

✓ 03.2_Seleccion_Features_Automatica

Objetivo

Ejecutar y comparar múltiples técnicas de selección automática de características (ANOVA, Información Mutua, RFECV, etc.) sobre los datos de entrenamiento finales.

El objetivo es identificar el subconjunto de variables más robusto y con mayor poder predictivo para guardar una nueva versión experimental de los splits (entrenamiento, validación y prueba) que contenga únicamente estas características seleccionadas.

Entradas (Inputs)

- data/splits/final/X_train.parquet
- data/splits/final/y_train.parquet
- data/splits/final/X_val.parquet
- data/splits/final/y_val.parquet
- data/splits/final/X_test.parquet
- data/splits/final/y_test.parquet

Salidas (Outputs)

Splits:

- data/splits/experiments/X_train_14.parquet
- data/splits/experiments/X_val_14.parquet
- data/splits/experiments/X_test_14.parquet
- data/splits/experiments/y_train_14.parquet
- data/splits/experiments/y_val_14.parquet
- data/splits/experiments/y_test_14.parquet

Artefactos:

- artifacts/experiments/03_2_selected_features_rfecv_14.json
 - artifacts/experiments/03_2_pipeline_rfecv14_logistic.pkl
 - artifacts/experiments/03_2_pipeline_pca_logistic.pkl
-

Resumen Ejecutivo

- El notebook automatiza la selección de características para un modelo de **Regresión Logística** sobre los “splits” de entrenamiento, validación y test previamente generados.

- Se emplean múltiples métodos de feature selection: **SelectKBest (ANOVA)**, **InfoMutua**, **Random Forest (importancia y umbral mediano)**, **RFECV** y un filtrado por frecuencia de selección.
 - Se compara también un enfoque de **reducción de dimensionalidad con PCA** (90 % varianza) frente a las selecciones basadas en árboles y univariadas.
 - Cada subconjunto resultante (p. ej., Top-20 de RFECV, intersección RFMI, PCA-64) se evalúa con **LogisticRegression** mediante métricas de precisión, recall, f1-score y accuracy en validación y test.
 - La validación CV sobre distintos tamaños de subset muestra un **plateau de f1_macro (~0.43)** a partir de k = 20 características.
 - La selección univariada (k=20) redujo accuracy (0.46→0.41), mientras que RFECV-Top-20 alcanzó ~0.43 y el pipeline **PCA+Logistic** obtuvo **0.46** en validación y **0.49** en test.
 - Se identifican características recurrentes (e.g., `TRADER_SCORE`, `S_Age`, `B4_log`, variables de series F31, G30) como las más predictivas.
 - Finalmente, los modelos y splits procesados se serializan en **formato Parquet** y **pkl** para facilitar su uso en fases posteriores.
-

1. Montaje de Google Drive, configuración de rutas y carga de datos finales

Monta Google Drive, añade la raíz del proyecto al sys.path, carga las rutas desde el módulo config y lee los archivos Parquet de los splits finales. Luego detecta y reordena variables derivadas si existen.

```
from collections import Counter
import json
import sys
from pathlib import Path

from google.colab import drive
import joblib
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import SelectKBest, f_classif, mutual_info_classif, Select
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import cross_val_score, StratifiedKFold
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

# Montar Google Drive
```

```

drive.mount('/content/drive', force_remount=True)

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

# Importar las rutas necesarias desde el archivo de configuración
import config # Import the config module itself
from config import FINAL_SPLITS_DIR, EXP_SPLITS_DIR, EXP_ARTIFACTS_DIR

print("Drive montado y configuración de rutas cargada correctamente.")

# Carga de splits FINALES y variables derivadas

print(f"Cargando datos FINALES desde: {config.FINAL_SPLITS_DIR}")

try:
    # Cargar splits usando la ruta FINAL del archivo config
    X_train = pd.read_parquet(config.FINAL_SPLITS_DIR / 'X_train.parquet')
    y_train = pd.read_parquet(config.FINAL_SPLITS_DIR / 'y_train.parquet').squeeze()

    X_val = pd.read_parquet(config.FINAL_SPLITS_DIR / 'X_val.parquet')
    y_val = pd.read_parquet(config.FINAL_SPLITS_DIR / 'y_val.parquet').squeeze()

    X_test = pd.read_parquet(config.FINAL_SPLITS_DIR / 'X_test.parquet')
    y_test = pd.read_parquet(config.FINAL_SPLITS_DIR / 'y_test.parquet').squeeze()

    print("Datos cargados correctamente desde formato Parquet.")

    print("\n Shapes tras cargar splits:")
    print(f" • X_train: {X_train.shape}, y_train: {y_train.shape}")
    print(f" • X_val: {X_val.shape}, y_val: {y_val.shape}")
    print(f" • X_test: {X_test.shape}, y_test: {y_test.shape}")

    # Variables derivadas esperadas
    derived_vars = [v for v in ['FL_SCORE', 'PORTFOLIO_DIVERSITY', 'TRADER_SCORE'] if v in

    if derived_vars:
        print("\nVariables derivadas encontradas:", derived_vars)
        # Reordenar columnas para poner las derivadas al principio
        X_train = X_train[derived_vars + [c for c in X_train.columns if c not in derived_
        X_val = X_val[derived_vars + [c for c in X_val.columns if c not in derived_vars
        X_test = X_test[derived_vars + [c for c in X_test.columns if c not in derived_va
        print(" Reordenadas variables derivadas en los DataFrames")
    else:
        print("\n No se encontraron variables derivadas en X_train.")

except Exception as e:
    print(f"\nOcurrió un error inesperado al cargar los datos: {e}")

```

→ Mounted at /content/drive

Módulo de configuración cargado y estructura de carpetas asegurada.

Drive montado y configuración de rutas cargada correctamente.

Cargando datos FINALES desde: /content/drive/MyDrive/TFG/ML/Calculo-Riesgo/d

Datos cargados correctamente desde formato Parquet.

Shapes tras cargar splits:

- X_train: (1976, 92), y_train: (1976,)
- X_val: (424, 92), y_val: (424,)
- X_test: (424, 92), y_test: (424,)

Variables derivadas encontradas: ['FL_SCORE', 'PORTFOLIO_DIVERSITY', 'TRADER_SCORE']
Reordenadas variables derivadas en los DataFrames

▼ 2. Creación y aplicación del escalador a los datos

Instancia un StandardScaler, lo ajusta únicamente con los datos de entrenamiento y transforma los conjuntos de entrenamiento, validación y prueba; finalmente convierte los arrays escalados de vuelta a DataFrames.

```
# Crear el objeto escalador
scaler = StandardScaler()

# Ajustar el escalador SOLO con los datos de entrenamiento y transformarlos.
X_train_scaled_np = scaler.fit_transform(X_train)

# Aplicar la MISMA transformación (sin volver a ajustar) a los conjuntos de validación y
X_val_scaled_np = scaler.transform(X_val)
X_test_scaled_np = scaler.transform(X_test)

# Convertir los arrays de NumPy de vuelta a DataFrames de Pandas.
X_train_trans = pd.DataFrame(X_train_scaled_np, columns=X_train.columns)
X_val_trans = pd.DataFrame(X_val_scaled_np, columns=X_val.columns)
X_test_trans = pd.DataFrame(X_test_scaled_np, columns=X_test.columns)

print(" Datos transformados y escalados correctamente.")
print(f" • Variable 'X_train_trans' creada con shape: {X_train_trans.shape}")
print(f" • Variable 'X_val_trans' creada con shape: {X_val_trans.shape}")
```

→ Datos transformados y escalados correctamente.
• Variable 'X_train_trans' creada con shape: (1976, 92)
• Variable 'X_val_trans' creada con shape: (424, 92)

▼ 3. Selección de características univariada con ANOVA y evaluación básica

Aplica SelectKBest con función ANOVA (f_classif) para quedarse con las 20 mejores variables, entrena un LogisticRegression con y sin selección, y muestra los informes de clasificación en validación.

```

# Selección univariada ANOVA
k = 20
selector = SelectKBest(score_func=f_classif, k=k)
selector.fit(X_train_trans, y_train)

selected_f = X_train_trans.columns[selector.get_support()].tolist()
print(" Top-k ANOVA (f_classif):", selected_f)

# Entrenar Logistic con y sin selección
# Sin selección
log_basic = LogisticRegression(max_iter=1000, class_weight='balanced', random_state=42)
log_basic.fit(X_train_trans, y_train)
pred_val_basic = log_basic.predict(X_val_trans)

# Con selección
X_train_sel = selector.transform(X_train_trans)
X_val_sel = selector.transform(X_val_trans)
log_sel = LogisticRegression(max_iter=1000, class_weight='balanced', random_state=42)
log_sel.fit(X_train_sel, y_train)
pred_val_sel = log_sel.predict(X_val_sel)

print(" Métricas - Logistic sin selección (validación):")
print(classification_report(y_val, pred_val_basic))

print(" Métricas - Logistic con SelectKBest (validación):")
print(classification_report(y_val, pred_val_sel))

→ Top-k ANOVA (f_classif): ['TRADER_SCORE', 'B35', 'B26', 'C22_4', 'F30_3', 'F30_6', '...
/usr/local/lib/python3.11/dist-packages/sklearn/feature_selection/_univariate_selection.py:116: UserWarning: Features %s are constant." % constant_features_idx
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
/usr/local/lib/python3.11/dist-packages/sklearn/feature_selection/_univariate_selection.py:116: UserWarning: Features %s are constant." % constant_features_idx
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
f = msb / msw

Métricas - Logistic sin selección (validación):
      precision    recall  f1-score   support
          1.0      0.28      0.47      0.35       34
          2.0      0.39      0.41      0.40      116
          3.0      0.69      0.46      0.55      233
          4.0      0.28      0.61      0.39       41

      accuracy                           0.46      424
     macro avg      0.41      0.49      0.42      424
  weighted avg      0.54      0.46      0.48      424

Métricas - Logistic con SelectKBest (validación):
      precision    recall  f1-score   support
          1.0      0.34      0.50      0.40       34
          2.0      0.38      0.41      0.40      116
          3.0      0.73      0.35      0.47      233
          4.0      0.21      0.68      0.32       41

      accuracy                           0.41      424
     macro avg      0.41      0.49      0.40      424
  weighted avg      0.55      0.41      0.43      424

```

✓ 4. Comparación de selección ANOVA vs. InfoMutua

Vuelve a calcular la selección ANOVA, añade SelectKBest con InfoMutua, e identifica solapamientos y diferencias entre ambas listas de features.

```
# ANOVA (ya calculado como selected_f), repetimos para asegurar
selector_f = SelectKBest(score_func=f_classif, k=k)
selector_f.fit(X_train_trans, y_train)
selected_f = X_train_trans.columns[selector_f.get_support()].tolist()

# Mutual Information
selector_mi = SelectKBest(score_func=mutual_info_classif, k=k)
selector_mi.fit(X_train_trans, y_train)
selected_mi = X_train_trans.columns[selector_mi.get_support()].tolist()

solapamiento = set(selected_f).intersection(selected_mi)
solo_anova = set(selected_f) - solapamiento
solo_mi = set(selected_mi) - solapamiento

print(" Top-k con ANOVA (f_classif):", selected_f)
print(" Top-k con InfoMutua:", selected_mi)
print(f" Solapamiento ({len(solapamiento)}):", solapamiento)
print(f" Solo en ANOVA ({len(solo_anova)}):", solo_anova)
print(f" Solo en InfoMutua ({len(solo_mi)}):", solo_mi)

→ /usr/local/lib/python3.11/dist-packages/sklearn/feature_selection/_univariate_selection.py:200: UserWarning: Features %s are constant." % constant_features_idx
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
/usr/local/lib/python3.11/dist-packages/sklearn/feature_selection/_univariate_selection.py:200: UserWarning: Features %s are constant." % constant_features_idx
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
f = msb / msw
Top-k con ANOVA (f_classif): ['TRADER_SCORE', 'B35', 'B26', 'C22_4', 'F30_3', 'F30_6', 'D3', 'D21', 'D30', 'F30_4', 'F31_5', 'B35', 'C22_3', 'F31_2', 'F31_4', 'G30_3', 'G1', 'G2', 'G30_4', 'F31_8', 'F31_6', 'TRADER_SCORE', 'F30_7', 'B26', 'F31_11', 'D3', 'D21', 'C22_3', 'F30_5', 'B3', 'F30_4', 'D30', 'F31_12', 'F31_10', 'F31_9', 'F31_13', 'F31_14', 'F31_15', 'F31_16', 'F31_17', 'F31_18', 'F31_19', 'F31_20', 'F31_21', 'F31_22', 'F31_23', 'F31_24', 'F31_25', 'F31_26', 'F31_27', 'F31_28', 'F31_29', 'F31_30', 'F31_31', 'F31_32', 'F31_33', 'F31_34', 'F31_35', 'F31_36', 'F31_37', 'F31_38', 'F31_39', 'F31_40', 'F31_41', 'F31_42', 'F31_43', 'F31_44', 'F31_45', 'F31_46', 'F31_47', 'F31_48', 'F31_49', 'F31_50', 'F31_51', 'F31_52', 'F31_53', 'F31_54', 'F31_55', 'F31_56', 'F31_57', 'F31_58', 'F31_59', 'F31_60', 'F31_61', 'F31_62', 'F31_63', 'F31_64', 'F31_65', 'F31_66', 'F31_67', 'F31_68', 'F31_69', 'F31_70', 'F31_71', 'F31_72', 'F31_73', 'F31_74', 'F31_75', 'F31_76', 'F31_77', 'F31_78', 'F31_79', 'F31_80', 'F31_81', 'F31_82', 'F31_83', 'F31_84', 'F31_85', 'F31_86', 'F31_87', 'F31_88', 'F31_89', 'F31_90', 'F31_91', 'F31_92', 'F31_93', 'F31_94', 'F31_95', 'F31_96', 'F31_97', 'F31_98', 'F31_99', 'F31_100', 'F31_101', 'F31_102', 'F31_103', 'F31_104', 'F31_105', 'F31_106', 'F31_107', 'F31_108', 'F31_109', 'F31_110', 'F31_111', 'F31_112', 'F31_113', 'F31_114', 'F31_115', 'F31_116', 'F31_117', 'F31_118', 'F31_119', 'F31_120', 'F31_121', 'F31_122', 'F31_123', 'F31_124', 'F31_125', 'F31_126', 'F31_127', 'F31_128', 'F31_129', 'F31_130', 'F31_131', 'F31_132', 'F31_133', 'F31_134', 'F31_135', 'F31_136', 'F31_137', 'F31_138', 'F31_139', 'F31_140', 'F31_141', 'F31_142', 'F31_143', 'F31_144', 'F31_145', 'F31_146', 'F31_147', 'F31_148', 'F31_149', 'F31_150', 'F31_151', 'F31_152', 'F31_153', 'F31_154', 'F31_155', 'F31_156', 'F31_157', 'F31_158', 'F31_159', 'F31_160', 'F31_161', 'F31_162', 'F31_163', 'F31_164', 'F31_165', 'F31_166', 'F31_167', 'F31_168', 'F31_169', 'F31_170', 'F31_171', 'F31_172', 'F31_173', 'F31_174', 'F31_175', 'F31_176', 'F31_177', 'F31_178', 'F31_179', 'F31_180', 'F31_181', 'F31_182', 'F31_183', 'F31_184', 'F31_185', 'F31_186', 'F31_187', 'F31_188', 'F31_189', 'F31_190', 'F31_191', 'F31_192', 'F31_193', 'F31_194', 'F31_195', 'F31_196', 'F31_197', 'F31_198', 'F31_199', 'F31_200', 'F31_201', 'F31_202', 'F31_203', 'F31_204', 'F31_205', 'F31_206', 'F31_207', 'F31_208', 'F31_209', 'F31_210', 'F31_211', 'F31_212', 'F31_213', 'F31_214', 'F31_215', 'F31_216', 'F31_217', 'F31_218', 'F31_219', 'F31_220', 'F31_221', 'F31_222', 'F31_223', 'F31_224', 'F31_225', 'F31_226', 'F31_227', 'F31_228', 'F31_229', 'F31_230', 'F31_231', 'F31_232', 'F31_233', 'F31_234', 'F31_235', 'F31_236', 'F31_237', 'F31_238', 'F31_239', 'F31_240', 'F31_241', 'F31_242', 'F31_243', 'F31_244', 'F31_245', 'F31_246', 'F31_247', 'F31_248', 'F31_249', 'F31_250', 'F31_251', 'F31_252', 'F31_253', 'F31_254', 'F31_255', 'F31_256', 'F31_257', 'F31_258', 'F31_259', 'F31_260', 'F31_261', 'F31_262', 'F31_263', 'F31_264', 'F31_265', 'F31_266', 'F31_267', 'F31_268', 'F31_269', 'F31_270', 'F31_271', 'F31_272', 'F31_273', 'F31_274', 'F31_275', 'F31_276', 'F31_277', 'F31_278', 'F31_279', 'F31_280', 'F31_281', 'F31_282', 'F31_283', 'F31_284', 'F31_285', 'F31_286', 'F31_287', 'F31_288', 'F31_289', 'F31_290', 'F31_291', 'F31_292', 'F31_293', 'F31_294', 'F31_295', 'F31_296', 'F31_297', 'F31_298', 'F31_299', 'F31_300', 'F31_301', 'F31_302', 'F31_303', 'F31_304', 'F31_305', 'F31_306', 'F31_307', 'F31_308', 'F31_309', 'F31_310', 'F31_311', 'F31_312', 'F31_313', 'F31_314', 'F31_315', 'F31_316', 'F31_317', 'F31_318', 'F31_319', 'F31_320', 'F31_321', 'F31_322', 'F31_323', 'F31_324', 'F31_325', 'F31_326', 'F31_327', 'F31_328', 'F31_329', 'F31_330', 'F31_331', 'F31_332', 'F31_333', 'F31_334', 'F31_335', 'F31_336', 'F31_337', 'F31_338', 'F31_339', 'F31_340', 'F31_341', 'F31_342', 'F31_343', 'F31_344', 'F31_345', 'F31_346', 'F31_347', 'F31_348', 'F31_349', 'F31_350', 'F31_351', 'F31_352', 'F31_353', 'F31_354', 'F31_355', 'F31_356', 'F31_357', 'F31_358', 'F31_359', 'F31_360', 'F31_361', 'F31_362', 'F31_363', 'F31_364', 'F31_365', 'F31_366', 'F31_367', 'F31_368', 'F31_369', 'F31_370', 'F31_371', 'F31_372', 'F31_373', 'F31_374', 'F31_375', 'F31_376', 'F31_377', 'F31_378', 'F31_379', 'F31_380', 'F31_381', 'F31_382', 'F31_383', 'F31_384', 'F31_385', 'F31_386', 'F31_387', 'F31_388', 'F31_389', 'F31_390', 'F31_391', 'F31_392', 'F31_393', 'F31_394', 'F31_395', 'F31_396', 'F31_397', 'F31_398', 'F31_399', 'F31_400', 'F31_401', 'F31_402', 'F31_403', 'F31_404', 'F31_405', 'F31_406', 'F31_407', 'F31_408', 'F31_409', 'F31_410', 'F31_411', 'F31_412', 'F31_413', 'F31_414', 'F31_415', 'F31_416', 'F31_417', 'F31_418', 'F31_419', 'F31_420', 'F31_421', 'F31_422', 'F31_423', 'F31_424', 'F31_425', 'F31_426', 'F31_427', 'F31_428', 'F31_429', 'F31_430', 'F31_431', 'F31_432', 'F31_433', 'F31_434', 'F31_435', 'F31_436', 'F31_437', 'F31_438', 'F31_439', 'F31_440', 'F31_441', 'F31_442', 'F31_443', 'F31_444', 'F31_445', 'F31_446', 'F31_447', 'F31_448', 'F31_449', 'F31_450', 'F31_451', 'F31_452', 'F31_453', 'F31_454', 'F31_455', 'F31_456', 'F31_457', 'F31_458', 'F31_459', 'F31_460', 'F31_461', 'F31_462', 'F31_463', 'F31_464', 'F31_465', 'F31_466', 'F31_467', 'F31_468', 'F31_469', 'F31_470', 'F31_471', 'F31_472', 'F31_473', 'F31_474', 'F31_475', 'F31_476', 'F31_477', 'F31_478', 'F31_479', 'F31_480', 'F31_481', 'F31_482', 'F31_483', 'F31_484', 'F31_485', 'F31_486', 'F31_487', 'F31_488', 'F31_489', 'F31_490', 'F31_491', 'F31_492', 'F31_493', 'F31_494', 'F31_495', 'F31_496', 'F31_497', 'F31_498', 'F31_499', 'F31_500', 'F31_501', 'F31_502', 'F31_503', 'F31_504', 'F31_505', 'F31_506', 'F31_507', 'F31_508', 'F31_509', 'F31_510', 'F31_511', 'F31_512', 'F31_513', 'F31_514', 'F31_515', 'F31_516', 'F31_517', 'F31_518', 'F31_519', 'F31_520', 'F31_521', 'F31_522', 'F31_523', 'F31_524', 'F31_525', 'F31_526', 'F31_527', 'F31_528', 'F31_529', 'F31_530', 'F31_531', 'F31_532', 'F31_533', 'F31_534', 'F31_535', 'F31_536', 'F31_537', 'F31_538', 'F31_539', 'F31_540', 'F31_541', 'F31_542', 'F31_543', 'F31_544', 'F31_545', 'F31_546', 'F31_547', 'F31_548', 'F31_549', 'F31_550', 'F31_551', 'F31_552', 'F31_553', 'F31_554', 'F31_555', 'F31_556', 'F31_557', 'F31_558', 'F31_559', 'F31_560', 'F31_561', 'F31_562', 'F31_563', 'F31_564', 'F31_565', 'F31_566', 'F31_567', 'F31_568', 'F31_569', 'F31_570', 'F31_571', 'F31_572', 'F31_573', 'F31_574', 'F31_575', 'F31_576', 'F31_577', 'F31_578', 'F31_579', 'F31_580', 'F31_581', 'F31_582', 'F31_583', 'F31_584', 'F31_585', 'F31_586', 'F31_587', 'F31_588', 'F31_589', 'F31_590', 'F31_591', 'F31_592', 'F31_593', 'F31_594', 'F31_595', 'F31_596', 'F31_597', 'F31_598', 'F31_599', 'F31_600', 'F31_601', 'F31_602', 'F31_603', 'F31_604', 'F31_605', 'F31_606', 'F31_607', 'F31_608', 'F31_609', 'F31_610', 'F31_611', 'F31_612', 'F31_613', 'F31_614', 'F31_615', 'F31_616', 'F31_617', 'F31_618', 'F31_619', 'F31_620', 'F31_621', 'F31_622', 'F31_623', 'F31_624', 'F31_625', 'F31_626', 'F31_627', 'F31_628', 'F31_629', 'F31_630', 'F31_631', 'F31_632', 'F31_633', 'F31_634', 'F31_635', 'F31_636', 'F31_637', 'F31_638', 'F31_639', 'F31_640', 'F31_641', 'F31_642', 'F31_643', 'F31_644', 'F31_645', 'F31_646', 'F31_647', 'F31_648', 'F31_649', 'F31_650', 'F31_651', 'F31_652', 'F31_653', 'F31_654', 'F31_655', 'F31_656', 'F31_657', 'F31_658', 'F31_659', 'F31_660', 'F31_661', 'F31_662', 'F31_663', 'F31_664', 'F31_665', 'F31_666', 'F31_667', 'F31_668', 'F31_669', 'F31_670', 'F31_671', 'F31_672', 'F31_673', 'F31_674', 'F31_675', 'F31_676', 'F31_677', 'F31_678', 'F31_679', 'F31_680', 'F31_681', 'F31_682', 'F31_683', 'F31_684', 'F31_685', 'F31_686', 'F31_687', 'F31_688', 'F31_689', 'F31_690', 'F31_691', 'F31_692', 'F31_693', 'F31_694', 'F31_695', 'F31_696', 'F31_697', 'F31_698', 'F31_699', 'F31_700', 'F31_701', 'F31_702', 'F31_703', 'F31_704', 'F31_705', 'F31_706', 'F31_707', 'F31_708', 'F31_709', 'F31_710', 'F31_711', 'F31_712', 'F31_713', 'F31_714', 'F31_715', 'F31_716', 'F31_717', 'F31_718', 'F31_719', 'F31_720', 'F31_721', 'F31_722', 'F31_723', 'F31_724', 'F31_725', 'F31_726', 'F31_727', 'F31_728', 'F31_729', 'F31_730', 'F31_731', 'F31_732', 'F31_733', 'F31_734', 'F31_735', 'F31_736', 'F31_737', 'F31_738', 'F31_739', 'F31_740', 'F31_741', 'F31_742', 'F31_743', 'F31_744', 'F31_745', 'F31_746', 'F31_747', 'F31_748', 'F31_749', 'F31_750', 'F31_751', 'F31_752', 'F31_753', 'F31_754', 'F31_755', 'F31_756', 'F31_757', 'F31_758', 'F31_759', 'F31_760', 'F31_761', 'F31_762', 'F31_763', 'F31_764', 'F31_765', 'F31_766', 'F31_767', 'F31_768', 'F31_769', 'F31_770', 'F31_771', 'F31_772', 'F31_773', 'F31_774', 'F31_775', 'F31_776', 'F31_777', 'F31_778', 'F31_779', 'F31_780', 'F31_781', 'F31_782', 'F31_783', 'F31_784', 'F31_785', 'F31_786', 'F31_787', 'F31_788', 'F31_789', 'F31_790', 'F31_791', 'F31_792', 'F31_793', 'F31_794', 'F31_795', 'F31_796', 'F31_797', 'F31_798', 'F31_799', 'F31_800', 'F31_801', 'F31_802', 'F31_803', 'F31_804', 'F31_805', 'F31_806', 'F31_807', 'F31_808', 'F31_809', 'F31_810', 'F31_811', 'F31_812', 'F31_813', 'F31_814', 'F31_815', 'F31_816', 'F31_817', 'F31_818', 'F31_819', 'F31_820', 'F31_821', 'F31_822', 'F31_823', 'F31_824', 'F31_825', 'F31_826', 'F31_827', 'F31_828', 'F31_829', 'F31_830', 'F31_831', 'F31_832', 'F31_833', 'F31_834', 'F31_835', 'F31_836', 'F31_837', 'F31_838', 'F31_839', 'F31_840', 'F31_841', 'F31_842', 'F31_843', 'F31_844', 'F31_845', 'F31_846', 'F31_847', 'F31_848', 'F31_849', 'F31_850', 'F31_851', 'F31_852', 'F31_853', 'F31_854', 'F31_855', 'F31_856', 'F31_857', 'F31_858', 'F31_859', 'F31_860', 'F31_861', 'F31_862', 'F31_863', 'F31_864', 'F31_865', 'F31_866', 'F31_867', 'F31_868', 'F31_869', 'F31_870', 'F31_871', 'F31_872', 'F31_873', 'F31_874', 'F31_875', 'F31_876', 'F31_877', 'F31_878', 'F31_879', 'F31_880', 'F31_881', 'F31_882', 'F31_883', 'F31_884', 'F31_885', 'F31_886', 'F31_887', 'F31_888', 'F31_889', 'F31_890', 'F31_891', 'F31_892', 'F31_893', 'F31_894', 'F31_895', 'F31_896', 'F31_897', 'F31_898', 'F31_899', 'F31_900', 'F31_901', 'F31_902', 'F31_903', 'F31_904', 'F31_905', 'F31_906', 'F31_907', 'F31_908', 'F31_909', 'F31_910', 'F31_911', 'F31_912', 'F31_913', 'F31_914', 'F31_915', 'F31_916', 'F31_917', 'F31_918', 'F31_919', 'F31_920', 'F31_921', 'F31_922', 'F31_923', 'F31_924', 'F31_925', 'F31_926', 'F31_927', 'F31_928', 'F31_929', 'F31_930', 'F31_931', 'F31_932', 'F31_933', 'F31_934', 'F31_935', 'F31_936', 'F31_937', 'F31_938', 'F31_939', 'F31_940', 'F31_941', 'F31_942', 'F31_943', 'F31_944', 'F31_945', 'F31_946', 'F31_947', 'F31_948', 'F31_949', 'F31_950', 'F31_951', 'F31_952', 'F31_953', 'F31_954', 'F31_955', 'F31_956', 'F31_957', 'F31_958', 'F31_959', 'F31_960', 'F31_961', 'F31_962', 'F31_963', 'F31_964', 'F31_965', 'F31_966', 'F31_967', 'F31_968', 'F31_969', 'F31_970', 'F31_971', 'F31_972', 'F31_973', 'F31_974', 'F31_975', 'F31_976', 'F31_977', 'F31_978', 'F31_979', 'F31_980', 'F31_981', 'F31_982', 'F31_983', 'F31_984', 'F31_985', 'F31_986', 'F31_987', 'F31_988', 'F31_989', 'F31_990', 'F31_991', 'F31_992', 'F31_993', 'F31_994', 'F31_995', 'F31_996', 'F31_997', 'F31_998', 'F31_999', 'F31_1000', 'F31_1001', 'F31_1002', 'F31_1003', 'F31_1004', 'F31_1005', 'F31_1006', 'F31_1007', 'F31_1008', 'F31_1009', 'F31_1010', 'F31_1011', 'F31_1012', 'F31_1013', 'F31_1014', 'F31_1015', 'F31_1016', 'F31_1017', 'F31_1018', 'F31_1019', 'F31_1020', 'F31_1021', 'F31_1022', 'F31_1023', 'F31_1024', 'F31_1025', 'F31_1026', 'F31_1027', 'F31_1028', 'F31_1029', 'F31_1030', 'F31_1031', 'F31_1032', 'F31_1033', 'F31_1034', 'F31_1035', 'F31_1036', 'F31_1037', 'F31_1038', 'F31_1039', 'F31_1040', 'F31_1041', 'F31_1042', 'F31_1043', 'F31_1044', 'F31_1045', 'F31_1046', 'F31_1047', 'F31_1048', 'F31_1049', 'F31_1050', 'F31_1051', 'F31_1052', 'F31_1053', 'F31_1054', 'F31_1055', 'F31_1056', 'F31_1057', 'F31_1058', 'F31_1059', 'F31_1060', 'F31_1061', 'F31_1062', 'F31_1063', 'F31_1064', 'F31_1065', 'F31_1066', 'F31_1067', 'F31_1068', 'F31_1069', 'F31_1070', 'F31_1071', 'F31_1072', 'F31_1073', 'F31_1074', 'F31_1075', 'F31_1076', 'F31_1077', 'F31_1078', 'F31_1079', 'F31_1080', 'F31_1081', 'F31_1082', 'F31_1083', 'F31_1084', 'F31_1085', 'F31_1086', 'F31_1087', 'F31_1088', 'F31_1089', 'F31_1090', 'F31_1091', 'F31_1092', 'F31_1093', 'F31_1094', 'F31_1095', 'F31_1096', 'F31_1097', 'F31_1098', 'F31_1099', 'F31_1100', 'F31_1101', 'F31_1102', 'F31_1103', 'F31_1104', 'F31_1105', 'F31_1106', 'F31_1107', 'F31_1108', 'F31_1109', 'F31_1110', 'F31_1111', 'F31_1112', 'F31_1113', 'F31_1114', 'F31_1115', 'F31_1116', 'F31_1117', 'F31_1118', 'F31_1119', 'F31_1120', 'F31_1121', 'F31_1122', 'F31_1123', 'F31_1124', 'F31_1125', 'F31_1126', 'F31_1127', 'F31_1128', 'F31_1129', 'F31_1130', 'F31_1131', 'F31_1132', 'F31_1133', 'F31_1134', 'F31_1135', 'F31_1136', 'F31_1137', 'F31_1138', 'F31_1139', 'F31_1140', 'F31_1141', 'F31_1142', 'F31_1143', 'F31_1144', 'F31_1145', 'F31_1146', 'F31_1147', 'F31_1148', 'F31_1149', 'F31_1150', 'F31_1151', 'F31_1152', 'F31_1153', 'F31_1154', 'F31_1155', 'F31_1156
```

→ Número de features seleccionadas por RF (umbral=mediana): 46
Features seleccionadas por RF: ['FL_SCORE', 'PORTFOLIO_DIVERSITY', 'TRADER_SCORE', '

6. Selección recursiva de características con RFECV y visualización

Configura y ajusta RFECV empleando un LogisticRegression y validación cruzada estratificada, obtiene el número óptimo de features, lista seleccionada y dibuja F1_macro vs. número de features.

```
estimator_log = LogisticRegression(max_iter=1000, class_weight='balanced', random_state=42)

rfecv = RFECV(
    estimator=estimator_log,
    step=1,
    cv=StratifiedKFold(n_splits=5, shuffle=True, random_state=42),
    scoring='f1_macro',
    min_features_to_select=5,
    n_jobs=-1
)
rfecv.fit(X_train_trans, y_train)

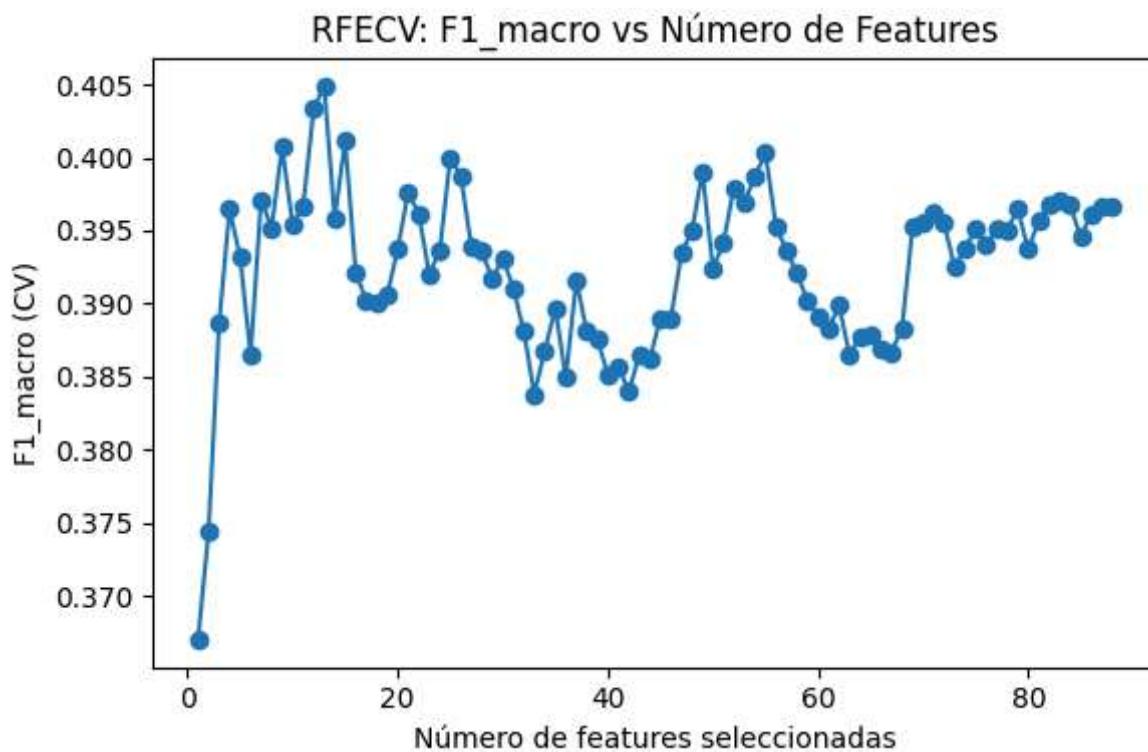
optimal_n = rfecv.n_features_
selected_rfecv = X_train_trans.columns[rfecv.support_].tolist()

print(f" Número óptimo de features según RFECV: {optimal_n}")
print(" Features seleccionadas por RFECV:", selected_rfecv)

# Gráfico de F1_macro vs número de features
plt.figure(figsize=(6,4))
plt.title("RFECV: F1_macro vs Número de Features")
plt.xlabel("Número de features seleccionadas")
plt.ylabel("F1_macro (CV)")

# Se calculan manualmente:
scores = rfecv.cv_results_['mean_test_score']
plt.plot(range(1, len(scores) + 1), scores, marker='o')
plt.tight_layout()
plt.show()
```

→ Número óptimo de features según RFECV: 17
 Features seleccionadas por RFECV: ['TRADER_SCORE', 'A1', 'B2_2', 'B11', 'B26', 'C24']



7. Cálculo de frecuencia de selección de features y filtrado por umbral

Combina las listas de features obtenidas por ANOVA, InfoMutua, Random Forest y RFECV, cuenta cuántas veces aparece cada feature, crea un DataFrame ordenado y selecciona aquellas con frecuencia ≥ 2 .

```
listas = {
    'ANOVA': selected_f,
    'InfoMutua': selected_mi,
    'RF_importance': sf_features,
    'RFECV': selected_rfecv
}

all_selected = selected_f + selected_mi + sf_features + selected_rfecv
freq_counter = Counter(all_selected)

df_freq = pd.DataFrame({
    'feature': list(freq_counter.keys()),
    'frequency': list(freq_counter.values())
}).sort_values('frequency', ascending=False)

print(" Frecuencia de selección de cada feature:")
display(df_freq)

threshold = 2
```

```
final_features_2 = df_freq.loc[df_freq['frequency'] >= threshold, 'feature'].tolist()
print(f" Features finales (frecuencia >= {threshold}):", final_features_2)
```

→ Frecuencia de selección de cada feature:

	feature	frequency
16	G2	4
19	S_Age	4
0	TRADER_SCORE	3
1	B35	3
11	F31_4	3
24	D30	3
18	G30_4	3
17	G30_3	3
15	G1	3
12	F31_5	3
2	B26	3
10	F31_2	3
25	F30_4	3
23	D21	3
7	F30_8	2
13	F31_6	2
6	F30_7	2
4	F30_3	2
3	C22_4	2
8	F30_9	2
21	C22_3	2
20	B3	2
22	D3	2
35	C24	2
48	G30_2	2
55	B4_log	2
32	B11	2
5	F30_6	1
27	F31_11	1
26	F30_5	1
14	F31_8	1
9	F30_11	1

8.3 Evaluación del modelo con features de consenso (umbral ≥ 3)

Filtrá las variables cuya frecuencia de selección sea al menos 3, ajusta un LogisticRegression sobre ese subconjunto y presenta métricas de validación y test.

```

30 B30
Filtrá las variables cuya frecuencia de selección sea al menos 3, ajusta un LogisticRegression
sobre ese subconjunto y presenta métricas de validación y test.

31 B31
32 B32
33 B33
34 B34
35 B35
36 B36
37 B37
38 B38
39 B39
40 B40
41 B41
42 B42
43 B43
44 B44
45 B45
46 B46
47 B47
48 B48
49 B49
50 B50
51 B51
52 B52
53 B53
54 B54
55 B55
56 B56
57 B57
58 B58
59 B59
60 B60
61 B61
62 B62
63 B63
64 B64
65 B65
66 B66
67 B67
68 B68
69 B69
70 B70
71 B71
72 B72
73 B73
74 B74
75 B75
76 B76
77 B77
78 B78
79 B79
80 B80
81 B81
82 B82
83 B83
84 B84
85 B85
86 B86
87 B87
88 B88
89 B89
90 B90
91 B91
92 B92
93 B93
94 B94
95 B95
96 B96
97 B97
98 B98
99 B99
100 B100
101 B101
102 B102
103 B103
104 B104
105 B105
106 B106
107 B107
108 B108
109 B109
110 B110
111 B111
112 B112
113 B113
114 B114
115 B115
116 B116
117 B117
118 B118
119 B119
120 B120
121 B121
122 B122
123 B123
124 B124
125 B125
126 B126
127 B127
128 B128
129 B129
130 B130
131 B131
132 B132
133 B133
134 B134
135 B135
136 B136
137 B137
138 B138
139 B139
140 B140
141 B141
142 B142
143 B143
144 B144
145 B145
146 B146
147 B147
148 B148
149 B149
150 B150
151 B151
152 B152
153 B153
154 B154
155 B155
156 B156
157 B157
158 B158
159 B159
160 B160
161 B161
162 B162
163 B163
164 B164
165 B165
166 B166
167 B167
168 B168
169 B169
170 B170
171 B171
172 B172
173 B173
174 B174
175 B175
176 B176
177 B177
178 B178
179 B179
180 B180
181 B181
182 B182
183 B183
184 B184
185 B185
186 B186
187 B187
188 B188
189 B189
190 B190
191 B191
192 B192
193 B193
194 B194
195 B195
196 B196
197 B197
198 B198
199 B199
200 B200
201 B201
202 B202
203 B203
204 B204
205 B205
206 B206
207 B207
208 B208
209 B209
210 B210
211 B211
212 B212
213 B213
214 B214
215 B215
216 B216
217 B217
218 B218
219 B219
220 B220
221 B221
222 B222
223 B223
224 B224
225 B225
226 B226
227 B227
228 B228
229 B229
230 B230
231 B231
232 B232
233 B233
234 B234
235 B235
236 B236
237 B237
238 B238
239 B239
240 B240
241 B241
242 B242
243 B243
244 B244
245 B245
246 B246
247 B247
248 B248
249 B249
250 B250
251 B251
252 B252
253 B253
254 B254
255 B255
256 B256
257 B257
258 B258
259 B259
260 B260
261 B261
262 B262
263 B263
264 B264
265 B265
266 B266
267 B267
268 B268
269 B269
270 B270
271 B271
272 B272
273 B273
274 B274
275 B275
276 B276
277 B277
278 B278
279 B279
280 B280
281 B281
282 B282
283 B283
284 B284
285 B285
286 B286
287 B287
288 B288
289 B289
290 B290
291 B291
292 B292
293 B293
294 B294
295 B295
296 B296
297 B297
298 B298
299 B299
300 B300
301 B301
302 B302
303 B303
304 B304
305 B305
306 B306
307 B307
308 B308
309 B309
310 B310
311 B311
312 B312
313 B313
314 B314
315 B315
316 B316
317 B317
318 B318
319 B319
320 B320
321 B321
322 B322
323 B323
324 B324
325 B325
326 B326
327 B327
328 B328
329 B329
330 B330
331 B331
332 B332
333 B333
334 B334
335 B335
336 B336
337 B337
338 B338
339 B339
340 B340
341 B341
342 B342
343 B343
344 B344
345 B345
346 B346
347 B347
348 B348
349 B349
350 B350
351 B351
352 B352
353 B353
354 B354
355 B355
356 B356
357 B357
358 B358
359 B359
360 B360
361 B361
362 B362
363 B363
364 B364
365 B365
366 B366
367 B367
368 B368
369 B369
370 B370
371 B371
372 B372
373 B373
374 B374
375 B375
376 B376
377 B377
378 B378
379 B379
380 B380
381 B381
382 B382
383 B383
384 B384
385 B385
386 B386
387 B387
388 B388
389 B389
390 B390
391 B391
392 B392
393 B393
394 B394
395 B395
396 B396
397 B397
398 B398
399 B399
400 B400
401 B401
402 B402
403 B403
404 B404
405 B405
406 B406
407 B407
408 B408
409 B409
410 B410
411 B411
412 B412
413 B413
414 B414
415 B415
416 B416
417 B417
418 B418
419 B419
420 B420
421 B421
422 B422
423 B423
424 B424
425 B425
426 B426
427 B427
428 B428
429 B429
430 B430
431 B431
432 B432
433 B433
434 B434
435 B435
436 B436
437 B437
438 B438
439 B439
440 B440
441 B441
442 B442
443 B443
444 B444
445 B445
446 B446
447 B447
448 B448
449 B449
450 B450
451 B451
452 B452
453 B453
454 B454
455 B455
456 B456
457 B457
458 B458
459 B459
460 B460
461 B461
462 B462
463 B463
464 B464
465 B465
466 B466
467 B467
468 B468
469 B469
470 B470
471 B471
472 B472
473 B473
474 B474
475 B475
476 B476
477 B477
478 B478
479 B479
480 B480
481 B481
482 B482
483 B483
484 B484
485 B485
486 B486
487 B487
488 B488
489 B489
490 B490
491 B491
492 B492
493 B493
494 B494
495 B495
496 B496
497 B497
498 B498
499 B499
500 B500
501 B501
502 B502
503 B503
504 B504
505 B505
506 B506
507 B507
508 B508
509 B509
510 B510
511 B511
512 B512
513 B513
514 B514
515 B515
516 B516
517 B517
518 B518
519 B519
520 B520
521 B521
522 B522
523 B523
524 B524
525 B525
526 B526
527 B527
528 B528
529 B529
530 B530
531 B531
532 B532
533 B533
534 B534
535 B535
536 B536
537 B537
538 B538
539 B539
540 B540
541 B541
542 B542
543 B543
544 B544
545 B545
546 B546
547 B547
548 B548
549 B549
550 B550
551 B551
552 B552
553 B553
554 B554
555 B555
556 B556
557 B557
558 B558
559 B559
560 B560
561 B561
562 B562
563 B563
564 B564
565 B565
566 B566
567 B567
568 B568
569 B569
570 B570
571 B571
572 B572
573 B573
574 B574
575 B575
576 B576
577 B577
578 B578
579 B579
580 B580
581 B581
582 B582
583 B583
584 B584
585 B585
586 B586
587 B587
588 B588
589 B589
590 B590
591 B591
592 B592
593 B593
594 B594
595 B595
596 B596
597 B597
598 B598
599 B599
600 B600
601 B601
602 B602
603 B603
604 B604
605 B605
606 B606
607 B607
608 B608
609 B609
610 B610
611 B611
612 B612
613 B613
614 B614
615 B615
616 B616
617 B617
618 B618
619 B619
620 B620
621 B621
622 B622
623 B623
624 B624
625 B625
626 B626
627 B627
628 B628
629 B629
630 B630
631 B631
632 B632
633 B633
634 B634
635 B635
636 B636
637 B637
638 B638
639 B639
640 B640
641 B641
642 B642
643 B643
644 B644
645 B645
646 B646
647 B647
648 B648
649 B649
650 B650
651 B651
652 B652
653 B653
654 B654
655 B655
656 B656
657 B657
658 B658
659 B659
660 B660
661 B661
662 B662
663 B663
664 B664
665 B665
666 B666
667 B667
668 B668
669 B669
670 B670
671 B671
672 B672
673 B673
674 B674
675 B675
676 B676
677 B677
678 B678
679 B679
680 B680
681 B681
682 B682
683 B683
684 B684
685 B685
686 B686
687 B687
688 B688
689 B689
690 B690
691 B691
692 B692
693 B693
694 B694
695 B695
696 B696
697 B697
698 B698
699 B699
700 B700
701 B701
702 B702
703 B703
704 B704
705 B705
706 B706
707 B707
708 B708
709 B709
710 B710
711 B711
712 B712
713 B713
714 B714
715 B715
716 B716
717 B717
718 B718
719 B719
720 B720
721 B721
722 B722
723 B723
724 B724
725 B725
726 B726
727 B727
728 B728
729 B729
730 B730
731 B731
732 B732
733 B733
734 B734
735 B735
736 B736
737 B737
738 B738
739 B739
740 B740
741 B741
742 B742
743 B743
744 B744
745 B745
746 B746
747 B747
748 B748
749 B749
750 B750
751 B751
752 B752
753 B753
754 B754
755 B755
756 B756
757 B757
758 B758
759 B759
760 B760
761 B761
762 B762
763 B763
764 B764
765 B765
766 B766
767 B767
768 B768
769 B769
770 B770
771 B771
772 B772
773 B773
774 B774
775 B775
776 B776
777 B777
778 B778
779 B779
780 B780
781 B781
782 B782
783 B783
784 B784
785 B785
786 B786
787 B787
788 B788
789 B789
790 B790
791 B791
792 B792
793 B793
794 B794
795 B795
796 B796
797 B797
798 B798
799 B799
800 B800
801 B801
802 B802
803 B803
804 B804
805 B805
806 B806
807 B807
808 B808
809 B809
810 B810
811 B811
812 B812
813 B813
814 B814
815 B815
816 B816
817 B817
818 B818
819 B819
820 B820
821 B821
822 B822
823 B823
824 B824
825 B825
826 B826
827 B827
828 B828
829 B829
830 B830
831 B831
832 B832
833 B833
834 B834
835 B835
836 B836
837 B837
838 B838
839 B839
840 B840
841 B841
842 B842
843 B843
844 B844
845 B845
846 B846
847 B847
848 B848
849 B849
850 B850
851 B851
852 B852
853 B853
854 B854
855 B855
856 B856
857 B857
858 B858
859 B859
860 B860
861 B861
862 B862
863 B863
864 B864
865 B865
866 B866
867 B867
868 B868
869 B869
870 B870
871 B871
872 B872
873 B873
874 B874
875 B875
876 B876
877 B877
878 B878
```

macro avg	0.42	0.53	0.41	424
weighted avg	0.55	0.41	0.41	424

✓ 9. Evaluación del modelo con top-20 features de RFECV

Toma las primeras 20 variables ordenadas por ranking de RFECV, ajusta un LogisticRegression y muestra los informes de clasificación en validación y test.

```
# Seleccionar las 20 primeras de selected_rfecv
rfecv_top20 = selected_rfecv[:20]
print(" RFECV Top-20 features:", rfecv_top20)

X_train_rf20 = X_train_trans[rfecv_top20].copy()
X_val_rf20 = X_val_trans[rfecv_top20].copy()
X_test_rf20 = X_test_trans[rfecv_top20].copy()

print(" Shapes RFECV-Top-20:")
print(" • X_train_rf20:", X_train_rf20.shape)
print(" • X_val_rf20: ", X_val_rf20.shape)

pipe_rfecv20 = Pipeline([
    ('clf', LogisticRegression(max_iter=1000, class_weight='balanced', random_state=42))
])
pipe_rfecv20.fit(X_train_rf20, y_train)

print(" Métricas - RFECV-Top-20 (validación):")
print(classification_report(y_val, pipe_rfecv20.predict(X_val_rf20)))

print(" Métricas - RFECV-Top-20 (test):")
print(classification_report(y_test, pipe_rfecv20.predict(X_test_rf20)))
```

→ RFECV Top-20 features: ['TRADER_SCORE', 'A1', 'B2_2', 'B11', 'B26', 'C24', 'D21', 'E22', 'F23', 'G25', 'H27', 'I28', 'J29', 'K30', 'L31', 'M32', 'N33', 'O34', 'P35', 'Q36', 'R37', 'S38', 'T39', 'U30', 'V31', 'W32', 'X33', 'Y34', 'Z35']

Shapes RFECV-Top-20:

- X_train_rf20: (1976, 17)
- X_val_rf20: (424, 17)

Métricas - RFECV-Top-20 (validación):

	precision	recall	f1-score	support
1.0	0.27	0.53	0.36	34
2.0	0.33	0.33	0.33	116
3.0	0.70	0.44	0.54	233
4.0	0.25	0.59	0.35	41
accuracy			0.43	424
macro avg	0.39	0.47	0.39	424
weighted avg	0.52	0.43	0.45	424

Métricas - RFECV-Top-20 (test):

	precision	recall	f1-score	support
1.0	0.27	0.56	0.36	34
2.0	0.46	0.47	0.47	116

3.0	0.71	0.35	0.47	233
4.0	0.26	0.73	0.38	41
accuracy			0.44	424
macro avg	0.42	0.53	0.42	424
weighted avg	0.56	0.44	0.45	424

▼ 10. Evaluación con features en la intersección RF ∩ InfoMutua

Calcula la intersección entre las features seleccionadas por Random Forest y por InfoMutua, entrena un LogisticRegression y muestra métricas en validación y test.

```
rf_mi_intersect = list(set(sf_features).intersection(set(selected_mi)))
print(" RF n InfoMutua:", rf_mi_intersect)

X_train_rfmi = X_train_trans[rf_mi_intersect].copy()
X_val_rfmi = X_val_trans[rf_mi_intersect].copy()
X_test_rfmi = X_test_trans[rf_mi_intersect].copy()

print(" Shapes RFnMI:")
print(" • X_train_rfmi:", X_train_rfmi.shape)
print(" • X_val_rfmi: ", X_val_rfmi.shape)

pipe_rfmi = Pipeline([
    ('clf', LogisticRegression(max_iter=1000, class_weight='balanced', random_state=42))
])
pipe_rfmi.fit(X_train_rfmi, y_train)

print(" Métricas - RF n InfoMutua (validación):")
print(classification_report(y_val, pipe_rfmi.predict(X_val_rfmi)))

print(" Métricas - RF n InfoMutua (test):")
print(classification_report(y_test, pipe_rfmi.predict(X_test_rfmi)))
```

→ RF n InfoMutua: ['F31_5', 'B35', 'F30_4', 'G30_3', 'G2', 'G1', 'D30', 'B3', 'G30_4', Shapes RFnMI:

- X_train_rfmi: (1976, 13)
- X_val_rfmi: (424, 13)

Métricas - RF n InfoMutua (validación):

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

1.0	0.38	0.53	0.44	34
2.0	0.41	0.48	0.44	116
3.0	0.71	0.35	0.47	233
4.0	0.21	0.66	0.32	41

accuracy			0.43	424
macro avg	0.43	0.50	0.42	424
weighted avg	0.55	0.43	0.44	424

Métricas - RF n InfoMutua (test):

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

1.0	0.31	0.59	0.40	34
2.0	0.43	0.50	0.46	116
3.0	0.66	0.25	0.37	233
4.0	0.22	0.71	0.33	41
accuracy			0.39	424
macro avg	0.40	0.51	0.39	424
weighted avg	0.52	0.39	0.39	424

11. Rendimiento en CV para distintos tamaños de subset de RFECV

Define una función para evaluar por validación cruzada el LogisticRegression con los primeros k features de RFECV, calcula medias y desviaciones para varios k y presenta un gráfico de error.

```

rfecv_feats = selected_rfecv # lista de 30 features con ranking=1

def eval_k_features(feature_list, X, y, cv=5):
    model = LogisticRegression(max_iter=1000, class_weight='balanced', random_state=42)
    scores = cross_val_score(model, X[feature_list], y, cv=cv, scoring='f1_macro', n_jobs=-1)
    return scores.mean(), scores.std()

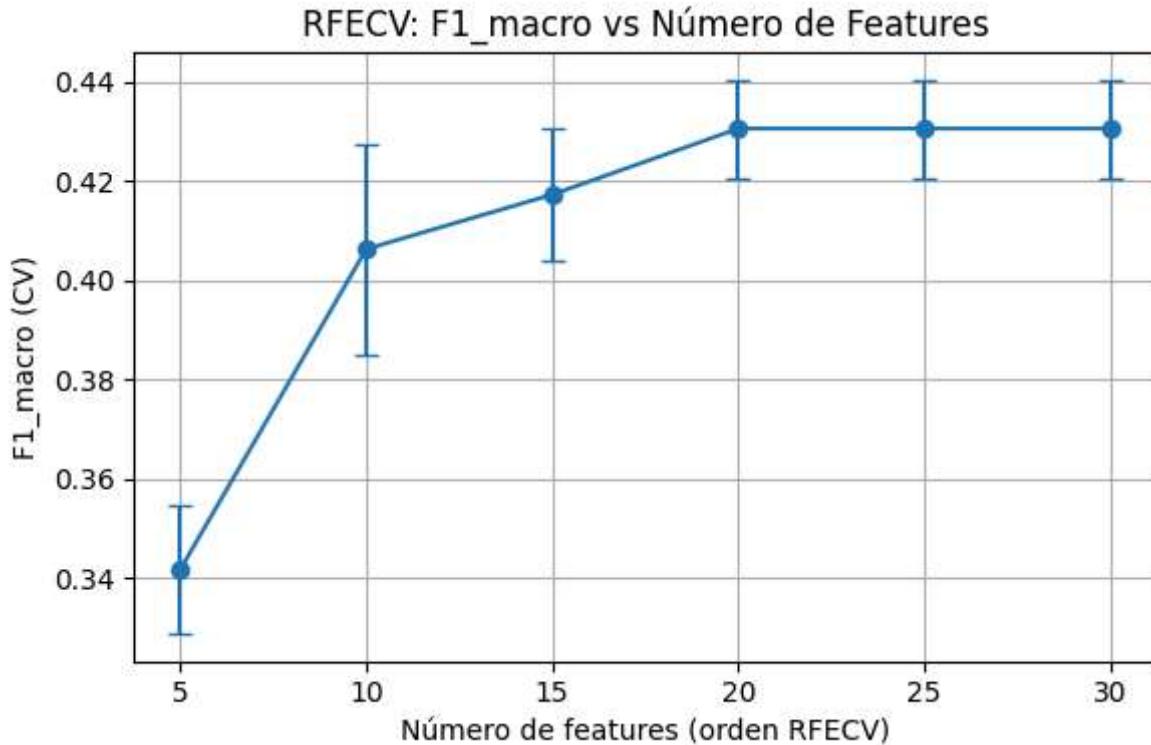
k_list = [5, 10, 15, 20, 25, 30]
means = []
stds = []

for k in k_list:
    feats_k = rfecv_feats[:k]
    mean_k, std_k = eval_k_features(feats_k, X_train_trans, y_train, cv=StratifiedKFold(5))
    means.append(mean_k)
    stds.append(std_k)
    print(f"k={k} → f1_macro (CV): {mean_k:.3f} ± {std_k:.3f}")

plt.figure(figsize=(6,4))
plt.errorbar(k_list, means, yerr=stds, fmt='-o', capsize=4)
plt.title("RFECV: F1_macro vs Número de Features")
plt.xlabel("Número de features (orden RFECV)")
plt.ylabel("F1_macro (CV)")
plt.grid(True)
plt.tight_layout()
plt.show()

```

→ k=5 → f1_macro (CV): 0.342 ± 0.013
 k=10 → f1_macro (CV): 0.406 ± 0.021
 k=15 → f1_macro (CV): 0.417 ± 0.013
 k=20 → f1_macro (CV): 0.431 ± 0.010
 k=25 → f1_macro (CV): 0.431 ± 0.010
 k=30 → f1_macro (CV): 0.431 ± 0.010



12. Análisis y evaluación de PCA para reducción de dimensionalidad

Ajusta un PCA que capture el 90 % de la varianza, transforma los datos, informa el número de componentes y las shapes resultantes, entrena un LogisticRegression sobre la representación PCA y muestra métricas.

```
pca = PCA(n_components=0.90, random_state=42)
pca.fit(X_train_trans)

print(f" Número de componentes para 90% varianza: {pca.n_components_}")

X_train_pca = pca.transform(X_train_trans)
X_val_pca = pca.transform(X_val_trans)
X_test_pca = pca.transform(X_test_trans)

print(" Shapes tras PCA:")
print("   • X_train_pca: ", X_train_pca.shape)
print("   • X_val_pca: ", X_val_pca.shape)
print("   • X_test_pca: ", X_test_pca.shape)

pipe_pca = Pipeline([
    ('clf', LogisticRegression(max_iter=1000, class_weight='balanced', random_state=42))
```

```
])
pipe_pca.fit(X_train_pca, y_train)

print(" Métricas - Logistic sobre PCA (validación):")
print(classification_report(y_val, pipe_pca.predict(X_val_pca)))

print(" Métricas finales - Logistic sobre PCA (test):")
print(classification_report(y_test, pipe_pca.predict(X_test_pca)))
```

→ Número de componentes para 90% varianza: 64

Shapes tras PCA:

- X_train_pca: (1976, 64)
- X_val_pca: (424, 64)
- X_test_pca: (424, 64)

Métricas - Logistic sobre PCA (validación):

	precision	recall	f1-score	support
1.0	0.31	0.50	0.38	34
2.0	0.43	0.48	0.46	116
3.0	0.67	0.42	0.52	233
4.0	0.26	0.59	0.36	41
accuracy			0.46	424
macro avg	0.42	0.50	0.43	424
weighted avg	0.54	0.46	0.48	424

Métricas finales - Logistic sobre PCA (test):

	precision	recall	f1-score	support
1.0	0.34	0.68	0.45	34
2.0	0.48	0.55	0.52	116
3.0	0.72	0.40	0.51	233
4.0	0.28	0.63	0.39	41
accuracy			0.49	424
macro avg	0.45	0.57	0.47	424
weighted avg	0.58	0.49	0.50	424

13. Pipeline definitivo de LogisticRegression con características seleccionadas por RFECV

Construye y ajusta un pipeline de LogisticRegression usando el conjunto de features de RFECV, evalúa en validación y test, y guarda el modelo para producción.

```
# Pipeline definitivo: RFECV Top-14 + LogisticRegression

# Tomar las 20 primeras variables de RFECV (ordenadas por ranking)
# (seleccionadas en el punto 7, donde RFECV devolvió 30 features)
rfecv_top20_features = selected_rfecv[:20]

# Construir DataFrames con solo esas 20 columnas
X_train_rf20 = X_train_trans[rfecv_top20_features].copy()
```

```

X_val_rf20 = X_val_trans[rfecv_top20_features].copy()
X_test_rf20 = X_test_trans[rfecv_top20_features].copy()

print(" Shapes RFECV-Top-20:")
print(f" • X_train_rf20: {X_train_rf20.shape}")
print(f" • X_val_rf20: {X_val_rf20.shape}")
print(f" • X_test_rf20: {X_test_rf20.shape}")

# Definir y entrenar LogisticRegression solo con estas 20 features
pipe_rfecv20 = Pipeline([
    ('clf', LogisticRegression(
        max_iter=1000,
        class_weight='balanced',
        random_state=42
    ))
])
pipe_rfecv20.fit(X_train_rf20, y_train)

# Evaluar en validación
print(" Métricas - RFECV Top-20 (validación):")
print(classification_report(y_val, pipe_rfecv20.predict(X_val_rf20)))

# Evaluar en test
print(" Métricas - RFECV Top-20 (test):")
print(classification_report(y_test, pipe_rfecv20.predict(X_test_rf20)))

# Guardar el pipeline para producción
PIPE_DIR = Path('/content/drive/MyDrive/Digitech/TFG/ML/Calculo-Riesgo/artifacts/pipeline')
PIPE_DIR.mkdir(exist_ok=True, parents=True)
joblib.dump(pipe_rfecv20, PIPE_DIR / 'pipeline_rfecv20_logistic.pkl')
print(f" Pipeline RFECV Top-20 + Logistic guardado en: {PIPE_DIR / 'pipeline_rfecv20_logistic.pkl'}")

```

→ Shapes RFECV-Top-20:

- X_train_rf20: (1976, 17)
- X_val_rf20: (424, 17)
- X_test_rf20: (424, 17)

Métricas - RFECV Top-20 (validación):

	precision	recall	f1-score	support
1.0	0.27	0.53	0.36	34
2.0	0.33	0.33	0.33	116
3.0	0.70	0.44	0.54	233
4.0	0.25	0.59	0.35	41
accuracy			0.43	424
macro avg	0.39	0.47	0.39	424
weighted avg	0.52	0.43	0.45	424

Métricas - RFECV Top-20 (test):

	precision	recall	f1-score	support
1.0	0.27	0.56	0.36	34
2.0	0.46	0.47	0.47	116
3.0	0.71	0.35	0.47	233
4.0	0.26	0.73	0.38	41
accuracy			0.44	424

macro avg	0.42	0.53	0.42	424
weighted avg	0.56	0.44	0.45	424

Pipeline RFECV Top-20 + Logistic guardado en: /content/drive/MyDrive/Digitech/TFG/ML

▼ 14. Pipeline alternativo: PCA + LogisticRegression

Define un pipeline que combina PCA (90 % varianza) y LogisticRegression, lo ajusta, evalúa en validación y test, y almacena el pipeline en disco. Primera línea de la celda de código:

```
# Pipeline alternativo (PCA + LogisticRegression)

# Definir pipeline PCA(90% varianza) + LogisticRegression
pipe_pca_final = Pipeline([
    ('pca', PCA(n_components=0.90, random_state=42)),
    ('clf', LogisticRegression(
        max_iter=1000,
        class_weight='balanced',
        random_state=42
    ))
])

# Ajustar el pipeline con todas las variables (X_train_trans, y_train)
pipe_pca_final.fit(X_train_trans, y_train)

# Evaluar en validación
print(" Evaluación PCA + Logistic (validación):")
print(classification_report(y_val, pipe_pca_final.predict(X_val_trans)))

# Evaluar en test
print(" Evaluación PCA + Logistic (test):")
print(classification_report(y_test, pipe_pca_final.predict(X_test_trans)))

# Guardar el pipeline en disco (opcional)
PIPE_DIR = Path('/content/drive/MyDrive/Digitech/TFG/ML/Calculo-Riesgo/artifacts/pipeline')
PIPE_DIR.mkdir(exist_ok=True, parents=True)
joblib.dump(pipe_pca_final, PIPE_DIR / 'pipeline_pca_logistic.pkl')
print(f" Pipeline PCA + Logistic guardado en: {PIPE_DIR / 'pipeline_pca_logistic.pkl'}")
```

→ Evaluación PCA + Logistic (validación):

	precision	recall	f1-score	support
1.0	0.31	0.50	0.38	34
2.0	0.43	0.48	0.46	116
3.0	0.67	0.42	0.52	233
4.0	0.26	0.59	0.36	41
accuracy			0.46	424
macro avg	0.42	0.50	0.43	424
weighted avg	0.54	0.46	0.48	424

Evaluación PCA + Logistic (test):

	precision	recall	f1-score	support
1.0	0.34	0.68	0.45	34
2.0	0.48	0.55	0.52	116
3.0	0.72	0.40	0.51	233
4.0	0.28	0.63	0.39	41
accuracy			0.49	424
macro avg	0.45	0.57	0.47	424
weighted avg	0.58	0.49	0.50	424

Pipeline PCA + Logistic guardado en: /content/drive/MyDrive/Digitech/TFG/ML/Calculo-

▼ 15. Guardado de splits experimentales y artefactos de pipelines

Establece parámetros de versión, crea la lista “Dream Team” de features, guarda esa lista como JSON y persiste los pipelines experimentales en la carpeta correspondiente de config.EXP_ARTIFACTS_DIR.

```
# Guardar los splits EXPERIMENTALES y los artefactos

# --- PARÁMETRO DE CONFIGURACIÓN ---
DATASET_VERSION = '14'
NOTEBOOK_ID = '03_2'
# -----

# Define tu 'Dream Team' de 14 variables (las que encontró RFECV en la celda 14)
dream_team_14 = selected_rfecv

print(f" Dream Team de {len(dream_team_14)} features seleccionado: {dream_team_14}")

# Guarda la lista de features como un artefacto experimental con nombre mejorado
features_path = config.EXP_ARTIFACTS_DIR / f'{NOTEBOOK_ID}_selected_features_rfecv_{DATASET_VERSION}.json'
with open(features_path, 'w') as f:
    json.dump(dream_team_14, f, indent=4)
print(f" Artefacto (lista de features) guardado en: {features_path}")

# Filtra los DataFrames para quedarte solo con esas columnas
X_train_exp = X_train[dream_team_14].copy()
X_val_exp = X_val[dream_team_14].copy()
X_test_exp = X_test[dream_team_14].copy()

# Guarda los nuevos DataFrames en la carpeta de splits EXPERIMENTALES
# Usamos config.EXP_SPLITS_DIR para no sobrescribir los datos finales
print(f"\n Guardando splits EXPERIMENTALES en: {config.EXP_SPLITS_DIR}")

X_train_exp.to_parquet(config.EXP_SPLITS_DIR / f'X_train_{DATASET_VERSION}.parquet')
X_val_exp.to_parquet(config.EXP_SPLITS_DIR / f'X_val_{DATASET_VERSION}.parquet')
X_test_exp.to_parquet(config.EXP_SPLITS_DIR / f'X_test_{DATASET_VERSION}.parquet')

# Guardamos las 'y' correspondientes en la misma carpeta experimental para tener el set completo
y_train_exp.to_parquet(config.EXP_SPLITS_DIR / f'y_train_{DATASET_VERSION}.parquet')
y_val_exp.to_parquet(config.EXP_SPLITS_DIR / f'y_val_{DATASET_VERSION}.parquet')
y_test_exp.to_parquet(config.EXP_SPLITS_DIR / f'y_test_{DATASET_VERSION}.parquet')
```