随机森林建模

简介

随机森林是由一堆决策树组成的，每一个决策树有一个结果，看有多少个决策树对同一个`Y`进行投票来确定`Y`。**分类就是少数服从多数，回归就是各个决策树取平均值。**随机森林是平均多个深决策树以降低方差的一种方法，其中，决策树是在一个数据集上的不同部分进行训练的。这是以偏差的小幅增加和一些可解释性的丧失为代价的，但是在最终的模型中通常会大大提高性能。

建模过程

1. 用有抽样放回的方法（`bootstrap`）从样本集中选取n个样本作为一个训练集
2. 用抽样得到的样本集生成一棵决策树。在生成的每一个结点随机不重复地选择`d`个特征，利用这`d`个特征分别对样本集进行划分，找到最佳的划分特征（可用基尼系数、增益率或者信息增益判别）
3. 重复步骤 1 到步骤 2 共 k 次，`k`即为随机森林中决策树的个数
4. 用训练得到的随机森林对测试样本进行预测，并用票选法决定预测的结果

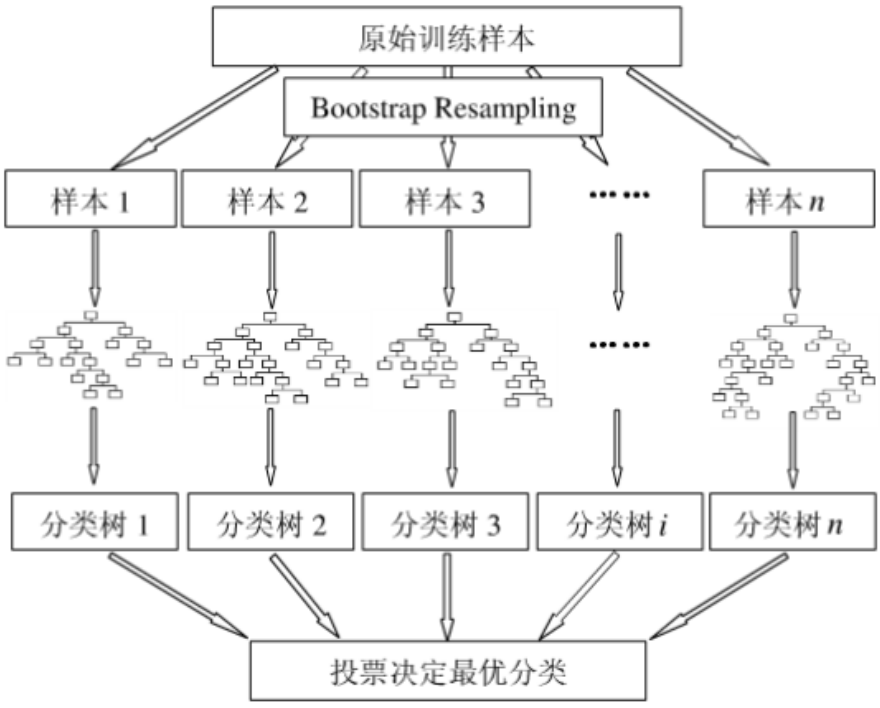


图1：随机森林算法示意图

实例分析

```python
# 随机森林
from sklearn.ensemble import RandomForestClassifier

classifier_rf = RandomForestClassifier(n_estimators=10, max_depth=8, criterion='entropy', random_state=42)
classifier_rf.fit(X_train, y_train)
# Predicting the Test set results
y_pred_rf = classifier_rf.predict(X_test)
print(pd.crosstab(y_test, y_pred_rf, rownames=['Actual Class'], colnames=['Predicted Class']))
print(accuracy_score(y_test, y_pred_rf))
print(recall_score(y_test, y_pred_rf))
print(f1_score(y_test, y_pred_rf))
print(sklearn.metrics.roc_auc_score(y_test, y_pred_rf))
```

这是银行客户流失数据的随机森林，使用了随机森林分类算法，其所有基评估器都是决策树

`n_estimators`:

这是森林中树木的数量，即基评估器的数量。这个参数对随机森林模型的精确性影响是单调的，`n_estimators`**越大，模型的效果往往越好。**但是相应的，任何模型都有决策边界，`n_estimators`达到一定的程度之后，随机森林的精确性往往不再上升或开始波动，并且，`n_estimators`越大，需要的计算量和内存也越大，训练的时间也会越来越长。对于这个参数，我们是渴望在训练难度和模型效果之间取得平衡。

`max_depth`:

树的最大深度，超过最大深度的树枝都会被剪掉

`criterion`:

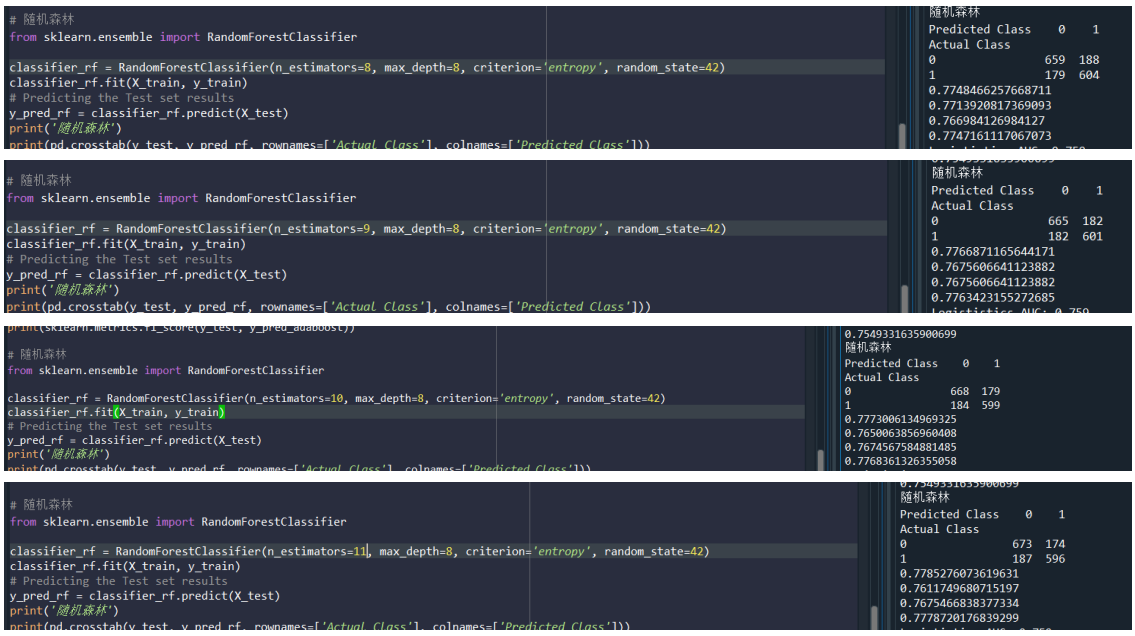
不纯度的衡量指标，有基尼系数和信息熵两种选择

`random_state`:

对于随机森林这个模型，它本质上是随机的，设置不同的随机状态（或者不设置`random_state`参数）可以彻底改变构建的模型，固定`random_state`后，每次构建的模型是相同的、生成的数据集是相同的、每次的拆分结果也是相同的

改变参数分析性能变化

1. 改变`n_estimators`，观察`accuracy_score`变化



```
# 随机森林
from sklearn.ensemble import RandomForestClassifier

classifier_rf = RandomForestClassifier(n_estimators=12, max_depth=8, criterion='entropy', random_state=42)
classifier_rf.fit(X_train, y_train)
# Predicting the Test set results
y_pred_rf = classifier_rf.predict(X_test)
print('随机森林')
print(pd.crosstab(y_test, y_pred_rf, rownames=['Actual Class'], colnames=['Predicted Class']))
```

```
0.7549331635900699
随机森林
Predicted Class    0    1
Actual Class
0                673  174
1                189  594
0.7773006134969325
0.7586206896551724
0.7659574468085106
0.7765948784757563
```

```
# 随机森林
from sklearn.ensemble import RandomForestClassifier

classifier_rf = RandomForestClassifier(n_estimators=13, max_depth=8, criterion='entropy', random_state=42)
classifier_rf.fit(X_train, y_train)
# Predicting the Test set results
y_pred_rf = classifier_rf.predict(X_test)
print('随机森林')
print(pd.crosstab(y_test, y_pred_rf, rownames=['Actual Class'], colnames=['Predicted Class']))
```

```
随机森林
Predicted Class    0    1
Actual Class
0                675  172
1                191  592
0.7773006134969325
0.756066411238825
0.7653522947640594
0.7764983768118564
```

可以发现，随着`n_estimators`从`8`增加到`13`，`accuracy_score`先增大，之后就基本不变了，说明`n_estimators`增大，一定范围内可以改善`accuracy_score`，达到一定程度后`accuracy_score`就基本不变了，达到了决策边界

2. 改变`max_depth`，观察`accuracy_score`变化

```
# 随机森林
from sklearn.ensemble import RandomForestClassifier

classifier_rf = RandomForestClassifier(n_estimators=10, max_depth=6, criterion='entropy', random_state=42)
classifier_rf.fit(X_train, y_train)
# Predicting the Test set results
y_pred_rf = classifier_rf.predict(X_test)
print('随机森林')
print(pd.crosstab(y_test, y_pred_rf, rownames=['Actual Class'], colnames=['Predicted Class']))
```

```
随机森林
Predicted Class    0    1
Actual Class
0                663  184
1                228  555
0.747239263803681
0.7088122605363985
0.7293035479632062
0.7457874761949997
```

```
# 随机森林
from sklearn.ensemble import RandomForestClassifier

classifier_rf = RandomForestClassifier(n_estimators=10, max_depth=7, criterion='entropy', random_state=42)
classifier_rf.fit(X_train, y_train)
# Predicting the Test set results
y_pred_rf = classifier_rf.predict(X_test)
print('随机森林')
```

```
随机森林
Predicted Class    0    1
Actual Class
0                689  158
1                218  565
0.7693251533742331
0.7215836526181354
0.7503320053120851
0.767521460311429
```

```
# 随机森林
from sklearn.ensemble import RandomForestClassifier

classifier_rf = RandomForestClassifier(n_estimators=10, max_depth=8, criterion='entropy', random_state=42)
classifier_rf.fit(X_train, y_train)
# Predicting the Test set results
y_pred_rf = classifier_rf.predict(X_test)
print('随机森林')
print(pd.crosstab(y_test, y_pred_rf, rownames=['Actual Class'], colnames=['Predicted Class']))
print(accuracy_score(y_test, y_pred_rf))
```

```
随机森林
Predicted Class    0    1
Actual Class
0                668  179
1                184  599
0.7773006134969325
0.7650063856960408
0.7674567584881485
0.7768361326355058
Logististics AUC: 0.759
```

```
# 随机森林
from sklearn.ensemble import RandomForestClassifier

classifier_rf = RandomForestClassifier(n_estimators=10, max_depth=9, criterion='entropy', random_state=42)
classifier_rf.fit(X_train, y_train)
# Predicting the Test set results
y_pred_rf = classifier_rf.predict(X_test)
print('随机森林')
print(pd.crosstab(y_test, y_pred_rf, rownames=['Actual Class'], colnames=['Predicted Class']))
```

```
随机森林
Predicted Class    0    1
Actual Class
0                678  169
1                189  594
0.7803680981595092
0.7586206896551724
0.7684346701164295
0.779546472336441
Logististics AUC: 0.759
```

```
# 随机森林
from sklearn.ensemble import RandomForestClassifier

classifier_rf = RandomForestClassifier(n_estimators=10, max_depth=10, criterion='entropy', random_state=42)
classifier_rf.fit(X_train, y_train)
# Predicting the Test set results
y_pred_rf = classifier_rf.predict(X_test)
print('随机森林')
print(pd.crosstab(y_test, y_pred_rf, rownames=['Actual Class'], colnames=['Predicted Class']))
```

```
随机森林
Predicted Class    0    1
Actual Class
0                670  177
1                177  606
0.7828220858895706
0.7739463601532567
0.7739463601532567
0.7824867574083876
```

```
# 随机森林
from sklearn.ensemble import RandomForestClassifier

classifier_rf = RandomForestClassifier(n_estimators=10, max_depth=11, criterion='entropy', random_state=42)
classifier_rf.fit(X_train, y_train)
# Predicting the Test set results
y_pred_rf = classifier_rf.predict(X_test)
print('随机森林')
print(pd.crosstab(y_test, y_pred_rf, rownames=['Actual Class'], colnames=['Predicted Class']))
```

```
随机森林
Predicted Class    0    1
Actual Class
0                685  162
1                166  617
0.7987730061349694
0.7879948914431673
0.7900128040973112
0.7983658046353971
Logististics AUC: 0.759
```

可以看出，随着`max_depth`从`6`增加到`11`，`accuracy_score`一直在增加，说明`max_depth`增大，`accuracy_score`也在变大

3. 改变`criterion`，观察`accuracy_score`变化

```
# 随机森林
from sklearn.ensemble import RandomForestClassifier

classifier_rf = RandomForestClassifier(n_estimators=10, max_depth=8, criterion='entropy', random_state=42)
classifier_rf.fit(X_train, y_train)
# Predicting the Test set results
y_pred_rf = classifier_rf.predict(X_test)
print('随机森林')
print(pd.crosstab(y_test, y_pred_rf, rownames=['Actual Class'], colnames=['Predicted Class']))
print(accuracy_score(y_test, y_pred_rf))
```

```
随机森林
Predicted Class    0    1
Actual Class
0                668  179
1                184  599
0.7773006134969325
0.7650063856960408
0.7674567584881485
0.7768361326355058
Logististics AUC: 0.759
```

```
# 随机森林
from sklearn.ensemble import RandomForestClassifier

classifier_rf = RandomForestClassifier(n_estimators=10, max_depth=8, criterion='gini', random_state=42)
classifier_rf.fit(X_train, y_train)
# Predicting the Test set results
y_pred_rf = classifier_rf.predict(X_test)
print('随机森林')
print(pd.crosstab(y_test, y_pred_rf, rownames=['Actual Class'], colnames=['Predicted Class']))
```

```
随机森林
Predicted Class    0    1
Actual Class
0                679  168
1                190  593
0.7803680981595092
0.7573435504469987
0.7681347150259068
0.7794982215044911
```

```python
# 随机森林
from sklearn.ensemble import RandomForestClassifier

classifier_rf = RandomForestClassifier(n_estimators=10, max_depth=9, criterion='entropy', random_state=42)
classifier_rf.fit(X_train, y_train)
# Predicting the Test set results
y_pred_rf = classifier_rf.predict(X_test)
print('随机森林')
print(pd.crosstab(y_test, y_pred_rf, rownames=['Actual Class'], colnames=['Predicted Class']))
print(accuracy_score(y_test, y_pred_rf))
```

```
随机森林
Predicted Class    0    1
Actual Class
0                678  169
1                189  594
0.7803680981595092
0.7586206896551724
0.7684346701164295
0.779546472336441
Logistics AUC: 0.759
```

```python
# 随机森林
from sklearn.ensemble import RandomForestClassifier

classifier_rf = RandomForestClassifier(n_estimators=10, max_depth=9, criterion='gini', random_state=42)
classifier_rf.fit(X_train, y_train)
# Predicting the Test set results
y_pred_rf = classifier_rf.predict(X_test)
print('随机森林')
print(pd.crosstab(y_test, y_pred_rf, rownames=['Actual Class'], colnames=['Predicted Class']))
print(accuracy_score(y_test, y_pred_rf))
print(recall_score(y_test, y_pred_rf))
```

```
随机森林
Predicted Class    0    1
Actual Class
0                669  178
1                176  607
0.7828220858895706
0.7752234993614304
0.7742346938775511
0.782535082403374
Logistics AUC: 0.759
SVM AUC: 0.695
```

```python
# 随机森林
from sklearn.ensemble import RandomForestClassifier

classifier_rf = RandomForestClassifier(n_estimators=11, max_depth=8, criterion='entropy', random_state=42)
classifier_rf.fit(X_train, y_train)
# Predicting the Test set results
y_pred_rf = classifier_rf.predict(X_test)
print('随机森林')
print(pd.crosstab(y_test, y_pred_rf, rownames=['Actual Class'], colnames=['Predicted Class']))
print(accuracy_score(y_test, y_pred_rf))
```

```
随机森林
Predicted Class    0    1
Actual Class
0                673  174
1                187  596
0.7785276073619631
0.7611749680715197
0.7675466838377334
0.7778720176839299
Logistics AUC: 0.759
```

```python
# 随机森林
from sklearn.ensemble import RandomForestClassifier

classifier_rf = RandomForestClassifier(n_estimators=11, max_depth=8, criterion='gini', random_state=42)
classifier_rf.fit(X_train, y_train)
# Predicting the Test set results
y_pred_rf = classifier_rf.predict(X_test)
print('随机森林')
print(pd.crosstab(y_test, y_pred_rf, rownames=['Actual Class'], colnames=['Predicted Class']))
print(accuracy_score(y_test, y_pred_rf))
print(recall_score(y_test, y_pred_rf))
```

```
随机森林
Predicted Class    0    1
Actual Class
0                682  165
1                187  596
0.7840490797546013
0.7611749680715197
0.772020725388601
0.7831848866331625
Logistics AUC: 0.759
SVM AUC: 0.695
```

通过改变`criterion`，分别改变`n_estimators`、`max_depth`，发现在相同条件下，`gini`比`entropy`的`accuracy_score`更大