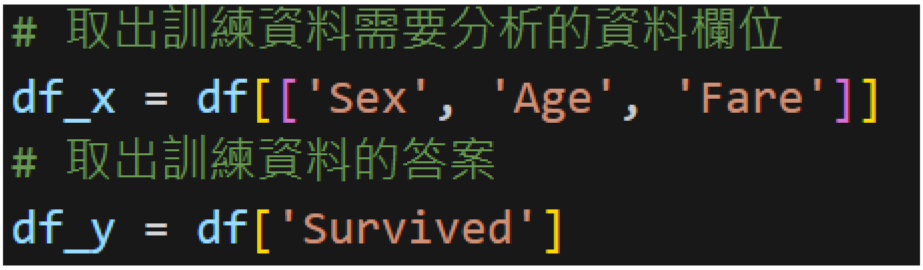
Report

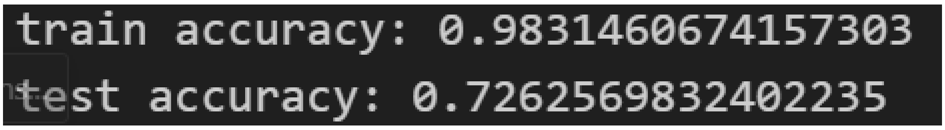
- The way to improve:

1. 增加輸入特徵

   原本:

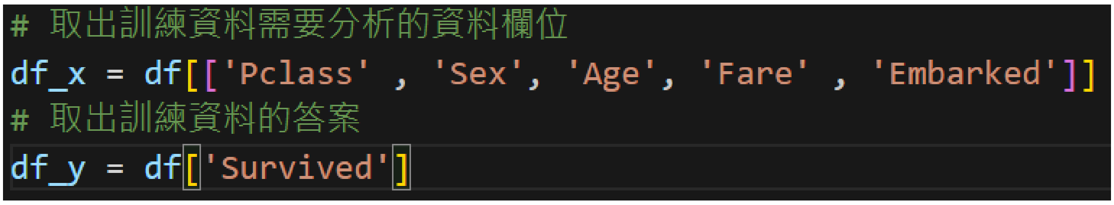


```
# 取出訓練資料需要分析的資料欄位
df_x = df[['Sex', 'Age', 'Fare']]
# 取出訓練資料的答案
df_y = df['Survived']
```

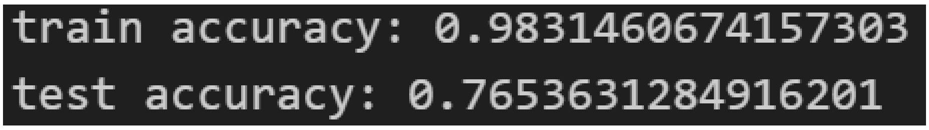


```
train accuracy: 0.9831460674157303
test accuracy: 0.7262569832402235
```

   增加 Pclass 與 Embarked(但 Embarked 需跟 Sex 一樣先做 LabelEncoder)



```
# 取出訓練資料需要分析的資料欄位
df_x = df[['Pclass' , 'Sex', 'Age', 'Fare' , 'Embarked']]
# 取出訓練資料的答案
df_y = df['Survived']
```

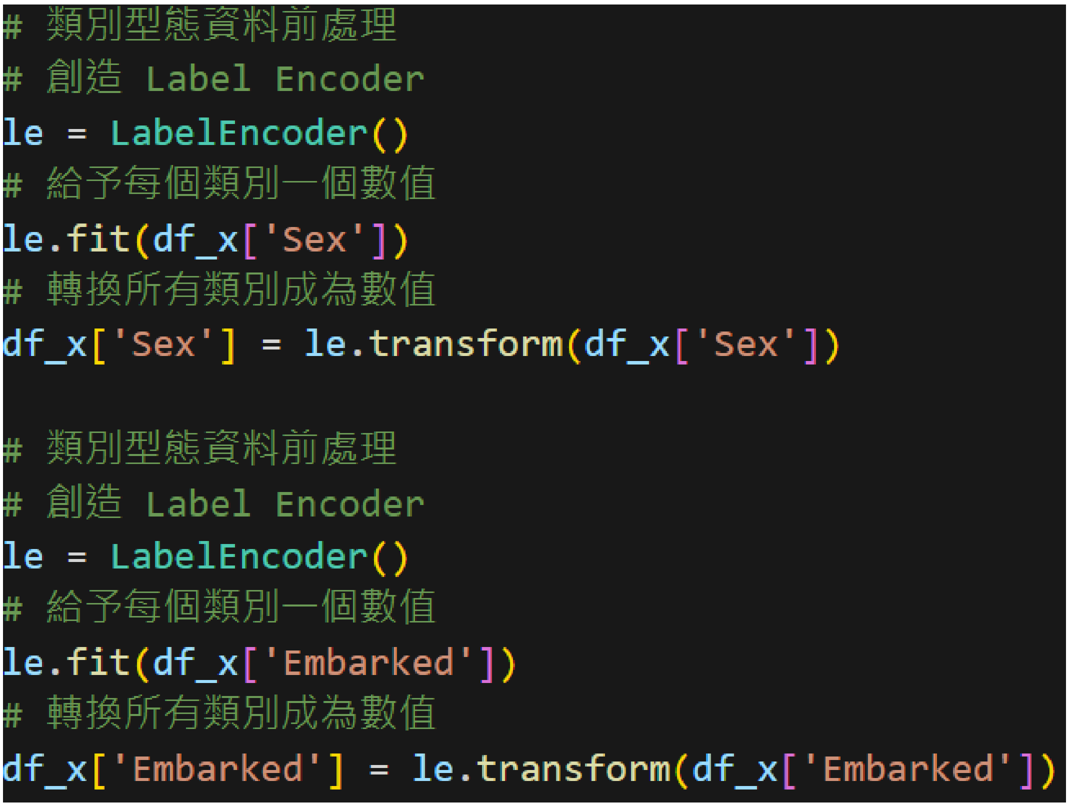


```
train accuracy: 0.9831460674157303
test accuracy: 0.7653631284916201
```

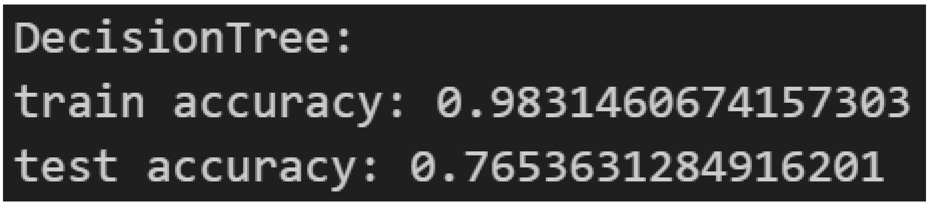
2. 使用不同的前處理方法

   此處增加將 Embarked 進行 LabelEncoder，因為 Embarked 與 Sex 一樣是 string，而模型中是利用數學計算，string 餵進去會爆掉，因此做與 Sex 一樣的動作，比較不一樣的是我新增兩個欄位，分別是 sex 與 embarked，讓新增的欄位的 type 不為 object，此做法是因為第二題會用到 XGBClassifier，XGBClassifier 的訓練資料不可為 object，因此事先轉換，再將原本的那兩個欄位刪掉，且此作法不影響 DecisionTree 的表現

   原本:

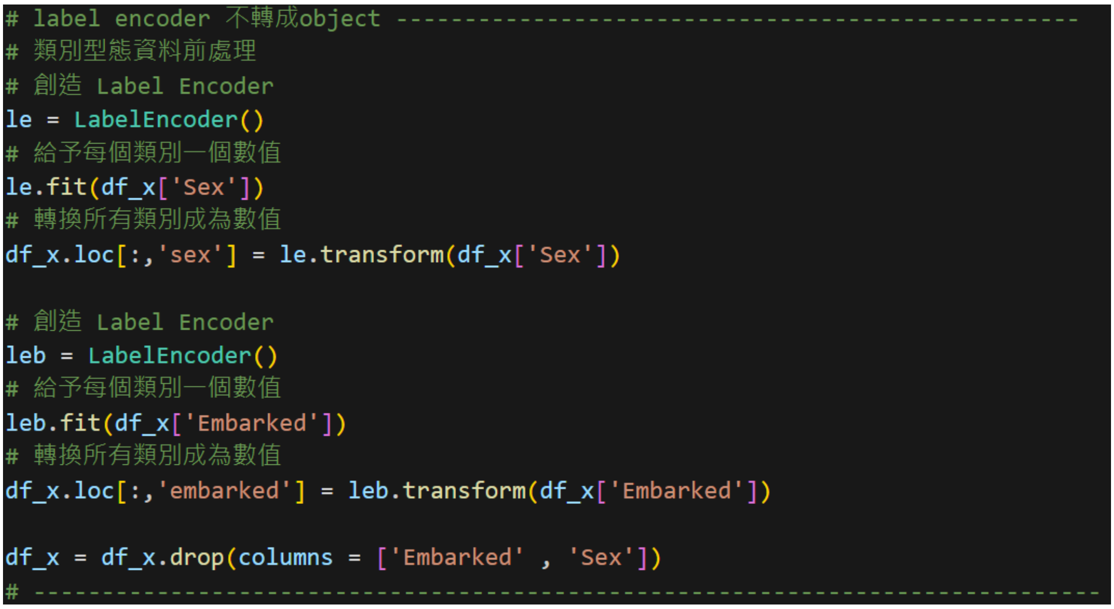


```
# 類別型態資料前處理
# 創造 Label Encoder
le = LabelEncoder()
# 給予每個類別一個數值
le.fit(df_x['Sex'])
# 轉換所有類別成為數值
df_x['Sex'] = le.transform(df_x['Sex'])

# 類別型態資料前處理
# 創造 Label Encoder
le = LabelEncoder()
# 給予每個類別一個數值
le.fit(df_x['Embarked'])
# 轉換所有類別成為數值
df_x['Embarked'] = le.transform(df_x['Embarked'])
```

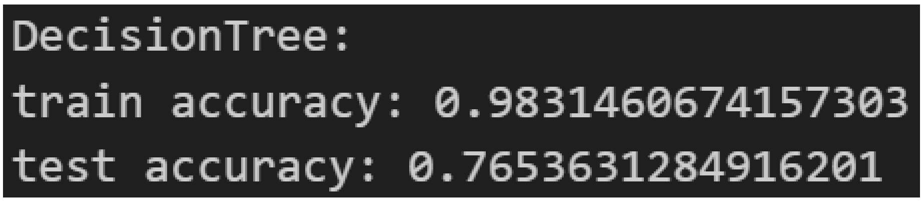


```
DecisionTree:
train accuracy: 0.9831460674157303
test accuracy: 0.7653631284916201
```

更改後:

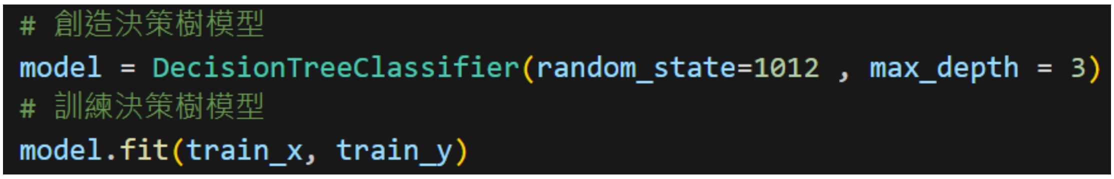


```
# label encoder 不轉成object ------------------------------------------------
# 類別型態資料前處理
# 創造 Label Encoder
le = LabelEncoder()
# 給予每個類別一個數值
le.fit(df_x['Sex'])
# 轉換所有類別成為數值
df_x.loc[:,'sex'] = le.transform(df_x['Sex'])

# 創造 Label Encoder
leb = LabelEncoder()
# 給予每個類別一個數值
leb.fit(df_x['Embarked'])
# 轉換所有類別成為數值
df_x.loc[:,'embarked'] = leb.transform(df_x['Embarked'])

df_x = df_x.drop(columns = ['Embarked' , 'Sex'])
# ------------------------------------------------------------------------------
```

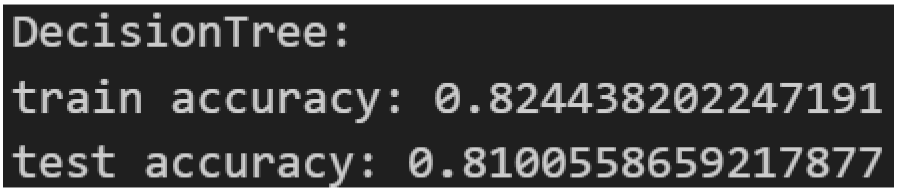


```
DecisionTree:
train accuracy: 0.9831460674157303
test accuracy: 0.7653631284916201
```

3. 調整超參數



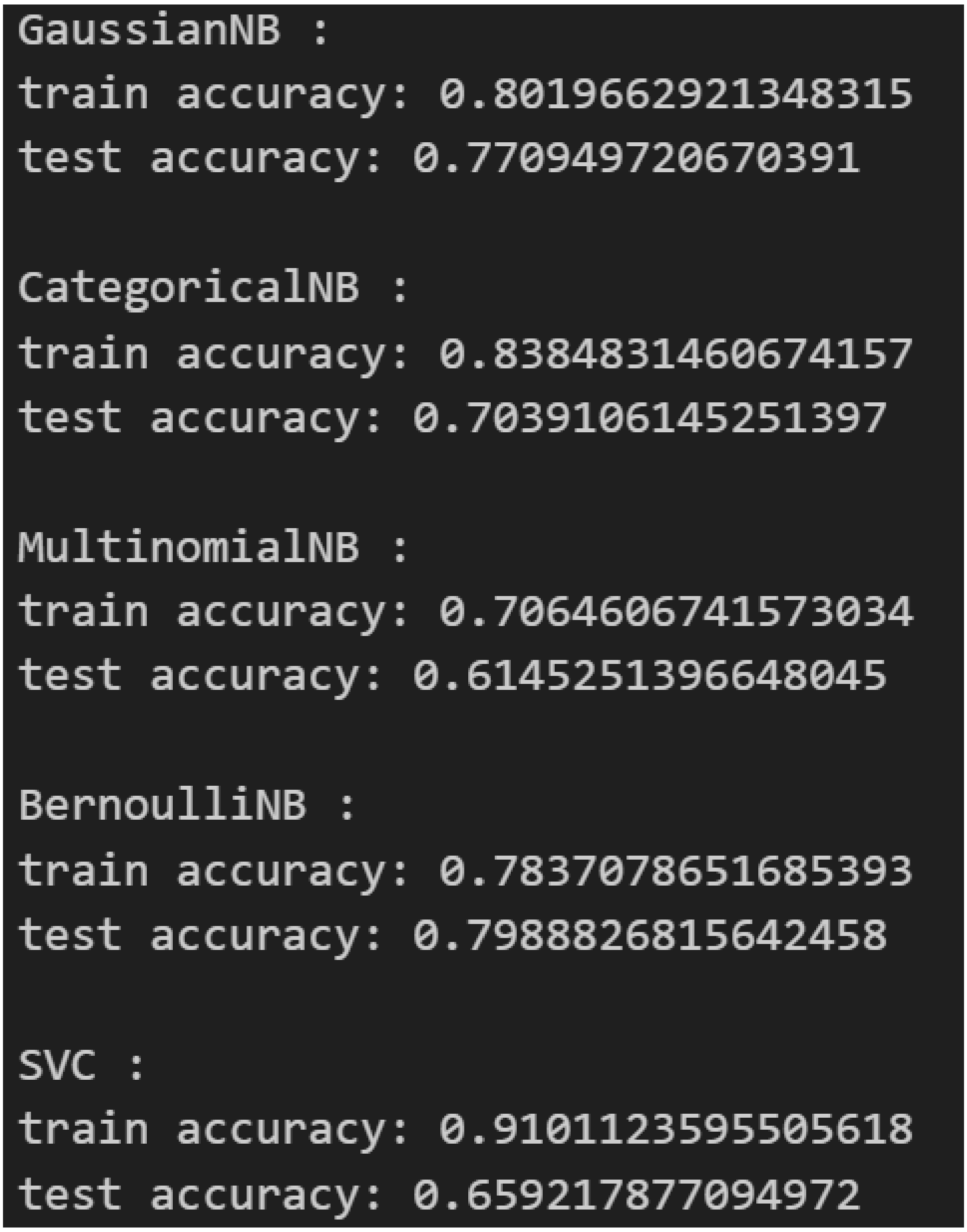
```
# 創造決策樹模型
model = DecisionTreeClassifier(random_state=1012 , max_depth = 3)
# 訓練決策樹模型
model.fit(train_x, train_y)
```



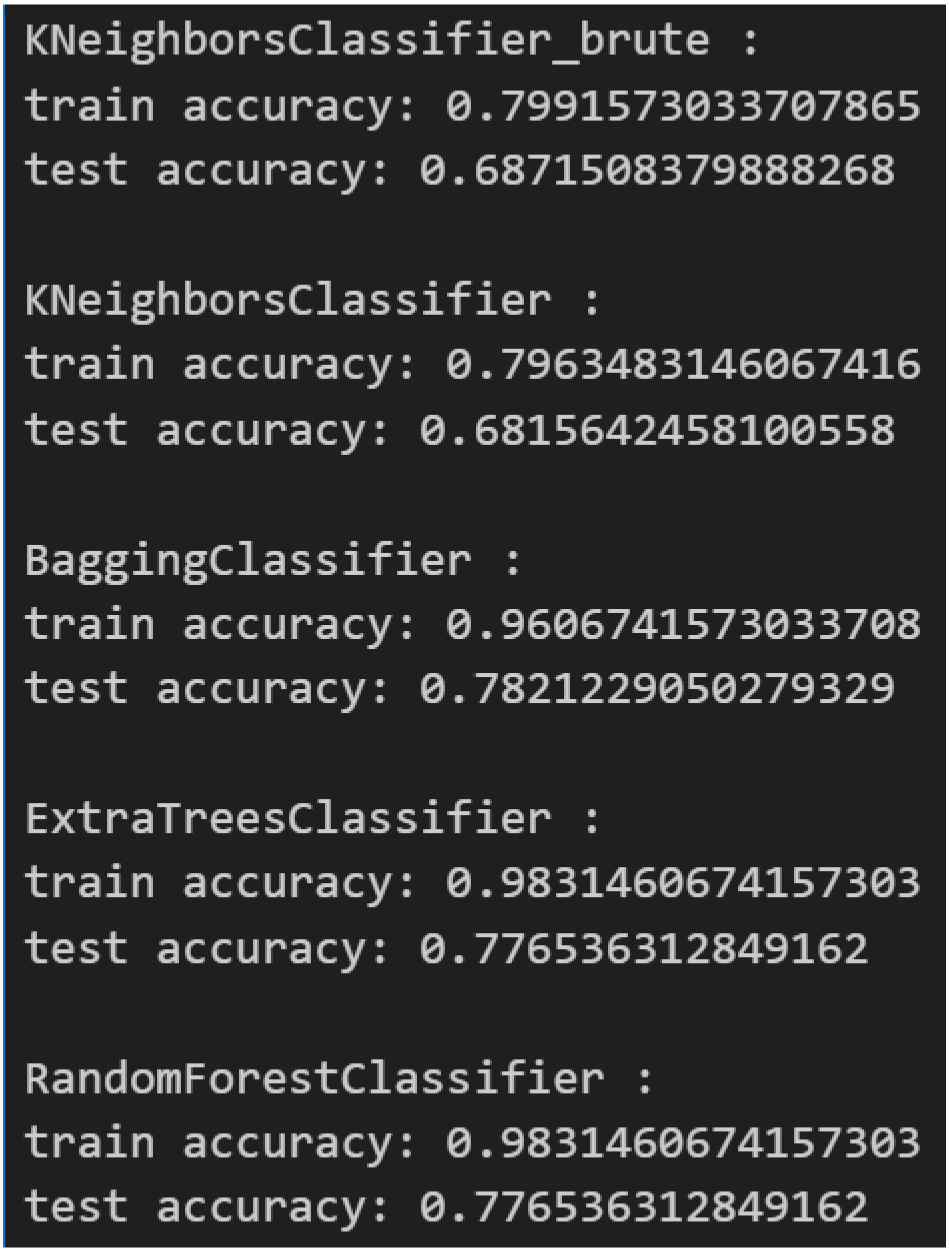
```
DecisionTree:
train accuracy: 0.8244382022471910
test accuracy: 0.8100558659217877
```

更改的地方為設定樹深，避免他 overfitting(可從上面的看出 train accuracy 從 0.98 掉到 0.82，但 test accuracy 卻從 0.76 升到 0.81)

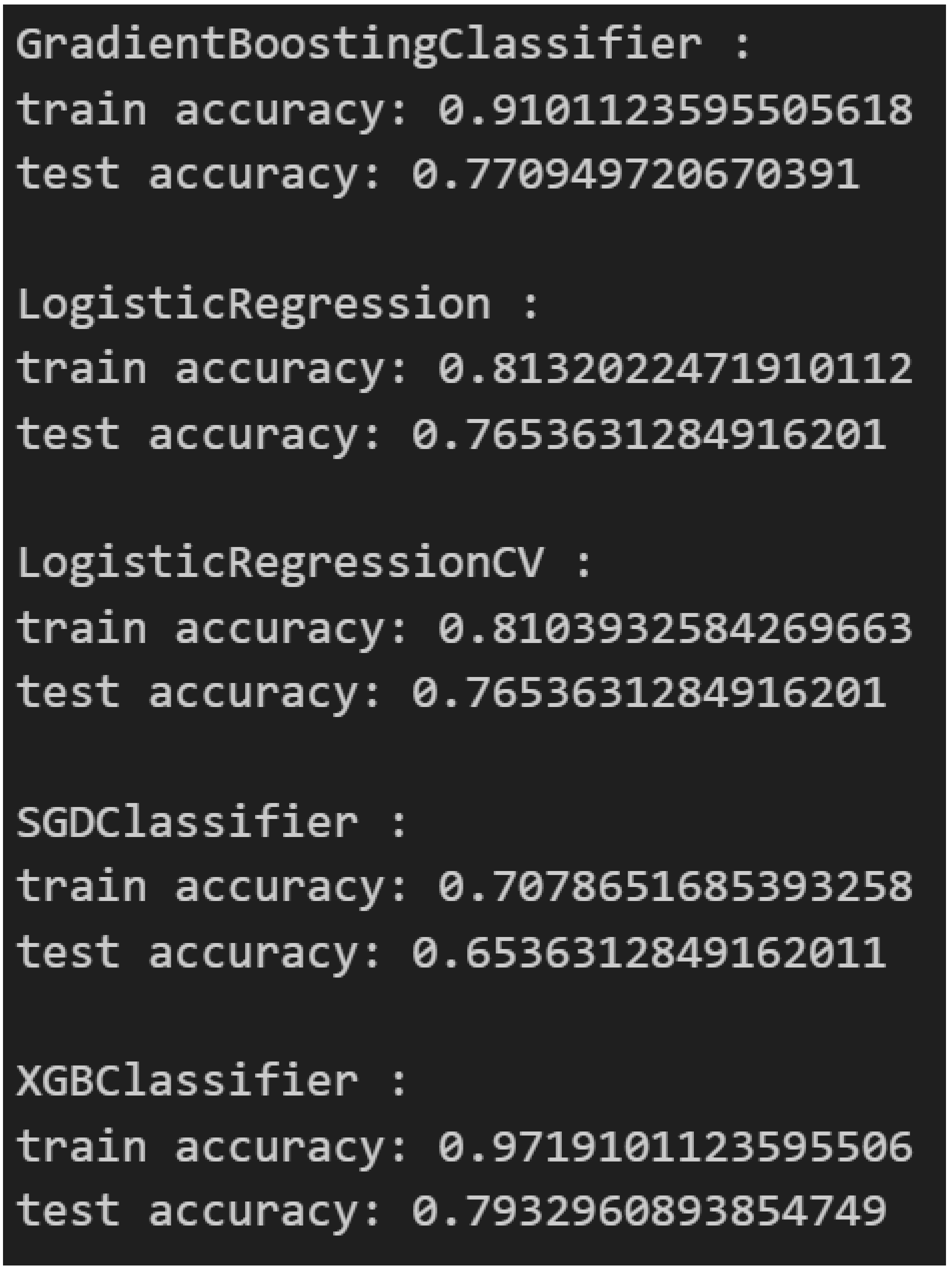
- Different model comparison:



```
GaussianNB :
train accuracy: 0.8019662921348315
test accuracy: 0.770949720670391

CategoricalNB :
train accuracy: 0.8384831460674157
test accuracy: 0.7039106145251397

MultinomialNB :
train accuracy: 0.7064606741573034
test accuracy: 0.6145251396648045

BernoulliNB :
train accuracy: 0.7837078651685393
test accuracy: 0.7988826815642458

SVC :
train accuracy: 0.9101123595505618
test accuracy: 0.659217877094972
```



```
KNeighborsClassifier_brute :
train accuracy: 0.7991573033707865
test accuracy: 0.6871508379888268

KNeighborsClassifier :
train accuracy: 0.7963483146067416
test accuracy: 0.6815642458100558

BaggingClassifier :
train accuracy: 0.9606741573033708
test accuracy: 0.7821229050279329

ExtraTreesClassifier :
train accuracy: 0.9831460674157303
test accuracy: 0.776536312849162

RandomForestClassifier :
train accuracy: 0.9831460674157303
test accuracy: 0.776536312849162
```



```
GradientBoostingClassifier :
train accuracy: 0.9101123595505618
test accuracy: 0.770949720670391

LogisticRegression :
train accuracy: 0.8132022471910112
test accuracy: 0.7653631284916201

LogisticRegressionCV :
train accuracy: 0.8103932584269663
test accuracy: 0.7653631284916201

SGDClassifier :
train accuracy: 0.7078651685393258
test accuracy: 0.6536312849162011

XGBClassifier :
train accuracy: 0.9719101123595506
test accuracy: 0.7932960893854749
```