[SQUEAKING]

[RUSTLING]

[CLICKING]

MICHAEL SCOTT ASATO CUTHBERT: OK. Let's get started right on time, both to set a good precedent, and there's quite a bit to do today. Oh, I will put that back down for one second so that those who just got in can see the agenda for today.

First two things will just be like a minute or two, and then we'll spend the rest of the time on what everybody thinks is the coolest thing in the world these days in computation, artificial intelligence and machine learning. But this is not a class in that. We'll spend most of our time on feature extraction, the necessary evil before doing great things with artificial intelligence and machine learning when you're dealing with data that's not already numbers.

I want to do a really quick discussion on something on problem set 7/8. I'm still listening to the compositions. Some of them are really, really cool. Just, they're all great, that I've heard so far. But some really went into it. But there was one thing that, among people who attempted every problem, ended up being a consistent error. Actually, two things.

One that a lot of people missed here was in-- let's say you were in C major, because that'll make life easier. I gave some chords that were like this. And what do we call this in C major? What's the sca-- in Roman numeral. We're doing the Roman numeral exercise for a second. What scale degree?

**STUDENT:** A ii.

MICHAEL SCOTT ASATO CUTHBERT: ii? Good. And what is the quality of the chord?

**STUDENT:** Diminished.

MICHAEL SCOTT ASATO CUTHBERT: Diminished, great. So you don't have your problem set back for two reasons. One, there are a couple more songs I'm listening to. And also-- so how would we write ii diminished?

**STUDENT:** Circle.

MICHAEL SCOTT ASATO CUTHBERT: With a circle, yeah. And then I noticed that my thing said, or if it's vii diminished, write this. And I did not say, in general, if it's diminished, write this.

So I think that some people seem to write specific code for making sure that it only happened on vii. But ii diminished happens quite often. How does it happen in C major? You're borrowing from C minor for a little bit.

So these borrowed chords, either flat and natural. And then you end up with that D-F-A flat chord. So that's where that comes in, but I want to make sure. I'm going to go back because there was an ambiguity in the instructions, and I want to make sure that everybody gets the little point back who made that, who didn't put that in.

So that was one of the two systematic issues, and that came from an ambiguity for me. Here is the other one that came up a lot that I want to address. Oh, yeah. So we just went from C major.

[PLAYS PIANO]

And we played--

[PLAYS PIANO]

--that ii diminished sound. Then what about this sound?

[PLAYS PIANO]

Anyone identify that chord by ear? Or just the-- what's that?

**STUDENT:** Augmented.

MICHAEL SCOTT ASATO CUTHBERT: Augmented, great.

[PLAYS PIANO]

Good. I was in, doing that borrowing from minor thing. And I left my blue pen, which probably shows up better. I was borrowing from minor and I started on the third scale degree. And so this was not part of your Roman numeral thing, but it came on the roots question. What is the root of that chord?

**STUDENT:** It could be any of them.

MICHAEL SCOTT ASATO CUTHBERT: No, it cannot. Ah, we'll get to that. That's where something happens. Yep, exactly. It's a great, great idea. John?

**STUDENT:** E flat.

MICHAEL SCOTT ASATO CUTHBERT: E flat. Now, we'll get to it in just a second. What is the root of that chord?

**STUDENT:** G.

MICHAEL SCOTT ASATO CUTHBERT: G. And let's see if I can make one. We'll go back to our E flat. What is the root of that one?

**STUDENT:** C flat.

MICHAEL SCOTT ASATO CUTHBERT: C flat. Now, why is it? And so, just so we all have it in our ears again--

[PLAYS PIANO]

Three chords, three exact same sound. But roots, the concept of root is something that comes from a triadic, sort of a spelling-enabled thing. So where do we find the root? We talked about the algorithm last time. What do we want to find? A bunch of stacked--

**STUDENT:** Thirds.

MICHAEL SCOTT ASATO CUTHBERT: Thirds. Good, I heard it. So stacked thirds. So here, we have a third, followed by a third, or a third and a fifth, depending on how you're counting. But here, what's this interval?

**STUDENT:** Diminished.

**STUDENT:** Diminished iv.

**STUDENT:** Diminished iv.

MICHAEL SCOTT ASATO CUTHBERT: Diminished iv. Yeah, a iv of some sort. So we have to move it up the octave in order to get our stacked iii's. And here, we can move them both up an octave in order to get our stacked iii's. So the diminished--

The augmented triad is one of the places where the concept of root has everything to do with spelling. There's one other very common chord where the concept of root has everything to do with spelling. You saw a big smile, and it's not a triad. Go ahead.

**STUDENT:** Diminished vii?

MICHAEL SCOTT ASATO CUTHBERT: Diminished vii chord, yeah. Here, I'll just keep it in the light. Assuming that's good. So yeah, diminished vii chord. Let's say we're in C minor. We're going to start on B. Would somebody shout out the next note?

**STUDENT:** D.

MICHAEL SCOTT ASATO CUTHBERT: D. Keep going.

**STUDENT:** F.

MICHAEL SCOTT ASATO CUTHBERT: F. And then?

**STUDENT:** A flat.

MICHAEL SCOTT ASATO CUTHBERT: A flat, yeah. So vii, then we have vii with a root on B. But we can change this to move it up an octave and flip the enharmonic. And suddenly, we have iii, iii, iii, and our root becomes D.

We've got a lot more to get through today, so I will leave it to a test to prove to yourself that we can keep doing this somehow, and get all, every one of the four notes, as a diminished-- as having its own, that note as a root. So each of the chord factors, depending on how you spell. And that's one of the reasons, by the way, this gets away from the computational side, but maybe a cool computational compositional tool.

One of the reasons why a diminished vii chord is often a very useful way to modulate to remote keys, because you just sort of start emphasizing a different note as your root. Any questions on this, or on other things? We're all eager to get to feature extraction? On this, there should be a notebook that says, under today, 2023-ai-template-blank. That one. Is it?

**STUDENT:** Yep, it's there.

MICHAEL SCOTT ASATO CUTHBERT: It's there. Good, good, good. So feel free to download that and put that where you are, and run it. It will install. If your hard drive is almost completely full, it will install a learning toolkit called Orange, which also installs another machine learning toolkit called scikit-learn, and I think they might be a little bit big. Might be-- whatever.

So make sure to run all first, and that will get you in a pretty decent shape. I'm going to get started on-- what we're talking about today is something called feature extraction. And so feature extraction is a conversion of elements, in this case, musical elements, into single numbers or lists, arrays, vectors, tensors, whatever you want to call them, of numbers.

It is the most boring thing I will teach in the class. And please don't correct me if you're like, Cuthbert, there's plenty of other boring things you've done. But no, it's really one of the most boring things in the world. And here's a reason to hate it.

It really sucks the lifeblood out of music. Everything that you love about music is going to be converted into a few numbers. It is-- sucks the life out. It is the zombie of music world. Adam's very excited. Yes, zombie invasion!

But there's a lot of amazing things, we know, happening in artificial intelligence, machine learning, deep learning, algorithms, all these things. And as far as I know, they all have one thing in common. And that is, they work on collections, big collections of numbers.

There's some ways that you can work with images that, effectively, are kind of like two- or multi-dimensional arrays of numbers. But basically, they're all going to be numbers. So when I put some music notation on the board, we're not going to be able to work with that, even an image of it, even a JPEG, unless we convert it to pixels and convert it to numbers. When you hear a sound, [VOCALIZING] something like that, you're going to have to convert that into numbers in some ways.

So we're going to be just working on converting things to numbers. So it's pretty simple, and-- but how to do it, and what to extract, and how to make your extractions, is the deep art that takes a while. Now, I'll get to a little bit later some of the things that people are doing today, where feature extraction is less-- or careful feature extraction is less important.

And I'll tell you what those things tend to have in common, and that is, they work on much bigger data sets than what we have. So we're going to get started on this. Yep, that's my template here. And we're going to start with, why not? We always start with our favorite Bach chorale. What's the number?

**STUDENT:** 66.6.

MICHAEL SCOTT ASATO CUTHBERT: 66.6. Conjure up the bwv66.6. I'm going to show it, get it back. OK, great.

So we're going to do some manual feature extraction first. We're going to convert this into numbers. I'm going to extract two things from this, number of parts and initial number of beats per measure. So let's just do our--

You don't have to type this if you don't want. numParts equals 4. Time signature. Numerator equals 4. There, I have done manual feature extraction, and describing the piece as two numbers. They seem to look like 4-4, but I didn't ever just extract the denominator. That just happens to be coincidence they're both 4.

So I don't know if we'd even call this feature extraction. But you can see, it's pretty-- we were not talking about very, very much here. So I'm going to define an automatic feature extractor. So we'll call it extract_meter.

And extract_meter, you can save time by not putting these annotations. It takes in some sort of a stream. I'm calling it score, even though it's not. And it returns two integers. What do you think the two integers are going to be, if you're extracting the meter?

**STUDENT:** Numerator.

MICHAEL SCOTT ASATO CUTHBERT: Numerator and denominator, great. Super. So we can say, all the meters in the piece equals the score meter, TimeSignature, recurse. And the first one, we'll just get the first one. first_meter, all_meters 0. And we'll return first_meter's numerator and first_meter's denominator. I know I don't need the parentheses, but I just kind of got used to that.

So I'll let people get caught up. Now, by the way, how many time signatures does this little chorale have? Are we going overkill by getting the first one? Do you want to take a guess? You all know it by now. 1?

It actually has 4, the way music 21 counts. 1, 2, 3, 4. They're all the same. But this is an error I make quite a bit often, that each of the parts has its own time signature. So this is something to be careful on. And now, we'll get to the first sort of ethical dilemma of feature extraction and machine learning, of which there will be many.

And quite a number today. What happens if a score doesn't have any time signatures? Do we want this? Will this crash, first off? And if you think it will crash, tell me what the error that's probably going to be returned.

**STUDENT:** List index out of range.

MICHAEL SCOTT ASATO CUTHBERT: List index out of range, or index error. Yep, that we're trying to get 0. So we can say something like, if not all_meters. Now, so here's the ethical dilemma. I'm going to return 0 comma 0 as a special value.

That's also an integer, tuple of two integers, that says, I have no time signature because I've raised as a medieval musicologist, so I have Gregorian chant lying around, and other meterless time signature pieces. But the other reason is that we sometimes, in case of an error or an out-of-bound value, we return some sort of, we call it the sentinel, some kind of special meaning number that says some sort of an error happened.

Why? Because you're going to someday have a gigantic data set. Maybe you're going to be doing this up in the cloud somewhere or something, and you're going to have-- you're going to be letting it run overnight, consuming a huge amount of processor money. And eventually, there's going to be some bad data in there that's going to crash your system.

And you want to be like, OK, I just let this run for 47 hours, and now I have to start all over again. Because, of course, I didn't save intermediary values, because I'm a theoretical person, not a practical programmer. But always save. Write to disk every once in a while.

But so this type of thing helps to save you from a little bit of bad data. What it also does, though, is it can swallow up systematic errors in your thought. You know what I'm saying? That you didn't realize that-- the one that I always get is, I'm looking at the pitch of everything in the piece, pitch of everything in there.

And then I forget that these structures called chords exist, which don't just have a pitch, they have multiple. And so I could end up silently throwing out all of my data about chords, rather than realizing I have a mental model problem in there. And this is some of the things that can lead to AI that doesn't include all of us, that programmers forget that women exist or that non-white people exist. And therefore, oh, that's data that's out of bounds.

So we'll go with a default response, rather than learning how to deal with facial recognition on things like that. So here, it's not such a big-- I don't think that we're going to have a problem with systematically discriminating against Gregorian chant. But it's these kinds of issues that lead to so many of the other ethical issues in artificial intelligence.

With that aside, we've all caught up. Let's see, did I, somebody-- and everybody just save me time when I type something wrong. Check it. But let's see if that works. Whew. extract_meter(bach). We have our first, our first automatic feature extractor. Is this topic, so far, as boring as I promised?

**STUDENT:** No.

MICHAEL SCOTT ASATO CUTHBERT: No OK, Somebody likes it. Yeah, it's pretty good. But let's go with something else. And we're going to start with another ethical dilemma. We're going to be working with Ryan's Mammoth Collection. And I think, if you type "blooming," you should get something that's not an error.

Do you? Because you've already-- if you've run everything, you've already run corpus, parse, BloomingMeadowsJig. And let's see that. So apologies we didn't get to the jig composition yet. I do plan to pick that up next week, but I figure that there's some other things that people need for their final projects, more importantly than working with that.

So you all see something like this, but we're going to be back on jigs again today for a bit. Now, Ryan's Mammoth Collection is a little bit of an ethical dilemma, and I'm trying to replace it. It is a collection of, I think, something like 1.079 fiddle tunes from the 19th century. And that has a bunch of jigs, hornpipes, things that if you take Joe Maurer's Sea Shanties and English Folk Music class and all those things, you can learn more about them. It also is a large collection of musical experiences of 19th century African-Americans that has been parodied and/or appropriated into white minstrel shows.

So I've tried to remove things that are offensive, on first sight, from the collection, titles and things like that. But the deeper history, a lot of these songs, has to do with a lot of appropriation. So I'm hoping to be able to get a collection to replace it at some point, because it's not exactly everything that I want to do with it. But anyhow, so we have-- for going through, we have the one jig, Blooming Meadows jig.

And I gave you just a contrast, a reel, R-E-E-L, and you should already have the reel, I think, already loaded. So you don't need to type this. It's called the "7th Regiment Reel." And hopefully, I got that. And it looks a little bit different.

So what we're going to do today is try to build a system that can tell the difference between a jig and a reel. Or a jig and-- try to fit it into the remaining 52 minutes between a jig and a not-jig. The whole, all of music is divided into jigs and not jigs. It's great.

So we we'll be looking-- how would you define the difference? You can look down at your own so I don't have to keep scrolling back and forth. Just, if you're somebody who didn't know, hadn't encountered a reel or a jig before, what is the-- what would you say is an obvious difference? Matthew? Do you--

**STUDENT:** 2/4 versus 6/8?

MICHAEL SCOTT ASATO CUTHBERT: 2/4 versus 6/8. Good. Another signature?

**STUDENT:** It's the essential one, I think.

MICHAEL SCOTT ASATO CUTHBERT: What's that?

**STUDENT:** It's the essential one, the time signature.

MICHAEL SCOTT ASATO CUTHBERT: The same one, time signature? Good. Yeah.

**STUDENT:** So the [INAUDIBLE] seems more syncopated.

MICHAEL SCOTT ASATO CUTHBERT: Which one seems more syncopated?

**STUDENT:** The jig.

MICHAEL SCOTT ASATO CUTHBERT: The jig, a little bit more syncopated? Especially things-- there's always things like this that are not, technically, rhythmically syncopated, but kind of feel a little bit. Mm-hmm. Good. Anything else? Yeah, Marlon.

[NOTIFICATION TONE]

**STUDENT:** [INAUDIBLE]

MICHAEL SCOTT ASATO CUTHBERT: What's that?

**STUDENT:** I said 16th versus eighth notes.

MICHAEL SCOTT ASATO CUTHBERT: 16th versus eighth notes. Thank you. I'm going to put that in, and I'm also going to disable my, my what do you call it, my notifications. Supposed to happen automatically. OK, yeah. 16th versus eighth notes. We're actually going to possibly use that as something.

So we're going to try to have a system that can do the difference. You should have-- if you type ryan, you should be getting something. I'm going to really quickly just cut and paste what I did to get to ryan and explain it. I first created something called ryanM, where I find everything in Ryan's Mammoth Collection.

And then what I've done is, here's another cut and paste, 1059, I think, yep, is that I've gone through everything in there. And said, first off, I only want to look at the first 500 of them. And then, I'm looking, without parsing, if the name of the file has jig in it. If it doesn't have jig in it, and if you're over file 100, continue.

So why did I do all this stuff? Just so that I can get something that is approximately 50/50 jig, not jig. So we end up with about 200. Here, we can-- we end up with 212 pieces, and I happen to know that 59% of them are jigs. So that's pretty good. These are all what we call metadata entries, so they haven't been parsed yet.

OK, great. Everybody with me? There. Great. I'm going to define a helper function that is-- how many ethical dilemmas are we doing today? This one's ethical, but this one's sort of a research thing called is_jig, which determines if something is a jig by looking at the file name and lowercasing it, and seeing if the word jig is in that.

I've done this with rags before, ragtime, a really great piano genre. Scott Joplin, the most famous person. And a lot of rags have "rag" in the title. "Maple Leaf Rag," and so on. But then, there's "The Entertainer," [VOCALIZING]

--which doesn't have the word "rag" in the title. So we're possibly having what we call a bad ground truth. The ground truth is the definition that is correct beforehand. And so, is this song a jig? Is this not a jig? This is probably not the greatest ground truth there.

The other thing that will come up a lot is that there is some pieces that will be very hard to tell if they are. And two different, intelligent people could disagree on something that's a jig. This comes up a lot in genre classification, where people can't really decide, is this country or is this rock? Is this funk? Is this hip-hop? Is this romantic? Is this classical?

So somebody once told me, when they realized I came from a music history training, musicology, they said, the worst part about musicologists getting into this field is that they give us really crappy ground truths. You all won't say what the right answer is. And that's, I hope, an ambiguity we like as humans. But anyhow, so we'll have our is_jig.

We'll actually see, is_jig(blooming). I think that was the one that is a jig. And what was the other one? A reel. OK. So with our garbage in, garbage out, terrible ground truth, we-- seems to be working OK. And I happen to know, 127 of the things in there is jigs.

So let's get out second feature extractor. This one's going to be pretty easy. We'll just get the number of sharps in the piece. We started working on jigs in G, so we happen to know that there are a whole bunch of them.

So maybe we'll do this. And here, we can just do, again, a try, return the scores key.KeySignature, the number of sharps. If it's two flats, what's it going to return?

**STUDENT:** Negative 2.

MICHAEL SCOTT ASATO CUTHBERT: Negative 2, yep. Great. And if there's no key signature-- this one's actually not a terrible assumption, right? Something without a key signature probably has a key signature of no sharps or flats. I don't know, maybe. Maybe we should, in this case, analyze the key and do something like that. But good enough for just a little bit.

So let's see. The blooming, hopefully this will return 1. Oops. Oh, I know what I did. This is why you should never do something like this. Except IndexError, we said, and we didn't get an index error. What did we get? We got an AttributeError?

Yep, there is no sharps, because I forgot to say that we want the first key signatures sharps. You can also do the dot 0 or bracket 0, however you'd like that. And so let me try the get_sharps again. Getting 1. And what's the answer for Bach?

**STUDENT:** 3.

MICHAEL SCOTT ASATO CUTHBERT: 3, good. You can either do that by hand or make sure [INAUDIBLE]. Great. And we'll define one more feature extractor so that we can-- oops. We're actually making good time this-- we'll do eighth_fraction based on the observation that there's more eighth notes in one piece than the other.

So we'll take in a stream.Score. And this time, we'll return a float. So usually, you can return an int, which we call a continuous-- a discrete vari-- answer or a float, which is a continuous variable. So let's just do this. We'll get the number of notes in the piece.

Actually, you do this. You give me-- go ahead and try coding this for a little bit, because we are a couple minutes ahead of time. How do we get the returns? The fra-- you don't need to put this. Returns the fraction of notes in a score that are eighth notes.

OK, you can keep doing yours. If you're stuck, I'm going to give what I do. Sign like--

[TYPING]

Not the most efficient use of iterators. I've done two.

[TYPING]

**STUDENT:** You're missing an H.

MICHAEL SCOTT ASATO CUTHBERT: I'm-- where am I missing it? It's somewhere in eighths, right?

**STUDENT:** Yeah.

MICHAEL SCOTT ASATO CUTHBERT: Where's--

**STUDENT:** So second to last.

MICHAEL SCOTT ASATO CUTHBERT: Ah, eighth. That's-- yep, eighths. Thank you. Appreciate that. And go ahead and run your num eighths on Blooming, on Reel, and on Bach. And use your Spidey sense to figure out if your numbers are pretty meaningful.

If there's anything I can get from this class to just use in all of your programming things is, always use your intuition to see if something like that, something is working right. Ooh, wow. Look at that. The reel is incredibly low. Oops. Now I have too many eighth fracti-- I should have done it eighths fraction, but whatever. I'm--

That actually feels too low for that reel. Only 8%. But I guess there's a lot of 16th notes. OK, give me a snap if you're ready to move on.

[SNAPPING]

OK, hearing some snaps, not all. We'll give another 10 seconds or so. So so far, we have the first one that seems really to give, at least in this one piece, a big distinction among the three types of things.

OK, now I'm going to wade into something that people don't always say, is this a feature extractor because it returns a number or something? Or is, when you put them all together, is that the feature extractor? I'm not sure, but I'll use this term. So the feature extractor is going to run all of our different little feature extractors.

And in this case, I'm going to say, it's going to return a tuple of-- and I'll tell you what these are in a bit. But here, I'll write this out here. A string, two ints, a float, an int, and an int. And what are these going to be?

They're going to be the file name, which is pure metadata, just so that we can look at things later. They're going to be the numerator and the denominator of the piece using our function we wrote, extract meter. They're going to be-- what's the only thing we've written that returns a float as the answer?

**STUDENT:** Eighth.

MICHAEL SCOTT ASATO CUTHBERT: What's that?

**STUDENT:** Proportion of eighth.

MICHAEL SCOTT ASATO CUTHBERT: Eighth, yeah, eighth fraction, eighth proportion, whatever we are going to call it, the number of sharps, and then that last one we'll get to. That's going to be the ground truth. And it's doing too much JSON there. Stop

Doing trailing commas. So the file name, you get this from a score's metadata object, which we haven't really talked about, it's file path split. I'm just going to get the last bit so I'm not showing my entire directory tree to everything.

So I'll just get the last thing after the slash. Those of you who know Python's path library, you can probably do better. The numerator and the denominator will extract meter from the score.

The eighths we'll say is the eighth fraction of the score. Sharps feature is get sharps. You can call them whatever you'd like. And then the ground truth, is it a jig or is it not? But it has to be a number.

So we'll say integer of is jig. You could also say, if you like this Python, 1 if jig, else 0. Sometimes explicit's a little bit nicer to see what that's going to be. And then we want to return all that file name numerator, denominator, eighths-- I should do quarters. I can spell quarter-- feature, and ground truth.

I'll let everybody get caught up while I debug my own system. Whoa, some fast typists in this class. Feature extractor on bach. Let me just see-- feature extractor on blooming. OK.

ABC file extension, yep.

**STUDENT:** [INAUDIBLE]

MICHAEL SCOTT ASATO CUTHBERT: Yeah. A lot of these are all written as ABC files. We didn't really cover-- no, we didn't cover ABC files because they're tiny bit more complex than the easiest to encode handwritten things but not complex enough as like musicxml or something.

OK, so blooming metals jig, it's in 6/8. 77% of its notes are eighth notes. It's in one sharp. And that last one indicates what again?

**STUDENT:** Jig.

MICHAEL SCOTT ASATO CUTHBERT: Yep, so the one indicates jig, right? Great. OK, so what I'm going to do right now is I'm just going to run this for a second. So how would I get everything?

So we can do metadata. I'm just doing this so we can see how we do this. In the Ryan thing, piece equals metadata_entry.parse. And then we'll print the feature extractor on the piece.

Let me just try to see. Yep. Because of the way I did it, you'll see that at first, there's not too many jigs. And then later on, there's jigs.

OK, now, if you scroll back up to the top of your-- sorry, I'm going to have to get my blank template for a second. Oops, that's way smaller. There should be, on the top of your template if you downloaded it, something called write_ryan. And you just need to change your path to someplace that you're going to be able to remember it.

So change that, unless you've created a Cuthbert user because it makes it easier for you to cut and paste my code. You probably want to change that. And then you can run this while I explain a little bit what it is. Is write_ryan working for people when you get that? OK.

Great. So I'm going to have my own version here.

**STUDENT:** [INAUDIBLE]

MICHAEL SCOTT ASATO CUTHBERT: The write_ryan( ) command. W-- you don't have to do any of this, but it'll be kind of fun because then you'll have write_ryan and start.

OK, I've seen more people waiting than typing for a little bit, so I'll just explain what we do. Some of these things, if you work with files, you probably know them pretty well. This is really not necessarily the best way to do this, but I'm going to open two files.

One of them is the file that we're going to use to train our classifier, the thing that is going to try to learn if something is a jig or not. The other one is the file we're going to do to try to test whether it can do it or not. We're going to hope that we don't put the same data into both. So we're going to try to put some data in the testing, some data in the training.

There's a number of different ways you can do this type of work. Sometimes you put 90% in the training and 10% in the testing. Sometimes you put 50/50, which is what we're going to do. Some of it has to do with how much data you have.

So what I'm going to do-- maybe it'll be easier to see-- I'm just going to separate-- .tab means everything's going to be separated by tabs. So I'm going to first put a header line that says the names of the columns. These are just for us to read.

So we have a file name, numerator, denominator, eighths, sharp, and isjig. And so we're going to write that to the top of it because a lot of the classifiers that work on tab data expect this to be the first line. Now, we're going to hope that when running this system, that it doesn't peek at the file name. And we'll see because obviously, file name would give it away.

And then so the descriptions, what kind of data is this? The file name's a string. Then we have discrete, as we said for an integer, discrete, continuous discrete, discrete. So we'll write that out again to each of the two files.

And then we have third line, which is mostly blank for all these things. And the first one just says metadata. That is to say, don't peek at the file name-- do not use the file name to try to classify this-- then normal information and then the class, the thing that we're trying to figure out. And we really, really hope that the artificial intelligence thing does not peek at the class when it's trying to look.

You can really quickly create an incredible AI thing if you would allow it to peek. And so we'll put that out. So that's just all headers. And that's why I didn't want to spend too much time writing it.

Then what are we going to do? Same thing we were saying before, we're just going to get each piece into there. I've done enumerate for a reason so I can keep track of what the piece number is. And then we can extract the piece, all the features from it. And then we'll just write them out.

In this case, we're going to write them as a string because we can only write strings out to a text file. Write each of the features out, tab separated. Actually, we'll get ready to write it. We'll just put it into a string.

And then every other one, we're going to write to training and test, training and test. Probably, another way to do it would be to flip a coin each time and use a random number so you can see if things are going there. I'm not an expert on this type of work. But that's how that works. And I don't think--

Yeah, I don't think I'm going to run it now because I should have been running it during this time. You know what? I'm just going to put train data x and test data x so that while it's running, I have access to our baked one. So actually we can look at it, magic features. So we switch out of Python and look at the shell for a second-- train data-- wherever you put your thing.

And I can see that I've written out-- it's kind of weird that the tabs don't show up. You would love to have something that would automatically put it out. But we have the file name, all our headers, description of the data, the metadata here. And then yeah, we have all of our information written out. So that's pretty good. Oh that did finish, great.

OK. So now we've done our feature extraction. The feature extraction part of the lecture is done for a little bit. This is a small number of features. But I think that they're going to end up being halfway decent features because we thought about them a bit. And when the data is small, you want to think more about your features.

So you installed something called Orange 3. But it's a little bit weird. It comes as capital Orange. I just I don't know why. I don't like things capital. So I always import orange as lowercase orange. Great.

We're going to load our training data and our test data, so first training data, orange.data.table. A lot of these things are going to be very similar in a super modern artificial intelligence framework like PyTorch or TensorFlow or something like this. Similar idea, you want to load data in as a table.

If it's too big to load on your all at once, then there's special techniques and stuff like that. But we'll load the training data in. And now when we look at it, it just looks a little bit nicer.

Actually, where's our first-- I don't know-- 45 to 50? Does that work? Yeah. OK, so that might be a little bit better because we can see we have our features here two for 12% and 3 sharps. And this one happens to be a jig, so there. And so the afterward is what the class is.

Great, and we'll do the exact same thing with the test data. And we'll call it variable test data. So I'm just going to copy and paste. And hopefully, the test data looks a lot like the training data.

**STUDENT:** Should we change the file path?

MICHAEL SCOTT ASATO CUTHBERT: Oh, yeah, yeah. It looks a lot like the training data if we don't change the file path. You see how well it is. OK, so now we're going to create some, we'll call learners, classifiers, things that can classify the data. Here's the first one and I think, in some ways, the most important classifier that you'll ever use. And that is-- why did I switch into camelcase? OK, I guess I'm in camelcase now-- something called orange.classification.MajorityLearner.

And what a majority learner is it's kind of the control or the placebo of your system to make sure that you don't get too excited by how good your system is working. What it does is it looks at the training data, which has hundreds-- what maybe the training data has-- I don't know. Let's say it has 51 jigs and 49 nonjigs.

And it says, I will always say that any piece that's ever shown to me is a jig because then I will get over 50% right or I will get the highest answer of doing anything. So if you are not beating your majority learner, then you're not doing any classification. So the majority classifier-- you all who have taken a machine learning class, do you get taught this or no? Yeah. OK, good.

Some people, this will be totally old story and not. So the majority learner is just a generic thing. And now we have a classifier, which is one trained on our data. So we're going to say that the majority classifier will now know how to work with this particular data.

And then we'll use another one called k-nearest neighbors. This was-- I don't know-- maybe 20 years ago, the state of the art of machine learning. It's a little bit out of date now. But it does work pretty well on small data sets and few classifiers. So nothing that it's going to get you super excited at a top tech firm. But it still has its uses.

So we're going to do the same exact same thing. Everything that I put from majority learner, and just put it as KNN neighbors. Look at the K, which might be 4, might be 3, might be 5, the nearest neighbors in a multi-dimensional vector space to your one piece and try to learn where in this space it goes.

Snap if you're caught. Snap if you're not now, if you need a second. OK, good. Great. So I'm just going to keep track of how many correct the majority gets in a variable called majCorrect and how many correct the k-nearest neighbor gets in a variable called knnCorrect. And then total equals the length of the test data, not the training data.

So now we're going to be working on this. And we should have something like this. By the way, I lost half an hour in preparing this lecture on, why do I keep getting over 100%? Always make sure to reclear out your number correct as you recode or something or make sure to do it within your loop.

So good. So we have 106 things that we're going to be able to do now. That was for our loop. So we'll just do an enumeration of each row in the test data. So for i comma test row in enumerate test data.

And then the majority learner is going to take a guess for each one. So majority's guessed is going to be the majority classifier of the test row, which it's always going to guess the same thing. I guess in this case, it's always going to guess jig because there's four more jigs than there are non jigs.

And then our classifier, k-nearest neighbor classifier, will also take a guess, knnClassifier. I know somebody's probably saying "guess" is not the right word, but type works. And then we'll get the real is the testrows.get_class.

So as you said, the class is either 1 or 0, something like that. And we can do print out something. But print-- how about this? We'll just say majGuess, knnGuess, and real.

So I'll start this running for a second. But notice, we haven't put in anything on whether it got it right or not. But we can just look at that. Oh, one of them guesses.

So you can see that the majority one for every single piece just puts out-- yep, it's always a jig, always a jig, always a jig. K-nearest neighbor says, I don't think this is a jig. And here's the correct answer-- not a jig, not a jig, not a jig. And here, k-nearest neighbor is learning that, ooh, this one looks like a jig.

So I'll just modify this a tiny bit and say, if majGuess equals real, then majCorrect add 1. If knnGuess equals real, knnCorrect equals 1. When you're trying to figure out how to do things, you'll use more classifiers. There's a lot more. And so you probably want to do this in some of a loop. But great.

Good. And so now we'll see how we did. How did majority do? Here, I'll do this. Majority majCorrect-- I always making things look nice for humans.

Why am I anal enough to start already putting in some of a rounding thing to look? Because I do feel like when you see all those digits, it's really, really hard to figure out how important the difference is.

So the majority, this is what I happen to know that there's 59% of them are jigs. And so the majority learner got 59% of them. And then we'll do the same thing for KNN.

Here, I can cut and paste, right? We all know how to do that. What are we looking up? Great. And we want to make sure that that's knnCorrect. So our classifier is getting about 87% correct at this, just with these few little things.

Now, this is always the exciting moment. And then the hard part is almost always getting it from 87 to 95. And the next harder part is getting it from 95 to 97 and then going up and up and up from there. So that's the theoretical part aside.

I'm going to really quickly just go through some things that music 21 has that just make some of this a little bit easier. So I know it's already imported since I imported star. But there's something called features. And what we can do is set equals a features.DataSet classLabel equals is_jig.

And then we'll do the same thing with the test set. So far, it's basically the same type of thing. And then I believe if you type fes, you should have something there. I think I pasted this.

These are built-in feature extractor. And each one of them has an ID. And no, I don't have these memorized, but something about-- I can't remember what these are. You know what? I'm just going to cut and paste because you already have this and put it here.

**STUDENT:** You have fears?

MICHAEL SCOTT ASATO CUTHBERT: Fears, yes, that's my hidden-- thank you very much. Appreciate that. And think that should work.

And so these are some features that have already been put in. So one of them, I don't know, what's the range of the piece? How far is it from the lowest note to the highest note?

A lot of these, they're adapted from a really, really cool toolkit called jSymbolic part of jMIR by Cory Mckay up in Canada, and just ported them over because they're kind of neat. And then some of them are ones that I've written because I kind think that they're a little valuable or somebody in the Music 21 team has written.

So here, we have a lot more features here. And what we can do is we have our two data sets. So we'll say the training set addFeatureExtractors and add all those features extractors and then same thing for the test set.

And this, as I said, you don't need to be typing too much because I'm just going to go pretty fast. And what we'll do is I'm just going to cut and paste this because all we're doing is dividing things up into either the test or the training set, depending on if it's even or odd. And so we'll just add the data on here with some nice little features. And this will just generate all the headers for you.

Whoops, what did I do? Testing set is what I called it before. Sorry, training and testing, that makes sense. Did I do something wrong? Oh I have to change it up here. There we go. Starting all over, hopefully, I don't have any num correct.

Great, and that's just doing all the feature extraction of getting them in. That's going to take a little bit of time. So we're going to switch like they do in the cooking shows to the already baked version where it's all done.

OK. Oops. Here we go. Where do we go? Yep, so the next couple of things that you do, you want to process all your training set and write it out just like before, process it and write it out.

Why is it taking so much longer? Because we gave it a whole bunch of features. And some of them take a long time to do, like ones that have to do with variability between parts and parallel fifths, finding if it's a multi-part thing and so on.

But then the next row ends up being the exact same type of thing. We have a majority learner. We're going to still use the knnLearner and then try to see how well it did and keep looking.

And again, the majority learner is still 59%. You hope that doesn't change. And now, using all of these extra features, we've gotten our number from 87% all the way up to 61%. So sometimes, just throwing more feature extractors at the problem is just going to have you wait longer at the computer.

Sometimes, thinking really carefully about what you already think the data might look like is a pretty good thing, and especially if you're working with a small data set where it's not going to be able to do the kinds of deep neural network learning. You have to keep going through enough to really build a model of the set.

As some people say that look, the neural networks are getting better and better. And we just need to throw more data at it. But imagine if you wanted to have a program that was automatically reading handwriting. And you gave the program all these papers of handwriting. And you just gave them an audio file of how it drops, how it sounds when it drops and try to do it. You're giving it kind of the wrong data to learn on.

And so a lot of the extractors now, unless you're writing your own or unless you're using really musically sensitive ones, you can save the computer a lot of time by trying to already say, hey, let's process this as a symbols ahead of time or something like that.

A lot of, by the way, state of the art of symbolic music machine learning things are-- you all remember the piano roll where we put things like this to show length and stuff like that? Yeah, well, it's converting the entire musical score to a piano roll, saving that as an image, and then using image classifiers and image generators to generate pieces that look like the piano rolls and then translating them back.

So I'm hoping that over the next decade or so, we can do better and do some things that help these neural nets and classifiers work with data that's a little bit closer to the original numbers.

OK, I have time for just a little fun thing with my favorite low feature extraction thing. So let me just show a couple little things. OK. Yeah. First off, the importance of thinking about data sets and ground truths, this is, I think, one of the first pieces that showed deep learning, the kind of for people in the know moment like ChatGPT has been in the last year happened.

I think it was about 10 years ago when AlphaZero and AlphaGo suddenly became the world's greatest go and chess players. And so a lot of people think, well, this was with generalized knowledge, no rules being taught except for legal move or not.

But one of the things to think about the difference between a lot of the things that they were being trained on and what you have is that AlphaZero, the chess playing computer, played, what was it, 700,000 games or a million games or something like that. And at the end of every single game it played, it got perfect feedback on how it did-- you won, you lost, you tied, you won, you lost, you tied.

A lot of the things that we're interested in doing with music like music recommendation or things like this, predicting what's the next great hit, first off, we don't have that correct data. Maybe what's the Billboard chart or something like that?

But for how good was this piece, because it's going to be different every time. It's going to take us three minutes to listen to if it's computer generating a song. And then we're going to give it feedback that we might feel differently about that piece tomorrow. So we don't give perfect feedback. And it takes a long time. So this is one of the reasons I like working with some of the older things.

OK, we'll go back. Great. I don't think I presented this. No, I presented in other place. So this is what we'll end for the last 10 minutes and talking about some of the interesting things that come out with features.

So how would you distinguish choral music from Bach from Monteverdi? So we're doing a lot of Bach in this class. But we'll get a little bit more into the ear. Here's Bach.

[MUSIC PLAYING]

So this is Bach. And so then we'll switch to one of the other great composers about 100 years before Bach, Claudio Monteverdi. And so, some of you, this might be the first time you hear him.

[NO AUDIO]

One second with the Back again so we can get back. Great. What's something you heard different between the two of them?

**STUDENT:** Monteverdi was a bit more somber with more of the minor keys.

MICHAEL SCOTT ASATO CUTHBERT: OK, somber, more minor keys. We could do a key detection. Jonathan.

**STUDENT:** I mean, Bach had sounded like there were more people singing. But it's hard to tell.

MICHAEL SCOTT ASATO CUTHBERT: Yeah, but it could be. It could be more people, fuller than good. Yeah.

**STUDENT:** Bach is more together where Monteverdi has a lot of voices going off doing kind of thing.

MICHAEL SCOTT ASATO CUTHBERT: Bach more together, homophonic maybe, Monteverdi doing their own thing. Good. One last one. OK, that's a pretty good place to start.

So I threw everything in, all the kitchen sink of feature extractors and got to see what a particular algorithm did that seemed to work pretty well. And the first, here's a little quiz. It's one of the few quizzes that elementary school students do better than adults at.

So let's see if we can figure this out. 7802 is 3. 1065 is 2. If you've seen it before, then don't answer. 998 is 4. And 1773 is 0.

If you're new to this, 1, 2, 3, 3 circles, 1 circle, 2 circles, 1, 2, 3, 4 circles and no circles. So I bring that up to say that sometimes the computer is going to be looking at things that are really, really far out and trying to figure out things.

So there's all these things that we could be looking at. I'll bet that the proportion of instruments that are electronic isn't going to help with Bach versus Monteverdi. But one of the things I like about a particular classifier that is not close to the state of the art or to the getting the best results. But this tree learner can-- unlike being a black box, it can actually show what the path-- so it's of a decision tree, branching tree-- it took to get there.

So here's how it can say, to begin, if the note f sharp below middle C is used more than 14% of the time, the piece is by Bach. Otherwise, if the top number in the time signature is 3 or less, it is by Bach.

Otherwise-- I love this one. If the distance between the highest note in the soprano and the lowest note of the bass is less than 2 octaves and a minor sixth, then it's by Bach. And then there's three more things. We get to the final one, count the number of times each note is used. Find the note that is used the most often and the note that is used the second most often.

If the second most often used note is used less than 97% as often as the most commonly used note, the piece is by Bach. Otherwise, it's by Monteverdi. And this worked with 99% accuracy.

What I loved about this is that this was the first decision to distinguish. And I was just baffled by it until I realized, well, OK, Bach has more pieces in sharp keys than Monteverdi is. OK, that would make sense. Monteverdi was mostly hovering around no sharps or flats or one flat assigned.

But then it's minor, and there's raised leading tones and stuff like that, so why not? Well, the leading tone tends to get raised in the melody, which tends to be in a higher voice. The middle area, tenor, will only appear as f sharp if the key signature is already f sharp. So it's a way of learning some things like this.

And this particular one ended up being garbage in, garbage out, that Monteverdi doesn't using modern time signatures. And so the people who encoded this chose never to use 2-4, always put it into 4-4, because it also weren't bar lines. So you could decide how long.

So this ended up being that the computer learned not the difference between Bach and Monteverdi but the difference between Bach's editor, modern editor, and Monteverdi modern editor.

I have lost the reference, but I have been told about an audio classifier that tried to classify by genre. And it was doing very, very well until it turns out that different genres of music used different microphones in general. And it was picking up the difference between the microphones.

So if you sing country music on a hip hop microphone, it would classify as hip hop. So all of these things, we need to be really careful about what's going on with the potential for classification. And so this is where I like this old school-- imagine this as a tree diagram of things going left and right, I guess if you turn it sideways-- these old school feature classifiers a little bit just to check what might be happening in the latest black box.

So you guys did a remarkable stamina. You got through the zombie of music eating feature extraction. And I hope that some people will choose to find-- if you find this to be helpful in doing something in your final project, you now have some of the tools to do it.

On Friday, we'll continue with searching and similarity, and will be giving the first 15 minutes explaining what he's doing at MIT. He's a professor of computer science and what he's been doing with music on his work. So there'll be a start. And then I'll continue with some other things. Great, thanks.