# python代码

#缺失值的python代码

import pandas as pd

import matplotlib

from matplotlib import pyplot

import matplotlib.pyplot as plt

import missingno as msno

%matplotlib inline

df = pd.read\_csv('C:/Users/hmoye/Desktop/BPD8-23\_1.csv')

df

msno.matrix(df)

path ='C:/Users/hmoye/Desktop/'

fig = plt.figure(figsize=(16,8))

msno.matrix(df)

fig = plt.gcf()

plt.show()

fig.savefig(path+'missingno7.svg',format='svg',dip=300)

msno.bar(df)

fig = plt.gcf()

# 显示图形

plt.show()

# 保存图形为PDF

plt.savefig('missing\_values\_bar1.pdf')

#机器学习模型的拟合及SHAP的python代码

#### Load the necessary modules###

import pandas as pd

import numpy as np

import pickle

import matplotlib

from matplotlib import pyplot

import matplotlib.pyplot as plt

import seaborn as sns

sns.set(style='whitegrid', rc={'axes.grid': False})

from itertools import cycle

from sklearn.model\_selection import train\_test\_split,StratifiedKFold

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn import svm

from sklearn.svm import LinearSVC

from sklearn.svm import SVC

from sklearn.neural\_network import MLPClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

from sklearn.ensemble import AdaBoostClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn import model\_selection

from sklearn.model\_selection import train\_test\_split

from collections import Counter

from sklearn.model\_selection import cross\_val\_score

from sklearn import metrics

from sklearn.metrics import roc\_curve, auc, precision\_recall\_curve, f1\_score, roc\_auc\_score

from sklearn.metrics import accuracy\_score, classification\_report

from sklearn.metrics import cohen\_kappa\_score

from sklearn.metrics import confusion\_matrix

from mpl\_toolkits.mplot3d import Axes3D

import matplotlib.pyplot as plt

import numpy as np

from PIL import Image

from io import BytesIO

from scipy.stats import norm,t

from tqdm import tqdm

from sklearn.utils import resample

from sklearn.datasets import make\_classification

from collections import Counter

from matplotlib import pyplot

from numpy import where

from imblearn.over\_sampling import SMOTE

from imblearn.under\_sampling import RandomUnderSampler

from imblearn.pipeline import Pipeline

import shap

from sklearn.preprocessing import StandardScaler

from sklearn.calibration import CalibratedClassifierCV

from sklearn.base import clone

from scipy import interp

from collections import Counter

from matplotlib import pyplot

from numpy import where

from imblearn.over\_sampling import SMOTE

from imblearn.under\_sampling import RandomUnderSampler

from imblearn.pipeline import Pipeline

#Read in ML input

#samples as rows with last column as cohort for classification

df = pd.read\_csv('C:/Users/hmoye/Desktop/sepsis文件夹/5-31/BPD5-31\_25.csv')

#Set up ML variables: x1 is df with feature values; y1 is cohort for classification

X1 = df.drop(['sepsis'],axis=1)

y1 = df.sepsis

X1 = StandardScaler().fit\_transform(X1)

data = pd.DataFrame(X1)

data.describe([0.01,0.05,0.1,0.25,0.5,0.75,0.9,0.99]).T

from sklearn.datasets import make\_classification

#Random oversample with SMOTE

oversample = SMOTE()

X, y = oversample.fit\_resample(X1, y1)

# 初始化设置

shap.initjs()

plt.style.use('seaborn')

plt.rcParams['font.sans-serif'] = ['SimHei']

plt.rcParams['axes.unicode\_minus'] = False

# 配置参数

plt.style.use('seaborn')

seed = 1

models = [

('LR', LogisticRegression(random\_state=seed)),

('RF', RandomForestClassifier(random\_state=seed)),

("SVM", svm.SVC(probability=True,random\_state=seed)),

('DTREE', DecisionTreeClassifier(random\_state=seed)),

('ADB', AdaBoostClassifier(random\_state=seed)),

('NB', GaussianNB()),

('LDA', LinearDiscriminantAnalysis()),

('KNN', KNeighborsClassifier(n\_neighbors=3)),

('GB', GradientBoostingClassifier(random\_state=seed)),

('MLP', MLPClassifier())

]

n\_bootstraps = 1000

def bootstrap\_evaluation():

results = []

roc\_data = {}

mean\_fpr = np.linspace(0, 1, 100)

for name, model in tqdm(models, desc='Models'):

auc\_scores, f1\_scores = [], []

sensitivities, specificities = [], []

tprs = []

for \_ in tqdm(range(n\_bootstraps), desc=name, leave=False):

# Bootstrap抽样

idx = resample(np.arange(len(y)), stratify=y, replace=True)

X\_boot, y\_boot = X[idx], y[idx]

# 训练模型

model.fit(X\_boot, y\_boot)

# 预测完整数据集

if hasattr(model, "predict\_proba"):

y\_proba = model.predict\_proba(X)[:, 1]

else:

y\_proba = model.decision\_function(X)

y\_pred = (y\_proba > 0.5).astype(int)

# 计算指标

auc = roc\_auc\_score(y, y\_proba)

auc\_scores.append(roc\_auc\_score(y, y\_proba))

f1\_scores.append(f1\_score(y, y\_pred))

# 计算混淆矩阵

tn, fp, fn, tp = confusion\_matrix(y, y\_pred).ravel()

sensitivities.append(tp / (tp + fn))

specificities.append(tn / (tn + fp))

# 新增：计算ROC曲线的fpr和tpr

fpr, tpr, \_ = roc\_curve(y, y\_proba)

# 插值到mean\_fpr网格上

interp\_tpr = np.interp(mean\_fpr, fpr, tpr)

interp\_tpr[0] = 0.0 # 初始点设置为0

tprs.append(interp\_tpr)

# 计算统计量

def bootstrap\_stats(values):

mean = np.mean(values)

std = np.std(values)

ci = np.percentile(values, [2.5, 97.5])

return mean, std, ci

auc\_mean, auc\_std, auc\_ci = bootstrap\_stats(auc\_scores)

f1\_mean, f1\_std, f1\_ci = bootstrap\_stats(f1\_scores)

sens\_mean, sens\_std, \_ = bootstrap\_stats(sensitivities)

spec\_mean, spec\_std, \_ = bootstrap\_stats(specificities)

# 存储结果

results.append({

'Model': name,

'AUC Mean': auc\_mean,

'AUC CI': auc\_ci,

'F1 Mean': f1\_mean,

'F1 CI': f1\_ci,

'Sensitivity Mean': sens\_mean,

'Sensitivity Std': sens\_std,

'Specificity Mean': spec\_mean,

'Specificity Std': spec\_std

})

# 新增：保存ROC曲线数据

mean\_tpr = np.mean(tprs, axis=0)

mean\_tpr[-1] = 1.0 # 最后一个点设置为1

std\_tpr = np.std(tprs, axis=0)

tprs\_upper = np.minimum(mean\_tpr + 1.96\*std\_tpr, 1)

tprs\_lower = np.maximum(mean\_tpr - 1.96\*std\_tpr, 0)

roc\_data[name] = {

'mean\_fpr': mean\_fpr,

'mean\_tpr': mean\_tpr,

'tprs\_upper': tprs\_upper,

'tprs\_lower': tprs\_lower,

'auc\_mean': auc\_mean,

'auc\_ci': auc\_ci

}

# 新增：绘制所有模型的ROC曲线

plt.figure(figsize=(10, 8))

plt.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r', label='Chance', alpha=0.8)

colors = plt.cm.tab10(np.linspace(0, 1, 10))

for i,(name, data) in enumerate(roc\_data.items()):

current\_color = colors[i]

plt.plot(data['mean\_fpr'], data['mean\_tpr'], color = current\_color,

label=f'{name} (AUC = {data["auc\_mean"]:.2f} [{data["auc\_ci"][0]:.2f}-{data["auc\_ci"][1]:.2f}])',

lw=2, alpha=0.8)

plt.fill\_between(data['mean\_fpr'], data['tprs\_lower'], data['tprs\_upper'],color = current\_color,

alpha=0.2)

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC Curves of Multiple Models')

plt.legend(loc='lower right')

plt.savefig("output25.pdf", format="pdf", dpi=300, bbox\_inches="tight")

plt.show()

return pd.DataFrame(results)

print("\n正在进行Bootstrap评估...")

boot\_results = bootstrap\_evaluation()

print("\nBootstrap结果：")

print(boot\_results.round(3))

boot\_results.to\_csv('bootstrap\_results4.csv', index=False)

# 初始化设置

shap.initjs()

plt.style.use('seaborn')

plt.rcParams['font.sans-serif'] = ['SimHei']

plt.rcParams['axes.unicode\_minus'] = False

# 训练随机森林分类器

model = RandomForestClassifier(random\_state=1).fit(X,y)

# 创建SHAP解释器并计算SHAP值

explainer = shap.TreeExplainer(model)

shap\_values = explainer.shap\_values(X) # 获取Explanation对象

shap.summary\_plot(shap\_values[1], X.astype("float"),max\_display=36,show=False)

plt.savefig('shap\_rf.pdf')