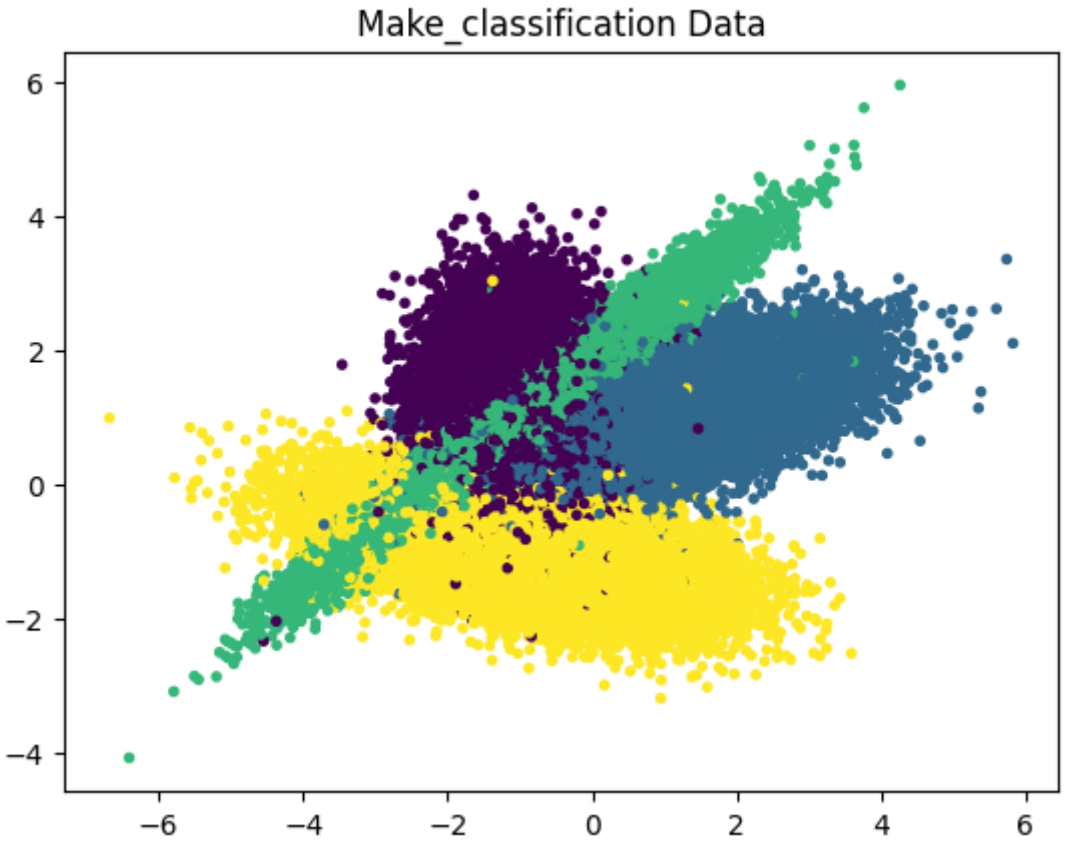
In [ ]:
```
#加载Intel的scikit-learn加速
from sklearnex import patch_sklearn
patch_sklearn(global_patch=True)
```

```
Scikit-learn was successfully globally patched by Intel(R) Extension for Scikit-learn
Intel(R) Extension for Scikit-learn* enabled (https://github.com/intel/scikit-learn-intelex)
```

In [ ]:
```
#用sklearn随机生成一个有3个分类的数据集，然后用KNN算法进行分类
from sklearn.datasets import make_classification
from sklearn.metrics import accuracy_score
from sklearn.metrics import r2_score # R2评分，R2值越接近1，表示模型越好，越接近0，表示模型越差
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score



x,y = make_classification(n_samples=100000,n_features=3,n_classes=4,n_informative=3,n_redundant=0,random_state=50,n_clusters_per_class=1)


#取其中的两个维度进行绘图
import matplotlib.pyplot as plt
plt.title('Make_classification Data')
plt.scatter(x[:,0],x[:,1],marker='.',c=y)
plt.show()

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.1)
```

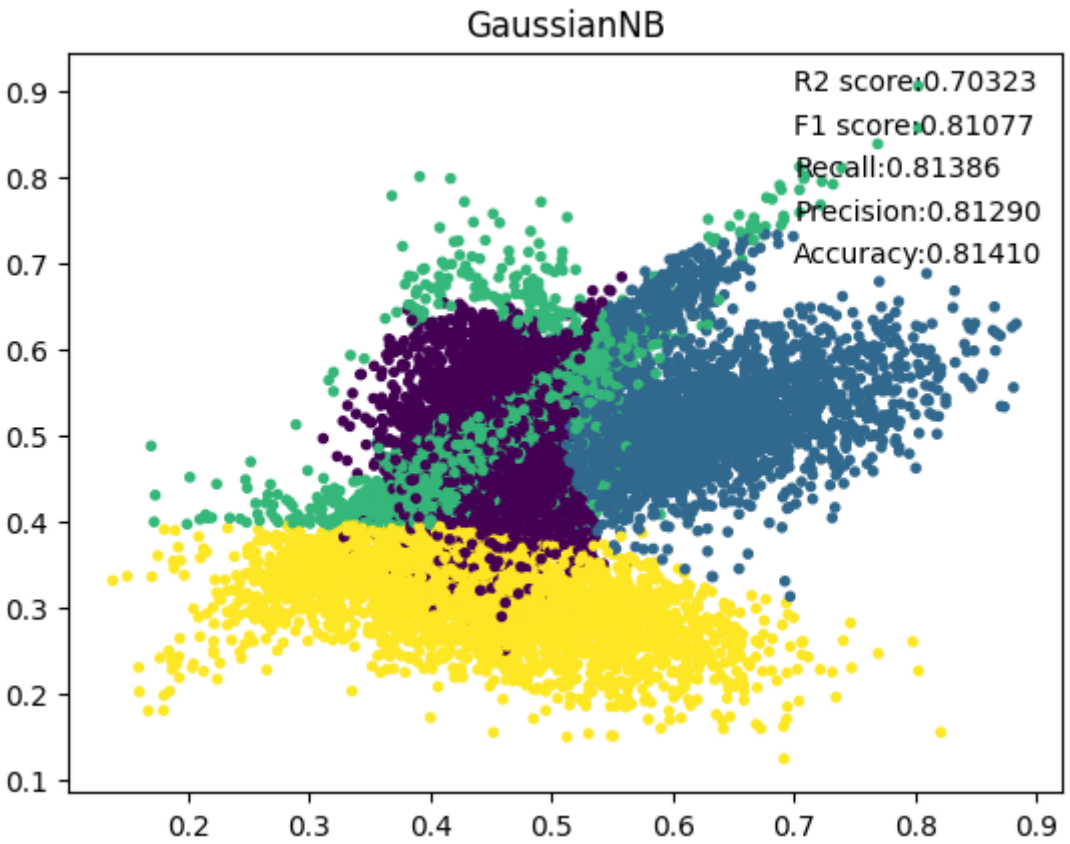


In [ ]:
```
#将数据归一化，数据都是正数
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(x_train)
x_train = scaler.transform(x_train)
x_test = scaler.transform(x_test)
```

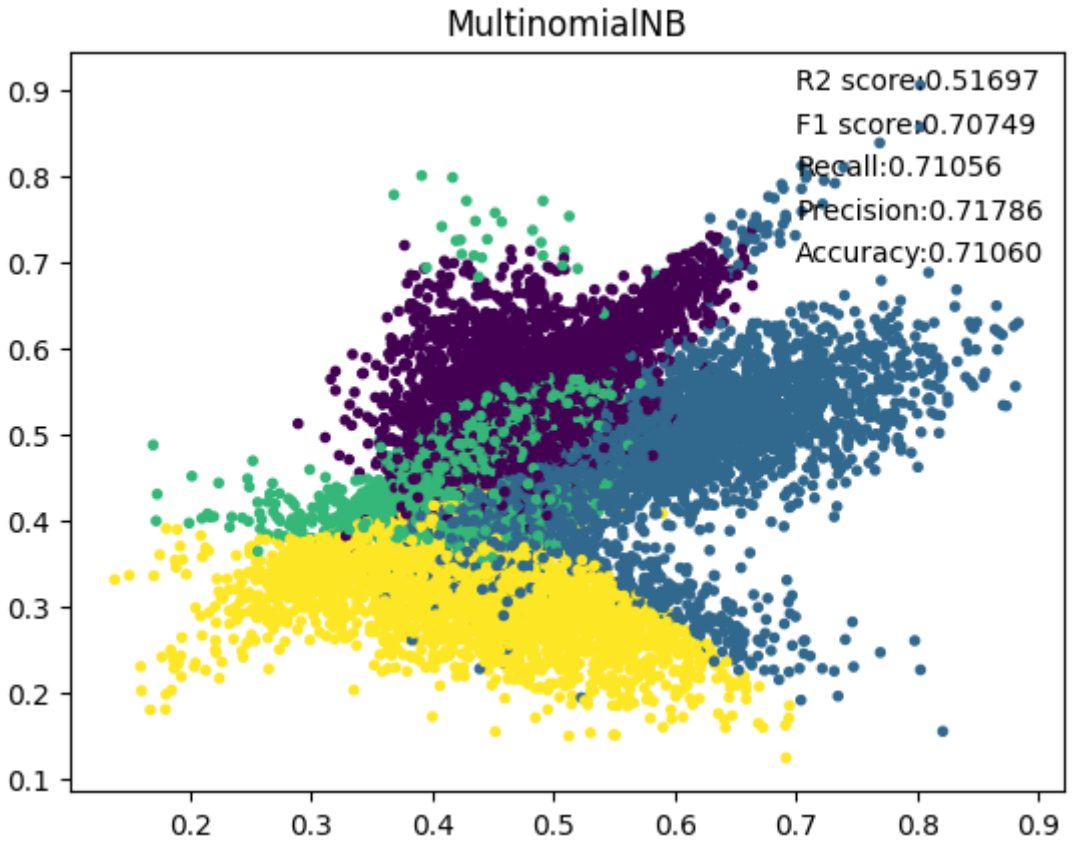
In [ ]:
```
def accurate(title):
    plt.title(title)
    plt.scatter(x_test[:, 0], x_test[:, 1], marker='.', c=y_predict)
    plt.text(0.7, 0.7, 'Accuracy:%.5f' % accuracy_score(y_test, y_predict))
    plt.text(0.7, 0.75, 'Precision:%.5f' % precision_score(y_test, y_predict, average='macro'))
    plt.text(0.7, 0.8, 'Recall:%.5f' % recall_score(y_test, y_predict, average='macro'))
    plt.text(0.7, 0.85, 'F1 score:%.5f' % f1_score(y_test, y_predict, average='macro'))
    plt.text(0.7, 0.9, 'R2 score:%.5f' % r2_score(y_test, y_predict))
    plt.show()
```

In [ ]:
```
# 用高斯贝叶斯算法进行分类
from sklearn.naive_bayes import GaussianNB, MultinomialNB
```

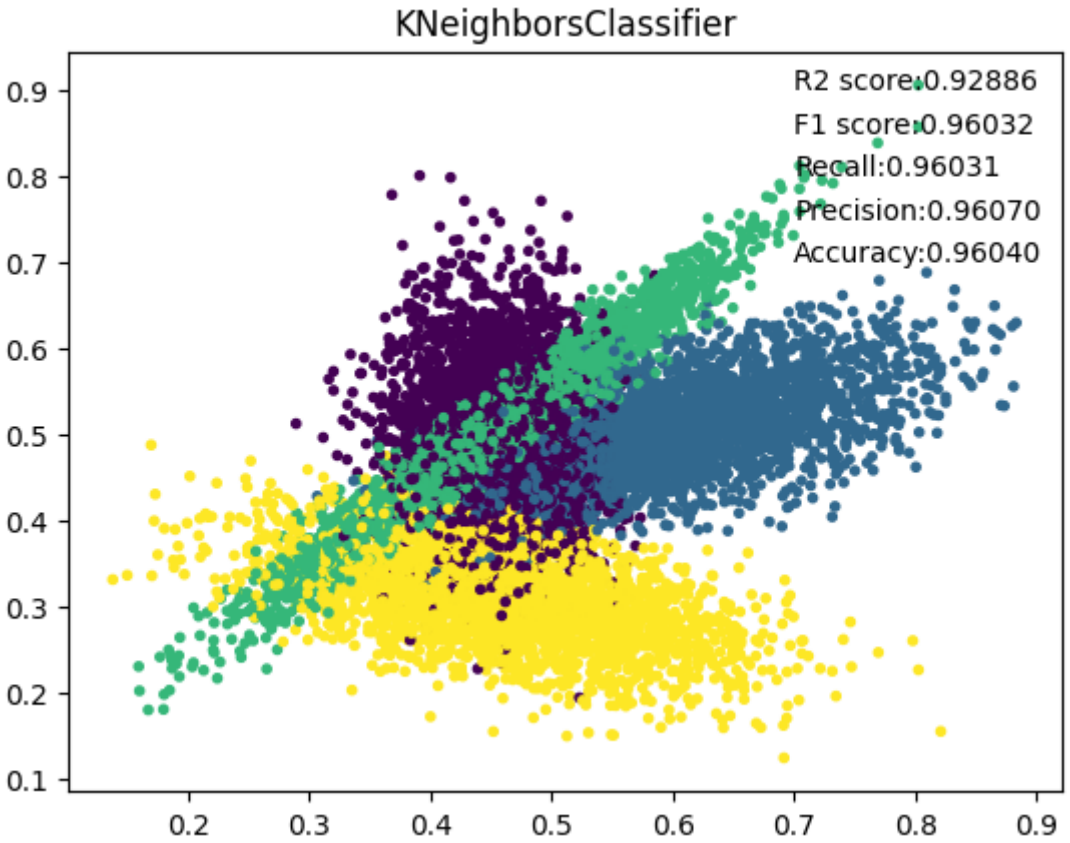
```
gnb = GaussianNB()
gnb.fit(x_train, y_train)
y_predict = gnb.predict(x_test)

accurate('GaussianNB')
```



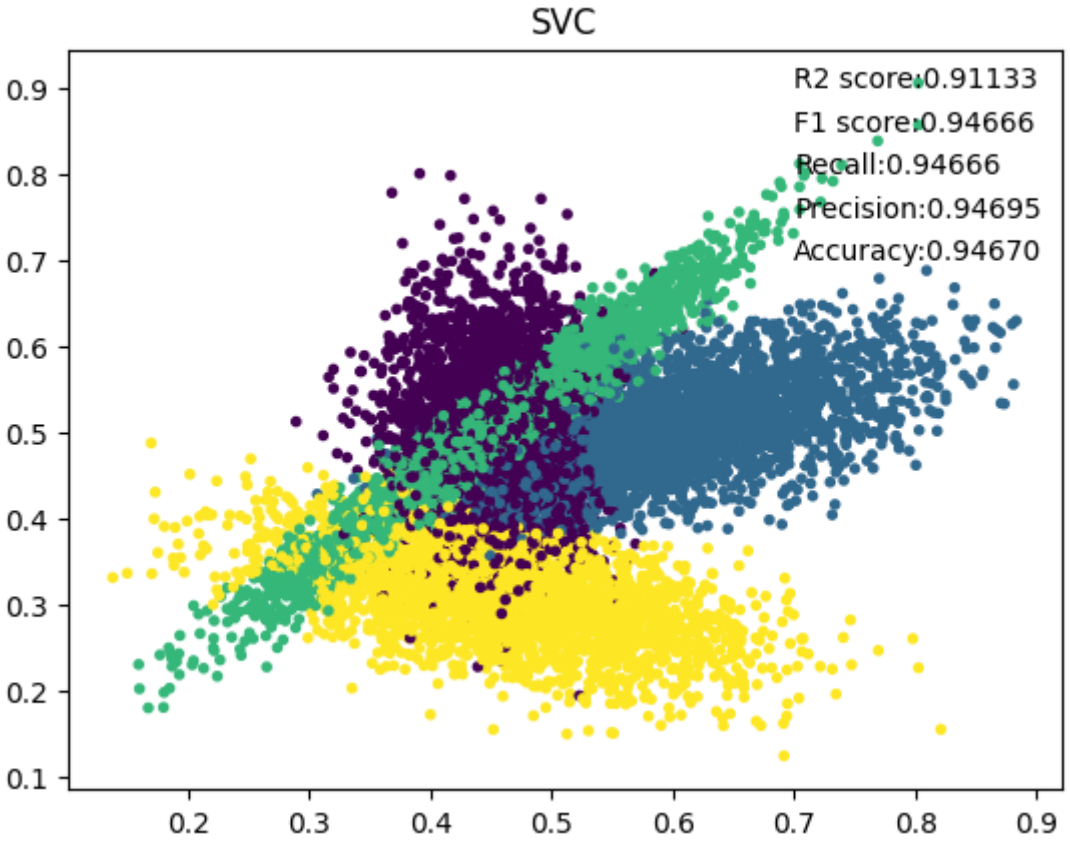
In [ ]:
```
#用多项式贝叶斯算法进行分类
mnb = MultinomialNB()
mnb.fit(x_train,y_train)
y_predict = mnb.predict(x_test)

accurate('MultinomialNB')
```



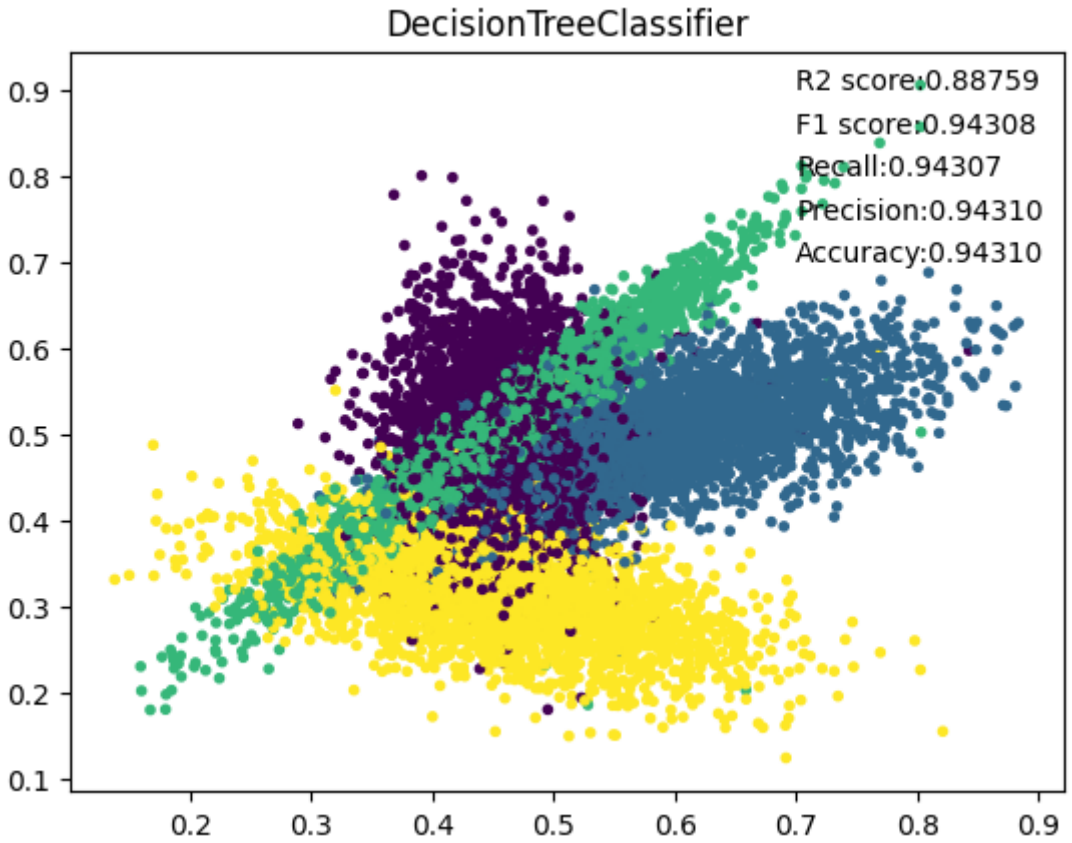
In [ ]:
```python
#用K近邻算法进行分类
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=30)
knn.fit(x_train,y_train)
y_predict = knn.predict(x_test)

accurate('KNeighborsClassifier')
```



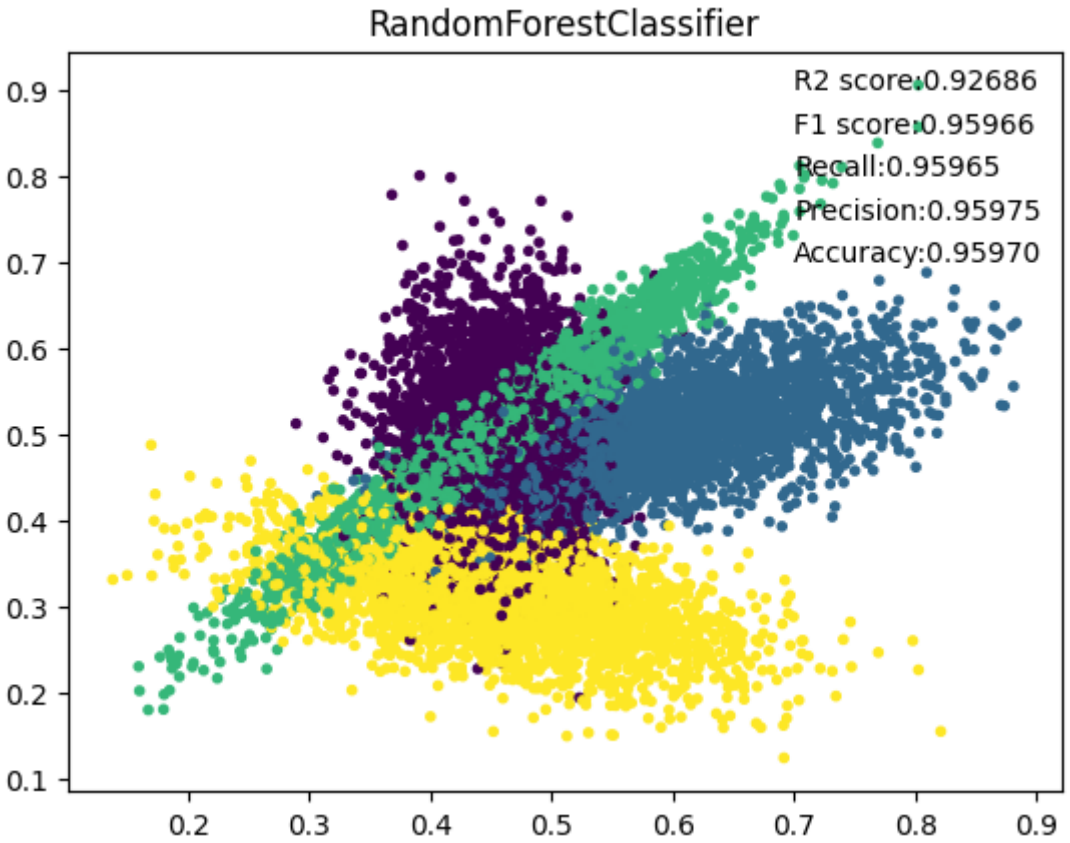
In [ ]:
```
#用SVM算法进行分类
from sklearn.svm import SVC
svm = SVC()
svm.fit(x_train,y_train)
y_predict = svm.predict(x_test)

accurate('SVC')
```



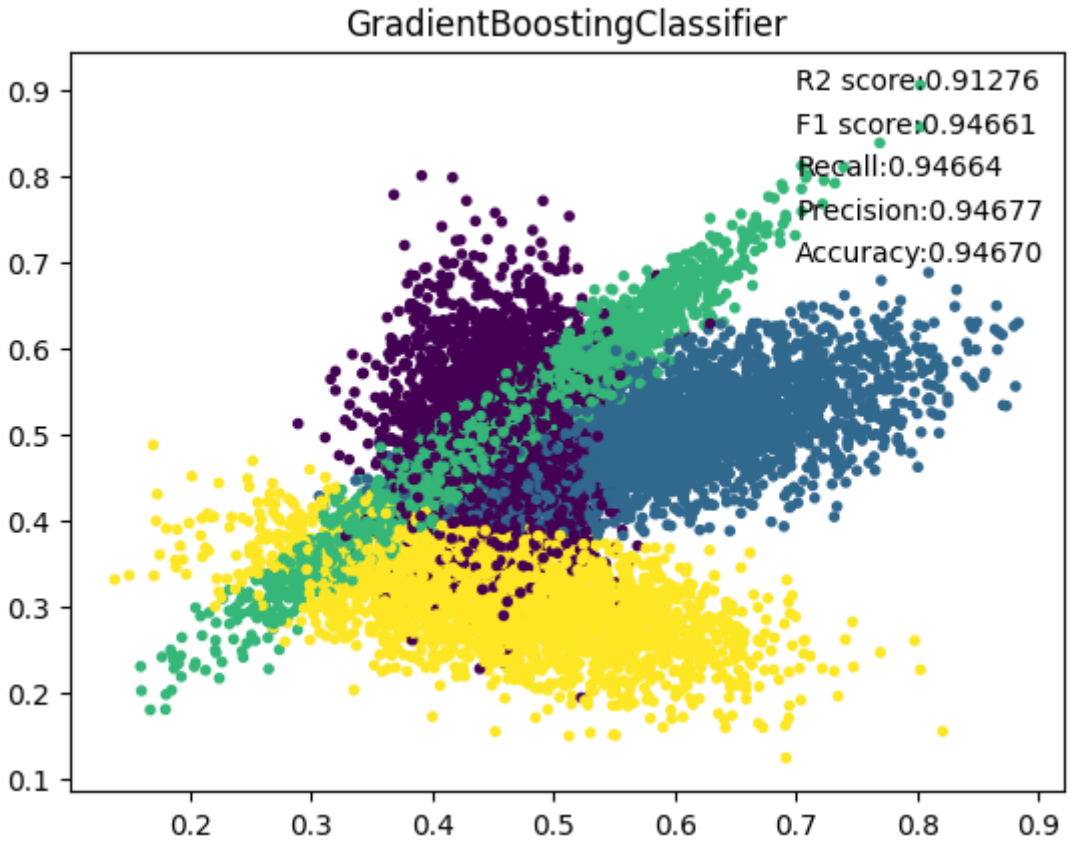
In [ ]:
```python
#用决策树算法进行分类
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
dtc.fit(x_train,y_train)
y_predict = dtc.predict(x_test)

accurate('DecisionTreeClassifier')
```



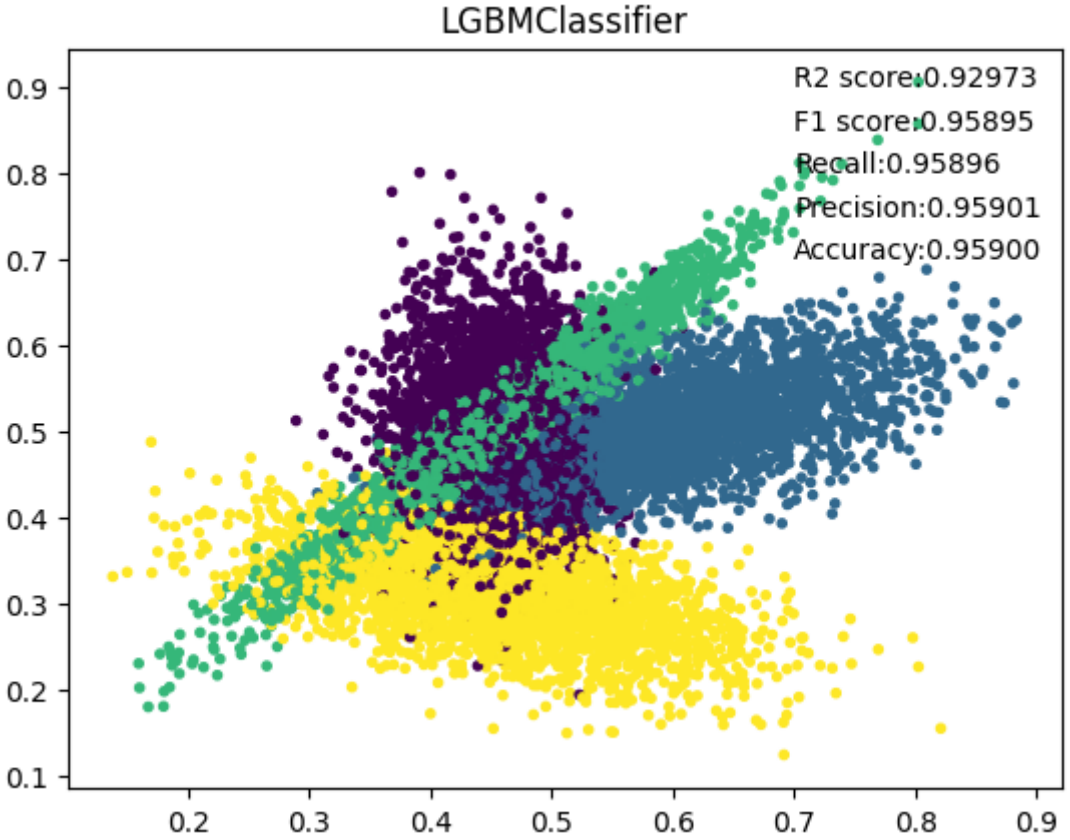
In [ ]:
```
#用随机森林算法进行分类
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)
y_predict = rfc.predict(x_test)

accurate('RandomForestClassifier')
```



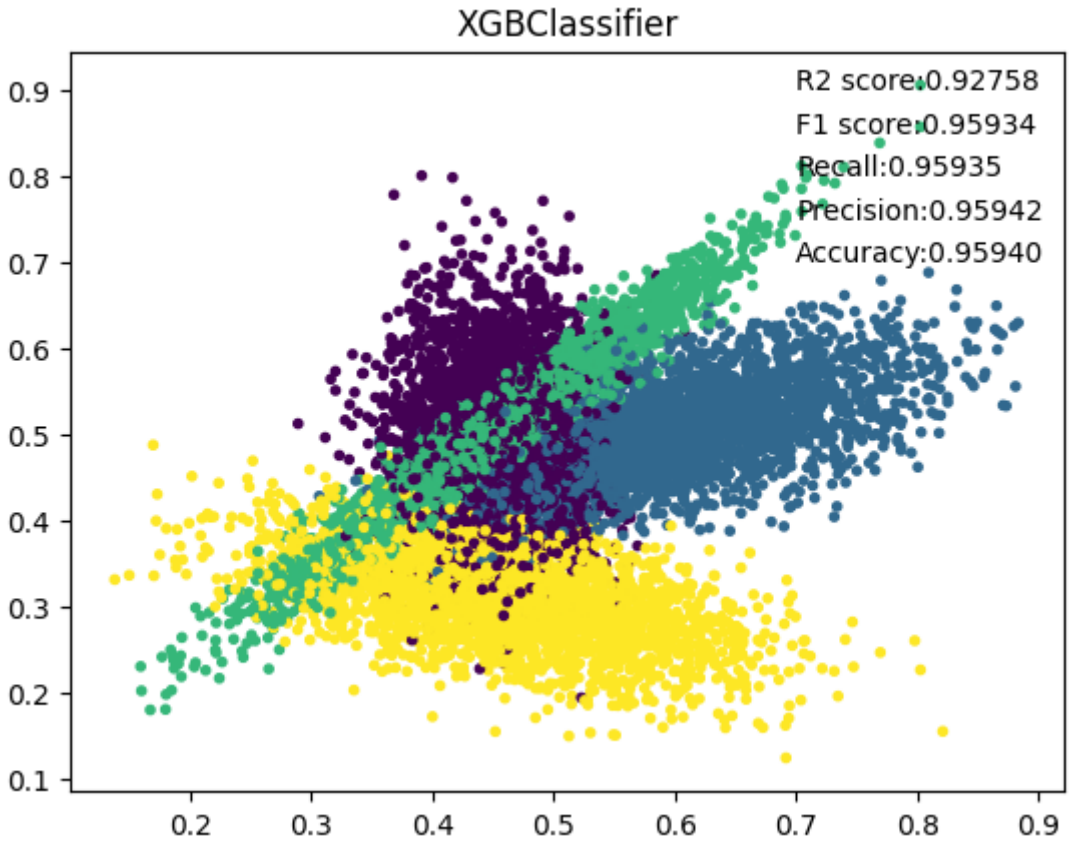
In [ ]:
```python
#用梯度提升算法进行分类
from sklearn.ensemble import GradientBoostingClassifier
gbc = GradientBoostingClassifier()
gbc.fit(x_train,y_train)
y_predict = gbc.predict(x_test)

accurate('GradientBoostingClassifier')
```



In [ ]:
```
#用LightGBM算法进行分类
from lightgbm import LGBMClassifier
lgbmc = LGBMClassifier()
lgbmc.fit(x_train,y_train)
y_predict = lgbmc.predict(x_test)

accurate('LGBMClassifier')
```

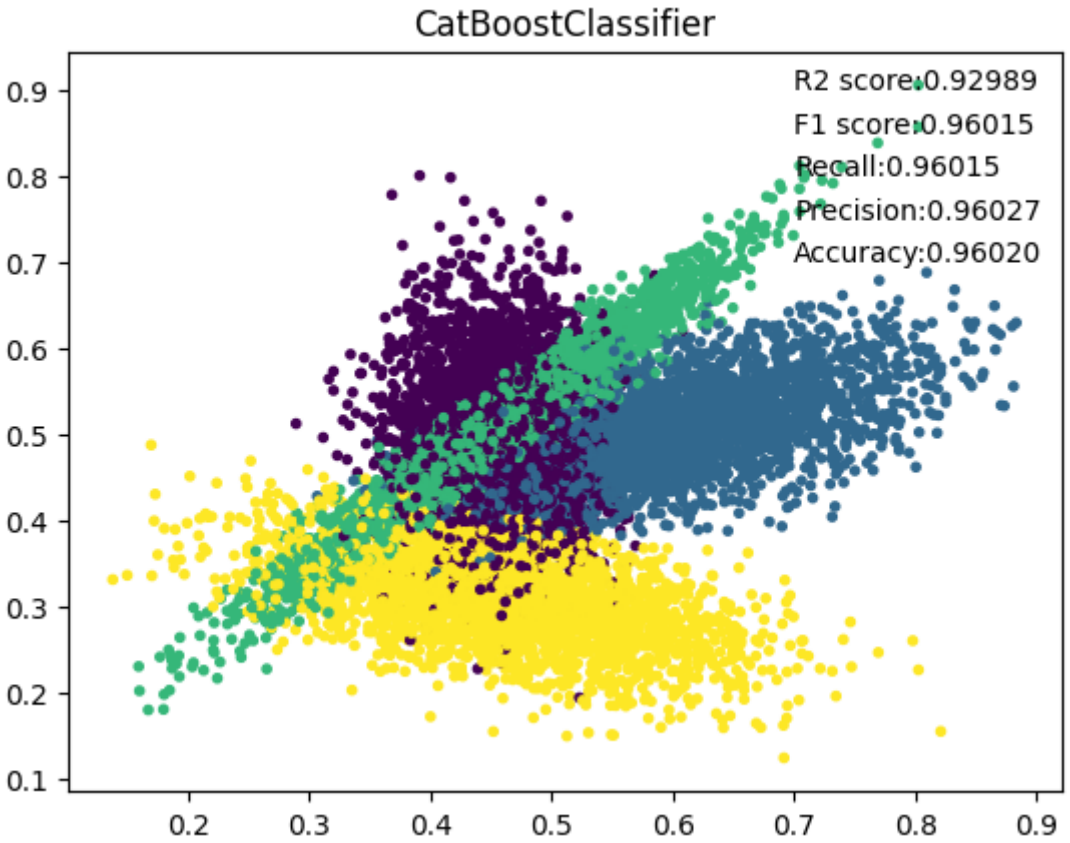


In [ ]:
```
#用XGBoost算法进行分类
from xgboost import XGBClassifier
xgbc = XGBClassifier()
xgbc.fit(x_train,y_train)
y_predict = xgbc.predict(x_test)

accurate('XGBClassifier')
```

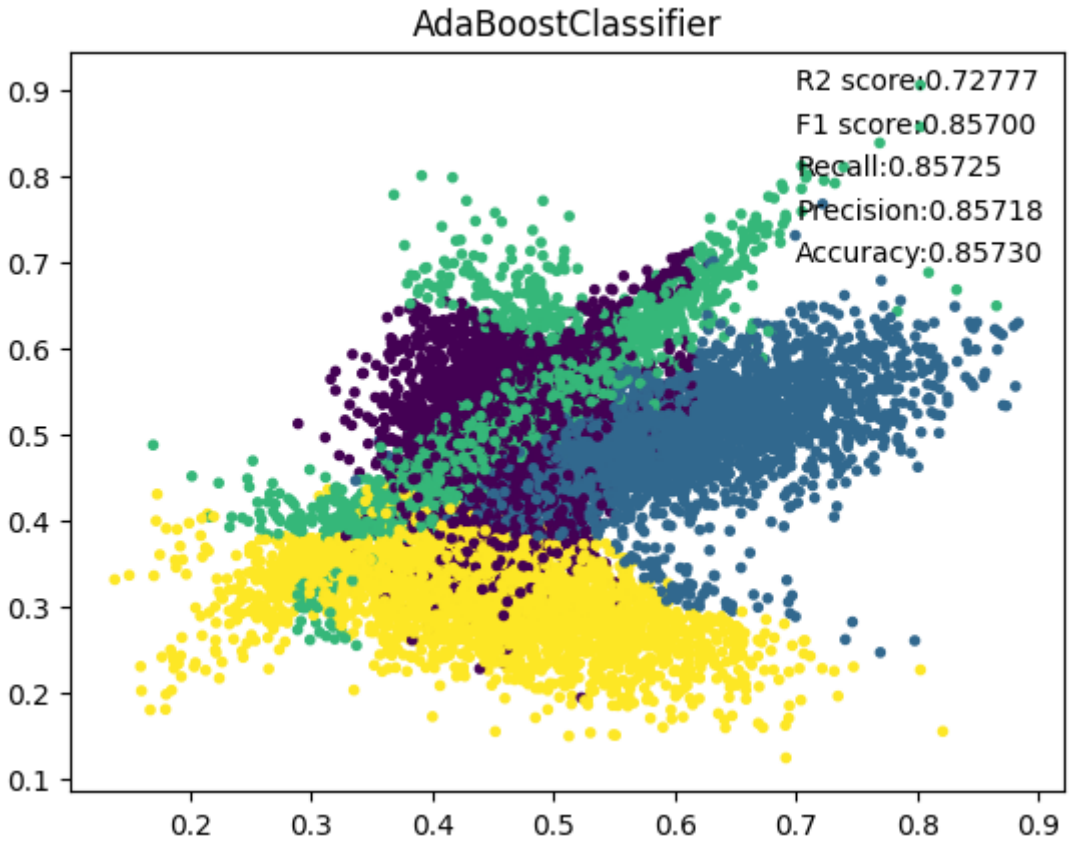


In [ ]:
```
#用CatBoost算法进行分类
from catboost import CatBoostClassifier
cbc = CatBoostClassifier()
cbc.fit(x_train,y_train)
y_predict = cbc.predict(x_test)
```

In [ ]:
```
accurate('CatBoostClassifier')
```

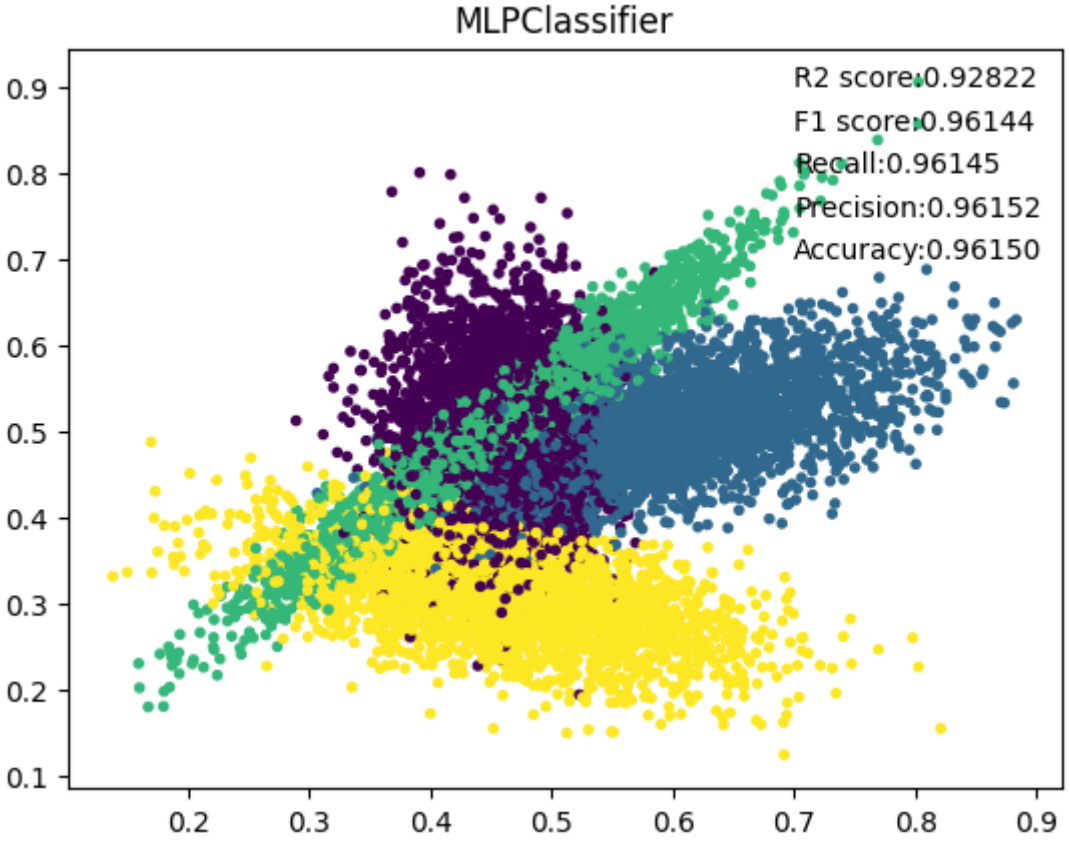


In [ ]:
```
#用AdaBoost算法进行分类
from sklearn.ensemble import AdaBoostClassifier
abc = AdaBoostClassifier()
abc.fit(x_train,y_train)
y_predict = abc.predict(x_test)

accurate('AdaBoostClassifier')
```



In [ ]:
```
#用神经网络算法进行分类
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier()
mlp.fit(x_train,y_train)
y_predict = mlp.predict(x_test)

accurate('MLPClassifier')
```

```
d:\Software\Python\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
```


MLPClassifier
R2 score:0.92822
F1 score:0.96144
Recall:0.96145
Precision:0.96152
Accuracy:0.96150
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9