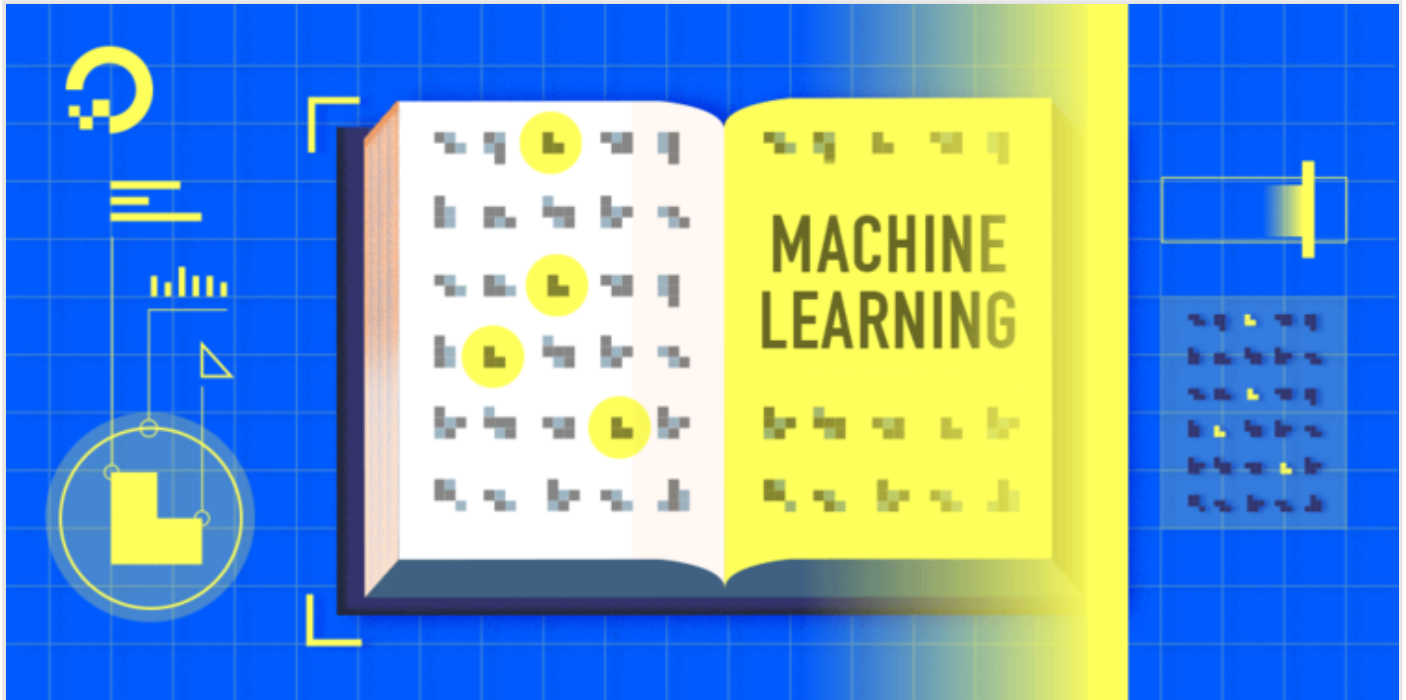




Subscribe

Share

Contents



# How To Build a Machine Learning Classifier in Python with Scikit-learn

  
22

Posted August 3, 2017 112.9k

PYTHON

DEVELOPMENT

PROGRAMMING PROJECT

DATA ANALYSIS

MACHINE LEARNING

By: Michelle Morales

## Introduction

Machine learning is a research field in computer science, artificial intelligence, and statistics. The focus of machine learning is to train algorithms to learn patterns and make predictions from data. Machine learning is especially valuable because it lets us use computers to automate decision-making processes.

SCROLL TO TOP

You'll find machine learning applications everywhere. Netflix and Amazon use machine learning to make new product recommendations. Banks use machine learning to detect fraudulent activity in credit card transactions, and healthcare companies are beginning to use machine learning to monitor, assess, and diagnose patients.

In this tutorial, you'll implement a simple machine learning algorithm in Python using [Scikit-learn](#), a machine learning tool for Python. Using a database of breast cancer tumor information, you'll use a [Naive Bayes \(NB\)](#) classifier that predicts whether or not a tumor is malignant or benign.

By the end of this tutorial, you'll know how to build your very own machine learning model in Python.

## Prerequisites

To complete this tutorial, you will need:

- Python 3 and a local programming environment set up on your computer. You can follow the [appropriate installation and set up guide for your operating system](#) to configure this.
  - If you are new to Python, you can explore [How to Code in Python 3](#) to get familiar with the language.
- [Jupyter Notebook](#) installed in the virtualenv for this tutorial. Jupyter Notebooks are extremely useful when running machine learning experiments. You can run short blocks of code and see the results quickly, making it easy to test and debug your code.

## Step 1 — Importing Scikit-learn

Let's begin by installing the Python module [Scikit-learn](#), one of the best and most documented machine learning libraries for Python.

To begin our coding project, let's activate our Python 3 programming environment. Make sure you're in the directory where your environment is located, and run the following command:

```
$ . my_env/bin/activate
```

With our programming environment activated, check to see if the Scikit-learn module is already installed:

```
(my_env) $ python -c "import sklearn"
```

[SCROLL TO TOP](#)

If `sklearn` is installed, this command will complete with no error. If it is not installed, you will see the following error message:

#### Output

```
Traceback (most recent call last): File "<string>", line 1, in <module> ImportError: No module
```

The error message indicates that `sklearn` is not installed, so download the library using `pip`:

```
(my_env) $ pip install scikit-learn[alldeps]
```

Once the installation completes, launch Jupyter Notebook:

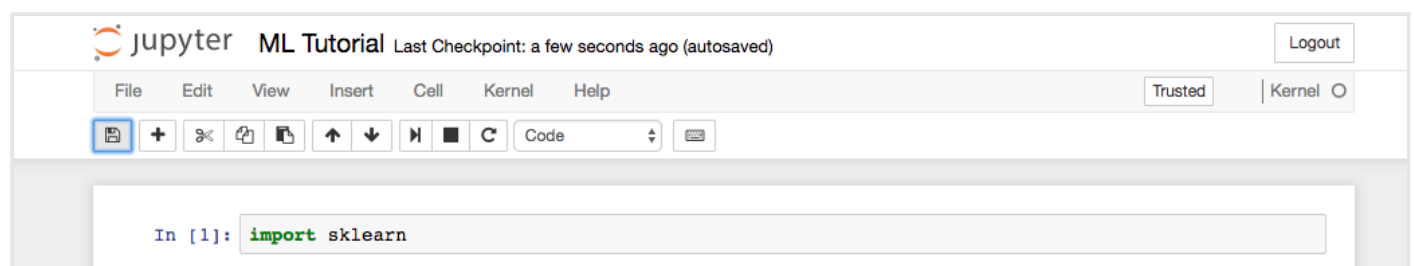
```
(my_env) $ jupyter notebook
```

In Jupyter, create a new Python Notebook called **ML Tutorial**. In the first cell of the Notebook, import the `sklearn` module:

ML Tutorial

```
import sklearn
```

Your notebook should look like the following figure:



Now that we have `sklearn` imported in our notebook, we can begin working with the dataset for our machine learning model.

## Step 2 — Importing Scikit-learn's Dataset

The dataset we will be working with in this tutorial is the Breast Cancer Wisconsin Diagnostic Database. The dataset includes various information about breast cancer tumors, classification labels of **malignant** or **benign**. The dataset has 569 *instances*, or data, on 369 tumors

SCROLL TO TOP

and includes information on 30 *attributes*, or features, such as the radius of the tumor, texture, smoothness, and area.

Using this dataset, we will build a machine learning model to use tumor information to predict whether or not a tumor is malignant or benign.

Scikit-learn comes installed with various datasets which we can load into Python, and the dataset we want is included. Import and load the dataset:

#### ML Tutorial

```
...  
  
from sklearn.datasets import load_breast_cancer  
  
# Load dataset  
data = load_breast_cancer()
```

The `data` variable represents a Python object that works like a dictionary. The important dictionary keys to consider are the classification label names (`target_names`), the actual labels (`target`), the attribute/feature names (`feature_names`), and the attributes (`data`).

Attributes are a critical part of any classifier. Attributes capture important characteristics about the nature of the data. Given the label we are trying to predict (malignant versus benign tumor), possible useful attributes include the size, radius, and texture of the tumor.

Create new variables for each important set of information and assign the data:

#### ML Tutorial

```
...  
  
# Organize our data  
label_names = data['target_names']  
labels = data['target']  
feature_names = data['feature_names']  
features = data['data']
```

We now have lists for each set of information. To get a better understanding of our dataset, let's take a look at our data by printing our class labels, the first data instance's label, our feature names, and the feature values for the first data instance:

[SCROLL TO TOP](#)

## ML Tutorial

```
...

# Look at our data
print(label_names)
print(labels[0])
print(feature_names[0])
print(features[0])
```

You'll see the following results if you run the code:

```
In [3]: # Look at our data
print(label_names)
print(labels[0])
print(feature_names[0])
print(features[0])

['malignant' 'benign']
0
mean radius
[ 1.79900000e+01  1.03800000e+01  1.22800000e+02  1.00100000e+03
 1.18400000e-01  2.77600000e-01  3.00100000e-01  1.47100000e-01
 2.41900000e-01  7.87100000e-02  1.09500000e+00  9.05300000e-01
 8.58900000e+00  1.53400000e+02  6.39900000e-03  4.90400000e-02
 5.37300000e-02  1.58700000e-02  3.00300000e-02  6.19300000e-03
 2.53800000e+01  1.73300000e+01  1.84600000e+02  2.01900000e+03
 1.62200000e-01  6.65600000e-01  7.11900000e-01  2.65400000e-01
 4.60100000e-01  1.18900000e-01]
```

As the image shows, our class names are **malignant** and **benign**, which are then mapped to binary values of **0** and **1**, where **0** represents malignant tumors and **1** represents benign tumors. Therefore, our first data instance is a malignant tumor whose *mean radius* is **1.79900000e+01**.

Now that we have our data loaded, we can work with our data to build our machine learning classifier.

## Step 3 — Organizing Data into Sets

To evaluate how well a classifier is performing, you should always test the model on unseen data. Therefore, before building a model, split your data into two parts: a *training set* and a *test set*.

You use the training set to train and evaluate the model during the development stage. You then use the trained model to make predictions on the unseen test set. This approach gives you a sense of the model's performance and robustness.

Fortunately, `sklearn` has a function called `train_test_split()`, which divides your data into these sets. Import the function and then use it to split the data:

[SCROLL TO TOP](#)

## ML Tutorial

...

```
from sklearn.model_selection import train_test_split

# Split our data
train, test, train_labels, test_labels = train_test_split(features,
                                                         labels,
                                                         test_size=0.33,
                                                         random_state=42)
```

The function randomly splits the data using the `test_size` parameter. In this example, we now have a test set (`test`) that represents 33% of the original dataset. The remaining data (`train`) then makes up the training data. We also have the respective labels for both the train/test variables, i.e. `train_labels` and `test_labels`.

We can now move on to training our first model.

## Step 4 — Building and Evaluating the Model

There are many models for machine learning, and each model has its own strengths and weaknesses. In this tutorial, we will focus on a simple algorithm that usually performs well in binary classification tasks, namely Naive Bayes (NB).

First, import the `GaussianNB` module. Then initialize the model with the `GaussianNB()` function, then train the model by fitting it to the data using `gnb.fit()`:

## ML Tutorial

...

```
from sklearn.naive_bayes import GaussianNB

# Initialize our classifier
gnb = GaussianNB()

# Train our classifier
model = gnb.fit(train, train_labels)
```

After we train the model, we can then use the trained model to make predictions which we do using the `predict()` function. The `predict()` function returns an array of predicted class labels.

[SCROLL TO TOP](#)

for each data instance in the test set. We can then print our predictions to get a sense of what the model determined.

Use the `predict()` function with the `test` set and print the results:

#### ML Tutorial

...

```
# Make predictions
preds = gnb.predict(test)
print(preds)
```

Run the code and you'll see the following results:

```
# initialize our classifier
gnb = GaussianNB()

# Train our classifier
model = gnb.fit(train, train_labels)

In [6]: # Make predictions
preds = gnb.predict(test)
print(preds)

[1 0 0 1 1 0 0 0 1 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0
 1 0 1 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 0 1 1 0 0 1 1 1 0 0 1 1 0 0 1 0
 1 1 1 1 1 1 0 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0 1 1 0
 1 1 0 0 0 1 1 1 0 0 1 1 0 1 0 0 1 1 0 0 0 1 1 1 0 1 1 0 0 1 0 1 1 0 1 0 0
 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 0 1 1 1 1 1 1 1 1 0 0
 0 1 1]
```

As you see in the Jupyter Notebook output, the `predict()` function returned an array of 0s and 1s which represent our predicted values for the tumor class (malignant vs. benign).

Now that we have our predictions, let's evaluate how well our classifier is performing.

## Step 5 — Evaluating the Model's Accuracy

Using the array of true class labels, we can evaluate the accuracy of our model's predicted values by comparing the two arrays (`test_labels` vs. `preds`). We will use the `sklearn` function `accuracy_score()` to determine the accuracy of our machine learning classifier.

#### ML Tutorial

...

```
from sklearn.metrics import accuracy_score
```

[SCROLL TO TOP](#)

```
# Evaluate accuracy
print(accuracy_score(test_labels, preds))
```

You'll see the following results:

```
In [7]: from sklearn.metrics import accuracy_score

# Evaluate accuracy
print(accuracy_score(test_labels, preds))

0.941489361702
```

As you see in the output, the NB classifier is 94.15% accurate. This means that 94.15 percent of the time the classifier is able to make the correct prediction as to whether or not the tumor is malignant or benign. These results suggest that our feature set of 30 attributes are good indicators of tumor class.

You have successfully built your first machine learning classifier. Let's reorganize the code by placing all `import` statements at the top of the Notebook or script. The final version of the code should look like this:

#### ML Tutorial

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

# Load dataset
data = load_breast_cancer()

# Organize our data
label_names = data['target_names']
labels = data['target']
feature_names = data['feature_names']
features = data['data']

# Look at our data
print(label_names)
print('Class label = ', labels[0])
print(feature_names)
print(features[0])

# Split our data
train, test, train_labels, test_labels = train_test_split(features,
```

[SCROLL TO TOP](#)



```
labels,  
test_size=0.33,  
random_state=42)  
  
# Initialize our classifier  
gnb = GaussianNB()  
  
# Train our classifier  
model = gnb.fit(train, train_labels)  
  
# Make predictions  
preds = gnb.predict(test)  
print(preds)  
  
# Evaluate accuracy  
print(accuracy_score(test_labels, preds))
```

Now you can continue to work with your code to see if you can make your classifier perform even better. You could experiment with different subsets of features or even try completely different algorithms. Check out [Scikit-learn's website](#) for more machine learning ideas.

## Conclusion

In this tutorial, you learned how to build a machine learning classifier in Python. Now you can load data, organize data, train, predict, and evaluate machine learning classifiers in Python using Scikit-learn. The steps in this tutorial should help you facilitate the process of working with your own data in Python.

By: Michelle Morales

♡ Upvote (22)

🔖 Subscribe

🔗 Share



Editor:  
Brian Hogan

SCROLL TO TOP

# Kubernetes ~~is hard.~~

We just made it easier for you to deploy faster.

TRY FREE

## Related Tutorials

[How To Install pygame and Create a Template for Developing Games in Python 3](#)

[How To Create a Twitterbot with Python 3 and the Tweepy Library](#)

[How To Create a Twitter App](#)

[How To Make a Simple Calculator Program in Python 3](#)

[How To Crawl A Web Page with Scrapy and Python 3](#)

## 7 Comments

**B** *I*      



Leave a comment...

Log In to Comment

SCROLL TO TOP

^ [abdelouahabb](#) August 12, 2017



0 Thank you for this wonderful tutorial.

but as a programmer, what comes next? I mean, what I would use for my program ? the training is the step to get the best parameters to use, the training step takes time, and the user will not stay everytime, he needs the best parameters to run the model, so how to do that?

---

^ [sandeshrjain](#) August 15, 2017



1 How can we load our **own dataset**\*\* instead breast cancer

---

^ [esteban9620](#) December 29, 2017



1 Hello, good afternoon, I am new to this from the sklearn bookstore and I already studied the tutorial, very good the truth, but I want to know if after having our training set as I can make a prediction, how could a possible cancer enter and this one tell me Is it benign or malefic? That's my question, how can I use the training set?

thank you very much for your attention.

---

^ [tntgalicia](#) April 8, 2018



0 Its amazing tuto, simple and coicisius

---

^ [ahmetturkmen](#) June 22, 2018



0 Clear, easy to implement and follow . Thank you very much for such a great article.

---

^ [sazzadur](#) December 6, 2018



0 Brilliant article. It is clearer than my class room learning.

---

^ [juliettetworsey](#) December 31, 2018



0 Thank you for this tutorial Brian!!


SCROLL TO TOP



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.



Copyright © 2019 DigitalOcean™ Inc.

[Community](#) [Tutorials](#) [Questions](#) [Projects](#) [Tags](#) [Newsletter](#) [RSS](#) 

[Distros & One-Click Apps](#) [Terms, Privacy, & Copyright](#) [Security](#) [Report a Bug](#) [Write for DOnations](#) [Shop](#)

SCROLL TO TOP