Summarizing Data and Deducing Probabilities

by Janani Ravi

Course Overview

Course Overview

Hi, my name is Janani Ravi, and welcome to this course on Summarizing Data and Deducing Probabilities. A little about myself. I have a masters degree in electrical engineering from Stanford and have worked at companies such as Microsoft, Google, and Flipkart. At Google, I was one of the first engineers working on real-time collaborative editing in Google Docs, and I hold four patents for its underlying technologies. I currently work on my own startup, Loonycorn, a studio for high-quality video content. Data science and data modeling are fast emerging as crucial capabilities that every enterprise and every technologist must possess these days. Increasingly, different organizations are using the same models and the same modeling tools. So what differs is how those models are applied to the data, so it's really important that you know your data well. In this course, you will gain the ability to summarize your data using univariate, bivariate, and multivariate statistics is a range of technologies. First, you'll learn how measures of mean and central tendency can be calculated in Microsoft Excel and Python. Next, you will discover how to use correlations and covariances to explore pairwise relationships. You will then see those constructs can be generalized to multiple variables using covariances and correlation matrices. Then you will understand and apply Bayes' Theorem, one of the most powerful and widely-used results in probability, to build a robust classifier. Finally, you will use Seaborn, a visualization

library, to represent statistics visually. When you're finished with this course, you will have the skills and knowledge to use univariate, bivariate, and multivariate descriptive statistics from Excel and Python in order to find relationships and calculate probabilities.

Understanding Descriptive Statistics for Data Analysis

Module Overview

Hi, and welcome to this course on summarizing data and deducing probabilities. In this module, we'll focus our attention on getting an intuitive understanding of the descriptive statistical measures that you can use for data analysis. We'll see as a whole that for exploratory data analysis, we can use two kinds of statistics, descriptive and inferential. Descriptive statistics are used to describe or summarize the data that we have available in our data set. Common descriptive statistics used with univariate data are measures of frequency, how often a particular value occurs in your data; measures of central tendency, what is the one number that can be used to represent this data; and finally, measures of dispersion or scatter. This measures how the numbers in your data set jump around or vary. Measures of frequency, central tendency, and dispersion typically apply to univariate data, data with just one variable. If you have a bivariate or multivariate data, that is data with two or more variables, you might want to explore the relationships that exist between these variables. That's when we'll use measures such as covariances and correlation.

Prerequisites and Course Outline

Before we dive into the actual course contents and demos, let's take a look at some of the prereqs that you need to have in order to make the most of your learning. As far as mathematics is concerned, the only prereq here is basic high school math. If you're familiar with high school mathematics, some basic statistics, you should be fine. Some of the demos of this course will be performed on Excel spreadsheets, so I'm assuming that you're familiar with Excel, you have used Excel formulas, you know how cell references and range references work. Other demos of this course will include some basic programming in the Python language, so I assume that you're familiar with Python and you know how to use Jupyter Notebooks to code, just basic Python

using a few simple data science and visualization libraries. Here is a quick outline of what the different modules of this course will cover. We'll first start with getting a basic and intuitive understanding of descriptive statistics that you might use to summarize and explore your data. And once we've got this understanding, we'll move on to exploratory data analysis using Excel workbooks. After working in Excel for a bit, we'll move on to Python programming where we'll see how we can summarize data using Python and its libraries. We'll then study Bayes' rule of conditional probabilities in some detail, and we'll see how we can apply Bayes' rule to a business problem. And finally, at the very end, we'll see how we can visualize statistical data using the Seaborn Python package.

Understanding Descriptive Statistics

In this clip, we'll talk about the role of statistics in understanding and getting an intuitive feel for the data that you're working with. So what exactly is statistics? Well, a famous man once said, there are two kinds of statistics, the kind you look up and the kind you make up. Making up statistics is fine when you want to obscure or put up a front, but that's not what we're interested in here. Let's talk about what exactly statistics is, a branch of mathematics that deals with collecting, organizing, analyzing, and interpreting data. Often, when you're working with very large and diverse data sets, it may be impossible to collect all of the data that you need to interpret. That's when you'll use samples and statistics. These are representative samples that assures us that inferences and conclusions that we draw can reasonably extend from the sample to the population as a whole. The statistics that we work with to understand our data are broadly divided into two categories, descriptive statistics and inferential statistics. A descriptive statistic is a summary statistic for data that quantitively describes or summarizes the features of a collection. Descriptive statistics are used to understand the data itself rather than using the information that we learn to learn about the population from which the data has been sampled. Inferential statistics, on the other hand, tries to use data analysis to understand the probability distribution, or the larger data, that the sample is drawn from. Both of these used together can be powerful tools to understand and explore your data. Depending on how many variables or features you're working with in your data, descriptive statistics may be categorized into univariate statistics, bivariate statistics, or multivariate statistics. Univariate statistics deal with just one variable and understanding that variable, bivariate statistics deal with relationships between pairs of variables, and multivariate statistics deal with relationship between multiple variables. When you work with inferential statistics, on the other hand, there are two kinds of data analysis that you might perform in order to understand the population from which your sample is drawn,

hypothesis testing and model fitting. Hypothesis testing involves making some kind of inference about the population as a whole and then using the sample of data that you have to test whether the hypothesis is true or false. Model fitting is where machine learning comes in. You understand relationships that exist in your data and use those relationships to make predictions on new data. Let's dig a little deeper into the world of descriptive statistics. We have univariate, bivariate, and multivariate statistics. Univariate statistics that you can calculate on your data are measures of frequency, central tendency, and dispersion. Measures of frequency simply give you an idea of how often a particular value occurs in your data set. You can think of these as simple counting measures. Measures of central tendency can be thought of as a single value that you can use to represent your data as a whole. This can be thought of as the central position or location of that set of data. And finally, measures of dispersion are also call variability, scatter, or spread. This is the extent to which a distribution is stretched or squeezed, how the numbers jump around for a variable. A measure that you might use to represent frequency for your variable are frequency tables. For every value in your data, you'll have a count of how often that particular value occurs. A better way to represent this frequency table is to use a histogram. Histograms can be used with raw data, or you might bin your data into ranges and represent using bars how often a particular category occurs in your data. Measures of central tendency you might be more familiar with. The average, or the arithmetic mean of your data set, is the most popular and widely known. This is where you sum up all of the values in your data and divide it by the count of values. If your data happens to have a lot of outliers, values that are very large or very small, the mean is not a great measure of central tendency. That's when you'll choose to use the median. The median can be described as the middle score for a set of data that has been arranged in order of magnitude. Another measure of central tendency is the mode, the value that occurs the most frequently in your data. There exist other less commonly used measures of central tendency as well, such as the geometric and harmonic mean, and each has its own specific use case. Common measures of dispersion or variation in your data are the range. This is basically just the difference between the maximum and minimum values in your data set. Another measure of dispersion is the interquartile range. We'll discuss this in more detail in a little bit. This is basically the difference between the value at the 75th percentile and 25th percentile of your data. And finally, the standard deviation, or the variance is another measure of dispersion. A low-standard deviation indicates that points are close to the mean. A high-standard deviation indicates that the points are away from the mean. Let's move on from our quick overview of univariate descriptive statistics to bivariate descriptive statistics. The most important here are correlation and covariance. Both of these are closely linked. Between these two, let's first discuss what covariance measures. This is a measure of the relationship between two variables. Specifically, it checks to see whether greater values of

one variable correspond to greater values in the other variable. If the answer here is yes, the covariance between these two variables is said to be positive. If greater values in one variable corresponds to smaller values in the other variable, the covariance between them is negative. Correlation can be thought of as a normalized version of the covariance. It's similar to covariance. It measures whether greater values of one variable correspond to greater values in the other, but the correlation coefficient between a pair of variables is scaled to always lie between +1 and -1. Another way to think of this is that the correlation is a measure of whether a linear relationship exists between two variables. Plus one indicates positive linear relationship, -1 a negative linear relationship, and variables that are independent exhibit 0 correlation. Covariances and correlations can be extended to multivariate data as well. Here, you'll represent the relationship between every pair of variables that exist in your data set using either a correlation matrix or a covariance matrix. Every row and every column of this matrix represents a variable in your data.

Measures of Central Tendency and Dispersion

In this clip, we'll understand intuitively the most common measures that we might use to describe univariate data, mean, variance, and standard deviation. Now let's say you had a number of data points in one dimension and you were asked for your thoughtful fact-based point of view on all of these numbers. If you wanted to use just one number or a headline to represent all of these data points, you'd use the arithmetic mean of your data. The mean or average is the one number that best represents all of these data points. You sum up all of the data points and divide by the number of data points that you have. That gives you the mean. On the other hand, the mean alone doesn't really give you much information about how the data is spread around. Do the numbers jump around? The mean can't tell you. That's when the variation in the data is important to capture. One way to capture the variation is to specify the range of your data. The maximum value minus the minimum value. One drawback when you use the range to capture the variation in your data is that this range is swayed by outliers. If you have one point that is very large in value and another point that's very small, that'll skew the representation of the variance in your data. In order to not be swayed by outliers that might be present in your data, you'll use the variance to represent variation. Variance is the second most important number to summarize the set of data points. So how do we calculate the variance? Let's say these are all of our data points, X1 all the way through to Xn, and X bar is the mean of this data. The mean deviation of your data is Xi-X bar. Subtract the mean from every data point, and that gives you the mean deviation. Once you have the mean deviation, you can simply square it to get a measure known as the squared mean deviation, Xi-X bar the whole square. And once you have the squared mean

deviation, the formula to capture the variance is very straightforward. Some of the squared mean deviation for all of the points that exist in your data set and divide by the number of points. That gives you the variance. When you're calculating variance in the real world, this formula is often tweaked to improve our estimate of the variance. We tweak the denominator by subtracting 1, and this correction is called Bessel's correction. We've seen here how we can calculate the mean and variance of a particular set of data. Mean and variance are often enough to succinctly summarize a set of numbers. Here is the arithmetic mean, and here is the variance as calculated using the mean. Rather than use the variance directly, another measure that you might have heard of is the standard deviation. The standard deviation is nothing but the square root of variance. Here is how we would calculate the variance in our data. The square root of this is just the standard deviation, so they are interchangeable. When you're working with data in the real world, you might find that the values that you have may not be exactly what you expect. You might find that certain points are outliers. Outliers might represent data errors, you haven't collected the data properly, or they might be genuinely rare points, which are legitimately present in your data set. It turns out that the most common measure of central tendency, that is the arithmetic mean of your data, is very, very sensitive to the presence of outliers in your data, which is why you might want to resort to using other measures of central tendency. We've also seen that when you use range to describe variance in your data, that's also sensitive to outliers. Another measure of variance in your data is the interquartile range. Think of the third quartile, or Q3, as the 75th percentile. This basically describes that data point such that 75% of the points in your data set is smaller than this value. Q1, on the other hand, represents the 25th percentile. Twenty-five percent of the points in your data set are smaller than Q1. Once you have these measures, the interquartile range is nothing but Q3 minus Q1, the 75th percentile minus the 25th percentile. The interquartile range gives you measure of where the middle 50% of your data is distributed. A measure of central tendency that is less that susceptible to the presence of outliers in your data set is the median. The median is basically the value at the 50th percentile. Fifty percent of the data points are smaller than the median, fifty percent are higher than the median. If you feel that your data contains outliers that are erroneous, you might want to use the median as a measure of central tendency. Unlike mean, median changes little due to outliers.

Understanding Variance

In this clip, we'll try to get an intuitive understanding of the variance of a data set and how it moves. Now let's imagine that you're tossing two coins. The first of these coins we'll refer to as Coin X. If Coin X shows up as heads, you will get $1, and if it shows up as tails, you lose $1. These

are small stakes. Loser pays $1. Winner takes $1. Let's say we also toss another coin. That is Coin Y. Here, if it shows up as heads, you'll get $1000, but for tails, you lose $1000. These are high stakes. Loser pays $1000. Winner takes $1000. Two coins are simple to understand and visualize. I'm going to set up a little table with all possible outcomes for Coin X and Coin Y and the payoffs associated with each of these outcomes. Of course, the implicit assumption here is that the coin is a fair one, and this is how it'll tend to show up. The first two columns are the results of the coin toss, Coin X and Coin Y, and the remaining two columns are the payoffs associated with Coin X and Y. Because each coin is a fair one, there is a 50% likelihood of it showing up as head or tails. So if you calculate the mean of the payoff associated with Coin X, we'll get a value of 0. This is intuitive and understandable. No matter what the stakes are, if you calculate the average payoff for Coin Y, we'll get a value of 0 once again. Simply sum up all of the payoffs, divide by the number of payoffs, that's 0. The mean was relatively straightforward. Let's go back to the payoff associated with Coin X and calculate the value. We've already studied what the formula of the variance is, and if you calculate the variance of Coin X, you'll find that it's equal to 1. The payoff in every case is either $1 or -$1, subtract the mean, square it, and then divide by the number of outcomes. This will give you 1. Now let's apply the same formula for variance to calculate the variance of the payoff associated with Coin Y, and here, you'll find that things are not really the same. The variance associated with the payoff for Coin Y is $1 million. And this is where variance is interesting. Observe that the mean for both payoffs are exactly the same. The variance is very different. As stakes grow, variance gets bigger faster than the mean. Let's go back to the original two coin tosses, X and Y. When we played for small stakes, the variance was small. But as the stakes grew 1000x, the variance grew by a million.

Measuring Relationships Using Covariance

Now that we understand variance, let's move on to understanding covariance and correlation. These are important statistics that we use to describe bivariate and multivariate data. Univariate data is data in one dimension, and we've already discussed a number of descriptive statistics that we could use to explore and understand univariate data such as mean, median, and standard deviation. When you work with data in the real world, it's often more insightful to view data in relation to some other related data. So you have two or more variables, and you explore the relationships between them. If you have bivariate data with X and Y variables, they can be plotted on a coordinate plane in two dimensions. We briefly discussed earlier the use of covariances as a measure of the relationship between two variables. It specifically measures whether greater values of one variable correspond to greater values in the other. Let's get an intuitive

understanding of covariance. For that, we'll need to use the mathematical formula for covariance. Here, we have data in two dimensions. There are X and Y variables, and X bar and Y bar represents the mean of X and the mean of Y respectively. For every point in this data set represented by Xi or Yi, the covariance of X, Y is given by this formula that you see here on screen. The covariance between two variables is basically the variance in two dimensions. You subtract the mean from X and Y, perform a summation across all data points in your data set, and divide by the number of points. For one specific point represented by Xi, Yi, the numerator of this formula represents the area of the rectangle that you see highlighted here on screen. The covariance between two variables, X and Y, involves summing up the areas of all rectangles made with the mean. A summation of all the rectangle areas will give you the numerator that you see in this formula, and you divide the summation by the number of data points. That's the denominator. Let's intuitively understand what it means for two variables to have positive covariance. Here are two variables, X and Y. X bar and the Y bar represent the mean of these two variables. The bars that you see here for X and Y are how the values of X and Y deviate about the mean. When two variables have positive covariance, you'll find that the deviations about the mean for these variables of the two series are in sync. If you then consider the distribution of two variables that have negative covariance, here are how the variables are distributed about the mean, you'll find that the deviations of the two variables about the mean are out of sync. We've already discussed earlier and got an intuitive understand of the variance of a variable. Let's say this variable is X and both of these figures here represent the deviations of the value of X about the mean. Covariance and variance are closely related. The variance of a variable is simply the covariance of a series with itself. And as you might imagine, the variance of a series with itself is always positive.

Covariance Matrices and Correlation Matrices

Covariance and correlations can be calculated for bivariate data, that is data with two variables. These can be extended to multivariate data as well. So if you have multiple variables in your data set, let's say X1 all the way through Xn, a covariance matrix summarizes the covariances of all columns in a data matrix. The rules are the variables; the columns are the variables; and individual cells or elements in a covariance matrix represents the covariance between pairs of variables. There are key variables here, X1 through Xk; there are k rows and k columns in this matrix; and every element of the covariance matrix contains the covariance of a pair of vectors. The first row of a covariance matrix corresponds to the variable X1. This contains the covariance of the first column, X1, with each of the other columns including itself. This is true for all of the other rows as well. The last row contains the covariance of the last column, Xk, that is the last variable, with

each of the other variables including itself. When you see a covariance matrix describing the pairwise relationships that exist between each pair of variables, you'll see the matrix is symmetric in nature. The value at row i and column j is the same as that at row j and column i. Along the diagonal of the covariance matrix is basically the covariance of every variable with itself, which as we know is just the variances of the corresponding columns. So a better way to represent this covariance matrix is that along the main diagonal, we have the variances of each of the individual variables, X1 through Kx. When we speak of covariances, we had also discussed the correlation between pairs of variables. Let's move on to understanding correlation. We know that it's similar to covariance, it measures whether greater values of one variable correspond to greater values in another, but this is a normalized representation of covariance. It's scaled to be between +1 and -1. Consider two variables represented in a two-dimensional plane that are positively correlated. E and D along the Y and X-axis. Here is how a distribution of their data points might look. The correlation coefficient between two variables tries to capture linear relationships that exist between those two variables. Two variables are perfectly positively correlated. They have a correlation of +1. If as X increases, Y increases linearly. Two variables are set to be perfectly negatively correlated, we have a correlation of -1. If as X increases, Y decreases linearly. And finally, if two variables are completely independent such that changes in the value of one variable are independent or do not affect changes in Y, the correlation coefficient is set to be 0. I should briefly explain that little asterisk that you see there. Now if the correlation coefficient between two variables are equal to 0, it's possible there is an edge case that the variable may not necessarily be independent. We've spoken of the correlation as a normalized representation of the covariance, and here is the mathematical formula that relates the correlation between X and Y to the covariance. Generally, in the real world, you may never use this mathematical formula directly, but it's important you understand how they are linked. And finally, one last detail to remember about covariance and correlation, independent variables have zero covariance and zero correlation. And with this, we come to the very end of this module where we discuss the importance and user statistics to explore and understand our data. We saw that statistics is divided into two broad categories, descriptive and inferential, and we focused our attention on using descriptive statistics to understand our data. In descriptive statistics, we discussed measures of frequency; measures of central tendency such as mean, mode, and median; measures of dispersion such as range, variance, and the interquartile range; measures of frequency, central tendency, and dispersion; apply to univariate data, data which has just one variable. If you're discussing bivariate or multivariate data, the measures that you'll use to explore relationships will be covariance and correlation. In the next module, we'll see how you can

calculate these descriptive statistics and perform exploratory data analysis using Excel spreadsheets.

Performing Exploratory Data Analysis in Spreadsheets

Module Overview

Hi, and welcome to this module on performing exploratory data analysis in spreadsheets. In this module, we'll work with Microsoft Excel workbooks to explore and understand our data; we'll use Excel to perform univariate, bivariate, and multivariate analysis; we'll calculate descriptive statistics for univariate data, plot and visualize bivariate relationships, as well as perform regression analysis, and we'll do the same for multivariate data as well; we'll also use Excel functions or formulas that are available to calculate measures of central tendency and measures of dispersion. Excel offers built-in functions to calculate the mean, median, and mode, as well as dispersion statistics, such as standard deviation and variance. In addition to calculating descriptive statistics, we'll also visualize relationships using charts. We'll use box plots, line charts, and pie charts in order to understand our data. We'll also see how we can perform regression analysis in Excel on bivariate, as well as multivariate data.

Working with Excel Workbooks

In order to continue our univariate analysis of height data, I've created a new sheet within this Excel workbook called Visualization. I've also copied in the single row of Height data that we worked on earlier. This is so that we have the Height data on the same sheet where we performed the analysis. Select the first column, column A, which contains your Height data, and let's visualize this information using a few charts. Go to your menu bar and click on Insert here and insert a new chart. The box and Whisker chart is a great tool for visualizing univariate data to convey important statistical information, and that's what I've selected here. Excel automatically displays a box and whisker plot for the data that you have selected. I'm going to make this bigger so that we can see exactly what's going on. Excel has some prepopulated access limits. You can see that the vertical axis here goes from 0 to 250. We don't need this entire range, so you can select this axis and change the Minimum and Maximum bounds for your box plot. I'll set 130 centimeters to be

the Minimum Y-bound and 210 to be the Maximum Y-bound so I can view the box plot more clearly. I'm going to close this format axis by clicking on the X on the top right, and I'm now going to edit the title of this chart. I'm going to call it Box Plot. The Box Plot is a single visualization, which conveys a number of different bits of information. The center line that you see here on screen is the median of your Height data. If you remember earlier, we calculated the median to be about 170.5, and that's where the center line is. The height of the box represents the interquartile range. The top edge of the box is the 75th percentile, and the bottom edge is the 25th percentile. And finally, the two horizontal lines that you see in the jutting out portion are called whiskers, that's why this is called the box and whisker plot, and they represent the range of your data. This is the default representation in Excel. Box plots can also be configured so that the center line represents the average and the whiskers represent 1.5 times the interquartile range or other statistics like this. Just with the single box plot visualization, you get across a number of different bits of information about your data. Let's take a look at another chart here. Select the A column, which contains your data, and select the Insert options, go to Chart, and this time, we'll plot a line chart. This line chart can give you a visual indicator of how all of the heights of the individuals in your data set fluctuates. Right now, our line chart is overlapping with our box plot. I'm simply going to select the line chart and drag it to the place on the sheet where I want it to be and expand it to make it bigger. you can see that the height of individuals in our data set varies fairly evenly within the range. There are no outliers.

Descriptive Statistics for Univariate Data

If you have a single variable, there are several descriptive statistics that you can use to get a feel for your data. The first is what is the maximum number present in your single variable? Let's apply an Excel formula to calculate this. I'm now going to move my cursor to the cell G2 where I'm going to write a formula to calculate the maximum height. This is the MAX function, and I simply type =MAX, and then I select the range of values that Excel should look at to calculate the maximum value. So I select the first cell with height in it. Then I scroll downward with my Shift key pressed down and select the last Cell at row 501. Take a look at the structure of this formula, =MAX, A2:A501, and these are the cells that we have selected. Even if the values in these cells change, this formula will calculate the maximum height. In our case, the Max Height is 199. Now that you know how to type in Excel functions and select a range of cell values on which the function should apply, let's go on and paste in some other functions in order to calculate other descriptive statistics in your Excel book. Just like the MAX function, Excel offers a MIN function to calculate the minimum height in your single variable. MIN of A2:A501 is 140 cm. The next useful

descriptive statistic is the range of height values present in our data. Now range is maximum minus minimum. We've already calculated the maximum and minimum values. I'm going to simply type in the range formula as G2-G3 where G2 contains the maximum height in my data and G3 contains the minimum height in my data. If you hit Enter, you'll get the result of this formula. That is 59 cm. Now that we have the range of this data set, let's calculate some measures of central tendency, starting with mode. There is a built-in MODE function that you can use to calculate the mode in Excel, and the mode of our data is 188. That is the most frequently appearing height. All measures of central tendency of univariate data are available as simple formulas. We have a formula for average as well. Simply type in =AVERAGE and specify the range of cells over which you want the average calculated. The average height for the 500 individuals we have in our data is about 170 cm. Let's calculate the median as well using the MEDIAN function. MEDIAN of A2: A501 is 170.5. You can see that the Mean and Median here are very close together. Along with range, another important measure of how your data set is dispersed or spread out across the mean is the variance of your data set. This is available in Excel using the VAR function. Variance, if you remember, is the square of the standard deviation. Once you hit Enter, you can see that the variance of height data is about 268. Observe that when you have a cell with the formula selected, the result is displayed in the cell and the formula is displayed in the formula bar on the top. Just like the variance, you can also calculate the standard deviation of your univariate data using a simple function. Standard Deviation of A2:A501 is about 16.37. Another important descriptive statistic that you might be interested in for the univariate data are quartiles. In Excel, you can calculate the 25th, 50th, and 75th percentiles using the QUARTILE function. In order to calculate the 25th percentile, you specify the range of data and then specify option 1 as your second argument to this Excel function. Option 1 represents 25th percentile; 2 is the Median, that is the 25th percentile; 3 represents the 75th percentile; 0 and 4 represent the minimum and maximum values in your data set. The result shows you that 25% of the individuals in our data set are under 156 cm. Let's calculate the 75th percentile in exactly the same way. Make sure that you specify the second input argument to this function after the range as 3 for the 75th percentile. Seventy-five percent of the individuals in our data set are under 184 cm. Now if you want to calculate the interquartile range, this is very simple. Simply subtract the 25th percentile from the 75th percentile to get the interquartile range of 28. Another statistical measure that can be used to define a univariate distribution is kurtosis. Kurtosis measures extreme values in the tail of the distribution. Kurtosis of a distribution is measured in relative terms. What is the kurtosis relative to the normal distribution. In our case, the Height data has a negative value for kurtosis, which implies that this particular distribution has lighter tails as compared to the normal distribution. Another measure to describe how univariate data is distributed is the skewness of a distribution.

This measures the asymmetry in a statistical distribution. Distributions can be skewed either to the left or to the right. They can be asymmetrical in both ways. The SKEW function in Excel calculates the skewness of your data, and negative values generally means that the distribution has its tail extended to the left, slightly in this case because it's just a small negative value. You've now seen that it's possible to use individual Excel formulas to calculate all of these descriptive statistics manually. In fact, you can do this in a far easier way in Excel if you go to Tools and Data Analysis. This Excel plugin that we enabled has the Descriptive Statistics analysis tool. This tool will work on your input data and generate all of the descriptive statistics that are useful automatically. Simply specify the input range of data on which you want this tool to operate. That is a2 to a501. You also need to specify where you want the descriptive statistics displayed. Select the Output Range box and then select any cell within your Excel sheet, and this is where the output will be displayed. For us, it's J2. Now if you hit OK, all of the summary statistics, this is the checkbox that you have enabled, will be displayed starting at J2. All of the statistics that we calculated manually using Excel formulas and a few more are all displayed here. You can give this a meaningful title. All of the descriptive statistics generated in one go using the Data Analysis plugin.

Visualizing Univariate Statistics

In order to continue our univariate analysis of Height data, I've created a new sheet within this Excel workbook called Visualization. I've also copied in the single row of Height data that we worked on earlier. This is so that we have the Height data on the same sheet where we performed the analysis. Select the first column, column A, which contains your Height data, and let's visualize this information using a few charts. Go to your menu bar and click on Insert here and insert a new chart. The box and whisker chart is a great tool for visualizing univariate data to convey important statistical information, and that's what I've selected here. Excel automatically displays a box and whisker plot for the data that you have selected. I'm going to make this bigger so that we can see exactly what's going on. Excel has some prepopulated axis limits. You can see that the vertical axis here goes from 0 to 250. We don't need this entire range, so you can select this axis and change the minimum and maximum bounds for your box plot. I'll set 130 cm to be the Minimum Y-bound and 210 to be the Maximum Y-bound so I can view the box plot more clearly. I'm going to close this Format Axis by clicking on the X on the top-right, and I'm now going to edit the title of this chart. I'm going to call it Box Plot. The Box Plot is a single visualization, which can raise a number of different bits of information. The center line that you see here on screen is the median of your Height data. If you remember earlier, we calculated the median to be about 170.5, and

that's where the center line is. The height of the box represents the interquartile range. The top edge of the box is the 75th percentile, and the bottom edge is the 25th percentile. And finally, the two horizontal lines that you see in the jutting out portion are called whiskers, that's why this is called the box and whisker plot, and they represent the range of your data. This is the default representation in Excel. Box plots can also be configured so that the center line represents the average and the whiskers represent 1.5 times the interquartile range or other statistics like this. Just with a single box plot visualization, you get across a number of different bits of information about your data. Let's take a look at another chart here. Select the A column, which contains your data, and select the Insert option, go to Chart, and this time, we'll plot a line chart. This line chart can give you a visual indicator of how all of the heights of the individuals in your data set fluctuates. Right now, our line chart is overlapping with our box plot. I'm simply going to select the line chart and drag it to the place on the sheet where I want it to be and expand it to make it bigger. You can see that the height of individuals in our data set varies fairly evenly within the range. There are no outliers.

Using Pivot Tables for Summary Statistics

We'll continue with univariate analysis of our data, and this time, we'll use pivot tables. I have a new sheet in my Excel workbook, and I have the Height data already copied into the first column. Pivot tables allow you to group values by a particular field and display summary statistics for that grouping. If you select the Insert option on the Excel menu, this will bring up a ribbon where you see the first option is that of a Pivot Table. Pivot tables are important tools for data analysis. When you select Pivot Table, it will bring up a dialog where you can specify the range of data that you want to use for your table. If you don't know what a pivot table is, don't worry. This demo will make that very clear in just a bit. Observe that we have a new menu option, Pivot Table Analyze. That's where we are located. And on to the right, we have the option to select PivotTable Fields. We have just one field in our data. That is the Height data, and this is what we choose for our pivot table. Notice that Excel automatically calculates the sum of all of the Height data and displays it with the new pivot table. I want this height to represent the rules of my pivot table, so I'm going to select Height from the PivotTable Fields and drag it down to the Rows section. Observe that our pivot table on this sheet is automatically updated. We have Row Labels, and for every row, we have the Sum of Height. So we've pivoted on the height itself. So for the row labeled 140, we have the Sum of Height for all individuals who have the height 140. Now this isn't really meaningful, which is why I'm going to change the summation value. Instead of Sum of Height, I'll change this by clicking on the little icon next to Sum of Height. This brings up a dialog,

which allows me to change the summarization for our pivot table to Count. So for every Height row label, we want to display the count of height, or the frequency, or the number of people at that particular height, and that's what we get in our resulting pivot table. Our data contains eight individuals who are 140 cm, eight at 141, nine at 142, and so on. If you scroll down to the bottom here, you'll see that the pivot table automatically calculates a grand total of 500. That is the size of our data set. So we pivot on the row label, that is the Height data, and we have the frequency of number of individuals at this height. If you want additional summary statistics, you can simply select the Height field name from your PivotTable Fields and drag this into the Values section. The default summary statistic, as you see, is the Sum for the particular row label. That's not meaningful, so I'm going to use the edit dialog that I bring up by clicking on this icon and change the summary statistic. Instead of summarizing, I want to display the data in a specific way. I click on Show data as, and I'll use this drop-down to display this data as a % of Grand Total. This way, I'll not only have a count of the number of people at each height, but I'll also know what percent of the total data they represent. And here is the resulting column in our pivot table. You can see that the number of individuals at different heights are pretty evenly distributed. And if you scroll down to the bottom, you can see that the grand total is 100%. If you really think about data, you'll see that the pivot table that we've created here isn't really that useful. Really, is there a difference between someone whose height is 140 cm versus 141? Not really. What we really want to do is to bucket this data and then create pivots with summary statistics, which we'll then visualize using charts. That's what we'll do in the next clip.

Performing Analysis Using Bucketing and Pivot Charts

Here we are on a new sheet of the Univariate_Analysis Excel workbook. This is called the Bucketing Data sheet, and we have the Height data already copied in. Head over to the Insert option in our Excel menu. Once again, we are going to insert a pivot table, but the way we pivot on our Height data will be a little different. Select the pivot table option and specify the range of data, that is a1 to a501, and the pivot table will be inserted into your sheet. We have just one field in our data, that is the Height data. Select Height within FIELD NAME and drag it onto the Rows section. So we are pivoting on individual heights. Before we move on to bucketing, let's change the summary statistic. Bring up the Summarize dialog and summarize by Count, so we want the Count of individuals for every height. Our pivot table now displays more meaningful information. For every value of Height, we have a count of individuals. Place your cursor on a specific row label. I've selected the very first row in my pivot table. And this selection within your pivot table will bring up some options within the pivot table Analyze menu. I'm going to select the Group

Selection option, which will allow me to bucket this data. Select Group Selection in this drop-down that you see here and specify the kind of bucket that you want. We want our entire range of Height data to be represented in the pivot table, so we leave the starting and ending values as 140 and 199, and we want to group by 12. This means that every bucket in our resulting pivot will contain 12 values, and that's what you see here in the result. The first bucket is 140 to 151, the second bucket is 152 to 163, and so on. And for each of these buckets in our pivot table, we have a frequency count of the number of individuals whose height falls within that bucket. In order to visualize this data, use your mouse and the Shift button to select your pivot table values. head over to the Insert option, and this will bring up an option to insert a PivotChart, and this PivotChart will automatically display a frequency histogram for the buckets that you have in your pivot table. I'm going to position this pivot chart so it doesn't overlap with the pivot table and make it bigger, and you can see that in our data set, most of the individuals have their heights between 176 and 187 cm. There are several ways in which you can customize this chart. I want to make the font bigger. I head over to the Home option, and I use this font drop-down to increase the size of the font within my chart to 16 pixels. Our histogram looks much nicer now. Let's now visualize the same data in a slightly different way. I'm going to use the Shift and the mouse to select all of the values in my pivot table, head over to the Insert menu, and once again insert a PivotChart for this data. Now the default representation of your pivot chart is the histogram that we've seen before, but I'm going to right-click on this chart and change the chart type to be something different. I want this to be a pie chart. There are some different pie chart options that you can choose from. I'll choose to go with the simple 2D pie representing proportions. Let's reposition and expand this pie chart so it doesn't overlap with the histogram that we had created earlier. The sections of this pie represent the proportion of individuals falling into each height bucket. Let's display some more information on this pie. I right-click and Add Data Labels, and the frequency associated with each section of the pie is now displayed. If you want these frequency counts to be displayed in a bigger font, right-click, choose the Font option, and change the font size. I'm going to make it 16, and you'll find that the resulting pie looks a lot better. For each chart that you have selected in your Excel sheet, the Format pane appears to the right. I'm going to use this pane to update the position of my labels to be at the center of the pie. The legend here is in really tiny font as well. I'm going to right-click, choose Font, and increase this font to be 16 as well. Now that our chart looks a lot better, I'm going to head over to the PivotChart Fields tab in our formatting pane off to the right. Here are all of the fields, just the height field has been displayed within this pivot chart. I'm going to change this so that it no longer displays the Count of Height as the summary statistic, but instead, displays something else. Right-click, go to Field Settings, and this brings up the same dialog that we've encountered earlier. I'm going to select the Show

data as tab and use this drop-down to display data in the form of percentages. Choose the Percentage option, change the field name, click on OK, and the sections of your pie are now percentage values rather than absolute counts. Now that we've seen how to set up pivot tables with buckets and visualize this information, let's go and see some manual Excel formulas that we can use to count the number of values in a particular bucket. Let's say you want to know how many individuals are within the height range 140 to 163 cm. There is a simple Excel formula for this, the COUNTIFS. Within a cell, specify COUNTIFS, and within brackets, specify the range of your data, A2:A501. Within your original data, you want to count those values if they satisfy the condition that you've specified, great than, equal to 140 and less than, equal to 163. Observe how the COUNTIFS function allows you to specify multiple ranges, here we are operating on the same range, and multiple conditions that you can apply on those ranges. You get the result, which satisfies both of those conditions. There are 184 individuals within this range.

Visualizing Bivariate Relationships

In this demo, we'll use Excel workbooks to perform bivariate analysis, analysis using two variables. We'll use visualizations, as well as regression analysis to understand the relationships between these two variables. Here we are on a brand-new Excel workbook. I'm on a sheet called Visualizing Relationships because visualization is what we want to do first and the data that we are going to work with contains two variables, Head_Size and Brain_Weight. The first thing that I'm interested in knowing is the correlation between these two variables. In statistics, correlation refers to any association between these two variables. Most commonly, correlations refer to linear relationships that might exist between these variables. Understanding the correlation between variables is useful because they can be used to indicate a predictive relationship where one variable, such as the Head_Size, can be used to predict another, the Brain_Weight. Excel makes it very easy for you to calculate correlations and data using the CORREL function. The CORREL function takes in two ranges. These are the ranges of the values for which you want to find the correlation. We have A2 to A238, which contains the Head_Size information, and B2 to B238, which contains the Brain_Weight information. And the CORREL function tells us that the correlation between these two is 0.79. Correlation values close to 0 tell us that the variables are independent. This is positive correlation. This tells us that as head size increases, brain weight also tends to increase. Let's visualize this bivariate relationship using a scatter plot. I'm going to select the columns for Head_Size and Brain_Weight. And using the menu, I'll choose the Insert option here and go ahead and insert an X Y Scatter chart. Here is the resulting scatter plot. I'm going to resize this and make a few formatting edits before we analyze the data that it displays. I'm going

to click on the x-axis here so that we change the bounds to zoom into the data. The minimum bound I'm going to set to 2500. Okay, my data is a little zoomed in. And I'm going to click on the y-axis and change the bounds there as well. I've set the minimum to be 800. We now have a much better scatter plot visualization. This upward-sloping scatter plot makes it very clear that people with larger heads tend to have heavier brains. Observe that I have no data here about whether heavier brains indicate higher intelligence or lower intelligence. You need to be careful that you don't draw the wrong conclusions from this data. As you can see here, if you have two variables and you want to explore the kind of relationships that might exist between these two, the scatter plot visualization is extremely useful.

Performing Regression Analysis on Bivariate Data

Let's say you have some data that you're exploring, and you have a prior or you suspect that there exist a cause-effect relationship between two variables in your data. One way to support your findings is to perform regression analysis, and we'll see how we can do that in Excel. Here I am in a brand-new Excel sheet within my Bivariate_Analysis notebook, and I'm continuing to work with the Head_Size and Brain_Weight data. Select the data columns in your Excel sheet and head over to the Tools menu. In the Tools menu, you have the option for Data Analysis, thanks to the plugin that we added in early on. Within this Analysis Tools dialog, you'll see that there is an option for Regression analysis. Click on OK, and this will bring up a dialog, which will help you specify on what you want this regression analysis performed. You're trying to figure out whether a cause-effect relationship exists between head size and brain weight. the Input Y Range refers to the effect. You have reason to believe that brain weight depends on the head size. B1 to b238 is my Input Y Range, and my Input X Range is a1:a238. I want the output of my regression analysis to be displayed starting at cell D2. I want the output on the same worksheet. You can also choose a new worksheet to display your output. Within Residuals, I want a Line Fit Plot, so I want to see how the regression line fits on the original data, and I also want probability plots, probability distribution plots for my data. I click on OK, and Excel will automatically perform the regression analysis for me. The SUMMARY OUTPUT gives you a bunch of statistical information about your data and the regression analysis. You can see how strong the relationship is between your variables. An important score that you can use to evaluate whether this regression was a strong one, whether the relationship is strong, is the R Squared and the Adjusted R Squared. The R Squared score is a measure that tries to capture how much of the variance in the underlying data, that is the Head_Size data, is captured by the regression that you just performed. The Adjusted R Squared is a measure that is more useful when you have multiple terms in your regression. Here,

we have just the one term, Head_Size. The Adjusted R Squared is the R Squared number that has been adjusted to account for the number of predictors that you've used in your regression model. Let's take a look at some of the other values that we have here. Here, you can see the coefficients of our regression, the coefficient associated with Head_Size and the Intercept. This regression analysis has tried to fit a straight line through our Head_Size data, so we have a formula of the form Y is equal to NX+C. That represents linear regression, and Intercept here is the C value, and the Head_Size coefficient is the value of N. Every coefficient is associated with a t Stat and a P-value. We don't need to go into the details of these, but these two values together specify how significant the Intercept and the Head_Size variables are. The coefficients are significant ff we have a positive value for the t Statistic and a very small value for the P statistic, or the P-value. P-values should be less than 5%. If you scroll over to the right, you'll see that our regression analysis has also generated two charts, the Head_Size Line Fit Plot and a Normal Probability Plot. I'm going to position these plots one on top of the other on the screen and increase their size. Here is the original scatter plot, and you can see that the orange dots represent the regression line. The regression analysis performed linear regression on our data. And if you scroll down here, you can see the Normal Probability Plot, which gives you how the Brain_Weight data is distributed. You have percentiles on the X-axis and the actual Brain_Weight on the Y-axis.

Covariance and Correlation Matrices for Multivariate Data

In this demo, we'll work with data that has many features, or X variables, to perform multivariate analysis. Here I am on a brand-new Excel workbook called Multivariate_Analysis, and I have a new worksheet named Relationships. I've already loaded in some data into this worksheet. This is the graduate student admissions data set. This data set is freely available on Kaggle at this URL. You can see that it contains a bunch of different information for graduate students, GRE Score; the TOEFL Score, that this the test of English as a foreign language; the rating of the university to which they've applied; a rating for their statement of purpose, letter of recommendation, and so on. And finally, the last column is the Chance of Admit. This gives a probability score to the student whether he or she will be admitted to the university to which he or she has applied. Now a lot of the X variables, such as the GRE Score, TOEFL Score, the CGPA of a student, might be related. Let's take a look at the correlation matrix for this data. Select the cell where the matrix will be displayed, head over to the Tools menu, and under Tools, you'll find the option for Data Analysis. This is the plugin that we enabled. In your Data Analysis tools, at the very top, you have the Correlation analysis tool, which is what we'll select. The Input Range is the input range of our data, a1 to h501. This includes all of the rows and columns in our data set. We specify to Excel that

there are labels in the first row of this data, so those should be ignored, and we specify the Output Range where this analysis should be displayed. You can select the Output Range text box and then select a cell in your worksheet. And you can see that Excel very helpfully throws up a correlation matrix, which shows you how every pair of variables is related to one another. Along this main diagonal, we have the correlation values of a variable with itself. That is always perfectly positively correlated equal to 1. A student has to write the graduate record examination in order to apply for grad school, and foreign students also have a TOEFL exam, Test of English as a Foreign Language, and you can see that these are highly positively correlated. You can also see that the probability that a student will be admitted to a university of his choice is highly positively correlated with the GRE score, as well as the CGPA of the student. Another way to understand the relationships that exist between the variables in your data is to use the Covariance Matrix. In statistics, the covariance is a measure of the joint variability of two random variables. If greater values of one variable mainly correspond with greater values of the other variable and the same is true for lower values of the first variable, the covariance between these two variables is said to be positive. Similarly, you can figure out what negative covariance means. Lesser values of one variable correspond to higher values of the second. Under Tools and Data Analysis in your Excel workbook, you can calculate the Covariance Matrix for your data set. This brings up a dialog similar to when we set up the Correlation Matrix. We have labels in the first row of our data set, the Input Range is a1 to h501, and we want the Output Range for the Covariance Matrix to be in this workbook. Go ahead and select the cell where you want this matrix to be displayed. Excel will automatically calculate the covariance between every pair of variables and display it on your worksheet. Now the magnitude of the covariance is not easy to interpret because it's not normalized, which is why we typically tend to use the correlation coefficient with just the normalized form of the covariance. You know that the CGPA and the TOEFL Score have a positive linear relationship with the GRE score; however, you can't really tell the magnitude of this positive linear relationship using just the covariance. The Correlation Matrix is much better for this.

Visualizing Multivariate Data Using Pivot Charts

Let's continue our analysis of the graduate student data set on a new sheet. This sheet is called Visualization, and I've already copied the data in here. We'll visualize this data using the web charts. Select a cell, go to the Insert option, and within Insert, select the PivotChart option. Observe that we haven't set up a pivot table first. We're directly using the pivot chart. This will bring up a dialog where you can specify the range of your data set, a1 to h501. When you click on OK, you'll see that Excel brings up a pivot chart, as well as a corresponding pivot table, which

holds the data associated with that chart. Let's configure this data that we want to view using this pivot chart. I'm going to select the University Rating field and drag it onto the Axis Categories. Observe that Excel has automatically updated the pivot table, as well as the pivot chart. The Row Labels are the full ratings for the universities, 1, 2, 3, 4, 5; and the default summary statistic is the Sum of University Rating, which doesn't really make sense, and that's what we'll set out to fix. Bring up the summary statistic Edit dialog. We are summarizing by Sum. Switch over to summarize by Count and hit OK. Now you can see how many students applied to the different universities with different ratings. Both the pivot table, as well as a bar graph are updated to reflect this information. If you don't want a bar chart for this information, you can right-click and change the chart type. There are a number of options for you to choose from. I'll go with a Line chart. And within the line chart, I want an area chart, so that's what I'm going to select here, and you can see the same information now displayed using an area chart. Let's display some more information in this area chart. I'm going to select the CGPA field. The default summarization is the Sum of CGPA, which is not interesting. Bring up the Edit dialog for the summarization, and let's switch this over to Average. The area chart, as well as the pivot table has been updated to reflect the average CGPA of students applying to the differently graded universities. You can see at the center for medium ranked university with a rating of 3, this has the highest number of applicants, and you can see that the CGPA is higher for higher rated universities. Let's add in more detail onto this area chart. I've selected the GRE Score field from our data set. And instead of Sum of GRE Score, I'm going to bring up the Edit dialog and calculate the average GRE score of students who've applied to the differently rated universities. Let's do the same for the Test of English as a Foreign Language, the TOEFL Score, as well. Add this field in. Excel has added this to the Axis categories by default. I'm going to drag it over to the summarization statistics. And instead the Sum of TOEFL Score, I'll bring up the Edit dialog and calculate the average TOEFL Score. We now have four bits of information represented using this pivot chart and pivot table. Close the properties panel, and let's take a look at the chart and the pivot table all at once. The pivot table shows you that for the highest rated universities with a rating of 5, the CGPA is really high. The average CGPA is 9.27 out of 10, the average GRE score is 327, and the average TOEFL scores for students are also very high.

Regression Analysis with Multivariate Data

Here we are on a new sheet in our Multivariate_Analysis Excel workbook. This sheet is called Regression Analysis, and we'll work the graduate student admissions data set. The technique to perform regression analysis on multivariate data is the same as with bivariate data. Use Tools,

Data Analysis, and within the Analysis Tools, choose Regression. We suspect that the probability of the student's admission, which is specified in the column H, is determined by the factor specified in the columns a through g. The first row of our data has column labels, so we check the labels box. We then specify the Output Range where this regression analysis should be displayed. In addition to the summary statistics of regression, we'll display the Line Fit Plots for our regression. Click on OK, and Excel will automatically compute the regression analysis for you. I'm going to move away all of the plots that are overlapping with my summary output so that we can take a look at the summary statistics for our regression. We have multiple predictors in our regression model, the GRE score, TOEFL Score, CGPA whether the student has done research or not, all is used to predict the probability of admit, or the chance of admit, for a student. Clearly, these are good indicators because the R Scared and the Adjusted R Squared of this regression model is really high, over 80%. You can see the weightage, or the coefficients for each of the input features in our regression. You can also see how significant these are looking at the t Stat and P-value. The GRE score, TOEFL Score, and the letter of recommendation rating for the student are clearly significant in determining his chances of admit to a school. But the statement of purpose has a low t Statistic, but more importantly, high P-value, over 5%, which means this is not really that significant. If you scroll over to the side, that's where I've placed my scatter plots, you can see how the chance of admit varies with the GRE score, the TOEFL Score, and the other input variables. You can see a linear relationship here. Even if your input data is categorical, has discrete value such as university rating or a rating for your statement of purpose, you can see that the chance of admit is linearly correlated. The probability of admit seems to rise with higher ratings. Let's take a look at the letter of recommendation and the CGPA. The same thing is true here. And let's take a look at the research as well. If you've done research, the probability of admission to a university of your choice seems to be higher.

Module Summary

And with this, we come to the very end of this module where we performed exploratory data analysis using Microsoft Excel spreadsheets. We saw that Excel is very simple and intuitive to use and offers powerful built-in functionality to work with univariate, bivariate, and multivariate statistics. We used Excel formulas, or functions as they are called, to calculate measures of central tendency and dispersion in our data. Excel offers built-in formulas to calculate the mean, median, and mode of your data set, as well as measures of dispersion such as variance and standard deviation. Excel also has a data analysis addon, which you can use to quickly compute all of the descriptive statistics on your data in one go. In addition to calculating descriptive statistics, we

also saw how we could use Excel to visualize relationships using the charts provided. Excel has box plots, line plots, pivot tables, pie charts, and a variety of other ways that you can use to dig into granular details of your data. We also performed regression analysis in Excel on both bivariate, as well as multivariate data. In the next module, we'll move on from Excel and use Python to explore and understand our data. We'll see how we could summarize our data and deduce probabilities using Python and its libraries.

Summarizing Data and Deducing Probabilities Using Python

Module Overview

Hi, and welcome to this module on summarizing data and deducing probabilities using Python. In this module, we'll get introduced to Azure Notebooks. These are hosted Jupyter notebooks on the Microsoft Azure platform, which we can use to write Python code in an interactive shell. Azure Notebooks come with all of the basic data science and visualization libraries installed, and we can use them out of the box. We'll use Python to perform univariate, bivariate, and multivariate analysis of our data. We can choose to use the native Python for the calculation of descriptive statistics, but that can get pretty tedious. There are, however, a whole host of Python libraries available to us, which make it very easy for us to explore and analyze our data. We'll specifically use the NumPy, Pandas, and StatsModels libraries for statistical analysis. We'll use Azure notebooks to write all of the Python code in this module. Azure Notebooks are basically notebooks hosted on the Azure cloud, which allow users to write Python code using Jupyter notebooks and execute it on a Microsoft Azure-hosted VM instance. Azure Notebooks are a great way to prototype your code. It's completely browser based and cloud hosted. Azure Notebooks offer an interactive shell, not only for Python versions 2 and 3, but also for R and F shop. if you're working on Microsoft Azure, it's useful to use Azure Notebooks for prototyping because it offers integration with a host of other Azure services.

Getting Started with Azure Notebooks

In all of the demos for this module, we'll work on the Azure cloud platform and write Python code and execute it using Azure Notebooks. Azure Notebooks are basically Jupyter notebooks posted on the Azure cloud, and they can be accessed at notebooks.azure .com. You'll need to create and work with an Azure account to use Azure Notebooks. I'm going to sign into my Azure account here using the Sign-in button on the top right. Cloud.user @ loonycorn.com is my login. This is a personal Azure account, not connected to an organization, so I select Personal account, specify my password, and then go ahead and sign in. Notebooks.azure .com directly takes us to Azure Notebooks. I'm going to create a brand-new project, which will host my notebooks. Click on create one now and specify the name and other project details. I'm simply going to call it Data_analysis, and my Project ID is also data-analysis. That is autogenerated. This is going to be a private project that is only for me to use. You can see from the status here that Azure Notebooks are running on Free Compute. That means you don't have to pay for these resources. This, of course, implies that you can't perform heavy-duty operations on Azure Notebooks, but it works great for our prototyping. Click on the plus icon here. This will allow you to create notebooks and folders. I'm going to create a new folder where I'm going to store my code. I'll call this folder code, click through. The code folder will hold my Python notebooks on the Azure cloud. I'm going to create a new subfolder, which will contain the data that I'll work on. I'll call this the dataset folder. Click through, and let's now upload some data sets from our local machine. Click on the upload button here, and I'm going to upload from my computer. This will bring up a dialog that will allow you to choose files from your local machine. Click on Choose files, thus loading up a finder window, and here are all of the CSV files that I'm going to upload to the Azure Notebook project. Click on the Upload button here and wait for all of the CSV files to be uploaded. This will be pretty quick. Once uploaded, click on Done, and you're now ready to get started.

Calculating Descriptive Statistics Using Python

In this demo, we'll see how we can calculate descriptive statistics on data using simple Python code. Here we are on notebooks.azure .com, our project name is Data_analysis, and we are within the code subfolder. I'm going to use this plus button drop-down here in order to create a new Azure Notebook. Within this dialog, specify a name for your notebook. I'm going to call it DescriptiveStatisticsUsingPython, and I'm going to be writing code using Python 3.6. You can see that Azure has automatically added the ipynb extension to your notebook. Click on New, and let's click through and start writing some Python code. You can see here that Azure Notebooks are simply Jupyter notebooks hosted on the Azure cloud, and they come with all of the libraries that you'll need built in. We'll be using the Pandas library to read in and work with data, so import

pandas as pd. Just like with local Jupyter notebooks, in order to execute this code cell, you need to hit Shift+Enter on your machine. The Pandas version that I'm working with is 0.23 .4. If you feel that the version of Pandas that you have in your project is an older version, simply use! pip install -U pandas to get the latest version. The data set that we'll work with is the head_size and brain_weight data set, which I'm going to read in into a Pandas data frame using pd.read_csv. The head function will display the top few records in this data. Here are 10 records, Head_Size and the corresponding Brain_Weight in grams. The shape property of the data frame tells us that there are a total of 237 rows or records, and every record has two columns, one for Head_Size and one for Brain_Weight. Let's say you want to calculate the average head size in this data set. You'll first need to perform the sum operation on the Head_Size column to get the total sum of head_size across the entire data set. We'll then need to calculate the number of records that you have, which is 237. With the total sum and the number of records, you can calculate the average, which is a measure of central tendencies. Sum_head_size divided by num_head_size will give us the mean value. The average head size in cubic centimeters is almost 3634. You can perform the same average calculation for the brain_weight as well. Sum up all of the brain_weights, divide by the number of records, and the average brain_weight in our data set is 1282 grams. We know that we are working with a total of 237 records, which is stored in the num_head_size variable. Let's take a look at the type of the brain_data column. We can see that it's a Pandas Series object. The built-in sorted function in Python can work with Pandas Series objects as well. Pass in the head_size data to sorted and get the sorted_head_size, so all of the head_size values sorted from the lowest to the highest. This is the first step that we need to perform in order to calculate the median value of head_size. The median of any data set requires us to first sort a data set and find the middle_index, which we'll do by calculating num_head_size plus 1 divided by two. We know that we have an odd number of elements, which means the median is the center element present in the middle_index index. If we print out middle_index, it's 119. The median value for head_size can now be retrieved by indexing into the sorted_head_size list, and the median value is 3615. You can see that a combination of using built-in functions in Python and manual calculation using code allows us to calculate practically every descriptive statistic that we might be interested in. As the stats get a little more complex, calculating them manually becomes more tedious. Here, we'll write some code to calculate the mode of our data set. Initialize the size_counts dictionary, which will keep track of the number of occurrences for each value of Brain_Weight. We run a for loop through the Brain_Weight column. And if a particular Brain_Weight is not present in size_counts, we initialize it to 1 otherwise we increment the existing value by 1. Once this code is executed the size_counts dictionary will have a frequency value associated with every Brain_Weight, as you see here on screen, and we can now iterate through this to find the mode of

the Brain_Weight data. Initialize count and size to 0 then run a for loop through the size_count dictionary items, and if the count is less than the current count that we encounter, that is c, we'll store that in the count variable and the corresponding size in the size variable. Finally, print out the size and frequency count, and you can see that the mode of the Brain_Weight data is 1350. There are 8 individuals in our data set with a Brain_Weight of 1350 grams. Let's see an example of a measure of dispersion, min and max values, and range. Min and max values can be calculated using the built-in min and max functions in Python. Here are the min and max values for Head_Size in cubic centimeters, and we can simply subtract the min from the max in order to get the range of Head_Size. Now if you're actually writing code in Python, you won't be hand calculating any of these descriptive statistics or other measures. The Python data science stack has a number of useful libraries that are available for you to use out of the box. You don't have to write code for these common functions, and that's what we'll see in the next clip.

Calculating Descriptive Statistics Using Python Libraries

In this demo, we'll see how you can calculate descriptive statistics for your data using some popular Python libraries for data science and statistics, NumPy, Pandas, and StatsModels. Here we are on a new Azure Notebook called DescriptiveStatisticsUsingNumpyPandasAndStatsmodels. First, set up the import statements for all of the libraries that you need, numpy, pandas, matplotlib to visualize your data, statsmodel, and the stats library from scipy. NumPy is an extremely popular Python package to work with and manipulate arrays. The version that we're using is 1.16 .2. Matplotlib is a package built on top of NumPy for visualizing your data. We're working with the latest version of matplotlib at the time of this recording, version 3. The StatsModels library in Python contains functions and libraries for useful statistical functions and techniques. We are working with version 0.9 .0. And SciPy is a library that you use for scientific and technical computing in Python. We are working with version 1.1. All of these are free and open source, and anybody can use them. We'll calculate descriptive statistics on our univariate data. That is the Height data. Here is a sample of 10 records from that data set. We've read in the CSV data into a Pandas data frame. Let's take a look at this data. There are 500 records with just 1 column in every record. That is the Height data. Let's take a look at some central tendency measures. The np.mean function from the NumPy library allows us to calculate the average of this data set. The average height is 169.9. Similarly, the np.median function in the NumPy library gives us the median of our data, which is 170.5. NumPy has library functions to calculate the mean and the median, but not for mode. Np.mode gives you an error. There is no attribute mode in your NumPy module. Since you have your data already in a Pandas data frame, you don't need to use NumPy. You can

use Pandas directly. In the height_data data frame, I've accessed the height column and called the .mean function, and this gives me the mean of the Height data. All numeric column support these common descriptive statistics. You can calculate the median in exactly the same way as well, and in fact, you can call the .mode function and calculate the mode of your Height data as well. So if you're working with Pandas data frames, you don't need to use NumPy at all. You can directly use the Pandas functions to calculate these measures of central tendency. The StatsModel library offers a module called DescrStatsW, which calculates commonly used descriptive statistics. Import this module and instantiate a DescrStats object by passing in our Height data, and the .mean property will give you the average of this data set. If you want to calculate the mode of your data set using NumPy, you'll use the SciPy library. From scipy, import the stats module and call stats.mode, and it'll show you that 188 with the count of 15 is the mode of your Height data. From measures of central tendency, let's move on to calculating measures of dispersion. You can use NumPy to calculate the minimum and maximum values in your data set. Here, I've used the NumPy functions np.min and np.max to calculate the minimum and maximum values of my Height data. Let's print out these values, and you can see that they are 140 and 199. You could use NumPy to calculate the range of your data set using the ptp function. Ptp stands for peak to peak. This will give you the difference between the minimum and maximum values of Height. It's 59, and you can confirm this by taking a look at min_height and max_height and subtracting one from the other. You can see that the results of both are the same. NumPy has very a very useful function called np.percentile to calculate percentile values in your data set. You can calculate the 25th percentile by passing in 25 as your second input argument. If the percentile that you've specified lies between two data points, you can specify the kind of interpolation you want to use to choose the right value. Interpolation= lower will choose the lower of the two data points. The 25th percentile of our Height data is 156. Instead of using NumPy, you can calculate these percentile values directly in Pandas. Invoke the quantile function and pass in 0.25 for the 25th percentile, and you'll get the result 156 as you would expect. We can also calculate percentiles using the StatsModels descriptive statistics object. Invoke the quantile function on this object and pass in the probability. Zero point two five will give you the 25th percentile. Let's go back to using NumPy. And using the np.percentile method, let's calculate the 75th percentile of our data set. Interpolation is lower once again. The 75th percentile is 184. And of course, you can use the quantile function on your Pandas series object and calculate the 75th percentile, and you can use the quantile function on the Height descriptor object and calculate the 75th percentile as well. If you want to calculate the interquartile range, that is the difference between the 75th and 25th percentile, you can use the stats module from the SciPy library. Stats.iqr will give you the interquartile range for your Height data. We have the 75th percentile stored in the variable q3 and

the 25th percentile in q1. Q3 minus q1 also gives us 28. That is our interquartile range. We discussed earlier that manually calculating measures of dispersion, such as variance and standard deviation, is pretty painful. But using the NumPy library, you simply invoke np.var on your Height data to get the variance in heights. Other variance is also directly available in your descriptive statistics stats model object. Simply call height_descr.var to get the variance. NumPy also has a useful function to calculate the standard deviation, np.std. This is the standard deviation of our Heights data. Similarly, height_descr.std will give you the standard deviation. This is the StatsModels object. You can always confirm that the standard deviation is the square root of the variance. Call np.sqrt on height_descr.var.

Calculating Skewness Kurtosis and Simple Visualizations

The SciPy library also has functions allowing you to calculate the skew in your data. Simply call stats.skew and pass in the Height data, and here is the skewness value. The same value that we got using Excel. The probably distribution of our Height data has a slight skew towards the left. Now there are different techniques to calculate the skewness in your data. When you use the stattools module from statsmodels, you can invoke the robust skewness function and get skew values using four different techniques. The first is the standard skewness, which is the result that we got earlier using Excel and the SciPy library. There is also quartile-based skewness, mean, median, different skewness, and so on. Exactly how those techniques work are beyond the scope of this course. You should know that calculating all of these different types are skewness are possible using StatsModels. The SciPy library also offers a helper function to calculate the kurtosis of your distribution called stats.kurtosis, and this is the same kurtosis result we got using Excel. Let's now use StatsModels. Stattools.robust_kurtosis will give you four different kurtosis measures for your data. The first is the standard kurtosis, the same result that we got using SciPy. There is also kurtosis estimator based on octiles, exceedance expectations, and other statistical techniques. If you want a summary of descriptive statistics on your data, you can use the SciPy library called stats.describe, and you'll get the number of observations, mean, min, max, the variance standard deviation, skewness, kurtosis, all useful statistics in one go. Or your data is already loaded into Pandas. You can simply call describe on your Pandas data frame, and this will print out the basic measures of central tendency and dispersion for your data, mean, standard deviation, min, max, and the quartiles. If you want to visualize this data, there are several ways to do it, and the easiest is by calling the boxplot method on your Pandas data frame. This will display a box and whiskers plot that we first studied when we worked with Excel. Pandas integrates with the matplotlib library under the hood to display this visualization. You can plot a line plot using

your data as well. Simply call height_data.plot .line, and here is a line plot representation of the Height data. Pandas is great to plot things quickly. But if you want to customize your visualization, you should use matplotlib directly. Here we are plotting a bar graph by sorting the Height first. This bar graph will give us a frequency count for every value of Height. Here is what the resulting bar graph looks like, but what's more interesting is a histogram representation. Let's plot that as well. Call height_data plot.hist, and this will plot a histogram representation of the heights in your data set. As before, you can see that most of the individuals in our data set have heights between 185 and 190 cm.

Bivariate Analysis

In this demo, we'll see how we can perform bivariate analysis, that is analysis using two variables in Python. Set up the import statements for the Python libraries that we've used, pandas to read in and work with the data set, seaborn and matplotlib to visualize our data. Seaborn is a statistical visualization tool built on top of matplotlib. I'm going to use Pandas to read in the admissions predict dataset that we are familiar with into a Pandas data frame, admissions_data. Here is what the data looks like. We have the GRE score, the TOEFL Score, University Rating, and a bunch of other bits of information about a student, and the probability that he or she got into a college of their choice. The best way to visualize a relationship between a pair of variables is to use the scatter plot I'll use to plot a scatter plot of the GRE score of a student versus his or her chances of admit. We have the GRE score on the X-axis and the chance of admit on the Y-axis. If you take a look at the upward-sloping nature of the scatter plot, you can see that a linear relationship exists between the two. Higher the GRE score, higher the probability the student was admitted. Let's use a scatter plot to analyze another relationship, the TOEFL score versus the probability of admit. Once again, this upward-sloping nature of the scatter plot shows you that a linear relationship exists between the two. What about the GPA of a student versus his chances of being admitted to a university of his choice. Once again, very stark relationship. Higher the GPA, higher the chances of admit. Let's now use the scatter plot to view a relationship where one of the values is categorical in nature. University Rating comprises of discrete values 1 through 5. That is the categorical feature. There resulting visualization shows us how GRE scores are spread out for the differently rated universities. You can see that when a university is more highly rated, say 5, the GRE scores of the students who applied to that university also tend to be higher. We saw how we could view relationships using scatter plots. Let's take a look at the correlation between the different pairs of variables that exist in our data set. You can calculate the correlation matrix in a Pandas data frame by simply invoking the .corr function. Here is the resulting correlation matrix

for our admissions data. You can see that several of our features are highly correlated with the chances of admit. The GRE score, TOEFL Score, whether the student has done research or not are all positively correlated with the probability of admit. The correlation matrix here also tells us that students with high GPAs also tend to have high GRE and TOEFL scores. They are positively correlated. Viewing correlation data in the form of numbers is not really very intuitive, which is why there is this very useful visualization called the heat map available in Seaborn. Invoke the sns.heatmap function and pass in your correlation matrix, and this will generate a visualization in the form of a grid where the colors of the grid cell represent whether a pair of variables are positively correlated or negatively correlated, and the different shades represents the strength of the correlation. You can see that the CGPA and the GRE score are strongly correlated as is the probability of admit and the GRE score.

Simple Regression on Bivariate Data Using Scipy

In this demo, we'll see how we can perform regression analysis in Python on bivariate data using the StatsModels from the SciPy library. Now there are many ways to perform regression on your data. There are analytical methods and machine learning techniques. Analytical methods involve the use of a formula to calculate regression values whereas with machine learning techniques, there is no underlying formula, only the training of your model. Here, we'll use the SciPy library, specifically the StatsModels in the SpiPy library, to perform analytical regression on our bivariate data. The bivariate data that we'll work with is the head_size and brain_weight data set. Read the CSV file in in the form of a Pandas data frame. Here are a few samples of the records in our data frame. Here, we are going to explicitly perform linear regression. Linear regression involves fitting a straight line on our underlying data in order to use that straight line for prediction later on. We believe that there is a linear relationship between head_size and brain_weight. We call stats.linregress, pass in the head_size, as well as the brain_weight to this function to perform linear regression. Linregress will perform regression using analytical techniques by fitting a formula on your data, and it'll return the slope, the intercept, and the R-squared score for your regression. It returns other values as well, such as the P-statistic to indicate the significance of this relationship and the standard error of our linear regression. We'll ignore those for now. Let's print out the R-square value of our regression, and you can see that it's 0.63. Remember that the R-squared is a measure of how much of the variance in the underlying data has been captured by our straight line. Higher values are better, that means more variance is captures. Very high values can often be suspect. Here is the slope of the line that we just fit on our data, 0.26, and here is the value of the intercept. These are the same values that we got when we used Excel if you

remember. Let's now use matpoplotlib to visualize the line that this linear regression model produced and see how it fits on the underlying data. I'll first have a scatter plot, which plots the original data values, head_size versus brain_weight. I'll then use only the x variable, that is the head_size data and use the slope and intercept generated from my linear regression model to get the predicted y values for brain_weight. This I'll label the fitted line of our linear regression model. And if you take a look at the resulting visualization, you can see the original data in the form of a scatter plot, and you can see that the fitted line models the underlying data pretty well.

Regression on Multivariate Data Using Statsmodels and scikit-learn

In this demo, we'll perform regression analysis once again, but this time, with multivariate data. We'll also use some different Python libraries. We'll perform analytical regression using StatsModels and regression with machine learning techniques using the scikit-learn library. Here we are in a new Azure Notebook, let's go ahead and install the latest version of scikit-learn, which we'll use to fit a linear regression model. This demo uses scikit-learn version 0.20 .3, the latest version at the time of this recording. We'll import all of the usual data science libraries, such as Pandas and NumPy, to work with our data. In addition, we'll import the StatsModels API, which will perform analytical regression by fitting a formula, and we'll import linear regression from the scikit-learn library to perform linear regression using machine learning techniques. We'll work with the graduate admissions data set, reading the CSV file into the admission_data data frame. The machine learning model that we'll build using the scikit-learn library, we'll use for prediction. So I'm going to go ahead and split my data into X variables and Y values. the X variables are all column values, other than the chance of admit, and what we are going to try to predict is the Y variable, the probability of admission for a particular student. I'm now going to split this data set into training data and test data. Training data is what we use to train our machine learning model. Test data is data that our machine learning model hasn't seen before. That is what we'll use to evaluate or measure our machine learning model. We'll use 80% of the data that we have for training and 20% to evaluate our model. That is our test data. The test-train split is a very useful function available in the scikit-learn library to split your data into training set and the test set. Now let's go ahead and see the shape of the training data. We'll use 400 records to train our machine learning model and 100 records to evaluate it. The y data contains the chances of admit for each student. This is what we'll try and predict once we've trained our model. But before we move on to machine learning, let's first perform regression analysis using the StatsModels library. This uses analytical techniques to perform regression by fitting a formula on the underlying data. We'll first add a constant to the training data. This is the intercept for our regression formula.

We'll then instantiate the OLS object that performs ordinary least-squares regression. We'll pass in the x_train_with_intercept, and we'll regress on y_train. We invoke the regression process by calling stats_model.fit. Once we have the fitted model, we can take a look at the summary statistics. Call fit_model.summary, and it'll print out statistics very similar to what we saw in Excel. Observe that the R-squared and the Adjusted R-squared here are very high, over 80%, indicating that this is a good regression analysis. The F-statistic and the Probability score for the F-statistic also shows that this model is significant. A model is significant if its F-statistic is a positive value, which it is here, and the P-value should be under 5%, which once again, it is. Let's move on and build a regression model on this data using machine learning technique. Linear regression using ML does not fit a formula, but instead will try to tweak the model parameters to fit a straight line on your data. First thing you need to do is to instantiate a linear regression estimator object in scikit-learn. This object is what we'll use to train our model, and we'll then use the train model for prediction. Normalize=True is simply a way of tweaking the numeric features in your data so that it's more robust when fed into an ML model. Normalization scales all of the numeric values in your data to be between 0 and 1. The fit method here is what starts the training process of your machine learning model. We'll pass in x_train to train the model and y_train. These are the output values that the model needs to tweak the model parameters. Once this code executes, we have a fully-trained model. We can then use it for prediction by calling linear_model.predict. I'm now going to predict the y values on my training data. I pass in x_train and store the result in y_pred_train. Let's calculate the R-squared value of our model on the training data. The scikit-learn library has a very helpful function, r2_score, and you invoke r2_score on the predicted values versus the actual values to get the R-squared score for this regression model on the training data. The R-squared score is about 76%, which is pretty decent. Let's now perform prediction using this model on the test data. Remember, this is data that our machine learning model hasn't seen before. We haven't used this data to train our model. The predicted values from our model are stored in y_pred_test. Let's calculate the r2_score for the predicted values of the test data, and you can see that it's 83%. The fact that the R-squared score is higher on the test data as opposed to the training data shows us that this linear regression model is actually a very good one.

Module Summary

With this, we come to the very end of this module where we used the Python programming language to explore and analyze our data. We wrote all of our Python code using Python 3 on Azure Notebooks on the Microsoft Azure platform. Azure Notebooks are simply Jupyter

notebooks hosted on an Azure VM instance, and it comes with a variety of data science and visualization tools built in. We used Python to perform univariate, bivariate, and multivariate analysis. We saw that we could use native Python to calculate measures of central tendency, as well as measures of dispersion, but that was tedious. We saw that the Python programming language comes with a rich set of libraries for data analysis, and you can use those libraries for descriptive statistics and to explore relationships. We specifically used the NumPy, Pandas, and StatsModels libraries for statistical analysis. In the next module, we'll move on to studying Bayes' rule and its application in business problems. We'll see how Bayes' rule involves the use of conditional probabilities for data analysis.

Understanding and Applying Bayes' Rule

Module Overview

Hi, and welcome to this module on understanding and applying Bayes' Rule. We'll use Bayes' rule to solve a problem from the real world. We'll first start off by understanding the intuition that lies behind Bayes' Rule. Bayes' Rule is a mathematical model based on statistics and probability that aims to calculate the probability of one scenario based on its relationship with another scenario. Once we've understood the intuition behind Bayes' Rule, we'll move on to the mathematical formulation for calculating conditional probabilities. We'll see how we can use the formula for Bayes' Rule to calculate the sentiment for review text. Bayes' Rule in the real world is often used for data analysis, such as predicational classification problem. We'll apply Bayes' Rule in data analysis where we'll predict the survival of a passenger of the Titanic using a Gaussian Naive Bayes classifier. When we work with data in the real world, it's important to understand the intuition and the mathematical formula behind Bayes' Rule. You won't be actually using the mathematics. It'll be done for you using built-in libraries that are available.

Intuition behind Bayes' Theorem

Bayes' Rule, or Bayes' Theorem, is a widely-popular and commonly used theorem to deduce probabilities. Bayes' Rule can be used to calculate the probability that a particular event will occur given existing knowledge about similar events or scenarios. Bayes' Theorem might seem

mathematically complicated. Let's first understand the intuition that lies behind this theorem. Swoosh as a binary classification problem. Let's say you have runners and police officers, and you want to classify a person who jogs past you on the street as one of these, a runner or a police officer. Now it's possible that you have some up-front knowledge about the conditions today. These are known as A priori probabilities. Individuals that job past you on the street could be runners or police officers. Those are the only two choices that we are considering here. Let's consider that for every nine runners, we have one police officer out there. That's because we know that today is the city marathon and there are more runners than police officers out on the streets. The A priori probabilities tell us that out of every 10 individuals who run past you, 9 are runners, thanks to the city marathon, and 1 is a police officer. Using just the information that we have available, the A priori probabilities, the probability of an individual being a runner is 9 out of 10 and the probability that an individual is a police officer is 1 on 10. Now these are known as A priori probabilities because these are priorities that exist before anything specific about the running individual is known. Someone is jogging past you on the street. You know nothing about that person. You only know that today is the city marathon. Now if you observe the individual who is running, you might get more information to predict whether this person is a runner or a police officer. Let's see you took a close look, and you saw that this individual is carrying handcuffs, a walkie-talkie, and has on running shoes. This is our second observation, conditions that are specific to the individual who is running. It's pretty clear from this list that there specific items that appear more often associated with one category of individuals than with the other. Observing these specific items allow us to set up conditional probabilities for our deduction. Now let's set these up in the form of a table. The items that individuals might carry include handcuffs, running shoes, guns, badges, or walkie-talkies. The second column here is the number of occurrences of these items that we saw with police officers. Six of them were carrying handcuffs, two running shoes, nine guns, eight badges, and eight walkie-talkies; and the third column is the occurrences of these items with runners. Of the jogging individuals, eight of them wore running shoes, three were carrying walkie-talkies, none of them had handcuffs, guns, or badges. Once you have some idea of these conditioning probabilities, you can use this information to make deductions for an individual who jogs past you. Upon closer examination, you saw that that person had handcuffs and carried a badge. The person that zipped past you carried these two items. It's pretty clear that with this additional information, the deduction that you can make about this individual is significantly improved. And here is where you'll use Bayes' theorem to calculate conditional probabilities. Figure out the probability that an individual is a runner given that you know that he or she is carrying handcuffs and a badge. This is the first conditional probability that you'll calculate. Exactly how you'll calculate these probabilities we won't discuss

right now. We'll look at that in the next clip. Now the second probability that you need to calculate is the probability that the individual who zipped past you is a police officer provided he or she was carrying handcuffs and a badge. After having evaluated both of these conditional probabilities, you'll now perform a comparison. You'll compare the probability that the individual is a police officer provided he or she was carrying handcuffs and a badge and the probability of the individual was a runner given the handcuffs and the badge. You'll then make your prediction about this individual by picking the label with the higher probability, and that's it. This is the intuition that you'll use to apply Bayes' theorem to make deductions.

A Priori and Conditional Probabilities to Classify Data

Bayes' theorem in the real world is often used for classification problems. Should I lend to this individual or not given his past lending history? Should I buy this stock or not given current market conditions? Naive Bayes' refers to the fact that all of the features that go with calculating conditional probabilities are considered to be independent. That's what naive refers to. In order to make predictions in the real world, you might set up a machine learning model. Let's consider an ML-based binary classifier. We have a training data that we'll feed to train a classification algorithm, and we'll get an ML-based classifier as a result. You can have this classification algorithm use Naive Bayes', that is apply Bayes' theorem of conditional probabilities. Now let's work with our training data first. Let's use some reviews data. These are some example reviews that we have for movies, and they have been labeled as positive or negative reviews. Let's assume that these five reviews here represent the entire corpus of our training data, just for simplicity. We'll now apply Bayes' theorem to probability information from this training data and use that to classify new problem instances. A priori probabilities refers to information that we have up front about the data as a whole. We have reviews that can be positive or negative, and we have a total number of reviews. Out of the reviews that we saw, three were positive, two were negative, with a total of five. So the first observation that makes up our A priori probabilities are the fact that there are more positive reviews than negative reviews in the training data. If we only consider A priori probabilities and we have no specific information about a review, for any new review that comes in, we'll basically say that the probability that it's a positive review is 3 by 5, the probability it's negative is 2 by 5, probability of positive greater than probability of negative, hence it's a positive review. Well, that isn't great. Remember that these are just A priori probabilities, probabilities that exist before anything specific about the review contents is known. Let's go back to our corpus of reviews that make up our training data, and let's evaluate some conditional probabilities. There is a second observation that we can make here. There are specific words that

occur more in one type of review than in the other. Some words tend to be in positive reviews and others in negative reviews. A specific example from our tiny corpus here is the word up. It appears twice in positive reviews, but 0 times in negative reviews. Similarly, you'll find that the world worst does not appear in positive reviews at all, but it appears twice in negative reviews. Well, this is great and useful information. Let's expand these conditional probabilities and calculate the number of occurrences of each word in our corpus in negative, as well as positive reviews, and here is a table that you might get. At the bottom here, you can see that there are a total of 9 words that appear in positive reviews and 10 words that appear in negative reviews. Now on this small and simple data set, it's fairly straightforward to calculate conditional probabilities. Let's set up the probabilities of occurrences of these words in positive reviews. Amazing is 1 by 9, two is 1 by 9, thumbs is 1 by 9, and so on. We also have a second column giving us the probability of occurrences of these words in negative reviews. Worst is 2 by 10, movie is 1 by 10, ever is 1 by 10, and so on. Let's consider just one of these words here, the word amazing. The probability that it occurs in a positive review is 1 by 9. What we have here is a conditional probability, the probability that the text contains amazing given that the label is a positive review is 1 by 9, and correspondingly, the probability that the text contains amazing given that the review is labeled negative is 0. For our training data, let's examine these conditional probabilities for a different word here, the word worst. This occurs 2 out of 10 times in negative reviews. Here are the conditional probabilities. The probability that the review text contains worst given that it's a positive review is 0, and the probability that the review contains worst given that it's a negative review is 2 by 10. We worked with a small data set to get a feel for the mathematics, but as you can imagine, no matter what the size of your data, you can calculate conditional probabilities for each word that occurs in your data.

Applying Bayes' Theorem to Make Predictions

At this point, we have the A priori probabilities for positive and negative reviews, and we have conditional probabilities for individuals words in a review. Let's use this information to classify a new problem instance. Given the review really bad, the worst, is it positive or negative? So what we are actually trying to figure out or predict is given the words in this review, let's call these words t, is this review likely or negative? And this is where we'll apply Bayes' theorem of conditional probabilities. We want to find the probability that given the text t, this is a positive review. So this is a probability label is equal to positive given the text really bad, the worst. Step one is to find the probability that this particular review is actually positive. We have step two where we try to calculate the probability of this review being a negative review. What is the

probability that this review is negative conditional on the fact that the text is really bad, the worst? We'll talk about how exactly to calculate both of these probabilities, but once you have them, you'll simply compare the two. Compare the probability that the review is positive given t to the probability that the review is negative given t if P. And then in order to classify this review instance, we'll pick the label with the higher probability. If P of Positive given t is greater than P of negative given t, we'll classify t as positive otherwise you'll classify t as negative. Now the t that we talk about here in this problem instance refers to the words that make up our review text, but t can be any features based on the problem that you're trying to solve. When you use Naive Bayes' to calculate conditional probabilities, it makes naive or strong assumptions about the independence of features. It assumes all of the features that you're using to train your model are independent. They are not related in any way. Now that we have the intuition, let's look at the mathematical formulation of calculating conditional probabilities. The probability that the review is positive given t is given this formula that you see here on screen. Let's just focus on the numerator for now. It's equal to the probability that the text appears in positive reviews multiplied by the probability that the review is positive overall based on A priori probabilities. Don't worry if this is getting complicated. We'll work through this. Now let's calculate the conditional probability that a review is negative. Remember, this was our step two, the probability that the review is actually negative given the text of the review. Once again, let's focus on the numerator. The first term in the numerator i the probability that this particular text appears in negative reviews multiplied by the overall probability that the review is negative, the A priori probabilities. Now let's attack these two mathematical formulations, which calculate conditional probabilities one on top of another, and you'll immediately observe something, the denominator of both of these are exactly the same. Given the denominator is the same, we can simply ignore this and focus only on the numerator to compare these two. When you have two fractions with the same denominator, only calculating the numerator will allow us to compare those fractions. That's what we want to take advantage of now. Let's take a look at these two terms here, the probability that a review is positive or negative. Both of these terms highlighted in red refer to the A priori probabilities that we had calculated up front. Remember, A priori probabilities are those before we know anything about the review contents. We know that three out of five reviews in our training data were positive reviews and two out of five were negative reviews. There are more positive reviews than negative reviews in our training data. Given that we know this, let's replace the denominator with actual numbers, P of Positive is 3 out of 5 and P of Negative is 2 out of 5. Now let's mov on to the second term that we are yet to calculate in these numerators. The probability that the text occurs in positive reviews and negative reviews. The problem instance that we are trying to classify is the text really bad, the worst. Let's see the probability that this text appears in positive reviews. This

is equal to the probability that the text contains really bad, the worst, and an an operation or a multiplicative operation between these probabilities. What we've done here is taken individual words in our review text and calculated the probabilities that they appear in positive reviews. If you remember the table that we had set up using our training data, we have these conditional probabilities. We have the word worst, bad, and the probabilities that these words occur in positive, as well as negative reviews. Let's just focus on one of these words here, the word worst. The probability that the text contains worst in positive reviews is 0 and the probability that the text contains worst in a negative review is 2 on 10. Let's plug this number into the Bayes' theorem formula that we were working with. The probability that this particular text, really bad, the worst, is present in a negative review is 2 on 1000. The word really does not appear in our training data, so we can ignore that probability, and we simply multiply the probability of occurrence of other words, bad, the worst. And let's compare this to the probability that this text occurs in a positive review. These are our conditional probabilities that we had calculated earlier. We can see that it's 0. That's because the probability that the word bad and worst occurring in positive reviews is equal to 0, which is why we get a result of 0. Once we have these number, we can plug this into the Bayes' theorem formula. So the probability that this is a positive review is 0 multiplied by 3 by 5, and the probability that this is a negative review is 2 by 1000 multiplied by 2 by 5. And when you apply this theorem, you pick the label that has the higher probability, which means it's very obvious here. If this were the new instance that you want your ML model to classify, you'd see that this is a negative review, and this is how you'd use conditional probabilities calculated use Bayes' theorem to solve a business problem in the real world. And really, when you're working in the real world, you just need an intuitive understanding of the mathematics involved. Libraries will take care of the calculations for you.

Calculating a Priori Probabilities of Survival on the Titanic

In this demo, we'll work with an example where we'll apply Bayes' Rule in data analysis. We'll train a Naive Bayes' classifier in the scikit-learn library on training data and use that classifier to make predictions. Here we are on the Azure Notebooks portal in the data analysis project that we had set up earlier. Within the dataset folder here, I'm going to upload some data that I'm going to work with. This data is available in the form of a CSV file on my local machine. Choose From Computer, select the Choose files option, use the Explorer or Finder window to locate this data, titanic.csv. upload this data to the dataset folder on the Azure cloud, and let's work on a brand-new notebook to perform our analysis. Click on Done, and you'll find that this data set has been successfully uploaded. This titanic data set contains passenger information for the various

passengers who traveled on the Titanic, and we'll use this information to predict whether a passenger survived or not, and we'll apply Bayes' Rule for this. I'm going to write all of my code in this notebook, NaiveBayesClassifier. We won't be performing the Naive Bayes' mathematical computation manually. We'll use the GaussianNB estimator available in the scikit-learn library. Once we make our prediction after applying Naive Bayes', we'll check whether our predictions where accurate using the accuracy score function from sklearn.metrics. This titanic data set is freely available to use and download on kaggle.com, and you can check it out at this URL if you're interested. I'll read in the contents of the CSV file into a Pandas data frame, and let's take a look at a sample of the data that we just loaded. As you can see, this contains passenger information for people who traveled on the Titanic. The Survived column here is 1 if an individual survived the sinking of the Titanic. It's 0 otherwise. The remaining columns are passenger details such as the class in which the passenger 1, 2, or 3; the name; sex; age; and other details of passengers. if you take a look at the shape of this data frame, you'll see that we have about 1300 records, 1300 passengers, and 13 columns for each record. We won't use all of the information that we have available. We'll only focus on two columns, the sex, or the gender of individuals, and whether they survived or not. The Sex column here is categorical in nature and comprises of discrete strings. I'm going to convert these to category codes so that they are represented in the form of numbers. Convert this column to be type category and extract the category codes. And if you take a look at the resulting data, you'll see that females are represented using the integer 0 and males are represented using the integer 1. Before we use this data for analysis, let's check whether there are any null values. Yes, indeed, they are in the Sex column, so let's go ahead and drop those records with null or missing values. We are left with a total of 1309 records. I'm going to extract the information that we are going to work with into two separate data frames, features and label. Features include the Sex column and the Survived column, and the label includes just the Survived column. You're not going to using this data directly to train an ML model yet, so don't be confused by the fact that the Survived column information is present in the featured data frame. We still have some A priori probabilities that we need to calculate. I'll now split this data into training and test data. The training data is what we'll use to calculate A priori probabilities for our Bayes' theorem. This data shuffle and split is very easily performed using the train, test, split function from scikit-learn. You can see that we have about 1000 instances in our training data, that is 80% of the original data set, and about 262 instances in the test data, 20% of our data set. What do we know up front? What are our A priori probabilities? We know the number of people who survived or did not. Let's calculate these values for our training data by calling y_train.value_counts. The information that we have up front from our training data is that 645 people did not survive, 402 did. Let's express this information in the form of percentages.

These make up our A priori probabilities of survival. Survival probability is 38.39. Before we know anything about a new instance for a new passenger, the A priori probabilities tell us that the probability that a particular passenger will survive the sinking of the Titanic is just 38.39 %. Let's calculate these same survival probabilities for the test data. Out of the test data, you can see that 164 passengers did not survive, 98 did. Let's convert these numbers to percentage values for the test data. And the probability of survival on the test data is 37.4. You can see here that it's very close to the probability of survival that we got on the training data. So if you consider the data set overall, you can see that the A priori probabilities work well in order to calculate how many people survived or did not in the test data. But let's say we know something more about passengers. Let's consider individuals by gender. Here are all of the men in our test data set, 166 of them. We know that the male gender is represented using the category code 1. So I've just extracted all of the men from my test data. Let's do the same thing for the women passengers in our test data set. I'm going to extract all of the women, and you can see that there are a total of 96 passengers in the test data who are women. If you remember, we represented the female sex using the category code 0. Now if you have this additional information about the gender of passengers, let's see the number of men who survived the sinking of the Titanic. You can see that 138 did not and only 28 did. This should make sense given what we have heard about the ship. Women and Children first on the lifeboats. If you calculate the probability of survival of men on our test data, you can see that it's very low, only 16.86 %. If you were to use A priori probabilities on a passenger without taking into account whether the passenger is male or female, you can see that we can get our survival prediction quite badly wrong. Let's do the same for women. Of all of the women in our test data set, let's see how many survived and how many did not. Seventy of them survived, 26 did not. It's pretty clear that if you were a woman on the Titanic, you're more likely to have to have survived. Let's calculate the survival probabilities for women on the test data set, and it's 72.9 %. Once again, it's clear that for a passenger picked at random, if you don't take into account whether they are male or female, you can get their survival probabilities very wrong. And this is where applying Bayes' theorem to calculate the probability of survival would really help, and that what we'll do in the next clip.

Applying Bayes' Rule Using the Naive Bayes Classifier

We'll apply Naive Bayes' theorem to predicting the survival of a passenger on the Titanic using the Gaussian Naive Bayes' classifier from the scikit-learn library. The training features that we'll use to train our classifier will include only the gender, or sex, of the passenger, so I'm going to go ahead and drop the Survived column from our x_train data. I'll drop the Survived column from the

test data as well. We'll use this information to calculate A priori probabilities. We are done with that. We can get rid of this column. Now the training data that we'll use to train our machine learning model and the test that we'll use to evaluate our Naive Bayes' classifier contains just one column, the gender of the passenger. Like I said earlier, in the real world, when you apply Bayes' Rule, you won't actually perform the mathematical computations for conditional probability use. Simply use an existing library, and here, we instantiate the Gaussian Naive Bayes' estimator object from the scikit-learn library. We call model.fit, pass in the training data and the Y labels indicating whether the passenger survived or not. This is what we'll use to train the mode. Under the hood, this classifier applies Bayes' theorem to calculate conditional probabilities. We have just one feature here, that is the gender of a passenger, and Bayes' theorem will be applied on that bit of information. Once we have a fully-trained Naive Bayes' classifier model, we'll use this model to predict on the test data. Remember the test data are instances that a model hasn't encountered before. The accuracy of this model's predictions on the test data will allow us to evaluate how good this model is, and we can compute this by calling the accuracy_score function. Using Naive Bayes' to calculate conditional probabilities based on a person's gender gives us an accuracy of 79%. Seventy-nine percent of the predictions of this model were correct, and this is a huge improvement over using just the A priori probabilities. Let's see how our Naive Bayes' prediction model performs when we segregate our passengers by gender. Before we do that, I'm going to set up two additional columns in my test data field, Actual Survived versus Predicted Survived. The actual survival labels come from the original data set, and the predicted values come from our model. So our test data frame now contains the sex or gender of the passengers whether they actually survived and what our model predicted for them. Let's now extract the male and female passengers into separate data frames, x_test_men and x_test_women. With this data separated, let's calculate the accuracy of our model on the men, and you can see that it's 83% and the accuracy of our model in making predictions for women, and you can see that it's 72.9 %. Applying Bayes' Rule and calculating conditional probabilities of survival based on gender has greatly improved our prediction.

Module Summary

And with this, we come to the very end of this module where we started the application of Bayes' Rule in data analysis. We started this module off by first getting an intuitive understanding of Bayes' Rule and how it can be applied to predictive analytics. Bayes' Rule involves the calculation of conditional probabilities. Calculating the probability that a particular event will occur given that another similar event has already occurred. Once we have the intuitive understanding, we moved

on to the mathematical formulation of Bayes' theorem and applied this mathematical formulation to classify text as positive or negative. And finally, we saw how we could apply Bayes' Rule in data analysis. We worked with the titanic data set, and we saw how A priori probabilities of whether passengers survived the sinking of the Titanic or not didn't really work well when we took specific conditions of passengers into account, such as their gender. We then trained a Naive Bayes' classifier using gender information of passengers and got very accurate predictions of survival. In the next module, we'll focus on visualization. We'll see how we can visualize probabilistic and statistical data using the Seaborn Python library.

Visualizing Probabilistic and Statistical Data Using Seaborn

Module Overview

Hi, and welcome to this module on visualizing and probabilistic and statistical data using the Seaborn Python library. Seaborn is a very commonly used library for data visualization and analysis. We'll first see how we can visualize univariate distributions using Seaborn, we'll visualize and explore our data using histograms, KDE plots, and rug plots. We'll then visualize bivariate distributions to explore relationships between variables. We'll use visualization techniques such as joint plots, Hexbin plots, KDE plots, and heat maps. We'll see that Seaborn offers this very useful visualization tool to explore pairwise relationships, the pair grid and the pair plot. We'll then use regression plots in Seaborn to see whether linear relationships exist between pairs of variables. We'll also see how we can visualize categorical data using specialized plots, such as strip plots, swamp plots, box plots, and violin plots.

Understanding Kernel Density Estimation

When you visualize statistical data, a plot that you'll encounter often is the KDE plot, or the Kernel Density Estimation plot, let's understand what that is. Now you might have heard this statement being made, Michael Jordan is a once-in-a-lifetime player. What does that mean? If you plot the basketball skill of players on this number line where the highly-skilled players are off to the right,

you'll find that Michael Jordan is an outlier, a once-in-a-lifetime player is an outlier represented by a data point that is far from the pack. If you were to plot basketball skill along this line and then plot a frequency distribution of players at this skill line, in reality, you'll find that most ordinary folks will be clustered around an average level of skill. A vast majority of the people is represented by this single tall bar that you see at the center. NBA players would be outliers in themselves. Michael Jordan would be an even greater outlier. You can imagine this chart as frequency distribution of players at a particular skill level. This chart tells us how common a specific level of skill is. The tall bar at the center shows you that an average level of skill is the most common. You can see that the shape of this chart resembles a bell. This is a normal probability distribution, or a bell curve. This bell curve, or the normal probability distribution, occurs to many things that occur in nature. Height and weight distributions, skill, intelligence, all of these can be represented using the normal probability distribution. The normal probability distribution makes it very clear that average levels of skill, intelligence, etc. are very common. Very high and very low values are both unusual. This bell curve occurs everywhere in nature. Now let's say for any data, you want to be able to answer this question, what is the probability of any specific value x occurring in the data? The answer is lies in a probability distribution function, and the normal distribution is one such probability distribution function. The probability distribution function is often referred to using the acronym PDF. With that in mind, let's move on to discussing the Kernel Density Estimation, or the KDE. If you have raw data and you express this data in the form of a histogram, the Kernel Density Estimation is a mathematical technique that you can use to get a small probability distribution from this histogram. You have a bunch of raw data. These are your data points. Given a set of points, here is a histogram representation of their points. Use this histogram representation to figure out the probability distribution. Of this data, the probability distribution can be expressed using a PDF, or a probability distribution function. The area under the curve that you've just drawn here must sum up to one. The KDE, or the Kernel Density Estimation, is a standard technique that is used to find the probability distribution function that applies to your data. This is a non-parametric smoothing technique. The kernel used for Kernel Density Estimation is often a Gaussian Kernel. This is when your probability distribution function is the normal distribution, or the Gaussian probability distribution, defined by a mean or average value of new and a standard deviation of sigma. You're already familiar with the concept of mean. Thea mean u is the center point of your data. The standard deviation is a measure of variance and often referred to as the bandwidth of the KDE. The bandwidth is a hyper parameter for your Gaussian Kernel. The bandwidth determines the shape of your kernel whether it's a tall, skinny probability distribution function or a short and a broad one.

Histograms, KDE Plots, and Rug Plots for Univariate Analysis

In this demo, we'll visualize statistical data and observe relationships using the Seaborn library for visualizations. Let's get the latest version of Seaborn. The Azure default version is not the latest one at this point in time. Simply call pip install -U seaborn. Once this latest version has been downloaded and installed onto your machine, we can set up the import statements for the libraries that we'll use. Pandas, NumPy, Seaborn, as well as matplotlib, remember? Seaborn is built on top of matplotlib. For univariate analysis, we'll work with the Height data set that we are familiar with. We'll use pd.read_csv to read this into a data frame. Seaborn allows you to view the probability distribution of a variable using the distplot, or the distribution plot. Call sms.distplot, and let's pass in the Height data and see what the distribution looks like. Now the default representation of the distribution is a histogram of the Height values. On the x-axis, we have the bucketed values for Height, and on the y-axis represents the frequency counts for every height bucket in terms of proportions. In addition, there is also a KDE plot, the Kernel Density Estimation curve. Kernel Density Estimation is a fundamental data smoothing problem where inferences about the population are made based on a finite data sample. Let's say you're interested only in the histogram of the Height data and not the Kernel Density Estimation curve, you can specify kde=False for the sns.distplot function. The KDE curve has disappeared in the resulting visualization if you just have the histogram, and on the y-axis, we have the actual frequency counts. What if we want a horizontal representation of this histogram? Specify kde=False and vertical=True. This will switch all of the bars to be horizontally oriented. The buckets of the Height data are now along the vertical axis. That's why vertical=True. Along with the histogram of the data, if you want to view the locations of the actual data points, that's possible using the distplot as well. Rug=True will indicate all of the individual data points in the form of sticks on the axis. Every little stick here is an actual data point in our data set. You can use other input arguments to the distplot function to customize your histogram further. Bills=40 will divide your data into 40 different bins, and here is the resulting histogram. If you're only interested in the actual data points, not the histogram or the KDE curve, you can call sns.rugplot and specify a height for every data point stick. The height that we've specified here is 0.5. You can see that this rug plot comprises of only sticks representing individual data points with the height of 0.5. This shows us very clearly that our height data is fairly evenly distributed within the min and max of this range. Seaborn offers a very rich set of color and palette customizations as well. You can set color_codes=True and specify shade=True in order to get a shaded background for your visual. The kdeplot for the univariate data plots just the Kernel Density Estimation curve. Here is our KDE curve with a shaded background. The area under the KDE curve is also shaded the default blue

color. You can change the color under the KDE curve by specifying shade=True, color=g. Here is the same kdeplot. Now the area under the curve is shaded a bright green color based on our customization. Let's take a look at one last interesting kdeplot, or multiple kdeplots on the same visual. When we simply invoke the kdeplot function, Seaborn uses a default value for the bandwidth. But this is something that you can configure. Here, I've plotted a kdeplot on the same chart, but with a much smaller value of bandwidth than the default, bandwidth of 0.1. And finally, I've plotted a third kdeplot with a bandwidth of 1. For this particular data set, this is again smaller than the default value of bandwidth, which is calculated using a technique under the hood. Take a look at the resulting visualization. The orange line represents a KDE curve with a bandwidth of 0.1. You can see that it's skinny, and it's very spikey and not smooth; whereas, the green line is bandwidth of 1, it once again has more curves and fluctuations than the default KDE curve, but it's much smaller than the KDE curve with a bandwidth of 0.1.

Scatter Plots, Joint Plots, Hexbin Plots for Univariate Analysis

In this demo, we'll perform some bivariate analysis of data using Seaborn visualizations using scatter plots, joint plots, Hexbin plots, and so on. Set up the import statements for all of the libraries that we'll use, Pandas, Seaborn, and matplotlib. I'm going to use Pandas to read in a new CSV file containing some data about tips at restaurants, tips.csv. This is an extremely interesting data set. If you take a look at the columns, you'll see that we know the total bill, the tip paid, whether it was a male or a female, smoker or nonsmoker, whether they had lunch or dinner and the size of the party. This is the data that we'll analyze using Seaborn. Let's take a look at our first Seaborn scatter plot, which we can plot by calling sns.scatterplot. We'll use the scatter plot to view the relationship between total_bill and tip. All of the data is present in the tips_data data frame that we pass in. And you can see from the scatter plot that this is almost a linear relationship. Higher the bill, higher the tip. That makes sense. To this bivariate analysis, we can add an additional dimension using the hue of the points that we plot. Hue=smoker. This will point plots for smokers and nonsmokers in different colors. From the distribution here, you can tell that there is no real difference of the tipping habits of smokers versus nonsmokers. I'll plot the same scatter plot as before, but instead of distinguishing between smokers and nonsmokers, I'll distinguish based on the time of day that that meal was had, dinner or lunch. The time of day will be represented by the hue, as well as the style of the markers. There's a pretty clear indication in this scatter plot that people tend to tip more at dinner. Dinner is represented by the blue circle. To our original bivariate analysis, the relationship between total_bill and tips, I'm going to add two additional dimensions. I'm going to represent the day of the week using the hue, and I'm going to

use the size of the bubbles to represent the size of the party at lunch or dinner. you can see that seaborn handles this very well, but as you add additional dimensions to the scatter plot, the data becomes harder to interprate. So we have different sizes of bubbles for the size of the party and different colors for the different days of the week. Let's make interpreting our scatter plot a little easier. I'm only going to represent one additional dimension, that is the size of the party, but I'm going to represent it using the hue, as well as the size of the bubbles, and I'm going to make the bubbles larger on the whole. I've specified that the bubble size ranges from 30 to 300, and I want a full legend. Let's take a look at the resulting visualization, and you can see that this is much easier to interpret. The larger bubbles in the darker hue are larger parties. There are more people having lunch or dinner, and they tend to tip quite well. Histograms, which give you an idea of the frequency distribution of data, are typically used for univariate analysis. Hexbin plots are the bivariate analog of histograms. Let's take a look at what a Hexbin plot looks like before we continue our discussion. You can see that it's kind of like a scatter plot. We have the tip on the y-axis and the total bill on the x-axis, and the entire area of the plot is divided into hexagonal bins. The color of every hexagonal bin gives you an indication of the count of observations in each bin. You can see that most of the meals hovered around the $12 range and the tip was around $2. A joint plot in Seaborn gives you two bits of data. It's a scatter plot, allowing you to see the relationship with bivariate data, and it also plots a frequency distribution of the individual variables. So in addition to the scatter plot showing you how the tip varies with the total bill, you can see a histogram representation of the total_bill and the tip variable individually. the joint plot is a pretty customizable API. You can use it to view relationships with different kinds of plots. For example, you can plot a Hexbin plot using the joint plot API. I'm set the axes_style to white here so that we have a white background, and I'm using a jointplot Hexbin representation of my data, and the color used to represent the frequency counts should be red, and here is what the resulting Hexbin plot looks like. The jointplot function can be used to plot KDE plots, scatter plots, and other kinds of plots as well. You can also use the kdeplot function directly if you're looking for a KDE representation for your bivariate data. You can see that we have the same relationship now expressed in the form of the Kernel Density Estimation curve. And as we have discussed earlier, if you're looking to understand the relationship that exists between pairs of variables, the correlation data is a very useful bit of information. You can generate the correlation matrix by calling the .corr function on your Pandas data frame. Here, you can see that the size of the tip is positively correlated with the total bill, we already knew that, and the size of the tip is also positively correlated with the size party having the meal. All you need to do now is to pass this correlation matrix into a heat map. And with this, we have the heat map representation of our correlation data.

Regression Analysis on Bivariate Data

In this demo, we'll see how we can perform regression analysis using Seaborn visualizations. Seaborn offers two different visualizations for regression analysis, the lmplot and the regplot. The implementation under the hood is the same. The lmplot is a figure-level function so it has fewer customizations. Let's take a look at the lmplot first. We'll work with the tips data set that we are familiar with. Here is the data set. The shape of this data set shows us that there are 244 records and 7 columns of data. Let's plot the sns.lmplot to see the relationship between the total_bill on the x-axis and the tip on the y-axis. The lmplot representation will display a scatter plot of the original data and also try to fit the regression line along with a confidence interval on this data. This is great for exploration because it allows you to see clearly whether a linear regression analysis makes sense on your data. You can use lmplot to fit multiple regression lines based on an additional field. Simply specify the hue attribute. Here, we are going to distinguish between smokers and nonsmokers. Observe that the data points are in two different hues. Orange points represent smokers, and there are two separate regression lines, one for smokers and one for nonsmokers. The slope of the two lines seems to indicate that nonsmokers perhaps tend to tip more. We can't be sure though. You can use the lmplot to visualize regression on categorical data as well. Here, x is the size of the party, which is a discrete number from 1 all the way through to 6, and y is the tip. We've also changed the aspect ratio of the resulting graph with an aspect of 1.5. And here is what the data looks like. You can see that parties of size 4 tend to tip much more. Observe that for each of these categories, all of the data points are kind of clumped together. If you want to view the data points separately to see how many data points there are, you can see x_jitter to assert a numeric value, and you can see that the resulting plot has the data points spread out for each category. This makes it much clearer that there are many more parties of size 2 in our data set. You can also use lmplot to display multiple regression plots based on a third dimension. Here, I want two regression plots based on whether party was a smoking party or not. The resulting visualization is two separate plots for smokers and nonsmokers, allowing you to view the differences between the two side by side. Here is another additional customization. I want different regression plots for total_bill versus tip. The hue is by day, and I want the columns also to be by day. We have information for four different weekdays here, Thursday through Sunday, and you can see that each of these regression plots are displayed in a different hue or color. The slope of the line for Sunday in blue is kind of flat-ish, indicating that tips on Sunday tend to be lower. Seaborn also offers the regplot function for regression plots. It's very similar to lmplot. The functionality for regplot is a subset of lmplot. It has fewer customizations that you can use. I've plotted a simple regression plot here of tips versus total_bill. You can see that the result

is the same. It's the scatter plot along with a regression line. There, the regplot API has additional flexibility is in how you specify the user input to the plot. It can accept arrays and Pandas series objects as well. Lmplot cannot. Observe that instead of passing in the column names of our data frame for the x and y values, I pass in the columns directly. These are Pandas series objects, and the regression plot works with these inputs as well. So if you want more flexibility in your inputs, the regplot is what you'll use. For our last regression analysis in Seaborn, let's do something interesting. I'm going to add a new column to my data frame called bit_tip. If I find that the tip is greater than 17.5 % of the total bill, I'll assume that is big tip. Here is the resulting data frame with the new column added in. Big_tip is either true or false for each of these records. Now I'm going to plot a regression plotusing sns.regplot. On the x-axis, I have the total bill. On the y-axis, I have the True/False values, whether it was big tip or not for that bill, and I want this regplot to perform logistic regression, that is fit an s-curve on this data. This fitting of an s-curve, or logistic regression, is typically used for classification problems. Here is what the resulting visualization looks like. Seaborn has tried its best to fit an s-curve on the data that we specified.

Representing Pairwise Relationships Using the Pairplot and Pairgrid

In this demo, we'll see how Seaborn visualizations can help us explore pairwise relationships in data. We'll work with the data set that we are familiar with and have used before, the graduate student admissions data, which we'll read in into a data frame from our CSV file. Here is what the data frame looks like. Now I won't work with all of the columns available in this data set. I'm going to drop a few columns so that the visualizations are clearer to see, so I'm going to drop the TOEFL Score, the SOP rating, the letter of recommendation, and the university rating from the original data frame. We are left with just a few columns. It'll help us better analyze our data, GRE Score, CGPA, whether the student has done research or not, and the probability of admission. Now bivariate relationships are best expressed and viewed in scatter plots. The pairplot is a way for you to explore a bunch of relationships in one go. It plots all of the pairwise relationships that might exist in your data. In the resulting matrix, you'll find that the main diagonal represent univariate relationships, and we want that to be represented using histograms. This will make much more sense when you take a look at the resulting pairplot. Observe that this is a grid and every chart here represents a relationship between a pair of variables. Observe that the diagonals are histograms. This is a univariate distribution of just the GRE score, both the row and the column represent the GRE score. Here is a bivariate relationship, the CGPA plotted against the GRE score. The first column is the GRE score, and this row is the CGPA. The pairplot is a quick way for you to explore what relationships might be useful. Here is the relationship of probability

of admit versus the GRE score, the CGPA, and whether the student has done research or not. There are some customizations you can make with the pairplot. You can specify that the diagonal should be a kde curve rather than a histogram, and that's what you see in the result here. The pairplot, in addition to pairwise relationships, can also represent an additional dimension using the hue of the scatter plots and their kde or histograms. We are going to represent students who have done research or not using different colors. And in the resulting pairplot, you can see that the diagonal has two KDE curves, one for students who have done research and another for those who haven't, and the scatter plots have the data points colored in two different colors as well. The pairplot is a higher-level API. If you want additional customizations for your pairwise relationships, you should use a pairgrid. In order work with the pairgrid, I'm going to load in the iris_data set, which is available as a built-in data set in Seaborn. The iris_data set is a data set of iris flowers with the sepal and petal lengths, as well as widths. There are 150 records in this data set, and every iris flower belongs to one of three species, versicolor, virginica, and setosa. The pairplot that we've worked with so far under the hood uses the pairgrid, which is a lower-level API to plot pairwise relationships. It gives us more flexibility and control over the plot. Initialize the pairgrid with the iris_data set that we are going to work with and call grid.map, and let's plot a scatter plot using this Pairgrid. All of the plots, including the diagonals you can see, are scatter plots. If you don't want all of the data to be represented in your pairwise relationships, you can specify a subset using the vars input argument. Here, we'll just plot the petal_length and petal_width in the form of a scatter plot. This makes the pairwise relationship easy to view since we just have two variables. Let's add to the customizations of this PairGrid. I'll color the data points based on the iris species of flowers that they represent. I want the diagonal to be represented in terms of a histogram. That's why I've said map_diag plt.hist. The diagonal represents univariate data. The bivariate data, the off-diagonal elements, should be represented using scatter plots, and I also want a legend for this grid. And here is what the resulting PairGrid looks like. The diagonals are histograms as we had specified, and there are three sequences of histograms based on the iris species. All of the off-diagonal elements are scatter plots, and each kind of iris species are represented in a different color. For example, the green color is for the virginica species.

Visualizing Categorical Data Using Strip Plots and Swarm Plots

In this demo, we'll see how we can visualize categorical data, that is data which has discrete values using strip plots and swarm plots. Start with a new Azure Notebook, import the libraries that you'll need, and let's read in the tips data set. This is the data set that we are familiar with. It needs no introduction. Often, the data that you're working with is categorical in nature, that is,

made up of categories or discrete values. You can use a stripplot with categorical data. On the x-axis, we have the sex, or the gender, of the diners. And on the y-axis, we'll represent the tips that they give. This is what a strip plot looks like. Notice the categories on the x-axis, and notice how the tip data is spread out on the y-axis. You can see that among males, there are a few tippers who are outliers and give very large tips. The stripplot has jitter enabled by default so that you can see the individual data points. If you set jitter to false, all of the data points will be available as a single line. They are not spread out. Let's represent a different category on the x-axis. This time, the categorical data is the day of the week on which the diners dined out. We've also assigned a numeric value to the jitter. The points will not be very spread out jitters equal to 0.04. This stripplot indicates very clearly that tips on Saturday tend to be higher and have outliers as compared with Friday. A variation of the stripplot is the swarmplot. This is also used to represent categorical data; however, the difference is it draws a categorical scatter plot with nonoverlapping points. Let's understand this for plotting just the tip data using the swarmplot. The points are adjusted along the categorical axis so that they don't overlap. All of the individual points will be visual. Because you can see individual data points, this gives you a better representation of the actual distribution. You can see that most of the tips here are clustered around $2. Let's plot a swarm plot with the size of the party represented along the x-axis and the tips along the y-axis. Here is what the result looks like. You can see that when the party size is just 1, tips tend to be low. Tips tend to be the highest when the party size is around 4. Let's add in an additional dimension to our swarm plot. Along the x-axis, we have the size of the party, y represents the tip, and I'll also have the hue of the data points represent the gender of the party. And here is the resulting swarm plot. You can see here that for a party of size 4, all of the outlier tips are by males. Let's take a look at one last interesting customization of a swarm plot before we move on. You can set the palette of your plot to be something different so that they are represented in different colors. Here, palette is Set 1. And you can set dodge=True if you want to separate the data points based on the hue. If this wasn't clear, the visualization should make things clearer. Observe that the red data points and the blue data points are now separated out so you can see the data points for males and females separately. They are not overlapping with one another.

Visualizing Data Using Box Plots and Violin Plots

In this demo, we'll work with statistical measures for categorical data using box plots and violin plots in Seaborn. Go ahead and import all of the libraries that you need. We'll work with the admissions data set that we are very familiar with. It needs no introduction. A box plot, as you

already know, is great for viewing statistical measures on your data. Let's view a single box plot for the GRE score distribution among the students who applied to US universities. You can see that the scores range between 290 and 340 with a median around 318. Box plots are usually viewed in the vertical orientation. Simply specify orient=v in order to see the same box plot set up vertically. Let's specify some categorical data along the x-axis of the box plot. Along the x-axis, we have the university rating. And along the y-axis, we have the GRE score. You can see that for higher rated universities, the distribution of the GRE scores tend to be on the higher end. You can represent an additional dimension of data in your box plot using the hue. Here, I'm going to set up separate box plots based on whether the student has conducted research or not. And I'm going to use this box plot to see how the GRE scores for these students differ. You can see that for higher rated universities, students who have conducted research tend to have higher GRE scores. The violin plot is used to represent the exact same data and distribution as the box plot, but instead of a box, it uses the Kernel Density Estimation of the data points. Here is what the violin of the GRE score distribution among students looked like. Rather than a simple box, this is represented using the KDE curve. In addition to information on the descriptive statistics, this gives you an idea of how the actual data points are distributed. Let's now take a look at the violin plot representation of the university rating and the distribution of the GRE score of the applicants to these universities. We've seen this before. As the universities tend to be more highly rated, the GRE scores of students applying to those universities tend to be higher. Just like with box plots and violin plots, you can also separate the data based on another dimension. Here, we'll use hue=Research to color the data differently based on whether the student has done research or not. We've seen earlier that students who have done research in their undergrad often tend to have higher GRE scores. Let's make our visualization more interesting by adding in a new column to our data frame. For any student who has a CGPA of greater than 8.5, we'll say that the student has achieved first class. This is kind of like an honors student. If we take a look at the resulting data frame, we have an additional column where First Class is set to True or False based on whether the student's CGPA is greater than 8.5. Once again, let's visualize data in the form of a violin plot. On the x-axis, we have the University Rating, y-axis as the GRE score, and we add the First Class column as the hue in this categorical plot. We also set an additional configuration here, dodge=False, so that the data is present one on top of another and not split up, as you can see here on screen. The orange violin plot represents students who are First Class students, and you can see that uniformly their GRE scores tend to be a lot higher. In Seaborn, it's very easy to include multiple visualizations in the same chart. Here, we have a box plot for the GRE score versus the University Rating, and in addition, we'll plot a swarm plot with the actual data points as

well, and here is the resulting visualization, box plot for the distribution and a swarm plot for the actual data points.

Visualizing Categorical Data Using Bar Plots, Point Plots, and Cat Plots

In this demo, we'll study some other visualizations that we can use to explore categorical data such as bar plots, point plots, and categorical plots. We'll continue working with the tips data set. This is once again something that we are familiar with. Read the CSV file in into a Pandas data frame, and let's see how we can visualize this data using a Seaborn bar plot. On the x-axis, we have the size of the party, and on the y-axis, we have the total_bill. By default, Seaborn plots bars, giving you an estimate of the central tendency, or the average of the total bill, for each party size. The little vertical lines that you see are error estimates. These error bars are indications of uncertainty around those estimates. For every category that you plot on the x-axis, if you want a count of observations for each of these categories, you can use the sns.countplot. Along the x-axis, we have the size of the party, and you can see that the largest bar is associated with parties of size 2. Most of the diners are couples. You can also have a group bar graph. Call sns.barplot and specify a hue, and this will show you a bar plot for smokers, as well as nonsmokers because that was our hue specification. For the little data that we have, you can see that smokers on an average tend to have a higher total bill. The black vertical bars on a bar plot represent the error estimates. If you wanted to represent the standard deviation, you simply say ci= sd in your barplot specification. So you now have a bar plot, which represents the average total bill for the different sizes of your party, and you also have those little bars, which represent the standard deviation measures. I'll now add a new Boolean column to the tips data. This Boolean represents whether the parties dining were a couple or not. So whenever size of the party is equal to 2, I'll set couple equal to True. Here is the data frame with the new column added in. Now I'm going to plot a bar chart for the average total bill for the different party sizes, but I want to emphasize the parties that are couples. I want to color them in a different hue. Dodge=False will make this a stacked bar plot, a single bar for each category. In the exploration of your data, if there are specific categories you want to highlight, this is how you'd do it. You can see that the couple bar is orange in color. A point plot can also be used for categorical data. It'll represent the data in the form of points on a graph connected by a line. Here, we will represent the average of the total bill, and you can see that these points are connected using a line showing the upward trend. As the size of the party grows, the average total bill goes up. We know this already. Now if you want multiple point plots and you want them separated by sex, you'll specify hue= sex, and here is

what the resulting visualization looks like. The male gender is represented using the orange color, and you can see that they are outliers in the average total bill for a party of six. The catplot in Seaborn is a general API for all kinds of categorical plots, like such as the strip plot, the box plot, the bar plot. All of these are plots that we've studied separately earlier. Here is how you specify a box plot to be represented using the catplot function kind= box. here is a box plot representation for the distribution of the total bill based on the day of the week that the meal was had. The default kind for a categorical plot is a strict plot. If you specify hue= time, the data points in your strip plot will be colored based on the time, whether it's lunch or dinner. And here is how you'd specify a categorical plot displaying the form of a violin plot, kind= violin. As you can see, there are many ways to display the same data in Seaborn. You'll choose the one that makes the most sense for you.

Summary and Further Study

And with this, we come to the very end of this module where we used the Seaborn library to visualize statistical relationships in our data. We first worked with univariate data, and we used visualization techniques such as histograms, KDE plots, and rug plots for univariate analysis. We then moved on to visualizing bivariate distributions using joint plots, Hexbin plots, scatter plots, and heat maps. We saw that we could quickly explore any relationship that occurs between pairs of variables using the pair plot and the pair grid. We also saw how we could quickly explore linear relationships that exist in data using regression plots. We also saw how we could visualize and explore categorical data, data which is made up of discrete values. We used specialized plots such as strip plots, swarm plots, and also used the common box plot and violin plot, its variation. And with this module, we come to the very end of this course as well. If you're interested in working with data on the Microsoft Azure platform, here are some other course on Pluralsight that you could watch. Representing, Processing, and Preparing Data. This is a course that uses Excel, Python, and relational databases to work with data. Another course that you might find interesting is Communicating Data Insights. Here, we'll explore a number of different Python packages, matplotlib, plotly, as well as Seaborn to extract and represent insights gleaned from data. And with this, it's time for me to say goodbye. That's it for me here today. Thank you for listening.

Course author

Janani Ravi

Janani has a Masters degree from Stanford and worked for 7+ years at Google. She was one of the original engineers on Google Docs and holds 4 patents for its real-time collaborative editing...

Course info

Level	Intermediate
Rating	★★★★★ (10)
My rating	★★★★★
Duration	2h 48m
Released	20 Jun 2019

Share course