Tema 2: Programando IA con Python

Capítulo 1: Preprocesado

El preprocesado de datos es una etapa fundamental en cualquier proyecto de Machine Learning (ML) que busca preparar los datos brutos para que puedan ser utilizados por los algoritmos de aprendizaje. Este proceso impacta directamente en la calidad, eficiencia y precisión del modelo de ML.

A continuación, se presenta un resumen de los puntos clave relacionados con el preprocesado, extraídos de las fuentes y la conversación previa:

- **Importancia de los Datos en ML:** Las fuentes destacan la importancia de los datos en el aprendizaje automático, considerándolos tan cruciales como el código. Cualquier cambio en los datos puede afectar significativamente el rendimiento del modelo, por lo que el preprocesado adecuado se vuelve esencial.
- **Pipeline de Datos en MLOps:** Dentro de los componentes de MLOps, se menciona el **"pipeline de datos"**, que se encarga de la **"recopilación, limpieza, transformación y almacenamiento de datos para el entrenamiento y la inferencia del modelo"**. Esto implica que el preprocesado forma parte integral de este pipeline.
- **Big Data e IA:** Las fuentes resaltan la relación simbiótica entre el Big Data y la IA. El Big Data proporciona la cantidad masiva de datos necesaria para entrenar modelos de IA, pero estos datos suelen ser heterogéneos y desorganizados. El preprocesado es crucial para limpiar, transformar y estructurar estos datos antes de su uso en el entrenamiento de modelos.
- **Ejemplos de Código Python:** Si bien los ejemplos de código en Python se centran en los fundamentos del lenguaje, algunos ejemplos, como la manipulación de cadenas y el trabajo con estructuras de datos, pueden ser aplicables en el preprocesado de datos. Por ejemplo:
    - Las funciones para **eliminar espacios en blanco**, **convertir a mayúsculas o minúsculas**, **reemplazar subcadenas** o **segmentar cadenas** pueden ser útiles para limpiar y normalizar datos de texto.
    - El trabajo con **listas, tuplas y diccionarios** permite organizar y manipular datos para su posterior análisis.

A partir de estos puntos, se puede inferir que el preprocesado de datos en ML abarca una serie de tareas como:

- **Análisis Exploratorio:** Se revisan los datos y se realiza un análisis de los mismos para identificar errores, inconsistencias u otros problemas.
- **Limpieza de Datos:** Eliminar o corregir datos erróneos, incompletos o inconsistentes.

- **Transformación de Datos:** Convertir los datos a un formato adecuado para el algoritmo de aprendizaje. Esto puede incluir:
    - **Escalado de características:** Ajustar las variables a una misma escala para evitar que algunas dominen el entrenamiento del modelo.
    - **Codificación de variables categóricas:** Transformar variables no numéricas en representaciones numéricas.
- **Reducción de dimensionalidad:** Seleccionar las características más relevantes para el modelo y reducir la complejidad del conjunto de datos.
- **Manejo de valores faltantes:** Decidir cómo tratar los valores ausentes en el conjunto de datos.

Capítulo 2: Pandas

Pandas es una biblioteca de software escrita para el lenguaje de programación Python para la manipulación y análisis de datos. Las fuentes proporcionan ejemplos de cómo usar Pandas para leer y escribir archivos CSV y Excel, acceder a columnas, filtrar filas, realizar agrupaciones, obtener estadísticas descriptivas, eliminar duplicados y valores faltantes, fusionar DataFrames y otras operaciones comunes.

Desafortunadamente, las fuentes no proporcionan una descripción detallada de la librería Pandas. Para obtener más información sobre esta biblioteca, se recomienda consultar la documentación oficial de Pandas u otros recursos disponibles en línea.

Capítulo 3: Numpy

Las fuentes proporcionadas ofrecen una visión general de la programación de IA y ejemplos de código Python, pero solo mencionan brevemente a NumPy. Se indica que las fuentes muestran ejemplos de cómo:

- **Importar NumPy**
- **Crear arrays**
- **Acceder a sus atributos**
- **Realizar operaciones matemáticas con arrays**

Información adicional:

Para obtener un resumen completo de NumPy, se necesitarían fuentes adicionales que profundicen en sus funcionalidades. NumPy es una biblioteca fundamental para la computación científica en Python, y ofrece una amplia gama de funciones para trabajar con arrays multidimensionales, incluyendo:

- **Operaciones matemáticas:** NumPy permite realizar operaciones matemáticas eficientes en arrays, como suma, resta, multiplicación, división, exponenciación, etc.
- **Álgebra lineal:** La biblioteca incluye funciones para realizar operaciones de álgebra lineal, como la multiplicación de matrices, la descomposición de matrices, el cálculo de determinantes, etc.

- **Transformada de Fourier:** NumPy ofrece funciones para realizar la transformada de Fourier discreta, que es una herramienta fundamental en el procesamiento de señales.
- **Generación de números aleatorios:** La biblioteca incluye un módulo para generar números aleatorios con diferentes distribuciones.

Capítulo 4: Sci-kit Learn

Scikit-learn es una biblioteca de aprendizaje automático de código abierto para Python. Proporciona una amplia gama de algoritmos para tareas como:

- **Clasificación:** Predicción de una etiqueta categórica (por ejemplo, spam/no spam).
- **Regresión:** Predicción de un valor continuo (por ejemplo, el precio de una casa).
- **Agrupación:** Agrupación de datos en grupos similares.
- **Reducción de dimensionalidad:** Reducción del número de variables en un conjunto de datos.
- **Selección de modelo:** Comparación y selección del mejor modelo para una tarea dada.
- **Preprocesamiento de datos:** Transformación de datos en un formato adecuado para los algoritmos de aprendizaje automático.

Características clave de Scikit-learn:

- **API consistente:** Todos los algoritmos se implementan con una API similar, lo que facilita su uso e intercambio.
- **Documentación completa:** Scikit-learn tiene una documentación detallada con ejemplos y tutoriales.
- **Comunidad activa:** Una gran comunidad de usuarios y desarrolladores proporciona soporte y contribuye a la biblioteca.

Integración con otras bibliotecas: Scikit-learn se integra bien con otras bibliotecas de Python para el análisis de datos, como NumPy, Pandas y Matplotlib.

Capítulo 5: Algoritmos ML

Los algoritmos de machine learning son el núcleo del campo del aprendizaje automático, una rama de la inteligencia artificial que busca dotar a las máquinas con la capacidad de aprender y generalizar el conocimiento a partir de experiencias.

- **Imitar Comportamientos Inteligentes:** El objetivo principal de la inteligencia artificial es crear máquinas que puedan imitar comportamientos inteligentes. Estos comportamientos pueden incluir la conducción, el análisis de patrones, el reconocimiento de voz y la capacidad de ganar juegos.
- **Tipos de IA:**
    - **IA Débil:** Sistemas que solo pueden realizar un conjunto limitado de tareas.
    - **IA Fuerte:** Sistemas que pueden aplicarse a una variedad de problemas y dominios. Actualmente, todas las IA existentes se consideran débiles.

- **Machine Learning como Capacidad de Aprendizaje:**
    - El machine learning se centra en la capacidad de las máquinas para aprender.
    - El aprendizaje automático implica la generalización del conocimiento a partir de un conjunto de experiencias.
    - Los algoritmos de aprendizaje automático se pueden clasificar en tres grupos: aprendizaje supervisado, no supervisado y reforzado.
- **Aprendizaje Supervisado:**
    - Se basa en descubrir la relación entre variables de entrada y variables de salida.
    - El algoritmo aprende de ejemplos que le muestran el resultado deseado para un valor dado.
    - Una vez entrenado, el algoritmo puede predecir resultados para valores que no ha visto antes.
    - Ejemplos: detección de spam, diagnóstico de enfermedades, predicción del precio de la vivienda.
- **Aprendizaje No Supervisado:**
    - Genera conocimiento únicamente a partir de los datos de entrada, sin ejemplos de salida.
    - Busca patrones de similitud en los datos de entrada para descubrir estructuras y relaciones.
    - Ejemplos: clusterización (agrupamiento de datos), aprendizaje de espacios latentes, reconocimiento de patrones.
- **Técnicas de Machine Learning:**
    - **Árboles de Decisión:** Modelos predictivos que utilizan una estructura de árbol para tomar decisiones.
    - **Modelos de Regresión:** Predicen una variable continua (como el precio de la vivienda) a partir de una o varias variables de entrada.
    - **Modelos de Clasificación:** Clasifican los datos en categorías discretas (como spam o no spam).
    - **Técnicas de Clusterización:** Agrupan datos en función de su similitud.
    - **Redes Neuronales:**
        - Inspiran en la estructura del cerebro humano.
        - Aprenden jerárquicamente, con capas que procesan información de forma cada vez más abstracta.
        - El **Deep Learning** se refiere a redes neuronales con muchas capas, lo que permite el aprendizaje de patrones complejos.
        - El Deep Learning es especialmente útil para procesar grandes cantidades de datos (**Big Data**).

Los algoritmos de machine learning están impulsando la revolución de la inteligencia artificial, permitiendo a las máquinas realizar tareas cada vez más complejas y sofisticadas. A medida que la tecnología avanza y se dispone de más datos, el aprendizaje automático seguirá desempeñando un papel fundamental en la configuración de nuestro futuro.

Capítulo 6: Redes Neuronales

1. Concepto y Estructura:

- **Definición:** Una red neuronal es un modelo computacional inspirado en el cerebro humano, compuesto por unidades básicas llamadas neuronas artificiales.
- **Neuronas:**
    - Son funciones matemáticas que reciben estímulos (entradas), realizan un cálculo interno y producen una salida.
    - Realizan una suma ponderada de las entradas, similar a una regresión lineal.
    - Incluyen un término de sesgo (bias) para ajustar la función.
    - Utilizan una función de activación para introducir no linealidad y permitir la creación de modelos más complejos.
- **Organización:** Las neuronas se organizan en capas: capa de entrada, capas ocultas y capa de salida.
    - La información fluye de la capa de entrada a la de salida, pasando por las capas ocultas.
    - Las capas ocultas permiten a la red aprender jerarquías de conocimiento, procesando la información de forma cada vez más abstracta.

2. Funciones de Activación:

- **Importancia:** Las funciones de activación son cruciales para que la red neuronal pueda modelar relaciones no lineales entre las variables.
- **Tipos:**
    - **Escalonada:** Asigna 0 o 1 a la salida en función de un umbral. No se usa en la práctica por sus limitaciones para el aprendizaje.
    - **Sigmoide:** Transforma la salida a un rango entre 0 y 1, útil para representar probabilidades.
    - **Tangente hiperbólica:** Similar a la sigmoide pero con un rango de -1 a 1.
    - **ReLU (Rectificada Lineal):** Se comporta como una función lineal para valores positivos y es 0 para valores negativos. A menudo se prefiere por su velocidad de entrenamiento.

3. Aprendizaje y Entrenamiento:

- **Objetivo:** El objetivo del entrenamiento es ajustar los parámetros de la red (pesos y bias) para que pueda realizar la tarea deseada (clasificación, regresión, etc.).
- **Algoritmo Backpropagation:** Es el algoritmo fundamental para el entrenamiento de redes neuronales.
    - Permite calcular el error de cada neurona y la influencia de cada parámetro en el error final.
    - Realiza una retropropagación del error desde la capa de salida hacia la capa de entrada, ajustando los pesos en cada paso.

- **Descenso del Gradiente:** Se utiliza en conjunto con backpropagation para minimizar el error de la red.
    - Ajusta los parámetros de la red en la dirección que reduce el error.

4. Representación Geométrica:

- **Visualización:** El proceso de aprendizaje se puede visualizar como la búsqueda de una frontera de separación entre diferentes clases de datos.
- **Efecto de las Funciones de Activación:** Las funciones de activación deforman el espacio de características, permitiendo a la red encontrar fronteras de separación más complejas.
- **Complejidad con Múltiples Neuronas:** Al combinar múltiples neuronas con diferentes funciones de activación, la red puede crear fronteras de separación altamente complejas, como circunferencias o formas irregulares.

5. Ejemplos y Aplicaciones:

- Las fuentes proporcionan ejemplos de uso de redes neuronales para:
    - **Clasificación de puntos en un plano:** Separar nubes de puntos con diferentes formas.
    - **Modelar puertas lógicas:** AND, OR, XOR.
    - **Clasificación de imágenes:** Identificar células cancerosas en una imagen (mencionado como ejemplo).

Capítulo 7: Procesamiento del Lenguaje Natural (NLP)

Las fuentes proporcionadas se centran en la introducción al Procesamiento de Lenguaje Natural (NLP), explicando cómo se pueden representar las palabras numéricamente para que las redes neuronales puedan procesarlas, y cómo los modelos de lenguaje han evolucionado a lo largo del tiempo. A continuación, se presenta un resumen de los puntos clave:

1. El Reto del NLP:

- **Objetivo:** El NLP busca que las máquinas puedan entender el lenguaje humano, tanto escrito como hablado.
- **Dificultades:** El lenguaje humano es complejo, con matices, ambigüedades, y está en constante evolución.
    - Intentar codificar todas las reglas gramaticales y lingüísticas manualmente sería una tarea extremadamente difícil y requeriría actualizaciones constantes.

2. De las Reglas a Machine Learning:

- **Dos Enfoques Históricos:**
    - **Codificación Manual:** Basado en reglas gramaticales y lingüísticas predefinidas.
    - **Machine Learning:** Los algoritmos aprenden patrones del lenguaje a partir de grandes corpus de texto.
- **Tendencia Actual:** En los últimos años, el enfoque basado en Machine Learning ha cobrado mayor importancia.

3. La Necesidad de Vectorizar el Texto:

- **Redes Neuronales y Datos Numéricos:** Las redes neuronales solo pueden procesar datos numéricos, no texto directamente.
- **Representación Numérica:** Para que las redes neuronales puedan trabajar con texto, es necesario convertir las palabras en una representación numérica.

4. Tokenización y Representación Inicial:

- **Tokenización:** División del texto en unidades individuales (tokens), que pueden ser palabras, caracteres o subpalabras.
- **Etiquetado Numérico:** Asignación de una etiqueta numérica única a cada token.
- **Limitaciones:** Esta representación inicial no refleja las relaciones semánticas entre palabras.

5. One-Hot Encoding y Representación Vectorial:

- **One-Hot Encoding:** Cada palabra se representa como un vector con tantas posiciones como palabras haya en el vocabulario, marcando con un 1 la posición correspondiente a la palabra.
- **Ventajas:** Evita la ordenación arbitraria de palabras.
- **Desventajas:**
    - Asigna la misma distancia entre todas las palabras, sin reflejar la proximidad semántica.
    - Genera vectores muy grandes y dispersos, con la mayoría de elementos en cero.

6. Embeddings - Una Representación Más Inteligente:

- **Concepto:** Los embeddings son representaciones vectoriales densas que capturan las relaciones semánticas entre palabras.
- **Aprendizaje:** Se entrenan redes neuronales para aprender estos embeddings a partir de grandes corpus de texto.
- **Ventajas:**
    - Palabras con significado similar tienen vectores cercanos en el espacio vectorial.
    - Reducen la dimensionalidad de los datos, haciendo el procesamiento más eficiente.
- **Ejemplos:** Word2Vec es un ejemplo de un sistema que genera embeddings pre-entrenados.

7. Evolución de los Modelos de Lenguaje:

- **Aprendizaje Supervisado:** Inicialmente, los modelos de lenguaje se entrenaban con datasets etiquetados manualmente, lo cual era costoso y limitaba la cantidad de datos.
- **Aprendizaje Autosupervisado:** Los modelos modernos, como los basados en Transformers, se entrenan con tareas autosupervisadas (como predecir la siguiente palabra en una frase), lo que permite usar grandes cantidades de texto sin necesidad de etiquetado manual.

- **Modelos de Lenguaje a Gran Escala (LLMs):** El aumento en la capacidad de procesamiento y la disponibilidad de grandes datasets ha llevado a la creación de LLMs, como GPT-3, que son capaces de realizar una variedad de tareas de NLP con gran precisión.

8. Transformers - Una Arquitectura Clave:

- **Mecanismos de Atención:** Los Transformers se basan en mecanismos de atención, que permiten a la red focalizar su atención en las partes más relevantes del texto para una tarea dada.
- **Ventajas:**
    - Superan las limitaciones de las redes neuronales recurrentes (RNNs) en cuanto a la memoria a largo plazo.
    - Han impulsado grandes avances en el NLP, como la traducción automática y la generación de texto.

9. Aplicaciones del NLP:

- **Análisis de Sentimientos:** Determinar la emoción expresada en un texto (positivo, negativo, neutral).
- **Traducción Automática:** Traducir texto de un idioma a otro.
- **Generación de Texto:** Crear texto coherente y gramaticalmente correcto, como en chatbots o sistemas de resumen automático.
- **Clasificación de Texto:** Asignar un texto a una categoría predefinida.