

Module 2: Framework for Machine Learning and Logistic Regression

Table of Contents

Module 2: Framework for Machine Learning and Logistic Regression.....	1
Module 2 Overview.....	2
Module 2 Introduction.....	2
Lesson 2-1: Machine Learning Overview.....	18
Lesson 2-1.1 Inference, Prediction, and Experimentation.....	18
Lesson 2-1.2 Categories of ML Models, Part 1, Types of Data and Terms.....	30
Lesson 2-1.3 Categories of ML Models, Part 2, Categories of Algos.....	39
Lesson 2-1.4 How Machine Learning Works in General	51
Lesson 2-1.5 Evaluating ML Model Quality.....	60
Lesson 2-2: Business Problem.....	70
Lesson 2-2.1 Introduce the Business Problem to Solve	70
Lesson 2-2.2 Introduce the Data.....	75
Lesson 2-3: Logistic Regression.....	83
Lesson 2-3.1 Introduction to Logistic Regression	83
Lesson 2-3.2 Logistic Regression Hands on - One Variable (Part1).....	93
Lesson 2-3.3 Logistic Regression Hands on - One Variable (Part2).....	103
Lesson 2-3.4 Logistic Regression Hands on - One Variable (Part3).....	117
Lesson 2-3.5 Logistic Regression Hands on - Multiple Variables	131
Module 2 Review	140
Module 2 Conclusion	140

Module 2 Overview

Module 2 Introduction

Data Today:
Volume, velocity, variety of data
is increasing.

(Google Ngram Viewer, 2019)

The slide features a blue background with a red 'I' logo in the top right corner. On the left, there is a screenshot of the Google Books Ngram Viewer showing a sharp increase in the frequency of the term 'big data' starting around 2000. On the right, a man in a dark suit and pink shirt is gesturing with his right hand while speaking.

In this module, we'll do two things. First, we'll introduce you to the world of machine learning. Second, we'll examine a classic classification algorithm logistic regression. More and more managers find themselves drowning in data. It's not just the everyday transactions and sales data that stalks them. It's the shipping, the logistic data, customer loyalty data, scanners and IoT data, that's machines talking to other machines, supplier data, and employee, and HR data. The volume, velocity, variety of data is only increasing if the use of the term big data is any indication. It's not just hyperbole to say that we have more data available to analyze now than we ever have before.



Commentators state that over 2.5 quintillion bytes of data are created every single day. That more than a megabyte of data is created every second for every person on Earth. That means that in the time you've been sitting here listening to this video so far, more data has been created just from you than most personal computers could handle in the year 2000. Now think of your own employer. How much data is available to analyze? How quickly does that data get created? Just keeping up with this data is a process in and of itself. But you are tasked with not only understanding these mountains of data, but also extracting actionable business insights from this data.



Machines can help us analyze big data.

Luckily, that's what this course is about. Helping you develop the ability to use tools to extract the business insights from your data. Clearly, for most of us, there's too much data to analyze by hand. Rather we need help. That's where machines come in.



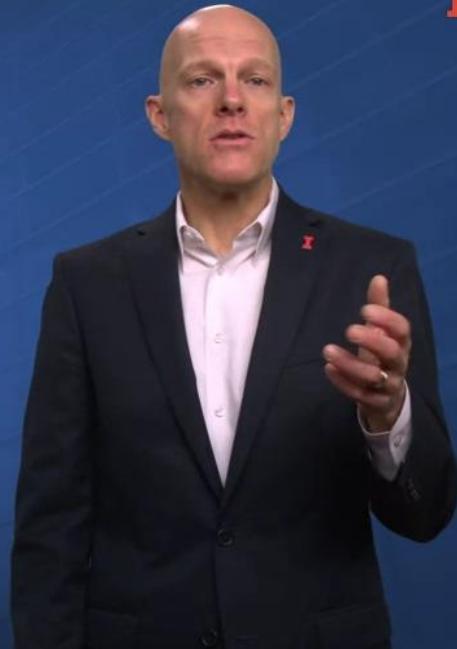
This course deals with data modeling and using machines to gather insights from data. After your data is acquired, cleansed, and ready for use, and explored, and understood it at a general level, it's time to use statistics to draw actionable insight.

In This Module:

The general process of machine learning

The menu of options available for analyzing data

The similarities among these models



This module will discuss that general process, the menu of options available for analyzing data, and general similarities that models have.

Machine Learning:

Solve a problem by gathering data and then using a machine to follow an algorithm that builds a statistical model based upon the data to gain actionable information from the data.



Machine learning has many definitions. But generally refers to solving a problem by gathering data and then using a machine to follow an algorithm that builds a statistical model based upon the data to gain actionable information from the data.

Machine Learning:

Machine learning algorithm
= program with instructions



I put it in another way. A machine learning algorithm is like a program with instructions. When you apply the algorithm to data, it creates a model. The model has new data along with instructions for how to make predictions with the data.

Learning in Machine Learning:

The algorithm can take the data and use the statistics and parameters from input and arrive at info without being explicitly programmed.



The learning part of machine learning refers to the fact that the algorithm can take the data and use the statistics and parameters that you give it and arrive at information without being explicitly programmed to arrive at that information every step of the way. Thus, the computer takes the general directions you give it and finds information that you did not explicitly tell it to find.



Before we move forward, it's useful to discuss some additional terminology. Let's first look at data science versus data analytics, versus business analytics. Unfortunately, for those of you who like clear definitions and distinct differentiation between terms, you'll be disappointed to learn that these terms are often used interchangeably. Most would consider data science as the broadest term and both data analytics and business analytics to be subsets of data science.

Data science:

Methods and procedures to extract knowledge from data with computers



Data science comprises methods and procedures to extract knowledge and insights from data with the aid of computers.

Data analytics:

A subset of data science, focusing on describing and visualizing data for specific objectives



Data analytics is similar, but a subset of data science that is more focused on describing and visualizing data for specific objectives.

Business analytics:

A subset of data analytics,
focusing on data and questions
related to business and involves
relatively few statistics



Business analytics is a subset of data analytics that focuses data and questions related to business and involves relatively few statistics.

Machine Learning, Artificial Intelligence (AI), Data Mining and Deep Learning



Next, let's define and distinguish machine learning, artificial intelligence or AI, data mining, and deep learning, etc. What do each of these terms mean and how are they different?

All are disciplines in data science

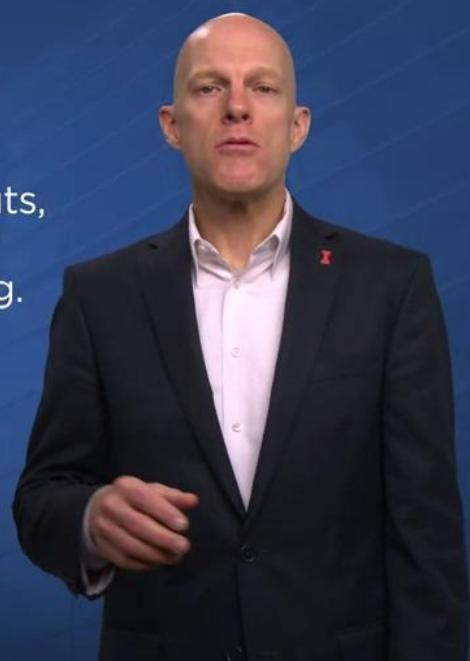
AI is the broadest term, referring to the ability of machines to perform intelligent and cognitive tasks

Machine learning and data mining are both subsets of AI.



Well, all of these are disciplines in Data Science. Artificial intelligence is the broadest term. It refers to the ability of machines to perform intelligent and cognitive tasks. Thus, AI simulates thinking. Machine learning defined earlier, and data mining are both subsets of AI.

Data mining is the cousin of machine learning, referring to analyzing inputs to detect outputs, but relying on direct human intuition, rather than self-learning.



Data mining is a cousin, if you will, to machine learning. It analyzes inputs to detect outputs, but relies on direct human intuition rather than self-learning.

Deep learning is a direct subset of machine learning, characterized by using progressive layers of learning.

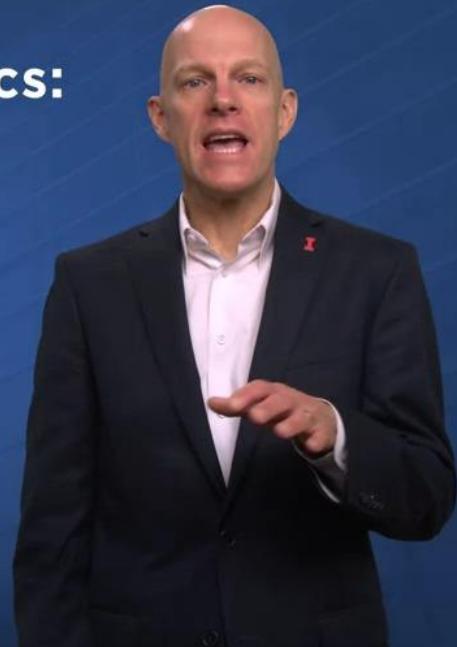
While many basic or shallow machine learning models derive output directly from their input variables, deep learning produces output based upon prior layers in the model.



Finally, deep learning is a direct subset of machine learning that is characterized by using progressive layers of learning. That is, while many basic or shallow machine learning models derive output directly from their input variables, deep learning produces output based upon prior levels and prior layers of the model.

Origin of Data Analytics:

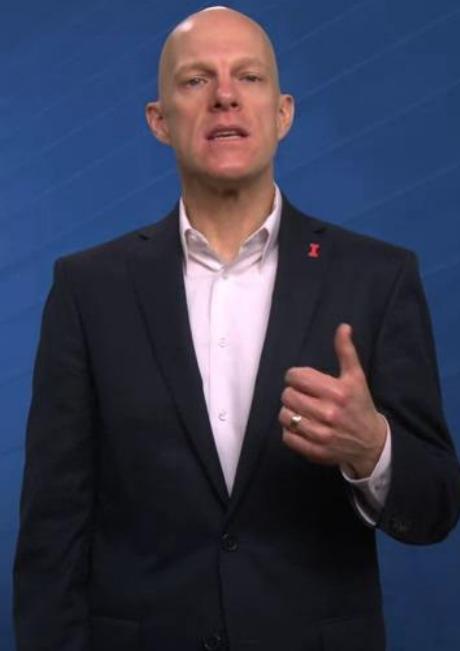
It combines computer science, math, statistics, information technology, operations management, and business analytics.



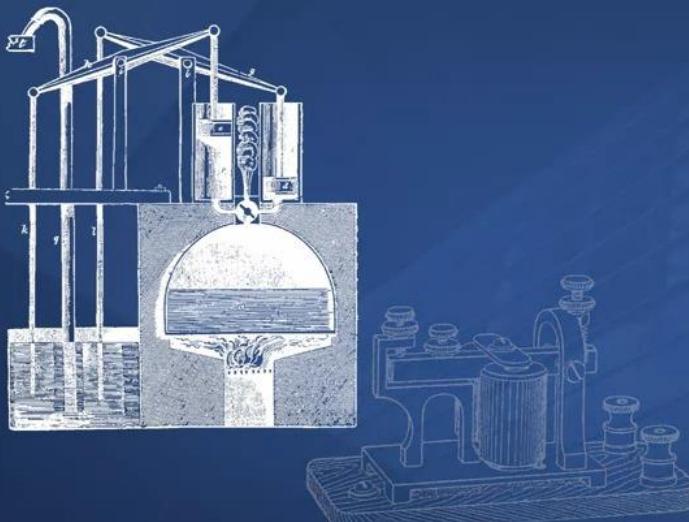
Next, it's useful to understand where this relatively new field of learning, data analytics and machine-learning came from. Data analytics is a confluence of many different disciplines, including computer science, mathematics, statistics, information technology, operations management, and business analytics. Thus, a data scientist and a data oriented business analyst is a relatively new creation.

Why do we care about machine learning at all?

How does machine learning help our business?



But perhaps at this point, it might be time to step back a little bit and ask, why we care about machine learning at all? How does machine learning help us in business?



1826 Galloway / Public Domain / Wikimedia /
Leupold's high pressure steam engine 1720

1905 The New International Encyclopædia /
Public Domain / *Wikimedia / Telegraph*



One way to think machine learning is just to relate it to the ways machines have helped businesses throughout history. Certainly, no one would doubt the usefulness to business of the steam engine or the telegraph. Like these early machines, computers and business learning process data at phenomenal rates of speed and draw connections, correlations, and insights that the analysts could get to know other way.



© 2013 Jurvetson S. / CC BY 2.0 / Wikimedia / *Caught Coding*

Of course, don't be misled by the many movies predicting machines are taking over.



Maximalfocus Public Domain / Unsplash /
The humanoid android robot Alter recreates human movements at the Mirakian museum in Tokyo

Machine learning helps the analyst derive useful and actionable insights from the data.

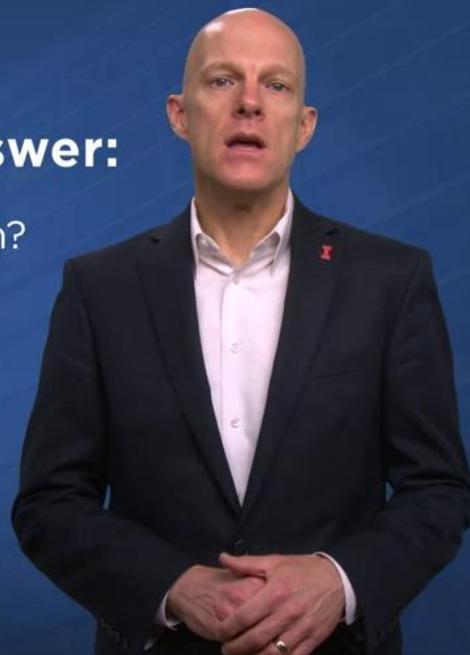
Questions Machine Learning Can Help Answer:

Numerical prediction. How much?

Cause and effect relationships.

What factors matter?

Will this work?

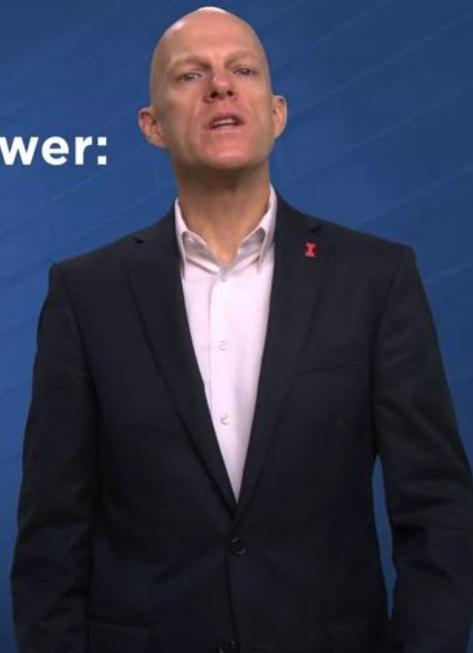


But it is still up to the analyst to act on these insights at least initially. Thus, machine learning is like a great blood hound. It helps you gain insight, but must be directed cautiously, or else it will run off chasing a squirrel. Finally, let's conclude by listing some questions that machine learning can answer. Here are some categories of problems that machine learning can help with and specific examples of problems from each category. First, numerical prediction as seen in regression. What do we expect, for example, costs to be next quarter in a segment? Next, cause and effect relationships between variables. What what factors matter? Will this work? For example, which factors have the biggest effect on shipping delay? Or does a particular change in our website improve click-through traffic? Or does this layout of our store improve sales?

Questions Machine Learning Can Help Answer:

Classification of examples.
Which category does this observation belong in?

Grouping things together.
Which group does this belong to?



Next, would be classification examples. What category does this observation belong in? For example, is this transaction fraudulent? Or what factors increase the likelihood of a purchase? Another category would be grouping things together. For example, which group does this belong to? Which customers should receive which type of targeted promotions? There's many others.

Lesson 2-1: Machine Learning Overview

Lesson 2-1.1 Inference, Prediction, and Experimentation

Categories of Machine Learning Insights

Prediction and predictive modeling.
Or generally what will happen?

Inference. Generally, why does something happen?

Experimentation. Generally, does changing something make a difference?

Machine learning uses statistical models to gather information from your data. The business analysts can then use that information to improve their business. There are three primary categories into which we group machine learning insights. In other words, machine learning helps us do the following three things. One, prediction and predictive modeling, generally what will happen? Two inference, generally why does something happen? Then three, experimentation. Generally, does changing something make a difference? I'll speak of each of these categories in turn and describe some examples of each.

Prediction (1/2)

Machine learning predicts outcome from new data based on existing data.

Prediction may be about numeric value, transaction class level, data group



Prediction is one important use of machine learning. With prediction, you're asking machine learning to look at the existing data and then to predict an outcome from new input data. This prediction may be about a numerical value. For example, based upon past data, how much do I predict sales will be next quarter? The prediction maybe about which level of a class I expect new data to be from. For example, based on the data I have, will this new information belong to a fraudulent or a non fraudulent class or transaction? The prediction may also be about what group the data belongs to. For example, based upon the data I've analyzed, a new data example or observation might be from a customer who will likely become a loyalty customer versus one who will not.

Prediction (2/2)

Prediction accuracy important,
not why model predicts outcome



If we are concerned only about prediction, we're not concerned about why the machine learning model predicts a certain outcome, only with how accurate the machine learning model is that prediction. Thus, we are indifferent about the underlying mechanism generating the prediction. We don't care why it predicts accurately, we just want to maximize accurate prediction. Thus, if we predict sales for the next quarter for a certain retail store, we're not concerned with why sales will be a certain number, but rather only whether the predicted number will be accurate.

Inference (1/2)

Use data to make inferences
about variables' effect on future
outcomes



But a second goal of machine learning is to understand why something happens or why a certain outcome occurs. Under this paradigm, we're seeking to use the data we have to make inferences about whether the variables we are examining will affect future outcomes that we're interested in. Thus, we are concerned with which variables in our data affect a target value of interest. Additionally, we are concerned with how these variables affect this outcome variable.

Inference (2/2)

How response + input variable associate

Make inferences based on stochastic models

A true model generates data



We're seeking to develop stochastic models which fit the data and then to make inferences about the data generating mechanism based on the structure of those models. We're thus assuming that there's a true model generating the data. We're concerned about the cause of the effects we see in the data. For example, instead of predicting next quarter's sales, we might be interested in what caused that quarter's sales and how our input variables affected those sales. Perhaps a percentage of our sales that come from a certain product negatively affect sales. That is, the more of that product that we sell, the lower sales actually are. This might happen if the sale of this product reduces the sale of another more profitable product.



Johnson Liu / Public Domain / Unsplash / *Untitled*



Of course, frequently we care about both prediction and inference. That is, we care about predicting future sales, for example, and we care about what variables affect future sales. As another example, we might use machine learning to predict the selling price of a house based upon that house's location, number of bedrooms, square footage,



2020 Shaylyn / Public Domain / Unsplash /
Brown Brick House With White Wooden Door Photo



color of the front door,



2019 Triquet / Public Domain / Unsplash / *Lake and Trees Photo*



lakeside location, etcetera.



© 2013 Prayitno / CC BY 2.0 / Wikimedia / *Freeway 405 & 105*



In addition, we might want to know that the location of the house next to a loud highway decreases the price of the house.



2016 Kock / Public Domain / Unsplash /
Red Bicycle Parked Beside Black Metal Gate in Front House

The color of the front door does not significantly affect the price of the house



2019 Trek / Public Domain / Unsplash /
Trees on Calm Body of Water Under Clear Blue Sky at Daytime

and the location on a lake increases the price of the house.

Machine Learning

Make predictions

Draw inferences



The magic of statistics allow us to address these questions and more. Understanding the ability of machine learning to make predictions and draw inferences helps us understand the usefulness of machine learning.

**Our goal is to use
data to build better
businesses.**



Finally, these two categories highlight the need to use data to capture valuable business insight. It's not enough to have data or even to understand our data. Our goal should be to use our data to build better businesses.

Machine Learning:

Be creative and explore.

Experimentation is a final
and important use of
machine learning.



Thus, we should use tools like Machine Learning to experiment, be creative, and explore. Experimentation is a final, important use of machine learning.

A/B Testing:



© 2004 Violetriga / CC BY-SA 3.0 / Wikipedia / *Web Traffic*

For example, a website retailer could test whether a certain change in the websites or treatments increases or decreases traffic in that part of the website. Thus, the manager leaves the website alone for some people, but then changes it for others and then uses machine learning to determine whether this changed, increased or decreased traffic on the website. This is often called A/B testing since the A group does not receive the treatment, but the B group does. A is then compared to B using machine learning to see if the treatment had a successful affect.

Build a Culture of Experimentation:

Data trumps opinions and opinions should be supported by data.



Stephan Tom Key and others argue that companies should be actively building a culture of experimentation. He believes that data trumps opinions and that opinions should be supported by data. Again, machine learning makes it possible to experiment and to know if those experiments succeed or fail.

**Machine Learning =
predict, infer, test
data, and let your
data speak.**



In some, machine learning gives you the ability to predict, infer, and test and to let your data speak.

[Lesson 2-1.2 Categories of ML Models, Part 1, Types of Data and Terms](#)

unique_id	transaction_id	unformatted_date	customer_id	product_id	product_name	category_id	category_name
0	20181016 961 1 100 2656161	2019-01-09	NA	18977	G07E10M	1	Fuel Premium
1	20170218 991 2 1 1894895	2017-05-14	6426302	6263	COKE CHRY 20OZ	250	Pk Bev-csd/pop (71)
2	20190530 1959 1 1 2043460	2019-08-22	NA	8830	MIKES HDR BLK CHRY 16OZ CAN	231	Flavored Malt
3	20170207 219 3 1 1630288	2017-05-03	NA	4546	HAMMS 6PK CAN	210	Beer-budget (44)
4	20170508 916 2 1 1154523	2017-08-01	NA	530	DT PEPSI .5LT 6PK	250	Pk Bev-csd/pop (71)
5	20190418 981 2 1 3117936	2019-07-12	NA	3823	NUT HAR PISTACHIOS 2.25OZ	55	Nuts & seeds
6	20170819 966 7 3 1815282	2017-11-12	NA	1738	BUD LT 18PK CAN	196	Beer-premium (42)
7	20180328 184 2 2 145024	2018-06-21	7364740	10989	TRKY SOURDOUGH	274	Cold Food
8	20171224 834 3 1 3141922	2018-03-19	NA	3738	LAYS CLASSIC 2.75OZ	52	Potato Chips

Our goal in this video is to look at all of machine learning, the whole landscape and build a foundation that will help you as you move forward and discuss individual algorithms in more detail. To do this, we'll need to define some terms. Our first step in building this foundation or this framework, is to understand important terms and phrases. Thus, we will discuss terms used to describe the typical dataset used for machine learning and define many aspects of it. Next, we'll introduce several different categories of machine learning algorithms. Finally, we'll introduce some of the key machine learning algorithms themselves. Let's start building our framework for machine learning. In order to extract information from data, machine learning algorithms need data, and data is why we're all here today. Broadly, there are two categories of data that we'll talk about in machine learning; a structured data and unstructured data.

Structured Data:

Each example has the same features

These features are organized in a form that the computer can understand easily



In this class, we're most concerned with structured data. Structured data is where each example has the same features. These features are organized in a form that the computer can understand easily. Thus, you can think of this as a spreadsheets or a matrix.

Unstructured Data:

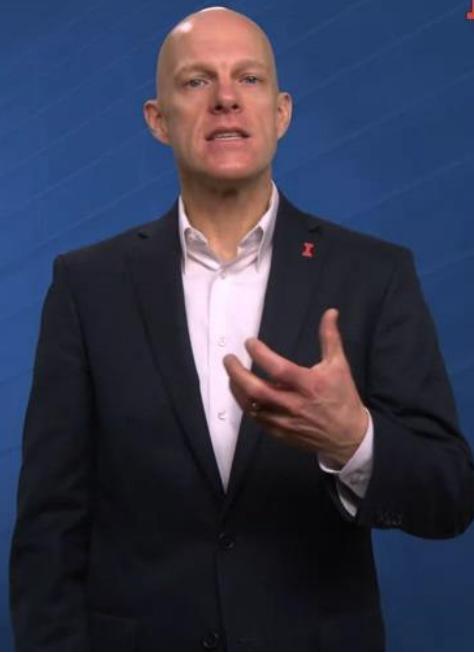
Sounds

Pictures

Social media posts



2018 Geralt / Public Domain / Pixabay /
Images by Machine Learn Photos Cat Human



Each example is a row and each feature is a column. The second type of data is unstructured data. These data are not structured into spreadsheets. They could be sounds, pictures, and social media posts. In order to use these data will need to be structured in some way.

Two Types of Variables:

1. Continuous Variable

Each increment has meaning

Differences between two variables have meaning



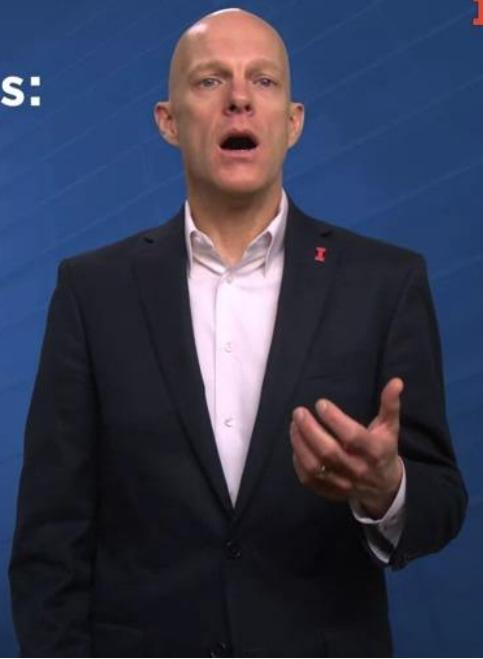
Again, in this class, we will focus on structured data in the form of a table or matrix or a spreadsheet. Within a spreadsheet, there are two broad categories of data. One, quantitative variables and two qualitative or categorical variables. Now a quantitative or continuous variable is a number where each increments and the number higher or lower has meaning. Further, the difference between two entries, for example, 10 and 12 has meaning.

Two Types of Variables:

2. Categorical (discrete) Variable

Dichotomous

Differences between two levels of a variable do not have meaning



For example, you can think of revenue recorded for a product. A \$10 product has meaning and that is different from a \$10.50 product. Also, that 50 cents difference between those two has meaning as well. On the other hand, there's another category of variable for which in-between measurements between two categories are not meaningful. These variables are thus just categories, and we often call them categorical variables or discrete variables. If there are just two categories like yes or no, true or false, we'll call them dichotomous. A good example of this is a feature that lists cities in the United States; Burlington and Waterloo. Two different cities, for example, are categories that have no order. They're categories or classes because the difference between these two means nothing.

Two Types of Variables:

1. Continuous Variable
2. Categorical (discrete) Variable

Differences between two levels of a variable do not have meaning

Note: can be a number, but they are arbitrary numbers.

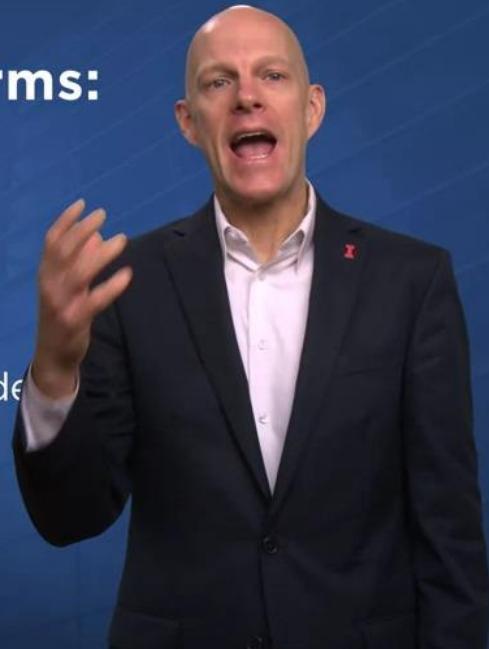


That is half of Burlington, doesn't mean anything in this context, and R would store this as a character. Now let's make one more important point. What if our column of cities did not have names in it, like Burlington and Waterloo, but instead, replace those names with a number with this column, then be categorical or continuous? This is an important question because it changes how the machine it should treat and process that variable. It's important that we get this right. It might be a bit confusing because this is clearly a number, but the correct answer here is that this is a categorical variable still. The numbers here are just arbitrary numbers that represent the cities and nothing else. Thus, even though they're numbers, the difference between two cities, in this case two numbers is still not meaningful.

Machine Learning Terms:

Unit of observation is the smallest entity with measured properties of interest for study.

Unit of analysis is the smallest unit from which inference is made.



Now to be successful at machine learning, we need to speak the language of machine learning. That is, we need to understand the terms that are routinely used. Thus, let's continue to explore the terms of machine learning. Please be warned that different people call these same things different names. Machine learning is not owned by any one academic discipline. It's part of statistics, part of computer science, and a bit of lots of other disciplines. Thus, the different disciplines give these things different names. I'll do my best to give you the most common names. A unit of observation is the smallest entity with measured properties of interests for study. Thus, for example, one product sold in a transaction could be a unit of observation. Next, a unit of analysis is the smallest unit from which inference is made. Thus, while one product from a transaction may be the smallest unit of observation, you may be more interested in aggregated sales over the course of a month. Thus, the unit of analysis might be the monthly sales in a product category.

Machine Learning Terms:

Examples, cases, values, and observations are instances of the unit of observation.
Each example is a row.

Features, variables, dimensions, attributes, inputs, and predictors are properties or attributes.
Each feature is a column.



Next, examples, cases, values, and observations are instances of the unit of observation. In our case, since we're using structured data, this is a matrix or a spreadsheet, and each example is a row. Next, features, variables, dimensions, attributes, inputs, and predictors are properties or attributes of examples. Again, in our case, using matrix data, each feature would be a column. For example, if we're examining products sold in individual transactions, the feature could be the product itself, time of sale, location of sale, whether a customer had a loyalty card and etc.

Machine Learning Terms:

Target feature can be called the dependent variable, the output variable, the response variable, or y .

The other features will be called the independent variables, the input features, or x .



Frequently in machine learning, we're examining the effect of multiple feature of variables on one target variable. We want to predict that target variable or examine how the other features affect the target feature. When this is the case, the target feature can be called the dependent variable, the output variable, the sponsor variable or y . The other features will then be called the independent variables, the input variables or x . For example, we might want to predict next quarter's revenue for a store that we manage. Thus, the target or dependent variable is next quarter's revenue and the independent variables or the input variables, could be current revenue, direct costs, costs of goods sold, seasonal demand, etc.

[Lesson 2-1.3 Categories of ML Models, Part 2, Categories of Algos](#)

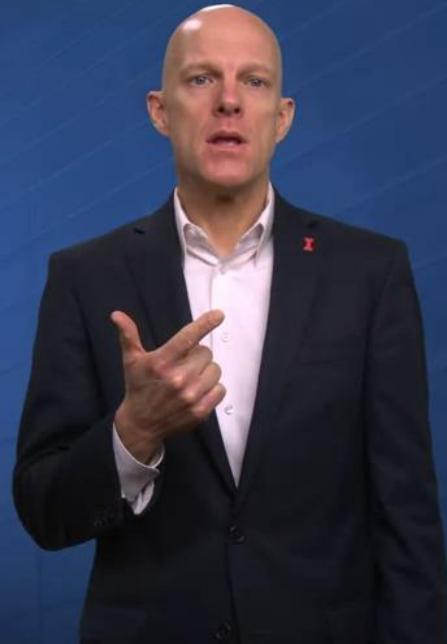
**Different situations
require different
algorithms.**



Next, in our effort to build a framework for understanding machine learning, let's explore the many different types of learning algorithms that exist. It would be really convenient and this class would be much shorter if there were one algorithm that would work in every situation. Unfortunately, every situation for which we can use machine learning is different since no two datasets are the same. Thus, as D. H. Wolpert says, there's no free lunch in machine learning. There is no one model that works best for all possible situations. There are generally three distinct categories of machine learning algorithms:

Types of Machine Learning Algorithms:

- Supervised
- Unsupervised
- Reinforcement
- Semi-supervised learning
- Data mining



supervised, unsupervised, and reinforcement. In addition to these three main categories, we'll also talk about semi-supervised learning and also data mining.

Supervised Learning:

Rely on labeled dataset.
All features have a value.

A model analyzes relationship between features and an existing target feature.

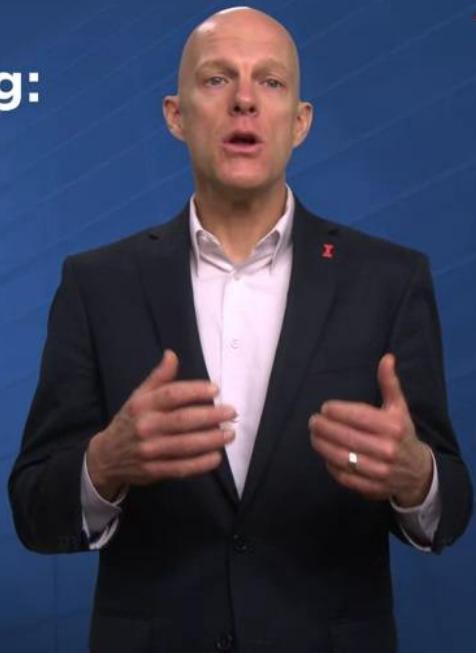


Supervised learning then is characterized as having a labeled dataset. That means that all features, including the feature of interest are labeled, and that is, they all have a value. Thus, a supervised learning algorithm uses data to create a model that finds a combination of features that results in a target feature or label, which can be a number or a category or class.

Unsupervised Learning:

Rely on unlabeled dataset, or features without a target feature

Feature transforms into another vector or a value.



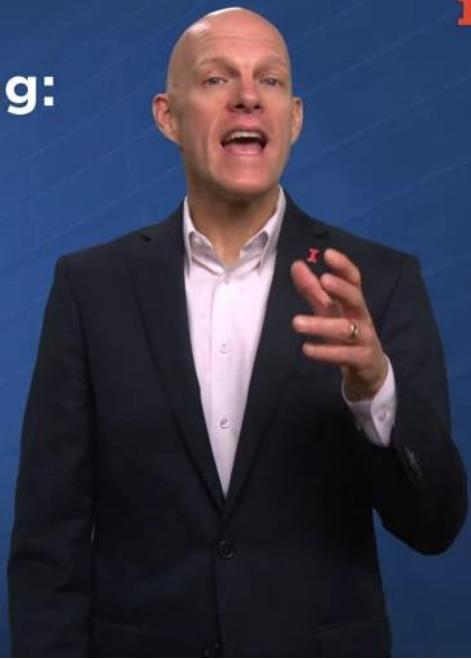
Let's state this in a slightly different and maybe more helpful way. A supervised learning algorithm uses a model to analyze and decipher the relationship between features and an existing target feature. For example, regression is a supervised learning algorithm. They could use features such as revenue last quarter, revenue last year, forecasted weather, etc, to predict the target feature of revenue for the next quarter. Next, a classification algorithm is a supervised algorithm that could use features such as time of day, amount of purchase, location of purchase, etc, to predict a label for the target variable of whether, for example, a purchase is fraudulent or not. Unsupervised learning is characterized by a dataset of unlabeled examples, that is, features without a target feature. The unsupervised algorithm transforms the features either into another vector as in dimensionality reduction, or into a value as in clustering.



Let's put this into different, perhaps easier terms. Since only input variables are present and no targets or output variable exist, these algorithms analyzes relationships between input variables and uncover hidden patterns that can be extracted to create new labels for output variables. For example, in a clustering algorithm, you take features such as an item purchased, time of a purchase, etc, to cluster customers into groups. For example, the model might determine, based upon convenience store purchases, that a customer belongs to a fuel group, or a fountain soda group, or a weekend alcohol group, or a daily lottery group.

Reinforcement Learning:

Uses data to create a policy that takes features of the data and outputs an optimal action for that state.



Next, reinforcement learning uses data to create a policy that takes features of the data and outputs an optimal action for that state. For example, think of learning to play a video game.

Semi-supervised Learning:

Just like supervised learning,
but some of the target variables
are missing

Training a model with labeled
and unlabeled data



You want to maximize the outcome of success and at each stage, you learn more and use that learning to reinforce and influence subsequent gameplay. For example, reinforcement learning can help a car learn not to crash and can help a computer learning from scratch how to play chess. Next, semi-supervised is just like supervised learning, but some of the target variables are missing. Thus, you're training a model with labeled and unlabeled data. This can happen when labeling the data manually all the way through. It's too costly. Thus, you could build a model with labeled cases and then use that same model to label the remaining cases.

Data Mining:

Refers to different machine learning techniques.

Analyzes inputs to detect outputs

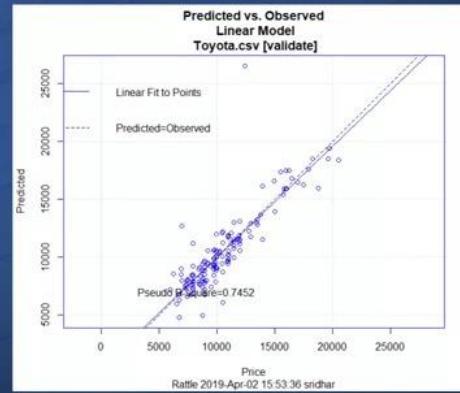
Relies more on human intuition than self-learning



Finally, data mining is frequently used to refer to different machine learning techniques. Thus, many sources see data mining as the very same thing as machine learning. If we're trying to distinguish data mining from machine learning, we would say that machine learning analyzes inputs to detect outputs, but relies more on direct human intuition than on self-learning.

Supervised Learning: Regression Algorithm

Uses regression to predict and examine target variables that are numerical and continuous.

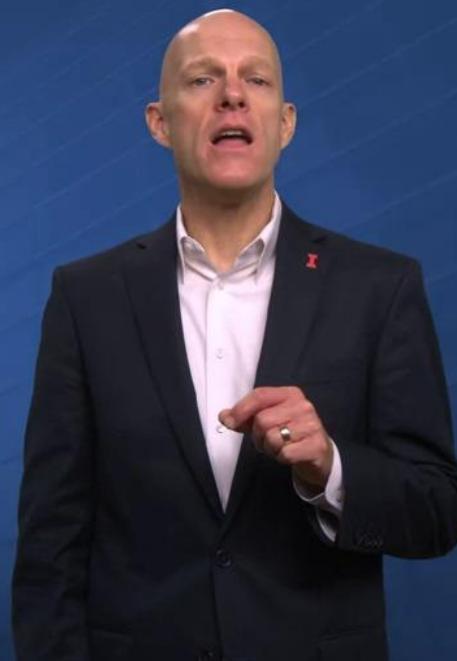


Rattle GUI / Togaware

Finally, let's mention just a few of the key algorithms in supervised and unsupervised learning. Supervised learning uses regression to predict and examine target variables that are numerical or continuous. An example of a regression problem, that is, a business problem for which a regression algorithm could provide insight, is investigating the quarterly sales by store of a set of convenience stores.

Supervised Learning: Regression Algorithm

Uses regression to predict and examine target variables that are numerical and continuous.

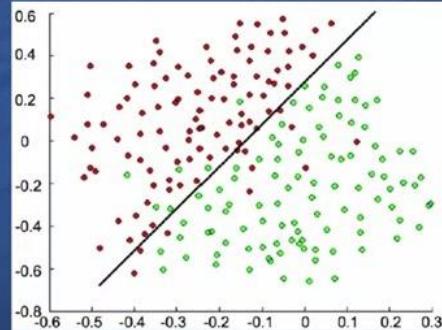


A regression model could use input variables such as store location, past sales, and current costs to predict future sales. This model would allow the analyst to examine the effect of these input variables on the output variable, quarterly sales.

I Supervised Learning: Classification Algorithms

Includes logistic regression, decision trees, random forests, and k-nearest neighbors.

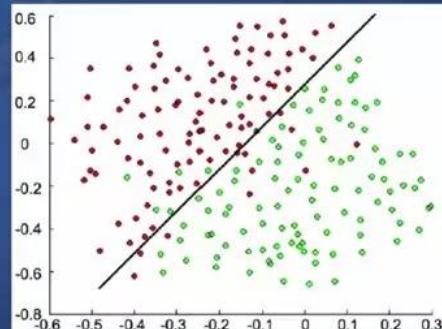
They predict and examine target variables that are categorical.



It would also be useful to predict next quarter sales, which is very valuable for forecasting and resource planning. Notice that the target feature in regression is a continuous variable. On the other hand, classification algorithms such as logistic regression, decision trees, random forests, and k-nearest neighbors, predict and examine target variables that are categorical.

I Supervised Learning: Classification Algorithms

They create decision boundaries that separate target variables into different classes.



They create decision boundaries that separate target variables into different classes.

Unsupervised Learning:

Includes clustering, k-means clustering, and density-based clustering.

Groups observations together based on features.

Works on only input variables.



An example of a classification problem is a company that wants to predict whether a purchase will be from a loyalty customer or a non-loyalty customer. A logistic regression algorithm could use a model to take the input independent variables and predict whether the output dependent variable is a loyalty customer or not. This model would be helpful because it can be used to predict what type of product is often purchased by a loyalty customer, then that product could be used to promote the loyalty program, particularly in locations that have poor loyalty system adoption. Next, in unsupervised learning, clustering, k-means clustering, density-based clustering, group observations together based upon their features. Clustering algorithms, unlike regression and classification, work on only input variables, often called unlabeled data, to put different observations into groups based upon those variables.

Unsupervised Learning:

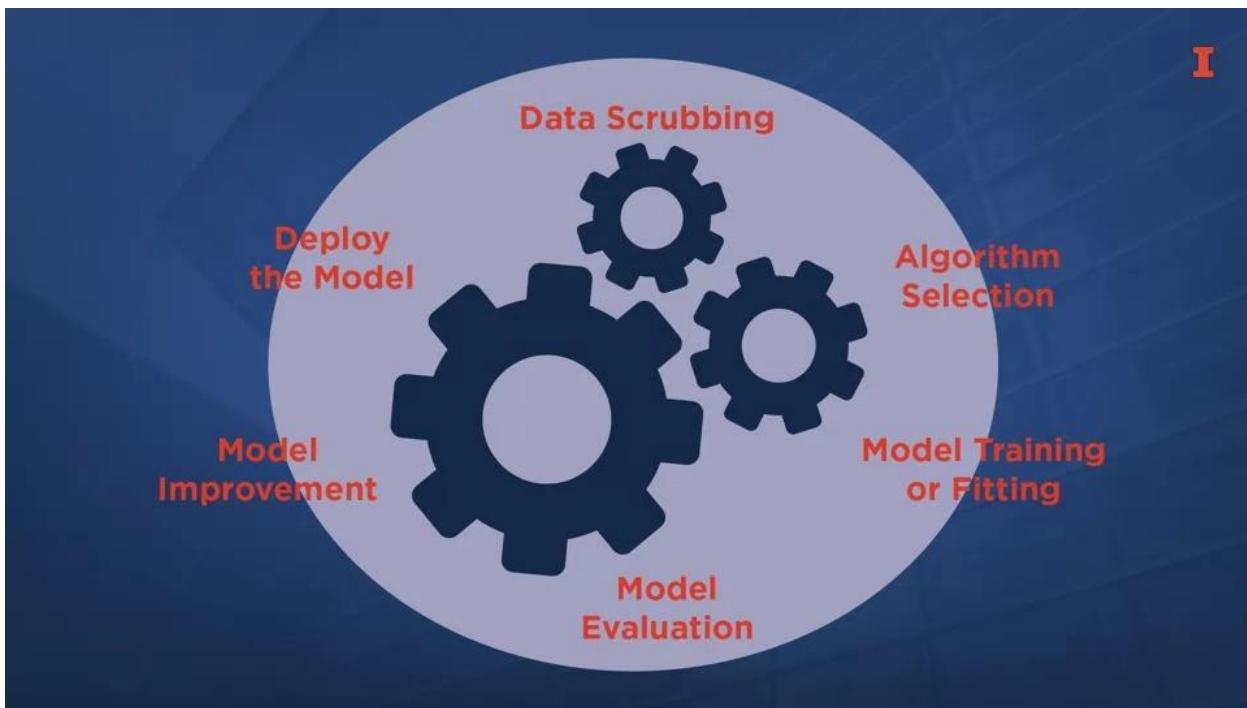
Rattle GUI / Togaware

A clustering problem could be that a convenience store business wants to categorize its customers into different groups, such as a gasoline group, a morning coffee and breakfast group, and the lunch group. Being able to identify customers in this way could lead to targeted promotions and discounts to increase traffic and increase the size of purchases of each cluster.

Unsupervised Learning:

Here's a final graphic that includes a sample of the whole landscape of machine learning.

[Lesson 2-1.4 How Machine Learning Works in General](#)



In this video, we'll discuss the basic level, how machine learning algorithms and models work. Now it's important to keep in mind that although there are many different machine learning models, options for those models called hyperparameters and questions that can be addressed with machine learning, all machine learning algorithms follow the same basic process. They take data and identify patterns that form the basis for further action. There's six general steps necessary to use machine learning. These are the following: one, data scrubbing, two, algorithm selection, three, model training or fitting, four, model evaluation, five, model improvement, and six, deploy the model. I'll talk about each of these in turn.

Data Scrubbing:

Extract, transform, load

Feature selecting – eliminating
and selecting redundant columns

Replace text in categorical
variables with numbers



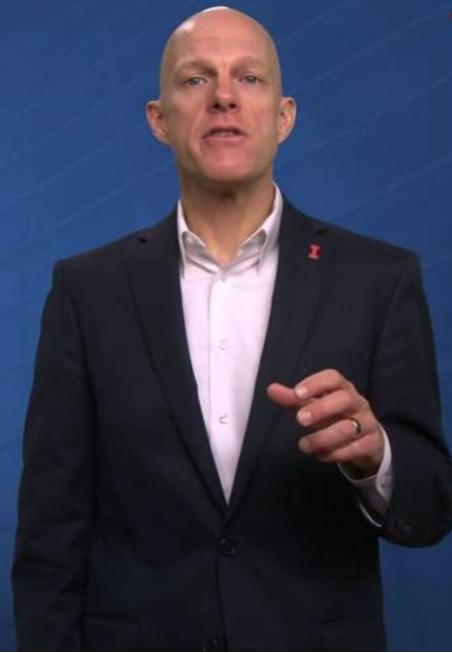
The first step in running a machine learning algorithm is getting, loading, transforming, and cleaning data. After the initial ETL or extract, transform and load process is complete, the data is in a usable form and relatively free of mistakes and problems. There are usually a few more data cleaning issues that might be necessary based upon the particular algorithm that will be used. For example, redundant columns might need to be eliminated. For example, a data set that lists products and has one column for the product name, and another column with a unique numerical identifier for each product has a redundancy. In this case, one of the columns should be removed. Eliminating and selecting columns is often called feature selection. Another data scrubbing activity would be to replace text in categorical variables with numbers. This is sometimes called one-hot encoding. For example, a column that lists cats and dogs could be changed to a column that lists zero for cats and one for dogs. This is also called dummy coding.

Data Scrubbing:

Standardization

Amount of data needed.

Rule of thumb: 10 times as many data points as number of features



Another data preprocessing technique that may be beneficial given the machine learning algorithm that you use is standardization. Standardization, it takes variables that are on different scales and puts them on the same scale so that they have the same variance. For example, assume your data set lists financial statement numbers for a company, and that's some of the numbers are in thousands while others are in millions, standardizing these figures by subtracting the mean and dividing by the standard deviation would give both measures a mean of zero and a standard deviation of one. Standardizing in this way is generally recommended for principal components, support vector machines, k-nearest neighbors, and other algorithms that use distance measures. A final topic of interest is not how the data should be scrubbed but how much data is needed in the first place. Well, there's no single definitive rule here, there are rules of thumb. Ideally, you have at least some observations for each combination of levels of the variables. For example, if a data set has two categorical variables with two levels each, for example, animal with cat and dog and age, old or young, it would be ideal to have data for old cats, young cats, old dogs, and young dogs. This way, things won't unravel at the side of unseen combinations in trying to predict the data in the future. A rule of thumb for the minimum amount of data to have to run machine learning algorithms is 10 times as many data points as the number of features. Ultimately, how much data is required depends on the algorithm that's being used. Clustering and dimensionality reduction algorithms can be effective with less than 10,000 observations. Regression and classification algorithms can get by with less than 100,000 observations while neural networks need more than 100,000 observations.

Algorithm Selection

Selecting the best algorithm for your analysis is one of the hardest jobs for business analysts.

Select best algorithm:
examine question and dataset.
Then determine solution needed.



The next step is algorithm selection. Selecting an algorithm that's best for your analysis project is one of the business analysts hardest jobs. The analyst examines their question, their data set, and they need a solution and picks accordingly. Unfortunately, there's no one model that's best for every business problem and business data set. Analysts might be tempted to reuse models that they're most comfortable with, but the analyst should be aware of all the models available and also needs to be aware of recent advances in the field.

Model Training/Fitting:

Apply algorithm to data to create model

Split training and testing datasets

Model created on training data set

Data split to check if model works for current and new data



The next step then is to apply the algorithm to the data to create a model. The model takes the data set, following the rules of the algorithm, and converts it into information that can be used to solve a business problem. That is, the machine learning model includes new data generated from applying the algorithm to the data set, as well as instructions for the machine for how to use that data to create predictions on new data. Frequently, but depending on the algorithm, the model is created on a subset of the data called the training data set. Specifically, the data set is split into two separate parts, called the training data and the testing data. For example, the data set might be split 70 percent for the training and 30 percent for testing. A 60, 40 or a 80, 20 split could also be used. Now this is sometimes called split validation. Once the model is trained on the training data, the testing data can be used to check the accuracy of the model. Thus, the point of splitting the data is to check whether the model that is created works not just on the data that we have, the training data, but also on the new data, the holdout data, or the testing data as well. This avoids the situation in which our model fits the data we have perfectly, but does not generalize to the other data. That is, it does not work for new data. This is important because the goal of many machine learning solutions is to speak not only to the data we have, but also to predict future outcomes and future relationships between variables.

Model Evaluation:

Examine measures of model's accuracy



After the model is trained, the next step is to evaluate how accurate the model is. As just mentioned, the goal of creating the model is to explain or predict the data we have, but also to explain and predict new data. Since our true goal and training a model is to solve a business problem, we want our model to predict the truth as closely as possible.

Measures of Accuracy for Classification Models:

Area under the curve (AUC)

Receive operating characteristic
(ROC)

Confusion matrix

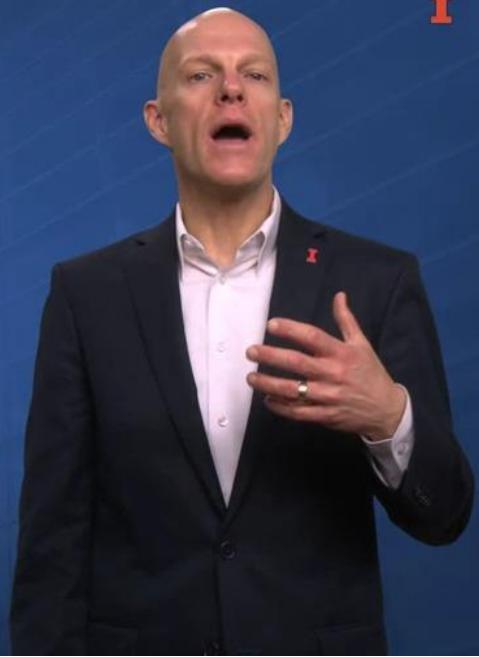
Recall



To evaluate the model's ability to do that, we examine measures of accuracy for the model. For example, if our machine learning task is to accurately classify whether a purchase is fraudulent or not a classification task, we would examine how accurate the model is at predicting fraud within our training data set. Next, we would examine how accurate the model is at predicting fraud and the testing data set. If accuracy is relatively high for both the testing and the training data sets, we'll have some assurance that the model will predict fraud well for new data. Measures of accuracy for classification models, include the area under the curve, AUC, the receiving operating characteristic, the ROC, a confusion matrix, recall and etc.

Model Improvement:

- Hyperparameter tuning
- Collect more data
- Feature selection
- Pick a new algorithm



If the accuracy of the model is not good enough, the next step is to improve the model. Many methods exist to improve the quality of the model. In fact, competitions to create the most accurate and useful model are a significant part of the machine learning culture. A first step for seeking to improve the accuracy of the model is to change the instructions given to the algorithm. Algorithms have options that control and impact how the resulting model learns and which patterns to identify and analyze. These options are called hyperparameters. These options can be changed by the analysts to fine tune the model to better fit the data set. Although fitting the model too closely to the data set can cause a problem called overfitting. Next, the analysts might increase the amount of data. All data has noise or irreducible error caused by missing variables or a unique, idiosyncratic variation. Increasing the amount of data can reduce that variation. Similarly, feature selection, that is adding additional variables that strongly affect the target variable and removing less useful variables may help. Thus, it's important to understand that the data scrubbing and cleansing process can sometimes just beyond going. Finally, the analyst might need to pick an entirely new algorithm.

Deploy the Model:

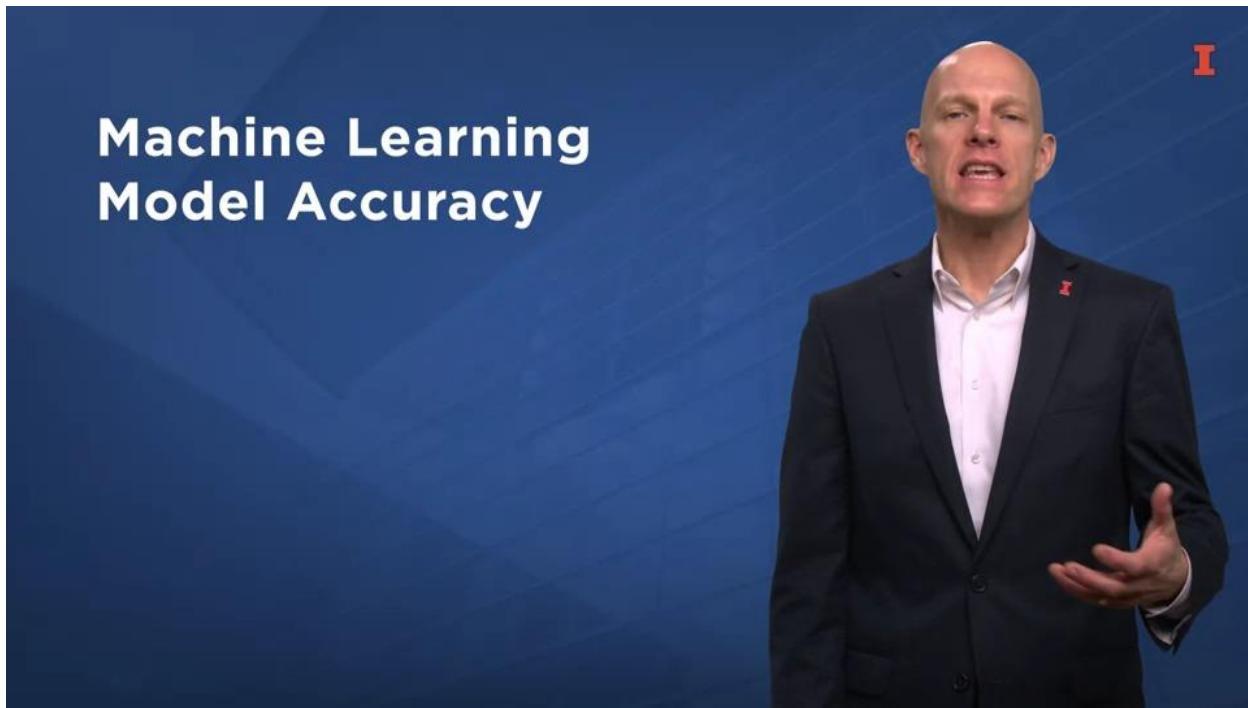
An accurate model:

- Gives support to implement change
- Reallocate resources
- Forecast future financial results
- Classify new data as helpful or problematic



The final step in this process is to deploy the model and put it into production. The whole point of using machine learning in the first place is to create a model from your data that helps you address business problems. An accurate model can give you support to implement and test a change in your business, reallocate resources to create or emphasize different classes of products or customers, forecast future financial results for a segment of your business, and classify new data as helpful or problematic.

[Lesson 2-1.5 Evaluating ML Model Quality](#)



In this video, we'll discuss the important topic of machine learning model accuracy. Now, this is a nuanced topic with no simple, easy solution. The business analyst has to make trade-offs, and it's important for us to spend a little time to understand the issues at hand.

**We attempt to create
a model that predicts
or uncovers truth of
some kind.**



At first, it's important to remember the goal of using machine learning in the first place. Though machine learning can be used for a lot of reasons, at its core, we're interested in creating a model based upon our data that predicts or uncovers truth of some kind. Once that truth is found, we can use it to make better business decisions and solve business problems. For example, we might want to know the truth about how a certain store will perform next quarter. Alternatively, we might want to know the truth about whether a new purchase is a real purchase or a fraudulent purchase.

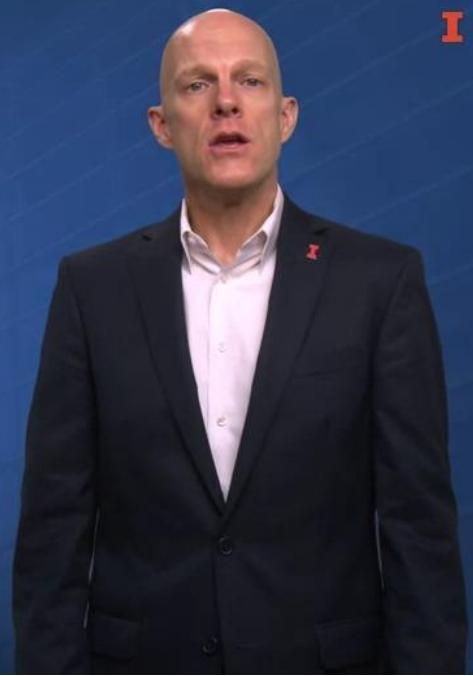
All data has irreducible noise.



The problem, of course, is that it's difficult to know the truth. Now, we're not trying to get into philosophy here. We're just stating the fact that the dataset we're using to find this truth is incomplete. That is, all data has irreducible noise. Unless you're looking at every single data point that you care about, some, hopefully, a lot of your data's characteristics and relationships will translate to other similar data. But there will always be some portion that does not translate. This could be because you don't have all of the important features of the real world in your data, or just because all the data is a little bit unique. For example, say you have a dataset that lists transactions and whether those transactions are fraudulent or not. If your business problem is to predict whether a transaction is fraudulent or not, your reason for building a model is not to perfectly fit your current data, but rather to use that information to predict whether a future transaction is fraudulent. Your current data can help with that, but it does not contain every possible transaction of every variable that would explain whether a transaction is fraudulent. Further, the data you collected last Tuesday, for example, will be unique, at least in part to last Tuesday, and will not be 100 percent representative of every Tuesday ever in the future. Furthermore, there might be small errors or mistakes in that data.

Overfitting:

If your model matches your data too perfectly, it will likely NOT match new data very well.



The key point is that because all data has some noise or irreducible error in it, if you want your machine learning to apply to more than just this data, you need to make an intelligent trade-off between your model being accurate, but not too accurate. Because if your model matches your data too perfectly, it will likely not match new data very well. This is what we call overfitting. Thus, there's always a trade-off between training a model that's good, but not too perfect, not too good. You want your model to generalize, but not to memorize.

Split Validation:

Split data into testing and training segments

80% training set and 20% testing set

Or 70% training set and 30% testing set



How does this work in practice? Suppose you have some data and you want to use a machine-learning algorithm to train a model that will predict a future data point. One way to avoid overfitting is to split your data into testing and training segments called split validation. Thus, you split your data into two groups of 80 percent and 20 percent, or 70 percent and 30 percent.

Split Validation:

Randomly sort data before splitting into training and testing sets.



A key step to remember is certainly to randomly sort your data before splitting it into your training and testing sets. Most datasets come sorted in some logical order. If you keep the data in this order, it's likely that your training set will be systematically different from your testing set since they're on different ends of meaningfully ordered data. For example, the data might be sorted alphabetically based upon date. Thus, the testing data would be systematically newer than the training data.

Split Validation:

Train your model on training set;
test model on testing set.

Then compare the accuracy of
your model.



Next, to help make sure your model will generalize to new data, you train your model on the bigger portion, the training data, and then test the model on the smaller portion, the testing data. Then you compare the accuracy of your model in the training data to that in the testing data,

Split Validation:

Underfitting:

When the accuracy of the
model is low on testing in
the training set



you would like the accuracy of the model to be high in the training data. If it's not, we call that underfitting. If the accuracy is low in the training data,

Split Validation:

What to do when underfitting:

Add complexity to your model

Pick a new algorithm with a
more flexible model



you'll need to add complexity to your model or pick a new algorithm with a more flexible model. This will help the model match the data points more specifically.

Split Validation:

A model that underfits the
training data is said to
have high bias.



A model that underfits the training data is said to have high bias.

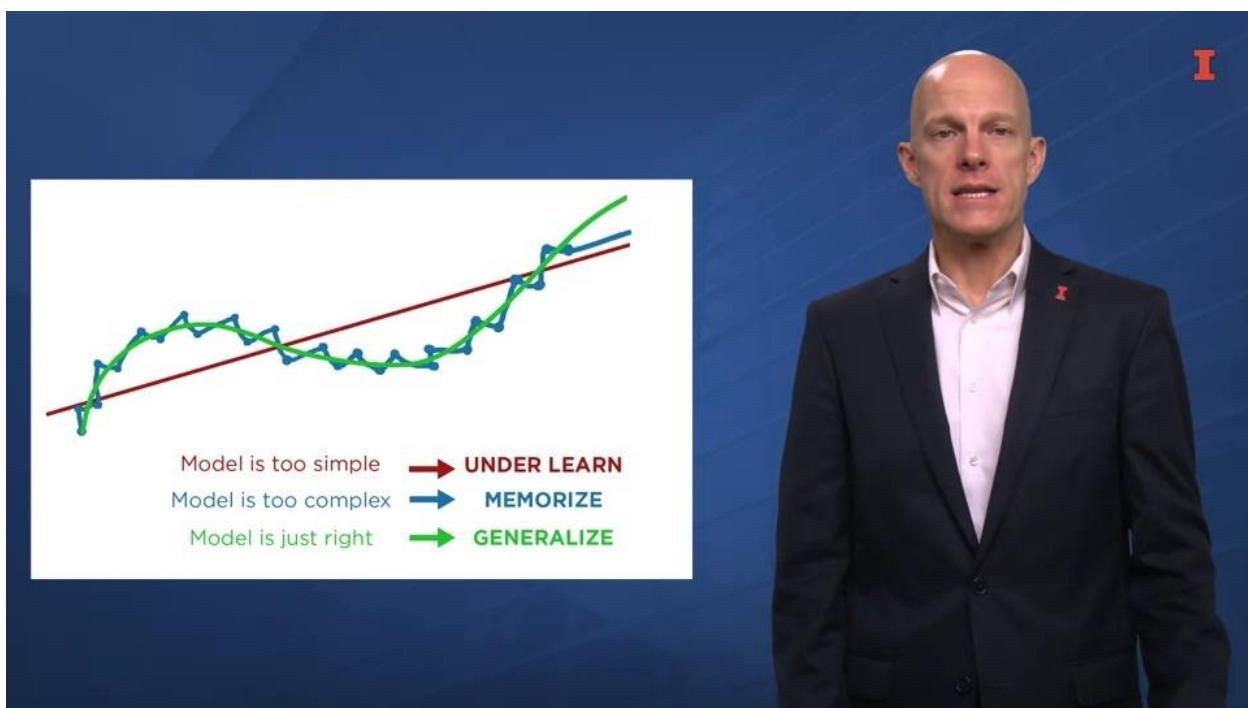
Split Validation:

After overcoming underfitting,
run your model to the test data

A model that overfits the
training data is said to have
high variance.



Next, you run the model on the test data. You want the accuracy of the model to also be high in the test data. If it's not high, you have overfit your model to the training data. The model is too specific and does not generalize. This means the model is responding too much to random chance or noise that's in the training data. If this is the case, you can add more data to the training set and/or reduce the complexity of your model, or pick a new algorithm with a less flexible model. A model that overfits the training data is said to have high variance.



So our constant effort as analysts is to pick algorithms and tune models to avoid underfitting and overfitting, that have low bias and low variance. Let's briefly summarize this discussion by using this picture. Suppose the dots here represent your training data and suppose that the three lines are the predictions of three separate machine learning models, the red line is too simple and biased and underfits the data. However, the blue model appears to go too far. It's too complex and specific, it has too much variance and overfits the data hitting every point exactly. It's likely that when this model is run on the testing data, it will not fit the testing data well, because it so specifically matches the training data. Finally, the green model seems just right. It's more complex and specific than the red model, and thus does have some more variance, but it does match the data well, and thus reduces bias. We'll have to check with the testing data, but it appears that the green model avoids both overfitting and underfitting and will thus generalize well.

Lesson 2-2: Business Problem

Lesson 2-2.1 Introduce the Business Problem to Solve



In this video, we'll set up a realistic hypothetical business problem. In subsequent videos, we'll use a new data analytics tool to gain useful insight into this problem. The hypothetical company will be helping is TECA. TECA owns over a 150 convenience stores and gas stations throughout the middle of the country. TECA knows from prior research and general industry knowledge, that a new loyalty customer is worth about \$200 earn extra revenue a year. A loyalty customer is an individual who has provided certain information to TECA convenience store in order to receive a loyalty card. Loyalty cards provide discounts and benefits for those who use them frequently. You probably have a loyalty card from your favorite grocery store, convenience store, ice cream shop, or movie theater. These businesses know that the small discounts they give you on your loyalty card are more than compensated for by the increase in business you provide to them.

Loyalty Customers:

Spend more money

Buy higher margin products



Not only do loyalty customers spend more money, as can be seen in this graph, loyalty customers also buy higher margin products. This graph shows the gross margin profit for non loyalty and loyalty customers. You can see that the loyalty column looks significantly higher than the non loyalty column. Thus, loyalty customers not only come into the store more often, they also buy more profitable items when they do come.

Why Predict if Transaction is by Loyalty Customer?

Increase percentage of loyalty customers

Increase number of transactions those customers make



TECA would like to be able to predict when a transaction is more likely to be made by a loyalty customer for two reasons. First, they'd like to increase the percentage of their customers who are loyalty customers. Second, they would like to increase the number of transactions those customers make. Thus, TECA is embarking on a research project to examine how they can make transactions more appealing to loyalty customers.



What TECA would like to do is predict when a transaction will be from a loyalty customer. Perhaps certain products drive loyalty purchases. If that's the case, TECA can promote these products to non loyalty customers to encourage participation in the loyalty program. Perhaps there are certain times of the year when loyalty purchases are more likely to occur. If so, TECA could advertise accordingly by focusing on promotional programs on those months in which loyalty is most or least likely to occur. Finally, particular regions of the country might be better at selling to loyalty customers. If those areas and stores are identified, TECA could investigate their underlying success and replicate it elsewhere to make sure that more purchases are made by loyalty customers.

Business problem:

Build a model that predicts when a transaction is likely to be from a loyalty customer versus a non-loyalty customer.

Explore how different products, store location, and time of year affects that model.



Thus, TECA would like to build a model that predicts when a transaction is likely to be from a loyalty customer versus a non loyalty customer. In addition, TECA would like to explore how different products, store location, and the time of year, affect that model.

Lesson 2-2.2 Introduce the Data



In this video, we'll take a brief look at the data we'll be analyzing throughout this module. We'll be using a neat data set created for this course called the TECA dataset.



TECA is a fictional company that owns over 150 convenience stores and gas stations throughout the middle of the country; from Wyoming to Minnesota, from Arkansas to Alabama and many states in between.



The particular dataset we'll be using is a random selection of customer transactions from these convenience stores for the years 2017-2019. These transactions include, for example, purchases of fuel, lottery tickets, soda and candy. Though our dataset is relatively large, including over a million transactions, it only includes a pretty small random sample from all of the transactions in this period. The full dataset would be over a 145 times larger than this and consist of hundreds of millions of transactions.

Data Aggregation:

Level of aggregation =
what each row represents

Transaction/product level



It's important to know the level of aggregation of the data. That is, what each row represents. In this case, each row represents a single product purchased as part of a single transaction. Thus, the data is organized at the transaction product level. Each row has an entry in the column unique ID. This number is unique to that row, meaning that no other row has that number.



Additionally, each row has an entry for transaction ID. This number is not necessarily unique since one transaction can have multiple products. For example, unique ID equals 15 could represent the purchase of an extra large fountain drink as part of transaction ID 20189999434333452532. While most transactions are for only one product, it is possible that this same transaction also included the purchase of an apple. Of course, the apple would be unique ID equals 16, since each product for each transaction has its own unique ID.

5 Categories in the Data Set

unique_id	transaction_id	unformatted_date	customer_id	product_id	product_name	category_id	category_name
0	20180106 961 1 100 2656161	2019-01-09	NA	18977	G87E10M	1	Fuel Premium
1	20170218 991 2 1 1894895	2017-05-14	6426302	6263	COKE CHRY 20OZ	250	Pk Bev-csd/pop (71)
2	20190530 1959 1 1 2043460	2019-08-22	NA	8830	MIKES HRDR BLK CHRY 16OZ CAN	231	Flavored Malt
3	20170207 219 3 1 1630288	2017-05-03	NA	4546	HAMMS 6PK CAN	210	Beer-budget (44)
4	20170508 916 2 1 154523	2017-08-01	NA	530	DT PEPSI .5LT 6PK	250	Pk Bev-csd/pop (71)
5	20190418 981 2 1 3117936	2019-07-12	NA	3823	NUT HAR PISTACHIOS 2.25OZ	55	Nuts & seeds
6	20170819 966 7 3 1815282	2017-11-12	NA	1738	BUD LT 18PK CAN	196	Beer-premium (42)
7	20180328 184 2 2 145024	2018-06-21	7364740	10989	TRKY SOURDOUGH	274	Cold Food
8	20171224 834 3 1 3141922	2018-03-19	NA	3738	LAYS CLASSIC 2.75OZ	52	Potato Chips

There's dozens of features or columns in the data. We won't describe them all here, but they can be grouped into five categories. We'll describe the five categories and the main columns or features in each category.

Transaction and Customer Groups

Transaction group includes unique identifier, transaction number, date

unique_id	transaction_id	unformatted_date
0	20180106 961 1 100 2656161	2019-01-09
1	20170218 991 2 1 1894895	2017-05-14
2	20190530 1959 1 1 2043460	2019-08-22
3	20170207 219 3 1 1630288	2017-05-03
4	20170508 916 2 1 154523	2017-08-01
5	20190418 981 2 1 3117936	2019-07-12
6	20170819 966 7 3 1815282	2017-11-12
7	20180328 184 2 2 145024	2018-06-21
8	20171224 834 3 1 3141922	2018-03-19

First, the first group of columns is the transaction group. These columns include the unique identifier for the row, the transaction number, and the date of the transaction.



Transaction and Customer Groups

Transaction group includes unique identifier, transaction number, date

Customer group identifies customer when known

unique_id	transaction_id	unformatted_date	customer_id
0	20181016[961]1 100 2656161	2019-01-09	NA
1	20170218[991]2 1 1894895	2017-05-14	6426.302
2	20190530[1959]1 1 2043460	2019-08-22	NA
3	20170207[193]1 1 630288	2017-05-03	NA
4	20170508[16]2 1 1154523	2017-08-01	NA
5	20190418[981]2 1 3117936	2019-07-12	NA
6	20170619[966]7 3 1815282	2017-11-12	NA
7	20180328[184]2 2 145024	2018-06-21	7364.740
8	20171224[834]3 1 3141922	2018-03-19	NA

Second, the next group of columns is the customer group. Actually this is not a group of columns, but only one column called Customer ID. This column identifies the customer making the purchase when that customer is known. Customers are only known when they are loyalty reward members of the company and made their purchase with a loyalty rewards number or card.

Product and Category Group

Identifies product and name

Hierarchy of product categories

product_id	product_name	category_id	category_name
18977	G87E10M	1	Fuel Premium
6263	COKE CHRY 20OZ	250	Pk Bev-csd/pop (71)
8830	MIKES HRDR BLK CHRY 16OZ CAN	231	Flavored Malt
4546	HAMMIS 6PK CAN	210	Beer-budget (44)
530	DT PEPSI .5LT 6PK	250	Pk Bev-csd/pop (71)
3823	NUT HAR PISTACHIOS 2.25OZ	55	Nuts & seeds
1738	BUD LT 18PK CAN	196	Beer-premium (42)
10989	TRKY SOURDOUGH	274	Cold Food
3738	LAYS CLASSIC 2.75OZ	52	Potato Chips

The third group is the product and category group. This group of columns identifies the product and its name. It also identifies the type of category of product. All products are a part of a hierarchy of product categories. Thus, each product belongs to a category and that category has a parent category or a superset of that category.

Site Group and Revenue and Profit Group

Site group:
name, address,
longitude/
latitude

Revenue and
profit for
product sold

site_id	site_name	address	city	zip	latitude	longitude	site_status
477	975 Boulder	1905 28Th St	Boulder	80301	40.0201	-105.2600	ACTIVE
137	368 Harrisburg	306 State St	Harrisburg	69345	41.5565	-103.7410	ACTIVE
140	370 Dunning	403 Jewett Ave	Dunning	68833	41.8281	-100.1050	ACTIVE
434	914 Centennial	2221 E Arapahoe Rd	Centennial	80122	39.5957	-104.9630	ACTIVE
452	939 Campo	183 W 4Th St	Campo	81029	37.1055	-102.5800	ACTIVE
410	878 Forgan	115 S Broadway St	Forgan	73938	36.9063	-100.5350	ACTIVE
157	387 Big Flat	100 Community Dr	Big Flat	72617	36.0018	-92.4051	ACTIVE

Next, the fourth group is the site group. These columns identify the convenience store where the purchase took place, including the site name, the address of the site, and the longitude and latitude. Please know, however, that to obscure the actual company using this dataset, the addresses are masked by referring to post offices instead of the actual convenience stores themselves. Nevertheless, the addresses are close to the convenience store. Finally, five, the last group of columns, present the amount of revenue and profit for the product that was sold. These columns are critical to all of our analysis going forward since TECA is a business and is interested in making money.

Lesson 2-3: Logistic Regression

Lesson 2-3.1 Introduction to Logistic Regression



Our hypothetical company TECA wants to be able to predict whether an individual purchase is going to be made by a loyalty customer or a regular non loyalty customer. Loyalty customers are those who applied for a loyalty card with TECA in order to get discounts for purchases. TECA has learned that they make about two \$100 per new loyalty customer, and that loyalty customers buy more profitable products on average than non loyalty customers do. By predicting whether a sale is made by a loyalty or non loyalty customer take a hopes to increase the number of loyalty customers and the number of products that they buy.

**Target variable or
dependent variable
is “loyalty” and it
is a categorical
variable.**



While there may be many ways to investigate this issue, TECA has approached the problem by labeling each of their purchases as being made by a loyalty customer labeled as loyal or made by a non loyalty customer labeled as not loyal. Thus the targets or dependent variable here is loyalty and it's a categorical variable.

Independent Variables:

Category:

Fuel

Pop (587)

Cold Dispensed Beverage

Juice/Tonics



The other features, or independent variables that will be used to predict the target feature are the following category. This variable list, the top 20 most frequently purchased products and they are the following fuel, which is gasoline purchased at the gas pumps, pop, which is bottled and canned soda, cold dispensed beverage, which is fountain soda. Next juice and tonics, these are bottled juices,

Independent Variables:

Category:

- Lottery
- Energy
- Candy/Gum
- Hot Dispensed Beverage



lottery, these are lottery tickets, both those purchased and paid out if it's a winning ticket. Energy, these are bottled and canned energy drinks. Next candy and gum just self explanatory, hot dispense to beverage, these is coffee, tea and hot chocolate from a dispenser.

Independent Variables:

Category:

- Beer
- Salty Snacks
- Cigarettes
- Pizza



Next beer and then salty snacks, these include peanuts and sunflower seeds, for example. Cigarettes, pizza,

Independent Variables:

Category:

Roller Grill

Bread & Cakes

Breakfast Sandwiches

Bakery



roller grill, these items are hot dogs and burritos that are kept warm on a roller grill. Breads and cakes, breakfast sandwiches, these refreshing take and heat breakfast sandwiches. Then bakery, these are donuts and cupcakes.

Independent Variables:

Category:

Smokeless

Hot Sandwiches & Chicken

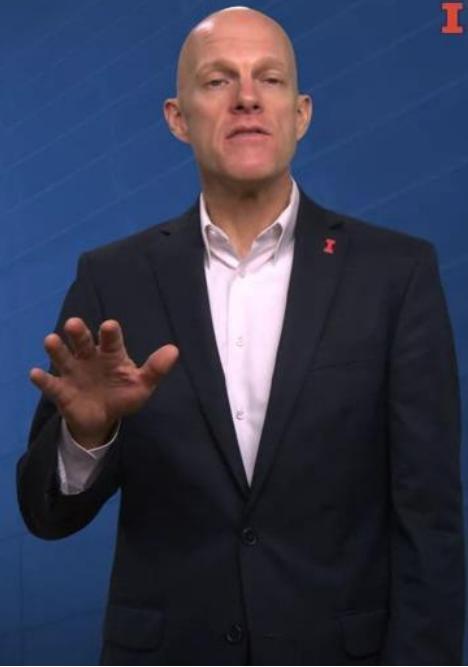
Cigars

Chips

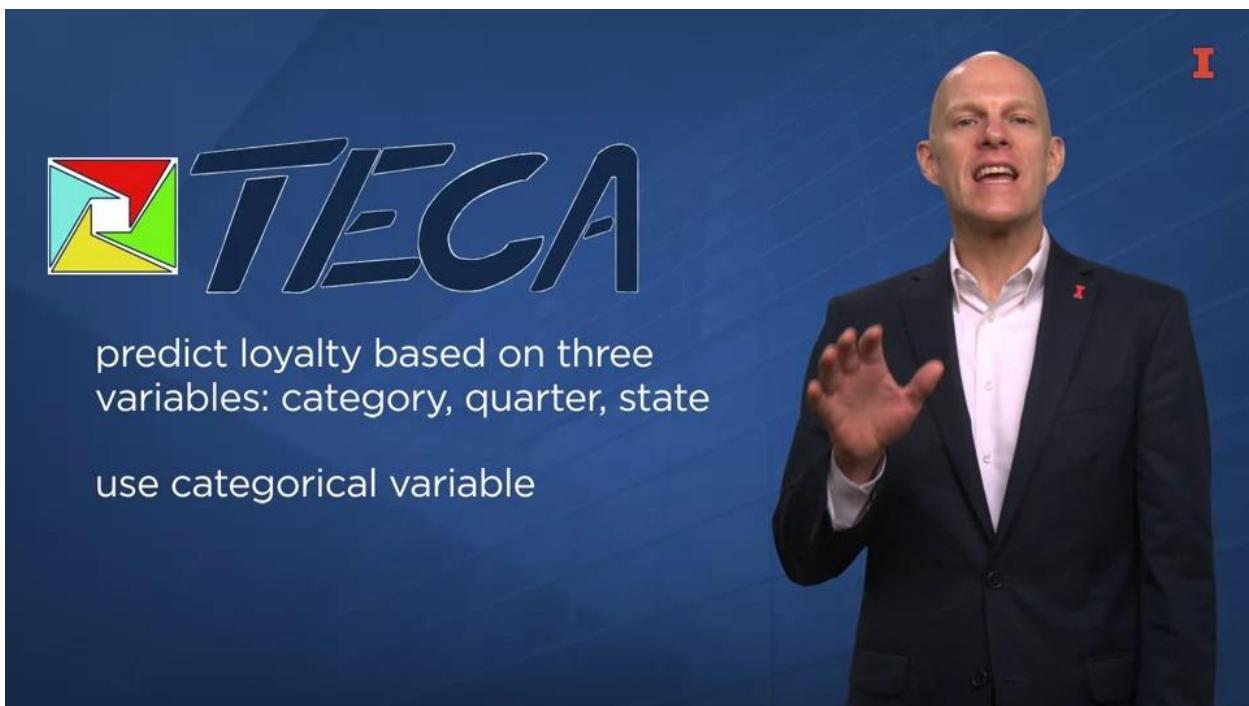


Another one is smokeless these, this is chewing tobacco and then hot sandwiches and chicken, these are ready to purchase sandwiches and chicken cigars and then chips. Here's a snapshot of the data that you can see.

	loyalty	category	quarter	state
1	loyal	Bread & Cakes	3	Colorado
2	loyal	Candy/Gum	1	Wyoming
3	loyal	Pop (587)	3	Wyoming
4	loyal	Pop (587)	4	Colorado
5	loyal	Chips	1	Oklahoma
6	loyal	Juice/tonics	2	Colorado
7	loyal	Juice/tonics	3	Wyoming
8	loyal	Bread & Cakes	2	Alabama
9	loyal	Beer	3	Alabama
10	loyal	Chips	2	Oklahoma
11	loyal	Cigarettes	1	Nebraska
12	loyal	Roller Grill	4	Arkansas
13	loyal	Pop (587)	4	Iowa
14	loyal	Salty Snacks	2	Alabama
15	loyal	Pizza	1	Nebraska
16	loyal	Salty Snacks	3	Colorado
17	loyal	Cigarettes	2	Missouri
18	loyal	Roller Grill	3	Wyoming
19	loyal	Fuel	3	Wyoming



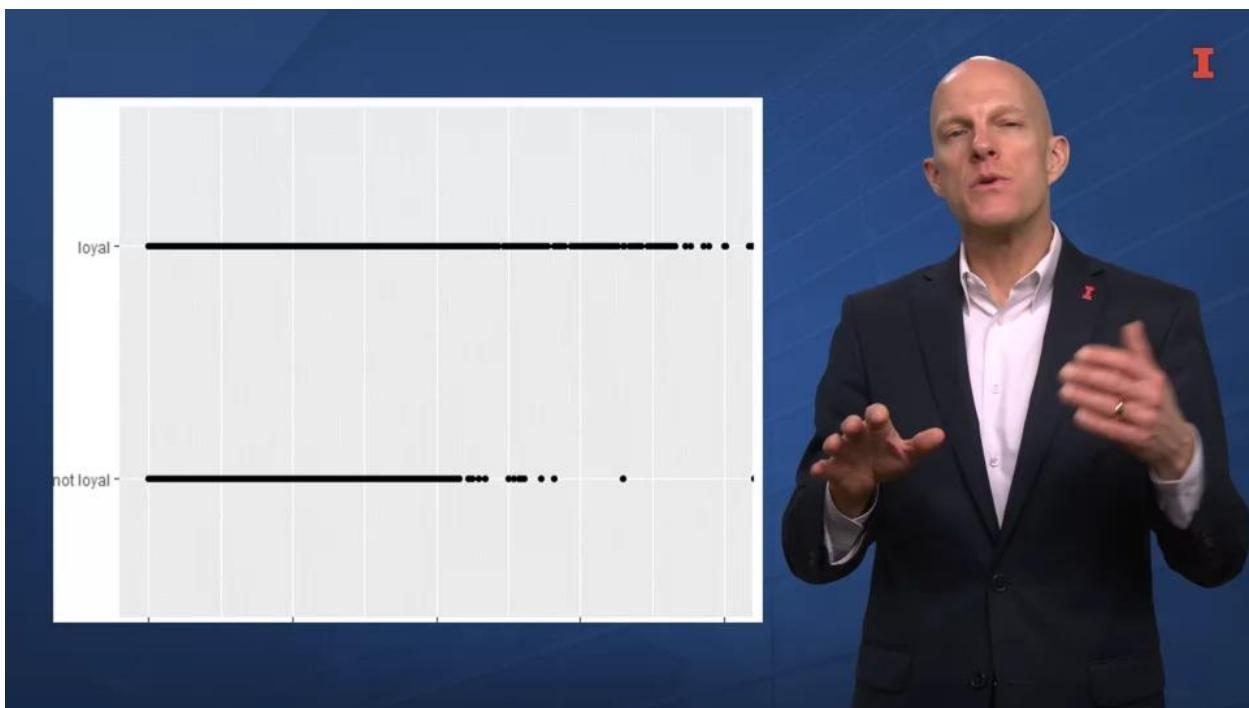
This data has been significantly cleaned and manipulated for use in this module, null values and outliers have been removed and categories have been combined and eliminated. Finally, the data has been adjusted so that the number of loyalty purchases and non-loyalty purchases are relatively similar.



predict loyalty based on three variables: category, quarter, state

use categorical variable

Now take a wants to predict loyalty based upon the three variables, category, Quarterl, and state. This feels like a solution that regression could handle, certainly regression examines the effect of independent variables on dependent variables and predicts an outcome. Unfortunately there is one major difference between a regression problem and our problem, that is regression predicts a continuous dependent variable. While we need to predict a categorical variable



specifically, as can be seen here, we have binary data that is all of our data lies at two levels loyal and not loyal. Regression would pick a regression trend line that gives a continuous response and would fall outside of the range of our dependent measure. Rather than that we need a function that converts the independent variables into an expression of probability between zero and one in relation to the dependent variable which would be not loyal and loyal. That is we need a function that gives outputs between zero and one that finds coefficients to maximize the likelihood of predicting a high probability for observations that actually belong to class one, which in our case is loyalty. And predicting a low probability for observations that actually belong to class zero, which in our case is not loyal. This will then produce an S shaped curve that can convert any number and map it into a numerical value between zero and 1.

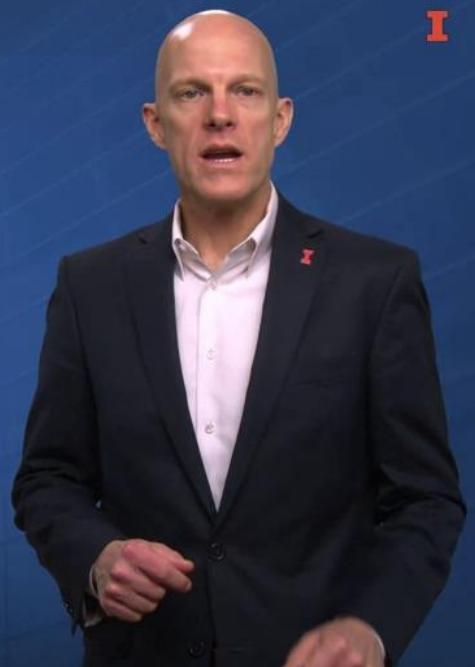
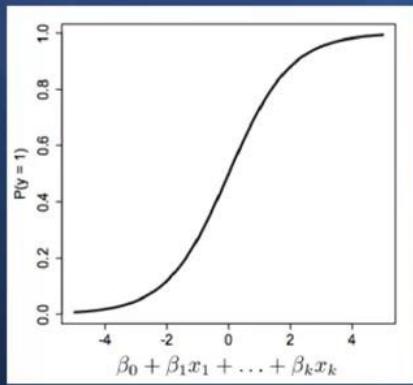
Logistic function or sigmoid function

$$P(y = 1) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_k x_k)}}$$



Now luckily there are many functions that can do this for logistic aggression will pick the logistic function or the sigmoid function. Here's a picture of a multi variant version of that,

Logistic function or sigmoid function



this function provides a nice S curve that gives us precisely what we're looking for. Thus this function models the probability that why the dependent variable belongs to a particular category given the x i the independent variables.



In other words, based upon the found probabilities of the independent variables, logistic regression assigns each data point to a discrete class, in our case, loyal and not loyal. Unlike the ridge regression prediction trendline, the logistic hyper plane represents a classification decision boundary that divides the data set into classes. Finally, the estimation of coefficients of this model is done via maximum likelihood. In general terms you use maximum likelihood to find the intercept and coefficient of X such that the predicted probability of Y is as close to possible to 1 for true y equals 1, and zero for true y equals zero. In summary, as it's probably clear by now, logistic regression is a classification machine learning algorithm. It's useful for prediction and exploration of categorical target variables. It's just one of many different classification algorithms.

[Lesson 2-3.2 Logistic Regression Hands on - One Variable \(Part1\)](#)



In this video, we'll use the algorithm to predict whether a transaction is from a loyalty customer or not. We'll use the TEKA data set. TEKA desires to increase their loyalty customers and the number of products a loyalty customer buys because loyalty customers create more business, and more profitable business for TEKA. As always, we encourage you to resist the urge to just watch these videos. Rather, please follow along doing each of the steps that we're doing so that you can get the critical hands-on experience necessary to learn this valuable tool. First, let's go ahead and bring in the data-sets and the needed packages.

```

Classification-type machine learning algorithm. In this video, we will use the algorithm to predict whether a transaction is
from a loyalty customer or not.
12 We will use the TECA dataset.
13
14 TECA desires to increase their loyalty customers and the number of products a loyalty customer buys, because loyalty customers
create more business and more profitable business for TECA.
15
16 As always, we encourage you to resist the urge to just watch these videos. Rather, please follow along, doing each of the steps
we are doing, so that you can get the critical hands-on experience necessary to learn this valuable tool.
17 First, let's bring in the data and the needed packages.
18
19+ # Bring in packages and data
20+ ---[r] read_rdsfile()
21 library(tidyverse)
22 df1_even_kp <- read_rds('logistic1.rds')
23+

```

Registered S3 methods overwritten by 'dbplyr':

method	from
print.tbl_lazy	
print.tbl_sql	

-- Attaching packages --

v ggplot2	3.3.3	v purrr	0.3.4
v tibble	3.0.4	v dplyr	1.0.2
v tidyr	1.1.2	v stringr	1.4.0
v readr	1.4.0	v forcats	0.5.0

-- Conflicts --

✓ dplyr::filter()	masks stats::filter()
✓ dplyr::lag()	masks stats::lag()

tidyverse 1.3.0 --

tidyverse_conflicts() --

24
25 Next, run the following lines of code. In these lines I have created a very basic function that will help us evaluate the
quality of our logistic regression model, later on.
26
27 we will use this code a few times, so, it made sense to put it into a function that can be reused, without having to cut and
paste all of this code every time we want to use it.
28
29+ # Create function for confusion matrix, to be used later
30+ ---[r]
31 # Make reusable Confusion Matrix function
32 my_confusion_matrix <- function(cf_table) {
33 true_positive <- cf_table[4]

I'll do that here. The package we're going to use is tidy verse. We put that in the library function. If you don't have the tidy verse package, you'll need to download that yourself. Then we bring in a file that we'll call df1_even_kp. We'll use the read_rds function and the rds file that we provide for you as logistics one.



Next, let's run the following lines of code. In these lines, I've created a very basic function that will help us evaluate the quality of our logistic regression model later on. We'll use this code several times so it makes sense to put it into a function that can be re-used without having to cut and paste all the code every time that we want to use it.

```

26
27 We will use this code a few times, so, it made sense to put it into a function that can be reused, without having to cut and
28 paste all of this code every time we want to use it.
29 # Create function for confusion matrix, to be used later
30 ...{r}
31 # Make reusable Confusion Matrix function
32 my_confusion_matrix <- function(cf_table) {
33   true_positive <- cf_table[4]
34   tp_rate <- cf_table[4]/(cf_table[3]+cf_table[2]+cf_table[1])
35   true_negative <- cf_table[1]
36   tn_rate <- cf_table[1]/(cf_table[3]+cf_table[2]+cf_table[4])
37   false_positive <- cf_table[2]
38   fp_rate <- cf_table[2]/(cf_table[3]+cf_table[4]+cf_table[1])
39   false_negative <- cf_table[3]
40   fn_rate <- cf_table[3]/(cf_table[4]+cf_table[2]+cf_table[1])
41   accuracy <- (true_positive + true_negative) / (true_positive + true_negative + false_positive + false_negative)
42   sensitivity_recall <- true_positive / (true_positive + false_negative)
43   specificity_selectivity <- true_negative / (true_negative + false_positive)
44   precision <- true_positive / (true_positive + false_positive)
45   neg_pred_value <- true_negative/(true_negative + false_negative)
46   print(cf_table)
47   my_list <- list(sprintf("%1.4f (%1.4f) = True Positive (TP, Hit", true_positive, tp_rate),
48     sprintf("%1.4f (%1.4f) = True Negative (TN, Rejection", true_negative, tn_rate),
49     sprintf("%1.4f (%1.4f) = False Positive (FP), Type 1 Error", false_positive, fp_rate),
50     sprintf("%1.4f (%1.4f) = False Negative (FN), Type 2 Error", false_negative, fn_rate),
51     sprintf("%1.4f = Accuracy (TP+TN/(TP+TN+FP+FN))", accuracy),
52     sprintf("%1.4f = Sensitivity, Recall, Hit Rate, True Positive Rate (How many positives did the model get
right? TP/(TP+FN))", sensitivity_recall),
53     sprintf("%1.4f = Specificity, Selectivity, True Negative Rate (How many negatives did the model get right?
TN/(TN+FP))", specificity_selectivity),
54     sprintf("%1.4f = Precision, Positive Predictive Value (How good are the model's positive predictions?
TP/(TP+FP))", precision),
55     sprintf("%1.4f = Negative Predictive Value (How good are the model's negative predictions? TN/(TN+FN)",
neg_pred_value)
56   )
57   return(my_list)
58 }
59 ...
60
61 Now, let's go through our six steps of for using a machine learning algorithm. Recall that these steps are the following:
62 1. Data scrubbing

```

I won't describe this here, but you'll need to go ahead and run it.



Next, let's go through our six steps for using a machine learning algorithm. Recall that these steps are the following: one, data scrubbing; two, algorithms selection; three, model training or model fitting; four, model evaluation; and five, model improvement with; six, deploying the model. We've done all the data scrubbing already for you so number 1 is already complete. For number 2, we selected logistic regression as our classification algorithm so let's move on now to Number 3. We'll break this down into multiple parts. Our goal is to train a univariate logistic regression model using our data. That is, we'll predict whether a particular transaction is from a loyalty customer or not.

```
75
76 # View data
77 ````{r}
78 slice_sample(df1_even_kp, n=10)
79 ````
```

	loyalty	category	quarter	state
961903	not loyal	Cigarettes	3	Iowa
878681	not loyal	Cigars	2	Nebraska
1243339	not loyal	Hot Dispensed Beverage	3	Alabama
443547	loyal	Cold Dispensed Beverage	3	Alabama
989743	not loyal	Hot Dispensed Beverage	1	Alabama
771964	not loyal	Energy	2	Alabama
88842	loyal	Bakery	2	Alabama
1561530	not loyal	Cold Dispensed Beverage	1	Missouri
1015308	not loyal	Fuel	1	Oklahoma
1041006	not loyal	Fuel	1	Colorado

Let's look at the data to remind ourselves what it looks like. To do that, let's run this slice_sample function. Just to get a sense of what the data is, we'll pass n equals to 10 to the function so that we can view and visualize just the first 10 rows. Loyalty, this column right here, this first column, that's our dependence or output or target variable. I'll use those terms interchangeably. Then we need to pick one of the other three, category, quarter, or state, to be the independent variable.



Since TEKA is concerned with predicting what transactions will be loyalty transactions, it probably makes the most sense to use category as our independent variable. Now, before we train our model, we have a few things to do to get ready. First, let's look at the baseline percentage of the occurrence of loyalty. We'll use the function table to do that. This function uses the levels of a categorical variable to build a contingency table of the counts at each combination of variable levels. We'd like to build a table of how many loyal and non loyal transactions that we have and then calculate a percentage of loyal transactions. Let's describe what's being done here. First, the table function is used to create a table from the loyalty variable. Second, we print the table. Next, we print the percentage of loyal transactions divided by the total transactions. This is done by indexing the table and using indexing to select individual parts of the table with the square brackets. That's for example, loyalty Table 2 is 519744.

```

83 Before we train our model we have a few more things to do to get ready.
84
85 First, let's look at the baseline percentage of the occurrence of `loyalty`. First, we will use the function `table()`. This
86 function uses the levels of categorical variables to build a contingency table of the counts at each combination of variable
87 levels. We would like to build a table of how many loyal and no loyal transactions we have and then calculate a percentage of
88 loyal transactions.
89
90 Let's describe what is done here. First, the `table()` function is used to create a table from the `loyalty` variable. Second,
91 we printed that table. Next, we printed the percentage of loyal transactions divided by the total transactions. This is done by
92 indexing the table and using indexing to select individual parts of the table with the square brackets `[]` . Thus, for example,
93 `loyal_table[2]` = 519744.
94

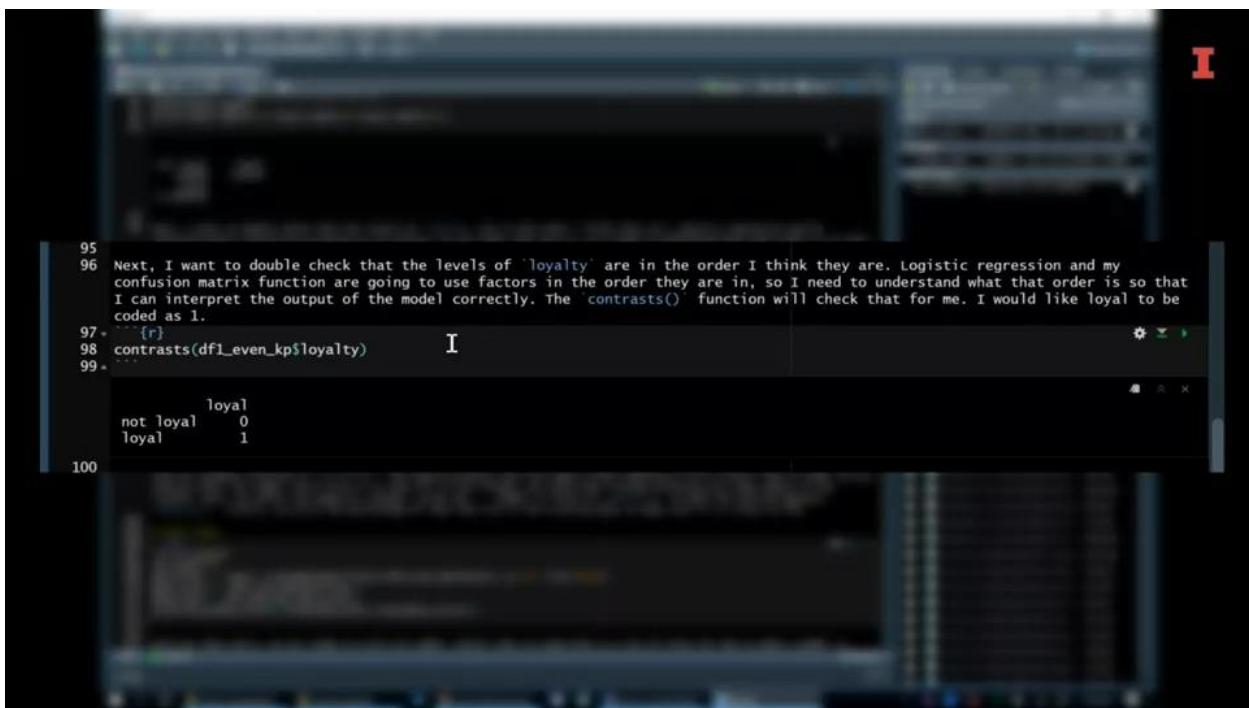
95 # Baseline occurrence of `loyalty`
96 ## (r)
97 loyal_table <- table(df1_even_kp$loyalty)
98 print(loyal_table)
99 print@{loyal_table[2]/(loyal_table[1]+loyal_table[2])}
100

101 not loyal      loyal
102             520000     519744
103             0.4998769

104
105 # Split data
106 ## (r)
107 contrasts(df1_even_kp$loyalty)
108
109
110 This is correct, so we can move on to our next step. The next step in training our data is to split it into training and testing
111 proportions. Recall, that we want to train our model on one set of data, but then make sure it works on other data, so that we
112 are not overfitting our model. I will use the `caret` package to split the data, though there are myriad ways of doing this.
113 However, as you continue to explore and work on classification models it is likely you will use the `caret` package.
114
115 The first line sets a seed so we can get the same split again. The second line uses #using the `caret` package to select 75% of
116 the entries in the column `loyalty` - we could have used a different percentage-like 80% or 70%. This creates a matrix of
117 rows and columns. We can use to select the rows of the dataset that we are using for training data. This is done in the third
118 line where we create a dataset called `data_train` by taking the original dataset and telling it to only keep the
119 rows we randomly selected in `partition` . The square brackets here are again using indexing to tell R which rows to take. We put
120 nothing after the comma, which tells R to take all of the columns. We could have instead selected only some subset of the
121 columns. Next, we select the opposite columns, using the ` -` symbol in front of `partition` to make the testing dataset,
122 `data_test` . Finally, we print the percentage of rows that are in the training data to make sure it is close to 75%.
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
617
618
619
619
620
621
622
623
624
625
626
627
627
628
629
629
630
631
632
633
634
635
635
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
```



Overall, loyal transactions happen about 49, almost 50 percent of the time, as you can see from this number here. Next, I want to double-check that the levels of loyalty are in the order I think they are. Logistic regression and the confusion matrix that I created, that first thing we ran way before, is going to use factors in the order that they're in, in that variable. I need to understand what the correct order is so that I can interpret the output of the model correctly. The contrast function will check that for me. I'd like to have loyal be coded as one.



A screenshot of a computer monitor displaying an RStudio interface. The code in the console window is:

```
95 Next, I want to double check that the levels of 'loyalty' are in the order I think they are. Logistic regression and my
96 confusion matrix function are going to use factors in the order they are in, so I need to understand what that order is so that
97 I can interpret the output of the model correctly. The contrasts() function will check that for me. I would like loyal to be
98 coded as 1.
99
100
```

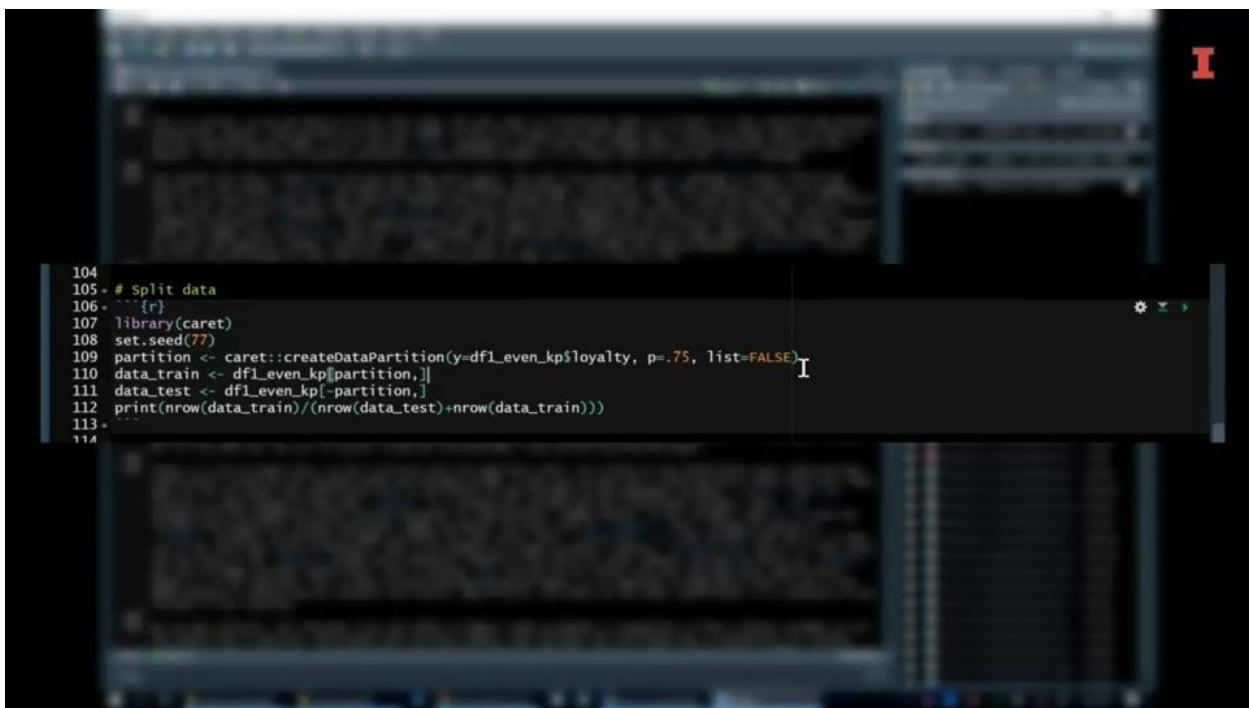
The output window shows the result of the `contrasts` function:

	loyal
not loyal	0
loyal	1

Let's go ahead and run that. Again, I'm simply passing the one column, loyalty column, in that contrast table. Sure enough, that's correct. I can see that zero is not loyal and one is equal to loyal. We can move on to our next step.



The next step in training our data is to split it into training and testing proportions. Recall that we want to train our model on one set of data but then make sure it works on the other data so that we're not over feeding our model. I'll use the caret package to split the data though there are a million of ways of doing this. However, as you continue to explore and work on classification models, it is likely that you'll probably use the caret package in the future.



A screenshot of an RStudio interface. The top bar shows the title 'Machine Learning Algorithms with R in Business Analytics' and the professor's names. The main area is a code editor with the following R code:

```
104 # Split data
105 ````{r}
106 library(caret)
107 set.seed(77)
108 partition <- caret::createDataPartition(y=df1_even_kp$loyalty, p=.75, list=FALSE)
109 data_train <- df1_even_kp[partition,]
110 data_test <- df1_even_kp[-partition,]
111 print(nrow(data_train)/(nrow(data_test)+nrow(data_train)))
112
113 ````
```

This first line here just brings in the caret package using the library function. This next line uses set.seed and then just a number I put in there, 77, to create a seed so that in the third line, the create data partition is going to use the same seed or same starting point every time so we get the same split of data every time. This line then uses the caret package to select 75 percent of the entries in the column loyalty. We could have used a different percentage. We could have used 80 percent, we could have used 70 percent. Let's go ahead and go with 75. This, then creates a matrix of numbers that we can use to select the rows from our data set that will become the training data. That is what we've done in the fourth line here. Here, we create the data set called data train by selecting the original data set and telling it to only keep the rows we randomly selected in the partition that we created right above. The square brackets here are again using indexing to tell R which rows to take, which rows to use. We put nothing after the comma right here which tells R to take all of the columns. Just the rows from partition, but all of the columns. We could have instead selected only some subset of the columns. Next, we select the opposite columns in data test using this minus or dashed symbol in front of partition to make the testing data set, data_test. Finally, we print the percentage of rows that are in the training data just to make sure that it's really close to 75 percent as we intended. We run that, and sure enough we have 75 percent here.

[Lesson 2-3.3 Logistic Regression Hands on - One Variable \(Part2\)](#)



With our Data Split, we're ready to train our model. Recall that an algorithm is a set of rules for how to make a model. A model takes our data and the options we select to create new data, new information, and additional rules for how to use that model. Below we create our model called model train. Then summarize it to view the output, we use the GLM function with the family equals binomial argument to select logistic regression model. The syntax is quite simple here, our dependent variable, loyalty goes first. Then we use the tilde sign, that squiggly line, to act like an equal sign, followed by our independent variable, category in this case. Finally, we list the data set. We are of course using the training data since we're training the model so far.

Let's go ahead and run that and we'll also summarize it with the summary function. I'll take a moment to train that and you can see we're running it and there it is.



There's a lot of data here and let's pick out only the most important things and talk about those.

```

127 ->

call:
glm(formula = loyalty ~ category, family = binomial, data = data_train)

Deviance Residuals:
    Min      1Q  Median      3Q     Max 
-1.5152 -1.1350 -0.8436  1.1812   1.5530 

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept)  0.30575  0.01539 19.870 < 2e-16 ***
categoryBeer -1.15590  0.02101 -55.009 < 2e-16 ***
categoryBread & Cakes -0.11000  0.02173 -5.063 4.14e-07 ***
categoryBreakfast Sandwiches 0.22179  0.02156 10.288 < 2e-16 ***
categoryCandy/Gum -0.33491  0.01816 -18.442 < 2e-16 ***
categoryChips -0.20758  0.02392 -8.677 < 2e-16 ***
categoryCigarettes -0.48665  0.01769 -27.517 < 2e-16 ***
categoryCigars -0.95134  0.02544 -37.395 < 2e-16 ***
categoryCold Dispensed Beverage 0.46045  0.01706 26.991 < 2e-16 ***
categoryEnergy 0.02773  0.01797  1.544 0.12269    
categoryFuel -0.63600  0.01622 -39.211 < 2e-16 ***
categoryHot Dispensed Beverage -0.01130  0.01866 -0.605 0.54487  
categoryHot Sandwiches & Chicken -0.06145  0.02298 -2.674 0.00749 ** 
categoryJuice/tonics -0.40630  0.01752 -23.196 < 2e-16 ***
categoryLottery -0.91991  0.01865 -49.313 < 2e-16 ***
categoryPizza 0.10643  0.02130  4.996 5.84e-07 ***
categoryPop (587) -0.31478  0.01706 -18.448 < 2e-16 ***
categoryRoller Grill -0.18406  0.02183 -8.430 < 2e-16 ***
categorySalty Snacks -0.28467  0.01997 -14.252 < 2e-16 ***
categorySmokeless (951) -0.67115  0.02315 -28.989 < 2e-16 *** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1081043  on 779807  degrees of freedom
Residual deviance: 1050668  on 779788  degrees of freedom
AIC: 1050708

Number of Fisher Scoring iterations: 4

```

We'll focus first on the coefficients area. There we have coefficients listed for our variable under the estimate column, right here. Here's the coefficients for each of those category levels.



Away over to the right here we have the z-values and the p-values, here and here. What does all this mean? The coefficients tells us the effect of that particular variable or level of variable on the dependent variable which is loyalty. The GLM function is smart enough to know that category is a categorical variable and not a continuous variable. Thus, the function automatically did something called one-hot encoding. Equivalently, we could call that dummy coding. It did that to category all on its own. That is, it took the variable and essentially created individual dummy variables for each level of category. You can see those here. Thus, instead of having one category variable, we now have 19 dummy variables for each level of category or almost each level. Thus, category beer is a new variable that has a one if that row is as a beer purchase and a zero otherwise. That's an example of a dummy variable. Likewise, category pizza, for example, has a one if that level row has pizza in the category column and a 0 otherwise.

```

127 ->

call:
glm(formula = loyalty ~ category, family = binomial, data = data_train)

Deviance Residuals:
    Min      1Q  Median      3Q     Max 
-1.5152 -1.1350 -0.8436  1.1812  1.5530 

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) 0.30575  0.01539 19.870 < 2e-16 ***
categoryBeer -1.15590  0.02101 -55.009 < 2e-16 ***
categoryBread & Cakes -0.11000  0.02173 -5.063 4.14e-07 ***
categoryBreakfast Sandwiches 0.22179  0.02156 10.288 < 2e-16 ***
categoryCandy/Gum -0.33491  0.01816 -18.442 < 2e-16 ***
categoryChips -0.20758  0.02392 -8.677 < 2e-16 ***
categoryCigarettes -0.48665  0.01769 -27.517 < 2e-16 ***
categoryCigars -0.95134  0.02544 -37.395 < 2e-16 ***
categoryCold Dispensed Beverage 0.46045  0.01706 26.991 < 2e-16 ***
categoryEnergy 0.02773  0.01797  1.544 0.12269  
categoryFuel -0.63600  0.01622 -39.211 < 2e-16 ***
categoryHot Dispensed Beverage -0.01130  0.01866 -0.605 0.54487  
categoryHot Sandwiches & Chicken -0.06145  0.02298 -2.674 0.00749 ** 
categoryJuice/tonics -0.40630  0.01752 -23.196 < 2e-16 ***
categoryLottery -0.91991  0.01865 -49.313 < 2e-16 ***
categoryPizza 0.10643  0.02130  4.996 5.84e-07 *** 
categoryPop (587) -0.31478  0.01706 -18.448 < 2e-16 ***
categoryRoller Grill -0.18406  0.02183 -8.430 < 2e-16 ***
categorySalty Snacks -0.28467  0.01997 -14.252 < 2e-16 ***
categorySmokeless (951) -0.67115  0.02315 -28.989 < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1081043  on 779807  degrees of freedom
Residual deviance: 1050668  on 779788  degrees of freedom
AIC: 1050708

Number of Fisher Scoring iterations: 4

```

Now hopefully you noticed an issue. We have 20 levels of category, if you remember back from above, but we only have 19 dummy variables here. That's because GLM smartly puts one of the levels of our category variable in the intercept term, the intercept right here.



Specifically, bakery is in the intercept. This is done to avoid over specifying the model. It's important to know which level is not in the model and which is in the intercept because it affects how we interpret the results. Specifically, the effect of the other coefficients is in reference to the variable that's in the intercept which in this case is bakery. We can now interpret. The coefficient on intercept here lists the effect of bakery items on whether a transaction is from a loyalty customer or not.

```

127 ->

call:
glm(formula = loyalty ~ category, family = binomial, data = data_train)

Deviance Residuals:
    Min      1Q  Median      3Q     Max 
-1.5152 -1.1350 -0.8436  1.1812  1.5530 

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) 0.30575   0.01539 19.870 < 2e-16 ***
categoryBeer -1.15590   0.02101 -55.009 < 2e-16 ***
categoryBread & Cakes -0.11000   0.02173 -5.063 4.14e-07 ***
categoryBreakfast Sandwiches 0.22179   0.02156 10.288 < 2e-16 ***
categoryCandy/Gum -0.33491   0.01816 -18.442 < 2e-16 ***
categoryChips -0.20758   0.02392 -8.677 < 2e-16 ***
categoryCigarettes -0.48665   0.01769 -27.517 < 2e-16 ***
categoryCigars -0.95134   0.02544 -37.395 < 2e-16 ***
categoryCold Dispensed Beverage 0.46045   0.01706 26.991 < 2e-16 ***
categoryEnergy 0.02773   0.01797  1.544 0.12269  
categoryFuel -0.63600   0.01622 -39.211 < 2e-16 ***
categoryHot Dispensed Beverage -0.01130   0.01866 -0.605 0.54487  
categoryHot Sandwiches & Chicken -0.06145   0.02298 -2.674 0.00749 ** 
categoryJuice/tonics -0.40630   0.01752 -23.196 < 2e-16 ***
categoryLottery -0.91991   0.01865 -49.313 < 2e-16 ***
categoryPizza 0.10643   0.02130  4.996 5.84e-07 *** 
categoryPop (587) -0.31478   0.01706 -18.448 < 2e-16 ***
categoryRoller Grill -0.18406   0.02183 -8.430 < 2e-16 ***
categorySalty Snacks -0.28467   0.01997 -14.252 < 2e-16 ***
categorySmokeless (951) -0.67115   0.02315 -28.989 < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1081043  on 779807  degrees of freedom
Residual deviance: 1050668  on 779788  degrees of freedom
AIC: 1050708

Number of Fisher Scoring iterations: 4

```

The coefficient here is positive, which means that selling a bakery item increases the likelihood that a transaction is a loyalty transaction.



More precisely, purchasing this item increases the log odds of a loyalty purchase happening by 0.30575. Now we won't go into log odds, but it suffices to say that buying a bakery item increases the likelihood of a transaction being from a loyalty customer. Let's look at some more of our dummy variable coefficients here.

```

127 -> call:
glm(formula = loyalty ~ category, family = binomial, data = data_train)

Deviance Residuals:
    Min      1Q  Median      3Q     Max 
-1.5152 -1.1350 -0.8436  1.1812   1.5530 

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) 0.30575  0.01539 19.870 < 2e-16 ***
categoryBeer -1.15590  0.02101 -55.009 < 2e-16 ***
categoryBread & Cakes -0.11000  0.02173 -5.063 4.14e-07 ***
categoryBreakfast Sandwiches 0.22179  0.02156 10.288 < 2e-16 ***
categoryCandy/Gum -0.33491  0.01816 -18.442 < 2e-16 ***
categoryChips -0.20758  0.02392 -8.677 < 2e-16 ***
categoryCigarettes -0.48665  0.01769 -27.517 < 2e-16 ***
categoryCigars -0.95134  0.02544 -37.395 < 2e-16 ***
categoryCold Dispensed Beverage 0.46045  0.01706 26.991 < 2e-16 ***
categoryEnergy 0.02773  0.01797  1.544 0.12269  
categoryFuel -0.63600  0.01622 -39.211 < 2e-16 ***
categoryHot Dispensed Beverage -0.21130  0.01866 -0.605 0.54487  
categoryHot Sandwiches & Chicken -0.56145  0.02298 -2.674 0.00749 ** 
categoryJuice/tonics -0.40630  0.01752 -23.196 < 2e-16 ***
categoryLottery -0.91991  0.01865 -49.313 < 2e-16 ***
categoryPizza 0.10643  0.02130  4.996 5.84e-07 *** 
categoryPop (587) -0.31478  0.01706 -18.448 < 2e-16 ***
categoryRoller Grill -0.18406  0.02183 -8.430 < 2e-16 ***
categorySalty Snacks -0.28467  0.01997 -14.252 < 2e-16 ***
categorySmokeless (951) -0.67115  0.02315 -28.989 < 2e-16 *** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1081043  on 779807  degrees of freedom
Residual deviance: 1050668  on 779788  degrees of freedom
AIC: 1050708

Number of Fisher Scoring iterations: 4

```

Let's look at fuel for example. Fuel actually decreases the chance of the transaction being from a loyalty customer relative to a bakery item. Again, bakery is in the intercept,



as can be seen by this negative coefficient here. More specifically, the coefficient is still negative even relative to the intercept, which again is bakery. That is, fuels coefficient plus the intercept is still negative if we add those two together. This means that fuel decreases the chance that a loyalty customer makes a purchase. Now this does make sense since bakery item is one that you go into the store for, which is more likely to happen for a loyalty customer, whereas anyone can just drive by and purchase gas. Next, let's look at fountain soda or cold dispensed beverage. This increases the chance the purchase was from a loyalty customer, even more than purchasing a bakery item. Thus, fountain soda seems to be a big draw for our loyalty customers and it could be a way that take I could bring increased loyalty purchases. Again, we noticed that specifically by the positive coefficient,

```

127 ->

call:
glm(formula = loyalty ~ category, family = binomial, data = data_train)

Deviance Residuals:
    Min      1Q  Median      3Q     Max 
-1.5152 -1.1350 -0.8436  1.1812   1.5530 

Coefficients:
                Estimate Std. Error z value Pr(>|z|)    
(Intercept)       0.30575  0.01539 19.870 < 2e-16 ***
categoryBeer     -1.15590  0.02101 -55.009 < 2e-16 ***
categoryBread & Cakes -0.11000  0.02173 -5.063 4.14e-07 ***
categoryBreakfast Sandwiches 0.22179  0.02156 10.288 < 2e-16 ***
categoryCandy/Gum   -0.33491  0.01816 -18.442 < 2e-16 ***
categoryChips        -0.20758  0.02392 -8.677 < 2e-16 ***
categoryCigarettes    -0.48665  0.01769 -27.517 < 2e-16 ***
categoryCigars         -0.95134  0.02544 -37.395 < 2e-16 ***
categoryCold Dispensed Beverage 0.46045  0.01706 26.991 < 2e-16 ***
categoryEnergy        0.02773  0.01797  1.544 0.12269  
categoryFuel          -0.63600  0.01622 -39.211 < 2e-16 ***
categoryHot Dispensed Beverage -0.01130  0.01866 -0.605 0.54487  
categoryHot Sandwiches & Chicken -0.06145  0.02298 -2.674 0.00749 ** 
categoryJuice/tonics    -0.40630  0.01752 -23.196 < 2e-16 ***
categoryLottery        -0.91991  0.01865 -49.313 < 2e-16 ***
categoryPizza          0.10643  0.02130  4.996 5.84e-07 ***
categoryPop (587)      -0.31478  0.01706 -18.448 < 2e-16 ***
categoryRoller Grill   -0.18406  0.02183 -8.430 < 2e-16 ***
categorySalty Snacks    -0.28467  0.01997 -14.252 < 2e-16 ***
categorySmokeless (951) -0.67115  0.02315 -28.989 < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1081043  on 779807  degrees of freedom
Residual deviance: 1050668  on 779788  degrees of freedom
AIC: 1050708

Number of Fisher Scoring iterations: 4

```

the fact that the 0.40645 is even larger than the 0.30575. Finally, let's look at the effect of bread and cakes. What effect does this have? It's a small negative coefficient,



but if we add this negative coefficient, the negative 0.11000, to the intercepts, we see that the difference is actually positive. Again, adding those two together. Thus, bread and cakes has a significant effect on the chance of purchase as a loyalty purchase. Again, we have to compare with the intercept because the bakeries in the intercept and we don't want to over specify our model. Next, it's not enough to see what the sign of the coefficients are that's what we've been doing. But we need to see whether the coefficient has a statistically significant effect on loyalty, we see this in at least two ways. First, we can look at the p level.

```

127 ->

call:
glm(formula = loyalty ~ category, family = binomial, data = data_train)

Deviance Residuals:
    Min      1Q  Median      3Q     Max 
-1.5152 -1.1350 -0.8436  1.1812  1.5530 

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) 0.30575  0.01539 19.870 < 2e-16 ***
categoryBeer -1.15590  0.02101 -55.009 < 2e-16 ***
categoryBread & Cakes -0.11000  0.02173 -5.063 4.14e-07 ***
categoryBreakfast Sandwiches 0.22179  0.02156 10.288 < 2e-16 ***
categoryCandy/Gum -0.33491  0.01816 -18.442 < 2e-16 ***
categoryChips -0.20758  0.02392 -8.677 < 2e-16 ***
categoryCigarettes -0.48665  0.01769 -27.517 < 2e-16 ***
categoryCigars -0.95134  0.02544 -37.395 < 2e-16 ***
categoryCold Dispensed Beverage 0.46045  0.01706 26.991 < 2e-16 ***
categoryEnergy 0.02773  0.01797  1.544 0.12269  
categoryFuel -0.63600  0.01622 -39.211 < 2e-16 ***
categoryHot Dispensed Beverage -0.01130  0.01866 -0.605 0.54487  
categoryHot Sandwiches & Chicken -0.06145  0.02298 -2.674 0.00749 ** 
categoryJuice/tonics -0.40630  0.01752 -23.196 < 2e-16 ***
categoryLottery -0.91991  0.01865 -49.313 < 2e-16 ***
categoryPizza 0.10643  0.02130  4.996 5.84e-07 *** 
categoryPop (587) -0.31478  0.01706 -18.448 < 2e-16 ***
categoryRoller Grill -0.18406  0.02183 -8.430 < 2e-16 ***
categorySalty Snacks -0.28467  0.01997 -14.252 < 2e-16 ***
categorySmokeless (951) -0.67115  0.02315 -28.989 < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1081043  on 779807  degrees of freedom
Residual deviance: 1050668  on 779788  degrees of freedom
AIC: 1050708

Number of Fisher Scoring iterations: 4

```

The p are greater than z right here, this last column or next to last column. This is the probability that the coefficient we're particularly looking at is equal to zero.



This is not the technical definition for that but it's a point to remember. The thing that you should retain is that the lower the p-value, that means that the coefficient is not equal to zero, and thus is statistically significant. Now, additionally, just to the right of that,

```

127 ->

call:
glm(formula = loyalty ~ category, family = binomial, data = data_train)

Deviance Residuals:
    Min      1Q  Median      3Q     Max 
-1.5152 -1.1350 -0.8436  1.1812  1.5530 

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) 0.30575  0.01539 19.870 < 2e-16 ***
categoryBeer -1.15590  0.02101 -55.009 < 2e-16 ***
categoryBread & Cakes -0.11000  0.02173 -5.063 4.14e-07 ***
categoryBreakfast Sandwiches 0.22179  0.02156 10.288 < 2e-16 ***
categoryCandy/Gum -0.33491  0.01816 -18.442 < 2e-16 ***
categoryChips -0.20758  0.02392 -8.677 < 2e-16 ***
categoryCigarettes -0.48665  0.01769 -27.517 < 2e-16 ***
categoryCigars -0.95134  0.02544 -37.395 < 2e-16 ***
categoryCold Dispensed Beverage 0.46045  0.01706 26.991 < 2e-16 ***
categoryEnergy 0.02773  0.01797  1.544 0.12269    
categoryFuel -0.63600  0.01622 -39.211 < 2e-16 ***
categoryHot Dispensed Beverage -0.01130  0.01866 -0.605 0.54487  
categoryHot Sandwiches & Chicken -0.06145  0.02298 -2.674 0.00749 **  
categoryJuice/tonics -0.40630  0.01752 -23.196 < 2e-16 ***
categoryLottery -0.91991  0.01865 -49.313 < 2e-16 ***
categoryPizza 0.10643  0.02130  4.996 5.84e-07 ***  
categoryPop (587) -0.31478  0.01706 -18.448 < 2e-16 ***
categoryRoller Grill -0.18406  0.02183 -8.430 < 2e-16 ***
categorySalty Snacks -0.28467  0.01997 -14.252 < 2e-16 ***
categorySmokeless (951) -0.67115  0.02315 -28.989 < 2e-16 ***  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1081043  on 779807  degrees of freedom
Residual deviance: 1050668  on 779788  degrees of freedom
AIC: 1050708

Number of Fisher Scoring iterations: 4

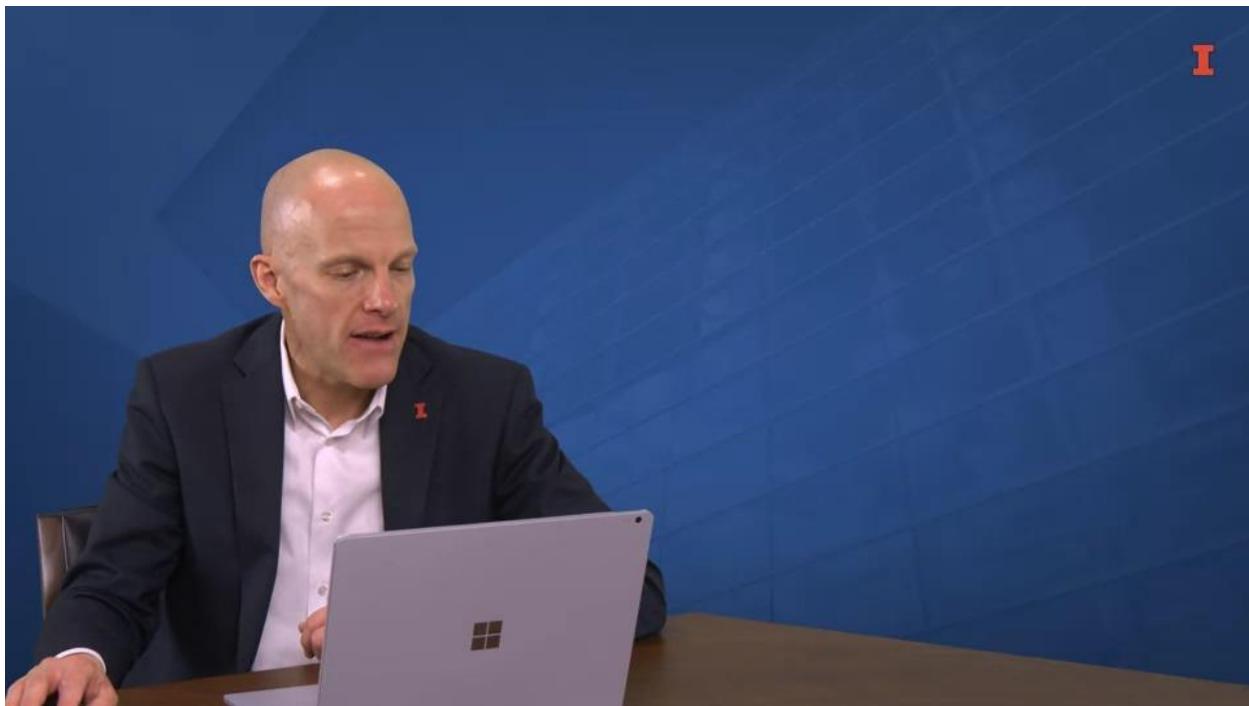
```

you see these asterix here, these little stars. These also indicate the significance of the coefficient.



The more stars, the more significant it is, the more likely that the coefficient is not zero. We have a lot of data here and even small coefficients are likely to be significant. Sure enough, only energy drinks if we look down, and hot sandwiches and chicken are not significant relative to bakery. Overall, what do we learn from this output? We know now that several categories, such as fountain sodas, bakery items, breakfast sandwiches, help increase the chance that a customer will be a loyalty customer. Tikka could promote these products to try to draw in more loyalty customers.

[Lesson 2-3.4 Logistic Regression Hands on - One Variable \(Part3\)](#)



So our next step is to evaluate the model that we've created by predicting the probabilities of each row and examining how accurate the model is at predicting a loyalty purchase. Recall that logistic regression uses the independent variables to create a probability for each row that maximizes the likelihood that the row is close to the true category, loyal or not loyal. Whichever one is the true category. We should predict those probabilities and examine their accuracy since we've labeled data and we know what the true category is for each row. So first. Let's go ahead and predict the probabilities. We really want to do this with the testing data. But we'll do this with the training data first and then compare to the real data and the testing data after that.

```

predicted probabilities rather than the log odds, which are more difficult to interpret. Next, we print a summary of the
probabilities, just so we can get a sense of what they look like. To further evaluate the model, we take the probabilities,
predict_train , and put them into the training data as a new column, data_train$prediction . Finally, we take a look at a
snapshot of the data. You can see that the probabilities here make sense, generally. All of these first 10 rows are from loyalty
customers, and they are all relatively high, considering our highest probability is only 68%.
134
135 # Predict the probabilities of each row on the training data
136 ````{r}
137 predict_train <- predict(model_train, newdata=data_train, type='response')
138 print(summary(predict_train))
139 data_train$prediction <- predict_train
140 head(data_train, n=10)
141 ````{r}

R Console      data.frame [10 x 5]
  loyalty category quarter state prediction
  <fctr>   <chr>    <dbl>   <chr>      <dbl>
1 loyal    Bread & Cakes 3       Colorado  0.5487819
2 loyal    Candy/Gum   1       Wyoming  0.4927105
3 loyal    Pop (587)   3       Wyoming  0.4977430
4 loyal    Pop (587)   4       Colorado 0.4977430
5 loyal    Chips       1       Oklahoma 0.5245229
6 loyal    Juice/tonics 2       Colorado 0.4748839
7 loyal    Juice/tonics 3       Wyoming  0.4748839
8 loyal    Beer        3       Alabama  0.2994030
9 loyal    Chips       2       Oklahoma 0.5245229
10 loyal   Cigarettes  1       Nebraska 0.4548983
11 loyal   Cigarettes 1       Nebraska 0.4548983

1-10 of 10 rows

142
143 Next, let's evaluate our model in another way, by looking at how accurate the model is at making correct predictions. There are
a lot of ways to think about accuracy of this model. You can think of how accurate the model was at detecting loyalty, but you
can also think about how accurate the model was at detecting not loyalty. Both of these types of accuracy to us, since we want
an overall accurate model. A typical way to visualize accuracy is by using a confusion or classification matrix. Again, we are

```

So, let's examine this code here, line by line. The first line is the predictions of the probabilities. We're predicting using the model we just created, model train. All right. So we have the predictive function. We're going to call this new prediction object predict train. Again, we have the training data. Next, as you can see, we make sure we're using the training data by having new data equal data train. The next argument, type equals response, gives the predicted probabilities rather than the law gods, which are more difficult to interpret. So next we're going to print a summary of the probability just so that we can get a sense of what they look like. To further evaluate the model, we take the probabilities, predict underscore train and we're going to put them into the training data as a new column called prediction. Finally, we're going to take a look at a snapshot of the data. So let's run this. We train up our model, we go ahead and do head to look at the top ten rows of our training data. All of these first ten rows are from the loyalty customers as we look. They're all from loyalty and they're all relatively high under the prediction column here. Considering that all of them are above 58% or so.



So next we want to evaluate our model in another way by looking at how accurate the model is in making correct predictions. So there's a lot of ways to think about accuracy of the model. We can think about how accurate the model is at detecting loyalty, but you can also think about how accurate the model is at detecting not loyal or a non loyalty transaction. Both of these types of accuracy makes sense since we want the overall accuracy of the model to be really good. A typical way to visualize accuracy is by using a confusion matrix or a classification matrix. Again, we're using the training data here. So just so we can compare later on to the testing data which we want to use ultimately. So let's again look at the code line by line.

```
# Confusion/classification matrix for training data
```
[1] "table1 <- table(predict_train>0.5, data_train$loyalty) #prediction on left and truth on top
my_confusion_matrix(table1)
```

not loyal    loyal
FALSE      265829  204677
TRUE       124171  185131
[[1]]
[1] "185131 = True Positive (TP), Hit"
[[2]]
[1] "265829 = True Negative (TN), Rejection"
[[3]]
[1] "124171 = False Positive (FP), Type 1 Error"
[[4]]
[1] "204677 = False Negative (FN), Type 2 Error"
[[5]]
[1] "0.5783 = Accuracy (TP+TN/(TP+TN+FP+FN))"
[[6]]
[1] "0.4749 = Sensitivity, Recall, Hit Rate, True Positive Rate (How many positives did the model get right? TP/(TP+FN))"
[[7]]
[1] "0.6816 = Specificity, Selectivity, True Negative Rate (How many negatives did the model get right? TN/(TN+FP))"
[[8]]
[1] "0.5985 = Precision, Positive Predictive Value (How good are the model's positive predictions? TP/(TP+FP))"
[[9]]
[1] "0.5650 = Negative Predictive Value (How good are the model's negative predictions? TN/(TN+FN))"

However, first, let's take our trained model and predict/run it on the testing data. Recall, that we train on the training data and test on the testing data. We want to use new data to test on so that we aren't just trying to memorize the training data, but rather are creating a model that can work on any data.
```

The first line creates a table using the table function that we saw before. All right. This table takes the probability predictions from above, predict underscore train and writes false if the probability is less than 0.5 and true if it's more than 0.5. These are our predictions using again the training data. So let's run that and you can see here at the top what we have. So you can see the faults and true categories here on the left of the table. So it's going to compare these with the truth, which is the loyalty column. So not loyal and loyal here on the top. So the second line then up here from the code is going to run the function from way above that we used at the beginning and ran at the beginning of the notebook.



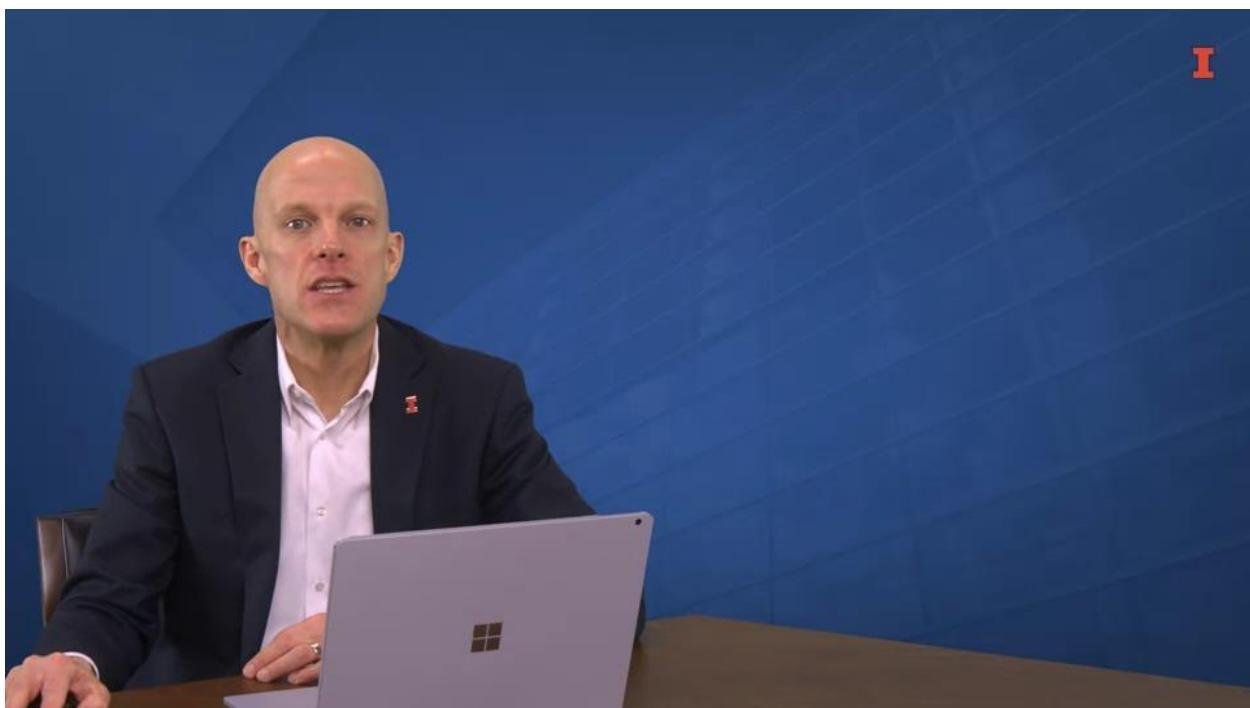
And this function just takes the four cells of this table and labels them and manipulates them to create various types of accuracy displayed right here below. So you have that all here and let's go ahead and explore this. The table shows the following output. First, when a transaction is actually truthful, a loyalty transaction, the model correctly classifies it

```
# Confusion/classification matrix for training data
```
r
table1 <- table(predict_train>0.5, data_train$loyalty) #prediction on left and truth on top
my_confusion_matrix(table1)
```

not loyal loyal
FALSE 265829 204677
TRUE 124171 185131
[[1]]
[1] "185131 = True Positive (TP), Hit"
[[2]]
[1] "265829 = True Negative (TN), Rejection"
[[3]]
[1] "124171 = False Positive (FP), Type 1 Error"
[[4]]
[1] "204677 = False Negative (FN), Type 2 Error"
[[5]]
[1] "0.5783 = Accuracy (TP+TN/(TP+TN+FP+FN))"
[[6]]
[1] "0.4749 = Sensitivity, Recall, Hit Rate, True Positive Rate (How many positives did the model get right? TP/(TP+FN))"
[[7]]
[1] "0.6816 = Specificity, Selectivity, True Negative Rate (How many negatives did the model get right? TN/(TN+FP))"
[[8]]
[1] "0.5985 = Precision, Positive Predictive value (How good are the model's positive predictions? TP/(TP+FP))"
[[9]]
[1] "0.5650 = Negative Predictive Value (How good are the model's negative predictions? TN/(TN+FN))"

However, first, let's take our trained model and predict/run it on the testing data. Recall, that we train on the training data and test on the testing data. We want to use new data to test on so that we aren't just trying to memorize the training data, but rather are creating a model that can work on any data.
```

as such by saying true. So that's right. That's this right here. The 185, 185,131.



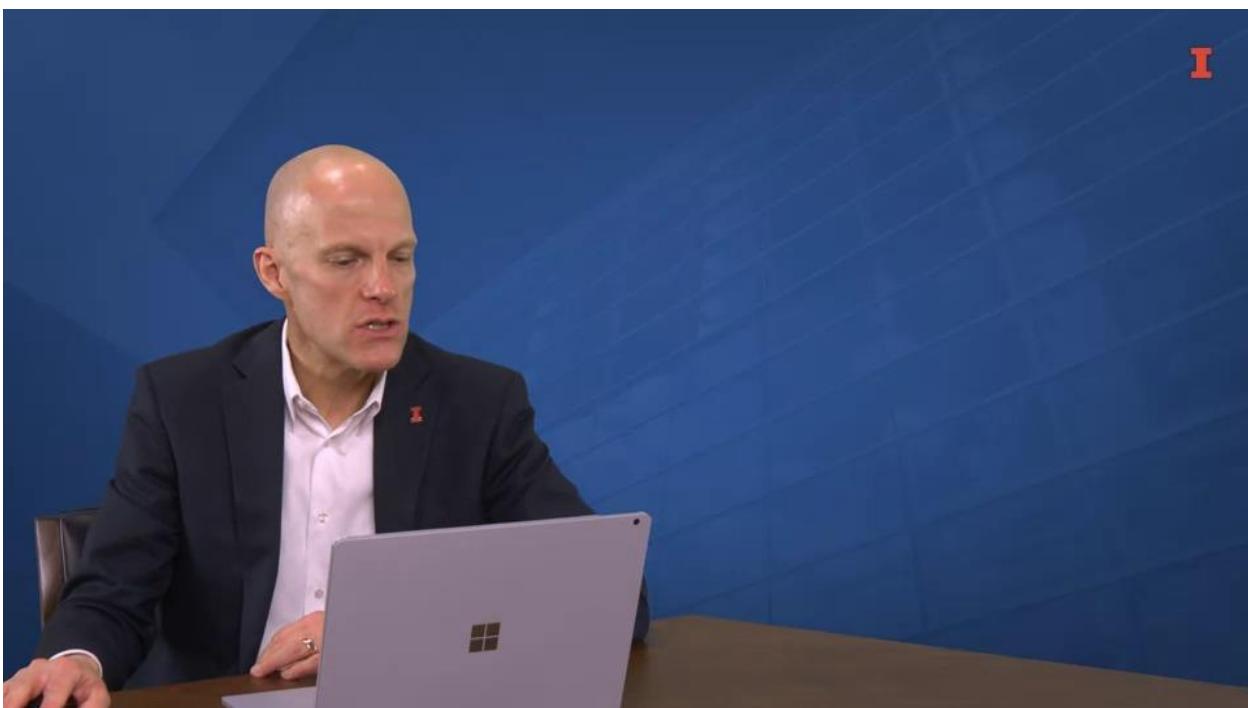
So this is called a true positive or a hit. Now, when a transaction is truthfully a non loyalty transaction, okay,

```
# Confusion/classification matrix for training data
{r}
table1 <- table(predict_train>0.5, data_train$loyalty) #prediction on left and truth on top
my_confusion_matrix(table1)

      not loyal   loyal
FALSE     265829  204677
TRUE      124171  185131
[[1]]
[1] "185131 = True Positive (TP), Hit"
[[2]]
[1] "265829 = True Negative (TN), Rejection"
[[3]]
[1] "124171 = False Positive (FP), Type 1 Error"
[[4]]
[1] "204677 = False Negative (FN), Type 2 Error"
[[5]]
[1] "0.5783 = Accuracy (TP+TN/(TP+TN+FP+FN))"
[[6]]
[1] "0.4749 = Sensitivity, Recall, Hit Rate, True Positive Rate (How many positives did the model get right? TP/(TP+FN))"
[[7]]
[1] "0.6816 = Specificity, Selectivity, True Negative Rate (How many negatives did the model get right? TN/(TN+FP))"
[[8]]
[1] "0.5985 = Precision, Positive Predictive Value (How good are the model's positive predictions? TP/(TP+FP))"
[[9]]
[1] "0.5650 = Negative Predictive Value (How good are the model's negative predictions? TN/(TN+FN))"
```

However, first, let's take our trained model and predict/run it on the testing data. Recall, that we train on the training data and test on the testing data. We want to use new data to test on so that we aren't just trying to memorize the training data, but rather are creating a model that can work on any data.

so the truth is not loyal and indeed our model says fault says It's not loyal. That happens, 265,829 times. This is called a true negative or rejection and again, that's also listed right down here.



On the other hand, the model makes two kinds of errors. When the model transaction is not a loyal transaction, but the model incorrectly says that it is, this is called a fault positive or a type one error. And this happens 124,171 time. So that's here.

```
# Confusion/classification matrix for training data
```
{r}
table1 <- table(predict_train>0.5, data_train$loyalty) #prediction on left and truth on top
my_confusion_matrix(table1)
```

not loyal loyal
FALSE    265829 204677
TRUE     124171 185131
[[1]]
[1] "185131 = True Positive (TP), Hit"
[[2]]
[1] "265829 = True Negative (TN), Rejection"
[[3]]
[1] "124171 = False Positive (FP), Type 1 Error"
[[4]]
[1] "204677 = False Negative (FN), Type 2 Error"
[[5]]
[1] "0.5783 = Accuracy (TP+TN/(TP+TN+FP+FN))"
[[6]]
[1] "0.4749 = Sensitivity, Recall, Hit Rate, True Positive Rate (How many positives did the model get right? TP/(TP+FN))"
[[7]]
[1] "0.6816 = Specificity, Selectivity, True Negative Rate (How many negatives did the model get right? TN/(TN+FP))"
[[8]]
[1] "0.5985 = Precision, Positive Predictive Value (How good are the model's positive predictions? TP/(TP+FP))"
[[9]]
[1] "0.5650 = Negative Predictive Value (How good are the model's negative predictions? TN/(TN+FN))"
```

However, first, let's take our trained model and predict/run it on the testing data. Recall, that we train on the training data and test on the testing data. We want to use new data to test on so that we aren't just trying to memorize the training data, but rather are creating a model that can work on any data.

Again, this is not a loyalty transaction because this is truth. But the model says that it is and so that we call that a type one error.



Finally, when a transaction is a loyalty transaction and the model says it is not, which here happens 204,677 times it's called a faults negative or a type one error.

```
# Confusion/classification matrix for training data
##r
table1 <- table(predict_train>0.5, data_train$loyalty) #prediction on left and truth on top
my_confusion_matrix(table1)

not loyal    loyal
FALSE      265829  204677
TRUE       124171 185131
[[1]]
[1] "185131 = True Positive (TP), Hit"
[[2]]
[1] "265829 = True Negative (TN), Rejection"
[[3]]
[1] "124171 = False Positive (FP), Type 1 Error"
[[4]]
[1] "204677 = False Negative (FN), Type 2 Error"
[[5]]
[1] "0.5783 = Accuracy (TP+TN/(TP+TN+FP+FN))"
[[6]]
[1] "0.4749 = Sensitivity, Recall, Hit Rate, True Positive Rate (How many positives did the model get right? TP/(TP+FN))"
[[7]]
[1] "0.6816 = Specificity, Selectivity, True Negative Rate (How many negatives did the model get right? TN/(TN+FP))"
[[8]]
[1] "0.5985 = Precision, Positive Predictive Value (How good are the model's positive predictions? TP/(TP+FP))"
[[9]]
[1] "0.5650 = Negative Predictive Value (How good are the model's negative predictions? TN/(TN+FN))"

However, first, let's take our trained model and predict/run it on the testing data. Recall, that we train on the training data and test on the testing data. We want to use new data to test on so that we aren't just trying to memorize the training data, but rather are creating a model that can work on any data.
```

Again, we see that here where it actually is a loyalty transaction but the model says it is not.



So these numbers can then be manipulated, multiplied and divided to create different measures of accuracy that are commonly used in practice. For example,

```
# Confusion/classification matrix for training data
{r}
table1 <- table(predict_train>0.5, data_train$loyalty) #prediction on left and truth on top
my_confusion_matrix(table1)

not loyal    loyal
FALSE      265829  204677
TRUE       124171 185131
[[1]]
[1] "185131 = True Positive (TP), Hit"
[[2]]
[1] "265829 = True Negative (TN), Rejection"
[[3]]
[1] "124171 = False Positive (FP), Type 1 Error"
[[4]]
[1] "204677 = False Negative (FN), Type 2 Error"
[[5]]
[1] "0.5783 = Accuracy (TP+TN/(TP+TN+FP+FN))"
[[6]]
[1] "0.4749 = Sensitivity, Recall, Hit Rate, True Positive Rate (How many positives did the model get right? TP/(TP+FN))"
[[7]]
[1] "0.6816 = Specificity, Selectivity, True Negative Rate (How many negatives did the model get right? TN/(TN+FP))"
[[8]]
[1] "0.5985 = Precision, Positive Predictive Value (How good are the model's positive predictions? TP/(TP+FP))"
[[9]]
[1] "0.5650 = Negative Predictive Value (How good are the model's negative predictions? TN/(TN+FN))"

However, first, let's take our trained model and predict/run it on the testing data. Recall, that we train on the training data and test on the testing data. We want to use new data to test on so that we aren't just trying to memorize the training data, but rather are creating a model that can work on any data.
```

overall accuracy is probably the most important thing that we're interested in here. Overall accuracy is 57, about 58%.



Next we have sensitivity. This is also called the recall rates or the true positive rates or the hit rate. Basically, how many positives did the model get right.

```
# Confusion/classification matrix for training data
{r}
table1 <- table(predict_train>0.5, data_train$loyalty) #prediction on left and truth on top
my_confusion_matrix(table1)

      not loyal  loyal
FALSE     265829 204677
TRUE      124171 185131
[[1]]
[1] "185131 = True Positive (TP), Hit"
[[2]]
[1] "265829 = True Negative (TN), Rejection"
[[3]]
[1] "124171 = False Positive (FP), Type 1 Error"
[[4]]
[1] "204677 = False Negative (FN), Type 2 Error"
[[5]]
[1] "0.5783 = Accuracy (TP+TN/(TP+TN+FP+FN))"
[[6]]
[1] "0.4749 = Sensitivity, Recall, Hit Rate, True Positive Rate (How many positives did the model get right? TP/(TP+FN))"
[[7]]
[1] "0.6816 = Specificity, Selectivity, True Negative Rate (How many negatives did the model get right? TN/(TN+FP))"
[[8]]
[1] "0.5985 = Precision, Positive Predictive Value (How good are the model's positive predictions? TP/(TP+FP))"
[[9]]
[1] "0.5650 = Negative Predictive Value (How good are the model's negative predictions? TN/(TN+FN))"

However, first, let's take our trained model and predict/run it on the testing data. Recall, that we train on the training data and test on the testing data. We want to use new data to test on so that we aren't just trying to memorize the training data, but rather are creating a model that can work on any data.
```

And here we're actually below 50% at 47%. Next we have specificity also called selectivity, the true negative.



This basically answers how many negatives did the model get right. And here we're at 68%.

```
# Confusion/classification matrix for training data
## [r]
table1 <- table(predict_train>0.5, data_train$loyalty) #prediction on left and truth on top
my_confusion_matrix(table1)
...


|       | not loyal | loyal  |
|-------|-----------|--------|
| FALSE | 265829    | 204677 |
| TRUE  | 124171    | 185131 |


[[1]]
[1] "185131 = True Positive (TP), Hit"
[[2]]
[1] "265829 = True Negative (TN), Rejection"
[[3]]
[1] "124171 = False Positive (FP), Type 1 Error"
[[4]]
[1] "204677 = False Negative (FN), Type 2 Error"
[[5]]
[1] "0.5783 = Accuracy (TP+TN/(TP+TN+FP+FN))"
[[6]]
[1] "0.4749 = Sensitivity, Recall, Hit Rate, True Positive Rate (How many positives did the model get right? TP/(TP+FN))"
[[7]]
[1] "0.6816 = Specificity, Selectivity, True Negative Rate (How many negatives did the model get right? TN/(TN+FP))"
[[8]]
[1] "0.5985 = Precision, Positive Predictive Value (How good are the model's positive predictions? TP/(TP+FP))"
[[9]]
[1] "0.5650 = Negative Predictive Value (How good are the model's negative predictions? TN/(TN+FN))"

However, first, let's take our trained model and predict/run it on the testing data. Recall, that we train on the training data and test on the testing data. We want to use new data to test on so that we aren't just trying to memorize the training data, but rather are creating a model that can work on any data.

```

Next we have another common measure of accuracy and that's precision. Also called the positive predictive value. This basically says how good are the model's positive predictions. And again, in each case I've indicated how these are calculated, so you can refer back to that. Last, we have the negative predictive value, and this here is basically saying how good are the model's negative predictions.



Thus, overall, our model predicts better than chance, that overall accuracy level was about 57, 58%. Recall that loyalty transactions happen about 50% of the time. Thus, if we had no model at all [LAUGH] and just flipped the coin every time to guess whether a transaction was going to be a loyalty transaction or not, we'd get 50%. We'd get it right about half the time. So our model has helped us to get it right a little bit more than that. It helps us get it right 58% of the time, which is not super high, but is better than chance. Where the model really kind of falls down and is not great, is in the sensitivity area. That is, the model is pretty good at predicting not loyal,

```
# Confusion/classification matrix for training data
``[r]
table1 <- table(predict_train>0.5, data_train$loyalty) #prediction on left and truth on top
my_confusion_matrix(table1)
``

not loyal loyal
FALSE 265829 204677
TRUE 124171 185131
[[1]]
[1] "185131 = True Positive (TP), Hit"
[[2]]
[1] "265829 = True Negative (TN), Rejection"
[[3]]
[1] "124171 = False Positive (FP), Type 1 Error"
[[4]]
[1] "204677 = False Negative (FN), Type 2 Error"
[[5]]
[1] "0.5783 = Accuracy (TP+TN/(TP+TN+FP+FN))"
[[6]]
[1] "0.4749 = Sensitivity, Recall, Hit Rate, True Positive Rate (How many positives did the model get right? TP/(TP+FN))"
[[7]]
[1] "0.6816 = Specificity, Selectivity, True Negative Rate (How many negatives did the model get right? TN/(TN+FP))"
[[8]]
[1] "0.5985 = Precision, Positive Predictive Value (How good are the model's positive predictions? TP/(TP+FP))"
[[9]]
[1] "0.5650 = Negative Predictive Value (How good are the model's negative predictions? TN/(TN+FN))"

However, first, let's take our trained model and predict/run it on the testing data. Recall, that we train on the training data and test on the testing data. We want to use new data to test on so that we aren't just trying to memorize the training data, but rather are creating a model that can work on any data.
```

68% right here. The specificity is good but bad at predicting loyal, getting it right, only 47% of the time. That's worse than flipping a coin. So our next step is to improve this model.



However, before we do that, let's take our training model and predict or run it on the testing data. Recall that we train on the training data, we test on the testing data. So we want to use new data to test on so that we're not just trying to memorize the training data, so that we don't over fit our model. But rather we're creating a model that we can work on any data.

```

# Predict and evaluate the model on the test data
...{r}
predict_test <- predict(model_train, newdata=data_test, type='response')
print(summary(predict_test))
data_test$prediction <- predict_test
head(data_test, n=10)
table2 <- table(predict_test>.5, data_test$loyalty) #prediction on left and truth on top
my_confusion_matrix(table2)

[[2]]
[1] "88248 (0.5140) = True Negative (TN), Rejection"

[[3]]
[1] "41752 (0.1914) = False Positive (FP), Type 1 Error"

[[4]]
[1] "67847 (0.3532) = False Negative (FN), Type 2 Error"

[[5]]
[1] "0.5784 = Accuracy (TP+TN/(TP+TN+FP+FN))"

[[6]]
[1] "0.4778 = Sensitivity, Recall, Hit Rate, True Positive Rate (How many positives did the model get right? TP/(TP+FN))"

[[7]]
[1] "0.6788 = Specificity, Selectivity, True Negative Rate (How many negatives did the model get right? TN/(TN+FP))"

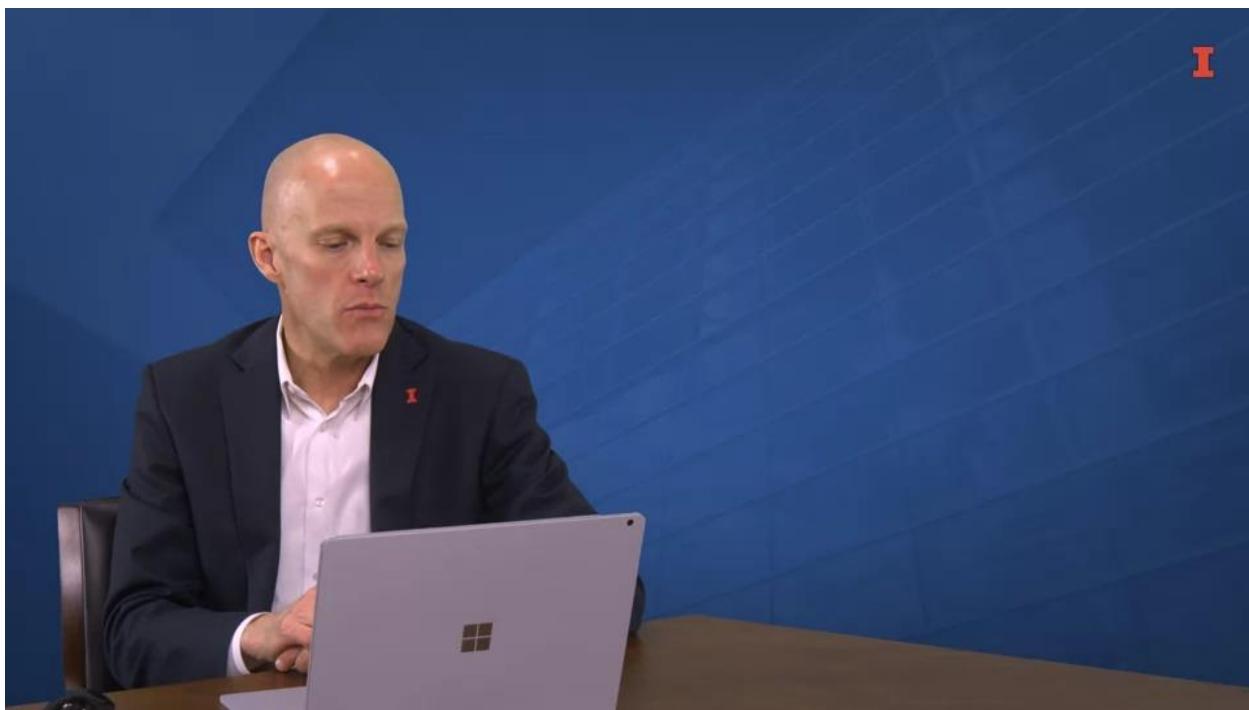
[[8]]
[1] "0.5979 = Precision, Positive Predictive Value (How good are the model's positive predictions? TP/(TP+FP))"

[[9]]
[1] "0.5653 = Negative Predictive Value (How good are the model's negative predictions? TN/(TN+FN))"

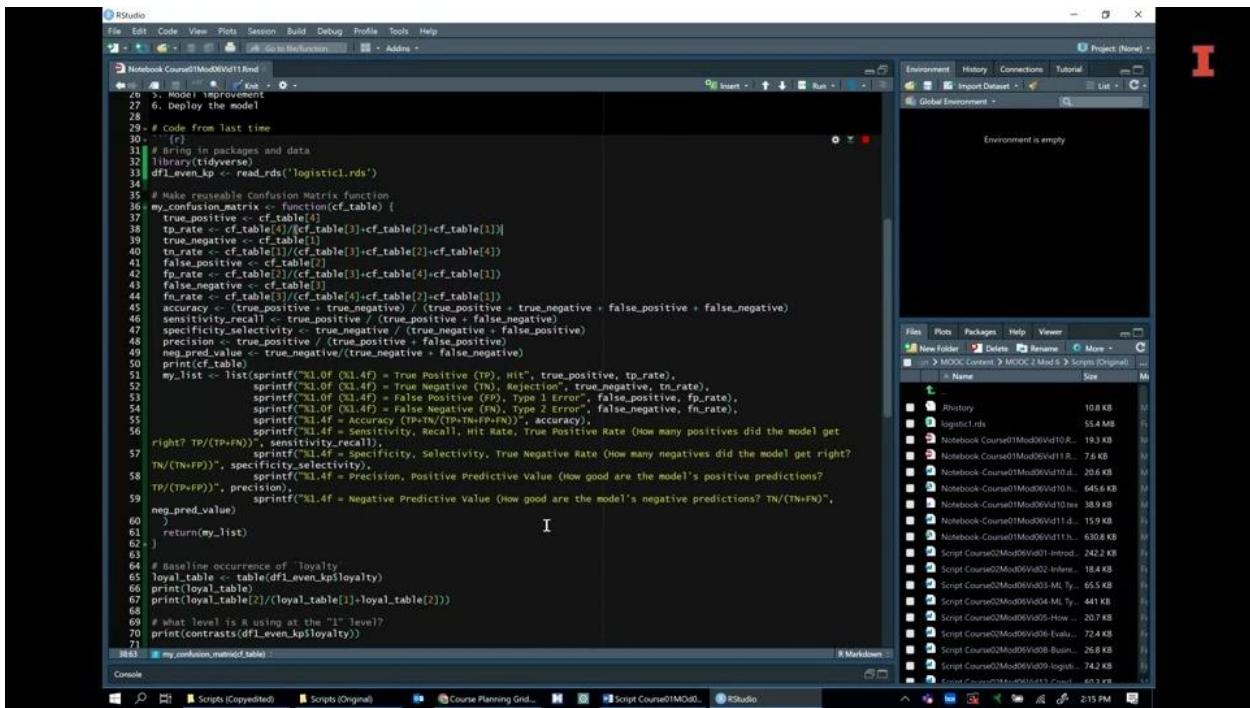
```

So let's do that now and run this. This code of course, looks very similar. So what we did above, the key exception here is that the data being used is now the test data. So that's the big critical key difference that you don't want to miss. So how does our model perform in the new data. Well, it actually does a little bit better than the training data. So while accuracy is pretty similar here, you know, still about 58%. Our problem from before, the low sensitivity is slightly improved. Not much, but slightly improved.

[Lesson 2-3.5 Logistic Regression Hands on - Multiple Variables](#)



In this video, we'll continue exploring logistic regression. Logistic regression is a valuable classification type machine learning algorithm. In this video, we'll use the algorithm to predict whether a transaction is from a loyalty customer or not. We'll use the TEKA data set. Now, TEKA desires to increase the loyalty customers that they have and the number of products that their loyalty customers purchased because loyalty customers creates more business and more profitable business for them. As always, we encourage you to resist the urge to just watch the videos rather, please follow along, doing each of the steps that we're doing so that you can get the critical hands-on experience necessary to learn this valuable tool. First, let's bring in the data and the needed packages. Last time we predicted loyalty using a single variable category but this time, we'll use all of our variables. We'll see if this improves the prediction accuracy of our model. Before we get to the model statement, the code will be identical to the prior video when we used only the one variable so we'll run all of that here just right now



The screenshot shows an RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Go to Function, Addins.
- Code Editor:** Notebook Course01Mod08Vid1.Rnw containing R code for model improvement, including a confusion matrix function and its usage.
- Environment Tab:** Global Environment - Environment is empty.
- File Explorer:** Shows a file tree for the course project, including files like Rhistory, logistic1.rds, and various notebook and script files.
- Console:** Displays the command `my.confusion_matrix`.
- Status Bar:** Shows the date and time as 2/15 PM.

all the way up to the model statement. Of our six steps for using a machine learning algorithm,



we're now focused on step 5. The first step is data scrubbing, the second was algorithms selection, third was model training or fitting, fourth was model evaluation, and five is model improvement, six is deploying the model, so we're firmly into the number five here.

```

53         sprintf("%1.4f = Accuracy ((TP+TN)/(TP+TN+FP+FN))", accuracy),
54         sprintf("%1.4f = Sensitivity, Recall, Hit Rate, True Positive Rate (How many positives did the model get
55 right? TP/(TP+FN))", sensitivity_recall),
56         sprintf("%1.4f = Specificity, Selectivity, True Negative Rate (How many negatives did the model get right?
57 TN/(TN+FP))", specificity_selectivity),
58         sprintf("%1.4f = Precision, Positive Predictive Value (How good are the model's positive predictions?
59 TP/(TP+FP))", precision),
60         sprintf("%1.4f = Negative Predictive Value (How good are the model's negative predictions? TN/(TN+FN))",
61 neg_pred_value)
62     )
63     return(my_list)
62 = }
63
64 # Baseline occurrence of `loyalty`
65 loyal_table <- table(df1_even_kp$loyalty)
66 print(loyal_table)
67 print(loyal_table[2]/(loyal_table[1]+loyal_table[2]))
68
69 # what level is R using at the "1" level?
70 print(contrasts(df1_even_kp$loyalty))
71
72 # Split data
73 library(caret)
74 set.seed(77)
75 partition <- caret::createDataPartition(y=df1_even_kp$loyalty, p=.75, list=FALSE)
76 data_train <- df1_even_kp[partition,]
77 data_test <- df1_even_kp[-partition,]
78 print(nrow(data_train)/(nrow(data_test)+nrow(data_train)))
79 =

```

Registered S3 methods overwritten by 'dbplyr':
method from
print.tbl_lazy
print.tbl_sql
-- Attaching packages --tidyverse 1.3.0 --
v ggplot2 3.3.3 v purrr 0.3.4
v tibble 3.0.4 v dplyr 1.0.2
v tidyverse 1.1.2 v stringr 1.4.0
v readr 1.4.0 v forcats 0.5.0
-- Conflicts --tidyverse_conflicts()
#> dplyr::filter() masks stats::filter()
#> dplyr::lag() masks stats::lag()

All right. So as you can see as you scroll through this code again, this is just identical from last time when we had a univariate logistical regression model. Now, with a multivariate regression model with multiple independent variables, everything will be the same until we get to the model statement.



Great. So our data split. We're now ready to train our model. Recall that an algorithm is a set of rules for how to make a model. A model takes our data and options that we provide it to select and create new data and information and additional rules for how to use the model.

```

Deviance Residuals:
    Min      1Q  Median      3Q     Max 
-1.7136 -1.1285 -0.6539  1.1315  1.8153 

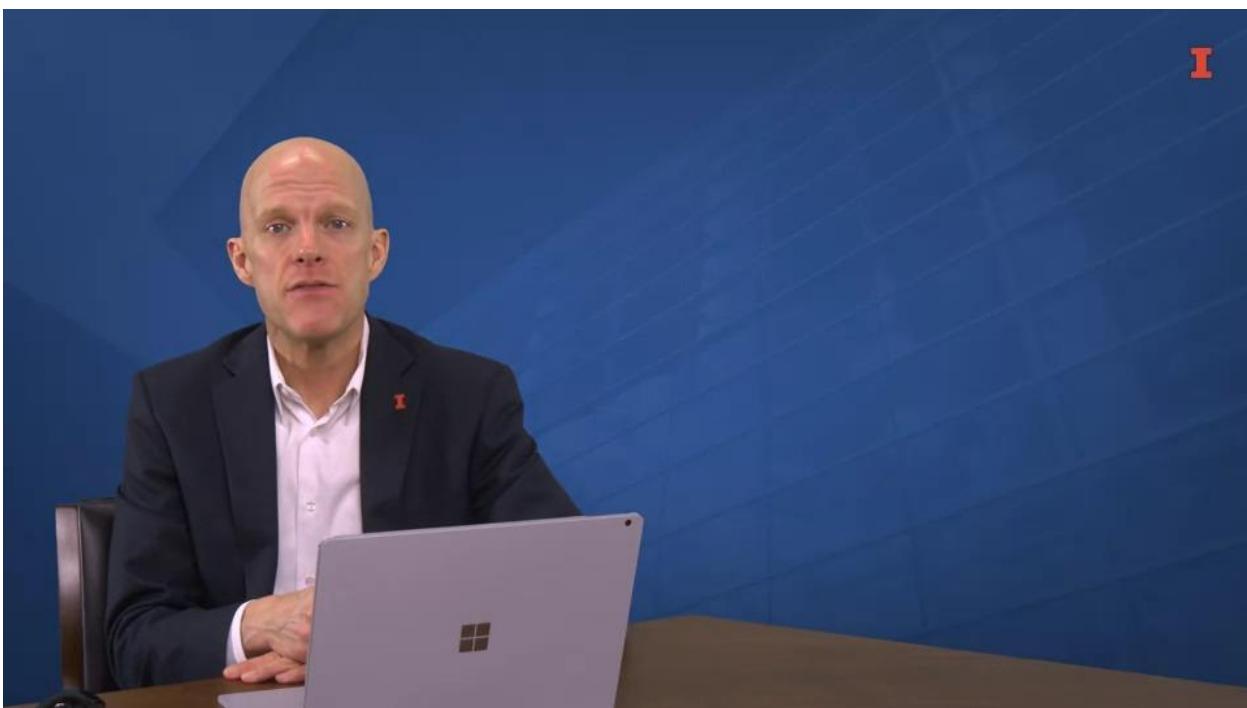
Coefficients:
                Estimate Std. Error z value Pr(>|z|)    
(Intercept)       0.254291  0.016633 15.289 < 2e-16 ***
categoryBeer     -1.096064  0.021205 -51.689 < 2e-16 ***
categoryBread & Cakes -0.114865  0.021934 -5.237 1.63e-07 ***
categoryBreakfast Sandwiches 0.220034  0.021724 10.129 < 2e-16 ***
categoryCandy/Gum   -0.341969  0.018316 -18.670 < 2e-16 ***
categoryChips      -0.204701  0.024131 -8.483 < 2e-16 ***
categoryCigarettes -0.511199  0.017849 -28.641 < 2e-16 ***
categoryCigars      -0.966302  0.025690 -37.614 < 2e-16 ***
categoryCold Dispensed Beverage 0.447609  0.017227 25.982 < 2e-16 ***
categoryEnergy      0.015697  0.018120  0.866  0.38633    
categoryFuel       -0.638023  0.016359 -39.002 < 2e-16 ***
categoryHot Dispensed Beverage -0.009300  0.018821 -0.494  0.62122  
categoryHot Sandwiches & Chicken 0.059721  0.023156 -2.579  0.00991 **  
categoryJuice/tonics   -0.433122  0.017665 -24.519 < 2e-16 ***
categoryLottery      -0.991808  0.018851 -52.612 < 2e-16 ***
categoryPizza       0.096907  0.021468  4.514  6.36e-06 ***
categoryPop (587)   -0.279395  0.017215 -16.229 < 2e-16 ***
categoryRoller Grill -0.192345  0.022021 -8.735 < 2e-16 ***
categorySalty Snacks -0.281628  0.020151 -13.976 < 2e-16 ***
categorySmokeless (951) -0.691139  0.023350 -29.608 < 2e-16 ***
factor(quarter)2    0.194020  0.006887 28.170 < 2e-16 ***
factor(quarter)3    0.216750  0.006653 32.580 < 2e-16 ***
factor(quarter)4    0.228426  0.006624 34.486 < 2e-16 ***  

stateArkansas      -0.341681  0.008364 -40.851 < 2e-16 ***
stateColorado      0.116604  0.007972 14.627 < 2e-16 ***
stateIowa          -0.285021  0.007312 -38.980 < 2e-16 ***
stateMinnesota     -0.285307  0.030645 -9.310 < 2e-16 ***
stateMissouri      0.276047  0.008295 33.281 < 2e-16 ***
stateNebraska      -0.182782  0.010315 -17.720 < 2e-16 ***
stateOklahoma      -0.592161  0.009407 -62.947 < 2e-16 ***
stateSouth Dakota -0.165378  0.021925 -7.543 4.60e-14 ***
stateWyoming        0.085997  0.013440  6.399 1.57e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

```

Below, we create our model called model_train and then summarize it to view the output. So we're again, using the GLM function with family equal to binomial to indicate that it's a logistic regression. Our dependent variable continues here to be loyalty, but this time will include all of our independent variables in the model, not just category. This is multivariate logistic regression so we have multiple independent variables. We are, of course, using the training data here, since we're training the model. So let's go ahead and run this and summarize as we train our model using both quarter, category and state. All right. So here's our output. Now, last time we examined the impact of category on loyalty. So let's look this time at quarter, right down here.



The quarter is just a quarter of the year. We made it into a factor here so that R knows the category. There's some debate about whether we could leave this as a continuous variable or not, since the difference between one quarter of the year and another quarter of the year does have meaning but for our purposes, we'll leave it as a factor or as a categorical variable. The first quarter of the year is the reference level of these multiple dummy variables. Thus, when we evaluate the effects of these quarters, it's in relationship to the effect of quarter 1. Looking at the signs of the coefficients and the p-values that are highlighted here, we can see that each quarter significantly increases the probability that a transaction will be a loyalty transaction. For example, the third quarter of the year has a positive coefficient

```

Deviance Residuals:
    Min      1Q  Median      3Q     Max 
-1.7136 -1.1285 -0.6539  1.1315  1.8153 

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) 0.254291  0.016633 15.289 < 2e-16 ***
categoryBeer -0.096064  0.021205 -51.689 < 2e-16 ***
categoryBread & Cakes -0.114865  0.021934 -5.237 1.63e-07 ***
categoryBreakfast Sandwiches 0.220034  0.021724 10.129 < 2e-16 ***
categoryCandy/Gum -0.341969  0.018316 -18.670 < 2e-16 ***
categoryChips -0.204701  0.024131 -8.483 < 2e-16 ***
categoryCigarettes -0.511199  0.017849 -28.641 < 2e-16 ***
categoryCigars -0.966302  0.025690 -37.614 < 2e-16 ***
categoryCold Dispensed Beverage 0.447609  0.017227 25.982 < 2e-16 ***
categoryEnergy 0.015697  0.018120  0.866 0.38633    
categoryFuel -0.638023  0.016359 -39.002 < 2e-16 ***
categoryHot Dispensed Beverage -0.009300  0.018821 -0.494 0.62122  
categoryHot Sandwiches & Chicken -0.059721  0.023156 -2.579 0.00991 **  
categoryJuice/tonics -0.433122  0.017665 -24.519 < 2e-16 ***
categoryLottery -0.991808  0.018851 -52.612 < 2e-16 ***
categoryPizza 0.096907  0.021468  4.514 6.36e-06 ***
categoryPop (587) -0.279395  0.017215 -16.229 < 2e-16 ***
categoryRoller Grill -0.192345  0.022021 -8.735 < 2e-16 ***
categorySalty Snacks -0.281628  0.020151 -13.976 < 2e-16 ***
categorySmokeless (951) -0.691339  0.023350 -29.608 < 2e-16 ***
factor(quarter)2 0.194020  0.006887 28.170 < 2e-16 ***
factor(quarter)3 0.216750  0.006653 32.580 < 2e-16 ***I
factor(quarter)4 0.228426  0.006624 34.486 < 2e-16 ***
stateArkansas -0.341681  0.008364 -40.851 < 2e-16 ***
stateColorado 0.116604  0.007972 14.627 < 2e-16 ***
stateIowa -0.285021  0.007312 -38.980 < 2e-16 ***
stateMinnesota -0.285307  0.030645 -9.310 < 2e-16 ***
stateMissouri 0.276047  0.008295 33.281 < 2e-16 ***
stateNebraska -0.182782  0.010315 -17.720 < 2e-16 ***
stateOklahoma -0.592161  0.009407 -62.947 < 2e-16 ***
stateSouth Dakota -0.165378  0.021925 -7.543 4.60e-14 ***
stateWyoming 0.085997  0.013440  6.399 1.57e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

```

of 216750 and a very low p-value less than two with a 16 zeros in front of it.



This means that relative to, again, quarter 1, quarter 3 has more transactions that are from the loyalty customers. With this knowledge, TEKA could consider focusing on the beginning of the year in an effort to increase the number of loyalty customers that they have. Additionally, TEKA could focus on the summer and fall as key times when their loyalty customers all are already active to promote loyalty to them and help increase those loyalty customers' purchases even more. Next, let's look at how each state affects the occurrence

```

Deviance Residuals:
    Min      1Q  Median      3Q     Max 
-1.7136 -1.1285 -0.6539  1.1315  1.8153 

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) 0.254291  0.016633 15.289 < 2e-16 ***
categoryBeer -1.096064  0.021205 -51.689 < 2e-16 ***
categoryBread & Cakes -0.114865  0.021934 -5.237 1.63e-07 ***
categoryBreakfast Sandwiches 0.220034  0.021724 10.129 < 2e-16 ***
categoryCandy/Gum -0.341969  0.018316 -18.670 < 2e-16 ***
categoryChips -0.204701  0.024131 -8.483 < 2e-16 ***
categoryCigarettes -0.511199  0.017849 -28.641 < 2e-16 ***
categoryCigars -0.966302  0.025690 -37.614 < 2e-16 ***
categoryCold Dispensed Beverage 0.447609  0.017227 25.982 < 2e-16 ***
categoryEnergy 0.015697  0.018120  0.866  0.38633  
categoryFuel -0.638023  0.016359 -39.002 < 2e-16 ***
categoryHot Dispensed Beverage -0.009300  0.018821 -0.494  0.62122 
categoryHot Sandwiches & Chicken 0.059721  0.023156 -2.579  0.00991 ** 
categoryJuice/tonics -0.433122  0.017665 -24.519 < 2e-16 ***
categoryLottery -0.991808  0.018851 -52.612 < 2e-16 ***
categoryPizza 0.096907  0.021468  4.514  6.36e-06 ***
categoryPop (587) -0.279395  0.017215 -16.229 < 2e-16 ***
categoryRoller Grill -0.192345  0.022021 -8.735 < 2e-16 ***
categorySalty Snacks -0.281628  0.020151 -13.976 < 2e-16 ***
categorySmokeless (951) -0.691139  0.023350 -29.608 < 2e-16 ***
factor(quarter)2 0.194020  0.006887 28.170 < 2e-16 ***
factor(quarter)3 0.216750  0.006653 32.580 < 2e-16 ***
factor(quarter)4 0.228426  0.006624 34.486 < 2e-16 ***
stateArkansas -0.341681  0.008364 -40.851 < 2e-16 ***
stateColorado 0.116604  0.007972 14.627 < 2e-16 ***
stateIowa -0.285021  0.007312 -38.980 < 2e-16 ***
stateMinnesota -0.285307  0.030645 -9.310 < 2e-16 ***
stateMissouri 0.276047  0.008295 33.281 < 2e-16 ***
stateNebraska -0.182782  0.010315 -17.720 < 2e-16 ***
stateOklahoma -0.592161  0.009407 -62.947 < 2e-16 ***
stateSouth Dakota -0.165378  0.021925 -7.543 4.60e-14 ***
stateWyoming 0.085997  0.013440  6.399 1.57e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

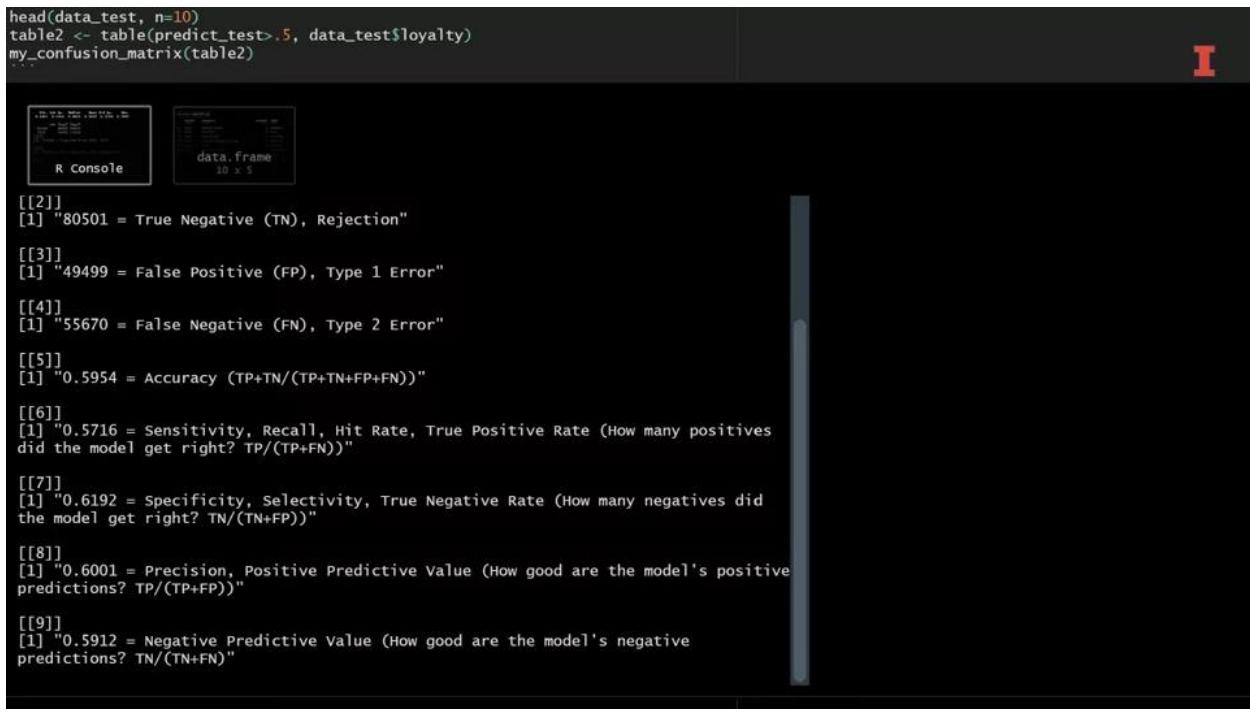
```

of our loyalty transactions. These are the dummy variables here. The reference state, the state up and intercept, is Alabama.



So relative to Alabama, the states that are most likely to generate loyalty transactions are Colorado, Missouri, and Wyoming. Missouri, in particular, has a strong effect. This can be seen by their relatively large coefficient. So TEKA should investigate key stores in this state to see what's going on right here. Finally, let's examine our larger, more complicated model on our test data. Recall that our single variable model had a relatively modest accuracy rate of about 57, 58 percent. The more complex model, let's look at that now.

```
head(data_test, n=10)
table2 <- table(predict_test[,5], data_test$loyalty)
my_confusion_matrix(table2)
```



```

[[2]]
[1] "80501 = True Negative (TN), Rejection"

[[3]]
[1] "49499 = False Positive (FP), Type 1 Error"

[[4]]
[1] "55670 = False Negative (FN), Type 2 Error"

[[5]]
[1] "0.5954 = Accuracy (TP+TN/(TP+TN+FP+FN))"

[[6]]
[1] "0.5716 = Sensitivity, Recall, Hit Rate, True Positive Rate (How many positives did the model get right? TP/(TP+FN))"

[[7]]
[1] "0.6192 = Specificity, Selectivity, True Negative Rate (How many negatives did the model get right? TN/(TN+FP))"

[[8]]
[1] "0.6001 = Precision, Positive Predictive Value (How good are the model's positive predictions? TP/(TP+FP))"

[[9]]
[1] "0.5912 = Negative Predictive Value (How good are the model's negative predictions? TN/(TN+FN))"

```


```

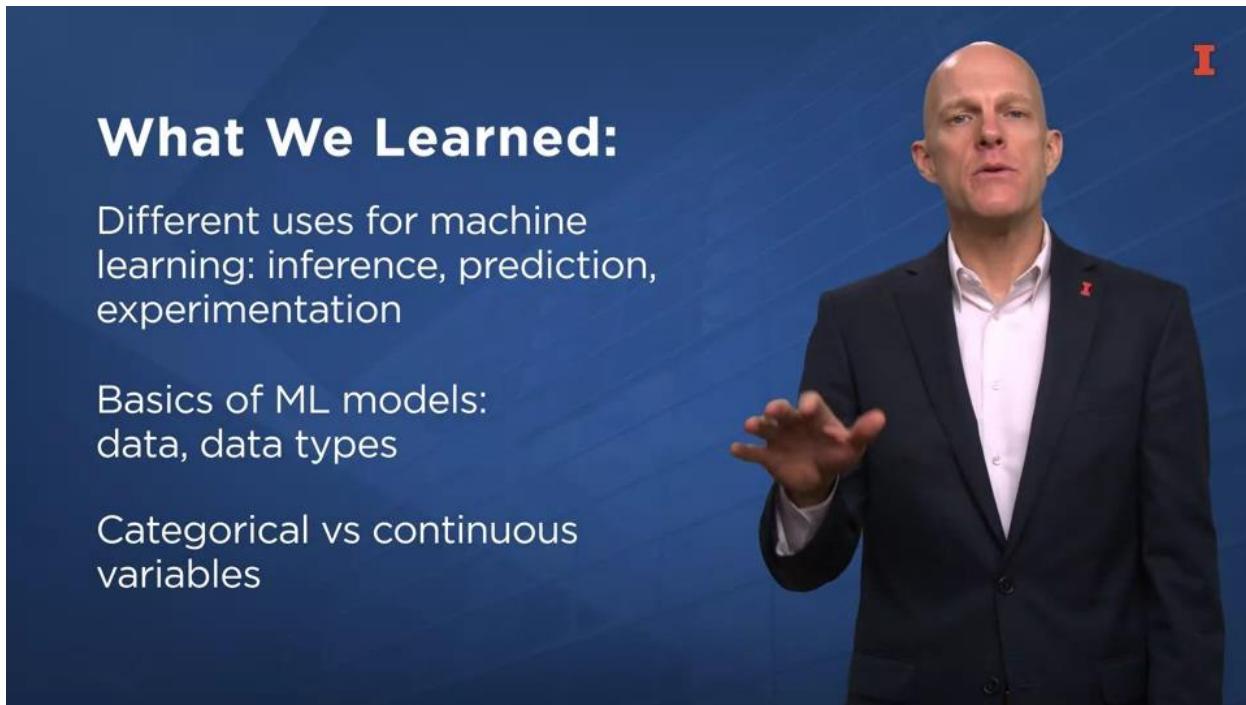
So again, here we've included our test data because we're going to run this model on the test data. We want to see how well our trained data predicts on our test data. We've run summary right here so we can view that, and then we've added our predict tests, our probabilities, back to our data set. If we wanted to, we could look at that and we see it right here. Then we've run heads so we can look at these. Finally, we are going to run this confusion matrix that we looked at last time. The more complex model has increased the accuracy slightly up to about almost 60 percent as you can see here but perhaps more importantly, the sensitivity of the model is much improved right here. Now, recall that before univariate logistic regression we had, the sensitivity or the accuracy of that mode predicting not loyal was below 50 percent,



worse than a coin flip, but this new model has increased that all the way up to 57 percent. So overall, this is not a great model. We could certainly add additional variables to make it more complex. Additionally, recall that this data is just a subset of the available data so adding data would also improve the predictability of our model.

## Module 2 Review

### Module 2 Conclusion



**What We Learned:**

- Different uses for machine learning: inference, prediction, experimentation
- Basics of ML models: data, data types
- Categorical vs continuous variables

Congratulations. In this module we've really covered a lot of ground. First, we've done an overview of machine learning to provide you with a solid framework or foundation for how to understand machine learning. Thus in the future, in this class or your own learning efforts, when you encounter a new model, that you've never encountered before, you'll be able to slide into the framework, understand it quickly, and move more quickly through the process of understanding the new algorithm. More importantly, you'll be able to put that algorithm to work for you in solving business problems. In this module, we did the following. We discussed different uses for machine learning, inference, prediction, and experimentation. In all three of these paradigms, data is king. Data and our statistical analysis of data drives our decisions. Next, we examine the building blocks for machine learning models, data and data types. We explored the differences between categorical, and continuous variables.

## What We Learned:

Supervised, unsupervised,  
and reinforcement learning

Regression, classification,  
and clustering algos

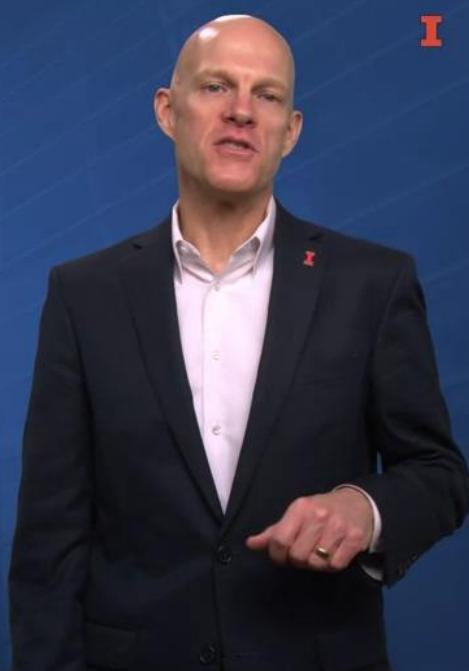


Next, we talked about the different types of machine learning, and machine learning algorithms. We talked about supervised, unsupervised and reinforcement learning, their definitions and their differences. After that, we discussed regression, classification, and clustering algorithms and how they differ. We then showed a graphic to put everything in its place.

## What We Learned:

Six steps of machine learning

Difference between machine  
learning algorithms and models



We then walked through a series of six steps for machine learning. We also discussed the difference between machine learning algorithms, and models.



Finally, we introduced a business problem from our hypothetical company TECA. We learned a classification algorithm, logistic regression and used the algorithm, and the model to gain insight into when TECA's transactions are made to loyalty versus non loyalty customers. This can help the company increase their number of loyalty customers, as well as the number of transactions their loyalty customers make.