Guía de estudio N°1— Introducción al Machine learning

¡Hola! Te damos la bienvenida a esta nueva guía de estudio.

¿En qué consiste esta guía?

Queremos invitarte a alcanzar un alto nivel de comprensión sobre temas como la inteligencia artificial y el machine learning, que son fundamentales en el campo del Data Science. A lo largo de esta guía, también aprenderás a seleccionar la tarea adecuada según los diferentes problemas existentes en este campo tan fascinante.

Nuestro objetivo es brindarte los conocimientos necesarios para que puedas comprender los conceptos clave y las aplicaciones prácticas del aprendizaje de máquinas. Te familiarizaras con herramientas y librerías ampliamente utilizadas en esta área, lo que te permitirá poner en práctica los conocimientos adquiridos.

A través de ejemplos, podrás explorar tanto los aspectos conceptuales como prácticos del aprendizaje de máquinas. Con ello, obtendrás una visión integral y podrás desarrollar habilidades para aplicar estas técnicas en diferentes escenarios.

¡Prepárate para sumergirte en el mundo del aprendizaje de máquinas! ¡Vamos a explorar juntos los fundamentos y las aplicaciones de esta emocionante disciplina!

¡Vamos con todo!

Tabla de contenidos

¡Comencemos!

¿Qué es el Machine Learning?

El Machine Learning, o aprendizaje de máquinas, es una rama de la inteligencia artificial que se centra en el desarrollo de algoritmos y modelos que permiten a las computadoras aprender y tomar decisiones automáticamente, sin ser programadas explícitamente para ello. Se basa en la idea de que las máquinas pueden aprender de los datos y mejorar su rendimiento a medida que se les proporciona más información.

El objetivo principal del Machine Learning es permitir que las computadoras puedan identificar patrones, realizar predicciones y tomar decisiones basadas en datos, de manera similar a cómo lo haría un ser humano. Esto se logra a través de algoritmos y modelos que analizan los datos, encuentran correlaciones y regularidades, y utilizan esa información para hacer predicciones o tomar decisiones en nuevos datos.

La historia del Machine Learning se remonta a mediados del siglo XX, cuando los científicos comenzaron a explorar la idea de crear máquinas que pudieran aprender de la experiencia. Uno de los hitos más importantes en el desarrollo del Machine Learning fue el desarrollo del perceptrón por Frank Rosenblatt en 1957, que sentó las bases para los primeros algoritmos de aprendizaje automático.

En las décadas siguientes se produjeron avances significativos en el campo del Machine Learning. Uno de los hitos más destacados fue el desarrollo de los algoritmos de aprendizaje supervisado, como el algoritmo de regresión logística y las máquinas de vectores de soporte (SVM), que permitieron la clasificación y la predicción precisas.

En los últimos años, el Machine Learning ha experimentado un crecimiento exponencial debido a varios factores, como el aumento masivo en la disponibilidad de datos, los avances en la capacidad de procesamiento y almacenamiento, y los algoritmos más sofisticados como las redes neuronales profundas. Estos avances han permitido la aplicación del Machine Learning en una amplia gama de industrias y campos, desde la medicina y la investigación científica hasta la industria del entretenimiento y el comercio electrónico.

En resumen, el Machine Learning es una rama de la inteligencia artificial que permite a las computadoras aprender y tomar decisiones automáticamente. A lo largo de su historia, ha experimentado importantes hitos y avances, y actualmente desempeña un papel crucial en diversos campos y sectores.

¿Qué es un modelo?

En el contexto del Machine Learning, un modelo es una representación matemática o algorítmica de un problema o fenómeno que se desea analizar o predecir. El modelo se crea a partir de los datos de entrenamiento y tiene la capacidad de generalizar a nuevos datos para hacer predicciones o tomar decisiones.

En términos sencillos, puedes imaginar un modelo como una "caja negra" que recibe ciertos datos de entrada y produce una respuesta o resultado esperado. El objetivo es que el modelo aprenda a través de los datos o la experiencia para generar respuestas cada vez más precisas y acertadas.

El proceso de construcción de un modelo generalmente involucra los siguientes pasos:

1. **Recopilación de datos:** Se reúnen datos relevantes y representativos del problema que se desea abordar. Estos datos consisten en pares de características o variables de entrada y su correspondiente resultado o etiqueta.
2. **Entrenamiento del modelo**: Utilizando los datos de entrenamiento, se ajustan los parámetros del modelo para que pueda aprender los patrones y relaciones presentes en los datos. Durante el entrenamiento, el modelo realiza cálculos y ajustes internos para mejorar su capacidad de hacer predicciones precisas.
3. **Evaluación del modelo:** Una vez entrenado, el modelo se evalúa utilizando datos de prueba que no se utilizaron en el entrenamiento. Esto ayuda a medir la capacidad del modelo para generalizar y hacer predicciones precisas en nuevos datos.
4. **Uso del modelo:** Una vez que el modelo ha sido entrenado y evaluado, se puede utilizar para hacer predicciones o tomar decisiones en nuevos datos. Se le proporcionan las características de entrada y el modelo genera una respuesta o predicción basada en los patrones aprendidos durante el entrenamiento.

Un ejemplo sencillo de un modelo es un modelo de predicción del clima. Supongamos que tenemos datos históricos de temperatura, humedad y presión atmosférica, junto con los correspondientes pronósticos del clima. Utilizando estos datos de entrenamiento, se puede construir un modelo que aprenda a reconocer los patrones climáticos y hacer predicciones sobre el clima futuro. Una vez que el modelo está entrenado, se le pueden proporcionar los datos actuales de temperatura, humedad y presión atmosférica, y el modelo generará una predicción sobre el clima esperado, como "soleado", "lluvioso" o "nublado".

Tipos de Aprendizajes

En Machine Learning, existen diferentes tipos de aprendizaje que se utilizan para abordar distintos tipos de problemas. A continuación, se proporciona una descripción de los principales tipos de aprendizaje: supervisado, no supervisado y por refuerzo.

Aprendizaje Supervisado

En el aprendizaje supervisado se utilizan conjuntos de datos etiquetados, es decir, se tienen ejemplos de entrada y su correspondiente salida deseada. El objetivo es construir un modelo que pueda aprender a mapear las características de entrada a las salidas esperadas. Esto implica encontrar una función que pueda predecir correctamente las salidas para nuevas entradas.

El aprendizaje supervisado se divide en dos categorías principales:

a. **Clasificación**: Se utiliza cuando la salida esperada o variable objetivo es una etiqueta o categoría discreta. Por ejemplo, clasificar correos electrónicos como spam o no spam.
b. **Regresión**: Se utiliza cuando la salida esperada o variable objetivo es un valor continuo. Por ejemplo, predecir el precio de una vivienda basado en sus características.

Consideraciones

- Se requiere un conjunto de datos etiquetados para entrenar el modelo, es decir, se necesita la variable objetivo o respuesta de aquel comportamiento que se quiere modelar.
- La calidad y representatividad de los datos de entrenamiento son fundamentales para el rendimiento del modelo.
- El principal objetivo de los modelos de aprendizaje supervisado es la de generalizar el problema para datos no vistos previamente por el modelo.

Aprendizaje No Supervisado

En el aprendizaje no supervisado, no se dispone de datos etiquetados. El objetivo principal es descubrir patrones o estructuras ocultas en los datos de entrada. En lugar de predecir una salida específica, se busca agrupar los datos en función de similitudes o características comunes.

El aprendizaje no supervisado se divide en dos categorías principales:

a. **Clustering (Segmentación)**: Se utilizan algoritmos para agrupar los datos en diferentes grupos basados en similitudes. Por ejemplo, agrupar clientes en segmentos de mercado según su comportamiento de compra.
b. **Reducción de dimensionalidad**: Se utilizan técnicas para reducir la cantidad de características o variables en un conjunto de datos. Esto ayuda a visualizar los datos o simplificar problemas complejos. Por ejemplo, reducir la dimensión de un conjunto de imágenes para su posterior análisis.

Consideraciones

- No se dispone de salidas etiquetadas, lo que dificulta la evaluación del rendimiento del modelo.
- Se utilizan medidas de similitud o distancia para agrupar o reducir los datos.
- El aprendizaje no supervisado es útil para descubrir patrones ocultos, detectar anomalías o simplificar la representación de los datos.

- Para los algoritmos de clusterización es relevante la validación de negocio respecto a los segmentos encontrados.

Aprendizaje por Reforzamiento

El aprendizaje reforzado o por reforzamiento es un enfoque del Machine Learning en el cual un agente aprende a tomar decisiones en un entorno interactivo para maximizar una recompensa acumulativa. El agente toma acciones en un entorno y recibe retroalimentación en forma de recompensas o penalizaciones, lo que le permite aprender a tomar las acciones correctas en diferentes situaciones.

El aprendizaje reforzado se basa en el concepto de prueba y error, donde el agente explora el entorno, toma decisiones y aprende de las consecuencias de sus acciones. Al utilizar técnicas como la exploración y la explotación, el agente busca encontrar la estrategia óptima que maximice la recompensa a largo plazo.

Un ejemplo común de aprendizaje reforzado es entrenar a un agente para jugar juegos de mesa, como el ajedrez o el Go. El agente aprende a través de interacciones con el entorno y busca aprender estrategias que le permitan ganar el juego.

En el aprendizaje por refuerzo, hay varias partes importantes que debes comprender para desarrollar un sistema de aprendizaje automático efectivo. Estas son algunas de las partes clave:

1. **Agente:** Es la entidad que realiza las acciones en el entorno. El agente puede ser un robot físico, un programa de software, etc. Su objetivo es aprender a tomar decisiones óptimas para maximizar la recompensa acumulativa.

2. **Entorno:** Es el contexto en el que el agente interactúa y toma acciones. Puede ser un entorno físico o virtual, como un juego de mesa, una simulación de vuelo, etc. El agente recibe información del entorno y toma decisiones basadas en esa información.

3. **Acciones:** Son las opciones disponibles para el agente en cada paso de tiempo. Pueden ser acciones discretas (por ejemplo, moverse en una dirección específica) o acciones continuas (por ejemplo, ajustar un valor de control).

4. **Estados:** Representan la información relevante del entorno en un momento dado. Los estados pueden ser observables directamente o pueden ser inferidos por el agente a partir de las observaciones.

5. **Recompensas:** Son señales numéricas de retroalimentación que el agente recibe del entorno después de realizar una acción. Indican la calidad de la acción tomada y se utilizan para guiar el proceso de aprendizaje. El objetivo del agente es maximizar la recompensa acumulativa a largo plazo.

6. **Política:** Es la estrategia o conjunto de reglas que el agente sigue para seleccionar acciones en función del estado actual. La política puede ser determinista o estocástica, y puede ser representada por una función o una tabla de valores.

7. **Función de valor:** Es una función que asigna un valor a cada estado o par estado-acción, indicando la utilidad o el valor esperado de estar en ese estado o tomar esa acción en función de la recompensa esperada a largo plazo.

8. **Exploración y explotación:** En el aprendizaje por refuerzo, el agente debe encontrar un equilibrio entre explorar el entorno para descubrir nuevas acciones y explotar el conocimiento actual para maximizar la recompensa. La exploración permite descubrir acciones más óptimas, mientras que la explotación se basa en acciones ya conocidas.

9. **Algoritmos de aprendizaje por refuerzo:** Existen diferentes algoritmos para el aprendizaje por refuerzo, como Q-Learning, SARSA, Deep Q-Networks (DQN) y Proximal Policy Optimization (PPO). Estos algoritmos utilizan diferentes técnicas para actualizar la política o la función de valor del agente y mejorar su rendimiento a lo largo del tiempo.

Comprender estas partes clave te ayudará a desarrollar un entendimiento más profundo del aprendizaje por refuerzo y te permitirá diseñar y entrenar sistemas de aprendizaje automático que puedan tomar decisiones óptimas en entornos interactivos.

¡Es tu turno de pensar algunas aplicaciones para cada tipo de aprendizaje en tu área de interés!

Machine Learning con Python

Python ha ganado popularidad en el campo del aprendizaje automático y la ciencia de datos debido a varias ventajas y beneficios que ofrece. Su diseño elegante y su sintaxis legible hacen que sea una opción ideal para el desarrollo de proyectos de machine learning. A continuación, explicaremos las ventajas y consejos para aprovechar al máximo Python en el desarrollo de modelos de machine learning.

Una de las principales ventajas de utilizar Python para machine learning es la amplia comunidad de programadores en este lenguaje, que permiten tener respuestas rápidas y una gran disponibilidad de bibliotecas especializadas como Scikit-learn, TensorFlow, Keras y PyTorch que ofrecen una amplia gama de algoritmos, herramientas y funciones para el desarrollo eficiente de modelos de machine learning.

Python se destaca por su sintaxis simple y legible, lo que facilita el proceso de desarrollo y mantenimiento del código. Su sintaxis se asemeja al lenguaje humano, lo que permite a los desarrolladores expresar ideas de manera concisa y comprensible. Esto resulta especialmente útil al trabajar con algoritmos complejos de machine learning, ya que se puede comprender fácilmente el flujo y la lógica del código.

La facilidad de aprendizaje es otra ventaja significativa de Python. Incluso aquellos sin experiencia previa en programación pueden aprender Python de manera relativamente sencilla. La comunidad de Python es muy activa y cuenta con una amplia gama de recursos educativos, tutoriales y ejemplos de código que facilitan el proceso de aprendizaje del lenguaje y su aplicación en el campo del machine learning.

Python se integra sin problemas con otras bibliotecas y herramientas ampliamente utilizadas en el campo del aprendizaje automático. Por ejemplo, se puede combinar Python con bibliotecas como NumPy para realizar operaciones numéricas eficientes, Pandas para manipulación de datos, Matplotlib para visualización de resultados y Jupyter Notebook para el desarrollo interactivo. Esta interoperabilidad permite un flujo de trabajo fluido y una mayor productividad durante el desarrollo de modelos de machine learning.

Para aprovechar al máximo Python en el desarrollo de modelos de machine learning, se recomienda explorar y utilizar las bibliotecas existentes. Además, es importante practicar y experimentar con conjuntos de datos reales para obtener experiencia práctica. Aprovechar la comunidad de Python y los recursos educativos disponibles también es fundamental para aprender y mejorar en el campo del machine learning con Python.

En resumen, Python ofrece varias ventajas clave para el desarrollo de modelos de machine learning, incluyendo la disponibilidad de bibliotecas especializadas, su sintaxis simple y legible, la facilidad de aprendizaje, la interfaz con otras bibliotecas y herramientas, y la activa comunidad de Python. Al aprovechar estas ventajas y seguir los consejos mencionados, se puede potenciar el uso de Python en el campo del machine learning y lograr resultados efectivos y eficientes.

Librerías en python

Algunas de las librerías más utilizadas en el campo son:

1. **Scikit-learn**: es una biblioteca de aprendizaje automático de propósito general que ofrece una amplia gama de algoritmos y herramientas para el preprocesamiento de datos, la selección de características, la construcción de modelos y la evaluación de rendimiento. Es conocida por su facilidad de uso y su enfoque en la implementación de algoritmos eficientes y escalables. Scikit-learn es ampliamente utilizado en la comunidad de aprendizaje automático y ofrece una documentación exhaustiva y muchos ejemplos prácticos.

2. **TensorFlow**: es una biblioteca de código abierto desarrollada por Google que se centra en el aprendizaje automático y el aprendizaje profundo. Es especialmente adecuada para construir y entrenar redes neuronales profundas en entornos distribuidos y con aceleración por GPU. TensorFlow proporciona una interfaz flexible y de alto nivel para construir modelos de aprendizaje automático y también permite la implementación de algoritmos personalizados. Es ampliamente utilizado en la investigación y la industria del aprendizaje automático.

3. **Keras**: Es una biblioteca de alto nivel que se ejecuta sobre TensorFlow y proporciona una API simple y elegante para construir y entrenar redes neuronales. Es especialmente popular entre los principiantes y permite la construcción rápida de modelos de aprendizaje profundo utilizando bloques de construcción predefinidos. Keras es conocido por su facilidad de uso, su enfoque en la experimentación rápida y su capacidad de integración con otras bibliotecas de Python.

4. **PyTorch:** Otra biblioteca de aprendizaje automático y aprendizaje profundo de código abierto que se ha vuelto muy popular en los últimos años. Se destaca por su capacidad de calcular gradientes de forma automática, lo que facilita la implementación y el entrenamiento de modelos complejos. PyTorch proporciona una API dinámica y flexible que permite un flujo de trabajo más intuitivo y una mayor flexibilidad en la construcción de modelos. También se ha convertido en una elección popular para la investigación en aprendizaje profundo.

5. **XGBoost:** Esta es una biblioteca optimizada de gradient boosting que se utiliza ampliamente en competencias de ciencia de datos y análisis predictivo. Ofrece una implementación eficiente y escalable de algoritmos de boosting y puede manejar tanto problemas de regresión como de clasificación. XGBoost es conocido por su rendimiento de alta calidad y su capacidad para manejar características numéricas y categóricas.

Estas son solo algunas de las bibliotecas más destacadas en Python para el aprendizaje automático, pero hay muchas otras disponibles, como LightGBM, CatBoost, Theano, entre otras. Cada una de estas bibliotecas tiene sus propias fortalezas y características distintivas, por lo que es recomendable explorarlas y elegir la más adecuada para tu proyecto o caso de uso específico.

¡Sigue explorando y aprendiendo!

Scikit - Learn

La principal librería que vamos a utilizar durante el módulo es scikit learn debido a la amplia gama de algoritmos que estarán implementados de manera eficiente, tanto de aprendizaje supervisado, como de aprendizaje no supervisado, la facilidad de aprendizaje y uso de esta herramienta debido a su enfoque en usabilidad y su documentación, y finalmente la compatibilidad con otras librerías como pandas, numpy y matplotlib.

En general, Scikit-learn es una biblioteca popular y poderosa para el aprendizaje automático en Python, que combina una amplia variedad de algoritmos con una interfaz fácil de usar y eficiencia computacional. Es una opción sólida tanto para principiantes como para expertos en el campo.

Dentro de sus principales usos podemos encontrar lo siguiente:

1. **Dataset de prueba:** podemos cargar por medio de su api una gran variedad de set de datos para practicar y probar algoritmos supervisados y no supervisados.
2. **Preprocesamiento:** cuenta con una amplia gama de métodos para preprocesar los datos, escalarlos, generar set de entrenamiento y testeo, como pipelines de entrenamiento y mucho más.
3. **Algoritmos:** tiene implementaciones eficientes y flexibles de una gran cantidad de algoritmos de aprendizaje supervisado (clasificación y regresión), como de aprendizajes no supervisados (clustering y reducción de dimensionalidad)
4. **Métricas de evaluación:** cuenta con varios métodos para evaluar los diferentes algoritmos de forma sencilla.

¡Tú puedes descubrir más usos! ¡Adelante!

Regresión Lineal con Machine Learning

Teoría de la Regresión Lineal

La regresión lineal es un algoritmo ampliamente utilizado en machine learning para predecir una variable continua basada en una o más variables predictoras. La idea fundamental detrás de la regresión lineal es encontrar la mejor línea recta que se ajuste a los datos y pueda utilizarse para hacer predicciones.

En el contexto del machine learning, la regresión lineal se trata de encontrar los coeficientes óptimos que minimizan la diferencia entre los valores reales y los valores predichos por el

modelo lineal. Esto se logra utilizando técnicas de optimización, como el método de los mínimos cuadrados.

$$Y = \beta_0 + \sum_{i=1}^{n} \beta_i * X_i$$

Fórmula general de la regresión lineal

Donde Y es la variable dependiente que queremos predecir, β_0 es el término de intersección o sesgo, β_i son los coeficientes que multiplican a las variables predictoras X_i .

El objetivo es encontrar los valores óptimos de β_i que minimicen el error entre los valores predichos y los valores reales. Una vez que se han calculado los coeficientes óptimos, el modelo de regresión lineal se puede utilizar para hacer predicciones sobre nuevos datos. Simplemente se ingresan los valores de las variables predictoras en la ecuación del modelo y se obtiene la predicción correspondiente.

Es importante tener en cuenta que la regresión lineal asume una relación lineal entre las variables predictoras y la variable dependiente. Si esta suposición no se cumple, el modelo puede no ajustarse bien a los datos y las predicciones pueden ser inexactas. En esos casos, se pueden explorar otros modelos de regresión más complejos.

Entrenamiento de la regresión lineal

El entrenamiento de un modelo de regresión lineal se basa en encontrar los valores óptimos de los coeficientes que minimizan el error entre las predicciones del modelo y los valores reales en el conjunto de entrenamiento. Esto se puede lograr utilizando el método de los mínimos cuadrados o el método de descenso de gradiente, que son dos enfoques comunes para optimizar los coeficientes en la regresión lineal.

$$\beta = argmin(\sum_{i=1}^{n}(y - y_{pred})^2) = argmin(\sum_{i=1}^{n}(y - \beta_0 + (\sum_{i=1}^{n} \beta_i * X_i))^2)$$

Problema de optimización

1. **Método de los mínimos cuadrados:** En este método, se calculan los coeficientes óptimos minimizando la suma de los cuadrados de las diferencias entre las predicciones del modelo y los valores reales en el conjunto de entrenamiento. Esto se puede hacer mediante técnicas matemáticas, como la inversión de matrices. Una vez calculados los coeficientes, el modelo está entrenado y se puede utilizar para hacer predicciones.

2. **Método de descenso de gradiente:** En este método, se inicializan los coeficientes con valores aleatorios y se actualizan iterativamente utilizando el gradiente

descendente. El gradiente descendente es un algoritmo de optimización que busca minimizar la función de costo, que mide la diferencia entre las predicciones del modelo y los valores reales. En cada iteración, se calcula el gradiente de la función de costo con respecto a los coeficientes y se ajustan los coeficientes en la dirección opuesta al gradiente para reducir la función de costo. Este proceso se repite hasta que se alcanza un criterio de convergencia.

Durante el entrenamiento, es común dividir los datos en un conjunto de entrenamiento y un conjunto de prueba o validación. El conjunto de entrenamiento se utiliza para calcular los coeficientes óptimos, mientras que el conjunto de prueba se utiliza para evaluar el rendimiento del modelo en datos no vistos. Esto permite verificar si el modelo generaliza bien y es capaz de hacer predicciones precisas en nuevos datos.

Sesgo y varianza

El trueque entre sesgo y varianza es un concepto fundamental en el aprendizaje automático y se refiere a la relación inversa entre ambos errores en un modelo. Introduciremos brevemente este tema, que desarrollaremos en sesiones posteriores.

El sesgo se refiere al error sistemático que se produce cuando el modelo asume una simplificación excesiva o incorrecta sobre los datos. Un modelo con alto sesgo tiende a tener un rendimiento pobre en el conjunto de entrenamiento y en el conjunto de prueba, ya que no puede capturar la complejidad de los datos.

Por otro lado, la varianza se refiere a la sensibilidad del modelo a las variaciones en los datos de entrenamiento. Un modelo con alta varianza se ajusta demasiado a los datos de entrenamiento y tiene dificultades para generalizar a nuevos datos. Esto puede llevar a un buen rendimiento en el conjunto de entrenamiento, pero un mal rendimiento en el conjunto de prueba.

El trueque entre sesgo y varianza se presenta de la siguiente manera: al tratar de reducir el sesgo de un modelo, se corre el riesgo de aumentar su varianza, y viceversa. En otras palabras, si se aumenta la complejidad del modelo para reducir el sesgo y mejorar el ajuste a los datos de entrenamiento, se puede aumentar la varianza y provocar un sobreajuste. Por otro lado, si se reduce la complejidad del modelo para disminuir la varianza y mejorar la generalización, se puede aumentar el sesgo y subajustar los datos.

En resumen, el objetivo es encontrar un equilibrio entre sesgo y varianza que minimice el error total del modelo. Esto se logra mediante la selección de un modelo adecuado, el ajuste de los hiperparámetros y la utilización de técnicas como la validación cruzada para evaluar el rendimiento del modelo en conjuntos de datos diferentes.

Es importante comprender el trueque entre sesgo y varianza para tomar decisiones informadas al desarrollar y ajustar modelos de aprendizaje automático, y así lograr un

equilibrio óptimo entre el sesgo y la varianza para obtener un rendimiento generalizado y preciso en la predicción de nuevos datos.

Train - Test Split

La división del conjunto de datos en conjuntos de entrenamiento y prueba es una práctica común en el aprendizaje automático y es fundamental para evaluar y validar el rendimiento del modelo.

La razón principal para dividir el conjunto de datos en conjuntos de entrenamiento y prueba es evaluar cómo se generaliza el modelo a nuevos datos. El conjunto de entrenamiento se utiliza para ajustar los parámetros del modelo, es decir, para enseñar al modelo a aprender de los datos y realizar predicciones. Por otro lado, el conjunto de prueba se utiliza para evaluar el rendimiento del modelo en datos que no ha visto durante el entrenamiento.

La idea es simular un escenario en el que el modelo se encuentra con datos completamente nuevos y verificar cómo se comporta. Al evaluar el modelo en un conjunto de datos independiente, se puede obtener una estimación realista del rendimiento y la capacidad de generalización del modelo.

La división en conjuntos de entrenamiento y prueba también ayuda a detectar y evitar el sobreajuste (overfitting) del modelo. El sobreajuste ocurre cuando el modelo se ajusta demasiado a los datos de entrenamiento y no generaliza bien a nuevos datos. Al tener un conjunto de prueba independiente, se puede evaluar si el modelo está sobreajustado o no.

Además de la división en conjuntos de entrenamiento y prueba, es común utilizar técnicas como la validación cruzada para obtener estimaciones más robustas del rendimiento del modelo y ajustar los hiper parámetros del modelo de manera más precisa.

Esta es una de las principales diferencias con el enfoque estadístico, donde el objetivo era interpretar las relaciones entre las variables de entrada y la variables de salida, pero ahora queremos asegurarnos de tener un algoritmo lo suficientemente bueno para generalizar el problema a nuevos datos.

En resumen, la división en conjuntos de entrenamiento y prueba es esencial para evaluar el rendimiento y la generalización del modelo, detectar el sobreajuste y tomar decisiones informadas sobre la elección del modelo y sus parámetros.

Multicolinealidad y factor de inflación de varianza

La multicolinealidad y el factor de inflación de la varianza (VIF) están relacionados y se utilizan para detectar y cuantificar la presencia de alta correlación entre variables predictoras en un modelo de regresión lineal múltiple.

La multicolinealidad es una situación en la que existe una fuerte correlación entre dos o más variables predictoras en un modelo. Esto puede ser problemático porque puede hacer que sea difícil identificar la relación individual de cada variable con la variable de respuesta y puede afectar la interpretación de los coeficientes de regresión. Además, la multicolinealidad puede provocar inestabilidad en los coeficientes estimados y hacer que sean difíciles de interpretar.

El factor de inflación de la varianza (VIF) es una medida que cuantifica la magnitud de la multicolinealidad entre las variables predictoras. Se calcula para cada variable predictora y se basa en la idea de que si una variable está altamente correlacionada con otras variables, su varianza se infla. Un VIF alto indica una alta multicolinealidad, lo que significa que la variable está altamente correlacionada con otras variables en el modelo.

El VIF se calcula para cada variable predictora como el cociente entre la varianza de los coeficientes estimados en el modelo completo y la varianza de los coeficientes estimados en un modelo que solo incluye esa variable predictora. Un VIF de 1 indica que no hay multicolinealidad, mientras que un VIF mayor que 1 indica la presencia de multicolinealidad. Generalmente, se considera que un VIF superior a 5 o 10 indica una alta multicolinealidad.

La interpretación del VIF es que cuanto mayor sea el valor, mayor será la inflación de la varianza debido a la multicolinealidad. Esto implica que los coeficientes estimados pueden tener una mayor varianza y ser menos confiables. En términos prácticos, si se encuentra una alta multicolinealidad (VIF alto), es recomendable tomar medidas para abordar el problema, como eliminar una de las variables altamente correlacionadas o transformar las variables de alguna manera.

Estos aspectos tendrán vital importancia en módulos posteriores.

Actividad guiada: Técnicas de Machine Learning

Para continuar el enfoque de Machine Learning vamos a aplicar los siguientes pasos:

1. Cargar un dataset desde la librería Scikit-Learn
2. Dividir el dataset en un dataset de entrenamiento y uno de testeo
3. Entrenar el modelo en el dataset de entrenamiento
4. Predecir en el dataset de testeo
5. Evaluar el modelo con las métricas indicadas

Para instalar scikit-learn debes ejecutar la siguiente línea de código:

python -m pip install scikit-learn

Para comenzar, cargaremos las bibliotecas necesarias y los datos.

```
import matplotlib.pyplot as plt
import numpy as np

#Librerías de machine learning
from sklearn.datasets import fetch_california_housing
from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

#si queremos leer la data desde un archivo de datos, necesitaremos
cargar pandas y utilizar pd.read
#data=pd.read_csv(archivo)
data = fetch_california_housing()
X, y = data.data, data.target

#también puede hacerse de esta manera
#X, y = load_wine(return_X_y=True)
```

Dividimos la muestra, en entrenamiento y testeo utilizando **train test split.** En este método, test_size=0.3 nos indica que el 30% de los datos se utilizará para prueba (naturalmente se puede modificar) y **random_state=123** corresponde a una semilla aleatoria, para que cuando repitamos la separación obtengamos el mismo resultado (y diferente si cambiamos este valor)

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 0.3,
random_state=123)
```

Entrenaremos ahora la regresión lineal

```
# Instanciar el objeto de la regresión lineal
regr = linear_model.LinearRegression()
# Entrenar el objeto con el método fit
regr.fit(X_train, y_train)
```

Utilizaremos ahora el modelo para predecir.

```
y_pred = regr.predict(X_test)
print("Coeficientes: \n", regr.coef_)
```

Podemos evaluar ahora el modelo, calculando el error cuadrado medio y el coeficiente de determinación

```
Cprint('\n Error cuadrado medio: %.2f' % mean_squared_error(y_test,
y_pred))

r2 = r2_score(y_test, y_pred)
print("Coeficiente de determinación (R^2):", r2)
```

Finalmente, podemos verificar alguna de nuestras predicciones (en nuestro caso, para la primera columna) creando un scatter-plot que considere los datos reales y las predicciones

```
plt.scatter(y_test, y_pred)
plt.xlabel("Valores reales")
plt.ylabel("Predicciones")
plt.title("Predicciones vs Valores reales")
plt.show()
```

¡Lo lograste! / ¡Felicitaciones!

Reflexiona:

- ¿Qué es Machine Learning? ¿Cómo se relaciona con el concepto de Inteligencia Artificial?
- ¿Qué tipos de tareas pueden realizar los algoritmos de Machine Learning?
- ¿Qué librerías se usan para trabajar con Machine Learning en Python?
- ¿En qué se diferencia la regresión lineal vista en el módulo pasado, con la vista desde la perspectiva de Machine Learning?

Referencias bibliográficas

1. Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.
2. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.
3. Scikit-learn Documentation: https://scikit-learn.org/stable/documentation.html
4. Python Data Science Handbook: https://jakevdp.github.io/PythonDataScienceHandbook

¡Continúa aprendiendo y practicando!