**CargarPreprocesar\_Datos**

**Este código carga datos de un archivo CSV, convierte el CONCEPTO en características numéricas, entrena un modelo Naive Bayes multinomial para clasificar CATEGORIAS y permite hacer predicciones sobre las CATEGORIAS que corresponderían a los nuevos CONCEPTOS. En el mismo programa se ve el ejemplo. El fichero de entrada es un .csv de prueba con los apuntes del banco.**

**Importaciones**

**python**

**import pandas as pd**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn.feature\_extraction.text import CountVectorizer**

**from sklearn.preprocessing import LabelEncoder**

**from sklearn.naive\_bayes import MultinomialNB**

**from sklearn.metrics import accuracy\_score**

* **import pandas as pd: Importa la biblioteca pandas y la asigna al alias pd, que se usa para manipulación y análisis de datos.**
* **from sklearn.model\_selection import train\_test\_split: Importa la función train\_test\_split de Scikit-learn, que se utiliza para dividir los datos en conjuntos de entrenamiento y prueba.**
* **from sklearn.feature\_extraction.text import CountVectorizer: Importa CountVectorizer, que convierte una colección de textos en una matriz de conteos de términos.**
* **from sklearn.preprocessing import LabelEncoder: Importa LabelEncoder, que transforma etiquetas categóricas en números.**
* **from sklearn.naive\_bayes import MultinomialNB: Importa MultinomialNB, un clasificador Naive Bayes para datos discretos.**
* **from sklearn.metrics import accuracy\_score: Importa accuracy\_score, una métrica para calcular la precisión del modelo.**

**Cargar Datos**

**python**

**datos = pd.read\_csv('recibos1.csv', sep=';', encoding='latin1')**

* **Carga de datos: Utiliza pd.read\_csv para leer un archivo CSV llamado recibos1.csv.**
  + **sep=';': Especifica que el separador de campos en el archivo CSV es ;.**
  + **encoding='latin1': Especifica la codificación de caracteres para leer el archivo.**

**Convertir Texto en Características Numéricas**

**python**

**vectorizer = CountVectorizer()**

**X = vectorizer.fit\_transform(datos['Concepto'])**

* **Crear CountVectorizer: vectorizer = CountVectorizer() crea una instancia de CountVectorizer.**
* **Transformar Texto: vectorizer.fit\_transform(datos['Concepto']) ajusta el vectorizador al texto de la columna Concepto y transforma los textos en una matriz de conteos de términos.**
  + **fit\_transform: Ajusta el vectorizador a los datos (aprende el vocabulario) y luego transforma el texto en características numéricas.**
  + **datos['Concepto']: Selecciona la columna Concepto del DataFrame datos.**
  + **Resultado X: La matriz resultante se almacena en X.**

**Codificar Categorías en Números**

**python**

**encoder = LabelEncoder()**

**y = encoder.fit\_transform(datos['Categoría'])**

* **Crear LabelEncoder: encoder = LabelEncoder() crea una instancia de LabelEncoder.**
* **Transformar Etiquetas: encoder.fit\_transform(datos['Categoría']) ajusta el codificador a las etiquetas categóricas de la columna Categoría y las transforma en números.**
  + **fit\_transform: Ajusta el codificador a las etiquetas y luego las transforma.**
  + **datos['Categoría']: Selecciona la columna Categoría del DataFrame datos.**
  + **Resultado y: Las etiquetas codificadas se almacenan en y.**

**Dividir los Datos en Entrenamiento y Prueba**

**python**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)**

* **Dividir los Datos: train\_test\_split divide los datos en conjuntos de entrenamiento y prueba.**
  + **X: Matriz de características (texto transformado).**
  + **y: Etiquetas codificadas (categorías).**
  + **test\_size=0.2: El 20% de los datos se utilizará para prueba.**
  + **random\_state=0: Asegura que la división sea reproducible.**
  + **Resultados:**
    - **X\_train: Características de entrenamiento.**
    - **X\_test: Características de prueba.**
    - **y\_train: Etiquetas de entrenamiento.**
    - **y\_test: Etiquetas de prueba.**

**Crear y Entrenar el Modelo**

**python**

**modelo = MultinomialNB()**

**modelo.fit(X\_train, y\_train)**

* **Crear Modelo: modelo = MultinomialNB() crea una instancia del clasificador Naive Bayes multinomial.**
* **Entrenar Modelo: modelo.fit(X\_train, y\_train) ajusta el modelo a los datos de entrenamiento.**
  + **fit: Ajusta el modelo a los datos de entrada (X\_train) y las etiquetas (y\_train).**

**Hacer Predicciones**

**python**

**y\_pred = modelo.predict(X\_test)**

* **Predicciones: modelo.predict(X\_test) utiliza el modelo entrenado para hacer predicciones en el conjunto de prueba (X\_test).**
  + **Resultado y\_pred: Las predicciones del modelo se almacenan en y\_pred.**

**Evaluar el Modelo**

**python**

**precision = accuracy\_score(y\_test, y\_pred)**

**print(f"Precisión del modelo: {precision}")**

* **Calcular Precisión: accuracy\_score(y\_test, y\_pred) calcula la precisión del modelo comparando las etiquetas predichas (y\_pred) con las etiquetas reales (y\_test).**
* **Imprimir Precisión: print(f"Precisión del modelo: {precision}") imprime la precisión del modelo.**

**Función para Predecir Categorías de Nuevos Conceptos**

**python**

**def predecir\_categoria(concepto):**

**concepto\_vectorizado = vectorizer.transform([concepto])**

**categoria\_codificada = modelo.predict(concepto\_vectorizado)[0]**

**return encoder.inverse\_transform([categoria\_codificada])[0]**

* **Definir Función predecir\_categoria: Crea una función para predecir la categoría de un nuevo concepto.**
* **Vectorizar Concepto: vectorizer.transform([concepto]) transforma el nuevo concepto en una matriz de conteos de términos.**
  + **concepto\_vectorizado: Matriz de conteos del nuevo concepto.**
* **Predecir Categoría: modelo.predict(concepto\_vectorizado)[0] utiliza el modelo entrenado para predecir la categoría del nuevo concepto.**
  + **categoria\_codificada: Categoría predicha en forma codificada.**
* **Decodificar Categoría: encoder.inverse\_transform([categoria\_codificada])[0] transforma la categoría codificada de vuelta a su forma original.**
  + **Resultado: La categoría original del nuevo concepto.**

**Ejemplo de Uso**

**python**

**nuevo\_concepto = "RECIB /NATURGY"**

**categoria = predecir\_categoria(nuevo\_concepto)**

**print(f"El concepto '{nuevo\_concepto}' pertenece a la categoría: {categoria}")**

* **Nuevo Concepto: Define un nuevo concepto nuevo\_concepto con el valor "RECIB /NATURGY".**
* **Predecir Categoría: predecir\_categoria(nuevo\_concepto) utiliza la función predecir\_categoria para predecir la categoría del nuevo concepto.**
  + **Resultado categoria: Categoría predicha para el nuevo concepto.**
* **Imprimir Resultado: print(f"El concepto '{nuevo\_concepto}' pertenece a la categoría: {categoria}") imprime el concepto y su categoría predicha.**

**Este código carga datos de un archivo CSV, convierte el texto en características numéricas, entrena un modelo Naive Bayes multinomial para clasificar categorías y permite hacer predicciones sobre nuevos conceptos.**