Understanding Machine Learning with R

by Jerry Kurata

Course Overview

Course Overview

Hi, my name is Jerry Kurata and welcome to my course, Understanding Machine Learning with R. These days, machine learning is all around us, from helping doctors diagnose patients to detecting fraudulent credit card transactions. We likely encounter machine learning applications a dozen times a day and may not even realize it. It silently scans our inbox for spam emails and makes sure we see an ad on every web page for those shoes we looked at last week. This course will introduce you to machine learning and the technology behind it. You will see why companies are in such a rush to use machine learning to grow their business and increase profits. You will learn how developers and data scientists use machine learning to predict events based on data, specifically, how to format your problem to be solvable, where to get data, and how to combine that data with algorithms to create models that can predict the future. Throughout this course, we'll use the R language, the best known machine learning language. We'll utilize R and its libraries to make it easy to build and test machine learning solutions. However, you don't need prior R experience. In this course, we will learn by doing and the R code you will use will be explained in the examples. By the end of this course, you'll know the how, when, where, and why of building a machine learning solution. You will have the skills you need to transform a one-line

problem statement into a tested prediction model that solves the problem. I look forward to you joining me on this journey of Understanding Machine Learning with R from Pluralsight.

Understanding Machine Learning With R

Introduction

Hi, I'm Jerry Kurata. Welcome to the Pluralsight course on Understanding Machine Learning with R. In this course, you'll learn how to apply machine learning to solve problems that are difficult, and some might say impossible to solve with standard coding techniques. In this module, we'll provide some basic information about machine learning. This includes examples of machine learning, a definition of machine learning, and importantly, how machine learning differs from traditional programming. We'll go over the two basic types of machine learning, supervised and unsupervised. We'll see each of these types in action, which will clarify how they differ, and when each type of machine learning should be used. After that, we'll review the contents of this course, and the skills you need, and do not need, for this course. We'll finish with a brief discussion of how machine learning fits into the larger subject of data science.

What Is Machine Learning?

Machine learning is one of those technologies that is being used all around us, and we may not even realize it. For example, machine learning is used to solve problems like determining if an email is spam, how people will vote in the next election, and what products people are likely to buy. Every day you see these types of machine learning solutions in action. When you open your email, and messages are automatically scanned, classified as spam, and moved to your spam folder. Since 2008, data scientist Nate Silver has been able to predict with amazing accuracy, the results of all major US elections, before a single ballot is cast. And data giants like Google and Amazon have been able to predict which items you're looking to buy, and ensure you see ads for those items on every webpage you visit. All of these useful, and sometimes annoying and creepy behaviors, are the result of machine learning, but before we proceed, what do you think when you hear the term "machine learning? " Perhaps something like this? While the thinking robots we see in movies likely use machine learning. Defining the complex logic that lets robots think is,

unfortunately, beyond the scope of this course, so let's define machine learning in general, and the specific type of machine learning this course focuses on. For this course, we'll define machine learning to be building a model from example inputs to make data-driven predictions, versus following strictly static programming instructions. This definition points out the key feature of machine learning, namely that the system learns how to solve the problem from example data, rather than you writing specific logic. This is a significant departure from how most programming is done. In more traditional program, we carefully analyze the problem, and define the code that will produce the desired results. We use constructs such as If statements, Case statements, and controlled loops implemented with While and Until statements. Each of these has tests that we have to define, and with the changing data typical of machine learning, can be difficult to maintain. In contrast with machine learning, we don't write the control logic that produces the result. Instead, we gather the data we need, and modify its format into a form machine learning can use. We then pass this data to an algorithm. The algorithm analyzes the data, and creates a model that implements the solution to solve the problem based on the data.

Types of Machine Learning

Machine learning algorithms learn from data by utilizing one of two techniques, supervised or unsupervised machine learning. While this course focuses on supervised machine learning, it is important to understand the difference between the two techniques, and when you should use one versus the other. In supervised machine learning, the subject of this course, each row of data has fields containing feature values and the value we want the algorithm to predict. To accomplish this, we would pass a set of training data with each row of the data containing features of the price, such as size, number of bedrooms, the year the house was built, and the value we want to predict, namely the price. We pass many rows of this training data to the algorithm. The algorithm analyzes the features, and the resultant price. It determines the relationship between the features, and creates a model that is trained to predict the price of the house, based on the features. Then when the train model is presented with data for new house, it executes its logic, and accurately predicts the price of the new house. Let's contrast this to unsupervised machine learning. In unsupervised machine learning, we are looking for clusters of like data. The algorithm analyzes input data, and identifies groups of data that share the same traits. For example, let's start with a recording of a room full of people talking. We can convert this recording into data, containing values for vocal attributes, such as pitch, intonation, and inflection. We can then pass this mixed voice data to an unsupervised learning algorithm. The algorithm can analyze the voices, and create a model that clusters data, words in this case, that

have certain patterns of pitch, intonation, and other vocal features. This results in the ability to isolate an individual's voice from the mixture of voices. Since it's important to understand the correct technique to use to solve a problem, let's recap the differences between supervised and unsupervised machine learning. A primary difference is what type of problem are you trying to solve? If we're looking to predict a value like the price of a house, then you're likely looking at a supervised machine learning problem. If you're going into a set of data, and trying to define groups or clusters of like items, then this is likely an unsupervised machine learning problem. The data we have also matters. Supervised machine learning requires that we have some training data that has the value we were trying to predict. With unsupervised machine learning, we don't have the value. We are trying to figure out the values. As we saw in the examples, in supervised machine learning, we use the data with the value, to train the model, so that it can predict future values when it sees new data, like a new house. In contrast with unsupervised machine learning, we get clusters of like data from the model. And importantly, in this course, we'll cover supervised machine learning, and not unsupervised machine learning. This choice was made because many of the machine learning problems you're likely to run into are predictions, and thus solved with supervised machine learning.

Course Overview

Now, let's take a few moments, and go over the content of the course. We'll start with an overview of the machine learning work flow. This work flow will provide the framework we use to approach our problem, and ensure we do not forget critical steps in developing the solution to the problem. Once we understand the structure of the work flow, we'll spend the majority of the course diving deep into each steps of the work flow. We will do this by applying the work flow steps to a sample problem of predicting if a flight will be on time. At the end of the course, we'll have a short review on what you have learned, and where you can go from here on your journey to learn more about machine learning, but before we go further, let's review the skills and experience you should have to get the most from this course. Let's start with the skills I do not expect you to have. First of all, as the "With R" in the title implies, we're going to be doing our programming in R. However, you don't need to have any prior experience with R. This is a "learn by doing" course. I will introduce the parts of R we need as we use them. Likewise, I will show you how to use R Studio, which is an IDE that wraps the R command line interface. While you could do this entire course without R Studio, R Studio has nice features like color coded editing, auto completion, script file integration, and data browsing that make development easier, and less error prone. Finally, I do not expect you to have advanced math or statistics knowledge. While the

algorithms are based on advanced math and statistics, we'll be focused on using existing algorithms, rather than creating new algorithms. Now, let's take a look at the skills and experience you do need to have. First, you need to have some sort of programming experience. That can be in C, C#, VBE, Java, Python, or almost any other language. The language specifics don't matter, but you need to be able to understand basic programming constructs such as assignment, condition statements, loops and passing parameters to functions. Second, you need to understand and have some experience working with data and tables and lists. That is, you need to understand that in a table, a given column contains data of the same type, and that rows contain one or more columns. Also, that in lists, all of the data is the same type. Third, you need to have some basic math and statistics knowledge. The statistics needed are not much beyond means, medians, max and min, but I am assuming this level of knowledge. Likewise, nothing more than basic algebra math skills are assumed, and finally, and most importantly, you need to have a curious mind and be enthusiastic. This is a course about seeking understanding through data. You learn how with a little manipulation, you can create models that predict the future, but getting there will take a little bit of effort.

Why This Course?

I'm going to assume that if you've made it this far, it's because you have an interest in machine learning, but just in case you have some doubts, let's talk about why you want to watch the rest of this course. One obvious reason for taking this course will to be to add a skill set, namely, machine learning. As developers, we always want to constantly add skills, and learn how to do things in new and interesting ways, and just like when you learned a new programming language, you needed a set of resources like this course to get you started. This course is about machine learning, and machine learning is one of the key technologies used in the broader field of study of data science. As we see, data science itself is a blend of mathematics and statistics, software development, and expertise in the problem subject area. Machine learning is one of the most important parts of data science, and is at the intersection of software development and math and statistics. I know you have the software development expertise, so in this course, we'll blend that with using math and statistical algorithms and models. And if you can add some subject matter expertise, you can become a tech unicorn. One of the things that has made data science so exciting now is the recognition of what machine learning can do for companies. In particular, the ability to predict outcomes can directly impact a company's bottom line by defining new business opportunities, and bringing in new customers. With this new awareness has been a substantial increase in the salaries for data science professionals, especially for tech unicorn, but beyond

financial gains, I hope this course can satisfy your curiosity about machine learning and how it's applied. Once you understand the basics of machine learning, who knows where your skills can take you? Perhaps, this will be your next project. If so, I, for one, welcome our new robot overlords. Now, let's get started on this machine learning journey. Head over to the next section where we'll discuss the machine learning workflow.

Understanding the Machine Learning Workflow

Machine Learning Workflow Overview

Building a machine learning solution is a complex set of tasks. Over the years, a machine learning workflow has been developed to organize these tasks in an efficient manner and ensure that no critical steps are missed. I am Jerry Kurata. Welcome back to the Pluralsight course on understanding machine learning with R. In this module, you'll get an overview of the various tasks that comprise the machine learning workflow and tips on how to best use the workflow. But first, let's define the machine learning workflow. While there are numerous definitions to the machine learning workflow, I like this definition. As you can see, the machine learning workflow is orchestrated and repeatable. This means work has been purposely organized into defined tasks and the same workflow can be used to develop solutions to different problems. Transforms and process information. That is, before we can use data in creating solutions, it must be transformed into a format we can use for processing. And finally, the product of the workflow is a prediction solution. Implicit in this is the prediction solution must meet the performance criteria we specify. With this definition in mind, let's take a look at the steps in the machine learning workflow. Our first step is to define the question we want the solution to answer. We need to define this question in a way that guides the remaining steps in the correct direction. This requires thinking carefully about the goals we want to achieve, the data we need, and the processing we can perform. Once the question is defined, we can gather the data we need to answer the question. This can be tricky, because data are often incomplete, inaccurate, and conflicting. When we have the question defined and the data we need, we can consider which algorithm to use. This is not an easy task, as there are many algorithms available, but if we have properly defined the question, the question will help us select the proper algorithm. Once we have the algorithm selected, we need to use a subset of the data we have to train the algorithm. This training process will result in

creating a trained model that predicts results on similar data. Once we have a trained model, we need to test its accuracy on new data that was not used to train the model. This testing will generate statistics with which we can determine if the model will meet our requirements or needs to be refined. If refinement is needed, it may be necessary to go back into the workflow and alter or get more data, change algorithms, adjust training parameters, and sometimes some combination of all three. When using the machine learning workflow, there are some important things to keep in mind. First, there is an inherent hierarchy of these steps, with the earliest steps being the most important since the latter steps are dependent upon them. That is, you need to correctly define the question for which you are creating solution. Then you have to get correct data which will allow you to train your algorithm to come up with a prediction. Only when you have the model trained can you evaluate its accuracy. When moving through the workflow, it's not unusual to have to return to a previous step. For example, as you work with data, it may be apparent that you are asking the question incorrectly. And regarding data, data that you find will almost never be in the format that you need. Expect to spend a considerable amount of time locating and transforming the data into the structure that you can use. Also, within reason, more data is usually better. Remember the mathematical equations needed to model your data may be complex with strange quirks. The more corner cases you can cover with the data, the better the model will be trained and the more accurate your resulting predictions will be. Finally, try not to push a bad solution. It's easy to fall into the trap of thinking with just a few more tweaks the stars will align and your model will start performing correctly. If you find yourself in this situation, it's better to take a step and ask yourself, Do I have the right data, do I need to pre-process data, or do I even have enough information to continue? Now that we have seen the structure of the machine learning workflow, let's see how we can apply it, and use the workflow to build a solution that will predict if a flight is going to be on time.

Asking the Right Question

From Question to Solution Statement

[Autogenerated] As with any software project, having a clear definition of what we want, the machine learning solution to achieve is critical. I'm Cherry Kurata and welcome back to the plural psych course on understanding Machine Learning with our in this section will go through the first step in the workflow, asking the right question. We will convert Our general goal of predicting for

flight will be on time into a solution statement that could be used to guide our work. At first glance, it may seem that asking the correct question is easy. Don't we already know that we want to predict if a flight will be on time. But this is a case of appearances being deceiving. We need a question that will direct invalidate her work. That is the question must drive the direction of the data we gather, how we mold the data, how we interpret the solution and the criteria we use to validate the solution. What we need is more than a simple one line question. Rather, we need a solution statement to finding the in goal starting point and how we achieve the end goal. This solution statement should define the scope of the solution, including data sources. Defined target performance measurements for the solution. Determine the context of using the solution and find how the solution will be created. Let's start adding these requirements and see how the impact the solution statement. We use our original goal of predicting if a flight would be on time as the starting point based on the needs of the solution, we could make some assumptions that can help with the definition of the scope of solution. Specifically, let's make the following assumptions U. S. Flights on Lee. That's well limited data we have to process but should be abroad. No pool lights to get good data arrival and departure at U. S. Airports only again. This will limit the number of sources needed to gather the required data and will also ensure data reporting procedures are the same at the arrival and departure airports. With these assumptions, we can do a preliminary evaluation of the data sources. There are many sources of datum flight performance, but since we restricted ourselves to US flights, the best source of data is provided by the U. S. Department of Transportation, better known as the D. O. T. What incorporating these assumptions. The statement changes, too, using d o T data predicted for flight would be on time. Now let's see if their date issues we need to handle. Once we have decided to use d o T data, we could make a preliminary review of duty standards to see if we need to adjust our statement. My review of the standard show that there are no on time statistics. Instead, there are delays statistics. This means that we should restate our statement in terms of delays. This results in the statement changing to using dio __ data predict that the flight would be delayed. One important aspect of any prediction algorithm is how accurate it is for our prediction. What prediction Accuracy should we reasonably expect? We're defining the valuation in a binary fashion, Delayed or not, delay historically, 70% is a pretty reasonable starting point, so let's use that as the bottom of the acceptable range. This changes the statement to be using the OT data predict with 70% or greater accuracy. If a flight would be delayed, the context in which a prediction we used needs to be considered. Since this contrive changes on data to capture and mold, for example, we're predicting delays. What does it mean for a flight to be delayed? Doesn't mean leaving late. Does that only mean arriving late for our problem? The choose delayed as arriving late and then the question is how late is delayed here? The D O T gives us help. They have legally to find a lead as

arriving 15 minutes or more after the scheduled time. Let's incorporate this timeframe into the statement. The statement now is using Theo Tito data predict, with 70% of greater accuracy. If a flight would arrive 15 minutes or more after the scheduled arrival time, it is always useful to put down on paper how you're going to approach a problem with machine learning. I like to put down the general steps for our solution. These are use the workflow process. Data from D O T site transformed data as required. This changes our problems statement quite a bit. The statement is now used the machine learning workflow to process and transform D o T data To create a prediction model. This model must predict whether flight would arrive 15 or more minutes after the scheduled arrival time was 70% of greater accuracy. We now have a final solution statement. This statement defines the scope of our solution in the data sources. We will use performance expected, the context in which will be used and even how we'll go about creating the solution. Join me in the next section. We'll see how to use the machine learning workflow to implement a solution we have to find.

Preparing Your Data

Introduction to Data Preparation

I'm Jerry Kurata, and welcome back to the Pluralsight course on Understanding Machine Learning with R In the previous module we worked through the first step in the Machine Learning Workflow, asking the right question. In that module we developed a solution statement for our prediction. In this module we use that solution statement to guide us in getting and preparing data. And just like preparing a surface before painting, good data preparation will make the task of creating a well trained model much easier. Likewise, poor preparation can make the task painful and full of corrective and re-do work. With this in mind, let's look at what we will do in this module to get and prepare our data. We will start by discussing how to find the data we need. Once we have the data we need to load it into R and inspect and clean the data. We also need to explore the data to understand its structure and make any necessary alterations. Finally, we need to mold the data into Tidy Data, which works best with machine learning algorithms. We will do all this work in R, running inside R Studio. But right now, you may be asking yourself what is this Tidy Data we are trying to produce? Here is the definition of Tidy Data. If you have SQL experience, as you read through this definition it may strike you as very familiar. In fact, Tidy Data has attributes of what we try to achieve in database normalization. You want each variable in a

column, each row is a unique observation, and we store them in a table which is in code's third normal form. But Tidy Data is a bit more pragmatic, and focuses on presenting the data in a clean format versus underlying structures like keys. I like to think they data's tidy when it looks like it was produced by a well designed view. Getting our data to tidy form can take a lot of time. In the literature you will find estimates that 50 to 80% of the time in a Machine Learning project is spent getting, cleaning, and organizing data. At first, I didn't believe this, but experience has taught me that these numbers are accurate. Why is this?

Getting Data

Today we are swimming in seas of data, but even with all that data, the data that we need may be difficult to find, so where can we look? Google, obviously, but be careful. Not surprisingly, quality varies widely. Be extra careful when looking for data on controversial topics like global warming, vaccination side effects etc. Fake, unvalidated data abound online. Government sources are also good, they have lots of data and usually have good documentation on meaning of data, that is, the metadata. Professional groups and companies are also good sources. Professional societies provide data sets on their specialties, and companies like Twitter provide access to tweets and a lot of analysis done on the tweets to determine tenure. Financial data can be retrieved from the free APIs on Yahoo. If you walk for a company, your own IT department can be a good source of company specific data, similarly, you department likely maintains department specific data. And then sometimes when it's just all of the above. A lot of times you need data from many sources and you have to weave them together into Tidy Data. With these sources in mind, let's go ge the data we need to solve our problem of predicting if a flight will be on time. When it comes to getting data, we're lucky, because of how we structured our solution statement, the data we need to predict if a flight will be on time is readily available. The US DOT has mandated that on-time data statistics must be gathered for all airlines operating in the United States. Furthermore, this data is required to be available to the public in easily readable and analyzable formats. So to get our data, we just need to go up to the DOT On-time website. I have put this Bitly link in to make it easier for you to get this data. Let's follow the link and get the data we need. As we see, the site makes it easy to extract data. First we select the month and year of the data we want. We will select January of 2015, next we select the time period fields we want, since we already know the month and the year, let's just select the day of the week in the day of the month. Moving down to the airline section, we'll select unique carrier, carrier, tail number and flight number fields. Now let's look at the flight origin information. From this information we'll select the origin airport, origin airport sequence ID, and the origin. For the destination we select similar columns to those

in the origin, namely, the destination airport ID, destination airport sequence ID, and destination. For departure performance we'll select the departure time, departure delay 15, and departure time block. Notice that departure delay 15 was selected instead of the actual number of minutes. This is because we only care about whether the flight was delayed or not, and departure delay 15 is a binary with one signifying more than a 15-minute delay. For the arrival performance, we select the arrival time and the arrival delay 15. For cancelations and diversions, we select the cancelled and diverted fields. For flight summaries we'll select distance since perhaps this is a factor. The cause of delay, gate return and diverted airport information looks interesting, but may be more than we need right now. Now we can press the download button, and the data will be downloaded as a Zip File. Inside the Zip File will be a CSV file containing the data we requested. As we saw, this data has a lot of great fields, in fact it even tells us directly when a flight arrives more than 15 minutes after scheduled. And this is great because of data rule number one: the closer the data is to what you're predicting, the better. This may sound obvious, but it's true. Having a single field saying that the flight arrived late or not is great. In a lot of cases you do not have what you're trying to predict so cleanly defined in the source data, rather, you have to infer it by combining columns from multiple sources, and each combination brings up the potential of misinterpretation or otherwise coming to the wrong conclusion. But no matter how good the data appears at first glance, it will almost never be quite as good as we need. Or, better stated, data rule number two: data will never be in the format you need. If you've done any data processing, you know how true this is. You will always have to do something to data to reformat it. In R we want data organized into data frames with columns containing data of the same type, and rows containing columns of data. We'll see this rule in action as we start working with the data we downloaded.

Loading, Cleaning, and Inspecting Data

In this demo, we'll go over loading the data we downloaded from the DOT site into R, exploring that data to see what sort of information the DOT has provided, and cleaning the data of extraneous fields that might cause us difficulties in utilizing the data. Prior to beginning this demo I unzipped the file we downloaded and extracted the CSV file created by the DOT website. This is the R Studio environment. On the left side we have the command line interpreter, on the right side we have an environment window which will show, in realtime, the values of our various data structures. Both R and R Studio are available for most operating systems. R is free to use and R Studio comes in both free and commercial versions. Throughout this course, we'll be using the free version of R Studio. Now let's start by loading the data fromt he CSV file. We will load the

data into an R frame with columns and rows. We specify the file path, separator, with (mumbles) the file's first row's column names, and how to treat strings. You may have also noticed how R uses the less than and line symbols to indicate assignment. R also supports the equal sign for assignment, but most R programmers prefer to use the less than and line for assignment. Now that we have the data frame loaded, let's see how many rows we have. We use the end row function to get the number rows in the data frame. Wow, that's a lot of rows to process and it's really going to slow us down. To speed things us, let's restrict data to flights only between certain large airports. We do this by creating an R factor, a list containing the airport three-letter codes. Now we use a subset function to get the data for only the flights between these airports. As we see, subset is doing a select operation. Let's check the number of rows again. Now we are down to a more manageable number of rows. Remember the rule: data will never be in the format that you need? This is true even with very clean data sources like our DOT data. As is it now, there is data that needs to be removed, altered or created to make it work for our needs. Let's visually inspect the data to find any obvious errors. We'll use the head command to look at the first two rows. As we can see, we have a lot of columns. Most are as expected, but notice the last column named X, with the values of N/A, that's suspicious. The data exported from the DOT site did not contain an X column. The X column was created as part of importing the CSV, and the N/A means the data was not available. It looks like that column can be dropped, but to be sure, let's check the end of the data frame with the tail function. It definitely appears that the X column has no value, so let's remove this column. In R a column can be removed by setting its value to null. Let's check the data again. Good, the X column is now gone. In general we want to eliminate any columns that we do not need. In particular, we want to eliminate columns that don't have any values or are full of N/A values, but we also want to eliminate columns that are duplicates or provide the same information in a different format. We can visually inspect columns to see if they're really the same, but visual inspection is error prone and does not deal with the critical issue of correlated columns. Correlated columns state the same information in different ways, such as an ID and the text value for the ID. These correlated columns do not add information about how the data causes changes in the results. Worse, they can amplify a bias because some algorithms naively treat every column as being independent and just as important. For example, imagine what would happen if we predicted house prices, and the size of the house was in both square feet and square meters. Size would result in being twice as important as it should. Let's look at some data and see if we can locate some possibly correlated fields. In looking at the data, origin airport sequence ID and origin airport seem to change together, as does destination sequence ID and destination airport ID. I'm not sure if we use these fields, but if they are correlated, we only need one from each pair. We can verify the correlation of values using the

core function. The closer the values return to R to one, the more correlated. Wow, a perfect one. So origin airport sequence ID and origin airport ID are moving in lockstep. Now let's check destination sequence ID and destination airport ID. Another perfect one, so destination airport sequence ID and destination airport ID are moving in lockstep. Let's drop the columns: origin airport sequence ID and destination airport sequence ID since they're not providing any new information. Using the core function works for checking correlation with numeric columns, but it will not work with string columns. I'm pretty sure I saw some potential duplicate string values, let's check the data. Notice that unique carrier and carrier also look related, actually, they look identical. We can see if they're identical by filtering the rows to those where unique carrier and carrier are different. R makes this easy, no loops to write. All the iteration is done for us. We know the row filter criteria and get a list of all the rows where carrier is not equal to unique carrier. Notice the comma right before the closing brace, R can filter a data frame based on a list of rows and a the list of columns. The comma separates the filter for the list of rows from the filter for the list of columns. Since we only want to filter by row values, putting a comma with nothing following tells R to filter only by rows. Now we can check the number of mismatched rows within row. The zero mismatch means that carrier and unique carrier are the same for all rows, so let's drop unique carrier. Let's see what the data frame looks like now. Looks like we have cleaned up the data frame pretty well, now we can move on to more advanced changes to the data.

Molding Data

We have done some good clean up by removing some columns we don't need, now we need to mold the data into data we can use for training. This involves reviewing the data values within the rows and making a decision about whether to keep the row, ensuring that the data types of the columns are what we need. And, if required, creating new columns based on existing data. Let's recall that our solution statement said we are trying to find flights that arrive 15 or more minutes late, that is those flights which arrive delay 15 is set to one. Since arrived delay 15 is the result for which we are predicting, we cannot process any rows for which the value is not zero for false, or one for true. Unfortunately, if the column does not have a value the CSV import process will set the value in the column to N/A. To make matter worse, the DOT export process will sometimes set columns to an empty string when it does not have a value, so we need to drop all rows for which arrive at delay 15 is set to N/A, or an empty string. We should also do the same filtering for departure delay 15 since we might want to use that field in our prediction. We can easily filter the rows for N/A or empty values in either column. Notice that this code stores the filtered rows in a new data frame named on-time data. I like to create new data frames when I reach major

checkpoints like starting to drop rows. This lets me easily go back to the before state. Now that we've filtered the data, let's compare the number of rows in the new and old data frames. The difference in the number of rows show that there were some rows that we needed to drop. Another basic way to mold data is to change columns into a more useful data type. Let's use the Environment tab in our studio to look at data types of some fields. We some fields like distance that should be numeric or strings. This can make this field less useful because algorithms may not see this as a continues set of numbers, but rather as discreet set of strings. So let's change the format from string to numeric. Changing the format of a column and all the data for the rows in that column is hard in some languages, but simple in R. We just type in a simple command, and we change all the rows in a column in one line. We can also convert other strings to integers. Now let's take a look at the arrival and departure delay 15 fields, sometimes algorithms perform better when you change columns into factors. Factors are like numerated types in other languages. Changing the column to a factor allows the algorithm to use counts of the number of times a column was a certain discrete value, like the number of times a value was one, versus the number of times a value was zero. Let's also change some other columns to factors. Notice that for each factor column, the number of distinct values is listed. Also notice that this number of distinct values is relatively low for the 20, 000 plus rows. If this number of values was not low, then the field probably should not have been changed to a factor. We can easily change the data type back by using R's as numeric and as string functions. Now, reviewing the data, looks like we have it in a form we can use. However, before we can use this data to train our production, we need to ensure the distribution of the data will allow us to train our prediction. This is driven by data rule number three: accurately predicting rare events is difficult. The basic issue with our event is they're rare, so the probability of you having one in your data is low, and the probability of accurately training the algorithm to recognize this event is also low. We only need a few percentage of the event to be able to train, but more is better. To ensure we have a reasonable chance of correctly predicting if the flight will be delayed, let's check our newly clean data. Remember, we're using arrival delay 15 equals 1 is true, and arrival delay 15 equals zero is false. Let's see how many delayed versus non-delayed arrivals occur in the data. We use the t. apply function to see how many times arrival delay 15 is true and how many times it is false. We definitely have a number of delays so that is good, let's compute the prevalence of delayed flights in the data. We have about 20% delayed flights, which means we can reasonably train our prediction model using the data we have. So, congratulations, we have our training data. But doesn't anyone remember how we got here? This brings up data rule number four: track how you manipulate data. A lot of the manipulation of data you would do in Machine Learning is trial and error. When you manipulate data it's easy to change meaning of data, this can be done

intentionally, but also is done unintentionally. By keeping track of the manipulations of data you can always reproduce your data at any step, and see where you went wrong. With R and R Studio the easiest way to keep track is to create an R Script File. You can copy commands from the command line to the R Script File, and then save them to disk. Like all coding, you should ensure that you have some sort of source control to track changes and allow you to restore things that you might've accidentally deleted or altered. Let's summarize how we prepared our data. We first discussed various data sources and went to the DOT site to select the data that we are using. Once we had the data selected we downloaded it as a Zip File that contained a CSV file. Using R we loaded the contents of the CSV file into a data frame. From there we cleaned the data by removing extraneous fields, then we molded the data by removing rows we could not use and changing the data types in columns to what we needed. We also verified that we could use the data for prediction. And along the way we discussed data rules that guided our work. In the next module we'll select the algorithm that will be trained by our data to produce our prediction model.

Selecting Your Algorithm

Introduction to Algorithm Selection

I'm Jerry Kurata and welcome back to the Pluralsight course of Understanding Machine Learning with R. In the previous modules, we went through the workflow steps of asking the right question and preparing our data. In this module, we'll go through the process of selecting algorithms. We'll use our problem knowledge to help us decide the algorithm to use. More specifically, in this module, we will discuss the role of the algorithm and the machine learning process. We will then select our initial algorithm by utilizing the requirements identified in the solution statement, as a guide, and discuss at a high level, the characteristics of some specific algorithms. Finally, we will select one algorithm to be our initial algorithm. Notice that I said initial algorithm. As mentioned previously, in machine learning, we often cycle through the workflow. In our search to find the best solution, it is likely we will need to train and evaluate multiple algorithms. It is important that we understand the role of the algorithm in the machine learning process. So let's review how the algorithm is involved in this process. One could say that the algorithm is the engine that drives the entire process and we will see this illustrated as we walk through the process. The algorithm's code is accessed by a training function often named train. When the function is called, training

data is passed. For our prediction problem, this data contains examples of the value we are trying to predict and features that enable predicting the value. The algorithm executes its logic and processes the training data. Utilizing the training data, the algorithm produces a trained model. This model contains code with which it can evaluate data and parameters created during the training process. These parameters control the code's logic and allow the code to evaluate data. The model contains a prediction function, often named predict. When this function is called, real data is passed. The model then uses its code and parameters to evaluate with the data and produce results. Now that we understand the role of the algorithm, let's select our initial algorithm.

Selecting an Initial Algorithm

Since machine learning can be applied to a lot of problems, there are a lot of machine learning algorithms available. A quick check of one of the standard libraries show that there are over 50, and more being created. So how do we decide which algorithm to use? To decide which algorithms would be suitable, we can compare the algorithms on a number of factors. Not surprisingly, data scientists have different opinions on which factors are important in selecting algorithms. The decision factors presented here represent one possible set of factors. As you gain experience, you will likely develop your own set of factors and decision criteria. For our example, we're going to use these decision factors. We will evaluate algorithms based on what type of learning they support, result type the algorithm produces, the complexity of the algorithm, and whether the algorithm is basic or enhanced. We will use our solution statement and knowledge of the workflow to help guide us in evaluating these factors. Each algorithm has a set of problems it works best in. One way to divide them is to look at the type of learning they support. With that in mind, let's go back to the solution statement and see what guidance it offers. Reading the statement, we see our solution is about prediction. Prediction means supervised machine learning, so we can eliminate all algorithms that do not support supervised machine learning. That reduces the number of choices but we still have a lot. Let's see what else we can do. Let's see how the result type can help. Prediction results can be divided into two categories: Regression and classification. Regression means a continuous set of values. Back in our example of determining the price of a house, we used the features of the house, size, number of bedrooms, et cetera. We put those features into an equation and produced a price. Any change in any feature resulted in a direct change in the price. In contrast, classification problems have a discrete set of values, such as small, medium, and large, one to 100, 101 to 200, 201 to 300, and more importantly for us, true or false. Changes in a feature value may or may not change the classification. So what

type of problem do we have? Again, the solution statement comes to our aid. From the solution statement, we see that the algorithm must predict whether a flight would arrive 15 or more minutes after the scheduled arrival time. Remember that we're using ARR_DEL15 to define if a flight is delayed. And ARR_DEL15 is a binary outcome. Since we are predicting a binary outcome, delayed or not, we can eliminate any algorithms that do not support classification in general, and binary classification in particular. That knocked out some more, but perhaps not as many as we might have thought. That's because many algorithms support both regression and classification. We're down to 20 algorithms but that's still too many. Since this is our initial algorithm, let's keep it simple. Let's also eliminate ensemble algorithms. These are special container algorithms that contain multiple algorithms under a single interface. These algorithms are most often used when we need to tune the model to increase performance, and we are still trying to do our initial training. Also, ensemble algorithms tend to be complex and difficult to diagnose and troubleshoot when the results are going awry. We still have a few algorithms left. We can divide these into two groups, enhanced and basic. Enhanced algorithms are variations on the basic algorithms, and they have been enhanced to perform better, and/or add additional functionality. As a result, they are more complex to understand and use properly. Since this is our initial algorithm, let's stick to basic algorithms. These have the advantage of being less complex and therefore, easier to understand. For our problem, let's look at these three basic algorithms as possibilities for use in our initial training and evaluation. We'll look at Naive Bayes, Logistic Regression, and Decision Trees Each of these algorithms are classic machine learning algorithms. And understanding a little bit about each one can greatly help understanding more complex algorithms that use these algorithms as building blocks. The Naive Bayes algorithm is based on Bayes' theorem. This theorem calculates a probability of the flight being delayed by using the likelihood of delay based on previous data combined with the probability of delay based on nearby feature values. It makes a naive assumption that all the features we pass in are independent of each other and equally impact the results. This assumption that every feature is independent of each other allows for fast convergence and therefore, requires a smaller amount of data to train. The Logistic Regression algorithm has a somewhat confusing name. In statistics, regression often implies continuous values, but Logistic Regression returns a binary result. The algorithm measures the relationship of each feature and weights them based on their impact on the result. The resultant value is mapped against a curve with two values, one and zero, which in our case is equivalent to delayed and not delayed. The Decision Tree algorithm can be nicely visualized. The algorithm uses a binary tree structure with each node making a decision based on the value of the feature. At each node, the feature value causes us to go down one path or another. A lot of data may be required to find the value which defines taking one branch or another. As we see, the Decision Tree has the

advantage of having tools available to produce a picture of the tree. This makes it easy for us to follow along and visualize how the trained model works. From these three, let's select Logistic Regression as the initial algorithm for training and evaluation. Logistic Regression has a number of characteristics that make it useful as an initial algorithm. First, it is simple to understand. Once the value hits a certain threshold, the result is true, delayed in our case. Below that threshold, the result is false, not delayed. Second, the algorithm is fast. Training time of some of the more complex algorithms can be 100 times or more longer. In a machine learning, you often have repeated cycles of running the algorithm, evaluating the results, and tweaking the parameters. Having a fast algorithm makes these cycles much quicker. Finally, the algorithm is stable. With some algorithms, as you work with the data, small changes in training parameters can cause the results to vary widely, or worse, the algorithm fails and returns nonsensical results. Both of these can send you on a long and frustrating debugging cycle. Logistic Regression does not exhibit these behaviors. Let's summarize our algorithm selection process. First, we need to acknowledge that there are a lot of algorithms available and more being created every day. With all these algorithms available, we need to select the one that will be initially trained with our data into a model. To select this algorithm, we can use the features of our problem that we captured when we defined our solution statement. From the solution statement, we know we have a supervised machine learning problem, which removes all of the unsupervised machine learning algorithms We also want binary results, not continuous results, which further reduces the number of viable algorithms. Since this is our first pass at algorithm training and evaluation, we also eliminated all ensemble algorithms, which combine multiple algorithms. We also decided that for initial training, we only wanted basic versions of the algorithms. After evaluating the algorithms, we selected Logistic Regression as our initial algorithm because it is simple, fast, and stable. In the next section, we will train the Logistic Regression algorithm with our data and see how well it can predict if a flight is going to be delayed.

Training the Model

Introduction to Training

Hi, I'm Jerry Kurata. Welcome back to the Pluralsight course Understanding Machine Learning with R. In previous modules, we covered the workflow steps of asking the right question, where we defined our solution statement, preparing data, in which we obtained raw data and

transformed it into data we could use for training, and selecting the algorithm, where we selected the initial algorithm we will train and evaluate. In this module, we'll put the pieces together and train the algorithm we selected with the data we prepared. When we are done with this training process, we will have a model that can predict if a flight will be delayed. In this module, we'll get a detailed understanding of the training process, introduce the Caret package which can make the training and evaluation process easier, then go back to R and train our algorithm with our DOT delay data and produce a train model. A good definition of machine learning training is letting specific data teach a Machine Learning algorithm to create a specific forecast model. Notice the use of the term "specific. " Data drives the training. If data changes over time or new data is used, in many cases we need to go back and retrain. And we want to retrain if the data changes. Retraining will ensure that our model can take advantage of the new data to make better predictions and also verify the algorithm can still create a high-performance model with the new data.

The Training Process

Let's review the supervised machine learning training tasks we need to perform. First, we will split the prepared DOT data into two sets of data, one for training and one for testing the train model. Typically, about 70% of the data goes into the training set, and about 30% goes into the testing set. Then we will train the algorithm with the training data. We'll hold the test data aside for evaluation. This training process produces a train model with parameters based on the training data. But I bet you're thinking what about the test data? We'll only use the training data to train. What are we going to do with the test data? You're correct in questioning this. Do you have any thoughts on why we split the data and then only used one of the sets of data? If we used all of the data to train, the produced model we trained to both training and test data sets. Since data drives the training model, this additional data would impact the workings of the model. Also, when we evaluate the model with the test data, it would likely falsely perform well since the model has seen the data before and knows about the test data's biases. But the model could perform poorly when used with real-world data that had a different set of biases. To prevent us from being lulled into thinking that the model performs better than it really does, we hold the test data aside and only train with the training data. In the next module, we'll use this test data to evaluate the training model. But for now, let's focus on the first two steps, splitting the data and training the model. But before we get started training, we need to select the features, i. e. columns that we will train with. Since the data drives how the model is trained, we want to ensure that we train with only the minimum number of features. This makes the training go faster and

often more accurately. We can look at the data and reason about the context of the problem to make our selection. For airline flights, we see there are some columns that stand out as good candidates for explaining why a flight would be delayed. Origin and destination make sense since airports have certain features such as a history of bad weather and few runways that make them more likely to have delays. Day of the week is selected, because as anyone who travels knows, there are certain days of the week that are worse for traveling. This is often because the high volume of passengers in departing and arrival flights create a perfect storm for delays. Carriers have, how shall I put this, certain patterns of behaviors that affect delays. These include overbooking, a history of late departure arrival, and other scheduling issues. We also selected departure time block. Departure time can affect delays because traffic, weather, and passenger loads vary throughout the day. And delayed departure means a late arrival. Since leaving at 7:30 is not much different than leaving at 7:31, but may cause the time to be treated independently by the algorithm, we use departure time block. Departure time block groups departures into one hour or longer segments such as 6 a. m. to 7 a. m., 7 a. m. to 8 a. m., et cetera. And finally, we need to include arrival delay 15. Since this is supervised machine learning, we need to include what we are predicting as part of the training data.

Training with R

Now that we understand in detail how to do the training, we're just about ready to go. But before we start coding, let me ask you a question. Do you usually write all your code from scratch? I bet not. I bet most of the time, you're like me and will try to find a library that can do some of the repetitive tasks for you. In R, there is such a library for handling machine learning training and evaluation tasks. The library is a package called Caret. Let's take a look at the Caret package and see how it can lessen our work load. The name Caret has nothing to do with vegetables or diamond rings. Rather, it's from the package's purpose, which is classification and regression training. The Caret package is a tool set that makes it much easier to perform training tasks such as splitting the data into training and test sets, pre-processing data, selecting which features of the data are most important, and tuning the model for better performance. In addition to these features, it provides a common interface for accessing algorithms. Now that we know about Caret, let's get into R and split the data and train the model. We have to load the Caret package before we can use it. To do this, we use the install. packages and library functions. Install. packages needs to only be read once. It will download the Caret package from the online repository and install it in our local installation of R. Once a package is installed, we can use the library function to load the package into the current R session. Remember that we need to use

the library function to load the package whenever we start a new R session. Now we are ready to split the data into training and testing data. But before we do that, we need to set the seed we'll use for random number generation. If you're not familiar with setting the seed, let me explain. Inside the training algorithm will likely exist code that uses random numbers. Random number generators in computers generate random numbers in a sequence based on a starting point called a seed number. Code within R or the operating system will set the seed number for us if we don't set it. For example, if we generate integers between 1 and 10, the sequence might start with 1, 9, 5, and 8. If we do not set the seed, the next time we generate the numbers, they might be 2, 9, 4, and 7. This difference in sequence can change our training results. We want to ensure if there's a change in result between training runs, it's because of something we change, not because R or some other code decided to use a different set of random numbers. We can get the same random numbers by setting the seed. This will ensure that when we run our code multiple times, it generates random numbers in the same sequence. As we see, we use the set. seed function to set the seed value. The actual value we pass to the function does not matter, just that we pass the same value each run. With that understanding, let's set the seed for our training run. From our previous review of the data, we know that we only need a few of the columns in the data frame to train the algorithm. Specifically, we need arrival delay 15, the result, day of the week, carrier, destination, origin, and departure time block as features used to predict the result. We stored these column names in a vector. This makes it easy to change columns if we decide we need to add or remove columns. Then we create a subset of the on-time data that contains only these columns. Notice that I again created a new data frame so I could easily get back to the data before the columns were filtered. Next, we need to split the data into training and testing data. If you think about this for a second, this can be kind of tricky. We need to ensure that we have the correct percentage of rows in the training and test data frames. In addition, we need to ensure that the distribution of true and false values is the same in both data frames. That is, for arrival delay 15, the ratio of true and false values is the same in the training and test data frame. We'll let Caret's Create Data Partition function deal with this issue. We'll tell the function which feature it should ensure the percentage of its values are the same in the training and test data frames, namely arrival delay 15. Next we'll tell the function what proportion of the data we want in the training set. 70% is 0. 7, so that's what we'll pass. Finally we'll tell the function not to create a list we want one item per row. We run the function, and it quickly creates a column vector containing a list of column indices. Let's look at these indices with the head function. If we look at the first ten rows, we see create data has populated the column with the selected row indices. Notice how the row indices skip around certain rows. This is because the rows were selected to ensure that the proportion of arrival delay 15 with the value true is the same in this 70% of the data as it is in

the original data. We use as row vector as the indices to select the rows that make up the training data by simply specifying the row indices as a list of rows to select. Pretty easy, huh? But wait, that took care of the 70% of data we want in the training set. R also makes it easy to select the 30% that are in the test set. We simply put the minus in front of the vector, and R will only select the rows whose indices are not in the vector. That is nice. So in five lines, we have partitioned our data into training and test data frames. But before we train with this data, we should verify that we really have a 70/30 split of data. We can do this by verifying the proportion of data in each data frame. First we check the training data frame. It should be near 70%. That's very close to 70%. Then we can check the testing data frame which should be near 30%. Very close to 30%. So on to training. To train the model, we use the Caret Train function. In the first parameter, we passed the names of the columns whose value we are trying to predict, arrival delay 15 in a list of columns used to predict that value. We separate the value and its predictors with the tilde character. Since we filtered the columns to only include those we needed to predict the value and the value itself, we can put a period after the tilde. This says all columns except the one on the left side of the tilde are used to predict the value. Next we'll specify the data we'll use in training, which is our training data frame. Then we specify the method to use. The Caret documentation contains a list of abbreviations. We are training using logistic regression, and this is a special case of generalized linear regression. So we pass GLM to ensure we use the logistics regression algorithm, we also need to pass a secondary parameter called family set to binomial. We can now press the Return key, and R starts training. When the training completes, we'll have a trained model that we can use to predict arrival delays. And that's all we had to do to split our data and train the model. R and the Caret package made this really easy. We just had to enter a few commands. You might be thinking we should take a look at the train model and see how it works. So let's do that. Unfortunately, as you see, the data is not too usable, unless you happen to be the person that wrote the code for the algorithm or the model. The best we can do is get some statistics on the model. But don't worry, we'll see much better information when we evaluate the model in the next module. Let's briefly summarize what we did in this module. We started with an overview of the training process. This provided us with an understanding of the tasks we needed to do to split the data into training and test data and create a trained model. We then went into R and split the data and trained our model. To help us with some of these tasks, we used the Caret package. This package contains functions that made it easy to split the data into training and test sets and then train the model. Join me in the next module, where we will evaluate the performance of this newly-trained model.

Testing Your Model's Accuracy

Introduction to Evaluating the Model

Hello I'm Jerry Kurata. Welcome back to the Pluralsight course on Understanding Machine Learning with R. In previous modules we went through the workflow steps of defining the solution statement, getting our data, and selecting an initial algorithm. In the last module we produced a model trained with our training data. In this module, we will evaluate this trained model and see how well it can predict if a flight will be delayed. In this module we will evaluate our trained model by using a set of test data. Remember this test data was not used to train the model so it should give us an accurate estimate of the real world performance of our model. This evaluation will provide us with a series of results that we can use to decide if the performance of the model is acceptable. The results will also give us some ideas on how we might revise the workflow steps to improve performance. Throughout the evaluation process we need to keep in mind that statistics only provide us with data. We are the ones that interpret this data and determine if it is good or bad. And, we need to define good and bad in the context of how we will use our model. But, enough theory for now, let's go back to R and evaluate the model.

Evaluating the Model with R

Remember we split our data into training and test data frames. 70% of the data went into the training data frame and 30% into the test data frame. Now, we will use the test data frame to evaluate the predictive capabilities of the model. We'll do this by calling the predict function and passing to the function the trained model and the test data frame. The function will use the trained model to predict delays from the test data frame and return an object containing the predictions. Now that we have the prediction we can evaluate how well the model predicts flight delays. To make it easier to evaluate the model's performance we use Caret's interestingly named "Confusion Matrix Function". We pass to the function the prediction object and the column we are predicting. The Confusion Matrix provides us with groups of performance statistics about the model's predictive capabilities. The first set of statistics is a matrix that compares the predictive and actual values for delays on the prediction and reference axes. Let's label these values with letters so we can use them in equations. A is the number of flights in the test data that were not delayed when the model predicted they would not be delayed. B is the number of flights in the

test data that were delayed when the model predicted it would not be delayed. C is the number of flights in the test data that were not delayed when the model predicted it would be delayed. D is the number of flights in the test data that were delayed when the model predicted it would be delayed. Below the matrix is a set of statistics derived from A, B, C, and D. These statistics quantify the capabilities of the model in terms of standard measurements. Let's review a few key statistics about our model. Accuracy is the ratio of the model prediction to the correct answer. That is predicting that a flight will be delayed when it is delayed, and predicting that a flight will not be delayed when it is not delayed. Mathematically this is A plus D divided by the number of test rows. Our model seems pretty good on accuracy achieving 0. 7997, which is better than the target goal. But, before we celebrate let's take a look at some other statistics. Sensitivity is the measure of how the model predicts no delay when there is no delay. This A divided by A plus C, and at 0. 9970 it is excellent. Specificity is the measure of the model's ability to predict delay when there is delay. Specificity is D divided by B plus D. Unfortunately, here the model is poor at 0. 016. Notice this is mostly due to D, predicted delay arrivals at 31 being so much smaller than the actual number of delay arrivals, B plus D 1938. We definitely need to increase the model's ability to predict arrival delays. A few more interesting statistics are the predictive values. The positive predictive value which predicts when there will be no delay is 0. 801 which is pretty good. However, the negative predictive value, which predicts when there will be a delay is 0. 5741 not so good. So, even though we hit our target for accuracy we still have some work to do.

Improving Model Performance

Before we make any attempts to improve performance let's do some quick analysis. Our problem is that we have too many flights in the test data that are delayed that are not being predicted as delayed. We need to determine a way to increase the accuracy of our predictions. What are our options for improving the accuracy of our predictions? Remember when we went over the workflow we discussed these options at a high level. We could go back to the prepared data and get additional data, or we could select a new algorithm that worked better with our data, or retrain the model with different parameters that could improve the performance. This updating and performing previous workflow steps is very common in machine learning. Let's see how these options translate into changes we can make in our problem's workflow. For our solution revisiting workflow steps would map to the following options. The simplest form of getting additional data means adding additional predictor columns from our existing data. The most obvious column and one that will definitely improve performace is Departure Delay 15, but think about that for a second. While it is true if your flight leaves late it is likely to arrive late, is this a practical feature

for what we want to predict? You can only detect if a flight is going to leave late a little bit before it departs, so how useful is that? It is likely that using delay departure as a feature for predicting if a flight is going to have delayed arrival is not useful in that many usages of the model. The second option would be trying to adjust the training settings, but I spent a while doing this and the improvements were pretty subtle. Notice that the problem is with unpredicted delay. Are there algorithms that could do better in this area? This is an area where ensemble algorithms can be useful. Let's try a Random Forest. Random Forest is an ensemble algorithm that creates multiple trees and uses a technique called Bagging to improve predictive performance. We can determine if the Random Forest has improved performance by checking the number of unpredicted delays and the resulting specificity value. We need to first load the Random Forest library. Then we need to train a model with the Random Forest algorithm. We'll only use a couple of the algorithm's many options. And, to make things interesting let's not use Caret's train method, rather let's see the raw Random Forest constructor so we can see what Caret highlights and standardizes. The train data filter minus one is R's way of creating a new data frame which excludes the first column which in our data happens to be Arrival Delay 15. The next parameter is the column to predict for, namely Arrival Delay 15. The last two parameters tell the Random Forest algorithm that we care about proximity and that the algorithm should use the importance of each feature in creating the trees. We press the return key and the algorithm processes the data to create a trained model. And, we wait for a long time. I sped up the screen. The actual training time was around eight minutes versus the five seconds for the Logistics Regression. As we discussed earlier trees can take some time to train and the multiple trees used by this ensemble algorithm take even longer. Now, let's use the new Random Forest model to predict which flights in the test data will be delayed. Finally, let's evaluate the performance of the model by looking at the Confusion Matrix. Looking at the matrix we see there are 1938 delays in the test data. With the Random Foest algorithm we predicted 229 of these which is much better than the 31 we predicted with Logistics Regression. And, this difference boosted the specificty from 0. 016 to 0. 1182, over seven times, but there is still room for more improvement. So, what can we try now? We could try to further improve the performance by adjusting some of the parameters through Random Forest, looking into doing some pre-processing of the data, or perhaps trying other algorithms. After each of these changes we will repeat the training and evaluating the results as before. But perhaps, there is another option rethinking about the problem. What do we know about flights? As we mentioned earlier flights that depart late will almost always arrive late. So, what causes a flight departure to be delayed? Do we know? Perhaps a little subject knowledge is required. If we read through the information on the DOT site we see that weather is a significant cause of delays. So maybe we should add weather data. Where would we go in the machine running workflow to

determine which weather data to add and how to add that data? Why, we are back at preparing data. All set for another cycle of the workflow with the addition of weather data. We would first clean and mold the weather and flight data. Then we would follow on to the subsequent workflow steps of selecting the algorithm, training it, and finally re-evaluating the performance of the new model with the new data. But because this is an introductory course and we have done these steps before let's leave getting and preparing weather data and incorporating it into flight data to another time, or perhaps as an exercise for you to try. You have seen these steps before so you know what to do and how to do it. But, for right now let's recap performance improvement. We have seen that our efforts are making improvements in the performance of our prediction model. And, for each cycle of changing the data, settings, and/or algorithms we'll hopefully see more performance improvements. But, these improvements can take some effort to yield results, so we need to weigh these efforts against the results and make a decision on how much performance is good enough. It seems like sometimes the hardest thing with Machine Learning is determining when to stop cycling through the machine learning workflow in search of better performance. In this module we evaluated the performance of our trained model against test data. We did this by using the predict function to generate a prediction of how well the model predicts if a flight will be delayed. We use the Confusion Matrix function to review the prediction performance of the model and inform us of areas that needed improvement. We chose to improve the performance by using a different algorithm, Random Forest. Changing the algorithm improved the performance substantially, but as always there was room for further improvement. Finally, we revisited the problem and used some domain expertise and reasoned that weather data would be useful in a future effort to improve the prediction of whether a flight would be delayed. In the next module we will summarize this course and go over some additional information that will be helpful in your machine learning journey. See you there.

Summary

The Journey so Far

As we complete this course, let's spend a few minutes recapping our machine learning journey so far, and going over some suggestions for future learning. We started this course describing some of the ways machine learning is working in our lives today. We saw how machine learning affects our lives in many subtle and not so subtle ways. A key point pointed out early on is that machine

learning is data driven with data defining the logic used to solve the problem. We discussed the machine learning workflow, which provides a defined path for creating machine learning solutions. Throughout the remainder the course, this workflow guided our work. Let's review how we used the workflow. We started by defining the question we are trying to answer. We used requirements and knowledge to enhance and transform our simple question of whether our flight would arrive on time. The end result was the solution statement defining where we would get our data, how we would use that data to create a solution, and the performance we expected from our solution. Following the workflow, we retrieved data from the DOT website. Using R, we cleaned this data. We removed columns that were either created as a side effect of the import process or contained correlated values that might cause the training to be misled. We molded the data by transforming the data types of columns into data types that work best for our machine learning. We also eliminated rows that we could not use because the value we wanted to predict was either missing or set to NA for not available. Once we had our data, we selected an algorithm to use. From the large selection of available algorithms, we made our initial algorithm selection by matching learning type they supported to our problem, ensuring the result type produced was what we needed, reviewing the complexity of the implementation, and choosing basic over enhanced versions of the algorithm. After we selected the algorithm, we used it with our data to create a trained model. We did this all in R, utilizing the care package. This involves splitting the data into training and test data. We decide to do use 70% of the data for training and 30% was reserved for testing. We passed this training data to the algorithm to create our trained model. We evaluated this trained model with R. We created a prediction with the test data. We used the Caret to Confusion Matrix function to evaluate the prediction. The predictive power of the model was not the best, so we reviewed our options for increasing predictive capabilities. In the end, we switched to a different algorithm, Random Forest, and trained it with the training data. When we tested it, the Random Forest algorithm showed significant improvement, but as always, there was still room for more improvement. We could have trained other algorithms or changed parameters, but instead, we thought about the data and did some investigation on the DOT site. This Works suggested that adding weather data could be a useful change. The addition of weather data and retraining and evaluating were left as a future exercise.

Guides for Your Journey

There are a number of resources available to you to help you continue on your machine learning journey. These Pluralsight courses can be very helpful on your journey. R Programming Fundamentals. We used R and learned some of its features and function. R Programming

Fundamentals can help you take the next step and dig into R to get the most from the language. RStudio: Getting Started. We used RStudio as RID, but we only used a small subset of its capabilities. RStudio: Getting Started can help you take better advantage of RStudio as made to your R development. Other sources that can be useful include these online resources. CRAN. CRAN is the abbreviation for the Comprehensive R Archive Network. From the site, you can download R and many of its packages, such as Caret. In fact, you may have noticed that when we used Install Package to download Caret, it went to this site. In addition to software, the documentation for both R and the packages are also available on the CRAN site. UCI Machine Learning Repository. This site contains data sets that can be readily used to learn about machine learning. The data is relatively clean but still has enough quirks to make it interesting. Also, many of these data sets are classics in machine learning and referenced in many of the articles you will find online.

The Journey from Here

You now understand some of the basics of machine learning and quite a bit about supervised machine learning. But as the Data Science diagram shows, there is a lot more to know. You started this journey with good development skills. Throughout this course, we focused on adding machine learning skills. We have done this through both code and understanding a tiny bit of math and statistics, and beyond machine learning, especially in evaluating the model, we tried throughout to add subject matter expertise in the area of airline flights. So hopefully you moved a little closer to becoming a tech unicorn, and who knows where this new knowledge will take you. Maybe here? No matter where it takes you, I hope you will respect your new skills. Just remember the words of Uncle Ben in Spider-man: "Build something cool and amazing, but not too creepy. " Happy programming.

Course author

Jerry Kurata

Jerry has Bachelor of Science degrees in Geology and Physics. His plans to work in the oil exploration industry were sidetracked when he discovered he preferred to work with computers on simulation...

Course info

Level	Beginner
Rating	★★★★½ (301)
My rating	★★★★★
Duration	1h 25m
Released	17 Feb 2016

Share course