### 14. XGBoost + PCA案例：用户流失风险预测

**问题背景**：某视频平台需预测用户是否会取消会员订阅，用户特征含 50 项（如日均观看时长、互动评论数、付费历史等），存在大量冗余特征（如 “周均观看时长” 与 “日均观看时长” 强相关），导致模型过拟合。

**数据**：

* 10000 名会员的 50 项特征及流失标签（1 = 流失，0 = 留存）。

**要求**：用 PCA 降维至 10 个主成分，输入 XGBoost 模型，对比降维前后的预测精度（AUC 值）和训练时间，分析关键流失因素（如 “连续 3 天未登录”）。

### 14. XGBoost + PCA代码：用户流失风险预测

|  |
| --- |
| import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  from sklearn.decomposition import PCA  from sklearn.model\_selection import train\_test\_split  from sklearn.preprocessing import StandardScaler  from sklearn.metrics import roc\_auc\_score, classification\_report  import xgboost as xgb  # 1. 数据准备（10000名会员的50项特征）  np.random.seed(42)  n\_samples = 10000  n\_features = 50  # 生成高维特征（模拟用户行为：观看时长、评论数、付费次数等）  X = np.random.randn(n\_samples, n\_features)  # 核心特征（与流失强相关）  core\_features = np.random.choice(n\_features, 5, replace=False)  for i, idx in enumerate(core\_features):  if i < 3: # 正向特征（值越高越不易流失）  X[:, idx] = np.random.normal(5, 1, n\_samples)  else: # 负向特征（值越高越易流失）  X[:, idx] = np.random.normal(2, 1, n\_samples)  # 生成流失标签（1=流失，0=留存）  # 核心逻辑：核心特征决定大部分流失概率  流失概率 = 0.5 - 0.1\*X[:, core\_features[0]] - 0.1\*X[:, core\_features[1]] - 0.1\*X[:, core\_features[2]] + \  0.1\*X[:, core\_features[3]] + 0.1\*X[:, core\_features[4]]  流失概率 = np.clip(流失概率, 0, 1)  y = np.where(np.random.rand(n\_samples) < 流失概率, 1, 0)  print(f"用户流失率：{sum(y)/n\_samples:.2%}")  # 2. 数据标准化  scaler = StandardScaler()  X\_scaled = scaler.fit\_transform(X)  # 3. PCA降维  pca = PCA(n\_components=0.95, random\_state=42) # 保留95%信息  X\_pca = pca.fit\_transform(X\_scaled)  print(f"原始特征数：{n\_features}，降维后特征数：{pca.n\_components\_}")  # 4. 划分数据集  X\_train\_raw, X\_test\_raw, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.3, random\_state=42)  X\_train\_pca, X\_test\_pca = train\_test\_split(X\_pca, test\_size=0.3, random\_state=42)  # 5. XGBoost模型训练与评估  def train\_xgboost(X\_train, X\_test, y\_train, y\_test, title):  model = xgb.XGBClassifier(  objective='binary:logistic',  n\_estimators=100,  max\_depth=5,  learning\_rate=0.1,  random\_state=42  )  model.fit(X\_train, y\_train)  y\_pred = model.predict(X\_test)  y\_prob = model.predict\_proba(X\_test)[:, 1]  auc = roc\_auc\_score(y\_test, y\_prob)  print(f"\n{title} AUC：{auc:.4f}")  print("分类报告：")  print(classification\_report(y\_test, y\_pred, target\_names=['留存', '流失']))  return model, auc  # 基于原始特征的XGBoost  model\_raw, auc\_raw = train\_xgboost(X\_train\_raw, X\_test\_raw, y\_train, y\_test, "原始特征XGBoost")  # 基于PCA降维特征的XGBoost  model\_pca, auc\_pca = train\_xgboost(X\_train\_pca, X\_test\_pca, y\_train, y\_test, "PCA降维XGBoost")  # 6. 结果对比  plt.figure(figsize=(8, 5))  plt.bar(['原始特征', 'PCA降维'], [auc\_raw, auc\_pca], color=['blue', 'orange'])  plt.ylim(0.5, 1.0)  plt.ylabel('AUC值（越大越好）')  plt.title('XGBoost模型AUC对比')  plt.grid(axis='y', alpha=0.3)  plt.show()  # 7. 特征重要性分析（降维后模型）  if hasattr(model\_pca, 'feature\_importances\_'):  plt.figure(figsize=(10, 6))  xgb.plot\_importance(model\_pca, max\_num\_features=10, title='PCA降维后特征重要性')  plt.show() |