41071102H 徐敏皓 HW7

Code:
https://colab.research.google.com/drive/11QgE0Ma3WojrJOGUkvJZHsM_Mf_MvNfJ?usp=sharing

K-Neighbors classification

呼叫方式　　KNeighborsClassifier

```
from sklearn.neighbors import KNeighborsClassifier
# 5. 建立 KNeighborsClassifier 模型
knn = KNeighborsClassifier(n_neighbors=4, weights='uniform', metric='minkowski')

# 6. 訓練模型
knn.fit(X_train_scaled, y_train)
```

KNeighborsRegressor

```
from sklearn.neighbors import KNeighborsRegressor
# 6. 建立 KNeighborsRegressor 模型
knn = KNeighborsRegressor(n_neighbors=4, weights='uniform', metric='minkowski')

# 7. 訓練模型
knn.fit(X_train_scaled, y_train)
```

- n_neighbors: 鄰居數 (預設為 5)
- weights: 距離權重 (可選 'uniform' 或 'distance')
- metric: 距離計算方法 (預設 'minkowski')

此分類器必須將特徵標準化，因為如果data的不同特徵有不同的量級（例如一個特徵範圍是 [0, 1]，另一個是 [0, 1000]），則特徵範圍大的變量會主導距離計算，導致模型偏向該特徵。
此外 Classifier 的 class label 必須為 discrete，而 Regressor 的 class label 必須為 continuous。

Naïve Bayes Classifiers

呼叫方式　　Gaussian Naive Bayes

```
from  sklearn.naive_bayes  import  GaussianNB
#  4.  初始化  GaussianNB  分類器
gnb  =  GaussianNB()

#  5.  訓練模型
gnb.fit(X_train,  y_train)
```

MultinomialNB

```
from  sklearn.naive_bayes  import  MultinomialNB
#  初始化  MultinomialNB
model  =  MultinomialNB(alpha=1.0,  fit_prior=True)

#  訓練模型
model.fit(X_train,  y_train)
```

- alpha: 拉普拉斯平滑參數
- fit_prior: 是否學習先驗分佈

使用 MultinomialNB 時，特徵必須為非負整數，因為 MultinomialNB 假設特徵是計數資料，通常表示為非負整數（例如，單詞的出現次數或其他計數型資料）。

Decision Trees Classification

呼叫方式

DecisionTreeClassifier

```
from sklearn.tree import DecisionTreeClassifier
# 建立 DecisionTreeClassifier 模型
model = DecisionTreeClassifier(criterion='gini', max_depth=5)

# 訓練模型
model.fit(X_train, y_train)
```

DecisionTreeRegressor

```
from sklearn.tree import DecisionTreeRegressor
# 5. 初始化 DecisionTreeRegressor
regressor = DecisionTreeRegressor(criterion='squared_error', max_depth=5)

# 6. 訓練模型
regressor.fit(X_train, y_train)
```

- criterion: 分割標準
- max_depth: 樹的最大深度

Classifier 的 class label 必須為 discrete，而 Regressor 的 class label 必須為 continuous。

SVM Classification

呼叫方式

LinearSVC

```
from sklearn.svm import LinearSVC
# 5. 建立 LinearSVC 模型
model = LinearSVC(C=1.0, max_iter=1000)

# 6. 訓練模型
model.fit(X_train_scaled, y_train)
```

SVC

```
from sklearn.svm import SVC
# 5. 初始化並訓練 SVC 模型
svc_model = SVC(kernel='rbf', C=10, gamma='scale')
svc_model.fit(X_train_scaled, y_train)
```

- C: 正則化參數
- max_iter: 最大迭代次數
- kernal: 核函數
- gamma: 核係數

此分類器必須將特徵標準化，因為如果data的不同特徵有不同的量級（例如一個特徵範圍是 [0, 1]，另一個是 [0, 1000]），則特徵範圍大的變量會主導距離計算，導致模型偏向該特徵。

ANN Classification

呼叫方式

MLPClassifier

```
from sklearn.neural_network import MLPClassifier
# 建立 MLP 模型
mlp = MLPClassifier(hidden_layer_sizes=(100, 50), activation='relu', solver='adam', max_iter=400)

# 訓練模型
mlp.fit(X_train_scaled, y_train)
```

MLPRegressor

```
from sklearn.neural_network import MLPRegressor
# 6. 初始化並訓練 MLPRegressor
mlp = MLPRegressor(hidden_layer_sizes=(100,50), activation='relu', solver='adam', max_iter=600)
mlp.fit(X_train_scaled, y_train)
```

- hidden_layer_sizes: 隱藏層神經元的數量
- activation: 激活函數
- solver: 權重優化算法
- max_iter: 最大迭代次數

此分類器必須將特徵標準化，因為如果data的不同特徵有不同的量級（例如一個特徵範圍是 [0, 1]，另一個是 [0, 1000]），則特徵範圍大的變量會主導距離計算，導致模型偏向該特徵。
此外 Classifier 的 class label 必須為 discrete，而 Regressor 的 class label 必須為 continuous。

Ensemble classifier

呼叫方式　　RandomForestClassifier

```
from sklearn.ensemble import RandomForestClassifier
# 4. 初始化並訓練 RandomForestClassifier
model = RandomForestClassifier(n_estimators=100, criterion='gini', max_depth=None)
model.fit(X_train, y_train)
```

- n_estimators: 隨機森林中樹的數量
- criterion: 分割標準
- max_depth: 樹的最大深度

Ensemble classifier (cont.)

呼叫方式　　GradientBoostingClassifier

```
from sklearn.ensemble import GradientBoostingClassifier
# 建立 GradientBoostingClassifier 模型
clf = GradientBoostingClassifier(n_estimators=100, learning_rate=0.1, max_depth=3)

# 訓練模型
clf.fit(X_train, y_train)
```

GradientBoostingRegressor

```
from sklearn.ensemble import GradientBoostingRegressor
# 建立 GradientBoostingRegressor 模型
model = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1, max_depth=3)

# 訓練模型
model.fit(X_train, y_train)
```

- n_estimators: 弱學習器（決策樹）的數量
- learning_rate: 用於控制每棵樹對最終預測的貢獻
- max_depth: 樹的最大深度

Classifier 的 class label 必須為 discrete，而 Regressor 的 class label 必須為 continuous。

Evaluation

呼叫方式

KFold

```
from sklearn.model_selection import KFold
# 初始化 KFold
kf = KFold(n_splits=5, shuffle=True)
```

ShuffleSplit

```
from sklearn.model_selection import ShuffleSplit
# 創建 ShuffleSplit 交叉驗證物件
ss = ShuffleSplit(n_splits=5, test_size=0.2)
```

- n_splits: 訓練/測試分割的次數
- shuffle: 是否在分割前隨機打亂數據
- test_size: 測試集的比例

執行結果

```
Fold 1
Accuracy: 54.00%
Fold 2
Accuracy: 64.50%
Fold 3
Accuracy: 60.50%
Fold 4
Accuracy: 61.50%
Fold 5
Accuracy: 60.50%

Average Accuracy: 60.20%

Accuracy of round 1: 60.00%
Accuracy of round 2: 60.50%
Accuracy of round 3: 60.50%
Accuracy of round 4: 60.50%
Accuracy of round 5: 64.50%

Average Accuracy: 61.20%
```

	KFold	ShuffleSplit
每次分割data時是否重疊(同一筆data被當作測試集兩次以上)	不會重疊	可能重疊
是否按比例分割	不按比例，但根據K分成K折	按比例分割

Evaluation (cont.)

呼叫方式　　confusion_matrix

```
from sklearn.metrics import confusion_matrix
# 計算混淆矩陣
cm = confusion_matrix(y_true = y_test, y_pred = y_pred)
```

- y_true: 真實label
- y_pred: 預測label

執行結果

```
Confusion Matrix:
[[14  0  1  1  0  0  0  0  1  3]
 [ 0 13  0  0  0  0  0  0  0  0]
 [ 2  1 21  1  0  1  0  0  0  1]
 [ 1  0  2 10  1  0  0  4  1  2]
 [ 1  0  1  3  7  0  0  1  1  1]
 [ 3  3  3  1  0 10  0  0  1  1]
 [ 1  0  0  2  1  0 18  0  0  3]
 [ 0  0  1  0  1  0  0 11  0  0]
 [ 1  2  3  3  5  1  0  0  8  0]
 [ 0  0  5  3  3  0  2  1  2  5]]
```

Confusion matrix對角線代表正確分類的數量，例如cm[0][0] = 14，代表實際是類別0且預測為類別0有14個。

非對角線代表錯誤分類的情況，例如cm[0][2] = 1，代表實際類別是0，但預測為類別2的數量有1個。

Evaluation (cont.)

呼叫方式　classification_report

```
from sklearn.metrics import classification_report
report_dict = classification_report(y_true = y_test, y_pred = y_pred, output_dict=True)
```

- y_true: 實際的label
- y_pred: 預測的label
- output_dict: 輸出是否為字典格式

執行結果

```
blues: {'precision': 0.6086956521739131, 'recall': 0.7, 'f1-score': 0.6511627906976745, 'support': 20.0}
classical: {'precision': 0.6842105263157895, 'recall': 1.0, 'f1-score': 0.8125, 'support': 13.0}
country: {'precision': 0.5675675675675675, 'recall': 0.7777777777777778, 'f1-score': 0.65625, 'support': 27.0}
disco: {'precision': 0.4166666666666667, 'recall': 0.47619047619047616, 'f1-score': 0.4444444444444444, 'support': 21.0}
hiphop: {'precision': 0.3888888888888889, 'recall': 0.4666666666666667, 'f1-score': 0.42424242424242425, 'support': 15.0}
jazz: {'precision': 0.8333333333333334, 'recall': 0.45454545454545453, 'f1-score': 0.5882352941176471, 'support': 22.0}
metal: {'precision': 0.9, 'recall': 0.72, 'f1-score': 0.8, 'support': 25.0}
pop: {'precision': 0.6470588235294118, 'recall': 0.8461538461538461, 'f1-score': 0.7333333333333333, 'support': 13.0}
reggae: {'precision': 0.5714285714285714, 'recall': 0.34782608695652173, 'f1-score': 0.43243243243243246, 'support': 23.0}
rock: {'precision': 0.3125, 'recall': 0.23809523809523808, 'f1-score': 0.2702702702702703, 'support': 21.0}
accuracy: 0.585
macro avg: {'precision': 0.5930350029904142, 'recall': 0.6027255546385981, 'f1-score': 0.5812870989538227, 'support': 200.0}
weighted avg: {'precision': 0.59963381362657, 'recall': 0.585, 'f1-score': 0.575488034682332, 'support': 200.0}
```

前10項代表每個label各自的"precision", "recall", "f1-score", "support"，其中support代表label的樣本數。

"macro avg"是對所有label的 precision, recall, f1-score 計算算術平均數。

"weighted avg"是根據每個label的support對 precision, recall, f1-score 加權後的平均值。

Evaluation (cont.)

呼叫方式　　f1_score

```
from sklearn.metrics import f1_score
f1_macro = f1_score(y_true = y_test, y_pred = y_pred, average='weighted')
```

- y_true: 真實label
- y_pred: 預測label
- average: 計算方式

執行結果

```
F1 Score (macro): 0.58
```

average 設為 macro 是針對所有 label 計算 F1 分數的簡單平均值，
無論每個類別中樣本數量的比例如何。

Evaluation (cont.)

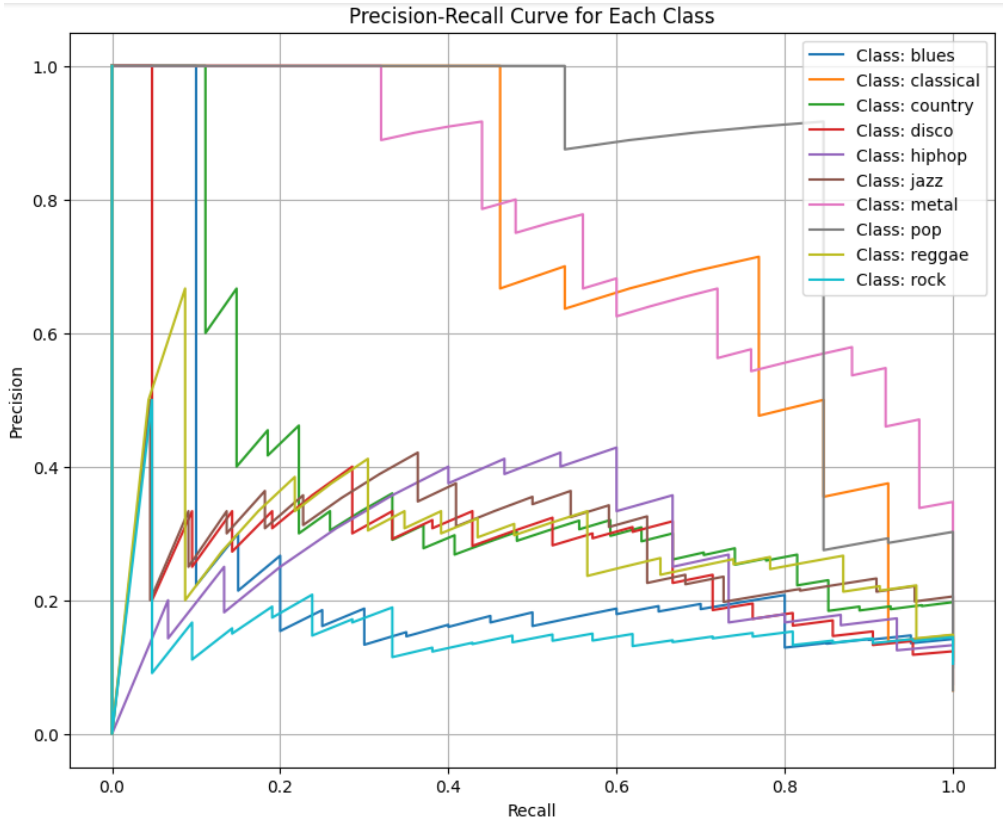
呼叫方式　precision_recall_curve

```
from sklearn.metrics import precision_recall_curve
# 將每個類別的 precision-recall 曲線計算出來
precision, recall, thresholds = {}, {}, {}

for i, class_label in enumerate(label_encoder.classes_):
    precision[class_label], recall[class_label], thresholds[class_label] = precision_recall_curve(y_test == i, y_scores[:, i])
```

- y_true(y_test == i): 真實label
- probas_pred(y_scores[:, i]): 預測的分數或機率

執行結果



- **橫軸 (Recall)**：
表示召回率，代表模型能正確預測出所有實際正樣本的比例。召回率越高，代表模型能抓住更多的正樣本。
- **縱軸 (Precision)**：
表示精確率，代表模型預測為正樣本的結果中，實際正樣本的比例。精確率越高，代表模型的預測結果越準確。
- **每條曲線**：
每條曲線對應一個類別，展示了模型在針對該類別進行分類時的精確率與召回率的權衡。隨著閾值的改變，精確率和召回率會產生變化。
- **解讀曲線形狀**：
- 如果某條曲線在高召回率下仍然保持較高的精確率，代表模型對該類別的分類能力較好。
- 如果曲線趨於平坦或靠近隨機水平，說明模型對該類別的區分能力較差。

分類器分類效果比較(Classification model)

Classification model 使用 “Accuracy”, “Precision”, “Recall”, “F1-score” 做評估。

KNeighborsClassifier

accuracy: 0.585
macro avg: {'precision': 0.5930350029904142, 'recall': 0.6027255546385981, 'f1-score': 0.5812870989538227, 'support': 200.0}
weighted avg: {'precision': 0.59963381362657, 'recall': 0.585, 'f1-score': 0.575488034682332, 'support': 200.0}

Gaussian Naive Bayes

accuracy: 0.35
macro avg: {'precision': 0.3048167061696474, 'recall': 0.37152296495774756, 'f1-score': 0.3048111363957416, 'support': 200.0}
weighted avg: {'precision': 0.3048779178338002, 'recall': 0.35, 'f1-score': 0.2978031144711699, 'support': 200.0}

MultinomialNB

accuracy: 0.315
macro avg: {'precision': 0.2889806903622693, 'recall': 0.3451937096719705, 'f1-score': 0.27446685205901794, 'support': 200.0}
weighted avg: {'precision': 0.2979920967190704, 'recall': 0.315, 'f1-score': 0.2680486257214368, 'support': 200.0}

DecisionTreeClassifier

accuracy: 0.42
macro avg: {'precision': 0.43304933532881956, 'recall': 0.46359948746905266, 'f1-score': 0.4101790669209341, 'support': 200.0}
weighted avg: {'precision': 0.40856048559445635, 'recall': 0.42, 'f1-score': 0.3772463382269081, 'support': 200.0}

LinearSVC

accuracy: 0.61
macro avg: {'precision': 0.6037224454999393, 'recall': 0.6333539294843643, 'f1-score': 0.6069840395672843, 'support': 200.0}
weighted avg: {'precision': 0.6043844537815126, 'recall': 0.61, 'f1-score': 0.5970933494204583, 'support': 200.0}

分類器分類效果比較（Classification model ）（cont.）

Classification model 使用 “Accuracy”, “Precision”, “Recall”, “F1-score” 做評估。

SVC

accuracy: 0.675
macro avg: {'precision': 0.6663565011562723, 'recall': 0.6884442095746444, 'f1-score': 0.6736103003104783, 'support': 200.0}
weighted avg: {'precision': 0.6667889364110761, 'recall': 0.675, 'f1-score': 0.6675487718267463, 'support': 200.0}

MLPClassifier

accuracy: 0.66
macro avg: {'precision': 0.6687988064389261, 'recall': 0.6763489714794062, 'f1-score': 0.6650407986936631, 'support': 200.0}
weighted avg: {'precision': 0.6673196920246401, 'recall': 0.66, 'f1-score': 0.656762589024926, 'support': 200.0}

RandomForestClassifier

accuracy: 0.585
macro avg: {'precision': 0.5853039553512699, 'recall': 0.6066158865289301, 'f1-score': 0.577069786079074, 'support': 200.0}
weighted avg: {'precision': 0.5949359717080306, 'recall': 0.585, 'f1-score': 0.5726398027497098, 'support': 200.0}

GradientBoostingClassifier

accuracy: 0.615
macro avg: {'precision': 0.613439574775892, 'recall': 0.6393145501841153, 'f1-score': 0.6193548387096773, 'support': 200.0}
weighted avg: {'precision': 0.6130066748043091, 'recall': 0.615, 'f1-score': 0.6081433691756272, 'support': 200.0}

分類器分類效果比較(Classification model）（cont.）

Classification model 使用 “Accuracy”, “Precision”, “Recall”, “F1-score” 做評估。

1. Accuracy	SVC (0.675) 和 MLPClassifier (0.66) 的 Accuracy 最高，說明它們在總體分類上性能最好。 MultinomialNB 的 Accuracy 最低 (0.315) ，說明該模型對這個資料集不太適合。
2. Precision	Precision 高的模型（如 SVC: 0.666、MLPClassifier: 0.667）更能準確區分正類別。 MultinomialNB Precision 最低 (0.298)，說明該模型誤報較多。
3. Recall	Recall 高的模型（如 SVC: 0.684、MLPClassifier: 0.676）更能識別正樣本。 MultinomialNB 的 Recall 最低 (0.345)，說明漏報較多。
4. F1-score	F1-score 綜合了 Precision 和 Recall，SVC (0.675)、MLPClassifier (0.665) 的 F1-score 最高，代表它們在精確性和召回率間取得平衡。 MultinomialNB 的 F1-score 最低 (0.274)，表明該模型在精確性和召回率上都表現較差。

分類器分類效果比較(Regression model)

Regression model使用 “Mean Squared Error(MSE)” 做評估。

KNeighborsRegressor	Mean Squared Error: 5.79
DecisionTreeRegressor	Mean Squared Error: 7.27
MLPRegressor	Mean Squared Error: 8.43
GradientBoostingRegressor	Mean Squared Error: 5.48

MSE 越低代表模型預測的誤差越小、越準確。因此從上面的結果可知 GradientBoostingRegressor 和 KNeighborsRegressor 在此資料集表現最佳，預測最準。而 MLPRegressor 對資料集的擬合效果較差，預測誤差較大。

分類器分類效果比較（分類器的優缺）

分類器	優點	缺點	適用情境
KNeighborsClassifier	實現簡單，對非線性問題效果好	計算效率低，對高維數據效果差	小規模數據，低維度，分佈有區域性模式
Gaussian Naive Bayes	訓練速度快，對高維數據有優勢	假設特徵獨立，對複雜關係的數據效果差	標籤有強先驗分佈，特徵是連續數據
MultinomialNB	高效處理離散數據，特別適合文本分類	對於特徵值為連續數據時表現不佳	文本分類（如垃圾郵件檢測）
DecisionTreeClassifier	可解釋性高，對異質數據有良好效果	容易過擬合，需要剪枝或正則化	需要可視化或解釋的分類問題
LinearSVC / SVC	高效處理線性與非線性分類，對小樣本數據表現優異	計算成本高，對噪聲敏感	高維數據，分類邊界清晰或可調的核函數情境
MLPClassifier	強大的非線性建模能力，可解決複雜問題	訓練時間長，對超參數敏感	大量數據，特徵與標籤關係複雜
RandomForestClassifier	高效，對異常值和多維數據適應性強	模型複雜，難以解釋	需要高準確度，對解釋性要求不高的場景
GradientBoostingClassifier	高效，對不平衡數據和異常值敏感，適合應對精細的分類問題	訓練時間長，對參數敏感	對性能要求高，數據分布異常，且特徵間關係複雜的情境