	<pre>import numpy as np import pandas as pd import matplotlib.pyplot as plt from sklearn.datasets import load_breast_cancer from sklearn.model_selection import train_test_split</pre>
	from sklearn.neighbors import KNeighborsClassifier Task 1. k-nearest neighbors algorithm (KNN)
	1. Load Dataset [1 point]
[2]:	 Load the csv file, 'final_shuffled_breast_cancer100.csv' as df df = pd.read_csv("final_shuffled_breast_cancer100.csv")
;	 2. Split the independent variable set and the target variable set [1 point] Assign X to the independent variable dataset
[3]:	 Assign y to the target variable dataset X=df.drop('target', axis=1)
	y = df['target'] 3. Split Dataset into the train & testset [1 point]
,	** When you use scikit-learn method to split the train & test set : • Set random_state to zero.
	 the ratio of train set and test set is as follows: 80% train set / 20% test set Assign the variable names as follow: X_train, X_test, y_train, y_test
[4]:	<pre>X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.8, random_state=0)</pre> 4. Load a KNN model by scikit-learn. [1 point]
	 Assign KNN model as variable name KNN Set the n_neighbors hyperparameter as 5
[5]:	KNN = KNeighborsClassifier(n_neighbors = 5) 5. Predict on your test set. [1 point]
[6]:	<pre>KNN.fit(X_train,y_train) y_pred = KNN.predict(X_test)</pre>
F-7.1.	print(y_pred) [1. 0. 0. 1. 1. 0. 0. 0. 1. 0. 0. 1. 0. 0. 1. 0. 1. 0. 1.]
[7]:	<pre>#check conf_matrix before making cal_confusion from sklearn.metrics import confusion_matrix, classification_report conf_matrix = confusion_matrix(y_test, y_pred)</pre>
	<pre>print(conf_matrix) [[12 2] [0 6]]</pre>
[8]:	len(y_pred) 20
[9]:	<pre>len(y_test) 20</pre>
t[9]:	6. Evaluate the prediction result of your model.
	 Calculate the confusion matrix which consists of TP, FP, TN, FN (True Positive, False Positive, True Negative, False Negative) Calculate accuracy rate, sensitivity, specificity Fill in the blank function in order to accomplish the aforementioned tasks. (DO NOT USE PACKAGES IN THIS TASK)
ı	6.1 Calculate the confusion matrix which consists of TP, FP, TN, FN (True Positive, False Positive, True Negative, False Negative) [5 points]
[10]:	 Fill in the blank function in order to accomplish the aforementioned tasks. (DO NOT USE PACKAGES IN THIS TASK) def cal_confusion(y_true, y_pred): TP =0
	FP = 0 $TN = 0$ $FN = 0$
	<pre>for i in range(len(y_true)): if y_true[i]==1 and y_pred[i]==1: TP = TP + 1 if y_pred[i]==1 and y_true[i]!=y_pred[i]:</pre>
	<pre>FP = FP + 1 if y_true[i]==0 and y_pred[i]==0: TN = TN + 1 if y_pred[i]==0 and y_true[i]!=y_pred[i]:</pre>
	FN = FN + 1 return TP, FP, TN, FN
[11]:	cal_confusion(y_test, y_pred)
	(6, 2, 12, 0) 6.2 Calculate accuracy rate by filling in the blank of cal_accuracy function. [5 points]
[12]:	• Fill in the blank function in order to accomplish the aforementioned tasks. (DO NOT USE PACKAGES IN THIS TASK) def cal_accuracy(y_true, y_pred):
	TP =0 FP =0 TN = 0
	FN = 0 for i in range(len(y_true)): if y_true[i]==1 and y_pred[i]==1: TP = TP + 1
	<pre>if y_pred[i]==1 and y_true[i]!=y_pred[i]: FP = FP + 1 if y_true[i]==0 and y_pred[i]==0:</pre>
	<pre>TN = TN + 1 if y_pred[i]==0 and y_true[i]!=y_pred[i]: FN = FN + 1</pre>
	<pre>accuracy = 0 accuracy = (TP+TN)/ (TP+FP+TN+FN) return accuracy</pre>
[13]:	<pre>cal_accuracy(y_test, y_pred)</pre>
[13]:	6.3 Calculate sensitivity by filling in the blank of cal_sensitivity function. [5 points]
	 Fill in the blank function in order to accomplish the aforementioned tasks. (DO NOT USE PACKAGES IN THIS TASK) When it comes to the cal_sensitivity function, we didn't specify the exact input variables, but just include all of TP, FP, TN, FN. You have to choose two of them and use th
[14]:	as the input variables of the cal_sensitivity function. def cal_sensitivity(TP,FP,TN,FN):
	<pre>sensitivity = 0 sensitivity = (TP)/ (TP+FN)</pre>
	return sensitivity
[15]: [15]:	<pre>cal_sensitivity(6,2,12,0) 1.0</pre>
ı	6.4 Calculate specificity by filling in the blank of cal_specificity function. [5 points]
	 Fill in the blank function in order to accomplish the aforementioned tasks. (DO NOT USE PACKAGES IN THIS TASK) When it comes to the cal_specificity function, we didn't specify the exact input variables, but just include all of TP, FP, TN, FN. You have to choose two of them and use the input variables of the cal_specificity function.
[16]:	def cal_specificity(TP,FP,TN,FN):
	<pre>specificity=0 specificity= (TN)/(TN+FP)</pre>
[17]:	
4 .	<pre>return specificity cal_specificity(6,2,12,0)</pre>
	cal_specificity(6,2,12,0) 0.8571428571428571
	cal_specificity(6,2,12,0) 0.8571428571428571 6.5 Print all of the results [1 point] • print all of the results (confusion matrix, accuracy, sensitivity, specificity)
(cal_specificity(6, 2, 12, 0) 0.8571428571428571 6.5 Print all of the results [1 point] • print all of the results (confusion matrix, accuracy, sensitivity, specificity) • fill in the below print function by your own results TP = cal_confusion(y_test,y_pred)[0]
(cal_specificity(6, 2, 12, 0) 0.8571428571428571 6.5 Print all of the results [1 point] • print all of the results (confusion matrix, accuracy, sensitivity, specificity) • fill in the below print function by your own results
	cal_specificity(6, 2, 12, 0) 0.8571428571428571 6.5 Print all of the results [1 point] • print all of the results (confusion matrix, accuracy, sensitivity, specificity) • fill in the below print function by your own results TP = cal_confusion(y_test,y_pred)[0] FP = cal_confusion(y_test,y_pred)[1] TN = cal_confusion(y_test,y_pred)[2] FN = cal_confusion(y_test,y_pred)[3] print('confusion matrix: ', cal_confusion(y_test, y_pred)) print('accuracy: ', cal_accuracy(y_test, y_pred)) print('sensitivity: ', cal_sensitivity(TP, FP, TN, FN))
	cal_specificity(6,2,12,0) 0.8571428571428571 6.5 Print all of the results [1 point] • print all of the results (confusion matrix, accuracy, sensitivity, specificity) • fill in the below print function by your own results TP = cal_confusion(y_test, y_pred)[0] FP = cal_confusion(y_test, y_pred)[1] TN = cal_confusion(y_test, y_pred)[2] FN = cal_confusion(y_test, y_pred)[3] print('confusion matrix: ', cal_confusion(y_test, y_pred)) print('sensitivity: ', cal_sensitivity(TP, FP, TN, FN)) print('sensitivity: ', cal_specificity(TP, FP, TN, FN)) confusion matrix: (6, 2, 12, 0) accuracy: 0.9
[18]:	cal_specificity(6,2,12,0) 0.8571428571428571 6.5 Print all of the results [1 point] • print all of the results (confusion matrix, accuracy, sensitivity, specificity) • fill in the below print function by your own results TP = cal_confusion(y_test,y_pred)[0] FP = cal_confusion(y_test,y_pred)[1] TN = cal_confusion(y_test,y_pred)[2] FN = cal_confusion(y_test,y_pred)[3] print('confusion matrix: ', cal_confusion(y_test, y_pred)) print('accuracy: ', cal_accuracy(y_test, y_pred)) print('specificity: ', cal_specificity(TP, FP, TN, FN)) print('specificity: ', cal_specificity(TP, FP, TN, FN)) confusion matrix: (6, 2, 12, 0)
[18]:	cal_specificity(6, 2, 12, 0) 0.8571428571428571 6.5 Print all of the results [1 point] • print all of the results (confusion matrix, accuracy, sensitivity, specificity) • fill in the below print function by your own results TP = cal_confusion(y_test, y_pred)[0] FP = cal_confusion(y_test, y_pred)[1] TN = cal_confusion(y_test, y_pred)[2] FN = cal_confusion(y_test, y_pred)[3] print('confusion matrix: ', cal_confusion(y_test, y_pred) print('accuracy: ', cal_accuracy(y_test, y_pred) print('specificity: ', cal_specificity(TP, FP, TN, FN)) print('specificity: ', cal_specificity(TP, FP, TN, FN)) confusion matrix: (6, 2, 12, 0) accuracy: 0.9 sensitivity: 1.8 sensitivity: 1.8 Sensitivity: 0.8571428571428571 6.6 Plot accuracy results as you change the K values. [3 points] • Plot the accuracy results from changing the number of K = (1,2,3,4,5,10)
[18]:	cal_specificity(6,2,12,0) 0.8571428571428571 6.5 Print all of the results [1 point] • print all of the results (confusion matrix, accuracy, sensitivity, specificity) • fill in the below print function by your own results TP = cal_confusion(y_test, y_pred)[0] FP = cal_confusion(y_test, y_pred)[1] TN = cal_confusion(y_test, y_pred)[2] FN = cal_confusion(y_test, y_pred)[3] print('confusion matrix: ', cal_confusion(y_test, y_pred) print('accuracy' ', cal_accuracy(y_test, y_pred) print('sensitivity' ', cal_sensitivity' (FP, FP, TN, FN) print('sensitivity' (FP, FP, TN, FN) print('sensiti
[18]:	cal_specificity(6,2,12,0) 0.8571428571428571 6.5 Print all of the results [1 point] • print all of the results (confusion matrix, accuracy, sensitivity, specificity) • fill in the below print function by your own results TP = cal_confusion(y_test, y_pred) [0] FP = cal_confusion(y_test, y_pred) [1] TN = cal_confusion(y_test, y_pred) [2] FN = cal_confusion(y_test, y_pred) [3] print('confusion matrix: ', cal_confusion(y_test, y_pred)) print('accuracy: ', cal_accuracy(y_test, y_pred)) print('sensitivity: ', cal_sensitivity(TP, FP, TN, FN)) print('specificity: ', cal_sensitivity(TP, FP, TN, FN)) confusion matrix: (6, 2, 12, 0) accuracy: 0.9 sensitivity: 0.8571428571428571 6.6 Plot accuracy results as you change the K values. [3 points] • Plot the accuracy results from changing the number of K = (1,2,3,4,5,10) k_range = [1,2,3,4,5,10] accuracy = []
[18]:	acuracy: acusers: (6, 2, 12, 0) e. serial confusion matrix: (acuracy) sensitivity, specificity) print('confusion matrix: ', cal_confusion(y_test, y_pred)[2] print('confusion matrix: ', cal_confusion(y_test, y_pred)[3] print('confusion matrix: ', cal_confusion(y_test, y_pred)[3] print('confusion matrix: ', cal_confusion(y_test, y_pred)[3] print('sonsitivity: ', cal_securacy(y_test, y_pred)) print('securacy: ', cal_securacy(
[18]:	cal_specificity(6,2,12,8) 0.887i428571428571 6.5 Print all of the results [1 point] • print all of the results (confusion matrix, accuracy, sensitivity, specificity) • fill in the below print function by your own results TP = cal_confusion(y_test, y_pred)[0] FP = cal_confusion(y_test, y_pred)[1] FN = cal_confusion(y_test, y_pred)[1] FN = cal_confusion(y_test, y_pred)[2] FN = cal_confusion(y_test, y_pred)[3] FN = cal_confusion matrix: (6, 2, 12, 0) confusion matrix: (1, 0, 2, 2, 12, 0) confusion matrix: (1, 0, 2, 2, 2, 0) confusion matrix: (1, 0, 2, 2, 2, 2, 0) confusion matrix: (1, 0, 2, 2, 2, 2,
[18]:	cal specificity(6, 2, 12, 8) 0.8671428671428671 0.5 Print all of the results [1 point] • print all of the results (confusion matrix, accuracy, sensitivity, specificity) • fill in the below print function by your own results TP = cal_confusion(y_test_y_pred)[0] FP = cal_confusion(y_test_y_pred)[1] TN = cal_confusion(y_test_y_pred)[2] FN = cal_confusion(y_test_y_pred)[2] FN = cal_confusion(y_test_y_pred)[2] FN = cal_confusion(y_test_y_pred)[2] Print('accuracy' ', cal_accuracy(y_test_y_pred) ') Print('accuracy' ', cal_accuracy(y_test_y_pred) ') Print('specificity' ', cal_aspecificity(TP, FP, TM, FN) ') Print('specificity' ', cal_aspecificity(TP, FP, TM, FN) ') Print('specificity' : cal_aspecificity(TP, FP, TM, FN) ') ### Confusion matrix: (6, 2, 12, 0) ### Accuracy of the confusion matrix: (7, cal_accuracy) the confusion matrix: (8, cal_accuracy) the cal_accuracy the confusion matrix: (8, cal_accuracy) the cal_accuracy the confusion matrix: (8, cal_accuracy) the confusion matrix: (8, cal_accuracy) the cal_accuracy the confusion matrix: (8, cal_accuracy) the cal_accuracy the confusion matrix: (8, cal_accuracy) the cal_accuracy the cal_acc
[18]:	cal_specificity(0,2,12,0) 0.857i4285714285714 6.5 Print all of the results [1 point] • print all of the results [2 point] • print all of the results (contision matrix, accuracy, sensitivity, specificity) • fill in the below print function by your own results TP = cal_confusion(y_test, y_pred)[0] FP = cal_confusion(y_test, y_pred)[1] FN = cal_confusion(y_test,
[18]: [20]:	us_l_specificity(6,2,12,0) 0.8571478571478571478571 6.5 Print all of the results (1 point) • print all of the results (confusion matrix, accuracy, sensitivity, specificity) • till in the below print function by your own results TP = cal_confusion(v_test_v_pred)[0] FP = cal_confusion(v_test_v_pred)[1] FP = cal_confusion(v_test_v
[18]: [20]:	cal specificity(6, 2, 12, 0) 0. 8671428671428671 6.5 Print all of the results (confusion matrix, accuracy, sensitivity, specificity) • fill in the below print: [function by your own results IP = cul_confusion(y_text, y_pred)[0] PF = cal_confusion(y_text, y_pred)[1] FF = cal_confusion(y_text, y_pred)[3] PF = cal_confusion(
[19]:	cal. specificity (6, 2, 12, 6) 0. 8571428971428571 6.5 Pint all of the results (1 point) - print all of the results (2 point) - print all of the results (2 point) - print (3 point
[19]:	cal_specificity(6,2.12,0) 0.8871428871428971 6.5 Print all of the results [1 point] • point all of the results (comusion mamx, accuracy, sensitivity, specificity) • fill in the below print; function by your own results **P = cal_contractor(y_test_y_pred)[0] **P = cal_contractor(y_test_y_pred)[2] **P = cal_contractor(y
[19]:	cal_specificativ(s,2,12,0) 0.8571428571428571 6.5 Print all of the results (all point) • print all of the results (all point) • print all of the results (control by your own results) • print all of the results (control by your own results) • print (all points (text, y pred) (all prints) • print (all points) (text, y pred) (all prints) • print (all points) (text, y pred) (all prints) • print (all points) (text, y pred) (all prints) • print (all points) (text, y pred) (all prints) • print (all points) (text, y pred) (all prints) • print (all points) (text, y pred) (all prints) • print (all points) (all points) (text, y pred) (all prints) • print (all points) (all points) (text, y pred) • print (all points) (all points) (all points) • Plot the accuracy results as you change the K values (B points) • Plot the accuracy results from changing the number of K = (1,2,3,4,5,10) K_range = (1,2,3,4,5,30) • print (all points) (all points) • Plot the accuracy (all points) (all points) • Plot the accuracy (all points) (all points) • print (all points) (all points) (all points) • print (all points) (all points) (all points) (all points) • print (all points) (all
[19]:	cal_specificativ(d, 2, 12, 0) 0.8571428571428571 6.5 Print all of the results (Dinit) • print all of the results (Dinit) • print all of the results (Dinit) • fill in the below (print) (tout, y prod) (0) FP = cal_confusion(y tout, y prod) (1) FP = cal_confusion(y tout, y prod) (1