110652036 邱立昕

Explain the difference between parametric and non - parametric models .

parametric：

參數化模型對於數據的基本分布形式作出明確的假設。

這些模型具有固定數量的參數，在訓練階段確定。

參數化模型的示例包括線性回歸、邏輯回歸和高斯朴素貝葉斯分類器。

一旦從數據中估計出參數，模型的結構就固定了，不受訓練數據集大小的影響。

參數化模型在處理大型數據集時可能效率更高，因為它們有一個固定數量的參數需要估計。

non - parametric：

非參數化模型不對數據的基本分布形式做出明確的假設。

這些模型具有彈性的參數數量，隨著訓練數據集的大小而增長。

非參數化模型的示例包括 k 最近鄰（KNN）、決策樹和支持向量機與非線性核。

非參數化模型可以捕捉到更複雜的數據關係，因為它們不受固定形式的限制。

然而，非參數化模型在處理大型數據集時可能會計算成本更高，因為它們的複雜度隨著訓練數據集的大小而增加。

What is ensemble learning ?

通過結合多個基本模型的預測來改進整體的預測性能。這些基本模型可以是同類型的模型或者是不同類型的模型。集成學習的核心思想是通過結合多個模型的預測，以獲得比任何單個模型更好的預測效果

Please explain the difference between bagging boosting and stacking .

Bagging（自助法）：

Bagging 通常用於降低模型的方差。

在 Bagging 中，我們從原始數據集中隨機抽樣生成多個子樣本，然後用每個子樣本來訓練一個基本模型。這些基本模型可以是相同的算法，也可以是不同的算法。

最後，通過對所有基本模型的預測進行平均或投票，來進行最終的預測。

典型的 Bagging 算法包括隨機森林（Random Forest）。

Boosting（提升法）：

Boosting 通常用於降低模型的偏差。

在 Boosting 中，基本模型是按照一定的順序逐個訓練的，每個模型都會根據前一個模型的表現進行修正。

Boosting 的核心思想是通過逐步地提高模型對錯誤樣本的關注度，來提高整體模型的性能。

常見的 Boosting 算法包括 AdaBoost 和梯度提升樹（Gradient Boosting Trees）。

Stacking（堆疊法）：

Stacking 通常用於進一步提高預測性能。

在 Stacking 中，我們訓練多個基本模型，然後將這些基本模型的預測結果作為新的特徵，再訓練一個元模型（meta-model）來組合這些基本模型的預測。

基本模型的預測結果通常被視為元特徵，用於訓練元模型。這使得 Stacking 可以在不同的基本模型之間進行組合，以獲得更好的預測性能。

Stacking 的設計通常比較複雜，需要額外的交叉驗證來訓練元模型。

Explain the meaning of the " n_neighbors ” parameter in KNeighborsClassifier , n_estimators " in RandomForestClassifier and AdaBoostClassifier .

KNeighborsClassifier 中的 n_neighbors：

n_neighbors 指定 K 最近鄰算法中使用的鄰居數量。

K 最近鄰（KNN）是一種基於實例的學習或懶惰學習的方法，其中函數僅在本地進行近似，所有計算都延遲到函數評估時進行。

該算法計算查詢點與所有訓練點之間的距離。然後，它選擇指定數量的最近鄰居（基於 n_neighbors 參數），並將查詢點分配給這些鄰居中的多數類（對於分類任務）或計算這些鄰居的值的平均值/中位數（對於回歸任務）。

n_neighbors 控制模型的複雜度：較小的值會使模型更靈活且可能過度擬合，而較大的值會使模型更平滑，但可能出現欠擬合。

RandomForestClassifier 中的 n_estimators：

n_estimators 指定隨機森林集成中的決策樹數量。

隨機森林是一種集成學習方法，在訓練期間構造了許多決策樹並輸出了這些樹的類別模式（分類）或平均預測（回歸）。

隨機森林中的每棵樹都是在訓練數據的隨機子集上訓練的，並且在每個分裂點考慮了隨機的一部分特徵，這引入了隨機性並減少了過擬合。

n_estimators 控制集成中的樹的數量。增加樹的數量通常會提高性能，但也會增加計算時間。

AdaBoostClassifier 中的 n_estimators：

n_estimators 指定 AdaBoost 集成中將組合為最終強學習者的最大弱學習者數量（通常是決策樹）。

AdaBoost（自適應提升）是一種集成學習方法，依次結合多個弱學習者。它在每次迭代中將更高的權重分配給錯誤分類的實例，因此後續的弱學習者會更多地關注這些實例。

n_estimators 控制將組合的弱學習者的數量。增加弱學習者的數量可以提高性能，但太多可能會導致過度擬合。

Explain the meaning of four numbers in the confusion matrix .

		Predicted Class	
		Class 0	Class 1
Actual Class	Class 0	TN	FP
	Class 1	FN	TP

True Negatives (TN)：實際標籤是負類（Class 0），且模型正確地預測為負類的數量。換句話說，這是模型正確地將負樣本分類為負樣本的次數。

False Positives (FP)：實際標籤是負類（Class 0），但模型錯誤地預測為正類的數量。換句話說，這是模型將負樣本錯誤分類為正樣本的次數。

False Negatives (FN)：實際標籤是正類（Class 1），但模型錯誤地預測為負類的數量。換句話說，這是模型將正樣本錯誤分類為負樣本的次數。

True Positives (TP)：實際標籤是正類（Class 1），且模型正確地預測為正類的數量。換句話說，這是模型正確地將正樣本分類為正樣本的次數。

這四個數字提供了有關模型性能的重要信息，通過這些數字可以計算出許多常見的性能指標，如準確率、精確率、召回率、F1 分數等，以進一步評估模型的效果。

In addition to " Accuracy " , " Precision " and " Recall " are two common metrics in classification tasks , how to calculate them , and under what circumstances would you use them instead of " Accuracy " .

Precision：

Precision 衡量模型在所有正類預測中的真正正類比例。

它關注正類預測的準確性。

Precision 計算公式為：

Precision = (True Positives+False Positives)/True Positives

高 Precision 表示模型做出的 False Positives 預測較少，即當它預測為 True Positives 時，更有可能是正確的。

當 False Positives 的成本較高，且希望減少誤報時，Precision 特別有用。

recall：

recall 衡量數據集中所有 True Positives 中的真正 True Positives 預測比例。

它關注捕獲所有 True Positives 實例的能力。

recall 計算公式為：

(True Positives+False Negatives)/True Positives

高 recall 表示模型有效地捕獲了數據集中的大多數 True Positives 實例。

當 False Negatives 的成本較高，且希望最小化漏報時，recall 特別有用。

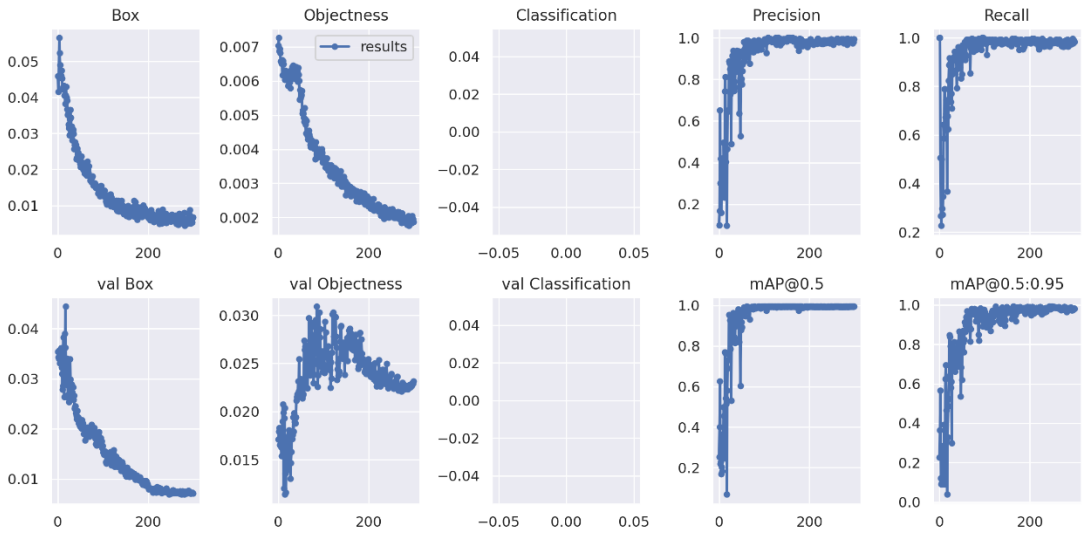
何時使用精確率和 recall 而不是準確率：

不平衡數據集：當數據集存在較大的類別不平衡時，其中一個類別（通常是少數類別）的實例明顯少於另一個類別，準確率可能不是一個可靠的指標。在這種情況下，精確率和 recall 提供了對模型性能更具信息性的評估。

不同的錯誤分類成本：當 False Positive 和 False Negatives 的成本不相等時，精確率和 recall 允許根據這些具體的錯誤分類成本來評估模型的性能。

不同方面的優先級：如果您的目標是最小化 False Positive（最大化 Precision）或最小化 False Negatives（最大化 recall），則精確率和 recall 是更適合考慮的指標。在這些情況下，準確率可能無法有效地優先考慮這些方面。

在 accurancy 和 parking slots 圖中，不同方法的準確率呈現出不同的趨勢，AdaBoost 和 RandomForest 有更好的表現，而 KNN 學習的沒這麼好，答對率最低



遇到問題:

我在剛開始做作業時其實看不太懂 code，就算助教上課時有說明，花了點時間和 chatGPT 交流一下才了解剛做甚麼，在 colab 的部分也卡了很久，找 detect.py 很久，結果是在 yolov7 的資料夾等等，總之這次作業很大部分都在了解題目，不過都一一解決了