**Chapter 3: Classification**

**Introduction**

* In contrast to regression tasks which predict values (like predicting housing prices, as explored in Chapter 2), **classification tasks predict classes**.
* Chapter 3 focuses on classification systems.

**MNIST Dataset**

* The MNIST dataset is used as an example in this chapter to illustrate the complexity of the classification task.
* It is already split into a **training set** (the first 60,000 images) and a **test set** (the last 10,000 images).
* The training set is **shuffled**, which ensures that all cross-validation folds are similar and prevents learning algorithms sensitive to instance order from performing poorly due to many similar instances in a row.

**Evaluating Classifiers**

* Evaluating a classifier is often **significantly trickier** than evaluating a regressor.
* A good way to evaluate a model is to use **cross-validation**, similar to what was done in Chapter 2.
* You can implement cross-validation yourself for more control, which involves splitting the training set into folds, training a clone of the classifier on training folds, and making predictions on the test fold.
* Scikit-Learn's cross\_val\_score() function can be used for K-fold cross-validation. For example, with three folds, the training set is split into three, and the model is trained and evaluated 10 times, each time picking a different fold for evaluation and training on the remaining nine.
* The StratifiedKFold class performs **stratified sampling** to produce folds that contain a representative ratio of each class.

**Performance Measures**

* There are many performance measures available for classifiers.
  + **Accuracy**: The ratio of correct predictions. However, accuracy can be misleading, especially with skewed datasets (e.g., a binary classifier where only 10% are positive instances might achieve 90% accuracy by always predicting "negative", which is not useful).
  + **Precision**: This is the accuracy of the positive predictions. It's the ratio of the true positives (TP) to the sum of true positives and false positives (FP).
    - *Precision = TP / (TP + FP)* [not explicitly in sources, but implied by text and common definition]
    - A trivial way to achieve perfect precision is to make only one correct positive prediction (precision = 1/1 = 100%), but this is not useful as it ignores most positive instances. Precision is typically used alongside recall.
  + **Recall**: Also called **sensitivity** or **True Positive Rate (TPR)**. This is the ratio of positive instances that are correctly detected by the classifier. It is the ratio of true positives (TP) to the sum of true positives and false negatives (FN).
    - **recall = TP / (TP + FN)**
  + **F1 Score**: A single metric used to combine precision and recall, especially when comparing two classifiers simply. It is the **harmonic mean** of precision and recall. The harmonic mean gives more weight to low values, so a classifier gets a high F1 score only if both recall and precision are high.
    - **F1 = 2 / (1 / precision + 1 / recall)**
    - **F1 = 2 × precision × recall / (precision + recall)**
    - **F1 = TP / (TP + FN + FP / 2)**
    - The F1 score favors classifiers with similar precision and recall. However, the choice of prioritizing precision or recall depends on the context of the project. For example, detecting safe videos for kids might prioritize high precision (low recall is acceptable), while detecting shoplifters might prioritize high recall (lower precision is acceptable).
  + **Precision/Recall Tradeoff**: It is often necessary to balance precision and recall. This can be done by adjusting the classification threshold. A higher threshold generally leads to higher precision and lower recall, and vice versa. Plotting precision directly against recall helps visualize this tradeoff. You can select a tradeoff point based on the project needs (e.g., aiming for a certain precision level and finding the corresponding threshold). If someone states a precision goal (e.g., 99% precision), you should ask "at what recall?".
  + **ROC Curve**: The Receiver Operating Characteristic (ROC) curve is another common tool for binary classifiers. It plots the **True Positive Rate (recall)** against the **False Positive Rate (FPR)**. The FPR is the ratio of negative instances incorrectly classified as positive. It is equal to one minus the True Negative Rate (TNR), which is the ratio of negative instances correctly classified as negative. The TNR is also called **specificity**. Hence, the ROC curve plots sensitivity (recall) versus 1 – specificity.

**Error Analysis**

* Analyzing the types of errors a classifier makes is a way to find ways to improve a promising model.
* Analyzing individual errors can provide insights into what the classifier is doing and why it is failing, although it is more difficult and time-consuming.
* Plotting examples of misclassified instances (like threes and fives in the MNIST dataset) can help in error analysis.