Generación de Variables de Riesgo - Todos los Usuarios

Home Credit Default Risk

Objetivo: Crear un DataFrame con todas las variables de riesgo para los 307,511 usuarios

---

In [1]:
```
import pandas as pd
import numpy as np
from tqdm import tqdm
import warnings
warnings.filterwarnings('ignore')

pd.set_option('display.max_columns', None)
pd.set_option('display.float_format', lambda x: '%.3f' % x)

print("✓ Librerías cargadas")
```

```
✓ Librerías cargadas
```

1. Carga de Datos

In [2]:
```
!pwd
```

```
/Users/jeronimo.deli/Desktop/other/Vs/MM/home_credit_risk_default/src/eda
```

In [3]:
```
print("Cargando datasets...\n")

app_train = pd.read_csv('../../data/home-credit-default-risk/application_tra
bureau = pd.read_csv('../../data/home-credit-default-risk/bureau.csv')
bureau_balance = pd.read_csv('../../data/home-credit-default-risk/bureau_bal
prev_app = pd.read_csv('../../data/home-credit-default-risk/previous_applica
pos_cash = pd.read_csv('../../data/home-credit-default-risk/POS_CASH_balance
credit_card = pd.read_csv('../../data/home-credit-default-risk/credit_card_b
installments = pd.read_csv('../../data/home-credit-default-risk/installments

print(f"✓ Application Train: {app_train.shape[0]:,} usuarios")
print(f"✓ Bureau: {bureau.shape[0]:,} registros")
print(f"✓ Bureau Balance: {bureau_balance.shape[0]:,} registros")
print(f"✓ Previous Application: {prev_app.shape[0]:,} registros")
print(f"✓ POS Cash: {pos_cash.shape[0]:,} registros")
print(f"✓ Credit Card: {credit_card.shape[0]:,} registros")
print(f"✓ Installments: {installments.shape[0]:,} registros")
print(f"\n✓ Datos cargados correctamente")
```

```
Cargando datasets...

✓ Application Train: 307,511 usuarios
✓ Bureau: 1,716,428 registros
✓ Bureau Balance: 27,299,925 registros
✓ Previous Application: 1,670,214 registros
✓ POS Cash: 10,001,358 registros
✓ Credit Card: 3,840,312 registros
✓ Installments: 13,605,401 registros

✓ Datos cargados correctamente
```

3. Procesamiento de Todos los Usuarios

Generamos el DataFrame completo con todas las variables para los 307,511 usuarios:

In [4]:

```
# ================================================================================
# GENERACIÓN DE FEATURES - VERSIÓN OPTIMIZADA
# ================================================================================
print("="*80)
print("GENERANDO DATASET COMPLETO - VERSIÓN OPTIMIZADA")
print("="*80)
print(f"\nTotal de usuarios: {len(app_train):,}")
print(f"Variables por usuario: 40\n")

# PARTE 1: FEATURES BASE DE APPLICATION
print("Generando features base...")

df = app_train[['SK_ID_CURR', 'TARGET']].copy()

# Features básicas
df['AMT_CREDIT'] = app_train['AMT_CREDIT']
df['AMT_ANNUITY'] = app_train['AMT_ANNUITY']
df['AMT_INCOME_TOTAL'] = app_train['AMT_INCOME_TOTAL']

# Edad
df['DAYS_BIRTH'] = app_train['DAYS_BIRTH']
df['EDAD_ANOS'] = abs(app_train['DAYS_BIRTH']) / 365.25

# Scores externos
df['EXT_SOURCE_1'] = app_train['EXT_SOURCE_1']
df['EXT_SOURCE_2'] = app_train['EXT_SOURCE_2']
df['EXT_SOURCE_3'] = app_train['EXT_SOURCE_3']
df['SCORE_PROMEDIO'] = app_train[['EXT_SOURCE_1', 'EXT_SOURCE_2', 'EXT_SOURC

# Ratios
df['CREDIT_INCOME_RATIO'] = df['AMT_CREDIT'] / df['AMT_INCOME_TOTAL']

# Familia
df['NAME_FAMILY_STATUS'] = app_train['NAME_FAMILY_STATUS']
df['CNT_CHILDREN'] = app_train['CNT_CHILDREN']
df['CNT_FAM_MEMBERS'] = app_train['CNT_FAM_MEMBERS']
df['CODE_GENDER'] = app_train['CODE_GENDER']
df['NAME_EDUCATION_TYPE'] = app_train['NAME_EDUCATION_TYPE']
```

```
df['INGRESO_PER_CAPITA'] = np.where(
    df['CNT_FAM_MEMBERS'] > 0,
    df['AMT_INCOME_TOTAL'] / df['CNT_FAM_MEMBERS'],
    df['AMT_INCOME_TOTAL']
)

# Activos
df['FLAG_OWN_CAR'] = app_train['FLAG_OWN_CAR']
df['FLAG_OWN_REALTY'] = app_train['FLAG_OWN_REALTY']
df['NUM_ACTIVOS'] = (app_train['FLAG_OWN_CAR'] == 'Y').astype(int) + \
                    (app_train['FLAG_OWN_REALTY'] == 'Y').astype(int)

# Consultas buró
bureau_cols = ['AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CREDIT_BUREAU_DAY',
               'AMT_REQ_CREDIT_BUREAU_WEEK', 'AMT_REQ_CREDIT_BUREAU_MON',
               'AMT_REQ_CREDIT_BUREAU_QRT', 'AMT_REQ_CREDIT_BUREAU_YEAR']
df['TOTAL_CONSULTAS_BURO'] = app_train[bureau_cols].fillna(0).sum(axis=1)

print("✓ Features base generadas")

# PARTE 2: FEATURES DE BUREAU
print("Procesando features de Bureau...")

bureau_agg = bureau.groupby('SK_ID_CURR').agg({
    'AMT_CREDIT_SUM_LIMIT': 'sum',
    'AMT_CREDIT_SUM': 'sum',
    'AMT_CREDIT_SUM_DEBT': 'sum',
    'CREDIT_DAY_OVERDUE': 'max',
    'SK_ID_BUREAU': 'count'
}).reset_index()

bureau_agg.columns = ['SK_ID_CURR', 'TOTAL_CREDITO_DISPONIBLE',
                      'TOTAL_CREDITO_OTORGADO', 'TOTAL_DEUDA_ACTUAL',
                      'MAX_DIAS_MORA', 'CANTIDAD_CREDITOS_BURO']

# Créditos activos y cerrados
bureau_active = bureau[bureau['CREDIT_ACTIVE'] == 'Active'].groupby('SK_ID_C
bureau_closed = bureau[bureau['CREDIT_ACTIVE'] == 'Closed'].groupby('SK_ID_C

bureau_agg = bureau_agg.merge(bureau_active, on='SK_ID_CURR', how='left')
bureau_agg = bureau_agg.merge(bureau_closed, on='SK_ID_CURR', how='left')

bureau_agg['CREDITOS_ACTIVOS'] = bureau_agg['CREDITOS_ACTIVOS'].fillna(0)
bureau_agg['CREDITOS_CERRADOS'] = bureau_agg['CREDITOS_CERRADOS'].fillna(0)
bureau_agg['TIENE_IMPAGOS'] = (bureau_agg['MAX_DIAS_MORA'] > 0).astype(int)

# Merge con df principal
df = df.merge(bureau_agg, on='SK_ID_CURR', how='left')

# Llenar NaN
bureau_fill_cols = ['TOTAL_CREDITO_DISPONIBLE', 'TOTAL_CREDITO_OTORGADO',
                    'TOTAL_DEUDA_ACTUAL', 'CREDITOS_ACTIVOS', 'CREDITOS_CERR
                    'MAX_DIAS_MORA', 'TIENE_IMPAGOS', 'CANTIDAD_CREDITOS_BUR
df[bureau_fill_cols] = df[bureau_fill_cols].fillna(0)

df['ES_PRIMER_CREDITO'] = (df['CANTIDAD_CREDITOS_BURO'] == 0).astype(int)
```

```
print("✓ Features de Bureau generadas")

# PARTE 3: FEATURES DE BUREAU BALANCE
print("Procesando Bureau Balance...")

bureau_balance_merged = bureau_balance.merge(
    bureau[['SK_ID_BUREAU', 'SK_ID_CURR']],
    on='SK_ID_BUREAU',
    how='left'
)

mora_status = ['1', '2', '3', '4', '5']
bureau_balance_merged['EN_MORA'] = bureau_balance_merged['STATUS'].isin(mora

balance_agg = bureau_balance_merged.groupby('SK_ID_CURR').agg({
    'EN_MORA': 'sum',
    'STATUS': 'count'
}).reset_index()

balance_agg.columns = ['SK_ID_CURR', 'MESES_CON_MORA', 'TOTAL_MESES']
balance_agg['PCT_MESES_MORA'] = (balance_agg['MESES_CON_MORA'] / balance_agg

# Créditos con impago
creditos_impago = bureau_balance_merged.groupby(['SK_ID_CURR', 'SK_ID_BUREAU
creditos_impago_count = creditos_impago.groupby('SK_ID_CURR')['EN_MORA'].sum
creditos_impago_count.columns = ['SK_ID_CURR', 'CREDITOS_CON_IMPAGO']

balance_agg = balance_agg.merge(creditos_impago_count, on='SK_ID_CURR', how=

df = df.merge(balance_agg[['SK_ID_CURR', 'MESES_CON_MORA', 'PCT_MESES_MORA',
              on='SK_ID_CURR', how='left')

df[['MESES_CON_MORA', 'PCT_MESES_MORA', 'CREDITOS_CON_IMPAGO']] = \
    df[['MESES_CON_MORA', 'PCT_MESES_MORA', 'CREDITOS_CON_IMPAGO']].fillna(0

print("✓ Features de Bureau Balance generadas")

# PARTE 4: FEATURES DE PREVIOUS APPLICATION
print("Procesando Previous Applications...")

prev_agg = prev_app.groupby('SK_ID_CURR').agg({
    'SK_ID_PREV': 'count',
    'RATE_INTEREST_PRIMARY': 'mean',
    'CNT_PAYMENT': 'mean',
    'AMT_CREDIT': ['mean', 'sum']
}).reset_index()

prev_agg.columns = ['SK_ID_CURR', 'NUM_PRESTAMOS_PREVIOS', 'TASA_INTERES_PRO
                    'PLAZO_PROMEDIO', 'MONTO_PROMEDIO_PREVIO', 'TOTAL_CREDIT

df = df.merge(prev_agg, on='SK_ID_CURR', how='left')

df[['NUM_PRESTAMOS_PREVIOS', 'MONTO_PROMEDIO_PREVIO', 'TOTAL_CREDITO_HISTORI
    df[['NUM_PRESTAMOS_PREVIOS', 'MONTO_PROMEDIO_PREVIO', 'TOTAL_CREDITO_HIS
```

```
print("✓ Features de Previous Applications generadas")

# PARTE 5: FEATURES DE INSTALLMENTS
print("Procesando Installments...")

valid_installments = installments[installments['AMT_INSTALMENT'] > 0].copy()
valid_installments['PAYMENT_RATIO'] = valid_installments['AMT_PAYMENT'] / va

install_agg = valid_installments.groupby('SK_ID_CURR')['PAYMENT_RATIO'].mear
install_agg.columns = ['SK_ID_CURR', 'RATIO_PAGO_CUOTA']

df = df.merge(install_agg, on='SK_ID_CURR', how='left')

print("✓ Features de Installments generadas")

# PARTE 6: FEATURES DE CREDIT CARD
print("Procesando Credit Card Balance...")

valid_cc = credit_card[credit_card['AMT_INST_MIN_REGULARITY'] > 0].copy()
valid_cc['PAYMENT_MIN_RATIO'] = valid_cc['AMT_PAYMENT_CURRENT'] / valid_cc['

cc_agg = valid_cc.groupby('SK_ID_CURR')['PAYMENT_MIN_RATIO'].mean().reset_ir
cc_agg.columns = ['SK_ID_CURR', 'RATIO_PAGO_MINIMO_TC']

df = df.merge(cc_agg, on='SK_ID_CURR', how='left')

print("✓ Features de Credit Card generadas")

# RESULTADO FINAL
print("\n" + "="*80)
print("✅ DATASET COMPLETO GENERADO")
print("="*80)
print(f"Usuarios procesados: {len(df):,}")
print(f"Variables: {len(df.columns)}")
print(f"Shape: {df.shape}")
print(f"Memoria: {df.memory_usage(deep=True).sum() / 1024**2:.2f} MB")
print("="*80)
```

```
================================================================================
====
GENERANDO DATASET COMPLETO - VERSIÓN OPTIMIZADA
================================================================================
====

Total de usuarios: 307,511
Variables por usuario: 40

Generando features base...
✓ Features base generadas
Procesando features de Bureau...
✓ Features de Bureau generadas
Procesando Bureau Balance...
✓ Features de Bureau Balance generadas
Procesando Previous Applications...
✓ Features de Previous Applications generadas
Procesando Installments...
✓ Features de Installments generadas
Procesando Credit Card Balance...
✓ Features de Credit Card generadas

================================================================================
====
✅ DATASET COMPLETO GENERADO
================================================================================
====
Usuarios procesados: 307,511
Variables: 41
Shape: (307511, 41)
Memoria: 167.42 MB
================================================================================
====
```

In [5]:
```
df
```

Out[5]:

	SK_ID_CURR	TARGET	AMT_CREDIT	AMT_ANNUITY	AMT_INCOME_TOTAL	I
0	100002	1	406597.500	24700.500	202500.000	
1	100003	0	1293502.500	35698.500	270000.000	
2	100004	0	135000.000	6750.000	67500.000	
3	100006	0	312682.500	29686.500	135000.000	
4	100007	0	513000.000	21865.500	121500.000	
...	...	...	...	...	...	
307506	456251	0	254700.000	27558.000	157500.000	
307507	456252	0	269550.000	12001.500	72000.000	
307508	456253	0	677664.000	29979.000	153000.000	
307509	456254	1	370107.000	20205.000	171000.000	
307510	456255	0	675000.000	49117.500	157500.000	

307511 rows × 41 columns

In [6]:

```
# Drop de variables redundantes/derivadas
df = df.drop(columns=[
    # Edad
    'DAYS_BIRTH',

    # Montos base
    'AMT_CREDIT',
    'AMT_INCOME_TOTAL',
    'AMT_ANNUITY',

    # Scores individuales
    'EXT_SOURCE_1',
    'EXT_SOURCE_2',
    'EXT_SOURCE_3',

    # Familia
    'CNT_FAM_MEMBERS',

    # Activos
    'FLAG_OWN_CAR',
    'FLAG_OWN_REALTY',

    # Consultas buró total
    'TOTAL_CONSULTAS_BURO',
```

```
    # Componentes de ratios (solo MESES_CON_MORA está en df)
    'MESES_CON_MORA',

    # Variables binarias derivadas
    'TIENE_IMPAGOS',
    'ES_PRIMER_CREDITO',

    # Créditos buró
    'CANTIDAD_CREDITOS_BURO'
])

print(f"Variables restantes: {df.shape[1]}")
print(f"Observaciones: {df.shape[0]:,}")
```

```
Variables restantes: 26
Observaciones: 307,511
```

4. Vista Previa del Dataset

In [7]:
```
print("Primeras 10 filas del dataset:\n")
df
```

```
Primeras 10 filas del dataset:
```

Out[7]:

	SK_ID_CURR	TARGET	EDAD_ANOS	SCORE_PROMEDIO	CREDIT_INCOME_RA
0	100002	1	25.903	0.162	2.
1	100003	0	45.900	0.467	4
2	100004	0	52.145	0.643	2.
3	100006	0	52.033	0.650	2
4	100007	0	54.571	0.323	4.
...	...	...	...	...	
307506	456251	0	25.536	0.414	1
307507	456252	0	56.879	0.116	3.
307508	456253	0	40.975	0.500	4.
307509	456254	1	32.747	0.588	2
307510	456255	0	46.149	0.519	4.

307511 rows × 26 columns

5. Información del Dataset

In [8]:
```
print("\n📋 INFORMACIÓN DEL DATASET")
print("="*80)
print(f"\nColumnas ({len(df.columns)}):")
print(df.columns.tolist())

print(f"\nTipos de datos:")
print(df.dtypes.value_counts())

print(f"\nMemoria utilizada: {df.memory_usage(deep=True).sum() / 1024**2:.2f

print(f"\nDistribución del TARGET:")
print(df['TARGET'].value_counts())
print(f"\nTasa de default: {df['TARGET'].mean() * 100:.2f}%")
```

```
📋 INFORMACIÓN DEL DATASET
================================================================================
====

Columnas (26):
['SK_ID_CURR', 'TARGET', 'EDAD_ANOS', 'SCORE_PROMEDIO', 'CREDIT_INCOME_RATI
O', 'NAME_FAMILY_STATUS', 'CNT_CHILDREN', 'CODE_GENDER', 'NAME_EDUCATION_TYP
E', 'INGRESO_PER_CAPITA', 'NUM_ACTIVOS', 'TOTAL_CREDITO_DISPONIBLE', 'TOTAL_
CREDITO_OTORGADO', 'TOTAL_DEUDA_ACTUAL', 'MAX_DIAS_MORA', 'CREDITOS_ACTIVO
S', 'CREDITOS_CERRADOS', 'PCT_MESES_MORA', 'CREDITOS_CON_IMPAGO', 'NUM_PREST
AMOS_PREVIOS', 'TASA_INTERES_PROMEDIO', 'PLAZO_PROMEDIO', 'MONTO_PROMEDIO_PR
EVIO', 'TOTAL_CREDITO_HISTORICO', 'RATIO_PAGO_CUOTA', 'RATIO_PAGO_MINIMO_T
C']

Tipos de datos:
float64    19
int64       4
object      3
Name: count, dtype: int64

Memoria utilizada: 107.59 MB

Distribución del TARGET:
TARGET
0    282686
1     24825
Name: count, dtype: int64

Tasa de default: 8.07%
```

6. Estadísticas Descriptivas

In [9]:
```
print("\n📊 Estadísticas de variables numéricas:\n")
numeric_cols = df.select_dtypes(include=[np.number]).columns
df[numeric_cols].describe()
```

📊 Estadísticas de variables numéricas:

Out[9]:

	SK_ID_CURR	TARGET	EDAD_ANOS	SCORE_PROMEDIO	CREDIT_INCOME_R
count	307511.000	307511.000	307511.000	307339.000	307511
mean	278180.519	0.081	43.907	0.509	3
std	102790.175	0.272	11.948	0.150	2
min	100002.000	0.000	20.504	0.000	0
25%	189145.500	0.000	33.985	0.414	2
50%	278202.000	0.000	43.121	0.525	3
75%	367142.500	0.000	53.886	0.623	5
max	456255.000	1.000	69.073	0.879	84

7. Valores Faltantes

In [10]:
```
print("\n📉 Porcentaje de valores faltantes por variable:\n")
missing_pct = (df.isna().sum() / len(df) * 100).sort_values(ascending=False)
missing_pct[missing_pct > 0]
```

📉 Porcentaje de valores faltantes por variable:

Out[10]:
```
TASA_INTERES_PROMEDIO   98.501
RATIO_PAGO_MINIMO_TC    80.726
PLAZO_PROMEDIO           5.486
RATIO_PAGO_CUOTA         5.164
SCORE_PROMEDIO           0.056
dtype: float64
```

8. Análisis por TARGET

In [11]:
```
print("\n COMPARACIÓN: Pagó (0) vs Default (1)")
print("="*80)

vars_comparar = [
    'EDAD_ANOS', 'AMT_CREDIT', 'AMT_INCOME_TOTAL', 'CREDIT_INCOME_RATIO',
    'SCORE_PROMEDIO', 'TOTAL_DEUDA_ACTUAL', 'CREDITOS_ACTIVOS',
    'PCT_MESES_MORA', 'CREDITOS_CON_IMPAGO', 'NUM_ACTIVOS',
    'RATIO_PAGO_CUOTA', 'RATIO_PAGO_MINIMO_TC'
]

vars_comparar = [v for v in vars_comparar if v in df.columns]

comparacion = df.groupby('TARGET')[vars_comparar].mean().T
comparacion.columns = ['Pagó (0)', 'Default (1)']
comparacion['Diferencia'] = comparacion['Default (1)'] - comparacion['Pagó (
comparacion['% Cambio'] = (comparacion['Diferencia'] / comparacion['Pagó (0)
```

```
print(comparacion)
print("\n" + "="*80)
```

```
 COMPARACIÓN: Pagó (0) vs Default (1)
================================================================================
====
                        Pagó (0)  Default (1)  Diferencia  % Cambio
EDAD_ANOS                 44.184       40.752      -3.431    -7.766
CREDIT_INCOME_RATIO        3.964        3.887      -0.076    -1.925
SCORE_PROMEDIO             0.519        0.397      -0.122   -23.522
TOTAL_DEUDA_ACTUAL    548083.039   558718.014   10634.975     1.940
CREDITOS_ACTIVOS           1.739        2.028       0.289    16.600
PCT_MESES_MORA             0.504        0.874       0.370    73.412
CREDITOS_CON_IMPAGO        0.216        0.283       0.067    31.274
NUM_ACTIVOS                1.038        0.989      -0.048    -4.665
RATIO_PAGO_CUOTA           1.371        1.521       0.150    10.966
RATIO_PAGO_MINIMO_TC      21.597        6.278     -15.318   -70.929

================================================================================
====
```

In [12]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (
    classification_report, confusion_matrix, roc_auc_score,
    roc_curve, precision_recall_curve, average_precision_score
)
import pickle
import warnings
warnings.filterwarnings('ignore')

# ============================================================================
# CONFIGURACIÓN
# ============================================================================
TEST_SIZE = 0.2
RANDOM_STATE = 42
CV_FOLDS = 5

print("="*80)
print("MODELO DE REGRESION LOGISTICA - HOME CREDIT DEFAULT RISK")
print("="*80)

# ============================================================================
# 1. VERIFICACIÓN DE DATOS
# ============================================================================
print("\n[1/15] Verificando datos existentes...")
print(f"Dataset en memoria: {df.shape}")
print(f"Usuarios: {len(df):,}")
print(f"Variables: {len(df.columns)}")
print(f"Tasa de default: {df['TARGET'].mean()*100:.2f}%")
```

```python
# ============================================================================
# 2. PREPARACIÓN DE DATOS
# ============================================================================
print("\n[2/15] Preparando datos...")

X = df.drop(['SK_ID_CURR', 'TARGET'], axis=1)
y = df['TARGET']

print(f"Variables predictoras: {X.shape[1]}")
print(f"Variable objetivo: {y.name}")

numeric_vars = X.select_dtypes(include=[np.number]).columns.tolist()
categorical_vars = X.select_dtypes(include=['object']).columns.tolist()

print(f"Variables numericas: {len(numeric_vars)}")
print(f"Variables categoricas: {len(categorical_vars)}")

# ============================================================================
# 3. MANEJO DE VALORES FALTANTES
# ============================================================================
print("\n[3/15] Analizando valores faltantes...")

missing_pct = (X.isnull().sum() / len(X) * 100).sort_values(ascending=False)
missing_vars = missing_pct[missing_pct > 0]

if len(missing_vars) > 0:
    print(f"Variables con valores faltantes: {len(missing_vars)}")
    print(missing_vars.head(10))

    print("Imputando valores faltantes...")
    for col in numeric_vars:
        if X[col].isnull().sum() > 0:
            X[col].fillna(X[col].median(), inplace=True)

    for col in categorical_vars:
        if X[col].isnull().sum() > 0:
            X[col].fillna(X[col].mode()[0], inplace=True)

    print("Valores faltantes imputados")
else:
    print("No hay valores faltantes")

# ============================================================================
# 4. CODIFICACIÓN DE VARIABLES CATEGÓRICAS
# ============================================================================
print("\n[4/15] Codificando variables categoricas...")

label_encoders = {}

if len(categorical_vars) > 0:
    for col in categorical_vars:
        le = LabelEncoder()
        X[col] = le.fit_transform(X[col].astype(str))
        label_encoders[col] = le

    print(f"{len(categorical_vars)} variables codificadas")
```

```
    print(f"Categorias: {categorical_vars}")

# ============================================================================
# 5. DIVISIÓN TRAIN/TEST
# ============================================================================
print("\n[5/15] Dividiendo datos en Train/Test...")

X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=TEST_SIZE,
    random_state=RANDOM_STATE,
    stratify=y
)

print(f"Train set: {X_train.shape[0]:,} usuarios ({X_train.shape[0]/len(X)*1
print(f"Test set:  {X_test.shape[0]:,} usuarios ({X_test.shape[0]/len(X)*100
print(f"\nDistribucion TARGET en Train:")
print(f"  Clase 0 (Pago): {(y_train==0).sum():,} ({(y_train==0).sum()/len(y_
print(f"  Clase 1 (Default): {(y_train==1).sum():,} ({(y_train==1).sum()/ler

# ============================================================================
# 6. NORMALIZACIÓN
# ============================================================================
print("\n[6/15] Normalizando variables (StandardScaler)...")

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

print("Datos normalizados (media=0, std=1)")
print(f"Media de features en train: {X_train_scaled.mean():.6f}")
print(f"Std de features en train: {X_train_scaled.std():.6f}")

# ============================================================================
# 7. ENTRENAMIENTO DEL MODELO
# ============================================================================
print("\n[7/15] Entrenando Regresion Logistica...")

model = LogisticRegression(
    penalty='l2',
    C=1.0,
    solver='lbfgs',
    max_iter=1000,
    class_weight='balanced',
    random_state=RANDOM_STATE,
    n_jobs=-1
)

model.fit(X_train_scaled, y_train)

print("Modelo entrenado exitosamente")
print(f"Iteraciones: {model.n_iter_[0]}")
print(f"Clases: {model.classes_}")

# ============================================================================
# 8. VALIDACIÓN CRUZADA
```

```python
# ============================================================================
print(f"\n[8/15] Validacion Cruzada ({CV_FOLDS}-fold en Train)...")

cv_scores = cross_val_score(
    model, X_train_scaled, y_train,
    cv=CV_FOLDS,
    scoring='roc_auc',
    n_jobs=-1
)

print(f"ROC-AUC por fold: {cv_scores}")
print(f"Media: {cv_scores.mean():.4f} (+/- {cv_scores.std():.4f})")

# ============================================================================
# 9. PREDICCIONES
# ============================================================================
print("\n[9/15] Generando predicciones...")

y_train_pred = model.predict(X_train_scaled)
y_train_proba = model.predict_proba(X_train_scaled)[:, 1]

y_test_pred = model.predict(X_test_scaled)
y_test_proba = model.predict_proba(X_test_scaled)[:, 1]

print("Predicciones generadas")

# ============================================================================
# 10. EVALUACIÓN EN TRAIN
# ============================================================================
print("\n[10/15] Evaluando modelo en TRAIN SET...")
print("="*80)

train_auc = roc_auc_score(y_train, y_train_proba)
train_ap = average_precision_score(y_train, y_train_proba)

print(f"\nMetricas Globales:")
print(f"ROC-AUC Score: {train_auc:.4f}")
print(f"Average Precision: {train_ap:.4f}")

print(f"\nClassification Report:")
print(classification_report(y_train, y_train_pred,
                            target_names=['Pago (0)', 'Default (1)']))

print(f"\nConfusion Matrix:")
cm_train = confusion_matrix(y_train, y_train_pred)
print(cm_train)
print(f"\nInterpretacion:")
print(f"  Verdaderos Negativos (TN): {cm_train[0,0]:,} - Predijo 'Pago' y si
print(f"  Falsos Positivos (FP): {cm_train[0,1]:,} - Predijo 'Default' pero
print(f"  Falsos Negativos (FN): {cm_train[1,0]:,} - Predijo 'Pago' pero hiz
print(f"  Verdaderos Positivos (TP): {cm_train[1,1]:,} - Predijo 'Default' y

# ============================================================================
# 11. EVALUACIÓN EN TEST
# ============================================================================
print("\n[11/15] Evaluando modelo en TEST SET (metricas definitivas)...")
```

```
print("="*80)

test_auc = roc_auc_score(y_test, y_test_proba)
test_ap = average_precision_score(y_test, y_test_proba)

print(f"\nMetricas Globales:")
print(f"ROC-AUC Score: {test_auc:.4f}")
print(f"Average Precision: {test_ap:.4f}")

print(f"\nClassification Report:")
print(classification_report(y_test, y_test_pred,
                            target_names=['Pago (0)', 'Default (1)']))

print(f"\nConfusion Matrix:")
cm_test = confusion_matrix(y_test, y_test_pred)
print(cm_test)
print(f"\nInterpretacion:")
print(f"  Verdaderos Negativos (TN): {cm_test[0,0]:,} - Predijo 'Pago' y si
print(f"  Falsos Positivos (FP): {cm_test[0,1]:,} - Predijo 'Default' pero p
print(f"  Falsos Negativos (FN): {cm_test[1,0]:,} - Predijo 'Pago' pero hizo
print(f"  Verdaderos Positivos (TP): {cm_test[1,1]:,} - Predijo 'Default' y

# ============================================================================
# 12. COMPARACIÓN TRAIN VS TEST
# ============================================================================
print("\n[12/15] Comparando TRAIN vs TEST (analisis de overfitting)...")
print("="*80)

print(f"\n{'Metrica':<25} {'Train':<12} {'Test':<12} {'Diferencia':<12}")
print("-" * 65)
print(f"{'ROC-AUC':<25} {train_auc:<12.4f} {test_auc:<12.4f} {abs(train_auc-
print(f"{'Average Precision':<25} {train_ap:<12.4f} {test_ap:<12.4f} {abs(tr

diff_auc = abs(train_auc - test_auc)
if diff_auc < 0.02:
    print("\nExcelente generalizacion (diferencia < 2%)")
elif diff_auc < 0.05:
    print("\nBuena generalizacion (diferencia < 5%)")
elif diff_auc < 0.10:
    print("\nPosible ligero overfitting (diferencia 5-10%)")
else:
    print("\nOverfitting detectado (diferencia > 10%)")

# ============================================================================
# 13. IMPORTANCIA DE VARIABLES
# ============================================================================
print("\n[13/15] Calculando importancia de variables...")
print("="*80)

feature_importance = pd.DataFrame({
    'Variable': X.columns,
    'Coeficiente': model.coef_[0],
    'Importancia_Abs': np.abs(model.coef_[0])
}).sort_values('Importancia_Abs', ascending=False)

print("\nTop 20 variables que mas influyen en el riesgo de default:\n")
```

```
print(feature_importance.head(20).to_string(index=False))

print("\nInterpretacion de coeficientes:")
print("  Coeficiente positivo (+): Aumenta probabilidad de DEFAULT")
print("  Coeficiente negativo (-): Disminuye probabilidad de DEFAULT")

# ============================================================================
# 14. VISUALIZACIONES
# ============================================================================
print("\n[14/15] Generando visualizaciones...")

# Calcular deciles
test_results = pd.DataFrame({
    'y_true': y_test,
    'y_proba': y_test_proba
})
test_results['decil'] = pd.qcut(test_results['y_proba'], q=10, labels=False,

# Calcular TPR por decil
decil_stats = test_results.groupby('decil').agg({
    'y_true': ['sum', 'count', 'mean']
}).reset_index()
decil_stats.columns = ['decil', 'positivos', 'total', 'tpr']
decil_stats = decil_stats.sort_values('decil', ascending=False)

# Crear gráficas
fig, axes = plt.subplots(1, 3, figsize=(20, 6))

# ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_test_proba)
axes[0].plot(fpr, tpr, linewidth=2, label=f'ROC (AUC = {test_auc:.4f})')
axes[0].plot([0, 1], [0, 1], 'k--', linewidth=1, label='Random')
axes[0].set_xlabel('False Positive Rate', fontsize=12)
axes[0].set_ylabel('True Positive Rate', fontsize=12)
axes[0].set_title('ROC Curve - Test Set', fontsize=14, fontweight='bold')
axes[0].legend()
axes[0].grid(alpha=0.3)

# Precision-Recall Curve (Recall en X, Precision en Y)
precision, recall, thresholds_pr = precision_recall_curve(y_test, y_test_prc

# Ordenar por recall ascendente para graficar correctamente
sort_idx = np.argsort(recall)
recall_sorted = recall[sort_idx]
precision_sorted = precision[sort_idx]

axes[1].plot(recall_sorted, precision_sorted, linewidth=2, label=f'PR (AP =
axes[1].axhline(y=y_test.mean(), color='gray', linestyle='--', linewidth=1,
axes[1].set_xlabel('Recall (TPR)', fontsize=12)
axes[1].set_ylabel('Precision', fontsize=12)
axes[1].set_title('Precision-Recall Curve - Test Set', fontsize=14, fontweig
axes[1].set_xlim([0, 1])
axes[1].set_ylim([0, 1])
axes[1].legend()
axes[1].grid(alpha=0.3)
```

```python
# TPR por Decil
axes[2].bar(decil_stats['decil'], decil_stats['tpr'], color='steelblue', alp
axes[2].set_xlabel('Decil de Score (10 = Mayor Riesgo)', fontsize=12)
axes[2].set_ylabel('True Positive Rate (TPR)', fontsize=12)
axes[2].set_title('TPR por Decil de Probabilidad', fontsize=14, fontweight='
axes[2].set_xticks(range(1, 11))
axes[2].grid(alpha=0.3, axis='y')
axes[2].set_ylim(0, 1)

# Añadir valores en las barras
for i, row in decil_stats.iterrows():
    axes[2].text(row['decil'], row['tpr'] + 0.02, f"{row['tpr']:.2f}",
                 ha='center', va='bottom', fontsize=10)

plt.tight_layout()
plt.savefig('logistic_regression_curves.png', dpi=300, bbox_inches='tight')
print("Graficas guardadas: logistic_regression_curves.png")

# Imprimir tabla de deciles
print("\nTabla de TPR por Decil:")
print(decil_stats.to_string(index=False))

# ============================================================================
# 15. GUARDAR MODELO Y RESULTADOS
# ============================================================================
print("\n[15/15] Guardando modelo y resultados...")

with open('logistic_model.pkl', 'wb') as f:
    pickle.dump(model, f)

with open('scaler.pkl', 'wb') as f:
    pickle.dump(scaler, f)

with open('label_encoders.pkl', 'wb') as f:
    pickle.dump(label_encoders, f)

feature_importance.to_csv('feature_importance.csv', index=False)

test_predictions = pd.DataFrame({
    'SK_ID_CURR': df.iloc[X_test.index]['SK_ID_CURR'].values,
    'TARGET_Real': y_test.values,
    'TARGET_Predicho': y_test_pred,
    'Probabilidad_Default': y_test_proba
})
test_predictions.to_csv('test_predictions.csv', index=False)

print("Archivos guardados:")
print("  - logistic_model.pkl")
print("  - scaler.pkl")
print("  - label_encoders.pkl")
print("  - feature_importance.csv")
print("  - test_predictions.csv")

# ============================================================================
# RESUMEN FINAL
# ============================================================================
```

```
print("\n" + "="*80)
print("ANALISIS COMPLETADO")
print("="*80)
print(f"\nResultados Finales (TEST SET):")
print(f"  ROC-AUC: {test_auc:.4f}")
print(f"  Average Precision: {test_ap:.4f}")
print(f"  Accuracy: {(y_test_pred == y_test).mean():.4f}")
print(f"\nEl modelo esta listo para produccion")
print("="*80)
```

```
================================================================================
MODELO DE REGRESION LOGISTICA - HOME CREDIT DEFAULT RISK
================================================================================

[1/15] Verificando datos existentes...
Dataset en memoria: (307511, 26)
Usuarios: 307,511
Variables: 26
Tasa de default: 8.07%

[2/15] Preparando datos...
Variables predictoras: 24
Variable objetivo: TARGET
Variables numericas: 21
Variables categoricas: 3

[3/15] Analizando valores faltantes...
Variables con valores faltantes: 5
TASA_INTERES_PROMEDIO   98.501
RATIO_PAGO_MINIMO_TC    80.726
PLAZO_PROMEDIO           5.486
RATIO_PAGO_CUOTA         5.164
SCORE_PROMEDIO           0.056
dtype: float64
Imputando valores faltantes...
Valores faltantes imputados

[4/15] Codificando variables categoricas...
3 variables codificadas
Categorias: ['NAME_FAMILY_STATUS', 'CODE_GENDER', 'NAME_EDUCATION_TYPE']

[5/15] Dividiendo datos en Train/Test...
Train set: 246,008 usuarios (80.0%)
Test set:  61,503 usuarios (20.0%)

Distribucion TARGET en Train:
  Clase 0 (Pago): 226,148 (91.93%)
  Clase 1 (Default): 19,860 (8.07%)

[6/15] Normalizando variables (StandardScaler)...
Datos normalizados (media=0, std=1)
Media de features en train: -0.000000
Std de features en train: 1.000000

[7/15] Entrenando Regresion Logistica...
Modelo entrenado exitosamente
Iteraciones: 26
Clases: [0 1]

[8/15] Validacion Cruzada (5-fold en Train)...
ROC-AUC por fold: [0.73107992 0.72574752 0.72971804 0.73457814 0.73818964]
Media: 0.7319 (+/- 0.0042)

[9/15] Generando predicciones...
```

```
Predicciones generadas

[10/15] Evaluando modelo en TRAIN SET...
================================================================================
====

Metricas Globales:
ROC-AUC Score: 0.7323
Average Precision: 0.2049

Classification Report:
              precision    recall  f1-score   support

    Pago (0)       0.96      0.68      0.80    226148
 Default (1)       0.15      0.66      0.25     19860

    accuracy                           0.68    246008
   macro avg       0.56      0.67      0.52    246008
weighted avg       0.89      0.68      0.75    246008


Confusion Matrix:
[[154449  71699]
 [  6739  13121]]

Interpretacion:
  Verdaderos Negativos (TN): 154,449 - Predijo 'Pago' y si pago
  Falsos Positivos (FP): 71,699 - Predijo 'Default' pero pago
  Falsos Negativos (FN): 6,739 - Predijo 'Pago' pero hizo default [CRITICO]
  Verdaderos Positivos (TP): 13,121 - Predijo 'Default' y si hizo default

[11/15] Evaluando modelo en TEST SET (metricas definitivas)...
================================================================================
====

Metricas Globales:
ROC-AUC Score: 0.7335
Average Precision: 0.2112

Classification Report:
              precision    recall  f1-score   support

    Pago (0)       0.96      0.69      0.80     56538
 Default (1)       0.15      0.66      0.25      4965

    accuracy                           0.68     61503
   macro avg       0.56      0.67      0.52     61503
weighted avg       0.89      0.68      0.75     61503


Confusion Matrix:
[[38762 17776]
 [ 1710  3255]]

Interpretacion:
  Verdaderos Negativos (TN): 38,762 - Predijo 'Pago' y si pago
```

```
  Falsos Positivos (FP): 17,776 - Predijo 'Default' pero pago
  Falsos Negativos (FN): 1,710 - Predijo 'Pago' pero hizo default [CRITICO]
  Verdaderos Positivos (TP): 3,255 - Predijo 'Default' y si hizo default

[12/15] Comparando TRAIN vs TEST (analisis de overfitting)...
================================================================================
====

Metrica                   Train        Test         Diferencia
------------------------------------------------------------------
ROC-AUC                   0.7323       0.7335       0.0013
Average Precision         0.2049       0.2112       0.0063

Excelente generalizacion (diferencia < 2%)

[13/15] Calculando importancia de variables...
================================================================================
====

Top 20 variables que mas influyen en el riesgo de default:

                Variable  Coeficiente  Importancia_Abs
          SCORE_PROMEDIO       -0.685            0.685
    RATIO_PAGO_MINIMO_TC       -0.366            0.366
     NAME_EDUCATION_TYPE        0.172            0.172
             CODE_GENDER        0.166            0.166
          PLAZO_PROMEDIO        0.162            0.162
        CREDITOS_ACTIVOS        0.149            0.149
               EDAD_ANOS       -0.134            0.134
       CREDITOS_CERRADOS       -0.131            0.131
  TOTAL_CREDITO_OTORGADO       -0.112            0.112
   MONTO_PROMEDIO_PREVIO       -0.088            0.088
     CREDIT_INCOME_RATIO        0.077            0.077
      INGRESO_PER_CAPITA        0.069            0.069
             NUM_ACTIVOS       -0.065            0.065
 TOTAL_CREDITO_HISTORICO        0.056            0.056
TOTAL_CREDITO_DISPONIBLE       -0.050            0.050
          PCT_MESES_MORA        0.043            0.043
            CNT_CHILDREN        0.026            0.026
     CREDITOS_CON_IMPAGO        0.024            0.024
      TOTAL_DEUDA_ACTUAL        0.023            0.023
   NUM_PRESTAMOS_PREVIOS        0.008            0.008

Interpretacion de coeficientes:
  Coeficiente positivo (+): Aumenta probabilidad de DEFAULT
  Coeficiente negativo (-): Disminuye probabilidad de DEFAULT

[14/15] Generando visualizaciones...
Graficas guardadas: logistic_regression_curves.png

Tabla de TPR por Decil:
 decil  positivos  total   tpr
    10       1514   6151 0.246
     9        898   6150 0.146
     8        634   6150 0.103
     7        491   6150 0.080
```

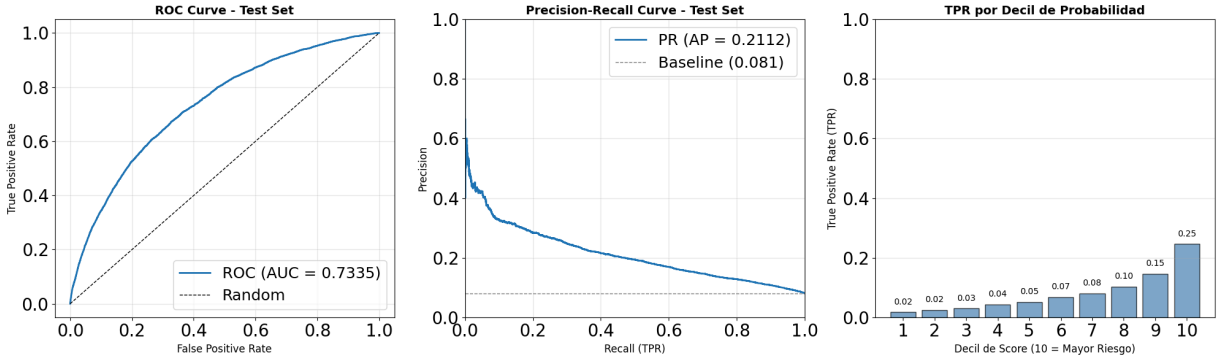
```
     6        414   6150 0.067
     5        312   6151 0.051
     4        266   6150 0.043
     3        183   6150 0.030
     2        148   6150 0.024
     1        105   6151 0.017

[15/15] Guardando modelo y resultados...
Archivos guardados:
  - logistic_model.pkl
  - scaler.pkl
  - label_encoders.pkl
  - feature_importance.csv
  - test_predictions.csv

================================================================================
====
ANALISIS COMPLETADO
================================================================================
====

Resultados Finales (TEST SET):
  ROC-AUC: 0.7335
  Average Precision: 0.2112
  Accuracy: 0.6832

El modelo esta listo para produccion
================================================================================
====
```



CORR

In [14]:

```python
# Matriz de correlación
df_numerico = df.select_dtypes(include=[np.number])
correlacion = df_numerico.corr()


# Ver las primeras filas
print(correlacion)

# O visualizarla con heatmap
import seaborn as sns
import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(20, 16))
sns.heatmap(correlacion, annot=False, cmap='coolwarm', center=0,
            linewidths=0.5, cbar_kws={"shrink": 0.8})
plt.title('Matriz de Correlación', fontsize=16)
plt.tight_layout()
plt.show()
```

	SK_ID_CURR	TARGET	EDAD_ANOS	SCORE_PROMEDIO
SK_ID_CURR	1.000	-0.002	0.001	0.001
TARGET	-0.002	1.000	-0.078	-0.222
EDAD_ANOS	0.001	-0.078	1.000	0.280
SCORE_PROMEDIO	0.001	-0.222	0.280	1.000
CREDIT_INCOME_RATIO	-0.002	-0.008	0.122	0.069
CNT_CHILDREN	-0.001	0.019	-0.331	-0.066
INGRESO_PER_CAPITA	0.000	-0.007	0.034	0.047
NUM_ACTIVOS	0.001	-0.020	-0.010	0.038
TOTAL_CREDITO_DISPONIBLE	-0.000	-0.011	0.012	0.001
TOTAL_CREDITO_OTORGADO	0.003	-0.018	-0.023	0.032
TOTAL_DEUDA_ACTUAL	-0.001	0.002	-0.064	-0.017
MAX_DIAS_MORA	0.001	0.004	0.014	-0.015
CREDITOS_ACTIVOS	0.001	0.044	0.009	-0.146
CREDITOS_CERRADOS	0.001	-0.037	0.087	0.042
PCT_MESES_MORA	-0.002	0.032	-0.019	-0.068
CREDITOS_CON_IMPAGO	0.000	0.021	-0.004	-0.069
NUM_PRESTAMOS_PREVIOS	0.003	0.024	0.072	-0.057
TASA_INTERES_PROMEDIO	0.037	-0.000	0.071	0.032
PLAZO_PROMEDIO	0.002	0.028	0.173	0.008
MONTO_PROMEDIO_PREVIO	0.001	-0.011	0.127	0.083
TOTAL_CREDITO_HISTORICO	0.002	0.011	0.118	0.012
RATIO_PAGO_CUOTA	-0.005	0.001	-0.001	-0.001
RATIO_PAGO_MINIMO_TC	-0.005	-0.002	-0.003	0.009

	CREDIT_INCOME_RATIO	CNT_CHILDREN
SK_ID_CURR	-0.002	-0.001
TARGET	-0.008	0.019
EDAD_ANOS	0.122	-0.331
SCORE_PROMEDIO	0.069	-0.066
CREDIT_INCOME_RATIO	1.000	-0.016
CNT_CHILDREN	-0.016	1.000
INGRESO_PER_CAPITA	-0.163	-0.232
NUM_ACTIVOS	-0.064	0.072
TOTAL_CREDITO_DISPONIBLE	-0.009	-0.001
TOTAL_CREDITO_OTORGADO	-0.057	0.035
TOTAL_DEUDA_ACTUAL	-0.063	0.046
MAX_DIAS_MORA	0.006	-0.002
CREDITOS_ACTIVOS	-0.056	0.016
CREDITOS_CERRADOS	-0.016	0.007
PCT_MESES_MORA	-0.025	0.003
CREDITOS_CON_IMPAGO	-0.041	0.006
NUM_PRESTAMOS_PREVIOS	-0.078	-0.019
TASA_INTERES_PROMEDIO	0.028	-0.034
PLAZO_PROMEDIO	0.010	-0.066
MONTO_PROMEDIO_PREVIO	-0.002	-0.044
TOTAL_CREDITO_HISTORICO	-0.036	-0.045
RATIO_PAGO_CUOTA	-0.001	0.002
RATIO_PAGO_MINIMO_TC	-0.001	0.003

	INGRESO_PER_CAPITA	NUM_ACTIVOS
SK_ID_CURR	0.000	0.001
TARGET	-0.007	-0.020
EDAD_ANOS	0.034	-0.010
SCORE_PROMEDIO	0.047	0.038
CREDIT_INCOME_RATIO	-0.163	-0.064

CNT_CHILDREN	-0.232	0.072
INGRESO_PER_CAPITA	1.000	0.027
NUM_ACTIVOS	0.027	1.000
TOTAL_CREDITO_DISPONIBLE	0.050	0.014
TOTAL_CREDITO_OTORGADO	0.136	0.091
TOTAL_DEUDA_ACTUAL	0.104	0.071
MAX_DIAS_MORA	-0.004	-0.003
CREDITOS_ACTIVOS	0.066	0.026
CREDITOS_CERRADOS	0.038	0.035
PCT_MESES_MORA	0.006	0.001
CREDITOS_CON_IMPAGO	0.025	0.015
NUM_PRESTAMOS_PREVIOS	0.027	0.046
TASA_INTERES_PROMEDIO	-0.000	-0.021
PLAZO_PROMEDIO	0.034	0.003
MONTO_PROMEDIO_PREVIO	0.105	0.071
TOTAL_CREDITO_HISTORICO	0.086	0.057
RATIO_PAGO_CUOTA	0.003	0.001
RATIO_PAGO_MINIMO_TC	-0.001	0.007

	TOTAL_CREDITO_DISPONIBLE	TOTAL_CREDITO_OTORGADO
SK_ID_CURR	-0.000	0.003
TARGET	-0.011	-0.018
EDAD_ANOS	0.012	-0.023
SCORE_PROMEDIO	0.001	0.032
CREDIT_INCOME_RATIO	-0.009	-0.057
CNT_CHILDREN	-0.001	0.035
INGRESO_PER_CAPITA	0.050	0.136
NUM_ACTIVOS	0.014	0.091
TOTAL_CREDITO_DISPONIBLE	1.000	0.119
TOTAL_CREDITO_OTORGADO	0.119	1.000
TOTAL_DEUDA_ACTUAL	0.044	0.582
MAX_DIAS_MORA	0.011	0.003
CREDITOS_ACTIVOS	0.255	0.307
CREDITOS_CERRADOS	0.155	0.309
PCT_MESES_MORA	-0.008	0.007
CREDITOS_CON_IMPAGO	0.020	0.095
NUM_PRESTAMOS_PREVIOS	0.033	0.034
TASA_INTERES_PROMEDIO	-0.005	-0.020
PLAZO_PROMEDIO	0.031	0.065
MONTO_PROMEDIO_PREVIO	0.051	0.131
TOTAL_CREDITO_HISTORICO	0.056	0.112
RATIO_PAGO_CUOTA	-0.002	0.004
RATIO_PAGO_MINIMO_TC	-0.001	-0.001

	TOTAL_DEUDA_ACTUAL	MAX_DIAS_MORA	CREDITOS_ACTIVOS
SK_ID_CURR	-0.001	0.001	0.001
TARGET	0.002	0.004	0.004
EDAD_ANOS	-0.064	0.014	0.009
SCORE_PROMEDIO	-0.017	-0.015	-0.146
CREDIT_INCOME_RATIO	-0.063	0.006	-0.05

6
CNT_CHILDREN   0.046   -0.002   0.01
6
INGRESO_PER_CAPITA   0.104   -0.004   0.06
6
NUM_ACTIVOS   0.071   -0.003   0.02
6
TOTAL_CREDITO_DISPONIBLE   0.044   0.011   0.25
5
TOTAL_CREDITO_OTORGADO   0.582   0.003   0.30
7
TOTAL_DEUDA_ACTUAL   1.000   -0.000   0.30
0
MAX_DIAS_MORA   -0.000   1.000   0.03
6
CREDITOS_ACTIVOS   0.300   0.036   1.00
0
CREDITOS_CERRADOS   0.159   0.020   0.45
6
PCT_MESES_MORA   0.005   0.008   0.01
9
CREDITOS_CON_IMPAGO   0.064   0.010   0.16
8
NUM_PRESTAMOS_PREVIOS   0.023   0.007   0.13
6
TASA_INTERES_PROMEDIO   -0.021   -0.003   -0.02
1
PLAZO_PROMEDIO   0.054   0.006   0.11
4
MONTO_PROMEDIO_PREVIO   0.096   0.000   0.07
6
TOTAL_CREDITO_HISTORICO   0.083   0.005   0.13
3
RATIO_PAGO_CUOTA   0.006   -0.001   0.00
1
RATIO_PAGO_MINIMO_TC   -0.002   -0.000   -0.00
5

                        CREDITOS_CERRADOS PCT_MESES_MORA \
SK_ID_CURR   0.001   -0.002
TARGET   -0.037   0.032
EDAD_ANOS   0.087   -0.019
SCORE_PROMEDIO   0.042   -0.068
CREDIT_INCOME_RATIO   -0.016   -0.025
CNT_CHILDREN   0.007   0.003
INGRESO_PER_CAPITA   0.038   0.006
NUM_ACTIVOS   0.035   0.001
TOTAL_CREDITO_DISPONIBLE   0.155   -0.008
TOTAL_CREDITO_OTORGADO   0.309   0.007
TOTAL_DEUDA_ACTUAL   0.159   0.005
MAX_DIAS_MORA   0.020   0.008
CREDITOS_ACTIVOS   0.456   0.019
CREDITOS_CERRADOS   1.000   -0.007
PCT_MESES_MORA   -0.007   1.000
CREDITOS_CON_IMPAGO   0.191   0.462
NUM_PRESTAMOS_PREVIOS   0.185   0.023

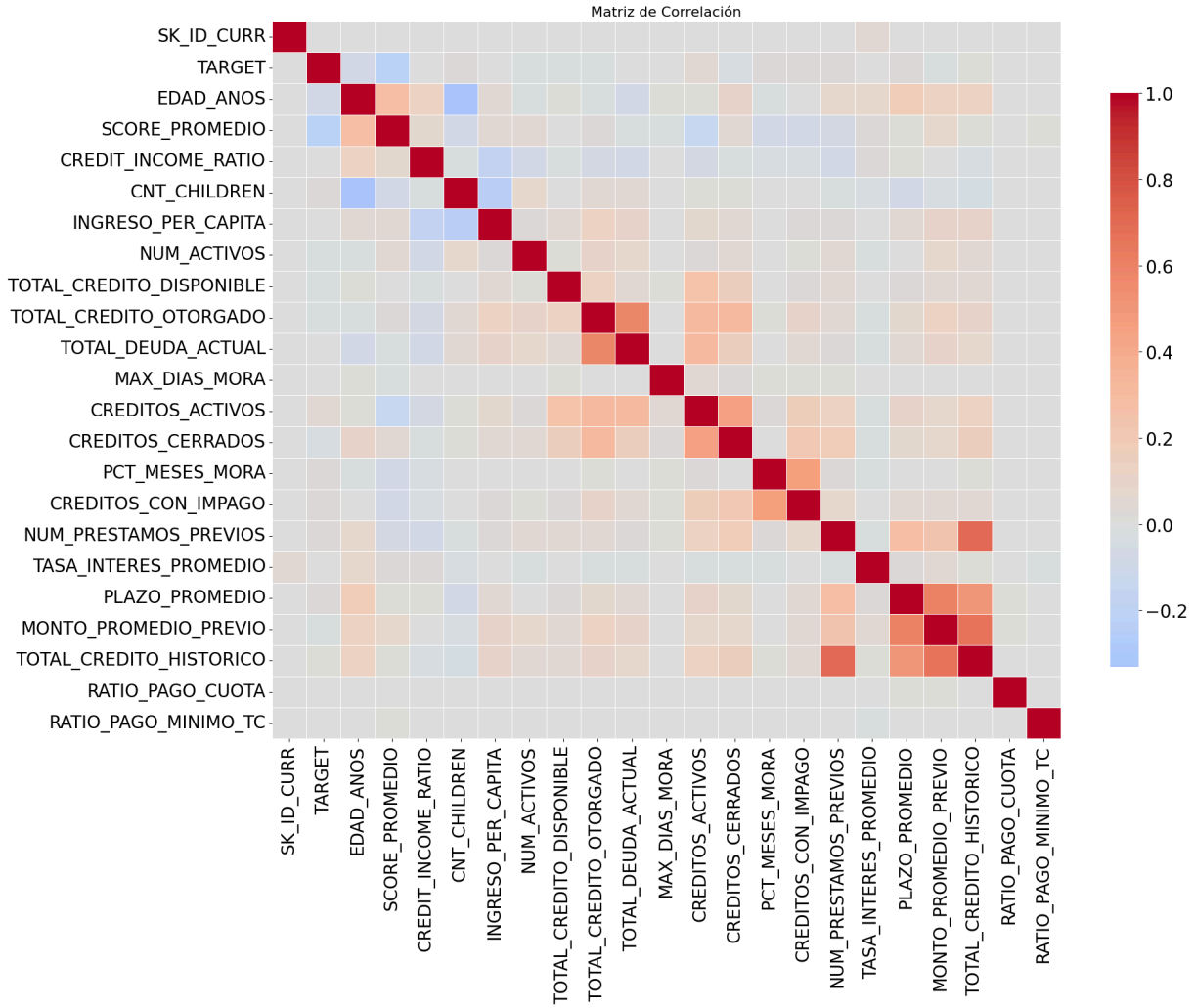
TASA_INTERES_PROMEDIO	-0.024	-0.012
PLAZO_PROMEDIO	0.064	0.006
MONTO_PROMEDIO_PREVIO	0.072	0.007
TOTAL_CREDITO_HISTORICO	0.148	0.014
RATIO_PAGO_CUOTA	0.000	-0.001
RATIO_PAGO_MINIMO_TC	-0.004	-0.001

	CREDITOS_CON_IMPAGO	NUM_PRESTAMOS_PREVIOS	\
SK_ID_CURR	0.000	0.003	
TARGET	0.021	0.024	
EDAD_ANOS	-0.004	0.072	
SCORE_PROMEDIO	-0.069	-0.057	
CREDIT_INCOME_RATIO	-0.041	-0.078	
CNT_CHILDREN	0.006	-0.019	
INGRESO_PER_CAPITA	0.025	0.027	
NUM_ACTIVOS	0.015	0.046	
TOTAL_CREDITO_DISPONIBLE	0.020	0.033	
TOTAL_CREDITO_OTORGADO	0.095	0.034	
TOTAL_DEUDA_ACTUAL	0.064	0.023	
MAX_DIAS_MORA	0.010	0.007	
CREDITOS_ACTIVOS	0.168	0.136	
CREDITOS_CERRADOS	0.191	0.185	
PCT_MESES_MORA	0.462	0.023	
CREDITOS_CON_IMPAGO	1.000	0.072	
NUM_PRESTAMOS_PREVIOS	0.072	1.000	
TASA_INTERES_PROMEDIO	-0.003	-0.012	
PLAZO_PROMEDIO	0.033	0.275	
MONTO_PROMEDIO_PREVIO	0.039	0.235	
TOTAL_CREDITO_HISTORICO	0.062	0.701	
RATIO_PAGO_CUOTA	0.000	0.003	
RATIO_PAGO_MINIMO_TC	-0.002	0.002	

	TASA_INTERES_PROMEDIO	PLAZO_PROMEDIO	\
SK_ID_CURR	0.037	0.002	
TARGET	-0.000	0.028	
EDAD_ANOS	0.071	0.173	
SCORE_PROMEDIO	0.032	0.008	
CREDIT_INCOME_RATIO	0.028	0.010	
CNT_CHILDREN	-0.034	-0.066	
INGRESO_PER_CAPITA	-0.000	0.034	
NUM_ACTIVOS	-0.021	0.003	
TOTAL_CREDITO_DISPONIBLE	-0.005	0.031	
TOTAL_CREDITO_OTORGADO	-0.020	0.065	
TOTAL_DEUDA_ACTUAL	-0.021	0.054	
MAX_DIAS_MORA	-0.003	0.006	
CREDITOS_ACTIVOS	-0.021	0.114	
CREDITOS_CERRADOS	-0.024	0.064	
PCT_MESES_MORA	-0.012	0.006	
CREDITOS_CON_IMPAGO	-0.003	0.033	
NUM_PRESTAMOS_PREVIOS	-0.012	0.275	
TASA_INTERES_PROMEDIO	1.000	0.019	
PLAZO_PROMEDIO	0.019	1.000	
MONTO_PROMEDIO_PREVIO	0.043	0.602	
TOTAL_CREDITO_HISTORICO	0.015	0.507	
RATIO_PAGO_CUOTA	-0.007	0.010	
RATIO_PAGO_MINIMO_TC	-0.023	-0.005	

	MONTO_PROMEDIO_PREVIO	TOTAL_CREDITO_HISTORICO	\
SK_ID_CURR	0.001	0.002	
TARGET	-0.011	0.011	
EDAD_ANOS	0.127	0.118	
SCORE_PROMEDIO	0.083	0.012	
CREDIT_INCOME_RATIO	-0.002	-0.036	
CNT_CHILDREN	-0.044	-0.045	
INGRESO_PER_CAPITA	0.105	0.086	
NUM_ACTIVOS	0.071	0.057	
TOTAL_CREDITO_DISPONIBLE	0.051	0.056	
TOTAL_CREDITO_OTORGADO	0.131	0.112	
TOTAL_DEUDA_ACTUAL	0.096	0.083	
MAX_DIAS_MORA	0.000	0.005	
CREDITOS_ACTIVOS	0.076	0.133	
CREDITOS_CERRADOS	0.072	0.148	
PCT_MESES_MORA	0.007	0.014	
CREDITOS_CON_IMPAGO	0.039	0.062	
NUM_PRESTAMOS_PREVIOS	0.235	0.701	
TASA_INTERES_PROMEDIO	0.043	0.015	
PLAZO_PROMEDIO	0.602	0.507	
MONTO_PROMEDIO_PREVIO	1.000	0.659	
TOTAL_CREDITO_HISTORICO	0.659	1.000	
RATIO_PAGO_CUOTA	0.012	0.006	
RATIO_PAGO_MINIMO_TC	-0.003	-0.002	

	RATIO_PAGO_CUOTA	RATIO_PAGO_MINIMO_TC
SK_ID_CURR	-0.005	-0.005
TARGET	0.001	-0.002
EDAD_ANOS	-0.001	-0.003
SCORE_PROMEDIO	-0.001	0.009
CREDIT_INCOME_RATIO	-0.001	-0.001
CNT_CHILDREN	0.002	0.003
INGRESO_PER_CAPITA	0.003	-0.001
NUM_ACTIVOS	0.001	0.007
TOTAL_CREDITO_DISPONIBLE	-0.002	-0.001
TOTAL_CREDITO_OTORGADO	0.004	-0.001
TOTAL_DEUDA_ACTUAL	0.006	-0.002
MAX_DIAS_MORA	-0.001	-0.000
CREDITOS_ACTIVOS	0.001	-0.005
CREDITOS_CERRADOS	0.000	-0.004
PCT_MESES_MORA	-0.001	-0.001
CREDITOS_CON_IMPAGO	0.000	-0.002
NUM_PRESTAMOS_PREVIOS	0.003	0.002
TASA_INTERES_PROMEDIO	-0.007	-0.023
PLAZO_PROMEDIO	0.010	-0.005
MONTO_PROMEDIO_PREVIO	0.012	-0.003
TOTAL_CREDITO_HISTORICO	0.006	-0.002
RATIO_PAGO_CUOTA	1.000	-0.000
RATIO_PAGO_MINIMO_TC	-0.000	1.000



In [15]:
```
# Top correlaciones (excluyendo diagonal)
mask = np.triu(np.ones_like(correlacion, dtype=bool))
corr_melted = correlacion.where(~mask).stack().reset_index()
corr_melted.columns = ['Variable_1', 'Variable_2', 'Correlación']
top_corr = corr_melted.reindex(corr_melted['Correlación'].abs().sort_values(
print(top_corr.head(20))
```

	Variable_1	Variable_2	Correlación
206	TOTAL_CREDITO_HISTORICO	NUM_PRESTAMOS_PREVIOS	0.701
209	TOTAL_CREDITO_HISTORICO	MONTO_PROMEDIO_PREVIO	0.659
189	MONTO_PROMEDIO_PREVIO	PLAZO_PROMEDIO	0.602
54	TOTAL_DEUDA_ACTUAL	TOTAL_CREDITO_OTORGADO	0.582
208	TOTAL_CREDITO_HISTORICO	PLAZO_PROMEDIO	0.507
119	CREDITOS_CON_IMPAGO	PCT_MESES_MORA	0.462
90	CREDITOS_CERRADOS	CREDITOS_ACTIVOS	0.456
12	CNT_CHILDREN	EDAD_ANOS	−0.331
87	CREDITOS_CERRADOS	TOTAL_CREDITO_OTORGADO	0.309
75	CREDITOS_ACTIVOS	TOTAL_CREDITO_OTORGADO	0.307
76	CREDITOS_ACTIVOS	TOTAL_DEUDA_ACTUAL	0.300
5	SCORE_PROMEDIO	EDAD_ANOS	0.280
169	PLAZO_PROMEDIO	NUM_PRESTAMOS_PREVIOS	0.275
74	CREDITOS_ACTIVOS	TOTAL_CREDITO_DISPONIBLE	0.255
187	MONTO_PROMEDIO_PREVIO	NUM_PRESTAMOS_PREVIOS	0.235
20	INGRESO_PER_CAPITA	CNT_CHILDREN	−0.232
4	SCORE_PROMEDIO	TARGET	−0.222
118	CREDITOS_CON_IMPAGO	CREDITOS_CERRADOS	0.191
133	NUM_PRESTAMOS_PREVIOS	CREDITOS_CERRADOS	0.185
155	PLAZO_PROMEDIO	EDAD_ANOS	0.173