


# **PLAN DE TRABAJO DEL ESTUDIANTE**

## 1. INFORMACIÓN GENERAL

Apellidos y Nombres:	GANDY WILLIAM HUMIRI QUISPE	ID:	1546329@SENATI.PE
Dirección Zonal/CFP:	Tacna/Moquegua		
Carrera:	Ingeniería de Software con Inteligencia Artificial	Semestre:	4
Instructor	YUSELENIN ANQUISE JIHUAÑA		

## 2. ENTREGABLES:

Durante la investigación de estudio, deberán de dar solución a los planteamientos de cada entregable:

Nº	ENTREGABLE 1			
1	 Archivos CSV 'ventas.csv' y 'usuarios.csv' creados con éxito.			
	Datos de Ventas:			
	ProductoID	UsuarioID	CantidadVendida	FechaVenta
	0	151	2	3 2023-01-01
	1	192	64	7 2023-01-02
	2	114	60	4 2023-01-03
	3	171	21	9 2023-01-04
	4	160	33	3 2023-01-05
	Datos de Usuarios:			
	UsuarioID	TiempoEnSitio	PaginasVistas	ProductosVistos
	0	44	20.362362	11 8
	1	8	46.406284	19 6
	2	47	36.636328	17 3
	3	35	17.000498	8 1
	4	78	28.951755	3 3
Productos más vendidos:				
ProductoID				
174	14			
152	9			
102	9			
187	9			
171	9			
192	7			
186	7			
137	7			
101	5			
120	5			
129	4			
123	4			
114	4			
Nº	ENTREGABLE 1			



**2**

```

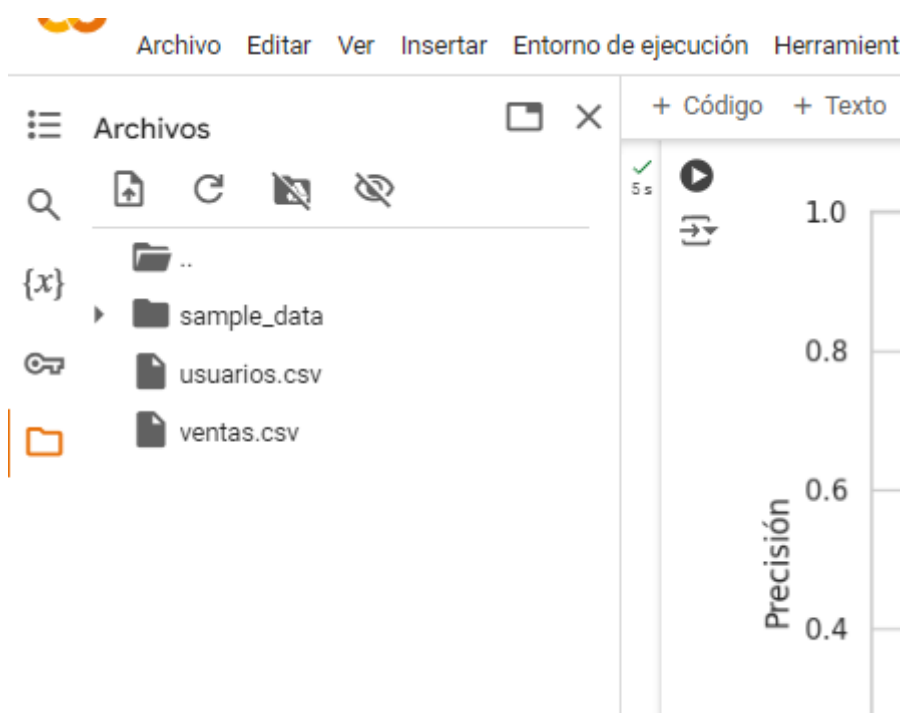
182      3
121      2
199      2
Name: CantidadVendida, dtype: int64

Precisión del modelo KNN: 0.75

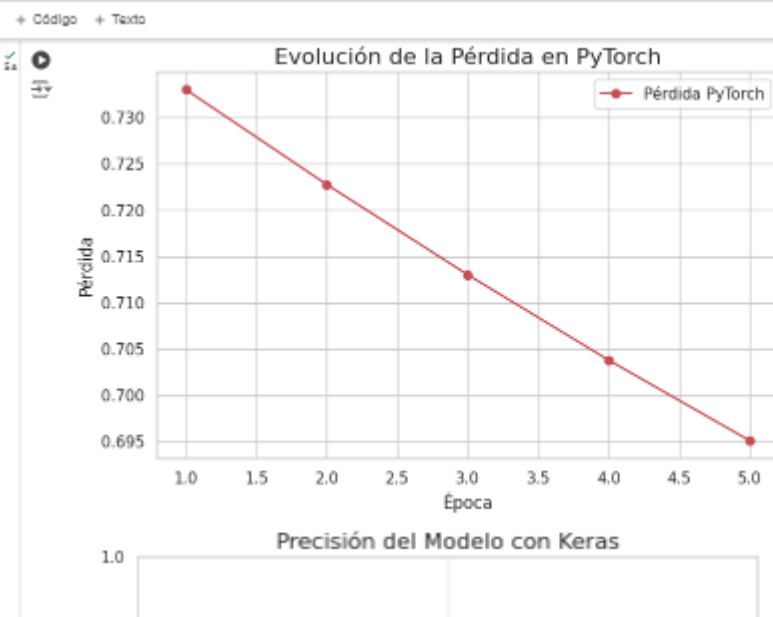
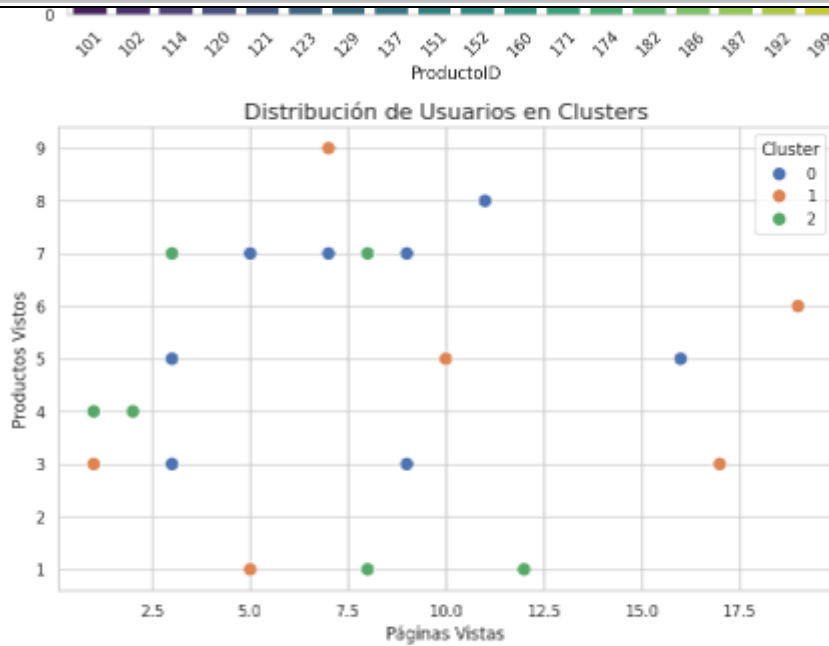
Usuarios agrupados en clusters:
  UsuarioID  Cluster
0         44        0
1          8        1
2         47        1
3         35        2
4         78        0
Época 1, Pérdida: 0.7917
Época 2, Pérdida: 0.7790
Época 3, Pérdida: 0.7668
Época 4, Pérdida: 0.7550
Época 5, Pérdida: 0.7438
Epoch 1/5
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default va
super().__init__(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
5/5 ----- 1s 4ms/step - accuracy: 0.6347 - loss: 0.9935
Epoch 2/5
5/5 ----- 0s 5ms/step - accuracy: 0.5306 - loss: 1.0739
Epoch 3/5
5/5 ----- 0s 4ms/step - accuracy: 0.4542 - loss: 1.0373
Epoch 4/5
5/5 ----- 0s 4ms/step - accuracy: 0.5792 - loss: 0.7413
Epoch 5/5
5/5 ----- 0s 4ms/step - accuracy: 0.5681 - loss: 0.8384
1/1 ----- 0s 204ms/step - accuracy: 0.4500 - loss: 0.7852

Precisión del modelo con Keras: 0.45

```



**Desarrollo del Entregable N°1- DIBUJO / ESQUEMA / DIAGRAMA**  
(Adicionar páginas que sean necesarias)



**HOJA DE PLANIFICACIÓN (Entregable 1)**

```
# 1: CREACIÓN DE ARCHIVOS CSV CON DATOS ALEATORIOS

import pandas as pd
import numpy as np
np.random.seed(42)

# Generamos datos aleatorios para las ventas
ventas_data = {
    'ProductoID': np.random.randint(100, 200, 20),
    'UsuarioID': np.random.randint(1, 100, 20),
    'CantidadVendida': np.random.randint(1, 10, 20),
    'FechaVenta': pd.date_range(start='2023-01-01', periods=20, freq='D').strftime('%Y-%m-%d')
}
ventas_df = pd.DataFrame(ventas_data)

# Guardamos el DataFrame como un archivo CSV
ventas_df.to_csv('ventas.csv', index=False)
# Generamos datos aleatorios para el comportamiento de usuario
usuarios_data = {
    'UsuarioID': np.random.randint(1, 100, 20),
    'TiempoEnSitio': np.random.uniform(1, 50, 20),
    'PaginasVistas': np.random.randint(1, 20, 20),
    'ProductosVistos': np.random.randint(1, 10, 20)
}
usuarios_df = pd.DataFrame(usuarios_data)
usuarios_df.to_csv('usuarios.csv', index=False)

print("Archivos CSV 'ventas.csv' y 'usuarios.csv' creados con éxito.")
```

```
# BLOQUE 2: LECTURA Y ANÁLISIS DE DATOS
ventas = pd.read_csv('ventas.csv')
usuarios = pd.read_csv('usuarios.csv')
# Mostramos los primeros registros de ambos archivos
print("\nDatos de Ventas:")
print(ventas.head())

print("\nDatos de Usuarios:")
print(usuarios.head())
# Análisis de productos más vendidos
productos_mas_vendidos = ventas.groupby('ProductoID')['CantidadVendida'].sum().sort_values(ascending=False)
print("\nProductos más vendidos:")
print(productos_mas_vendidos)

#3: MACHINE LEARNING CON SCIKIT-LEARN

# Importamos las librerías de Scikit-Learn
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.cluster import KMeans
from sklearn.metrics import accuracy_score

# Añadimos una columna de simulación de compra (0: no compra, 1: compra)
usuarios['Compra'] = np.random.randint(0, 2, usuarios.shape[0])
X = usuarios[['TiempoEnSitio', 'PaginasVistas', 'ProductosVistos']]
y = usuarios['Compra']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# --- Modelo KNN (Clasificación) ---
# Creamos y entrenamos un modelo de KNN
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
# Realizamos predicciones
```

## Desarrollo del Entregable N°2 - DIBUJO / ESQUEMA / DIAGRAMA

(Adicionar páginas que sean necesarias)

```

y_pred = knn.predict(x_test)
# Calculamos la precisión del modelo
accuracy = accuracy_score(y_test, y_pred)
print(f"\nPrecisión del modelo KNN: {accuracy:.2f}")

# --- Modelo K-Means (Clustering) ---
# Creamos y entrenamos un modelo de K-Means
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X)

# Asignamos los usuarios a clusters
usuarios['cluster'] = kmeans.labels_

print("\nUsuarios agrupados en clusters:")
print(usuarios[['UsuarioID', 'cluster']].head())

#4: MACHINE LEARNING CON PYTORCH

# Importamos las librerías de PyTorch
import torch
import torch.nn as nn
import torch.optim as optim

# Definimos una red neuronal simple con PyTorch
class RedNeuronal(nn.Module):
    def __init__(self):
        super(RedNeuronal, self).__init__()
        self.fc1 = nn.Linear(3, 10)
        self.fc2 = nn.Linear(10, 1)

    def forward(self, x):
        x = torch.relu(self.fc1(x))
        x = torch.sigmoid(self.fc2(x))
        return x

```

