End-to-End Data Science with SAS®

A Hands-On Programming Guide

James Gearheart

The correct bibliographic citation for this manual is as follows: Gearhart, James. 2020. *End-to-End Data Science with SAS®: A Hands-On Programming Guide*. Cary, NC: SAS Institute Inc.

End-to-End Data Science with SAS®: A Hands-On Programming Guide

ISBN 978-1-64295-808-9 (Hard cover)
ISBN 978-1-64295-804-1 (Paperback)
ISBN 978-1-64295-805-8 (Web PDF)
ISBN 978-1-64295-806-5 (Epub)
ISBN 978-1-64295-807-2 (Kindle)

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

June 2020

Contents

About This Book

What Does This Book Cover?

Hello, my name is James, and I'm an addict. I'm addicted to data science books, web courses, instructional videos, blogs, data science podcasts, predictive modeling competitions, and coding. This addiction takes up the majority of my mental energy. From the time that I wake up until I fall asleep (and all through my dreams), I'm generally thinking about data science concepts and coding. I'm going to bet that many of you are in a similar situation. If so, I'm sure that you have been as frustrated as I have been about the massive hole in the instructional data science market.

It is essential that a data scientist who is working in a SAS environment be able to develop and implement machine learning models in ANY SAS environment.

The market is overrun with data science books for Python, R, and Hadoop. These books provide an overview of data science and in-depth instructions on the various machine learning models, and they provide the associated development code for those particular programming languages. Although these books are great resources for data scientists, they do not offer direct programming instruction to the most popular programming language in the business community. SAS is used by 95% of Fortune 100 companies, and these companies are the leading employers of data scientists. There is an incredible opportunity to fill the need of professional data scientists for hands-on machine learning training with real-world examples.

The unfortunate reality for many SAS programmers is that we often do not have access to the latest and greatest SAS products. SAS Enterprise Miner, SAS Visual Analytics, SAS Forecast Server, and SAS Viya are all incredible products, but they are not universally available to all SAS programmers. It is essential that a data scientist who is working in a SAS environment be able to develop and implement machine learning models in any SAS environment. Even if data scientists have access to SAS Viya, it is incredibly beneficial for them to have a solid understanding of the programming code that drives the models that they develop in SAS Viya.

This book, *End-to-End Data Science in SAS®,* provides all SAS programmers insight into the models, methodology, and SAS coding required to develop machine learning models in any industry. It also serves as a reference for programmers of any language who either want to expand their knowledge base or who have just been hired into a data scientist position where SAS is the preferred language.

The goal of this book is to provide clear and practical explanations of the data science environment, machine learning techniques, and the SAS code necessary for the proper development and evaluation of these highly desired techniques. These explanations are demonstrated with real-world business applications across a variety of industries. All code and data sets are publicly available in a dedicated GitHub repository.

Is This Book for You?

If you are interested in this book, then you (or most likely the organization that you work for) have SAS installed on your computer. However, not all SAS installations are created equal. Some programmers work in Base SAS (also called PC SAS). Others have a variety of SAS software available to them:

- SAS Enterprise Guide
- SAS Enterprise Miner
- Visual Analytics
- SAS Studio
- SAS Viya

We are only limited by our ingenuity.

This list is just a sample of the many SAS products available. In addition to these products, there are several software components that SAS offers:

- SAS/ACCESS software
- SAS/ETS software
- SAS/IML software
- SAS ODS Graphics Editor
- SAS/OR software
- SAS/STAT software

Your company's IT department generally dictates the SAS products that you have and the software components that are available to you. If you desperately want SAS Viya or SAS/ETS, you will often have to "fight the power" to get it. I sincerely hope that you can access one or many of these SAS products because they are awesome, and they will make your life as a data scientist much easier and much more productive. However, if you are like me and you have to develop predictive models without the benefit of all the toys that SAS has to offer, then this book is what you have been waiting for.

SAS Software Requirements

The minimum requirement for the majority of procedures detailed in this book is SAS 9.2 with SAS/STAT installed. This requirement should cover most SAS users. With this minimum requirement, we will be able to develop:

- Linear regressions
- Logistic regressions
- Clustering
- Decision trees

Some of the more advanced procedures will require SAS Enterprise Miner to be installed. These procedures will include:

- PROC HPFOREST
- PROC TREEBOOST
- PROC SVM

Don’t panic! Just because you are limited to Base SAS with SAS/STAT installed does not mean that we are limited to the predefined procedures that come with SAS/STAT. We are limited only by our ingenuity. I will include methods of how to perform some of these more advanced procedures by developing them from scratch, using only the procedures available in SAS/STAT. The code to create these procedures from scratch can get pretty complicated, but I’ll provide step-by-step explanations and all code will be available in the repository.

Programming Knowledge Assumed

If you only have experience working with Microsoft Excel spreadsheets, then don’t worry; we can get you up and running in SAS. However, if you’ve never worked with data in any capacity, including Excel spreadsheets, then maybe this book isn’t for you.

I will assume that you have some experience writing basic formulas. For example, in Excel:

```
=AVERAGE(B12: B32)
=SUM(A1: A14)
```

I will also assume that you can perform basic computer commands such as create a new file, save a file, open a file, and so on.

As long as you have met these minimum requirements and have a desire to learn awesome new skills that are guaranteed to increase your value to any organization, then we are good to go.

Icons Used in This Book

Warning This icon indicates a warning or a difficult subject

Decision Time This icon indicates a decision that the data scientist has to make

Example Code and Data

All data used for examples in this book have been accessed through public-use data repositories. These repositories provide a wide range of data sets that span across a variety of disciplines. The code that is demonstrated in the book is primarily created by the author. Any code that has been shown that was not created by the author has been cited with the appropriate credit to the developer. There is a fantastic community of SAS developers who openly share their code with the world. I strongly encourage you to search the web and connect with these communities on the SAS website, GitHub, Stack Overflow, and several other communities.

You can access the example code and data for this book by visiting the GitHub repository at https://github.com/Gearhj/End-to-End-Data-Science.

SAS University Edition

This book is compatible with SAS University Edition. If you are using SAS University Edition, then begin here: https://support.sas.com/ue-data.

We Want to Hear from You

Do you have questions about a SAS Press book that you are reading? Contact us at saspress@sas.com.

SAS Press books are written *by* SAS Users *for* SAS Users. Please visit sas.com/books to sign up to request information on how to become a SAS Press author.

We welcome your participation in the development of new books and your feedback on SAS Press books that you are using. Please visit sas.com/books to sign up to review a book.

Learn about new books and exclusive discounts. Sign up for our new books mailing list today at https://support.sas.com/en/books/subscribe-books.html.

Learn more about this author by visiting his author page at http://support.sas.com/gearheart. There you can download free book excerpts, access example code and data, read the latest reviews, get updates, and more.

Author Acknowledgments

I would like to thank SAS for allowing me the opportunity to publish this book through SAS Press. It is an honor to be part of the SAS author community. I would also like to give special thanks to my editor Suzanne Morgen for her patience and her incredible eye for detail.

Additional thanks are given to all of the reviewers who painstakingly reviewed raw chapters and provided needed critical feedback:

- Christopher Battiston
- David Ghan
- Funda Gunes
- Sunil Gupta
- Mark Jordan
- Robin Langford
- Premkumar Varma

I would still be a poor musician in Warren, Ohio if it were not for Lorin Ranbom, Mitali Ghatak and the Health Services Research Section at the Ohio Department of Jobs and Family Services who took a chance on hiring me right out of college. Thanks to Dan Hecht, Donna Bush, Lora Summers, Kendy Markmen, Dave Dorsky, Hope McGonigle, Tracy Cloud and Eric Edwards for introducing me to SAS and providing me all of the support that I needed.

A continuous thank you to the best SAS programmer that I ever met, Matt Bates. I cannot count how many times that I bothered Matt about how to perform tasks from the mundane to the insanely complex. He has always come through for me and I am eternally grateful.

A huge thank you to Mihai Gavril, Vijay Narapsetty, Dan Gedrich, and Jay Das for providing a work environment that is collaborative, innovative and focused on excellence. I couldn’t ask for a better team.

Thank you to my parents Richard and Lou Ann Gearheart for all of their love and support and my brothers Scott, Glen, Kenny, George, and Tod. You have always been there for me.

My beautiful son, Jacob Gearheart, is the pride of my life and I am amazed every day with your wit, charm, intelligence, and kindness. I pray that I will be more like you every day.

And of course, the love of my life, Tanya Gearheart. I must have written you a thousand love letters and none of them have come close to how much that I love you and need you in my life. You are my entire world and the only reason that I wake up every day is to spend more time with you. Always, madly.

About The Author

James Gearheart is an experienced Senior Data Scientist and Machine Learning Engineer who has developed transformative machine learning and artificial intelligence models. He has over 20 years of experience in developing and analyzing statistical models and machine learning products that optimize business performance across several industries including digital marketing, financial services, public health care, environmental services, social security and worker's compensation.

James has been both a data science manager and an individual contributor on projects ranging from logistics, anti-money laundering, CRM, cash demand forecasting, customer segmentation, digital marketing, A/B testing, attribution and attrition. His research interests include applied data science, machine learning, and artificial intelligence.

Learn more about this author by visiting his author page at http://support.sas.com/Gearhart. There you can download free book excerpts, access example code and data, read the latest reviews, get updates, and more.

Chapter 1: Data Science Overview

Introduction to This Book

Over the past decade, there has been an explosion of interest in the application of statistics to the vast amounts of data being generated every second by nearly everyone on the planet. This interest has been driven primarily by tech giants that learned how to monetize this information by providing free services to individuals in exchange for the data that these individuals generate. Every click, tweet, picture upload, page view, like, share, follow, purchase, comment, retweet, email, and logon is stored in vast data warehouses maintained by tech giants and made available to their army of data scientists to turn your data into actionable insights.

You may feel flattered that you have a secret admirer that obsesses about your every click, your interests, your hopes and dreams, your wants, your insecurities, and your wildest fantasies. You may envision tech billionaires like Mark Zuckerberg or Jeff Bezos lying awake at night thinking about you and how to give you exactly what you want. Maybe that image is a bit of a stretch, but these CEOs, and all of their competitors, do have the customer at the center of their business models. This phenomenon is not limited to tech companies. Traditional brick and mortar stores (Walmart, Target, Best Buy), telecommunication companies (AT&T, Verizon, Comcast), medical institutions, entertainment conglomerates, utility companies, and government offices all have teams of data scientists whose sole function is to know what you want before you even know it yourself.

Amazon knows that because of your interest in data science books that you might be interested in this one. Netflix knows that because you binged the entire first three seasons of *Black Mirror* over the past weekend, that you will also be interested in the reboot of *The Outer Limits*. These insights are the result of teams of data scientists who access the information that you have freely provided and develop models that categorize individuals into similar groups based on a variety of attributes. They also develop predictive models that try to determine what you will do or want in the future.

Don't believe me? Here is a quick test. Based on the only information that I have about you (your interest in data science books focusing on SAS), I will assume that you are a male aged between 25 and 40 years old with graduate-level education and live in an English-speaking country. You probably work in a large corporation. Maybe your desk is a bit messy, but you have a method to the madness.

How'd I do? If these guesses were a complete miss, then you must be a unique individual to not fall into any of the traditional demographic and interest categories of data scientists. But I would guess that at least half of these descriptions pertain to you. These guesses might appear to be a bit broad or general. But remember, they are based on only a single general piece of information: your interest in data science books focused on SAS.

Now imagine that I have a database of your entire web history, including all your past purchases, your social connections, your financial statements, your search terms, your emails, and your viewing habits. Imagine that I took all of this information and input it into a complex predictive model that was built with the sole purpose of calculating the probability that you would be interested in a specific product. Imagine that I gathered this same database of information for ten million individuals (a small sample size in the world of digital information where the data sets can contain several trillion data points) and built a deep learning model that calculated the probability that you would be interested in a travel rewards credit card offer with a 2% cashback program.

That's a pretty specifically defined target variable! Given the quantity and quality of information that I have about the population and a well-defined target variable, I should be able to build a predictive model that can calculate, with a reasonable degree of confidence, the probability of an individual responding to this credit card offer. This information could be used to allocate my marketing budget efficiently. I'd probably achieve a higher response rate with this approach than if I spent the budget on a radio commercial broadcast to a broad population. If the response rate of the travel website campaign is 1.5% while the response rate of the radio ad is 0.03%, then our predictive model provided a **50X** lift in response rate! That is one way to optimize your marketing budget.

In this example, all of this data has been used to deliver a highly targeted ad for the right product to the right individual at the right time with the right message. This targeting mechanism is just one example of how data science is used every day by a wide variety of businesses. Major corporations have used data science techniques to target, classify, segment, cluster, recommend, adjust offers, personalize, upsell, cross-sell, influence and predict every aspect of customer behavior.

Minimum Effective Dose

This book will follow the Minimum Effective Dose (MED) method of teaching data science. The goal of this method is to get you up and running with sufficient background information on the various data science techniques and provide you with specific examples and access to real data and also provide SAS code that is thoroughly explained. I do not want you to get bogged down with a bunch of mathematical theories on the inner workings of the procedures. We will touch on the formulas, but I will not drone on about it. If you are interested in a thorough explanation of the mathematics of data science, I suggest that you check out *The Elements of Statistical Learning* by Hastie, Tibshirani, and Friedman. It is the bible of statistical mathematics for the data science community.

So, what does MED mean?

- No math! (well...very little math)
- An understandable and practical explanation of data science methods:
    - Which method to use for a given situation
    - What to watch out for
    - How data science works with real business data (you will not find the standard academic data sets such as the iris data set or the MNIST data set in this book)
- Step-by-step instructions on how to do specific procedures
- How to evaluate the performance of your model:
    - How to apply your model to new data
    - How to understand the SAS code provided

The Current Data Science Landscape

Over the past few years, the job titles for business analysts have changed. These changes in job titles reflect the additional demands of the business analyst. If you were in the business analytics field before 2012, maybe you've had one of these job titles:

- Analyst
- Statistician
- Quant
- Researcher
- Analytical Consultant
- Data Architect
- Database Administrator
- "Data Guy"

Since 2012, many of these job titles have changed to:

- Data Scientist
- Machine Learning Engineer
- Artificial Intelligence Engineer

One of the first questions that you may have is, "what is the difference between a data scientist and an analyst"? The initial answer is that 80% of their work is the same:

- Access data
- Clean data
- Combine data sets
- Summarize data
- Answer business questions
- Create reports

However, data scientists have some exciting tools in their toolbox, including:

- Accessing and using unstructured data: Web scraping, Twitter tweets, Facebook comments, and so on
- Making sense of data that they do not know using unsupervised machine learning methods: Clustering and PCA
- Predicting future outcomes: Forecasting and predictive modeling

These additional functions of the data scientist take their role to a new level by being able to move past the traditional descriptive statistics of reporting and move into the exciting world of predictive analytics, prescriptive analytics, and optimization.

Types of Analytics

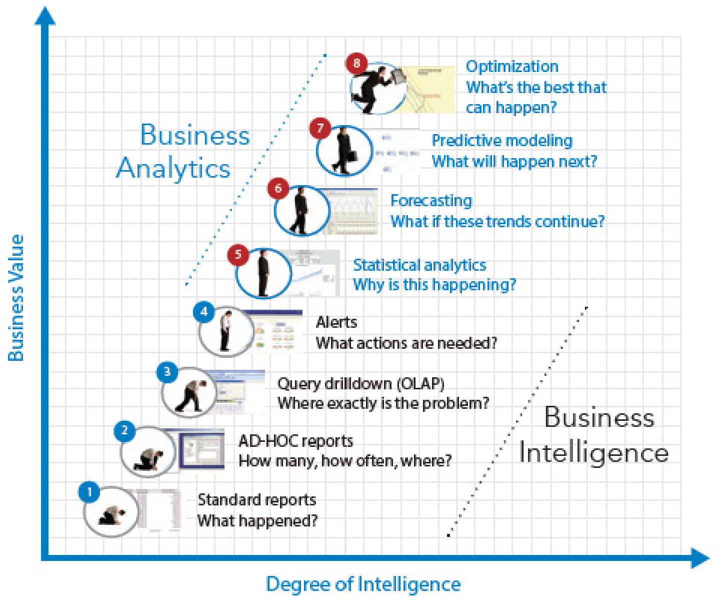
Let’s look at a few different types of analytics that data scientists should be familiar with.

- **Descriptive Analytics** – This is a retrospective look at what has happened in the past and what is currently happening. Firms generate a lot of descriptive analytics to understand business trends, customer demand, production thresholds, and countless other business metrics. Descriptive reporting is not flashy, but it is the life's blood of all business. Since these are events that have already been realized, this reporting focuses on reporting actual figures along with their associated statistics (averages, standard deviations, and so on).
- **Predictive Analytics** – Historical data is used to make predictions about the future. These are statistical estimates of future possibilities based on historical data. This type of reporting is accompanied by confidence intervals, odds ratios, and probabilities.

- **Prescriptive Analytics** – Also called *optimization*, prescriptive analytics identifies impactful factors and recommends the optimal values of these factors to achieve a predefined result.

SAS created Figure 1.1. This figure breaks down the previous list even further into eight levels of analytics. By increasing the level of intelligence in these analytical products, we are expanding the business value of analytics to the organization. The first four stages are based on reporting functions of analytics, while the last four stages are based on probabilistic functions of analytics. The business value of these functions is incredibly beneficial to businesses.

Figure 1.1: Eight Levels of Analytics

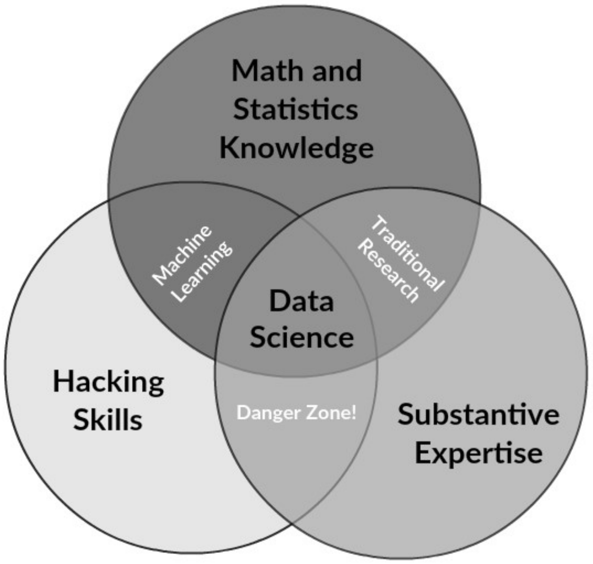

Business Value
Degree of Intelligence
Business Analytics
Business Intelligence
8 Optimization What's the best that can happen?
7 Predictive modeling What will happen next?
6 Forecasting What if these trends continue?
5 Statistical analytics Why is this happening?
4 Alerts What actions are needed?
3 Query drilldown (OLAP) Where exactly is the problem?
2 AD-HOC reports How many, how often, where?
1 Standard reports What happened?

For example, if you are at the horse track and getting ready to place your bet, would you rather have a report on the historical performance of the horses, or would you rather have some probabilistic information on what might happen in the upcoming race? My money would be on the probabilistic report.

Data Science Skills

I’m sure that most of you are as tired as I am of looking at the graphic in Figure 1.2 below. We have seen it on every data science blog, presentation, and instructional medium. Regardless of the ubiquity of the graphic, there is still a clear message that is conveyed. It shows the vital interplay and intersection of three diverse fields that had traditionally been performed separately.

Figure 1.2: Data Science Venn Diagram



Adapted from: Conway, Drew. (September 2010). *The Data Science Venn Diagram*. Retrieved from http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram.

The main sections in the diagram in Figure 1.2 are as follows:

- **Computer skills** – represented in the “Hacking Skills” section of the diagram. This skill set is essential for accessing and manipulating structured and unstructured data. Data scientists need to be able to gather data, clean it, merge it with other data sources, prepare it for modeling, and implement machine learning algorithms on this data to answer a specific question.
- **Math and Statistics Knowledge** – A data scientist needs to have the prerequisite statistical background to understand the goal, process, and function of a predictive algorithm.
- **Business knowledge** – represented in the “Substantive Expertise” section of the diagram. In this context, the term “business” can represent any field of study that requires knowledge of that specific environment. Machine learning models are developed to answer business questions. A data scientist needs to have knowledge of that environment to develop the initial business questions that can be modeled. This

type of knowledge is critical for knowing what type of data will be required to answer the question, what direction the data should be moving, and what answers are counterintuitive.

The subsections in the diagram in Figure 1.2 are:

- **Machine Learning** – The intersection of "Hacking Skills" and "Math and Statistics Knowledge" excludes the "Substantive Expertise" skill set. This subsection is a reference to computer scientists who can develop sophisticated predictive models based solely on finding correlations in given data sets. Regardless of how sophisticated or accurate these models may be, these models lack the background business knowledge and are therefore prone to spurious correlations, weak inferences and ethical concerns.
- **Traditional Research** – The intersection of "Math and Statistics Knowledge" and "Substantive Expertise" is excluding the "Hacking Skills" section of the graph. Before the application of computers to solve all our daily problems, researchers had to apply mathematical knowledge to their field of expertise to develop theoretical models. If you received your advanced degree before 1995, then I'm talking about you.
- **Danger Zone** – Certainly my favorite section of the graphic. The intersection of "Hacking Skills" and "Substantive Expertise" that excludes the "Math and Statistics Knowledge" highlights the dangerous area of misaligned models, misinterpreted results, and unfounded conclusions.

Introduction to Data Science Concepts

The language of data science is based on a series of concepts that represent the issues that data scientists have to consider when they are developing a project. These terms and definitions are a kind of shorthand for deep theoretical issues that data scientists must consider at each step of their project. For example, if I complain to another colleague that my go-to regression model is not working well on this new project, he might respond, "No free lunch." That doesn't mean that he assumed I was going to steal his lunchbox!

In this section, I will outline some of the most common concepts in data science so that we have a common understanding of these concepts.

Supervised Versus Unsupervised

There are two main categories that machine learning models can fall into. These are supervised and unsupervised models. The presence of a target variable differentiates them. The target variable represents the item that you are interested in. It is also called the response variable. If there is a value that you are trying to predict, you have a supervised method. Supervised models have a target variable, while unsupervised models do not have a target variable.

Many data science terms are used interchangeably. Predictor variables are also called descriptive variables, independent variables, or input variables. The target variable is also called the response variable or the dependent variable.

Supervised Models

Most of the models that you have been exposed to have been supervised models. These are the main predictive models that filter out your spam, identify your handwriting, recommend your next song, decide which ad to show you, and make a thousand other selections for you every day.

These models include, but are not limited to:

- Regression Models
- Decision Trees
- Random Forests
- Gradient Boosting
- Neural Networks
- Support Vector Machines (SVM)
- K-Nearest Neighbors
- Naïve Bayes Models

Each one of these models is built on a data set that has a target variable along with one to several thousand predictor variables. If a data set contains a target variable, then supervised machine learning algorithms are most commonly used with this type of data. These machine learning algorithms identify the associations between the predictor variables and the target variable. The result is a formula that can be used on new observations where the target value is not known.

For example, if you want to predict the quality ranking of a bottle of wine, then you would generally have a data set that contains several observations that have the ranking of wine for each observation. The quality ranking would be the target variable. The data set would also include several descriptive variables along with the target variable. These descriptive variables describe several different factors that influence the target variable. Table 1.1 shows nine predictive variables that are associated with the target variable "quality."

In mathematical terms, the predictor variable would traditionally be labeled as "X," and multiple predictor variables would be listed as X1, X2, X3, and so on. The target variable would be labeled as "y," so that for each observation of the predictor measurement X_i there is an associated response measurement y_i where i represents the ith observation.

Table 1.1: Wine Quality Data

Fixed Acidity	Volatile Acidity	Citric Acid	Sugar	Chlorides	Density	pH	Sulfates	Alcohol	Quality
8.6	0.23	0.4	4.2	0.035	0.995	3.1	0.53	9.70	5
7.9	0.18	0.37	1.2	0.040	0.992	3.2	0.63	10.80	5
6.6	0.16	0.4	1.5	0.044	0.991	3.5	0.52	12.40	7
8.3	0.42	0.62	19.25	0.040	1.000	3	0.67	9.70	5
6.6	0.17	0.38	1.5	0.032	0.991	3.3	0.55	11.40	7
6.3	0.48	0.04	1.1	0.046	0.993	3.2	0.36	9.60	6
6.2	0.66	0.48	1.2	0.029	0.989	3.3	0.39	12.80	8

In the case of our wine quality prediction example, you could build a regression model that results in a formula like this:

$$\begin{aligned} Quality = 0.796 &+ FixedAcidity(0.0349) + VolatileAcidity(-0.8287) \\ &+ Cholorides(-0.0464) + pH(-0.203) + Sulphates(0.1386) \\ &+ Alcohol(0.4663) \end{aligned}$$

This formula states that the quality score of wine is a function of six main features of the wine. The model determines which features are associated with the wine quality score by assessing the correlation between each feature and the quality score. The model also produces model weights (also called coefficients) that are used to predict the quality score of new bottles of wine that have not been scored yet.

If we were given an information sheet on a brand new bottle of wine that has not been scored for quality, then we would be able to input the information from each of these predictors into our formula and predict the quality of the wine, as shown in Table 1.2.

Table 1.2: Wine Data Set Predictive Weights

		New Wine Info	Predictive Weights
	Model Intercept	Unknown	0.796
Predictors	**fixed acidity**	7.2	0.0349
	volatile acidity	0.22	-0.8287
	chlorides	0.042	-0.0464
	pH	3.2	-0.203
	sulphates	0.61	0.1386
	alcohol	10.2	0.4663

		Predicted Score
quality (Target)		5.0542

Each value of the new wine information is multiplied by the predictive weight (coefficient) produced by the model. These calculated values are combined with the model intercept to generate a predicted score.

According to our predictive model, this new bottle of wine should have a quality score of 5. Notice that not all the predictive variables were used in the final predictive model. The model retained only the predictors that had a statistically significant relationship with the target variable. Don't worry; these concepts will be explained in detail in the regression modeling chapter of the book.

Unsupervised Models

Unsupervised models are built on data sets that do not have a target variable. This type of analysis is slightly more difficult because of the lack of direction that is usually provided by the target variable. The goal of unsupervised models is inherently different from supervised models. Supervised models are generally constructed to predict the response of future observations or to understand better the relationship between the predictors and the target variable. On the other hand, unsupervised models do not predict anything. They are built to understand the relationship between the given variables. The goal of unsupervised models is to identify naturally occurring groups or classifications of the observations.

The most common types of unsupervised learning models include:

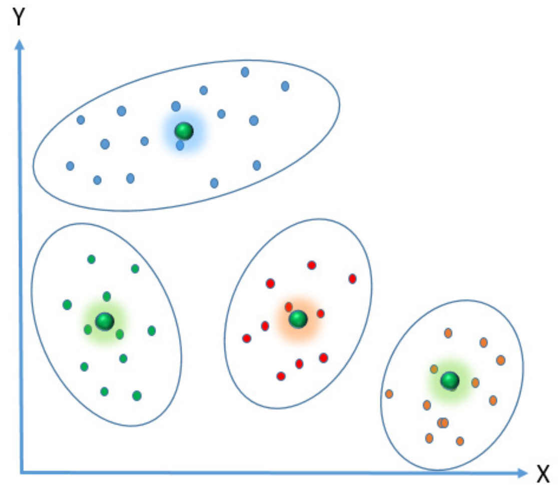
- k-Means Clustering
- Hierarchical Cluster Analysis (HCA)
- Principal Component Analysis (PCA)
- t-distributed Stochastic Neighbor Embedding (t-SNE)

For example, suppose that you have a database of customer attributes and you want to know if you can detect groups of similar customers. These groups could be beneficial in marketing campaigns or for targeting purposes. It is a common practice to run a *k-means clustering algorithm* that will assign each observation to one of *k* groups based on the features that are provided. The "*k*" in k-means clustering represents the number of groups that you want to be assigned. Data points that are relatively close to one another in the feature space are assigned to the same group or cluster.

The takeaway from this section is that the clusters form organically without the direction of a target variable. The relationship between variables is the basis of unsupervised learning.

One of the nice features of clustering algorithms is that they are commonly expressed through visualizations. The clustering algorithm is easily understood in a scatter plot visualization where the observation points can be colored or symbolized according to the cluster that they have been assigned. Figure 1.3 shows a typical clustering model visualization. This visualization demonstrates four distinct groups of observations based on the X and Y features. Each of the groups contains a centroid point that establishes the cluster in the feature space.

Figure 1.3: Clustering Model Visualization



One of the most powerful applications of unsupervised learning models is *dimensionality reduction*. It is not uncommon to have a modeling data set with hundreds of predictors. Although we always want more data, the truth is that more data can lead to more problems. A large number of predictors quickly runs into the curse of dimensionality (explained in detail in the next few sections) and issues of correlation. Dimensionality reduction is the ability to simplify the data without losing too much information.

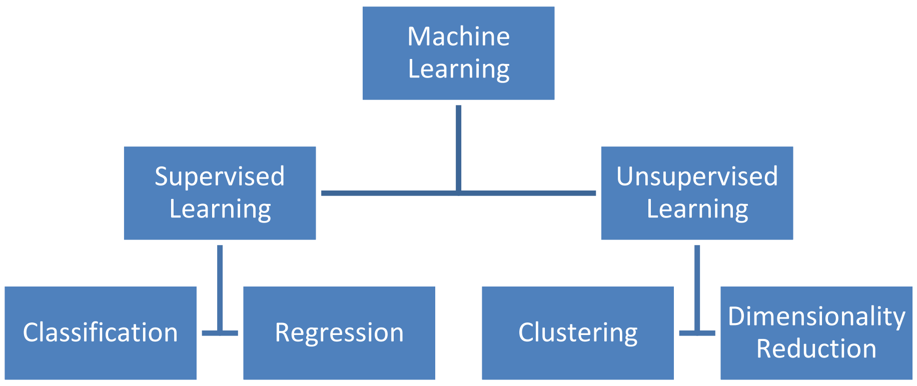
Algorithms such as *principal component analysis* (PCA) collect correlated variables and merge them into a single feature that represents an important dimension of your data. For example, the square footage of a home and its number of bedrooms are highly correlated. These features can be combined into a single attribute that might represent the family size.

Although the purpose of supervised learning models and unsupervised learning models are inherently different from one another, they often work together. A data scientist can employ unsupervised modeling techniques to develop clusters and perform dimensionality reduction, which will create features that can be used as inputs to a supervised learning model.

Machine Learning Categories

Figure 1.4 demonstrates the high-level categorization of machine learning algorithms. The broad category of machine learning can be broken into supervised learning and unsupervised learning models. Supervised learning can be further broken into classification and regression models. Unsupervised learning can be further broken down into clustering and dimensionality reduction models. Each of these concepts will be explained in detail in the upcoming sections.

Figure 1.4: Machine Learning Categories



Parametric Versus Non-parametric

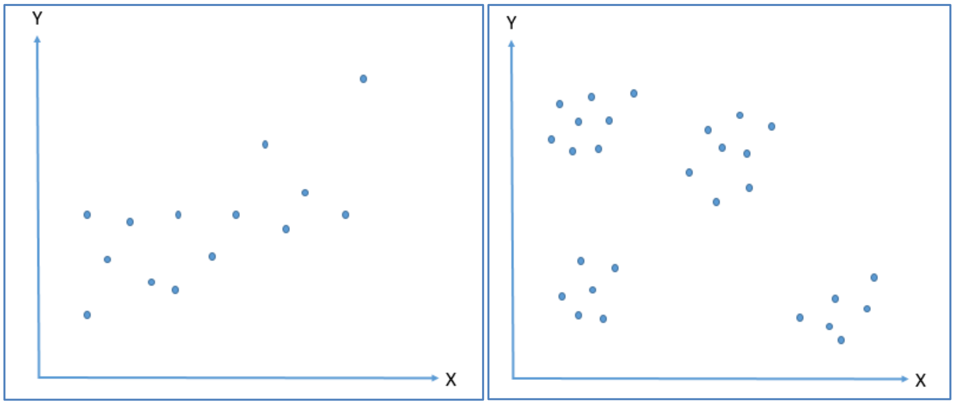
Once you examine your data and decide that you are going to build a supervised model, the next step is to try to determine what kind of model will best fit the data. Several methods will be covered later in this book that pertain to variable examination, plotting data, and correlation analysis. When you perform these techniques, you are trying to understand the *functional form* of the relationship between the predictors and the target variable. When you examine the data, you need to ask yourself:

- Is the data linear?
- Is the data clustered?
- Is the data curved?
- Does it look like there is a pattern to the distribution?
- Is the data random?

These methods will provide you with clues as to what type of model you should apply to the data. Figure 1.5 shows two very different types of data distributions. The graph on the left shows a linear relationship between the X and Y variable, while the graph on the right does not display a linear relationship.

When we are trying to decide what type of model to apply to our data, we need to make assumptions about the functional form of the relationship between X and Y. A linear model will attempt to draw a straight line through the data that has the shortest distance between the line and the observation points. If we applied a linear regression model to the first set of data, I think that we would get a decent fit. However, if we applied a linear regression model to the second set of data, we would get a very poorly fit model.

Figure 1.5: Data Distribution Types



There are two general classifications that models fall into. These are *parametric* and *non-parametric* models.

Parametric Models

Parametric models make an assumption about the functional form of the underlying relationship between the predictors and the target variable. Through your examination of the data, you might determine that there exists a linear relationship between the target and the predictors. In this case, you are assuming a linear functional form:

$$f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

Since we assumed that $f(X)$ is linear, then we just need to estimate the coefficients, $\beta_0, \beta_1, \ldots \beta_p$. This is most commonly performed through *ordinary least squares* (this concept will be explained in detail in the regression chapter).

Regression models are classified as *parametric* because they reduce the *p*-dimensional function (p is the total number of predictors) down to estimating a given set of parameters.

Parametric models include, but are not limited to:

- Linear regression
- Logistic regression
- Naïve Bayes
- Linear Discriminate Analysis
- Simple neural networks

The benefits of parametric models are:

- **Transparency** – These models are relatively simple and easy to understand. You know exactly what the inputs are and what the coefficients are.
- **They are faster** – Since these models do not have to search through the entire feature space to find an arbitrary *p*-dimensional function, they are much faster to train than non-parametric models.
- **They need less data** – Parametric models can be developed on a limited set of data compared to non-parametric models. Although the coefficients will adjust as more data is added, there is a point of diminishing returns where additional observations will not yield discernable model fit improvements.

Non-Parametric Models

If parametric models assume the functional form of f, then you could probably guess that non-parametric models do not assume the functional form of f. These models are free to learn any functional form of the data because they are not constrained by assumptions. Without the restrictions of a predetermined functional form, these algorithms attempt to estimate a function that gets as close as possible to the training data but will generalize well to unseen data. The significant advantage of non-parametric models is that they can potentially fit a much wider range of functional forms than parametric models.

For example, a nearest neighbor's model does not assume any functional form of the data. It just looks for observations in close proximity to one another in the feature space, regardless of how the data is distributed. The only assumption that is made is that observations close to one another in the feature space have a similar target value.

Non-parametric models include, but are not limited to:

- k-Nearest Neighbors
- Decision Trees
- Complex Neural Networks
- Support Vector Machines (SVM)

The benefits of non-parametric models are:

- **Flexibility** – These models do not assume the functional form; therefore, they can detect a wide range of functional forms.
- **Accuracy** – These models generally result in more accurate models.

Non-parametric models can be applied to linearly distributed data. The disadvantages of applying a non-parametric model to a data set that could be modeled by a parametric model are:

- **Data requirements** – Non-parametric models generally require more data to adequately train a model. If the size of the data set is limited, the parametric model will provide a more accurate model fit.
- **Training Time** – Since non-parametric models do not assume the functional form of the data, these models have to search the *p*-dimensional feature space to assess the best model fit.
- **Transparency** – Many types of non-parametric models are called *black-box* models because it is difficult to understand how the model prediction is constructed. The parametric model can be easily understood by examining the weights of the coefficients. Most non-parametric models do not output model weights and are therefore difficult to interpret.

Table 1.3 provides a summary of the differences between parametric and non-parametric models

Table 1.3: Parametric versus Non-Parametric Models

Model Type	Data	Speed	Transparency	Functional Forms	Accuracy
Parametric	Less	Fast	Transparent	Limited	Less
Non-Parametric	More	Slow	Black Box	Unlimited	More

Regression Versus Classification

Variables can fall into one of two main categories. *Quantitative* variables take on numerical values. These values are continuous values (1.1, 1.2, 1.3 …). Examples of these types of variables are height, weight, money, distance, time, and any other quantity that can be expressed as a value.

Variables can also be *qualitative*. These variables represent a value in a category. For example, a home price variable could be categorized as high, medium, or low. A person's gender can be categorized as male, female, or other. Any descriptive label can be made into a qualitative variable.

Machine learning models can be categorized as either *regression* models or *classification* models. This categorization depends on whether the target variable is quantitative or qualitative. If a data set has a target variable that is a continuous value, such as "annual income," then the machine learning model that is built on this data set will be a regression model. On the other hand, if a target variable is qualitative, such as "yes" or "no," then the machine learning model will be a classification model.

The terms "regression" and "classification" can lead to some semantic problems and muddy our understanding of these models. A linear regression model will most likely have a continuous target variable; however, a logistic regression model will often have a binary target variable. Even though they both have the term "regression" in their name, they are performing two very different calculations. A linear regression model is a regression model, while a logistic regression model is actually a classification model.

Also, many machine learning algorithms can perform both regression and classification functions. Decision trees, k-nearest neighbors, neural networks, and random forests can all be applied to either quantitative or qualitative target variables.

My best advice is not to get caught up in the names of the models. Once you understand how each of the models function and their relative strengths and weaknesses, you will have a much better feel for which model to use in a given situation.

Table 1.4 provides some general descriptions of regression and classification models.

Table 1.4: Regression and Classification Models

	Regression	**Classification**
General Description	Regression means to predict the output **value** using training data	Classification means to output the group into a **class**
Example	Regression to predict the **value** ($ amount) of an insurance claim using training data	Classification to predict the **type** of an insurance claim (fraud vs non-fraud) using training data
Target Variable	If it is a real number/continuous, then it is a regression problem	If it is a discrete/categorical variable, then it is a classification problem

Overfitting Versus Underfitting

The concepts of overfitting and underfitting are some of the most important issues that data scientists have to understand. A considerable amount of a data scientist's work is spent evaluating whether their model is overfitting or underfitting the data. To understand these concepts, we first need to understand the difference between the terms "signal" and "noise."

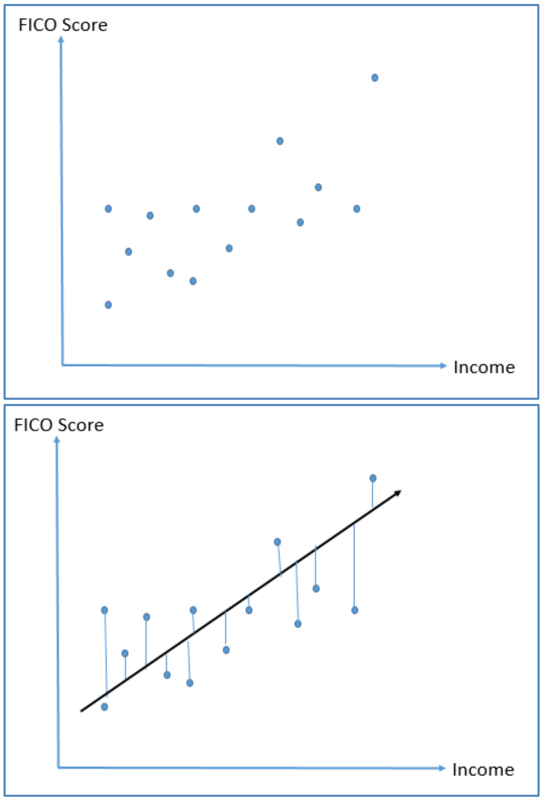
Every machine learning model attempts to capture the underlying relationship between the target variable and the predictor variables. If a relationship exists, then the model should identify this relationship and use it to predict new cases. However, in every data set, there exists a certain amount of random error. This random error can come from variables that have only a weak association or no association at all with the target variable. It can also come from outliers, skewed data, and a myriad of other data issues. This random error is called "noise."

Overfitting

When a machine learning model describes the random error (noise) rather than the underlying relationship of the predictors and the target, the model is said to be overfitting. Overfitting often occurs when a model is too complex for a given set of data. If the machine learning model incorporates too many predictors relative to the number of observations, it will begin to model the random error instead of the desired underlying relationship. The overfitted model will be highly tuned to the data set that it was developed on (called the training data set). However, it will not generalize well to new data. This lack of generalization will lead to poor model performance on new data.

Let's look at a quick example. In the finance world, we often use a customer's FICO score to determine credit worthiness. The FICO score is a three-digit number between 350 and 800 that is calculated and reported by the credit bureau agencies and represents a customer's payment history, credit tenure, credit delinquencies, and many other financial factors. We can construct a very simple data set based on the relationship between a customer's FICO score and their income (Figure 1.6).

Figure 1.6: Simple Linear Relationship



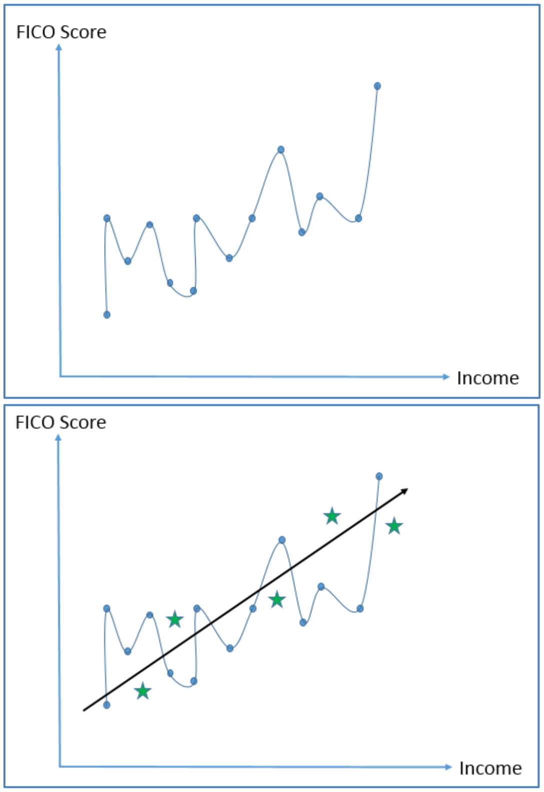
The graph on the top of Figure 1.6 shows a scatter plot graph of the distribution of FICO score and income. We can see that there is a positive relationship between the predictor variable (Income) and the target variable (FICO score). Although there is a positive relationship, it is not a one-to-one perfectly fit relationship. We can see that there is a general upward trend, but it does not look like a perfect match between the two variables.

The graph on the bottom of Figure 1.6 shows a linear trend line and error bars that demonstrate the residual error of the linear model. If you remember your Statistics 101 class, several different lines can be fit to this data set, but the best fit is the one that has the lowest total residual errors. We can easily find this by calculating the distance between the observation and the line; then we square this difference to make sure that we have positive values and then we add them all together to get the total residual error.

So, you might ask yourself, what if I created a polynomial model (a curved line created from polynomial variables) where my line goes through every data point? That would mean that I would have a total residual error of zero. That would be awesome! Unfortunately, that would not be awesome. Even though you would be meeting your goal of reducing the residual error, your model would be so highly tuned to the data set that it was developed on that it would not generalize well to new data.

Figure 1.7 provides a visual example. The graph on the left shows a polynomial line that goes through all data points and has a residual error rate of zero. However, the graph on the bottom shows new data points represented as stars. The polynomial line does not provide a good prediction for these new data points, but the linear prediction does provide a good estimate of the predicted value. Using the linear model, I can predict the value of a customer's FICO score if I am given their income. If I used the polynomial line, my prediction would be much worse.

Figure 1.7: High Degree Polynomial Model



There are several methods to detect overfitting, and there are easy solutions to this important issue. These methods will be described in detail later in the book, but a few highlights are:

- Cross-validation
- Simplifying the model by reducing the number of parameters or reducing your higher-order polynomial
- Regularization

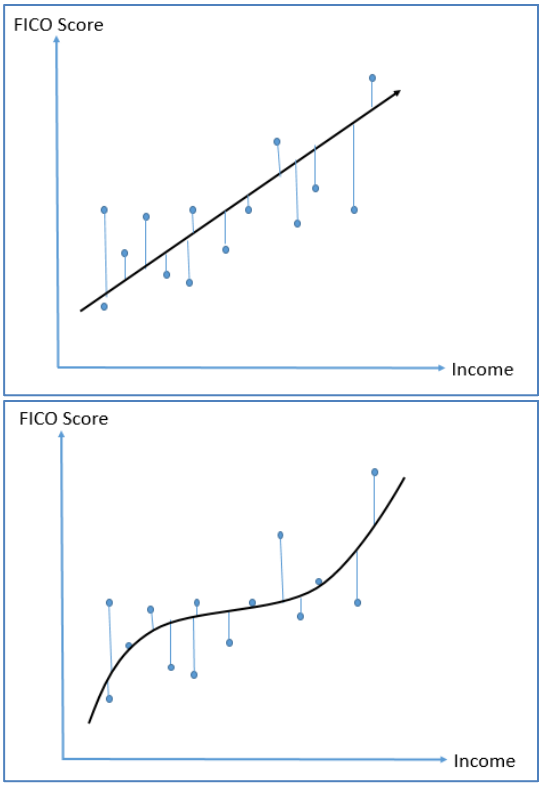
- Cleaning your data set by finding and fixing the outliers and skewed data elements
- Getting more data (more observations, not necessarily more parameters)

Underfitting

The concept of underfitting is the polar opposite of overfitting. In the case of underfitting, the model is not identifying the underlying relationship between the predictors and the target. It is not detecting the signal that is present in the data. In layman's terms, you have a weak model.

If we use the same example as we did in the overfitting demonstration, we can see that the linear model is actually underfitting the data. Figure 1.8 compares the residual error generated from the linear model to the residual error generated from a lower-degree polynomial model. This moderate polynomial model captures some of the dynamic relationships between the predictor and the target that the linear model does not capture. The overall residual error of the moderate polynomial model is lower than the linear model; therefore, it is a better fit.

Figure 1.8: Lower Degree Polynomial Model



There are methods to address underfitting the data. These include:

- Increasing model complexity. Either through more parameters, feature engineering, or selecting a more powerful modeling algorithm
- Decreasing regularization (reducing the constraints on the model)

Batch Versus Online Learning

Machine learning models can be further classified into either “batch” or “online” learning models. The difference between these types of models is whether the model can learn from an incoming stream of new data.

Batch Models

In batch models, the algorithm (the model equation that we use to predict our target values) has been previously developed, and new observations are scored according to a predefined model. The positive aspect of this model is that since it has been previously developed, the learning algorithm can consider all the available training data and take as long as necessary to output a scoring algorithm. Some deep learning models have been known to run for over a month before the final model weights are constructed.

By developing the model before implementation, the data scientist can try several different approaches and conduct various types of tests to determine the efficacy of the model prior to implementation. If a data scientist wants to use new data for a batch model, then they would have to train a completely new model from scratch on this new data set.

It is common to train a new data set weekly or even every night on the most recent data available. As long as the modeling pipeline is established and new data can be imported, cleaned, prepared for modeling, modeled, and evaluated, then new models can be implemented relatively seamlessly.
Batch models generally require a lot of data, computational resources, and time to process. These requirements can result in a substantial cost to an organization. Due to these resource issues, model designs have been developed that are able to learn incrementally.

Online Learning Models

A solution to the resource requirements of batch models is to instead feed the model either a continuous stream of new data or to feed the model small batches (called “mini-batches”) of data. This data can be evaluated by the model algorithm and quickly discarded after it has been used. This model design is great for data environments that receive a constant stream of data such as web clicks or stock prices. An online learning model can adapt quickly to changes in the data stream.

One of the main cautions with this type of system is that these models need to be continuously monitored because they are so sensitive to new data. New data can be erroneous, or it can be purposefully harmful (online attacks, fraud, spamming, bad sensors, system gaming, and so on). This type of new data can have a detrimental impact on an online learning model.

Data scientists generally combat these types of runaway scenarios in two ways:

1. **Continuously monitor model results** – Data scientists build systems to monitor model results in real time. These systems will have threshold levels that trigger a warning if a value meets or exceeds a given level.
2. **Adjust the learning rate** – In this context, the learning rate is a measure of how fast the model incorporates new data and adjusts the model results based on this new data. This adjustment can be a delicate balance because if you set the learning rate too high, the model will adapt to new data very quickly and you can get runaway results. However, if you set the learning rate too low, then the model will not pick up on new trends in the data and this could defeat the purpose of implementing an online learning model.

Bias-Variance Tradeoff

Now it's time to talk about one of the most essential concepts in machine learning. A large part of the work that a data scientist does is directly related to the concept of the bias-variance tradeoff.
A data scientist needs to be able to determine which type of algorithm they will apply to a given business problem and how complex that algorithm needs to be. Even a simple model such as a simple linear regression with one predictor variable can become complex if we transform that single variable into a higher-order polynomial. For example:

Simple: $FICO\ Score = Income(x)$
Complex: $FICO\ Score = Income(x)^{10}$

Even though each of the models contains only one variable, the higher-order polynomial is considered a complex model.

Model complexity can be increased by:

- Increasing the number of predictors
- Feature engineering
- Variable transformations
- Interaction terms
- Model construction
  - Number of branches in a decision tree
  - Regularization value in a regression model
  - Number of hidden nodes in a neural network

   - Value of C in an SVM
   - Many, many, many others

The goal of a data scientist is to develop a model that captures the signal in the data while being able to generalize well to new unseen data. Although this might sound easy, there is an unfortunate tradeoff that must occur. To understand this tradeoff, we must define the concepts of *bias* and *variance*. Fortunately for us, these concepts are closely related to *underfitting* and *overfitting* that we just covered a few pages ago.

Bias

Bias occurs when we make simplistic assumptions about the data or apply simplistic models to a data set. These simplistic models are not able to adequately identify the underlying signal in the data. The models will underfit the data and result in non-optimal model predictions.

For example, the data represented in Figure 1.8 shows a linear model fit to the data on the top while a polynomial model is fit on the bottom. The model on the top suffers from high bias because the model is not adequately representative of the underlying data. It suffers from a systemic lack of fit due to poor model choice. As we increase model complexity by transforming the predictor variable into a polynomial, the model will begin to better represent the underlying signal in the data. This increase in model complexity reduces the bias of the model.

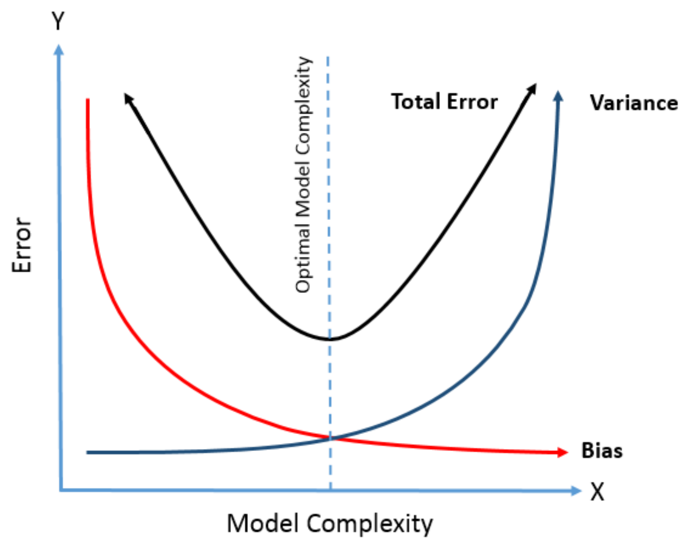
Variance

Variance is the exact opposite problem compared to bias. High variance occurs when you are overfitting the data by applying too many or too strong of data transformations to the training data, or you are using an overly complex model to a data set that does not require it. For example, the predictive model demonstrated in Figure 1.7 shows a high variance model. It is highly overfit to the data and cannot be generalized to new observations.

A common-sense explanation of why we use the term *variance* is that if we **vary** the underlying training data (that is, changed it over and over), we would get very different models each time that we changed the underlying data. This is called “high variance.” This type of model is capturing the noise in the data and not representing the signal.

So, how do we know when we have the right balance of model complexity? Figure 1.9 shows the tradeoff between bias and variance. When model complexity is low, we have a high bias. This is because a more complex model will generally capture the underlying relationship in the data better than a very simple model. A simple model with high bias will have a high total model error.

Figure 1.9: Optimal Model Complexity



On the other hand, an overly complex model will have a high variance. This model is so highly tuned to the training data that it does not generalize well to new observations. An overly complex model with high variance will also have a high total model error.

There exists a balance point where the total model error is reduced to its lowest possible point. This point is the point of *optimal model complexity*. Although this is not the point with the lowest possible bias or the lowest possible variance, it is the balance point between the two. Of course, your next question has to be, *how do I find the point of optimal model complexity*?

Training and Testing Data Sets

An important technique in data science that is reviewed in detail later in the book is to split your data set into two sections. These sections are called the "training data set" and the "testing data set." As a rule of thumb, about 70% of your data will be in the *training* data set. This is the data set that you will use to build your model. The remaining 30% of your data will be in your hold-out *testing* data set. This is the data set that you will apply your model to and evaluate the results.

There are two important caveats to creating your training and testing data sets:

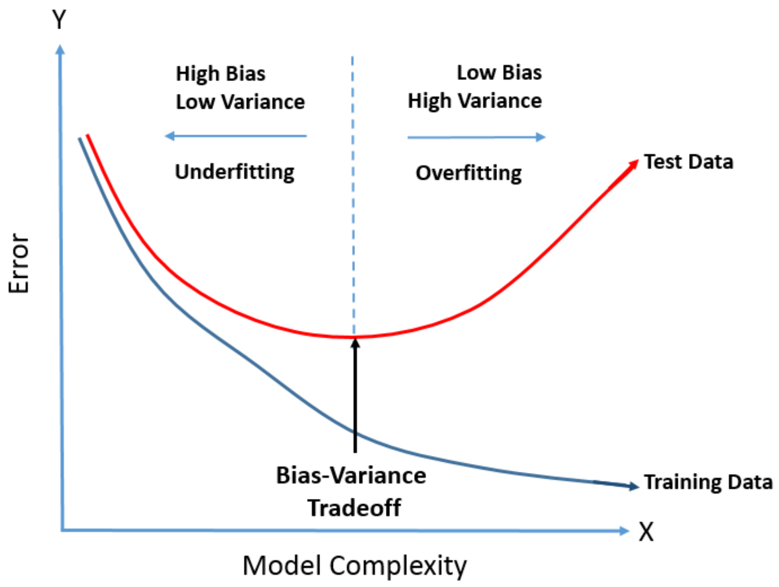
1. The data needs to be randomly assigned to either the training or testing data set. This randomization will avoid selection bias. If the data is not randomly sampled, then the resulting sample data set is not representative of the population intended to be analyzed.
2. No peeking! You are not allowed to evaluate the test data set prior to building the model on the training data set. Any information that you get from investigating the test data set could influence how you build the model on the training data set. This type of

insider knowledge would not be available once the model is in production. This type of cheating can lead to significant performance differences between a model development environment and a production environment.

Figure 1.10 demonstrates how the training and testing data sets are associated with the concepts of underfitting, overfitting, bias, variance, signal, and noise. If we look at the left side of the graph, this section represents models with low complexity (think of a simple linear regression model). When we evaluate the performance of the model when it is applied to both the training data set and the testing data set, there is a high error rate. This simple model does not capture the underlying signal in the data. There is high bias because the model is not capturing the signal and there is low variance because the model results will not change much if we pulled a different sample to build the model.

As we move to the right and increase model complexity, both the training data set error rate and the testing data set error rate begin to fall. As we increase the model complexity, we are capturing more of the signal in the data. A result of increasing model complexity is that the model bias will begin to fall while the model variance will begin to increase. This effect is because we are capturing more of the signal in the data, but the model is becoming more specified to the underlying training data. If we had pulled a different random sample to create the training data, we would get a slightly different model. However, because we are better able to identify the signal in the data, the training error rate and the testing error rate begin to fall.

Figure 1.10: Bias-Variance Tradeoff



As we continue to increase model complexity, both the training error rate and the testing error rate will continue to fall until we hit the point of optimal model complexity. At this point, the model captures the signal in the data, and the testing error rate is at its lowest point. Once we increase model complexity beyond the optimal point, the testing error rate will begin to increase. This increase occurs because the model is now overfit to the training data and is starting to not generalize well to unseen data.

Warning

Interestingly, the training data error rate will continue to fall as we increase model complexity. This is where many inexperienced data scientists fall into trouble. Watching the error rate on your training data set continue to fall as you increase model complexity can give you a false sense of accomplishment. It's easy to fool yourself into believing that you are really nailing it! Be aware that you are actually overfitting your model to the training data set and it will result in very high error rates when your model goes into production and is applied to unseen data.

Step-by-Step Example of Finding Optimal Model Complexity

Ok, that's a lot of words, and if you are like me, then you are easily confused by lengthy explanations. I learn much better through pictures. So let's see a visual example of adjusting a model to reach the optimal model complexity (and thus minimizing the error rate).

Let's revisit our example of predicting a person's FICO score. This prediction can be beneficial for marketing or for dynamically adjusting offers based on a person's credit score. Our background business knowledge of how FICO scores are created leads us to believe that we will need predictive variables related to a person's income, payment history, overall balance, and length of credit history. This list of variables should give us a good predictive model to try to determine a new customer's FICO score.

Step 1 – Simple Linear Regression

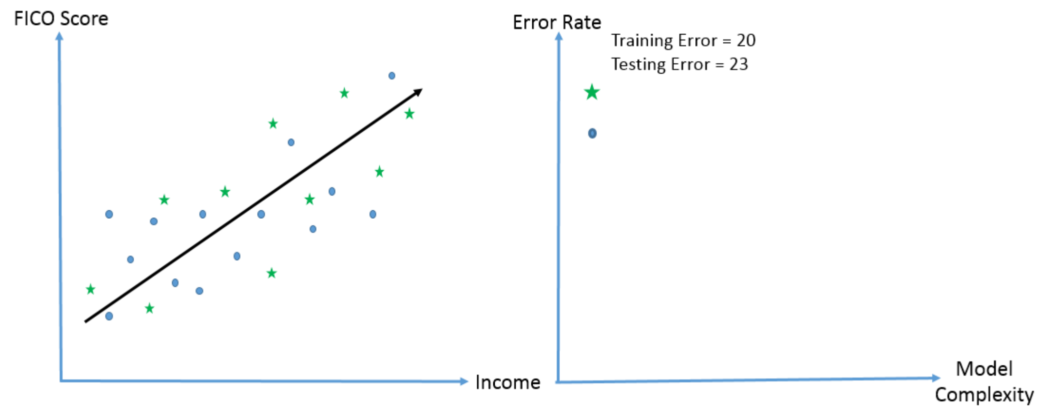
Figure 1.11 shows a simple data set that has been divided into training data (dots) and testing data (stars). The first model that was constructed is a simple linear regression. We can think of this model in the form of:

$$FICO\ Score = Intercept + Income(x) + \ \varepsilon$$

Remember that for linear regression models, we will have the target variable (FICO Score), the model intercept, the coefficient (Income), and the error term (ε).

This simple linear regression with a single predictor variable will result in a high error rate for both the training and test data sets. It captures some of the general signal in the data, but it is not dynamic to the true relationship of the underlying data.

Figure 1.11: Simple Linear Regression



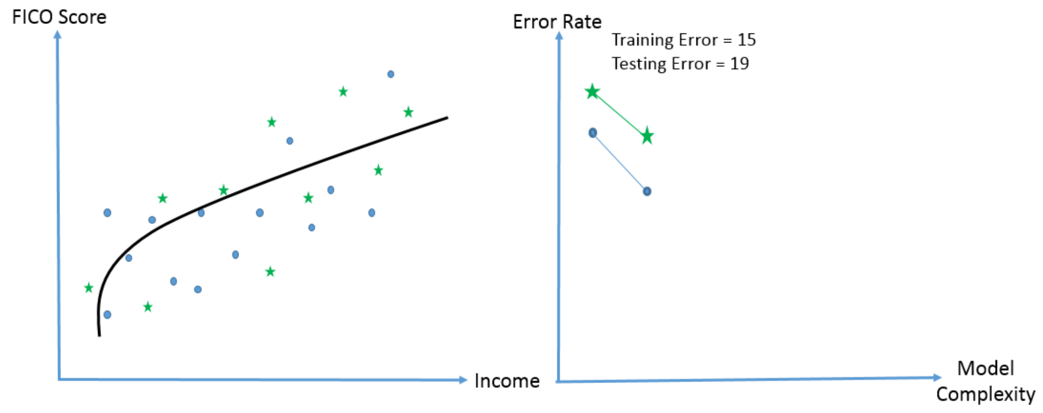
Step 2 – Linear Regression with Two Variables

Figure 1.12 demonstrates the effect of adding another predictor term to the model. In addition to the Income term, we have added a Payments term to the model:

$$FICO\ Score = Intercept + Income(x) + Payments(x) + \ \varepsilon$$

By considering a person's payment history in addition to their income, we can get a better prediction of their FICO score. The new model shows that the additional predictor term lowered both the training and the testing error rate. Since the testing error rate has decreased, we can be confident that the addition of the new predictor term benefits our model.

Figure 1.12: Linear Regression with Two Variables

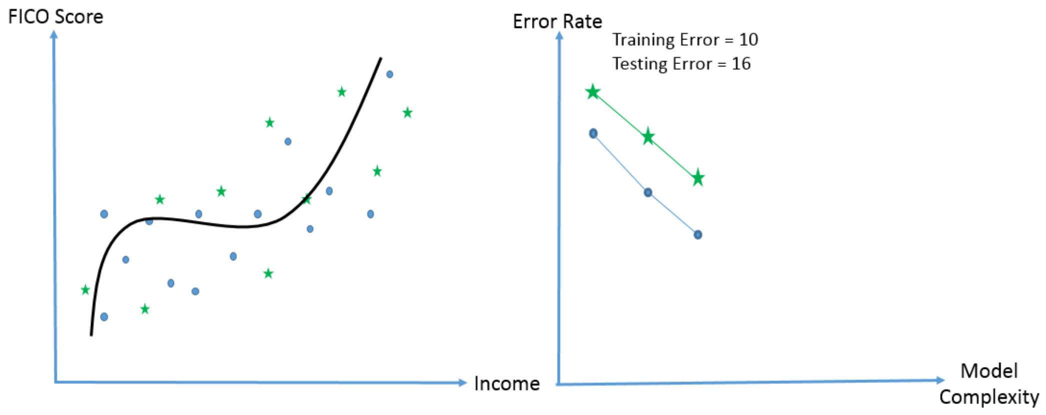


Step 3 – Linear Regression with Three Variables

Figure 1.13 shows the effect of adding a third predictor to the model. The Balance term represents a person's outstanding balance across their accounts. The updated model equation contains all three predictor terms:

$$FICO\ Score = Intercept + Income(x) + Payments(x) + Balance(x) + \ \varepsilon$$

Figure 1.13: Linear Regression with Three Variables



The addition of the Balance term allows the model to adjust and capture more of the signal in the data. This additional variable decreases the error rate in both the training and testing data sets.

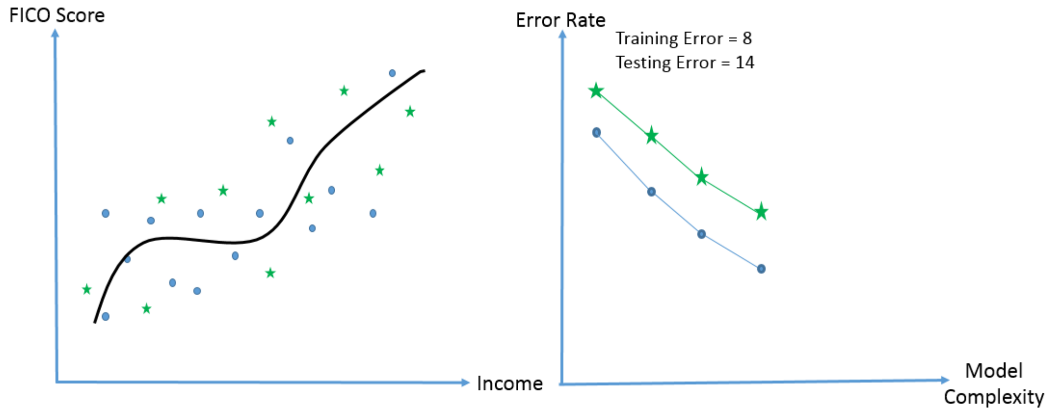
Step 4 – Linear Regression with Four Variables

Figure 1.14 shows the effect of adding a fourth predictor term to the model. The addition of the Credit Tenure variable allows the model to adjust to a person's length of credit history. The updated equation is now:

$$FICO\ Score = Intercept + Income(x) + Payments(x) + Balance(x) + Credit\ Tenure(x) + \ \varepsilon$$

The addition of the Credit Tenure variable lowers the error rate for both the training data and the testing data. We can be confident that the addition of this fourth variable is bringing us closer to optimal model complexity.

Figure 1.14: Linear Regression with Four Variables



At this point, we have exhausted all the variables that we selected based on our background business knowledge. We now have variables related to income, payment history, overall balance, and credit history. The model is dynamic to the underlying data. It captures the signal in the data and is not overfit to the noise in the data.

What happens if we continue to increase model complexity?

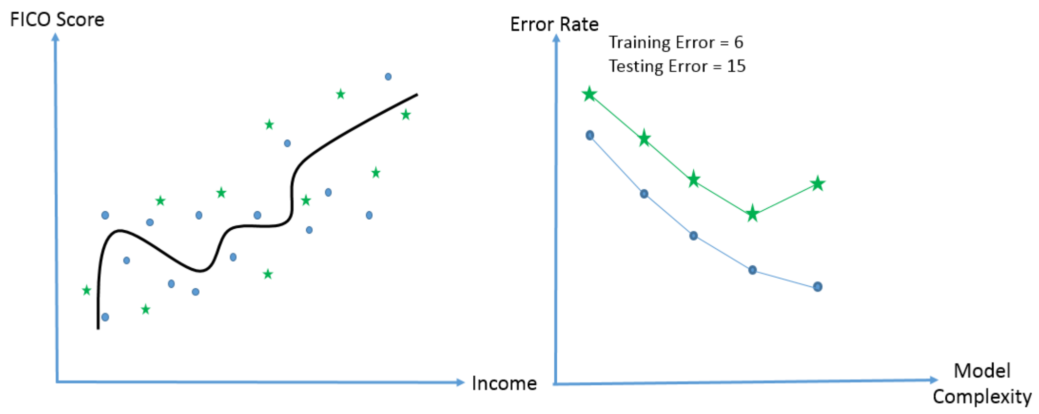
Step 5 – Linear Regression with Five Variables

Figure 1.15 demonstrates the effect of adding a polynomial term of Income to the model. The addition of the squared value of income changes the model equation to:

$$FICO\ Score = Intercept + Income(x) + Payments(x) + Balance(x) + Credit\ Tenure(x) + Income(x)^2 + \varepsilon$$

The model is becoming overly tuned to the underlying training data and is starting to not generalize well to the unseen testing data. This over tuning is demonstrated by examining the error rate of the two data sets. The training rate continues to fall because the model is becoming highly tuned to this data set. It is starting to model the noise along with the signal. However, the testing data error rate has begun to **increase**. This increase is because the model that was developed on the training data is not generalizing well to the testing data. The overly fitted model does a poor job in predicting new cases and results in a higher testing error rate.

Figure 1.15: Linear Regression with Five Variables



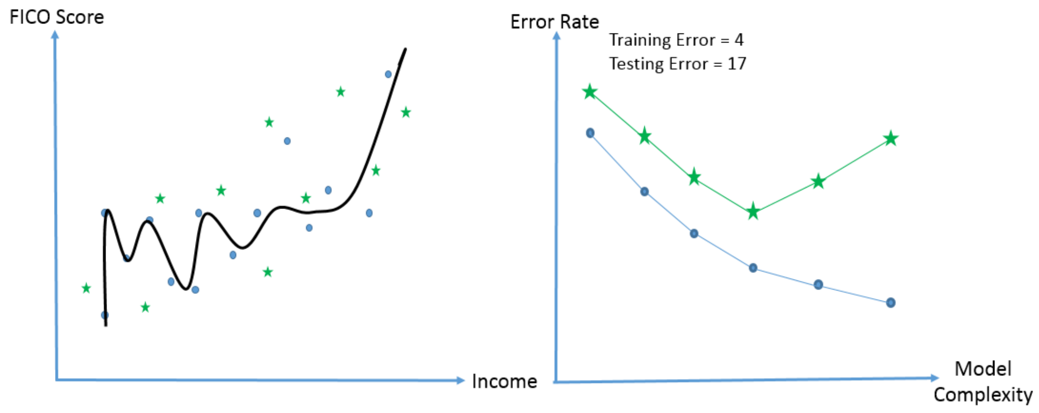
Step 6 – Linear Regression with Six Variables

In order to really drive home the point, we can add a sixth predictor term that is the polynomial of the Balance term. The addition of this term changes our equation to:

$$\begin{aligned} FICO\ Score = {} & Intercept + Income(x) + Payments(x) + Balance(x) \\ & + Credit\ Tenure(x) + Income(x)^2 + Balance(x)^2 + \varepsilon \end{aligned}$$

Figure 1.16 demonstrates how overly fit this model has become. The model is now so highly tuned to the training data that it completely misses on the testing data. The training error rate has continued to decrease while the testing error rate has continued to increase.

Figure 1.16: Linear Regression with Six Variables



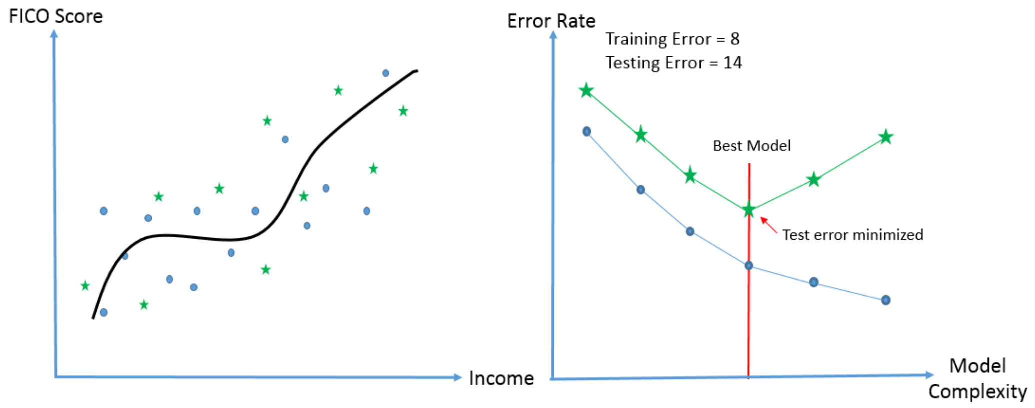
Step 7 – Optimal Linear Regression Model

This visual demonstration allows us to determine where things went wrong for our model. By evaluating the testing error rate, we can determine that the model with four predictors is the best fit. This model is where the testing error rate is minimized.

$$FICO\ Score = Intercept + Income(x) + Payments(x) + Balance(x) + Credit\ Tenure(x) + \varepsilon$$

Figure 1.17 shows the optimal model complexity for this data set. This model is able to capture the signal in the training data set and use it to accurately predict values in the testing data set.

Figure 1.17: Optimal Linear Regression Model



A lot of this book will focus on each concept of finding the optimal model complexity. This simplistic example above looked at a simple linear regression model. In actuality, data scientists need to be able to select a range of model types that would work for their underlying data. This range of model types would need to be narrowed down to one or two model types, and finally, a single model type would be chosen. Then they would need to tune the *hyperparameters* of the selected model in the same manner that we just did above. Hyperparameters are the pre-set parameters of a model that control the performance of the model. These values depend on the model type but may include learning rate, sampling rate, train fraction, alpha, and many other options.

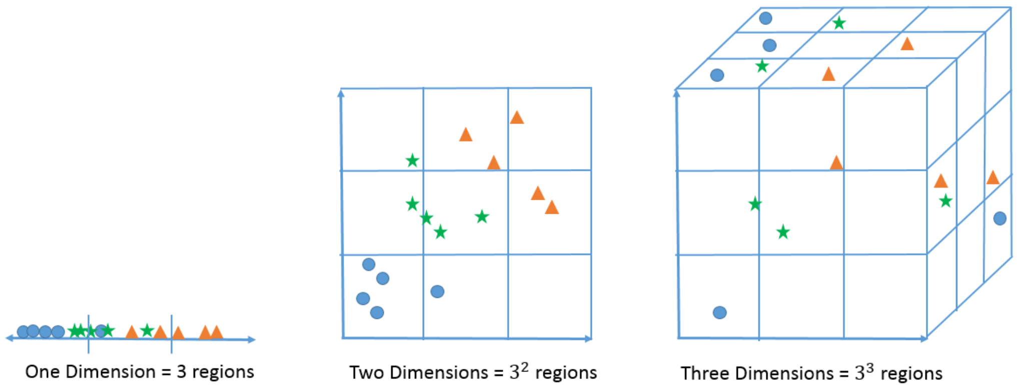
Curse of Dimensionality

The *curse of dimensionality* is problematic because it goes against everything that we've been taught about data, namely, that more data is good. It is a data scientist's job to identify a lot of good data attributes to build a predictive model. Some modeling data sets can have thousands or even millions of predictor variables. This sounds great, right? You may think that you have a nearly endless number of predictors with which to build your model. This idea starts to erode when we think about what the model is actually doing.

If we take a classification model (for example, think about a KNN or decision tree model), we are attempting to identify observations that are close to one another in a certain region of dimensional space. The model will generally count the number of observations in a given region and use those observations to construct a predictor. If a new observation falls within this region, then we can assume that it is similar to the other observations in that region and we can infer a predictive value for the new observation based on the other observations in that space.

Figure 1.18 shows what begins to happen when we increase the number of dimensions. Remember that the number of dimensions is the same as the number of predictors in your model. A model with a single predictor can be visualized by the graph on the left of Figure 1.18. This simple model has only one dimension, so it is common for there to be many variables in a single region. This model has the benefit of many predictor observations to infer values to a new observation, but since it only represents one aspect of the data (the one dimension), it will not tell us much about the new observation.

Figure 1.18: Dimension Increase


One Dimension = 3 regions
Two Dimensions = 3^2 regions
Three Dimensions = 3^3 regions

The middle graph of Figure 1.18 shows the same observations spread out over two dimensions. The additional dimension adds more information to the data and allows us to infer more information about new variables that appear in a given space; however, there are now fewer observations in a given area because the observations are being distributed across two dimensions rather than a single dimension.

The graph on the right of Figure 1.18 shows the same data spread out over three dimensions. Again, more information is gained by the additional predictor variables (dimensions), but the data is becoming even more dispersed. There are far fewer observations in each region to generate predictions.

Remember that we are distributing our limited number of observations across a feature-dependent dimensional space. The algorithm has to calculate the distance between observation points to identify groups of observations. The Euclidean distance between two Cartesian coordinates is expressed as the following formula, where p is the first coordinate and q is the second coordinate:

$$d(p,q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

(Sorry for the math, but sometimes it's necessary to explain things sufficiently). This equation shows that for each additional predictor term that we add, the distance between observation points increases. When we continue to add predictors but keep the number of observations constant, the result is a sparse feature space.

Our limited brains have a hard time imagining four-dimensional space, and we cannot really conceive of a 10,000-dimensional space. However, mathematics shows us that the number of observations needed to populate higher-dimensional space at a consistent density increases exponentially. Table 1.5 shows the exponential increase required to maintain an equivalent density in the feature space.

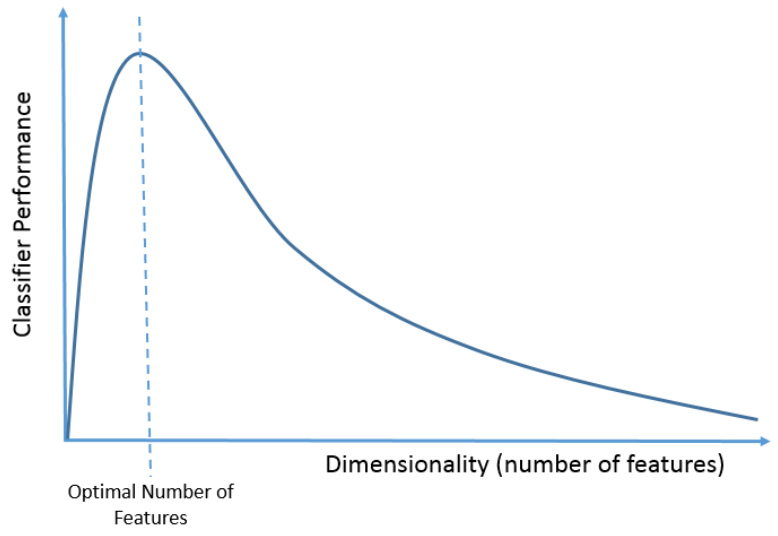
Table 1.5: Consistent Density in High-Dimensional Space

Dimension	Exponential Increase	Observations
1 Dimension	10^1	10
2 Dimensions	10^2	100
3 Dimensions	10^3	1000
n Dimensions	10^n	A whole bunch

Hughes Phenomenon

Hughes Phenomenon describes the tradeoff between the number of predictors in a model and the overall performance of the model. Figure 1.19 provides a graphical display of this concept. Hughes phenomenon simply states that as the number of predictor variables (features) increases, model performance will increase until it reaches the point of the optimal number of features for the given data set. If we add additional features beyond this optimal point while maintaining the same number of observations, the model performance degrades.

Figure 1.19: Hughes Phenomenon



Simply stated, the curse of dimensionality is this: as you add more predictors to your model, you will need to add *exponentially* more observations to maintain model performance.

Transparent Versus Black Box Models

Simple models are easy to explain. For a simple linear regression, you can easily interpret the equation of the line. Nothing is hidden in this type of model. We know exactly what the intercept, coefficient weight, and error terms are. A simple decision tree is also easy to interpret. We can easily see the logic of the decision path and understand how each split point was calculated. The methodology for these models is open for inspection and easily understood. In fact, they are so straightforward that even your manager can understand these models. (Zing! Take that management!)

However, as we increase model complexity, the transparent nature of the model begins to change. Some models, like random forests, take the basic design of a decision tree and add randomization to the construction of many decision trees and summarize their collective predictions. It’s not easy to give a solid explanation of what is happening to each variable to generate the result because that explanation changes for each of the hundreds of decision trees in a random forest.

Gradient boosting decision trees compiles the results of many decision trees by systematically taking the residual from the previous tree and using it as the target value for the next tree. In this type of model, the target value is continually changing. This loop is repeated many times over. The result is usually highly accurate, but it is not easy to explain what is happening at the predictor level.

The term “black-box model” describes a type of model that is complex and generates output without clear insight into how the prediction was created. Neural networks have famously become the poster child for black-box models. These types of models work great for image classification; however, there is no clear understanding of what particular factors contributed to the prediction. Was it the curve of the jaw or the angle of the lips or the skin tone of the person or the dress that they are wearing? Although all these factors are considered in a neural network model, you cannot assign a given quantity to a given aspect of the picture. In many cases, a very slight change in an image or a slightly different angle of the image can result in a completely different prediction.

Ethics

Black-box models are a massive concern to machine learning ethicists. As we become increasingly reliant on these types of models due to their incredible predictive accuracy, we are also moving further away from explainability and, consequently, justification of their use. Here is another example of the tradeoffs that are made with machine learning. The tradeoff of explainability and accuracy is not just a theoretical issue. It can have incredible consequences for your daily life.

Imagine that you are charged with a crime. The new AI criminal justice system calculates that you are guilty and automatically assigns you a sentence. You might want to know what reasons it gave to convict you. You might be disappointed if the answer is a cold, "the algorithm said so." That is a dystopian novel scenario with an unquestionable and unfeeling AI overlord determining your fate.

There are many examples of black-box algorithms producing results where we demand an explanation concerning its decision but are not provided one:

- Credit denial
- Identification of a cancerous cyst
- A self-driving car goes off the road
- Criminal sentencing

The ironic reality is that we rely on these black-box models due to their accuracy and automation. They are extremely fast, precise, and accurate. However, the more we integrate these algorithms into our daily lives, the more we will demand a human-level explanation of the model results. It can be a frightening thought that even the model developers who selected the inputs, model design, and the parameters cannot provide a clear explanation of how decisions are made within their own design. But on the bright side, I don't need to manually tag my friends in photos. I guess that's a win.

No Free Lunch

Data scientists have built an array of powerful models that can perform incredible tasks. Some of these model types are continually used to perform the most difficult machine learning tasks. Gradient boosting machines often win predictive modeling competitions and deep neural networks often solve complex image classification problems. You might ask yourself, so which model is the best? If deep neural networks are so great, then why don't we use them on every data problem?

The unfortunate answer is that there is no "skeleton-key model" that would work best for all possible data sets. For many data sets, we make assumptions about the distribution of the data. If a data set appears linear, then we use a linear model. If a data set appears clustered, then we can use a KNN model. However, if we relaxed all assumptions and approached a data set without any preconceived ideas about its distribution, there is no one single model that we can apply to all data sets that would outperform all other models.

This is because models need to be appropriately selected based on the underlying data distribution. Some data sets are linear by design while others are clustered. Some are sparse while some are dense. Some have stable patterns while others are erratic. Each one of these data sets has a specific model type that would work best.

The concept of no free lunch simply means that you cannot take the easy way out and use your trusty neural network model for every problem. For many data sets, a simple linear regression would work much better than a deep neural network. The only way to figure out which model to use is to evaluate the underlying data and make some assumptions about the data distribution. Then you need to select an array of model types that are aligned with that type of data distribution and apply them to the data. Finally, you must compare the results from these different model types and select which one works best for the given data to answer a given problem at a given time frame for a given population.

Chapter Review

This chapter is foundational to your understanding of data science. Nearly all the major concepts and issues of data science have been explained in detail. All these concepts will be used throughout the rest of this book and in every data science project that you will develop. It is very important that you understand these base concepts before you move forward into developing data science projects and constructing modeling algorithms.

Regardless of how well your project is designed or how sophisticated your modeling algorithm is, you will never get away from the underlying concepts explained in this chapter. Every data science project will have to address issues of the bias-variance tradeoff, underlying data distribution, optimal model complexity, the curse of dimensionality, and many other concepts covered in this chapter.