Machine Learning Foundations Unit 1: Machine Learning in a Nutshell

Table of Contents

Module Introduction: Use ML for Industrial Decision Making

Module Introduction: Recognize ML Problem Types

- Ask the Expert: Laurens van der Maaten on ML
- Quiz: Check Your Knowledge: Identifying ML Problem Types
- Assignment: Unit 1 Assignment - Part 2: Recognizing ML Problem Types
- Module Wrap-up: Recognize ML Problem Types

Module Introduction: The ML Life Cycle

- Watch: The ML Process: End to End
- Quiz: Check Your Knowledge: Identifying The Process
- Watch: Your Role as an MLE
- Watch: ML Problem Formulation
- Watch: Case Study: Recommendation Systems Revisited
- Assignment: Unit 1 Assignment - Part 3: The ML Life Cycle
- Module Wrap-up: The ML Life Cycle

Module Introduction: The ML Tech Stack

- Watch: ML Workflow Applications
- Watch: Demo: Linux
- Read: Machine Learning Workflow Applications
- Read: Useful Features of the IPython Interpreter
- Tool: The Structure of a Notebook
- Read: Coding Exercise Submissions and Grading
- Tool: Linux and IPython Cheat Sheet
- Assignment: Jupyter Notebook Practice
- Watch: ML Python Packages
- Tool: ML Python Packages
- Quiz: Check Your Knowledge: Identifying Which Package to Use Part 1
- Watch: Demo: Packages
- Assignment: Practice NumPy
- Tool: Numpy Array Tip Sheet
- Assignment: Practice Pandas
- Tool: Pandas DataFrame Cheat Sheet
- Quiz: Check Your Knowledge: Identifying Which Package to Use Part 2
- Module Wrap-up: The ML Tech Stack

Homepage

Video Transcript

If you're here watching this video, you're probably already aware that artificial intelligence and machine learning are important technologies. I'd go as far as saying they are critical in today's digital society. Nearly every action in today's economy involves data, and AI helps organizations more objectively and efficiently perform the core tasks, such as delivering content or assessing risk. AI can and will impact us all as consumers in this modern economy.

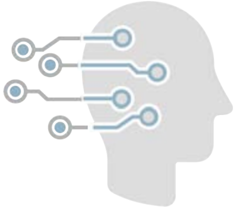
All of the AI is built by a small set of people with highly specialized and sought-after skills. Given the breadth of AI's reach, it is important that the people building it are able to understand the diverse perspectives and lived experiences of the people it affects. As the use of AI matures, we're hearing more and more about how AI might be used as a tool for oppression, silently capturing and reinforcing society's biases.

But AI-driven problems can be contained and constrained with responsible and inclusive design and testing guardrails that look for potential problems. This is an important part you will play and why diversity in the people who build AI systems is critical. Diversity in tech is how AI gets built for the people by the people.

What you'll do

- **Understand the machine learning life cycle and explore common machine learning packages**
- **Perform exploratory analysis to understand your data and prepare your data for machine learning applications**
- **Train and optimize two popular supervised learning algorithms: k-nearest neighbors and decision trees**
- **Understand the mechanics of linear models and implement a common linear model from scratch**
- **Define the model evaluation metrics for specific applications by selecting the appropriate model candidates and hyperparameters for testing**
- **Understand the principle of ensemble models and how to use them to improve model performance**
- **Explore the fields of computer vision and natural language processing then implement deep learning models to solve problems in these areas**

- **Identify performance issues and societal failures then find solutions to address these issues**



Course Description

Machine learning (ML) is increasingly a part of people's daily lives. Think about some of the technologies you use every day, such as the suggestions that appear on YouTube and emails being diverted to the spam folder. All are practical applications of ML, a branch of artificial intelligence (AI) that allows computer programs to automatically improve through experience. Looking ahead, some of the world's most complex problems — such as future pandemics — could depend on ML as a solution.

With a curriculum taught by Cornell Tech's Visiting Lecturer Brian D'Alessandro, with content from Cornell University and Boston University faculty, and developed in concert with industry leaders, this "Machine Learning Foundations" summer course offers you the skills that will enable you to build ML solutions in real-world conditions through an ethical and inclusive lens. In this skills-based program, you will work with industry-relevant tools to analyze real-world data sets to identify patterns and relationships in data. By the end of this nine-week course, you will have hands-on experience solving real-world problems by working through the ML life cycle to build machine learning models.

Weekly synchronous lab sessions will give you the opportunity to explore these skills in a collaborative environment and gain hands-on experience in machine learning and data science. Ultimately, the foundational skills you acquire in this "Machine Learning Foundations" curriculum will prepare you to take on real-world industry challenges in Studio this fall.

During this part of the **Break Through Tech** machine learning program, you can expect to spend about eight to ten hours per week on asynchronous online content and about three hours per week on synchronous lab sessions with other students. For a detailed list of what is required to successfully complete this course, visit the "Syllabus" link under **Course Resources** in the left navigation menu.

Brian D'Alessandro

Head of Data Science, Social Impact

Instagram

Brian D'Alessandro is a practicing data science executive with 20 years of experience building machine learning and statistical models for industrial decision making. Mr. D'Alessandro currently leads data science for Instagram's Well Being and Integrity teams. Prior to Instagram, he held leadership roles at Capital One, Zocdoc, and Dstillery. Within these roles, Mr. D'Alessandro led the development and execution of AI and ML systems enhancing digital experiences serving healthcare and financial service needs. Mr. D'Alessandro has also served as an adjunct professor for NYU's Center for Data Science Master's of Data Science program. He developed and taught core ML classes for incoming Master's students and has helped over 1,000 students start their careers as data scientists and machine learning engineers.

Dr. Kilian Weinberger

Associate Professor

Computing and Information Science, Cornell University

Dr. Kilian Weinberger is an Associate Professor in the Department of Computer Science at Cornell University. He received his Ph.D. from the University of Pennsylvania in Machine Learning under the supervision of Lawrence Saul and his undergraduate degree in Mathematics and Computer Science from the University of Oxford. During his career, Dr. Weinberger has won several best paper awards at ICML (2004), CVPR (2004, 2017), AISTATS (2005), and KDD (2014, runner-up award). In 2011, he was awarded the Outstanding AAAI Senior Program Chair Award, and in 2012 he received an NSF CAREER award. Dr. Weinberger was elected co-program chair for ICML 2016 and for AAAI 2018. In 2016, he was the recipient of the Daniel M. Lazar '29 Excellence in Teaching Award. Dr. Weinberger's research focuses on machine learning and its applications; in particular, he focuses on learning under resource constraints, metric learning, machine-learned web-search ranking, computer vision, and deep learning. Before joining Cornell University, Dr. Weinberger was an Associate Professor at Washington University in St. Louis and had previously worked as a research scientist at Yahoo! Research.

Dr. Linda K. Nozick

Professor and Director of Civil and Environmental Engineering

College of Engineering, Cornell University

Dr. Linda K. Nozick is Professor and Director of Civil and Environmental Engineering at Cornell University. She is a past Director of the College Program in Systems Engineering, a program she co-founded. Dr. Nozick has been the recipient of several awards, including a CAREER award from the National Science Foundation and a Presidential Early Career Award for Scientists and Engineers from President Clinton for "the development of innovative solutions to problems associated with the transportation of hazardous waste." She has authored over 60 peer-reviewed publications, many focused on transportation, the movement of hazardous materials, and the modeling of critical infrastructure systems. Dr. Nozick has been an associate editor for Naval Research Logistics and a member of the editorial board of Transportation Research Part A. She has served on two National Academy Committees to advise the U.S. Department of Energy on renewal of their infrastructure. During the 1998-1999 academic year, she was a Visiting Associate Professor in the Operations Research Department at the Naval Postgraduate School in Monterey, California. Dr. Nozick holds a B.S. in Systems Analysis and Engineering from George Washington University and an M.S.E and Ph.D. in Systems Engineering from the University of Pennsylvania.

Dr. Chris Myers
Senior Research Associate and Adjunct Professor
Center for Advanced Computing, Cornell University

Dr. Chris Myers has been with the Center for Advanced Computing at Cornell University since 2017, having previously been a member of the research staff of the Bioinformatics Facility of the Institute of Biotechnology (2007-2017) and the Cornell Theory Center (1993-1997, 1998-2007). In addition, he is an Adjunct Professor in the Department of Physics at Cornell and a member of the graduate faculty in the fields of Physics, Computational Biology, Applied Mathematics, and Computational Science and Engineering. Dr. Myers works primarily in the field of computational biology, addressing problems in the systems biology of cellular regulation, signaling, metabolism, development, virulence and immunity, and in host-pathogen interactions and the spread of infectious diseases on populations, networks, and landscapes.

Dr. Mehrnoosh Sameki
Adjunct Professor, Boston University
Senior Technical Program Manager, Microsoft

Dr. Mehrnoosh Sameki is a senior technical program manager at Microsoft, responsible for leading the product efforts on machine learning interpretability and fairness within the Open Source and Azure Machine Learning platform. She co-founded Fairlearn and Responsible-AI-widgets and has been a contributor to the InterpretML offering. Dr. Sameki earned her Ph.D. degree in Computer Science at Boston University, where she currently serves as an Adjunct Assistant Professor offering courses in responsible AI. Previously, she was a data scientist in the retail space, incorporating data science and machine learning to enhance customers' personalized shopping experiences.

Watch: Course Introduction

In this course, you will practice the skills that unlock value in unstructured data sets and aid you in making recommendations about what models to use when solving machine learning problems. You will become familiar with factors to consider and questions to ask to make sure they're implemented in the most effective way, including how to train a more powerful model using advanced evaluation and hyperparameter tuning methods.

Video Transcript

Think about some of the technologies you use every day: the voice-automated assistant on your phone, the suggestions that appear on YouTube or Netflix, or even the chatbots on websites you frequently visit. These are all practical applications of machine learning, a branch of artificial intelligence that allows computer programs to automatically learn through experience.

Machine learning is a field that is grounded in decades of academic research. The people who are putting machine learning into practice and thus creating the chatbots, recommender systems, and voice assistants we often interact with are called machine learning engineers. Machine learning engineering as a discipline naturally pulls from academic research, but it is also a practical trade. Becoming a machine learning engineer takes time with focused study and experience.

My goal is to prepare you to master the core skills necessary to become a successful machine learning engineer. You will walk away from this course with hands-on experience in building and optimizing machine learning models from scratch. Our aim is for you to have enough knowledge and experience to be able to independently formulate and execute machine learning problems.

Finally, with your career in mind, we hope that this program will set you up to be successful in landing your first job or internship in the field as a machine learning engineer.

Back to Table of Contents

Read: Course Resources

The contributing faculty and experts who designed this course have identified a number of additional resources that may be useful if you wish to delve further into the topics covered throughout the summer. You may choose to listen/read/watch any of these alongside your course work now or in the future.

ML Resources

- **AI Nation**: An NPR podcast that reveals how machine learning and predictive analytics are affecting every major aspect of our modern lives.
- **Coded Bias Trailer**: A trailer for a documentary exploring the idea of technologies like AI being built on systemic racial and gender-based prejudices. The woman in the trailer is Joy Buolamwini, a computer scientist and digital activist. Buolamwini founded an organization, Algorithmic Justice League, to challenge bias in decision making software.
- **Kate Crawford's book**: Kate Crawford is a researcher at Microsoft research who recently wrote a book about bias in AI. This video is a segment from one of her many talks where she promotes the importance of this topic.
- **Machine Learning and AI Roles at Google**: You can find information on various positions at Google that are in the ML and AI field.
- **Machine Learning and AI at Google**: Information on Google's AI research, responsibilities, and tools.
- **Machine Learning for Media and Entertainment Industry**: An overview by Allie K Miller on how ML is applied to the media and entertainment industry.
- **Data Science from Scratch**: A textbook written by Joel Grus that covers the foundation of python basics for data science including the principles, libraries, frameworks, toolkits and more.
- **Linear Digressions:** A podcast that explores ML and DS through various applications by Ben Jaffe and Katie Malone.

Python Resources

The following are links to a variety of downloadable Python resources.

- **Core Python Data Science Ecosystem Components**: A tool listing core Python ecosystem components relevant to data science such as the Python Standard Library, IPython, Pandas, Numpy, and more.

- **Built-in Python Container Data Types**: A list of Python's built-in container data types such as lists and dictionaries.
- **IPython Tip Sheet**: Helpful tips for using IPython.

eCornell Keynotes on the Future of Tech

- **The Coming AI Revolution**: Keynote session hosted by Ray Jayawardhana, the Harold Tanner Dean of Arts & Sciences, as part of the Arts Unplugged series. An interactive discussion with leading experts on changes and considerations in how we can enact policy that supports democracy and an ethical society.
- **The New Jim Code**: Keynote session hosted by Ruha Benjamin, Professor of African American Studies from Princeton University, which discusses how technology, while appearing to be a neutral arbiter, deepens racial discrimination and ways in which technology encodes inequality and amplifies racial hierarchies.
- **Machine Learning**: Keynote session hosted by Kilian Weinberger, Associate Professor of Cornell Bowers College of Computing and Information Science, discussing the newest advances in machine learning and AI and how it improves our modern day sectors such as medical care, transportation, and communications.

Back to Table of Contents

Read: Course Data Sets

Throughout this program, you will be working with multiple data sets. You do not need to download the data sets now in order to complete each of the coding exercises, assignments, and labs. However, if you'd like to save them for future reference, you may choose to download them now.

- Becker, Barry & Kohavi, Ronny (1996). *Adult Data Set* (1994) [Data set]. UCI Machine Learning Repository. **https://archive.ics.uci.edu/ml/datasets/adult**
- Bogacz, Michal. (2021). Restaurant Business Rankings 2020 [Data set]. Kaggle. **https://www.kaggle.com/datasets/michau96/restaurant-business-rankings-2020**
- Bozsolik, Timo. (2020). *Superbowl History 1967 - 2020* [Data set]. Kaggle. **https://doi.org/10.34740/KAGGLE/DSV/926384**
- Duke University. *Telecom churn (cell2cell)* [Data set]. Kaggle. **https://www.kaggle.com/datasets/jpacse/datasets-for-churn-telecom/code**
- Garnepudi, Venkatesh. (2019). *Spam Mails Dataset.* [Data set]. Kaggle. **https://www.kaggle.com/datasets/venky73/spam-mails-dataset**
- Helliwell, J., Layard, R., & Sachs, J. (2018). *World Happiness Report 2018* [Data set]. New York: Sustainable Development Solutions Network. **https://worldhappiness.report/ed/2018/**
- *Inside Airbnb Data* [Data sets]. Inside Airbnb. **http://insideairbnb.com/get-the-data**
- John Blitzer, Mark Dredze, Fernando Pereira. Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. Association of Computational Linguistics (ACL), 2007. **https://www.cs.jhu.edu/~mdredze/datasets/sentiment/**

Back to Table of Contents

Unit 1: Machine Learning in a Nutshell Overview

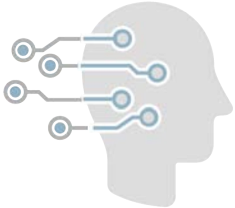
Video Transcript

The first set of lectures here are designed to build a high level overview of everything we'll be learning throughout the course. I will first cover some basics, such as: what are machine learning models?; what are their goals?; and why do we need them? I'll then provide a basic taxonomy of different styles of machine learning models and show how they fit with different problem types. The first three modules will be more conceptual in nature. I believe it is very important to understand the big picture before jumping into specific topics.

The second half will be split between presenting the end-to-end model development life cycle and setting up the tools to execute the life cycle. In later sessions, we'll deep dive into individual components of the development life cycle and provide opportunities to practice those components with actual coding. But if you're eager to get coding, I'll set you up for that in the last module here, where I'll introduce the Python tools most commonly used for preparing data and building machine learning models.

What you'll do:

- **Identify which machine learning problem types are appropriate for a given problem specification**
- **Articulate the complete ML development process end to end**
- **Understand the responsibility of ML engineers in developing fair models**
- **Become familiar with various tools and packages used in the industry**



Unit Description

Machine learning (ML) is the use and development of computer systems with the ability to learn and discover patterns in data. You even encounter some of these systems on a daily basis; for example, a computer program can determine whether an email is spam or not spam, and a computer program can find patterns among shoppers and recommend products tailored toward their needs and interests. Mastering the ability to analyze and visualize data in meaningful ways is a critical step in your study of machine learning.

In the first unit of this program, you will start by exploring the role that machine learning plays in the industry for decision making and thinking about your role as a machine learning engineer (MLE). The characteristics of a particular problem, the data you have to work with, and the questions you want to answer will dictate what type of ML approach, method, and algorithm needs to be used. Once you have gone through the basic role of machine learning and the process from start to finish, Mr. D'Alessandro introduces industry-relevant tools such as Jupyter Notebooks, NumPy, and Pandas.

Back to Table of Contents

Tool: Unit 1 Glossary

Though most of the new terms you will learn about throughout the unit are defined and described in context, there are a few vital terms that are likely new to you but undefined in the course videos. While you won't need to know these terms until later in the unit, it can be helpful to glance at them briefly now. Click on the link to the right to view and download the new terms for this unit.

Download the Tool

Use this **Unit 1 Glossary** as a reference as you work through the unit and encounter unfamiliar terms.

Back to Table of Contents

Discussion: Unit 1: Machine Learning in a Nutshell

Welcome to the discussion thread for Unit 1! This is a space where you can ask any questions you have about the foundational concepts of machine learning covered in this unit.

To get the most helpful responses from your peers and instructional teams, be specific about the concepts you're struggling with and share what you've understood so far.

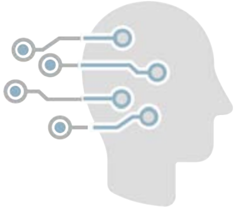
Guidelines for Posting in Discussion Threads:

1. **Stay On Topic:** Make sure your questions and contributions are relevant to the unit's content. This helps keep discussions focused and beneficial for everyone.
2. **Be Specific:** When you post a question, include specific details and context. Mention particular concepts, lectures, assignments, you're referring to. This makes it easier for others to provide accurate and helpful responses.
3. **Quote or Cite Materials:** If your question is about a specific part of the course materials, quote or cite the section to help others understand and reference the context of your query. You can even use a link to the specific course page!
4. **Constructive Language:** Use respectful and constructive language in all your interactions. Encourage and appreciate responses, and offer positive feedback when others help you.
5. **Use Clear and Concise Language:** Keep your questions and comments clear and to the point. Avoid overly complex language that might confuse your peers or instructional teams.
6. **Collaborative Learning:** Encourage discussion by not only asking questions but also responding to others. Sharing your insights and understanding can contribute significantly to collective learning.
7. **Confidentiality and Respect:** Do not share personal information or content that hasn’t been publicly discussed in the course materials. Always respect the privacy and intellectual rights of your peers and instructors.
8. **Search Before Posting:** Before posting a question, check if someone else has asked something similar and whether it has already been answered. This helps reduce duplicate questions and leverages existing discussions.

9. **Use Proper Formatting:** If you’re sharing code, equations, or specific formatting-dependent content, use the appropriate formatting tools provided by the discussion platform to enhance readability.
10. **Review and Edit Before Posting:** Before you submit your post, review it for clarity and errors. A well-written post is more likely to receive useful responses.
11. **Maintain Academic Integrity:** Do not post exact answers to assignment questions. Instead, focus on discussing concepts and problem-solving approaches. This helps preserve the integrity of the learning process and ensures everyone benefits fairly from the course materials.

Back to Table of Contents

Module Introduction: Use ML for Industrial Decision Making



Machine learning (ML) is one of the fastest-growing fields in tech right now, but what exactly is machine learning and how does it relate to other seemingly similar topics such as AI, deep learning, and data science? How does ML help in the real world and in business?

In this module, Mr. D'Alessandro discusses how machine learning is a new paradigm of programming for implementing AI, and how it is different compared to traditional input/output paradigms. He also introduces new terms and definitions relating to disciplines such as deep learning, data science, and statistics. At end of this module, you will begin this unit's assignment, where you will identify a real-life company and a product, feature, or application that is driven by machine learning.

Back to Table of Contents

Watch: ML: A New Paradigm of Programming

While machine learning is derived from many mathematical and statistical traditions, it can be helpful to think of it as a new paradigm of programming. Just like other forms of programming, machine learning aims to use software to solve problems. Watch as Mr. D'Alessandro further describes machine learning.

Video Transcript

While machine learning is derived and proven through various mathematical and statistical traditions, it is helpful to think of it also as a new paradigm for programming. Learning the underlying math and theory is very important to gain true mastery and be able to push the state of the art forward. However, to apply machine learning in practice, it is often sufficient to approach it as first, a new paradigm of programming, fitting it into a larger family of methods aimed at using software to solve problems.

This course will present many of the building blocks that will enable you to create machine learning models from scratch. We can't teach you everything you're going to need to know for machine learning and software engineering in general, but this course should set you up to both immediately apply machine learning to common problems and to develop the requisite knowledge and skills to enable further learning.

We can think of machine learning as just another type of math. In some cases, ML really is just an equation that is easy to express in pure mathematical terms. We will use the term 'model' a lot. A model is the core output of a machine learning process. It is technically usually a data object that represents patterns we've observed in data. It is usually used like a function, where some inputs are mapped to an output to achieve some decision-making purpose.

In practice, we'll shift from expressing machine learning, or ML, as pure math, and instead treat it as just another programming function. A function lets us encapsulate logic for repeated use in our applications. This is no different when that logic happens to be driven by machine learning. When writing functions in traditional programming, we express our logic directly using the rules and heuristics we consider appropriate for the problem. With machine learning, the logic is learned by finding patterns in data.

As machine learning engineers, we control how that logic is learned. But the logic is ultimately driven by an algorithm that optimizes some type of objectives that we specify. In the hello_ml function here, that logic would be encapsulated in some data object we call a 'model' here. We apply that logic using the predict method. We'll cover these ideas in more detail later. Right now, we're just focused on the high-level concept.

Ultimately, any machine learning logic you build will fit into a broader application designed to execute some sort of business purpose. When thinking about the larger system, we can be agnostic to whether the core application logic is generated by machine learning or business experts. Designing such systems is out of the scope of this class, but I find it useful to at least see at a very high level how machine learning fits into traditional software engineering.

Back to Table of Contents

Read: ML vs. Related Topics

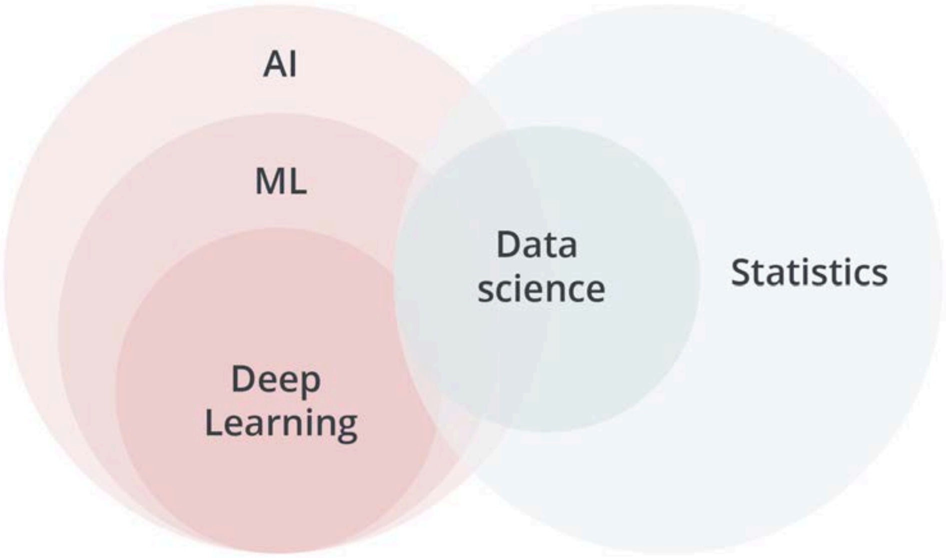
The field of artificial intelligence has experienced dramatic growth in recent decades. We hear terms such as "machine learning," "artificial intelligence," and "data science" all the time, but what does each one really mean, and how do they relate to one another? Let’s go over each of these terms and their related concepts.

☆ Key Points

Deep learning is a subset of machine learning, and machine learning is a subset of artificial intelligence.

Data science utilizes ML and statistics to uncover patterns and insights in data.

ML-based AI learns its behavior from data.



Artificial intelligence

Artificial intelligence, or AI for short, is a broad, all-encompassing term that captures the research and implementation of systems that are capable of performing tasks intelligently in a given environment. These tasks can be common tasks such as driving a car or can be as complicated as landing a rocket. An AI system is different from a traditional rules-based system in the following ways:

- It uses its environment to shape its behavior, whether through experience or on the fly.

- It can deal with the unknown and provide a generalized response for previously unseen situations

In a nutshell, AI is able to operate in an environment with previously unseen stimuli and make appropriate decisions without human intervention.

Machine learning

Machine learning (ML) is a subset, or implementation, of AI that deals with the research and implementation of systems that shape their behavior based on experience. Essentially, a machine learning model is a program that solves problems by "learning" from past data in order to make decisions. When implementing a machine learning model, instead of telling it what to do, we instruct it how to learn. A machine learning model needs historical data in order to perform well. It goes through a training phase, where it consumes data, often in high volume, in order to update its inner data structure — the brain, so to speak. It is worth emphasizing that not all AI systems employ ML.

Deep learning

Deep learning is a subset of machine learning that uses a special type of ML model, neural networks, as its underlying algorithm. Neural networks can solve very complex, real-world problems and are often used in the fields of image recognition, language processing, and speech recognition.

Statistics

Statistics is simply the science of collecting and analyzing numerical data, especially for the purpose of inferring properties of those in a representative sample. Statistics is commonly categorized into two buckets: descriptive statistics and inferential statistics. Descriptive statistics seeks to describe the various properties of a data set, such as the degree of variation and, relatedly, its distribution. Inferential statistics is concerned with making inferences on a larger group based on a subset of sampled data. Some machine learning models, such as linear and logistic regression, originated as statistical methods.

Data science

Data science is the discipline concerned with finding patterns and providing insights from data. Machine learning aims to automate these processes. For some data science tasks, ML is the ideal candidate, while others may be more suited to using statistics. The fact that machine learning is capable of "learning on its own" from an enormous amount of data means it can pick up subtle relationships within data that could otherwise be missed using a traditional statistics approach.

Read: What Is Machine Learning?

Traditional programming works well in areas where the transformation between input and output is clearly defined; for example: If a button is pressed (input), dispense soda (output), else do nothing. Yet when it comes to performing tasks involving uncertainty, such as making a prediction or recognizing patterns, traditional programs are very much limited to the experiences and biases of their developers.

This is where machine learning (ML) is needed. Simply put, ML is a paradigm of programming that enables computer programs to learn from data. In machine learning, developers do not explicitly tell a program what to do in a given situation; instead, they tell the program how to learn from historical data. The result is a predictive program, known as a machine learning model. It is usually used like a function, where some inputs are mapped to an output to achieve a decision-making purpose. Some ML approaches aim to automatically discover general patterns in data, whereas other approaches aim to learn relationships between specific data elements in order to make predictions when encountering new, never-before-seen data.

☆ Key Points

Machine learning is also referred to as ML.

In ML, an algorithm learns from training data and builds a model, or a prediction algorithm, that can be then used to make accurate predictions when encountering new data.

An ML model’s goal is to generalize well to new, previously unseen data.

Training data contains examples, also referred to as data points.

For example:

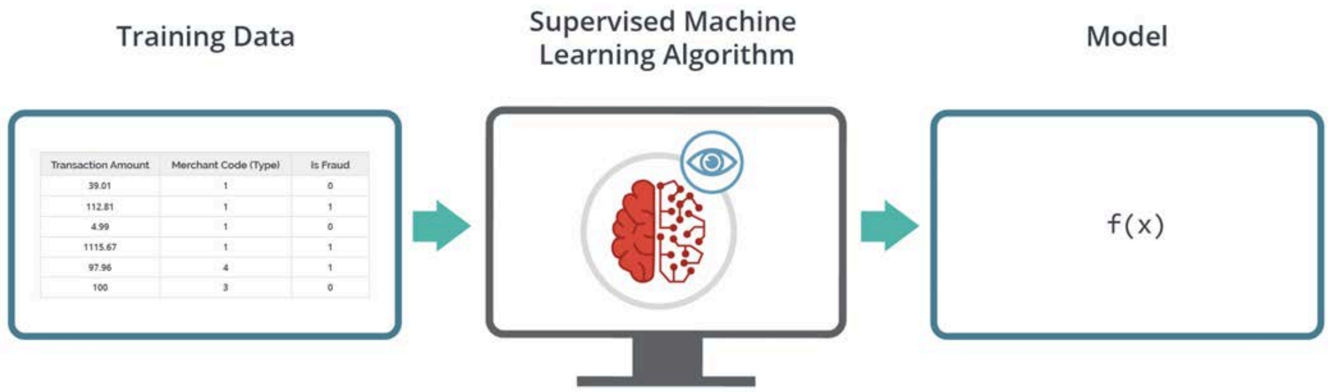
- A computer program can learn what constitutes spam or not spam from existing emails, then predict whether a new email is spam or not spam.
- A computer program can learn to identify a dog or a cat from existing images. It can then predict whether a new image is that of a dog or of a cat.
- A computer program can learn from historical housing data, then predict the price of a new house in a given area.
- A computer program can input consumer and product information, find patterns among the data, and group consumers together to recommend products tailored toward their needs and interests

Machine learning is not just one method or algorithm but a broad swath of different approaches, methods, algorithms, and techniques that address various kinds of problems. The first step in deciding how specifically to employ ML methods for data of interest is to identify the specific questions you want to answer and the type of data you have to work with.

You can solve most problems using one of two types of ML methods: supervised or unsupervised learning. Of the two, supervised learning is the most commonly used form of machine learning, and this program will focus on supervised learning.

▼ Supervised learning

You will delve into the specifics of supervised learning throughout the program, but it is worth discussing the unique features of this method. In supervised learning, a computer program learns from specially designed input data and improves itself until it can make accurate predictions on similar data in the future. This is known as training. During this training phase, a standard machine learning algorithm is run on input data, such as historical housing data, or images. This input data is also known as training data. The algorithm learns from that training data and produces a program, known as a machine learning model, that can be used to make future predictions on brand-new data. You can think of a supervised ML model as a function that takes in new data and outputs predictions. In this course, you may see the term “model” referred to as a “function.”



Transaction Amount	Merchant Code (Type)	Is Fraud
39.01	1	0
112.81	1	1
4.99	1	0
1115.67	1	1
97.96	4	1
100	3	0

This process is an iterative one. The produced ML model often goes through multiple iterations of training, evaluation, and optimization to improve until the optimal model is produced.

Thanks to the proliferation of data and computing power in today's world, the training phase can input millions of data points and improve the model's inner wiring — its brain, so to speak. Machine learning models can learn from hundreds, thousands, or millions of data points to inform their decision-making — a scale of data points that no human can possibly encounter. As a result, ML models are able to pick up subtle relations and patterns that are beyond the natural capabilities of the human mind.

As stated, the goal is to produce a model that has the ability to make accurate predictions on new, unseen data. A model is not perfect, however, and will not be able to guarantee correctness 100% of the time because, remember, the nature of the problem that it tries to tackle is inherently uncertain.

To illustrate this further, imagine we have a machine learning model that is designed to classify emails as spam or non-spam. During training, the model is exposed to millions of actual spam and non-spam emails, from which the model develops a good sense of what constitutes a spam email or a non-spam email. It can then predict whether a new email is spam or not spam. In reality, however, what is classified as a spam email to one person may actually be useful to another; therefore, it is not possible for a machine learning model to predict accurately 100% of the time. The best that this model can do is to **generalize** from past data in order to produce a best guess when facing new, previously unseen data; in this case, a new email. Developing a model that has the ability to generalize well to new data — i.e., make an accurate prediction when encountering new data) is the focus of this program.

What does the input data look like? The compilation of initial data is known as a data set.

▼ Data set

While the specifics of the data itself will vary and will be discussed throughout this program, you can picture the model's input data as organized in a similar manner in which data is organized in a database table. The table consists of columns and rows; this is known as a data matrix and in Python as a "dataframe." In this program, you will often see the terms "data set," "dataframe," and "data matrix" used interchangeably. Each row is called an "example" or a "data point." When we state that a model is trained on millions of actual spam and non-spam emails, we can instead say that a model is trained on millions of "examples" or "data points."

Email sender	Email subject	Recognized IP	Contains word “advertisement”	SPAM
Frank’s Restaurant and Bar	Eat at Frank's	No	Yes	Yes
Mom	Monday at 4	Yes	No	No
Trinket Land	10 percent off	Yes	Yes	Yes

Back to Table of Contents

Ask the Expert: Francesca Lazzeri on AI Applications in Real Life

Many artificial intelligence systems employ machine learning. In this video, Francesca Lazzeri gives some real-world examples of such AI systems and describes a few common algorithms that are used in their application in the real world.

Note: The job title listed below was held by our expert at the time of this interview.

Francesca Lazzeri
Curriculum Committee Industry Advisor, Microsoft

Dr. Francesca Lazzeri is an experienced scientist and machine learning practitioner with both academic and industry experience as an Adjunct Professor of AI and Machine Learning at Columbia University and Principal Cloud Advocate Manager at Microsoft. She also authored the book "Machine Learning for Time Series Forecasting with Python" (Wiley) and many other publications, including technology journals and conferences. At Microsoft she leads a large international team (across the U.S., Canada, U.K., and Russia) of engineers and cloud AI developer advocates, managing a large portfolio of customers in the research and academic sectors and building intelligent automated solutions on the cloud. Before joining Microsoft, Dr. Lazzeri was a research fellow at Harvard University in the Technology and Operations Management Unit. You can find her on Twitter at @frlazzeri.

What are some real-world examples of AI?

Video Transcript

So AI is really everywhere and also I would say machine learning because I like to always say that AI is here with us everywhere in our life thanks to machine learning algorithms. So we don't need to forget that the machine learning part of AI and intelligence applications in general.

Honestly, when they asked me this question of real-world examples of AI applications, I really like to measure like things that we use everyday, applications that we use every day, like for example, navigation applications. Every time that you jump in your car and you're driving to a specific destination, you're using an application that is using, AI is using a lot of the different machine learning algorithms to let you arrive to the destination where you have to arrive.

Another very common example of AI is Netflix. Every time with that you are watching Netflix and you see that they are suggesting you a new movie or new TV series, behind that there is a simple recommendation system that is most of the time is based on multiple recommender systems that we call it and these are just the algorithm that basically tries to understand what your profile is, what your preferences are based on history and also based on additional data, external data and they try to suggest you something to watch next. This is a classical example of AI application.

Another one that is very, very common is the chatbot. Every time that you'll probably tried to call a customer service or tried to interact with them and because you have specific questions and you are inputting the question, you are getting an answer and most of the time, that is a chatbot This is more like based on NLP type of algorithms and also text analytics algorithm.

I would say that another very interesting example of AI is for example, the algorithm that is also recommending you like a new item to buy and this is more like related to marketing use cases. So every time that you are on a website, you are shopping or you're just reading your news. You get suggestions about the different products and types of services to buy. Again, that is another example of AI. So really AI is at this point everywhere, and that's a consequence of machine-learning algorithms are a everywhere. They are learning from us and we are the first generators of the data that they use in order to learn more about the world and as a consequence to become more intelligent.

Back to Table of Contents

Watch: Generalization — A Fundamental Goal

While there is a great degree of variety among machine learning applications, there are also a few core principles that apply to all machine learning use cases. These principles can be used to build machine learning applications for new problems. One of the most important is generalization, which is a model's ability to make accurate predictions on new, previously unseen data. Mr. D'Alessandro walks through an example of this principle.

Video Transcript

Nearly all electronic forms of commerce and entertainment these days use some level of machine learning to operate. Machine learning is used in many industries. More traditional industries like health care and insurance have had to learn to adapt their operations to accommodate this new technology. Other industries like tech, e-commerce, and certain credit lenders have grown with machine learning central to their strategy. There is an enormous variety in the ways we can leverage machine learning models and also in the ways to develop those models.

Nonetheless, a few core principles will apply to all machine learning use cases. Learning these principles will set you up to build machine learning applications on new problems. Arguably, the most important principle to learn is the idea of generalization, which we can define as a model's ability to adapt to new, previously unseen data. In other words, we want the model to be able to make predictions on data and be accurate. Nearly everything you learn in a machine learning course is aimed at making you able to build models that generalize well. Let's walk through a quick example to illustrate this core principle.

Let's imagine we're using machine learning to detect if certain credit card transactions are fraud. First, let's assume we have historical data that represents past examples of this problem. This data here represents what transaction data might look like. Each column represents independent attributes of these transactions. First, we know whether historical transactions were fraud, and we also have information about the transactions, such as the location of the merchant versus the location of the account, the amount of the transaction, as well as the type of merchant. These, again, would be represented as columns in your data.

Now, your true goal here is to assess the riskiness of new transactions, for instance, a transaction like this. Again, on the back end, this transaction gets represented as data, and this data will be input into your fraud model. The data would look like the original data that was used to build the model, only this specific example is new — has never been seen before.

Since you've never seen this transaction before, there isn't technically an exact precedent to learn from. A well-built model gets us around this problem.

By observing historical examples that are at least similar to this new example, we can make much better predictions on the new example. We'll try to illustrate that here more abstractly with these two distributions. The blue shape represents the distribution of an outcome such as fraud or anything else for a full population. The orange shape represents the distribution of an outcome for a smaller subset where everyone is similar to each other with respect to some attributes of the observed data.

Rather than having to use just the population average, which would be done absent to machine learning model, machine learning helps us find the most relevant subset — basically similar users from before or similar transactions — and we make guesses based on that group. This is the idea of generalization and action. I know this is a bit abstract, but I also hope the core idea of generalization is clear.

You will eventually learn how to prepare data and design algorithms to achieve this idea of generalization. You'll also learn how to measure it and how to improve it when your first attempts just aren't quite right. Before we go, we'll leave you with a few key takeaways to remember. Machine learning is everywhere, but all applications of it leverage the same core principles. Generalization is the most important idea in machine learning, and nearly everything you learn in a machine learning course is centered around able to master the skill of generalization.

Back to Table of Contents

Quiz: Check Your Knowledge: ML or not ML?

In this quiz, you will be presented with various scenarios. You will determine if it is an example of machine learning.

You may take this quiz up to three times.

Please complete this activity in the course.

[Back to Table of Contents](#)

Watch: Practical Considerations of ML

Despite the wide utility and application of machine learning, it does have its limitations. In this video, Mr. D'Alessandro sets some ground rules to consider when using ML.

Video Transcript

Before we begin our journey of building machine learning models, I like to emphasize a few important points. Machine learning does not solve all problems. I realize this might sound obvious, but it needs to be said. It isn't uncommon to encounter colleagues in a professional setting who think that as long as your company is collecting data, machine learning is a viable answer to many problems. With this said, I want to introduce a few cautionary rules, as it is very important to consider important limitations up front.

Rule number one is let the application drive the solution, not the other way around. There is a popular saying that, "If all you have is a hammer, everything looks like a nail." That saying is important to remember as a machine learning engineer. Like a hammer, machine learning is just a tool. Only use machine learning if it is the right tool for the job. This course will teach you how to use the tools, but knowing when to use them will be developed and honed throughout your career.

And for the next rule, rule number two: to leverage machine learning, you need the right data. This is another rule that seems obvious at first, but it's worth saying out loud. Nearly every company these days is collecting massive amounts of data. But only people trained in machine learning are usually able to tell if the company has the right data. Assuming you're working on a problem that would be better with machine learning, as a machine learning engineer, you often need to assess whether you have the right data. We'll learn more about data preparation later, but for common machine learning applications, you'll need what we call a label where examples of what right answers look like.

You'll also need what we call features or data elements that you think would be predictive of the problem. For both, we'll need a sufficient number of examples. There is no single answer for how much data is enough, but we'll cover techniques in this course that will enable you to assess that for any given problem. The next rule is never underestimate the value of a good heuristic. I think this rule follows naturally from the first two, and is especially true when we find we lack adequate data to solve a problem. Breaking it down, a heuristic is just some well-reasoned or an end or tested business logic aimed at solving the same problem for which you might use machine learning.

As an example, if you're an online newspaper recommending articles, you can use machine learning, of course, but you can probably still get a lot of value out of just recommending the most-read articles or the most trending. In practice, you actually would often start with such a heuristic and as both your understanding of the problem and your data get better, you transition to a machine learning-based solution.

Finally, rule number four: with data comes great responsibility. As a machine learning engineer, you have quite a bit of power. You'll get access to a lot of data, and while you'll start to look at it as just numbers and strings, behind every row of data is often a real person. It's a privilege to have access to people's data, and it must be treated with care and respect. Most large companies have strong privacy policies and information security teams, so you won't have to navigate data privacy and security by yourself. But just remember that respecting individual privacy often starts with you.

Second, machine learning is used to make a lot of important decisions. Behind many predictions is a person looking for a mortgage, a job, or even a life partner. Machine learning lets us serve many people better at scale, but at the same time, it can also harm people at scale. Ethical AI is a growing field within machine learning, and we'll cover elements of it in this course. There is a lot of specialized knowledge required to evaluate models from a fairness perspective, and we'll introduce some of that. But the heart of ethical AI is still the fundamentals of machine learning, which is what we'll be covering throughout this course.

Back to Table of Contents

Watch: Case Study: Recommendation Systems

In this video, Mr. D'Alessandro provides a high level overview on recommendation systems. If you've ever tapped on suggested content in an app or watched a recommended video on a website, you've interacted with a recommendation system.

Video Transcript

Throughout this course, there will be a lot of hands-on practice, but another way to help the key points sink in is to see them in action. We'll introduce a few case studies from time to time to do this.

A great case study I like to consider is recommendation systems, which, at a high level, are machine learning systems designed to recommend items to you on various websites and apps. These are typically very complex and sophisticated, and there is usually no single way that these are built. This means that many of the core algorithms we learn can be used here.

First things first, recommendation systems are pretty ubiquitous. It is difficult to use the web or mobile apps these days without experiencing recommendations from a recommendation system. Let's start by discussing why this is so, or put another way, what value are these bringing to the companies that use them? Remember, we cover this point before anything else technical because we always start with the problem that we're trying to solve. Most of the companies that use recommender systems have the same user problem to solve. These are typically content providers or online merchants, and they have a lot of content or merchandise to provide.

Take YouTube, for instance. YouTube has around 30 billion videos and this is always going up. Many YouTube users want the same thing; that is, they want relevant videos. Defining relevance is one of the bigger challenges when building a recommender system, and we will defer this discussion for now. If we could plug ourselves into the team building the YouTube recommendation system, we might define a product goal like this. Ensure every user has access to relevant, informative, and/or entertaining content with as little friction as possible.

Now, notice that the goal here isn't to build a recommendation system using machine learning. Instead, the goal is framed around an outcome, which is access to relevant content to address a particular user problem, i.e., too much content. This architecture diagram will help us illustrate several of the key points we've made so far. We've already defined the problem we're trying to solve.

Looking at the diagram, we can abstract the solution a little bit. The recommendation engine is just one part of the bigger system. The recommendation engine acts as another function, taking in data and returning recommendations. This engine doesn't have to be driven by machine learning. Remember, a well-tuned heuristic may work just as well — so, just showing videos by the same video providers that you've already viewed. It is safe to assume that machine learning will help us better reach our problem goal than the heuristic over time.

We presented earlier that the primary goal of machine learning is generalization. In this context, we can think of generalization as recommending new, relevant videos based on videos that similar users have historically found to be relevant. This generalization goal can be precisely defined and measured, which will be necessary to actually build the system.

Finally, I want to emphasize the point about responsibility. Companies typically benefit from recommendation engines the most when they have a lot of scale, both in terms of content and users. While every company should take and be responsible with the use of their machine learning, those companies with the largest scale have the potential to have also the largest negative impact. There are different ways that recommendation systems can cause harm, both at individual and societal levels. Usually, this negative impact is unintentional. To reduce or avoid it altogether, awareness and intention are the first steps.

Ethical AI is its own specialized topic, and it is designed to address this specific problem. In this video, we covered recommendation systems at a pretty high level. Once we go deeper into specific machine learning procedures and algorithms, we can revisit this area to again see those new topics in action.

Back to Table of Contents

Assignment: Unit 1 Assignment - Part 1: Using ML for Industrial Decision Making

In this part of the assignment, you will identify a real-life company and a product, feature, or application that is driven by machine learning. You will use that real-life example to answer the questions in the assignment document linked below.

Completion of this assignment is a course requirement.

Instructions:

1. Download the **Unit 1 Assignment document**.
2. Complete Part One.
3. Save your work as one of these file types: .doc, .docx. No other file types will be accepted for submission.
4. You will not submit your unit assignment for grading and credit now. At the end of the unit, you will submit your completed assignment for facilitator review and grading.

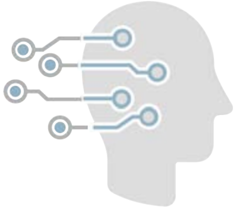
Back to Table of Contents

Module Wrap-up: Use ML for Industrial Decision Making

In this module, Mr. D'Alessandro explained how ML is a paradigm of programming that uses algorithms to create a model that learns to generalize from data. Simply put, generalization is the ability of a model to adapt to new, previously unseen data. Some machine learning implementations are a part of systems that you interact with every day, such as systems that detect fraudulent transactions when using a credit or debit card and systems that offer recommendations for new music or videos you may like. You completed the first part of this week's assignment and continued this discussion of ML being used in the real world where you chose an ML example, identified its objective and goal, and explained how ML can help achieve the goal.

Back to Table of Contents

Module Introduction: Recognize ML Problem Types



There are various approaches to machine learning that aim to solve different types of problems. In this module, Mr. D'Alessandro identifies the difference between the two most common families of machine learning methods: supervised learning and unsupervised learning, and some of the various approaches contained within each. Exploring the concepts that lie at the core of these two methods, such as features, labels, and the data matrix, will provide you with the foundation necessary for solving machine learning problems. By the end of this module, you will also have practiced identifying the difference between classification, regression, and clustering problems.

Back to Table of Contents

Watch: The ML Taxonomy

In this video, Mr. D'Alessandro introduces the two families of methods in machine learning known as supervised and unsupervised learning. There are a variety of machine learning algorithms used in each category. Mr. D'Alessandro gives an overview of each method and provides you with some examples.

Video Transcript

Machine learning is a general phrase that covers many different methods. We will cover some of the most used and fundamental algorithms that are used to build models for machine learning applications. We could think of each of these fundamental algorithms as building blocks for the more sophisticated and cutting edge methods. These algorithms fit into a few broad families, which we'll introduce here.

The two families of methods we're introducing are called 'supervised' and 'unsupervised learning.' We'll start with supervised learning, which we'll define as 'the process of inferring rules or functions from labeled data.' The important concept to anchor to here is the term 'label.' In other modules, we'll cover what it means to infer a rule or a function from the data, which is also the process of actually building a model.

Before defining exactly what a label is, let's introduce the basic terms and concepts that are typically part of a data matrix. We could think of a data matrix as a structured table consisting of rows and columns. Each row should represent an individual entity, which can be people, transactions, merchandise, or whatever is being modeled.

Let's use the credit card fraud model again. Individual card transactions would represent the entity of a single row. We often call these rows 'examples.' The columns represent individual attributes associated with each example. If these are credit card transactions, the first three columns might be 'transaction amount'; whether the card is physically present at the time of the transaction; and the category code of the merchant, like electronics store or grocery store. Most of these attributes will be used as an input to your supervised learning model. When used as inputs, these are frequently referred to as 'features,' and in the nomenclature, we often represent them as 'X' or a vector of individual Xs.

In supervised learning, there was one column or attribute that is special and is ultimately what differentiates supervised from unsupervised learning. This attribute is called a label, but can also be referred to as the 'target variable.' When written out, the nomenclature is usually to represent the 'label' as the letter 'y', much like the features with the letter 'x'. The label is the attribute we're trying to predict, and should be tied to the core goal of the problem. In our fraud data, it would be the column that tells us whether or not a specific transaction was fraudulent.

Features and labels are both columns or variables in your data. The main difference is the label is what you're trying to predict, and the features are the inputs for that prediction. Each example will be comprised of the same features and label, but usually with different values across the examples. We call it supervised learning because the label helps supervise the learning process. Specifically, during the model-building phase we call training, the label helps us understand the correctness of the model. Throughout the learning procedure, the correctness at each step is used to adapt the model to steadily improve itself.

Supervised learning is the core method we use for most prediction applications. In other words, we want a model to correctly estimate the label on a future example where the label itself isn't yet known. Again, with a music recommendation, I'd want to match you with a song that I predict you will like. Within supervised learning, there are two broad classes of methods which are defined by the type of label — whether it has a number or whether it is what we call a 'discrete class.' These are called 'Regression' and 'Classification,' and each is defined by the type of label.

In Regression, we are using any real valued number. For Classification, the label is a categorical value, which doesn't even have to be a number. Here are a few examples to illustrate the difference. For instance, what will the stock's price be tomorrow? Or how much will a new customer spend in the next 90 days? These are both regression problems, and in both cases, we're predicting a real valued number.

Here are some examples of classification problems. Is a particular email spam or not? This is a binary classification. Or what is the topic of a given article? This is what we would call a multiclass Classification because there are multiple, i.e., more than two classes. Also, what number is a particular digit in, maybe, some handwritten text? This, again, would be a multiclass classification example.

Again, you can see, with classification, we have two different types. We have the binary class or binary classification, and we have multiclass classification. The most common type is when the label is binary. You could think of these as problems where the question itself is a yes or no question. This is probably, again, the most common type of supervised learning that are going to be used in business applications. The other type, multiclass, is where there are three or more discrete label values. Notice how the labels can take on non-numeric values, for instance, when doing topic classifications, maybe for text or newspaper articles, the labels can represent different text categories, and there is no limit on how many different categories there might be.

Now let's switch to unsupervised learning. In a sense, unsupervised learning is defined as not being supervised learning. What I mean here is the process of finding patterns in data that do not have labels falls into unsupervised learning. One of the most common forms of unsupervised learning is finding clusters or segments of examples in your data. We'll provide more detailed examples later.

But one area where this technique is used commonly is in music streaming services. We can represent individual listeners by the songs and artists they listen to. A common goal is to find groups of listeners that have similar music tastes. We would find these groups using unsupervised learning techniques, and once we have the groups, we can use this to inform our music recommendations.

Back to Table of Contents

Read: Key Machine Learning Terminology

☆ Key Points

- A data matrix consists of a collection of examples that are either labeled or unlabeled.
- Machine learning programs are called models, and they are trained with examples before they can be used to make predictions.
- Supervised learning models are trained with labeled examples.
- Unsupervised learning models learn with unlabeled examples.
- >Regression predicts continuous numbers while classification predicts categorical values (classes).

As you progress through this program, you will frequently encounter a number of key terminologies relating to machine learning. It is important that you understand their meanings and make them second nature.

Below is a list of the most important terms used in machine learning. Take a moment to read through them before moving on. Refer back to this page and/or the glossary as you continue through the course.

Data set and data matrix

A data set is a collection of data. Data used in machine learning is stored in a data matrix. A data matrix is a structured table consisting of rows and columns.

Features

Features are input variables. They are the data contained in the columns of a data matrix. One feature value is contained in one column. Each individual feature is an attribute that describes the data.

For example, patient data has features such as *height*, *weight*, *age, body temperature, blood pressure,* and *heart rate*. Features can be numerical, textual, boolean, or another data type. A machine learning model can be trained on anywhere from a small number of features to millions of features. Features are grouped together as a vector and are often referred to as a feature vector, represented by X. Each individual feature in the vector is represented by the following notation: x1, x2, x3,...,xn. For example, the patient data contains a feature vector X composed of the features height (x1), weight (x2), age (x3), body temperature (x4), blood pressure (x5), and heart rate (x6). A feature is often referred to as a dimension. The patient data feature vector contains six features and can therefore be referred to as having six dimensions.

In our credit card transaction example, some common features may be:

- Type of transaction
- Time of transaction
- Location of the merchant
- Type of merchant
- Transaction amount
- Proximity of merchant location to cardholder's address

Labels

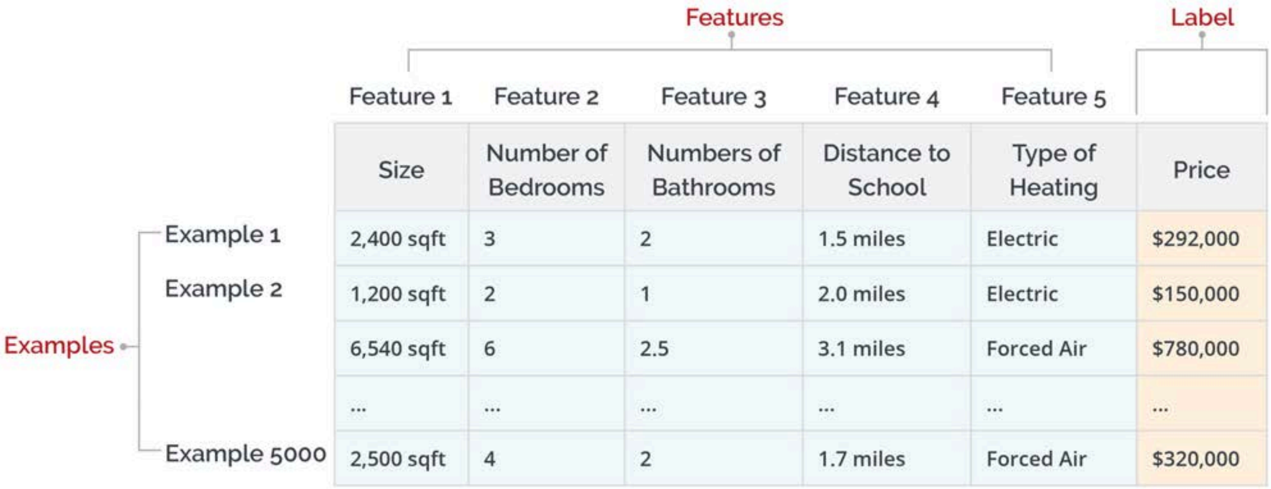
A label is the attribute that we want to predict. It is also called a target variable or an output variable and is represented by *y*. In the patient data example, the label is whether the patient is sick or healthy. In our credit card transaction example, the label is whether the transaction is fraudulent or not fraudulent.

Example — data point

An example is an instance of data; it is also called a data point. It corresponds to a row in the data matrix. This can be compared to a record in a database table. An example is either a labeled example or an unlabeled example.

Labeled example

A labeled example contains features and a label. Below is a data matrix containing 5,000 labeled examples. Note that with labeled examples, the label is associated with a set of features and makes up the last column in the data matrix.

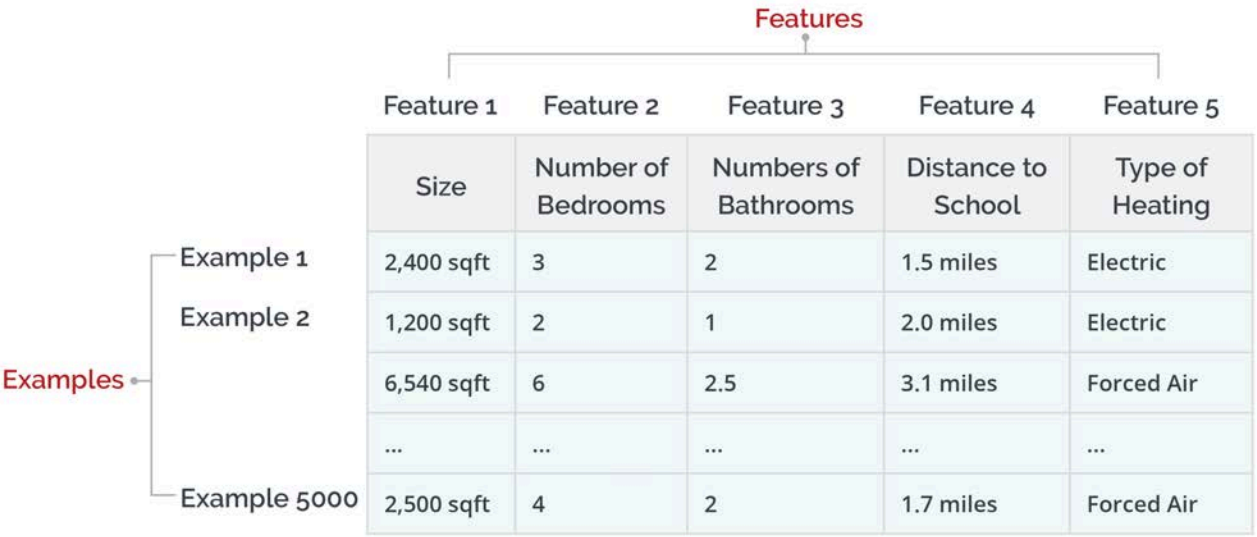

Features
Label
Examples

	Feature 1	Feature 2	Feature 3	Feature 4	Feature 5	
	Size	Number of Bedrooms	Numbers of Bathrooms	Distance to School	Type of Heating	Price
Example 1	2,400 sqft	3	2	1.5 miles	Electric	$292,000
Example 2	1,200 sqft	2	1	2.0 miles	Electric	$150,000
	6,540 sqft	6	2.5	3.1 miles	Forced Air	$780,000
	...	...	...	...	...	...
Example 5000	2,500 sqft	4	2	1.7 miles	Forced Air	$320,000

Figure 1. Data matrix containing 5,000 labeled examples.

Unlabeled example

An unlabeled example contains only features and no label. Below is the same housing data matrix containing only unlabeled examples.


Features
Examples

	Feature 1	Feature 2	Feature 3	Feature 4	Feature 5
	Size	Number of Bedrooms	Numbers of Bathrooms	Distance to School	Type of Heating
Example 1	2,400 sqft	3	2	1.5 miles	Electric
Example 2	1,200 sqft	2	1	2.0 miles	Electric
	6,540 sqft	6	2.5	3.1 miles	Forced Air
	...	...	...	...	...
Example 5000	2,500 sqft	4	2	1.7 miles	Forced Air

Figure 2. Data matrix containing only unlabeled examples.

Supervised learning

Supervised learning is the most common class of machine learning algorithms. Supervised learning attempts to discover the relationship between features and an associated label for the purpose of future prediction. Supervised learning creates a computer program, or model, that learns this relationship from past data to make predictions on similar yet never-before-seen data; it learns to predict a label.

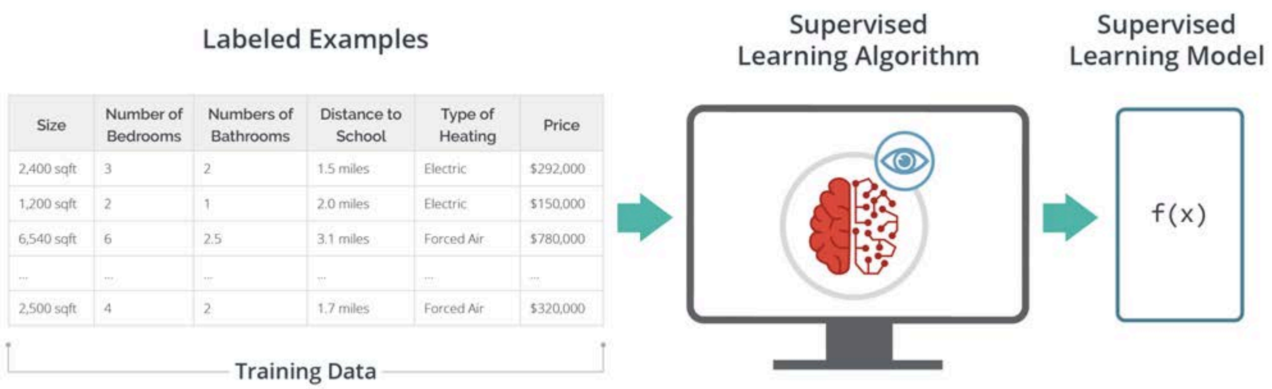
Consider a patient data model. Let’s say the features describe a patient in a hospital, e.g., height, weight, age, body temperature, various symptoms, and the label states whether the patient is sick or healthy. This data from past medical records can be used to create a model that is able to determine a future patient's diagnosis based on their symptoms. For an incoming patient, when you observe features, you can apply the model to predict whether this new patient is sick or healthy.

Because supervised learning models learn by using prior knowledge of an expected output (label) based on a given input (features), supervised learning algorithms use labeled examples as input to train models.

^ Training

Training supervised learning models means creating — also called "learning" — the model. During training, a standard supervised machine learning algorithm is run on training data comprised of labeled examples, enabling the algorithm to gradually learn the relationship between features and labels. As part of discovering this relationship, it may associate certain features more strongly with a specific label. Considering patient data, some features have more bearing on a patient’s overall health, such as body temperature, rather than height.

During this training process, a predictive model is built and iteratively improved by using the label as means to check its correctness. For example, if the model is inaccurately predicting the label (a patient’s health), the model can be tweaked to consider some features (a patient’s symptoms) as carrying more weight than others (a patient’s height or age). A model is often trained with a large amount of data — sometimes thousands or millions of labeled examples. Once training is completed, the result is an optimal model that can be used to make future predictions.



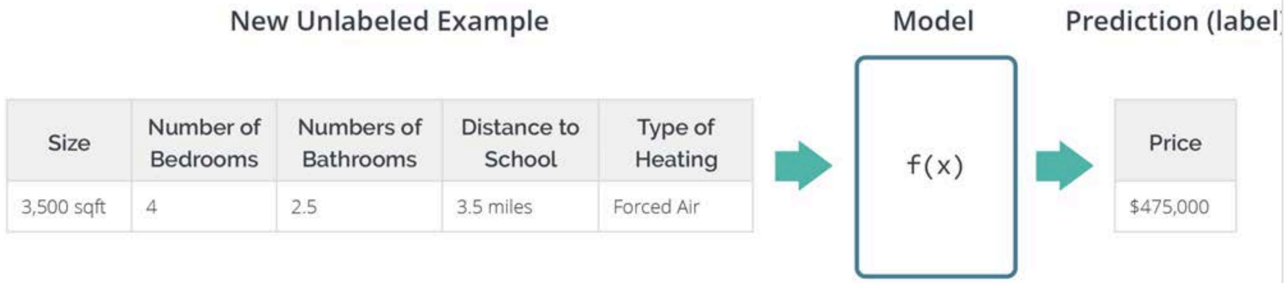
Size	Number of Bedrooms	Numbers of Bathrooms	Distance to School	Type of Heating	Price
2,400 sqft	3	2	1.5 miles	Electric	$292,000
1,200 sqft	2	1	2.0 miles	Electric	$150,000
6,540 sqft	6	2.5	3.1 miles	Forced Air	$780,000
...	...	...	...	...	...
2,500 sqft	4	2	1.7 miles	Forced Air	$320,000

Figure 3. Training of a supervised learning model.

Model

As you just discovered, a supervised learning model is the learned program that is able to make predictions on new, previously unseen data. A supervised learning model is said to have two phases. The first phase is the training phase, in which the model is built, and the second phase is the prediction phase — often called inference phase or testing phase — in which the model is used to make predictions.

After a model is trained with labeled examples, we can input a never-before-seen unlabeled example into the model and the model will output a label, or prediction. As you can see below, the unlabeled example consists of new housing data, and the output of the machine learning model is the price.



Size	Number of Bedrooms	Numbers of Bathrooms	Distance to School	Type of Heating
3,500 sqft	4	2.5	3.5 miles	Forced Air

Price
$475,000

Figure 4. Prediction phase with unlabeled example.

Generalization

Generalization is a model's ability to adapt to new, previously unseen data that is similar to the data that it has been trained on. Let's further explain this principle.

In supervised learning, we train a model to make predictions on a specific number of labeled examples. During the training phase, we tweak the model until it makes as few mistakes as possible. While a model may do well at making predictions on the training data, the ultimate goal is to create a model that can do well on future data. For example, if we make an email spam detector model, our goal is not only to accurately classify the emails that the model was trained on but to properly classify tomorrow's email. In other words, our goal is to ensure that our model generalizes to new examples.

Supervised learning algorithms

Supervised learning algorithms fall into two main categories: regression and classification.

Classification

The labels are discrete, or categorical values. Classification models predict these categorical values (classes) or the class probability membership (80% chance a patient is healthy).

For example, classification models can be used to make predictions such as:

- Whether an email is spam or not spam
- Whether a patient is healthy or not healthy
- Whether a credit card transaction is fraudulent or not fraudulent
- Whether an image contains an elephant, a dinosaur, or a human

Note that when dealing with classification problems, a label is commonly referred to as a *class label,* since the goal is to "classify" data. For example, an email can belong to the class "spam" or the class "not spam."

Regression

The labels are continuous, or any real valued number.

For example, regression models can be used to make predictions such as:

- The housing price of a new home in Manhattan, NY
- How much a new customer will spend in their first 12 months as a customer
- The minimum and maximum temperature on Friday

Unsupervised learning

Unsupervised learning is about discovering patterns in data without the use of training data containing labeled examples. It is called "unsupervised" learning because it is not provided the answer (the label); an unsupervised learning algorithm discovers patterns on its own. Unsupervised learning is sometimes the first step in a machine learning pipeline to identify structures to guide further analyses. It can then allow for analyses using supervised learning algorithms.

Unsupervised learning uses unlabeled examples, meaning only feature values. The algorithm discovers patterns in data typically by finding examples that are similar to each other and mapping them to different segments, or clusters. For example, when provided images of different animals, the algorithm might learn to cluster together animals of the same genus based on having common traits. The algorithm determines from pictures of humans that a human has two legs, two arms, walks upright, etc., and can then categorize other images of humans as such.

Clustering

Clustering is an unsupervised learning technique that groups subsets of data — “clusters” — that are collectively similar to one another based on the similarity of their feature values.

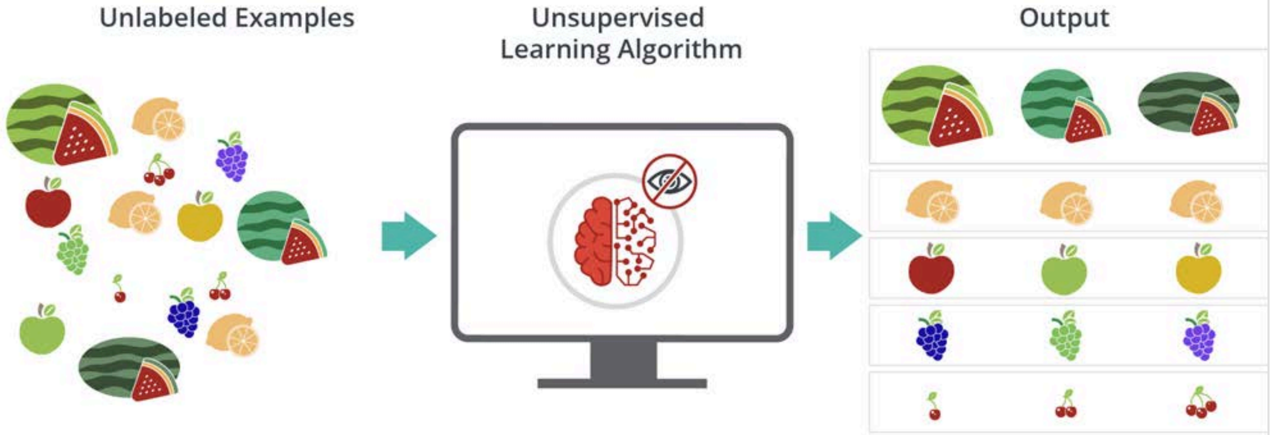


Figure 5. Clustering

In the example above, the algorithm would learn from the images that fruit comes in different sizes, colors, and shapes. After learning about their characteristics, it can then categorize them by types of fruit based off of color, shape, and size.

As stated, unsupervised learning is sometimes the first step in a machine learning pipeline. Supervised learning algorithms can then be used for further analysis. For example, even though the unsupervised learning algorithm above can identify different groups of fruit with similar characteristics, it does not "understand" that a fruit is watermelon or an apple or a cherry and therefore cannot label a cluster accordingly. Class labels can be added to the clusters for use with supervised learning algorithms, e.g., cluster 1 belongs to class "watermelon."

Back to Table of Contents

Watch: Supervised Learning Example

Supervised learning is the most commonly used form of machine learning, and will be the focus of this course. Supervised learning algorithms are provided features as well as a set of known labeled data, and build models that can be used to make predictions about new data. Supervised learning algorithms fall into two main categories:

- **Classification**: The labels are discrete, or categorical, values. Classification problems fall into one of two categories: binary classification and multiclass classification.
- **Regression**: The labels are real, or continuous numbers.

In this video, Mr. D'Alessandro demonstrates with a real business problem and explains how it can be solved using binary classification.

Video Transcript

In this video, we'll provide a visual example of a supervised learning problem in action. We'll leverage a real business problem, but we'll simplify it to make our points. The goal here is to establish a strong, intuitive understanding of supervised learning before we get into the technical details. I like to always start with the business problem.

Let's assume we're a new credit lending startup. We know that credit lending is a risky business, so we'd like to build a credit scoring system to make better lending decisions for people applying for credit. We know that machine learning is appropriate for this problem, so the first thing we need to do is translate the business problem into a machine learning problem. Doing this, we want to build a default prediction model where default is defined as the customer missing three consecutive payments in the first six months.

Notice a few things here. We were very precise in how we define default. Your problem statement ultimately needs to be translated into code, so you should express it as precisely as possible. The definition of default can be translated into a true-false question. So with this label, we have a binary classification problem. Of course, to build our model, we're going to need data. It is easy to assume that companies are awash in data, but it often happens that when you're building a product for the first time, you actually have no data to work with. This is called the cold start problem.

To deal with this issue, we start with an experiment where we give credit out to everyone who applies. This is clearly risky, but the extra risk can be worth it for what we learn in the long run. When people apply, we collect their age and income as our main predictors. We then observe them for the first six months to determine if they default or not. In the end, we have data that is suitable for supervised learning and specifically classification. When we plot the data, we can observe the relationship between age, income, and defaults.

Notice that we plotted the defaulting customers as red versus black. With machine learning, our goal is to generalize a pattern from observed examples from the data. In this case, that means looking beyond the individual data points and trying to find regions of the graph that have a higher density of defaulters. As an exercise, try to mentally draw a single line that best separates the red from the black points here. You likely won't be able to get a perfect separation, but try it anyways. In an age before machine learning, lenders used to draw these lines using heuristics.

As an example, they might have had a rule to only extend credit to people whose income was greater than $50 thousand. From a business perspective, this likely wasn't terrible. The rule is intuitive, and we can see from the picture that people with higher income are less likely to default, but there still are multiple problems here. This rule leaves out a lot of value on the table. There are people with lower incomes that still demonstrate good credit behavior.

Additionally, and historically, heuristics were often justified using faulty and discriminatory logic. This was so rampant that the US had to pass laws to prevent discrimination in credit lending based off of someone's protected class. Machine learning doesn't automatically solve these problems, but it does give us a path to greatly improve upon them. Here is the optimal line using a certain type of machine learning. This line doesn't perfectly separate the defaulters from the non-defaulters, but it does create region that ultimately reduce our error in predicting default incorrectly.

Revisiting our exercise from just before, does this line look like the line that you had visualized? I'll bet it was close. But now the mathematics behind machine learning enables to get a level of precision that our brains can't do alone. We do this by minimizing prediction errors through a process called model training. This visualization was easy enough to do with two variables or features, but think about what that would look like with a hundred. This is where machine learning becomes incredibly powerful for us.

Once we have our model, we can ignore the original training data. This leaves us with something like this. The single line represents our best generalization of the data. Now we can use this to solve our business problem. When new customers apply for credit, we'll again take their age and income, and use our model to find where they fall with respect to this line. What's cool is the two markers here represent the same risk to the company. Without machine learning, we might assume that the person with the higher income was a safer bet, but our data and model show us otherwise.

Also, when using this model in practice, we don't need to make simple binary decisions on whether to lend or not. Although the line separates the customers into high and low risk, there's actually a continuum between them. We can use the distance from the line to set appropriate credit limits. After all, there were still a lot of non-defaulters in the higher risk region, and one could argue that it's unfair to deny those people credit because people with similar profiles were more likely to default in the past.

Back to Table of Contents

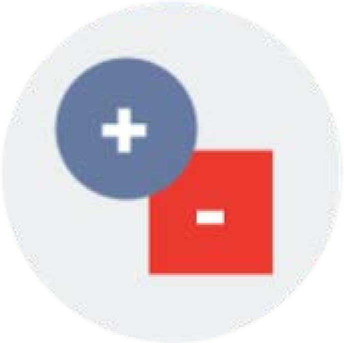
Read: The Labels of a Supervised Learning Problem

Recall that supervised learning is a class of machine learning algorithms concerned with discovering the relationship between features and an associated label for the purpose of predicting a label for a new, previously unseen data point or example.

The labels come in different forms depending on the type of ML model used and what we are trying to predict. Below are three of the most common labels used in practice today.

Binary

There are only two possible label values.



When the label we are trying to predict belongs to only two possible distinct values, we refer to this as a binary classification problem. Some examples of a binary classification problem and the corresponding labels are as follows:

- Is a given image that of a cat? True or false.
- Should we bring an umbrella today? Yes or no.
- What should we do with this loan application? Reject or approve.
- Is the email spam? Yes or no.

In the case of a binary classification problem, we can map each class to “+” or “-.” For example, with a spam email classification, an email is either spam or not spam. Spam could be mapped to "+" and email not considered spam to "-." In this program, you will often see two ways to denote binary labels: (+) and (-) notation or 1 and 0 notation (note that outside of this course these are not the one ways to denote binary labels).

Multiclass

There are multiple distinct label values.

When the label we are trying to predict belongs to multiple distinct values, we refer to this as a multiclass classification problem. Some examples of a multiclass classification problem and the corresponding labels are as follows:

- What animal is in a given image? Cat, dog, rabbit, or wookiee.
- What is the weather forecast for tomorrow? Sunny, cloudy, rain, or thunderstorm.

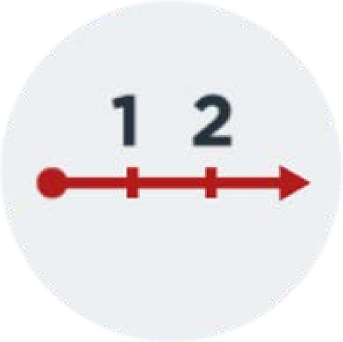
- What creditworthiness should we assign to a loan applicant? Poor, fair, or excellent.

Note that with multiclass classification, each potential value of the label would be distinguished as a separate class. For example, note the classes of the animal recognition problem:

- class1=”cat”
- class2=”dog”
- class3=”rabbit”
- class4=”wookiee”

Regression

There are infinite possible label values.



When the label we are trying to predict belongs to a real number, we refer to this as a regression problem. In regression problems, there are infinite possible label values. For example, if you want to predict for how much a particular house will sell, the label, i.e., the sale price, could be any non-negative value.

Some examples of a regression problem and the corresponding labels are as follows:

- What is the height of the animal in a given image?
- What is the temperature forecast for tomorrow?
- What is the credit score of a person of a certain financial profile?

Regression vs. classification

Some problems could be cast as either regression or multiclass; however, there is usually a most natural choice.

Assume, for example, that you are trying to predict the height (label) of a person based on other data (features), such as their gender, weight, or age. If you cast this as a multiclass classification problem, you might define each class as a rounded incremental value of 1cm. If we try to classify someone who is actually 183cm tall and return a label of 182cm, we are just as wrong as if we had predicted 140cm. Most people would consider a predicted height of 182cm to be much closer to the truth (183cm) than 140cm, but since each height is its own class and there is typically no assumption that one class is more similar to another in classification problems, we’ve simply failed to correctly predict the height. For this example, we should instead choose regression. In regression, we typically assume that because the label can be any real value, you will almost never hit it exactly but you can hope to get very close.

An example problem where multiclass classification would be the best choice is facial recognition. Bill Gates, Steve Jobs, and Linus Torvalds would each be their own class (i.e., class 1=''Bill Gates," class 2="Steve Jobs," class 3="Linus Torvalds"). We see here that each class is separate and distinct; selecting the wrong class is simply an incorrect label. There is no notion of similarity between classes since you are either right or wrong. For image classification problems, multiclass classification is the appropriate choice.

Back to Table of Contents

Watch: Unsupervised Learning Example

Unsupervised learning algorithms try to find patterns in data without labels associated with them. One common unsupervised learning algorithm is called clustering. Building on the credit lending problem, Mr. D'Alessandro will demonstrate how clustering could be used to build different lending products that are tailored to the specific needs of different customers.

Video Transcript

Let's go through an example of an unsupervised learning problem in action. We'll leverage again a real business problem, but simplify it to make our points. We'll use the same credit lending problem that was discussed in the supervised learning example. Like we did in that video, we'll start with the business problem.

Let's assume we have our credit model and are doing well using it to assess lending risk of our single product. But as our company evolves, we realize we're not serving the needs of all of our potential customers with just one product. We want to introduce a new business problem, which is, can we build different lending products that are tailored to the specific needs of our different customers and target them at the time of application?

When we translate this to an ML problem, we can say something like, 'Can we identify clusters of similar customers based on their observed attributes at application time?' Let's notice a few things. There is no mention of a specific outcome like default, for instance, that we're trying to predict. This is a good indicator that we're not using supervised learning. The keywords 'cluster' and 'similar' are also good clues that this might require unsupervised learning techniques. We were very specific to point out using attributes at application time. Once applicants become customers, we'll have richer data about them.

But if our goal is to match an applicant to the right product at time of application, we'll need to limit the data we use to what is available at application time. Like most machine learning projects, we might start with some basic visualization. Here, we're showing a simple scatterplot between age and income. What we want to do here is segment these points into discrete clusters, where members of each cluster should be similar to each other, and not similar to members outside of the cluster.

In the supervised learning example, we mentally drew a line to separate the defaulters from the non-defaulters. In this example, we'll do the same exercise, but now we want to draw two lines to split the data into four segments. Where would you draw such lines? Using one of our unsupervised learning clustering algorithms, we might end up clustering our customers into four groups like this. Now it is up to us to add meaning and utility to these clusters. We usually implement supervised learning models to make predictions, and we evaluate the accuracy of those predictions to understand the utility of the model. With unsupervised learning, we often need to add some subjective interpretation to derive the full benefit.

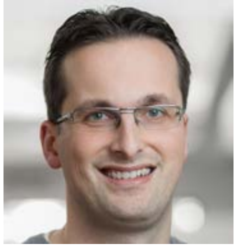
Here's an example of such a subjective interpretation. We can use the data we used in the initial segmentation to build stylized profiles of each segment. In many companies, this process might be complemented by deeper qualitative user research where individuals in each segment are interviewed or surveyed to build a strong human connection to them. We give each segment here a simple profile description. These descriptions might help us humanize the segments, and should inform our product development process. With a segmentation like this, we can then design a product to each specific need.

In an ideal world, everyone should have access to credit. But this needs to be designed in a way that doesn't put your company or the customer at risk of going bankrupt. Unsupervised learning is a common data-driven method supporting this type of business problem.

Back to Table of Contents

Ask the Expert: Laurens van der Maaten on ML

Note: The job title listed below was held by our expert at the time of this interview.



Laurens van der Maaten
Research Scientist, Facebook AI Research

Laurens van der Maaten is a research scientist at Facebook AI Research in New York, working on machine learning and computer vision. He previously worked as an assistant professor at Delft University of Technology and as a post-doctoral researcher at UC San Diego. Dr. van der Maaten received his Ph.D. from Tilburg University in 2009.

Dr. van der Maaten is interested in a variety of topics in machine learning and computer vision. Currently, he works on embedding models, large-scale weakly supervised learning, visual reasoning, and cost-sensitive learning.

How is machine learning helping us solve problems?

Video Transcript

So I think what's really cool about machine learning right now is that a bunch of things have come together that have allowed us to solve some really important problems like image classification, like speech recognition, and machine translation. And basically the things that have come together are sort of a set of algorithms, but then also data, right? So the availability of sort of large data sets that we can use for training of these models for these kinds of problems, and computation.

So what's been really important in recent years is the development of GPUs for the use of training machine learning models. And so I think that's what's really exciting; we've really sort of seen this confluence of these three different vectors that have allowed for breakthroughs in machine learning on a particular set of problems.

Back to Table of Contents

Quiz: Check Your Knowledge: Identifying ML Problem Types

In this quiz, you will be presented with various scenarios. You will determine which problem type it belongs to: Classification, Regression, or Clustering.

You may take this quiz up to three times.

Please complete this activity in the course.

Back to Table of Contents

Assignment: Unit 1 Assignment - Part 2: Recognizing ML Problem Types

In this part of the assignment, you will take your example from the previous part and further analyze its problem type: is it a classification or regression problem?

Completion of this assignment is a course requirement.

Instructions:

1. Open your saved unit assignment document, or download the **Unit 1 Assignment document**, if you have not already done so.
2. Complete Part Two.
3. Save your work as one of these file types: .doc, .docx. No other file types will be accepted for submission.
4. You will not submit your assignment for grading and credit now. At the end of the course, you will submit your completed assignment for facilitator review and grading.

Back to Table of Contents

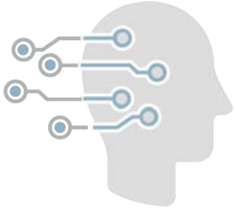
Module Wrap-up: Recognize ML Problem Types

In this module Mr. D'Alessandro introduced basic terminology frequently used in ML such as models, labels, and features. Mr. D'Alessandro also introduced the two most common problem types of supervised learning and unsupervised learning. Supervised learning models are trained on labeled data in order for them to make predictions on previously unseen data. Unsupervised learning, on the other hand, is trained on unlabeled data in order to identify patterns within the data itself.

By this point in the program, you have built up one of the most important skills in machine learning: knowing whether a real-world problem should be solved using supervised or unsupervised learning methods. You will continue to hone this skill with practice and experience. Continue to refer back to the foundational content in this module as you progress through the course.

Back to Table of Contents

Module Introduction: The ML Life Cycle



A machine learning engineer role can lead to a fast-growing career that is not only rewarding but impactful to society. In this module, Mr. D'Alessandro introduces Cross-Industry Standard Process for Data Mining (CRISP-DM), an industry-standard workflow for carrying out machine learning projects. You will practice identifying events that happen throughout the ML life cycle. You will also be introduced to MLE responsibilities, things to consider as an MLE in regards to fairness, and things to look out for in your data. You will then dive into the details of problem formulation in machine learning.

Back to Table of Contents

Watch: The ML Process: End to End

In this video, Mr. D'Alessandro describes the machine learning life cycle from analyzing your business problem, to preparing your data, to building your model and deploying it for public use. This program will walk you through the different steps in the life cycle to prepare you for a future as a Machine Learning Engineer or Data Scientist. The ML process can be represented by a diagram called CRISP-DM, or cross industry standard process for data mining. Mr. D'Alessandro walks through each of the six steps in this process.

Video Transcript

The core of machine learning is a set of computational algorithms driven by a mixture of mathematical techniques from calculus, linear algebra, and information theory. Although fundamentally driven by mathematics, most of it has been abstracted away by well-developed programming packages such as Python scikit-learn.

From a practitioner's perspective, the process behind building machine learning models is how we'll spend most of our time. We will cover the mechanics of fundamental algorithms, but I like starting with the overall machine learning process. Knowing the big picture helps us put the algorithms into the right context, and it doesn't hurt to get a little repetition around the process we'll often be repeating.

The process used for building machine learning models is very well established. As a matter of fact, one of the most popular ways to represent this process is a diagram that is at least two decades old. This is what is called the CRISP-DM, or Cross Industry Standard Process for Data Mining. No one really uses the phrase 'data mining' much anymore as data science and machine learning engineering are the de facto titles these days. Nonetheless, the process has held up very well, and we'll use it as our standard for machine learning model development.

Let's walk through each of the six steps at a high level. Before doing that, notice the complex arrow patterns in this chart and the fact that the the entire thing is represented as a circle. The outer circle represents the iterative nature of model development. It is fair to say that the work is seldom ever finished. Once we deploy a model, it is already time to start thinking about the next version of it. The inner arrows imply that we may not proceed in a purely linear fashion. For instance, we may revisit the first step after looking at our data or do more data prep if we find our modeling isn't performing the way we expected it to.

The first place to start is the business understanding. Although it is labeled as business understanding, this means this process can apply for all non-profit and for profit settings. What this emphasizes is we should really start with understanding the problem, both from a business or domain perspective and from a technical one. When building models in practice, it is common to start with a written project brief where you document your motivation, key constraints, planned approach, and timeline. This helps you get alignment with colleagues and peers to make sure you're heading in the right direction before writing any code.

Next, we'll bundle the data understanding and data preparation steps. Although presented separately, we can think of these as one continuous process. This can actually be one of the longer phases in the whole process. If you recall, one of the fundamental rules to do machine learning, you need the right data. This is the stage where you decide if that is true or not, and where you put in the hard work to ready yourself for the model building phases. While there aren't fancy algorithms to study here, remember the old saying: 'Garbage in, garbage out.'

Next, I'm going to bundle again the models building step as well as the evaluation. While evaluation requires its own set of techniques and methods that we'll learn later, it is usually done right after the modeling step. When we learn things like model selection, you'll see how intertwined these two actually are. While these are critical steps that actually define machine learning, you'll find that ironically, these are where you'll spend the least amount of time. It can take weeks to months to get to this stage, and then just a few hours to actually build or train a machine learning model. This is partly due to the fact that the methods used here are fairly independent of the domain or data, so it is relatively easier to automate this through good software.

Last in the development cycle is the deployment phase. Models don't provide any value if they're not used. This step is not to be neglected in practice. The elements of model deployment are more aligned with general software development, so are beyond the scope of this course. But this is the one area that typically differentiates machine learning engineers from data scientists. Although the specifics differ for each company, it is typical to find that machine learning engineers are responsible for end to end model development as well as deployment, whereas a lot of data scientists leave deployment to the engineers. Throughout this course, we'll be spending our time unpacking each of these steps in greater depth.

One philosophy I like to take, though, is to teach the art as much as the science. Most of the methods that will be used in the modeling and evaluation phases are well documented and pretty standard in conventional textbooks. The prep work that leads to these steps is often left for folks to learn on the job. I expect this to still be true for everyone going through this program, but nonetheless, I'll go into some depth in these sections to help transfer the knowledge I've gained practicing these for nearly two decades in the field.

Back to Table of Contents

Quiz: Check Your Knowledge: Identifying The Process

In this quiz, you will determine which part of the process the scenario is describing.

You may take this quiz up to three times.

Please complete this activity in the course.

Back to Table of Contents

Watch: Your Role as an MLE

Let's take a moment to examine two professional roles related to machine learning: the data scientist and the machine learning engineer. Mr. D'Alessandro has experience with both of these roles and offers insight into what to expect in an interview, the job itself, and also the risks involved in ML professions.

Video Transcript

I want to take a little time to discuss two professional roles related to machine learning. These are the data scientist and the machine learning engineer. I worked as a data science leader in many places and have managed both of these roles. The specifics of the two roles really do vary with each company, so it is always important to thoroughly review job descriptions to get the exact expectations or requirements for either role.

One generalization that I think holds true across most companies is that a machine learning engineer is an engineer first. I think this is a helpful rule of thumb because it provides clear guidance on how to prepare. A computer science degree isn't necessary, but knowing computer science usually is. Most interviews will expect you to know offhand fundamental algorithms, data structures, and complexity analysis.

It will also be important to know and understand software development and deployment processes. These are skills you'll develop better on the job, so an internship will be quite valuable in getting that first role post university. Last, of course, is the machine learning knowledge. Just as with computer science, there's a lot of knowledge to learn and hold for the machine learning portion of a job interview. Be prepared for a mix of both theory and practice questions. Although we can build models by treating the underlying algorithms as black boxes, it is always good to take the time to understand how they actually work.

At a high level, there are five themes to expect in a machine learning-focused interview. These represent a mixture of pure knowledge, both of processes and theory, and of what I call subjective evaluation. Being a machine learning engineer involves being able to make subjective tradeoffs, so this is an area that will likely be tested in interviews. This course was primarily designed to prepare you for the first four points here.

As a machine learning engineer, you can expect to be part of the full development lifecycle. One reason the machine learning engineer role exists and is separate from data scientist is the lack of consistency amongst data scientists in terms of being able to be full stack engineers. As a machine learning engineer, your primary responsibility will be to make sure models make it into product systems and don't break. In some places, you'll build the models in a bespoke way and manually push the results into the system. In more tech-forward companies, you'll actually be building systems that automate the full development cycle. In this type of environment, the machines are actually learning by themselves.

Machine learning can be highly valuable, but does come with risks. Learning how to manage these risks could fill an entire course. But at least we can start by raising awareness now. A lot of what you learn, even in a course that covers fundamentals, will help you in this area. I like to think of risk as falling into two main categories. These are system and ethical risks. System risks applied to all software systems. But I would argue that machine learning systems have a lot more additional complexity and more complexity means more failure points, so it is up to the machine learning engineer to implement the appropriate tests and monitors to be able to detect and mitigate such problems.

The other side is ethical risk. As the machine learning field matures in industries, we are learning more and more about its unintended effects. Machine learning models are typically evaluated on whole populations, which means we're not always aware of how they're performing on individual segments of society. It has been shown that models that perform well on average can often hurt non-majority and historically marginalized groups.

Both of these risks stem from one fundamental idea — companies are increasingly deferring decision making to automated systems. While such a design can improve the accuracy, consistency, and scalability of such decision making, if and when the decisions are being applied poorly, the scale of the error is going to be that much larger. Machine learning engineer is in the best position to anticipate and proactively manage these risks.

Back to Table of Contents

Watch: ML Problem Formulation

As you gain more experience with formulating future ML problems, you will find that it becomes easier and easier. In this video, Mr. D'Alessandro lists a few rules and principles to consider when performing this step.

Video Transcript

The first stage in the machine learning development life cycle is problem formulation. In the spectrum of art versus science, this stage leans more to the art side for sure. One challenge this presents is there aren't well-developed theories or methodologies for approaching this stage.

I'm going to present a few principles and rules to follow. This is ultimately an area that you'll strengthen on the job. When doing practice problems, typically the problem statement is presented to you. Practice problems are great for exercising the core mechanics of model development, but at the same time, the problem statements are generally provided to you. This means you won't always get the opportunity to practice problem formulation on your own.

I'm going to give you four questions to consider each time you have a practice problem. These are based on the four rules presented in an earlier video. By repeating these questions with each exercise, you'll develop the intuition necessary to formulate business problems as ML problems. These are: What problem is the model solving, and why is it better than a non-ML-based solution?; What kind of data would you need to solve this problem with ML?; How would you solve this problem without ML?; and what are the potential risks we should anticipate? Let's fast forward to the day we are a machine learning engineer working on the job.

The first rule I want to make is don't do this step alone. You may be the only person in your company that knows how to build a model, but you shouldn't be the only person who knows how to use the model. You should engage with your team as early as possible and various domain experts to really articulate the business problem at hand. Here are a common set of questions to consider when doing this. When engaging with others, you'll want to both understand the core problem goals, but also understand the constraints. Constraints always exist, and knowing these will help you determine what is and isn't a feasible solution. One strong rule of thumb I want to emphasize is to be agile and think iteratively.

Remember the earlier rule of not underestimating the value of a good heuristic. Oftentimes, such a heuristic can capture a lot of the potential value and help your system start operating in a shorter period of time. After that, you can scope the problem in such a way that lets you make improvements in regular shorter intervals. You'll often find that each iteration provides less and less value, and eventually, it is often more worthwhile to start a new problem than continue iterating on the same problem. This rule is true for both on-the-job model development and for the problem sets you'll be doing as practice. You can start practicing this idea today.

Back to Table of Contents

Watch: Case Study: Recommendation Systems Revisited

Now Mr. D'Alessandro explains the machine learning process, focusing on the first step of business understanding and problem formulation. Let's look again at the recommendation systems case study and some different approaches we can use to solve it.

Video Transcript

Let's revisit the recommendation systems case study. Another reason that this is a good problem to analyze is the inherent flexibility we can have in terms of how we solve it. Let's continue with the YouTube example and review both our product and machine learning goals. Recommendation systems for any type of product typically start with a common data pattern. We call this the 'user item matrix.'

In this matrix, the rows represent individual users or customers, and the columns represent items. For YouTube, the columns can represent individual videos, but hopefully you can see how this generalizes. For companies like Amazon or Walmart, the items can be individual products that are being sold. The values in the matrix can vary, but they should represent some level of consumption or engagement with the specific items. For instance, we can have binary entries that show whether or not the user viewed the video. We can have numeric entries, too, that represent a rating or a level of consumption. In this case, we have entries that are 1, 2, 3, 4, or 5. We can think of this as a normalized rating that tells us how much of the video was consumed.

To actually make the recommendations, we can take both supervised and unsupervised learning approaches. We'll start with a brief sketch on the supervised learning approach. The key is to identify a good label. In this case, the label can be the entries in the matrix. If those labels are binary, we have a classification problem. In this example, the labels are the 1-5 star ratings. These are numeric and ordered, so we can approach this as regression problems. The features of the model will be whatever information you have on both the user and the videos. For video recommendations, we might use prior videos watched as features as well as descriptions in the videos, such as the genre or video author. Many of the supervised learning algorithms we'll learn will work here, but there are actually specialized algorithms that have been developed just for this type of problem.

The unsupervised approach was actually one of the first ways recommendation systems were solved. The original method went by the name of 'collaborative filtering.' There are various flavors for how this might work, but here's the basic sketch of the algorithm. In this case, we're going to use a clustering algorithm and the key is to treat members of the same cluster as the group that will help select and rank the videos from a given user.

One thing I like about this approach is that we can flip the unit and apply it to the items as well as the users. For instance, if you've ever shopped online, you might see parts of a website that show similar items to the one you've been looking at. This type of recommendation is often driven by item based clustering. Recommendation systems are fairly well established, so it is natural to expect a high degree of sophistication built into them. In a lot of cases, we might expect a hybrid approach.

The supervised learning approach tends to be more accurate in practice, but there is one key challenge. There are often just too many items, and scoring every single item for each user is computationally expensive. If those items are videos on YouTube, there are several billion videos to score and rank for each different user. This is massive and too expensive to run in a brute force way. The hybrid approach solves this problem well. The unsupervised step can be done less often. We can use the resulting clusters to filter the billions of videos down to just a few thousand. Then we can use the more powerful, supervised learning model to score the smaller set, saving us a tremendous amount of computing power.

This overview of methods for recommendation systems was intentionally shallow. Their entire course is devoted to just this class of problem. If you're interested, I highly recommend looking for them. What I mostly want to highlight are two key points. First, as we saw here, there are usually multiple ways to solve the same problem. Knowing this, it is always important to do a little design planning in advance. As we discussed in a previous video, having clear goals and knowledge of your operating constraints is important to make the right choices.

Second, we can often get to highly specialized and customized solutions by just leveraging the fundamentals. The evolution of recommender systems follow the iterative approach we previously demonstrated. The unsupervised approach was first and helped prove the case for having recommendation systems in general. Machine learning engineers then discovered that supervised learning approaches could perform better. The hybrid method was then used to get the accuracy of the supervised learning approach with the scalability of the unsupervised approach.

Back to Table of Contents

Assignment: Unit 1 Assignment - Part 3: The ML Life Cycle

In this part of the unit assignment, you will play the role of someone working for a telecom company. The management of the company is looking to address the problem of customer churn. Your task is to predict which customers are likely to churn.

Completion of this assignment is a course requirement.

Instructions:

1. Open your saved assignment document, or download the **Unit 1 Assignment document**, if you have not already done so.
2. Complete Part Three.
3. Save your work as one of these file types: .doc, .docx. No other file types will be accepted for submission.
4. Now you will submit your completed Unit 1 Assignment document for review and credit.
5. Click the **Start Assignment** button on this page, attach your completed Unit 1 Assignment document, and then click **Submit Assignment** to send it to your facilitator for evaluation and credit.

Back to Table of Contents

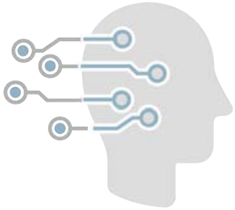
Module Wrap-up: The ML Life Cycle

In this module, Mr. D'Alessandro introduced a typical workflow of a machine learning project that can be distilled into six distinct steps as illustrated by CRISP-DM. The six steps are business understanding, data understanding, data preparation, modeling, evaluation, and deployment. Depending on the organization, the role of a machine learning engineer can range from owning any one of these steps or all of them.

Mr. D'Alessandro also listed things to consider when looking for patterns such as: is there a well-defined outcome associated with this problem? And is there a well-specified granularity in the problem? There are also common questions to consider when formulating your problem such as: What are the constraints, and what data is there? Remember to consider these questions as you delve into your ML journey.

Back to Table of Contents

Module Introduction: The ML Tech Stack



In this module, Mr. D'Alessandro introduces the programming tools commonly used to develop the ML models we have discussed through demos and tools for you to download. These tools and packages are in standard use throughout the industry, and you will use them throughout the course. You will practice working with the most popular Python packages for handling data, which will provide a solid foundation for working with data in the machine learning environment.

Back to Table of Contents

Watch: ML Workflow Applications

Before building your model, you need to think about the following:

- Where do you find your data?
- How do you access your data?
- Where do your build your model?

Watch as Mr. D'Alessandro gives a high-level overview to answer these questions.

Video Transcript

Now we're going to start getting more technical. There are a few things to know before building a model, and these are: Where do I find my data?; Where do I actually build the model?; and How do I access these different systems? This video will present a high-level overview of these questions. Let's start with data. In most of your problem sets, you're going to be just handed data. This, of course, isn't how things work on an actual job.

As a machine learning engineer in industry, you'll be accessing data sets in what is commonly called a data lake. Data is sent to the data lake through logging processes and SQL updates to databases from the application layer. We can think of the application layer as just the general software controlling the entire system. Data sent to a database is typically structured, meaning it has well-defined schema or format that is the same for every record. Standard log data is typically written to a distributed file system. These are typically Linux servers that host just plain flat files.

This kind of data can be structured, but is usually not. Unstructured data is typically easier for software developers to manage and maintain, so this is their general preference. Most of the data will work within problem sets as structured and will be provided as a flat file. In practice, it will be necessary to know SQL to query databases, as well as to know how to work with different formats to process unstructured data. Ultimately, most machine learning models require structured data as input, so converting unstructured records or examples to a fixed schema is a skill you will need to develop.

Now, once you know where your data lives and have studied its format, you'll need to access it for analysis, exploration, and modeling. We usually do this through an IDE, or an integrated development environment. The IDE is usually your main point for accessing data. Most data hosting systems let you read and write data directly to and from the IDE.

Probably the most popular IDE for model development is the Jupyter Notebook. This working format is so popular that you'll see versions of the idea in practice built by major cloud service providers as well as custom-built versions at various companies. These notebooks allow you to write and execute code like any other IDE, but they are specialized for working with data. It is pretty common to want to visualize data both by looking at the data directly or by creating various types of plots. The notebooks let you integrate visualization right in the main flow of code.

Another great feature of notebooks is they let you embed documentation directly in the flow via what they call a markdown language. This makes notebooks very shareable to the point that a lot of machine learning engineers share their work directly with notebooks instead of making formal presentations. We will use notebooks in this course, so you'll become very familiar with them.

Last, I want to introduce another tool that is very common in the machine learning tool kit workflow. This is the Linux command line interface. Big data sets don't typically live on laptop computers. To access the data, you typically need to log into a Linux-based server. These servers don't typically have nice point and click UIs, so you need to learn to navigate them using simple command line tools.

As a machine learning engineer, a few commands can go a long way. While you're a student and working with your laptop, you can still integrate the command line into your workflow. Most laptops offer a command line interface. Macs have a built-in app called iTerm that is easy to use. It is pretty common to use the command line to set up folders and do general navigation, and then to directly launch Jupyter notebooks. We'll demonstrate this in a subsequent video.

Back to Table of Contents

Watch: Demo: Linux

The Linux Command Line is one of the tools used in the machine learning workflow to access raw data stored on Linux servers. Watch as Mr. D'Alessandro demonstrates the basics in this video.

Video Transcript

I'm now going to present a demo that represents a common workflow in a given ML project. Specifically, we're going to set up a working environment. Here I'm going to show you a few examples of command-line tools to follow, then we're going to launch Jupyter Notebooks, do a little bit of work, and then save some of that work back into the laptop, and just verify that the work was completed successfully.

Now we're also going to launch Jupyter Notebooks from our computer. I need to mention that Jupyter Notebooks is something that you usually have to install yourself. It's not native to most like Mac or Windows installations, so usually you would use something like Anaconda or the Anaconda distribution to get Jupyter Notebook set up. What I have here is just an example of the iTerm command-line interface, which is native to a Mac.

So, first thing I might want to do is start from a clean slate. I'm going to use this cd command, and when I say 'cd' with nothing else after it and hit 'Enter', it's going to take me to my computer's directory, so I can use 'pwd', which is 'present working directory,' to tell me where I am. And you can see the output is right here. The first thing I want to do is set up a working environment. So I already have a folder called 'mle', which is within my 'Documents' folder. I'm going to start there, and the first thing I'm going to do is make a folder for this course. Let's call it 'btt_course', then I'm going to change into that particular folder, and again, if I say 'pwd', you can see the full file path right down here.

Now, I want to set up some subfolders within this environment, so first thing we want is we want a data folder. 'mkdirdata'. Next, I'm also going to put a folder in place for my code. I think it's generally smart to keep your code and your data separately. Oftentimes, code you want to back up or commit to a collaborative repo like GitHub, but you don't generally want to store your data there as well, so it's good to have them separated. I'm also going to create a folder for the output just so we can keep our images and any illustrations that we create separate. Let me just check that everything is here. This 'ls' command lists everything that's available within a folder. That '-ltr' presents it in this what I consider easier to read format with a lot of extra details. If I just use 'ls', you could see it's like this as well.

Next thing we want to do is we want to put data into the data folder, so something that I did before the demo is I downloaded data from the Internet; it went into a folder called 'Downloads,' so let's double-check we're in the home. Now I'm going to go into the 'Downloads' folder. I also want to verify that the folder is there. I remember that the file had the word 'cell' on it, but I forgot exactly the name of the file. When I use this ls command and I take a snippet of the file name and I wrap it around these asterisks — these are wild cards — so it'll say, basically this will return any file in this folder that has the word 'cell' in it. I can confirm that the data is there, so let me copy it.

Now, the next thing I want to do is I want to move this file, 'cell2cell_data' and put it into the folders we just created. I started the file path with this tilde — this is actually shorthand for my root directory, and then the rest is 'btt_course' and then 'data' and then 'cell2cell'. And so it doesn't tell you anything, so you have to go back into that folder and just verify yourself that it was there. Yeah, we can see it's there, so now let's go on to the next steps. I'm going to use this command cd and then dot dot. This actually takes us just to one folder prior in the file path, so now I'm in the root directory of this particular project.

Again, we can look what's here; now, I want to — oh, notice that the the data is listed last because this is one that had the most recent update. Now I'm going to go into notebooks, this is where I want my actual code to live, so to launch the notebook, I use the command 'jupyter notebook'. Again, Jupyter is likely something that you'll have to install yourself; the Anaconda distribution is highly recommended. Once I hit this, it's going to process for a few seconds; mine is defaulted to Firefox, so you can see in here, this is what the Jupyter Notebook environment looks like.

Please note that Mr. D'Alessandro uses the term "root" to describe "home", but there is only one "root" and one "home."

Back to Table of Contents

Read: Machine Learning Workflow Applications

Linux Command Line Interface

One of the tools used in the machine learning workflow is the Linux Command Line. Data is often stored on a Linux-based server; to access this data directory, you have to log in to the server and, using a terminal, navigate to the data using typical command-line tools.

One key thing to understand when working with Linux commands is the concept of file paths. A path is a representation of the location of a file or directory (folder), often including one or more directories leading up to that location. For example, let's say you create a folder on your Desktop named "myFolder" and in that folder is a file named "myFile.txt." The path to "myFile.text" would be "/Users/myUserName/Desktop/myFolder/myFile.txt." Keep in mind that it's important that your "current working directory" is the directory in which you are currently working.

It is worth noting that you don't always have to be logged in to a Linux server in order to use a terminal and Linux commands; you can install a Linux terminal on your local computer (macOS comes with a Linux terminal) and use Linux commands to access your local files and work on your own machine learning project.

Commonly used Linux commands:

- **pwd:** present working directory or print working directory
  - `pwd`: outputs your current working directory
- **cd**: change directory command.
  - `cd` OR `cd ~`: navigates to your home directory
  - `cd /`: navigates to your root directory
  - `cd [directory name]`: navigates to a directory in the current working directory
  - `cd [path name/directory name]`: navigates to a directory in a specified location
- **mkdir:** make directory command
  - `mkdir [directory name]`: creates a directory in your current working directory
  - `mkdir [path name/ directory name]`: creates a directory in a specified location
- **ls**: lists files and directories
  - `ls`: lists all files in the current working directory
  - `ls [path]`: lists all files in a specified location
  - `ls -ltr`: using the `-ltr` option with the `ls` command prints more information about files in your current working directory or in a specified directory

Jupyter Notebooks

Another tool used in the machine learning workflow is an integrated development environment (IDE) that allows you to import your input data and build your model. Jupyter Notebooks is a rich IDE that has become a dominant mode of development and communication within the field of machine learning. The Jupyter Notebook contains much more than just code; it enables you to integrate code, documentation, commentary, results, and data visualizations so that you can package together your machine learning models and analyses to better convey the results of those analyses.

Jupyter is a very general framework that supports interactive computing for a number of different programming languages, but in this course, it will be used with the Python programming language.

Jupyter Notebooks supports the IPython interpreter

Python is an interpreted language, which means that a separate program called an interpreter is required when you actually want to run Python code that you have written. While the programming language adheres to a specified set of rules, there exist different interpreters that can augment support for the Python language with additional features.

The default interpreter for Python is a program called "Python" — possibly "python.exe" on a Windows system. That program understands only the Python programming language and is especially useful for running whole programs that you have written in a code editor and want to run from the command line. Yet if you want to do interactive work with Python, where you might load some data, inspect it, do some preliminary analyses, or make some plots, a different interpreter called "IPython" should be used; it contains additional features to support that kind of interactivity. In fact, IPython also supports the use of file-related Linux commands. IPython forms the core of Jupyter Notebooks.

Jupyter Notebook file extension

A Jupyter Notebook file is different than the Python source code files that you create. Jupyter Notebooks define a different file format for storing code, documentation text, and figures. A Jupyter Notebook ends with exntention `.ipynb`, short for "IPython notebook," rather than ending in the standard `.py` extension for Python code files.

Note: In this program, you will not have to install Jupyter Notebooks on your computer or work with it in a web browser; the Jupyter Notebook IDE is embedded directly in this course and ready for you to begin your work.

Back to Table of Contents

Read: Useful Features of the IPython Interpreter

The IPython interpreter forms the core of Jupyter notebooks. It is an alternative interpreter for Python, intended to support interactive work. With enhanced support for accessing help and documentation, command-line editing, and special "magic" functions, IPython is especially useful when you want to explore your data and try out some new analyses. Below is a list of IPython features that you will use in your Jupyter Notebooks. Refer back to this page as needed as you work through your exercises.

☆ Key Points

IPython is an alternative interpreter for Python that's useful when you want to explore your data and try out new analyses.

IPython provides magic functions that are useful in an interactive context but aren't recognized by the default Python interpreter.

IPython forms the core of Jupyter notebooks.

It is important to keep in mind, however, that some of the features that IPython provides — especially the magic functions — are not part of the Python language itself. Magic functions in IPython begin with a percent sign, `%`, such as in the magic functions `%ls` (to list files in the current directory), `%run` (to run a Python code file in the interpreter), or `%timeit` (to time how long it takes to execute particular operations). And while these functions are useful in an interactive context, they will not be recognized by the default "Python" interpreter that you might use for command-line, Python-only programs.

Here are some other useful built-in IPython functions and magic functions you can use in Jupyter Notebooks:

- IPython provides several ways of getting documentation about datatypes and functions.
  - The built-in `help()` function: prints out help about an object; run `help(object_name)`. You can also use the `help()` function to get information on general types, e.g. `help(list)`, `help(dict)`, `help(int)`. To exit the interactive help system, type `q` and press enter or return.
  - You can also get information about an object by typing `?` after the name of an object or function, e.g. `list?` will provide short documentation about that object.
  - The built-in function `dir()` prints out just the group of names of methods (and potentially other attributes) associated with an object, without all the documentation

    that `help()` provides; run `dir(object_name)`. You can also use the `dir()` function to get information on general types, e.g. `dir(list)`.

- IPython magic functions are very useful for querying and manipulating data in the interpreter.
    - `%lsmagic` provides a list of available magic functions.
    - `%run` allows you to run your code in a source file within the interpreter.
    - `%history` allows you to see your input command history, and `%pdoc` prints the documentation text (docstrings) associated with an object, e.g. `%pdoc list`.
    - Another useful pair of magics are `%who` and `%whos`. `%who` describes what variables have been created in your main namespace and %whos provides summary information about those variables.
    - If you want more information about a particular magic, you can just type the name of the magic and append ? to it. For example, `%who?`.
- Many IPython magic functions correspond to familiar terminal commands used in a Linux/Unix environment. These commands are simply prepended with %. Examples of commonly used commands are `cat`, `cd`, `cp`, `less`, `ls`, `man`, `mv`, and `pwd`. These commands enable you to manipulate and interrogate files on your local machine.
    - Type `%ls` to list files in your current directory. This can be very useful if you are trying to read in a file or run some code, and want to check that the file is where you think it is.
    - If you're not sure what directory you are in, type `%pwd` to print the working directory.
    - If you would like to move to another directory, type `%cd` followed by name of the directory, or path to the directory.

Back to Table of Contents

Tool: The Structure of a Notebook

In this course, you will be coding and submitting assignments using Jupyter Notebooks. Thus, it is important for you to familiarize yourself with how to use its core features. Use the guide linked to the right to explore how to run code, reset code, and submit code in a Jupyter Notebook. You will also find steps in the guide for how to download and save a Jupyter Notebook locally. No need to memorize the content now, but do have it handy as you work your way through the next practice activity.

Download the Tool

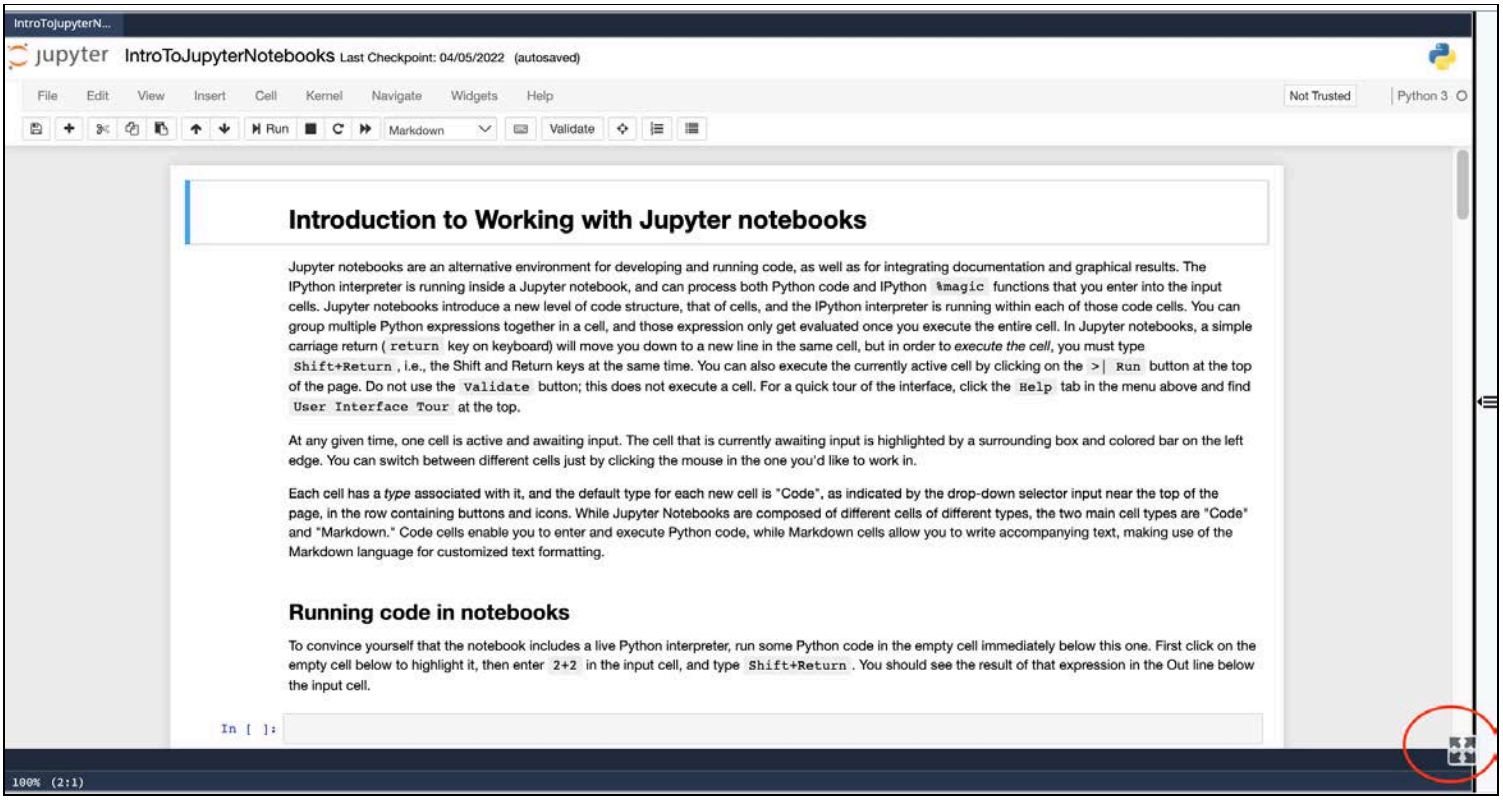
Use **The Structure of a Notebook** tool as a guide to navigate Jupyter Notebook.

Back to Table of Contents

Read: Coding Exercise Submissions and Grading

This program will contain a variety of assignments, such as written assignments, quizzes, coding assignments and labs. Throughout each week you will have coding exercises in which you will work in a Jupyter Notebook.

As mentioned, the Jupyter Notebook IDE will be embedded directly in this course and will be ready for you to begin working. For best experience, work in fullscreen mode by clicking on the lower right button as indicated in the image below:



Introduction to Working with Jupyter notebooks

Jupyter notebooks are an alternative environment for developing and running code, as well as for integrating documentation and graphical results. The IPython interpreter is running inside a Jupyter notebook, and can process both Python code and IPython `%magic` functions that you enter into the input cells. Jupyter notebooks introduce a new level of code structure, that of cells, and the IPython interpreter is running within each of those code cells. You can group multiple Python expressions together in a cell, and those expression only get evaluated once you execute the entire cell. In Jupyter notebooks, a simple carriage return (`return` key on keyboard) will move you down to a new line in the same cell, but in order to *execute the cell*, you must type `Shift+Return` , i.e., the Shift and Return keys at the same time. You can also execute the currently active cell by clicking on the `>| Run` button at the top of the page. Do not use the `Validate` button; this does not execute a cell. For a quick tour of the interface, click the `Help` tab in the menu above and find `User Interface Tour` at the top.

At any given time, one cell is active and awaiting input. The cell that is currently awaiting input is highlighted by a surrounding box and colored bar on the left edge. You can switch between different cells just by clicking the mouse in the one you'd like to work in.

Each cell has a *type* associated with it, and the default type for each new cell is "Code", as indicated by the drop-down selector input near the top of the page, in the row containing buttons and icons. While Jupyter Notebooks are composed of different cells of different types, the two main cell types are "Code" and "Markdown." Code cells enable you to enter and execute Python code, while Markdown cells allow you to write accompanying text, making use of the Markdown language for customized text formatting.

Running code in notebooks

To convince yourself that the notebook includes a live Python interpreter, run some Python code in the empty cell immediately below this one. First click on the empty cell below to highlight it, then enter `2+2` in the input cell, and type `Shift+Return` . You should see the result of that expression in the Out line below the input cell.

There will be three types of coding exercises:

- Ungraded practice activities
- Graded assignments that are automatically graded
- Graded assignments that will be graded by your facilitator

Each exercise will be designated accordingly.

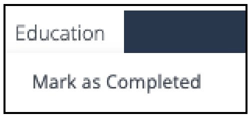
Type	Description	Icon
Ungraded Practice Activities	Ungraded practice activities are intended for you to practice techniques and concepts taught throughout the week.	</>
Graded Assignments	All graded assignments will contain the bull's eye icon in the exercise title.	🎯

Graded Assignments:

All graded assignments require that you submit your work in the following manner:

When you finish your work:

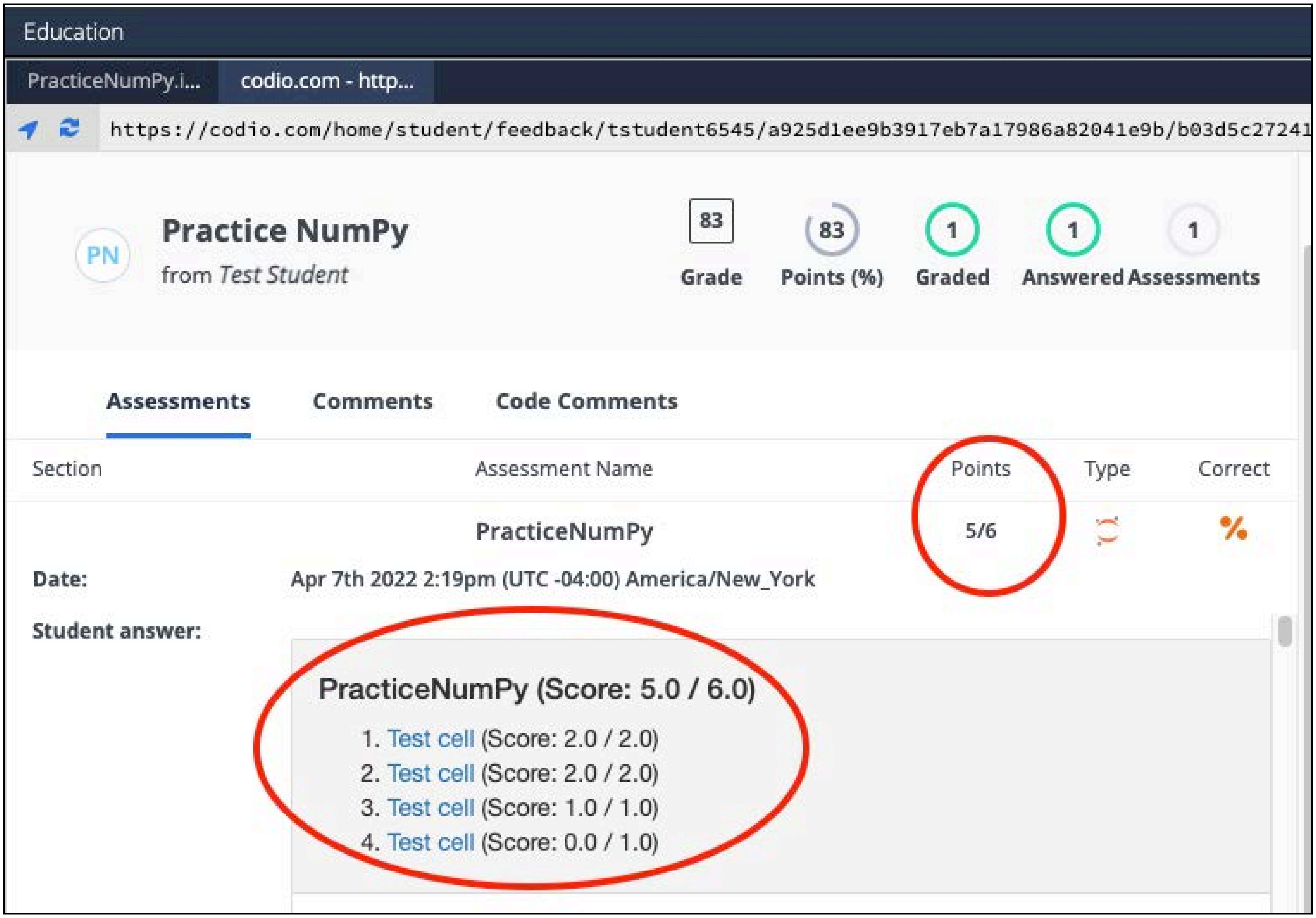
1. Save your Jupyter notebook by selecting the "Save and Checkpoint" entry from the "File" menu at the top of the notebook. If you do not save your notebook, some of your work may be lost.
2. Submit your work by clicking **Education —> Mark as Completed** in the upper left of the Activity window



Automatically Graded (auto-graded) Assignments:

Some exercises will be "auto-graded," or automatically graded. In such cases, you will receive a grade upon submission.

Once you click on **Education —> Mark as Completed,** a tab will appear next to your Jupyter Notebook containing information about your grade. The tab will contain a display that lists your overall grade, in addition to a break down of points per cell. You can click on an individual **Test Cell** to see more information.



https://codio.com/home/student/feedback/tstudent6545/a925d1ee9b3917eb7a17986a82041e9b/b03d5c27241

PN **Practice NumPy**
from *Test Student*

Grade	Points (%)	Graded	Answered	Assessments
83	83	1	1	1

Section	Assessment Name	Points	Type	Correct
	PracticeNumPy	5/6		%

Date: Apr 7th 2022 2:19pm (UTC -04:00) America/New_York

Student answer:

PracticeNumPy (Score: 5.0 / 6.0)

1. Test cell (Score: 2.0 / 2.0)
2. Test cell (Score: 2.0 / 2.0)
3. Test cell (Score: 1.0 / 1.0)
4. Test cell (Score: 0.0 / 1.0)

Facilitator-Graded Assignments:

Some assignments will be graded by your facilitator. In such cases, after you click on **Education —> Mark as Completed,** you will have to wait for a grade from your facilitator. Just as in automatically graded assignments, as soon as you submit your work, a tab will appear next to your Jupyter Notebook. This will not contain a grade and should be ignored.

Back to Table of Contents

Familiarize Yourself with Jupyter Notebooks

In this exercise, you will practice working with Jupyter Notebooks. The tool "The Structure of a Notebook" contains some tips on how to use Jupyter Notebooks. Refer to this tool as you progress through the program. The tool is also contained in a collapsable panel to the right of every Jupyter Notebook.

- To access the tool, click on the small black arrow ☰.
- To close the guide containing the tool, click on the "collapse" button contained at the top of the guide Collapse.

This activity will not be graded. Ungraded activities will contain the following icon in the exercise title: </>

This activity will not be graded.

Please complete this activity in the course.

Back to Table of Contents

Tool: Linux and IPython Cheat Sheet

Use this tool as your cheat sheet to IPython and Linux.

For full online documentation about IPython, visit this link:

https://ipython.readthedocs.io/en/stable/index.html.

For full documentation on magic functions, check out this page:

https://ipython.readthedocs.io/en/stable/interactive/magics.html.

Back to Table of Contents

Download the Tool

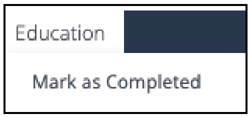
Use the **Linux and IPython Tip Sheet** as your guide to complete exercises.

Assignment: Jupyter Notebook Practice

In this exercise, you will get more practice working in a Jupyter Notebook.

This exercise will be graded. Most exercises in this program will be graded. Graded exercises will contain the bull's eye icon in the exercise title.

When you finish your work:

1. Save your notebook by selecting the "Save and Checkpoint" entry from the "File" menu at the top of the notebook. If you do not save your notebook, some of your work may be lost.
2. Submit your work by clicking **Education —> Mark as Completed** in the upper left of the Activity window 

3. This assignment will be auto-graded and can be resubmitted. After submission, the Jupyter Notebook will remain accessible in the first tabbed window of the exercise. To reattempt the work, you will first need to click **Education —> Mark as Uncompleted,** then proceed to make edits to the notebook. Once you are ready to resubmit, follow steps one and two.

For general information on using Jupyter Notebooks, you can expand the right-hand panel adjacent to the Notebook by clicking on the icon.

Note: To execute a cell in a Jupyter Notebook, press shift-enter or select **Run** in the menu bar. Prolonged inactivity or navigating to a different page will cause the notebook session to timeout. To resume your work where you left off, be sure to re-run all of the notebook cells in order, starting again at the very beginning.

This exercise will be auto-graded.

Please complete this activity in the course.

Back to Table of Contents

Watch: ML Python Packages

As a machine learning practitioner, you'll work with many different packages from the Python ML ecosystem. In this video, Mr. D'Alessandro focuses on the ones we will be using in the course. These include packages such as NumPy, Pandas, Matplotlib and Seaborn, and Scikit-learn.

Video Transcript

A machine learning engineer will need to have a solid foundation of computer science and software development. This course will emphasize what is needed to work with data and build models solely within the Python environment. In this video, I'll introduce a few of the core packages we will be working with. I think one of the reasons that the roles of data scientist and machine learning engineer have been so successful is the rich set of open source tools that are available to suit the specific needs of the field. We owe a lot to the various Python developers that have contributed to the core libraries that every machine learning engineer should know.

We'll go through each of these at a high level, between various demos and the exercises, you will have ample opportunity to practice using them. A lot of the packages we use for analysis and data manipulation leverage NumPy on the back end. NumPy is a general purpose array processing package. Almost all machine learning applications can be reduced down to operations on arrays. NumPy makes array processing much faster than using base python.

Two core properties of NumPy to call out are vectorization and broadcasting. Vectorization is a property that significantly speeds up your code. When working with logic where loops might be appropriate, it is generally better to replace loops with NumPy array operations when possible. The other is broadcasting; broadcasting is a process that enables us to perform array operations like addition and multiplication with very simple and efficient code expressions. Python has its core data objects such as lists, tuples, and dictionaries. Pandas are so popular that the core data object, the data frame, has earned its status as a core Python data object to know.

Some of the highlights to call out are that Pandas enables a lot of relational data operations that are necessary for working with structured data. Pandas uses a different syntax than common SQL, but the type of operations are generally the same. Additionally, Pandas is well integrated into the machine learning ecosystem. Nearly all of the common machine learning libraries enable directly working off of Pandas data frames, making the transition from data preparation to modern building pretty seamless.

Visualization is one of the areas where we have a lot of options to choose from. In this course, we will use the combination of Matplotlib and Seaborn. Matplotlib is easily the most common visualization tool in Python. It is incredibly flexible and offers a large range of plotting options; at the same time, its interface can be challenging to learn. Seaborn is built off of Matplotlib, but has a much easier interface to work with. For most of your basic plotting needs, Seaborn will be enough. Using them together enables us to build our knowledge of Matplotlib, but also enjoy the ease of use with Seaborn.

Scikit-learn is easily the most important library for a machine learning engineer to know. It is probably the most complete modeling package available. Being free, so complete, and easy to use, it is also the most popular package by far. You will use Scikit-learn for most of the machine learning needs in this course. There are a few areas where other packages would be better, though; these are for neural networks and advanced natural language processing, and even some more traditional statistical modeling.

This course will offer opportunities to practice using all of these packages. We won't cover all of the package features in our instructional material, but there is ample documentation around what each package covers and how to use them. One skill to develop is accessing the packages' official documentation and learning the APIs. The popularity of these packages means the documentation is generally incredibly thorough, and there are many examples and discussions across various blogs and sites like Stack Overflow.

Back to Table of Contents

Tool: ML Python Packages

This tool is a quick reference guide for the packages discussed in the videos. The packages are listed in the order in which they were introduced in the videos, which groups things together that are related or provide similar sorts of functionality.

Back to Table of Contents

Download the Tool

Use this helpful list of **ML Python Packages** as a quick reference guide for identifying the appropriate Python ecosystem component(s) for your machine learning projects.

Quiz: Check Your Knowledge: Identifying Which Package to Use Part 1

In this quiz, you will be presented with various scenarios. You will determine which package you would use to help solve the problem.

You may take this quiz up to three times.

Please complete this activity in the course.

Back to Table of Contents

Watch: Demo: Packages

In this video, Mr. D'Alessandro demonstrates how to import and work with some of the packages.

Note: It will be easier to see the text in this demo when you watch in full screen.

Video Transcript

So, once I hit this, it's going to process for a few seconds. Mine is defaulted to Firefox, so you can see in here, this is what Jupyter Notebook environment looks like. First thing I need to do is create a new file. Let's call this one 'Demo'. So what we want to do in this demo is read our data, put it into a data frame, and then do some plots, and then save that plot.

So the first thing I need to do is actually import the packages I'm going to use. This is running; notice how when it's running, there's a bit of — it displays an asterisk in these little brackets, and when it's done, it shows a number — again, that's how you know it's done. Here are the three packages that we're using: Pandas, Matplotlib, and Seaborne. notice I've created aliases for these; these just make it easier to basically type your code.

Now, one thing I'm going to do is I want to just create a variable for the root directory. It's something I plan to be using several times but I forgot exactly what it was called. I'm going to go back in here. So sometimes I might just go in here and copy and paste. Go back to Firefox. Now our infile is rootdir + it was in data, and it was called cell2cell_data.csv. These are all strings, so we're just concatenating two strings. I didn't say this earlier, but when you want to run within the Jupyter Notebook, you can use the UI and hit 'Run', but on a Mac, at least I can hit 'Return' and 'Shift' at the same time, and it runs the cell for me.

Next, what I'm going to do is read the file, so this is a csv file, and I want to put it into a data frame, so Pandas has a method called 'read_csv'. The most simplest use is just basically give it the file path and I'll put it into a data frame that we call df. Something I also like to do is just use this head command, df.head. This display is the first five records. If I wanted to make it 10 records, I could do that — shows me more. And this is a great just kind of gut check to do on your data just to make sure that all the columns were read incorrectly, that the output actually looks the way you would expect it.

What we want to do, you know, let's assume there's a business problem we're looking to solve and we needed to understand what is the distribution of this revenue feature? When we think about distributions, a good tool to use is the histogram. Seaborn has a histogram method called 'Histplot' and the simplest way to use it is just call Histplot, and the data frame is df, and then I'm going to reference the revenue column, just 'df.revenue', hit 'Return'. And you can see we have a histogram with a skewed distribution; this is pretty common for feature types that include revenue or income or anything representing currency. Now, this looks good.

So what I want to do next is I want to save this. Let's remember we had a rootdir and we made a specific folder called 'output', and let's just call this 'revenue_histogram.jpg' and then we're going to run it. Looks like it returned something, so let's check where we are. Now what we want to do is let's go into the Output folder. The revenue histogram is here.

Let's assume that we're done, so let's go in and save our work and just one more thing to check. Let's go into the Notebooks folder. And again, you can see this is where our demo is. This is a common workflow, and one of the main reasons to show it this way is to demonstrate the fluidity of having to work within your IDE as well as your computer and using the Command line to quickly navigate between folders and move files around and things like that.

Please note that when Mr. D'Alessandro says "return and shift runs the cell," he means to say, "shift + return." Refer back to the Machine Learning Workflow Applications read page for the correct command and more.

Back to Table of Contents

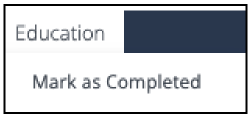
Assignment: Practice NumPy

In this exercise, you will practice working with the NumPy package in a Jupyter Notebook.

This exercise will be graded. Most exercises in this program will be graded. Graded exercises will contain the bull's eye icon in the exercise title.

When you finish your work:

1. Save your notebook by selecting the "Save and Checkpoint" entry from the "File" menu at the top of the notebook. If you do not save your notebook, some of your work may be lost.
2. Submit your work by clicking **Education —> Mark as Completed** in the upper left of the Activity window



3. This assignment will be auto-graded and can be resubmitted. After submission, the Jupyter Notebook will remain accessible in the first tabbed window of the exercise. To reattempt the work, you will first need to click **Education —> Mark as Uncompleted,** then proceed to make edits to the notebook. Once you are ready to resubmit, follow steps one and two.

For general information on using Jupyter Notebooks, you can expand the right-hand panel adjacent to the Notebook by clicking on the icon.

Note: To execute a cell in a Jupyter Notebook, press shift-enter or select **Run** in the menu bar. Prolonged inactivity or navigating to a different page will cause the notebook session to timeout. To resume your work where you left off, be sure to re-run all of the notebook cells in order, starting again at the very beginning.

This exercise will be auto-graded.

Please complete this activity in the course.

Back to Table of Contents

Tool: Numpy Array Tip Sheet

NumPy arrays support a wide range of operations. The tip sheet linked from this page lists many common operations.

Back to Table of Contents

Download the Tool

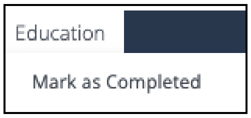
Use this **Numpy Array Tip Sheet** to aid you when working with Numpy.

Assignment: Practice Pandas

In this exercise, you will practice working with the Pandas package in a Jupyter Notebook.

This exercise will be graded. Most exercises in this program will be graded. Graded exercises will contain the bull's eye icon in the exercise title.

When you finish your work:

1. Save your notebook by selecting the "Save and Checkpoint" entry from the "File" menu at the top of the notebook. If you do not save your notebook, some of your work may be lost.
2. Submit your work by clicking **Education —> Mark as Completed** in the upper left of the Activity window 

3. This assignment will be auto-graded and can be resubmitted. After submission, the Jupyter Notebook will remain accessible in the first tabbed window of the exercise. To reattempt the work, you will first need to click **Education —> Mark as Uncompleted,** then proceed to make edits to the notebook. Once you are ready to resubmit, follow steps one and two.

For general information on using Jupyter Notebooks, you can expand the right-hand panel adjacent to the Notebook by clicking on the icon.

Note: To execute a cell in a Jupyter Notebook, press shift-enter or select **Run** in the menu bar. Prolonged inactivity or navigating to a different page will cause the notebook session to timeout. To resume your work where you left off, be sure to re-run all of the notebook cells in order, starting again at the very beginning.

This exercise will be auto-graded.

Please complete this activity in the course.

Back to Table of Contents

Tool: Pandas DataFrame Cheat Sheet

Using Pandas in conjunction with your Python programming knowledge enables you to develop expressive custom workflows and data analyses. You can use the provided Pandas DataFrame Cheat Sheet to remind you of the syntax of DataFrames and how to use them to group and reshape data.

Download the Tool

Use this **Structure of a DataFrame Object** tool to aid you when working with Pandas DataFrames.

Refer to the **documentation on Pandas** from pydata as well.

Back to Table of Contents

Quiz: Check Your Knowledge: Identifying Which Package to Use Part 2

In this quiz, you will be presented with various scenarios. You will determine which package(s) you would use to help solve the problem.

You may take this quiz up to three times.

Please complete this activity in the course.

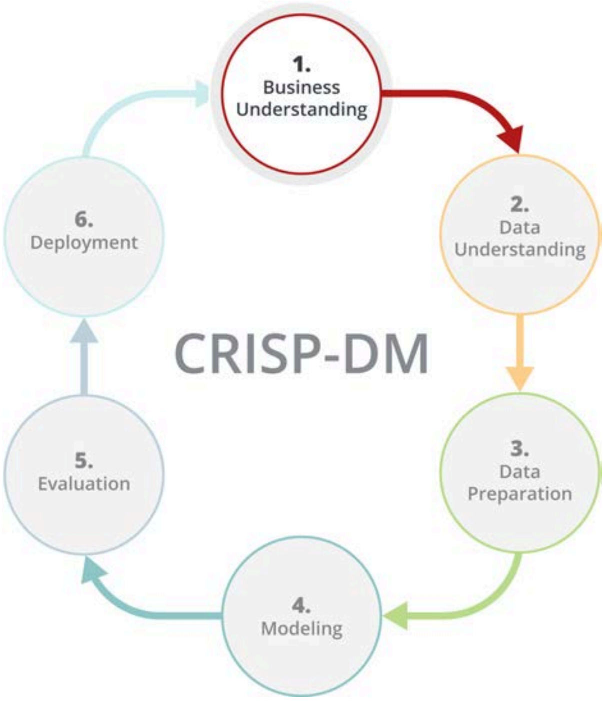
Back to Table of Contents

Module Wrap-up: The ML Tech Stack

In this module, you explored some of the most important tools used within the machine learning industry. Like learning how to ride a bike, the first few times you attempt to use these tools might require you to use "training wheels." Use the tools in this module as your guides and support, and refer back to them often. With practice, patience, and repetition, you will eventually become so familiar with and comfortable using most of the tools from this module that you will forget why you needed guides to begin with.

Back to Table of Contents

Lab 1 Overview: ML Life Cycle: Business Understanding and Problem Formulation



In this lab, you will practice the first step of the machine learning life cycle: business understanding and problem formulation. You will also continue working with the ML Python tools NumPy and Pandas. Your practice will provide comfort and confidence in working with NumPy arrays and Pandas DataFrames. You will be working in a Jupyter Notebook.

This three-hour lab session will include:

- **20 minutes** - Icebreaker
- **20 minutes** - Week 1 Overview and Q&A
- **20 minutes** - Breakout Groups: Big-Picture Questions
- **10 minutes** - Class Discussion
- **10 minutes - Break**
- **35 minutes** - Breakout Groups: Lab Assignment Working Session 1
- **10 minutes** - Working Session 1 Debrief
- **35 minutes** - Breakout Groups: Lab Assignment Working Session 2
- **10 minutes** - Working Session 2 Debrief
- **10 minutes -** Concluding Remarks and Survey

In Lab 1, you will:

- Practice working with ML Python tools:
  - Use NumPy to create arrays and perform operations on them.
  - Create Pandas DataFrames from data.
  - Load a dataset and use NumPy and Pandas to analyze the data.
  - Visualize the data with Matplotlib.
- Practice machine learning problem formulation using four different datasets.

Back to Table of Contents

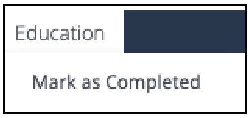
Assignment: Lab 1 Assignment

In this lab, you will be working with the **Airbnb "listings" data set**, which contains New York City Airbnb listings from December 2021. The data set has been modified for this program. For more information about these modifications, click **here**.

You will also work with the **Housing Prices** data set, the **Restaurant Business Rankings 2020** data set, the **Zoo Animal Classification** data set, and the **Flight Delay** data set.

This assignment will be graded.

When you finish your work:

1. Save your notebook by selecting the "Save and Checkpoint" entry from the "File" menu at the top of the notebook. If you do not save your notebook, some of your work may be lost.
2. Submit your work by clicking **Education —> Mark as Completed** in the upper left of the Activity window .
3. Note: This assignment will be manually graded by your facilitator. You cannot resubmit.

This lab assignment will be graded by your facilitator.

Please complete this activity in the course.