Machine Learning Data Lifecycle in Production

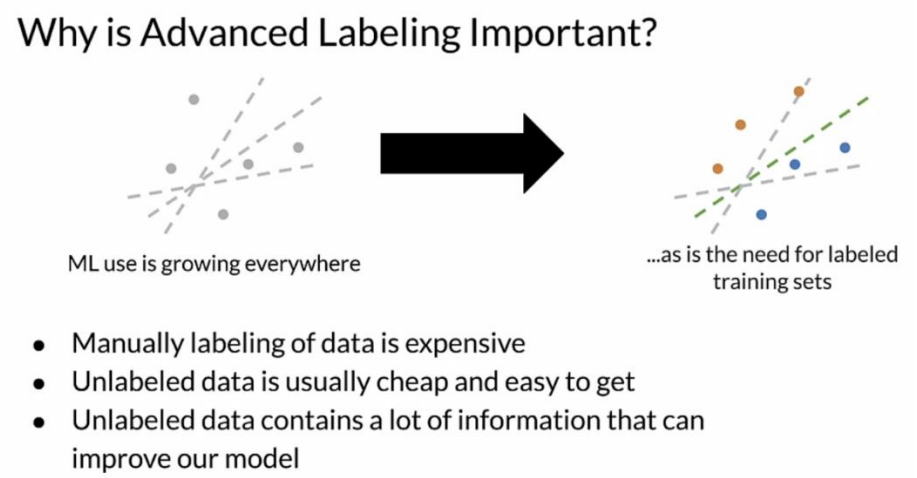
In the second course of Machine Learning Engineering for Production Specialization, you will build data pipelines by gathering, cleaning, and validating datasets and assessing data quality; implement feature engineering, transformation, and selection with TensorFlow Extended and get the most predictive power out of your data; and establish the data lifecycle by leveraging data lineage and provenance metadata tools and follow data evolution with enterprise data schemas.

Understanding machine learning and deep learning concepts is essential, but if you're looking to build an effective AI career, you need production engineering capabilities as well. Machine learning engineering for production combines the foundational concepts of machine learning with the functional expertise of modern software development and engineering roles to help you develop production-ready skills.

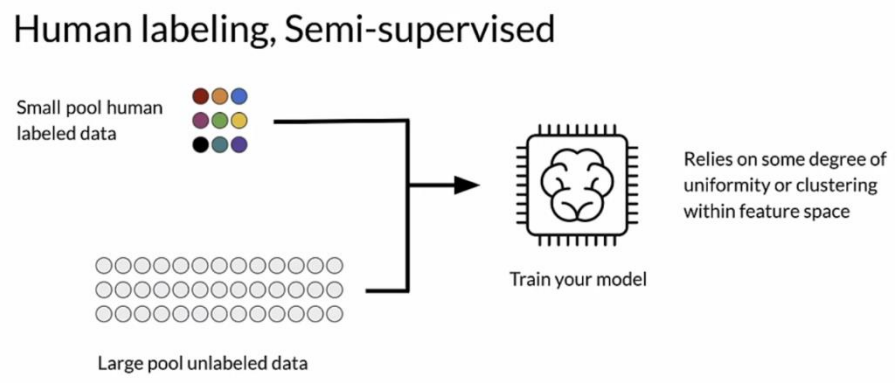
Week 4: Advanced Labelling, Augmentation and Data Preprocessing

Contents

Semi-supervised Learning

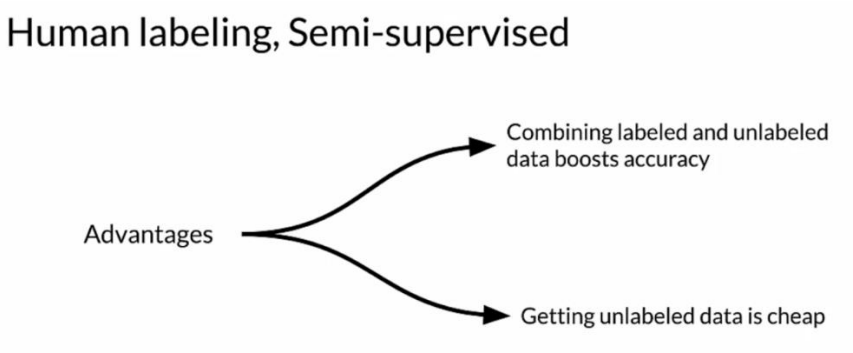


- Let's address an important question first. How can we assign labels in other ways other than going through each example manually?
- In other words, can we automate the process **even at the expense of maybe introducing inaccuracies** in the labeling process?
- This lesson is all about exploring some powerful advanced labeling techniques.
- The first stop in this journey is exploring how semi-supervised labeling works and how you can use it to **improve your model's performance by expanding your labeled dataset**.
- The next stop is **active learning**, which uses **intelligence sampling** to assign labels based on the existing data to unlabeled data.
- The last stop is weak supervision, which is a way to **programmatically label data**, typically by using heuristics which are designed by subject matter experts.
- Snorkel is a friendly framework for applying weak supervision.
- First, why is advanced labeling important? Well, machine learning is growing everywhere and machine learning requires training data, labeled data, at least if you're doing supervised learning.
- That means we need labeled training sets. But manually labeling data is often expensive and difficult, while unlabeled data is typically pretty cheap and easy to get.
- An unlabeled data contains a lot of information that can help improve our model.
- Advanced labeling techniques help us reduce the cost of labeling data while leveraging the information in large amounts of unlabeled data.

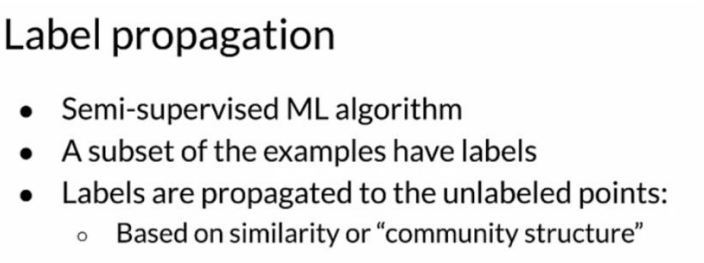


- With Semi-supervised labeling, you start with a relatively small dataset that's been labeled by humans.
- Then you'll combine that labeled data with a large amount of unlabeled data, where you'll infer the labels for the unlabeled data by looking at how the different human labeled classes are clustered or structured within the feature space.

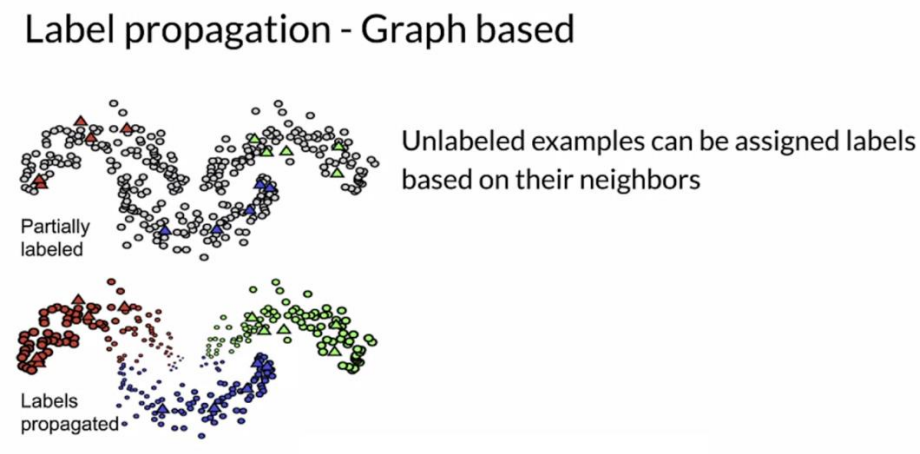
- Then you'll train your model using the combination of the two datasets. This method is based on the **assumption that different label classes will cluster together or have some recognizable structure within the feature space.**



- Using Semi-supervised labeling is advantageous for really two main reasons
- Combining labeled and unlabeled data can improve the accuracy of machine learning models.
- Getting unlabeled data is often very inexpensive, since it doesn't require people to assign labels. Often unlabeled data is easily available in large quantities.



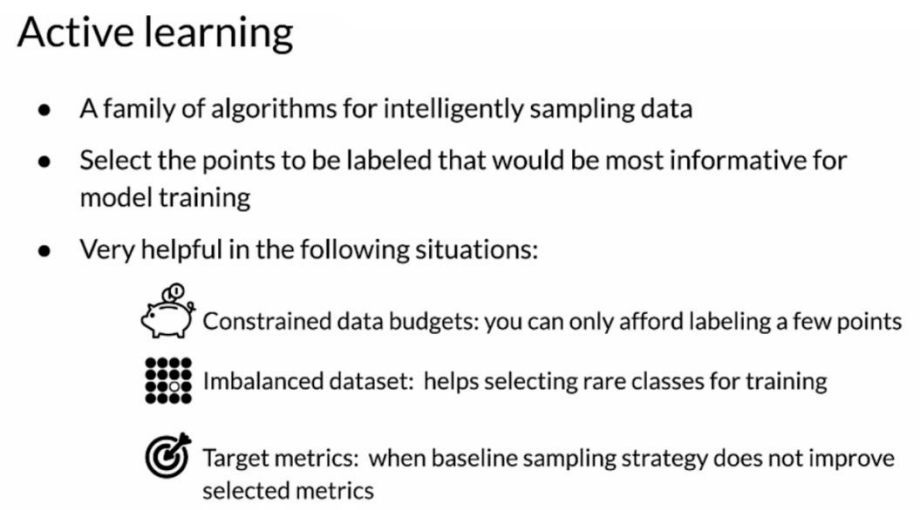
- **Label propagation** is an **algorithm that assigns labels to previously unlabeled examples**.
- This makes it a semi-supervised algorithm where a subset of the data points have labels. The algorithm propagates the labels to data points without labels.
- It does that based on the **similarity** or community structure of the labeled data points and the unlabeled data points. This similarity or structure is used to assign labels to the unlabeled data.



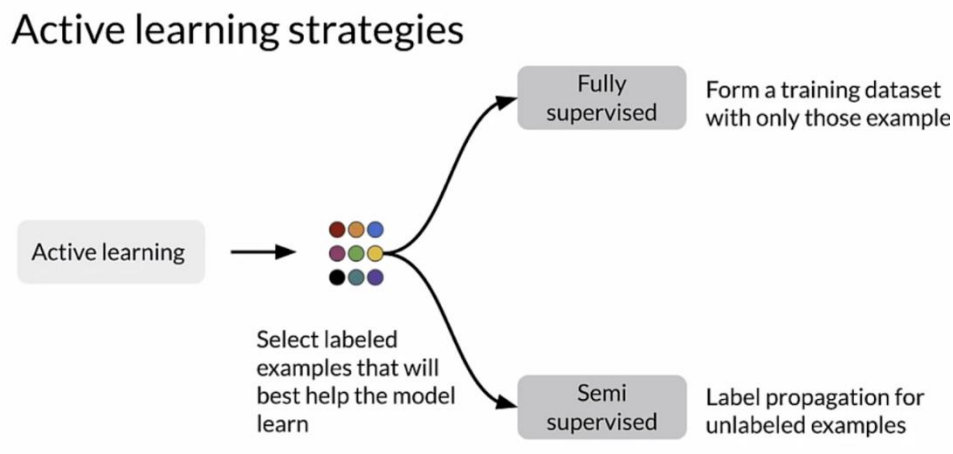
- For example, there's graph-based, and in this figure you can see some labeled data, the red, blue, and green triangles here, and a lot of unlabeled data, which are the gray circles.
- With this method, you assign labels to the unlabeled examples based on their neighbors.
- The labels are then propagated to the rest of the clusters as shown here by the colors.

- We should mention that there are many different ways to do label propagation. Graph-based label propagation is only one of several techniques.
- Label propagation itself is considered **transductive learning**, meaning that we're **mapping from the examples themselves without learning a function for the mapping.**

Active Learning

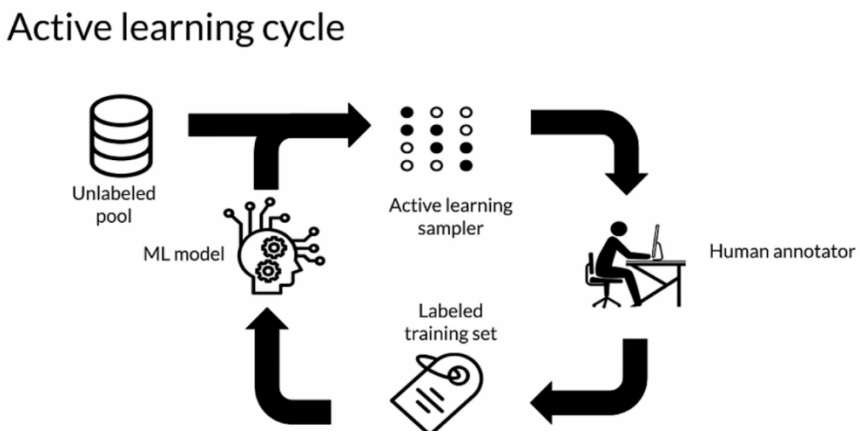


- Active learning is a way to intelligently sample your data.
- Intelligent sampling **selects unlabeled points that bring the most predictive value to your model**.
- This is very helpful in a variety of contexts: First of all, limited data budget. It caused money to label data, especially when you're using human experts to look at the data and assign a label to it, for example, maybe in healthcare
- Active learning helps offset this cost and burden.
- If you have an imbalanced dataset, active learning is an efficient way to **select rare classes at the training stage**.
- If standard sampling techniques don't help with improving accuracy and other target metrics, active learning can find ways to achieve or help achieve the desired accuracy.

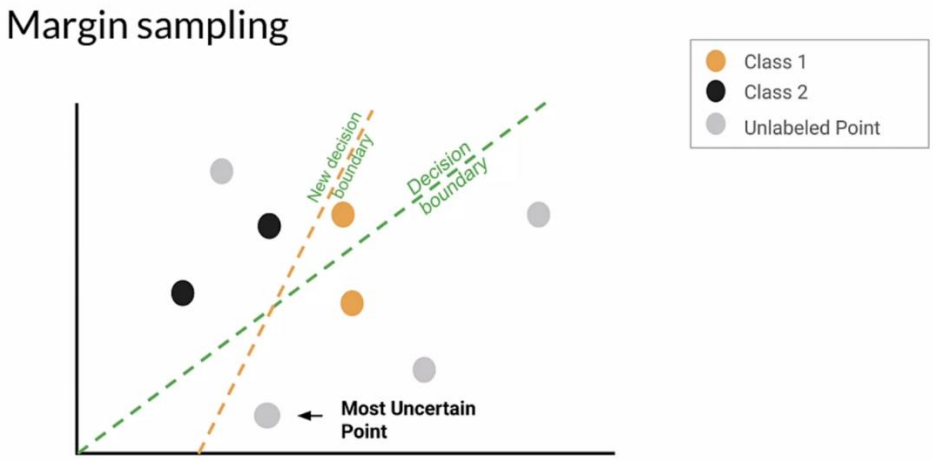


- Active learning strategy works by selecting **labeled** examples that will best help the model learn.
- In a fully supervised setting, the training dataset consists of only the examples that you've labeled.

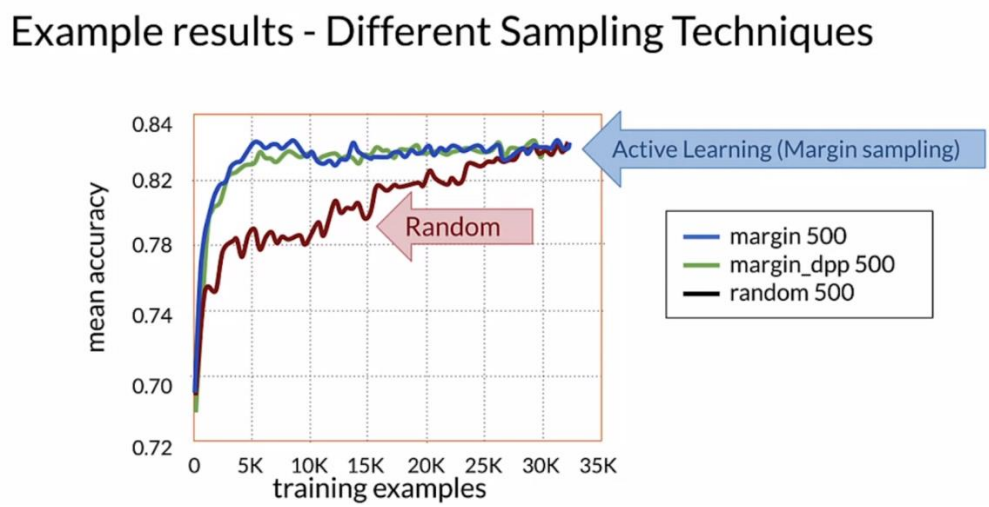
- In a semi-supervised setting, you leverage those examples to perform **label propagation**, so that's in addition to active learning.



- This is the typical active learning lifecycle: you start with a **pool of unlabeled data**, and then active learning selects a few examples using intelligent sampling, and we'll talk about what intelligent sampling is in a second, and then you annotate the data with human annotators or by leveraging some other technique.
- This annotation or labeling procedure generates up labeled training dataset.
- Finally, you use this labeled data to train a model and make predictions.
- The cycle goes on, but this begs the question, **how do we do intelligent sampling**?



- **Margin sampling** is one widely used technique for doing intelligent sampling.
- In this example, the data belong to two classes, additionally, they're unlabeled data points.
- In this setting, the simplest strategy is, we're just going to train a binary linear classifier model that just makes it simple. We're going to just use a linear model for the example.
- We're going to train that on the labeled data and that's going to give us a decision boundary.
- Now, among the unlabeled data, the **most uncertain point is the one that is closest to the decision boundary**.
- **With active learning, you'll select that most uncertain point to be labeled** next and added to the dataset.
- Now, using this new labeled data point as part of the dataset, you retrain the model to learn a new classification boundary.
- Moving the boundary, the model learns a bit better to separate those classes.
- Next, you find the most uncertain data point, again, and repeat the process **until the model doesn't improve**.



- This plot shows model accuracy as a function of the number of training examples for sampling techniques. The red line shows the results of just selecting points at random to label.
- The blue and green line shows the performance of two margins sampling algorithms using active learning.
- As you can see, the margin sampling methods achieve a higher accuracy with fewer training examples than the random sampling technique.
- But of course, eventually, as a **higher percentage of the unlabeled data is labeled with even using random sampling, it will catch up to margin sampling**, as we see here, but it **requires a lot more data to be labeled**.

Active learning sampling techniques

Margin sampling: Label points the current model is least confident in.

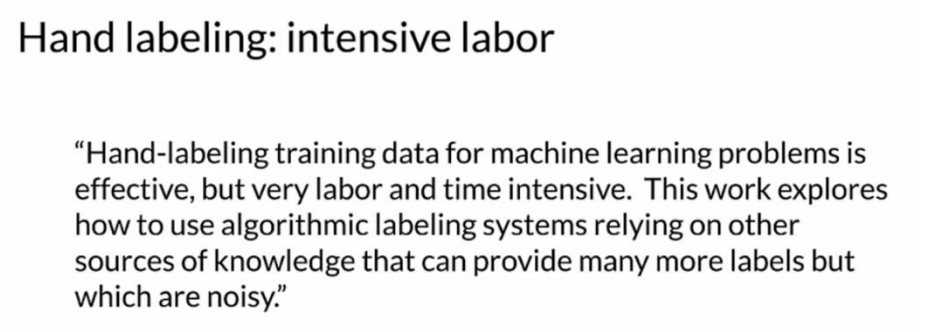
Cluster-based sampling: sample from well-formed clusters to "cover" the entire space.

Query-by-committee: train an ensemble of models and sample points that generate disagreement.

Region-based sampling: Runs several active learning algorithms in different partitions of the space.

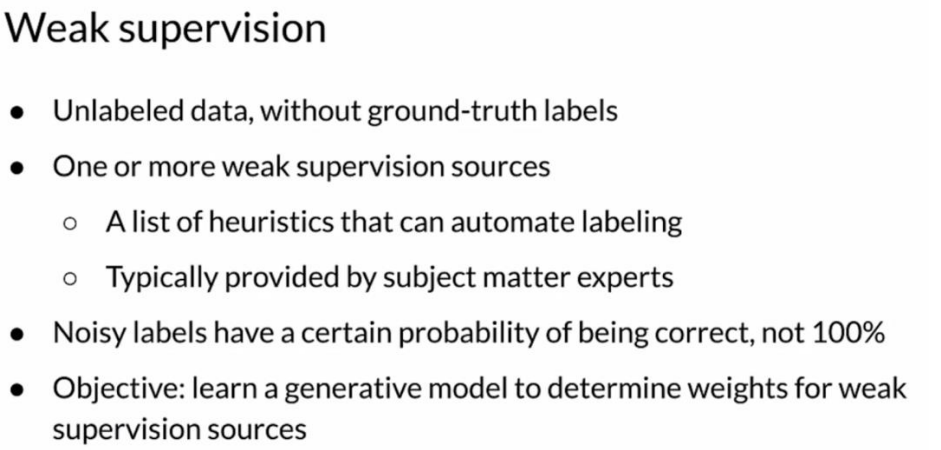
- There are several common active learning sampling techniques.
- With margin sampling, as we just saw, you assign labels to the most uncertain points based on their distance from the decision boundary.
- With cluster-based sampling, you select a diverse set of points by using clustering methods over your feature space.
- With query-by-committee, you train several models and select the data points with the highest disagreement among those models.
- Finally, region-based sampling is a relatively new algorithm. At a high level, this algorithm works by dividing the input space into separate regions and running an active learning algorithm in those regions.

Weak Supervision



- Let's get started with weak supervision, the final advanced labeling technique that we'll be covering.
- This is a quote from Jeff Dean, who leads the machine learning research group at Google.
- He's commenting on the cost in terms of both time and labor, of labeling data.
- He's also commenting specifically on the work to create algorithmic approaches the labeling data that rely on other sources of information, to produce noisy labels.
- Weak supervision which we'll discuss now, is the primary way of doing this kind of labeling.

- Weak supervision is a way to generate labels using information from one or more sources.
- Usually these are subject matter experts, and usually they're designing heuristics.
- The resulting labels are **noisy** rather than the deterministic labels that we're used to.
- More specifically weak supervision comprises one or more noisy conditional distributions over unlabeled data.
- And the main objective, is to learn a generative model that determines the relevance of each of these noisy sources.



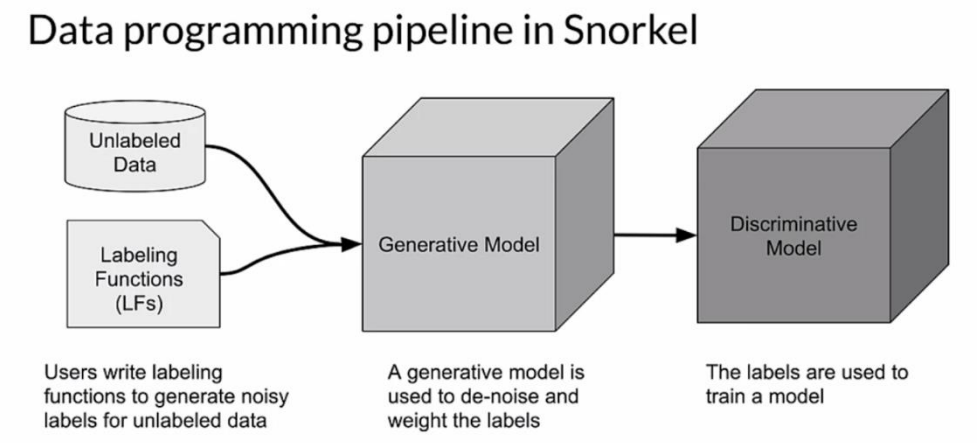
- So, you start with **unlabeled** data for which you don't know the true labels.

- Then you add to the mix one or more **weak supervision sources**. These sources are a **list of heuristic procedures** that implement noisy, and imperfect automated labeling.
- Subject matter experts are the most common sources for designing these heuristics, which typically consists of a coverage set, and an expected probability of the true label over the coverage set.
- **It uses heuristics or labelling functions instead of deterministic labels**
- By noisy we mean that the label has a certain probability of being correct, rather than the 100 certainty that were used to, for labels in our labeled data.
- The main goal is to learn the **trustworthiness** of each supervision source, and this is done by **training a generative model.**
- Overall, it takes advantage of input from experts in the field, and creates and structures training datasets through programming

Snorkel

- Project started at Stanford in 2016
- Programmatically building and managing training datasets without manual labeling
- Automatically: models, cleans, and integrates the resulting training data
- Applies novel, theoretically-grounded techniques
- Also offers data augmentation and slicing

- The Snorkel framework came out of Stanford in 2016 and is the most widely used framework for implementing weak supervision.
- It doesn't require manual labeling, so the system programmatically builds and manages training dataset.
- Snorkel provides tools to clean, model, and integrate the resulting training data rising through the weak supervision pipeline.
- Snorkel uses a novel theoretically grounded technique to get the job done promptly and efficiently.
- And in addition, Snorkel also offers data augmentation and slicing.



- With Snorkel, you'll start with unlabeled data and apply labeling functions. Which are the heuristics which are designed by subject matter experts, to generate noisy labels.
- You'll then use a **generative model to de-noise the noisy labels, and assign importance weights to the different labeling functions** i.e. determine weights for **supervision sources**
- Finally, you'll train a **discriminative** model, i.e. your model, with the de-noised labels.

Snorkel labeling functions

```
from snorkel.labeling import labeling_function

@labeling_function()
def lf_keyword_my(x):
    """Many spam comments talk about 'my channel', 'my video', etc."""
    return SPAM if "my" in x.text.lower() else ABSTAIN

@labeling_function()
def lf_short_comment(x):
    """Non-spam comments are often short, such as 'cool video!'."""
    return NOT_SPAM if len(x.text.split()) < 5 else ABSTAIN
```

- So let's take a look at what a couple of simple labeling functions might look like in code. Here I'll show a simple way to create functions to label spam using snorkel.
- The first step is to import the labeling function from snorkel. And then with this functional label messages spam, if it contains the word **my**.
- This is just an example it's an easy example to show, so messages with 'my' don't have to be spam.
- Otherwise the function returns abstained, which means that it has **no opinion** on what the labels should be.
- The second function labels a message spam if it's longer than five words.
- So what we're showing here is you were **using multiple labeling functions to try to arrive at what the label should be.**

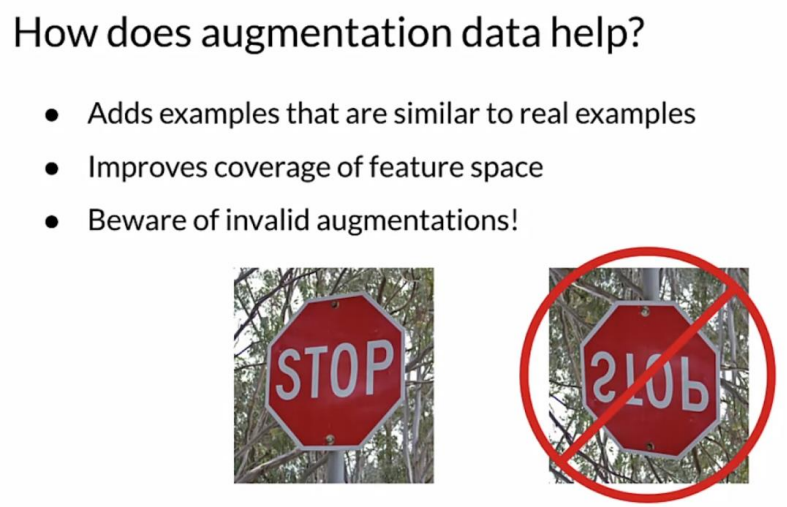
Key points

- Semi-supervised learning:
  - Applies the best of supervised and unsupervised approaches
  - Using a small amount of labeled data boosts model accuracy
- Active learning:
  - Selects the most important examples to label
  - Improves predictive accuracy
- Weak supervision:
  - Uses heuristics to apply noisy labels to unlabeled examples
  - Snorkel is handy framework for weak supervision

- Supervised learning requires labeled data, but labeling data is often an expensive, difficult, and slow
- Semi-supervised learning is one possible way to add labels to unlabeled data. So this method falls between unsupervised learning and supervised learning.
- It works by combining a small amount of labeled data with a large amount of unlabeled data, and this improves learning accuracy.
- Active learning is another advanced labeling method. It relies on intelligence sampling techniques that **select the most important examples** to label and to add to the data set.
- Active learning improves predictive accuracy while minimizing labeling cost.
- The last method you explored was weak supervision.
- Weak supervision leverages noisy, limited, or inaccurate label sources inside a supervised learning environment, to **test labeling accuracy**.
- Snorkel is a compact and user friendly system to manage all these operations for weak supervision and establish training data sets using weak supervision.

Data Augmentation

- Previously you explored methods for getting more labeled data by labeling unlabeled data.
- But another method is to augment your existing data to create more labeled examples.
- With data augmentation you expand the dataset by adding slightly modified copies of existing data, or create new synthetic data from your existing data.
- With the existing data, it is possible to create more data by making minor alterations or perturbations in the existing samples.
- Simple tasks like flips or rotations and images are an easy way to double or triple the number of images in a dataset.



- Data augmentation is a way to improve your model's performance, add new valid examples that fall into **regions of the feature space** that aren't covered by your real examples and add valid examples in this way, improves coverage of your feature space.
- Keep in mind that if you add invalid examples, you run the risk of learning the wrong answer or at the very least introducing unwanted noise. So be careful to only augment your data in valid ways.

- CIFAR is the Canadian Institute for Advanced Research. The CIFAR-10 dataset is a collection of images commonly used to train machine learning models and computer vision models.
- It's one of the most widely used datasets for machine learning research.
- CIFAR-10 contains 60,000 32 by 32 color images. There's 10 different classes with 6000 images in each class.

Data augmentation on CIFAR-10

Let's augment the CIFAR-10 dataset with the following steps:

1. Pad the image with a black, four-pixel border
2. Randomly crop a 32 x 32 region from the padded image
3. Flip a coin to determine if the image should be flipped horizontally left/right

- So let's take a practical look at data augmentation with the CIFAR-10 dataset.
- We will add a border padding the image, and we will crop the padded images to a 32 x 32 image and will flip or rotate the padded and cropped images based on a coin toss or random variable.
- This is so that the model is not overfitting to particular rotations or scales.

Defining the augment operation

```
def augment(x, height, width, num_channels):
    x = tf.image.resize_with_crop_or_pad(x, height + 8, width + 8)
    x = tf.image.random_crop(x, [height, width, num_channels])
    x = tf.image.random_flip_left_right(x)
    return x
```

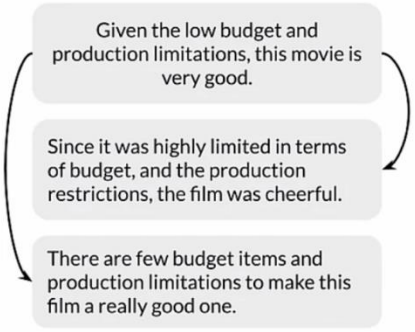
- Since color images are tensors, tensorflow provides very useful functions to perform augmentations on image datasets.
- Let's look at a chunk of code that encapsulates the steps for doing a left, right flip.
- First we define a function that will use to augment image datasets.
- Then we'll use tf.image.resize with crop_or_pad which allows you to resize or crop the image.
- In this case we're going to pad.
- tf.image.random crop generates a crop of size height by width across all channels.
- And then tf.image.random flip left, right does the same thing with rotations like flipping left or right.
- So the function returns the altered image to be appended to the dataset.

Augmented examples

- Here's some examples after applying the padding, cropping and left, right flip.

Other Advanced techniques

- Semi-supervised data augmentation e.g., UDA, semi-supervised learning with GANs
- Policy-based data augmentation e.g., AutoAugment



- Apart from simple image manipulation, there are other advanced techniques for data augmentation, such as semi supervised data augmentation such as the unsupervised data augmentation or UDA.
- Or other methods using policy based data augmentation like AutoAugment.
- On the right, there's an example of a movie review, and then using policy augmentation, we generate a variant example.
- So we have a new example perfectly valid that we've been able to generate with augmentation.
- We can do it again, and we can generate another one.

Key points on data augmentation

- It generates artificial data by creating new examples which are variants of the original data
- It increases the diversity and number of examples in the training data
- Provides means to improves accuracy, generalization, and avoiding overfitting

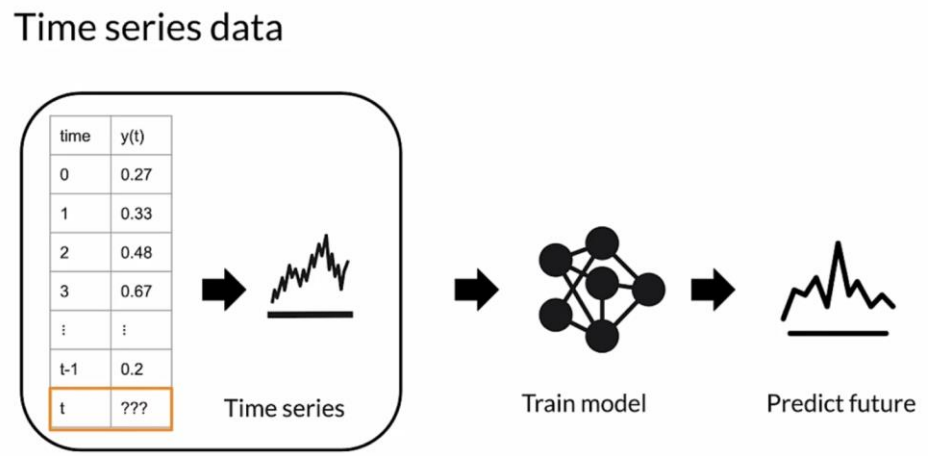
Time Series

A note on different types of data

- TFX pre-processing capabilities for multiple data types:
  - Images
  - Video
  - Text
  - Audio
  - Time series
- Optional notebook on images
- Two optional notebooks on time series

- First, let us briefly mention some different types of data depending on the problem that you are working on in the data that you have.
- You might be working with different types of data, each of which require different preprocessing, and in some cases, different modeling techniques.

- For example, you might be working with images or video or text or audio or time series data. There are too many different types of data like this to discuss thoroughly in this class. So we have provided some optional materials for you to explore some of these on your own.
- There is an optional notebook to work with for the CIFAR-10 image data set, and there's two other notebooks to work on with time series data.
- One data is on weather data and the other contains data from an accelerometer, and other sensors available on most cell phones.
- Of all the data types listed here, time series is probably the one that most developers are the least familiar with. So let us get started by reviewing the key aspects of time series.



- Time series are a sequence of data points in time, often from recorded events where the time dimension indicates when the event occurred.
- They may or may not be ordered in the raw data, but you almost always want them to be in **order by time for modeling.**
- We want to predict the value in a typical situation. We want to predict the value of y at time t sometime in the future based on previous measurements.
- The goal is to train a model that predicts future outputs with acceptable accuracy.

- Karl Kristian Steincke pointed out that it's difficult to make predictions, especially about the future.
- The corollary to this of course is that it is relatively easy to make predictions about the past since that is already happened.

Time series forecasting

- Time Series forecasting predicts future events by analyzing data from the past
- Time series forecasting makes predictions on data indexed by time
- Example:
  - Predict future temperature at a given location based on historical meteorological data

- Time series forecasting does exactly that it tries to predict the future.
- It does it by analyzing data from the past. This requires time index data for example, to predict future temperature at a given location. We could use other meteorological variables such as atmospheric pressure, wind, direction and velocity, et cetera, that have been recorded previously at some time in the past.
- Let us look at a concrete example of making weather predictions.

Time series dataset: Weather prediction

We will use the weather time series dataset recorded by the Max Planck Institute for Biogeochemistry:

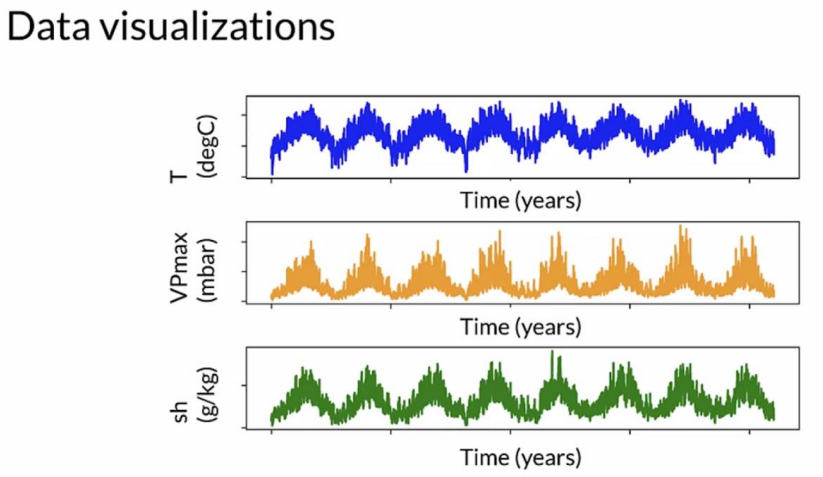
- to preprocess time series data with TensorFlow Transform
- to convert data into sequences of time steps:
  - Making data ready to train a long short-term memory (LSTM) recurrent neural network (RNN)

- In this example, you are going to be working with a time series data set that was recorded by the Max Planck Institute for bio geochemistry.
- This dataset contains 14 different features including air temperature, atmospheric pressure and humidity. They were recorded every 10 minutes beginning in 2003.
- Your job is to preprocess the features with the TFX pipeline, and convert the data into time sequences
- This format is required to train recurrent neural networks such as long, short term memory models or LSTMs.

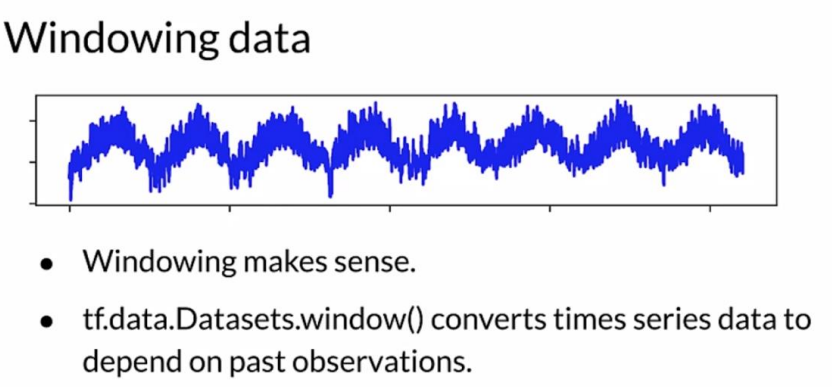
Weather time series dataset

- There are 14 variables:
  - P(mbar), T (degC), Tdew (degC), rh (%), VPmax (mbar), VPact (mbar), VPdef (mbar), sh (g/kg), H20C (mmol/mol), rho (g/m**3), wv (m/s), max.xv (m/s), wd (deg)
  - Target is T (degC)
- Observations recorded every 10 minutes
  - 6 observations per hour
  - 144 (6 X 24) observations in a day.

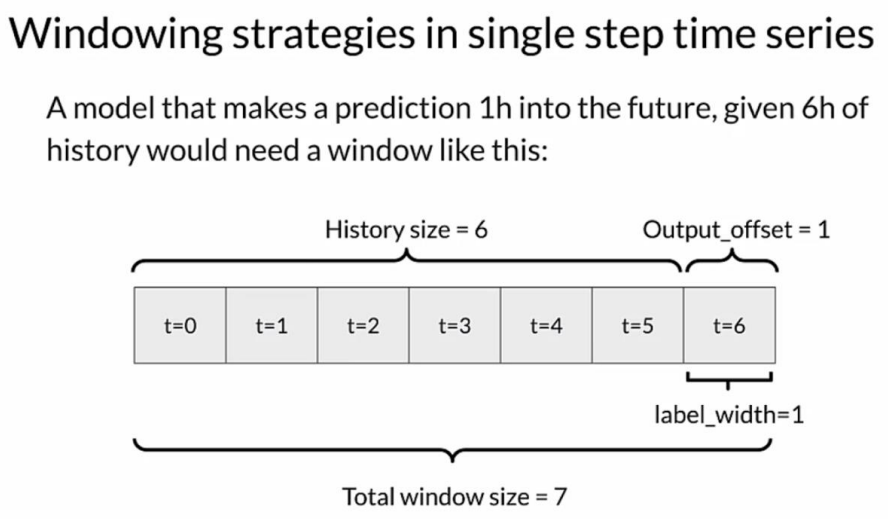
- Let us take a closer look at how the data is organized and collected. There are 14 variables, including measurements related to humidity, wind, direction and velocity, temperature and atmospheric pressure.
- The target for prediction is the Temperature.
- The sampling rate is one observation every 10 minutes. So, you have six observations per hour and 100 and 144 in a given Day.



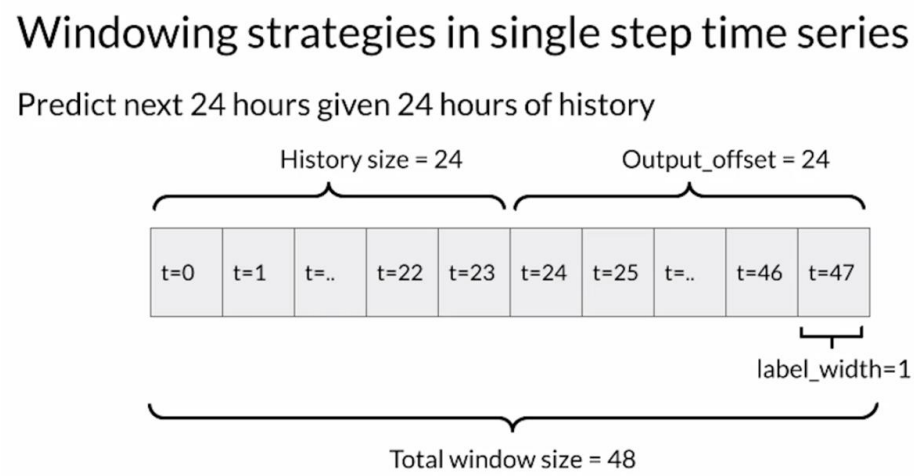
- These are plots for a few of the features over time, and the target variable T, which is temperature.
- You can see that there is a pattern to this which repeats over specific intervals of time.
- There is clear seasonality here, which we need to consider when doing feature engineering for this data.
- We should consider doing **seasonal decomposition**, but to keep things simple in this example, we would not be doing that.



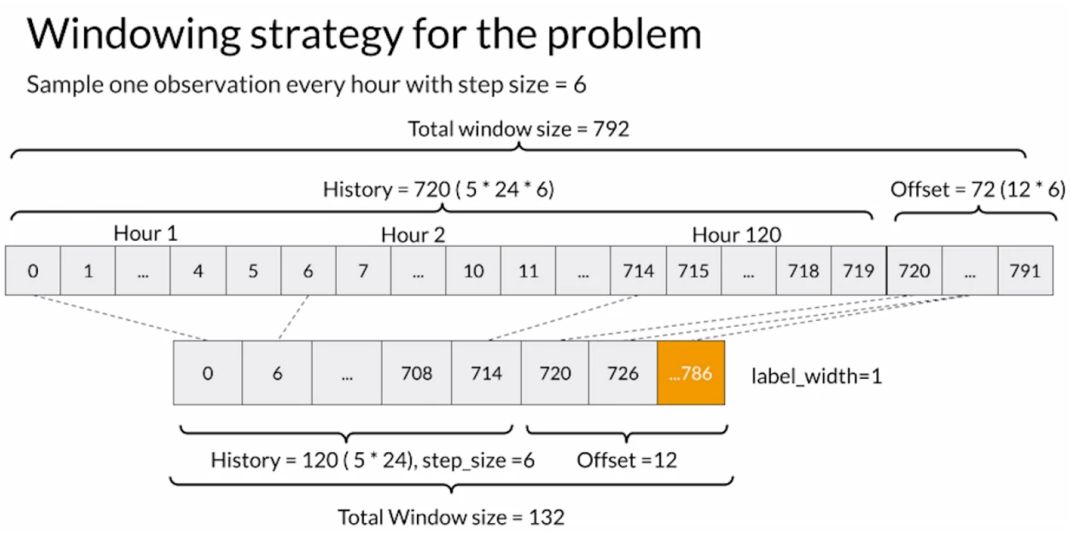
- The data exhibits clear periodicity or seasonality, which in this case is actually most likely due to the typical seasonal progression over the year.
- But remember seasonality is a characteristic of data, and often has nothing to do with the actual seasons of the year. It is really about periodicity.
- Using a **windowing strategy** to look at dependencies with the past data seems a natural past the take.
- And luckily TFX has this functionality already built in in the notebook.
- You will see the window() function tf.data, and we'll use that to group the data set into windows. So let us see exactly how this actually works using weather data as an example.



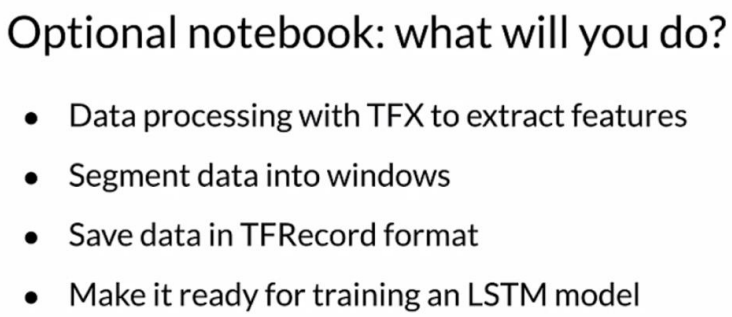
- Windowing strategies in time series become pretty important, and they are kind of unique to time series and similar types of sequence data.
- In this example, you have a model that you can use to make a prediction one hour into the future and given a history of six hours, we will use a sliding window with a window size of six and an offset of one.
- So the total window sizes 7, 6 plus one.



- In another example, suppose you make a prediction 24 hours into the future given 24 hours of history.
- So that in that case your history size is 24, the offset size is also 24.
- So you could use a total window size of 48 which would be the history plus the output offset.
- It is also important to consider when now is, and **not** include data in the future which is referred to as time travel.
- In this example, if now is at T equals 24 then we need to be careful not to include the data from T=25 to T=47 in our training data.
- We could do that in feature engineering or by reducing the window to include only the history and the label.



- Let us talk about the sampling strategy. You already know that there are six observations per hour. In our example, one observation every 10 minutes
- In a day, there is going to be 144 observations. If you take **five days of past observations** and make a prediction six hours into the future.
- That means that our history size will be five times 144, or 720 observations, and the output offset will be 12 times six, or 72.
- So the total window size in time is 792 since observations in an hour are unlikely to change too much.
- Let us sample one observation per hour. So instead of one every 10 minutes we are going to go to one per hour. So say we could take the first observation in the hour as the sample or even better, you can take the median of the observations in each hour
- Then our history size becomes five times 24 times one or 120 and our output offset will be six.
- So our total window size becomes 126.
- So we have reduced the size of our feature vector from 792 to 126 by either **sampling within each hour** or aggregating the data for each hour by taking the median.
- So I know the numbers might be a little confusing here, but the important point is to think about how we are treating the data in our window.
- The above is an example of taking a finite window of data plus **down-sampling** for **slow time varying signals**.



- What will you do in the optional notebook?
- In that notebook, you are going to start with preprocessing the weather time series dataset using TFX.
- And you will use TF.data.dataset window to create the windows that you will be using to build your examples, and you'll save the transform and preprocessed data in TF record format.
- And finally, you will create training and test data sets in a way that the features can be easily used to train an LSTM model using either TensorFlow or Keras with windowing strategies or in some other framework.
- The training is not implemented in the notebook since we are focusing on the data itself.

Sensors and Signals

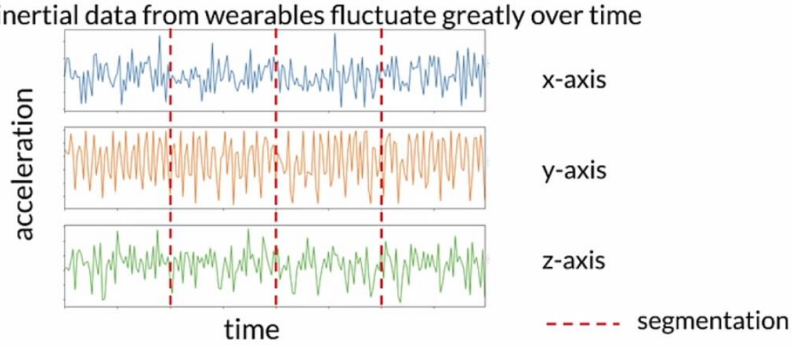
Sensors and Signals

- Signals are sequences of data collected from real time sensors
- Each data point is indexed by a timestamp
- Sensors and signals data is thus time series data
- Example: classify sequences of accelerometer data recorded by the sensors on smartphones to identify the associate activity

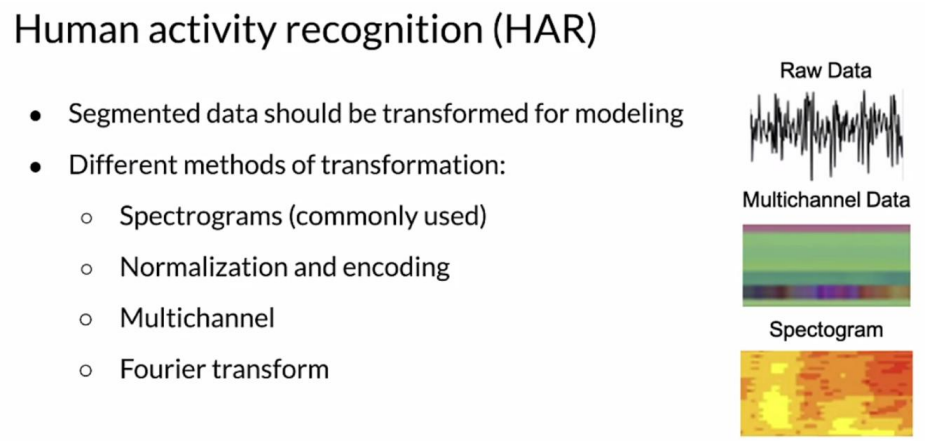
- Let's review sensor generated data which is often referred to as signals.
- Let's start by defining important terminology by means of an example using a collected accelerometer data from smartphones.
- The accelerometer is a sensor
- Signals are sequences of data collected by a sensor in real time
- Data points are indexed by time stamps. So therefore this is a time series data and we're going to use time series analysis.
- Let's look at a concrete problem, **human activity recognition or HAR**: Can we infer the activity from the accelerometer patterns in time?

Human activity recognition (HAR)

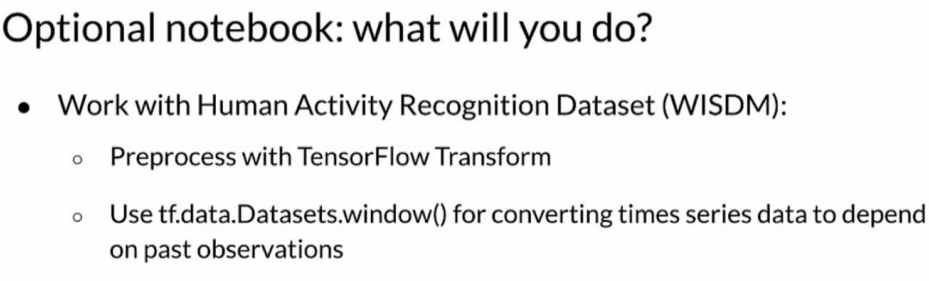
- HAR tasks require segmentation operations
  - Raw inertial data from wearables fluctuate greatly over time



- Let's look at some of the challenges of activity recognition.
- Properly segmenting the sensor data is key to successful activity recognition.
- This is similar to the windowing strategy that we just discussed for weather data.
- Inertial data like this fluctuates greatly over time, so **segmentation is key to detecting proper activity chunks**.
- The raw time-dependent acceleration data set is separated into segments during the data segmentation process. All of the following HAR related operations, including feature extraction, classification and validation etc. are based on these segments.
- The length of the segments depends on the application context and sampling rate of the sensors
- Segments for HAR are typically 1- 10 seconds long.



- Now that the data is segmented, let's talk about typical transformations. The segment of data should be transformed for modeling because it helps with model accuracy and performance
- There are different ways to go about transforming your data.
- A spectrogram of an inertial signal is a new representation of the signal as a function of frequency and time.
- Spectrogram representation is a powerful method for extracting interpretable features that represent the **intensity differences** among different inertial data points.
- Spectrogram displays amplitude changes for each frequency as a function of time
- Other options include normalization and encoding, multichannel processing and applying Fourier transform.



- In the optional notebook, you will try to **recognize human activities from sensor data.**
- The wireless sensor data mining dataset contains data collected through controlled laboratory conditions.
- The data set is a test bench for activity recognition using cell phone accelerometer data,
- In this notebook, you'll preprocess the data within a TFX pipeline. There, you will use tf.data.Dataset.window to group the data into windows, which can be used with an RNN or similar model.

References

- Hand Labeling
- Weak supervision
- Snorkel
- How do you get more data?
- Advanced Techniques
- Images in tensorflow
- https://www.cs.toronto.edu/~kriz/cifar.html
- https://www.tensorflow.org/datasets/catalog/cifar10
- Weather dataset
- Human Activity Recognition
- Label Propagation: Iscen, A., Tolias, G., Avrithis, Y., & Chum, O. (2019). Label propagation for deep semi-supervised learning. https://arxiv.org/pdf/1904.04717.pdf