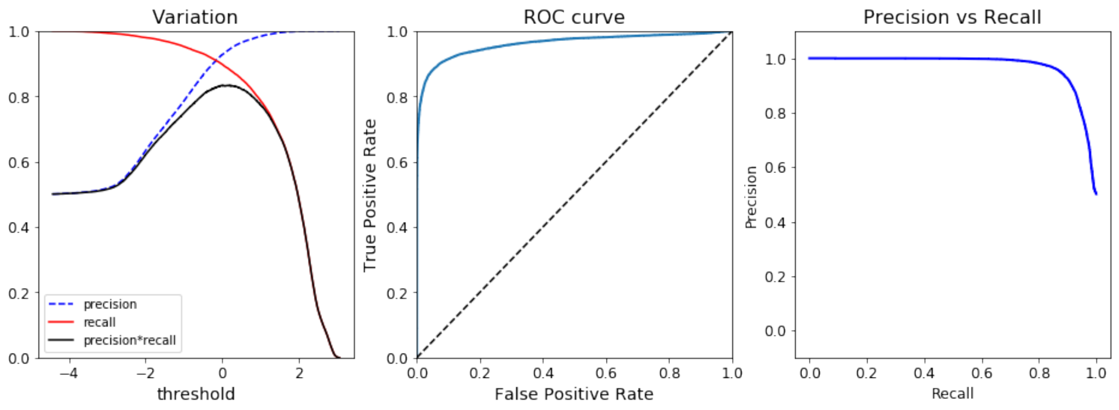
MachineLearning

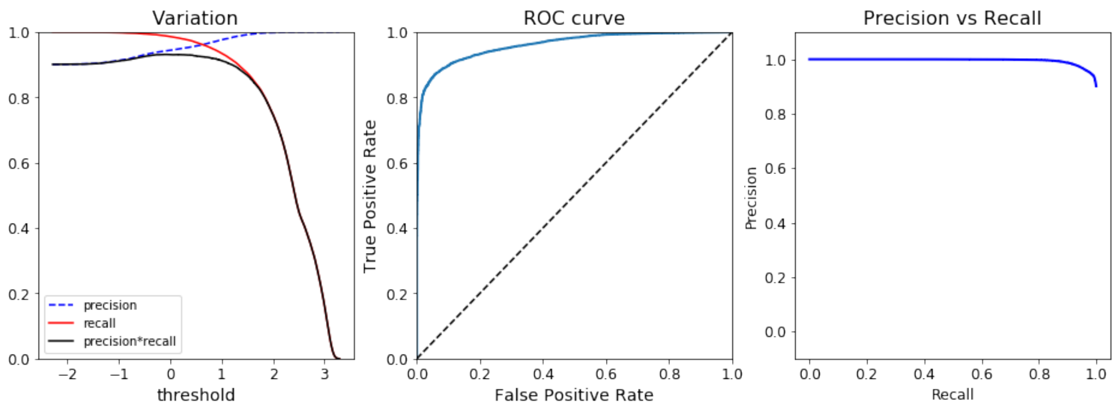
May 29, 2019

```
In [41]:
        #SGDClassifier

        from sklearn.linear_model import SGDClassifier
        from sklearn.pipeline import Pipeline
        from sklearn.decomposition import KernelPCA
        sgd_clf = SGDClassifier(random_state=42)
        sgd_clf.fit(data_trainF, data_trainL)
        print("on train sample")
        plotResult(sgd_clf, data_trainF, data_trainL, 3)
        print("on test sample")
        fpr_sgd, tpr_sgd = plotResult(sgd_clf, data_testF, data_testL, 3)
```

```
on train sample
('precision*recal is ', 0.83282072170042065)
('the area under ROC is : ', 0.96117332026504021)
```

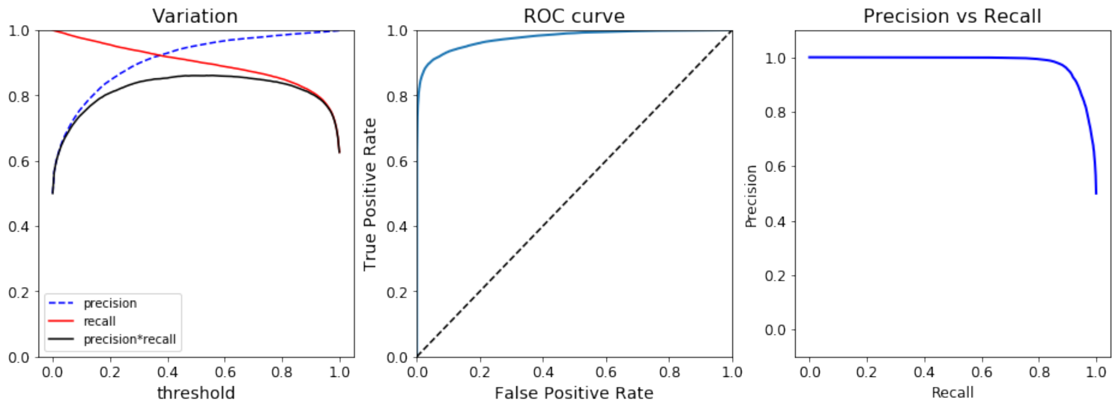


```
on test sample
('precision*recal is ', 0.93083332394343754)
('the area under ROC is : ', 0.96071645783663717)
```

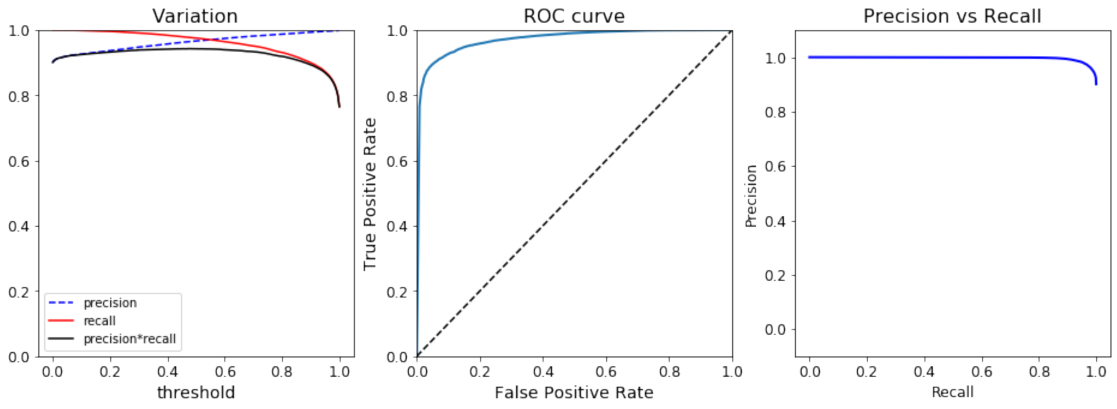


```
In [8]: from sklearn.ensemble import RandomForestClassifier
        forest_clf = RandomForestClassifier(n_estimators=10, random_state=42)
        #grid search
        param_grid = [
                      {'n_estimators': [3, 10, 30], 'max_features': [2, 3, 4, 5]},
                      {'bootstrap': [False], 'n_estimators': [3, 10], 'max_features': [2, 3, 4]}
                      ]

        #random search
        param_distribs = {'n_estimators':randint(low=1, high=200),
            'max_features':randint(low=1, high=6),}

        rf_clf = best_model(forest_clf, param_grid, param_distribs, data_trainF, data_trainL);

        sort_features(rf_clf)
        print("on train sample")
        plotResult(rf_clf, data_trainF, data_trainL, 3)
        print("on test sample")
        fpr_rf, tpr_rf = plotResult(rf_clf, data_testF, data_testL, 3)
```

```
after grid search :
('search.best_params_', {'max_features': 4, 'n_estimators': 30})
('search.best_estimator_', RandomForestClassifier(bootstrap=True, class_weight=None, criterion='
            max_depth=None, max_features=4, 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, n_estimators=30, n_jobs=1,
            oob_score=False, random_state=42, verbose=0, warm_start=False))
after random search :
('search.best_estimator_', RandomForestClassifier(bootstrap=True, class_weight=None, criterion='
            max_depth=None, max_features=2, 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, n_estimators=192, n_jobs=1,
            oob_score=False, random_state=42, verbose=0, warm_start=False))
('features : ', Index([u'thrust', u'CParameter', u'DParameter', u'HeavyMass',
       u'WideBroadening', u'totalBroadening'],
      dtype='object'))
the importance of each feature listed in the following:
('thrust', 0.29154291816365913)
('CParameter', 0.18962083845285427)
('DParameter', 0.21961095829660662)
('HeavyMass', 0.094253703415084419)
('WideBroadening', 0.056533798832532504)
('totalBroadening', 0.14843778283926329)
on train sample
('precision*recal is ', 0.86018832233493814)
('the area under ROC is : ', 0.97578586022451286)
```

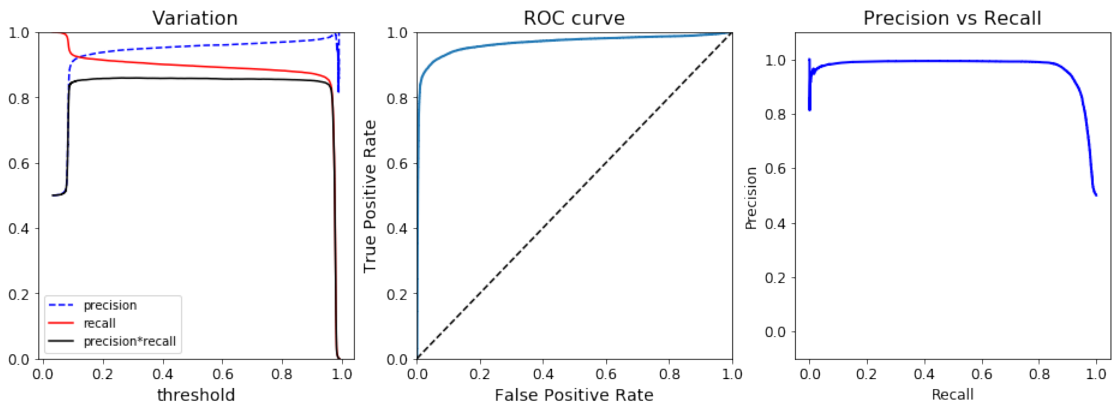


```
on test sample
('precision*recal is ', 0.94231861972627540)
('the area under ROC is : ', 0.97113840593413847)
```

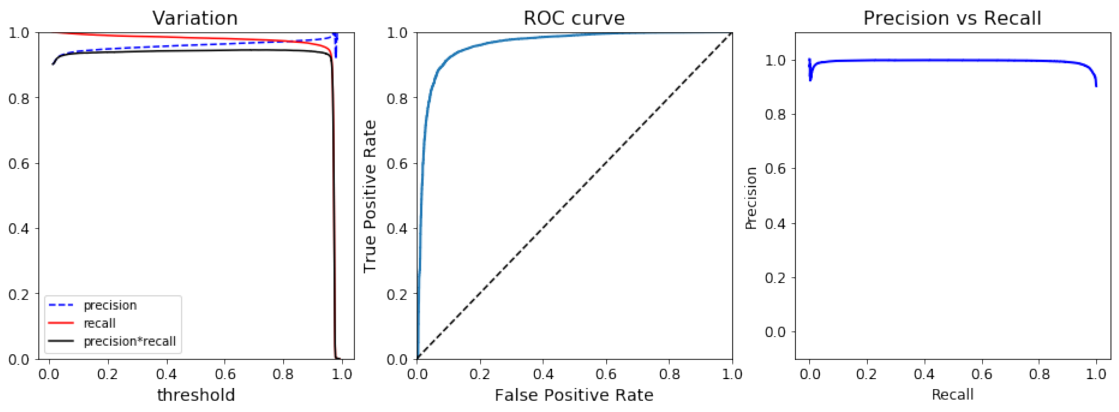


```
In [9]: #Support Vector Machine
        from sklearn.svm import SVC
        from sklearn.preprocessing import StandardScaler
        from sklearn.pipeline import Pipeline

        #rbf
        rbf_kernel_svm_clf = Pipeline([
                                        ("scaler", StandardScaler()),
                                        ("svm_clf", SVC(kernel="rbf", gamma=5, C = 0.1, probabili
                                        ])

        rbf_kernel_svm_clf.fit(data_trainF, data_trainL)

        print("on train sample")
        plotResult(rbf_kernel_svm_clf, data_trainF, data_trainL, 3)
        print("on test sample")
        fpr_svm_rbf, tpr_svm_rbf = plotResult(rbf_kernel_svm_clf, data_testF, data_testL, 3)
```

```
on train sample
('precision*recal is ', 0.85620920960532698)
('the area under ROC is : ', 0.9653197456121394)
```

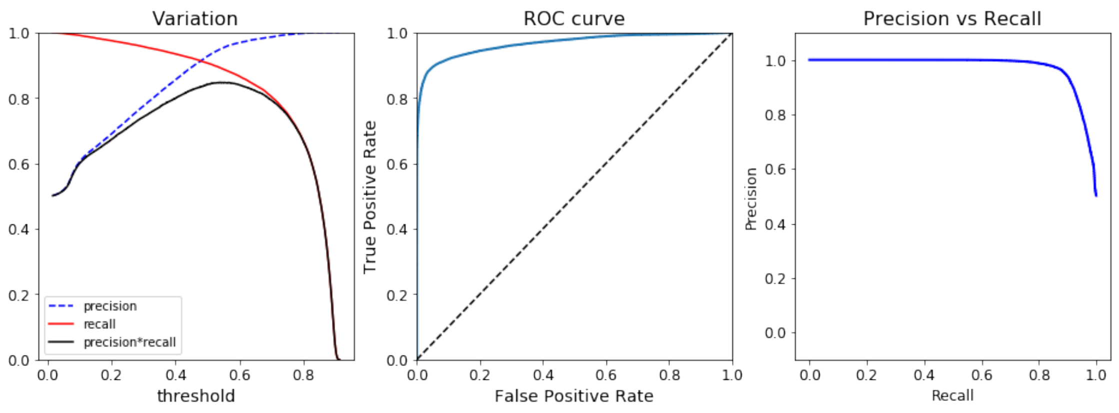


```
on test sample
('precision*recal is ', 0.94305850368794664)
('the area under ROC is : ', 0.95956936963550032)
```

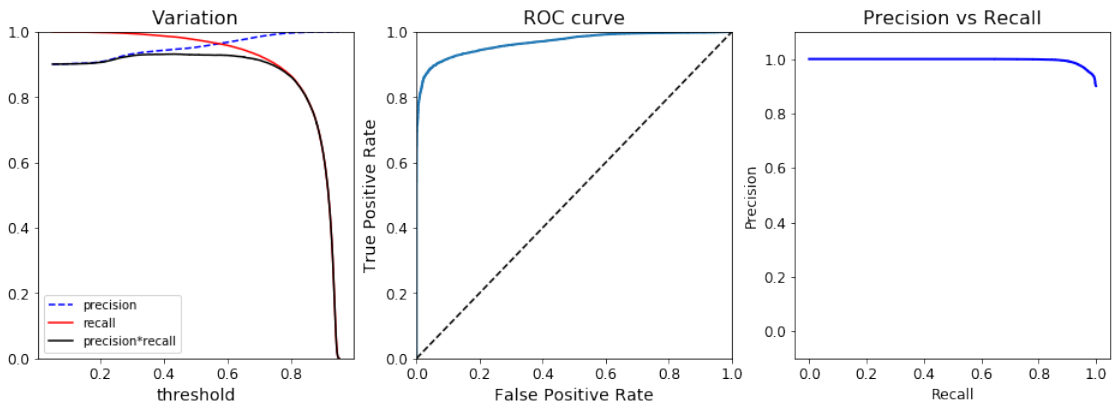


```
#Softmax Regression
from sklearn.linear_model import LogisticRegression
softmax_reg = LogisticRegression(multi_class="multinomial",solver="lbfgs",C=10)
softmax_reg.fit(data_trainF, data_trainL)
#sort_features(softmax_reg)
print("on train sample")
plotResult(softmax_reg, data_trainF, data_trainL, 3)
print("on test smaple")
fpr_sftmx, tpr_sftmx = plotResult(softmax_reg, data_testF, data_testL, 3)
```

```
on train sample
('precision*recal is ', 0.841377140081429O5)
('the area under ROC is : ', 0.96643261432248628)
```



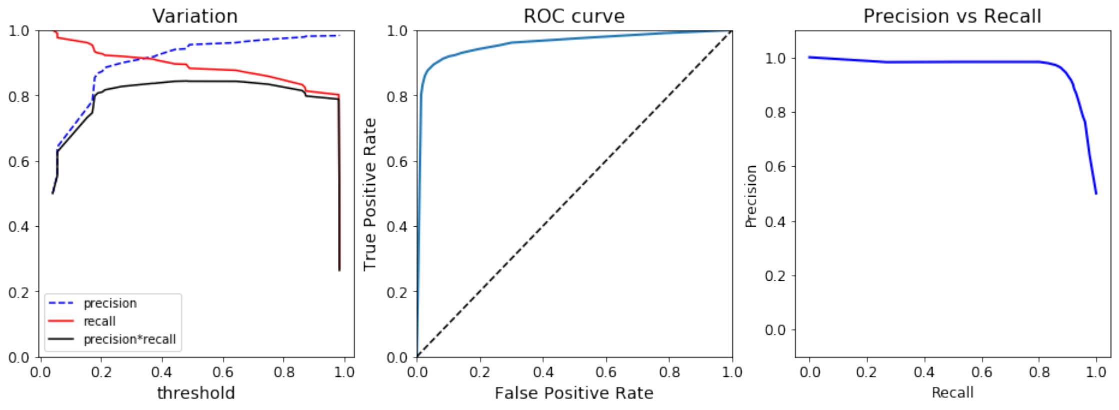
```
on test smaple
('precision*recal is ', 0.92941465348342334)
('the area under ROC is : ', 0.96736839942326813)
```



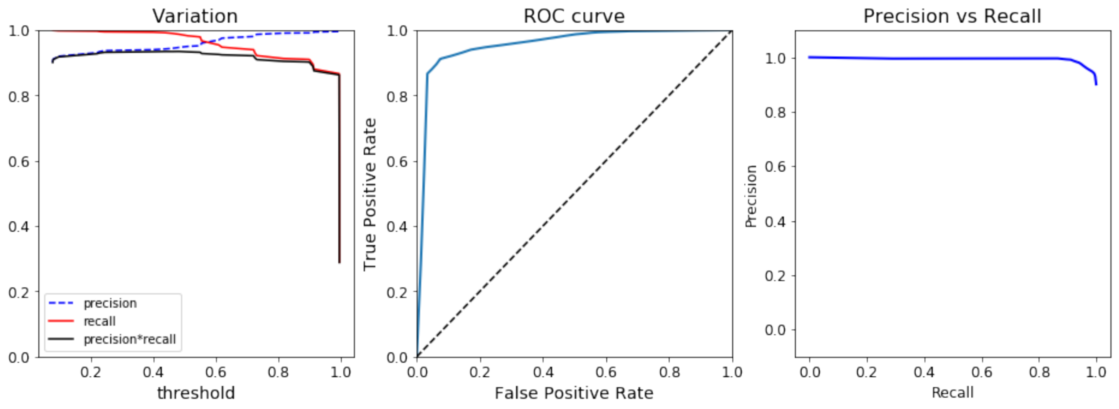
```
In [11]: #Decision tree

         from sklearn.tree import DecisionTreeClassifier
         newtree_clf = DecisionTreeClassifier(splitter = "random", max_depth = 6, min_samples_le
         params = {'max_leaf_nodes':list(range(2, 10)), 'min_samples_split':list(range(50, 103))
         best_gridModel = grid_search(newtree_clf, params, 3, "roc_auc", data_trainF, data_train
         tree_clf = best_gridModel
         tree_clf.fit(data_trainF, data_trainL)
         sort_features(tree_clf)
         print("on train sample")
         plotResult(tree_clf, data_trainF, data_trainL, 3)
         print("on test sample")
         fpr_tree, tpr_tree = plotResult(tree_clf, data_testF, data_testL, 3)
```

```
after grid search :
('search.best_params_', {'min_samples_split': 50, 'max_leaf_nodes': 9})
('search.best_estimator_', DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth
            max_features=None, max_leaf_nodes=9, min_impurity_decrease=0.0,
            min_impurity_split=None, min_samples_leaf=4,
            min_samples_split=50, min_weight_fraction_leaf=0.0,
            presort=False, random_state=42, splitter='random'))
('features : ', Index([u'thrust', u'CParameter', u'DParameter', u'HeavyMass',
       u'WideBroadening', u'totalBroadening'],
      dtype='object'))
the importance of each feature listed in the following:
('thrust', 0.9491707683545684 8)
('CParameter', 0.023756337321780025)
('DParameter', 0.019510926871802503)
('HeavyMass', 0.0)
('WideBroadening', 0.007561967451848989 8)
('totalBroadening', 0.0)
on train sample
('precision*recal is ', 0.8431087327212586 4)
```

```
('the area under ROC is : ', 0.95782228662678159)
```

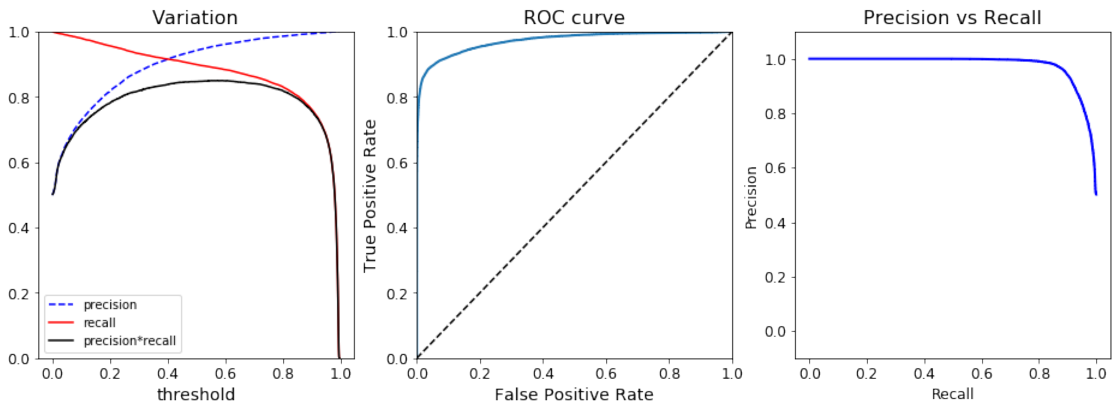


```
on test sample
('precision*recal is ', 0.93324712463507598)
('the area under ROC is : ', 0.9542481386478211)
```

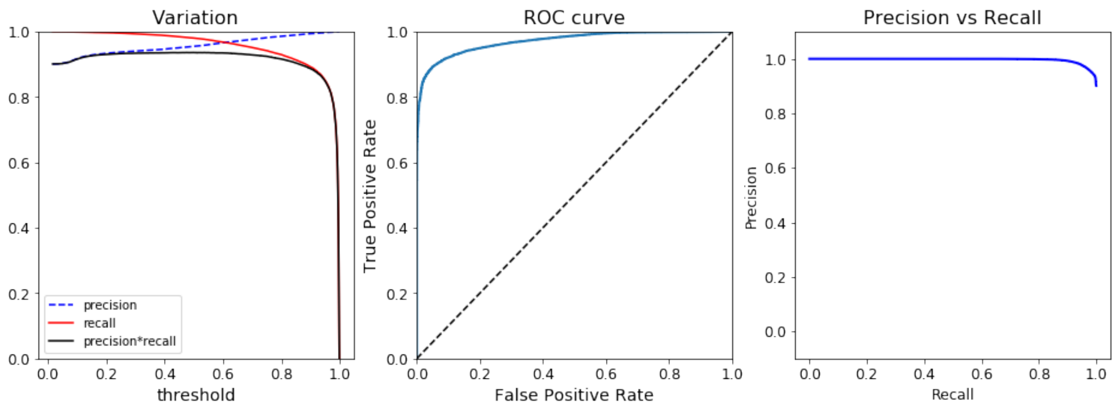


```
In [12]: #ensemble different kind of classfiers

         from sklearn.ensemble import RandomForestClassifier
         from sklearn.ensemble import VotingClassifier
         from sklearn.linear_model import LogisticRegression
         from sklearn.svm import SVC
         log_clf = LogisticRegression(solver="liblinear", random_state=42)
         rnd_clf = RandomForestClassifier(n_estimators=10, random_state=42)
         svm_clf = SVC(gamma="auto", probability=True, random_state=42)
         voting_clf = VotingClassifier(estimators=[('lr', log_clf), ('rf', rnd_clf), ('svc', svm
         voting_clf.fit(data_trainF, data_trainL)
         print("on train sample")
         plotResult(voting_clf, data_trainF, data_trainL, 3)
         print("on test sample")
         fpr_vot, tpr_vot = plotResult(voting_clf, data_testF, data_testL, 3)
```

```
on train sample
('precision*recal is ', 0.84684583642935252)
('the area under ROC is : ', 0.97177798227232781)
```



```
on test sample
('precision*recal is ', 0.93598202005155851)
('the area under ROC is : ', 0.97061901002975581)
```

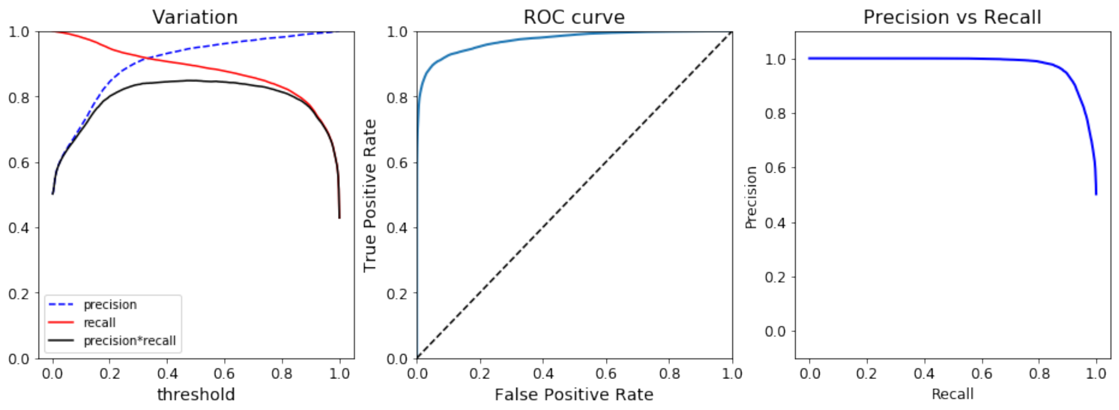


```
('LogisticRegression', 0.89425377321231903, 'f1_score :', 0.94454751597608744)
('the area under ROC is : ', 0.96380058556813497)
```

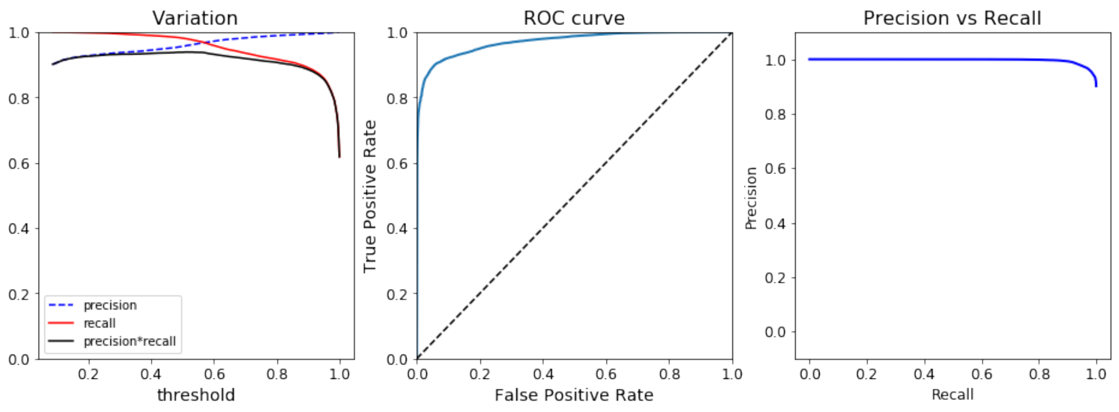
```
('RandomForestClassifier', 0.89895008919864372, 'f1_score :', 0.94705397610218378)
('the area under ROC is : ', 0.97113840593413847)
('SVC', 0.8835817230813805, 'f1_score :', 0.93852196632810203)
('the area under ROC is : ', 0.96683081413924887)
('VotingClassifier', 0.89404428952993908, 'f1_score :', 0.94438521066208081)
('the area under ROC is : ', 0.97061901002975581)
```

```
In [23]: #ensemble the same classfier

         from sklearn.ensemble import BaggingClassifier
         from sklearn.tree import DecisionTreeClassifier
         bag_clf = BaggingClassifier(DecisionTreeClassifier(), n_estimators=500, max_samples=100
                                     bootstrap=True, n_jobs=-1, oob_score=True)
         bag_clf.fit(data_trainF, data_trainL)
         print("on train sample")
         plotResult(bag_clf, data_trainF, data_trainL, 3)
         print("on test sample")
         fpr_bag, tpr_bag = plotResult(bag_clf, data_testF, data_testL, 3)
```

```
on train sample
('precision*recal is ', 0.84872839757770421)
('the area under ROC is : ', 0.97231079898932393)
```

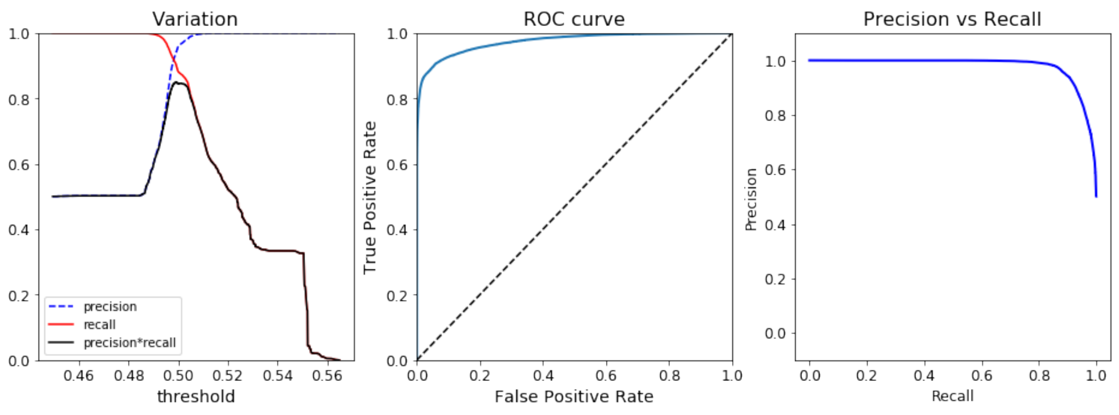


```
on test sample
('precision*recal is ', 0.93665108999243185)
('the area under ROC is : ', 0.97020863809293023)
```



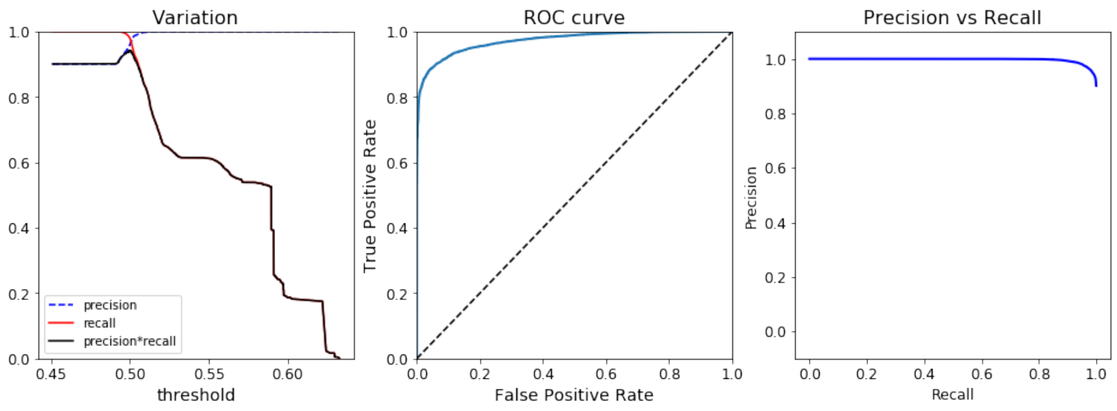
```
In [28]: #boosting

         from sklearn.ensemble import  AdaBoostClassifier
         ada_clf = AdaBoostClassifier(DecisionTreeClassifier(max_depth=1), n_estimators=200, alg
                                      learning_rate=0.5)
         ada_clf.fit(data_trainF, data_trainL)
         print("on training set")
         plotResult(ada_clf, data_trainF, data_trainL, 3)
         print("on testing set")
         fpr_ada, tpr_ada = plotResult(ada_clf, data_testF, data_testL, 3)
         sort_features(ada_clf)
```

```
on training set
('precision*recal is ', 0.846373689992290001)
('the area under ROC is : ', 0.97412642992169174)
```



```
on testing set
('precision*recal is ', 0.94009619125648891)
```

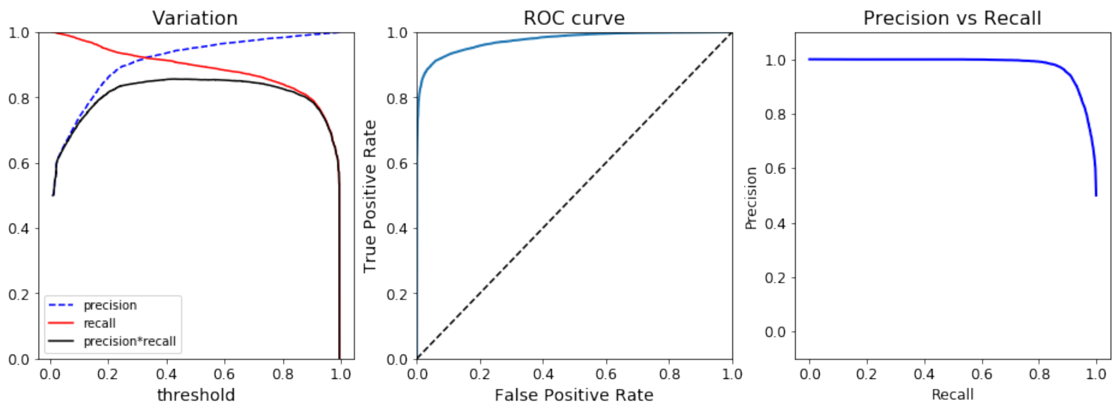
```
('the area under ROC is : ', 0.97257402408533633)
```



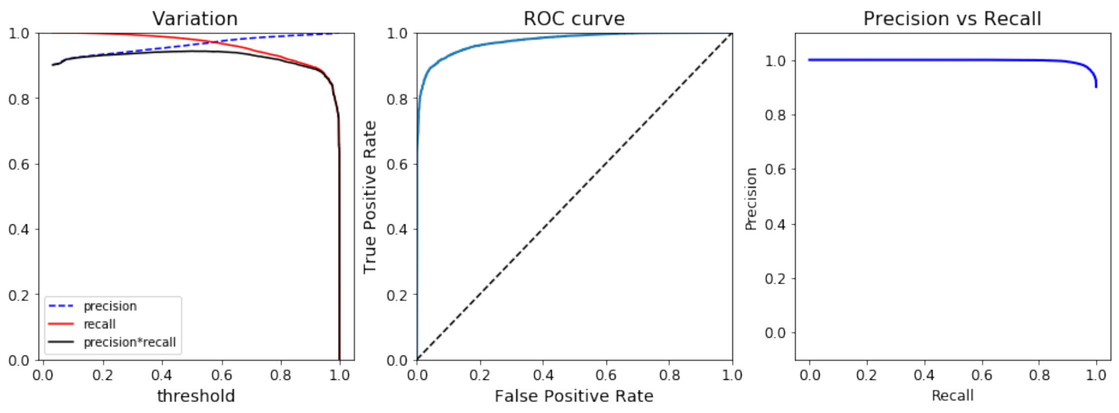
```
('features : ', Index([u'thrust', u'CParameter', u'DParameter', u'HeavyMass',
       u'WideBroadening', u'totalBroadening'],
      dtype='object'))
the importance of each feature listed in the following:
('thrust', 0.19500000000000001)
('CParameter', 0.245)
('DParameter', 0.085000000000000006)
('HeavyMass', 0.044999999999999998)
('WideBroadening', 0.245)
('totalBroadening', 0.185)
```

```
In [24]: #gradient boosting
         from sklearn.ensemble import GradientBoostingClassifier
         grb_clf = GradientBoostingClassifier(max_depth=2, n_estimators=120)
         n_estimators = NEstimators(grb_clf, data_trainF, data_trainL, data_testF, data_testL)
         print("n_estimators : ", n_estimators)
         grb_clf = GradientBoostingClassifier(max_depth=2, n_estimators=n_estimators)

         print("on training set")
         plotResult(grb_best, data_trainF, data_trainL, 3)
         print("on testing set")
         fpr_grb, tpr_grb = plotResult(grb_best, data_testF, data_testL, 3)
```

```
('n_estimators : ', 111)
on training set
('precision*recal is ', 0.85492789126047786)
('the area under ROC is : ', 0.97476142892020434)
```



```
on testing set
('precision*recal is ', 0.94320220101402719)
('the area under ROC is : ', 0.97410806997488042)
```


Variation
precision
recall
precision*recall
threshold
ROC curve
True Positive Rate
False Positive Rate
Precision vs Recall
Precision
Recall