# 因為這語法是先於colab上執行  
所以需要安裝 optuna 才能做超參數調整優化

| !pip install optuna |
| --- |

# 匯入所需的套件

| import pandas as pd  import numpy as np  from sklearn.model\_selection import train\_test\_split  from sklearn.ensemble import RandomForestClassifier  from sklearn.feature\_selection import RFE, SelectKBest, f\_classif  from sklearn.metrics import confusion\_matrix, accuracy\_score  import optuna |
| --- |

# 1. 載入數據

| train\_url = "https://raw.githubusercontent.com/LonelyCaesar/-Titanic-Survival-Prediction/main/train.csv"  test\_url = "https://raw.githubusercontent.com/LonelyCaesar/-Titanic-Survival-Prediction/main/test.csv"  train = pd.read\_csv(train\_url)  test = pd.read\_csv(test\_url) |
| --- |

# 2. 數據前處理

| # 填補缺失值  train["Age"].fillna(train["Age"].median(), inplace=True)  test["Age"].fillna(test["Age"].median(), inplace=True)  test["Fare"].fillna(test["Fare"].median(), inplace=True)  # 類別特徵編碼  train["Sex"] = train["Sex"].map({"male": 0, "female": 1})  test["Sex"] = test["Sex"].map({"male": 0, "female": 1})  train["Embarked"].fillna("S", inplace=True)  train["Embarked"] = train["Embarked"].map({"C": 0, "Q": 1, "S": 2})  test["Embarked"] = test["Embarked"].map({"C": 0, "Q": 1, "S": 2})  # 選擇特徵  features = ["Pclass", "Sex", "Age", "SibSp", "Parch", "Fare", "Embarked"]  X = train[features]  y = train["Survived"]  X\_test = test[features]  # 分割數據集  **分割數據**：將訓練數據分為訓練集（80%）和驗證集（20%）。  X\_train, X\_valid, y\_train, y\_valid = train\_test\_split(X, y, test\_size=0.2, random\_state=42) |
| --- |

# 3. 特徵選擇

使用 SelectKBest 根據統計測試（f\_classif）篩選出排名前 n\_features 的特徵。

訓練隨機森林模型並計算驗證集的準確率，作為 Optuna 的評估指標。

| # 使用 SelectKBest  skb = SelectKBest(score\_func=f\_classif, k=5)  X\_train\_skb = skb.fit\_transform(X\_train, y\_train)  X\_valid\_skb = skb.transform(X\_valid)  X\_test\_skb = skb.transform(X\_test)  # 使用 RFE  model = RandomForestClassifier(random\_state=42)  rfe = RFE(model, n\_features\_to\_select=5)  X\_train\_rfe = rfe.fit\_transform(X\_train, y\_train)  X\_valid\_rfe = rfe.transform(X\_valid)  X\_test\_rfe = rfe.transform(X\_test) |
| --- |

# 4. Optuna 超參數調優

| def objective(trial):  n\_features = trial.suggest\_int("n\_features", 3, len(features))  model = RandomForestClassifier(random\_state=42)    skb = SelectKBest(score\_func=f\_classif, k=n\_features)  X\_train\_selected = skb.fit\_transform(X\_train, y\_train)  X\_valid\_selected = skb.transform(X\_valid)    model.fit(X\_train\_selected, y\_train)  y\_pred = model.predict(X\_valid\_selected)  return accuracy\_score(y\_valid, y\_pred)  study = optuna.create\_study(direction="maximize")  study.optimize(objective, n\_trials=50)  # 使用最佳特徵數  best\_k = study.best\_params["n\_features"]  skb = SelectKBest(score\_func=f\_classif, k=best\_k)  X\_train\_best = skb.fit\_transform(X\_train, y\_train)  X\_test\_best = skb.transform(X\_test) |
| --- |

# 5. 模型訓練與評估

| model = RandomForestClassifier(random\_state=42)  # 對驗證集和測試集進行相同的特徵選擇處理  X\_valid\_best = skb.transform(X\_valid)  X\_test\_best = skb.transform(X\_test)  # 訓練模型  model.fit(X\_train\_best, y\_train)  # 預測驗證集與測試集  y\_pred\_valid = model.predict(X\_valid\_best)  y\_pred\_test = model.predict(X\_test\_best)  # 混淆矩陣與準確率  conf\_matrix = confusion\_matrix(y\_valid, y\_pred\_valid)  accuracy = accuracy\_score(y\_valid, y\_pred\_valid)  print("混淆矩陣:\n", conf\_matrix)  print("準確率: {:.2f}%".format(accuracy \* 100)) |
| --- |

# 6. 儲存預測結果

| submission = pd.DataFrame({"PassengerId": test["PassengerId"], "Survived": y\_pred\_test})  submission.to\_csv("submission.csv", index=False)  print("結果已儲存至 submission.csv") |
| --- |

準確率: 81.56%

將 submission.csv 上傳至  
https://www.kaggle.com/competitions/titanic/submissions

