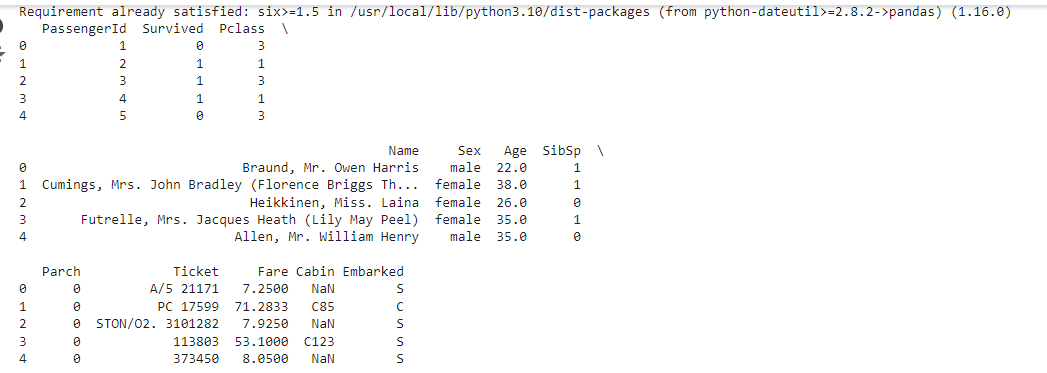
HW4-1

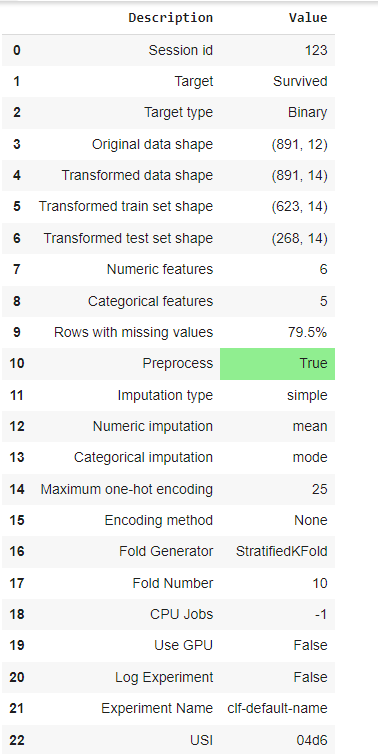
HW4-1

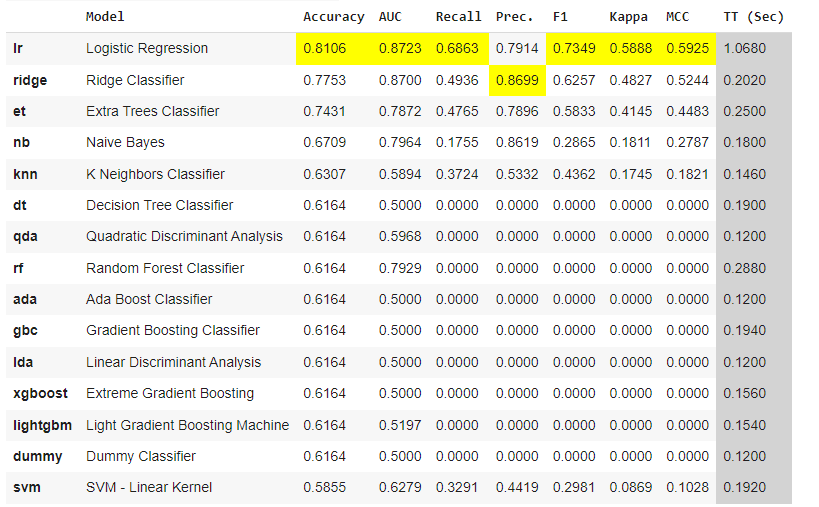
HW4-2

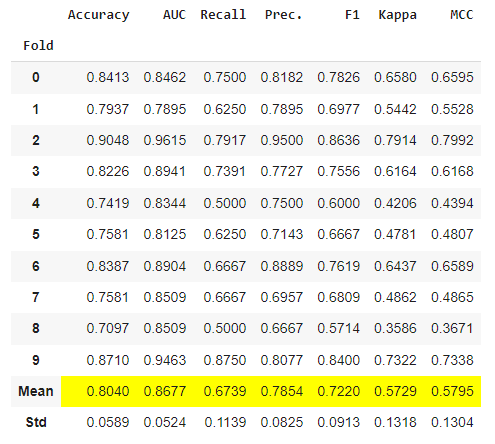
HW4-1 Pycarat to compare ML agorithms on classification problem 16 Model (multi-features titanic is ok , )

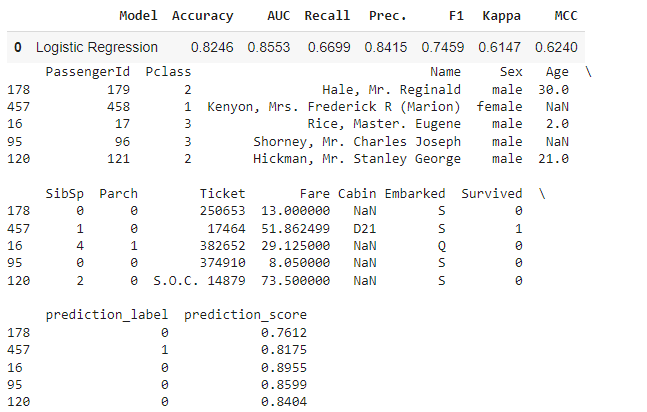
| # 安裝必要的套件  !pip install pycaret  !pip install pandas  # 載入必要的庫  import pandas as pd  from pycaret.classification import \*  # 下載 Titanic 資料集  url = 'https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv'  titanic\_data = pd.read\_csv(url)  # 顯示資料集的前幾行  print(titanic\_data.head())  # 使用 PyCaret 來設置環境  # 設定目標變數 'Survived'，session\_id 用於確保可重現性  clf1 = setup(data=titanic\_data, target='Survived', session\_id=123)  # 比較多個模型，並選擇最佳的模型  best\_model = compare\_models(fold=5)  # 建立一個邏輯回歸模型  lr\_model = create\_model('lr')  # 進行模型預測  predictions = predict\_model(lr\_model)  # 顯示預測結果  print(predictions.head())  # 評估模型效能  evaluate\_model(lr\_model)  # 儲存訓練好的邏輯回歸模型  save\_model(lr\_model, 'titanic\_lr\_model')  # 從檔案加載已儲存的模型  loaded\_model = load\_model('titanic\_lr\_model')  # 使用已訓練的模型進行預測  loaded\_predictions = predict\_model(loaded\_model)  print(loaded\_predictions.head())  # 建立一個 Stack 模型，將多個基礎模型進行集成  # 注意：meta\_model 應該傳入訓練好的模型物件，而不是字串  stacked\_model = stack\_models([lr\_model, best\_model], meta\_model=lr\_model)  # 評估堆疊模型效能  evaluate\_model(stacked\_model)  # 儲存預測結果為 CSV 檔案  stacked\_predictions = predict\_model(stacked\_model)  stacked\_predictions.to\_csv('titanic\_stacked\_predictions.csv', index=False)  # 顯示堆疊模型的預測結果  print(stacked\_predictions.head()) |
| --- |

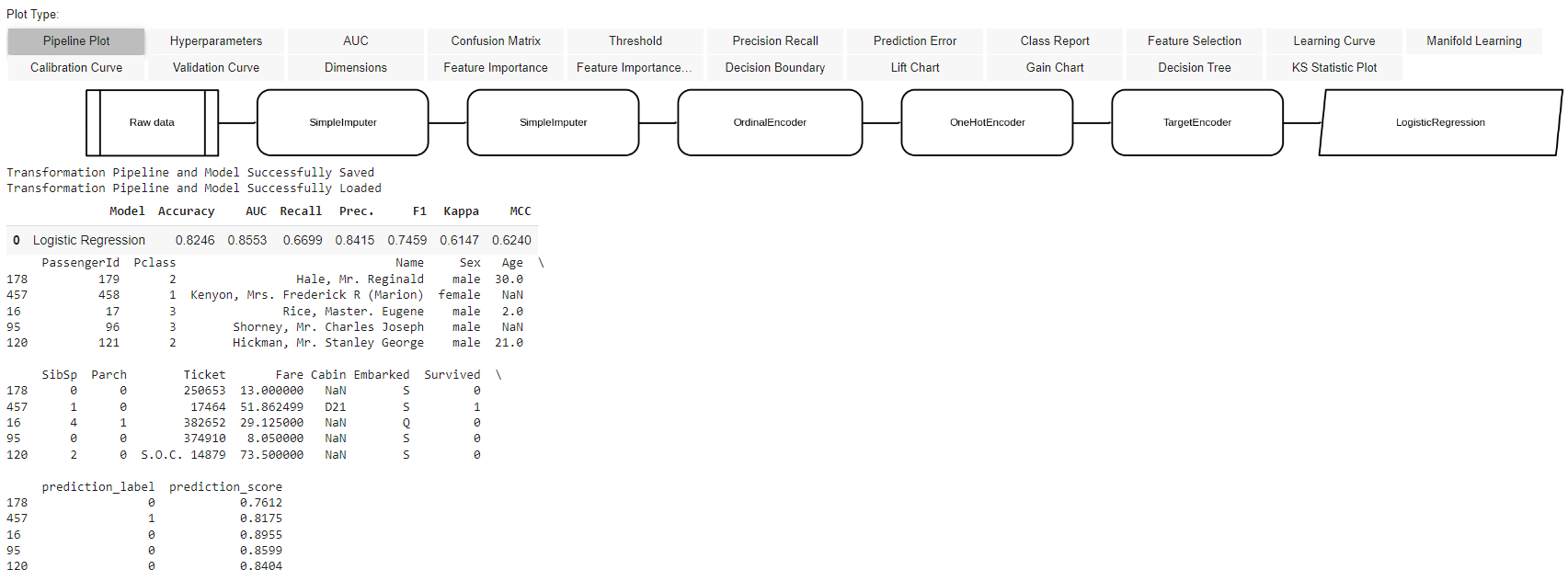


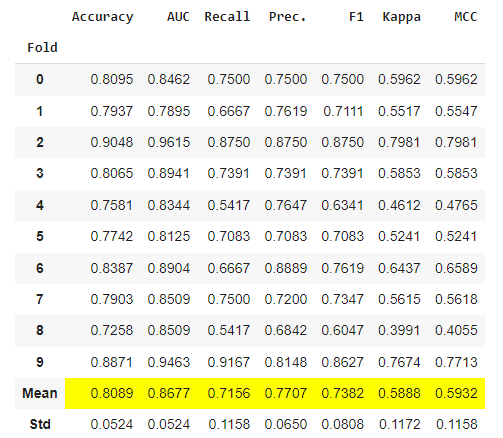


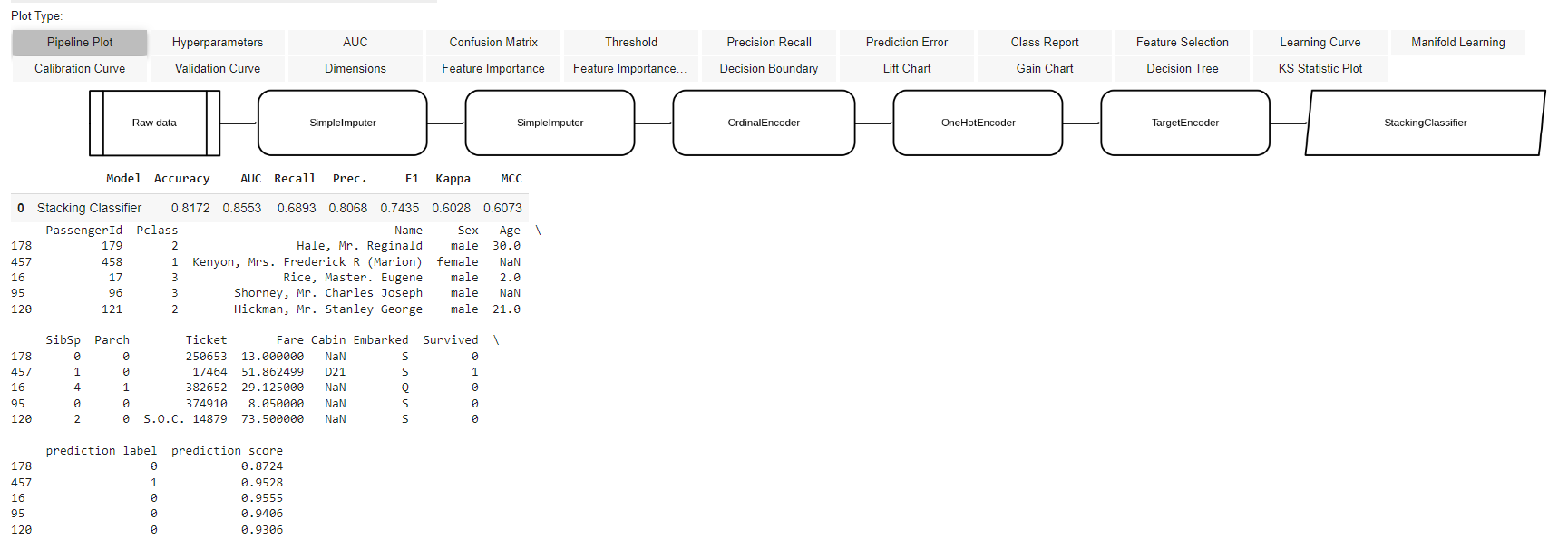












HW4-2

HW4-1

HW4-2

HW4-2 對 model optimization using pycarat or optuna or other AutoML,meta-heuristic

1. Feature engineering, 2. model selection , 3 . training 超參數優化

| !pip install optuna scikit-learn  import pandas as pd  import optuna  from sklearn.datasets import load\_breast\_cancer  from sklearn.model\_selection import train\_test\_split  from sklearn.ensemble import RandomForestClassifier  from sklearn.svm import SVC  from sklearn.linear\_model import LogisticRegression  from sklearn.metrics import accuracy\_score  # 讀取資料集  data = load\_breast\_cancer()  df = pd.DataFrame(data.data, columns=data.feature\_names)  df['target'] = data.target  # 特徵工程  X = df.drop('target', axis=1)  y = df['target']  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  # 模型選擇：訓練並比較不同模型  rf\_model = RandomForestClassifier(random\_state=42)  svm\_model = SVC(random\_state=42)  lr\_model = LogisticRegression(random\_state=42)  models = [rf\_model, svm\_model, lr\_model]  model\_names = ["Random Forest", "SVM", "Logistic Regression"]  for model, name in zip(models, model\_names):  model.fit(X\_train, y\_train)  predictions = model.predict(X\_test)  accuracy = accuracy\_score(y\_test, predictions)  print(f"Model: {name}, Accuracy: {accuracy:.4f}")  # Optuna 超參數優化  def objective(trial):  n\_estimators = trial.suggest\_int('n\_estimators', 50, 200)  max\_depth = trial.suggest\_int('max\_depth', 3, 10)  min\_samples\_split = trial.suggest\_int('min\_samples\_split', 2, 10)  model = RandomForestClassifier(  n\_estimators=n\_estimators,  max\_depth=max\_depth,  min\_samples\_split=min\_samples\_split,  random\_state=42  )  model.fit(X\_train, y\_train)  predictions = model.predict(X\_test)  accuracy = accuracy\_score(y\_test, predictions)  return accuracy  study = optuna.create\_study(direction='maximize')  study.optimize(objective, n\_trials=100)  print("Best parameters: ", study.best\_params)  print("Best accuracy: ", study.best\_value) |
| --- |

Best parameters: {'n\_estimators': 130, 'max\_depth': 10, 'min\_samples\_split': 10}

Best accuracy: 0.9649122807017544