ANEXOS:
CREA PROGRAMAS CON ALGORITMOS DE APRENDIZAJE SUPERVISADO

SESION 2 / SEMANA 9

Nota: _____

Anexo 01: Tarea Participación 01-B

Fecha Entrega: 03/11/2024 23:59:59

Nombre de Equipo: Anexo 01

Estudiantes (Máximo 4 personas):

- Juan Piero Vincha
- Gianella Ayca
- Karol Mancheco

1. La empresa AMAZON requiere conocer estadísticamente el total (Suma de todos los subtotales), promedio, mediana, desviación estándar (de columna subtotal) y el producto más vendido tanto por tienda como por toda la empresa en función de la columna total. Las ventas de cada tienda esta registrado en archivos CSV de donde extraen los datos y los analizaran con las bibliotecas como numpy, pandas u otros que permitan calcular los datos solicitados (10PTOS)

   Adjunto CSV:
   ventas_amazon_arequipa.csv
   ventas_amazon_ica.csv
   ventas_amazon_lima.csv
   ventas_amazon_tacna.csv

   Rpta: Código Python y pantalla resultado

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report,
confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

def crear_datos_entrenamiento():
    """
    Crea un conjunto de datos más extenso para entrenamiento del
modelo
```

```
    """
    productos = ['Laptop', 'Mouse', 'Teclado', 'Monitor',
'Auriculares', 'Tablet', 'Impresora']

    # Crear características para cada producto
    data = []
    np.random.seed(42)

    for _ in range(1000):  # Crear 1000 registros
        producto = np.random.choice(productos)

        # Características que pueden influir en las ventas
        precio = np.random.normal({
            'Laptop': 1200,
            'Mouse': 25,
            'Teclado': 45,
            'Monitor': 300,
            'Auriculares': 50,
            'Tablet': 350,
            'Impresora': 200
        }[producto], scale=50)

        mes = np.random.randint(1, 13)
        dia_semana = np.random.randint(1, 8)

        # Variable objetivo: 1 si se vendió, 0 si no
        vendido = np.random.binomial(1, {
            'Laptop': 0.7,
            'Mouse': 0.8,
            'Teclado': 0.75,
            'Monitor': 0.6,
            'Auriculares': 0.65,
            'Tablet': 0.55,
            'Impresora': 0.5
        }[producto])

        data.append([producto, precio, mes, dia_semana, vendido])

    return pd.DataFrame(data, columns=['producto', 'precio',
'mes', 'dia_semana', 'vendido'])

def preparar_datos(df):
    """
    Prepara los datos para el modelo de machine learning
    """
```

```
    # Codificar la variable de producto
    le = LabelEncoder()
    df['producto_encoded'] = le.fit_transform(df['producto'])

    # Normalizar características numéricas
    scaler = StandardScaler()
    X = scaler.fit_transform(df[['producto_encoded', 'precio',
'mes', 'dia_semana']])
    y = df['vendido']

    return X, y, le

def entrenar_modelo(X, y):
    """
    Entrena el modelo de regresión logística
    """
    # 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)

    # Crear y entrenar el modelo
    modelo = LogisticRegression(random_state=42)
    modelo.fit(X_train, y_train)

    # Evaluar el modelo
    predicciones = modelo.predict(X_test)
    probabilidades = modelo.predict_proba(X_test)

    return modelo, X_test, y_test, predicciones, probabilidades

def visualizar_resultados(modelo, X_test, y_test, predicciones,
le, df):
    """
    Visualiza los resultados del modelo
    """
    # Crear figura con subplots
    plt.figure(figsize=(15, 10))

    # 1. Matriz de confusión
    plt.subplot(2, 2, 1)
    cm = confusion_matrix(y_test, predicciones)
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title('Matriz de Confusión')
    plt.xlabel('Predicho')
    plt.ylabel('Real')
```

```
    # 2. Probabilidades de venta por producto
    plt.subplot(2, 2, 2)
    productos_prob = []
    for producto in le.classes_:
        producto_encoded = le.transform([producto])[0]
        precio_medio = df[df['producto'] ==
producto]['precio'].mean()
        X_pred = np.array([[producto_encoded, precio_medio, 6,
3]])  # mes=6, dia=3 como ejemplo
        prob = modelo.predict_proba(X_pred)[0][1]
        productos_prob.append((producto, prob))

    productos_prob.sort(key=lambda x: x[1], reverse=True)
    productos, probs = zip(*productos_prob)

    plt.barh(productos, probs)
    plt.title('Probabilidad de Venta por Producto')
    plt.xlabel('Probabilidad')

    # 3. Importancia de características
    plt.subplot(2, 2, 3)
    caracteristicas = ['Producto', 'Precio', 'Mes', 'Día']
    importancia = abs(modelo.coef_[0])
    plt.bar(caracteristicas, importancia)
    plt.title('Importancia de Características')
    plt.xticks(rotation=45)

    plt.tight_layout()
    plt.savefig('resultados_modelo.png')
    plt.close()

def main():
    # Crear y preparar datos
    print("Creando conjunto de datos de entrenamiento...")
    df = crear_datos_entrenamiento()

    print("\nPreparando datos para el modelo...")
    X, y, le = preparar_datos(df)

    print("\nEntrenando modelo...")
    modelo, X_test, y_test, predicciones, probabilidades =
entrenar_modelo(X, y)

    print("\nVisualizando resultados...")
```

```
    visualizar_resultados(modelo, X_test, y_test, predicciones,
le, df)

    # Imprimir reporte de clasificación
    print("\nReporte de Clasificación:")
    print(classification_report(y_test, predicciones))

    # Mostrar probabilidades de venta por producto
    print("\nProbabilidad de venta por producto:")
    productos_prob = []
    for producto in le.classes_:
        producto_encoded = le.transform([producto])[0]
        precio_medio = df[df['producto'] ==
producto]['precio'].mean()
        X_pred = np.array([[producto_encoded, precio_medio, 6,
3]])  # mes=6, dia=3 como ejemplo
        prob = modelo.predict_proba(X_pred)[0][1]
        productos_prob.append((producto, prob))

    productos_prob.sort(key=lambda x: x[1], reverse=True)
    for producto, prob in productos_prob:
        print(f"{producto}: {prob*100:.2f}%")

if __name__ == "__main__":
    main()
```

```
@Allenaig →/workspaces/Practicas-1 (main) $ python tarea122.py
Creando conjunto de datos de entrenamiento...

Preparando datos para el modelo...

Entrenando modelo...

Visualizando resultados...

Reporte de Clasificación:
/home/codespace/.local/lib/python3.12/site-packages/sklearn/metrics/_clas
abels with no predicted samples. Use `zero_division` parameter to control
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/home/codespace/.local/lib/python3.12/site-packages/sklearn/metrics/_clas
abels with no predicted samples. Use `zero_division` parameter to control
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/home/codespace/.local/lib/python3.12/site-packages/sklearn/metrics/_clas
abels with no predicted samples. Use `zero_division` parameter to control
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        81
           1       0.59      1.00      0.75       119

    accuracy                           0.59       200
   macro avg       0.30      0.50      0.37       200
weighted avg       0.35      0.59      0.44       200


Probabilidad de venta por producto:
Laptop: 100.00%
Monitor: 100.00%
Tablet: 100.00%
Impresora: 100.00%
Teclado: 99.98%
Auriculares: 99.94%
Mouse: 99.81%
```

2. Luego analizar los datos, implementar el siguiente modelo aprendizaje supervisado de Machine Learning para determinar que producto es más probable de vender. Rpta en código python y pantalla de resultado. (10PTOS)

```python
import pandas as pd
import numpy as np
import os

def crear_datos_ejemplo():
    """
    Crea archivos CSV de ejemplo para demostración.
    """
    # Datos para la primera tienda
    datos_tienda1 = {
        'producto': ['Laptop', 'Mouse', 'Teclado', 'Monitor', 'Auriculares'],
        'subtotal': [1200, 25, 45, 300, 50],
        'total': [1400, 30, 55, 350, 60]
    }

    # Datos para la segunda tienda
    datos_tienda2 = {
        'producto': ['Laptop', 'Impresora', 'Tablet', 'Monitor', 'Mouse'],
        'subtotal': [1100, 200, 350, 280, 20],
        'total': [1300, 240, 420, 330, 25]
```

```
    }

    # Crear directorio si no existe
    if not os.path.exists('data'):
        os.makedirs('data')

    # Guardar los datos en archivos CSV
    pd.DataFrame(datos_tienda1).to_csv('data/tienda1.csv', index=False)
    pd.DataFrame(datos_tienda2).to_csv('data/tienda2.csv', index=False)

def analizar_ventas_tienda(archivo):
    """
    Analiza los datos de ventas de una tienda individual.

    Args:
        archivo: Ruta al archivo CSV de la tienda
    Returns:
        tuple: (DataFrame de la tienda, diccionario con estadísticas)
    """
    # Leer el archivo CSV
    df = pd.read_csv(archivo)

    # Calcular estadísticas
    estadisticas = {
        'nombre_tienda': os.path.basename(archivo).replace('.csv', ''),
        'total_ventas': df['subtotal'].sum(),
        'promedio_ventas': df['subtotal'].mean(),
        'mediana_ventas': df['subtotal'].median(),
        'desviacion_estandar': df['subtotal'].std(),
        'producto_mas_vendido': df.loc[df['total'].idxmax(), 'producto']
    }

    return df, estadisticas

def analizar_todas_tiendas(directorio_csvs):
    """
    Analiza los datos de ventas de todas las tiendas.
    """
    # Listas para almacenar DataFrames y estadísticas
    dfs_tiendas = []
    estadisticas_tiendas = []

    # Verificar si el directorio existe
    if not os.path.exists(directorio_csvs):
        raise FileNotFoundError(f"El directorio {directorio_csvs} no existe")
```

```
    # Procesar cada archivo CSV en el directorio
    archivos_csv = [f for f in os.listdir(directorio_csvs) if f.endswith('.csv')]

    if not archivos_csv:
        raise FileNotFoundError("No se encontraron archivos CSV en el
directorio")

    for archivo in archivos_csv:
        ruta_completa = os.path.join(directorio_csvs, archivo)
        df_tienda, stats_tienda = analizar_ventas_tienda(ruta_completa)
        dfs_tiendas.append(df_tienda)
        estadisticas_tiendas.append(stats_tienda)

    # Combinar todos los DataFrames
    df_global = pd.concat(dfs_tiendas, ignore_index=True)

    # Calcular estadísticas globales
    estadisticas_globales = {
        'total_empresa': df_global['subtotal'].sum(),
        'promedio_empresa': df_global['subtotal'].mean(),
        'mediana_empresa': df_global['subtotal'].median(),
        'desviacion_estandar_empresa': df_global['subtotal'].std(),
        'producto_mas_vendido_empresa':
df_global.loc[df_global['total'].idxmax(), 'producto']
    }

    return {
        'estadisticas_globales': estadisticas_globales,
        'estadisticas_tiendas': estadisticas_tiendas
    }

def imprimir_resultados(resultados):
    """
    Imprime los resultados del análisis de manera formateada.
    """
    print("\n=== ESTADÍSTICAS GLOBALES DE LA EMPRESA ===")
    print(f"Total de ventas:
${resultados['estadisticas_globales']['total_empresa']:,.2f}")
    print(f"Promedio de ventas:
${resultados['estadisticas_globales']['promedio_empresa']:,.2f}")
    print(f"Mediana de ventas:
${resultados['estadisticas_globales']['mediana_empresa']:,.2f}")
```

```
    print(f"Desviación estándar:
${resultados['estadisticas_globales']['desviacion_estandar_empresa']:,.2f}
")
    print(f"Producto más vendido:
{resultados['estadisticas_globales']['producto_mas_vendido_empresa']}")

    print("\n=== ESTADÍSTICAS POR TIENDA ===")
    for tienda in resultados['estadisticas_tiendas']:
        print(f"\nTienda: {tienda['nombre_tienda']}")
        print(f"Total de ventas: ${tienda['total_ventas']:,.2f}")
        print(f"Promedio de ventas: ${tienda['promedio_ventas']:,.2f}")
        print(f"Mediana de ventas: ${tienda['mediana_ventas']:,.2f}")
        print(f"Desviación estándar: ${tienda['desviacion_estandar']:,.2f}")
        print(f"Producto más vendido: {tienda['producto_mas_vendido']}")

if __name__ == "__main__":
    try:
        # Crear datos de ejemplo
        crear_datos_ejemplo()

        # Analizar los datos
        resultados = analizar_todas_tiendas('data')
        imprimir_resultados(resultados)

    except Exception as e:
        print(f"Error al procesar los datos: {str(e)}")
```

```
@Allenaig ➜/workspaces/Practicas-1 (main) $ python tarea123.py

=== ESTADÍSTICAS GLOBALES DE LA EMPRESA ===
Total de ventas: $3,570.00
Promedio de ventas: $357.00
Mediana de ventas: $240.00
Desviación estándar: $436.14
Producto más vendido: Laptop

=== ESTADÍSTICAS POR TIENDA ===

Tienda: tienda1
Total de ventas: $1,620.00
Promedio de ventas: $324.00
Mediana de ventas: $50.00
Desviación estándar: $502.56
Producto más vendido: Laptop

Tienda: tienda2
Total de ventas: $1,950.00
Promedio de ventas: $390.00
Mediana de ventas: $280.00
Desviación estándar: $415.57
Producto más vendido: Laptop
@Allenaig ➜/workspaces/Practicas-1 (main) $
```