Week 16
Video Transcripts

Video 1 (6:42): Machine Learning - Rocks and Mines Dataset

We're going to learn a little bit about machine learning just to make things clear. This is a very very superficial introduction to machine learning, which is a very complex and deep topic. What we're going to do really is, focus on a few Python libraries that are available for machine learning, and actually a few parts of these libraries, and try to get a sense for what is it that is important when you're doing machine learning exercises. If you want to actually do some machine learning, then you should probably read up more on it.

As we go along, I'll point you to various resources and, but this should be a pretty, you know, quick, a pretty decent introduction. So, let's start by looking at machine learning, and our goal here is, so we look at 'regression', and our goal is to really walk through the steps in regression and see how we load the data, how we do the regression, and then how we can interpret the results so that it's not making sense to us. Because interpretation is really the key to understanding what the algorithm or the computer program has given to you. So, a general step is, of course, to read the data and then generate a few summary statistics and then try to understand what these things do. So, the data sets we're going to use for the machine learning exercise are mostly available online, and the first data set that we're going to focus on is the rocks versus mines data set.

This data set contains sonar soundings at different frequencies of a location underwater, so to speak, actually it's underwater, where there are rocks and there are mines, and the goal is to figure out whether we can learn - what is a rock and what is a mine from this...from this data. Obviously, the reason is that you don't want to accidentally step on a mine or swim into a mine. So, think of it like this, right, like, let's say this is, though this is underwater stuff but let's say, you're looking at a mine field in a war-torn... war-torn zone of some sort and you can see that there are rocks and there are mines and they look alike, they just look alike. But you send sonar sounds, soundwaves to the rocks and mines and you get different results back. So, the question now is that can we integrate these results to figure out which one is a rock, which one is a mine, and how—with what confidence can we walk through this minefield without blowing ourselves up, you know. That's the goal here, right! So, because the only real way to test it, of course, is to step on it, but that's not a good idea. So, that's—with that in mind, let's see how this analysis... this analysis will go, right! So, let's go to the data set first. The data set contains a bunch of independent variables, and these are sonar soundings at different frequencies, and the frequencies range increase slowly so you can expect that one frequency and the one imagery next to it are going to be highly correlated, right!

So, that would be an expectation here and the dependent variable is whether something is a rock or a mine. So, I guess, what... what we're saying is that we already have some data that tells us that for these frequency, for a certain rock, we send sonar sounds and we got a set of frequency, some set of data back, and that. So, we know for that data, there's a rock. And then, similarly, we know for some data, there's a mine. We already know this, right! So, this is our—sample is available to us and we've done this

over one field and now we want to expand this into other fields, so to speak, right! Remember, this is actually underwater but I'm going to be using the over land analogies, it's a lot simpler to understand. So...so that's our data. Typically, in a machine learning exercise that's called supervised learning. What happens is that you already have data that gives you both the values of the independent variables as well as that of dependent variable, and you want to train your algorithm or your machine to use that data to figure out whether, you know, in the future, when you get some new data arriving, whether that fits, in this case a rock or a mine, or whatever dependent variable you have.

So, this is an example of supervised learning, which is actually probably more common than anything else, but it is, that's where it is. So, the data we have is sitting in this URL over here, and you can just put the URL there and notice that 'read csv', pandas 'read csv' function over here, and all we need to do is, give it the URL and it'll get the data from the URL. So, csv data doesn't have to be sitting in your local computer but it can be anywhere on the internet. You give the URL and you get it back there. So, what do we do? Well, we import 'pandas', we import the 'dataFrame' but we don't have to, we can do 'pd' to 'dataFrame', and then we run this, and we get our data. So, this is what the data looks like, right! So, 'df' dot 'describe', remember, describes all the numerical columns in a data frame. So, we have a bunch of columns, so we can see the columns go from '0' all the way up to '60'—'59', right! So, there are '60' columns in the data set, and they contain, this is the data containing them. Okay, so this is the data that we have but these are all the numerical columns.

So, there's one additional column, which we will see in a second, which is the categorical variable, which tells us if something is a rock or a mine, that won't come up in describe. So, this tells us a lot of stuff. As you can see, we get the mean, the standard deviation, the minimum, and the quantiles—quartiles rather, of the data over here. So, we can...we can check that. And, just to show you a couple of things in pandas, you notice from '9' to '50', we have a dot dot dot over there. What this is telling us is that the data frame is too large to show on a screen. So, it's essentially trying to give us a sample of the data. Rather than showing all '60' columns, it's showing us a sample of the columns, '1' to '9', and then '50' to '59'. So, you can tell pandas that you want to see them all, and the way to do that is, to use what's called the 'options' command here, and the 'options' have many many things. You can look them up, you can display the maximum number of columns, the maximum of rows. We saw earlier that, when we had large data sets, that when you're viewing the data set, it puts dot dots in the middle. So, if you have like '10,000' lines, you'll see maybe '100', maybe '20' lines in the front and '20' lines in the back, but you can force it to show them all. So, what we're going to do is we say, show me all the columns. So, now when we look at this...this thing here, we get all the columns, okay! Not that it doesn't matters for this, it's just another thing that you should know.

Video 2 (9:44): Machine Learning - Understanding the Data

So, let's take a look at one column, because when we, whenever we look at data, we want to really understand what the, what's going on inside our data before we do anything right! We've already seen this when we did our data cleaning but in this case, the data is clean, but we still want to understand what is the nature of this data. So, let's examine the distribution of the data in 'column 4'. And if you look at 'column 4' here, you have the quantile data for 'column 4', or actually column 5, I guess, should

be column 5. Okay, '0.67', oh, here, 'column 4', I'm sorry, '0.0067'. So, this tells us that— let me push this here. So, the minimum is '0.0067', that's this number there, and the maximum is '0.4010'. So, our data ranges from this to this. All we want to see is, whether the quartiles are evenly distributed or not, because that, you know, tells us how well distributed the data is. So, we can see here that 'quartile 1' ranges from '.0067' to '.038', which is about '.03' in range. This is '.03805' to '.0625', '.0625' to '.100275'. So, these three are reasonably similar, and the only difference that we have here is that the last quartile actually is humongous. It ranges from .1 to .4. It has a point three range. So, it's much larger than other quartiles.

So, this, what this does is, it tells us that, "Hey! maybe we have some outliers in our data." And, outliers can mess things up, because when you have outliers, it's harder to predict, right! You're trying to get may be a line or predict line or predict something in the future, but an outlier can mess stuff up. So let's go ahead and look for outliers in the data, and see what's, what's happening with them. We can use a quantile plot, which— to help us identify outliers, and here let's take a look at the quantile plot. So, what this does is, it'll show us a draw line, which is the straight line from our...our data, and then it plots all the data points as we go from decreasing order to increasing order in our dataset. So, if they were uniformly distributed or evenly distributed, I should say, then there would be on or close to this red line that we see in the plot. But we can see that we have a bunch of outliers, a few on the bottom, not a lot, but there's a lot on the top, right! On the top there's a whole bunch of outliers.

So, that's not so great, but, you know, we got to live with it, because that's our life for us. So...so, that's the first step and when you look at outliers, you have to decide whether you want to, you know, keep them or take them out or whatever. Typically, taking out outliers is a process that is best avoided because the bottom line is you want to get as good a result as possible, and if the outliers are, you know, fat enough, that means the enough outliers, then you, when you test your model in the real world, you're going to have a higher probability of getting blown up, which is not so great. On the other hand, if you are, if you believe that the cost associated with not correctly dealing with the outliers is low, then you can throw them out because in that case, you'll get a much, a far more robust result in the part that is reasonably following the red line in our plot here. In this plot, if you can, you know, for a large chunk of the data that's on or close to the red line, if you can get a reasonable estimate, and if the cost associated with...with missing outliers or not using, not dealing with outliers probably, properly is not high, then, you know, you can just ignore it. So, for now, and say, let's chuck them out. For now, let's keep them in because we want to see what we get from all this stuff here. So, next thing you want to look at is what are the unique values of the dependent variable, because we saw that we have 0 to 59 independent variables, and there's the 60th column, and that's a dependent variable, and we find that's an array with 'R' and 'M'. 'R' is a rock, 'M' is a mine, okay!

So, that's the stuff there. So, we know that we have two dependent—two values for our dependent variables, so we've got, really looking forward, or looking at this stuff, we can see that we need to guess the categories of the data. As we get sonar soundings, we want to guess what category they're going to fall into, right! So, we have a categorical learning problem over here, and our categories are two. It's called a binomial classification problem. So now, let's look at the correlations between the dependent variables, and this is also kind of crucial because if your dependent variable are highly correlated, then adding lots of dependent variables is not going to help you that much, right! So, typically in regression,

you want to find relatively uncorrelated dependent variables, and, sorry, independent variables, and use them, rather than using a whole bunch of correlated ones. So, looking at this, we can just eyeball this and see a '0' and the '0' obviously is a '1', diagonal elements are all 1's, and '0' to '1' is '0.73', '0.57', '0.49', '0.34', and this makes intuitive sense to us because we know that the sonars— we're just increasing the sound, the frequency of the soundwaves slowly. So, the idea that '1' and '0' and '1' and '2' will give you pretty similar results is not, soundings is not, you know, it's very reasonable.

So, the correlation is going to be high and then it'll keep decreasing as we go by. If you look through this stuff here, we find that it keeps decreasing, if you look at the row '1', and this...this one, right, keeps decreasing, keeps decreasing, decreasing, and actually becomes negative around here, right! So, in the middle range, it's negative, and then as you go up there, it starts increasing again. So, oddly enough, we start getting higher correlations as our frequencies get really divergent, okay, which is kind of interesting but that's something to note here. So, there are various techniques in linear regression, in regression to find the set of uncorrelated variables that will explain your result more than, more. So, you want to try to look up those things if you want to, but we are going to, for our exercise, we're going to just use them all. So, we can, we have these correlations, it's kind of hard to read this whole thing. So what we'll do is, we'll draw a little color plot that shows our correlation, it's kind of a neat way of looking at correlations. So, we give it our correlation matrix, that's the dataframe really here, the dataframe of correlations that we compute, this one. And...and use the...the 'matplotlib' 'pyplot' function call 'plot' dot 'pcolor'. So, what pcolor is going to do is it plots these in color. Now, this is not very exciting looking but I can explain this to you here. So, what we get along the, oops, work along the diagonal over there the yellow one, it's very high correlations. As we go into the lighter yellow and then into green, the correlations are dropping.

As we go into the blues, the correlations are becoming negative, and then we go back into more positive territory, ending up with positive correlations again. So, if you look, for example, at the zeroth column, right, the very first column, '0', and watch it as it goes along. So, we see that '0' is initially, the correlation between '0' and 1, and maybe 2 is high, and then it starts dropping off, and then it goes on until we get to about 21 or 22, where it turns blue, right? So, that becomes negative, and that goes on till about 30 something, and then it starts going positive again, and finally at about '60', it's reasonably positive. Nowhere near, of course, the high correlations we saw earlier, but the correlation is out there. So, this tells us something about our data. That the correlations are, you know, tightly correlated across the various variables, but there's a lot of room to play in the mid area where there's negative correlations and low correlations in the...in the middle area. So, we can...we can use that to our advantage, but, for now, we'll just stick with this. So, let's look at the correlations of '1' with this, and this will really, intuitively, will show us what our, what we've been talking about so far. What this shows us is that the correlation of 0 to 0 is, you know, I had one, then 0 to 0 is point 9 7 8 9 something like that, and then it drops rapidly, right! As you go to 2, 3, 4, 5, it drops rapidly down.

So, probably around 5, it's already dropped all the way to '0.2'. And then, it keeps dropping, it's sort of juggles around a little bit there, it keeps dropping into negative territory, around 25, and then it rises, rises, rises, ending at about '0.2'. So clearly, our situation with... with this, with one anyway, and for most of them this is the case, that the—for the two or three columns that are around a column, plus, minus 2 or 3 columns, the correlations are very high, but then it sort of tapers off. So, in general, high—

highly correlated items are not good, low correlated items are good, and you want to actually try to pull out the ones that are highly correlated if you can, right! But, in our case, we're going to have a hard time doing that because...because of the way the data is, we, if we pull out one, we still have, you know, everything is correlated to everything nearby. So, we can't put all the nearby columns of everything out. So, we might have to do like one and five and 15 and 20, and maybe it's not a bad idea to try that. So, and, of course, if the correlation is high with a dependent variable, then that's...that's good, you know, that's something that you want to look at. So, our dependent variable here is categorical. So, we don't really have correlations. So, we skip that step.

Video 3 (7:04): Machine Learning - Wines Dataset

The second data set that we look at is the wine data. So, let's take a look at that as well. So, wine data has a bunch of 'independent variables' and these variables are the composition of wine. And the composition is really the chemical composition. What's the alcohol? What kind of sulphites? What kind of acidity it has? You know, that kind of stuff. So, it tells you, if you took your wine sample and then analyze it chemically what you would see. And, the dependent variable is interesting. It takes a panel of three wine tasters and they taste the wine. And they rate the wine quality between 0 and I think it's 10 but not 100 percent sure. So, 0 and say, let's say 10, 0 and something. So, they rate the wine and then we take the average of the three ratings, and we get a rating for the wine.
So, the goal here is to see if we can take a wine sample and chemically analyze it and then predict whether its quality is going to be rated high or low, by these three wine experts, anyway. But in general assuming that these three wine experts are reflective of the entire population of wine experts. So, let's get that data. That data's at this URL over here, okay! So, you can get that in and again we do a read CSV. We get that. We had a zero, and the separator in this data is not comma, which is a default for read CSV but it's a semicolon. So, we give this additional argument over here. Let's say separator equals semicolon, right! So, that's what read, it says read CSV, it can actually read any file structure as long as you know what the, what separates the data columns in that, data values in that file. So, let's take a look at this. So, we get a WDF and we look at this here. And we see, we have all these independent variable columns 'fixed acidity', 'volatile acidity', whatever. And it tells us, we have '1599' data points here, quite a bit. And the last column is the quality. And this is the average rating of the, the 3, 3, 5, 6, 8, 6. These are the quantiles. The average is 5.63. And the standard deviation is 0.807, okay! So, that's our variable over there.
So, that's...that's a different example. The first example had a, our rocks and mines data, had categorical variables and the wine data has continuous variables. So, let's take a look at the 'volatile acidity' column. Run that and this is the values we get over here. So, they're all nicely, this thing. And, we have lots of data here where we can take a look at the correlations, and do our plot. So, it's looking better and this tells us that the correlations from '0' to '0' is like one, which is big surprise but nothing else is really that tightly correlated, right! We have some, little bit correlations like between '6' and '5'. There's a little bit of correlation. but there is, otherwise the correlations are pretty mixed. So, this is relatively

uncorrelated data. And so that's good for us, right! We can also look at the correlation of one with something else.

So, we look at the correlation of one with each of the others. That is 'fixed acidity' with each of the others. So, that tells us that there is some correlation here like 0.5 something with 'citric acid' and 0.6 something with, I guess, 'sulfur dioxide'. but it's still all over the place. We can use a scatter matrix, a scatter matrix to visualize the correlation between features but though this is kind of not too much data. So, we can actually show this here quickly but scatter...scatter matrix is a fairly 'processor intensive' function. And, but it's interesting to see that. So, what we're going to do is, we are using the scatter matrix function here and we're giving it our alpha, if you know alpha is a density thing. So, the figure and we're giving it stuff here. So, we get this. So, the diagonal, this is a scatter matrix and what it tell us is that the, it looks as the... as the scatter plots of each pair of independent variables and of each variables in this case. So, we see, for example, let's take sulphates, right! So, because that's clearer and alcohol. So, we've got sulphates and alcohol and we look at it. And, we see it's relatively uncorrelated because the blobs are no distinct pattern. If we look on the other hand at alcohol and— I can't read these things very well. Let me see if I can find one that looks clear enough. Let's say the top, the third element from the top, on the top row. That is the...that is the top row is fixed acidity, and the third element, citric acid. That's the third element from the... from the left on the bottom row. Then we see that that's, you know, the scatter plot looks more increasing, as one increases, the other increases.

They're still pretty chunky. They're not that tightly correlated. Some of them are a lot more tightly correlated as we'll see but this one not that chunky, as we can see if we look at these graphs. So, scatter plots give us a good sense for how tightly correlated two variables are. The middle, the diagonal columns over here are the distribution of the variables themselves. So, we can see that, you know, for example, the very first top-left corner, we can see it has a fat tail on the right-hand side, on the higher side. And in fact, looking through these things, there are lots of fat tails and some of them are really well poorly behaved. Like citric acid is poorly behaved, right! Citric acid is the third from the top and the third from the left. So, that's poorly behaved. Something that is like this one over here, which is couple of them over here which are pH, and the one before that are reasonably well-behaved, right! They look kind of mostly normal. So, and if you look at the dependent variable wine quality, we find a problem because it has really got two so to speak midpoints, right! So, two...two highpoints in this curve here. So, it's not at all well-behaved at all. So, we're going to have a problem with that. We know that upfront, right! So, this probably won't work very well with regression. So, we won't use regression, we'll use something else for this. So, we've done that. We can also do quantile plots just like we did with the rocks and mines data. And, that again shows us some outliers over there, and, so that's really the first step. The first step is to take all your data and examine the distributions.

Examine the correlation between variables, look for outliers, and try to understand what the data looks like. What is the distribution of the data and you know whether you're going to get results that will make sense or not. So, looking at this, for example, the quality, the...the shape of the quality data. We can see that regression isn't going to work very well with this because it's not going to find a single, nice line that we're going to fit this thing here, right! So, you want to see what kind of technique will work better with it as well. So, that's our data segment. We're going to move on to actually doing some regression with this thing here.

Video 4 (7:05): Setting Up Regression

So, now let's see if we can train a classifier on rocks versus mines. So, to do this, we're going to use a package in Python called 'sklearn', or sci-kit-learn, sci dash...sci dash kit dot learn, or something like that, anyway. We've got this here. So, we get sklearn. We import data sets, and we import the linear model. We're going to use the linear model here. And we will import—this I'll do again anyway, but these are the metrics we're going to focus on, the roc_curve, and the auc and I'll explain that as we go along and we can work with this. Yes, it should be imported before I forget. And, we also import NumPy and random because they're useful for building our training and testing samples. Then we read the data, of course, so we've got that. We've already done that before. And now, the first thing we want to do is we want to take our dependent variable, which are currently two characters, 'R' and 'M', and convert them into '0' and '1' because when you're doing a regression, our dependent variable has to be numerical. So, we convert that into '0', '1', and we're going to use our 'np' dot 'where', if you recall this one. This is like the Excel F. And what it says is wherever 'df[60]', which is the dependent variable column, equals 'R', you pop a '0', and otherwise, you put a '1'. And we know that from our earlier examination that unique values are only 'R' and 'M'. So, we won't have any NAs or anything else to deal with. So, we take our 'df[60]' and we convert it from 'R' and 'M' into '0' and '1'. The next thing we need to do is to divide the data set into training and test samples. Obviously we can take our entire data set and run it, and, you know, get the result, and then hope for the best when something new comes in. But that's not going to be really a great idea because what we want to do is, we want to see whether the results that we get are robust or not. If they're not robust, then we don't want to use them when we send our troops or people into the minefield. To test whether it's robust or not, typically what you do is, you take your data set and divide it into two parts, a training and a testing sample. You train the machine on the training sample, and then test it on the testing sample to see whether the results you got from the training sample actually make sense with some data that you haven't used in training, in...in trying to parameterize or train your model.

So, that's...that's really an important step. If you don't do that, you know, you get into trouble. You have to understand, you have to make sure that your results are going to be reasonably robust, otherwise, you're going to be in trouble. So, training and testing and in fact, often people use a third, holdout sample, completely unseen sample. And then after they have trained the model, training and testing, and they have tested it out and looked at different ways of training it, then they finally take a third sample that...that hasn't been used at all in the process and see whether the results hold over there. For example, you might train your model using one classification technique, and then try another classification technique and try a third classification technique and then you pick the one that looks the best. But at this point, you've only trained on one set of data, tested on another set of data, and then you used that to pick a model.

So, now you want to see whether that process of training and testing has sort of biased your results to whatever data was in the training and testing samples. So, now you pick a third one and test it out there, and hopefully it will work out well. We'll...we'll use only a training and testing sample and 'sklearn' has

a...a...a model selection package, which contains a function called 'train_test_split'. What that does is, you give it the fraction of data that you want in your test sample. I think test size equals '0.3' over here. You give it a dataframe and it's going to split the thing into two dataframes, a training and testing one, and then we've got that. So, once we have a training and testing, we're going to pull out from our dataframe, the independent variables. That's column 0, sorry, row '0' onwards to '0:60', which is from the training. We have two things here, training and testing. Let me run this and we can see this for a second, okay! 'Insert', cell above. So, here if I look at train, we see it's a dataframe, and it has row '6', row '193', row '179', row '139', row '94', row '24' in it. So, it's randomly picked from a certain set of rules from our dataframe, and that goes in the training and the other rows have gone into testing, the ones that are not in...in this thing here. So, that's our 'x_train' and from the training on this dataframe, that is our training dataframe, the one that we just saw here, we pick the columns '0' through '60', that is '0' through 59, remember the last one that I have included. So, these are independent variable columns, and they go into the 'x_train', which is our data, our training data set, the independent variables in our training data set. And the 60th column, that is actually 61st, but that goes into 'y_train' which is our dependent variable, and similarly we do the same for 'x_test' and 'y_test' for the testing sample. And notice, we use the 'iloc' here because we're using row indices to pick data and we use row indices to use 'iloc'. If you use the row numbers, zero, one, two, three, four, then you use 'iloc'. If you use the actual index, we don't have an index here but if you add an index, and you use an actual index, you could use dot loc instead.

So, this gives us our thing here and we see that y training contains '6', '193', '179', which pretty much corresponds to the '6', '193', '179', '139', '94'. So, they...they are aligned, you know. We want aligned data in our training and testing samples. So, once we have that, then we build our model and fit the training data and this is the standard procedure regardless of which particular model you're using for machine learning from the sklearn package, you're going to use the same two steps. You're going to first start by building a model itself. So...so, you...you get this and we're going to use linear regression over here. I should point out that for our data. Logistic regression is another option, which is probably slightly better but because I want to explain how to, you know, the various methods for evaluating results, we'll use linear regression here. To use logistic regression, you just replace linear regression by logistic regression, and then, you know, most of what we are do is, probably going to work equally well with that. And, but anyway, so we have here, we're going from the linear models in our, in sklearn, we're going to be picking the linear regression package and recreate this model here. And then what we want to do is, we want to fit the model to our training data. And to fit the model, you tell it what your training sample, independent variable values are, and what the corresponding dependent variables are, and that will fit the model for us.

So, this is pretty much what we will do regardless of which particular model we happen to be using. Once we've done that, we fit it in our model, and now we can start thinking about how to interpret the results.

Video 5 (8.33): Introduction to Optimization - Example1

Hello, everyone. We'll learn about optimization models. This is an important pillar in this course and in fact, in the field of analytics, where this has wide applicability in almost any real world problem that you might think of, where typically we are trying to find the best possible use or most efficient use of limited resources. So this field of optimization provides a formal framework of modeling a decision problem, a real-world decision problem, as a mathematical model and developing algorithms to find the best possible decisions or solutions efficiently.

So in this formal framework, we use variables to represent decisions and constraints to model problem dynamics, and encode the problem-specific constraints that we have and the performance objective that we want to optimize is the objective. Our goal in this course would be to understand various optimization formulations with a strong emphasis on modeling and using off-the-shelf commercially available solvers like R and Gurobi to solve those optimization formulations. These solvers are very powerful and can solve very large-scale problems quite efficiently, which makes them very widely applicable in today's real-world problems at the scale of problems that we need to solve today.

Let me begin by describing the common optimization frameworks. So, linear optimization is a very powerful framework and very tractable and widely applicable. So in this framework, both constraints and objectives are linear functions of the decision variables. Another very powerful model is integer optimization, where in addition to constraints and objective being linear functions of the decision variables, we can have some of the decision variables to be constrained to take integer values. This is a very powerful model and allows us to model a large class of applications that just linear optimization is not able to model. But together these two models, linear optimization and integer optimization, these are very widely applicable and we have solvers that can solve extremely large-scale problems.

Solvers like Gurobi can handle, for linear optimization, problems with millions of variables; in integer optimization, problems with tens of thousands of variables in a very reasonable time. So, you can imagine that these are very useful frameworks which are widely applicable, made applicable by presence of these very powerful solvers. Another framework is convex optimization, where in addition to allowing linear objective and constraints, we allow for convex objectives and constraints and this is very powerful. It has applications in machine learning, for instance, logistic regression is a convex optimization problem.

So this is another very powerful framework. And more generally, we can have nonlinear optimization framework where we allow for general nonlinear constraints and objectives. We'll also focus on a very important framework, that of optimization under uncertainty. As you may know, at the planning phase or optimization phase, many of the problem parameters are not known. So we need to handle uncertainty in problem parameters while solving the decision problem and there are frameworks like stochastic optimization and robust optimization, where something is known about the uncertainty. Or more generally, online optimization where we have very little information about these uncertain parameters.

So we'll touch on these topics in the course as well, but let me start with an example on linear optimization, that's the simplest of the frameworks that I discussed above, and in particular, let me start with an example about transportation problem. So, this is a toy example where we have two factories

where the supplies of 100 and 200 units of some product and we have three shops with demands of 50, 100, and 150, respectively, and we want to ship these units from factories to shops, spending the least possible on the shipping costs. Okay? So, this table here describes the shipping cost.
For instance, shipping from Factory 1 to Shop 1, the cost is $10 per unit. And this table describes the shipping costs between different factories and different shops. Our goal is to find a shipping plan that ships the units from factories to demand, satisfies the demand of all the shops, and minimizes the total shipping cost. So we can represent this problem as a network. With these two nodes representing the factories that want to ship the units to the shops which are nodes on the right side. Okay? Now, with their respective demands. Now the first thing to formulate this problem as a linear optimization problem is, decide on what are the variables. So here I can have a variable, which is xij, which tells me the number of units that I want to ship from factory 'i' to shop 'j'. And here I'm denoting it as a number, but I am thinking xij can be fractional. And the objective is to minimize the shipping cost, which I can actually write as a function of these decision variables.
So for instance, x11 is the number of units shipped from Factory 1 to Shop 1, and the total shipping cost for this would be 10x11, okay? Similarly, I can find the shipping cost between every other factory and shop and summing up all the costs gives me the total shipping cost. Now what about the constraints?
So, we have three sets of constraints. The first constraint is on the supply side. So, let's look at Factory 1. So, this is the constraint for Factory 1. It says that the total units that are shipped out of Factory 1 cannot exceed 100, okay? Similarly, we have a constraint for Factory 2 where the total units out of Factory 2 cannot exceed 200. The second set of constraints correspond to the constraints on the demand nodes which says that for each demand node, let's say Shop 1, the number of units shipped to Shop 1 must satisfy its demand.
So, the number of units that are shipped to Shop 1 are from Factory 1 or Factory 2. This is 'x11' and this is 'x21'. So the sum of these two units must be at least 50, and a similar constraint for Shop 2 and Shop 3. And lastly, we have non-negativity constraints on all the units. Each xij must be non-negative. Putting it all together, we have the following linear program where we are minimizing a linear function of the decision variables. The constraints are linear functions of the decision variables and we have non-negativity constraints on the variables.

Video 6 (5.43): Example 2 - Cash Flow

Let us consider another example for formulating a decision problem as a linear optimization problem. So in this example, we want to maximize our cash flows, okay? In particular, we are given $100 to invest and we have three different investment options. Let me describe one of the options, for instance.
In option A, for every dollar invested now, we will get $0.10 a year from now and $1.30 three years from now. Similarly, we have options B and C with their respective return schedule. In addition, we also have a money market account where we can just put our cash in that account and we get a 2% return per year. The goal is to decide on how much to invest in which options such that our cash at the end of year 3 is maximized. So to formulate that problem as a linear optimization problem, our first task is to decide on the decision variables.

So the most natural decision variables are xa, xb, xc, and xd, which are dollars invested in each of the four options, xd being the money market option. Now in addition, we will need auxiliary variables y1, y2, y3, that represent cash at the end of year 1, 2, and 3, respectively. So once we have these auxiliary variables, in particular we have auxiliary variable y3, which is cash at end of year 3, our objective formulation is trivial. We want to maximize y3.

Let us now model the constraints on these seven decision variables and it's easy to model these constraints once we represent these investment options using a cash flow diagram, okay? So let me draw, so you have these three years. So option A says, if I put in $1.00 now, I get $0.10 here and $1.30 here. Option B says if I put in $1.00 now, I get $0.20 here and $1.10 here. So, I've drawn this $1.10 to the left of 3. That indicates this is end of year 2 or beginning of year 3.

Now option C only becomes available to me here. If I invest $1.00 now, I will get $1.50 at end of year 3, okay? So this represents the cash flows of various options. So now my decision variables are xa, xb, xc, and xd. So the cash at the end of year 1, y1, is going to be 0.1 xa plus 0.2 xb plus 1.02 xd. That's the money market return on xd, okay? So now, in year 2, or beginning of year 2, I have option of investing in option C. Okay?

So we have xc, which can be at most y1, because that is the cash at the end of year 1; I cannot invest more than the cash I have available at the end of year 1. So now with these decision variables, I can write y2 as, I get 1.1 xb plus y1 minus xc. This is the cash I put in money market, times 1.02 and therefore I get y3 is 1.02 y2 because that's the cash at the end of year 2, again in money market and I get a 2% return plus 1.3 xa plus 1.5 xc.

So these are the constraints that model these problem dynamics; and in addition, I also have a constraint where xa plus xb plus xd can be at most 100. So this is the complete set of constraints for this problem. And I can write this as the following linear program where I'm maximizing y3, a linear function subject to linear constraints on the decision variables.

Video 7 (5.24): Introduction to Linear Optimization

So, let us consider the formal framework of linear optimizations. So suppose we have decision variables 'x1 to xn'. Objective function is a linear function in these decision variables and we can represent a linear function as 'c0 plus summation j 1 to n, cj xj'. Okay?

The constraints are of the form that a linear function is greater than or equal to 0. So, there can be constraints 'i' going from '1' to 'm'. So we have 'm' constraints where each of 'gi of x' is a linear function of the decision variables. And I can represent it as for instance 'ai0 plus summation j going from 1 to n aij xj greater or equal to 0'.

So if you want, we just model non-negativity of decision variables, you will have 'gi of x is xj greater than or equal to 0', that constraint will model non-negativity of decision variable 'xj'. Similarly if you want to model equalities through inequality, I can model 'gi of x is greater than or equal to 0' and 'negative gi of x is greater or equal to 0'. That would model 'gi of x is equal to 0'. So a general linear programming problem can be formulated in this framework, where both the objective and the constraints are linear functions of the decision variable.

So just to emphasize, I have some examples here where the functions are non-linear functions of decision variables. For instance, in the first example here, we have a quadratic term. So this first function is a quadratic function of the decision variable. The second and third examples similarly are nonlinear functions of the decision variables.

So in linear optimization, both constraints and objectives must be linear functions of the decision variables. So now that we know what a linear optimization formulation is, the first question you would ask, how to solve such linear optimization problems. So, to build in tuition towards that, let me consider the special case where we only have two decision variables. So in particular, let's consider this example where we have decision variables 'x1' and 'x2'. And we have three inequality constraints and non-negativity constraints on the decision variables.

Now, this problem with only two variables is very special because we can easily plot the feasible region and that's what we'll do. And that gives some very important insights towards the structure of optimal solutions in linear optimization. So let's first look at the easier set of constraints, 'x1' and 'x2' non-negative. This is the unbounded shaded region that I have on the figure. Now, let's add this constraint 'x1 plus x2 less than or equal to 80', okay? So the way we add this constraint is, we draw the following line: 'x1 plus x2 equals 80'.

Now this line divides the whole space 'R2' into two halves. One of the halves satisfies the constraint and the other half does not. And in this case, this direction satisfies the constraint. And therefore the resulting feasible region is the region shaded in yellow. Now similarly, we add the next constraint: '2x1 plus x2 less than or equal to 100', and figure out the resulting feasible region, which is the region shaded in yellow. And finally we can add this constraint: 'x1 less than equal to 40' to get the resulting feasible region.

So this shaded region, any point in this region, satisfies the constraints for the problem and we want to find the one that has the highest objective value. For this, we'll use the fundamental theorem in linear programming which says the following: that if the LP has a finite optimal value, and the feasible region has at least one corner point, then one of the corner points must be optimal. Or in particular, there exists a corner point optimal.

So that's very powerful because now what we can do is, we have this feasible region which is shaded in yellow but there are only five corner points. So I can go and check the objective value at each of these corner points and pick the one which has the highest objective value. So for instance in this case, we get this point which has the objective value of 180.

Video 8 (6.03): Example 1 - Linear Regression

Let us see some examples for linear formulations. So the first example that I want to discuss is linear regression, which we have seen earlier in the class where we're given a bunch of data points: x1</sub>,y1; x2,y2; xn, yn, and we want to find the best linear relationship between 'y' and 'x'. In particular, I want to find best 'a' and 'b', best line that explains the data 'xi, yi'. So I want to fit 'yi' as a linear function of 'x'. And to do that, we saw earlier that we can solve a least square problem, which is minimizing the sum of squared errors.

So we minimize over 'a' and 'b', sum of squared error, where error in the 'i'th data point is going to be a predictive value: a transpose xi plus b minus the true value square of that is the squared error, and I minimize the sum of squared error. Now, this is not a linear formulation because the objective is a quadratic function of the decision variables. So we can consider alternate formulations to find a linear fit that explains the data. So for instance, here I want to find the best linear model that explains the data such that the maximum error is minimized, and here error is given by the predicted value minus the true value. This is the error and I want to minimize the worst possible absolute value of the error, okay? Now as such this is not a linear optimization formulation, because first of all, we have a min-max problem and the objective also contains absolute values, which is not a linear function.

But we can find a linear formulation by introducing additional variables. So let's first linearize this part, which is the maximum of the absolute value of the errors, and then once I've linearized that, I'll just minimize a linear function, okay? So let me introduce a variable 'ei', which sort of tracks the error in the ith data point. So I will say 'ei is greater than a transpose xi plus b', which is the prediction, 'minus yi'. But this value can be a negative number. So I'll also add the following constraint: 'ei is greater than equal to negative of a transpose xi plus b minus yi'. So these two constraints guarantee that 'ei is going to be greater than or equal to absolute value of a transpose xi plus b minus yi', okay? And therefore I have a linear formulation for this absolute value function.

Now, I want to minimize the worst possible error. So let me introduce another auxiliary variable 'e'. And I'll say that 'e is going to be greater than or equal to ei for all i: 1 to n'. This is the formulation that we get, which is a linear formulation for this regression problem. So two things that we should learn from this example is linearizing this absolute value constraint. So, we can linearize such a constraint by the following two constraints, and linearizing max, and that can be done using the following constraint. So we were able to linearize both max and absolute value to get this linear formulation. So in a similar spirit, suppose we consider the regression problem where we want to minimize the sum of absolute value of the errors.

So, in a spirit similar to the previous reformulation, we can introduce additional variables here and formulate this as a linear optimization problem. In particular, we can introduce additional variables 'ei' as before and add the following two constraints and just write the objective as 'minimize sum of eis'. Now you may wonder that these two constraints only guarantee that 'ei' is greater that 'ei' is greater than absolute value of 'a transpose xi plus b minus yi'. But we wanted to model 'ei' as exactly equal to this absolute value. But note that since this is a minimization problem, 'eis' will be set to equal to absolute value at optimality and therefore this is an exact reformulation.