Ciclo de vida de los datos de aprendizaje automático en la producción

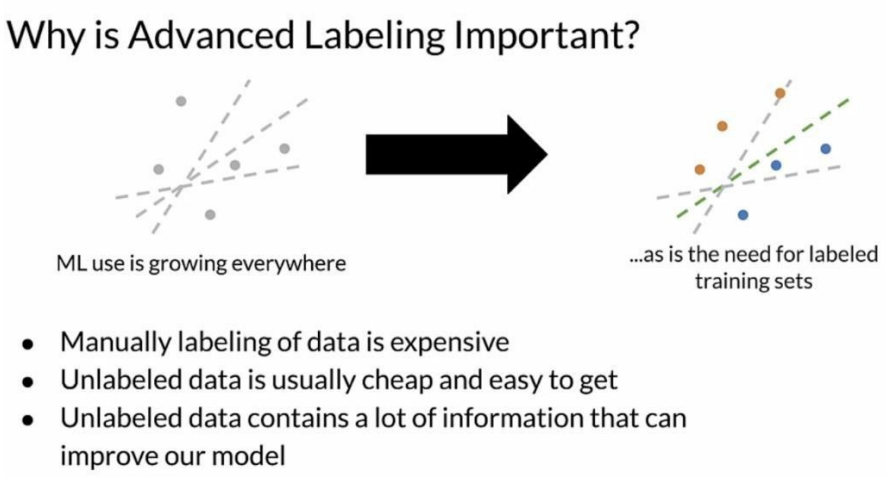
En el segundo curso de Ingeniería de Aprendizaje Automático para la Especialización de Producción, construirá pipelines de datos mediante la recopilación, limpieza y validación de conjuntos de datos y la evaluación de la calidad de los datos; implementará la ingeniería de características, la transformación y la selección con TensorFlow Extended y obtendrá el mayor poder predictivo de sus datos; y establecerá el ciclo de vida de los datos aprovechando las herramientas de metadatos de linaje y procedencia de los datos y seguirá la evolución de los datos con esquemas de datos empresariales.

Entender los conceptos de aprendizaje automático y aprendizaje profundo es esencial, pero si quieres construir una carrera eficaz en el campo de la IA, también necesitas capacidades de ingeniería de producción. La ingeniería de aprendizaje automático para la producción combina los conceptos fundamentales del aprendizaje automático con la experiencia funcional de las funciones modernas de desarrollo de software e ingeniería para ayudarte a desarrollar habilidades listas para la producción.

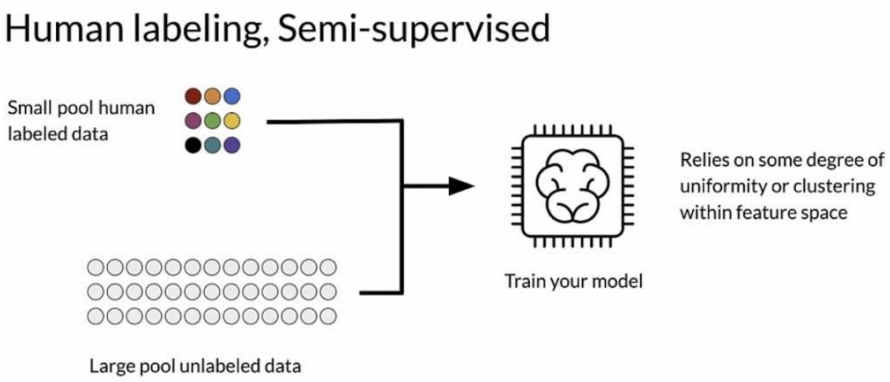
Semana 4: Etiquetado avanzado, aumento y preprocesamiento de datos

Contenido

Aprendizaje semisupervisado

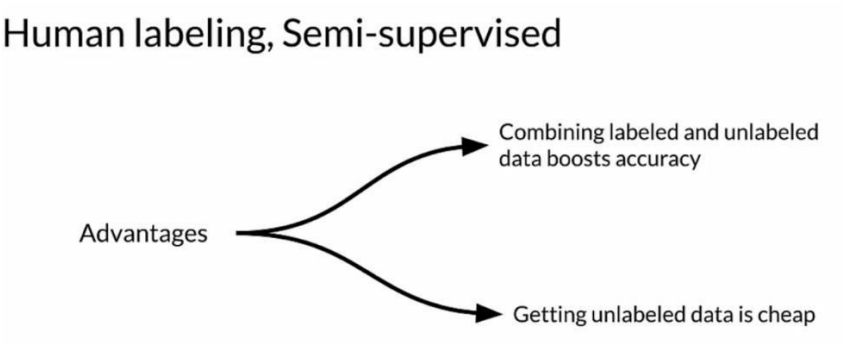


- Abordemos primero una cuestión importante. ¿Cómo podemos asignar etiquetas de otra manera que no sea pasando por cada ejemplo manualmente?
- En otras palabras, ¿podemos automatizar el proceso **incluso a costa de introducir quizás imprecisiones** en el proceso de etiquetado?
- Esta lección trata de explorar algunas poderosas técnicas avanzadas de etiquetado.
- La primera parada en este viaje es explorar cómo funciona el etiquetado semisupervisado y cómo puede utilizarlo para
  mejorar el rendimiento de su modelo ampliando su conjunto de datos etiquetados.
- La siguiente parada es el **aprendizaje activo**, que utiliza el **muestreo inteligente** para asignar etiquetas basadas en los datos existentes a los datos no etiquetados.
- La última parada es la supervisión débil, que es una forma de **etiquetar los datos de forma programada**, normalmente mediante el uso de heurísticas diseñadas por expertos en la materia.
- Snorkel es un marco amigable para aplicar una supervisión débil.
- En primer lugar, ¿por qué es importante el etiquetado avanzado? Bueno, el aprendizaje automático está creciendo en todas partes y el aprendizaje automático requiere datos de entrenamiento, datos etiquetados, al menos si estás haciendo aprendizaje supervisado.
- Eso significa que necesitamos conjuntos de entrenamiento etiquetados. Pero etiquetar manualmente los datos suele ser caro y difícil, mientras que los datos sin etiquetar suelen ser bastante baratos y fáciles de conseguir.
- Los datos no etiquetados contienen mucha información que puede ayudar a mejorar nuestro modelo.
- Las técnicas avanzadas de etiquetado nos ayudan a reducir el coste del etiquetado de los datos, al tiempo que se aprovecha la información de grandes cantidades de datos sin etiquetar.

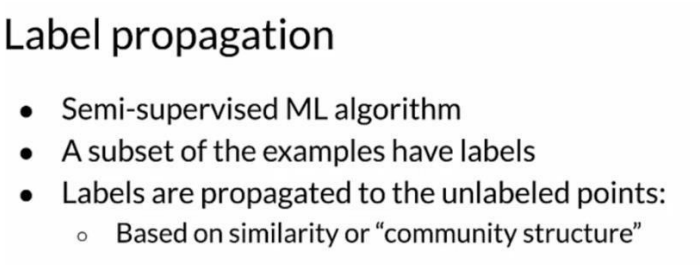


- Con el etiquetado semisupervisado, se empieza con un conjunto de datos relativamente pequeño que ha sido etiquetado por humanos.
- A continuación, combinará esos datos etiquetados con una gran cantidad de datos sin etiquetar, de los que deducirá las etiquetas de los datos sin etiquetar observando cómo se agrupan o estructuran las diferentes clases humanas etiquetadas dentro del espacio de características.

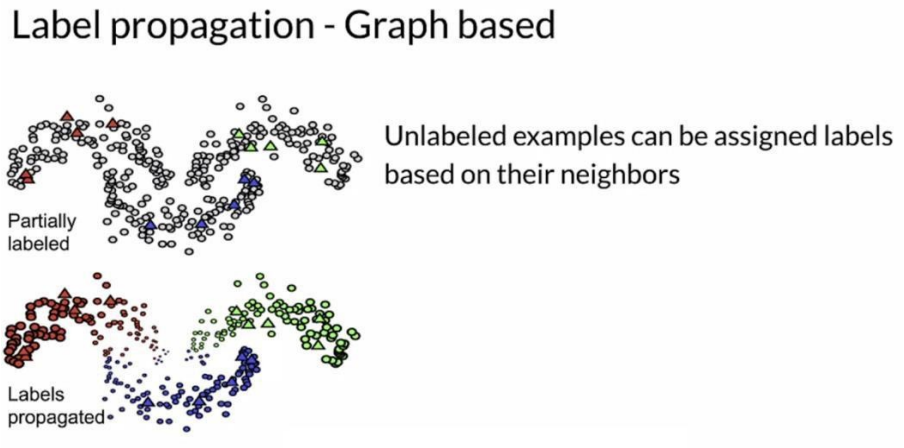
- A continuación, entrenará su modelo utilizando la combinación de los dos conjuntos de datos. Este método se basa en la **suposición de que las diferentes clases de etiquetas se agruparán o tendrán alguna estructura reconocible dentro del espacio de características.**



- El uso del etiquetado semisupervisado es ventajoso por dos razones principales
- La combinación de datos etiquetados y no etiquetados puede mejorar la precisión de los modelos de aprendizaje automático.
- La obtención de datos no etiquetados suele ser muy económica, ya que no requiere que las personas asignen etiquetas. A menudo, los datos sin etiquetar están fácilmente disponibles en grandes cantidades.



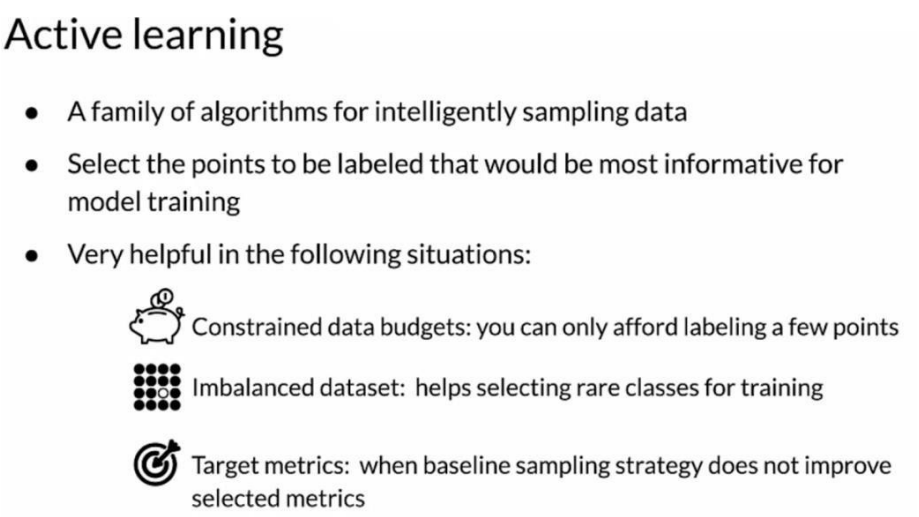
- **La propagación de etiquetas** es un **algoritmo que asigna etiquetas a ejemplos no etiquetados previamente**.
- Esto lo convierte en un algoritmo semisupervisado en el que un subconjunto de los puntos de datos tiene etiquetas. El algoritmo propaga las etiquetas a los puntos de datos sin etiquetas.
- Lo hace basándose en la **similitud** o la estructura de la comunidad de los puntos de datos etiquetados y los puntos de datos no etiquetados. Esta similitud o estructura se utiliza para asignar etiquetas a los datos no etiquetados.



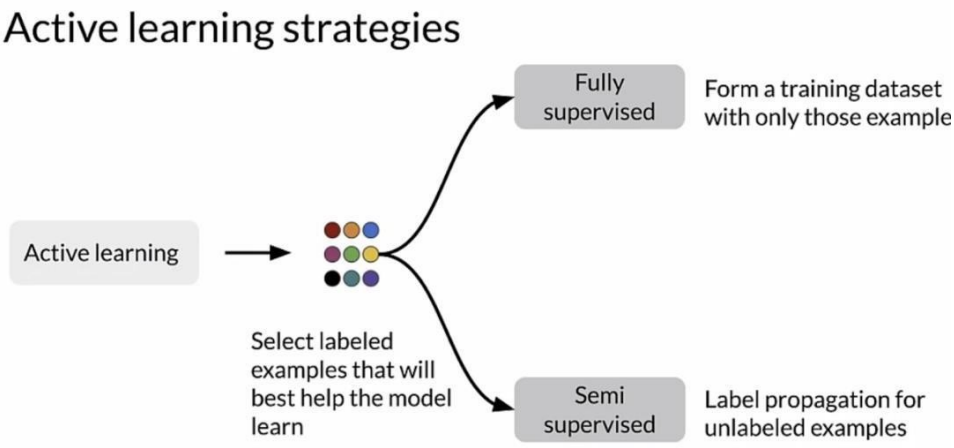
- Por ejemplo, está basado en un gráfico, y en esta figura puedes ver algunos datos etiquetados, los triángulos rojos, azules y verdes aquí, y un montón de datos sin etiquetar, que son los círculos grises.
- Con este método, se asignan etiquetas a los ejemplos no etiquetados basándose en sus vecinos.
- A continuación, las etiquetas se propagan al resto de los clústeres, como se muestra aquí por los colores.

- Debemos mencionar que hay muchas formas diferentes de hacer la propagación de etiquetas. La propagación de etiquetas basada en grafos es sólo una de las diversas técnicas.
- La propagación de etiquetas en sí misma se considera **aprendizaje transductivo**, lo que significa que estamos **mapeando a partir de los propios ejemplos sin aprender una función para el mapeo.**

Aprendizaje activo

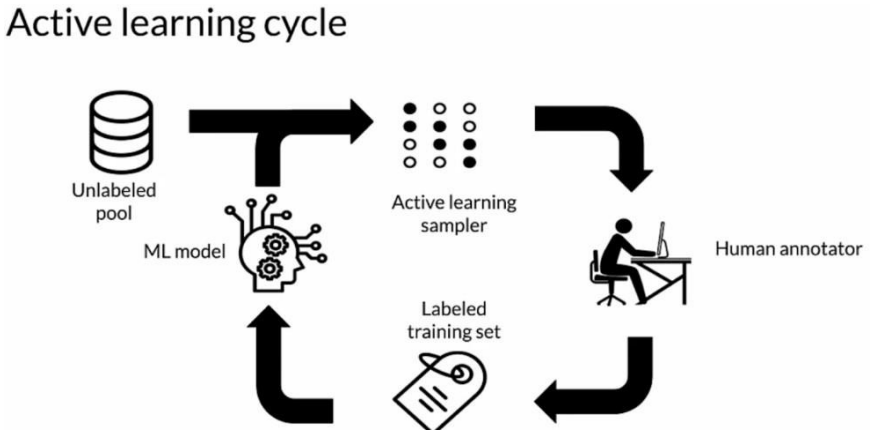


- El aprendizaje activo es una forma de muestrear los datos de forma inteligente.
- El muestreo inteligente **selecciona los puntos no etiquetados que aportan el mayor valor predictivo a su modelo**.
- Esto es muy útil en varios contextos: En primer lugar, un presupuesto de datos limitado. El etiquetado de los datos cuesta dinero, sobre todo cuando se recurre a expertos humanos para examinar los datos y asignarles una etiqueta, por ejemplo, en el ámbito de la sanidad.
- El aprendizaje activo ayuda a compensar este coste y esta carga.
- Si se dispone de un conjunto de datos desequilibrado, el aprendizaje activo es una forma eficaz de **seleccionar clases raras en la fase de entrenamiento**.
- Si las técnicas de muestreo estándar no ayudan a mejorar la precisión y otras métricas objetivo, el aprendizaje activo puede encontrar formas de alcanzar o ayudar a alcanzar la precisión deseada.

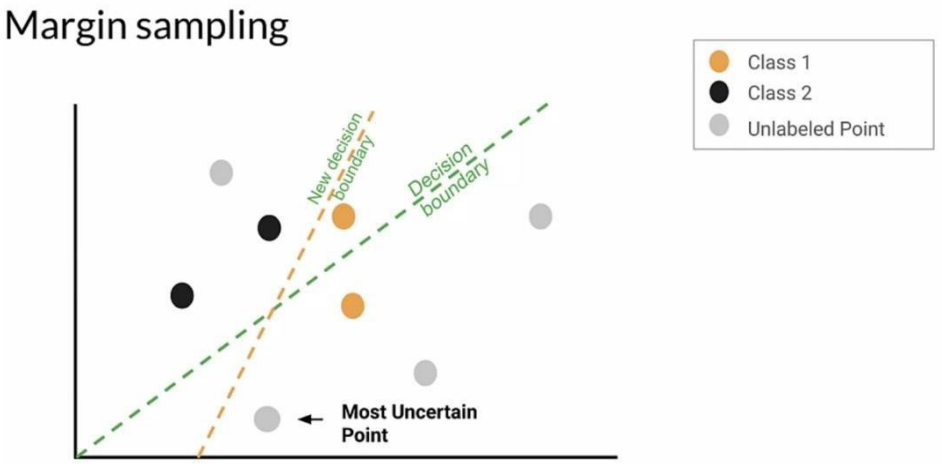


- La estrategia de aprendizaje activo funciona seleccionando los ejemplos **etiquetados** que mejor ayudarán al modelo a aprender.
- En un entorno totalmente supervisado, el conjunto de datos de entrenamiento consiste únicamente en los ejemplos que se han etiquetado.

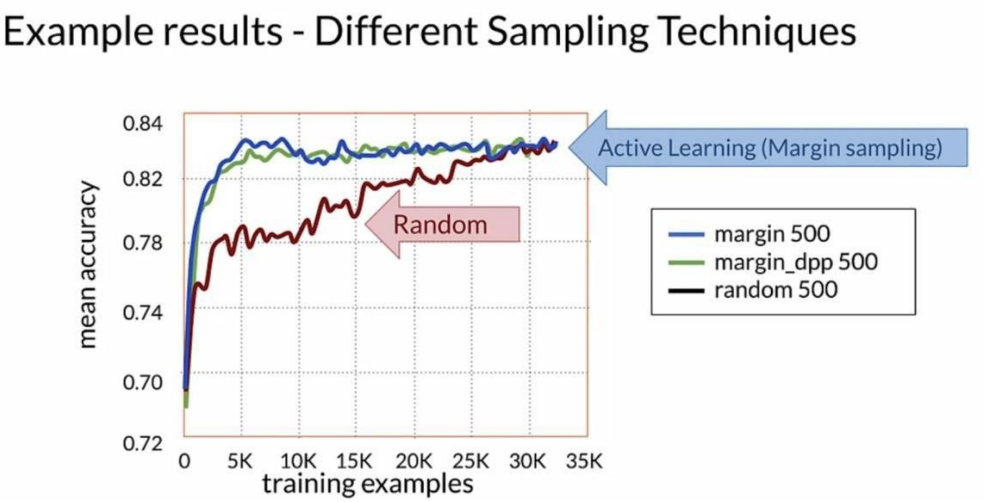
- En un entorno semisupervisado, se aprovechan esos ejemplos para realizar **la propagación de la etiqueta, por** lo que eso se suma al aprendizaje activo.



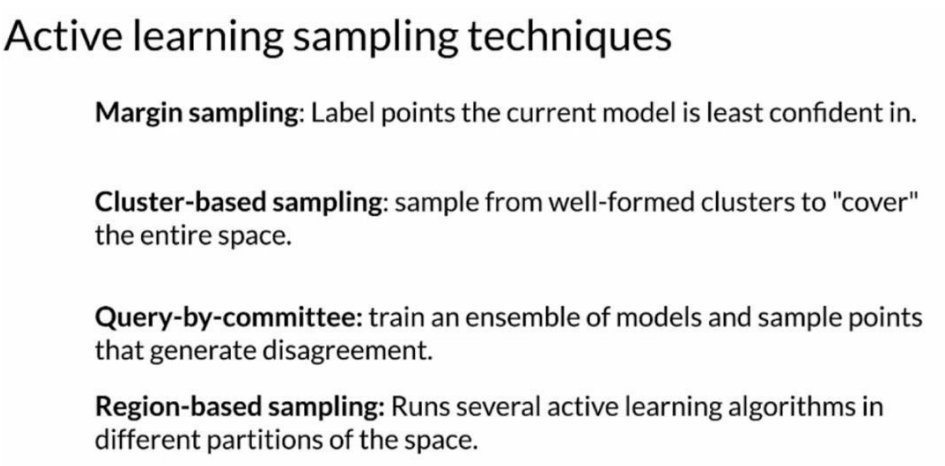
- Este es el típico ciclo de vida del aprendizaje activo: se empieza con un **conjunto de datos sin etiquetar** y, a continuación, el aprendizaje activo selecciona algunos ejemplos mediante un muestreo inteligente, del que hablaremos en un segundo, y luego se anotan los datos con anotadores humanos o aprovechando alguna otra técnica.
- Este procedimiento de anotación o etiquetado genera un conjunto de datos de entrenamiento etiquetado.
- Por último, se utilizan estos datos etiquetados para entrenar un modelo y hacer predicciones.
- El ciclo continúa, pero esto nos lleva a preguntarnos **cómo hacer un muestreo inteligente**.



- **El muestreo de márgenes** es una técnica muy utilizada para realizar un muestreo inteligente.
- En este ejemplo, los datos pertenecen a dos clases, además, son puntos de datos sin etiquetar.
- En este escenario, la estrategia más simple es, sólo vamos a entrenar un modelo clasificador lineal binario que lo hace simple. Vamos a utilizar un modelo lineal para el ejemplo.
- Vamos a entrenar eso en los datos etiquetados y eso nos va a dar un límite de decisión.
- Ahora, entre los datos no etiquetados, el **punto más incierto es el que está más cerca del límite de decisión**.
- **Con el aprendizaje activo, se seleccionará el punto más incierto para etiquetarlo** a continuación y añadirlo al conjunto de datos.
- Ahora, utilizando este nuevo punto de datos etiquetado como parte del conjunto de datos, se vuelve a entrenar el modelo para aprender un nuevo límite de clasificación.
- Moviendo el límite, el modelo aprende un poco mejor a separar esas clases.
- A continuación, encuentra el punto de datos más incierto, de nuevo, y repite el proceso **hasta que el modelo no mejore**.

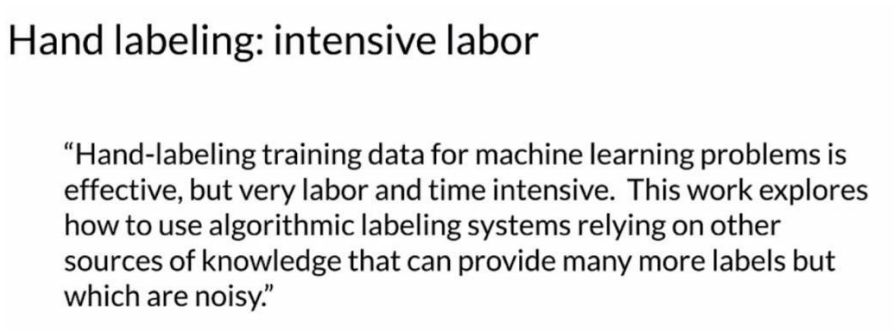


- Este gráfico muestra la precisión del modelo en función del número de ejemplos de entrenamiento para las técnicas de muestreo. La línea roja muestra los resultados de la simple selección de puntos al azar para etiquetar.
- La línea azul y verde muestra el rendimiento de dos algoritmos de muestreo de márgenes que utilizan el aprendizaje activo.
- Como puede ver, los métodos de muestreo de márgenes logran una mayor precisión con menos ejemplos de entrenamiento que la técnica de muestreo aleatorio.
- Pero, por supuesto, con el tiempo, a medida que un **mayor porcentaje de los datos no etiquetados se etiquetan con incluso utilizando el muestreo aleatorio, se pondrá al día con el muestreo de margen**, como vemos aquí, pero **requiere muchos más datos para ser etiquetados**.



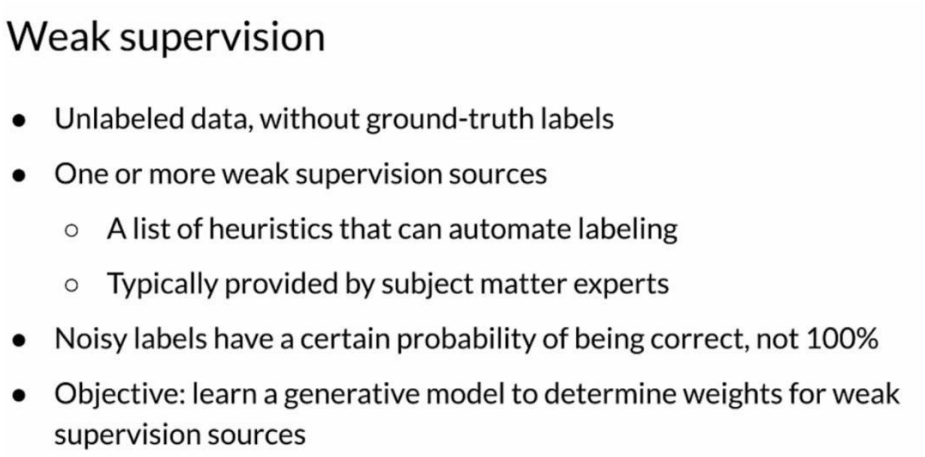
- Existen varias técnicas de muestreo de aprendizaje activo habituales.
- Con el muestreo de márgenes, como acabamos de ver, se asignan etiquetas a los puntos más inciertos en función de su distancia al límite de decisión.
- Con el muestreo basado en clústeres, se selecciona un conjunto diverso de puntos mediante el uso de métodos de clúster sobre su espacio de características.
- Con la consulta por comité, se entrenan varios modelos y se seleccionan los puntos de datos con mayor desacuerdo entre esos modelos.
- Por último, el muestreo por regiones es un algoritmo relativamente nuevo. A grandes rasgos, este algoritmo funciona dividiendo el espacio de entrada en regiones separadas y ejecutando un algoritmo de aprendizaje activo en esas regiones.

Supervisión deficiente



- Empecemos con la supervisión débil, la última técnica avanzada de etiquetado que vamos a tratar.
- Esta es una cita de Jeff Dean, que dirige el grupo de investigación de aprendizaje automático en Google.
- Está comentando el coste, tanto en tiempo como en trabajo, de etiquetar los datos.
- También comenta específicamente el trabajo para crear enfoques algorítmicos el etiquetado de datos que se basan en otras fuentes de información, para producir etiquetas ruidosas.
- La supervisión débil, de la que hablaremos ahora, es la principal forma de hacer este tipo de etiquetado.

- La supervisión débil es una forma de generar etiquetas utilizando información de una o más fuentes.
- Suelen ser expertos en la materia, y normalmente están diseñando heurísticos.
- Las etiquetas resultantes son **ruidosas** en lugar de las etiquetas deterministas a las que estamos acostumbrados.
- Más concretamente, la supervisión débil comprende una o más distribuciones condicionales ruidosas sobre datos no etiquetados.
- Y el objetivo principal, es aprender un modelo generativo que determine la relevancia de cada una de estas fuentes ruidosas.



- Así pues, se empieza con datos **no etiquetados** de los que no se conocen las verdaderas etiquetas.

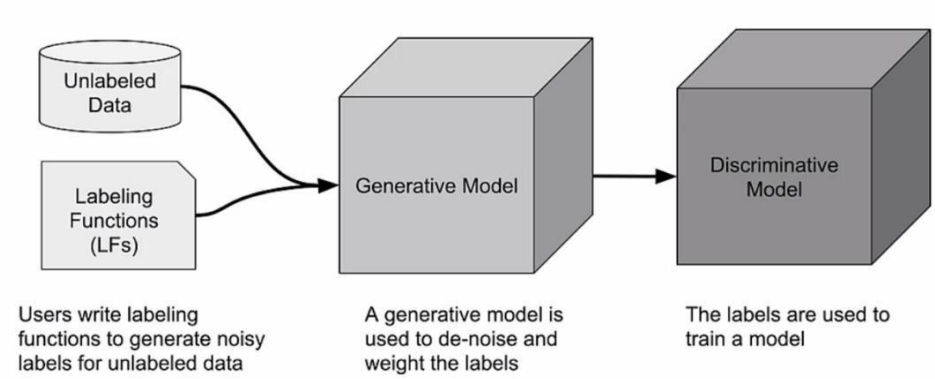
- Luego se añade a la mezcla una o más fuentes **de supervisión débil**. Estas fuentes son una **lista de procedimientos heurísticos** que implementan un etiquetado automatizado ruidoso e imperfecto.
- Los expertos en la materia son las fuentes más comunes para diseñar estas heurísticas, que suelen consistir en un conjunto de cobertura y una probabilidad esperada de la etiqueta verdadera sobre el conjunto de cobertura.
- **Utiliza funciones heurísticas o de etiquetado en lugar de etiquetas deterministas**
- Por ruidoso queremos decir que la etiqueta tiene una cierta probabilidad de ser correcta, en lugar de la certeza del 100 a la que estábamos acostumbrados, para las etiquetas en nuestros datos etiquetados.
- El objetivo principal es aprender la **fiabilidad** de cada fuente de supervisión, y esto se hace **entrenando un modelo generativo.**
- En general, aprovecha las aportaciones de los expertos en la materia y crea y estructura conjuntos de datos de entrenamiento mediante la programación

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

- El marco Snorkel salió de Stanford en 2016 y es el más utilizado para implementar la supervisión débil.
- No requiere etiquetado manual, por lo que el sistema construye y gestiona de forma programada el conjunto de datos de entrenamiento.
- Snorkel proporciona herramientas para limpiar, modelar e integrar los datos de entrenamiento resultantes que surgen a través de la tubería de supervisión débil.
- Snorkel utiliza una novedosa técnica de base teórica para realizar el trabajo con rapidez y eficacia.
- Y además, Snorkel también ofrece el aumento y el corte de datos.

Data programming pipeline in Snorkel



- Con Snorkel, empezarás con datos sin etiquetar y aplicarás funciones de etiquetado. Que son las heurísticas que están diseñadas por expertos en la materia, para generar etiquetas ruidosas.
- A continuación, se utilizará un **modelo generativo para eliminar el ruido de las etiquetas y asignar pesos de importancia a las diferentes funciones de etiquetado, es decir,** determinar los pesos de las **fuentes de supervisión**
- Por último, entrenará un modelo **discriminante**, es decir, su modelo, con las etiquetas desnotizadas.

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
```

- Así que echemos un vistazo a cómo podrían ser un par de funciones de etiquetado simples en el código. Aquí mostraré una forma sencilla de crear funciones para etiquetar spam usando snorkel.
- El primer paso es importar la función de etiquetado de snorkel. Y luego con esta etiqueta funcional mensajes de spam, si contiene la palabra **mi**.
- Esto es sólo un ejemplo es un ejemplo fácil de mostrar, por lo que los mensajes con 'mi' no tienen que ser spam.
- En caso contrario, la función devuelve abstención, lo que significa que **no** tiene **opinión** sobre cuáles deben ser las etiquetas.
- La segunda función etiqueta un mensaje como spam si tiene más de cinco palabras.
- Así que lo que estamos mostrando aquí es que usted estaba **usando múltiples funciones de etiquetado para tratar de llegar a lo que la etiqueta debe ser.**

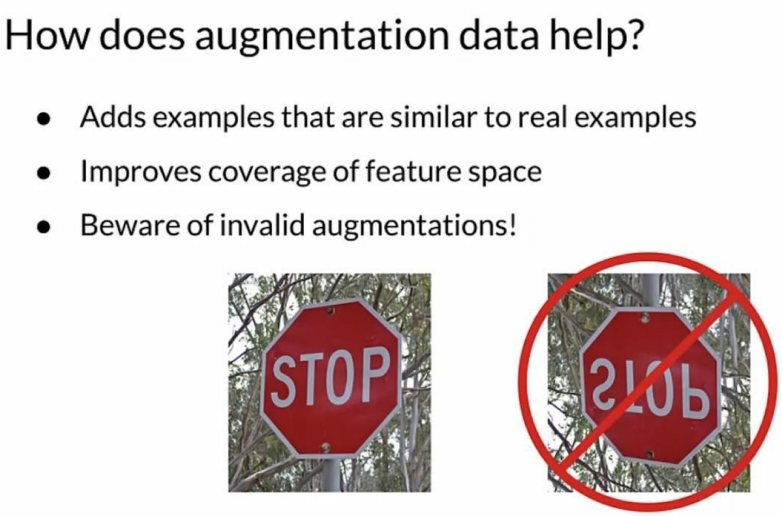
Key points

- 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

- El aprendizaje supervisado requiere datos etiquetados, pero el etiquetado de datos suele ser un proceso caro, difícil y lento.
- El aprendizaje semi-supervisado es una forma posible de añadir etiquetas a los datos no etiquetados. Por tanto, este método se sitúa entre el aprendizaje no supervisado y el supervisado.
- Funciona combinando una pequeña cantidad de datos etiquetados con una gran cantidad de datos sin etiquetar, y esto mejora la precisión del aprendizaje.
- El aprendizaje activo es otro método avanzado de etiquetado. Se basa en técnicas de muestreo de inteligencia que
  seleccionar los ejemplos más importantes para etiquetarlos y añadirlos al conjunto de datos.
- El aprendizaje activo mejora la precisión de la predicción y minimiza el coste del etiquetado.
- El último método que exploró fue la supervisión débil.
- La supervisión débil aprovecha las fuentes de etiquetas ruidosas, limitadas o inexactas dentro de un entorno de aprendizaje supervisado, para **comprobar la precisión del etiquetado**.
- Snorkel es un sistema compacto y fácil de usar para gestionar todas estas operaciones para la supervisión débil y establecer conjuntos de datos de entrenamiento utilizando la supervisión débil.

Aumento de datos

- Anteriormente exploró métodos para obtener más datos etiquetados mediante el etiquetado de datos no etiquetados.
- Pero otro método es aumentar los datos existentes para crear más ejemplos etiquetados.
- Con el aumento de datos se amplía el conjunto de datos añadiendo copias ligeramente modificadas de los datos existentes, o se crean nuevos datos sintéticos a partir de los datos existentes.
- Con los datos existentes, es posible crear más datos haciendo pequeñas alteraciones o perturbaciones en las muestras existentes.
- Tareas sencillas como giros o rotaciones e imágenes son una forma fácil de duplicar o triplicar el número de imágenes en un conjunto de datos.



- El aumento de datos es una forma de mejorar el rendimiento de su modelo, añadir nuevos ejemplos válidos que caen en **regiones del espacio de características** que no están cubiertas por sus ejemplos reales y añadir ejemplos válidos de esta manera, mejora la cobertura de su espacio de características.
- Tenga en cuenta que si añade ejemplos no válidos, corre el riesgo de aprender la respuesta equivocada o, como mínimo, de introducir ruido no deseado. Así que ten cuidado de aumentar tus datos sólo de forma válida.

- CIFAR es el Instituto Canadiense de Investigación Avanzada. El conjunto de datos CIFAR-10 es una colección de imágenes utilizadas habitualmente para entrenar modelos de aprendizaje automático y de visión por ordenador.
- Es uno de los conjuntos de datos más utilizados para la investigación del aprendizaje automático.
- CIFAR-10 contiene 60.000 imágenes en color de 32 por 32. Hay 10 clases diferentes con 6000 imágenes en cada clase.

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

- Así que vamos a echar un vistazo práctico al aumento de datos con el conjunto de datos CIFAR-10.
- Añadiremos un borde acolchado a la imagen, y recortaremos las imágenes acolchadas a una imagen de 32 x 32 y voltearemos o rotaremos las imágenes acolchadas y recortadas basándonos en el lanzamiento de una moneda o en una variable aleatoria.
- Esto es para que el modelo no se ajuste en exceso a rotaciones o escalas particulares.

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
```

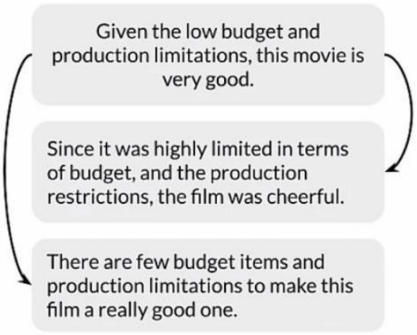
- Dado que las imágenes en color son tensores, tensorflow proporciona funciones muy útiles para realizar aumentos en conjuntos de datos de imágenes.
- Veamos un trozo de código que encapsula los pasos para hacer un giro a la izquierda y a la derecha.
- En primer lugar, definimos una función que se utilizará para aumentar los conjuntos de datos de imágenes.
- Luego usaremos tf.image.resize con crop_or_pad que permite redimensionar o recortar la imagen.
- En este caso vamos a rellenar.
- tf.image.random crop genera un recorte de tamaño alto por ancho en todos los canales.
- Y luego tf.image.random flip left, right hace lo mismo con las rotaciones como voltear a la izquierda o a la derecha.
- Así, la función devuelve la imagen alterada que se añadirá al conjunto de datos.

Augmented examples

- Aquí hay algunos ejemplos después de aplicar el relleno, el recorte y el giro a la izquierda y a la derecha.

Other Advanced techniques

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



- Aparte de la simple manipulación de imágenes, existen otras técnicas avanzadas para el aumento de datos, como el aumento de datos semi-supervisado o UDA.
- O bien otros métodos que utilizan el aumento de datos basado en políticas como AutoAugment.
- A la derecha, hay un ejemplo de una crítica de cine, y luego, utilizando el aumento de la política, generamos un ejemplo variante.
- Así que tenemos un nuevo ejemplo perfectamente válido que hemos podido generar con el aumento.
- Podemos volver a hacerlo y generar otro.

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

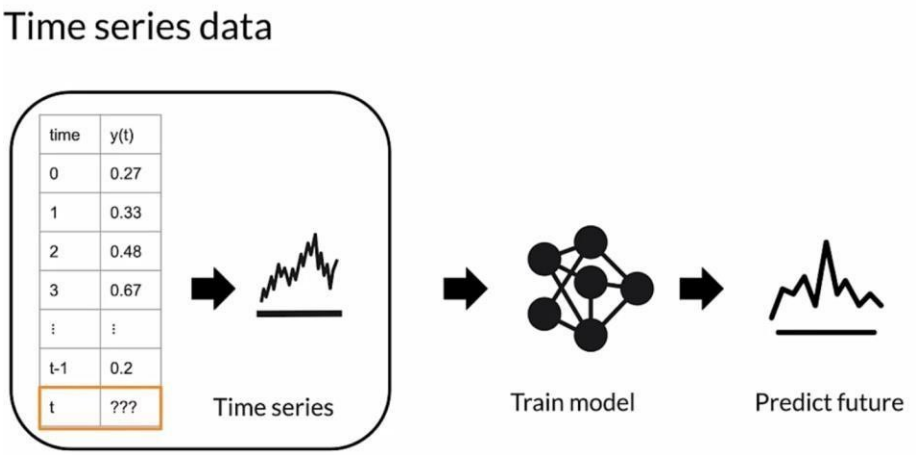
Series temporales

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

- En primer lugar, vamos a mencionar brevemente algunos tipos de datos diferentes en función del problema que está trabajando en los datos que tiene.
- Es posible que trabaje con diferentes tipos de datos, cada uno de los cuales requiere un preprocesamiento diferente y, en algunos casos, diferentes técnicas de modelado.

- Por ejemplo, puedes trabajar con imágenes, vídeo, texto, audio o datos de series temporales. Hay demasiados tipos de datos como estos para discutirlos a fondo en esta clase. Así que hemos proporcionado algunos materiales opcionales para que puedas explorar algunos de ellos por tu cuenta.
- Hay un cuaderno opcional para trabajar con el conjunto de datos de imágenes CIFAR-10, y hay otros dos cuadernos para trabajar con datos de series temporales.
- Uno de ellos es sobre datos meteorológicos y el otro contiene datos de un acelerómetro y otros sensores disponibles en la mayoría de los teléfonos móviles.
- De todos los tipos de datos enumerados aquí, las series temporales son probablemente las que la mayoría de los desarrolladores conocen menos. Así que empecemos por repasar los aspectos clave de las series temporales.



- Las series temporales son una secuencia de puntos de datos en el tiempo, a menudo procedentes de eventos registrados en los que la dimensión temporal indica cuándo se produjo el evento.
- Pueden o no estar ordenados en los datos brutos, pero casi siempre se quiere que estén **ordenados por tiempo para la modelización.**
- Queremos predecir el valor en una situación típica. Queremos predecir el valor de y en el momento t en algún momento del futuro basándonos en mediciones anteriores.
- El objetivo es entrenar un modelo que prediga los resultados futuros con una precisión aceptable.

- Karl Kristian Steincke señaló que es difícil hacer predicciones, especialmente sobre el futuro.
- El corolario de esto, por supuesto, es que es relativamente fácil hacer predicciones sobre el pasado, puesto que ya ha sucedido.

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

- La previsión de series temporales hace exactamente eso: intenta predecir el futuro.
- Lo hace analizando datos del pasado. Para ello se necesitan datos de índices temporales, por ejemplo, para predecir la temperatura futura en un lugar determinado. Podríamos utilizar otras variables meteorológicas, como la presión atmosférica, el viento, la dirección y la velocidad, etc., que se hayan registrado previamente en algún momento del pasado.
- Veamos un ejemplo concreto de cómo hacer predicciones meteorológicas.

Time series dataset: Weather prediction

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

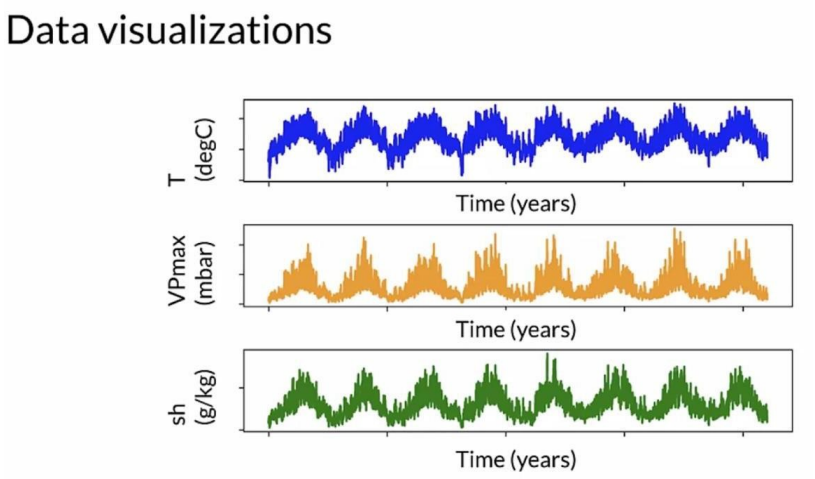
- to preprocess time series data with TensorFlow Transform
- to convert data into sequences of time steps:
  - Making data ready to train a long short-term memory (LSTM) recurrent neural network (RNN)

- En este ejemplo, se va a trabajar con un conjunto de datos de series temporales que fue registrado por el Instituto Max Planck de biogeoquímica.
- Este conjunto de datos contiene 14 características diferentes, como la temperatura del aire, la presión atmosférica y la humedad. Se registraron cada 10 minutos a partir de 2003.
- Tu trabajo consiste en preprocesar las características con el pipeline TFX, y convertir los datos en secuencias temporales
- Este formato es necesario para entrenar redes neuronales recurrentes, como los modelos de memoria a corto plazo o LSTM.

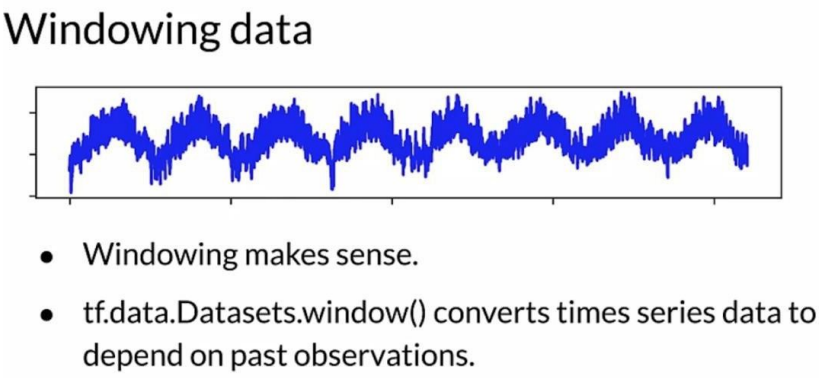
Weather time series dataset

- 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.

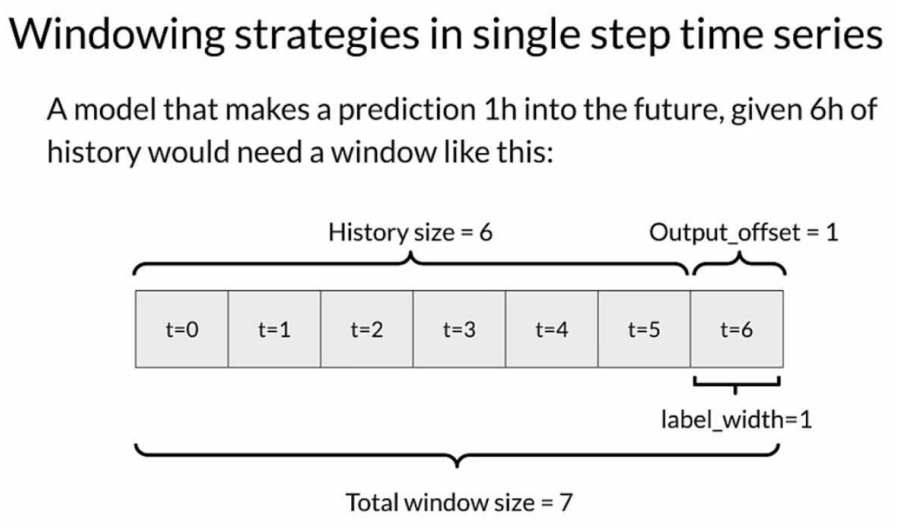
- Veamos con más detalle cómo se organizan y recogen los datos. Hay 14 variables, que incluyen mediciones relacionadas con la humedad, el viento, la dirección y la velocidad, la temperatura y la presión atmosférica.
- El objetivo de la predicción es la temperatura.
- La tasa de muestreo es de una observación cada 10 minutos. Por lo tanto, tiene seis observaciones por hora y 100 y 144 en un Día determinado.



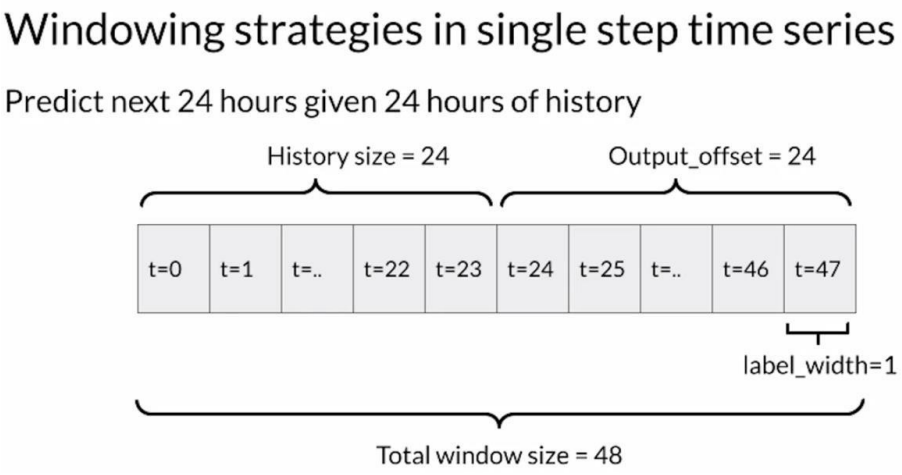
- Estos son los gráficos de algunas de las características a lo largo del tiempo, y la variable objetivo T, que es la temperatura.
- Puedes ver que hay un patrón que se repite en intervalos de tiempo específicos.
- Aquí hay una clara estacionalidad, que debemos tener en cuenta a la hora de hacer ingeniería de características para estos datos.
- Deberíamos considerar la posibilidad de hacer una **descomposición estacional**, pero para simplificar las cosas en este ejemplo, no lo haríamos.



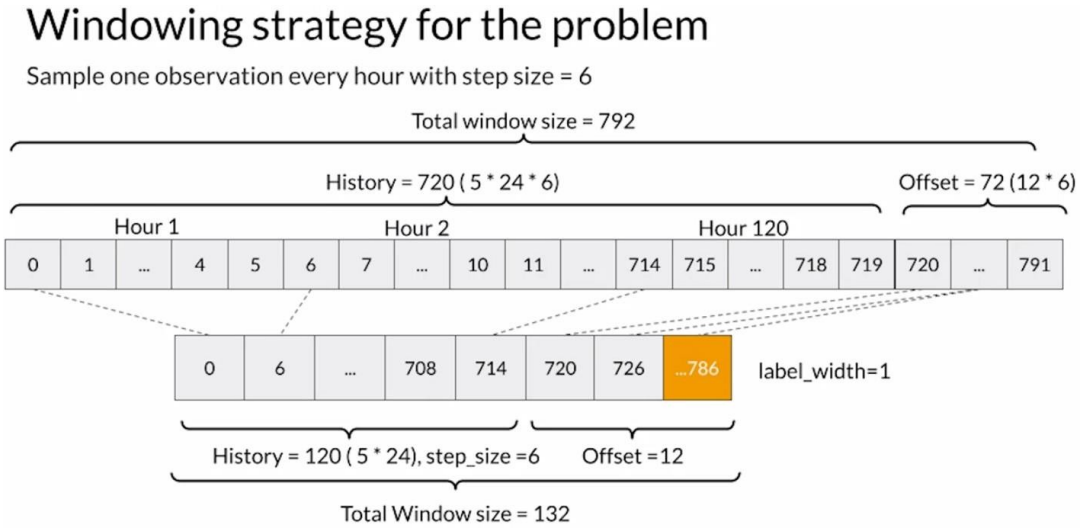
- Los datos muestran una clara periodicidad o estacionalidad, que en este caso se debe probablemente a la típica progresión estacional a lo largo del año.
- Pero recuerde que la estacionalidad es una característica de los datos, y a menudo no tiene nada que ver con las estaciones reales del año. En realidad, se trata de la periodicidad.
- El uso de una **estrategia de ventanas** para ver las dependencias con los datos pasados parece una toma natural.
- Y, por suerte, el TFX tiene esta funcionalidad ya incorporada en el portátil.
- Verás la función window() tf.data, y la usaremos para agrupar el conjunto de datos en ventanas. Así que vamos a ver exactamente cómo funciona esto usando los datos meteorológicos como ejemplo.



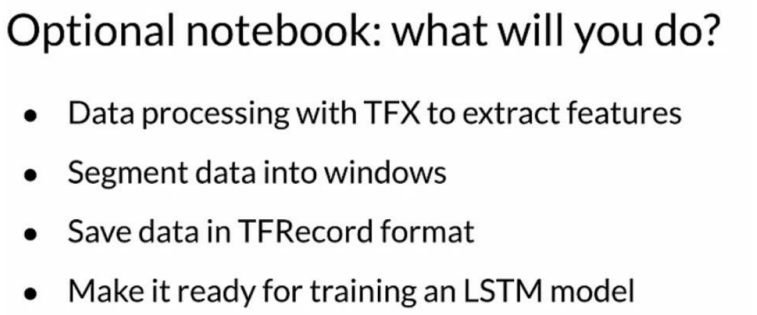
- Las estrategias de ventanas en las series temporales son bastante importantes, y son algo exclusivo de las series temporales y tipos similares de datos secuenciales.
- En este ejemplo, usted tiene un modelo que puede utilizar para hacer una predicción una hora en el futuro y dado un historial de seis horas, utilizaremos una ventana deslizante con un tamaño de ventana de seis y un desplazamiento de uno.
- Así que el total de la ventana es de 7, 6 más uno.



- En otro ejemplo, suponga que hace una predicción a 24 horas en el futuro dadas 24 horas de historia.
- Así que en ese caso el tamaño de tu historia es 24, el tamaño del offset también es 24.
- Así que podría utilizar un tamaño de ventana total de 48 que sería la historia más el desplazamiento de salida.
- También es importante tener en cuenta cuándo es el ahora, y **no** incluir datos en el futuro, lo que se conoce como viaje en el tiempo.
- En este ejemplo, si ahora está en T es igual a 24 entonces tenemos que tener cuidado de no incluir los datos de T=25 a T=47 en nuestros datos de entrenamiento.
- Podríamos hacerlo en la ingeniería de características o reduciendo la ventana para incluir sólo el historial y la etiqueta.



- Hablemos de la estrategia de muestreo. Ya sabe que hay seis observaciones por hora. En nuestro ejemplo, una observación cada 10 minutos
- En un día, habrá 144 observaciones. Si tomas **cinco días de observaciones pasadas** y haces una predicción a seis horas en el futuro.
- Esto significa que el tamaño de nuestra historia será cinco veces 144, o 720 observaciones, y el desplazamiento de salida será 12 veces seis, o 72.
- Así, el tamaño total de la ventana en el tiempo es de 792, ya que es poco probable que las observaciones en una hora cambien demasiado.
- Vamos a muestrear una observación por hora. Así que en lugar de una cada 10 minutos vamos a pasar a una por hora. Así que digamos que podemos tomar la primera observación de la hora como muestra o incluso mejor, se puede tomar la mediana de las observaciones de cada hora
- Entonces el tamaño de nuestra historia se convierte en cinco veces 24 veces uno o 120 y nuestro desplazamiento de salida será seis.
- Así, el tamaño total de nuestra ventana pasa a ser de 126.
- Así, hemos reducido el tamaño de nuestro vector de características de 792 a 126, ya sea **muestreando dentro de cada hora** o agregando los datos de cada hora tomando la mediana.
- Así que sé que los números pueden ser un poco confusos aquí, pero el punto importante es pensar en cómo estamos tratando los datos en nuestra ventana.
- Lo anterior es un ejemplo de tomar una ventana finita de datos más **un muestreo descendente** para las **señales lentas que varían en el tiempo** .



- ¿Qué harás en el cuaderno opcional?
- En ese cuaderno, vas a empezar con el preprocesamiento del conjunto de datos de las series temporales del tiempo utilizando TFX.
- Y utilizará la ventana TF.data.dataset para crear las ventanas que utilizará para construir sus ejemplos, y guardará la transformación y los datos preprocesados en formato de registro TF.
- Y por último, crearás conjuntos de datos de entrenamiento y de prueba de manera que las características puedan ser fácilmente utilizadas para entrenar un modelo LSTM utilizando TensorFlow o Keras con estrategias de ventanas o en algún otro marco.
- El entrenamiento no se implementa en el cuaderno ya que nos centramos en los datos en sí.

Sensores y señales

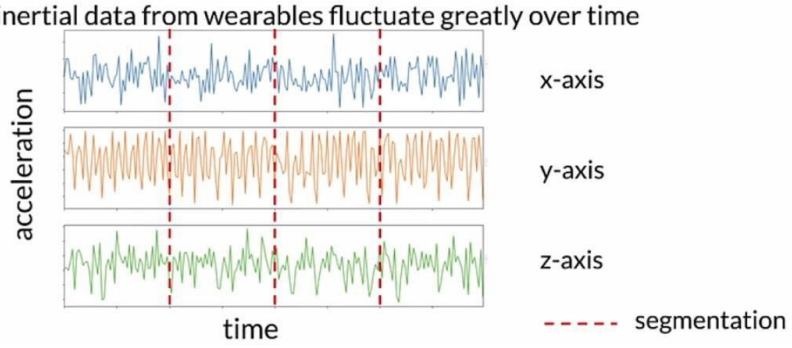
Sensors and Signals

- 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

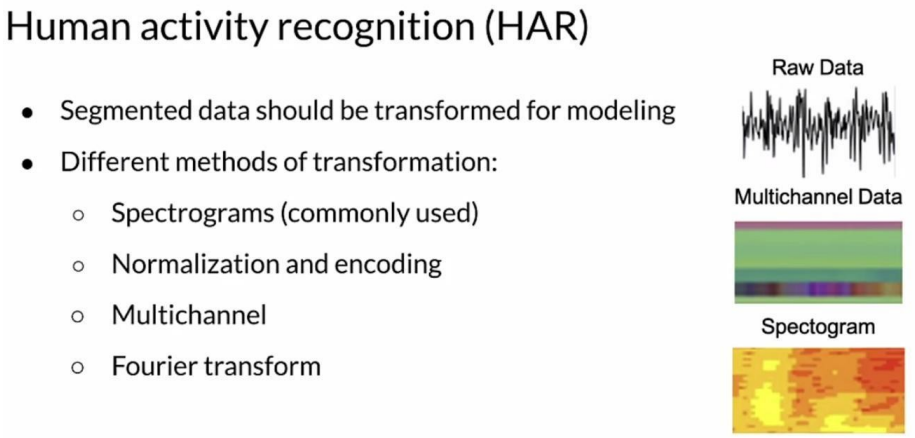
- Revisemos los datos generados por los sensores, que suelen denominarse señales.
- Empecemos por definir la terminología importante mediante un ejemplo en el que se utilizan los datos del acelerómetro recogidos de los teléfonos inteligentes.
- El acelerómetro es un sensor
- Las señales son secuencias de datos recogidas por un sensor en tiempo real
- Los puntos de datos están indexados por marcas de tiempo. Por lo tanto, se trata de datos de series temporales y vamos a utilizar el análisis de series temporales.
- Veamos un problema concreto, el **reconocimiento de la actividad humana o HAR**: ¿Podemos inferir la actividad a partir de los patrones del acelerómetro en el tiempo?

Human activity recognition (HAR)

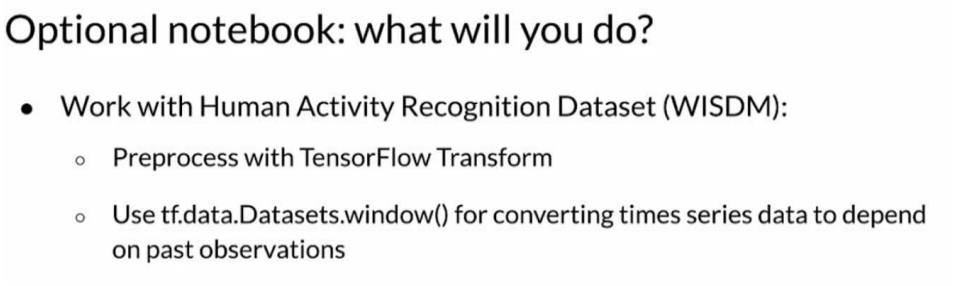
- HAR tasks require segmentation operations
  - Raw inertial data from wearables fluctuate greatly over time



- Veamos algunos de los retos del reconocimiento de actividades.
- La segmentación adecuada de los datos de los sensores es la clave para el éxito del reconocimiento de la actividad.
- Esto es similar a la estrategia de ventanas que acabamos de discutir para los datos meteorológicos.
- Los datos inerciales de este tipo fluctúan en gran medida a lo largo del tiempo, por lo que la **segmentación es clave para detectar los trozos de actividad adecuados**.
- El conjunto de datos de aceleración en función del tiempo se separa en segmentos durante el proceso de segmentación de datos. Todas las operaciones siguientes relacionadas con HAR, como la extracción de características, la clasificación y la validación, etc., se basan en estos segmentos.
- La longitud de los segmentos depende del contexto de la aplicación y de la frecuencia de muestreo de los sensores
- Los segmentos de HAR suelen durar entre 1 y 10 segundos.



- Ahora que los datos están segmentados, hablemos de las transformaciones típicas. El segmento de datos debe ser transformado para el modelado porque ayuda a la precisión y el rendimiento del modelo
- Hay diferentes maneras de transformar los datos.
- Un espectrograma de una señal inercial es una nueva representación de la señal en función de la frecuencia y el tiempo.
- La representación del espectrograma es un potente método para extraer características interpretables que representan el
  diferencias de intensidad entre distintos puntos de datos inerciales.
- El espectrograma muestra los cambios de amplitud de cada frecuencia en función del tiempo
- Otras opciones son la normalización y la codificación, el procesamiento multicanal y la aplicación de la transformada de Fourier.



- En el cuaderno opcional, intentarás **reconocer las actividades humanas a partir de los datos de los sensores.**
- El conjunto de datos de minería de datos de sensores inalámbricos contiene datos recogidos en condiciones controladas de laboratorio.
- El conjunto de datos es un banco de pruebas para el reconocimiento de la actividad utilizando los datos del acelerómetro del teléfono móvil,
- En este cuaderno, usted preprocesará los datos dentro de un pipeline TFX. Allí, utilizarás tf.data.Dataset.window para agrupar los datos en ventanas, que pueden ser utilizadas con un modelo RNN o similar.

Referencias

- Etiquetado a mano
- Supervisión deficiente
- Snorkel
- ¿Cómo se obtienen más datos?
- Técnicas avanzadas
- Imágenes en tensorflow
- https://www.cs.toronto.edu/~kriz/cifar.html
- https://www.tensorflow.org/datasets/catalog/cifar10
- Conjunto de datos meteorológicos
- Reconocimiento de la actividad humana
- Propagación de etiquetas: Iscen, A., Tolias, G., Avrithis, Y., & Chum, O. (2019). Propagación de etiquetas para el aprendizaje profundo semi-supervisado. https://arxiv.org/pdf/1904.04717.pdf