COMM04

# INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL APLICADA AL MARKETING

**Paso 2: Creación del modelo de Machine Learning en SQL**

```
CREATE OR REPLACE MODEL
`mi_proyecto.mi_dataset.modelo_conversion_clientes`

OPTIONS (model_type='logistic_reg') AS
```

```
SELECT

    edad,

    historial_compras,

    interaccion_ads,

    tiempo_en_sitio,

    case when compra_realizada = 'Sí' then 1 else 0 end as
objetivo

FROM `mi_proyecto.mi_dataset.usuarios`;
```

## Paso 3: Entrenamiento y evaluación del modelo

```
SELECT *

FROM                                        ML.EVALUATE(MODEL
`mi_proyecto.mi_dataset.modelo_conversion_clientes`,

(

    SELECT

        edad,

        historial_compras,

        interaccion_ads,

        tiempo_en_sitio,
```

```
        case when compra_realizada = 'Sí' then 1 else 0 end
as objetivo

    FROM `mi_proyecto.mi_dataset.usuarios`

));
```

## Paso 4: Generación de predicciones en tiempo real

Una vez que el modelo está entrenado y evaluado, se puede utilizar para predecir el comportamiento de nuevos usuarios:

```
SELECT

    edad,

    historial_compras,

    interaccion_ads,

    tiempo_en_sitio,

    ML.PREDICT(MODEL
`mi_proyecto.mi_dataset.modelo_conversion_clientes`,

        STRUCT(30 AS edad, 5 AS historial_compras, 8 AS
interaccion_ads, 15 AS tiempo_en_sitio)) AS prediccion

FROM `mi_proyecto.mi_dataset.usuarios`

LIMIT 10;
```

## Definición del problema y recopilación de datos

```
import pandas as pd


# Cargar datos desde un archivo CSV
datos = pd.read_csv("clientes.csv")


# Ver las primeras filas del dataset
print(datos.head())
```

## Preprocesamiento y limpieza de datos

```
# Eliminar valores nulos
datos.fillna(datos.median(), inplace=True)


# Convertir variables categóricas en numéricas
datos = pd.get_dummies(datos, columns=['genero',
'categoria_producto'])
```

## Selección del algoritmo de Machine Learning

```
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression



# Separar variables predictoras y objetivo

X = datos.drop(columns=['compra_realizada'])

y = datos['compra_realizada']



# Dividir datos en conjunto 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 logística

modelo = LogisticRegression()

modelo.fit(X_train, y_train)
```

## Evaluación del modelo

```
from sklearn.metrics import accuracy_score, classification_report



# Realizar predicciones

y_pred = modelo.predict(X_test)



# Evaluar precisión del modelo

print("Precisión:", accuracy_score(y_test, y_pred))

print("Reporte de clasificación:\n", classification_report(y_test, y_pred))
```

## Generación de predicciones y aplicación en marketing

Una vez validado el modelo, se puede utilizar para hacer predicciones sobre nuevos clientes y automatizar estrategias de marketing. El código en Python para generar predicciones es el siguiente:

```
# Datos de un nuevo cliente para predecir si comprará o no

nuevo_cliente = [[35, 10, 3, 15]]  # Edad, historial de compras, interacciones, tiempo en sitio



# Predicción

probabilidad_compra = modelo.predict_proba(nuevo_cliente)[:,1]

print("Probabilidad de compra:", probabilidad_compra[0])
```

## Implementación del modelo en una estrategia de marketing digital

```
pip install pandas numpy scikit-learn matplotlib seaborn
```

**matplotlib y seaborn** → Visualización de datos y métricas del modelo.

```
import pandas as pd

# Cargar datos de clientes desde un CSV
datos = pd.read_csv("clientes.csv")

# Mostrar las primeras filas del dataset
print(datos.head())

# Verificar información general del dataset
print(datos.info())
```

Después, los datos deben ser **limpiados y transformados** antes de entrenar un modelo de IA. El código en Python para limpiar y transformar datos es el siguiente:

```
# Eliminar filas con valores nulos
datos = datos.dropna()

# Convertir variables categóricas en numéricas
datos = pd.get_dummies(datos, columns=['genero', 'categoria_producto'])

# Normalizar datos numéricos para mejorar el rendimiento del modelo
from sklearn.preprocessing import StandardScaler

escalador = StandardScaler()
columnas_numericas = ['edad', 'historial_compras', 'interaccion_ads']
datos[columnas_numericas] = escalador.fit_transform(datos[columnas_numericas])
```

El código en Python para dividir los datos es:

```
from sklearn.model_selection import train_test_split


# Definir variables predictoras y objetivo
X = datos.drop(columns=['compra_realizada'])
y = datos['compra_realizada']


# Dividir datos en entrenamiento (80%) y prueba (20%)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

A continuación, vemos un ejemplo en el que se utilizará un **modelo de regresión logística**, que es ideal para predecir eventos binarios (como si un usuario comprará o no). El código en Python para entrenar el modelo sería el siguiente:

```
from sklearn.linear_model import LogisticRegression


# Crear y entrenar el modelo
modelo = LogisticRegression()
modelo.fit(X_train, y_train)
```

Una vez entrenado, se debe **evaluar su precisión** para asegurarse de que hace predicciones confiables. El código en Python para evaluar el modelo es el siguiente:

```
from sklearn.metrics import accuracy_score, classification_report


# Generar predicciones en el conjunto de prueba
y_pred = modelo.predict(X_test)


# Evaluar el rendimiento del modelo
print("Precisión:", accuracy_score(y_test, y_pred))
print("Reporte de clasificación:\n", classification_report(y_test, y_pred))
```

Una vez validado, el modelo puede utilizarse para hacer predicciones en nuevos clientes. El código en Python para hacer predicciones es:

```
# Datos de un nuevo cliente

nuevo_cliente = [[30, 5, 10]]  # Edad, historial de compras, interacciones con anuncios



# Predicción de compra

probabilidad_compra = modelo.predict_proba(nuevo_cliente)[:,1]

print("Probabilidad de compra:", probabilidad_compra[0])
```

A continuación, vemos un **esquema básico de un autoencoder:**

```
import tensorflow as tf

from tensorflow import keras

from tensorflow.keras.layers import Input, Dense

from tensorflow.keras.models import Model



# Definir el codificador (encoder)
```

```
input_dim = 100  # Número de características

encoding_dim = 32  # Representación comprimida



input_layer = Input(shape=(input_dim,))

encoded               =               Dense(encoding_dim,
activation='relu')(input_layer)



# Definir el decodificador (decoder)

decoded = Dense(input_dim, activation='sigmoid')(encoded)



# Construcción del autoencoder

autoencoder = Model(input_layer, decoded)



# Compilación del modelo

autoencoder.compile(optimizer='adam',
loss='binary_crossentropy')
```

Un **esquema básico de una CNN en Python** sería el siguiente:

```
import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D,
Flatten, Dense



# Definir la arquitectura de la CNN

modelo_cnn = Sequential([

    Conv2D(32, (3,3), activation='relu', input_shape=(64,
64, 3)),  # Capa convolucional

    MaxPooling2D(pool_size=(2,2)),  # Capa de pooling

    Flatten(),  # Aplanamiento de la información

    Dense(128, activation='relu'),    #  Capa  totalmente
conectada

    Dense(10, activation='softmax')  # Capa de salida para
clasificación

])



# Compilación del modelo

modelo_cnn.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])
```

El código en Python para cargar los datos es el siguiente:

```
import pandas as pd


# Cargar datos históricos de clientes
datos = pd.read_csv("suscripciones.csv")


# Mostrar las primeras filas del dataset
print(datos.head())
```

El código en Python para preprocesar datos es:

```
# Eliminar valores nulos

datos = datos.dropna()



# Convertir variables categóricas en numéricas

datos = pd.get_dummies(datos, columns=['tipo_suscripcion',
'pais'])



# Normalizar datos numéricos

from sklearn.preprocessing import StandardScaler

escalador = StandardScaler()

datos[['uso_diario', 'interacciones', 'facturacion']] =
escalador.fit_transform(datos[['uso_diario',
'interacciones', 'facturacion']])
```

El código en Python para entrenar un modelo de clasificación de clientes es el siguiente:

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

# Definir variables predictoras y objetivo
X = datos.drop(columns=['cancelacion'])
y = datos['cancelacion']

# Dividir datos en 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 Random Forest
modelo = RandomForestClassifier(n_estimators=100, random_state=42)
modelo.fit(X_train, y_train)
```

Una vez entrenado, se debe evaluar el rendimiento del modelo utilizando métricas de clasificación. El código en Python para evaluar el modelo es:

```
from sklearn.metrics import accuracy_score, classification_report

# Generar predicciones
y_pred = modelo.predict(X_test)

# Evaluar precisión del modelo
print("Precisión:", accuracy_score(y_test, y_pred))
print("Reporte de clasificación:\n", classification_report(y_test, y_pred))
```