0:00

This is the second week and we are talking about supervised learning more

0:05

specifically about classification and regression. We will also talk about the evaluation

0:12

of models and how the learning process works. To recap, the field of machine learning

0:19

is a sub field of artificial intelligence where we study models or algorithms that can learn from data and

0:26

generalize to unseen data without explicit instructions.

0:32

In supervised learning, we work with a data set that contains variables called features. These are the columns of a

0:38

data set and a dependent variable, also called the target or output.

0:44

Usually you have a table where you list the features and one of these is the target or class, the value you want to

0:51

predict. The features are also called attributes. Each column represents one.

0:57

At the end of the process using what we call a training algorithm, we learn how to predict the output given

1:04

the inputs or features. Essentially, we are learning a function

1:09

that maps the feature columns to the target. More specifically, in classification, we

1:15

are predicting a category. For example, we might predict whether an image is of a dog or a cat. Another example could be

1:23

predicting a person's eye color. There are many applications where the target

1:28

is discrete and categorical. In these cases, you have a limited number of

1:33

categories. When the target is a continuous numeric value, the task is called regression. In

1:41

regression, we are finding a function that receives inputs and predicts a number. For instance, if we want to

1:48

predict the price of a house based on its square footage, neighborhood, age, and other features, this is a regression

1:56

task. The target is not a categorical label, but a numeric value.

2:02

To summarize, classification involves predicting a discrete categorical value or label. Regression involves predicting

2:10

a continuous numeric value. We will now start talking about the

2:16

foundational machine learning models. This can serve as a recap session since

2:22

all of this content was introduced in the first week. Let's talk about the mother of all machine learning

2:28

algorithms, linear regression. It is widely studied in STEM and mathematically oriented courses. Despite

2:35

its simplicity, linear regression is indeed a machine learning algorithm and

2:40

clearly illustrates what machine learning is. In its simplest form,

2:46

linear regression tries to determine a linear relationship between two variables. For example, the input could

2:52

be years of experience and the goal is to predict salary. Essentially, you're

2:57

trying to fit a line to the data. There are other types of regression, but in linear regression, you fit a line to the

3:04

data. You use a learning algorithm that works during training to minimize the sum of squared distances between the

3:11

data points and the regression line. Suppose you start with a randomly drawn line. The goal is to adjust the slope

3:18

beta and the intercept epsilon so that the line fits the data better. The

3:24

initial line may not fit the data well, so you tweak beta and epsilon to improve the fit.

3:30

The learning algorithm updates beta and epsilon iteratively. You start with

3:35

random values for these parameters and update them to minimize the discrepancy between the predicted line and the

3:42

actual data points. This discrepancy is measured as the squared difference between each point and the line. Every

3:49

training algorithm in machine learning is an optimization process. You optimize

3:54

parameters here beta and epsilon so that these squared differences are minimized.

4:00

The reason we square the differences is that we don't care if the prediction error is positive or negative. Squaring

4:07

ensures all errors contribute positively to the total. Why do we minimize these

4:13

errors? Because we want to generalize because we want to generalize well to unseen data. Suppose you're hiring a new

4:20

person and want to estimate their salary based on 10 years of experience. Even if

4:26

you didn't have this exact data point in your training data, you can still make a prediction using the fitted line. When

4:33

the squared differences are small, you get a better fit. This process is formalized using a loss function. For

4:40

linear regression, the loss function is the mean squared error, MSE.

4:45

During training, the algorithm minimizes this loss function to find the best fitting line. Now, it's important to

4:52

distinguish between parameters and hyperparameters. A parameter is a value inside the model

4:58

that is learned during training. In this case, beta, this slope, and epsilon, the

5:04

intercept, are the model parameters. A hyperparameter is a configuration setting that controls the learning

5:10

process. For example, the learning rate determines how quickly the model updates

5:15

its parameters. Hyperparameters are set before training begins and require

5:21

experimentation to find good values. This example uses a one-dimensional

5:26

model where you have one feature experience predicting one output salary. The term dimension or dimensionality

5:32

refers to the number of features used as input to the model. If your data set includes only one feature column,

5:38

experience, and one target, salary, it's a one-dimensional regression. You can

5:43

increase the dimensionality by adding more variables. For example, to predict height, you might use arm length,

5:49

biological sex, athletic status. These additional variables form a multi-dimensional model, in this case,

5:56

three-dimensional, that better captures the variations in height. you'll still have an intercept. And in higher

6:02

dimensions, the fitted model is a linear combination of features. While we can graph up to three

6:08

dimensions, in n-dimensional problems, the model is still mathematically a line, or more precisely, a hyper plane.

6:14

Linear regression is a fundamental model that helps introduce core concepts such as parameters, hyperparameters,

6:20

dimensionality, training algorithms, and loss functions. To summarize the loss function again,

6:26

it's the function you minimize during training. For linear regression, the default is the mean squared error. You

6:32

sum the squared differences between predicted and actual values for all data points. The smaller this sum, the better

6:38

the fit. Although MSE is the common choice, different loss functions can be used depending on the task. Now let's

6:45

talk about logistic regression. It is a variant of linear regression and is one of the most basic classification

6:52

algorithms. Linear regression is used for regression tasks where the goal is to predict continuous values such as

6:59

height or salary. Logistic regression despite its name is actually used for

7:04

classification tasks. While the name includes regression and we will see why

7:10

its primary use is for predicting categorical outputs. For example, imagine we want to predict

7:16

whether a fruit is an apple or an orange based on two features weight and color.

7:21

Let's say color is represented numerically. Lower scores for green, higher scores for reddish shoes. You

7:28

would have a data set with color, weight, and a class label, apple or orange, based on historical data. As new

7:36

data comes in, the model should classify the fruit correctly. Using linear regression wouldn't help because we're

7:42

not predicting a continuous value. We're predicting a class. That's where logistic regression comes in. Instead of

7:49

fitting a straight line, logistic regression fits a sigmoid function. Suppose the output is zero for orange

7:56

and one for apple. In a simplified one-dimensional example using only the

8:01

color feature, a straight line would not effectively separate the two classes.

8:06

There might be overlap. Some oranges and apples may share similar color values, making it difficult to draw a straight

8:12

line that separates them cleanly. The sigmoid function, which has an S-shaped

8:18

curve, is better suited for this. It is bounded between 0 and one and outputs a

8:23

probability that a given data point belongs to a certain class. For example,

8:28

if a fruit falls at a point on the sigmoid curve that corresponds to 8, the

8:34

model would assign an 80% probability that it's an apple. A threshold is then

8:39

applied. If the predicted probability is greater than 0.5, classify as apple one.

8:45

If less, classify as orange, zero. This threshold can be adjusted depending on

8:51

the distribution of the historical data. For instance, if the data shows less overlap between classes, the threshold

8:58

might be raised or lowered. For example, predicting apple only for values greater than 7. The parameters of the sigmoid

9:06

curve and the threshold if tuned are learned through a training algorithm that minimizes a loss function. In this

9:14

classification context, the loss function often used is also the mean squared error MSE between the predicted

9:21

probability and the actual class label. For example, if the true label is one

9:27

apple and the model predicts.3, the error is 1 minus pin.3.

9:33

The training algorithm adjusts the model to minimize such losses across all training examples.

9:41

In summary, logistic regression is used for classification, not regression. It

9:47

uses a sigmoid function to map inputs to probabilities between 0 and one. The

9:53

next algorithm is called K nearest neighbors or KNN.

9:58

Although it is considered a machine learning algorithm, it does not have a training phase. It is known as a

10:04

non-parametric algorithm, meaning it doesn't involve learning model parameters during training. Instead, it

10:11

operates directly on the data during prediction. KN&N is used for both classification and

10:19

regression. It's considered a machine learning algorithm because it's widely applied in practice even though it

10:26

doesn't involve a traditional learning process. The idea is simple. Given a new

10:31

data point, for example, in a classification task, KN&N finds the K closest data points in

10:40

the training set and predicts the class based on them. Consider a case where you

10:46

want to classify a person as male or female based on their height. One approach would be to use logistic

10:53

regression. Another approach is to use KN&N. You find the five people with the

10:59

closest heights to the new person and assign the class that is most frequent

11:04

among those five. For instance, suppose the new person has a height of 179 cm.

11:11

You look at your data set and find the five closest heights. Let's say those five neighbors are 1

11:19

181 cm 2 180.5 cm 3 179 cm 4 178.5 cm 5

11:30

177.8 cm. Then check the gender of those five individuals. If the majority are

11:36

male, you assign the label male to the new person. The parameter K in this case

11:42

five is userdefined. KN&N is also used for regression. For

11:48

example, suppose you want to predict chest circumference based on height. A

11:53

new person comes in with a height of 180 cm. You find the three people in your

11:59

data set with the closest heights equals 52,000 + 55 + 51 equals 52.

12:06

The predicted chest circumference for the new person is 52 cm. This is the

12:12

essence of K nearest neighbors. It is simple yet effective and remains widely

12:18

used for both classification and regression tasks. Now let's see another model called

12:25

support vector machines SVMs. This is a machine learning algorithm with a

12:30

learning step. It was originally designed for classification tasks but can also be used for regression. The

12:36

core concept of SVM is to draw a decision boundary between data points in the training set so that when a new

12:43

unseen data point comes in, it can be classified accurately. For example, if

12:48

you're trying to classify animals by their weight and length of the nose to determine if they're cats or elephants,

12:53

you have a two-dimensional problem. The SVM creates a decision boundary and the simplest SVM creates a line to separate

13:01

cats from elephants. During training, SVM maximizes the

13:06

margin, which is the space between the outermost points of each class. These outermost points are called support

13:13

vectors. The loss function during SVM training focuses on maximizing this margin, making it an optimization

13:20

problem. SVM does not only create linear boundaries. It can create nonlinear

13:25

boundaries such as circular boundaries to separate classes that cannot be separated by a straight line. For

13:32

example, when separating apples and oranges by weight and color, a line may not separate the classes effectively,

13:38

but an SVM can create a circular decision boundary to separate them. To

13:43

create these curve boundaries, SVM uses kernel functions. This allows SVM to

13:49

handle complex nonlinear decision boundaries and makes it useful for regression tasks as well.

13:56

In summary, SVM can fit decision boundaries as lines, polomials, circles

14:01

or other functions. It is a simple yet powerful classifier and regression mechanism. Additionally, like linear

14:08

regression, logistic regression and KN&N, SVM is computationally efficient

14:14

and does not require extensive computational resources while providing strong classification and regression

14:19

capabilities. Another classifier is what we call naive bay. It belongs to a family of bay

14:27

classifiers which are probabilistic classifiers. Like logistic regression,

14:33

they output the probability of a class label. The naive Bay algorithm is the

14:38

simplest in this family, though there are other Baze classifiers with different assumptions. These classifiers

14:44

are based on BA theorem, a statistical theorem from the 1800s.

14:49

BA theorem calculates the probability of a hypothesis given some evidence. For example, it tells us the probability

14:56

that someone is sick given that their CO 19 test result is positive. This is

15:01

typically explained using a probability tree. A person may be sick or healthy.

15:07

Then they take a test which may come back positive or negative. If a sick person tests negative, that is a false

15:15

negative. A healthy person may also receive a positive test which is a false

15:20

positive. What we observe is the test result, the evidence, but we are

15:25

interested in the underlying condition, the hypothesis. BA's theorem lets us

15:30

compute the probability of the hypothesis being sick given the observed evidence a positive test result. The

15:38

mathematical notation this reads as the probability of being sick given that the

15:43

test result is positive. The vertical bar means given. There is a formal

15:48

equation for B theorem. It allows us to compute this probability from observed data. Students can be referred to

15:56

dedicated videos and readings to explore the derivation and computation of the formula in more detail.

16:02

In machine learning, this idea is applied to spam detection. The hypothesis is whether an email is spam

16:10

or not spam and the evidence is the presence of specific words in the email

16:17

such as prize, lottery, viagra. You may not directly observe whether an email is

16:23

spam or not, but based on the evidence, the presence of words, BA's theorem

16:29

provides a probability. For example, an email that includes the words Viagra and prize might have a 92.8%

16:37

probability of being spam, meaning a 7.2% probability of not being spam. The

16:44

term naive refers to the assumption that all pieces of evidence are independent

16:50

of each other. For instance, the algorithm assumes that the presence of the word viagra is independent of the

16:57

word penis even though they often co-occur. This independence assumption is rarely

17:04

true in real life, but it simplifies computation and the algorithm often

17:09

performs well regardless. Naive Bay is simple, efficient, and has a learning

17:14

phase. Here's how the learning works. You are given a data set of emails where

17:20

each email is labeled as spam or not spam. And each column represents the

17:25

presence of a particular word, lottery, prize, Viagra. The class label might be

17:32

zero for spam and one for not spam. From this, these probabilities are computed from the data set. During prediction,

17:39

the algorithm multiplies the learned probabilities for each piece of evidence to compute the final classification.

17:46

While naive bay does not minimize a loss function like some other algorithms, there is still a learning phase in which

17:52

probabilities are estimated from data. Decision trees are a machine learning model that uses a series of yes no

18:00

questions to partition data. For example, you can use a decision tree to

18:05

classify whether a person is low risk or high risk for a heart attack. You start

18:10

with a root node that asks a yes no question such as is the age less than

18:16

18. If yes, you check the weight. If it's less than 60, it's low risk. If

18:24

more than 60, it's high risk. If the age is between 18 and 30, it is classified

18:30

as low risk and so on. When a new data point comes in with attributes like age,

18:37

smoker status, and weight, the model uses the learned structure to classify

18:42

the individual. For instance, a 31-year-old smoker weighing 60 kg will

18:48

be classified by traversing the tree based on these attributes. During training, the algorithm learns the

18:54

splits or the parameters of these questions using a loss function based on

18:59

maximizing information gain. For example, if you have a data set with attributes such as outlook, windy,

19:06

temperature, humidity, and a target indicating if you can play outside, the algorithm evaluates which attribute to

19:14

split on first. It tests whether to split on outlook, windy, humidity, or

19:20

temperature. If splitting on Outlook results in an information gain of 0.247

19:26

and this is higher than splitting on other attributes, the algorithm will choose Outlook as the first split.

19:34

Information gain measures how well a split creates pure nodes. For instance,

19:40

if splitting on Outlook and getting overcast results in all yes for playing

19:45

outside, it creates a pure node indicating a good split. The tree

19:51

continues branching on the next attributes, selecting splits that maximize purity and information gain.

19:57

The algorithm recursively tests different splits, learning the parameters that lead to the best

20:02

classification performance. The goal is to create a tree where each split

20:08

results in pure or nearly pure nodes, improving classification accuracy.

20:14

Ensemble algorithms enhance decision trees by combining multiple trees,

20:19

resulting in a powerful model. One such ensemble method is random forests, where

20:25

multiple decision trees vote on the classification of new data. In random

20:31

forests, each tree is trained on the data set using the information gain method, but with random exclusions of

20:39

certain nodes or attributes, resulting in different trees with different structures.

20:44

Each tree votes on the class and the majority vote determines the final classification.

20:51

Random forests can be used for both classification and regression. They are

20:56

effective because the different trees capture different aspects of the data and their combined voting mechanism

21:03

leads to more robust and accurate predictions. Neural networks and we will

21:09

begin with the basic unit of a neural network known as a neuron or unit. This

21:17

concept is inspired by biological neurons in the human brain where neurons

21:23

or in this case units are interconnected and can be activated or inactive.

21:29

Active or inactive. Let's model this behavior mathematically

21:35

and understand how an artificial neuron works. Each basic unit

21:44

the neuron performs a mathematical operation. The calculation can be graphically

21:50

depicted as this figure here in the left. Here we depict the operation that the

21:57

neuron performs. The calculation is done inside the

22:03

computer and the unit will hold a value or will output a value. The value of the

22:11

calculation or the output of the neuron or even the

22:18

value that the neuron holds is called activation. So the activation will tell

22:24

if the unit is active or inactive. The operation is this. The neuron

22:31

receives the the the the neuron receives inputs

22:37

for example x1 and x2 and each of these inputs are multiplied

22:44

by weights w1 w2. And to that

22:50

we add a parameter or we add a value called the

22:58

bias term w0. The result of this op mathematical

23:04

operation which is depicted here as you take x1 multiply by w1 x2 ultip by w2

23:14

and sum up a bias term is then

23:19

passed through a function which we call a decision function or an activation

23:25

function. A famous activation function is the step function.

23:31

meaning that this is the g of whatever comes inside of that parenthesis.

23:38

So the step function is a function where up until zero everything is zero and

23:45

when it comes and touches zero then it jumps to the value of one.

23:53

Okay. So this is the basic building block of a neural network.

23:58

And here we're talking about a single neuron, not yet a full network.

24:04

Now let's illustrate how it this simple neuron can perform

24:11

and can learn and then perform a prediction.

24:16

Here is a data set. Here's a data set that shows that you

24:22

are allowed to play if mom or dad authorizes you.

24:29

But if both of them don't authorize you, you're not allowed to play.

24:36

So there are two dimensions to this data set or two columns. The first is if if

24:41

mom authorizes, one for yes, zero for no. The second dimension is the dad

24:48

column whether dad authorizes or no. And the third column is the target for your

24:54

supervised learning. This is what we call the or function

24:59

where play is allowed if at least one parent authorize.

25:06

So let's say that you want a neural network to learn this rule.

25:13

Learn to predict if you can play or not from the inputs X1, mom, X2, dad.

25:22

Let's use values for W1. W2 actually also receives one and W

25:32

receives minus one. And let's play with the data.

25:39

So let's first calculate this this situation where mom don't authorize

25:47

dad also don't authorize. Let's see what happens. Let's perform

25:54

the calculation. So first of all we're going to take x1 0

26:00

multiply by w1 which is 1. Then take x2

26:05

which is also zero multiply by w2 which is 1 and we're going to sum up w0 which

26:14

is a bias term. So this will result for me -1.

26:20

So whatever is inside the parenthesis for the function g is

26:26

minus1. So the result of my neural network is g of minus1. And if I'm using a step

26:34

function as g and I select minus one, you can see that the result is zero.

26:41

Exactly predicting exactly uh this situ this situation.

26:50

Now let's continue playing with the data set. Let's say for the input of

26:57

zero mom did not authorize and one for dad authorized.

27:05

What would change in this calculation is that x2 here is changing to the value of

27:11

1. And now I have 1 \* 1 - 1 = 2 the result of 0.

27:24

So now my neural my my neuron is outputting g of 0. g of0 if you go to

27:33

the function here is now one. So now I

27:39

can play and I correctly predicted this

27:44

and you can play with all the other numbers and I will do just one more situation.

27:51

This situation here. So this situation here is when both of them authorize you to go to play. So X1

28:01

authorize X2 authorized. What is changing in this calculation is that x1

28:07

now is one and your calculation

28:15

is uh going to be g of 1

28:22

right because 1 \* 1 + 1 \* 1 - 1 is 1 and this is if you go here is actually one

28:31

as well. So you're further to the right. And

28:38

here is what is happening. During training, a neural network model

28:43

learns these values, the values of these parameters which are the weights and the

28:49

bias term. I use the values here of w1 equal to 1, w2= to 1 and w

28:57

equal to minus1 because I already trained the neuron and I knew the answer. But the beauty of neural

29:03

networks is that during training the network will learn these parameters automatically just by showing this

29:12

historical data here.

29:17

Let's try and play with another function. Let's say that we want to

29:24

predict what we call an end function. So this data set now let's say you have

29:32

a two-dimensional data set as well and you're representing in f the first column in the first dimension if your

29:39

mom authorized you to play and in the second dimension if it's raining

29:45

the only way you can go play it is it's if your mom authorize you and

29:53

it is not raining. So I'm going to tell you what works in

30:01

terms of a neural network predicting this. What works is

30:07

a weight of w1 = 1, w2 = 1 and w 0 = to

30:14

minus2 and you can test it out. So let's say the first line here x1 is 0, x2 is

30:21

zero. This is x2 x1 and the result

30:28

is so x1 is 0 \* 1 + x2

30:34

0 \* 1 -2 whatever is inside the parentheses is

30:41

resulting in min -2 and g of -2 is zero exactly the right prediction.

30:51

Now let's go fast forward to this situation where you're modifying the

30:58

input one and one. So what's being modified in the calculation is that x1 is 1, x 2 is

31:08

1. And now you're outputting g of 0 because -2 + 1 + 1 is 0. And you know

31:17

that g of zero because of the step function is one.

31:23

So you're here and you will see that this

31:28

trained neural network with this parameters this set of parameters

31:34

actually can also predict a end function.

31:39

So you can see that each weight in the network contribute to the activation of

31:44

the neuron. So if it's if it's zero we say that the the output the the neuron

31:52

is inactive. If it's one it's active and the bias term term adjusts the

32:01

strength of the activation. For example, in the or function, you

32:07

could be you could be outputting one because

32:13

you're here g of zero. Your calculation was zero or your calculation was one

32:19

which is which was way to the right. This basic model with one neuron

32:27

performing a function is called a perceptron and it can generalize to any

32:33

number of inputs and uses this weighted sum followed by an activation function.

32:40

So for example, if you want to predict the height of a person, you can input the arm length, the sex, if the person

32:47

is sporty. And in the end of the day, what the perceptron is doing is is weighting the

32:55

inputs and adjusting the activation through the

33:02

bias. The step function is the simplest activation function but in practice we

33:09

use differentiable functions such as the sigmoid functions and

33:14

others. This allows for efficient optimization

33:19

and it has mathematical reasons to use this kinds of functions that are smooth

33:26

and not such a uh and non and not the ones that are not smooth as a step. So

33:32

the function that we you we use is kind of a a sigmoid function like this

33:41

and not only mathematical mathematically it's good but it gives more of a

33:48

probabilistic interpretation to the output. So say here you were you're more

33:55

you're even more allowed to play because you have mommy and daddy leting letting

34:01

you play. So you can see that the the threshold of your activation

34:12

um is probabilistic.

34:18

Neurons are organized into layers. Each neuron in one layer connects to all

34:24

neurons in the next layer. So

34:29

here is an example of what we call the multi-layer perceptron perceptron MLP.

34:36

So the weights in the activation of the neurons connected to um the other

34:43

neurons in the following layer are adjusting the activation and the

34:50

threshold of that particular neuron. So let's say that I'm this neuron. I'm

34:55

being activated by the influence of the the neurons that are connected to me and

35:02

the weights that of that connection plus my W0 which is

35:09

um adjusting the activation threshold.

35:15

An MLP a multi-layer perceptron can learn more complex nonlinear decision boundaries than a single neuron.

35:23

Of course it has more parameters.

35:29

Sometimes it can we will see that for deep learning this will get to billions

35:34

of parameters and these parameters will be adjusted in

35:40

the training stage. Neural networks can also learn from

35:47

images and it's a very good way to understand the horistics of how neural

35:52

network works. Consider a classification of handwritten

35:58

digits from 0 to 9. Traditional models like logistic regression require

36:05

carefully selected features to learn.

36:11

For example, let's say that we are learning how to class a logistic regression to classify if that a digit

36:18

is number one. When you're fitting the squiggle of logistic

36:24

regression, meaning that you're learning the parameters, it would be very

36:32

informative if you could select the pixels where everyone

36:38

is drawing a one. So suppose an image of the digit one and

36:45

it's activating this particular pixels. You would want to get the value of this

36:53

pixels as features to the logistic regression.

36:59

But if someone writes one in a different position, the model will fail because

37:06

the one is now activating another set of features.

37:13

So the logistic regression was

37:18

was trained to analyze this features here. The

37:24

columns of it. The data set you use to train a logistic regression would be

37:29

doing this analyzing if this particular pixels are one. But if a translation

37:35

happens the features are not that particular features are not informative anymore.

37:43

So it is very complex to make a digit detector for example.

37:50

And the beauty of a neural network by contrast is that they with because of

37:56

their comprehensive connectivity they can take raw features

38:02

and pass their values through several layers of neurons.

38:07

And the final layer will output and predict if that is number one or not

38:14

without specifically selecting the features

38:20

without specifically doing feature extraction.

38:26

Actually neural networks were a breakthrough in eliminating the need of previous feature extraction.

38:33

Obviously that feature extraction is still a great process to help models, but neural networks rely on all the

38:41

features and automatically extract information.

38:50

So neural networks and deep learning which we will talk about soon eliminates or

38:58

diminishes the need for feature extraction. Here is a good example to

39:03

understand that in earlier models, feature extraction was essential. In

39:09

X-ray diagnostics, for example, experts manually identified features such as

39:15

angles and gradients of X-rays.

39:21

So the the domain experts extracted this features and then trained

39:29

for example models such as logistic regressions to identify if a person had

39:36

lung cancer or not. With deep learning and neural networks,

39:44

the raw pixels of the X-ray can be fed directly into the network. The network

39:51

learns relevant features by itself during training and eliminates greatly

39:58

the need for manual extraction. This ability to learn from raw data is

40:06

one of the key strengths of deep learning. So what is deep learning?

40:16

Learning is the definition of large

40:21

neural networks that has several hidden layers, several layers of neurons.

40:31

Usually deep learning have several layers and

40:37

thousands, millions or billions of parameters. So this is a deep learning

40:43

architecture. It's a multi-layer perceptron and the inputs for example

40:51

are connected to all this is the weight of input one to neuron one of layer 1.

41:00

This one is input the weight of input one to neuron 2 and

41:06

so on. But actually a deep neural network or a deep learning network is

41:14

the one that have several hidden layers and the number of these parameters and

41:20

biases are immense. So not only you have the

41:25

weights that connect but you also have the biases for each one of the neurons.

41:34

Now neural networks can also perform multiclass classification.

41:40

For example, we were talking about digit classification to classify digits 0 1 2

41:47

3 and so on. The output layer would contain

41:52

one neuron for each class and the neuron with the highest

41:59

activations determines the predicted label. say that this is 0.1

42:08

fi 4 0.3 and this is 0.79.

42:13

So the the recommendation or the prediction of

42:19

the network is that the digit the digit is number two. This being number one,

42:26

number one, number two, number three up until our x is zero, one, two, three, up until

42:33

nine. So it's number one. The prediction is number one.

42:39

So each connection between neurons has an associated weight as we uh discussed

42:46

and a bias term. Now multi-layer perceptrons and neural

42:54

networks cannot only perform classification tasks but also regression

43:00

by using a single output neuron. So this architecture is used for

43:08

regression. The single output neuron returns a

43:14

continuous value. For example, given inputs like arm length, biological sex

43:20

and uh a flaticism, the network can predict height.

43:28

For that, the output activation function for the last neuron should be a

43:35

linear function so that the network can perform the linear regression.

43:41

For example, the last linear the the the last neuron will hold the predict value

43:48

whatever it is. For example, if the height is 180

43:54

180 cm. So you have the inputs

44:00

such as biological sex,

44:05

arm length and so on. You could have several.

44:10

And the the G function of this last neuron is a linear function. It has to

44:17

be a linear function because this can assume any continuous value.

44:23

So you cannot have a g like this

44:29

because this goes the the output of the G is only between zero and one. So

44:35

whatever calculations is being done here inside the

44:41

the the neurons in the hidden layers then the last neuron should if the

44:47

calculation is 180 the ne the last neuron will output exact a horistic

44:52

understanding of neural networks. I will use an example given by Jeffrey Hinton

44:59

the most recent physics Nobel Prize and godfather of AI.

45:04

So let's start off by describing how it would work for recognizing objects in

45:11

images. Let's suppose you want to uh say

45:17

classify whether there is a bird in the image

45:22

and let's suppose that the birds uh roughly clear in the images if they are

45:29

there. So you can think of an image. Let's suppose it's 100 by 100 uh pixels. So

45:38

100 by 100 pixels little squares. That's 10,000 pixels. Each pixels is three

45:45

colors, red, green, and blue. So that's 30,000 numbers.

45:52

And in computational terms, you have to take that 30,000 numbers and output a

45:57

single numbers. A single number that says if it's a bird or not. So you would have 30,000 inputs and a single output

46:06

neuron. People for many many years tried to do that but it's very hard.

46:16

So how would we use a multi-layer

46:21

perceptron a multi-layer neural network to do this? And Hinton started by

46:28

telling us how he would wire up the neural network by hand.

46:34

So what you do would be this. Uh you have a layer of the first layer

46:42

would be a feature dete a feature detector and it could detect features right like

46:50

for example uh edge in the image.

46:56

Uh so a typical feature detection an edge detector might have a big positive

47:05

connection uh strength coming from a horizontal

47:11

row of pixels. Right? So here and a uh and a negative

47:18

connection a big negative connection oop sorry to

47:23

go back to the neighboring row of pixels.

47:31

So and no connection strengths anywhere else. So if both call uh if if if um

47:40

if both uh rows of pixels are bright, you'll get

47:46

a big connection here and a negative connection there. And this neuron is not

47:54

activated. Now,

48:01

if one of these rows are bright

48:06

and the other one is not bright,

48:12

you will have an edge detector and this neuron will light up

48:20

and this neuron says, "Hey, I found the thing I like," which is an edge. You

48:26

could have several different feature detectors and you can think about this as gazillions of neurons.

48:34

So you can have all of these feature detectors in several parts of the image.

48:42

So he just told us how to wire up by hand using positive and negative

48:47

weights. So you would have found a little

48:52

horizontal edge and you can think about detecting

48:59

several neurons detecting different edges in different locations of the image. That will be the first layer.

49:06

Then the second layer you could put the weights of of some

49:12

neurons to activate when

49:19

uh you know like found an edge like this edge here that we're zooming here is the

49:24

horizontal edge and another edge here and it would activate

49:30

um if they had a fine angle between them. So

49:36

it would say hey this neuron would say hey maybe there's a beak here and all

49:44

sorts of other things. uh but it might be just a beak a beak.

49:50

So you can from edges to

49:55

for example you can go from edges to a whole bunch of forms detectors.

50:02

You can go with peak detectors, eye detectors and you layer by layer keep detecting

50:09

slightly more complicated things. Um,

50:15

so you have a you could have a potential beak uh in in a uh spatial ratio with an

50:23

eye that is potentially good. And so you have a potentially uh uh this neuron

50:31

here is potentially giving you that this is a bird and you would you would be output

50:38

outputting a bird. Uh so in the next layer that would be um

50:45

a wing detector a foot. So here is a here here is a a good ratio of foot a fe

50:54

a wing and voila there's a bird. Now instead of wiring it all by hand we

51:02

could imagine trying to learn it all. So

51:07

instead of wiring by hand, you want would start with random

51:13

connections right between the neurons. And so when you input the um image of a

51:19

bird and you go forward through this layers, it will be

51:26

bad prediction. It will com uh completely behave

51:31

randomly. Let's say that it outputs.3.

51:38

Um, it's going to say it's not a bird and but I'm sure it's a

51:45

bird, right? So, there's an error there. How can I change then you can ask a

51:52

question. How can I change all the connection strengths in the network so that I you know instead of saying.3 I

52:01

can say 0.31 it's a bird like to tweak how can I tweak the weights to minimize

52:08

my error and you

52:14

you take that difference and you send that difference backwards through the

52:20

network. And then you use calculus uh to tweak to compute a change or a

52:30

tweak in every single connection of the network to make the network predict

52:36

better the bird. And you can you just keep going like

52:43

that uh with lots and lots of birds and non-birds and eventually you'll discover

52:51

that um that this network is predicting and

52:56

performing the classification um well

53:02

and we will talk about how to tweak the weights

53:08

uh next. ly that number.

53:16

The learning algorithm used to train deep learning models is called gradient descent.

53:23

So we will talk about gradient descent and back propagation.

53:29

So let's talk about gradient descent. It is an optimization algorithm that iterative iteratively adjusts the

53:36

model's parameters, the weights and biases to minimize the error.

53:42

Minimize the error between the predicted and the actual output from the training

53:49

data set. The data set where you have examples and the target values

53:54

available. Suppose you're training a neural network to recognize a bird. Initially the

54:01

weights are random. If the network output one should output one for a bird

54:06

but instead outputs.3 the error is 1 minus.3

54:14

meaning 7. We want to red reduce the mean squared

54:21

error of that. So we want to reduce 1 minus.3 squared.

54:27

The algorithm then adjusts the weight slightly. After the adjustment, the error becomes

54:34

smaller. If we show another sample of a bird,

54:40

for example, from 1 - 3^ 2 to 1 -6^

54:48

2. This process is repeated gradually

54:54

reducing the error. So gradient descent in this graph to the

55:00

left this is um loss not cost loss.

55:05

Gradient descent can be visualized as a hiker descending a valley to find the

55:11

lowest point or a ball rolling downhill to reach the minimum of this loss

55:17

function. Each step of the training

55:23

um algorithm will go in the direction of the

55:30

steepest descent. That's why it's called gradient descent

55:36

because the gradient is a mathematical entity computed to show how can we tweak

55:43

the weights and minimize the loss in the

55:49

steepest steepest way or more optimized way.

55:54

Each training step is called epoch. So the epoch is a training step

56:04

meaning that I start my network with random val

56:11

values. I show the examples. I tweak my weights

56:19

and I will show the examples again. the arrow will have

56:26

decreased. This is when at epoch

56:34

it is possible that neural networks get stuck in what we call local minimum

56:42

and don't reach the optimum value of the model.

56:47

So to mitigate this training is often repeated multiple

56:53

times with different random initializations. So if you initialize your weights here

56:58

randomly, you will get stuck in the local minima because you will have

57:04

minimized and gradient descent will always minimize the loss because here

57:09

you will start the loss will start getting bigger. Your

57:14

algorithm will stop here. meaning that the weights that you should use that you you trained is this weights

57:22

obtained in this step. But what the way to overcome this is to

57:30

do several trainings initializing your weights in different

57:35

locations or with different random weights.

57:42

Starting from different points increases the chances of reaching the global minimum meaning that you would have an

57:49

optimal model with the best parameters to the prediction.

57:56

The specific implementation of gradient descent in neural networks is called back propagation.

58:03

And the way the process works is this. Run the neural network with current weights and compute the error.

58:12

So let's say that you want to try to to classify a canc lung cancer with all

58:20

these parameters. Run the network. Compute the error. Use

58:27

the gradient this the gradient calculation to adjust the weights that

58:35

contribute to the error. So adjust the weights of this f the last layer. You

58:41

start with the output layer adjusting the weights and you move backwards

58:46

adjusting the weight the weights updating each sets of set of weights.

58:53

This step-by-step reverse update is called back propagation. We've discussed multi-layer perceptrons

59:00

deep learning with a feed forward architecture. Um that means that the feed forward ar

59:08

there's lots of neural network architectures. These are just uh some

59:15

shown here. But the feed forward architecture is where the data flows from input with

59:22

output. So one neuron activates the next

59:28

neuron to the right or to the other layer.

59:33

But other important architectures include convolutional neural network is this architecture here.

59:40

We're not going to go into details but it's used for image classification and processing. So convolutional neural

59:48

networks are more effective than traditional MLPS for image tasks. You can imagine that the example of the bird

59:55

classification is a vanilla kind of old technology example because

1:00:05

the new technology used in convolutional neural networks use filters or

1:00:10

convolutions to detect patterns such as edges, corners, textures.

1:00:18

So CNN's as they're called are now the standard for visual recognition tasks.

1:00:24

The other important architecture is recurrent neural networks or RNNs.

1:00:29

Design this architecture here. They're designed for sequential or time series data forecasting

1:00:37

such as stock price forecasting, sequence uh forecasting, word

1:00:43

forecasting um and other types of trends forecasting.

1:00:51

In recurrent neural networks, the state of a neuron at a previous time step

1:00:56

influence the current activation. This creates a form of memory that

1:01:02

allows the network to model temporal dependencies. So here's an example. Let's say that you

1:01:08

want to forecast the stock price. Suppose you have the stock prices over

1:01:15

time. X1, X2, X3 and so on. These are the prices

1:01:21

of the stocks. And you want to do is from X1 you want

1:01:26

to predict X2. from X2 you want to predict X3 from X3 you want to predict

1:01:32

X4 and so on. So think about the the architecture of

1:01:40

the recurrent network has this additional connection between

1:01:46

neurons from layers subsequent layers changing the

1:01:53

activation of previous neurons. So let's let's say that you input X1 to predict

1:01:59

X2. You input X1, you will predict X2,

1:02:05

right? And say that I am this neuron here.

1:02:10

When you input X2 to predict X3, what is going to happen to me is that

1:02:17

not only X2 will influence and and all the the previous uh neurons that are

1:02:24

connected to me are going to influence my activation. but also the activation of the this

1:02:31

neuron here when the input was previous when the input was previous in time when

1:02:38

the input was x1. So you're kind of creating a memory in

1:02:45

the network. So each prediction uses the current input and the past internal state making

1:02:52

this very suitable for time series forecasting.

1:02:58

where the connections between time steps uh store

1:03:04

uh past information. Now that we've seen some models, let's

1:03:11

talk about model complexity. Model complexity refers to how sophisticated a machine learning model

1:03:18

is in its ability to capture patterns in data. A more complex model has more

1:03:24

parameters and can learn more complicated relationships like a neural network with many layers and billions of

1:03:30

parameters. A simpler model such as linear regression has fewer parameters.

1:03:38

Trade-off in complexity. Finding the right level of complexity is

1:03:43

crucial. A too simple model might not capture important patterns in the data.

1:03:48

A too complex model might fit the training data perfectly including the noise and thus fail to generalize.

1:03:56

For example, a linear regression might miss a curve in the data. A polomial

1:04:01

regression of appropriate degree can fit the trend better. A high order polomial

1:04:07

can perfectly match all training points but this often means it is overfitting.

1:04:13

Although such a complex model fits the training data exactly, it performs poorly on new unseen data. This is

1:04:21

because it has learned noise rather than the actual pattern. Noise in the data.

1:04:28

Suppose you measure a sinosoidal pattern with sensors. The true underlying data

1:04:34

follows a sine wave. But because of measurement error and sensor noise, the

1:04:39

observed data has small deviations. A well-fit model should capture the

1:04:45

underlying sign trend, not the scattered deviations. A very complex model might trace the

1:04:53

scattered points precisely resulting in a curve that is no longer sinosoidal.

1:04:58

This is overfitting. When a new input comes in, the overfit model will fail to generalize,

1:05:05

predicting an incorrect value. Underfitting and overfitting.

1:05:10

Underfitting happens when the model is too simple to capture the trend. Overfitting happens when the model is

1:05:17

too complex and fits the noise. In both cases, the model performs poorly on test

1:05:24

data, even if it performs well on training data. Generalization.

1:05:31

What we want is a model that is complex enough to capture the true structure of

1:05:37

the data, but not so complex that it memorizes noise and generalizes well to

1:05:43

unseen data. The model overfits. This is like a student who memorizes instead of

1:05:49

understanding. They perform well on homework, training data, but poorly on

1:05:55

the exam test data. The moment test loss begins to increase is when

1:06:00

generalization power is lost. Stopping training at this point is ideal

1:06:06

which is why techniques like early stopping are used. Uh I'll I'll do

1:06:11

something here. So when when when you're training

1:06:16

um when you're when you're training your uh your model and you have your loss

1:06:22

function and you want to minimize the loss function for example

1:06:28

um what what you need to do is this. you you you you keep training and your loss

1:06:34

function for example your mean squared error in a linear regression or in a regression um or a quadratic regression

1:06:41

polomial regression whatever is decreasing right so this is the the loss

1:06:47

in the training data and you want it as small as you can but if your model is

1:06:53

too complex it will be so so small that you're actually fitting the noise

1:06:59

so Um we what we want is that your uh

1:07:04

your your if you if you uh think about measuring

1:07:10

the mean squared er error on on the um

1:07:16

on the the test data you want to this is the test data. So if new data comes in,

1:07:24

we we discussed about the test data and the division of the the data set, you hold out part of your data set to test

1:07:31

your model, but you want that to after training, you want the the the loss for

1:07:37

the training data to be small. Okay.

1:07:43

Um what happens when your model overfits is that actually you're learning the noise.

1:07:51

So the loss uh the the if you're testing in a test data at some point if you keep

1:07:58

learning you're you're learning too much. It's like a student that is memorizing not understanding the data.

1:08:06

All right. So the the loss in the test data starts to starts to uh actually

1:08:13

increase meaning that your generalization power is lost. Okay. So

1:08:21

one of the mechanism to avoid overfitting is actually splitting the

1:08:28

data the into three three parts training validation and test. So you have a hold

1:08:35

out called the test data. Okay. And you you keep that uh to to test several

1:08:43

models and to choose the best one, right? Uh giving a metric. Now what you

1:08:48

do while you're training you keep assessing how your your loss is u

1:08:55

minimizing and you you keep a a uh you hold out a a

1:09:00

a part of the data set called the validation data set. So this data set

1:09:08

you will apply your model to the data set and you will see well you know it's

1:09:14

kind of a test set but because you use it to stop the training

1:09:20

um it's it's it's not an unseen data right so remember you always have to

1:09:26

test your model into this okay this test the unseen data this although it's not

1:09:34

used for training. Validation is not used for training. It's used to it's used uh to to kind of guide the

1:09:44

training. Okay. Kind to guide the training. So it

1:09:49

uh it it it will actually uh weight in the the the values that you're choosing

1:09:57

for your parameters. Okay. So uh the in the validation set the the loss

1:10:06

functions as as you as you train so this is the train time

1:10:14

or we will we will discuss the epochs in in in the in terms of neural networks

1:10:21

uh I'll I'll I'll bring this concept just in a bit but the validation error

1:10:27

drops drops drops drops drops And as soon as starts to get up, right, because

1:10:33

you're starting to overfit, you halt the train. You stop the training here. So

1:10:39

you stop the training, right? So that when you have the test

1:10:47

set, when you have the let me choose another color. when you have the test

1:10:53

set uh in the test set because you stopped when it was starting to kind of

1:10:59

get um it's start it's starting to overfit it will still be uh it will still be a

1:11:07

good model. So because you're stopping the training, you're using this data set

1:11:12

kind of to halt the the values of the weights here of the parameters of your

1:11:19

model here. It's uh this this validation although not using to training it is

1:11:25

kind of used to get your final model. So you still have to use an completely

1:11:30

unseen uh model uh data for the test. Okay. So let me introduce the concept of

1:11:37

epochs. Uh while we're training neural networks, we present the data for neural

1:11:44

networks several times and then adjust the weights via back propagation. Um and

1:11:53

each time we show the data. So each this each time we show the data, this is

1:11:59

epoch one. I have a loss. Then I adjust my weights of all the um of the um

1:12:08

of the layers. Then I show the data again and I adjust my weights again. So

1:12:14

every little twick that I do in my weights during training is called an

1:12:20

epoch. Okay. Now validation is a very common this

1:12:27

split here during training is very common training validation and test and

1:12:33

actually a a um there is a algorithm

1:12:41

uh called uh cross validation that uh that um kind of gets this concept a

1:12:50

little bit better. uh and it it um it does it uh to kind of

1:12:58

um decrease variability. So let's say that you've divided your training set,

1:13:05

your validation set and your test set and you had a result of a loss, right? So let's say that your loss

1:13:12

um after training uh so uh

1:13:17

uh sorry that you have a metric evaluation metric of 98%

1:13:22

of of goodness right your model is 98% good we will discuss measures of

1:13:28

goodness in a bit uh and then you if you change this uh randomly um um

1:13:37

the samples that are inside each of these groups groups you will have a 96% and then you will have a 94% and if you

1:13:44

keep doing you will have a 99%. So in average right with different holdouts

1:13:50

with different um um kind of divisions you will have

1:13:56

a um a metric that you can use right there is an algorithm called cross

1:14:02

validation that improves the evaluation of models by reducing variability.

1:14:07

Suppose you divide your data into training, validation, and test sets and obtain a model accuracy of 98%.

1:14:15

If you randomly reshuffle the samples among these sets, you might get different results, 96%, 94%, or 99%.

1:14:24

Cross validation provides a more reliable average performance metric across different splits. Although it's

1:14:31

called cross validation, a more accurate name would be cross testing.

1:14:37

It doesn't necessarily require a validation set. When training and testing, you face two issues. You're not

1:14:45

using the full data set for training because you're holding out the test set. And if you allocate more data for

1:14:50

training, the test set becomes smaller. Cross validation addresses these issues

1:14:55

by dividing the data set into equal parts or folds. For example, with five

1:15:01

folds labeled 1 through five, you perform five rounds of training and testing. In each round, four folds are

1:15:08

used for training and one fold is used for testing. You rotate the test fold in each round, so each part of the data is

1:15:15

used once for testing and multiple times for training. This approach ensures that

1:15:20

every data point is used in training and in testing, providing a more robust estimate of model performance. This is

1:15:27

called five-fold cross validation. At the end, you average the performance metrics from each round to obtain a

1:15:34

final evaluation score. You can optionally include a validation set in each fold for tasks like early stopping

1:15:41

to avoid overfitting, but it's not required. Cross validation can be used with or without validation sets. It

1:15:48

helps address the challenges of data partitioning and provides a better understanding of the model's generalization. For example, if you have

1:15:56

200 data points and divide them into five folds, each fold will have 40 data points. In each round, 160 data points

1:16:03

are used for training and 40 for testing. Across the five rounds, all 200

1:16:08

points are eventually used for both training and testing. Suppose your performance metrics in each fold are

1:16:14

98%, 95%, 99%, 89%, and 71%.

1:16:21

Averaging these results gives a more reliable measure of your model's performance rather than relying on a

1:16:26

single random train test split. We're talking about model evaluation and

1:16:31

metrics. For classification models, we use several important metrics. For regression, we use others such as RQ

1:16:39

squared and mean squared error MSE. MSE can be used as both a loss function

1:16:46

during training and an evaluation metric after training on the test set. In

1:16:51

regression, you have a numerical target and an attribute. You're fitting a line

1:16:56

to the training data. During training, you minimize the squared distances from

1:17:01

the line to the points. When a new data point arrives in the test set, you

1:17:06

compare the predicted value from the line to the true value. The squared difference becomes part of the

1:17:12

evaluation. The lower the average MSSE across test points, the better the model.

1:17:19

R 2 R 2 is another metric for regression. It represents how much of the variation in the target variable is

1:17:26

explained by the predictor. For instance, if you're predicting mouse weight from mouse height and get an R

1:17:32

squar of 81%, it means 81% of the variation is explained by height.

1:17:39

In contrast, if you predict mouse weight based on time spent sniffing a rock and

1:17:44

get an R squar of 6%. That means sniffing time explains only 6% of the

1:17:50

variation in weight. So, it's a poor predictor. Now, for classification metrics,

1:17:57

accuracy, F1 score, and area under the rock curve, A C O.

1:18:04

These metrics help evaluate the performance of classification models. In classification, assume you're

1:18:10

identifying whether someone has a disease. The positive class one represents sick and the negative class

1:18:16

zero represents healthy. If you predict someone is sick and they are sick, that's a true positive. If you predict

1:18:23

sick but the person is healthy, it's a false positive. If you predict healthy and the person is healthy, it's a true

1:18:29

negative. If you predict healthy but the person is sick, it's a false negative.

1:18:35

This structure forms the confusion matrix. In statistics, false positives

1:18:40

are called type one errors and false negatives are type two errors. Accuracy

1:18:46

is defined as the number of correct predictions, true positives plus true negatives, divided by the total number

1:18:52

of predictions. A high accuracy is good, but it can be misleading in imbalanced

1:18:58

data sets. For example, if only one in 100,000 people has a disease, a model

1:19:03

that classifies everyone as healthy will have very high accuracy, but it will miss the disease cases. To address that,

1:19:11

we use the F1 score, which combines precision, how many predicted positives are correct, and recall, how many actual

1:19:18

positives are correctly identified. F1 is more balanced and sensitive to

1:19:23

imbalanced classes. You calculate F1 from the confusion matrix. It penalizes

1:19:29

false negatives and false positives based on the context. Let's talk about thresholds.

1:19:35

Classification models often output probabilities. By default, values above

1:19:41

0.5 are considered class one and below 0.5 are class zero. Adjusting the

1:19:49

threshold affects classification. For example, lowering the threshold may reduce false negatives but increase

1:19:57

false positives. Visualizing the impact of threshold changes is helpful. Some

1:20:02

resources offer animations to show how the threshold affects outcomes.

1:20:08

A key metric for comparing thresholds is the ROC curve, receiver operating characteristic curve. It plots the true

1:20:15

positive rate against the false positive rate at different threshold values. The true positive rate is the proportion of

1:20:21

correctly identified positives out of all actual positives. The false positive

1:20:26

rate is the proportion of incorrectly identified positives out of all actual negatives. We want a classifier that

1:20:34

achieves a true positive rate close to one and a false positive rate close to zero. The AU area under the curve of the

1:20:43

ROC plot quantifies this performance. A perfect classifier has an AU of one. If

1:20:49

two classifiers have different ROC curves, the one with the higher AU is better. During training, engineers can

1:20:57

also penalize false negatives or false positives in the loss function. For example, in rare disease detection,

1:21:03

false negatives are costly, so models are trained to reduce them.

1:21:09

This concludes week two. For the prototyping activity, there's a 15minute video introducing Weta, a software tool

1:21:16

for machine learning. Students can choose between WKA and Orange, another tool. A five-minute video introduces

1:21:22

Thinkable Machine, a tool to run a classification on the go. Students will use these tools to build prototypes and

1:21:28

solve classification or regression problems using different models. Evaluation of the prototypes can be

1:21:34

discussed later. Okay, so that's for week two. um we're talking about just we could add

1:21:42

a um summarization section saying oh we we learned about supervised

1:21:49

learning uh you can do classification regression and forecast so classification is discrete regression is

1:21:56

continuous uh forecast is uh um for t

1:22:02

time right uh series and so on um we learned about how to how to

1:22:09

cross cross validation kinds of evaluation just wrap up everything and I

1:22:15

there are three main cases for the tutorials for the prototypes. So one is

1:22:20

for classification the other one is for regression the other one is for forecasting. So I will

1:22:27

here just pretend uh like I I will just show how I

1:22:34

think the tutorials would work. So here is the mini case. This is Harshida. She

1:22:39

wrote like a a very introductory level use case. So stopping the turn before it

1:22:45

starts building an AI retention assistant. You've joined the customer strategy uh team at telco, a midsize

1:22:52

telecom provider uh facing uh intense pressure from new market entrance and

1:22:58

shrinking customer loyalty and the the head wants to an AI powered

1:23:05

retention assistant that can identify customers at a risk of leaving, right?

1:23:11

uh your manager uh hands you a data set and asks you to develop a first version

1:23:18

of a prototype using no cool no code tools. So this is your role, this is

1:23:23

your business objective and so on. So the I uh the thing that I envision is

1:23:30

having a tutorial on WA N or even see if SAS has some kind of no tool code for

1:23:37

machine learning and then uh

1:23:43

um let them try and then just do like a uh

1:23:49

um rubric for it.

1:23:58

Hey everyone. Today I'd like to make a quick video that I hope will be concrete and useful for you. It's about the very

1:24:04

first machine learning library I ever tried and it's called Weta. What's great is that Weta comes with a guey that

1:24:10

makes it easy to visualize your data sets and train and compare different classifiers. And this is a really handy

1:24:16

tool to have while you're learning ML. I'll give you a quick walkthrough of how to use Weta from installation all the

1:24:22

way to running experiments and show you some of what it can do. I'll demo training models on two different data

1:24:28

sets. First, we'll predict if a patient has diabetes based on attributes like their glucose levels. And next, we'll

1:24:34

predict if a Congress person is a Democrat or Republican based on how they voted on different bills. I'll also show

1:24:40

you how to evaluate the results of these experiments and how to do things like feature selection to discover which

1:24:46

attributes are important. Okay, let's dive right in. The first thing we'll do is download and install Weta. And what's

1:24:53

neat is that it comes as a nicely packaged application you can run on Mac, Windows, or Linux. There's also a Java

1:24:58

API. Here, I'll download and install Weta. And now I'll start it up. There

1:25:04

are different interfaces, and we'll use the Explorer. There's a lot on the screen, but don't worry about it. You'll

1:25:09

get a feel for how this works in a moment. The first thing to do is open a data set. So, we'll hit open. And now

1:25:15

would be a good time to download one. You can find a bunch of prepackaged data sets on this page. And we'll start with

1:25:22

the UCI repository. It contains about 37 problems. And when you download it,

1:25:27

you'll get a jar. Now, you might be familiar with these if you're a Java developer, but if not, don't worry. You

1:25:32

can treat them as a zip. Here, I'll unzip it. And now we can see a directory of data sets. Let's return to Weta and

1:25:39

open one of these up. And we'll start with diabetes. All right. What do we see here? Let me walk you through it. First,

1:25:46

let's learn about the data set. At the top, you can see there are 768 examples or instances and nine attributes or

1:25:53

features. The best attribute to start with is class or the label we want to predict. And usually in Weta, that's the

1:26:00

last attribute in a data set. Clicking on that, we can see a histogram. The blue column on the left shows the number

1:26:06

of people who tested negative for diabetes, and the red column on the right shows those who tested positive.

1:26:12

Now, let's look at the attributes we'll use to predict if a patient has the disease. The descriptions here are

1:26:17

pretty short, but we can open up the data set in Sublime or your favorite text editor to learn more about what

1:26:22

they mean, as well as how the data set was collected. Now, Weta data sets come in an ARF format, and this is just a CSV

1:26:30

with some metadata included at the top. Scrolling down a bit, we can see a description of the attributes. And the

1:26:36

first tells us the number of times a patient was pregnant. And the second tells us their plasma glucose for

1:26:42

diabetes. I imagine one of these is more predicted than the other. Let's see if Weta can tell us that, too. Back in the

1:26:49

GUI, let's click on plasma. And what's cool is you can see a histogram of how different values correlate to the class

1:26:54

we want to predict. Recall that blue is negative and red is positive. And right off the bat, we can see this is a useful

1:27:01

attribute, meaning that if plasma is low, say below about 100, then it's unlikely the patient has diabetes. Most

1:27:09

of these values are blue. Whereas, as the value increases, it's increasingly likely that a patient has the disease.

1:27:16

Now, let's look at pregnancy. And to me, this doesn't look like a strong correlation. There may be one here, but

1:27:22

it's less obvious. Now, here's where things get really interesting. I have so much to show you,

1:27:27

but I want to jump in and classify this data. So, let's head over to the classify tab. There's a whole bunch of

1:27:34

built-in classifiers. Let's start with a decision tree, and J48 is one type of

1:27:39

tree that does pruning. We'll hit start, and bam, we're done. We just trained a

1:27:44

decision tree on the diabetes data set. Let's say we also wanted to train a linear classifier like logistic

1:27:50

regression. To do that, we can go into functions, logistic, and hit start. And there we go. This is great because we

1:27:57

can flip back and forth between the two and compare the results. There are many types of classifiers in Weta if you're

1:28:04

interested. Everything from naive bays to basic neural networks. Now, let's head back to the tree and see the

1:28:10

results. And there's a lot of information on the screen. So, let me walk you through it. Let's scroll to the

1:28:16

top and start there. First, you can see the train. As always, you read it from the top down. It's telling us to start

1:28:23

by looking at the value of plasma and this happens to be the most predictive attribute in the data set, but we'll

1:28:29

return to that later. Scrolling down, we can see the accuracy was about 73%. But

1:28:34

what exactly was the accuracy evaluated on? Well, here you can see Weta gives you three options. The first would be to

1:28:42

compute the accuracy on the training set. And if we do that, of course, it'll be higher. It goes up to 84% because

1:28:48

we've tested the tree on data it's already seen. Of course, this isn't useful in the real world. As always in

1:28:55

machine learning, our goal is to generalize from the training data. Ideally, we want a model that performs well on data it's never seen before. So,

1:29:02

how can we know if it does? Well, one way to simulate that is to have a separate test set. Basically, you can

1:29:08

divide the diabetes RF file into two separate files, one for training and one for testing. Use the testing file only

1:29:15

rarely to see how well your algorithm performs. Another thing we can do is use cross validation. And this sounds fancy,

1:29:22

but all it does is iteratively divide the data set into two chunks. The larger chunk is used for training and the

1:29:28

smaller one is used for testing. We train a model, evaluate it, and repeat this process a number of times, then

1:29:34

average the results. And Weta automates this process for you. Now, let's look more closely at the evaluation. You'll

1:29:42

see stats like these for any classifier you train. And importantly, notice that there are metrics like precision and

1:29:48

recall in addition to accuracy. Why? Well, although accuracy is the

1:29:53

first thing we think of when evaluating a classifier, it doesn't always tell us the whole story, especially in data sets

1:30:00

where one class is rare. For example, let me show you how to write a 99% accurate classifier that doesn't really

1:30:06

do anything at all. Imagine you're writing a program to assist a doctor like we're doing with this diabetes data

1:30:12

set. Now, imagine that the disease we want to predict is very rare. Say only one person in 100 is sick. So, how can

1:30:20

you train a 99% accurate classifier without using any ML at all? Well, it's

1:30:25

simple. It turns out you can write one line of Python. Deaf diagnose return

1:30:30

healthy. Because most people are healthy just by predicting that everyone is or

1:30:35

the majority class, we're 99% accurate but not useful. Our model will always be

1:30:41

wrong when the patient is sick. That's why when we evaluate classifiers, we have to look at accuracy both on the

1:30:48

positive and negative cases. And there are different ways to do this. A confusion matrix like we see below is

1:30:54

one of my favorites and you can find a link in the description to learn more about it. Now on to another topic.

1:31:00

Imagine we had asked the question which attributes in the data set are important. Here we don't want to train a

1:31:05

model. We just want to explore the data. There's a technique we can use called feature selection which can help. And

1:31:12

the first thing we can do is rank the attributes by their information gain. Let's head back to the diabetes data set

1:31:18

for an example. We can hit filters supervised attribute attribute selection. Then select info gain as the

1:31:25

method and ranker as the search. And when we run this, the attributes will be sorted

1:31:31

by how useful they are to predict the label. If you could know just one feature from the data set, you'd

1:31:36

probably want to know plasma. If you could know two things, you'd probably also want to know mass. But keep in

1:31:42

mind, we haven't done a search. It's possible that these two attributes are not the best combination. There are

1:31:48

other methods of selecting attributes like this if you want to find the best subset to use.

1:31:54

An exhaustive search can be computationally expensive, though. Now, let's look briefly at the vote data.

1:32:00

I'll move faster this time because training and evaluating a classifier uses the same pattern. This data set is

1:32:06

from the US Congress and the goal is to predict if a representative is a Democrat or a Republican based on how

1:32:12

they voted on different bills. As before, let's start with class. Here, blue are Democrats and red are

1:32:18

Republicans. And this data set was collected back in the 1980s. So, these ratios are different than they are

1:32:23

today. Each attribute describes how a Congress person voted on different bills, and many are predictive. If you

1:32:30

flip through them, you'll see that many votes are divided along party lines. As before, you can read details about the

1:32:36

bills in the ARF if you're interested. And if we train a tree, these are the rules you can use to predict the

1:32:41

political affiliation of a member of Congress based on their voting history. It's still amazing to me how easy this

1:32:47

tool is to use, and I find it helpful on a regular basis. I usually begin my experiments with a decision tree to

1:32:53

learn more about the data and as a sanity check for a baseline classifier before I move on to more complex models

1:32:58

like neural nets. Okay, I hope this was helpful and that Weta makes it easier for you to learn ML. Thanks everyone and

1:33:04

I'll see you next time. [Music]

1:33:15

We would generate a um kind of a a rubric. Um this is the the telco

1:33:22

customer churning data set. We opened in Wacka Explorer.

1:33:28

Um and you can see here all the attributes or columns of the data set. If you open the CSV

1:33:35

the of the data set, you will see that the data set has

1:33:43

customer ID as a column, gender, senior, citizen, partner, dependence, tenure,

1:33:50

uh, and several columns and the last column is the target to do supervised

1:33:55

learning meaning churn. Okay. So we have um a huge huge data set

1:34:07

um and you can open it in Excel and you can open also in in Wacka and you can

1:34:13

see that uh Wacka has um

1:34:18

uh a summary. So we have instances meaning the number of roles is 7,043

1:34:25

with 21 attributes. Um and you can select each attribute uh as um we saw in

1:34:33

the um tutorial video from Google and we can do our own tutorial. I mean it this

1:34:40

is just an example but for example gender you can see it's it's a it's a

1:34:46

it's a um an attribute that has female or male as an option. It has 3,488

1:34:55

females and 3,555 males. So, it's kind of a very um

1:35:02

balanced data set. The type of this attribute is nominal or categorical.

1:35:08

It's the same uh thing. And we will see more information about um

1:35:16

um this this kinds of stuff uh in the following weeks.

1:35:22

But um so what we want to do and again uh we

1:35:27

would um bring this data set already pre-processed for the students because

1:35:32

we're not focusing on pre-processing or anything now. Uh so we would have

1:35:38

removed the missing values. Um you can see that if there are any missing values

1:35:44

it would appear here but you have only female and male. you don't have a a like

1:35:49

a a not available or anything here. You can also add it to the data set so that

1:35:55

you can take a look if there's some instance that lacks information on the gender for example,

1:36:02

right? Uh another idea actually is doing one of the prototypes first and then

1:36:10

letting the students doing the do the the other two. I don't know. This is all

1:36:15

ideas but you can see that um the data set is

1:36:20

this. So you have lots of you know if the if the people got tech uh tax support

1:36:27

yes no or no internet services. Um um tenure is the um the

1:36:38

um how long the contract was signed for. Uh so uh

1:36:47

let me check if that's the meaning. Uh and I'll go here just one second. So the

1:36:54

ten is the number of months the customer has stayed. So there are customers that didn't stay not even a full month and

1:37:02

there are customers that turned after 72 months. Right? So you have a mean and a standard deviation. So these are all

1:37:08

informations about the features and obviously that as we

1:37:13

said some features will be informative for models uh to predict this classification. This is a classification

1:37:21

problem. So let's apply our knowledge now. uh we we if we need to be if we

1:37:27

need to predict a churn a churn is zero or one is churn or or no churn yes or no

1:37:35

so this is a discrete value this is a classification it's class no class yes

1:37:41

so we will use classification model uh classifications that are capable of uh

1:37:47

predict um uh class prediction classification prediction

1:37:53

Um one thing that is very important is to eliminate the customer ID. So customer ID is just it will be there

1:38:01

will be 70,000 diff 70,043 different labels. It's kind of the name

1:38:08

the ID it's a unique identifier and this is this should not participate in any

1:38:14

kind of uh machine learning prediction or AI prediction. uh the AI could just

1:38:21

like memorize easily because it's a computer system the 7,000 uh numbers and

1:38:29

um it would it would do very good in the training data set it would predict it

1:38:35

would like memorize that but when a new uh customer comes in uh it didn't it

1:38:42

didn't extract the knowledge from the other features but from uh memorization

1:38:48

of the label if a new label comes in it will uh perform poorly. So you would

1:38:54

just remove that. Okay. So

1:39:00

um the the data set would be prepared for the students and then we will go and

1:39:05

we go here to classify and we can

1:39:11

um let me just get rid of this. I was uh playing with some things here. Um, Weta

1:39:18

is a very easy um, kind of interface. It's very used in research.

1:39:25

There are some enterprises, I guess, some some industries that use WKA. Most of them use Nime, which is more

1:39:32

complicated, a little bit more complicated, but very much used in the industry. Um, I'm just showing Wacka

1:39:40

because it's it's uh, easier and in academia really. It's it's a very very

1:39:45

easy tool. Um and um I think that we we don't lose

1:39:52

that much time with other um tools because we will see in week four that

1:39:59

with AI with uh with uh sorry with large language models even this no tool no

1:40:06

code tools are um are uh kind of you know being left

1:40:13

aside uh and you can kind program a classifier for example

1:40:20

um using just natural language but we will see that on week four.

1:40:26

So classifiers we have several classifiers the the base types of

1:40:31

classifiers and we saw naive base uh we saw multi-layer perceptron

1:40:38

we saw SVMs right so um in in WA SVMs are called

1:40:45

SMOS you can see here if you uh stop your mouse on top of the classifier

1:40:53

oops sorry you will see that it's a implementation of support factor classifier.

1:41:00

Um you can you we have trees. So for example um

1:41:06

random forests we we we saw random forests. So you have several classifiers. I will choose base just

1:41:13

because it's very easy to show. Um and you can here are the test options. You

1:41:20

can test using your training set. This is uh it's not testing actually it's

1:41:26

just to see uh your performance during the training right um

1:41:34

but uh this is not what we want usually we want to test in an unseen data or we

1:41:41

want to do cross validation right so we want to do cross validation where uh if

1:41:46

you go back to the concepts we hold out uh pieces of the data set uh in this

1:41:52

case uh five-fold cross validation. You can also do a percentage split like

1:41:58

20 uh 80% of training data uh 20% of testing data. But I will do 10fold cross

1:42:05

validation. If you so I chose na naive base I chose my cross validation option. If I click

1:42:12

naive base there are more options and obviously that there are standard there

1:42:18

are standard there is a standard model and you can change this using

1:42:26

this hyperparameters right so these are hyperparameters as we explained it's par

1:42:32

there are parameters that they're not the model parameters but parameters to train the model so the number of decimal

1:42:39

places if you want to use a curve journal uh all of this there all of this kind of a very uh detailed things

1:42:48

uh people uh the students will kind of have to if they want to build the

1:42:54

prototype they will have to go for for it right so they will have to go a little bit more in depth and they can

1:43:00

find all of the information here right um

1:43:08

you know like they uh and they kind of they can go a little bit forward, right?

1:43:13

If you want to debug, if you want uh anyway, so

1:43:18

let's let's just use the the the standard and let's that let's run start.

1:43:25

You can see that um it took a time to build the model and here is the

1:43:31

information. The model correctly classified 5,116

1:43:36

um clients um so 72.6395%

1:43:42

of act of uh sorry of correct uh classifications

1:43:48

or accuracy rate and incorrectly 27.36. This is the confusion matrix as we saw

1:43:56

um the the confusion matrix is uh like uh uh the I have for example 3,617

1:44:07

class no and it was classified as no. So it's a it's a it's a

1:44:14

a true negative or positive depending on what why you you uh you consider

1:44:19

positive or negative. And here is all the uh detailed kind of um

1:44:27

uh um met evaluation metrics. So you have the area under the curve or the uh the

1:44:35

uh AU raw uh ROC um measure.

1:44:41

Um it's pretty decent, right? uh you have the F1 score or the F measure and

1:44:48

you have other kinds of measures as precision and recall and again pre precision and

1:44:54

recall if you go back I'm not sure if we gave details but these are the kinds of things that we can put in a PDF right or

1:45:01

or reading session uh meaning that precision gets true negatives a little

1:45:07

bit uh better than than uh uh sorry false negatives better than false positives but anyways like Um th this is

1:45:15

the way you train uh a knifebased model. We could change the model.

1:45:22

Um but before changing the model we will show uh we can show in the example

1:45:31

prototype or we can show in the um rubric

1:45:37

something very interesting that we will talk about in week five. that is the

1:45:43

advant of LLMs helping us to do data science. So first of all I can I can

1:45:50

even ask um LLM that I need to download this uh this data set. It gives me all

1:45:57

the alternatives. I I I downloaded it and I said I want to make data

1:46:04

prep-processing or data mining or or data science in this data set. Can you

1:46:09

make sense of all the columns and summarize the data set for me? And I

1:46:14

mean the LLM will work and will give you the the meaning of the column. So

1:46:20

customer ID, gender, senior citizen, whether the customer is senior or not and it's giving you like one or zero. So

1:46:27

this is a binary column. If uh tenure, the number of months the the customer

1:46:33

stayed. Um so it it comes and makes sense of the

1:46:38

data and it uses its kind of knowledge base and the knowledge in the internet to

1:46:46

make sense of the data. Okay, it identified some issues. Um but we will

1:46:52

see more of this in week three. um it identifies that I have uh number

1:46:58

of rows and 21 uh uh co columns and um

1:47:05

and here's here is a a a thing that we can ask is I am creating a naive base

1:47:15

model in weta for classifying

1:47:20

churn. I am not sure what to use as model as

1:47:27

model parameters during training.

1:47:32

The options are um oop sorry uh I didn't uh finish. The

1:47:41

options are and let's go get this options. So the options are batch size

1:47:49

and you can even copy this as an image.

1:47:55

So the options are and I'm going to paste

1:48:00

the image here.

1:48:07

Right? So here is the screenshot

1:48:14

and you'll see here's a clear practical explanation of that batch size. So you

1:48:21

can see that you can and you can ask for for example I am not

1:48:26

used to machine learning at can you explain in details what each

1:48:34

option is and what should I choose

1:48:42

given my data set.

1:48:48

So batch size and it explains uh debug debug what is the for example um

1:48:57

let me I don't know like decimal places controls the number of decimal places shown in output right or user kernel

1:49:05

um if if it's false assumes numeric data falls f follows a gshian distribution

1:49:13

um so it's that assumption that it's ident uh that that they are indep the

1:49:19

the attributes are independent and follow a gshian or not right

1:49:24

um and how it affects the changes anyway so this is uh what should you do

1:49:31

practically obviously that uh large language models can um not give you the

1:49:37

right answer but we will talk more about this here I'm just telling you that um

1:49:45

not only you can go to wackas website and check the doc documentation.

1:49:51

But right now you can al also uh play with LLMs to help you.

1:49:57

Um so now let's create a new classifier. Uh for example, let's say let me let me

1:50:04

get a a random forest, right? And there's a bunch a bunch of um a bunch of

1:50:12

options as well that you could uh you know make sense. Um but the thing is

1:50:18

that now that you have all this basic understanding you know what a random

1:50:23

forest is doing. You know that random forests can do classification and you know that you can create a prototype

1:50:31

even if it's not you know particularly um optimizing the hyperparameters.

1:50:37

Uh but you can do that and you can talk to the the the IT guys and you can talk

1:50:46

with engineers and you can talk with computer scientists to discuss the

1:50:52

feasibility of having a machine learning model to solve um decision making pro uh

1:50:58

problems. So I'm just going to create this uh and and and run

1:51:05

and um and you can see that here is you can see

1:51:10

that they are very similar models. So base 72%

1:51:16

this one 79 actually no there um random force not par like I'm I didn't not even

1:51:23

change the hyperparameters but it's doing very well here. Um

1:51:29

and you have precision recall you know you have uh the F1 score uh here

1:51:37

and the confusion matrix right um so you can do

1:51:42

you can find several uh functions because in the end of the day multi-layer perceptron apply functions

1:51:49

you can uh run a multi-layer perceptron and I'm running again like all the uh

1:51:55

the hyperparameters are the same I'm I'm running this uh multi-layer perceptron

1:52:02

uh in my training data it will take much longer than the other models

1:52:08

uh because so here we are

1:52:14

um let me stop the training just to just to

1:52:19

show you so we're in a multilayer perceptron we

1:52:25

can choose like two just two hidden layers just I don't want to do a deep learning this is the learning rate how

1:52:32

fast I update my weights there's several other things here that we have to uh

1:52:40

think about but here's training time is the number of epochs right so the number

1:52:45

of epochs to train through we know what epochs are now so it means that I will

1:52:51

present all my data set to the to the neural network

1:52:57

fix the weights 500 times in order to to

1:53:03

decrease the error. Okay. So anyways like oh and the number of uh neurons um

1:53:13

um in each hidden layer you can also uh parameterize this.

1:53:20

Uh I um let me just uh I want to do a a very small example. Um

1:53:28

this is fold one. Uh it it I mean you can get specialized

1:53:34

in in this tools, right? But it's totally no code. It's no code. Um you

1:53:40

you can also do the same thing in nime k n i m e. You can also do this maybe in a

1:53:48

SAS version of something. So you can see that multi-layer perceptron and the

1:53:53

trees are very similar. Um, this one failed. So, I can delete

1:53:58

this, right? It's the one that I stopped. But the the area under the curve is a

1:54:05

little bit um higher. The F score a little bit higher. But anyways, they're very similar. So, you are doing cross

1:54:12

validation. Okay. Uh you can also if you go to more options,

1:54:19

um let me see here. Oh, sorry. If you can

1:54:24

go, if you go here, you can also do a a validation. So remember cross validation

1:54:29

is kind of a actually a cross test. You don't need to uh have a validation to

1:54:35

early stop to your model to not overfit. But the validation

1:54:43

uh could be like I I'm I'm holding out 10% of my uh of my training set to to do

1:54:50

early stopping to kind of don't overfit. Let's see if something changes. Um

1:54:58

and uh uh oh um it it it is it got

1:55:05

actually a little bit worse. So early stopping um changed a little bit here. So I mean I'm

1:55:13

doing this very like a sloppy way but you you can you can now understand what

1:55:18

is to build the model. So after you built a model you can save your model.

1:55:24

For example, I'm going to save my MLP

1:55:30

uh model here. And when let's say that my uh a new set of customers

1:55:39

uh for the week came in right how can I test this unseen data how can

1:55:46

I predict if this uh customers will will turn or not

1:55:51

so the way to do this um the the the new

1:55:58

uh clients that came in for the I'm not putting the the the dimension or

1:56:07

the first column of ID because I did not train on that. So the the data sets they

1:56:13

have to be the same and the churn I just leave as a inter uh a question mark

1:56:21

and that's what we want to know. We want to predict we we won't know the answer because this is unseen data but we will

1:56:28

predict the data and in our models around 80% of the time we will get this

1:56:34

right. So what I'm going to do is I'm going to load the model

1:56:41

um and you can see here that the model information it was trained in 20 um

1:56:48

attributes. It starts with gender that that there was a churn u but in that

1:56:55

case you had the targets now you don't now you come here and you put supply to

1:57:01

test set and this is the CSV test you

1:57:07

will say that the class is churn but the the you don't have the targets the

1:57:12

question marks and then you don't hit start because start is for training

1:57:19

You come here and you say evaluate reevaluate model on current test set.

1:57:25

That's what you're going to do. And here you go. So you have three instances. The actual values are

1:57:32

unknown. And the predicted is that the first client is yes, the second client

1:57:37

no, the third client no. Um and this are this is the the um

1:57:45

neural network final um uh activation right. So uh one the closest to one is

1:57:53

no. Uh the closest to zero is yes. And uh you can you can actually you can um

1:58:01

this option to to show the prediction. Uh you come here in more options output

1:58:06

predictions and select plain text so that you have the predictions here. Okay. Uh we will see next week that you

1:58:15

can actually see which attributes are the ones that are making the

1:58:23

the the most important difference in the classification. Uh but this is this is something for

1:58:29

week three. Okay. Out there for Wacka Nime or um uh other no code platforms

1:58:39

uh and we will see that. So this classify thing here um you'll see that actually classify

1:58:48

they use classify as prediction because you can also do linear you can do

1:58:53

regression regression is a prediction that is not discrete as in classes but

1:58:59

it's continuous. So you can do you can go here function and you can see that linear regression if you try to do a

1:59:06

linear regression the button start won't work. Why is that? It's because the

1:59:11

churn which is the class and you can point out that this one is the class is

1:59:16

of is a nominal or discrete class of yes or no. Now if you want to if you're

1:59:21

doing a regression your class will be numerical and then the prediction here

1:59:29

will the linear regression for example will work but logistic works right because logistic is uh made to uh

1:59:38

classification so you can go there and um oop sorry uh cross validation I'm

1:59:45

training it so you will see that um the logistic regression is uh uh working

1:59:52

uh fine here. I think that's actually the it's the best model and then you can

1:59:58

um kind of deploy this into unseen data. So one thing that I was envisioning is

2:00:05

we can um point to resources where where students can get better uh at at the

2:00:13

tools per se. We could have workshops uh in different tools and we can find

2:00:19

out what tools are of interest for Ivy to offer. Um I know that nine for example

2:00:27

is less academic than Wacka. Um and maybe industries use more but but

2:00:35

but I um Wacka is very very very easy to use and simple tools. So it's a good

2:00:42

it's a good start. Anyway, this is what I was envisioning for kind of a

2:00:48

uh the first prototype. You you would have three, right? You would you would do um um forecast, regression, and um

2:00:59

classification, which is all in this classify tab. Um

2:01:05

and the other tabs are for the other weeks. Um you can visualize obviously

2:01:10

the data if you want. So like this uh you could see uh streaming TV versus um

2:01:17

churn for example or um

2:01:22

uh uh phone service versus churn and and you can kind of

2:01:29

um it it's it's a binary and binary uh kind of structure. So uh you can make a

2:01:37

sense of what what what is influencing or not. Uh and so on.