# 4 Types of Classification

**https://machinelearningmastery.com/types-of-classification-in-machine-learning/**

**Tutorial Overview**

This tutorial is divided into five parts; they are:

1. Classification Predictive Modeling
2. Binary Classification
3. Multi-Class Classification
4. Multi-Label Classification
5. Imbalanced Classification

**Classification Predictive Modeling**

In machine learning, [classification](https://en.wikipedia.org/wiki/Statistical_classification) refers to a predictive modeling problem where a class label is predicted for a given example of input data.

Examples of classification problems include:

* Given an example, classify if it is spam or not.
* Given a handwritten character, classify it as one of the known characters.
* Given recent user behavior, classify as churn or not.

From a modeling perspective, classification requires a training dataset with many examples of inputs and outputs from which to learn.

A model will use the training dataset and will calculate how to best map examples of input data to specific class labels. As such, the training dataset must be sufficiently representative of the problem and have many examples of each class label.

Class labels are often string values, e.g. “*spam*,” “*not spam*,” and must be mapped to numeric values before being provided to an algorithm for modeling. This is often referred to as [label encoding](https://machinelearningmastery.com/how-to-prepare-categorical-data-for-deep-learning-in-python/), where a unique integer is assigned to each class label, e.g. “*spam*” = 0, “*no spam*” = 1.

There are many different types of classification algorithms for modeling classification predictive modeling problems.

There is no good theory on how to map algorithms onto problem types; instead, it is generally recommended that a practitioner use controlled experiments and discover which algorithm and algorithm configuration results in the best performance for a given classification task.

Classification predictive modeling algorithms are evaluated based on their results. Classification accuracy is a popular metric used to evaluate the performance of a model based on the predicted class labels. [Classification accuracy is not perfect](https://machinelearningmastery.com/failure-of-accuracy-for-imbalanced-class-distributions/) but is a good starting point for many classification tasks.

Instead of class labels, some tasks may require the prediction of a [probability of class membership](https://machinelearningmastery.com/probability-metrics-for-imbalanced-classification/) for each example. This provides additional uncertainty in the prediction that an application or user can then interpret. A popular diagnostic for evaluating predicted probabilities is the [ROC Curve](https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/).

There are perhaps four main types of classification tasks that you may encounter; they are:

* Binary Classification
* Multi-Class Classification
* Multi-Label Classification
* Imbalanced Classification

**Binary Classification**

[Binary classification](https://en.wikipedia.org/wiki/Binary_classification) refers to those classification tasks that have two class labels.

Examples include:

* Email spam detection (spam or not).
* Churn prediction (churn or not).
* Conversion prediction (buy or not).

Typically, binary classification tasks involve one class that is the normal state and another class that is the abnormal state.

For example “*not spam*” is the normal state and “*spam*” is the abnormal state. Another example is “*cancer not detected*” is the normal state of a task that involves a medical test and “*cancer detected*” is the abnormal state.

The class for the normal state is assigned the class label 0 and the class with the abnormal state is assigned the class label 1.

It is common to model a binary classification task with a model that predicts a [Bernoulli probability distribution](https://machinelearningmastery.com/discrete-probability-distributions-for-machine-learning/) for each example.

The Bernoulli distribution is a discrete probability distribution that covers a case where an event will have a binary outcome as either a 0 or 1. For classification, this means that the model predicts a probability of an example belonging to class 1, or the abnormal state.

Popular algorithms that can be used for binary classification include:

* Logistic Regression
* k-Nearest Neighbors
* Decision Trees
* Support Vector Machine
* Naive Bayes

Some algorithms are specifically designed for binary classification and do not natively support more than two classes; examples include Logistic Regression and Support Vector Machines.

Next, let’s take a closer look at a dataset to develop an intuition for binary classification problems.

We can use the [make\_blobs() function](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_blobs.html) to generate a synthetic binary classification dataset.

The example below generates a dataset with 1,000 examples that belong to one of two classes, each with two input features.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | # example of binary classification task  from numpy import where  from collections import Counter  from sklearn.datasets import make\_blobs  from matplotlib import pyplot  # define dataset  X, y = make\_blobs(n\_samples=1000, centers=2, random\_state=1)  # summarize dataset shape  print(X.shape, y.shape)  # summarize observations by class label  counter = Counter(y)  print(counter)  # summarize first few examples  for i in range(10):  print(X[i], y[i])  # plot the dataset and color the by class label  for label, \_ in counter.items():  row\_ix = where(y == label)[0]  pyplot.scatter(X[row\_ix, 0], X[row\_ix, 1], label=str(label))  pyplot.legend()  pyplot.show() |

Running the example first summarizes the created dataset showing the 1,000 examples divided into input (*X*) and output (*y*) elements.

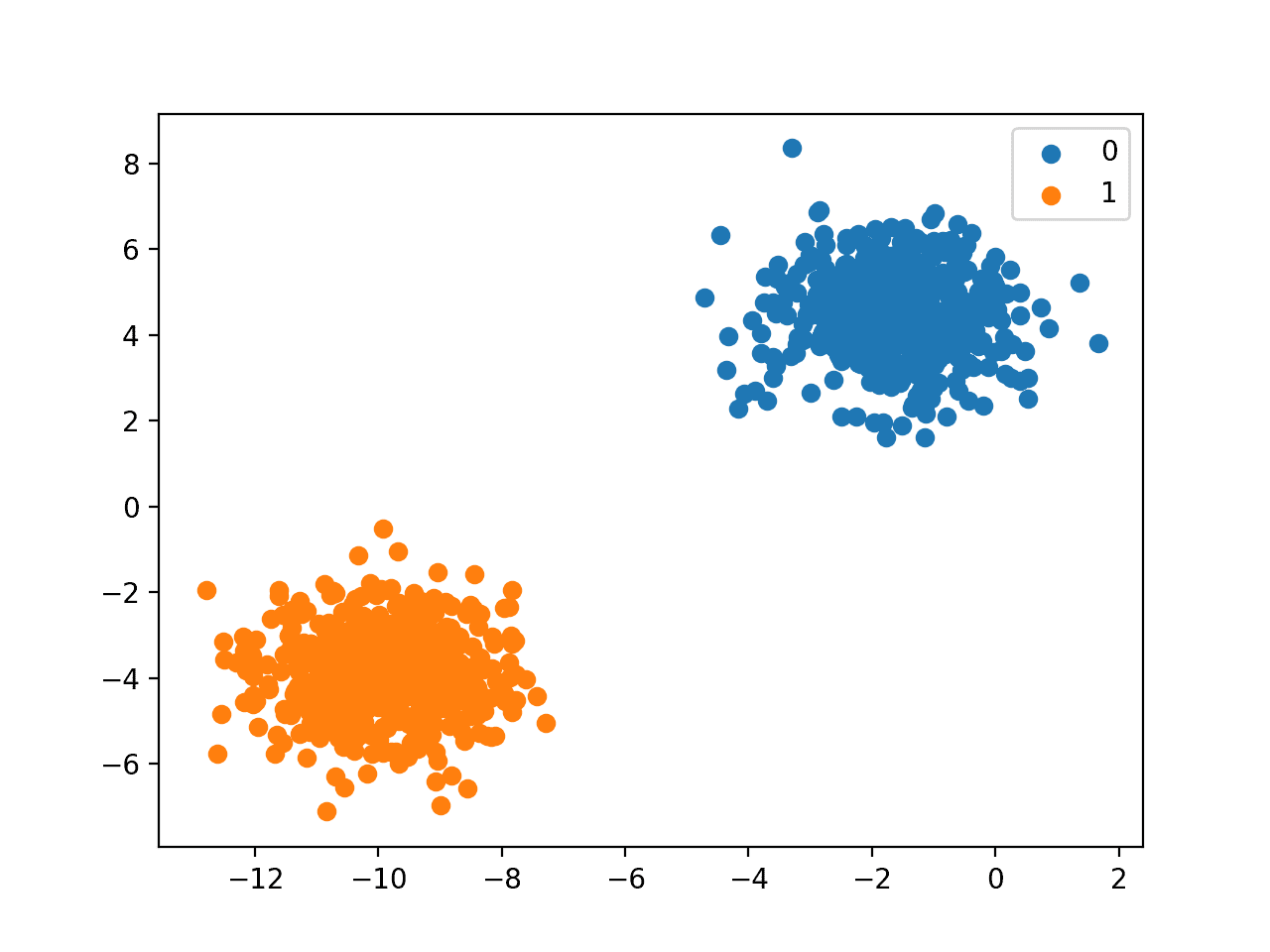
The distribution of the class labels is then summarized, showing that instances belong to either class 0 or class 1 and that there are 500 examples in each class.

Next, the first 10 examples in the dataset are summarized, showing the input values are numeric and the target values are integers that represent the class membership.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | (1000, 2) (1000,)    Counter({0: 500, 1: 500})    [-3.05837272  4.48825769] 0  [-8.60973869 -3.72714879] 1  [1.37129721 5.23107449] 0  [-9.33917563 -2.9544469 ] 1  [-11.57178593  -3.85275513] 1  [-11.42257341  -4.85679127] 1  [-10.44518578  -3.76476563] 1  [-10.44603561  -3.26065964] 1  [-0.61947075  3.48804983] 0  [-10.91115591  -4.5772537 ] 1 |

Finally, a scatter plot is created for the input variables in the dataset and the points are colored based on their class value.

We can see two distinct clusters that we might expect would be easy to discriminate.



Scatter Plot of Binary Classification Dataset

**Multi-Class Classification**

[Multi-class classification](https://en.wikipedia.org/wiki/Multiclass_classification) refers to those classification tasks that have more than two class labels.

Examples include:

* Face classification.
* Plant species classification.
* Optical character recognition.

Unlike binary classification, multi-class classification does not have the notion of normal and abnormal outcomes. Instead, examples are classified as belonging to one among a range of known classes.

The number of class labels may be very large on some problems. For example, a model may predict a photo as belonging to one among thousands or tens of thousands of faces in a face recognition system.

Problems that involve predicting a sequence of words, such as text translation models, may also be considered a special type of multi-class classification. Each word in the sequence of words to be predicted involves a multi-class classification where the size of the vocabulary defines the number of possible classes that may be predicted and could be tens or hundreds of thousands of words in size.

It is common to model a multi-class classification task with a model that predicts a [Multinoulli probability distribution](https://machinelearningmastery.com/discrete-probability-distributions-for-machine-learning/) for each example.

The Multinoulli distribution is a discrete probability distribution that covers a case where an event will have a categorical outcome, e.g. *K* in {1, 2, 3, …, *K*}. For classification, this means that the model predicts the probability of an example belonging to each class label.

Many algorithms used for binary classification can be used for multi-class classification.

Popular algorithms that can be used for multi-class classification include:

* k-Nearest Neighbors.
* Decision Trees.
* Naive Bayes.
* Random Forest.
* Gradient Boosting.

Algorithms that are designed for binary classification can be adapted for use for multi-class problems.

This involves using a strategy of fitting multiple binary classification models for each class vs. all other classes (called one-vs-rest) or one model for each pair of classes (called one-vs-one).

* **One-vs-Rest**: Fit one binary classification model for each class vs. all other classes.
* **One-vs-One**: Fit one binary classification model for each pair of classes.

Binary classification algorithms that can use these strategies for multi-class classification include:

* Logistic Regression.
* Support Vector Machine.

Next, let’s take a closer look at a dataset to develop an intuition for multi-class classification problems.

We can use the [make\_blobs() function](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_blobs.html) to generate a synthetic multi-class classification dataset.

The example below generates a dataset with 1,000 examples that belong to one of three classes, each with two input features.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | # example of multi-class classification task  from numpy import where  from collections import Counter  from sklearn.datasets import make\_blobs  from matplotlib import pyplot  # define dataset  X, y = make\_blobs(n\_samples=1000, centers=3, random\_state=1)  # summarize dataset shape  print(X.shape, y.shape)  # summarize observations by class label  counter = Counter(y)  print(counter)  # summarize first few examples  for i in range(10):  print(X[i], y[i])  # plot the dataset and color the by class label  for label, \_ in counter.items():  row\_ix = where(y == label)[0]  pyplot.scatter(X[row\_ix, 0], X[row\_ix, 1], label=str(label))  pyplot.legend()  pyplot.show() |

Running the example first summarizes the created dataset showing the 1,000 examples divided into input (*X*) and output (*y*) elements.

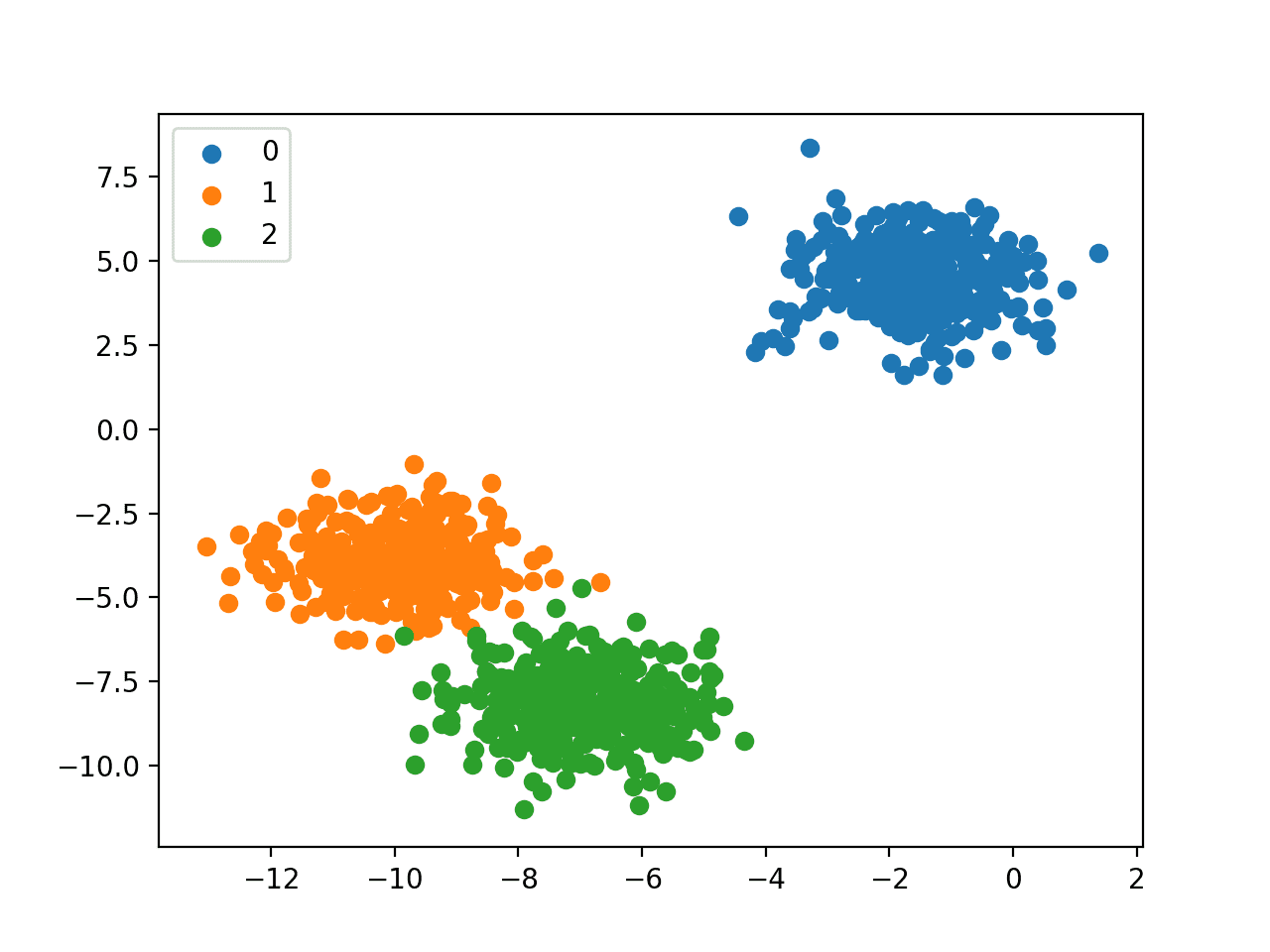
The distribution of the class labels is then summarized, showing that instances belong to class 0, class 1, or class 2 and that there are approximately 333 examples in each class.

Next, the first 10 examples in the dataset are summarized showing the input values are numeric and the target values are integers that represent the class membership.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | (1000, 2) (1000,)    Counter({0: 334, 1: 333, 2: 333})    [-3.05837272  4.48825769] 0  [-8.60973869 -3.72714879] 1  [1.37129721 5.23107449] 0  [-9.33917563 -2.9544469 ] 1  [-8.63895561 -8.05263469] 2  [-8.48974309 -9.05667083] 2  [-7.51235546 -7.96464519] 2  [-7.51320529 -7.46053919] 2  [-0.61947075  3.48804983] 0  [-10.91115591  -4.5772537 ] 1 |

Finally, a scatter plot is created for the input variables in the dataset and the points are colored based on their class value.

We can see three distinct clusters that we might expect would be easy to discriminate.



Scatter Plot of Multi-Class Classification Dataset

**Multi-Label Classification**

[Multi-label classification](https://en.wikipedia.org/wiki/Multi-label_classification) refers to those classification tasks that have two or more class labels, where one or more class labels may be predicted for each example.

Consider the example of [photo classification](https://machinelearningmastery.com/object-recognition-with-deep-learning/), where a given photo may have multiple objects in the scene and a model may predict the presence of multiple known objects in the photo, such as “*bicycle*,” “*apple*,” “*person*,” etc.

This is unlike binary classification and multi-class classification, where a single class label is predicted for each example.

It is common to model multi-label classification tasks with a model that predicts multiple outputs, with each output taking predicted as a Bernoulli probability distribution. This is essentially a model that makes multiple binary classification predictions for each example.

Classification algorithms used for binary or multi-class classification cannot be used directly for multi-label classification. Specialized versions of standard classification algorithms can be used, so-called multi-label versions of the algorithms, including:

* Multi-label Decision Trees
* Multi-label Random Forests
* Multi-label Gradient Boosting

Another approach is to use a separate classification algorithm to predict the labels for each class.

Next, let’s take a closer look at a dataset to develop an intuition for multi-label classification problems.

We can use the [make\_multilabel\_classification() function](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_classification.html) to generate a synthetic multi-label classification dataset.

The example below generates a dataset with 1,000 examples, each with two input features. There are three classes, each of which may take on one of two labels (0 or 1).

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | # example of a multi-label classification task  from sklearn.datasets import make\_multilabel\_classification  # define dataset  X, y = make\_multilabel\_classification(n\_samples=1000, n\_features=2, n\_classes=3, n\_labels=2, random\_state=1)  # summarize dataset shape  print(X.shape, y.shape)  # summarize first few examples  for i in range(10):  print(X[i], y[i]) |

Running the example first summarizes the created dataset showing the 1,000 examples divided into input (*X*) and output (*y*) elements.

Next, the first 10 examples in the dataset are summarized showing the input values are numeric and the target values are integers that represent the class label membership.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | (1000, 2) (1000, 3)    [18. 35.] [1 1 1]  [22. 33.] [1 1 1]  [26. 36.] [1 1 1]  [24. 28.] [1 1 0]  [23. 27.] [1 1 0]  [15. 31.] [0 1 0]  [20. 37.] [0 1 0]  [18. 31.] [1 1 1]  [29. 27.] [1 0 0]  [29. 28.] [1 1 0] |

**Imbalanced Classification**

[Imbalanced classification](https://machinelearningmastery.com/what-is-imbalanced-classification/) refers to classification tasks where the number of examples in each class is unequally distributed.

Typically, imbalanced classification tasks are binary classification tasks where the majority of examples in the training dataset belong to the normal class and a minority of examples belong to the abnormal class.

Examples include:

* Fraud detection.
* Outlier detection.
* Medical diagnostic tests.

These problems are modeled as binary classification tasks, although may require specialized techniques.

Specialized techniques may be used to change the composition of samples in the training dataset by undersampling the majority class or oversampling the minority class.

Examples include:

* [Random Undersampling](https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/).
* [SMOTE Oversampling](https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/).

Specialized modeling algorithms may be used that pay more attention to the minority class when fitting the model on the training dataset, such as cost-sensitive machine learning algorithms.

Examples include:

* [Cost-sensitive Logistic Regression](https://machinelearningmastery.com/cost-sensitive-logistic-regression/).
* Cost-sensitive Decision Trees.
* Cost-sensitive Support Vector Machines.

Finally, alternative performance metrics may be required as reporting the classification accuracy may be misleading.

Examples include:

* Precision.
* Recall.
* F-Measure.

Next, let’s take a closer look at a dataset to develop an intuition for imbalanced classification problems.

We can use the [make\_classification() function](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_classification.html) to generate a synthetic imbalanced binary classification dataset.

The example below generates a dataset with 1,000 examples that belong to one of two classes, each with two input features.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | # example of an imbalanced binary classification task  from numpy import where  from collections import Counter  from sklearn.datasets import make\_classification  from matplotlib import pyplot  # define dataset  X, y = make\_classification(n\_samples=1000, n\_features=2, n\_informative=2, n\_redundant=0, n\_classes=2, n\_clusters\_per\_class=1, weights=[0.99,0.01], random\_state=1)  # summarize dataset shape  print(X.shape, y.shape)  # summarize observations by class label  counter = Counter(y)  print(counter)  # summarize first few examples  for i in range(10):  print(X[i], y[i])  # plot the dataset and color the by class label  for label, \_ in counter.items():  row\_ix = where(y == label)[0]  pyplot.scatter(X[row\_ix, 0], X[row\_ix, 1], label=str(label))  pyplot.legend()  pyplot.show() |

Running the example first summarizes the created dataset showing the 1,000 examples divided into input (*X*) and output (*y*) elements.

The distribution of the class labels is then summarized, showing the severe class imbalance with about 980 examples belonging to class 0 and about 20 examples belonging to class 1.

Next, the first 10 examples in the dataset are summarized showing the input values are numeric and the target values are integers that represent the class membership. In this case, we can see that most examples belong to class 0, as we expect.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | (1000, 2) (1000,)    Counter({0: 983, 1: 17})    [0.86924745 1.18613612] 0  [1.55110839 1.81032905] 0  [1.29361936 1.01094607] 0  [1.11988947 1.63251786] 0  [1.04235568 1.12152929] 0  [1.18114858 0.92397607] 0  [1.1365562  1.17652556] 0  [0.46291729 0.72924998] 0  [0.18315826 1.07141766] 0  [0.32411648 0.53515376] 0 |

Finally, a scatter plot is created for the input variables in the dataset and the points are colored based on their class value.

We can see one main cluster for examples that belong to class 0 and a few scattered examples that belong to class 1. The intuition is that datasets with this property of imbalanced class labels are more challenging to model.

