XGBoost With Python

Gradient Boosted Trees With XGBoost and scikit-learn

Jason Brownlee

MACHINE LEARNING MASTERY

Disclaimer

Acknowledgements

Special thanks to my technical editor John Halfyard.

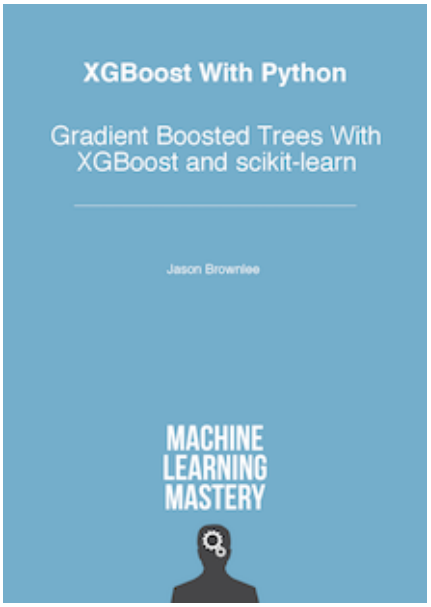
Copyright

XGBoost With Python

Edition: v1.15

This is Just a Sample

Thank-you for your interest in **XGBoost With Python**.
This is just a sample of the full text. You can purchase the complete book online from:
https://machinelearningmastery.com/xgboost-with-python/



Contents

Chapter 1

Welcome

Welcome to XGBoost With Python. This book is your guide to fast gradient boosting in Python. You will discover the XGBoost Python library for gradient boosting and how to use it to develop and evaluate gradient boosting models. In this book you will discover the techniques, recipes and skills with XGBoost that you can then bring to your own machine learning projects.

Gradient Boosting does have some fascinating math under the covers, but you do not need to know it to be able to pick it up as a tool and wield it on important projects to deliver real value. From the applied perspective, gradient boosting is quite a shallow field and a motivated developer can quickly pick it up and start making very real and impactful contributions. This is my goal for you and this book is your ticket to that outcome.

1.1 Book Organization

The tutorials in this book are divided into three parts:

- **XGBoost Basics**.
- **XGBoost Advanced**.
- **XGBoost Tuning**.

In addition to these three parts, the Conclusions part at the end of the book includes a list of resources for getting help and diving deeper into the field.

1.1.1 XGBoost Basics

This part provides a gentle introduction to the XGBoost library for use with the scikit-learn library in Python. After completing the tutorials in this section, you will know your way around basic XGBoost models. Specifically, this part covers:

- A gentle introduction to the gradient boosting algorithm.
- A gentle introduction to the XGBoost library and why it is so popular.
- How to develop your first XGBoost model from scratch.

- How to prepare data for use with XGBoost.
- How to evaluate the performance of trained XGBoost models.
- How to visualize boosted trees within an XGBoost model.

1.1.2 XGBoost Advanced

This part provides an introduction to some of the more advanced features and uses of the XGBoost library. After completing the tutorials in this section you will know how to implement some of the more advanced capabilities of gradient boosting and scale up your models for bigger hardware platforms. Specifically, this part covers:

- How to serialize trained models to file and later load and use them to make predictions.
- How to calculate importance scores and use them for feature selection.
- How to monitor the performance of a model during training and set conditions for early stopping.
- How to harness the parallel features of the XGBoost library for training models faster.
- How to rapidly speed up model training of XGBoost models using Amazon cloud infrastructure.

1.1.3 XGBoost Tuning

This part provides tutorials detailing how to configure and tune XGBoost hyperparameters. After completing the tutorials in this part, you will know how to design parameter tuning experiments to get the most from your models. Specifically, this part covers:

- An introduction to XGBoost parameters and heuristics for good parameter values.
- How to tune the number and size of trees in a model.
- How to tune the learning rate and number of trees in a model.
- How to tune the sampling rates in stochastic variation of the algorithm.

1.1.4 Python Recipes

Building up a catalog of code recipes is an important part of your XGBoost journey. Each time you learn about a new technique or new problem type, you should write up a short code recipe that demonstrates it. This will give you a starting point to use on your next machine learning project.

As part of this book you will receive a catalog of XGBoost recipes. This includes recipes for all of the tutorials presented in this book. You are strongly encouraged to add to and build upon this catalog of recipes as you expand your use and knowledge of XGBoost in Python.

1.2 Requirements For This Book

1.2.1 Python and SciPy

You do not need to be a Python expert, but it would be helpful if you knew how to install and setup Python and SciPy. The tutorials assume that you have a Python and SciPy environment available. This may be on your workstation or laptop, it may be in a VM or a Docker instance that you run, or it may be a server instance that you can configure in the cloud as taught in Part III of this book.

Technical Requirements: The technical requirements for the code and tutorials in this book are as follows:

- Python version 2 or 3 installed.
- SciPy and NumPy installed.
- Matplotlib installed.
- Pandas installed.
- scikit-learn installed.

You do not need to match the version exactly, but if you are having problems running a specific code example, please ensure that you update to the same or higher version as the library specified.

1.2.2 Machine Learning

You do not need to be a machine learning expert, but it would be helpful if you knew how to navigate a small machine learning problem using scikit-learn. Basic concepts like cross-validation and one hot encoding used in tutorials are described, but only briefly. There are resources to go into these topics in more detail at the end of the book, but some knowledge of these areas might make things easier for you.

1.2.3 Gradient Boosting

You do not need to know the math and theory of gradient boosting algorithms, but it would be helpful to have some basic idea of the field. You will get a crash course in gradient boosting terminology and models, but we will not go into much technical detail. Again, there will be resources for more information at the end of the book, but it might be helpful if you can start with some idea about the technical details of the technique.

1.3 Your Outcomes From Reading This Book

This book will lead you from being a developer who is interested in XGBoost with Python to a developer who has the resources and capabilities to work through a new dataset end-to-end using Python and develop accurate gradient boosted models. Specifically, you will know:

- How to prepare data in scikit-learn for gradient boosting.
- How to evaluate and visualize gradient boosted models.
- How to save and load trained gradient boosted models.
- How to visualize and evaluate the importance of input variables.
- How to configure and tune hyperparameters of gradient boosted models.

There are a few ways you can read this book. You can dip into the tutorials as your need or interests motivate you. Alternatively, you can work through the book end-to-end and take advantage of how the tutorials build in complexity and range. I recommend the latter approach.

To get the very most from this book, I recommend working through each tutorial and then build upon them. Attempt to improve the results, apply the method to a similar but different problem, and so on. Write up what you tried or learned and share it on your blog, social media or send me an email at `jason@MachineLearningMastery.com`. This book is really what you make of it and by putting in a little extra, you can quickly become a true force in applied gradient boosting.

1.4 What This Book is Not

This book solves a specific problem of getting you, a developer, up to speed applying XGBoost to your own machine learning projects in Python. As such, this book was not intended to be everything to everyone and it is very important to calibrate your expectations. Specifically:

- **This is not a gradient boosting textbook**. We will not cover the basic theory of how algorithms and related techniques work. There will be no equations. You are also expected to have some familiarity with machine learning basics, or be able to dive deeper into the theory yourself, if needed.
- **This is not a Python programming book**. We will not be spending a lot of time on Python syntax and programming (e.g. basic programming tasks in Python). You are expected to already be familiar with Python or a developer who can pick up a new C-like language relatively quickly.

You can still get a lot out of this book if you are weak in one or two of these areas, but you may struggle picking up the language or require some more explanation of the techniques. If this is the case, see the *Getting More Help* chapter at the end of the book and seek out a good companion reference text.

1.5 Summary

It is a special time right now. The tools for fast gradient boosting have never been so good and XGBoost is the top of the stack. The pace of change with applied machine learning feels like it has never been so fast, spurred by the amazing results that the methods are showing in such a broad range of fields. This is the start of your journey into using XGBoost and I am excited for

you. Take your time, have fun and I'm so excited to see where you can take this amazing new technology.

Next in Part II, you will get a gentle introduction to the gradient boosting algorithm as described by primary sources. This will lay the foundation for understanding and using the XGBoost library in Python.

Chapter 2

Develop Your First XGBoost Model in Python with scikit-learn

XGBoost is an implementation of gradient boosted decision trees designed for speed and performance that is dominant in competitive machine learning. In this tutorial you will discover how you can install and create your first XGBoost model in Python. After reading this tutorial you will know:

- How to install XGBoost on your system for use in Python.
- How to prepare data and train your first XGBoost model.
- How to make predictions using your XGBoost model.

Let's get started.

2.1 Install XGBoost for Use in Python

Assuming you have a working SciPy environment, XGBoost can be installed easily using `pip`. For example:

```
sudo pip install xgboost
```

Listing 2.1: Install XGBoost using `pip`.

To update your installation of XGBoost you can type:

```
sudo pip install --upgrade xgboost
```

Listing 2.2: Update XGBoost using `pip`.

An alternate way to install XGBoost if you cannot use `pip` or you want to run the latest code from GitHub requires that you make a clone of the XGBoost project and perform a manual build and installation. For example to build XGBoost without multithreading on Mac OS X (with GCC already installed via macports or homebrew), you can type:

```
git clone --recursive https://github.com/dmlc/xgboost
cd xgboost
cp make/minimum.mk ./config.mk
make -j4
```

```
cd python-package
sudo python setup.py install
```

Listing 2.3: Build and Install XGBoost on Mac OS X.

Below are some resources to help you with the installation of the XGBoost library on your platform.

- You can learn more about how to install XGBoost for different platforms on the *XGBoost Installation Guide*.
  http://xgboost.readthedocs.io/en/latest/build.html
- For up-to-date instructions for installing XGBoost for Python see the *XGBoost Python Package*.
  https://github.com/dmlc/xgboost/tree/master/python-package

XGBoost v1.0.2 is the latest at the time of writing and is used in this book.

2.2 Problem Description: Predict Onset of Diabetes

In this tutorial we are going to use the Pima Indians onset of diabetes dataset. This dataset is comprised of 8 input variables that describe medical details of patients and one output variable to indicate whether the patient will have an onset of diabetes within 5 years. You can learn more about this dataset from the published dataset details[1].

This is a good dataset for a first XGBoost model because all of the input variables are numeric and the problem is a simple binary classification problem. It is not necessarily a good problem for the XGBoost algorithm because it is a relatively small dataset and an easy problem to model.

```
6,148,72,35,0,33.6,0.627,50,1
1,85,66,29,0,26.6,0.351,31,0
8,183,64,0,0,23.3,0.672,32,1
1,89,66,23,94,28.1,0.167,21,0
0,137,40,35,168,43.1,2.288,33,1
```

Listing 2.4: Sample of the Pima Indians dataset.

Download this dataset[2] and place it into your current working directory with the file name `pima-indians-diabetes.csv`.

2.3 Load and Prepare Data

In this section we will load the data from file and prepare it for use for training and evaluating an XGBoost model. We will start off by importing the classes and functions we intend to use in this tutorial.

[1] https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.names
[2] https://goo.gl/bDdBiA

```
from numpy import loadtxt
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

Listing 2.5: Import Classes and Functions.

Next, we can load the CSV file as a NumPy array using the NumPy function `loadtext()`.

```
...
# load data
dataset = loadtxt('pima-indians-diabetes.csv', delimiter=",")
```

Listing 2.6: Load the Dataset.

We must separate the columns (attributes or features) of the dataset into input patterns (`X`) and output patterns (`Y`). We can do this easily by specifying the column indices in the NumPy array format.

```
...
# split data into X and y
X = dataset[:,0:8]
Y = dataset[:,8]
```

Listing 2.7: Separate Dataset into X and Y.

Finally, we must split the `X` and `Y` data into a training and test dataset. The training set will be used to prepare the XGBoost model and the test set will be used to make new predictions, from which we can evaluate the performance of the model. For this we will use the `train_test_split()` function from the scikit-learn library. We also specify a seed for the random number generator so that we always get the same split of data each time this example is executed.

```
...
# split data into train and test sets
seed = 7
test_size = 0.33
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=test_size,
    random_state=seed)
```

Listing 2.8: Separate Dataset into Train and Test sets.

We are now ready to train our model.

2.4 Train the XGBoost Model

XGBoost provides a wrapper class to allow models to be treated like classifiers or regressors in the scikit-learn framework. This means we can use the full scikit-learn library with XGBoost models. The XGBoost model for classification is called `XGBClassifier`. We can create and fit it to our training dataset. Models are fit using the scikit-learn API and the `model.fit()` function. Parameters for training the model can be passed to the model in the constructor. Here, we use the sensible defaults.

```
...
# fit model on training data
model = XGBClassifier()
model.fit(X_train, y_train)
```

Listing 2.9: Fit the XGBoost model.

You can see the parameters used in a trained model by printing the model, for example:

```
...
print(model)
```

Listing 2.10: Summarize the XGBoost model.

We are now ready to use the trained model to make predictions.

2.5 Make Predictions with XGBoost Model

We can make predictions using the fit model on the test dataset. To make predictions we use the scikit-learn function `model.predict()`.

```
...
# make predictions for test data
predictions = model.predict(X_test)
```

Listing 2.11: Make Predictions with the XGBoost model.

Now that we have used the fit model to make predictions on new data, we can evaluate the performance of the predictions by comparing them to the expected values. For this we will use the built in `accuracy_score()` function in scikit-learn.

```
...
# evaluate predictions
accuracy = accuracy_score(y_test, predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

Listing 2.12: Evaluate Accuracy of XGBoost Predictions.

2.6 Tie it All Together

We can tie all of these pieces together, below is the full code listing.

```
# First XGBoost model for Pima Indians dataset
from numpy import loadtxt
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
# load data
dataset = loadtxt('pima-indians-diabetes.csv', delimiter=",")
# split data into X and y
X = dataset[:,0:8]
Y = dataset[:,8]
# split data into train and test sets
seed = 7
```

```
test_size = 0.33
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=test_size,
    random_state=seed)
# fit model on training data
model = XGBClassifier()
model.fit(X_train, y_train)
# make predictions for test data
predictions = model.predict(X_test)
# evaluate predictions
accuracy = accuracy_score(y_test, predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

Listing 2.13: Complete Working Example of Your First XGBoost Model.

Running this example produces the following output.

```
Accuracy: 77.95%
```

Listing 2.14: Sample Output From First XGBoost Model.

This is a good accuracy score on this problem, which we would expect, given the capabilities of the model and the modest complexity of the problem.

2.7 Summary

In this tutorial you discovered how to develop your first XGBoost model in Python. Specifically, you learned:

- How to install XGBoost on your system ready for use with Python.
- How to prepare data and train your first XGBoost model on a standard machine learning dataset.
- How to make predictions and evaluate the performance of a trained XGBoost model using scikit-learn.

In the next tutorial, you will build upon these skills and learn how to best prepare your data for modeling with XGBoost.

This is Just a Sample

Thank-you for your interest in **XGBoost With Python**.
This is just a sample of the full text. You can purchase the complete book online from:
https://machinelearningmastery.com/xgboost-with-python/