Engineer

Aprendizaje automático

Crear software que aprende una tarea a partir de ejemplos

Qué es

El aprendizaje automático consiste en crear software que aprende una tarea a partir de ejemplos, en lugar de reglas que un programador escribe explícitamente. Se le proporcionan muchos casos cuya respuesta ya se conoce; encuentra una relación entre las entradas y las respuestas y la utiliza para predecir la respuesta de cada caso nuevo. El nombre es anterior a casi todo el software que hoy lo utiliza: A. L. Samuel lo incluyó en el título de un estudio del juego de las damas que IBM Journal of Research and Development publicó en julio de 1959.

El vocabulario se concreta fácilmente con scikit-learn, la biblioteca Python de código abierto cuya documentación sigue este artículo. Los ejemplos se organizan en una tabla. Cada fila es una muestra, como una factura, un cliente o una lectura de un sensor, y cada columna es una característica: un número que describe la muestra. La respuesta que se quiere que aprenda el software es la variable objetivo. En la terminología de scikit-learn, lo que aprende es un estimador. Todo estimador tiene un método `fit` que aprende de las filas que se le muestran. Los estimadores predictivos utilizan `predict` para dar respuestas sobre filas nuevas; los transformadores utilizan `transform` para producir una tabla de muestras transformada.

Dos familias abarcan la mayor parte del trabajo. En el aprendizaje supervisado, cada fila de entrenamiento incluye su variable objetivo. Si es una categoría, como pagada a tiempo o con retraso, la tarea es de clasificación; si es una cantidad, como los días hasta el pago, es de regresión. En el aprendizaje no supervisado no hay variable objetivo: el software busca estructuras por sí mismo, por ejemplo agrupando clientes con comportamientos similares. La documentación de scikit-learn dedica un capítulo a cada familia, seguido de otro sobre selección y evaluación de modelos, donde se concentra la mayor parte de la dificultad práctica.

El aprendizaje automático, al menos en este sentido, no designa cualquier software que parezca inteligente. Una regla que marca todas las facturas por encima de un importe fijo la ha escrito una persona y hace exactamente lo que dice. Un modelo que marca facturas ha deducido su regla de facturas anteriores: puede equivocarse de maneras que nadie escribió y su comportamiento cambia cuando cambian los datos de los que aprende.

Utilice el concepto en una decisión

Aplique «¿Qué es el aprendizaje automático?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. Un agente conectado a los sistemas que utiliza, probado con casos definidos por sus especialistas, supervisado por su personal y sin depender de un único modelo.

Explore nuestros servicios Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. Motor de 268 controles para el repositorio — 268 controles de commit y 146 pruebas de los propios controles

Motor de 268 controles para el repositorio Póngase en contacto →

Por qué importa

Algunas tareas son mucho más fáciles de mostrar con ejemplos que de describir. Nadie puede escribir una regla completa para determinar qué facturas se pagarán tarde, pero la mayoría de las empresas conserva años de facturas con el resultado registrado. El aprendizaje automático convierte ese historial en una regla operativa y puede hacer lo mismo con muchas preguntas de este tipo a la vez; por eso ha pasado de la investigación al software empresarial habitual.

El precio es que los datos hacen ahora el trabajo que antes hacía el código. Si el historial contiene errores, duplicados o lagunas, el modelo también los aprende, sin que nada en el código lo revele. D. Sculley y nueve colegas de Google describieron las consecuencias en *Hidden Technical Debt in Machine Learning Systems*, publicado en las actas de NIPS 2015. Su primera figura representa el código de aprendizaje como una pequeña caja negra en el centro de un sistema mucho mayor; el pie describe la infraestructura circundante como vasta y compleja. En un pasaje estiman que un sistema maduro puede acabar con un máximo del 5 % de código de aprendizaje automático y un mínimo del 95 % de código de enlace que introduce y extrae datos de paquetes de propósito general.

También sostienen que depender de datos cuesta más que depender de código. Los compiladores y los sistemas de compilación rastrean qué código depende de cuál; las herramientas equivalentes para datos son mucho menos comunes, por

lo que una columna puede cambiar de significado sin que nada lo detecte. Denominan CACE a un principio, por «Changing Anything Changes Everything», es decir, cambiar cualquier cosa lo cambia todo: si cambia la distribución de una entrada, también puede cambiar la forma en que el modelo utiliza todas las demás.

Por eso, crear un modelo es barato y mantenerlo es caro. El trabajo que determina su utilidad se centra sobre todo en los datos: de dónde proceden, si cada cliente aparece una o tres veces, si el resultado registrado es correcto y quién detectará que un sistema anterior en el flujo empieza a enviar algo distinto.

Cómo se aplica en la práctica

Pensemos en una empresa que quiere saber, el día en que emite una factura, cuáles tienen probabilidades de pagarse tarde, para llamar a esos clientes antes del vencimiento, en lugar de después. Una exportación contiene una fila por factura anterior, ordenada por fecha de emisión, con cuatro columnas conocidas el día de emitirla: el importe, los días de crédito concedidos, cuántas facturas recibió el cliente durante los doce meses anteriores y cuántas se pagaron tarde. Una quinta columna registra qué ocurrió: 1 si se pagó tarde y 0 si se pagó a tiempo.

El siguiente script se escribió para este artículo y se contrastó línea por línea con la documentación de scikit-learn 1.9.1 y NumPy 2.5 en octubre de 2026. Entrena un modelo con las facturas más antiguas y lo prueba con la cuarta parte más reciente del conjunto:

```
# invoices.csv: one row per invoice, sorted by issue date, numbers only.
# The last column is 1 if the invoice was paid late, 0 if on time.
import numpy as np
from sklearn.dummy import DummyClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import balanced_accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

data = np.loadtxt("invoices.csv", delimiter=",", skiprows=1)
X, y = data[:, :-1], data[:, -1].astype(int)

# Hold the newest quarter back: nothing learns from it.
X_old, X_new, y_old, y_new = train_test_split(
```

```
    X, y, test_size=0.25, shuffle=False
)

model = make_pipeline(StandardScaler(), LogisticRegression())
baseline = DummyClassifier(strategy="most_frequent")
for name, candidate in [("model", model), ("baseline", baseline)]:
    candidate.fit(X_old, y_old)
    score = balanced_accuracy_score(y_new, candidate.predict(X_new))
    print(name, round(score, 3))
```

Cuatro decisiones importan más que la elección del algoritmo.

La cuarta parte más reciente se separa antes de que nada aprenda. Con `shuffle=False`, `train_test_split` mantiene las filas ordenadas y coloca la última cuarta parte en el conjunto de prueba: el modelo aprende de facturas anteriores y se evalúa con las posteriores, como se utilizará en la práctica. Con el valor predeterminado, `shuffle=True`, la división sería aleatoria y el modelo se probaría en parte con facturas anteriores a algunas de las que aprendió.

El escalado está dentro de la cadena de procesamiento. `StandardScaler` resta la media de cada característica y la escala a una varianza unitaria. Como `make_pipeline` lo encadena con el clasificador, la media y la varianza proceden únicamente de las filas de entrenamiento; ninguna información de la cuarta parte más reciente se filtra al modelo.

La puntuación es la exactitud equilibrada. Promedia, entre las dos clases, la proporción que el modelo acertó en cada una: las facturas tardías que detectó y las puntuales que reconoció. Si una de cada diez facturas se paga tarde, la exactitud ordinaria recompensa a un modelo que responda siempre «a tiempo». La exactitud equilibrada no.

El modelo debe superar una referencia básica. `DummyClassifier` con `strategy="most_frequent"` ignora las características y responde siempre con la opción más frecuente. Con dos clases obtiene una exactitud equilibrada de 0,5: acierta todas las facturas puntuales y falla todas las tardías. Si un modelo no lo supera claramente, no ha aprendido nada útil, por buena que parezca su exactitud ordinaria.

Nada de esto salva a un modelo de unas filas incorrectas. Si un cliente existe como tres registros, el recuento de sus facturas tardías se reparte entre tres y el modelo aprende de un historial que nunca ocurrió. Nuestra página de Preparación de datos para IA trata esos cimientos: un registro único por cliente, cada persona con acceso solo a lo que le corresponde y toda respuesta rastreable hasta los datos que la sustentan. En un modelo, esto último permite recorrer el camino inverso desde una predicción extraña hasta las filas que la produjeron.

Dónde surgen los problemas

Se introduce información del futuro en el entrenamiento. La guía de errores comunes de scikit-learn lo llama fuga de datos: construir el modelo con información que no existirá cuando haga una predicción. En las facturas, la tentadora columna adicional es el número de recordatorios enviados, que predice estupendamente el retraso porque los recordatorios solo se envían después del vencimiento. La misma fuga ocurre de forma menos visible si se ajusta un escalador o un imputador con todas las filas antes de dividirlas; la regla de la guía es no llamar nunca a `fit` con los datos de prueba. Anote cuándo se conoce el valor de cada columna, divida primero y mantenga todos los pasos de preprocesamiento dentro de la cadena, para que se ajusten únicamente con filas de entrenamiento.

Se evalúa el modelo con lo que ya ha visto. Un modelo que solo repitiese los valores objetivo de su entrenamiento obtendría una puntuación perfecta con esas filas y no serviría para nada nuevo; la guía de validación cruzada lo llama sobreajuste. Una forma más sutil consiste en probar muchos ajustes y quedarse con el que mejor puntúe con las filas reservadas: el conocimiento de esas filas se filtra a la elección. Compare los ajustes mediante validación cruzada con la parte de entrenamiento, dividida en varios grupos que se turnan como conjunto de prueba, y consulte las filas reservadas una sola vez, al final.

Una exactitud alta que también alcanzaría una constante. Si una clase es infrecuente, la clase común domina la exactitud; la guía advierte de que, con clases raras, como en la detección de fraude, un grupo puede quedarse sin ningún ejemplo de la clase minoritaria. Informe de la exactitud equilibrada u otra medida por clase, compare con un `DummyClassifier` y use grupos estratificados, que mantienen proporciones de clases aproximadamente iguales en cada grupo. `cross_val_score` los usa de forma predeterminada para un clasificador cuando se le pasa un número de grupos.

Se divide aleatoriamente información que cambia con el tiempo. La guía de validación cruzada señala que la suposición de muestras independientes e idénticamente distribuidas rara vez se cumple en la práctica y que los datos ordenados temporalmente deben probarse con observaciones posteriores a las del entrenamiento. Una división aleatoria permite al modelo asomarse al futuro y anuncia una puntuación que no alcanzará. Divida por fecha, como en el ejemplo; para la validación cruzada use un divisor temporal como `TimeSeriesSplit`, cuyos grupos, según la guía, deberían abarcar intervalos de igual duración.

El preprocesamiento cambia entre entrenamiento y producción. Si el modelo aprendió con características escaladas y el sistema real envía valores sin transformar, las predicciones serán incorrectas sin que se produzca un error. La guía exige aplicar toda transformación del entrenamiento tanto a los datos de prueba como a los de producción. Distribuya la cadena ya ajustada como un único objeto, en lugar de un modelo acompañado de una nota sobre cómo preparar los datos.

El mundo cambia después del lanzamiento. Un modelo entrenado con las facturas del año pasado supone que los clientes siguen comportándose igual. Sculley y sus coautores proponen vigilar el sesgo de predicción: en un sistema que funciona según lo previsto, la proporción de cada etiqueta predicha normalmente debería coincidir con la realmente observada; una brecha suele ser la primera señal de que el mundo ha cambiado. Mantenga esa comparación, reentrene con datos recientes y conserve una copia congelada y versionada de toda señal de entrada que pertenezca a otro equipo, para que un cambio bienintencionado anterior en el flujo no modifique silenciosamente su modelo.

Lecturas adicionales

- Guía del usuario — la referencia de scikit-learn para aprendizaje supervisado y no supervisado y para selección y evaluación de modelos, capítulo a capítulo.
- Primeros pasos — la interfaz de los estimadores en ejemplos breves: ajuste y predicción, transformadores, cadenas de procesamiento y validación cruzada.
- Errores comunes y prácticas recomendadas — preprocesamiento incoherente, fuga de datos y aleatoriedad, con el código incorrecto y el correcto enfrentados.
- Validación cruzada: evaluar el rendimiento de los estimadores — por qué engaña probar con datos de entrenamiento y qué división conviene para datos agrupados, desequilibrados u ordenados temporalmente.

- Deuda técnica oculta en sistemas de aprendizaje automático — Sculley y sus colegas de Google, NIPS 2015, sobre el coste de ejecutar un modelo en producción más allá del propio modelo.

Glosario

Elija según sus restricciones reales

Describa el flujo de trabajo, las interfaces existentes y los usuarios que necesitan el cambio. Podemos definir una entrega comprobable y un traspaso que permita a su equipo mantener lo entregado.

Vea cuánto cuesta Póngase en contacto →

Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es el aprendizaje automático?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

Explore nuestros servicios

welcome@engineer.company +45 50 10 61 84 Copenhague, Dinamarca

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://engineer.company/es/glossary/machine-learning/