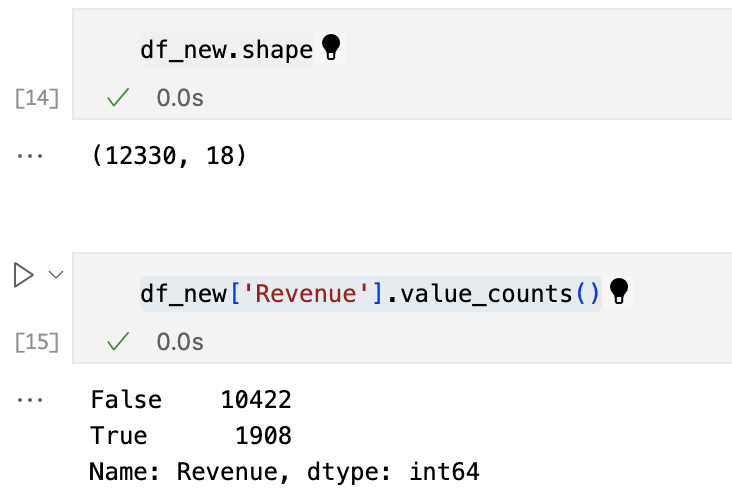
基于UCI网站不均衡数据集Online Shoppers Purchasing Intention Dataset的分析

孙澈20216491

**一、数据来源及数据特征**

1. **数据来源**

本文选取的Online Shoppers Purchasing Intention数据集来自UCI网站<https://archive.ics.uci.edu/dataset/468/online+shoppers+purchasing+intention+dataset> ，内容包括有关网站性能以及顾客在网站上执行的所有活动的信息。数据集包含了在为期12个月的时间内收集的12330个样本，可用于预测顾客在网站上的行为。在这个数据集中，目标变量存在不均衡，负类别（无购物意向）占样本的84.5%（10,422个），而正类别（有购物意向）占样本的15.5%（1908个）。该数据集非平衡率为5.46，即这种类别分布的不均衡性表现为正类别相对较少，而负类别相对较多。



数据集大小以及Revenue两个类别样本数

1. **数据特征**

2.1连续特征

Administrative（管理）：整数，与网站的管理相关的不同页面的访问数。

Informational（信息）：整数，与网站信息以及网站其他有用内容相关的不同页面的访问次数。

ProductRelated（产品相关）：整数，与网站上不同产品相关的不同页面的访问次数。

BounceRate（跳出率）：浮点数，离开网站的用户在主页上的百分比。

ExitRate（退出率）：浮点数，从用户访问的页面上离开的百分比。

Page Values（页面价值）：浮点数，用户在进行交易之前访问的页面的平均价值。

SpecialDay（节日）：浮点数，0表示会话的日期距离节日超过10天，0.1到0.5表示会话的日期在距离节日10天到5天之间，0.6到0.9表示会话的日期在距离节日4天到1天之间，1表示会话的日期是节日当天。

Bounce Rates、Exit Rates和Page Values等特征用于确定网站的性能。跳出率用于显示所有会话中单页面会话的百分比。退出率显示从页面离开的人的百分比。页面价值给出所有访问页面的交易的平均值，这个值均匀分布在所有页面上。

2.2分类特征

Administrative\_Duration（管理时间）：整数，在管理页面上花费的时间（秒）。

0：未在页面上花费时间。

1：在页面上花费不到233秒的时间。

2：在页面上花费超过233秒的时间。

Informational\_Duration（信息时间）：整数，在信息页面上花费的时间（秒）。

0：未在页面上花费时间。

1：在页面上花费更多时间。

ProductRelated\_Duration（产品相关时间）：整数，在与产品相关的页面上花费的时间（秒）。

0：未在页面上花费时间。

1：在页面上花费不到3384秒的时间。

2：在页面上花费超过3384秒的时间。

Browse（浏览器ID）：整数，用户登陆网站的浏览器ID。

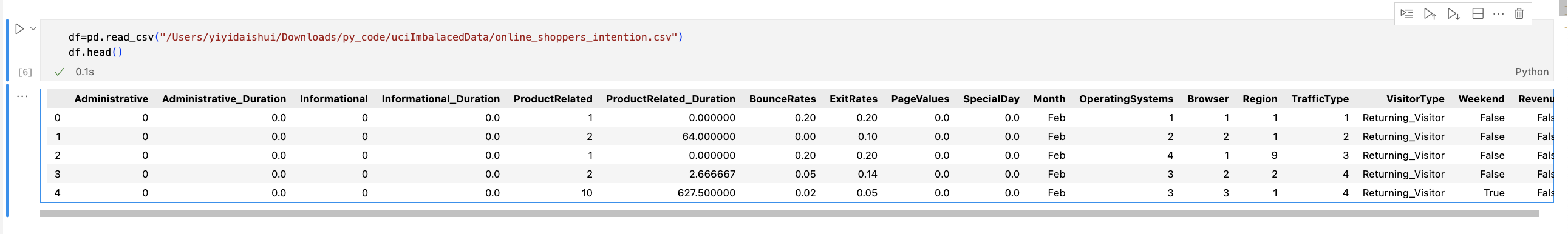
Region（地区ID）：整数，用户登陆网站时所在地区ID。

Traffic\_Type（流量类型ID）：整数，用户登陆网站的不同流量类型来源ID。

User\_Type（用户类型）：字符串，用户是退出网站的用户、新用户还是其他类型的用户。

Revenue（收入）：布尔值，用户是否购买。

Weekend（周末）：布尔值，会话是否发生在周末。



数据集结构

**二、不均衡数据对传统分类器的影响**

在这一部分，我选择以下模型进行评估：

1. 逻辑回归
2. 带RBF内核的支持向量机
3. 随机森林分类器
4. XGBoost分类器

# get models

def get\_models():

ct = get\_transformer()

models = {

"Dummy": make\_pipeline(ct, DummyClassifier()),

"LogisticRegression": make\_pipeline(

ct, LogisticRegression(max\_iter=1500)

), # helps convergence

"SVC": make\_pipeline(ct, SVC(probability=True)),

"RandomForest": make\_pipeline(ct, RandomForestClassifier()),

"XGB": make\_pipeline(

ct, xgb.XGBClassifier(use\_label\_encoder=False, eval\_metric="logloss")

),

}

return models

其中，模型评估的标准和方法为：

1. 执行五折交叉验证，查看平均结果
2. 通过对训练集进行交叉验证的预测来生成和分析混淆矩阵
3. 使用训练集上交叉验证的预测生成并分析召回曲线

# 执行五折交叉验证，查看平均结果

def cross\_validate\_models(

models,

X\_train,

y\_train,

cv=5,

metrics=["accuracy", "precision", "recall", "f1"],

):

results = {}

for name, model in models.items():

print(f"-- CV: {name}")

results[name] = get\_mean\_cv\_scores(

model, X\_train, y\_train, cv=cv, return\_train\_score=True, scoring=metrics

)

results\_df = pd.DataFrame(results)

return results\_df

# 通过对训练集进行交叉验证的预测来生成和分析混淆矩阵

def get\_confusion\_matrices(models, X\_train, y\_train):

# to plot all confusion matrices together

fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(13, 10), dpi=100)

# None zips with DummyClassifier which is not plotted

axes = [None, ax1, ax2, ax3, ax4]

labels = ["No Purchase", "Purchase"]

for (name, model), ax in zip(models.items(), axes):

if name == "Dummy":

continue

else:

y\_pred = cross\_val\_predict(model, X\_train, y\_train)

# creates base confusion matrix plot

ConfusionMatrixDisplay.from\_predictions(

y\_train, y\_pred, ax=ax, colorbar=False, display\_labels=labels

)

# sets the title of the confusion matrix

ax.set\_title(f"{name}")

# sets overall plot title

fig.suptitle("Model Confusion Matrices", y=0.94, fontsize=16)

return fig

# 使用训练集上交叉验证的预测生成并分析召回曲线

def get\_precision\_recall\_curves(models, X\_train, y\_train):

fig, ax = plt.subplots(figsize=(7, 5), dpi=100)

for name, model in models.items():

if name == "Dummy":

continue

else:

y\_pred = cross\_val\_predict(model, X\_train, y\_train, method="predict\_proba")[

:, 1

]

# creates base confusion matrix plot

PrecisionRecallDisplay.from\_predictions(y\_train, y\_pred, ax=ax, name=name)

ax.legend(loc="upper right")

ax.set\_title("Precision recall curves", fontsize=15)

return fig

不均衡数据对模型的影响：

1. 容易过拟合：下述随机森林模型（Random Forest）和 XGBoost 模型（XGB）都严重过度拟合训练集，因为训练集的准确率和精度都非常高。这也说明了集成模型在没有调整超参数的情况下，是比较容易过度拟合训练集的。

2. 准确率陷阱：因为准确率仅仅是正确分类的样本数与总样本数的比例，而不考虑类别分布。所以在不均衡数据中，即使分类器只预测多数类别，也能获得相对较高的准确率。所以，高准确率并不一定表示分类器在处理不均衡数据时是有效的。

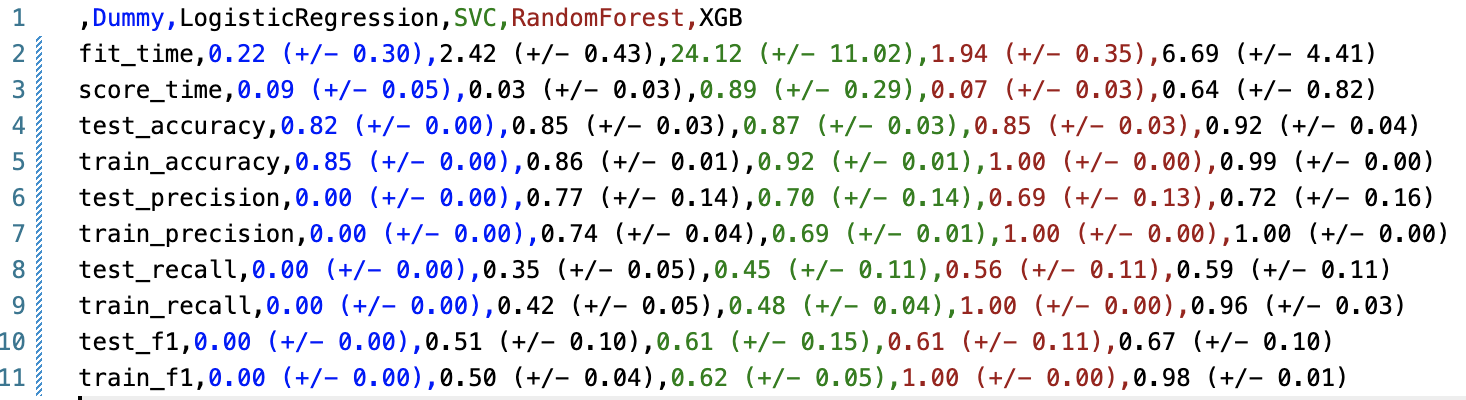
3. 召回率下降：在不均衡数据中，分类器可能会倾向于过度预测多数类别，导致少数类别的召回率下降。少数类别的样本被忽略，模型可能无法捕捉到真正的正例。

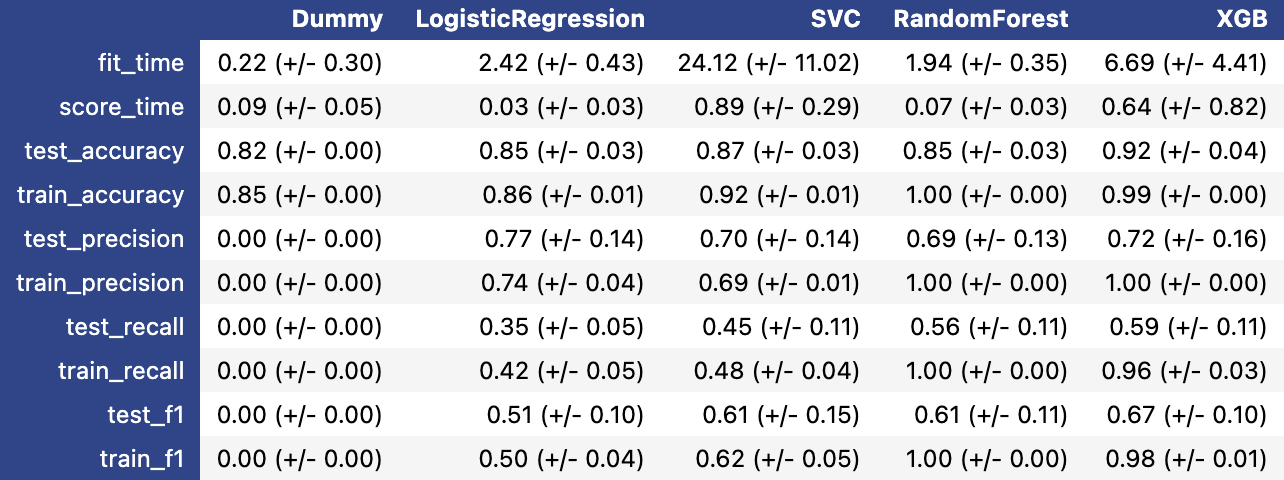
4. 精度不足：不均衡数据中，对于少数类别的精度通常较低。由于模型更倾向于预测为多数类别，所以在预测为少数类别时可能存在大量的误分类。

5. 模型泛化问题：在不均衡数据中，模型可能过于关注多数类别，导致在新的、均衡分布的数据上泛化能力下降。

**三、不同采样算法的效果对比及分析**

模型交叉验证结果（平均）：



模型交叉验证结果

从表中可以看出：

逻辑回归模型（Logistic Regression）在交叉验证期间的验证测试中具有精度最好。然而，该模型的召回分却非常差。

支持向量机（SVC）与逻辑回归模型（Logistic Regression）精度得分比较相近，并且召回分数较好。

随机森林模型（Random Forest）和 XGBoost 模型（XGB）都严重过度拟合训练集，因为训练集的准确率和精度都非常高。这也说明了集成模型在没有调整超参数的情况下，是比较容易过度拟合训练集的。

随机森林（Random Forest）和 XGBoost 模型（XGB）的测试集召回分数高于逻辑回归和支持向量机模型。

模型的F1分数与上述分析一致。

**四、对随机森林模型进行微调**

由上述分析可知，随机森林模型和XGBoost模型在处理该不均衡数据集的问题时表现较好，本部分选择随机森林模型，对该模型进行进一步微调。

根据sklearn文档：

<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

此部分选取部分超参数：

max\_depth、min\_samples\_leaf、min\_samples\_split、n\_estimators调整。

# 创建待调整的模型以及超参数搜索空间

def create\_model\_and\_params():

ct = get\_transformer()

model = make\_pipeline(ct, RandomForestClassifier())

search\_space = {

"randomforestclassifier\_\_n\_estimators": randint(100, 1000),

"randomforestclassifier\_\_criterion": ["gini", "entropy"],

"randomforestclassifier\_\_max\_depth": np.arange(10, 100, 5),

"randomforestclassifier\_\_max\_features": ["auto", "log2"],

"randomforestclassifier\_\_min\_samples\_split": [2, 4, 8],

"randomforestclassifier\_\_min\_samples\_leaf": [1, 2, 4],

"randomforestclassifier\_\_class\_weight": [

"balanced",

"balanced\_subsample",

None,

],

}

return model, search\_space

def perform\_random\_search(

X\_train, y\_train, model, search\_space, n\_iter=100, scoring="recall"

):

random\_search = RandomizedSearchCV(

model,

search\_space,

n\_iter=n\_iter,

scoring=scoring,

n\_jobs=-1,

random\_state=42,

return\_train\_score=True,

)

random\_search.fit(X\_train, y\_train)

return random\_search

def get\_search\_results(random\_search):

results = {

"best\_estimator": random\_search.best\_estimator\_,

"best\_params": random\_search.best\_params\_,

"best\_score": random\_search.best\_score\_,

"cv\_results\_": random\_search.cv\_results\_,

}

return results

def get\_final\_predictions(model, X\_train, y\_train, X\_test, y\_test):

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

y\_pred = model.predict(X\_test)

labels = ["No Purchase", "Purchase"]

cm = ConfusionMatrixDisplay.from\_predictions(y\_test, y\_pred, display\_labels=labels)

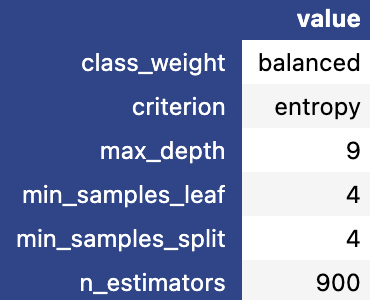
cm.ax\_.set\_title(f"Final Random Forest Confusion Matrix")

cm\_plot = cm.figure\_

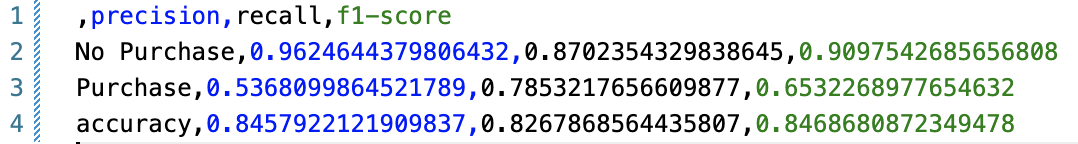
cr = classification\_report(y\_test, y\_pred, target\_names=labels, output\_dict=True)

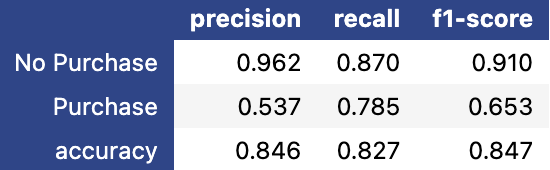
cr\_df = pd.DataFrame(cr).T

return cm\_plot, cr\_df



随机搜索交叉验证结果





调参后随机森林评估

可以看出，我们可以通过调整超参数优化模型处理不均衡数据集的问题。

**五、参考文献**

[1]Sakar, C., and Kasto, Yomi. 2018. “UCI Machine Learning Repository.” University of California, Irvine, School of Information; Computer Sciences. <https://archive-beta.ics.uci.edu/>

[2]Baati, K., Mohsil, M. (2020). Real-Time Prediction of Online Shoppers’ Purchasing Intention Using Random Forest. In: Maglogiannis, I., Iliadis, L., Pimenidis, E. (eds) Artificial Intelligence Applications and Innovations. AIAI 2020. IFIP Advances in Information and Communication Technology, vol 583. Springer, Cham. <https://doi.org/10.1007/978-3-030-49161-1_4>

[3]Sakar, C.O., Polat, S.O., Katircioglu, M., Kastro, Y.: Real-time prediction of online shoppers’ purchasing intention using multilayer perceptron and LSTM recurrent neural networks. Neural Comput. Appl. 31(10), 6893–6908 (2019). <https://doi.org/10.1007/s00521-018-3523-0>