ScikitLearn 操作記錄單 2

組別: ___team 15_____　　學號: ___41147046S_____　　姓名: ____楊子萱_____

Supervised Learning

1. 請根據以下教學資源操作: http://www.cse.msu.edu/~ptan/dmbook/tutorials/tutorial6/tutorial6.html
2. 請自行查詢了解下列 scikit-learn 模組的功能作用 https://scikit-learn.org/stable/

程式碼連結: https://docs.google.com/document/d/1m2JA2Qz0aGmkDo5zdG1GrYfHjv5bPhQHTHm0j2t8FYs/edit?usp=sharing

因為程式碼較不同，有分成兩半，下面那半專門放 classification: evaluate 的程式

連結點入為 google document 且功能皆已被註解，若要執行程式，需複製貼上到.py，並且把被註解掉的功能解開執行

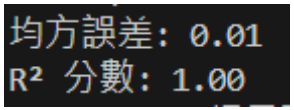
Classification	Module	Function	試寫程式，實驗該函式所提供功能及主要參數設定效果
K-Neighbors classification	sklearn.neighbors	KNighborsClassifier ()	KNeighborsClassifier(n_neighbors=5) n_neighbors 的選擇： • n_neighbors 是一個非常重要的超參數，通常需要通過交叉驗證來選擇最佳值。若 K 太小，模型可能會對噪音過度擬合；若 K 太大，模型則可能過於簡單，無法捕捉到數據的複雜模式。 weights 的選擇： • 當選擇 weights='uniform' 時，所有鄰居對分類的影響是相等的。若選擇 weights='distance'，則距離較近的鄰居會對分類結果有更大影響，這通常有助於提高分類效果，特別是當數據中存在不均勻分佈的情況時。 距離度量（metric）： • 默認使用歐氏距離（euclidean），但有時根據特定的問題或數據，使用其他距離度量（例如曼

			哈頓距離或馬氏距離）可能會提升模型表現。 參數: 輸入要分成幾群 分 5 群的結果: 模型準確率: 0.81 分 3 群的結果: 模型準確率: 0.79
	sklearn.neighbors	KNighborsRegressor ()	KNeighborsRegressor(n_neighbors=5) 回歸效果比較: 跟上述 KNighborsClassifier ()相同 參數: 輸入要分幾群做回歸預測 KNeighborsRegressor 模型均方誤差: 0.43 KNeighborsRegressor 模型 R² 分數: 0.95
Naïve Bayes Classifiers	sklearn.naive_bayes	Gaussian Naive Bayes()	GaussianNB(var_smoothing=1e-9) 與 K-Nearest Neighbors (KNN)： • KNN 是基於距離度量進行分類的，而 GaussianNB 是基於概率的分類方法。KNN 沒有假設特徵之間的獨立性，但會受到維度的影響，對大數據集比較慢。GaussianNB 需要特徵符合高斯分佈，但通常在數據較小或較簡單時運行更快，並且效果較好。 與 Decision Trees 或 Random Forests： • 決策樹和隨機森林是基於數據特徵進行劃分的非線性模型，而 GaussianNB 假設數據符合高斯分佈，且特徵之間是條件獨立的。這使得高斯在特徵相關性強或數據呈現複雜模式時可能無

			法表現很好。 與 Logistic Regression： • 邏輯回歸是一種基於線性假設的分類模型，適用於線性可分的數據。GaussianNB 假設每個特徵的分佈是高斯分佈，適用於特徵呈正態分佈的情況。 參數: var_smoothing=1e-9：為每個特徵的變異數添加微小的平滑因子，防止當特徵的變異數過小時計算不穩定。 priors=None：使用自動計算的類別先驗概率。 
	sklearn.naive_bayes	MultinomialNB()	回歸效果比較: 跟上述 Gaussian Naive Bayes()相同 參數: alpha=1.0：拉普拉斯平滑，防止特徵在某些類別中為零的情況，默認設為 1.0。 fit_prior=True：使用訓練集中的類別分佈作為類別先驗概率。*須先把負值標準化後才能使用 
Decision Trees Classification	sklearn.tree	DecisionTreeClassifier()	DecisionTreeClassifier(criterion="gini", max_depth=5, random_state=42) **與 K-Nearest Neighbors (KNN)**： • KNN 基於距離的度量，適合處理非線性邊界的數據，但在高維數據中會遇到維度災難。而決策樹在處理具有明確層級關係的數據時表現較好。 **與 Random Forest**：

			• 隨機森林是多棵決策樹的集成方法，它通常能提供比單棵決策樹更穩定和準確的結果。決策樹容易過擬合，而隨機森林則通過集成多棵樹來減少這種過擬合。 **與 Logistic Regression**： • 邏輯回歸假設數據是線性可分的，而決策樹則能處理非線性問題。對於非線性數據，決策樹的表現通常優於邏輯回歸。 參數: **criterion**：選擇決策樹分裂的準則。可以選擇 "gini"或 "entropy"。默認值是 "gini"。 **max_depth**：樹的最大深度。控制決策樹的複雜度，防止過擬合。默認值為 None，表示直到所有葉節點都為純淨或樣本數量少於 min_samples_split 時停止。 **min_samples_split**：分裂內部節點所需的最小樣本數。默認是 2，可以設為較大值來控制過擬合。 **min_samples_leaf**：每個葉節點最小的樣本數。增加這個值可以避免過擬合，尤其是在數據量較少的情況下。 **max_features**：決策樹每次分裂時考慮的最大特徵數量。 **random_state**：隨機數種子，保證結果可重現。 **class_weight**：每個類別的權重。默認為 None，這表示每個類別的權重相等 模型準確率: 0.98

	sklearn.tree	DecisionTreeRegressor()	DecisionTreeRegressor(criterion="mse", max_depth=5, random_state=42) **與線性回歸**： • 線性回歸假設特徵和目標之間存在線性關係，而決策樹能夠處理非線性關係。因此，決策樹在複雜、非線性的數據上比線性回歸有優勢。 **與 K-Nearest Neighbors (KNN)**： • KNN 是一種基於距離的算法，它會根據鄰近點的類型來進行預測，適合處理有較多噪音的數據，而決策樹通過學習數據的層次結構來進行分裂。兩者的區別在於 KNN 需要存儲所有訓練數據，而決策樹則是在訓練階段構建一個結構化模型。 **與隨機森林 (Random Forest)**： • 隨機森林是多棵決策樹的集成方法，通常能提供比單棵決策樹更穩定和準確的結果。隨機森林通過集成多棵樹來減少過擬合，尤其是在高維數據中。相比之下，單棵 DecisionTreeRegressor 很容易過擬合。 **與支持向量回歸 (SVR)**： • 支持向量回歸可以處理高維、非線性問題，並且能夠通過核技巧處理更為複雜的數據。相比之下，DecisionTreeRegressor 更加直觀，並且可以很好地處理具有層次結構的數據，但在高維數據上可能不如 SVR。 參數: **criterion**：評估分裂質量的準則。可以選擇 "mse"（均

			方誤差，默認）或 "mae"（平均絕對誤差）。mse 是常用的選擇，因為它對大誤差更加敏感。 **max_depth**：決策樹的最大深度，防止過擬合。 **min_samples_split**：每個節點分裂所需的最小樣本數，減少過擬合。 **min_samples_leaf**：每個葉節點的最小樣本數。 **max_features**：每次分裂時考慮的最大特徵數量。 **random_state**：隨機數種子，用來保證結果可重現。 
SVM Classification	Sklearn.svm	LinearSVC()	LinearSVC(C=1.0, max_iter=1000, penalty='l2', dual=False, tol=1e-4) 主要用於線性可分問題，並且表現通常與其他線性分類器（如 LogisticRegression）相似。當數據集的特徵數量比較大時，LinearSVC 可以比其他基於 SVM 的分類器（例如 SVC）更高效。LinearSVC 對於稀疏數據也有很好的效果。 與 KNeighborsClassifier 或 RandomForestClassifier 等其他非線性模型相比，LinearSVC 在處理大量特徵時的速度更快，但如果數據是高度非線性可分的，則其分類效果可能會比隨機森林等更差。因此，選擇 LinearSVC 的時候要考慮數據的特性和維度。 參數: **C**：正則化參數，控制對錯誤的懲罰。較大的 C 值會對錯誤分類的懲罰更大，有助於在訓練數據上獲得更

			準確的結果，但可能會導致過擬合。較小的 C 值會允許更多的錯誤，但模型會更具泛化能力。 **max_iter**：最大迭代次數，默認為 1000。如果迭代次數達到上限但尚未收斂，會拋出警告。 **penalty**：正則化方式，通常使用 'l2'（L2 正則化）來防止過擬合，也可以使用 'l1' 來進行稀疏化處理。 **dual**：是否使用對偶問題的解法，默認為 True。在特徵數量多於樣本數量時設置為 False 會更有效率。 **tol**：收斂容忍度，用於控制訓練過程中的收斂閾值 
	Sklearn.svm	SVC()	SVC(C=1.0, kernel='rbf', gamma='scale', degree=3, probability=True) 與 **KNeighborsClassifier**：SVC 在處理非線性可分數據時表現較好，尤其是當數據包含複雜的邊界時。相比之下，KNeighborsClassifier 更適合於基於距離的簡單分類問題，但在數據量較大時可能效率較低。 與 **RandomForestClassifier**：隨機森林對於大量特徵和數據能夠進行較好的處理，並且較不容易過擬合。而 SVC 在處理線性和非線性邊界的問題時可能會表現更好，但需要較長的訓練時間。 與 **LogisticRegression**：SVC 和邏輯回歸都可以解決線性可分問題，但 SVC 在面對複雜邊界時能夠表現更好。 參數: **C**：正則化參數，控制對錯誤的懲罰。較大的 C 值會使模型過於擬合訓練數據，較小的 C 值會使模型更具

			泛化能力。 **kernel**：核函數，SVC 可以使用不同的核函數來將數據映射到更高維度，使得在原始空間中非線性可分的數據在高維空間中變為線性可分。常見的核函數有： • 'linear'：線性核 • 'poly'：多項式核 • 'rbf'：徑向基核（默認核） • 'sigmoid'：Sigmoid 核 **degree**：如果使用多項式核，則此參數指定多項式的階數，默認為 3。 **gamma**：核函數的參數，控制每個訓練樣本的影響範圍，較大的值會使得模型擬合更多的樣本，較小的值會使模型更為簡單。 **coef0**：核函數中的常數項，影響多項式核和 Sigmoid 核的學習。 **probability**：如果設置為 True，則會啟用概率估計，這需要在訓練階段中進行額外的計算。 
ANN Classification	Sklearn. neural_network	MLPClassifier()	MLPClassifier(hidden_layer_sizes=(100, 50), activation='relu', solver='adam', max_iter=500, random_state=42, learning_rate_init=0.001) 與 **SVC**：MLPClassifier 和 SVC 都是強大的分類模型，能夠處理非線性數據，並且在不同的情況下表現相當好。不過，SVC 更加適合於處理少量樣本的問題，而 MLPClassifier 更加擅長於處理大量樣本和高維數據。

			與 **RandomForestClassifier**：MLPClassifier 和隨機森林都能夠進行複雜的模式識別，但 MLPClassifier 在特徵多樣且關聯性複雜的數據上可能會比隨機森林有更好的表現。隨機森林則對過擬合有更好的抗性。
			與 **KNeighborsClassifier**：KNeighborsClassifier 是一個基於鄰近點的分類方法，對於較小的數據集或者特徵較少的情況下，通常效果較好。而 MLPClassifier 通常能夠學到更多的數據模式，對於大規模且複雜的數據集表現更好。
			參數:
			hidden_layer_sizes=(100, 50)：設定兩層隱藏層，第一層包含 100 個神經元，第二層包含 50 個神經元。
			activation='relu'：選擇 ReLU 激活函數，ReLU 在處理深度神經網絡時通常能提供更好的效果。
			solver='adam'：選擇自適應學習率優化算法，這是目前最常用的優化方法之一。
			max_iter=200：最多訓練 200 次迭代。
			random_state=42：設置隨機種子，保證每次運行結果的一致性
			*結果尚未收斂，需再進行調整
			`Maximum iterations (500) reached and the optimization hasn't converged yet.`
			`warnings.warn(`
			`模型準確率: 0.97`

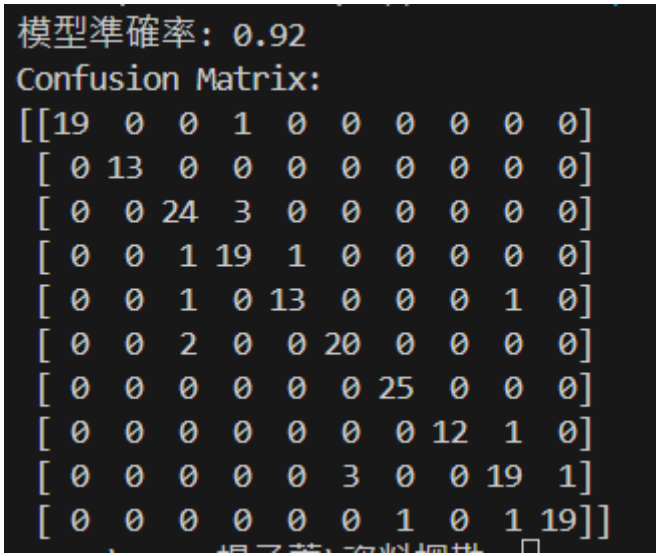
	Sklearn. neural_network	MLPRegressor()	MLPRegressor(hidden_layer_sizes=(100, 50), activation='relu', solver='adam', max_iter=500, random_state=42) MLPRegressor 的效果通常取決於數據的性質和超參數的選擇。它可以擅長處理非線性問題，並且比一些傳

			統的線性回歸方法（如線性回歸、Lasso 回歸等）更靈活。 MLPRegressor 的效果通常取決於數據的性質和超參數的選擇。它可以擅長處理非線性問題，並且比一些傳統的線性回歸方法（如線性回歸、Lasso 回歸等）更靈活。 參數: **hidden_layer_sizes**: 設定兩層隱藏層，第一層 100 個神經元，第二層 50 個神經元。 **activation**: 使用 'relu' 激活函數，它通常對回歸任務表現良好。 **solver**: 選擇 'adam' 優化算法，它是一種廣泛使用的優化器，適合大多數情況。 **max_iter**: 設定最大迭代次數為 500。你可以根據需要調整，來確保模型有足夠的迭代來收斂 均方誤差 (MSE): 0.10 R² 分數: 0.99
Ensemble classifier	Sklearn.ensemble	RandomForestClassifier ()	RandomForestClassifier(n_estimators=100, criterion='gini', max_depth=None, random_state=42) 高效處理非線性關係：隨機森林適用於大多數分類問題，尤其是那些具有非線性關係的問題。 泛化能力強：由於隨機森林的集成特性，它通常具有較強的泛化能力，能夠有效避免過擬合。 適用於多類別問題：隨機森林可以很好地處理多類別問題，比其他基於單一模型的分類器（如支持向量機）更具優勢。 參數:

			n_estimators=100：使用 100 顆樹，這是隨機森林模型的預設設置。 **criterion='gini'**：使用基尼不純度來進行決策樹的劃分。 **max_depth=None**：樹的最大深度不限制，直到所有葉節點是純的或滿足最小樣本數量要求。 **random_state=42**：隨機數種子，用於確保結果可重現。 模型準確率: 0.92
	Sklearn.ensemble	GradientBoostingClassifier() GradientBoostingRegressor()	GradientBoostingClassifier(n_estimators=100, learning_rate=0.1, max_depth=3, random_state=42) **高準確性**：Gradient Boosting 在許多應用中往往能提供比隨機森林更好的性能，尤其是在數據特徵具有高維或複雜性時。 **過擬合風險**：雖然 Gradient Boosting 防止過擬合，但如果超參數調整不當，模型仍然容易過擬合，特別是在訓練數據量較小時。 參數: **n_estimators=100**：使用 100 個基學習器（樹）。 **learning_rate=0.1**：每棵樹的貢獻權重，較小的學習率需要更多的樹來達到同樣的效果。 **max_depth=3**：每棵樹的最大深度，控制樹的複雜度。 **random_state=42**：設定隨機數種子，保證結果可重現。 模型準確率: 0.98

Evaluation	Sklearn.model_selection	KFold()	KFold(n_splits=5, shuffle=True, random_state=42) **準確性較高的模型**：使用 KFold 時，模型的性能估計將更加穩定，避免了因為一次隨機分割所造成的偏差。例如，隨機森林、支持向量機（SVM）和梯度提升模型等強大的分類器，使用 KFold 交叉驗證後，往往能夠展示穩定且較高的分類準確性。 **計算成本增加**：如果模型訓練時間較長（例如深度學習模型），交叉驗證可能會顯著增加計算時間，因此需要考慮到時間成本 參數; **n_splits=5**：將數據集分成 5 個折進行交叉驗證。 **shuffle=True**：對數據進行隨機打亂，確保交叉驗證過程中的數據分佈更加均勻。 **random_state=42**：固定隨機數種子，保證結果可重現。 **RandomForestClassifier**：這裡使用隨機森林模型作為分類器，這個模型會在每次交叉驗證中進行訓練和測試。 
	Sklearn.model_selection	ShuffleSplit()	ShuffleSplit(n_splits=5, test_size=0.2, random_state=42) **ShuffleSplit**: 每次隨機地拆分訓練集和測試集，因此每個樣本有可能在不同的拆分中被選為測試集或訓練集。這樣的隨機拆分方式可以幫助檢測模型在不同數據劃分下的穩定性。 **KFold**: 會將數據劃分為固定的折數，並確保每個樣本會在每個拆分中都被測試一次。這對於需要每個樣本

			都被評估的情況非常有用，並且能夠均勻分配每個樣本的訓練和測試次數。 參數: n_splits=5: 進行 5 次隨機拆分。 test_size=0.2: 每次拆分後，測試集佔整體數據的 20%。 random_state=42: 保證每次執行結果一致。 ShuffleSplit 交叉驗證平均準確率: 0.94
	Sklearn.metrics	confusion_matrix() classification_report() f1_score() precision_recall_curve()	confusion_matrix(y_test, y_pred) confusion_matrix() 是一個用來展示模型預測結果與實際標籤之間差異的工具。它本身不進行模型訓練或預測，而是對已經預測的結果進行評估。 classification_report() 提供了更詳細的分類指標，如精確率、召回率和 F1 分數，這些指標能夠更全面地展示模型的性能 參數: **y_true**: 真實的標籤值，通常是測試集的標籤。 **y_pred**: 模型的預測結果，通常是模型對測試集的預測值。 **labels**: 這個參數指定了類別的順序。如果不指定，則默認為 y_true 和 y_pred 中出現過的類別。 **sample_weight**: 用來加權每個樣本的權重。 **normalize**: 是否對混淆矩陣進行歸一化。 normalize='true' 表示顯示每類的準確率，而 normalize=None 則顯示原始的計數。

			 RandomForestClassifier(n_estimators=100, random_state=42) **精確度 (Precision)**: 真正例 / (真正例 + 假正例)，表示預測為正類的樣本中有多少是正確的。 **召回率 (Recall)**: 真正例 / (真正例 + 假負例)，表示實際為正類的樣本中有多少被正確預測為正類。 **F1 分數 (F1-score)**: 2 * (Precision * Recall) / (Precision + Recall)，綜合考慮精確度和召回率，對於不平衡類別非常重要。 **支持度 (Support)**: 每個類別在實際標籤中出現的樣本數 y_true: 實際的標籤（通常是測試集的目標變數 y_test）。 y_pred: 預測的標籤（通常是模型對測試集的預測結果 y_pred）。

target_names: 用來指定每個類別的名稱，若沒有傳入，將會使用數字標籤（如 0, 1, 2 等）。
labels: 用來指定在報告中顯示的標籤。默認會顯示所有標籤。
output_dict: 如果設置為 True，則會返回字典格式的結果，可以方便進一步處理。
digits: 設置報告中數值顯示的小數位數，默認為 2 位。
zero_division: 當計算精確度、召回率或 F1 分數時，如果遇到 0 除數問題，可以設置這個參數，zero_division=0 表示設為 0，zero_division=1 表示設為 1。

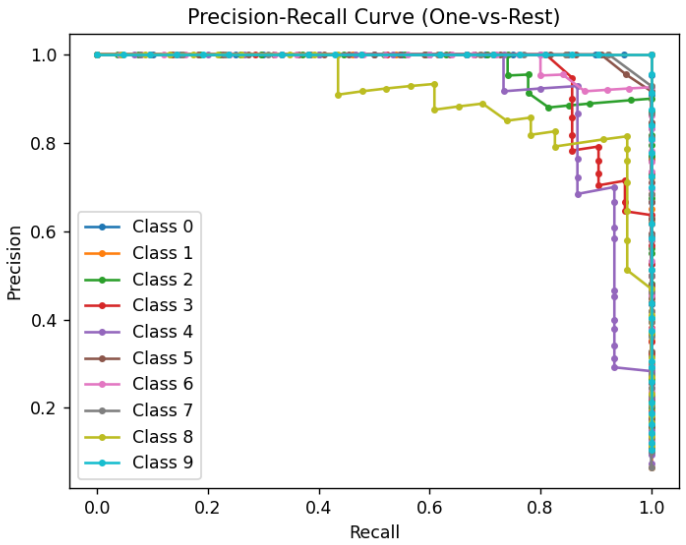
```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.95      0.97        20
           1       1.00      1.00      1.00        13
           2       0.86      0.89      0.87        27
           3       0.83      0.90      0.86        21
           4       0.93      0.87      0.90        15
           5       0.87      0.91      0.89        22
           6       0.96      1.00      0.98        25
           7       1.00      0.92      0.96        13
           8       0.86      0.83      0.84        23
           9       0.95      0.90      0.93        21

    accuracy                           0.92       200
   macro avg       0.93      0.92      0.92       200
weighted avg       0.92      0.92      0.92       200
```

f1_score(y_test, y_pred, average='weighted')

			`F1 分數: 0.92` precision_recall_curve(y_test == i, y_scores[:, i]) 參數: y_true: 實際的標籤（通常是測試集的目標變數 y_test）。 probas_pred: 模型預測為正類的概率（model.predict_proba()）。這是非常重要的一點，因為你需要提供的是模型的預測概率，而不是直接的預測標籤。 • 如果是二分類問題，model.predict_proba(X_test) 會返回每個樣本對應的類別概率，選擇第二列（即正類的概率）。 pos_label: 正類的標籤（預設為 1）。用來指定哪個類別是「正類」，若你的標籤是 0 和 1，則默認 1 是正類。 sample_weight: 樣本權重（默認為 None），如果每個樣本有不同的權重，可以用這個參數來指定。

補充(regression model)

http://www.cse.msu.edu/~ptan/dmbook/tutorials/tutorial5/tutorial5.html

其他參考資源:

- machine learning 參考書: "Introduction to Machine Learning with Python" 之 github code

https://github.com/amueller/introduction_to_ml_with_python/blob/master/02-supervised-learning.ipynb
https://github.com/amueller/introduction_to_ml_with_python/blob/master/05-model-evaluation-and-improvement.ipynb

Scikit Learn documentation(http://scikit-learn.org/stable/index.html)

- 尋搜尋其他可信網路資源