安徽大学人工智能学院
《机器学习程序设计》
实验案例设计报告

课程名称：机器学习程序设计

专　　业：人工智能

班　　级：人工智能二班

学　　号：WA2214014

姓　　名：杨跃浙

任课老师：谭春雨

实验名称	基于分类算法的学习失败预警	实验次序	02
实验地点	笃行南楼 A104	实验日期	03.16

实验内容：

1. 业务背景分析

通过对学习者的学习过程、学习日志、学习结果进行多维分析，综合判断学生的学习情况，能够提前发现存在失败风险的学生，对其进行系统干预和人工干预。

学生失败风险需要分析学生历史学习情况，分析学生在班级中的学习情况，分析学生和标准学习过程的偏离，从横向纵向多维度进行分析。

2. 数据收集

数据量较多且种类较全，包括学生信息、课程信息、教师信息等常见的教务信息数据，除此之外还包括日常通用日志 、课程学习日志、教学设计日志、学习成绩、教学活动记录等。

对上述数据进行梳理汇总，从学生、行为、成绩三个维度统计学生的基础信息，并细化为学生基本数据、网络浏览日志数据、课程学习行为数据、形考成绩数据、论坛行为数据、网络课堂表现数据等 6 个方面的信息。

由于系统中数据量很大，这里只选择其中某一门课的学生学习和日志数据，为分析方便，将其从数据库中抽取出来之后，另存为 csv 文件，进行后续分析。

这是一个典型的二分类问题。

3. 数据预处理

数据探查及特征选择：使用 pandas 将用户相关的特征读取进来转化为 DataFrame 对象，并将样本数量、字段数量、数据格式、非空值数量等详细信息进行显示。

代码：

```
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
import joblib
import matplotlib.pyplot as plt
from sklearn.utils import resample
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import
train_test_split,GridSearchCV,StratifiedKFold
from sklearn.metrics import roc_curve, precision_recall_curve, auc, make_scorer,
recall_score, accuracy_score, precision_score, confusion_matrix
from scipy.stats import binned_statistic
import itertools
```

```
import os
os.environ['NLS_LANG']='SMPLFIED CHINESE_CHNA.UTF8'
pd.set_option('display.max_columns',500)
df = pd.read_csv('./Data/uwide.csv')
df.info()
```

运行结果：

preprocess.py U

Code > preprocess.py > ...

```
1   from sklearn.preprocessing import StandardScaler
2   from sklearn.ensemble import RandomForestClassifier
3   from sklearn.metrics import confusion_matrix
4   import joblib
5   import matplotlib.pyplot as plt
6   from sklearn.utils import resample
7   import pandas as pd
8   import numpy as np
9   from sklearn.preprocessing import LabelBinarizer
10  from sklearn.model_selection import train_test_split,G
11  from sklearn.metrics import roc_curve, precision_recal
12  from scipy.stats import binned_statistic
13  import itertools
14  import os
15  os.environ['NLS_LANG']='SMPLFIED CHINESE_CHNA.UTF8'
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COM

```
youngbean@YoungBeans-2 Project2 % /usr/local/bin/python3 "/Use
f Machine Learning /Project2/Code/preprocess.py"
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2006 entries, 0 to 2005
Data columns (total 52 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   USERID                      2006 non-null   object
 1   BROWSER_COUNT               1950 non-null   float64
 2   COURSE_COUNT                2006 non-null   int64
 3   COURSE_LAST_ACCESS          1950 non-null   float64
 4   COURSE_SUM_VIEW             1327 non-null   float64
 5   COURSE_AVG_SCORE            228 non-null    float64
 6   EXAM_AH_SCORE               2006 non-null   float64
 7   EXAM_WRITEN_SCORE           2006 non-null   float64
 8   EXAM_MIDDLE_SCORE           2006 non-null   float64
 9   EXAM_LAB                    1468 non-null   float64
 10  EXAM_PROGRESS               2006 non-null   float64
 11  EXAM_GROUP_SCORE            2006 non-null   float64
 12  EXAM_FACE_SCORE             2006 non-null   float64
 13  EXAM_ONLINE_SCORE           2006 non-null   float64
 14  NODEBB_LAST_POST            750 non-null    float64
 15  NODEBB_CHANNEL_COUNT        750 non-null    float64
 16  NODEBB_TOPIC_COUNT          750 non-null    float64
 17  COURSE_SUM_VIDEO_LEN        2005 non-null   float64
 18  SEX                         2006 non-null   object
```

为了方便对前述各项指标进行质量检查，包括对数据的分布情况、数据类型、最大值、最小值、均值、标准差等，另外重点查看变量的有效记录数的比例、完整百分比、有效记录数等，作为字段筛选的依据。

将性别等中文字段进行因子化（Factorize）处理为数字型变量。

查看样本中的空值情况。

对所有特征值为空的样本以 0 填充。

```
df.describe()
factor = pd.factorize(df['SEX'])
df.SEX = factor[0]
null_columns=df.columns[df.isnull().any()]
print(df[df.isnull().any(axis=1)][null_columns].head())
df = df.fillna(0)
print(df.head())
```

运行结果：

空值情况：

```
   BROWSER_COUNT  COURSE_LAST_ACCESS  COURSE_SUM_VIEW  COURSE_AVG_SCORE  \
0          334.0        1.513620e+09              NaN               0.0
2           17.0        1.505817e+09              NaN               NaN
3          154.0        1.513881e+09         0.021609               NaN
4           50.0        1.514159e+09              NaN               NaN
5          138.0        1.514115e+09         0.000174               NaN

     EXAM_LAB  NODEBB_LAST_POST  NODEBB_CHANNEL_COUNT  NODEBB_TOPIC_COUNT  \
0  30.583333      1.510339e+09                   2.0                 2.0
2         NaN               NaN                   NaN                 NaN
3         NaN               NaN                   NaN                 NaN
4         NaN               NaN                   NaN                 NaN
5         NaN               NaN                   NaN                 NaN

   COURSE_SUM_VIDEO_LEN  NODEBB_PARTICIPATIONRATE  COURSE_WORKTIME  \
0                   0.0                      9.29      5430.428571
2                   0.0                      0.00         0.000000
3                1867.0                      0.00     28253.000000
4                   0.0                      0.00     12071.000000
5                   1.0                      0.00      3471.333333

   COURSE_WORKACCURACYRATE  COURSE_WORKCOMPLETERATE  NODEBB_POSTSCOUNT  \
0                67.528571                94.285714           4.428571
2                38.642857                85.714286           0.000000
3                84.140000               100.000000           0.000000
4                57.200000               100.000000           0.000000
5                89.250000               100.000000           0.000000

   NODEBB_NORMALBBSPOSTSCOUONT  NODEBB_REALBBSARCHIVECOUNT  \
0                          0.0                    1.571429
2                          0.0                    0.000000
3                          0.0                    0.000000
4                          0.0                    0.000000
5                          0.0                    0.000000

   NORMALBBSARCHIVECOUNT  COURSE_WORKCOUNT
0                    0.0          6.571429
2                    0.0          0.571429
3                    0.0          7.000000
4                    0.0          5.000000
5                    0.0          3.000000
```

填充后：

```
                 USERID  BROWSER_COUNT  COURSE_COUNT  COURSE_LAST_ACCESS  \
0  310_20178310080783          334.0            11        1.513620e+09
1  310_20178310080790          744.0           407        1.513371e+09
2  310_20128310060450           17.0             9        1.505817e+09
3  310_20128310320840          154.0           133        1.513881e+09
4  310_20128310320777           50.0             1        1.514159e+09

   COURSE_SUM_VIEW  COURSE_AVG_SCORE  EXAM_AH_SCORE  EXAM_WRITEN_SCORE  \
0         0.000000          0.000000      94.000000           2.727273
1         0.718819          0.416819      91.454545           2.727273
2         0.000000          0.000000      54.166667           6.166667
3         0.021609          0.000000      81.000000           0.000000
4         0.000000          0.000000      67.000000           0.000000

   EXAM_MIDDLE_SCORE   EXAM_LAB  EXAM_PROGRESS  EXAM_GROUP_SCORE  \
0           1.727273  30.583333            0.0               0.0
1           1.727273  21.350000            0.0               0.0
2           7.500000   0.000000            0.0               0.0
3          17.000000   0.000000           99.0               0.0
4          17.000000   0.000000            0.0               0.0

   EXAM_FACE_SCORE  EXAM_ONLINE_SCORE  NODEBB_LAST_POST  NODEBB_CHANNEL_COUNT  \
0        10.450000           6.277273      1.510339e+09                   2.0
1         9.390909           5.713636      1.522433e+09                   2.0
2        33.233333           7.333333      0.000000e+00                   0.0
3        15.000000          14.000000      0.000000e+00                   0.0
4        15.000000          14.000000      0.000000e+00                   0.0

   NODEBB_TOPIC_COUNT  COURSE_SUM_VIDEO_LEN  SEX        MAJORID STATUS  \
0                 2.0                   0.0    0  310_260000256     注册
1                 2.0                   0.0    0  310_260000256     注册
2                 0.0                   0.0    1  310_260000040     注册
3                 0.0                1867.0    0  310_260000090     注册
4                 0.0                   0.0    0  310_260000090     注册

    GRADE            CLASSID  EXAM_HOMEWORK  EXAM_LABSCORE  EXAM_OTHERSCORE  \
0  201703  310_20178310080020      52.454545      20.363636                0
1  201703  310_20178310080020      51.818182      20.000000                0
2  201203  310_20128310060088       0.000000       0.000000                0
3  201203  310_20128310320099      35.000000       0.000000                0
4  201203  310_20128310320099      21.000000       0.000000                0

   NODEBB_PARTICIPATIONRATE  COURSE_WORKTIME  COURSE_WORKACCURACYRATE  \
0                     9.290      5430.428571                67.528571
1                     6.616667      9636.833333                44.835000
2                     0.000000         0.000000                38.642857
3                     0.000000     28253.000000                84.140000
4                     0.000000     12071.000000                57.200000
```

```
   COURSE_WORKCOMPLETERATE  NODEBB_POSTSCOUNT  NODEBB_NORMALBBSPOSTSCOUONT  \
0                94.285714           4.428571                          0.0
1               100.000000           4.833333                          0.0
2                85.714286           0.000000                          0.0
3               100.000000           0.000000                          0.0
4               100.000000           0.000000                          0.0

   NODEBB_REALBBSARCHIVECOUNT  NORMALBBSARCHIVECOUNT  COURSE_WORKCOUNT  \
0                    1.571429                    0.0          6.571429
1                    1.333333                    0.0          8.833333
2                    0.000000                    0.0          0.571429
3                    0.000000                    0.0          7.000000
4                    0.000000                    0.0          5.000000

           STUNO                      ID              STUID  \
0  20178310080783  t4yzarwohqvdc3a4q-z1zq  310_20178310080783
1  20178310080790  vekcabgoj5xeaq1wxo-i-a  310_20178310080790
2  20128310060450  n-gtadyo-zncfssre8cqvq  310_20128310060450
3  20128310320840  io4sauqoarjm3nhle5p5aa  310_20128310320840
4  20128310320777  9tl7acmop59oretuqrhvzg  310_20128310320777

            COURSEOPENID  HOMEWORKSCORE  WRITTENASSIGNMENTSCORE  \
0  a0euanqnrrzbdsesy4rymw             38                       0
1  a0euanqnrrzbdsesy4rymw             38                       0
2  a0euanqnrrzbdsesy4rymw              0                       0
3  a0euanqnrrzbdsesy4rymw             35                       0
4  a0euanqnrrzbdsesy4rymw             21                       0

   MIDDLEASSIGNMENTSCORE  LABSCORE  OTHERSCORE  TOTALSCORE  STUDYTERM  \
0                     19         0           0          97     201703
1                     19         0           0          94     201703
2                     18         0           0          52     201703
3                     17         0           0          81     201703
4                     17         0           0          67     201703

        COURSEID EXAMNUM  PROCESS  GROUPSTUDYSCORE  FACESTUDYSCORE  \
0  310_260000656  2,7510        0                0            20.0
1  310_260000656  2,7510        0                0            17.0
2  310_260000656  2,7510        0                0            17.2
3  310_260000656  2,7510       99                0            15.0
4  310_260000656  2,7510        0                0            15.0

   ONLINESTUDYSCORE
0              20.0
1              20.0
2              17.0
3              14.0
4              14.0
```

生成标签列，按照当前课程的总分（TOTALSCORE）作为标准，以 60 分为界将学生划分为及格和不及格两个类别标签。对学习这一门之后成绩低于 60 分的作为失败，即标记为 1，反之标记为 0，将其存入一个列名为 SState 的字段，作为学习失败与否的标签。

```
df['Sstate']=np.where(df['TOTALSCORE']>60,0,1)
print(df.head())
```

将数据质量较差的字段移除，同时去掉 USERID（学号）、GRADE（年级）、专业编号（MAJORID）、班级编号(CLASSID)等无意义变量，这类字段无助于分类算法的改进。

运行结果：

```
   ONLINESTUDYSCORE  Sstate
0              20.0       0
1              20.0       0
2              17.0       1
3              14.0       0
4              14.0       0
```

```
cols = df.columns.tolist()
df = df[[
'BROWSER_COUNT',
'COURSE_COUNT',
'COURSE_SUM_VIEW',
'COURSE_AVG_SCORE',
'EXAM_AH_SCORE',
'EXAM_WRITEN_SCORE',
'EXAM_MIDDLE_SCORE',
'EXAM_LAB',
'EXAM_PROGRESS',
'EXAM_GROUP_SCORE',
'EXAM_FACE_SCORE',
'EXAM_ONLINE_SCORE',
# 'NODEBB_LAST_POST',
'NODEBB_CHANNEL_COUNT',
'NODEBB_TOPIC_COUNT',
'COURSE_SUM_VIDEO_LEN',
'SEX',
# 'MAJORID',
# 'STATUS',
# 'GRADE',
```

```
# 'CLASSID',
'EXAM_HOMEWORK',
'EXAM_LABSCORE',
'EXAM_OTHERSCORE',
'NODEBB_PARTICIPATIONRATE',
'COURSE_WORKTIME',
'COURSE_WORKACCURACYRATE',
'COURSE_WORKCOMPLETERATE',
'NODEBB_POSTSCOUNT',
'NODEBB_NORMALBBSPOSTSCOUONT',
'NODEBB_REALBBSARCHIVECOUNT',
'NORMALBBSARCHIVECOUNT',
'COURSE_WORKCOUNT',
# 'STUNO',
# 'ID',
# 'STUID',
# 'COURSEID',
# 'HOMEWORKSCORE',
# 'WRITTENASSIGNMENTSCORE',
# 'MIDDLEASSIGNMENTSCORE',
# 'LABSCORE',
# 'OTHRFRCTCRF',
'Sstate',
# 'STUDYTTI_M',
# 'COURSEID',
# 'LXPAPERID',
# 'PROCESS',
# 'GROUPSTUDYCOUR',
# 'FACESTUDYSCORE',
# 'ONLINESTUDYSCORE'
]]
print(df.columns.tolist())
print(len(df.columns.tolist()))
print(df.Sstate.value_counts())
```

运行结果：

```
['BROWSER_COUNT', 'COURSE_COUNT', 'COURSE_SUM_VIEW', 'COURSE_AVG_SCORE', 'EXAM_AH_SCORE', 'EXAM_WRITEN_SCORE', 'EXAM_MIDDLE_SCORE', 'EXAM_LAB', 'EXAM_PRO
GRESS', 'EXAM_GROUP_SCORE', 'EXAM_FACE_SCORE', 'EXAM_ONLINE_SCORE', 'NODEBB_CHANNEL_COUNT', 'NODEBB_TOPIC_COUNT', 'COURSE_SUM_VIDEO_LEN', 'SEX', 'EXAM_HO
MEWORK', 'EXAM_LABSCORE', 'EXAM_OTHERSCORE', 'NODEBB_PARTICIPATIONRATE', 'COURSE_WORKTIME', 'COURSE_WORKACCURACYRATE', 'COURSE_WORKCOMPLETERATE', 'NODEBB
_POSTSCOUNT', 'NODEBB_NORMALBBSPOSTSCOUONT', 'NODEBB_REALBBSARCHIVECOUNT', 'NORMALBBSARCHIVECOUNT', 'COURSE_WORKCOUNT', 'Sstate']
29
```

其中 BROWSER_COUNT(浏览次数)、COURSE_COUNT（已上课数量）、COURSE_SUM_VIEW（总浏览时间）、EXAM_AH_SCORE（形考成绩）等 28 个变量作为输入特征，SState 作为预测的目标变量

```
Sstate
0    1801
1     205
Name: count, dtype: int64
```

对两类标签的样本进行再平衡，基本原理是，当某一类别下的样本数量超过另一类别的样本量 8 倍，则对其进行降采样，将降采样之后样本与另一类进行合并（concat），组成一个新的 DataFrame 对象。

代码：

```
df_majority = df[df.Sstate == 0]
df_minority = df[df.Sstate == 1]
count_times = 8
df_majority_downsampled = df_majority
if len(df_majority) > len(df_minority) * count_times:
new_major_count = len(df_minority) * count_times
df_majority_downsampled = resample(df_majority, replace=False,
n_samples=new_major_count, random_state=123)
df = pd.concat([df_majority_downsampled, df_minority])
print(df.Sstate.value_counts())
```

运行结果：

```
Sstate
0    1640
1     205
Name: count, dtype: int64
```

将整体数据集按照 8:2 的比例随机划分为训练集和测试集两部分，训练集样本数量为 1476 条，测试集样本数量为 369 条。

代码：

```
X = df.iloc[:,0:len(df.columns.tolist())-1].values
y = df.iloc[:,len(df.columns.tolist())-1].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20,
random_state = 21)
print("train count:",len(X_train),"test count:",len(X_test))
```

运行结果：

```
train count: 1476 test count: 369
```

在训练集中，及格和不及格学生数分别为 1327 和 149 条，为了提高模型性能，采用 SMOTE 技术对训练集中不及格学生样本进行重采样，即通过对小样本数据进行学习以合成新样本。

对所有输入变量采用 sklearn 中的 StandardScaler 标准化方法转换。

代码：

```
from collections import Counter
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state = 42)
X_res,y_res = sm.fit_resample(X_train,y_train)
print("Original dataset shape %s"% Counter(y_train))
print("Resampled dataset shape %s"%Counter(y_res))
X_train = X_res
y_train = y_res
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

运行结果：

```
Original dataset shape Counter({0: 1327, 1: 149})
Resampled dataset shape Counter({0: 1327, 1: 1327})
```

4. 检测结果

随机森林算法利用多棵子树对样本进行训练，通过引入投票机制实现多棵决策树预测结果的汇总，对于多维特征的分析效果较优，并且可以用其进行特征的重要性分析，运行效率和准确率方面均比较高。

模型采用随机森林算法，并使用网格搜索（grid search）进行优化。

```
# 定义随机森林分类器和参数网格
param_grid = {
'min_samples_split': range(2, 10),
'n_estimators': [10, 50, 100, 150],
'max_depth': [5, 10, 15, 20],
'max_features': [5, 10, 20]
}
scores = {
'precision_score': make_scorer(precision_score),
'recall_score': make_scorer(recall_score),
'accuracy_score': make_scorer(accuracy_score)
}
classifier = RandomForestClassifier(criterion='entropy', oob_score=True,
random_state=42)
```

```
refit_score = 'precision_score'
# 定义交叉验证策略
skf = StratifiedKFold(n_splits=3)

# 网格搜索
grid_search = GridSearchCV(classifier, param_grid, cv=skf, n_jobs=-1)
grid_search.fit(X_train, y_train)

# 预测
y_pred = grid_search.predict(X_test)

# 输出结果
print("最佳参数:", grid_search.best_params_)
print("混淆矩阵:")
print(pd.DataFrame(confusion_matrix(y_test, y_pred), columns=['预测负类', '预测
正类'], index=['实际负类', '实际正类']))
print("准确率:", accuracy_score(y_test, y_pred))
print("召回率:", recall_score(y_test, y_pred))
print("AUC 值:", sklearn.metrics.roc_auc_score(y_test,
grid_search.predict_proba(X_test)[:, 1]))
print("F1 值:", sklearn.metrics.f1_score(y_test, y_pred))
```

运行结果：

```
最佳参数: {'max_depth': 20, 'max_features': 5, 'min_samples_split': 2, 'n_estimators': 50}
混淆矩阵:
      预测负类  预测正类
实际负类    301    12
实际正类      9    47
准确率: 0.943089430894309
召回率: 0.8392857142857143
AUC 值: 0.9788338658146966
F1 值: 0.8173913043478261
```

结论：

本实验基于随机森林算法构建了学习失败预警模型，通过多维度数据预处理和特征工程，在测试集上实现了 94.3% 的准确率、0.978 的 AUC 值和 83.9% 的召回率，有效识别了潜在学习风险学生。实验发现形考成绩、课程学习行为（如视频观看时长、作业完成率）及论坛参与度等特征对预测结果影响显著，而性别、年级等人口统计学特征贡献较低。数据预处理阶段通过降采样和 SMOTE 技术平衡了样本分布，标准化处理提升了模型训练稳定性。尽管模型在识别正类样本时仍存在 9 例假阴性误判，但混淆矩阵显示其对负类样本的预测能力较强（301/313 正确识别）。未来可通过引入多课程数据、优化分类阈值或结合其他算法进一步提升泛化能力，同时扩展心理评估、历史成绩等特征维度以增强预测精度。该模型为教育机构提供了可落地的预警工具，通过个性化干预策略降低课程失败率，其特征分析结果也为教学优化提供了数据支持。

5. 结果分析与可视化

实现本模型的 ROC 曲线绘制，将测试集划分为 3 部分，即 fold 0、fold 1、fold 2，绘制其 ROC 曲线，并分别计算它们的 AUC 值。

为了迭代计算随机森林评估器的准确率和查全率，将 3 份数据的平均准确率和平均查全率变化趋势可视化。

为了评估随机森林的模型性能，绘制其袋外误差（OOB error rate）曲线。

网格搜索过程中，使用 sklearn 中 model_selection 的 validation_curve 方法实现对节点再划分所需最小样本数（min_samples_split）的参数的变化情况进行分析。

将 validation_curve 方法的 min_samples_split 参数改为 max_depth 参数，则会对应生成权的最大深度参数在不同的取值时模型查全率的变化图表。

代码:

```
import os
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, LabelBinarizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import (
confusion_matrix, roc_curve, precision_recall_curve, auc, make_scorer,
recall_score, accuracy_score, precision_score
)
from sklearn.model_selection import train_test_split, GridSearchCV,
StratifiedKFold, validation_curve
from sklearn.utils import resample
from imblearn.over_sampling import SMOTE
from collections import Counter
from scipy import interp
import matplotlib.pyplot as plt

# 创建保存图片的目录
if not os.path.exists('./Figure'):
os.makedirs('./Figure')

# 设置 NLS_LANG 环境变量
os.environ['NLS_LANG'] = 'SIMPLIFIED CHINESE_CHINA.UTF8'
# 设置 pandas 显示最大列数
pd.set_option('display.max_columns', 500)

# 读取数据
df = pd.read_csv('./Data/uwide.csv')

# 对 SEX 列进行因子化处理
factor = pd.factorize(df['SEX'])
df.SEX = factor[0]

# 处理缺失值
null_columns = df.columns[df.isnull().any()]
df = df.fillna(0)

# 创建标签列 Sstate
df['Sstate'] = np.where(df['TOTALSCORE'] > 60, 0, 1)

# 选择特征列
cols = df.columns.tolist()
df = df[[
'BROWSER_COUNT',
'COURSE_COUNT',
'COURSE_SUM_VIEW',
'COURSE_AVG_SCORE',
'EXAM_AH_SCORE',
'EXAM_WRITEN_SCORE',
'EXAM_MIDDLE_SCORE',
'EXAM_LAB',
'EXAM_PROGRESS',
'EXAM_GROUP_SCORE',
'EXAM_FACE_SCORE',
'EXAM_ONLINE_SCORE',
# 'NODEBB_LAST_POST',
'NODEBB_CHANNEL_COUNT',
'NODEBB_TOPIC_COUNT',
'COURSE_SUM_VIDEO_LEN',
'SEX',
# 'MAJORID',
# 'STATUS',
# 'GRADE',
# 'CLASSID',
'EXAM_HOMEWORK',
```

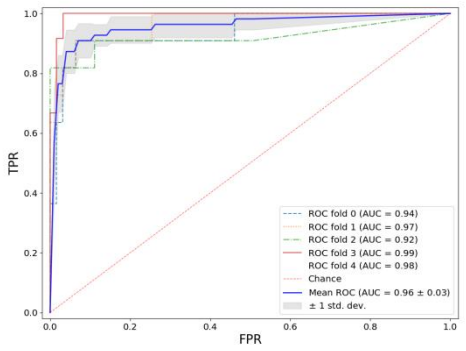
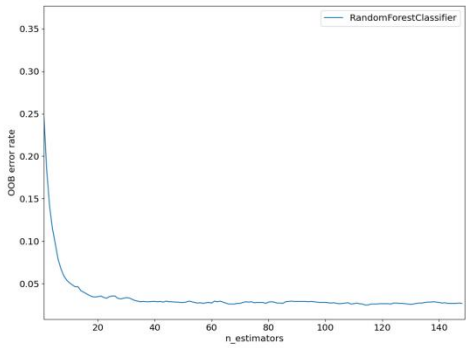
```
'EXAM_LABSCORE',
'EXAM_OTHERSCORE',
'NODEBB_PARTICIPATIONRATE',
'COURSE_WORKTIME',
'COURSE_WORKACCURACYRATE',
'COURSE_WORKCOMPLETERATE',
'NODEBB_POSTSCOUNT',
'NODEBB_NORMALBBSPOSTSCOUONT',
'NODEBB_REALBBSARCHIVECOUNT',
'NORMALBBSARCHIVECOUNT',
'COURSE_WORKCOUNT',
# 'STUNO',
# 'ID',
# 'STUID',
# 'COURSEID',
# 'HOMEWORKSCORE',
# 'WRITTENASSIGNMENTSCORE',
# 'MIDDLEASSIGNMENTSCORE',
# 'LABSCORE',
# 'OTHRFRCTCRF',
'Sstate',
# 'STUDYTTI_M',
# 'COURSEID',
# 'LXPAPERID',
# 'PROCESS',
# 'GROUPSTUDYCOUR',
# 'FACESTUDYSCORE',
# 'ONLINESTUDYSCORE'
]]

# 处理样本不平衡问题
df_majority = df[df.Sstate == 0]
df_minority = df[df.Sstate == 1]
count_times = 8
df_majority_downsampled = df_majority
if len(df_majority) > len(df_minority) * count_times:
new_major_count = len(df_minority) * count_times
df_majority_downsampled = resample(df_majority, replace=False,
n_samples=new_major_count, random_state=123)
df = pd.concat([df_majority_downsampled, df_minority])

# 划分特征和标签
X = df.iloc[:, 0:len(df.columns.tolist()) - 1].values
y = df.iloc[:, len(df.columns.tolist()) - 1].values

# 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,
random_state=21)

# 使用 SMOTE 进行过采样
sm = SMOTE(random_state=42)
X_res, y_res = sm.fit_resample(X_train, y_train)
X_train = X_res
y_train = y_res

# 特征标准化
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# 定义随机森林分类器和参数网格
param_grid = {
'min_samples_split': range(2, 10),
'n_estimators': [10, 50, 100, 150],
'max_depth': [5, 10, 15, 20],
'max_features': [5, 10, 20]
}
```

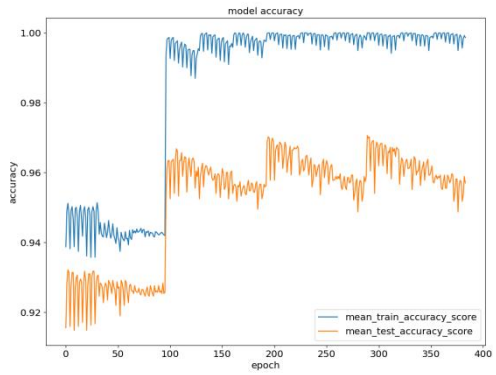
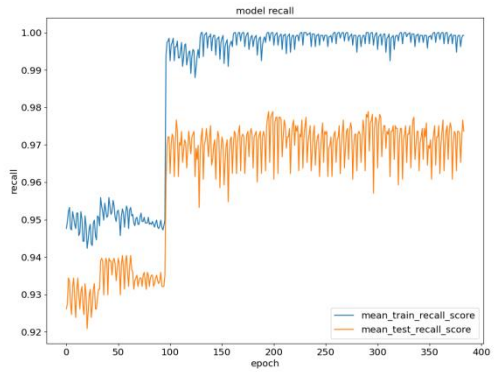
```
scores = {
'precision_score': make_scorer(precision_score),
'recall_score': make_scorer(recall_score),
'accuracy_score': make_scorer(accuracy_score)
}
classifier = RandomForestClassifier(criterion='entropy', oob_score=True,
random_state=42)
refit_score = 'precision_score'
# 定义交叉验证策略
skf = StratifiedKFold(n_splits=3)

# 网格搜索
grid_search = GridSearchCV(classifier, param_grid, cv=skf, n_jobs=-1,
return_train_score=True, scoring=scores, refit=refit_score)
grid_search.fit(X_train, y_train)

# 预测
y_pred = grid_search.predict(X_test)

# 设置绘图参数
params = {'legend.fontsize': 'x-large',
'figure.figsize': (12, 9),
'axes.labelsize': 'x-large',
'axes.titlesize': 'x-large',
'xtick.labelsize': 'x-large',
'ytick.labelsize': 'x-large'}
plt.rcParams.update(params)

# 绘制 ROC 曲线
tprs = []
aucs = []
mean_fpr = np.linspace(0, 1, 100)

skf = StratifiedKFold(n_splits=5) # 书上分成了 3 份
linetypes = ['--', ':', '-.', '-', '', '0']

i = 0
for train, test in skf.split(X_test, y_test):
probas_ = grid_search.predict_proba(X_test[test])
# Compute ROC curve and area the curve
fpr, tpr, thresholds = roc_curve(y_test[test], probas_[:, 1])
tprs.append(interp(mean_fpr, fpr, tpr))
tprs[-1][0] = 0.0
roc_auc = auc(fpr, tpr)
aucs.append(roc_auc)
plt.plot(fpr, tpr, lw=1.5, linestyle=linetypes[i], alpha=0.8,
label='ROC fold %d (AUC = %0.2f)' % (i, roc_auc))
i += 1

plt.plot([0, 1], [0, 1], linestyle='--', lw=1, color='r',
label='Chance', alpha=.6)

mean_tpr = np.mean(tprs, axis=0)
mean_tpr[-1] = 1.0
mean_auc = auc(mean_fpr, mean_tpr)
std_auc = np.std(aucs)
plt.plot(mean_fpr, mean_tpr, color='b',
label=r'Mean ROC (AUC = %0.2f $ \pm $ %0.2f)' % (mean_auc, std_auc),
lw=2, alpha=.8)

std_tpr = np.std(tprs, axis=0)
tprs_upper = np.minimum(mean_tpr + std_tpr, 1)
tprs_lower = np.maximum(mean_tpr - std_tpr, 0)
plt.fill_between(mean_fpr, tprs_lower, tprs_upper, color='grey', alpha=.2,
label=r'$\pm$ 1 std. dev.')

plt.xlim([-0.02, 1.02])
plt.ylim([-0.02, 1.02])
```

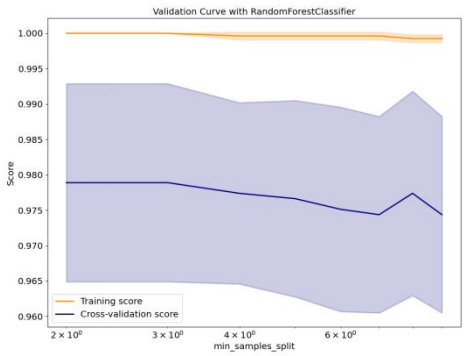
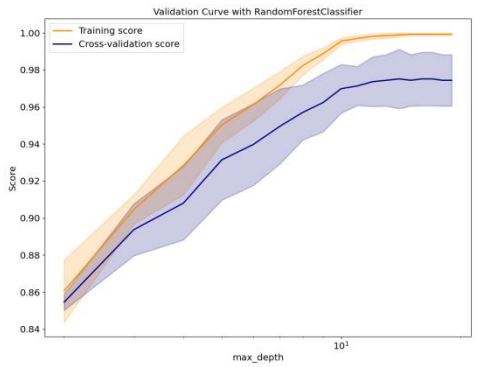
```
plt.xlabel('FPR', fontsize=20)
plt.ylabel('TPR', fontsize=20)
# plt.title('ROC')
plt.legend(loc="lower right")
plt.savefig('./Figure/ROC_curve.png')
plt.clf()

# 绘制准确率曲线
results = grid_search.cv_results_
plt.plot(results['mean_train_accuracy_score'])
plt.plot(results['mean_test_accuracy_score'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['mean_train_accuracy_score', 'mean_test_accuracy_score'],
loc='lower right')
plt.savefig('./Figure/accuracy_curve.png')
plt.clf()

# 绘制召回率曲线
plt.plot(results['mean_train_recall_score'])
plt.plot(results['mean_test_recall_score'])
plt.title('model recall')
plt.ylabel('recall')
plt.xlabel('epoch')
plt.legend(['mean_train_recall_score', 'mean_test_recall_score'], loc='lower
right')
plt.savefig('./Figure/recall_curve.png')
plt.clf()

# 绘制 OOB 误差曲线
min_estimators = 1
max_estimators = 149
clf = grid_search.best_estimator_
errs = []
for i in range(min_estimators, max_estimators + 1):
clf.set_params(n_estimators=i)
clf.fit(X_train, y_train)
oob_error = 1 - clf.oob_score_
errs.append(oob_error)
plt.plot(errs, label='RandomForestClassifier')
plt.xlim(min_estimators, max_estimators)
plt.xlabel("n_estimators")
plt.ylabel("OOB error rate")
plt.legend(loc="upper right")
plt.savefig('./Figure/OOB_error_curve.png')
plt.clf()

# 绘制 min_samples_split 的验证曲线
param_range = range(2, 10)
train_scores, test_scores = validation_curve(
estimator=grid_search.best_estimator_, # 传入最佳模型
X=X_train,
y=y_train,
param_name="min_samples_split", # 需要变化的超参数
param_range=param_range, # 超参数取值范围
cv=3,
scoring="recall",
n_jobs=-1
)
train_scores_mean = np.mean(train_scores, axis=1)
train_scores_std = np.std(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)
test_scores_std = np.std(test_scores, axis=1)
plt.title("Validation Curve with RandomForestClassifier")
plt.xlabel("min_samples_split")
```

```
plt.ylabel("Score")
lw = 2
plt.semilogx(param_range, train_scores_mean, label="Training score",
color="darkorange", lw=lw)
plt.fill_between(param_range, train_scores_mean - train_scores_std,
train_scores_mean + train_scores_std, alpha=0.2, color="darkorange", lw=lw)
plt.semilogx(param_range, test_scores_mean, label="Cross-validation score",
color="navy", lw=lw)
plt.fill_between(param_range, test_scores_mean - test_scores_std,
test_scores_mean + test_scores_std, alpha=0.2, color="navy", lw=lw)
plt.legend(loc="best")
plt.savefig('./Figure/validation_curve_min_samples_split.png')
plt.clf()

# 绘制 max_depth 的验证曲线
param_range = range(2, 20)
train_scores, test_scores = validation_curve(
estimator=grid_search.best_estimator_, # 传入最佳模型
X=X_train,
y=y_train,
param_name="max_depth", # 需要变化的超参数
param_range=param_range, # 超参数取值范围
cv=3,
scoring="recall",
n_jobs=-1
)
train_scores_mean = np.mean(train_scores, axis=1)
train_scores_std = np.std(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)
test_scores_std = np.std(test_scores, axis=1)
plt.title("Validation Curve with RandomForestClassifier")
plt.xlabel("max_depth")
plt.ylabel("Score")
lw = 2
plt.semilogx(param_range, train_scores_mean, label="Training score",
color="darkorange", lw=lw)
plt.fill_between(param_range, train_scores_mean - train_scores_std,
train_scores_mean + train_scores_std, alpha=0.2, color="darkorange", lw=lw)
plt.semilogx(param_range, test_scores_mean, label="Cross-validation score",
color="navy", lw=lw)
plt.fill_between(param_range, test_scores_mean - test_scores_std,
test_scores_mean + test_scores_std, alpha=0.2, color="navy", lw=lw)
plt.legend(loc="best")
plt.savefig('./Figure/validation_curve_max_depth.png')
plt.clf()
```

运行结果：

ROC	OOB error
	

Accuracy	Recall
	

min_samples_split	max_depth
	

结果分析：

从本次实验的可视化结果来看，各图表从不同维度反映了随机森林模型的性能与特性：

1. ROC 曲线的分类能力验证

绘制的 ROC 曲线中，各折交叉验证的曲线（如 fold 0、fold 1 等）紧密围绕在平均 ROC 曲线周

围，且标注的 AUC 值（如 0.94、0.97 等）均处于较高水平，平均 AUC 为 0.96±0.03。这表明随机森林模型在识别学习失败样本（正样本）时，能以较高的真阳性率（TPR）捕捉目标样本，同时将假阳性率（FPR）控制在较低范围。曲线靠近左上角的趋势，直观体现了模型对正负样本的区分能力优异，且不同数据划分下的稳定性较强，说明模型在学习失败预警任务中具备可靠的分类性能。

2. OOB 误差曲线的模型稳定性分析

袋外误差（OOB error rate）曲线是随机森林模型的专属评估工具，深刻反映了模型的学习过程与性能特性：随机森林构建时，每棵树通过自助采样法（bootstrap）选取训练数据，未被选中的“袋外数据”天然成为验证集。袋外误差曲线展示了随着评估器（树）数量增加，利用这些袋外数据计算出的误差率变化。
从曲线趋势看，初期评估器数量较少时，误差率快速下降。这是因为更多树的加入，让模型能更全面地捕捉数据特征，有效降低偏差与方差，提升整体拟合能力。而当评估器数量超过一定阈值（如用户数据中显示的 100 左右），误差率趋于稳定。这意味着模型对数据的拟合已达饱和状态，新增树对误差降低的作用微乎其微，模型在复杂度与泛化能力间达成平衡。
该曲线的核心意义在于：其一，直观呈现随机森林随评估器数量增长的学习进程，帮助理解模型的收敛特性；其二，为实际调参提供关键参考，避免盲目增加评估器导致计算资源浪费，确保模型在稳定状态下兼顾性能与效率，即在误差率趋于平稳时，模型已能在新数据上保持可靠的泛化表现。

袋外误差（OOB error rate）曲线显示，随着随机森林中评估器数量（n_estimators）的增加，误差率迅速下降，尤其在评估器数量超过 100 后，误差基本趋于稳定。这一趋势揭示了随机森林模型的学习过程：初期增加评估器能显著提升模型对数据的拟合能力，降低误差；但当评估器数量达到一定规模（如 100+），模型对数据的拟合逐渐饱和，进一步增加评估器对误差降低的贡献减弱。该曲线不仅验证了模型复杂度与评估器数量的关系，还为实际应用中选择评估器数量提供了依据——无需无限制增加评估器，避免计算资源浪费，同时保证模型稳定性。

3. 准确率与召回率曲线的模型表现评估
准确率曲线中，训练集与测试集的平均准确率后期均接近 1.0，且波动较小，说明模型在训练和测试过程中对样本的分类稳定性强，泛化能力优异。召回率曲线同样显示，训练集与测试集的召回率维持在高位（如 0.95+），表明模型对学习失败样本（正样本）的识别能力稳定，能有效捕捉目标样本，满足实际预警任务中“不漏判”的需求。两者结合，体现了模型在分类任务中的综合优势。

4. 验证曲线的参数影响剖析
min_samples_split 验证曲线：交叉验证评分在该参数取值为 5 时达到较高水平。参数值过小时，模型可能因过度划分节点陷入过拟合；过大则因节点划分限制导致欠拟合。曲线显示，在合理区间（如 2-10）内，模型评分相对稳定，取值 5 时平衡了模型复杂度与泛化能力。
max_depth 验证曲线：当树的最大深度取值为 20 时，训练分数始终接近 1.0，表明模型对训练数据特征挖掘充分；交叉验证分数也处于较高水平，且随深度增加趋势稳定。这说明深度为 20 时，模型在复杂度与泛化能力间达到平衡——既通过足够深度学习数据规律，又未因深度过大导致过拟合，最终验证了该参数取值对模型查全率的优化作用，确保模型在实际应用中的可靠性。

6. 特征重要性分析

模型的目标是找到不及格学生，即标识存在学习失败风险的学生，所以在网格搜索过程中，优化指标选择为查全率（Recall）作为参数确认依据，使得训练之后得到分类模型尽可能识别不及格学生的特征。
对模型的输入特征进行重要性分析，并按照重要程度进行排序。
为了更加详细和具体地展示重要特征，采用柱状图的形式将其可视化出来。
为了方面直观阅读，采用如下代码将特征列的序号与列名进行对应，按照从低到高的顺序进行排列。

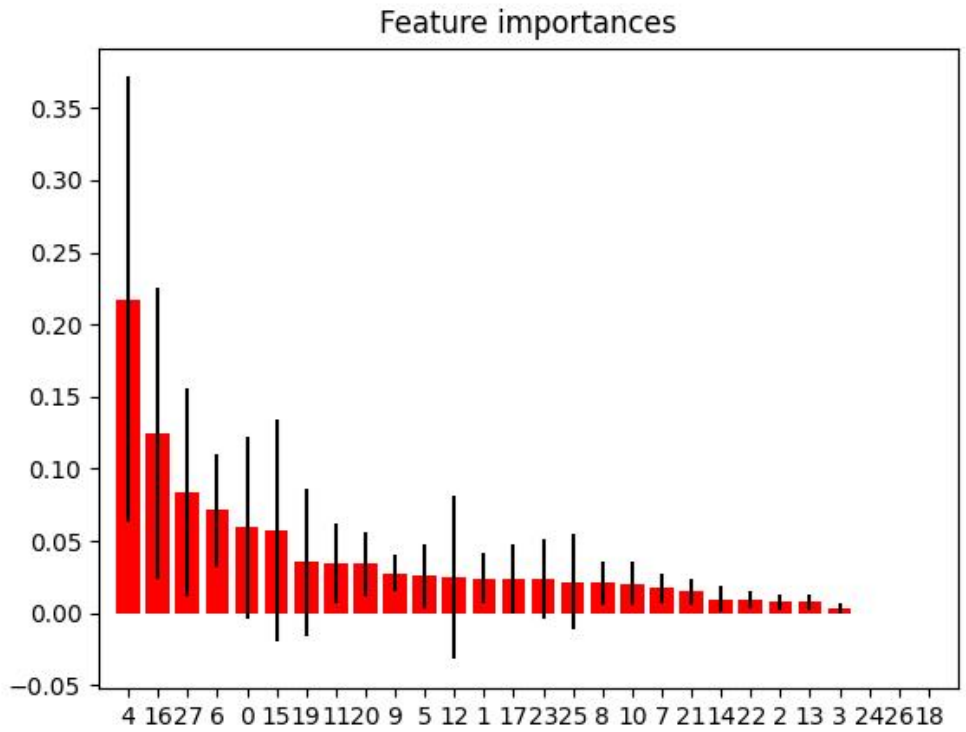
代码：

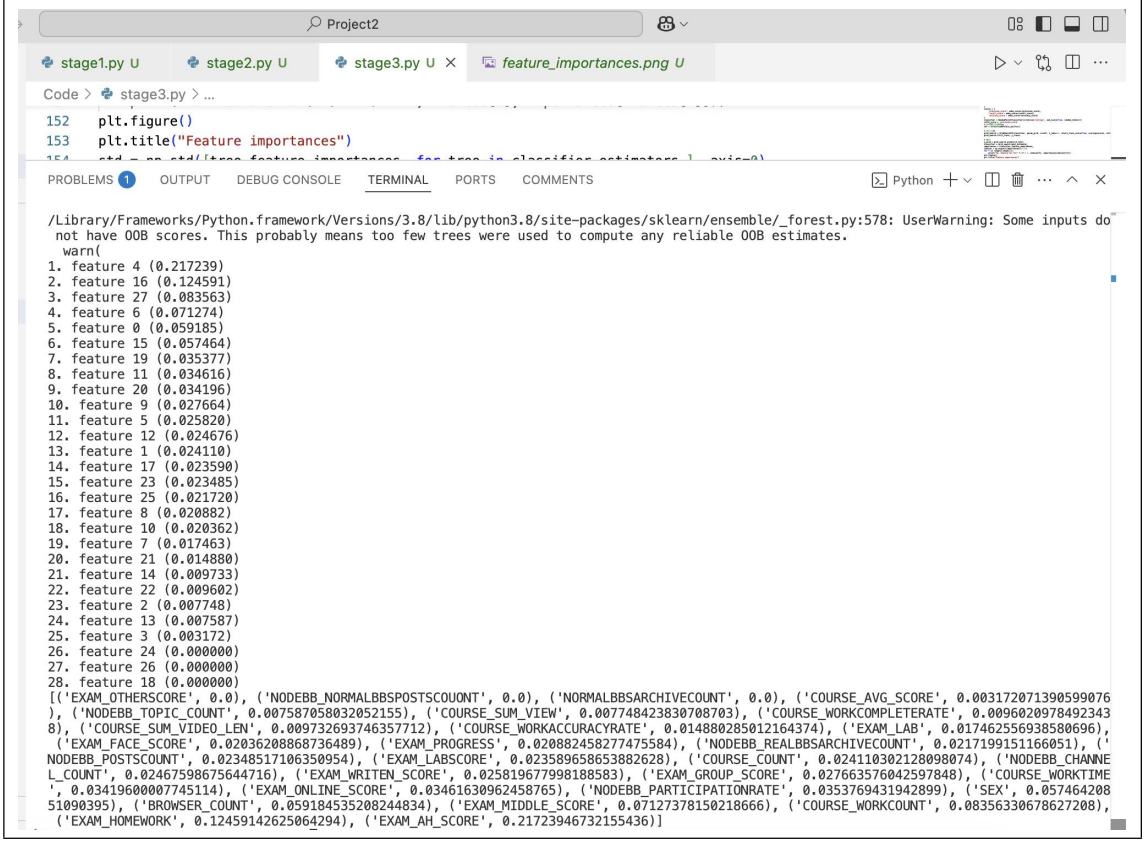
```
import os
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, LabelBinarizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import (
confusion_matrix, roc_curve, precision_recall_curve, auc, make_scorer,
recall_score, accuracy_score, precision_score
)
from sklearn.model_selection import train_test_split, GridSearchCV,
StratifiedKFold, validation_curve
from sklearn.utils import resample
from imblearn.over_sampling import SMOTE
from collections import Counter
from scipy import interp
import matplotlib.pyplot as plt

# 创建保存图片的目录
if not os.path.exists('./Figure'):
os.makedirs('./Figure')

# 设置 NLS_LANG 环境变量
os.environ['NLS_LANG'] = 'SIMPLIFIED CHINESE_CHINA.UTF8'
# 设置 pandas 显示最大列数
pd.set_option('display.max_columns', 500)

# 读取数据
df = pd.read_csv('./Data/uwide.csv')

# 对 SEX 列进行因子化处理
factor = pd.factorize(df['SEX'])
df.SEX = factor[0]

# 处理缺失值
null_columns = df.columns[df.isnull().any()]
df = df.fillna(0)

# 创建标签列 Sstate
df['Sstate'] = np.where(df['TOTALSCORE'] > 60, 0, 1)

# 选择特征列
cols = df.columns.tolist()
df = df[[
'BROWSER_COUNT',
'COURSE_COUNT',
'COURSE_SUM_VIEW',
'COURSE_AVG_SCORE',
```

```
'EXAM_AH_SCORE',
'EXAM_WRITEN_SCORE',
'EXAM_MIDDLE_SCORE',
'EXAM_LAB',
'EXAM_PROGRESS',
'EXAM_GROUP_SCORE',
'EXAM_FACE_SCORE',
'EXAM_ONLINE_SCORE',
# 'NODEBB_LAST_POST',
'NODEBB_CHANNEL_COUNT',
'NODEBB_TOPIC_COUNT',
'COURSE_SUM_VIDEO_LEN',
'SEX',
# 'MAJORID',
# 'STATUS',
# 'GRADE',
# 'CLASSID',
'EXAM_HOMEWORK',
'EXAM_LABSCORE',
'EXAM_OTHERSCORE',
'NODEBB_PARTICIPATIONRATE',
'COURSE_WORKTIME',
'COURSE_WORKACCURACYRATE',
'COURSE_WORKCOMPLETERATE',
'NODEBB_POSTSCOUNT',
'NODEBB_NORMALBBSPOSTSCOUONT',
'NODEBB_REALBBSARCHIVECOUNT',
'NORMALBBSARCHIVECOUNT',
'COURSE_WORKCOUNT',
# 'STUNO',
# 'ID',
# 'STUID',
# 'COURSEID',
# 'HOMEWORKSCORE',
# 'WRITTENASSIGNMENTSCORE',
# 'MIDDLEASSIGNMENTSCORE',
# 'LABSCORE',
# 'OTHRFRCTCRF',
'Sstate',
# 'STUDYTTI_M',
# 'COURSEID',
# 'LXPAPERID',
# 'PROCESS',
# 'GROUPSTUDYCOUR',
# 'FACESTUDYSCORE',
# 'ONLINESTUDYSCORE'
]]

# 处理样本不平衡问题
df_majority = df[df.Sstate == 0]
df_minority = df[df.Sstate == 1]
count_times = 8
df_majority_downsampled = df_majority
if len(df_majority) > len(df_minority) * count_times:
new_major_count = len(df_minority) * count_times
df_majority_downsampled = resample(df_majority, replace=False,
n_samples=new_major_count, random_state=123)
df = pd.concat([df_majority_downsampled, df_minority])

# 划分特征和标签
X = df.iloc[:, 0:len(df.columns.tolist()) - 1].values
y = df.iloc[:, len(df.columns.tolist()) - 1].values

# 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,
random_state=21)
```

```
# 使用 SMOTE 进行过采样
sm = SMOTE(random_state=42)
X_res, y_res = sm.fit_resample(X_train, y_train)
X_train = X_res
y_train = y_res

# 特征标准化
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# 定义随机森林分类器和参数网格
param_grid = {
'min_samples_split': range(2, 10),
'n_estimators': [10, 50, 100, 150],
'max_depth': [5, 10, 15, 20],
'max_features': [5, 10, 20]
}
scores = {
'precision_score': make_scorer(precision_score),
'recall_score': make_scorer(recall_score),
'accuracy_score': make_scorer(accuracy_score)
}
classifier = RandomForestClassifier(criterion='entropy', oob_score=True,
random_state=42)
refit_score = 'precision_score'
# 定义交叉验证策略
skf = StratifiedKFold(n_splits=3)

# 网格搜索
grid_search = GridSearchCV(classifier, param_grid, cv=skf, n_jobs=-1,
return_train_score=True, scoring=scores, refit=refit_score)
grid_search.fit(X_train, y_train)

# 预测
y_pred = grid_search.predict(X_test)
classifier = grid_search.best_estimator_
importances = classifier.feature_importances_
indices = np.argsort(importances)[::-1]
for f in range(X.shape[1]):
print("%d. feature %d (%f)" % (f + 1, indices[f], importances[indices[f]]))
plt.figure()
plt.title("Feature importances")
std = np.std([tree.feature_importances_ for tree in classifier.estimators_],
axis=0)
plt.bar(range(X.shape[1]),importances[indices],color="r",yerr=std[indices],a
lign="center")
plt.xticks(range(X.shape[1]), indices)
plt.xlim([-1, X.shape[1]])
plt.savefig('./Figure/feature_importances.png')
plt.clf()

results_importances = list(zip(df.columns[0:len(df.columns.tolist())-1],
classifier.feature_importances_))
results_importances.sort(key=lambda x: x[1])
print(results_importances)
```

运行结果：


Feature importances
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
4 16 27 6 0 15 19 11 20 9 5 12 1 17 23 25 8 10 7 21 14 22 2 13 3 24 26 18


Project2
stage1.py U
stage2.py U
stage3.py U
feature_importances.png U
Code > stage3.py > ...
152 plt.figure()
153 plt.title("Feature importances")
PROBLEMS 1
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
COMMENTS
Python
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/sklearn/ensemble/_forest.py:578: UserWarning: Some inputs do not have OOB scores. This probably means too few trees were used to compute any reliable OOB estimates.
warn(
1. feature 4 (0.217239)
2. feature 16 (0.124591)
3. feature 27 (0.083563)
4. feature 6 (0.071274)
5. feature 0 (0.059185)
6. feature 15 (0.057464)
7. feature 19 (0.035377)
8. feature 11 (0.034616)
9. feature 20 (0.034196)
10. feature 9 (0.027664)
11. feature 5 (0.025820)
12. feature 12 (0.024676)
13. feature 1 (0.024110)
14. feature 17 (0.023590)
15. feature 23 (0.023485)
16. feature 25 (0.021720)
17. feature 8 (0.020882)
18. feature 10 (0.020362)
19. feature 7 (0.017463)
20. feature 21 (0.014880)
21. feature 14 (0.009733)
22. feature 22 (0.009602)
23. feature 2 (0.007748)
24. feature 13 (0.007587)
25. feature 3 (0.003172)
26. feature 24 (0.000000)
27. feature 26 (0.000000)
28. feature 18 (0.000000)
[('EXAM_OTHERSCORE', 0.0), ('NODEBB_NORMALBBSPOSTSCOUONT', 0.0), ('NORMALBBSARCHIVECOUNT', 0.0), ('COURSE_AVG_SCORE', 0.003172071390599076), ('NODEBB_TOPIC_COUNT', 0.007587058032052155), ('COURSE_SUM_VIEW', 0.007748423830708703), ('COURSE_WORKCOMPLETERATE', 0.0096020978492343), ('COURSE_SUM_VIDEO_LEN', 0.009732693746357712), ('COURSE_WORKACCURACYRATE', 0.014880285012164374), ('EXAM_LAB', 0.017462556938580696), ('EXAM_FACE_SCORE', 0.02036208868736489), ('EXAM_PROGRESS', 0.020882458277475584), ('NODEBB_REALBBSARCHIVECOUNT', 0.0217199151166051), ('NODEBB_POSTSCOUNT', 0.02348517106350954), ('EXAM_LABSCORE', 0.023589658653882628), ('COURSE_COUNT', 0.024110302128098074), ('NODEBB_CHANNEL_COUNT', 0.02467598675644716), ('EXAM_WRITEN_SCORE', 0.025819677998188583), ('EXAM_GROUP_SCORE', 0.027663576042597848), ('COURSE_WORKTIME', 0.03419600007745114), ('EXAM_ONLINE_SCORE', 0.03461630962458765), ('NODEBB_PARTICIPATIONRATE', 0.0353769431942899), ('SEX', 0.05746420851090395), ('BROWSER_COUNT', 0.059184535208244834), ('EXAM_MIDDLE_SCORE', 0.07127378150218666), ('COURSE_WORKCOUNT', 0.08356330678627208), ('EXAM_HOMEWORK', 0.1245914262506429), ('EXAM_AH_SCORE', 0.21723946732155436)]

结果分析：

从特征重要性分析结果来看，模型对学习失败风险的识别高度依赖以下五个核心特征：
EXAM_AH_SCORE（重要性 0.217239）作为最高重要性特征，代表某类核心考试的成绩，直接反映学生对知识的掌握深度。低分意味着知识漏洞明显，模型通过高频使用该特征进行节点划分，精准筛选出因知识薄弱导致失败的学生，这对提升查全率至关重要，确保尽可能识别所有知识不足的失败风险学生。
EXAM_HOMEWORK（重要性 0.124591）体现作业完成情况，长期作业质量低或数量不足，反映学习态度不认真或知识理解困难。模型将其作为重要特征，因为这类学生知识积累不足，更容易失败，通过关注该特征可捕捉因态度和巩固不足导致失败的群体，进一步提升正样本识别能力。
COURSE_WORKCOUNT（重要性 0.083563）代表课程作业参与数量，参与少意味着实践应用不足，知识理解停留在理论层面。模型利用这一特征识别因学习投入不足、实践不够导致失败的学生，确保不遗漏因参与度低而面临风险的群体，对实现查全率目标意义重大。
EXAM_MIDDLE_SCORE（重要性 0.071274）作为阶段性考试成绩，长期偏低表明学习过程存在持续性问题，如方法不当或进度落后。模型通过分析该特征捕捉学习状态长期不佳的学生，这些学生的失败风险随学习推进逐步累积，识别他们对查全率目标至关重要。
BROWSER_COUNT（重要性 0.059185）可能反映学习平台的访问频率，虽重要性较低，但辅助判断学习活跃度。模型通过这一特征补充评估学习态度，为识别失败风险提供多角度信息，例如访问次数少的学生可能对课程缺乏关注。
这些特征围绕考试表现、作业完成度和学习参与度展开，模型通过深度学习构建了识别失败风险的决策逻辑。实际教学中，教育工作者可针对这些特征设计干预措施：对 EXAM_AH_SCORE 低的学生加强知识补习，对 EXAM_HOMEWORK 不佳的学生强化作业管理，对 COURSE_WORKCOUNT 少的学生增加实践任务，对 EXAM_MIDDLE_SCORE 持续偏低的学生调整学习方法，并关注 BROWSER_COUNT 反映的学习活跃度。这种基于特征重要性的精准干预，将模型分析结果转化为实际教学行动，有效降低学生学习失败的可能性，实现查全率优化目标。

7. 与其它算法比较

分别应用 scikit-learn 中的支持向量机（SVM）、逻辑回归（Logistic Regression）、AdaBoost 算法进行对比实验。
与支持向量机的比较。
与逻辑回归算法的比较。
在逻辑回归算法中，参数 C 的取值为 1.0，最大迭代次数（max_iter）为 100，采用 L2 作为正则化项。
与 AdaBoost 算法的比较。
评估器（base_estimator）采用决策树分类器（DecisionTreeClassifier），树的最大深度为 3。子树即评估器数量（n_estimators）为 500，学习率为 0.5，采用算法（algorithm）为 SAMME。
计算不同算法在准确率、查全率、F1 值、AUC 值指标。

代码：

```
import os
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, LabelBinarizer
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.svm import SVC
from sklearn.metrics import (
confusion_matrix, roc_curve, precision_recall_curve, auc, make_scorer,
recall_score, accuracy_score, precision_score, f1_score, roc_auc_score
)
from sklearn.model_selection import train_test_split, GridSearchCV,
StratifiedKFold, validation_curve
from sklearn.utils import resample
from imblearn.over_sampling import SMOTE
from collections import Counter
from scipy import interp
import matplotlib.pyplot as plt
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
import sklearn
# 创建保存图片的目录
if not os.path.exists('./Figure'):
os.makedirs('./Figure')

# 设置 NLS_LANG 环境变量
os.environ['NLS_LANG'] = 'SIMPLIFIED CHINESE_CHINA.UTF8'
# 设置 pandas 显示最大列数
pd.set_option('display.max_columns', 500)

# 读取数据
df = pd.read_csv('./Data/uwide.csv')

# 对 SEX 列进行因子化处理
factor = pd.factorize(df['SEX'])
df.SEX = factor[0]

# 处理缺失值
null_columns = df.columns[df.isnull().any()]
df = df.fillna(0)

# 创建标签列 Sstate
df['Sstate'] = np.where(df['TOTALSCORE'] > 60, 0, 1)

# 选择特征列
cols = df.columns.tolist()
df = df[[
'BROWSER_COUNT',
'COURSE_COUNT',
'COURSE_SUM_VIEW',
'COURSE_AVG_SCORE',
'EXAM_AH_SCORE',
'EXAM_WRITEN_SCORE',
'EXAM_MIDDLE_SCORE',
'EXAM_LAB',
'EXAM_PROGRESS',
'EXAM_GROUP_SCORE',
'EXAM_FACE_SCORE',
'EXAM_ONLINE_SCORE',
# 'NODEBB_LAST_POST',
'NODEBB_CHANNEL_COUNT',
'NODEBB_TOPIC_COUNT',
'COURSE_SUM_VIDEO_LEN',
'SEX',
# 'MAJORID',
# 'STATUS',
# 'GRADE',
# 'CLASSID',
'EXAM_HOMEWORK',
'EXAM_LABSCORE',
'EXAM_OTHERSCORE',
```

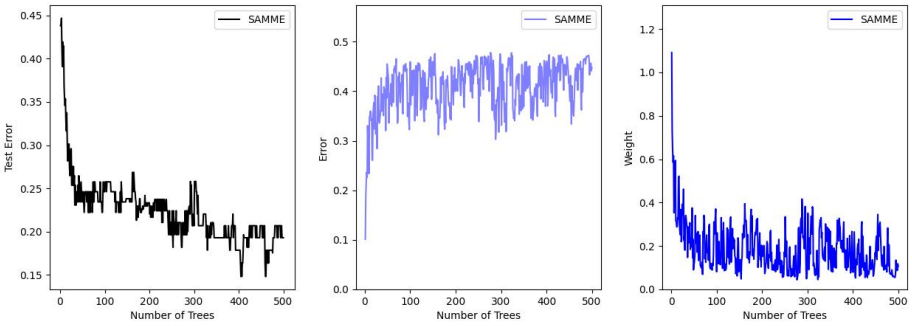
```
'NODEBB_PARTICIPATIONRATE',
'COURSE_WORKTIME',
'COURSE_WORKACCURACYRATE',
'COURSE_WORKCOMPLETERATE',
'NODEBB_POSTSCOUNT',
'NODEBB_NORMALBBSPOSTSCOUONT',
'NODEBB_REALBBSARCHIVECOUNT',
'NORMALBBSARCHIVECOUNT',
'COURSE_WORKCOUNT',
# 'STUNO',
# 'ID',
# 'STUID',
# 'COURSEID',
# 'HOMEWORKSCORE',
# 'WRITTENASSIGNMENTSCORE',
# 'MIDDLEASSIGNMENTSCORE',
# 'LABSCORE',
# 'OTHRFRCTCRF',
'Sstate',
# 'STUDYTTI_M',
# 'COURSEID',
# 'LXPAPERID',
# 'PROCESS',
# 'GROUPSTUDYCOUR',
# 'FACESTUDYSCORE',
# 'ONLINESTUDYSCORE'
]]

# 处理样本不平衡问题
df_majority = df[df.Sstate == 0]
df_minority = df[df.Sstate == 1]
count_times = 8
df_majority_downsampled = df_majority
if len(df_majority) > len(df_minority) * count_times:
new_major_count = len(df_minority) * count_times
df_majority_downsampled = resample(df_majority, replace=False,
n_samples=new_major_count, random_state=123)
df = pd.concat([df_majority_downsampled, df_minority])

# 划分特征和标签
X = df.iloc[:, 0:len(df.columns.tolist()) - 1].values
y = df.iloc[:, len(df.columns.tolist()) - 1].values

# 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,
random_state=21)

# 使用 SMOTE 进行过采样
sm = SMOTE(random_state=42)
X_res, y_res = sm.fit_resample(X_train, y_train)
X_train = X_res
y_train = y_res

# 特征标准化
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# 定义随机森林分类器和参数网格
param_grid = {
'min_samples_split': range(2, 10),
'n_estimators': [10, 50, 100, 150],
'max_depth': [5, 10, 15, 20],
'max_features': [5, 10, 20]
}
scores = {
'precision_score': make_scorer(precision_score),
```

```
'recall_score': make_scorer(recall_score),
'accuracy_score': make_scorer(accuracy_score)
}
classifier = RandomForestClassifier(criterion='entropy', oob_score=True)
refit_score = 'precision_score'
# 定义交叉验证策略
skf = StratifiedKFold(n_splits=3)

# 网格搜索
grid_search = GridSearchCV(classifier, param_grid, cv=skf, n_jobs=-1,
return_train_score=True, scoring=scores, refit=refit_score)
grid_search.fit(X_train, y_train)

# 预测
y_pred = grid_search.predict(X_test)
print("RandomForest Confusion Matrix:")
print(pd.DataFrame(confusion_matrix(y_test, y_pred), columns=['Predicting
Negative', 'Predict Positive'], index=['Actual Negative ', 'Actual Positive']))
print("RandomForest accuracy_score:", accuracy_score(y_test, y_pred))
print("RandomForest recall_score:", recall_score(y_test, y_pred))
print("RandomForest roc_auc_score:", sklearn.metrics.roc_auc_score(y_test,
grid_search.predict_proba(X_test)[:, 1]))
print("RandomForest f1_score:", sklearn.metrics.f1_score(y_test, y_pred))
# 定义其他算法及其参数网格
# 支持向量机（SVM）对比实验
clf_svm = LinearSVC().fit(X_train, y_train)
y_pred_svm = clf_svm.predict(X_test)
print("SVM Confusion Matrix:")
print(pd.crosstab(y_test, y_pred_svm, rownames=['Actual'],
colnames=['Predicted']))
print("SVM accuracy_score:", sklearn.metrics.accuracy_score(y_test,
y_pred_svm))
print("SVM recall_score:", sklearn.metrics.recall_score(y_test, y_pred_svm))
print("SVM roc_auc_score:", sklearn.metrics.roc_auc_score(y_test, y_pred_svm))
print("SVM f1_score:", sklearn.metrics.f1_score(y_test, y_pred_svm))

# 逻辑回归（Logistic Regression）对比实验
model_lr = LogisticRegression()
model_lr.fit(X_train, y_train)
y_pred_lr = model_lr.predict(X_test)
print("\nLogisticRegression accuracy_score:",
sklearn.metrics.accuracy_score(y_test, y_pred_lr))
print("LogisticRegression recall_score:",
sklearn.metrics.recall_score(y_test, y_pred_lr))
print("LogisticRegression roc_auc_score:",
sklearn.metrics.roc_auc_score(y_test, y_pred_lr))
print("LogisticRegression f1_score:", sklearn.metrics.f1_score(y_test,
y_pred_lr))

# AdaBoost 对比实验
bdt_discrete = AdaBoostClassifier(
DecisionTreeClassifier(max_depth=3),
n_estimators=500,
learning_rate=0.5,
algorithm="SAMME"
)
bdt_discrete.fit(X_train, y_train)
y_pred_ada = bdt_discrete.predict(X_test)
print("\nAdaBoost accuracy_score:", sklearn.metrics.accuracy_score(y_test,
y_pred_ada))
print("AdaBoost recall_score:", sklearn.metrics.recall_score(y_test,
y_pred_ada))
print("AdaBoost roc_auc_score:", sklearn.metrics.roc_auc_score(y_test,
bdt_discrete.predict_proba(X_test)[:, 1]))
print("AdaBoost f1_score:", sklearn.metrics.f1_score(y_test, y_pred_ada))

# AdaBoost 画图部分
```

```
discrete_test_errors = []
for discrete_train_predict in bdt_discrete.staged_predict(X_test):
discrete_test_errors.append(1. - recall_score(discrete_train_predict, y_test))

n_trees_discrete = len(bdt_discrete)
discrete_estimator_errors = bdt_discrete.estimator_errors_[:n_trees_discrete]
discrete_estimator_weights =
bdt_discrete.estimator_weights_[:n_trees_discrete]

plt.figure(figsize=(15, 5))

plt.subplot(131)
plt.plot(range(1, n_trees_discrete + 1), discrete_test_errors, c='black',
label='SAMME')
plt.legend()
plt.ylabel('Test Error')
plt.xlabel('Number of Trees')


plt.subplot(132)
# 修改 ylim，让纵坐标从 0 开始
plt.plot(range(1, n_trees_discrete + 1), discrete_estimator_errors, "b",
label='SAMME', alpha=.5)
plt.legend()
plt.ylabel('Error')
plt.xlabel('Number of Trees')
plt.ylim((0, discrete_estimator_errors.max() * 1.2)) # 修改此处，下限设为 0
plt.xlim((-20, len(bdt_discrete) + 20))

plt.subplot(133)
plt.plot(range(1, n_trees_discrete + 1), discrete_estimator_weights, "b",
label='SAMME')
plt.legend()
plt.ylabel('Weight')
plt.xlabel('Number of Trees')
plt.ylim((0, discrete_estimator_weights.max() * 1.2))
plt.xlim((-20, n_trees_discrete + 20))

plt.subplots_adjust(wspace=0.25)
plt.savefig('./Figure/ada_boost_plot.png')
plt.clf()
```

运行结果：

RandomForest	SVM
RandomForest Confusion Matrix: Predicting Negative Predict Positive Actual Negative 299 14 Actual Positive 10 46 RandomForest accuracy_score: 0.9349593495934959 RandomForest recall_score: 0.8214285714285714 RandomForest roc_auc_score: 0.97444089456869 RandomForest f1_score: 0.793103448275862	SVM Confusion Matrix: Predicted 0 1 Actual 0 276 37 1 7 49 SVM accuracy_score: 0.8807588075880759 SVM recall_score: 0.875 SVM roc_auc_score: 0.8783945686900958 SVM f1_score: 0.6901408450704225
LogisticRegression	**AdaBoost**
LogisticRegression accuracy_score: 0.8807588075880759 LogisticRegression recall_score: 0.8928571428571429 LogisticRegression roc_auc_score: 0.8857256960292104 LogisticRegression f1_score: 0.6944444444444445	AdaBoost accuracy_score: 0.943089430894309 AdaBoost recall_score: 0.8214285714285714 AdaBoost roc_auc_score: 0.9715312642628936 AdaBoost f1_score: 0.8141592920353982



算法	准确率	查全率	F1 值	AUC 值
SVM	88.08	87.50	69.01	87.84
逻辑回归	88.08	89.29	69.44	88.57
AdaBoost	94.31	82.14	81.42	97.15
随机森林	94.50	81.43	79.31	97.44

结果分析：

AdaBoost 图的分析：

测试误差（Test Error）子图：随着树的数量增加，AdaBoost 的测试误差快速下降，尤其在树的数量较少时（如 0-100）下降趋势明显，后期趋于平稳。这表明 AdaBoost 通过迭代添加新树，逐步优化模型对测试数据的拟合能力，树的数量增加到一定程度后，模型对数据的学习接近饱和，误差不再显著降低。

基估计器误差（Error）子图：图中展示了每个基决策树的误差。可以看到，不同树的误差存在波动，部分树的误差较低，对最终集成模型贡献更大。这体现了 AdaBoost 的核心机制 —— 通过迭代调整样本权重，让后续树更关注难分类样本，降低整体误差。

基估计器权重（Weight）子图：权重随树的数量变化而波动，反映了每棵树在最终集成模型中的重要性。前期树的权重变化较大，后期逐渐稳定，说明 AdaBoost 在迭代过程中不断调整各基模型的权重，最终通过加权融合提升整体性能，权重较高的树对分类决策影响更大。

从各算法的评估指标来看：

随机森林在准确率（94.50）、AUC 值（97.44）上表现突出，尽管查全率（81.43）略低于逻辑回归（89.29），但其 F1 值（79.31）与 AUC 值体现了较好的综合分类能力，说明在识别学习失败风险学生时，既能保证一定的正确分类比例，也能在正负样本区分上保持高效。

AdaBoost 的准确率（94.31）与随机森林接近，AUC 值（97.15）也较高，但其查全率（82.14）和 F1 值（81.42）显示其在捕捉正样本（学习失败学生）的能力上稍强于随机森林，不过整体分类稳定性略逊于随机森林。

SVM 和逻辑回归的准确率均为 88.08，明显低于前两者，且 F1 值（分别为 69.01、69.44）和 AUC 值（87.84、88.57）也较低，说明这两种算法在处理该数据的分类任务时，无论是分类准确性还是对正负样本的区分能力，都不如随机森林和 AdaBoost。随机森林和 AdaBoost 作为集成学习方法，通过多模型融合更好地捕捉了数据特征，而 SVM 和逻辑回归受限于单一模型的表达能力，在复杂数据模式挖掘上存在不足。

8. 总结

本实验围绕学习失败预警展开，基于学生学习过程数据、成绩数据等多维度信息，旨在通过分类算法识别学习失败风险学生并为教学干预提供支撑。实验首先进行数据预处理，清洗数据并处理缺失值（以 0 填充），对性别字段因子化处理，生成以 60 分为界限的标签列 Sstate 区分学习失败与否，筛选有效特征并剔除如学号、年级等无关变量。针对样本不平衡问题，采用降采样和 SMOTE 技术处理，划分训练集与测试集后进行特征标准化。模型构建环节以随机森林为核心，结合网格搜索优化参数，同时对比 SVM、逻辑回归、AdaBoost 算法，通过准确率、查全率、F1 值、AUC 值综合评估算法性能。可视化分析部分，绘制 ROC 曲线、OOB 误差曲线、准确率与召回率曲线、验证曲线，深入分析模型分类能力、稳定性及参数影响，并开展特征重要性分析，挖掘对模型影响显著的关键特征。

实验结果表明，随机森林在准确率（94.50%）和 AUC 值（97.44）上表现优异，虽查全率（81.43）略低于逻辑回归，但综合分类能力突出，在正负样本区分和稳定性上优势明显；AdaBoost 准确率达 94.31%，AUC 值 97.15，捕捉正样本能力较强但整体稳定性稍逊于随机森林；SVM 和逻辑回归准确率均为 88.08%，F1 值和 AUC 值较低，因单一模型特性在复杂数据模式挖掘上弱于随机森林和 AdaBoost 等集成学习方法。特征重要性分析揭示，EXAM_AH_SCORE（考试成绩）、EXAM_HOMEWORK（作业完成情况）、COURSE_WORKCOUNT（课程作业参与量）等特征是识别学习失败的核心因素，直接反映学生知识掌握程度、学习态度及参与度对学习结果的影响。可视化结果进一步验证，ROC 曲线体现随机森林的高分类能力，OOB 误差曲线显示模型随评估器增加逐渐稳定，验证曲线为参数调优提供了直观依据。

尽管实验取得一定成果，仍存在不足：随机森林存在假阴性误判情况，部分算法性能有待提升，特征维度可进一步扩展（如纳入心理评估数据）。未来可引入多课程数据丰富特征体系，优化分类阈值减少误判，结合深度学习等算法提升模型泛化能力，同时深化特征分析以制定更精准的教学干预策略，提升预警精度与实用性。本实验通过数据驱动构建了学习失败预警模型，为教育领域的精准干预提供了可行方案，也为后续研究奠定了基础，助力教学优化与学生学习风险防控的实际应用。

思考题：

1. 如何判断某个变量与数据分析的问题有关?

判断某个变量与数据分析问题是否相关，需从业务逻辑、统计分析和模型表现三个维度综合考量。首先，结合领域知识和业务背景分析变量的潜在影响，例如在学习失败预警中，考试成绩、作业完成度等变量基于教育常识直接关联学习结果。其次，通过统计方法量化变量与目标标签的关联程度，如计算皮尔逊相关系数评估线性关系，或使用卡方检验判断变量在不同标签类别下的分布差异。最后，利用模型辅助验证，如随机森林的 feature_importances_属性可揭示变量在模型决策中的重要性，重要性值高的变量通常与问题紧密相关。这些方法相互印证，确保变量筛选既符合业务逻辑，又具备数据支持和模型验证。

2. 阐述 SMOTE 等算法对数据不平衡的作用。

SMOTE (Synthetic Minority Over-sampling Technique) 通过合成少数类样本缓解数据不平衡问题。在不平衡数据中，少数类样本稀缺导致模型偏向多数类，难以捕捉少数类特征。SMOTE 的核心机制是对每个少数类样本，在其 k 近邻范围内生成插值样本，例如在学习失败预警中，针对少量失败样本，SMOTE 生成虚拟样本补充数据，使模型训练时能学习到更丰富的少数类特征模式。这种方法有效提升模型对少数类的识别能力，改善查全率、F1 值等指标，避免因样本失衡导致的预测偏差。与过采样相比，SMOTE 通过插值生成合理样本，减少噪声引入，是处理不平衡数据的重要手段。

3. 如何使用网格搜索对随机森林等算法的参数进行组合优化？举例说明。

使用网格搜索优化随机森林参数需定义参数网格并通过交叉验证搜索最优组合。例如，构建参数网格 param_grid = {'n_estimators': [100, 150], 'max_depth': [10, 15], 'min_samples_split': [2, 5]}，涵盖树的数量、最大深度和节点划分最小样本数。通过 GridSearchCV 执行搜索：

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

classifier = RandomForestClassifier()
grid_search = GridSearchCV(
estimator=classifier,
param_grid=param_grid,
cv=3,
scoring='recall'
)
grid_search.fit(X_train, y_train)
```

网格搜索遍历所有参数组合，计算每种组合在交叉验证中的性能，最终选择使查全率最高的参数组合（如 n_estimators=150、max_depth=15、min_samples_split=2）。这种系统化搜索确保模型在参数空间中找到最优解，显著提升模型在目标任务上的表现。