05.1_Ingenieria_Avanzada_95

Objetivo

Aplicar un conjunto de técnicas de ingeniería de características avanzadas sobre los datos con 95 variables iniciales, generando nuevos scores de dominio, perfiles conductuales e interacciones para obtener un dataset enriquecido que sirva de base para el modelado.

Entradas (Inputs)

- `data/splits/final/X_train.parquet`
- `data/splits/final/X_val.parquet`
- `data/splits/final/X_test.parquet`

Salidas (Outputs)

Splits (en `data/engineered/final/`):

- `X_train_95_ultimate.parquet`
- `X_val_95_ultimate.parquet`
- `X_test_95_ultimate.parquet`

Resumen Ejecutivo

El objetivo de este notebook es desarrollar y aplicar un pipeline de ingeniería de características “Ultimate” sobre un dataset de 95 variables originales, con el fin de enriquecerlo para futuros modelos de riesgo financiero. Tras montar el entorno y cargar el conjunto de entrenamiento (1 976 muestras × 92 atributos numéricos), se definen funciones especializadas para generar scores de conocimiento financiero, actitud ante el riesgo y diversidad, así como métricas de brecha de percepción y varias interacciones de primer y segundo orden. Al ejecutar el pipeline maestro, se crean 11 nuevas variables, elevando la dimensión del dataset a 103 características. Finalmente, el dataset enriquecido se guarda de forma reproducible para los siguientes pasos de modelado.

1. Configurar entorno de trabajo

Monta Google Drive en Colab, desactiva las advertencias, añade la raíz del proyecto a `sys.path` y carga las rutas de entrada y salida definidas en la configuración.

```
# Montar Drive, Cargar Librerías y Configuración (ACTUALIZADO)

# Google Colab
from google.colab import drive

# Standard library
import sys
import warnings
from pathlib import Path

# Data processing
import pandas as pd
import numpy as np

# 1. Montar Google Drive
drive.mount('/content/drive', force_remount=True)
warnings.filterwarnings('ignore')

# 2. Añadir la raíz del proyecto al path para importar módulos personalizados
ROOT_PATH_STR = '/content/drive/MyDrive/TFM-AntonioEsquinas'
if ROOT_PATH_STR not in sys.path:
    sys.path.append(ROOT_PATH_STR)

# 3. Importar las rutas FINALES desde el archivo de configuración
from config import FINAL_SPLITS_DIR, FINAL_ENGINEERED_DATA_DIR

print(" Entorno preparado.")
print(f"   -> Datos de entrada se leerán de: {FINAL_SPLITS_DIR}")
print(f"   -> Datos de salida se guardarán en: {FINAL_ENGINEERED_DATA_DIR}")
```

```
Mounted at /content/drive
✅ Entorno preparado.
   -> Datos de entrada se leerán de: /content/drive/MyDrive/Digitech/TFG/ML/Calculo-R
   -> Datos de salida se guardarán en: /content/drive/MyDrive/Digitech/TFG/ML/Calculc
```

2. Cargar dataset con 95 características

Lee el dataset original de partida con 95 características desde el directorio configurado en `FINAL_SPLITS_DIR`, almacenándolo en un DataFrame de Pandas y mostrando sus dimensiones.

```
# Cargar el Dataset de Partida (95 Características)

print(f"Cargando dataset base desde: '{FINAL_SPLITS_DIR}'...")

# Cargar desde la carpeta de splits finales
X_train = pd.read_parquet(FINAL_SPLITS_DIR / 'X_train.parquet')
X_val = pd.read_parquet(FINAL_SPLITS_DIR / 'X_val.parquet')
X_test = pd.read_parquet(FINAL_SPLITS_DIR / 'X_test.parquet')

print(f" Dataset cargado. Shape de X_train: {X_train.shape}")
```

```
Cargando dataset base desde: '/content/drive/MyDrive/Digitech/TFG/ML/Calculo-Riesgo/d
✅ Dataset cargado. Shape de X_train: (1976, 92)
```

3. Definir funciones de ingeniería de características

Implementa todas las funciones auxiliares necesarias para generar, transformar y combinar nuevas variables, que se usarán en el pipeline de ingeniería avanzada.

```
# Todas las Funciones de Ingeniería

# --- 1: Funciones para crear "Scores" de Dominio ---

def create_base_scores(df):
    """Crea los scores fundamentales de Conocimiento, Actitud y Diversidad."""
    df_eng = df.copy()

    # Score de Conocimiento Financiero (usando el mapeo correcto de respuestas)
    knowledge_q = {
        'G4': 1, 'G5': 1, 'G6': 2, 'G7': 1, 'G8': 1,
        'G11': 1, 'G12': 2, 'G13': 3, 'G21': 2
    }
    for col, correct_answer in knowledge_q.items():
        if col in df_eng.columns:
            df_eng[f'{col}_correct'] = (df_eng[col] == correct_answer).astype(int)

    knowledge_cols = [col for col in df_eng.columns if '_correct' in col]
    df_eng['Score_Conocimiento'] = df_eng[knowledge_cols].sum(axis=1)
    df_eng = df_eng.drop(columns=knowledge_cols)

    # Score de Actitud Pro-Riesgo
    c25_norm = (df_eng['C25'] - 1) / 9.0
    g1_norm = (df_eng['G1'] - 1) / 3.0
    d31_inverted_norm = 1 - ((df_eng['D31'] - 1) / 6.0)
    df_eng['Score_Actitud_ProRiesgo'] = (c25_norm + g1_norm + d31_inverted_norm) / 3

    # Score de Diversidad de Activos
    asset_cols = [col for col in df.columns if col.startswith('B2_')]
    df_eng['Score_Diversidad'] = (df[asset_cols] == 1).sum(axis=1)

    return df_eng

# --- 2: Funciones para crear "Perfiles Psicológicos y de Comportamiento" ---

def create_behavioral_profiles(df):
    """Crea perfiles que miden sesgos y comportamientos complejos."""
    df_eng = df.copy()

    # Perfil de Inconsistencia: Inversores preocupados con alta exposición
    # (Se asume que D31 y B11 existen, si no, habría que protegerlas también)
    worried_investor = (df_eng['D31'] >= 6)
```

```
        high_stock_exposure = (df_eng['B11'] >= 4) # B11: 4 => 50-74% en acciones
        df_eng['Perfil_Inconsistencia_Riesgo'] = (worried_investor & high_stock_exposure).ast

        # Perfil de Sobreconfianza (Dunning-Kruger)
        g2_norm = (df_eng['G2'] - 1) / 3.0
        # Asumimos que G7 existe para la normalización del score. Si no, habría que ajustar.
        score_k_norm = df_eng['Score_Conocimiento'] / df_eng['G7'].shape[0]
        df_eng['Gap_Confianza_vs_Conocimiento'] = g2_norm - score_k_norm

        # Perfil de Sofisticación del Inversor (protegido contra KeyErrors)

        # Comprueba si B6 existe. Si no, b6_used_margin será 0.
        if 'B6' in df_eng.columns:
            b6_used_margin = (df_eng['B6'] == 1).astype(int)
        else:
            b6_used_margin = 0

        # Comprueba si B34 existe. Si no, b34_used_options será 0.
        if 'B34' in df_eng.columns:
            b34_used_options = (df_eng['B34'] == 1).astype(int)
        else:
            b34_used_options = 0

        # Asumimos que ya existe 'Score_Conocimiento' y 'Score_Diversidad'
        df_eng['Perfil_Sofisticacion'] = df_eng['Score_Conocimiento'] + df_eng['Score_Diversi

        # Score de Comportamiento Emocional
        d2_norm = (df_eng['D2'] - 1) / 4.0
        d3_norm = (df_eng['D3'] - 1) / 4.0
        d21_inverted_norm = 1 - ((df_eng['D21'] - 1) / 3.0)
        df_eng['Score_Comportamiento_Emocional'] = (d2_norm + d3_norm + d21_inverted_norm) /

        return df_eng

    # --- 3: Funciones para crear Interacciones y Bins Avanzados ---

    def create_advanced_interactions_and_bins(df):
        """Crea interacciones y categorías con mucho sentido de negocio."""
        df_eng = df.copy()

        # Interacción: Cómo reacciona a caídas ponderado por su exposición a acciones
        df_eng['Reaccion_x_Exposicion'] = df_eng['D21'] * df_eng['B11']

        # Gap de Percepción de Riesgo
        df_eng['Gap_Riesgo_CriptoVsAcciones'] = df_eng['B24'] - df_eng['B25']

        # Interacción de segundo nivel: Actitud x Conocimiento
        df_eng['Actitud_x_Conocimiento'] = df_eng['Score_Actitud_ProRiesgo'] * df_eng['Score_

        # Binning de B4 (Valor de cartera)
        df_eng['B4_cat'] = pd.qcut(df_eng['B4'], q=5, labels=False, duplicates='drop')

        return df_eng

    print(" Funciones de ingeniería de características 'Ultimate' definidas.")
```

```
✅ Funciones de ingeniería de características 'Ultimate' definidas.
```

4. Crear y aplicar pipeline maestro de ingeniería

Define la función principal que orquesta las distintas funciones de ingeniería y aplica este pipeline al dataset de partida para obtener el dataset enriquecido.

```
# Función Maestra y Aplicación

def apply_ultimate_feature_engineering(df):
    """Aplica toda la cadena de ingeniería avanzada en el orden correcto."""
    df = create_base_scores(df)
    df = create_behavioral_profiles(df)
    df = create_advanced_interactions_and_bins(df)
    return df

print("Aplicando ingeniería de características 'Ultimate' a los datasets...")

# Creamos una copia para no modificar los dataframes originales
X_train_to_eng = X_train.copy()
X_val_to_eng = X_val.copy()
X_test_to_eng = X_test.copy()

# Aplicar la ingeniería
X_train_eng = apply_ultimate_feature_engineering(X_train_to_eng)
X_val_eng = apply_ultimate_feature_engineering(X_val_to_eng)
X_test_eng = apply_ultimate_feature_engineering(X_test_to_eng)

# Ahora, nos quedamos solo con las columnas NUEVAS que hemos creado
new_feature_cols = [col for col in X_train_eng.columns if col not in X_train.columns]

# Concatenamos las 95 originales con las nuevas para el dataset final
X_train_final = pd.concat([X_train, X_train_eng[new_feature_cols]], axis=1)
X_val_final = pd.concat([X_val, X_val_eng[new_feature_cols]], axis=1)
X_test_final = pd.concat([X_test, X_test_eng[new_feature_cols]], axis=1)

print("\n Ingeniería de características completada.")
print(f"Número de características originales: {X_train.shape[1]}")
print(f"Número de características nuevas creadas: {len(new_feature_cols)}")
print(f"Shape del nuevo dataset final enriquecido: {X_train_final.shape}")
```

```
Aplicando ingeniería de características 'Ultimate' a los datasets...

✅ Ingeniería de características completada.
Número de características originales: 92
Número de características nuevas creadas: 11
Shape del nuevo dataset final enriquecido: (1976, 103)
```

5. Guardar dataset enriquecido

Escribe el nuevo DataFrame enriquecido en formato Parquet en la carpeta de salida (`FINAL_ENGINEERED_DATA_DIR`), asegurando reproducibilidad y trazabilidad.

```
# Guardar el Nuevo Dataset Enriquecido

file_name_suffix = '95_ultimate'

X_train_final.to_parquet(FINAL_ENGINEERED_DATA_DIR / f'X_train_{file_name_suffix}.parquet
X_val_final.to_parquet(FINAL_ENGINEERED_DATA_DIR / f'X_val_{file_name_suffix}.parquet')
X_test_final.to_parquet(FINAL_ENGINEERED_DATA_DIR / f'X_test_{file_name_suffix}.parquet')

print(f" Dataset final '{file_name_suffix}' guardado en: {FINAL_ENGINEERED_DATA_DIR}")
```

```
Dataset final '95_ultimate' guardado en: /content/drive/MyDrive/Digitech/TFG/ML/C
```

Conclusiones Finales

- Se agregaron 11 variables derivadas que resumen información clave:
  - **Scores de dominio** (conocimiento, actitud, diversidad) que codifican respuestas cuestionario → valores cuantitativos.
  - **Gap de percepción de riesgo** (cripto vs. acciones) y **reacciones ponderadas** por exposición.
  - **Interacciones de primer y segundo nivel** (e.g., actitud × conocimiento).
  - **Binning** de valor de cartera en 5 categorías equitativas.
- Estas nuevas variables capturan tanto rasgos latentes del inversor como relaciones no lineales entre atributos originales, ofreciendo una representación de mayor expresión para modelos posteriores.
- El aumento de dimensionalidad (de 92 a 103 features) está controlado y orientado a mejorar la capacidad predictiva sin perder interpretabilidad.
- El dataset resultante, con 1 976 observaciones y 103 variables, queda listo para entrenar y evaluar modelos de machine learning con mayor riqueza de información.