Machine Learning in Python

Supervised Learning - Classification and Metrics

Cristian A. Marocico, A. Emin Tatar

Center for Information Technology
University of Groningen

Friday, July 4th 2025

Outline

1. Introduction to Classification
2. Classification Basics
3. Evaluation Metrics
4. k-Nearest Neighbours
5. Decision Trees
6. Support Vector Machines (SVM)

Introduction to Classification

Classification Definition

Introduction to Classification

Classification — Definition

Classification is a type of supervised learning where the model learns from labeled data to predict the class of new observations based on past data.

Introduction to Classification

Classification — Definition

Classification is a type of supervised learning where the model learns from labeled data to predict the class of new observations based on past data.

Classification vs. Regression is a key distinction in supervised learning:

- In classification, the target variable is categorical (e.g., "spam" or "not spam").

Introduction to Classification

Classification — Definition

Classification is a type of supervised learning where the model learns from labeled data to predict the class of new observations based on past data.

Classification vs. Regression is a key distinction in supervised learning:

- In classification, the target variable is categorical (e.g., "spam" or "not spam").
- In regression, the target variable is continuous (e.g., predicting a price).

Classification Types

Classification Types

Definition

Classification Types

Classification Types — Definition

Classification can be broadly divided into two types:

- Binary Classification: The target variable has two classes (e.g., "yes" or "no", "spam" or "not spam"). Numerically, this can always be represented as 0 and 1.

Classification Types

Classification Types

Definition

Classification can be broadly divided into two types:

- Binary Classification: The target variable has two classes (e.g., "yes" or "no", "spam" or "not spam"). Numerically, this can always be represented as 0 and 1.
- Multiclass Classification: The target variable has more than two classes (e.g., "cat", "dog", "weasel"). In this case, the model predicts one of several possible categories.

Classification Algorithms

Classification Algorithms

Definition

Classification Algorithms

Classification Algorithms — Definition

Classification algorithms are designed to learn from labeled data and make predictions about the class of new, unseen data. Some common algorithms include:

- Logistic Regression: Despite its name, it is used for binary classification. It models the probability that a given input belongs to a particular class.

Classification Algorithms

Classification Algorithms — Definition

Classification algorithms are designed to learn from labeled data and make predictions about the class of new, unseen data. Some common algorithms include:

- Logistic Regression: Despite its name, it is used for binary classification. It models the probability that a given input belongs to a particular class.
- k-Nearest Neighbors (k-NN): A non-parametric method that classifies a data point based on the classes of its nearest neighbors in the feature space.

Classification Algorithms

Classification Algorithms — Definition

Classification algorithms are designed to learn from labeled data and make predictions about the class of new, unseen data. Some common algorithms include:

- Logistic Regression: Despite its name, it is used for binary classification. It models the probability that a given input belongs to a particular class.
- k-Nearest Neighbors (k-NN): A non-parametric method that classifies a data point based on the classes of its nearest neighbors in the feature space.
- Decision Trees: A tree-like model that splits the data into subsets based on feature values, leading to a decision about the class label.

Classification Algorithms

Classification Algorithms — Definition

Classification algorithms are designed to learn from labeled data and make predictions about the class of new, unseen data. Some common algorithms include:

- Logistic Regression: Despite its name, it is used for binary classification. It models the probability that a given input belongs to a particular class.
- k-Nearest Neighbors (k-NN): A non-parametric method that classifies a data point based on the classes of its nearest neighbors in the feature space.
- Decision Trees: A tree-like model that splits the data into subsets based on feature values, leading to a decision about the class label.
- Support Vector Machines (SVM): A method that finds the hyperplane that best separates the classes in the feature space.

Classification Algorithms

Classification Algorithms

Definition

Classification algorithms are designed to learn from labeled data and make predictions about the class of new, unseen data. Some common algorithms include:

- Logistic Regression: Despite its name, it is used for binary classification. It models the probability that a given input belongs to a particular class.
- k-Nearest Neighbors (k-NN): A non-parametric method that classifies a data point based on the classes of its nearest neighbors in the feature space.
- Decision Trees: A tree-like model that splits the data into subsets based on feature values, leading to a decision about the class label.
- Support Vector Machines (SVM): A method that finds the hyperplane that best separates the classes in the feature space.
- More advanced algorithms like Random Forests, Gradient Boosting, and Neural Networks.

Logistic Regression

Logistic Regression

Definition

Logistic Regression

Logistic Regression — Definition

Logistic Regression models the probability that the target variable y belongs to a particular class. The logistic function (sigmoid) is used to map predicted values to probabilities between 0 and 1. The decision boundary is determined by the threshold (commonly 0.5) for classifying observations into different classes.

Logistic Regression

Logistic Regression — Definition

Logistic Regression models the probability that the target variable y belongs to a particular class. The logistic function (sigmoid) is used to map predicted values to probabilities between 0 and 1. The decision boundary is determined by the threshold (commonly 0.5) for classifying observations into different classes.

The sigmoid function is defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

where z is a linear combination of the input features.

Evaluation Metrics for Classification

Confusion Matrix

Definition

Evaluation Metrics for Classification

Confusion Matrix

Definition

The confusion matrix summarizes the performance of a classification algorithm by comparing predicted and actual labels.

Evaluation Metrics for Classification

Confusion Matrix — Definition

The confusion matrix summarizes the performance of a classification algorithm by comparing predicted and actual labels.

	Predicted $= 1$	Predicted $= 0$
Actual $= 1$	True Positive (TP)	False Negative (FN)
Actual $= 0$	False Positive (FP)	True Negative (TN)

From these four numbers we obtain the core metrics summarized next.

Accuracy (Overall Success Rate)

Accuracy (Overall Success Rate) — Definition

Accuracy is a measure of how often the classifier is correct across all predictions. It answers the question: "What fraction of all predictions are correct?"

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

Despite its simplicity, accuracy can be misleading, especially in imbalanced datasets where one class dominates.

Precision (Positive Predictive Value)

Precision (Positive Predictive Value) Definition

Precision is a measure of the accuracy of positive predictions. It answers the question: "When the classifier predicts 1, how often is it correct?"

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

High precision indicates that when the model predicts a positive class, it is likely correct, but it does not account for false negatives.

Recall (Sensitivity, True-Positive Rate)

Recall (Sensitivity, True-Positive Rate) — Definition

Recall is a measure of the model's ability to capture all positive instances. It answers the question: "Of all the actual 1 cases, how many did we catch?"

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

High recall indicates that the model is good at identifying positive cases, but it does not consider false positives.

F1-Score

F1-Score — Definition

F1-Score is the harmonic mean of precision and recall, providing a balance between the two metrics:

$$F_1 = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

It is particularly useful when the class distribution is imbalanced, as it considers both false positives and false negatives. High F1-Score indicates a good balance between precision and recall, while a low F1-Score suggests that either precision or recall (or both) are low.

Precision-Recall Trade-off

Precision-Recall Trade-off — Definition

The Precision-Recall Trade-off illustrates the inverse relationship between precision and recall:

Precision-Recall Curve — Definition

Precision-Recall Trade-off

Precision-Recall Trade-off — Definition

The Precision-Recall Trade-off illustrates the inverse relationship between precision and recall:

- Increasing the threshold for classifying a positive instance will increase precision but decrease recall.

Precision-Recall Curve — Definition

Precision-Recall Trade-off

Precision-Recall Trade-off (Definition)

The Precision-Recall Trade-off illustrates the inverse relationship between precision and recall:

- Increasing the threshold for classifying a positive instance will increase precision but decrease recall.
- Decreasing the threshold will increase recall but decrease precision.

Precision-Recall Curve (Definition)

Precision-Recall Trade-off

Precision-Recall Trade-off — Definition

The Precision-Recall Trade-off illustrates the inverse relationship between precision and recall:

- Increasing the threshold for classifying a positive instance will increase precision but decrease recall.
- Decreasing the threshold will increase recall but decrease precision.

Precision-Recall Curve — Definition

Precision-Recall Trade-off

Precision-Recall Trade-off — Definition

The Precision-Recall Trade-off illustrates the inverse relationship between precision and recall:

- Increasing the threshold for classifying a positive instance will increase precision but decrease recall.
- Decreasing the threshold will increase recall but decrease precision.

Precision-Recall Curve — Definition

Precision-Recall Curve is a graphical representation of the trade-off between precision and recall for different threshold values.

Receiver Operating Characteristic (ROC) Curve

Receiver Operating Characteristic (ROC) Curve — Definition

ROC Curve is a graphical representation of a classifier's performance across different thresholds. It plots the true positive rate (recall) against the false positive rate (FPR) at various threshold settings.

Area Under the Curve (AUC)

Area Under the Curve (AUC) Definition

Area Under the Curve (AUC)

Area Under the Curve (AUC) — Definition

Area Under the Curve (AUC) is a single scalar value that summarizes the performance of a classifier across all possible thresholds. It is calculated as the area under the ROC curve. AUC provides an aggregate measure of performance across all classification thresholds, making it useful for comparing different classifiers.

k-Nearest Neighbours (k-NN)

k-Nearest Neighbours (k-NN) — Definition

k-Nearest Neighbours (k-NN)

k-Nearest Neighbours (k-NN) — Definition

k-Nearest Neighbors (k-NN) is a non-parametric, instance-based learning algorithm that classifies a new observation based on the classes of its k nearest neighbors in the feature space. It does not explicitly learn a model but stores the training instances.

k-Nearest Neighbours (k-NN)

k-Nearest Neighbours (k-NN) — Definition

k-Nearest Neighbors (k-NN) is a non-parametric, instance-based learning algorithm that classifies a new observation based on the classes of its k nearest neighbors in the feature space. It does not explicitly learn a model but stores the training instances.

The algorithm works as follows:

- Compute the distance between the new observation and every point in the training set (Euclidean, Manhattan or Minkowski distances).

k-Nearest Neighbours (k-NN)

k-Nearest Neighbours (k-NN) — Definition

k-Nearest Neighbors (k-NN) is a non-parametric, instance-based learning algorithm that classifies a new observation based on the classes of its k nearest neighbors in the feature space. It does not explicitly learn a model but stores the training instances.

The algorithm works as follows:

- Compute the distance between the new observation and every point in the training set (Euclidean, Manhattan or Minkowski distances).
- Select the k closest neighbors.

k-Nearest Neighbours (k-NN)

k-Nearest Neighbours (k-NN) — Definition

k-Nearest Neighbors (k-NN) is a non-parametric, instance-based learning algorithm that classifies a new observation based on the classes of its k nearest neighbors in the feature space. It does not explicitly learn a model but stores the training instances.

The algorithm works as follows:

- Compute the distance between the new observation and every point in the training set (Euclidean, Manhattan or Minkowski distances).
- Select the k closest neighbors.
- Predict the class by majority vote among those neighbors.

k-Nearest Neighbours (*k*-NN)

k-Nearest Neighbours (*k*-NN) — Definition

k-Nearest Neighbors (*k*-NN) is a non-parametric, instance-based learning algorithm that classifies a new observation based on the classes of its k nearest neighbors in the feature space. It does not explicitly learn a model but stores the training instances.

The algorithm works as follows:

- Compute the distance between the new observation and every point in the training set (Euclidean, Manhattan or Minkowski distances).
- Select the k closest neighbors.
- Predict the class by majority vote among those neighbors.

k-Nearest Neighbours (k-NN)

k-Nearest Neighbours (k-NN) — Definition

k-Nearest Neighbors (k-NN) is a non-parametric, instance-based learning algorithm that classifies a new observation based on the classes of its k nearest neighbors in the feature space. It does not explicitly learn a model but stores the training instances.

The algorithm works as follows:

- Compute the distance between the new observation and every point in the training set (Euclidean, Manhattan or Minkowski distances).
- Select the k closest neighbors.
- Predict the class by majority vote among those neighbors.

The decision boundary is implicitly defined by the k nearest points, allowing the model to capture highly non-linear class boundaries without explicit training.

Mathematical Formulation

Mathematical Formulation

Definition

Given a training set $\{(x_i, y_i)\}_{i=1}^{n}$ where $y_i \in \{1, \ldots, C\}$, the prediction for a query point $\hat{x}$ is:

$$\hat{y} = \text{mode}\left(\{y_j | x_j \in N_k(\hat{x})\}\right),$$

where $N_k(\hat{x})$ denotes the set of k training points closest to $\hat{x}$.

Considerations for k-NN

Considerations for k-NN

- Choosing k:

Considerations for k-NN

Considerations for k-NN

- Choosing k:
  - A small k can lead to noise sensitivity (overfitting).

Considerations for k-NN

Considerations for k-NN

- Choosing k:
  - A small k can lead to noise sensitivity (overfitting).
  - A large k can smooth out class boundaries (underfitting).

Considerations for k-NN

Considerations for k-NN

- Choosing k:
  - A small k can lead to noise sensitivity (overfitting).
  - A large k can smooth out class boundaries (underfitting).
  - Cross-validation is often used to select the optimal k.

Considerations for k-NN

Considerations for k-NN

- Choosing k:
  - A small k can lead to noise sensitivity (overfitting).
  - A large k can smooth out class boundaries (underfitting).
  - Cross-validation is often used to select the optimal k.
- Advantages:

Considerations for k-NN

Considerations for k-NN

- Choosing k:
  - A small k can lead to noise sensitivity (overfitting).
  - A large k can smooth out class boundaries (underfitting).
  - Cross-validation is often used to select the optimal k.
- Advantages:
  - Simple and intuitive.

Considerations for k-NN

Considerations for k-NN

- Choosing k:
  - A small k can lead to noise sensitivity (overfitting).
  - A large k can smooth out class boundaries (underfitting).
  - Cross-validation is often used to select the optimal k.
- Advantages:
  - Simple and intuitive.
  - No training phase; all computation happens at prediction time.

Considerations for k-NN

Considerations for k-NN

- Choosing k:
  - A small k can lead to noise sensitivity (overfitting).
  - A large k can smooth out class boundaries (underfitting).
  - Cross-validation is often used to select the optimal k.
- Advantages:
  - Simple and intuitive.
  - No training phase; all computation happens at prediction time.
  - Naturally handles multi-class classification.

Considerations for k-NN

Considerations for k-NN

- Choosing k:
  - A small k can lead to noise sensitivity (overfitting).
  - A large k can smooth out class boundaries (underfitting).
  - Cross-validation is often used to select the optimal k.
- Advantages:
  - Simple and intuitive.
  - No training phase; all computation happens at prediction time.
  - Naturally handles multi-class classification.
- Disadvantages:

Considerations for k-NN

Considerations for k-NN

- Choosing k:
  - A small k can lead to noise sensitivity (overfitting).
  - A large k can smooth out class boundaries (underfitting).
  - Cross-validation is often used to select the optimal k.
- Advantages:
  - Simple and intuitive.
  - No training phase; all computation happens at prediction time.
  - Naturally handles multi-class classification.
- Disadvantages:
  - Computationally expensive at prediction time, especially with large datasets.

Considerations for k-NN

Considerations for k-NN

- Choosing k:
  - A small k can lead to noise sensitivity (overfitting).
  - A large k can smooth out class boundaries (underfitting).
  - Cross-validation is often used to select the optimal k.
- Advantages:
  - Simple and intuitive.
  - No training phase; all computation happens at prediction time.
  - Naturally handles multi-class classification.
- Disadvantages:
  - Computationally expensive at prediction time, especially with large datasets.
  - Sensitive to irrelevant features and feature scaling.

Considerations for k-NN

Considerations for k-NN

- Choosing k:
  - A small k can lead to noise sensitivity (overfitting).
  - A large k can smooth out class boundaries (underfitting).
  - Cross-validation is often used to select the optimal k.
- Advantages:
  - Simple and intuitive.
  - No training phase; all computation happens at prediction time.
  - Naturally handles multi-class classification.
- Disadvantages:
  - Computationally expensive at prediction time, especially with large datasets.
  - Sensitive to irrelevant features and feature scaling.
  - Requires a good choice of distance metric.

Decision Trees

Decision Trees — Definition

Decision Trees recursively partition the feature space into axis-aligned regions and assign a class label to each region. It consists of nodes (features), branches (decisions), and leaves (outcomes).

Decision Trees

Decision Trees — Definition

Decision Trees recursively partition the feature space into axis-aligned regions and assign a class label to each region. It consists of nodes (features), branches (decisions), and leaves (outcomes).

At each node, the algorithm chooses the split that maximizes a purity metric such as information gain or Gini impurity. The tree is built by recursively splitting the dataset into subsets until a stopping criterion is met (e.g., maximum depth, minimum samples per leaf).

Decision Trees

Decision Trees — Definition

Decision Trees recursively partition the feature space into axis-aligned regions and assign a class label to each region. It consists of nodes (features), branches (decisions), and leaves (outcomes).

At each node, the algorithm chooses the split that maximizes a purity metric such as information gain or Gini impurity. The tree is built by recursively splitting the dataset into subsets until a stopping criterion is met (e.g., maximum depth, minimum samples per leaf).

By hierarchically splitting the space, a tree can approximate complex decision boundaries while remaining interpretable-each internal node corresponds to a human-readable rule.

Information Gain

Information Gain — Definition

For a node containing a sample set S, the entropy is:

$$H(S) = -\sum_{c=1}^{C} p_c \log_2 p_c,$$

where p_c is the proportion of class c.

Information Gain

Information Gain — Definition

For a node containing a sample set S, the entropy is:

$$H(S) = -\sum_{c=1}^{C} p_c \log_2 p_c,$$

where p_c is the proportion of class c. A split of S into subsets S_L, S_R yields information gain:

$$IG = H(S) - \frac{|S_L|}{|S|} H(S_L) - \frac{|S_R|}{|S|} H(S_R).$$

The algorithm selects the split with the highest IG (or lowest Gini).

Information Gain

Information Gain — Definition

For a node containing a sample set S, the entropy is:

$$H(S) = -\sum_{c=1}^{C} p_c \log_2 p_c,$$

where p_c is the proportion of class c. A split of S into subsets S_L, S_R yields information gain:

$$IG = H(S) - \frac{|S_L|}{|S|} H(S_L) - \frac{|S_R|}{|S|} H(S_R).$$

The algorithm selects the split with the highest IG (or lowest Gini).

Considerations for Decision Trees

Considerations for Decision Trees — Definition

- Pruning: Techniques like cost-complexity pruning can be applied to reduce overfitting by removing branches that have little importance.

Considerations for Decision Trees

Considerations for Decision Trees

Definition

- **Pruning**: Techniques like cost-complexity pruning can be applied to reduce overfitting by removing branches that have little importance.
- **Advantages**:

Considerations for Decision Trees

Considerations for Decision Trees — Definition

- Pruning: Techniques like cost-complexity pruning can be applied to reduce overfitting by removing branches that have little importance.
- Advantages:
  - Easy to interpret and visualize.

Considerations for Decision Trees

Considerations for Decision Trees — Definition

- Pruning: Techniques like cost-complexity pruning can be applied to reduce overfitting by removing branches that have little importance.
- Advantages:
  - Easy to interpret and visualize.
  - Handles both numerical and categorical data.

Considerations for Decision Trees

Considerations for Decision Trees

Definition

- Pruning: Techniques like cost-complexity pruning can be applied to reduce overfitting by removing branches that have little importance.
- Advantages:
  - Easy to interpret and visualize.
  - Handles both numerical and categorical data.
  - Non-parametric: does not assume any specific distribution of the data.

Considerations for Decision Trees

Considerations for Decision Trees — Definition

- Pruning: Techniques like cost-complexity pruning can be applied to reduce overfitting by removing branches that have little importance.
- Advantages:
  - Easy to interpret and visualize.
  - Handles both numerical and categorical data.
  - Non-parametric: does not assume any specific distribution of the data.
- Disadvantages:

Considerations for Decision Trees

Considerations for Decision Trees — Definition

- Pruning: Techniques like cost-complexity pruning can be applied to reduce overfitting by removing branches that have little importance.
- Advantages:
  - Easy to interpret and visualize.
  - Handles both numerical and categorical data.
  - Non-parametric: does not assume any specific distribution of the data.
- Disadvantages:
  - Prone to overfitting, especially with deep trees.

Considerations for Decision Trees

Considerations for Decision Trees — Definition

- Pruning: Techniques like cost-complexity pruning can be applied to reduce overfitting by removing branches that have little importance.
- Advantages:
  - Easy to interpret and visualize.
  - Handles both numerical and categorical data.
  - Non-parametric: does not assume any specific distribution of the data.
- Disadvantages:
  - Prone to overfitting, especially with deep trees.
  - Sensitive to small changes in the data (can lead to different trees).

Support Vector Machines (SVM)

Support Vector Machines (SVM) — Definition

Support Vector Machines (SVM) are powerful algorithms that find the optimal hyperplane that separates different classes in the feature space. The goal is to maximize the margin between the closest points of each class, known as support vectors.

Support Vector Machines (SVM)

Support Vector Machines (SVM) — Definition

Support Vector Machines (SVM) are powerful algorithms that find the optimal hyperplane that separates different classes in the feature space. The goal is to maximize the margin between the closest points of each class, known as support vectors. The SVM algorithm works by solving the following optimization problem:

$$\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2 \quad \text{s.t.} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad \forall i,$$

where $\mathbf{w}$ is the weight vector, b is the bias term, and y_i is the class label of the i-th training example.

Support Vector Machines (SVM)

Support Vector Machines (SVM) — Definition

Support Vector Machines (SVM) are powerful algorithms that find the optimal hyperplane that separates different classes in the feature space. The goal is to maximize the margin between the closest points of each class, known as support vectors. The SVM algorithm works by solving the following optimization problem:

$$\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2 \quad \text{s.t.} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad \forall i,$$

where $\mathbf{w}$ is the weight vector, b is the bias term, and y_i is the class label of the i-th training example.

Visualization should help. . .

Linear SVMs

Linear SVMs — Definition

Linear SVMs are used when the data is linearly separable. The decision boundary is a hyperplane defined by the equation:

$$\mathbf{w}^T\mathbf{x} + b = 0,$$

where $\mathbf{w}$ is the weight vector and b is the bias term. The SVM finds the hyperplane that maximizes the margin between the two classes.

Non-Linear SVMs

Non-Linear SVMs — Definition

Non-Linear SVMs use kernel functions to transform the input space into a higher-dimensional space where a linear hyperplane can separate the classes. Common kernels include:

- Polynomial Kernel: Maps the input features into a polynomial feature space.
- Radial Basis Function (RBF) Kernel: Maps the input features into an infinite-dimensional space, allowing for complex decision boundaries.
- Sigmoid Kernel: Similar to the activation function in neural networks, it maps the input features into a space that can capture non-linear relationships.

Kernel Trick

Kernel Trick — Definition

The Kernel Trick allows SVMs to operate in high-dimensional feature spaces without explicitly computing the coordinates of the data in that space. Instead, it computes the inner products between the images of all pairs of data points in the feature space, which is computationally efficient.

This is done using a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ that computes the inner product in the transformed space:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j),$$

where ϕ is the mapping function that transforms the input features into the higher-dimensional space.