**计算机科学与工程学院**

**数据挖掘**

|  |  |
| --- | --- |
| 专业班级： |  |
|  |  |
| 姓名： |  |
|  |  |
| 学号： |  |
|  |  |
| 提交日期： |  |

1. **任务介绍**

本赛题以银行产品认购预测为背景，想让你来预测下客户是否会购买银行的产品。在和客户沟通的过程中，我们记录了和客户联系的次数，上一次联系的时长，上一次联系的时间间隔，同时在银行系统中我们保存了客户的基本信息，包括：年龄、职业、婚姻、之前是否有违约、是否有房贷等信息，此外我们还统计了当前市场的情况：就业、消费信息、银行同业拆解率等。

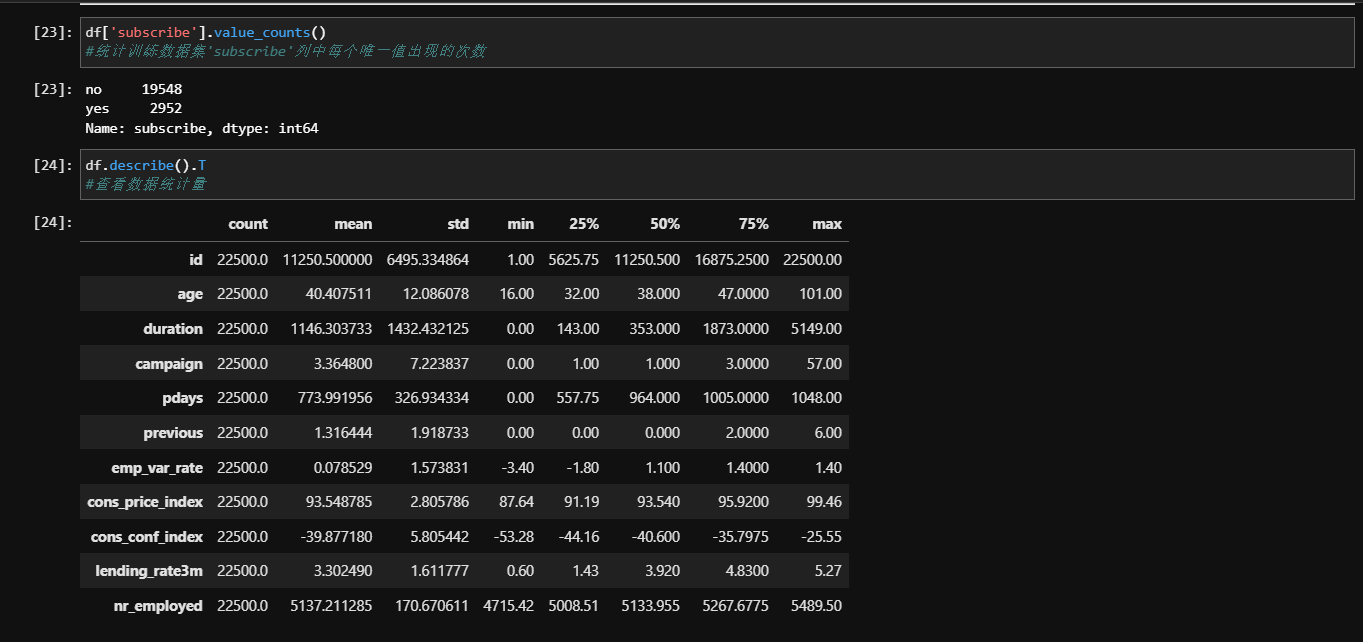
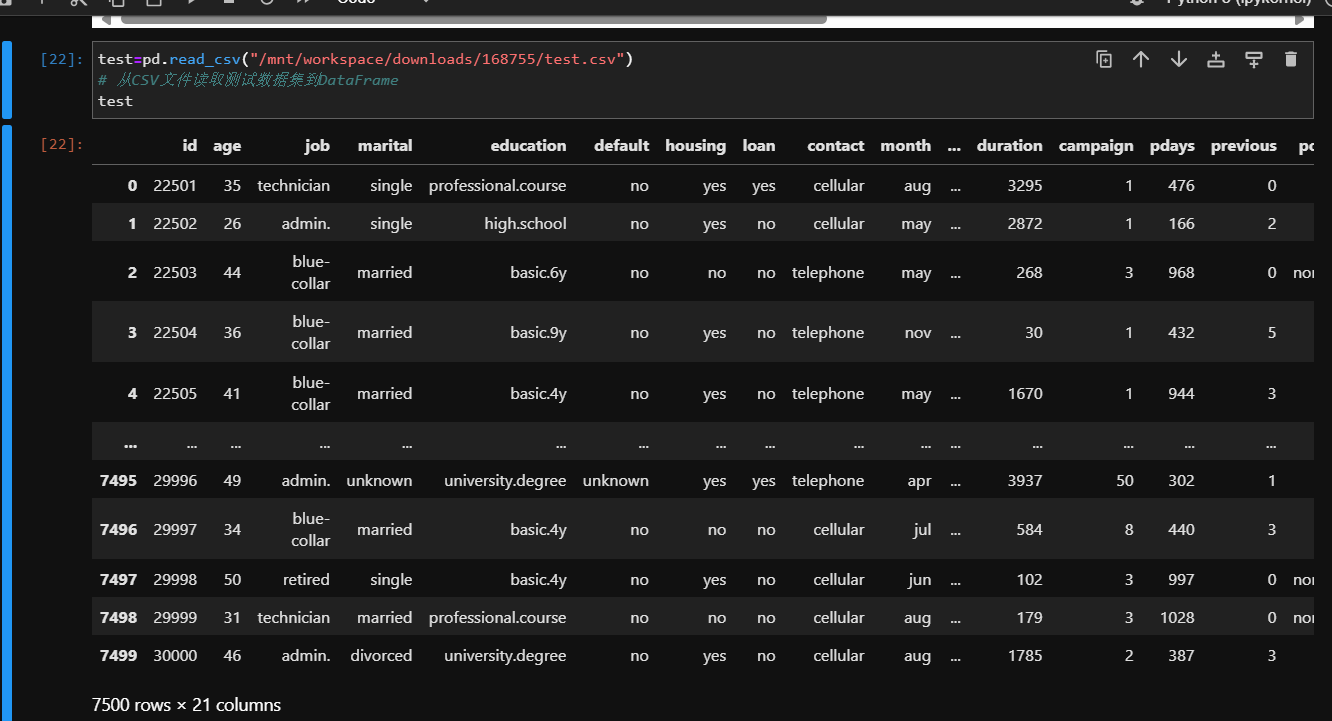
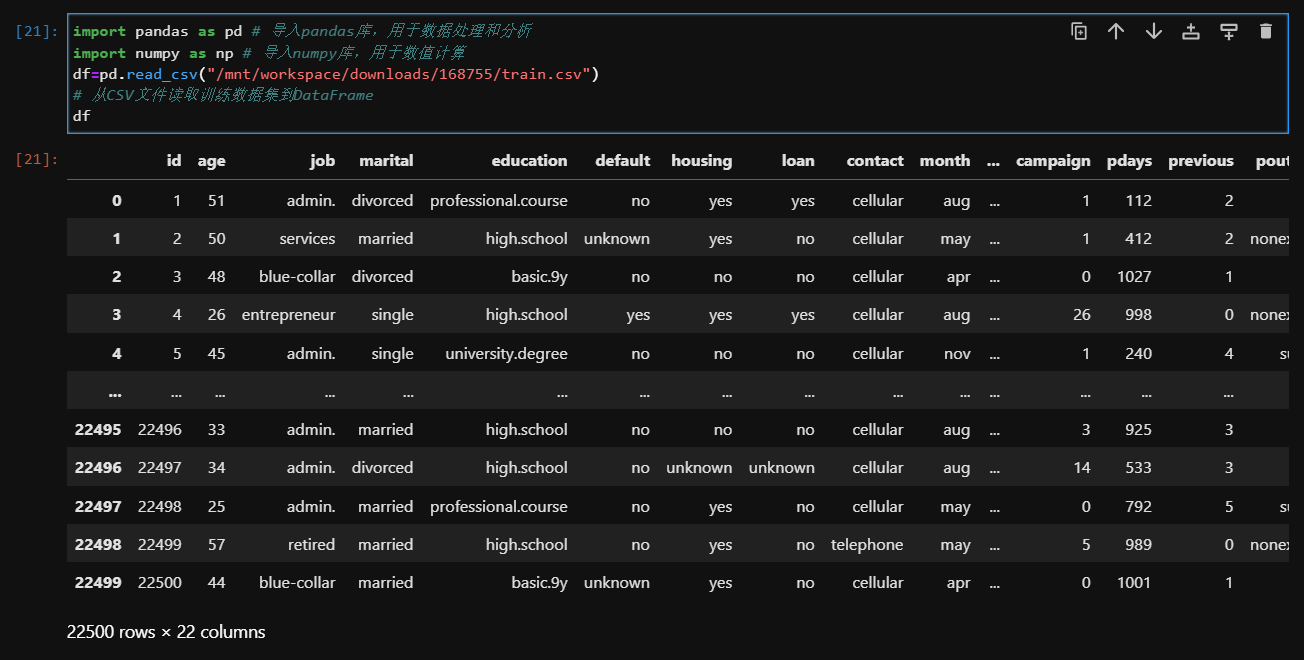
用户购买预测是数字化营销领域中的重要应用场景，通过这道赛题，鼓励学习者利用营销活动信息，为企业提供销售策略，也为消费者提供更适合的商品推荐。

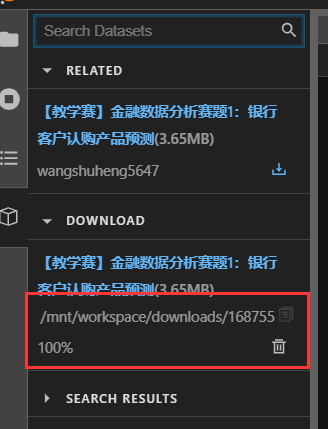
1. **任务详情**

将操作分为如下步骤：加载数据并预览、分离数值变量与分类变量、将计算特征相关性矩阵并绘制热力图、进行特征编码、统计目标变量的数量并绘制饼图、设置参数进行模型训练并保存csv文件、绘制ROC和KS曲线、进行特征输出（绘制图像）。下面将通过实验操作的代码以及过程中的注释来描述完成详情。

**1.加载数据**

这里要注意，记得将数据集下载下来，并且要把你下载的数据集所在位置给清楚，不然开始做的时候会一直卡在这里



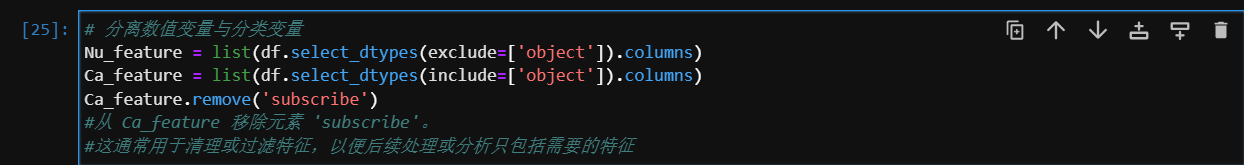
**注意：**

训练集是在这个文件路径下的，一定不要忽视这个路径。

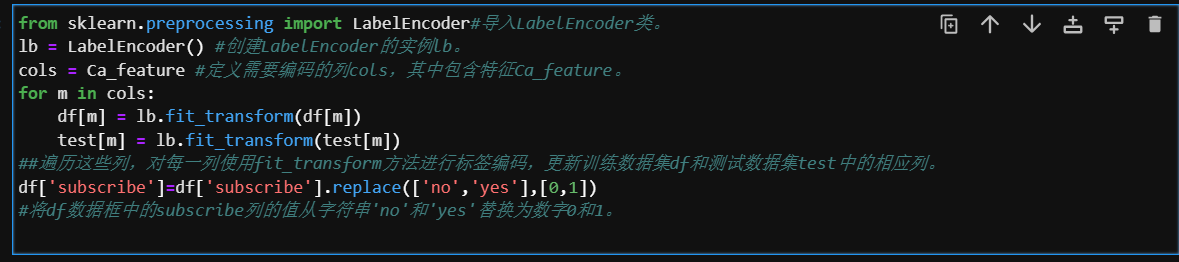
**2.分离数值变量与分类变量**

**#从 Ca\_feature 移除元素 'subscribe'。**

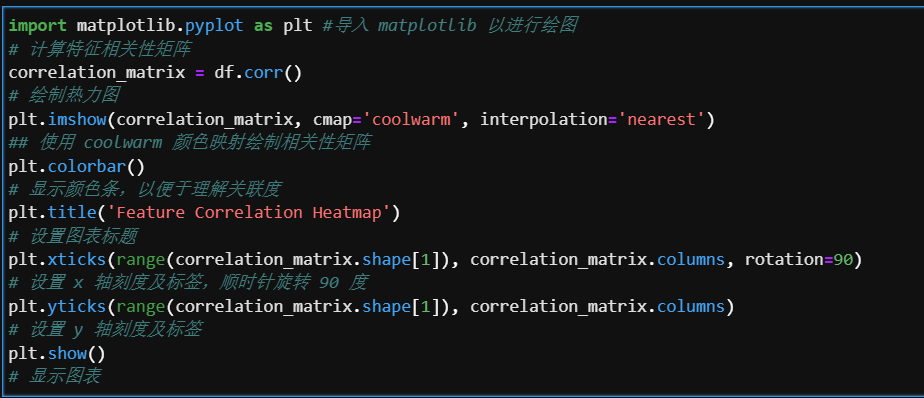
通常用于清理或过滤特征，以便后续处理或分析只包括需要的特征

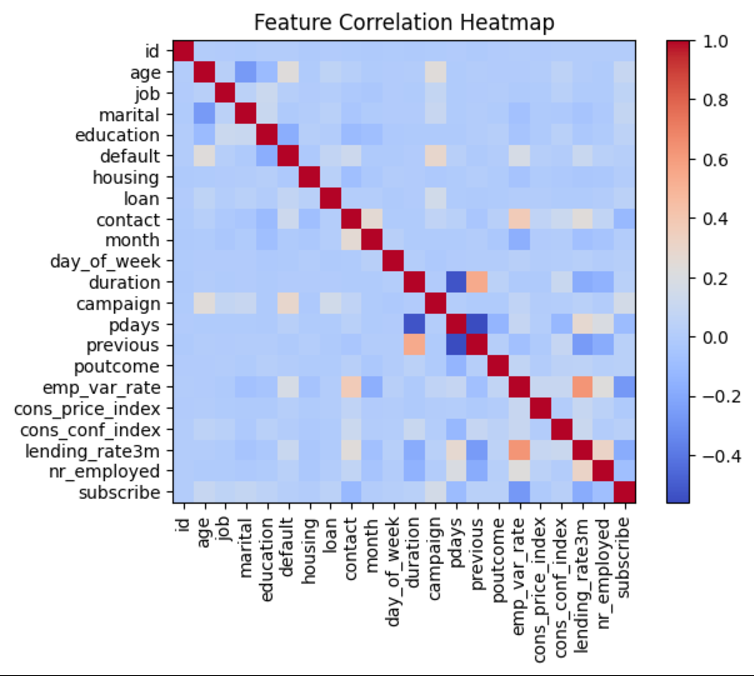


**3.进行特征编码，将类别特征转换为数值特征**

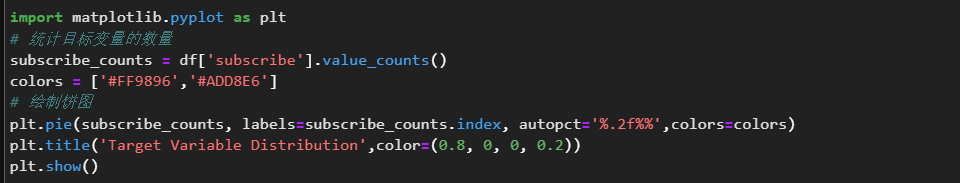
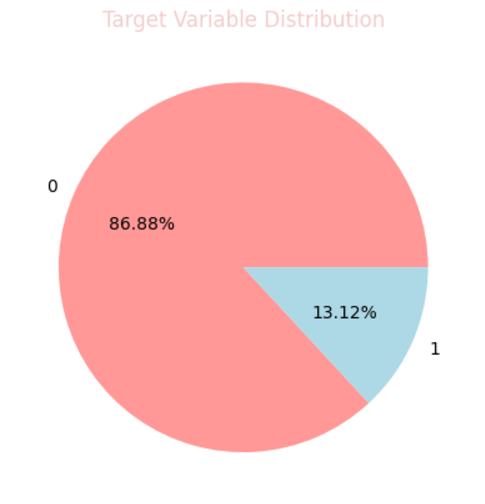


**4.热力图绘制**





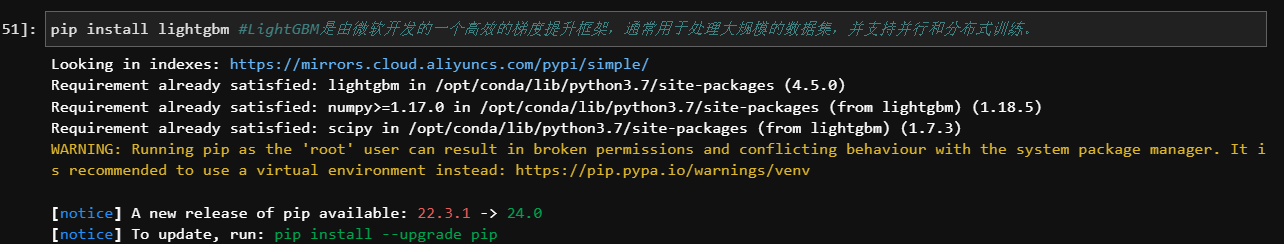
**5.** **统计目标变量的数量并绘制饼图**

在此之前都属于对数据数据进行预处理

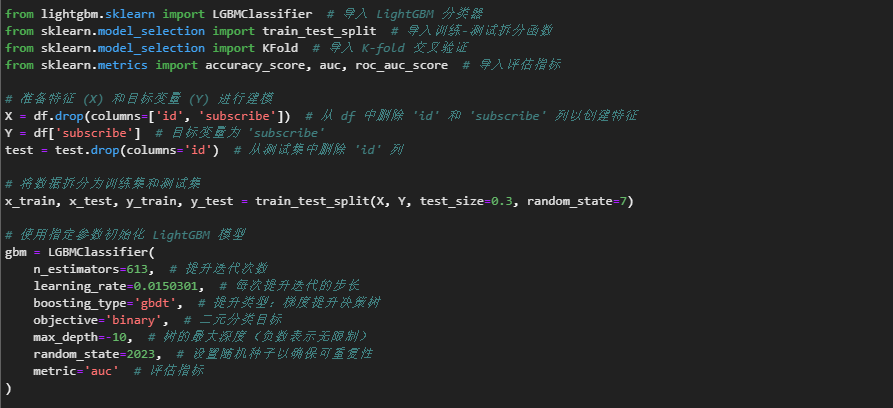
**6.模型训练与roc和ks图片绘制**

导入库：



构建模型：

模型的构建可以有多种选择，结合效果在这里选择了最优的LightGBM：

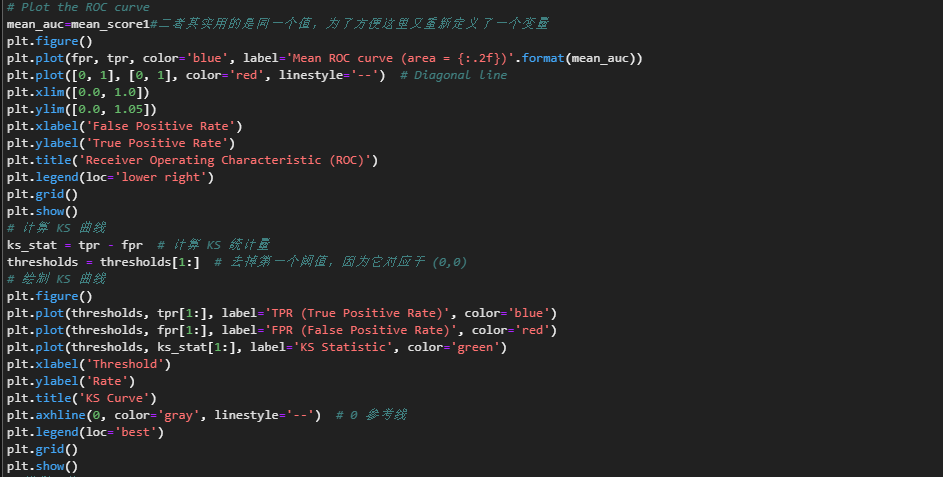


K折交叉验证：

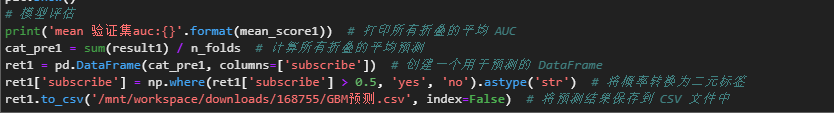


Roc曲线和ks曲线的绘制：

对模型进行评估



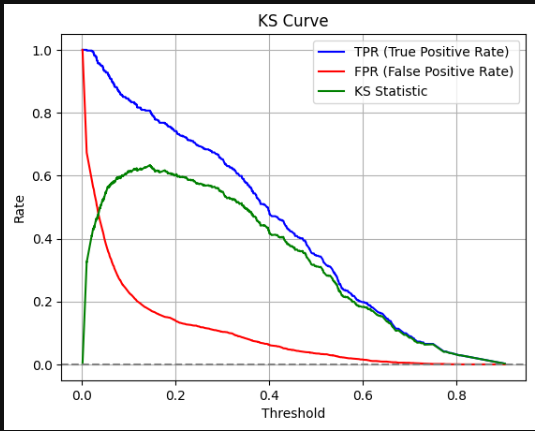
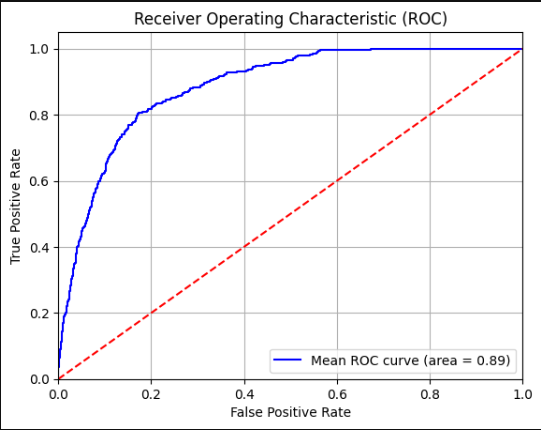
保存csv文件



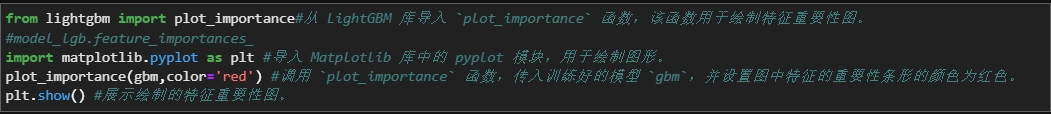
接下来是运行结果：



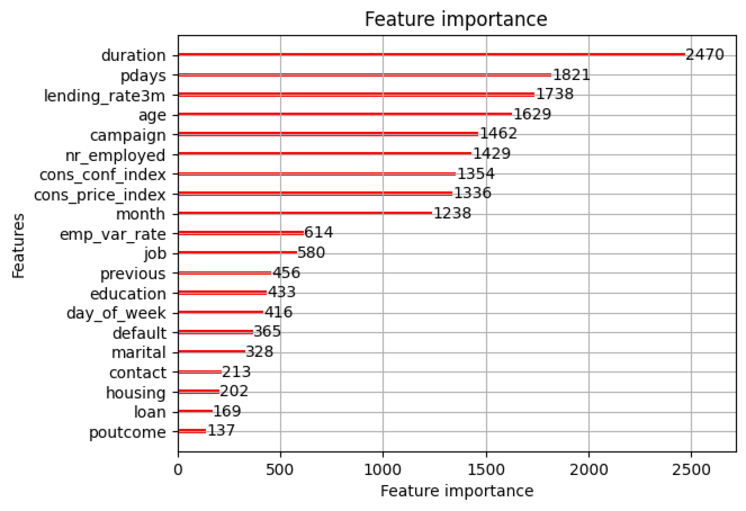
**ROC和ks曲线如下：**



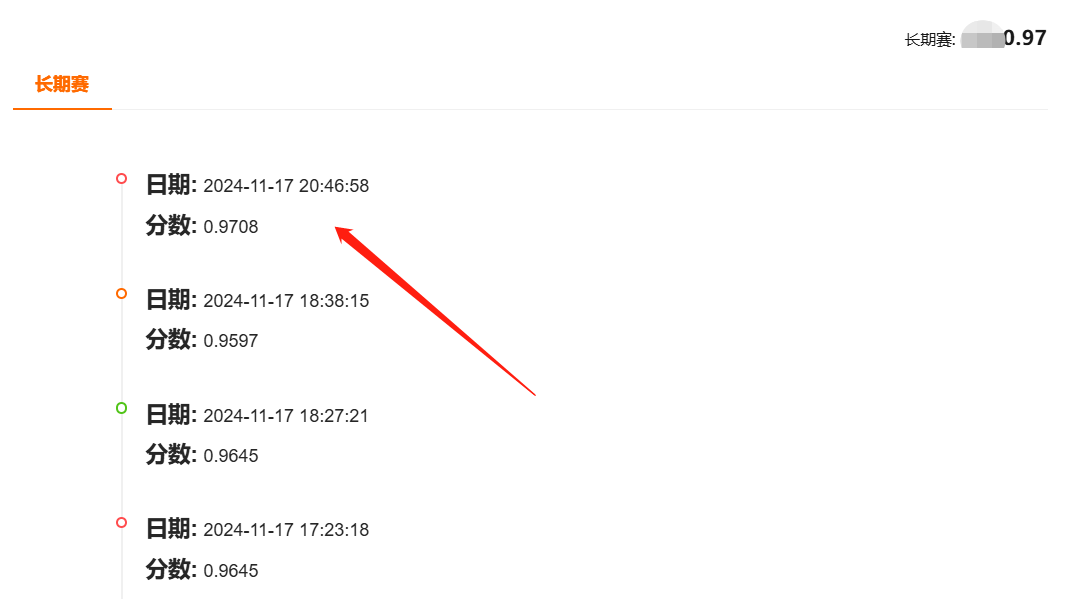
**7.进行特征输出**



图片如下：



最后经过反复的调试与调参，得到了一组最佳的模型参数，都在上述步骤和注释之中，这里就不在赘述了。最终的成绩为0.9708。



全过程的代码如下：

import pandas as pd # 导入pandas库，用于数据处理和分析

import numpy as np # 导入numpy库，用于数值计算

df=pd.read\_csv("/mnt/workspace/downloads/168755/train.csv")

# 从CSV文件读取训练数据集到DataFrame

# 这里地址要改成你自己的地址，下面的test.csv文档也是

df

test=pd.read\_csv("/mnt/workspace/downloads/168755/test.csv")

# 从CSV文件读取测试数据集到DataFrame

test

df['subscribe'].value\_counts()

#统计训练数据集'subscribe'列中每个唯一值出现的次数

df.describe().T

#查看数据统计量

# 分离数值变量与分类变量

Nu\_feature = list(df.select\_dtypes(exclude=['object']).columns)

Ca\_feature = list(df.select\_dtypes(include=['object']).columns)

Ca\_feature.remove('subscribe')

#从 Ca\_feature 移除元素 'subscribe'。

#这通常用于清理或过滤特征，以便后续处理或分析只包括需要的特征

from sklearn.preprocessing import LabelEncoder#导入LabelEncoder类。

lb = LabelEncoder() #创建LabelEncoder的实例lb。

cols = Ca\_feature #定义需要编码的列cols，其中包含特征Ca\_feature。

for m in cols:

    df[m] = lb.fit\_transform(df[m])

    test[m] = lb.fit\_transform(test[m])

##遍历这些列，对每一列使用fit\_transform方法进行标签编码，更新训练数据集df和测试数据集test中的相应列。

df['subscribe']=df['subscribe'].replace(['no','yes'],[0,1])

#将df数据框中的subscribe列的值从字符串'no'和'yes'替换为数字0和1

import matplotlib.pyplot as plt #导入 matplotlib 以进行绘图

# 计算特征相关性矩阵

correlation\_matrix = df.corr()

# 绘制热力图

plt.imshow(correlation\_matrix, cmap='coolwarm', interpolation='nearest')

## 使用 coolwarm 颜色映射绘制相关性矩阵

plt.colorbar()

# 显示颜色条，以便于理解关联度

plt.title('Feature Correlation Heatmap')

# 设置图表标题

plt.xticks(range(correlation\_matrix.shape[1]), correlation\_matrix.columns, rotation=90)

# 设置 x 轴刻度及标签，顺时针旋转 90 度

plt.yticks(range(correlation\_matrix.shape[1]), correlation\_matrix.columns)

# 设置 y 轴刻度及标签

plt.show()

# 显示图表

import matplotlib.pyplot as plt

# 统计目标变量的数量

subscribe\_counts = df['subscribe'].value\_counts()

colors = ['#FF9896','#ADD8E6']

# 绘制饼图

plt.pie(subscribe\_counts, labels=subscribe\_counts.index, autopct='%.2f%%',colors=colors)

plt.title('Target Variable Distribution',color=(0.8, 0, 0, 0.2))

plt.show()

pip install lightgbm #LightGBM是由微软开发的一个高效的梯度提升框架，通常用于处理大规模的数据集，并支持并行和分布式训练。

from lightgbm.sklearn import LGBMClassifier  # 导入 LightGBM 分类器

from sklearn.model\_selection import train\_test\_split  # 导入训练-测试拆分函数

from sklearn.model\_selection import KFold  # 导入 K-fold 交叉验证

from sklearn.metrics import accuracy\_score, auc, roc\_auc\_score  # 导入评估指标

# 准备特征 (X) 和目标变量 (Y) 进行建模

X = df.drop(columns=['id', 'subscribe'])  # 从 df 中删除 'id' 和 'subscribe' 列以创建特征

Y = df['subscribe']  # 目标变量为 'subscribe'

test = test.drop(columns='id')  # 从测试集中删除 'id' 列

# 将数据拆分为训练集和测试集

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.3, random\_state=7)

# 使用指定参数初始化 LightGBM 模型

gbm = LGBMClassifier(

    n\_estimators=613,  # 提升迭代次数

    learning\_rate=0.0150301,  # 每次提升迭代的步长

    boosting\_type='gbdt',  # 提升类型：梯度提升决策树

    objective='binary',  # 二元分类目标

    max\_depth=-10,  # 树的最大深度（负数表示无限制）

    random\_state=2023,  # 设置随机种子以确保可重复性

    metric='auc'  # 评估指标

)

# 初始化 K-fold 交叉验证的变量

result1 = []  # 用于存储每个折叠的预测结果

mean\_score1 = 0  # 用于计算平均 AUC

n\_folds = 7  # 交叉验证的折叠数

kf = KFold(n\_splits=n\_folds, shuffle=True, random\_state=2023)  # 初始化 KFold

mean\_auc=0

# 执行 K-fold 交叉验证

for train\_index, test\_index in kf.split(X):

    x\_train = X.iloc[train\_index]  # 当前折叠的训练特征

    y\_train = Y.iloc[train\_index]  # 当前折叠的训练目标

    x\_test = X.iloc[test\_index]  # 当前折叠的测试特征

    y\_test = Y.iloc[test\_index]  # 当前折叠的测试目标

    gbm.fit(x\_train, y\_train)  # 在训练数据上拟合模型

    y\_pred1 = gbm.predict\_proba(x\_test, num\_iteration=gbm.best\_iteration\_)[:, 1]  # 预测测试数据的概率

    fpr, tpr, thresholds = roc\_curve(y\_test, y\_pred1)

    print('验证集AUC:{}'.format(roc\_auc\_score(y\_test, y\_pred1)))  # 打印当前折叠的 AUC

    mean\_score1 += roc\_auc\_score(y\_test, y\_pred1) / n\_folds  # 累加平均 AUC

    y\_pred\_final1 = gbm.predict\_proba(test, num\_iteration=gbm.best\_iteration\_)[:, 1]  # 预测测试集的概率

    y\_pred\_test1 = y\_pred\_final1  # 存储测试预测结果

    result1.append(y\_pred\_test1)  # 添加到结果列表

# Plot the ROC curve

mean\_auc=mean\_score1#二者其实用的是同一个值，为了方便这里又重新定义了一个变量

plt.figure()

plt.plot(fpr, tpr, color='blue', label='Mean ROC curve (area = {:.2f})'.format(mean\_auc))

plt.plot([0, 1], [0, 1], color='red', linestyle='--')  # Diagonal line

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic (ROC)')

plt.legend(loc='lower right')

plt.grid()

plt.show()

# 计算 KS 曲线

ks\_stat = tpr - fpr  # 计算 KS 统计量

thresholds = thresholds[1:]  # 去掉第一个阈值，因为它对应于 (0,0)

# 绘制 KS 曲线

plt.figure()

plt.plot(thresholds, tpr[1:], label='TPR (True Positive Rate)', color='blue')

plt.plot(thresholds, fpr[1:], label='FPR (False Positive Rate)', color='red')

plt.plot(thresholds, ks\_stat[1:], label='KS Statistic', color='green')

plt.xlabel('Threshold')

plt.ylabel('Rate')

plt.title('KS Curve')

plt.axhline(0, color='gray', linestyle='--')  # 0 参考线

plt.legend(loc='best')

plt.grid()

plt.show()

# 模型评估

print('mean 验证集auc:{}'.format(mean\_score1))  # 打印所有折叠的平均 AUC

cat\_pre1 = sum(result1) / n\_folds  # 计算所有折叠的平均预测

ret1 = pd.DataFrame(cat\_pre1, columns=['subscribe'])  # 创建一个用于预测的 DataFrame

ret1['subscribe'] = np.where(ret1['subscribe'] > 0.5, 'yes', 'no').astype('str')  # 将概率转换为二元标签

ret1.to\_csv('/mnt/workspace/downloads/168755/GBM预测.csv', index=False)  # 将预测结果保存到 CSV 文件中

from lightgbm import plot\_importance#从 LightGBM 库导入 `plot\_importance` 函数，该函数用于绘制特征重要性图。

#model\_lgb.feature\_importances\_

import matplotlib.pyplot as plt #导入 Matplotlib 库中的 pyplot 模块，用于绘制图形。

plot\_importance(gbm,color='red') #调用 `plot\_importance` 函数，传入训练好的模型 `gbm`，并设置图中特征的重要性条形的颜色为红色。

plt.show() #展示绘制的特征重要性图。

1. **任务总结**

在参与天池杯银行客户认购产品预测项目的过程中，我深刻体会到了数据科学在金融领域的实际应用价值。通过对大量银行客户数据的分析和模型构建，我们不仅能够预测客户行为，还能为银行提供个性化服务和产品推荐，从而增强客户满意度和银行竞争力。

在模型选择阶段，我尝试了多种机器学习算法，包括逻辑回归、随机森林、梯度提升树和神经网络等。每种模型都有其优势和局限性，通过交叉验证，我们找到了最适合当前数据集的模型，即为LightGBM。此外，我还学习了如何使用不同的评估指标，如AUC、准确率、召回率和F1分数，来全面评估模型性能。

在本次任务中，我认为对于不同的项目处理上，所使用的算法不能固定的，使用的算法并不总是那一种，可以多测试几个算法，综合一下对负样本进行筛选。并要结合原始数据特征再次筛选。

不同的模型适用的情形不同，对于本次的任务，综合来看LightGBM是最佳选择，例如使用xgboos和catboost等模型之后，可以发现最终的准确率判断效果始终不如前者，最终的成绩只在0.960~0.965这一范围波动，始终得不到0.97以及更高。

模型的参数选择适当对于最终的结果准确率是有明显的提升的，在本任务中，通过对参数的多次调试，最终得出一个合理的参数组合，使得模型拟合达到一个基本理想的状态。

最后，通过参与天池杯银行客户认购产品预测这一比赛项目，我不仅提升了自己在数据挖掘和机器学习领域的专业技能，还增强了解决实际问题的能力。此次对该项目是一个宝贵的学习经历，它让我更加坚信数据科学和深度学习在金融领域的巨大潜力。同时在对数据挖掘这一门课程的学习过程中培养了数据思维意识，数据挖掘技术不仅是一种工具，更是一种思维方式。通过本课程培养了我对数据的敏感性和“数据思维意识”，这对于培养科学的世界观和方法论具有重要意义。除此之外，增强我的国家意识和历史使命感。在一些数据挖掘项目中，我们可以通过分析与国家经济、社会相关的数据，我们可以更好地理解国家、社会、市场的需求，为国家发展做出自己的贡献。总之，本次任务实践使我收获颇多，既学到了数据挖掘和机器学习领域的知识技能，深刻理解了数据科学的应用对社会发展的作用，增强了国家意识和社会责任感，将所学的知识应用到实际生产的应用与预测中，我认为这正是学习知识技能的真正目的所在。