Pima_Dataset

目標:

使用 Isolation Forest 模型找出 Pima 資料集中 5%的離群值並刪除，分別以 Logistic Regreession、KNN、Random Forest、XGB 對該資料集進行分類預測並且比較各別的績效，最後再繪製 ROC 曲線圖比較各分類器的 AUC 面積。

資料集介紹:

該資料集共有 768 筆資料，其中 Outcome =1 的為「有罹患糖尿病」Outcome =0 的為「未罹患糖尿病」，有罹病的筆數為 268 筆，未罹病的筆數為 500 筆，比例約為 1:2。

以下為各變量的解釋:

1. **Pregnancies**: 懷孕次數
2. **Glucose**: 口服葡萄糖耐量試驗中 2 小時後的血糖濃度
3. **BloodPressure**: 舒張壓（毫米汞柱）
4. **SkinThickness**: 三頭肌皮膚褶皺厚度（毫米）
5. **Insulin**: 2 小時血清胰島素（mu U/ml）
6. **BMI**: 體質指數（體重除以身高的平方）
7. **DiabetesPedigreeFunction**: 糖尿病家族函數
8. **Age**: 年齡（年）
9. **Outcome**: 結果變量（0 或 1，表示無糖尿病或有糖尿病）

一、導入資料並刪除離群值

```
df = pd.read_csv('pima.csv')
df.describe()

# 使用孤立森林模型刪除離群值
iforest = IsolationForest(n_estimators=300,
                          max_samples='auto',
                          contamination=0.05,
                          max_features=3,
                          n_jobs=-1,
                          random_state=1)

df_pred = iforest.fit_predict(df)
df_scores = iforest.decision_function(df)
df_anomaly_label = df_pred
pima_outlier = df[df_anomaly_label==-1]

print(pima_outlier.shape)
```

```
In [7]: print(pima_outlier.shape)
(39, 9)
```

#共有 39 筆離群值

二、資料分割

```
X = df_cleaned.iloc[:,:8]
y = df_cleaned["Outcome"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.3, random_state=123)
```

```
In [10]: X_train.shape, X_test.shape
Out[10]: ((510, 8), (219, 8))
```

訓練集: 510 筆，測試集: 219 筆

三、Logistic Regression

```
# parameter grid
param_grid_LR = {
    'C': [0.001, 0.01, 0.1, 1, 10, 100],
    'penalty': ['l1', 'l2'],
    'solver': ['liblinear', 'saga']
}

clf = LogisticRegression()
grid_search = GridSearchCV(clf, param_grid_LR, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)

# Build classifier
LR =LogisticRegression(penalty = 'l1', C = 10, solver = 'liblinear')
LR_pima = LR.fit(X_train, y_train)

# Predict the test subset
test_pred_LR = LR_pima.predict(X_test)
test_conf_LR=pd.crosstab(y_test, test_pred_LR, rownames=['real'],colnames=['pred'])
print(test_conf_LR)
print(classification_report(y_test, test_pred_LR))
```

	precision	recall	f1-score	support
0	0.82	0.87	0.84	143
1	0.73	0.63	0.68	76
accuracy			0.79	219
macro avg	0.77	0.75	0.76	219
weighted avg	0.79	0.79	0.79	219

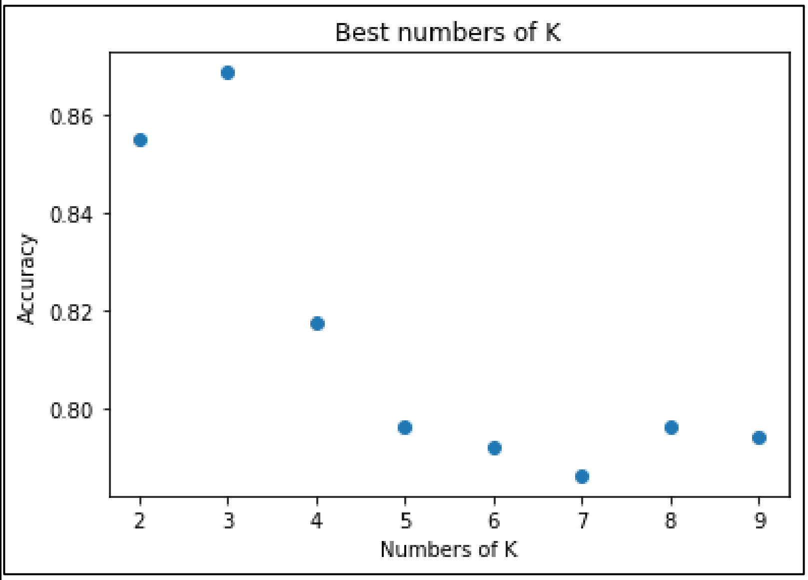
LR 測試集績效

四、K-Nearest Neighbors (KNN)

```
#選擇 K 的個數
k_range = np.arange(2, 10)
accur = []
table=[]

for i in k_range:
    knn = KNeighborsClassifier(n_neighbors = i)
    knn_clf = knn.fit(X_train, y_train)
    test_pred = knn_clf.predict(X_train)
    accu = metrics.accuracy_score(y_train, test_pred)
    accur.append(accu)
table=pd.Series(accur)
print("選擇 K 的個數\n\n",table)

best_k = accur.index(max(accur)) + k_range[0]
print("最佳 K 的個數:",best_k)
plt.scatter(k_range, accur)
plt.show()
```



K=3 時，Accuracy 最高

```
# Build classifier
knn = KNeighborsClassifier(n_neighbors = best_k)
knn_pima = knn.fit(X_train, y_train)

# Predict the test subset
test_pred_KNN = knn_pima.predict(X_test)
test_conf_KNN=pd.crosstab(y_test, test_pred_KNN, rownames=['real'],colnames=['pred'])
print(test_conf_KNN)
print(classification_report(y_test, test_pred_KNN))
```

```
              precision    recall  f1-score   support

           0       0.77      0.81      0.79       143
           1       0.61      0.55      0.58        76

    accuracy                           0.72       219
   macro avg       0.69      0.68      0.69       219
weighted avg       0.72      0.72      0.72       219
```

KNN 測試集績效

五、Random Forest

```
# parameter grid
params_RF = {'n_estimators':list(range(50,151,20)),
             'max_depth':list(range(2,5)),
             'max_features':list(range(2,5)),
             'min_samples_split':list(range(10,31,5)),
             'min_samples_leaf':list(range(5,21,5))}

# Grid Search in RF 使用網格搜尋法尋找超參數
tunnRF = GridSearchCV(RandomForestClassifier(random_state=(123)), params_RF, cv = 5)
tunnRF.fit(X_train,y_train)
print('\nBest parameters :',tunnRF.best_params_)
```

```
Best parameters : {'max_depth': 4, 'max_features': 3, 'min_samples_leaf': 5,
'min_samples_split': 10, 'n_estimators': 50}
```

```
# Build classifier
rf = RandomForestClassifier(criterion = 'gini',
                            n_estimators = tunnRF.best_estimator_.n_estimators,
                            max_depth = tunnRF.best_estimator_.max_depth,
                            max_features = tunnRF.best_estimator_.max_features,
                            min_samples_split = tunnRF.best_estimator_.min_samples_split,
                            min_samples_leaf = tunnRF.best_estimator_.min_samples_leaf)
rf_pima = rf.fit(X_train, y_train)

# Get importance of feature with sorting
pima_imp = pd.DataFrame({'Feature': X_train.columns,
                         'Importance':rf_pima.feature_importances_})
print('\nFeature importance:\n',pima_imp.sort_values(by=['Importance'],ascending=False))
print(classification_report(y_train, train_pred_rf))
```

```
Feature importance:
                    Feature  Importance
1                  Glucose    0.448341
5                      BMI    0.181717
7                      Age    0.168208
6 DiabetesPedigreeFunction    0.072102
4                  Insulin    0.045729
2            BloodPressure    0.036155
0              Pregnancies    0.027116
3            SkinThickness    0.020633
```

觀察變數重要性

```
# Predict the test subset
test_pred_rf = rf_pima.predict(X_test)
test_conf_rf = pd.crosstab(y_test, test_pred_rf ,rownames=['real'],colnames=['pred'])
print('\nConfusion Matrix Training:\n',test_conf_rf)
print(classification_report(y_test, test_pred_rf))
```

```
              precision    recall  f1-score   support

           0       0.76      0.86      0.81       143
           1       0.66      0.50      0.57        76

    accuracy                           0.74       219
   macro avg       0.71      0.68      0.69       219
weighted avg       0.73      0.74      0.73       219
```

#RF 測試集績效

六、eXtreme Gradient Boosting (XGB)

```
# parameter grid
params_XGB = {'n_estimators':list(range(50,151,20)),
              'max_depth':list(range(2,5)),
              'learning_rate':[0.1,0.05,0.01],
              'min_child_weight':[0.1,0.3,0.5],
              'colsample_bytree':[0.5,0.7,0.9]}
tunnXgb = GridSearchCV(XGBClassifier(eval_metric='error', random_state=(124)), params_XGB, cv = 5,
tunnXgb.fit(X_train, y_train)
print('\nBest parameters :',tunnXgb.best_params_)
```

```
Best parameters : {'colsample_bytree': 0.5, 'learning_rate': 0.05,
'max_depth': 2, 'min_child_weight': 0.1, 'n_estimators': 130}
```

```
# Build classifier
XGB = XGBClassifier(n_estimators=tunnXgb.best_estimator_.n_estimators,
                    learning_rate=tunnXgb.best_estimator_.learning_rate,
                    max_depth=tunnXgb.best_estimator_.max_depth,
                    min_child_weight=tunnXgb.best_estimator_.min_child_weight,
                    colsample_bytree=tunnXgb.best_estimator_.colsample_bytree,
                    eval_metric='error')
xgb_pima = XGB.fit(X_train, y_train)

# Get importance of feature with sorting
pima_imp_xgb = pd.DataFrame({'Feature':X_train.columns,'Importance':xgb_pima.feature_importances_})
print('\nFeature importance:\n', pima_imp_xgb.sort_values(by=['Importance'],ascending=False))
```

```
Feature importance:
                    Feature  Importance
1                  Glucose    0.243854
7                      Age    0.154603
5                      BMI    0.152430
4                  Insulin    0.151332
6 DiabetesPedigreeFunction    0.084186
0              Pregnancies    0.076254
3            SkinThickness    0.070898
2            BloodPressure    0.066443
```

觀察變數重要性

```
# Predict the test subset
test_pred_xgb = xgb_pima.predict(X_test)
test_conf_xgb = pd.crosstab(y_test, test_pred_xgb ,rownames=['real'],colnames=['pred'])
print('\nConfusion Matrix Training:\n',test_conf_xgb)
print(classification_report(y_test, test_pred_xgb))
```

```
              precision    recall  f1-score   support

           0       0.78      0.85      0.81       143
           1       0.66      0.55      0.60        76

    accuracy                           0.74       219
   macro avg       0.72      0.70      0.71       219
weighted avg       0.74      0.74      0.74       219
```

XGB 測試集績效

七、比較各模型績效

Model	Accuracy	Recall	Precision	F1 Score
Logistic Regression	0.789954	0.631579	0.727273	0.676056
KNN	0.721461	0.552632	0.608696	0.579310
Random Forest	0.735160	0.500000	0.655172	0.567164
XGB	0.744292	0.552632	0.656250	0.600000

八、ROC curve