  
**Hands-on Supervised Machine Learning Algorithms**

Learn the underpinnings of many supervised learning algorithms, and develop rich python coding practices in the process.

Supervised learning—help teach a machine to think for itself!

# **AUTHOR - TAYLOR SMITH**



# **DURATION -**

# **OVERVIEW**

Supervised machine learning is used in a wide range of industries across sectors like finance, online advertising, analytics, and it’s here to stay. Supervised learning allows you to train your system to make price predictions, recommendations, and to make decisions on its own. Understanding the core principles that drive how a machine “learns” is a critical skill for any would-be practitioner or consumer alike.

This course will guide you through the implementation and nuances of many popular **supervised** machine learning algorithms while facilitating a deep understanding along the way.

You will embark on this journey with a quick course overview of how supervised machine learning differs from unsupervised learning. Next, you’ll learn about parametric models such as linear and logistic regression, and non-parametric methods such as decision trees & various clustering techniques. As we proceed, you will work hands-on with recommender systems, which are widely used by ecommerce sites to increase user interaction and enrich their shopping experience. Finally, you’ll wrap up with a brief foray into neural networks and transfer learning.

By the end of the video course, you will be equipped with intimate knowledge of how these algorithms work, and will possess the know-how needed to apply them to new problems.

# **TARGET AUDIENCE**

This course is suitable for developers/aspiring data scientists who want to enter the field of data science & are new to machine learning. Intermediate knowledge of Python is required for the best course experience.

# **KEY FEATURES**

* Take a deep dive into supervised learning, grasping how a machine “learns” from data
* Follow detailed and thorough coding examples to implement popular machine learning algorithms from scratch, developing a deep understanding along the way
* Work your Python muscle! This course will help you grow as a developer by heavily relying some of the most popular scientific and mathematical libraries in the Python language

# **APPROACH**

1. This course will be a step-by-step guide to understanding complex mathematical concepts in a practical fashion. This video course highlights clean coding techniques, object-oriented class design, and general best practices in machine learning.
2. Though solutions may exist (i.e., implementations in various other Python libraries), this course adheres to a “learning by doing” pattern.

# **WHAT WILL YOU LEARN**

1) Crack how a machine learns a concept and generalizes its understanding to new data

2) Uncover the fundamental and theoretical differences between parametric and non-parametric models. Distinguish why you might opt for one over the other.

3) Implement and grok several well-known supervised learning algorithms from scratch; build out your github portfolio and show off what you’re capable of!

4) Work with model families like recommender systems, which are immediately applicable in domains such as ecommerce and marketing.

5) Expand your expertise using various algorithms like regression, decision trees, clustering and many more to become a much stronger Python developer

6) Build your own models capable of making predictions

7) Delve into some of the most popular topics in machine learning, like neural networks and transfer learning

# **ABOUT THE AUTHOR -**

*Taylor smith is a machine learning enthusiast with over five years of experience who loves to apply interesting computational solutions to challenging business problems. Currently working as Principal Data-scientist, Taylor is also an active open-source contributor and staunch pythonista.*[*LinkedIn*](https://www.linkedin.com/in/taylorgsmith/) *and* [*Github*](https://github.com/tgsmith61591) *profile.*

# **SUMMARY OF CONTENTS**

This course will cover several broad families of supervised learning models, introducing you to the concepts and mathematics behind the algorithm before moving on to actually implementing them from scratch in Python.

1. **First steps towards Supervised Learning** *- In this section, we’ll get up and running with our Anaconda environment and cover the fundamentals of supervised learning, including loss functions and hill climbing.*
2. **Implementing parametric models** *– In this section we will discuss parametric models, their pros and cons, and will implement several commonly used algorithms from scratch.*
3. **Working with non-parametric models** – *Here we’ll introduce non-parametric models, which come with a whole host of new pros and cons. We will implement several popular non-parametric algorithms.*
4. **Advanced topics in supervised ML** *– In this section, we’ll cover more specific, nuanced families of machine learning algorithms such as recommender systems, neural networks and transfer learning.*

# **COURSE ROADMAP**

### **SECTION ONE – First steps towards supervised learning**

*In section 1, we’ll introduce supervised machine learning and how it differs from unsupervised machine learning. We’ll also discuss the concept of loss functions and hill climbing, and they are central to all machine learning algorithms we will cover in the course.*

* **Video 1 – Getting our environment setup.** In this video, we’ll cover a brief real-world example of machine learning and how it impacts our everyday lives, and then we’ll get up and running with Anaconda and create our development environment for future videos.
* **Video 2 – Supervised learning.** At this point, we’re ready to begin with the course material. In this video, we will introduce the definition of supervised machine learning and how it differs from other methods (namely unsupervised).
* **Video 3 – Hill climbing & loss functions.** At the end of the day, machine learning is simply applied mathematics. This video will cover the optimization functions that are core to machine learning algorithms and how they drive the behavior of various algorithms.
* **Video 4 – Model evaluation and data splitting.** Though this course is more geared towards the algorithmic “guts” of models, it’s important we at least acknowledge the importance of data splitting in model development and evaluation so the terminology makes sense throughout the course.

### **SECTION TWO – Implementing parametric models**

*This section will cover parametric models and their pros/cons. In addition, we’ll implement various models from scratch and introduce various concepts that are crucial to understanding how machine learning works.*

* **Video 1 – Introduction to parametric models & linear regression.** In this video we’ll begin by explaining parametric models, starting with the most simple: linear regression.
* **Video 2 – Implementing linear regression from scratch.** In this video, we will code a linear regression class from scratch in order to get a better understanding of how it works under the hood.
* **Video 3 – Introduction to logistic regression models.** Logistic regression is similar in spirit to linear regression, but is used for discrete (i.e., classification) outputs rather than continuous. In this video, we’ll introduce the concept and mathematics behind logistic regression models.
* **Video 4 – Implementing logistic regression from scratch.** In this lesson, we will code a logistic regression in python to learn how various hyper-parameters affect its ability to learn.
* **Video 5 – Parametric models—pros/cons.** Here we’ll examine some of the parametric models we’ve discussed and go over some of their common strengths and weaknesses.

**SECTION THREE – Working with non-parametric models**

*Now that we’ve covered parametric models and some of their pros and cons, it’s time to consider alternative algorithmic approaches that complement the former.*

* **Video 1 – The bias/variance trade-off.** At the end of section two, we hinted at some weaknesses of parametric models. This video will cover those weaknesses in greater detail while introducing a sliding spectrum of “bias” and “variance” that we’ll reference frequently.
* **Video 2 – Introduction to non-parametric models & decision trees.** One of the most ubiquitous non-parametric models is a decision tree. In this lesson, we’ll introduce non-parametric models in conjunction with the decision tree and how it works.
* **Video 3 – Decision trees part 2.** In this video, we’ll begin to explore some code and explain the internal working of decision trees, namely the splitting criteria.
* **Video 4 – Implementing a decision tree from scratch.** In this lesson we will pick up where we left off in the last video, hand-coding the remainder of the decision tree from scratch.
* **Video 5 – Introduction to clustering methods.** Another common non-parametric family of models is clustering. We will go over several common methods, honing in on one in particular: k-nearest neighbors.
* **Video 6 – Implementing K-nearest neighbors from scratch.** Here, we’ll write code for a kNN classifier, per the lesson in video 5.
* **Video 7 – Non-parametric models—pros/cons (when to use vs. parametric models).** In this video, we’ll summarize the common pros and cons of non-parametric models with respect to how to are impacted by the bias/variance trade-off, etc.

### **SECTION FOUR – Advanced topics in supervised ML**

*After having learned the difference between parametric and non-parametric models, as well as about the bias/variance trade-off and various other modeling considerations, you might be ready to apply some modeling techniques in the real world. However, there are several applications of machine learning that don’t necessarily fit into the aforementioned categories, or that are so specific that they were not included there. In this section, we will cover such topics outside of the context of an entire devoted section.*

* **Video 1 – Recommender systems & an introduction to collaborative filtering.** Recommender systems are a quickly emerging field of study in machine learning. In this lesson, we’ll introduce collaborative filtering (CF) techniques with an emphasis on item-to-item CF.
* **Video 2 – Collaborative filtering part 2: matrix factorization.** In this video, we’ll introduce a new form of collaborative filtering: matrix completion techniques. Here we’ll cover the motivation and benefits of a factorization approach.
* **Video 3 – Collaborative filtering in Python.** We’ll pick up where we left off in the last video, jumping right into a coding example of collaborative filtering, where we’ll design both an item-to-item collaborative filtering recommender as well as an Alternating Least Squares recommendation engine.
* **Video 4 – Content-based filtering.** Content-based filtering is an alternative to collaborative filtering. In this video, we will cover how content-based methods can supplement collaborative filtering methods.
* **Video 5 – Neural networks & “deep learning”.** Deep learning is another emerging field in data science and machine learning. Here we’ll cover an introduction to how multilayer perceptrons work at a mathematical level.
* **Video 6 – Neural networks part 2.** In this video, we’ll continue our exploration of neural networks, coding one from scratch in Python.
* **Video 7 – Introduction to transfer learning.** Now that we’ve learned about neural networks, we’ll cover an advanced concept: transfer learning. Though there are libraries that can handle transfer learning elegantly, in this video we’ll demonstrate how to perform transfer learning by hand.

# **SETUP AND INSTALLATION**

### Minimum Hardware Requirements

For successful completion of this course, students will require the computer systems with at least the following:

* OS: Mac OS, Linux or Windows
* Processor: N/A (any modern processor should be fine)
* Memory: 4GB memory
* Storage: 4GB

### Recommended Hardware Requirements

For an optimal experience with hands-on labs and other practical activities, we recommend the following configuration:

* OS: Mac OS, Linux or Windows
* Processor: 2.5GHz+
* Memory: 16GB
* Storage: 4GB

### Software Requirements

* Operating system: N/A (any will work)
* PyCharm (Community Edition) IDE, Latest Version, or your favorite text editor (<https://www.jetbrains.com/pycharm/download/>)
* Anaconda 4+ installed ([http://anaconda.com](http://anaconda.com/))