Ejemplo: Uso de IA para Analizar Grandes Volúmenes de Datos en MySQL y Obtener Insights Útiles

Supongamos que tenemos una base de datos **MySQL** con datos de ventas y queremos usar **Inteligencia Artificial (IA)** para detectar patrones y generar predicciones. Implementaremos un modelo de **Machine Learning en Python** que se conecta a la base de datos, analiza los datos y predice las ventas futuras.

---

1. Instalación de Dependencias

Antes de empezar, asegúrate de tener instaladas las librerías necesarias:

pip install mysql-connector-python pandas scikit-learn matplotlib seaborn

---

2. Conectar a MySQL y Obtener Datos

Vamos a conectarnos a una base de datos de **ventas** y extraer información.

```
import mysql.connector
import pandas as pd

# Conectar a MySQL
db_connection = mysql.connector.connect(
    host="localhost",
    user="usuario",
    password="contraseña",
    database="ventas_db"
)

# Crear cursor y obtener datos
query = """
    SELECT fecha, producto, cantidad_vendida, precio_unitario,
(cantidad_vendida * precio_unitario) AS total_venta
    FROM ventas
"""
df = pd.read_sql(query, db_connection)

# Convertir fecha a datetime
df["fecha"] = pd.to_datetime(df["fecha"])

# Mostrar los primeros registros
print(df.head())
```

---

3. Análisis Exploratorio de Datos (EDA)

Antes de aplicar IA, visualizamos los datos para entender tendencias y patrones.

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Ventas por producto
plt.figure(figsize=(10, 5))
sns.barplot(x=df["producto"], y=df["total_venta"], estimator=sum)
plt.xticks(rotation=45)
plt.title("Ventas Totales por Producto")
plt.show()

# Tendencia de ventas a lo largo del tiempo
df_grouped = df.groupby("fecha")["total_venta"].sum()

plt.figure(figsize=(12, 5))
plt.plot(df_grouped.index, df_grouped.values, marker="o", linestyle="-")
plt.xlabel("Fecha")
plt.ylabel("Total de Ventas")
plt.title("Tendencia de Ventas en el Tiempo")
plt.grid()
plt.show()
```

---

4. Aplicar Machine Learning para Predecir Ventas Futuras

Usaremos **Regresión Lineal** para predecir las ventas de los próximos días.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import LabelEncoder

# Convertir fecha en números
df["fecha_ordinal"] = df["fecha"].map(pd.Timestamp.toordinal)

# Codificar productos en números
encoder = LabelEncoder()
df["producto_encoded"] = encoder.fit_transform(df["producto"])

# Variables predictoras (X) y variable objetivo (y)
X = df[["fecha_ordinal", "producto_encoded", "cantidad_vendida"]]
y = df["total_venta"]

# Dividir en datos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Entrenar modelo de Regresión Lineal
model = LinearRegression()
model.fit(X_train, y_train)

# Evaluar el modelo
score = model.score(X_test, y_test)
print(f"Precisión del modelo: {score:.2f}")
```

5. Predecir Ventas Futuras

Ahora generamos predicciones para los próximos días.

```
import numpy as np

# Crear fechas futuras
future_dates = pd.date_range(start=df["fecha"].max(), periods=10,
freq="D")
future_dates_ordinal = future_dates.map(pd.Timestamp.toordinal)

# Asumimos un producto específico para predicción
producto_pred = "Producto A"
producto_encoded = encoder.transform([producto_pred])[0]

# Simulación de predicción de ventas
cantidad_estimada = np.random.randint(10, 50, size=len(future_dates))

# Construir DataFrame con datos futuros
future_df = pd.DataFrame({
    "fecha": future_dates,
    "fecha_ordinal": future_dates_ordinal,
    "producto_encoded": producto_encoded,
    "cantidad_vendida": cantidad_estimada
})

# Realizar predicciones
future_df["total_venta_predicha"] =
model.predict(future_df[["fecha_ordinal", "producto_encoded",
"cantidad_vendida"]])

# Mostrar predicciones
import ace_tools as tools
tools.display_dataframe_to_user(name="Predicciones de Ventas",
dataframe=future_df)
```

Conclusión

Este proceso permite extraer insights y realizar predicciones en bases de datos **MySQL** con **Machine Learning**. Puedes expandir este modelo incorporando más variables como tendencias estacionales, días festivos o estrategias de marketing para mejorar la precisión.