Data Wrangling with Pandas for Machine Learning Engineers

by Mike West

Course Overview

Course Overview

Hello. My name is Mike West, and welcome to my course Data Wrangling with Pandas for Machine Learning Engineers. While artificial neural networks are getting all the attention, one of the most overlooked aspects of machine learning is the data. Regardless of the algorithm type, almost all machine learning models need well formatted, structured data to perform optimally. It's the job of the machine learning engineer to wrangle the data into a modellable state. Data wrangling is one of the most difficult and time-consuming parts of machine learning. In the real world, data is dirty and machine learning models are temperamental. These models only want highly-structured, well-cleansed data. In this course, we'll provide you with the foundation you need to wrangle those unruly datasets. The course will introduce you to applied data wrangling. You'll want to have developers take real-world datasets and wrangle them to highly-structured numerical entities machine learning models need. The core library used by machine learning engineers to wrangle their data in Python is called pandas. You'll learn how to manipulate tabular data in an array. The array is the core data object in machine learning. Once the data has been properly wrangled, you'll build a highly accurate model that will predict a person's survivability if they were aboard the Titanic at the time of the sinking. Python has become the gold standard in applied

machine learning, and a library called pandas, the preferred tool utilized by developers to massage their data into a well-cleansed state. By the end of the course, you'll be familiar with the basics of data wrangling and the process machine learning engineers use to create well-cleansed, model-ready datasets. I hope you will join me on this journey to learn more about data wrangling with Python at Pluralsight.

Getting Started in Data Wrangling

Introduction

Hello. My name is Mike West, and welcome to my course Data Wrangling with Pandas for Machine Learning Engineers. In this course, we're going to cover the basics of data wrangling. Machine learning is one of the most in-demand careers in the world and will be for a long time to come. A large part of real-world machine learning is data wrangling. This first module will provide some basic information about data wrangling. This includes an explanation of the two core types of machine learning, supervised and unsupervised learning. Machine learning is very process oriented. This first module will cover that process and explain how data wrangling fits into the larger picture. Machine learning models don't like poorly structured data. The cleaner the data oftentimes the better the model's performance. By the end of this module, you'll understand why data wrangling is so important to machine learning. We're also going to cover the skills you need for the course and the skills you do not. Finally, we're going to work through a simple example in Python and use pandas to remove our first attribute from our dataset.

Why Take This Course?

Machine learning is one of the most in-demand careers in the world and data wrangling one of the most in-demand skills within that career. Data wrangling isn't easy and it's a time-consuming endeavor. However, it's a skill all applied machine learning engineers will have to master in order to secure a top-level position within this space. Much of applied machine learning is data wrangling. Recent surveys by Kaggle and CrowdFlower found that machine learning engineers and data scientists spent up to 80% of their time wrangling data. Regardless of the true percentage of time spent massaging data, machine learning engineers will be spending a lot of

their daily routine molding their data into a modellable state. All supervised machine learning models ingest some kind of numerical data. If that data is incomplete the model's performance will likely suffer. The machine learning engineer's goal is to build a model with a high degree of accuracy. This won't be an easy task without a well-cleansed dataset. This course is about learning real-world skills you can immediately apply to your datasets before building your models. The cleaner the data oftentimes the better the model's performance.

Overview of the Course

Let's take a few minutes now to go over the contents of the course. The first thing we're going to cover is the pandas DataFrame. A DataFrame is similar in concept to an Excel spreadsheet or a table in a relational database, and it's the core object in pandas we're going to be working with. In the course, we're going to clean the famous Titanic dataset. The sinking of the Titanic is one of the most infamous shipwrecks in history. Although there was some element of luck involved in surviving the sinking, some groups of people were more likely to survive than others, such as women, children, and the upper class. Once our dataset has been properly cleansed we're going to build a model from it that will help us predict a person's survivability if they were aboard at the time of the sinking. Additionally, we're going to cover manipulating our data in pandas using a variety of functions. While our dataset is structured in a tabular fashion it does need some cleansing. We're going to cover several techniques you'll be able to use on just about any dataset you encounter for wrangling your real-world data. At the end of the course we'll have a short review on what you've learned and where to go from here on your journey to learn more about applied data wrangling. Before we move ahead, let's cover some skills you'll need for the course and some skills you will not need. You do not need to have any prior experience in machine learning. A basic understanding would be beneficial, but it's not required. While this course will cover the machine learning process, the focus will be on one library used in data wrangling. For this course, you won't need a deep understanding of Python or what the core data science libraries are. The course will introduce you to pandas, the most popular library for data wrangling in Python. You won't need any advanced math skills. One of the myths surrounding machine learning engineering is that the career requires a lot of advanced mathematics and statistics. This simply isn't the case. If you made it through high school geometry and algebra you'll be just fine. It's important to keep in mind machine learning engineers don't author any of these models. They use ones that already exist. While you don't need advanced Python development skills a basic understanding of how Python works will help you get the most out of this course. If you've installed Python locally and know the basics of Notebook navigation then you're ready for the

course. In machine learning we don't call them rows and columns, we call them attributes and observations. Regardless of the nomenclature, you'll need to have a fundamental understanding of how an Excel spreadsheet works or a table in a relational database. While it's true that advanced math and stats aren't required, basic math and stats are required. For example, we're going to be loading data into a table-like structure called an array so an understanding of matrixes and arrays would be very beneficial.

Supervised vs. Unsupervised Learning

Machine learning is divided into two separate types of tasks. Those tasks are called supervised learning and unsupervised learning. The first type is called supervised machine learning. In supervised machine learning we feed the machine learning model a dataset we have the answer to. That nicely cleansed dataset will be fed to our model for processing, and eventually the model will make a prediction based on the patterns it finds in the attributes of that dataset. In our sample, we have the first five observations of the Titanic dataset. This is called a subset of our data. All the observations are called a dataset. Most of us are familiar with rows and columns; however, in machine learning rows are called observations and columns are called attributes. In our dataset, we have a column or an attribute that has the answer we are looking for in every row in our dataset. We call this value the target variable. In our dataset, our target variable is binary, meaning there are only two possible outcomes. A 1 means the passenger survived the sinking and a 0 means the passenger did not. We ask the algorithm to learn patterns in that dataset based on all the other columns excluding the target variable in our dataset. We expect the algorithm to be able to make accurate predictions on the data it's never seen after we've trained it on our dataset. In the real world 90% of all modeling is done using supervised machine learning. The second type of machine learning is unsupervised. In unsupervised machine learning there is no existing data. Unsupervised machine learning is a more complex process. When people talk about computers teaching themselves to learn rather than us having to teach them they are often alluding to the process of unsupervised machine learning. In unsupervised learning there is no training dataset and the outcomes are unknown. The model goes into the problem blindly. In our example, the model looks at the images of humans and rabbits and is eventually able to classify them into two distinct groups. The model learns on its own and is eventually able to classify what images are rabbits and what images are humans. In our example, we only have two images. However, in a real unsupervised model the algorithm would have hundreds if not thousands of images. Incredible as it seems, unsupervised machine learning is the ability to solve complex problems using just the

input data and the binary on and off mechanisms that all computer systems are built on. There is no reference data at all.

The Machine Learning Process

Machine learning is very process oriented. Machine learning engineers and data scientists follow the same four steps every time they start the process of creating their predictive model. Most real-world machine learning is supervised. Therefore, the first step is to source the data. Data comes from relational databases and it's the number one reason, the number one required skill for a machine learning engineer is SQL. The next step is data cleansing. This is often the most time-consuming part of the process and it's the step where data wrangling takes place. In this course, we're going to spend most of our time at this step in the machine learning process. The third step in the process is model building. This is where we build and tune our models. These models are complicated mathematical equations that look for patterns in our data then make predictions based on those patterns. The final step in the process is prediction. We've built a great model and now we're ready to feed it data it's never seen. The end goal of the entire process is to have our model make accurate predictions on fresh data. Fresh data is data the model has never seen before. Now you might be thinking where does all this data come from? While it can come from many sources, there are typically three sources of real-world data. Relational databases have been around for decades. Corporations have amassed tons of structured data in every facet of their business. Much of this data remains untouched by analysis. However, the explosion of machine learning is changing that. Currently, most real-world machine models are sourced by relational databases, machine learning models like clean tabular datasets. An easy way to share these datasets without the overheard of a relational database is a CSV document. The word CSV is an acronym that stands for comma-separated value. These documents are most prevalent with new learners because many online document repositories and tutorials use CSV documents. While a lot of modeling is currently done with relational data, relational data is only the tip of the iceberg. Ninety percent of all data collected globally is unstructured or what we often refer to as big data. While this is the most prevalent data it's also the most difficult to work with and massage into a modellable state.

Removing an Attribute Demo

Let's complete a quick demo in Python 3. 6. The IDE for the course will be a Jupyter Notebook. The first step will be to import the pandas library into a DataFrame and massage the data. After

that, you're going to do some actual wrangling by removing an unimportant attribute. Finally, you'll peruse the results of your wrangled dataset. Open a Jupyter Notebook and give it a name. Let's load the first line of code. The first line of code loads the pandas library and creates an alias called pd that can be used instead of having to type out the name of the library every time the code is called. An alias will allow you to type less code. Once the code has run successfully a new cell will appear. When that cell appears load another line of code. A pandas DataFrame is a lot like an Excel spreadsheet. In this line of code, we are creating a variable called data to hold our dataset. Next, let's load the dataset into a DataFrame using the read_csv function. Take note you are aliasing pandas using the alias created in the code above. Once the code in the cell is run, let's see what's inside the data variable. Let's use another function called the head function. This function allows you to view the first five observations of the dataset. Machine learning models need clean numerical data. What attribute jumps out at you if your thought process is focusing on clean numerical data? Right, the name column. Not only is it not numerical, it's unique within the dataset. If the goal of the model is to look for numerical patterns in our data then a unique text column won't provide our model with anything relevant. That means it has to go. In order to remove that attribute let's use another function. The drop function can be used to remove that column from the dataset. In this line of code let's use the data variable that houses our dataset and call the drop function on the fourth column. This is a little confusing. If the goal is to remove the fourth column then why is there a 3 in the line of code? In Python the first position is always a 0. That means you have to start counting from the 0 position. If you start counting at the 0 position, the name column is at position 3 in the dataset. Once that cell is run let's load our next line of code. The next line of code calls the head function again. The name column has been removed from our dataset and you've just taken your first step towards wrangling the data into a more modellable state by removing the name attribute.

Summary

This module covered the difference between the two core types of machine learning, supervised and unsupervised. Supervised learning uses highly structured data for model building. Supervised learning is the most prevalent in the applied space and it's why data wrangling is so important to machine learning. The success of the model is more dependent on data than any other facet in the process. This is not a trivial statement. Many new to this space focus too much time on model building and are often disappointed to learn that in the real world much of your time will be spent wrangling data. In this module, you learned about the machine learning process. More specifically, the supervised machine learning process was covered because much of applied machine learning

is focused on supervised learning. Lastly, you took your first step in wrangling the dataset by removing the name attribute. This step was very simple, but it is indeed data wrangling.

Pandas DataFrame Basics

Introduction

Hello. I'm Mike West, and welcome back to Data Wrangling in Python for Machine Learning Engineers. This second module provides some more detailed information about the core object in pandas, the DataFrame. This includes defining the DataFrame, understanding why the DataFrame was built on top of NumPy arrays, and why we use methods to manipulate the data. The module will also cover data types. Data types define how data is stored. Using the correct data type is critical to the outcome of your model. For example, you can't use mathematical methods on an object data type so storing numbers correctly is very important to successful data wrangling. Additionally, this module will cover the sundry components of the DataFrame. The DataFrame has three core components and you'll learn what they are in this module. In SQL a select statement is used to retrieve data. In pandas indexers are used. This module will define and show you how indexers return data. After loading, merging, and preparing a dataset a familiar task is to compute group statistics or possibly pivot tables for reporting. In this module, grouping and aggregation will also be covered. Additionally, you'll continue wrangling the Titanic dataset. Upon completing each module, your dataset will be one more step closer to the highly structured numerical dataset these models need.

DataFrame Overview

The DataFrame is one of the core objects in pandas. Pandas DataFrames all share the same properties. All DataFrames are two-dimensional arrays. Arrays are the main data structure used in machine learning. In programming terminology, an array is a collection of data where each value has an index or location associated with it. Pandas DataFrames sit on top of NumPy arrays. NumPy is a Python library that can be used for scientific and numerical applications. The word NumPy is short for Numerical Python. The main data structure in NumPy is an ndarray, which is shorthand for N-dimensional array. When working with NumPy, data in an ndarray is simply referred to as an array. It's a fixed-sized array in memory that contains data of the same type, such as integers or floating points. A DataFrame is a table much like in SQL or Excel. It's similar in

structure also, making it possible to use similar operations such as aggregations, filtering, and pivoting. However, because DataFrames are built in Python it's possible to use Python to program more advanced operations and manipulations than SQL or Excel can offer. Additionally, because these DataFrames sit on NumPy arrays they are very fast. DataFrames are useful because more powerful methods are built into them. In Python, methods are associated with objects, so the data must be in the DataFrame to use these methods.

Data Types

In a recent demo, data was loaded into a pandas DataFrame. When that happened pandas assigned data types to each column based on what the column contained. That decision had important downstream implications on how the data was manipulated later. Data types determine what you can and what you can't store in a DataFrame. How data is stored in a DataFrame affects how you can manipulate it and the outputs of the calculations as well. For example, you can't perform mathematical calculations on a string or text data. This might seem obvious; however, sometimes numeric values are read into a DataFrame as strings. In this scenario, when you try to perform calculations on string-formatted numeric data you're going to receive an error. There are three core data types in a pandas DataFrame. The first one is the Int64. This data type stores whole numbers. One, two, three, and four are examples of whole numbers. The next data type is the float64, and it stores numbers with decimal places even if that decimal point is a 0. If we have a column that contains both integers and floating points, pandas will assign the entire column to a float data type so the decimal points are not lost. The next core object stores text, and it's called an object data type. In addition to storing text and string, the object data type can also contain numbers. A string might be a word, a sentence, or several sentences. The naming convention for this data type is a little strange and often confusing. In NumPy and Python text is stored as a string data type. The term object does tend to confuse people, and you'll often hear people say that text in pandas is stored as a string data type. However, the proper term is an object in pandas vernacular.

DataFrame Anatomy

Let's take some time to learn the anatomy of a DataFrame. A DataFrame is composed of three different components: the index, the columns, and the data. The first object, the index, represents the sequence of values on the far left-hand side of the DataFrame. Each individual value of an index is called a label. The second set of objects are called columns. They are a sequence of

values at the very top of the DataFrame. Each individual value of the columns is called a column, but can also be referred to as a column name or a column label. Lastly, we have our values. Everything that's housed inside our DataFrame is the data. Let's take a look at a sample DataFrame. You may be thinking that the unlabeled index in our DataFrame looks similar to the column labeled UserID. The label UserID is part of the dataset that was imported. This is likely a monotonically increasing primary key from a relational database, and it's not part of the structure of our DataFrame. The index will rarely, if ever, have a column header. You will sometimes hear DataFrames referred to as tabular data. This is just another name for rectangular data with rows and columns. It's also common to refer to columns and rows as axis. Collectively they are called axes. So, a row is an axis and a column is another axis. The word axis appears as a parameter in many DataFrame methods. Pandas allows you to choose a direction on how the method will work with the parameter.

DataFrame Navigation

Let's take a look now at DataFrame navigation. The process of recreating only the rows and columns we need in a Pandas DataFrame is called indexing. If you're familiar with relational databases then this is no different than using a SELECT statement on a table. The group of objects used to slice our DataFrames are called indexers. There are two core indexers. The two most commonly used ones are. loc and. iloc. The loc is short for location. The first indexer is called iloc and it stands for integer location. The integer is used to select rows and columns by number in the order they appear in the DataFrame. Remembering that iloc stands for integer location will help you remember the differences between the two. The. loc indexer selects data in a different way than just using the index. It can select subsets of rows or columns. Most importantly, it only selects data by the label of the rows and the columns.

Grouping

Categorizing a dataset and applying a function to each group, whether an aggregation or a transformation, is often a critical aspect of a data analysis workflow. Grouping lets you slice up the rows of a DataFrame into, well, groups that have the same values in one or more categorical variables. These are useful because you can easily calculate statistics for each group and aggregate the results into a new DataFrame. One of the reasons of the popularity of relational databases in SQL, which stands for a structured query language, is the ease with which data can be joined, filtered, transformed, and aggregated. However, query languages like SQL are rather

limited in the kinds of group operations that can be performed. Additionally, larger datasets often take a long time and a large amount of resources to complete. Fortunately, pandas provides all the functionality that SQL does but returns the results much faster. Using NumPy arrays, you can perform much more complex grouped operations by utilizing any function that accepts a pandas object or a NumPy array. This makes grouping and aggregating in pandas much more robust than any vendor flavor of SQL. In a DataFrame and a series all the most common aggregating attributes are accepted.

Navigation Demo

Like many things in machine learning, data wrangling is a process-oriented endeavor. The first task is to import the libraries needed for the project. Secondly, because this is pandas, you'll need a dataset to work with. In this demo, the Titanic dataset will be used. Once the dataset is in a variable, you'll use indexers to view various parts of it. Additionally, you'll sort and group the dataset. Finally, you'll wrangle another attribute of the dataset. Let's open a Jupyter Notebook and give it a name. Next, let's execute one line of code at a time. I'm using the Shift+Enter key shortcut on my Windows keyboard to execute each cell. On the first line of code the pandas library is being imported. Once that cell has been run it's time to import the dataset and store it in a variable. In our sample, the variable holding our dataset is called data. In the next line of code, the head function is being called. This will allow us to view the first five observations of our dataset. Skipping a few rows, let's navigate to the cell with iloc in it. In this line of code, iloc is being run on the first axis in our dataset. Recall that 0 is the first row or axis in the dataset. Let's stay with iloc. Since iloc is being used you need to feed integers in as parameters. This time, two integers are being fed into iloc. The first one is a 0 and the second one is an 8. This statement will return all the rows between axes positions 0 and 8. Don't forget that the axis will start at 0. Alright, now let's move on to loc. In this line of code, a label is being passed. The label being passed in this example is the name column. This line of code prints out all the observations of the name label. The next line of code is similar to the one we just executed. However, another label has been added. This label is the age column. This line of code will return all the observations in the dataset for the name and age attribute. In our next line of code, a condition is being added. That condition is equality. In Python, two equal signs is equality and one equal sign is assignment. This line of code is returning the name of the passenger we want passed into it. Take note that the match must be exact. If we leave a few letters off the name attribute Python won't return the observation. In this module, we learned about grouping and aggregating. Let's apply some commonly used groupby functions on our dataset. In the next line of code a groupby is being

applied to the sex and the survived attribute using the passengerID column as the item being counted. The output tells you males didn't fare so well surviving the sinking. In the next line of code, the groupby function is again being used. Here, the groupby returns the average or the mean age of the passenger in each class. Now that navigation has been covered, let's continue wrangling the dataset. In the first module the name column was removed. Let's execute the drop function on the name attribute. This is the same step that was completed in the demo for module 1. Now, let's execute a similar line of code against the passengerID column, which is at position 0 in the axis. The passengerID column was most likely a monotonically increasing key from a relational database. This adds no value to our dataset so it has to go. The dataset is looking much cleaner and we've only removed two attributes from it. The two attributes that were removed were the name attribute and the passengerID attribute.

Summary

In this module, the DataFrame took centerstage. The start of the module began detailing various core components of the DataFrame. The core object in machine learning for storing data is an array. A DataFrame is a table-like object that sits on top of an array. DataFrames are built on top of NumPy arrays. This architectural decision ensures that most operations performed on a DataFrame are very fast. Similar to tables in a relational database, DataFrames use data types to store various kinds of data. The three core data types are Int64, float64, and the object data type. DataFrames are composed of indexes, columns, and rows. The index will often be numeric and it will be visible on the left-hand side of the DataFrame. When navigating a DataFrame you'll use indexers. The two core indexers are iloc, which uses the index by number, and loc, which uses an object's labels. Lastly, grouping and aggregating in pandas are much faster thanks to the NumPy array.

Pandas Data Structures

Introduction

Hello. I'm Mike West, and welcome back to Data Wrangling in Python for Machine Learning Engineers. This third module will provide some more detailed information about the other core object in pandas, the series. A pandas series is a one-dimensional array of indexed data. The array is the main data structure in all of machine learning. In this module, the array and other similar

structures will be covered. For example, Google has a framework called TensorFlow. A tensor is nothing more than a multi-dimensional array. In this lesson, the importance of domain knowledge will be discussed. Thus far, you've been able to pick some easy attributes to remove. Removing the name attribute and an imported primary key didn't take a lot of real dataset knowledge. In order to wrangle the dataset much further you'll need more detailed knowledge about the dataset. Python uses a lot of methods and functions. In this module, the core functions used in data wrangling will be covered. For example, almost all datasets have missing values. Handling missing values correctly is critical to improving model performance. Additionally, the module will cover a real-world, abridged guide to data wrangling. This will include a step-by-step process of wrangling real-world data. Finally, you'll wrangle the rest of the dataset. Upon completion of this module the dataset will be model ready. That means the entire dataset will be composed of numbers.

The Series Object

A series represents a one-dimensional, labeled indexed array based on the NumPy ndarray. While that's an extremely technical definition, at first just think of a series as a column in a table with an index attached to it. The index is on the left-hand side of the series. The index has no column header and begins with a 0. Beside the index is our data. Recall that there are three core data types used in a DataFrame and the series. The first is integers and they are whole numbers. Second are floating points and they have decimals. And lastly is the object data type and it stores text and string data. Like an array, a series can hold 0 or more values of a single data type. A series can be created and initialized by passing either a scalar value, an ndarray, a Python list, or a Python dictionary.

Vendors and Arrays

The array is one of the most fundamental data objects within all of machine learning. An array is a collection of data or values where each value has an index or location associated with it. Regardless of the platform you use for your machine learning needs, the concept of an array is used throughout all of machine learning. Pandas DataFrames are built on NumPy arrays. When building models locally on your laptop or desktop, NumPy will be the array used most often. TensorFlow is Google's computational framework and the key object is a tensor, which is a multi-dimensional array. If you're building models on Google's cloud platform, often referred to as GCP, then the tensor will be the array of choice. MXNet and Gluon are two frameworks from Amazon

and their array is almost identical to the NumPy array. Their array is called an ndarray. The array is the core object using supervised machine learning to feed data into the models. Regardless of where you build your models, the array or structure similar to the array will be used to house that data.

Array Types

Arrays have different dimensions. The most basic array is the one-dimensional array. The data points move in only one direction. In our example, we have a one-dimensional array with four data points. In any array we use index positions to locate our data. If you need to locate a data point associated with passenger Owen Braund, you would say this data point is located at 0, 1 within our array. If you wanted to locate the age of Mr. Braund you would use 0, 2 to locate that data point within the array. The next type of array is a two-dimensional array. The data points in this array move in two directions. A set of data points moves horizontally and another set moves vertically. In our sample, if you wanted to locate the passenger with the name John Cumings you would say he's at data point 1, 1 within the array. Lastly, we have a three-dimensional array. Think of a three-dimensional array as a rubrics cube where each smaller cube represents a different number but moves throughout space as a single cube. The data points now move in three dimensions. One set of data points moves horizontally, one vertically, and one into a space that we can only view when visually representing a three-dimensional object, like a rubrics cube. Each dimension will have different data points on them. These three arrays are the foundation for all data modeling in supervised learning.

The Titanic Dataset

Thus far, you were able to remove certain attributes from the dataset without a granular understanding of the dataset. However, before moving forward, a more thorough examination of the dataset is in order. While names and database keys are easily removed, data wrangling involves understanding the data you are modeling. In data wrangling, there is no substitute for domain knowledge. Domain knowledge is knowledge about the data you are wrangling. Fortunately, the Titanic dataset has been hand-labeled and that means someone has gone through the dataset and massaged the data for you. Unfortunately, in the applied space this will be up to you. While many of the columns are self-explanatory a few are not. Let's cover each of them to be thorough. The name column is the passenger's name. This column has already been removed from the dataset based on analysis from the fist module. The column age is the age of

the passenger at the time of the sinking. The sex column defines whether the passenger was male or female. The column SibSp stands for siblings of passengers aboard. Did the passenger have a brother or sister aboard? The column Parch means what were the total number of parents and children aboard? Fare is how much the passenger paid for the ticket. The column Pclass stands for passenger class. There were three classes aboard the Titanic, first, second, and third. First-class passengers were on the upper level, second-class passengers were on the middle, and third-class passengers were on the bottom level. The Cabin column is the room they were in. And lastly, embarked is the location where the passengers boarded the boat. Now you know what all the columns and attributes mean. Start thinking about what attributes you believe will contribute to the passengers' survivability. Dataset familiarity is necessary in order to determine what columns will be removed from the dataset. This is one of the most important first steps in the data cleansing process.

Abridged Data Cleansing Steps

While there is no data wrangling playbook, these steps are a good starting point to begin the data cleansing process. The first step is to remove any unneeded attributes from your dataset. At this juncture, the question you'll need to ask of your data is what attributes can be easily removed that you know for certain are not needed? For example, when you import data from a relational database you'll often import the primary key. Since most relational database keys are surrogate keys, meaning the key has no business logic, you'll need to remove them. The second step in the process is to remove duplicate data or empty data. Recall that relational databases account for much of the data in machine learning. Relational data is often dirty, filled with holes, and duplicates. You'll need to remove all those duplicate values. Once those dupes are gone you'll need to fill those empty values in the DataFrame. Models don't like missing values. It's a much better practice to fill those empty values with contrived or artificial data than it is to leave them blank. In the third step, you'll want to ensure that all the data is formatted properly. All punctuation will have to be cleansed. Additionally, you'll want to update any incorrect data. The final dataset will need to be numeric and will have all the values in the DataFrame filled in.

Methods and Functions

Python relies heavily on methods and functions. While there are hundreds of functions in Python you'll use a few of them often to wrangle data. Pandas offers two functions to test for missing data, isnull and notnull. These are simple functions that return a Boolean value, indicating whether

the passed in argument value is in fact missing data. You have two options for handling these values. You can replace them or you can drop them. In pandas, missing values show up as NaN values. Recall that NaN stands for not a number. In order to drop all the NaN values from a DataFrame simply call the dropna function. Be careful though, calling dropna on the entire DataFrame will drop all occurrences of NaN values in the entire DataFrame. Dropping only one column can be done by passing the value parameter instead of the method. Most of the time you won't want to drop all those NaN values. Instead of dropping all those values another option is to call fillna. Fillna will change those values to the value you specify. The data wrangling process involves finding duplicate data and removing those duplicates. In order to find duplicates in a series call the duplicated function. Once you've found those duplicates you can remove them by calling drop_duplicates. Lastly, you'll often need to alter the column used in an index. Set_index will allow to change the column are columns you want to be the index column.

Final Wrangle Demo

In this demo, you'll start by importing pandas and reading the dataset into a DataFrame. These initial steps will almost always be the same. Missing values are a hindrance to most models. In this demonstration, you'll learn how to find those missing values and then replace them. Relational data will often have duplicate values. While you can remove them before exporting your data keep in mind that pandas DataFrames are much faster than most relational database systems. In this demonstration, finding and removing duplicate values will be covered. Recall the final dataset will need to be numerical. Therefore, you need to convert the sex column to integers. Lastly, you'll peruse the final dataset and clean anything you might have missed during the wrangling process. Before the final wrangle, let's cover removing duplicates. Since there are no dupes in the Titanic dataset, we're going to have to create a contrived dataset to work with. In our first line of code pandas is being imported. Next, we're going to create a dataset filled with contrived data. When our second line of code is executed the dataset is created and the contents are printed out. Take note there are three Mike Wests. While it appears there are two duplicate values there is really only one. The age value in one of those observations is 111. That's most like an error in the data; however, it's not a duplicate value for our purposes. If this were live data you'd have to have someone do some research to find out more information about that customer. In order to remove the dupes from the DataFrame we simply call drop_duplicates. After that's executed the DataFrame has no more duplicate values. Let's navigate back to the Titanic dataset and wrangle it from the very beginning. In the first two lines of code pandas is being imported and the dataset is loaded into a DataFrame. Next, the head function is being called on the first 10 rows. From an

attribute perspective, this is the entirety of the dataset. In the next line of code only the columns selected will be returned. It's already been established that the name and passengerID columns need to be removed. Let's return all the rows, excluding the name and passengerID columns. In this module, you learned that the tickets were randomly assigned to the passengers. Therefore, let's remove the ticket column. The embarked column is the location where the passengers boarded the boat. This is also a random occurrence and that means the embarked column can be wrangled out. Next up is the cabin attribute. Attributes with lots of NaN values tend to attract your attention. In the cabin attribute it appears there are lots of NaN values. Let's call the info function on all the attributes to see how many NaN values the column has. Viewing the results, you can see that the cabin attribute only has 204 values and that means the rest will be NaNs. If this were an important attribute to the dataset then we would have to stop our analysis on this dataset. Since the attribute doesn't provide us with any real value it can be removed. The fare column is the amount the passenger paid for the ticket. If the passenger class attribute didn't exist then the column might have more importance. Fortunately, the passenger class tells us where the passengers were located on the ship and that means we can remove the fare column. In this line of code, the head function was called on the first 15 rows or observations. The age column is one of our most important attributes. Women and children first was enforced. However, the age column does have a few NaN values. The results only show one value but one is too many. There are two options here, remove all the NaN values or replace them. For this wrangle let's replace them. A common approach to filling NaN values in a numeric column is to replace them with the mean value of that attribute. In order to do that the fillna will be called on the age column with the mean parameter. This will replace all the NaN values in that series with the mean age. That mean age is 29. At this juncture, the dataset looks much better and much cleaner. The one attribute that still jumps out is the sex column. It's still text. Let's fix that by mapping males to 0 and females to 1 in the dataset. You're done. The dataset is numeric and you've removed all the unimportant attributes. This dataset is now model ready. Finally, let's write out our cleansed dataset to a CSV file. This is our final cleansed dataset ready for the modeling process.

Summary

In this module the series object was covered. A series object is a column in a table with an index attached to it. The module also covered the array, the main data structure in all supervised machine learning. Regardless of the platform you build your models on, your data will need to be in an array or an array-like object. Wrangling involves domain knowledge. In the real world you'll have SMEs, or subject matter experts, you'll rely on to understand the data. Wrangling your

dataset requires you have intimate knowledge of it. The module also covered several functions you'll use often in pandas. There is no science behind filling empty or NaN values. You'll have to test various models on a variety of datasets. Most of the models I've built use the mean value of the attribute. Once you begin cleansing real-world data you'll develop your own routine and process. To get you started, this module provided you with some basic steps on how machine learning engineers approach data wrangling. Lastly, the demo walked you through a simple wrangle of the Titanic dataset. Data wrangling is one of the most important skills a machine learning engineer will need to possess in order to excel in the applied space.

Modeling the Cleansed Data

Introduction

Hello. I'm Mike West, and welcome back to Data Wrangling in Python for Machine Learning Engineers. This fourth module is about modeling our cleansed dataset. Machine learning engineers often affectionately refer to this as the fun part of the job. Building and tweaking models in order to get the best predictive capability is a rewarding process. In this module, the two core types of machine learning models will be covered. There are only two and you'll learn what they are in this module. Many machine learning models fall into four broad categories. For example, we will learn the difference between classification and regression, two categories of models that are often used in the applied space. In this module, the core steps of building a predictive model in scikit-learn will be covered. Model building, like data wrangling, is a process-oriented endeavor, and scikit-learn has become the gold standard for building traditional machine learning models in Python. Model selection for those new to machine learning can be very difficult. How do you know what model to choose for what project or task? This module will cover a general guide to model selection. Lastly, you're going to be building several models using the dataset that was cleansed throughout the course. Your data wrangling efforts will be rewarded with a successful real-world model.

The AI Hierarchy

Artificial intelligence makes it possible for machines to learn from experience, adjust to new inputs, and perform human-like tasks. Artificial intelligence is the top-level container that holds all of machine learning. Most AI examples that you hear about today, from chess-playing computers

to self-driving cars, rely heavily on deep learning and natural language processing. Using these technologies, computers can be trained to accomplish specific tasks by processing large amounts of data and recognizing patterns in the data. Machine learning can be defined as a set of algorithms that parse data, learn from them, and then apply what they've learned to make intelligent decisions. These models are often referred to as traditional models. In Python, these models can be built using a library called scikit-learn. In this module, you'll use scikit-learn to craft your models. A neural network is a computer system designed to work by classifying information in the same way your brain does. It can be taught to recognize, for example, images and classify them according to elements they contain. Artificial neural networks live under the machine learning umbrella. You might be thinking why would I use a traditional model over an artificial neural network? Aren't artificial neural networks better? While it is true that deep learning models will often outperform their counterparts the cost is very high. Deep learning models require a lot of data and are often very resource intensive. The model you'll be building can be constructed on a laptop with very little data and used in the real world.

The Four Machine Learning Categories

Earlier in the course you learned that there are two broad categories of machine learning, supervised and unsupervised learning. Under those two broad types were model categories. Let's cover the four broad categories or techniques that fall under supervised and unsupervised learning. The first two are supervised learning tasks and the latter two are unsupervised learning tasks. Possibly the most common type of applied machine learning is classification. A classification model attempts to draw some conclusions from observed values. Given one or more inputs, a classification model will try to predict the value of one or more outcomes. Outcomes are labels that can be applied to a dataset. For example, in this course you're building a binary classification model. There are only two possible outputs in the model, hence the term binary classification. A 1 means the person survived and a 0 means they did not. The next model also falls under the supervised learning umbrella and its linear regression. Simple linear regression is useful for finding relationships between two continuous variables. For example, predicting housing prices or stock prices are examples of linear regression. The idea is to determine a line that best fits the data. The next technique is called clustering and it falls under the unsupervised learning umbrella. Clustering is a machine learning technique that involves the grouping of data points. Given a set of data points, we can use a clustering algorithm to classify each data point into a specific group. In theory, data points that are in the same group should have similar properties or features while data points that are in different groups should have dissimilar

properties or features. The next technique is called association and it's an unsupervised learning approach. Association rule-based learning is a rule-based machine learning method for discovering relationships between variables in large databases. It's often referred to as market basket analysis. The goal is to find associations of items that occur together more often than you would expect from a random sampling of possibilities. The classic example is the famous beer and diapers association that is often mentioned in data mining books. The story goes men who go to the store to buy diapers will also tend to buy beer at the same time. If you were working for the beer company who discovered this unique association then you might suggest a beer display in the aisle where the diapers are sold. Machine learning is about finding patterns and making predictions in datasets. These patterns might not seem obvious to you and I at first; however, they make perfect sense to the algorithm.

SciKit-learn Model Building Steps

Scikit-learn is a library in Python used for building traditional machine learning models. It's important to remember the library is not used for building deep learning models. It's only used for building traditional models. When building any model in scikit-learn there are three core steps. After your data has been wrangled and you've chosen the model there are three things you need to do with every model. The first step is to fit the model. The term fitting the model is synonymous with training the model. In order to fit the model to the data the fit method is called. Now that you've trained your algorithm it's time to make some predictions. In order to do that, the predict method is called. This does exactly as the name implies. It will make predictions based on the patterns it finds in your data. Lastly, you'll need to evaluate the model's accuracy. In this course you're building a model to predict a person's survivability if they were aboard the Titanic at the time of the sinking. Classification problems are perhaps the most common type of machine learning problem and that means there are a lot of metrics that can be used to evaluate predictions for this problem. The most commonly used metric is accuracy. Classification accuracy is the number of correct predictions made as a ratio of all the predictions made. You might be thinking what's a good number for a model's accuracy? Well, that's a very good question. If your model's close to 50% then that's no better than the flip of a coin. If your model's accuracy is 90% then there's a good chance you're suffering from overfitting, a condition where the model learns the data too well. If your model is above 75 and below 90 then you're probably in a good accuracy zone.

Model Selection Process

When you're given a problem how do you know what model to choose? That's not an easy question. Fortunately, in the applied space some decisions are made for us. This makes the model selection process a little easier. For example, you've learned that most real-world models are supervised and that means you need data. Before we go any further, let's walk through a simple selection process. In this course we've been given a dataset to model. Therefore, it's a supervised learning problem. The next question is how much data have you been given? If you have less than 500 samples or rows then you'll need more data. The general rule is the more data the better. Our dataset has more than 500 samples so we can move forward. The next questions you'll need to ask are are you predicting a category and is the data labeled? If the answer is yes to predicting a category then you have a classification problem. If the data is not labeled then it's back to data wrangling. In the real world the person doing the labeling will often be the machine learning engineer. If your data wasn't labeled then you'd have to go back through the dataset and add the survived or not survived column. If you are predicting a quantity then you'll look to regression. You've already learned that the model is predicting a category so that means you have a classification problem. Recall that there are two possible outcomes for every sample, a 1 or a 0, and that means our model is a binary classifier. A suggested metric for deciding between a support vector machine and stochastic gradient descent is the amount of data involved. However, one of my first choices regardless of the amount of data is the support vector machine model. These models tend to do very well with binary classification problems. In the applied space many of your models will be binary classifiers. For example, did the customer buy a product? Will the driver survive an accident? Will the customer choose your restaurant? Additionally, in scikit-learn, you can easily switch between models often by altering only one or two lines of code. You'll start your testing by using an SVC but then eventually decide on a decision tree. Decision trees also work very well on binary classification problems. You might be thinking how do I know what models work on binary classifiers and which ones work best on natural language processing? Unfortunately, there is no concrete answer to this question. Only time and experience will aid in the model-selection process.

Model Building Demo

The dataset has been properly cleansed and now it's time to build a model from it. In this demo, we're going to import the libraries we need. In addition to importing pandas, you'll be importing models from a library called scikit-learn. The name used for the scikit-learn library is called scikit-learn. Once the dataset has been properly cleansed you'll build a support vector machine. This model works be creating the line that separates the data points. Once that line has been found

the model will separate any input based on where it falls specific to that line. After the SVC has been scored you'll build another model called a decision tree. A decision tree is a set of rules used to classify data into categories. It looks at the variables in a dataset, determines which are the most important, then comes up with the tree of decisions that best partitions the data. The tree is created by splitting data up by variables and then counting to see how many are in each bucket after the split. Once that's been completed, you'll choose the model that performed the best. This dataset was part of a competition hosted on a website called Kaggle. Because you took the time to properly cleanse your data, the final model would have scored in the top 2% in that competition. Additionally, this model is production ready. Many of the winning models on Kaggle are never used in the real world because they are over engineered. The one you build will be highly accurate and production ready. Alright, let's begin coding. In the first line of code you'll important pandas. Next, you'll load the dataset into a DataFrame. And lastly, you'll call the head function to peruse the dataset. On the next line of code, you'll important the columns you'll use for the model. Let's call the head function again to ensure your data looks clean. Our data still has text in the sex column so let's clean that up. Once that line of code is executed it's the head function once again to view our results. In our next line of code let's drop all the NaN values. This will reduce the dataset, but it will also give us a true representation of the data. Let's load our next line of code. The survived column is being dropped, and it's being mapped as our target variable. In the next line of code, train_test_split is being imported and executed against the dataset. This line of code splits the data into a training set and a testing set. In the next line of code, the support vector classifier is being imported. Note that scikit-learn calls this model an SVC for support vector classifier. It's the same model as a support vector machine, just named differently in the scikit-learn library. In the next line of code, let's call the fit method to train the model. When you call this method, you'll see that sundry parameters are printed out. These can be adjusted to improve model performance. The defaults will do just fine for our purposes. In our next line of code, the predict method is called. Additionally, the model is being scored. The score for this model is 77%. That's not a bad first run. However, we can do better. Let's try a different model. Take note that all the steps are identical expect for the two lines of code that import our decision tree model and the one that will allow us to determine the splits in our tree. Since all the code is identical, let's just walk through it until we arrive at the code that's different. Please take note that even importing the model is almost identical to importing the SVC. The only code that's changed is the name of the model. The next line of code is the important one. The min_samples_split specifies the minimum number of samples required to split an internal node. Tweaking this parameter allowed the model to score much higher. As an aside, tweaking parameters in machine learning is called hyperparameter tuning. After we fit, predict, and score

our model a score of 84% was achieved. This is a great score on this dataset. Additionally, it's a real-world model that can be immediately placed in production. Congratulations, you've just wrangled a dataset. And with a few lines of code and a little bit of tweaking you were able to score higher than 90% of all the people in the world who modeled this dataset.

Summary

In this module, the hierarchy of artificial intelligence was covered. Artificial intelligence is the top-level container. Machine learning is a type of artificial intelligence. And lastly, artificial neural networks are a type of machine learning. This module also covered the core four. At a high level, models can be grouped into four different categories. When you're building a supervised learning model you'll choose between classification and regression. When you're building an unsupervised learning model you'll choose between clustering and association. In scikit-learn there are three core steps to building your model once your data has been cleansed and your model has been chosen. They are fit, predict, and evaluate. You train the model on the data, the model makes predictions, and lastly the model is evaluated, often for accuracy. Model selection is not an easy process. In applied machine learning most problems are supervised. This makes the decision-making process a little easier. If you've been given a dataset or have to create one then you'll choose between classification and regression. In the demonstration, you used the dataset you wrangled throughout the course to build two different models. One was a support vector classifier, or an SVM, and one was a decision tree. The decision tree model was the one ultimately chosen as the winner.

Course Summary

As we wrap up this course, let's spend some time recapping our data wrangling journey. We started the course learning about the two core types of machine learning, which are supervised and unsupervised learning. We focused on supervised learning because most of applied machine learning involves building predictive models against existing datasets. The machine learning process was covered with an emphasis placed on data wrangling. The single most important facet to building accurate machine learning models is the data. The course focused on the pandas library in Python and an array-like object called a DataFrame. The DataFrame is the key object that houses the data you'll build your models against. You've also learned that data inside a DataFrame is stored based on data types. There are three core data types. Each data type holds a specific type of data. The Int64 holds integers, the float64 holds numbers with decimal places,

and the object data type holds textual information. These DataFrames are built on top of NumPy arrays making them very fast. The other core object in pandas, the series object, was covered. A series is nothing more than a column in a table with an index attached to it. The index simply provides you with a way to access the different data points within the DataFrame. Regardless of the platform you use for data wrangling or your model building needs, the array is the core data object in all of machine learning. TensorFlow, MXNet, Gluon, NumPy all use the same object for their data, a multidimensional array. Data wrangling is impossible without knowing the data. You'll need to understand what every single attribute means. In the applied space that means you'll need to rely on a subject-matter expert to understand the data. Data wrangling, like machine learning, is very process oriented. This course provided you with a simple step-by-step approach to wrangling just about any real-world dataset. If you're new to machine learning then this approach will help get you started with data wrangling. The course also covered the AI hierarchy. Artificial intelligence is the parent container to machine learning, and machine learning is the parent container to artificial neural networks. There are four categories of models in the machine learning space. The two you'll use most often are classification and regression. They are used most often because so much of applied machine learning is supervised and these are supervised machine learning model categories. The course covered scikit-learn, a library in Python that has become the gold standard for building production-ready traditional models. Scikit-learn has three core steps once you've chosen a model and the data has been cleansed. The data is fitted to the model, predictions are made, and lastly the model is evaluated on how well it did. Choosing a model isn't an easy task. In the applied space, you'll often be crafting supervised machine learning models and that means you'll choose between classification and regression. Once our data was properly cleansed we built two different models using scikit-learn, a support vector machine and a decision tree. After some testing and model tuning the decision tree was chosen as the winner. Congratulations, you've wrangled a dataset and created a model with a high degree of accuracy that is production ready.

Course author

Mike West

Mike has Bachelor of Science degrees in Business and Psychology. He started his career as a middle school psychologist prior to moving into the information technology space. His love of computers...

Course info

Level	Beginner
Rating	★★★★★ (30)
My rating	★★★★★
Duration	1h 0m
Released	9 Aug 2018

Share course