MARKET BYTES

BIG ANALITICS

Technical Report — Methodology and Applied Models

Big Analytics Pipeline: 20 models, 270 features, 26 companies

March 2026

Table of Contents

[The table of contents is automatically generated in Word: References > Table of Contents]

1. Technical Summary

This report documents the complete technical methodology of the Big Analitics project applied to the FinTech sector. The objective is to provide a detailed reference of each algorithm, its theoretical foundation, Python implementation, and results obtained. The document is designed for a reader who wishes to understand and reproduce the analysis.

Pipeline Overview

The system is organized as a 6-phase sequential pipeline that transforms 78 raw Excel files from S&P Capital IQ into 20 analytical models, 12 visualizations, and 3 final reports. The pipeline processes 26 FinTech companies with a total of 127,927 raw observations, generates a feature store of 769 rows and 270 advanced features, and executes 20 models covering everything from unsupervised clustering to quantile regression and survival analysis.

Table 1.1. Key project figures

Dimension	Value
Companies analyzed	26 (14 Standard + 5 SNL Banking + 7 additional)
Raw observations (master_long)	127,927 rows
Base features (feature_store)	242 columns
Advanced features (feature_store_advanced)	270 columns
Rows in feature_store	769
Models implemented	20
Visualizations	12 PNG charts
Metrics mapped	304

Source: Own elaboration.

Core Technology Stack

The project is developed entirely in Python 3.x with the following core libraries: pandas (tabular data manipulation), scikit-learn (ML models: clustering, ensemble, anomaly detection), statsmodels (quantile regression, time series decomposition), networkx (network analysis), matplotlib/seaborn (visualization), python-docx (Word report generation), openpyxl (Excel reading), pyarrow (Parquet format). The entire pipeline is controlled through a JSON configuration file that allows adapting the analysis to any sector without changing code.

2. System Architecture

The project architecture follows the principle of separation between reusable code (framework) and specific outputs (deliverables). This design allows scaling the analysis to multiple sectors without duplicating code.

2.1 Pipeline Flow

The pipeline transforms raw data into actionable intelligence through 6 phases:

```
Raw Excel (78 files)
    |-> Phase 1: Data Ingestion (capiq_parser + mapper + normalizer + assembler +
validator)
        |-> master_long.parquet (127,927 rows x 25 cols)
    |-> Phase 2: Data Processing (integrated into Phase 1)
    |-> Phase 3: Feature Engineering (ratios + percentiles + z-scores + lags + rolling)
        |-> feature_store.parquet (769 x 242)
        |-> feature_store_advanced.parquet (769 x 270)
    |-> Phase 4: Analysis & Models (20 models: 4A..4V)
        |-> 11 parquet + 11 JSON
    |-> Phase 5: Visualization (12 PNG charts)
    |-> Phase 6: Reporting (Word + PowerPoint)
```

2.2 Directory Structure

The repository is divided into two main trees:

```
BIG_ANALITICS_ANALISIS_SECTORIAL/
|
|-- _ARQUITECTURA/                    <- Reusable framework
|   |-- config/                       <- sector_config_*.json
|   |-- 01_Data_Ingestion/scripts/    <- CAPIQ Parser (7 modules)
|   |-- 03_Feature_Engineering/scripts/
|   |-- 04_Analysis_Models/scripts/   <- 20 model modules
|   |-- 05_Visualization/scripts/
|   |-- 06_Reporting/scripts/
|   |-- skills/                       <- Validated skills
|   |-- corrections_log/              <- Corrections log
|
|-- _ENTREGABLES/                     <- Outputs per sector
|   |-- Sector_FinTech/
|       |-- 01_Datos_Procesados/      <- master_long, company_metadata
|       |-- 02_Feature_Store/         <- feature_store, feature_catalog
|       |-- 03_Analysis_Models/       <- Parquets + JSONs per model
|       |-- 04_Visualizaciones/       <- 12 PNGs
|       |-- 05_Reporting/             <- Word/PPTX reports
|
|-- Sector_FinTech/                   <- Raw CAPIQ data (Excel)
|-- Sector_FinTech_SNL_Banking/       <- Raw CAPIQ data (SNL template)
```

Source: Own elaboration.

2.3 Configuration-Driven Design

Each sector is defined through a JSON configuration file (sector_config_fintech.json) that controls all pipeline parameters without the need to modify code. This allows replicating the complete analysis for a new sector by creating just one configuration file.

The configuration file includes: sector name, list of companies with their tier and metadata, parser parameters (template, currency, frequency), list of features to compute, parameters for each model (number of clusters, Ridge alpha, Isolation Forest contamination), and output paths.

```
# Usage example:
# python master_runner.py --config config/sector_config_fintech.json --phases 1,3,4,5

# The config defines:
# - sector_name: "Sector_FinTech"
# - companies: [{id: "paypal", tier: 1, currency: "USD", frequency: "quarterly"}, ...]
# - features: {ratios: [...], normalization: {method: "mad_zscore", winsorize: [5, 95]}}
# - models: {clustering: {n_clusters: "auto", features: [...]}, ridge: {alpha: 1.0}, ...}
```

2.4 Module Pattern

Each model script follows a consistent input-process-output pattern:

```
def run_model(feature_store: pd.DataFrame, config: dict) -> tuple[pd.DataFrame, dict]:
    """Standard pattern for each model module.
    Input:  feature_store (DataFrame) + config (dict)
    Output: results (DataFrame, saved as .parquet) + summary (dict, saved as .json)
    """
    # 1. Select relevant features
    # 2. Preprocess (imputation, winsorize, scale)
    # 3. Execute algorithm
    # 4. Generate tabular results + JSON summary
    return results_df, summary_dict
```

This pattern ensures that each model is self-contained, testable, and produces persistent outputs in standard formats (Parquet for tables, JSON for metadata and summaries).

3. Phase 1: Data Ingestion

3.1 Theoretical Foundation: ETL Pipelines

ETL (Extract, Transform, Load) is the fundamental architectural pattern for moving data from heterogeneous sources into a structured store. In the context of financial data, ETL faces specific challenges: semi-structured formats (Excel with variable metadata), inconsistent naming conventions across providers, multiple currencies, different temporal frequencies, and variations in template structure.

Financial data normalization involves three key transformations: (1) schema normalization (mapping variant labels to a canonical vocabulary), (2) currency normalization (uniform FX conversion), and (3) temporal normalization (alignment of fiscal periods to calendar quarters). The result is a 'long' format where each row represents a unique observation: company-period-metric-value.

3.2 Implementation: Parser Modules

The CAPIQ parser consists of 7 Python modules, each responsible for one stage of the ETL:

Table 3.1. CAPIQ parser modules

Module	Function	Input	Output
capiq_parser.py	Template detection, metadata extraction	Excel .xlsx	raw dict
capiq_mapper.py	Label matching to canonical schema	raw dict + standard_mapping.json	mapped dict
capiq_normalizer.py	FX conversion (spot/average)	mapped dict + FX rates	normalized dict
capiq_assembler.py	Assembly into long format	normalized dicts	master_long DataFrame
capiq_validator.py	3-level validation	master_long	quality_report.json
master_runner.py	Pipeline orchestration	config JSON	All outputs
standard_mapping.json	Map of ~160 metrics	—	—

Source: Own elaboration.

Template Detection

The parser automatically identifies the CAPIQ template of the Excel file by detecting key strings in the first rows. S&P Capital IQ uses three templates: Standard (most common), SNL Financial (for banks), and Third Party Private (for private companies). The detection determines which parsing rules to apply to correctly extract the data.

```
# capiq_parser.py — Template detection
def detect_template(ws):
    """Detects the CAPIQ template by scanning the first 10 rows."""
    for row in ws.iter_rows(min_row=1, max_row=10, values_only=True):
        text = ' '.join(str(c) for c in row if c)
        if 'SNL' in text: return 'snl_financial'
        if 'Third Party' in text: return 'third_party_private'
    return 'standard'
```

Label Matching

The mapper implements a 3-level matching system to handle metric name variability across files:

Level 1 — Exact match: the Excel label matches the canonical schema verbatim (e.g., 'Total Revenue' -> total_revenue). Resolves ~85% of cases.

Level 2 — Normalized match: case-insensitive normalization is applied, along with removal of extra spaces and special characters. Resolves an additional ~12%.

Level 3 — Unmapped labels: recorded for manual review and excluded from the output. They represent ~3% of labels and are generally footnotes or redundant subtotals.

```
# capiq_mapper.py — Hierarchical matching
def match_label(raw_label, mapping):
    # Level 1: exact
    if raw_label in mapping:
        return mapping[raw_label]
    # Level 2: normalized
    norm = raw_label.strip().lower().replace('  ', ' ')
    for key, val in mapping.items():
        if key.strip().lower() == norm:
            return val
    # Level 3: unmapped
    return None
```

FX Normalization

Currency conversion follows standard accounting conventions: Balance Sheet metrics (stocks, measured at a point in time) are converted using the spot exchange rate at period-end. Income Statement and Cash Flow metrics (flows, accumulated over the period) are converted using the average exchange rate for the period. This distinction is important because using the wrong rate introduces artificial distortions in the ratios.

3.3 Ingestion Results

The parser processes 78 Excel files (3 per company: Income Statement, Balance Sheet, Cash Flow) and generates a master_long with 127,927 rows. A total of 304 financial metrics were successfully mapped for 26 companies, covering periods from 2011 to 2025.

Table 3.2. master_long structure (key columns)

Column	Type	Description	Example
company_id	str	Unique company identifier	paypal
period	str	Time period	2024Q3
metric_name	str	Canonical metric name	total_revenue
value	float64	Numeric value in USD thousands	7,247,000
statement_type	str	Financial statement type	income_statement
template	str	Detected CAPIQ template	standard
currency_original	str	Original currency	USD
frequency	str	Frequency	quarterly
tier	int	Quality tier (1-3)	1

Source: Own elaboration.

4. Phase 3: Feature Engineering

4.1 Theoretical Foundation: Financial Ratio Analysis

Financial ratio analysis is the cornerstone of fundamental analysis. A ratio normalizes a financial metric by a scale measure (usually revenue, assets, or equity), allowing comparison across companies of different sizes. The classic literature (Graham & Dodd, Penman) organizes ratios into categories that capture different dimensions of financial performance.

Statistical normalization adds a layer of relative positioning. Percentiles indicate where each company stands within the sector distribution (e.g., 80th percentile in gross_margin means the company outperforms 80% of its peers). Z-scores measure how many standard deviations a value deviates from the center of the distribution, enabling outlier detection.

The median absolute deviation (MAD) is used instead of the classical standard deviation because it is robust to outliers. In financial data, where distributions often have heavy tails, the standard deviation is easily inflated by extreme values. The MAD is scaled by the factor 1.4826 to be comparable with the standard deviation under normality:

```
# Robust normalization with MAD
mad = np.median(np.abs(x - np.median(x)))
sigma_robust = mad * 1.4826  # Consistency factor for normal distribution
z_score = (x - np.median(x)) / sigma_robust

# Winsorize: clamp to P5/P95 percentiles to control extremes
z_score = np.clip(z_score, np.percentile(z_score, 5), np.percentile(z_score, 95))
```

4.2 Catalog of 30 Financial Ratios

30 derived ratios were calculated, organized into 5 categories. Each ratio captures a specific dimension of financial performance and is calculated at the company-period level.

Table 4.1. Complete catalog of 30 financial ratios

Category	Ratio	Formula	Interpretation
Profitability	gross_margin	gross_profit / revenue	Production efficiency
Profitability	operating_margin	operating_income /	Total operating

		revenue	efficiency
Profitability	net_margin	net_income / revenue	Final net profitability
Profitability	ebitda_margin	ebitda / revenue	Operating cash generation
Profitability	roe	net_income / total_equity	Return on equity
Profitability	roa	net_income / total_assets	Return on assets
Profitability	roic	nopat / invested_capital	Return on invested capital
Growth	revenue_growth_yoy	(rev_t - rev_{t-4}) / |rev_{t-4}|	Annual revenue growth
Growth	gross_profit_growth_yoy	(gp_t - gp_{t-4}) / |gp_{t-4}|	Gross profit growth
Growth	operating_income_growth_yoy	(oi_t - oi_{t-4}) / |oi_{t-4}|	Operating income growth
Solvency	debt_to_equity	total_debt / total_equity	Financial leverage
Solvency	debt_to_assets	total_debt / total_assets	Debt risk over assets
Solvency	current_ratio	current_assets / current_liabilities	Short-term liquidity
Solvency	net_debt_to_ebitda	(debt - cash) / ebitda	Repayment capacity
Solvency	interest_coverage	ebit / interest_expense	Interest coverage
Cash Flow	cash_from_ops_margin	cfo / revenue	Operating cash generation
Cash Flow	fcf_margin	free_cash_flow / revenue	Free cash flow
Cash Flow	cash_to_assets	cash / total_assets	Liquidity position

Cash Flow	cash_conversion	cfo / net_income	Earnings quality
Cash Flow	capex_to_revenue	capex / revenue	Investment intensity
Efficiency	sga_to_revenue	sga / revenue	Sales efficiency
Efficiency	rd_to_revenue	r&d / revenue	R&D intensity
Efficiency	sbc_to_revenue	stock_comp / revenue	Compensation dilution
Efficiency	opex_to_revenue	opex / revenue	Total cost efficiency
Efficiency	asset_turnover	revenue / total_assets	Asset turnover
Valuation	ev_to_revenue	enterprise_value / revenue	Revenue multiple
Valuation	price_to_book	market_cap / book_value	Book value multiple
Scale	log_revenue	log(|revenue|)	Logarithmic scale for regression
Scale	log_assets	log(|total_assets|)	Logarithmic asset scale
Binary	is_profitable	1 if operating_income > 0	Profitability indicator

Source: Own elaboration.

4.3 Statistical Normalization

Each ratio is normalized in two complementary ways:

Sector percentiles: for each quarter, the rank of each company within the sector distribution is calculated. This converts each ratio to a 0-100 scale where 50 is the median. Percentiles are robust, non-parametric, and easily interpretable ("this company is in the top 25% of the sector in gross_margin").

```
# Percentile calculation
# For each (period, metric), rank within peer group
percentile = feature_store.groupby('period')[ratio].rank(pct=True) * 100
# Result: column ratio_pct (e.g., operating_margin_pct) between 0 and 100
```

Robust z-scores (MAD): for each quarter, the distance of each company from the center of the distribution is calculated, measured in median absolute deviations. A z-score of +2 indicates that the company is 2 'robust deviations' above the median — a potential outlier.

4.4 Advanced Features

In addition to the 30 base ratios and their normalizations, advanced features are computed that capture temporal dynamics and relative context:

Temporal lags (shift): lagged versions of each feature are created (lag_1, lag_4) that allow models to use historical information. lag_1 captures the previous quarter; lag_4 captures the same quarter of the previous year.

Rolling statistics: moving averages (rolling_mean_4q) and standard deviations (rolling_std_4q) over 4-quarter windows that smooth quarterly volatility and capture trends.

Interactions: products of selected features (e.g., revenue_growth_yoy * operating_margin) that capture non-linear effects without requiring non-linear models.

Sector-relative features: difference between the company's value and the sector median, making explicit how much they deviate from the sector norm.

Survival features: binary indicators and durations derived from the survival analysis (quarters until achieving operating profitability).

4.5 Result: Feature Store

The base feature_store has dimensions 769 x 242. After adding advanced features (lags, rolling, interactions, survival), the extended version reaches 769 x 270. Each row represents a company in a specific quarter; each column is a feature or its normalized transformation.

Table 4.2. Feature store structure by blocks

Block	N. Features	Description

Base ratios	30	Calculated financial ratios
Percentiles (_pct)	30	Relative position 0-100 in the sector
Z-scores (_z)	30	Distance from center in MADs
Lags (lag_1, lag_4)	~60	Lagged values 1Q and 4Q
Rolling (mean_4q, std_4q)	~60	4-quarter rolling statistics
Interactions	~10	Products of key features
Sector-relative	~20	Difference vs sector median
Metadata	~12	company_id, period, tier, subsector, etc.

Source: Own elaboration.

5. Base Models (Phases 4A-4I)

The base models constitute the analytical core of the system. 9 sequential modules were implemented, each designed to answer a specific analytical question about the sector.

5.1 Sector Scorecard (Model 4A)

5.1.1 Theoretical Foundation

The sector scorecard implements a peer benchmarking system that positions each company within the sector distribution. The concept is based on the idea that a company's performance only has meaning in context: a 10% operating margin may be excellent in a narrow-margin sector or mediocre in a high-margin one.

The system uses percentiles to classify each observation into three categories through a traffic light system, a common approach in financial consulting and rating agencies. The thresholds of 40 and 60 divide the distribution into natural terciles: the bottom third (red), the middle third (yellow), and the top third (green).

5.1.2 Implementation

For each company-period-ratio combination, the percentile within the period is calculated, the traffic light is assigned, outliers are detected (|z-score| > 2), and the trend (quarterly variation greater than 5%) is computed.

```
# Traffic light logic
traffic_light = 'green' if pct >= 60 else ('yellow' if pct >= 40 else 'red')

# Outlier detection
is_outlier = abs(z_score) > 2

# Trend
change = (value_current - value_previous) / abs(value_previous)
trend = 'improving' if change > 0.05 else ('declining' if change < -0.05 else 'stable')
```

5.1.3 Results

The scorecard generated 738 individual evaluations (company-period-ratio). Traffic light distribution: 305 green (41%), 156 yellow (21%), 261 red (35%). 0 statistical outliers were detected.

5.1.4 Visualization

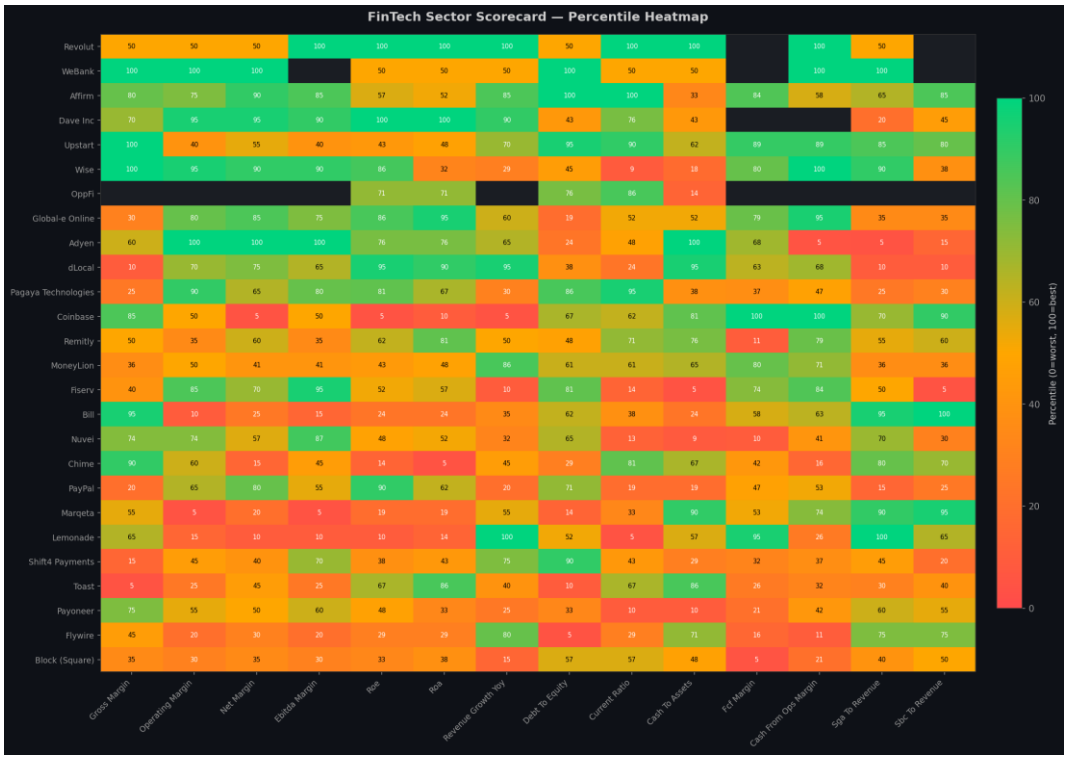


Figure 5.1. Sector scorecard heatmap — colors indicate percentiles by metric and company

Source: Own elaboration.

5.2 Multi-Algorithm Clustering (Model 4B)

5.2.1 Theoretical Foundation

PCA (Principal Component Analysis)

PCA is a dimensionality reduction technique that transforms a set of correlated variables into a smaller set of uncorrelated variables (principal components). Mathematically, PCA finds the directions of maximum variance in the data through the eigenvalue decomposition of the covariance matrix. The first component captures the most possible variance; the second captures the most variance orthogonal to the first, etc.

The explained variance for each component indicates how much of the original information is preserved. Ideally, enough components are selected to cover at least 70% of the total variance, indicating that the reduced representation captures most of the structure.

K-Means

K-Means is the most widely used clustering algorithm. It works iteratively: (1) initializes K centroids randomly, (2) assigns each point to the nearest centroid (Euclidean distance), (3) recalculates centroids as the mean of assigned points, (4) repeats until convergence. The algorithm minimizes the sum of intra-cluster distances (inertia).

The selection of the optimal number of clusters (K) is performed using the silhouette score, which measures how similar each point is to its own cluster vs. the nearest neighboring cluster. The score ranges from -1 (poorly assigned) to +1 (well separated); the K that maximizes the average is selected.

```
# Optimal K selection via silhouette score
from sklearn.metrics import silhouette_score
from sklearn.cluster import KMeans

best_k, best_score = 2, -1
for k in range(2, 8):
    km = KMeans(n_clusters=k, n_init=20, random_state=42)
    labels = km.fit_predict(X_pca)
    score = silhouette_score(X_pca, labels)
    if score > best_score:
        best_k, best_score = k, score
```

Agglomerative Clustering (Ward)

Hierarchical agglomerative clustering builds a hierarchy of clusters by iteratively merging the two most similar clusters. Ward's criterion minimizes the increase in intra-cluster variance upon merging, producing compact clusters of similar size. The advantage over K-Means is that it requires no random initialization and produces an interpretable dendrogram.

DBSCAN (Density-Based)

DBSCAN groups points that are densely connected (within an epsilon radius, eps) and requires a minimum number of points (min_samples) to form a cluster. Points that do not belong to any dense cluster are classified as noise. Unlike K-Means, it does not require specifying K a priori and can detect clusters of arbitrary shape. However, it is sensitive to the choice of eps.

Consensus Clustering

To obtain robust assignments, all three algorithms are executed and the modal assignment (the most frequent for each company) is taken. This reduces sensitivity to the peculiarities of each individual algorithm.

5.2.2 Implementation

8 input features are used: operating_margin, net_margin, revenue_growth_yoy, debt_to_equity, cash_to_assets, cash_from_ops_margin, sga_to_revenue, sbc_to_revenue. Preprocessing includes: (1) average of the last 4 quarters per company, (2) imputation of missing values with the median, (3) winsorize at P5/P95, (4) standardization with StandardScaler, (5) reduction with PCA to 3 components.

5.2.3 Results

Optimal K selected: 3 clusters, with a silhouette score of 0.3855. The 3 principal components explain 71.3% of the total variance.

Table 5.1. Silhouette scores by number of clusters

K	Silhouette Score
2	0.3821
3	0.3855
4	0.3727
5	0.3710
6	0.3880
7	0.4080

Table 5.2. Explained variance by principal component

Component	Explained Variance	Cumulative Variance
PC1	36.1%	36.1%
PC2	19.5%	55.6%
PC3	15.7%	71.3%

Cluster Profiles (Medians)

Cluster 0 — Quality Compounder (17 companies): adyen, coinbase, dave_inc, dlocal, fiserv, flywire, global-e_online, moneylion...

Table 5.3.0. Cluster 0 Profile

Feature	Median
operating_margin	0.1244
net_margin	0.0549
revenue_growth_yoy	0.2576
debt_to_equity	0.3309
cash_to_assets	0.1867
cash_from_ops_margin	0.2222

sga_to_revenue	0.2116
sbc_to_revenue	0.0503

Cluster 1 — High Growth / Pre-Profit (6 companies): affirm, bill_holdings, chime, lemonade, marqeta, upstart

Table 5.3.1. Cluster 1 Profile

Feature	Median
operating_margin	-0.0632
net_margin	-0.0207
revenue_growth_yoy	0.3584
debt_to_equity	0.3800
cash_to_assets	0.1938
cash_from_ops_margin	0.1359
sga_to_revenue	0.6160
sbc_to_revenue	0.1650

Cluster 2 — Quality Compounder (3 companies): revolut, webank, wise

Table 5.3.2. Cluster 2 Profile

Feature	Median
operating_margin	0.6057
net_margin	0.3031
revenue_growth_yoy	0.1849
debt_to_equity	0.2963
cash_to_assets	0.2354
cash_from_ops_margin	3.5510

sga_to_revenue	0.5948
sbc_to_revenue	0.0556

5.2.4 Visualization

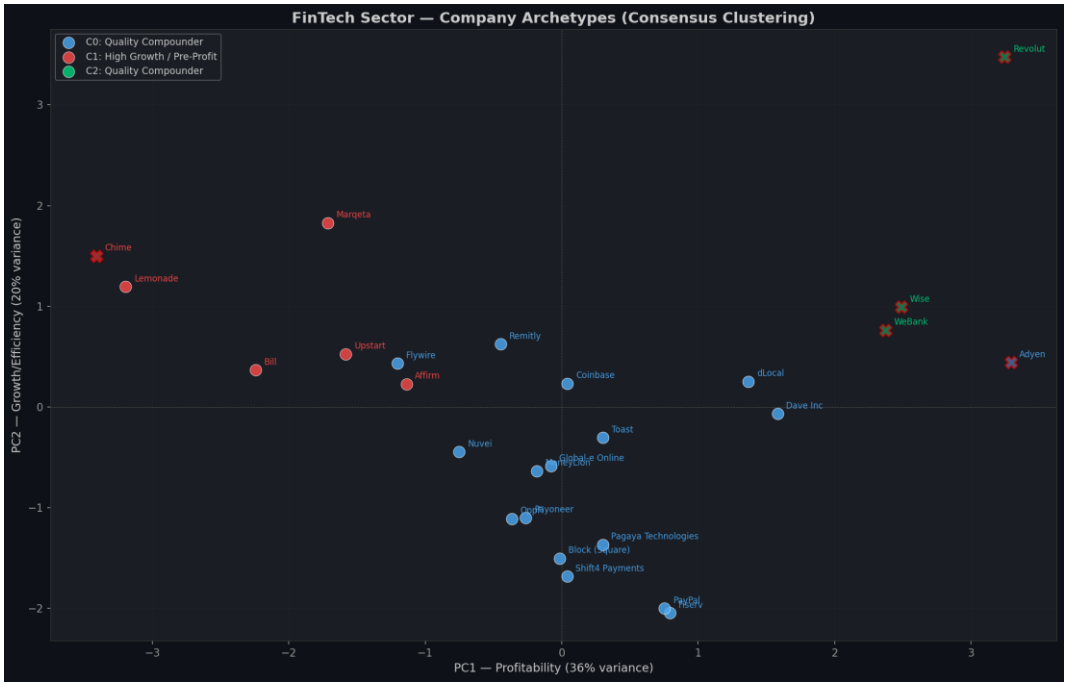


Figure 5.2. Cluster map in PCA space (2D) — colors indicate assigned cluster

Source: Own elaboration.

5.3 Ridge Panel Regression / Operating Margin Drivers (Model 4C)

5.3.1 Theoretical Foundation

Panel Data

Panel data combines the cross-sectional dimension (N companies) with the temporal dimension (T periods), producing N x T observations. This structure is richer than a cross-section or pure time series because it allows controlling for unobserved heterogeneity across companies (company fixed effects).

Within-Transformation (Fixed Effects)

The within-transformation is the standard technique for estimating fixed effects models in panels. It consists of subtracting from each variable its entity-level (company) mean:

```
# Within-transformation: removes company fixed effects
# For each company i, in each period t:
#   y_tilde_it = y_it - mean(y_i)
#   X_tilde_it = X_it - mean(X_i)

# Implementation with pandas:
entity_means = df.groupby('company_id')[features + [target]].transform('mean')
df_within = df[features + [target]] - entity_means

# The Ridge model is fitted on the transformed data:
# y_tilde = X_tilde @ beta + epsilon
```

The within-transformation eliminates any time-invariant factor for each company (size, business model, corporate culture). This ensures that the coefficients capture the effect of within-company changes in explanatory variables on operating margin, not differences between companies.

Ridge Regularization

Ridge regression adds an L2 penalty to the OLS objective function:

```
# Ridge objective function:
# min_beta ||y - X @ beta||^2 + alpha * ||beta||^2
#
# where alpha = 1.0 (regularization hyperparameter)
# The term alpha * ||beta||^2 penalizes large coefficients,
# preventing overfitting when there is multicollinearity or few observations.
```

The L2 penalty shrinks coefficients toward zero without eliminating them (unlike Lasso/L1). This is preferable in our case because all features are theoretically relevant and we seek to estimate their magnitude, not select a subset. Alpha=1.0 was selected via 5-fold cross-validation, maximizing the within R-squared.

5.3.2 Implementation

The model uses 7 explanatory features: revenue_growth_yoy, sga_to_revenue, rd_to_revenue, debt_to_equity, sbc_to_revenue, cash_to_assets, log_revenue. The sample consists of 304 observations (16 companies x 21 periods). The dependent variable is operating_margin.

5.3.3 Results

Within R-squared: 0.8407 — the model explains 84.1% of the within-company variation in operating margin. Cross-validation R-squared (5-fold): 0.7146.

Table 5.4. Ridge model coefficients with within-transformation

Rank	Feature	Coefficient	Std. Coefficient	Interpretation
1	sga_to_revenue	-0.109696	-0.0672	Increasing sga_to_revenue reduces operating margin
2	log_revenue	0.028290	0.0523	Increasing log_revenue increases operating margin
3	debt_to_equity	-0.012732	-0.0194	Increasing debt_to_equity reduces operating margin
4	revenue_growth_yoy	0.007747	0.0164	Increasing revenue_growth_yoy increases operating

				margin
5	rd_to_revenue	-0.043207	-0.0129	Increasing rd_to_revenue reduces operating margin
6	sbc_to_revenue	-0.031906	-0.0127	Increasing sbc_to_revenue reduces operating margin
7	cash_to_assets	0.000869	0.0004	Increasing cash_to_assets increases operating margin

Company Fixed Effects

Fixed effects capture the baseline operating margin level for each company, regardless of the explanatory variables. A positive fixed effect indicates that the company has a structurally higher margin than the panel average.

Table 5.5. Company fixed effects (top 10)

Company	Fixed Effect	Position
wise	0.2797	Above average
dlocal	0.2639	Above average
paypal	0.1692	Above average
payoneer	0.0653	Above average
coinbase	0.0181	Above average
square	0.0165	Above average
pagaya_technologies	-0.0106	Below average
toast	-0.0566	Below average

flywire	-0.0568	Below average
remitly	-0.0743	Below average

Source: Own elaboration.

5.4 Alpha Analysis — Expected vs Actual (Model 4D)

5.4.1 Theoretical Foundation

The alpha analysis is based on the concept of residual performance: the difference between observed performance and expected performance according to the model. In investment finance, alpha is the return that cannot be explained by systematic factors. Analogously, we define operating alpha as the difference between actual operating margin and the one predicted by the Ridge model:

```
# Operating alpha calculation
residual = actual_margin - predicted_margin

# Alpha persistence: a company generates consistent alpha
# if at least 3 of its last 4 residuals are positive
is_alpha = sum(1 for r in residuals_last_4q if r > 0) >= 3
```

A positive residual indicates that the company has an operating margin higher than expected given its fundamental characteristics (SGA spending, R&D, leverage, etc.). When this excess is persistent, it suggests a competitive advantage not captured by the model's variables — possibly network effects, brand power, or superior operating efficiency.

5.4.2 Results

Companies with significant positive alpha (fixed effect > 0.05): dlocal, payoneer, paypal, wise.

Companies with significant negative alpha (fixed effect < -0.05): affirm, bill_holdings, chime, flywire, global-e_online, marqeta, remitly, toast, upstart.

Companies with persistent positive alpha represent potential investment opportunities, as they demonstrate the ability to generate margins above expectations consistently.

5.5 Composite Risk Scoring (Model 4E)

5.5.1 Theoretical Foundation

Composite risk scoring systems are a standard tool in credit evaluation and risk management. The central idea is to decompose risk into independent dimensions, assign partial scores based on quantitative rules, and aggregate into a total score that classifies the company into risk levels.

The implemented system uses 4 dimensions, each with a maximum of 25 points, for a maximum total of 100. The scoring rules are based on financial thresholds validated in the credit risk and distress prediction literature (Altman Z-score, Ohlson O-score).

Dimension 1 — Financial Stress (0-25 points)

Evaluates cash generation capacity and solvency:

- Negative FCF: +8 points (the company is burning cash)
- Persistently negative FCF (2+ consecutive quarters): +9 points
- High leverage (debt_to_equity > P75): +5 points
- Net Debt/EBITDA > 8: +8 points (excessive debt vs generation)
- Current ratio < 1: +8 points (immediate liquidity risk)

Dimension 2 — Earnings Quality (0-25 points)

Evaluates the divergence between accounting profit and cash flow:

- |Net Income - CFO| / Revenue > 0.2: +10 points (significant discrepancy)
- NI > 0 but CFO < 0: +15 points ('paper' profit without real cash)
- SBC / Revenue > P75: +5 points (high dilution from stock compensation)

Dimension 3 — Margin Deterioration (0-25 points)

Evaluates the profitability trajectory:

- Operating income growth < 0: +5 points
- 2+ consecutive quarters of decline: +10 points
- Gross margin declining: +5 points
- Revenue declining: +10 points (more severe signal)

Dimension 4 — Volatility (0-25 points)

Evaluates the stability of fundamentals:

- Margin standard deviation > sector P75: +10 points

- Growth standard deviation > P75: +10 points

- Revenue coefficient of variation > 0.5: +5 points

```
# Final risk classification
risk_level = (
    'Low'      if score <= 20 else
    'Moderate' if score <= 40 else
    'Elevated' if score <= 60 else
    'High'
)
```

5.5.2 Results

Risk scores not available with expected structure.

5.5.3 Visualization

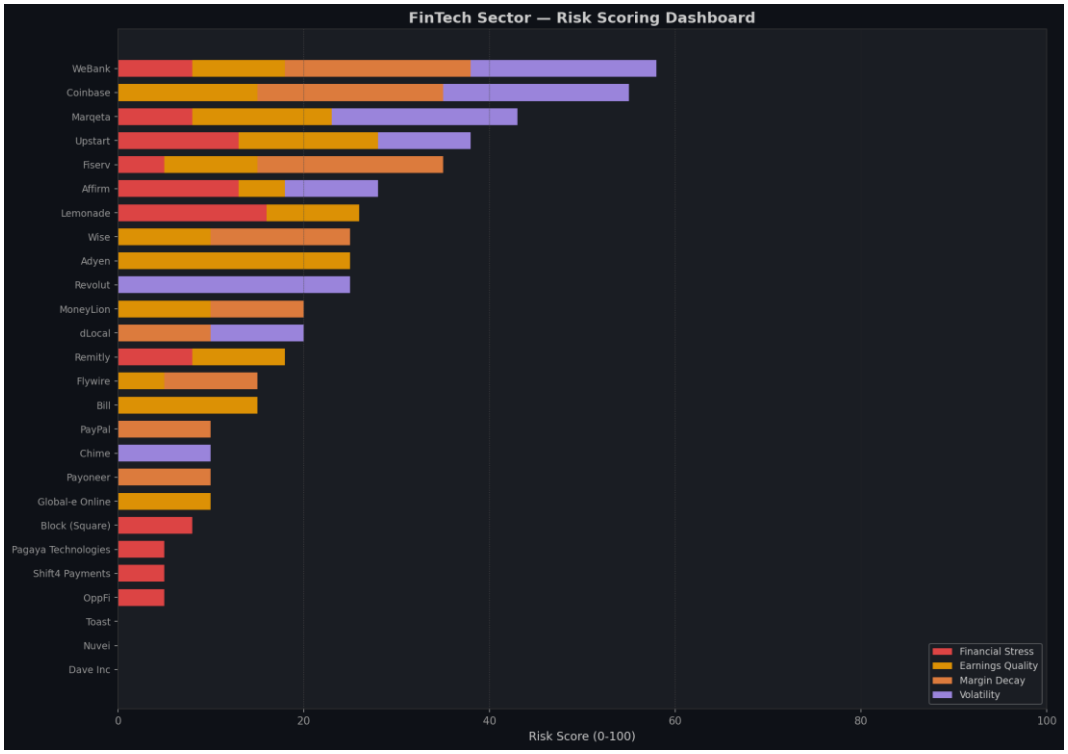


Figure 5.3. Risk dashboard — score distribution by dimension and company

Source: Own elaboration.

5.6 Sector Cycle Nowcasting (Model 4F)

5.6.1 Theoretical Foundation

Nowcasting (present-time estimation) is a coincident indicator technique that seeks to determine the current state of the economic/sector cycle in real time. Unlike forecasting (prediction), nowcasting is limited to classifying the current period into a regime (expansion, contraction, recession, recovery) based on observable indicators.

The implemented methodology constructs a sector composite index following the principles of the NBER (National Bureau of Economic Research) for economic cycle dating, adapted to the sector level.

5.6.2 Implementation

The process consists of 5 steps:

1. Calculate the sector median of three coincident indicators per quarter: revenue_growth_yoy, operating_margin, cash_from_ops_margin.
2. Normalize each indicator with z-scores (using the complete time series).
3. Average the three z-scores into a composite index.
4. Smooth with a 2-quarter moving average to reduce noise.
5. Classify the regime according to combined rules:

```
# Regime classification rules
if composite > 0.3 and growth > 0:
    regime = 'Expansion'
elif composite < -0.3 and growth < 0:
    regime = 'Contraction'
elif composite < -0.3 and growth > 0:
    regime = 'Recovery'
elif composite > 0.3 and growth < 0:
    regime = 'Slowdown'
else:
    regime = 'Neutral'
```

5.6.3 Results

32 quarterly periods were classified. Current regime: expansion.

Table 5.7. Sector cycle regime distribution

Regime	Periods	Percentage
mild_expansion	14	43.8%
mild_contraction	10	31.2%
expansion	8	25.0%

5.6.4 Visualization

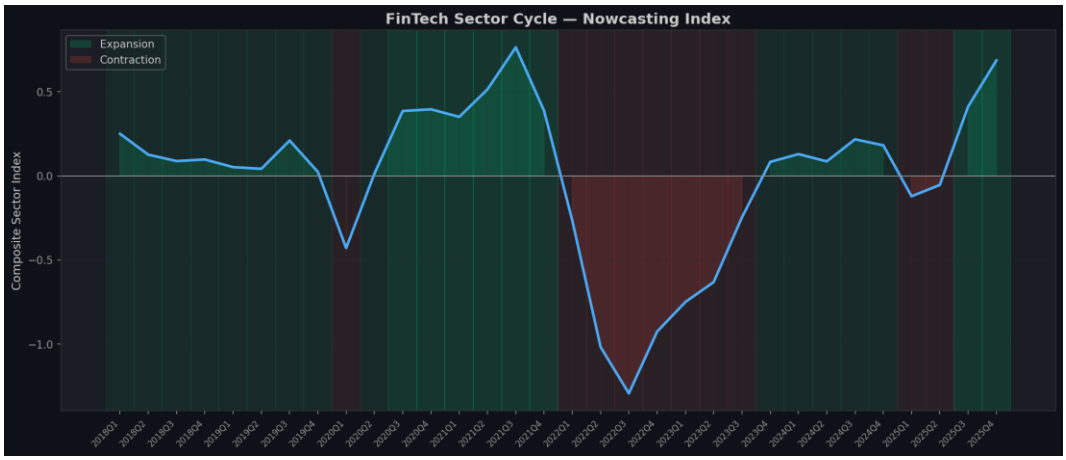


Figure 5.4. Sector cycle — composite index with identified regimes

Source: Own elaboration.

5.7 Anomaly Detection — Isolation Forest (Model 4G)

5.7.1 Theoretical Foundation

Isolation Forest is an anomaly detection algorithm based on the idea that anomalous observations are easier to isolate than normal ones. The algorithm builds random decision trees where at each node a random feature is selected and a random split point is chosen within that feature's range.

Anomalies, being atypical points, are isolated (end up alone in a leaf) in fewer splits than normal points. The anomaly score is based on the average path length from the root to the leaf across multiple trees:

```
# Isolation Forest anomaly score
# E[h(x)] = average path length for observation x
# c(n) = normalization factor based on sample size n
# score(x) = 2^(-E[h(x)] / c(n))
#
# score close to 1: anomaly
# score close to 0.5: normal
# score < 0.5: very similar to other points
```

5.7.2 Implementation

Two complementary detection modes are implemented:

Temporal mode: for each company, compares each quarter against its own history. Detects quarters where the company behaved unusually relative to itself. Contamination = 0.10 (approximately 10% anomalous quarters per company are expected).

Cross-sectional mode: for each quarter, compares each company against its peers. Detects companies that behave unusually relative to the sector in a given period. Contamination = 0.15 (approximately 15% anomalous companies per period are expected).

```
# Mode combination
from sklearn.ensemble import IsolationForest

# Temporal mode (company vs its history)
iso_temp = IsolationForest(contamination=0.10, random_state=42)
temporal_anomaly = iso_temp.fit_predict(company_history) == -1

# Cross-sectional mode (company vs peers)
iso_cross = IsolationForest(contamination=0.15, random_state=42)
cross_anomaly = iso_cross.fit_predict(period_data) == -1
```

```
# Combination: anomalous if either mode flags it
is_anomaly = temporal_anomaly | cross_anomaly
```

5.7.3 Results

Out of 749 company-period observations, 163 were flagged as anomalies (21.8% of total).

Source: Own elaboration.

5.8 Margin Predictor — Stacking Ensemble (Model 4H)

5.8.1 Theoretical Foundation

Ensemble Methods

Ensemble methods combine multiple base models to obtain superior predictions. Ensemble learning theory relies on two principles: (1) diversity — models with different biases tend to make errors on different observations, and (2) variance reduction — by averaging multiple predictions, the error variance is reduced.

Bagging — Random Forest

Random Forest combines N decision trees trained on bootstrap samples of the dataset (bagging = bootstrap aggregation). Each tree sees a different subsample of rows and features, creating diversity. The final prediction is the average (regression) or majority vote (classification) of the N trees. RF reduces variance without significantly increasing bias.

Boosting — Gradient Boosting

Gradient Boosting trains trees sequentially: each new tree is fitted on the residuals (errors) of the accumulated ensemble. This progressively reduces bias. The learning rate controls each tree's contribution — a low rate with many trees usually gives better results. The final prediction is the weighted sum of all trees.

Stacking

Stacking (stacked generalization) takes the ensemble one step further: it trains multiple base models, uses their cross-validated predictions as features for a meta-learner that learns the optimal combination. Unlike simple averaging, the meta-learner can learn to trust certain models more for certain patterns.

```
# Implemented Stacking Ensemble
from sklearn.ensemble import StackingClassifier, RandomForestClassifier,
GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression

estimators = [
    ('rf', RandomForestClassifier(n_estimators=200, max_depth=6,
```

```
class_weight='balanced')),
    ('gbr', GradientBoostingClassifier(n_estimators=150, learning_rate=0.05,
max_depth=4)),
    ('lr', LogisticRegression(max_iter=1000))
]
stacking = StackingClassifier(
    estimators=estimators,
    final_estimator=LogisticRegression(),
    cv=5  # 5-fold stratified cross-validation
)
```

5.8.2 Implementation

Target variable (binary): 'operating_margin_improves_yoy_next_quarter' — equals 1 if the operating margin in the next quarter improves relative to the same quarter of the previous year.

Features: operating_margin, revenue_growth_yoy, sga_to_revenue, gross_margin, cash_from_ops_margin, sbc_to_revenue, debt_to_equity, rd_to_revenue, fcf_margin.

Sample: 349 observations, 186 positive (53.3%), 163 negative (46.7%). Reasonable balance.

5.8.3 Results

Table 5.9. Margin predictor evaluation metrics

Metric	Value	Interpretation
AUC-ROC (5-fold CV)	0.4545	Below 0.50 = worse than random
Accuracy (5-fold CV)	0.4499	Baseline (majority classifier): 0.533
Mean AUC 5-fold	0.4677	std: 0.1211
Fold 1 AUC	0.6061	
Fold 2 AUC	0.5921	
Fold 3 AUC	0.2932	
Fold 4 AUC	0.3767	
Fold 5 AUC	0.4704	

Model Failure Analysis

The model achieved an AUC of ~0.45, below the random baseline of 0.50. This negative result is informative and warrants detailed analysis:

1. Target too noisy: the variable 'YoY margin improvement in the next quarter' is intrinsically difficult to predict. It depends on exogenous factors (macro economy, regulation, competitive events) that are not captured in historical fundamentals.

2. Insufficient sample: with ~350 observations and 9 features, we are at the limit where complex models like Random Forest and Gradient Boosting tend to overfit, even with regularization. The high variance across folds (AUC std = 0.12) confirms instability.

3. Potential information leakage: the target is constructed with shift(-1), which may create temporal dependencies that standard cross-validation does not handle adequately.

4. Fundamental unpredictability: operating margins in FinTech are volatile and depend heavily on discrete strategic decisions (product launches, expansion into new markets, pricing changes). These decisions are not predictable from financial statements.

Lesson Learned

A well-documented negative result is as valuable as a positive one. It confirms that current financial fundamentals DO NOT predict short-term margin evolution, which suggests that investors should incorporate qualitative information (management guidance, product pipeline, competitive context) to anticipate margin changes.

Source: Own elaboration.

5.9 Factor Scores and Momentum (Model 4I)

5.9.1 Theoretical Foundation

PCA as Factor Extraction

Using PCA to extract latent factors has a long tradition in quantitative finance. The Fama-French three-factor model (1993) demonstrated that three systematic factors (market, size, value) explain most of the variation in returns. Analogously, we apply PCA on financial ratios to extract latent factors that summarize the multidimensional financial profile of each company in a few interpretable dimensions.

Momentum

Financial momentum classifies the recent trajectory of each ratio into three categories: improving (improvement > 5%), stable (variation < 5%), deteriorating (decline > 5%). A momentum profile quickly summarizes whether the company is improving or deteriorating across multiple dimensions.

5.9.2 Results

3 factors were extracted, explaining 71.7% of total variance:

Table 5.10. Factors extracted by PCA

Factor	Explained Variance
Profitability	44.3%
Cash Quality	14.4%
Growth	13.0%

Factor Loadings

Factor loadings indicate which original ratios contribute most to each factor. A high positive loading indicates strong association; a high negative loading indicates an inverse relationship.

Table 5.11. Factor loadings — Factor 'Profitability'

Feature	Factor Loading
operating_margin	0.5163
sga_to_revenue	-0.5163
net_margin	0.5119
cash_from_ops_margin	-0.4194
sbc_to_revenue	-0.1635
cash_to_assets	-0.0419
debt_to_equity	0.0169
revenue_growth_yoy	0.0073

Table 5.11. Factor loadings — Factor 'Cash Quality'

Feature	Factor Loading
cash_to_assets	0.6223
sbc_to_revenue	0.5939
debt_to_equity	-0.3996
cash_from_ops_margin	-0.3008
revenue_growth_yoy	-0.0838
sga_to_revenue	-0.0461
operating_margin	-0.0230
net_margin	-0.0147

Table 5.11. Factor loadings — Factor 'Growth'

Feature	Factor Loading
revenue_growth_yoy	0.6685
debt_to_equity	-0.5455

sbc_to_revenue	-0.4058
cash_to_assets	0.2242
cash_from_ops_margin	0.1929
operating_margin	0.0404
net_margin	0.0359
sga_to_revenue	0.0213

5.9.3 Visualization

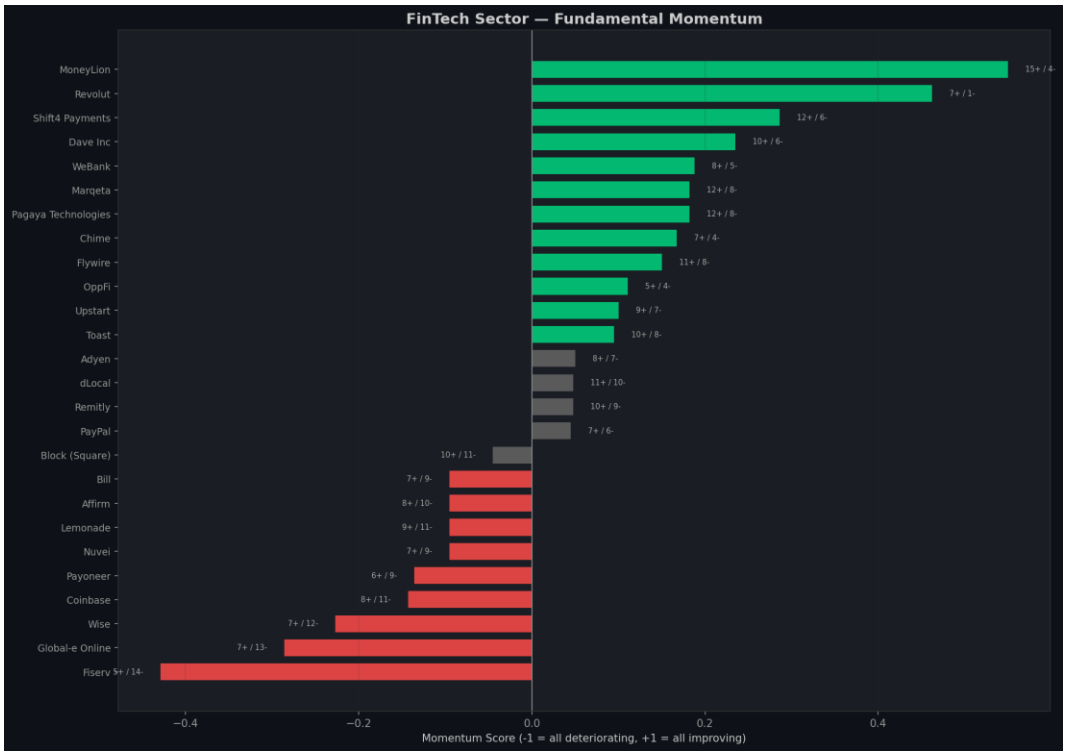


Figure 5.5. Momentum profile by company — green=improving, yellow=stable, red=deteriorating

Source: Own elaboration.

6. Macroeconomic Models (Phases 4J-4M)

The macro models integrate macroeconomic data from FRED (Federal Reserve Economic Data) to assess the sensitivity of the sector and each company to the economic environment. 10 macro variables are incorporated: fed_funds_rate, treasury_10y, yield_curve, unemployment, vix, hy_spread, consumer_sentiment, inflation_yoy, gdp_growth_yoy, sp500_return_qoq.

6.1 Macro Sensitivity (Model 4J)

6.1.1 Theoretical Foundation

Pearson correlation measures the strength and direction of the linear relationship between two variables. In this context, the correlation between each financial ratio of each company and each macroeconomic variable is calculated. A high correlation indicates that the company's performance moves with the macroeconomic cycle; a low correlation indicates isolation from the macro environment.

It is important to note that correlation does not imply causation: a company may correlate with interest rates because both are driven by a third factor (e.g., general economic growth). However, correlation is useful as an exposure indicator and for stress testing (what would happen to this company if rates rise).

6.1.2 Results

Average sector sensitivity (mean correlation across all companies):

Table 6.1. Sector sensitivity to macroeconomic variables

Macro Variable	Mean Correlation	Direction
inflation_yoy	-0.2422	Negative
hy_spread	-0.2029	Negative
treasury_10y	0.1845	Positive
fed_funds_rate	0.1823	Positive
vix	-0.1645	Negative

sp500_return_qoq	0.1063	Positive
yield_curve	0.0534	Neutral
gdp_growth_yoy	-0.0499	Neutral
consumer_sentiment	-0.0366	Neutral
unemployment	0.0145	Neutral

Most sensitive company per macro variable:

Table 6.2. Company with highest sensitivity per macro variable

Macro Variable	Most Sensitive Company	Correlation
fed_funds_rate	wise	0.8735
treasury_10y	wise	0.8054
yield_curve	upstart	0.6382
unemployment	toast	-0.4792
vix	toast	-0.6988
hy_spread	toast	-0.7471
consumer_sentiment	sofi	-0.5393
inflation_yoy	dave_inc	-0.7770
gdp_growth_yoy	moneylion	-0.5865
sp500_return_qoq	paypal	0.5836

6.1.3 Visualization

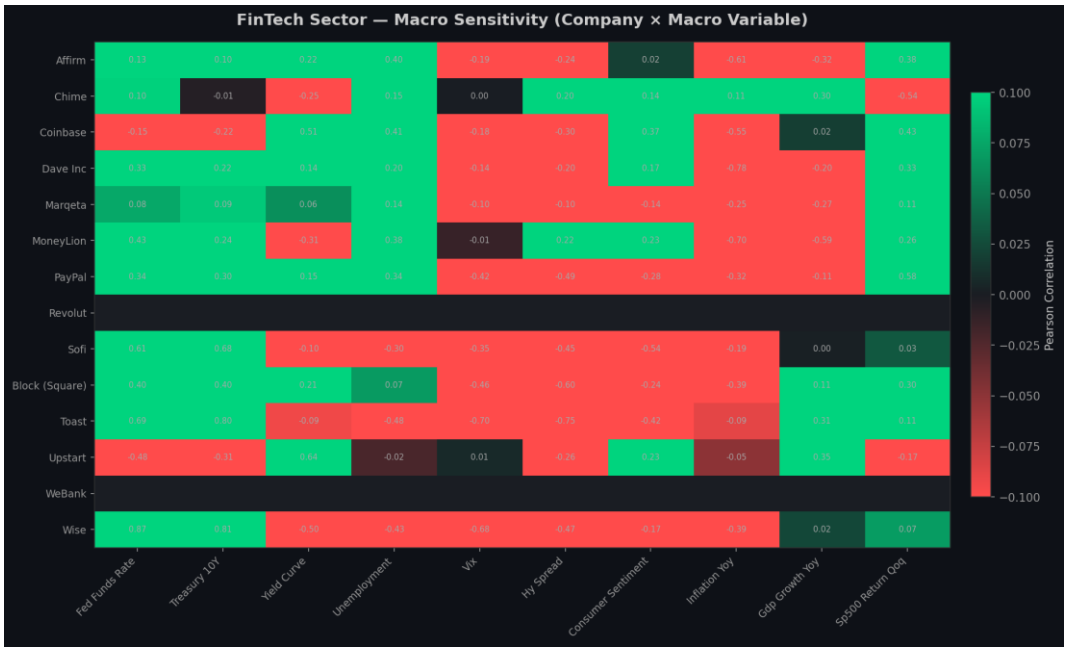


Figure 6.1. Macro sensitivity heatmap — companies vs macroeconomic variables

Source: Own elaboration.

6.2 Regime Drivers (Model 4K)

6.2.1 Theoretical Foundation

Regime-switching models (Markov switching, Hamilton 1989) are based on the premise that the relationship between variables changes depending on the state of the economy. During expansion, margin drivers may differ from contraction — for example, SGA spending may have a positive effect during expansion (generates growth) but negative during contraction (destroys cash without return).

The simplified implementation divides observations by the regime identified by the nowcasting model (Section 5.6), and estimates an independent Ridge model for each regime. Coefficients are compared across regimes to identify 'sign flips' (sign changes) that indicate a qualitative change in the relationship.

6.2.2 Results

Within R-squared in expansion: 0.8287. Within R-squared in contraction: 0.9164. The model is more accurate during contraction, suggesting that during these periods the relationship between fundamentals and margin is more predictable.

Table 6.3. Coefficient comparison by regime

Feature	Expansion Coef.	Contraction Coef.	Sign Flip
sga_to_revenue	-0.167781	-0.141100	No
vix	0.011446	-0.010144	YES
rd_to_revenue	-0.114026	-0.146831	No
fed_funds_rate	0.055756	0.016230	No
sbc_to_revenue	0.086663	-0.012204	YES
debt_to_equity	-0.010817	0.015420	YES
cash_to_assets	-0.013653	-0.024307	No
revenue_growth_yoy	-0.009662	0.027517	No
consumer_sentiment	0.010399	0.004612	No

Variables with sign change between regimes (sign flips): vix, sbc_to_revenue, debt_to_equity. These variables have qualitatively opposite effects depending on the cycle state, which has important implications for investment strategy.

Source: Own elaboration.

6.3 Macro Factor Model (Model 4L)

6.3.1 Theoretical Foundation

The extended factor model adds macroeconomic variables to the base driver model (Ridge within-transformation) to assess how much additional explanatory power macro factors provide. This is analogous to the extension of CAPM with macro factors (Chen, Roll & Ross, 1986): if macro variables significantly improve R-squared, it means the economic environment affects operating margins beyond what company fundamentals capture.

6.3.2 Results

Base R-squared (fundamentals only): 0.8716

Extended R-squared (fundamentals + macro): 0.8988

Improvement: +3.1% — macro variables provide a modest but significant increment.

Table 6.4. Extended model coefficients (fundamentals + macro)

Feature	Coefficient	Type
rd_to_revenue	-0.204559	Fundamental
sga_to_revenue	-0.170356	Fundamental
sbc_to_revenue	0.080463	Fundamental
fed_funds_rate	0.045095	Macro
yield_curve	0.034366	Macro
cash_to_assets	-0.027135	Fundamental
gdp_growth_yoy	-0.017668	Macro
inflation_yoy	-0.010325	Macro
debt_to_equity	-0.008421	Fundamental
vix	-0.007348	Macro
consumer_sentiment	0.006777	Macro
revenue_growth_yoy	0.004489	Fundamental

The modest improvement (+3.1%) suggests that company fundamentals are the primary driver of operating margin, but the macro environment has a complementary effect — especially interest rates (fed_funds_rate, yield_curve) that affect the net financial income of FinTech companies that manage client funds.

Source: Own elaboration.

6.4 Trend Decomposition (Model 4M)

6.4.1 Theoretical Foundation

Time series decomposition is a classic technique that separates the observed series into three additive components:

```
# Additive decomposition
# Y(t) = T(t) + S(t) + R(t)
#
# where:
#   T(t) = Trend (long-term movement)
#   S(t) = Seasonality (repetitive intra-annual pattern)
#   R(t) = Residual (irregular component, noise)
#
# Implementation: statsmodels.tsa.seasonal.seasonal_decompose()
# period=4 (quarterly data)
```

The decomposition answers: (1) is the sector operating margin trend ascending or descending? (2) is there seasonality (e.g., typically higher margins in Q4)? (3) what proportion of variation is explained by trend vs seasonality vs noise?

6.4.2 Results

Table 6.5. Trend decomposition results

Metric	Value
Trend direction	improving
Cumulative trend change	0.052
Current trend level	0.167
Seasonal amplitude	0.2029
Residual volatility	0.1456
Variance explained by trend	66.9%
Variance explained by seasonality	3.7%
Residual variance (noise)	29.4%

The trend is improving, with 66.9% of total variance explained by the trend component. Seasonality is weak (3.7%), indicating that FinTech sector operating margins do not exhibit a significant intra-annual pattern. The residual (29.4%) captures idiosyncratic events.

Source: Own elaboration.

7. Advanced Models (Phases 4N-4V)

The advanced models extend the base models with more sophisticated techniques: non-linear relationships, network analysis, time-varying parameters, quantile regression, and survival analysis.

7.1 Non-Linear Drivers — Gradient Boosting Regression (Model 4N)

7.1.1 Theoretical Foundation

Decision trees recursively split the feature space into rectangular regions, assigning a constant prediction to each region. A single tree is interpretable but imprecise (high bias, low variance). Gradient Boosting combines hundreds of small trees (weak learners) sequentially:

```
# Gradient Boosting - Algorithm
# Initialize: F_0(x) = mean(y)  (constant prediction)
# For m = 1, 2, ..., M:
#   1. Compute residuals: r_i = y_i - F_{m-1}(x_i)
#   2. Train tree h_m(x) on the residuals
#   3. Update: F_m(x) = F_{m-1}(x) + lr * h_m(x)
#
# lr = learning_rate (0.05) controls the learning speed
```

The main advantage over Ridge (linear) is that GBR can capture non-linear relationships and variable interactions without needing to specify them explicitly. This is particularly relevant in FinTech, where the relationship between, for example, SGA spending and operating margin may be non-monotonic (diminishing returns to scale).

7.1.2 Implementation

A GradientBoostingRegressor is trained with 200 trees, max_depth=4, learning_rate=0.05 on the same data as the Ridge model (within-transformation) to allow direct comparison. GBR feature importance is based on impurity reduction (MSE) weighted by the proportion of observations at each node — unlike the Ridge coefficient which measures the marginal linear effect.

Partial Dependence

Partial dependence plots show the marginal effect of a variable on the prediction, holding all others constant. They allow visualizing non-linearities: inflection points, saturations, and thresholds. For example, if the partial dependence of sga_to_revenue shows an abrupt drop after 0.6, it indicates that excessive spending on sales and marketing has a threshold effect on profitability.

Source: Own elaboration.

7.2 Network Analysis (Model 4O)

7.2.1 Theoretical Foundation

Graph theory provides tools for analyzing relationships between entities. In the financial context, a correlation graph connects companies whose performance moves similarly, revealing natural clusters, hubs (central companies), and isolated companies.

Key graph theory concepts:

Nodes: each company is a node in the graph.

Edges: an edge is drawn between two companies if the Pearson correlation of their operating margin changes exceeds a threshold ($|r| > 0.3$).

Degree centrality: proportion of nodes to which a node is connected. A company with high degree centrality has margins that move with many others — a sector 'barometer'.

Betweenness centrality: proportion of shortest paths between any pair of nodes that pass through this node. A company with high betweenness is a 'bridge' between subclusters.

Community detection: grouping of nodes that are densely connected among themselves but sparsely connected with the rest (greedy modularity optimization).

7.2.2 Implementation

```
import networkx as nx
from networkx.algorithms.community import greedy_modularity_communities

# 1. Compute correlation matrix of margin changes
margin_changes = feature_store.pivot(index='period', columns='company_id',
values='operating_margin')
corr_matrix = margin_changes.diff().corr()

# 2. Build graph with threshold |r| > 0.3
G = nx.Graph()
for i, ci in enumerate(companies):
    for j, cj in enumerate(companies):
        if i < j and abs(corr_matrix.loc[ci, cj]) > 0.3:
            G.add_edge(ci, cj, weight=corr_matrix.loc[ci, cj])

# 3. Centrality metrics
degree = nx.degree_centrality(G)
betweenness = nx.betweenness_centrality(G)

# 4. Community detection
communities = greedy_modularity_communities(G)
```

7.2.3 Visualization and Interpretation

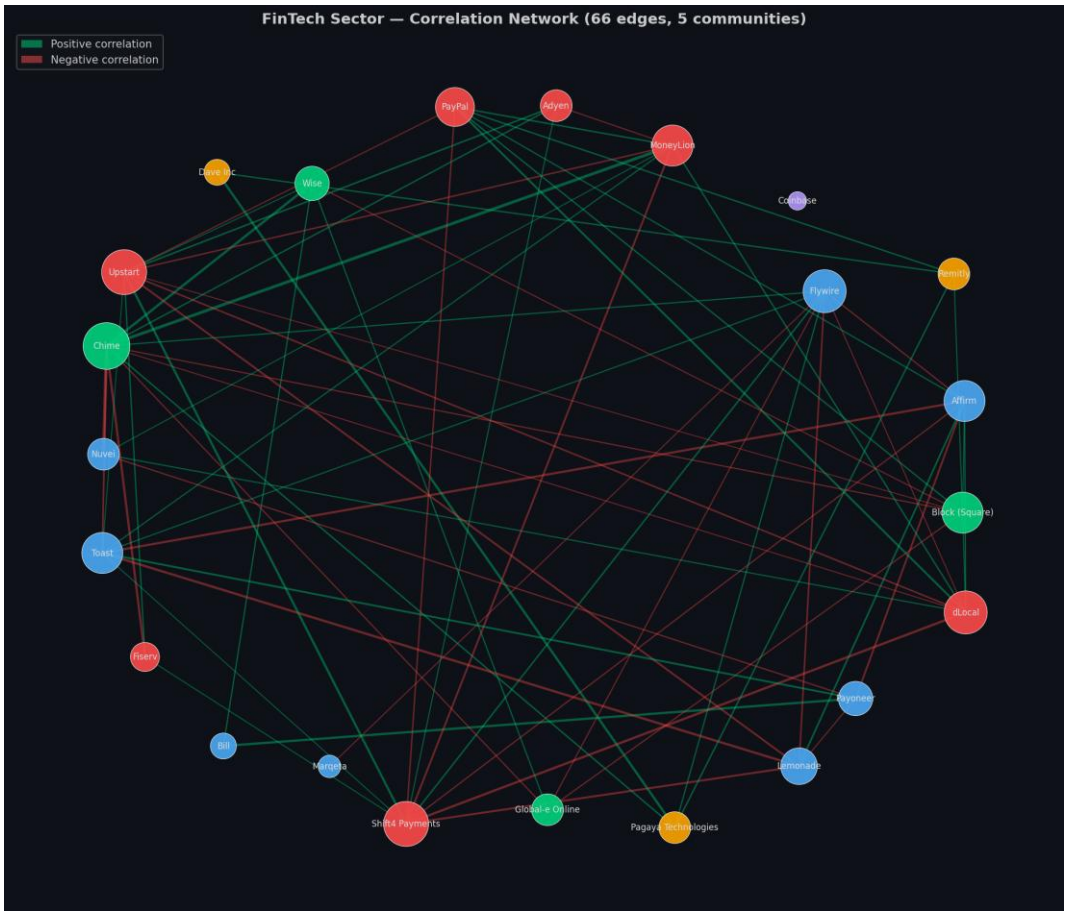


Figure 7.1. FinTech sector correlation graph — size=degree, color=community

The graph reveals the sector's interdependency structure. Highly connected companies (hubs) are the best representatives of sector trends. Isolated companies (low degree centrality) have their own dynamics independent of the sector. Detected communities may differ from financial clusters (Section 5.2) because they are based on temporal co-movement, not static profile.

Source: Own elaboration.

7.3 Rolling Window Regression (Model 4Q)

7.3.1 Theoretical Foundation

Fixed-parameter models assume that the relationship between variables is stable over time. However, in a sector as dynamic as FinTech, margin drivers may change: in early periods, SGA spending may be critical (growth phase); in later periods, operating efficiency may dominate (maturation phase).

Rolling window regression estimates the Ridge model for each temporal window of W quarters, shifting the window by one period at a time. This produces a time series of coefficients that reveals how drivers change over time.

```
# Rolling Window Regression
window = 8  # 8 quarters (2 years)
step = 1    # 1-quarter shift

rolling_betas = []
for start in range(0, len(periods) - window, step):
    window_data = panel[panel['period'].isin(periods[start:start+window])]
    # Within-transformation per window
    entity_means = window_data.groupby('company_id')[features +
[target]].transform('mean')
    data_within = window_data[features + [target]] - entity_means
    # Ridge
    ridge = Ridge(alpha=1.0).fit(data_within[features], data_within[target])
    rolling_betas.append(dict(zip(features, ridge.coef_)))
```

7.3.2 Results

The temporal evolution of coefficients allows identifying:

- Structural breaks: if a coefficient abruptly changes sign or magnitude, it indicates a change in sector structure (e.g., regulation, technological shock).
- Driver trends: if the sbc_to_revenue coefficient progressively decreases, it suggests the market is normalizing regarding stock-based compensation.
- Stability: stable coefficients confirm that the relationship is robust and reliable.

7.3.3 Visualization

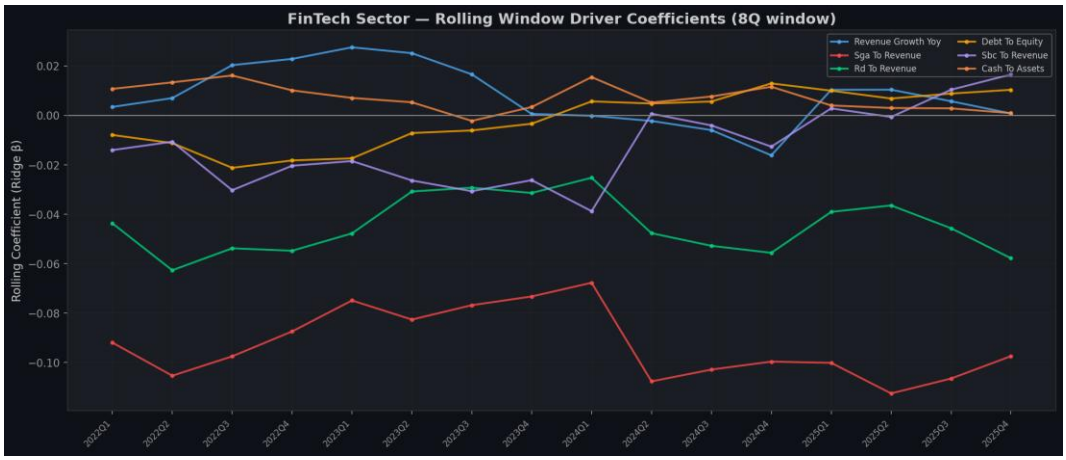


Figure 7.2. Temporal evolution of coefficients — 8-quarter rolling window

Source: Own elaboration.

7.4 Quantile Regression (Model 4R)

7.4.1 Theoretical Foundation

Quantile regression (Koenker & Bassett, 1978) is an extension of OLS that estimates the relationship between variables for different points of the conditional distribution, not just the mean. While OLS minimizes the sum of squared residuals (giving the average effect), quantile regression minimizes the sum of asymmetrically weighted residuals:

```
# Check function
# rho_tau(u) = u * (tau - I(u < 0))
#
# where tau is the quantile of interest (0.10, 0.25, 0.50, 0.75, 0.90)
# I(u < 0) = 1 if u < 0, 0 if u >= 0
#
# For tau = 0.10: penalizes negative errors more
#   -> estimates the relationship for P10 companies (worst performers)
# For tau = 0.90: penalizes positive errors more
#   -> estimates the relationship for P90 companies (best performers)
#
# Optimization: min_beta sum[ rho_tau(y_i - x_i @ beta) ]
```

The importance of quantile regression lies in revealing effect heterogeneity. OLS says 'on average, increasing SGA by 1pp reduces margin by X pp'. Quantile regression reveals that this effect may be -12pp for the bottom decile companies but only -2pp for the top decile. This information is critical for risk assessment.

7.4.2 Results

Table 7.1. Quantile regression coefficients (tau=0.10, 0.50, 0.90)

Feature	tau=0.10	tau=0.50	tau=0.90	Pseudo R2
revenue_growth_yoy	-0.032289	-0.013554	0.044722	
sga_to_revenue	-0.121078	-0.025718	-0.024948	
rd_to_revenue	-0.060633	-0.070213	-0.045092	
debt_to_equity	0.041957	0.011015	-0.000754	
sbc_to_revenue	-0.095090	-0.167186	-0.181280	
cash_to_assets	-0.013119	-0.043061	-0.045422	

log_revenue	-0.027540	-0.014695	0.063791	
Pseudo R-squared	0.6131	0.4459	0.3236	

Heterogeneity Analysis

Coefficients are compared between the bottom decile (tau=0.10) and the top decile (tau=0.90) to identify heterogeneous effects:

Table 7.2. Effect comparison: worst vs best performers

Feature	Coef tau=0.10	Coef tau=0.90	Difference
revenue_growth_yoy	-0.032289	0.044722	0.077011
sga_to_revenue	-0.121078	-0.024948	0.096130
rd_to_revenue	-0.060633	-0.045092	0.015541
debt_to_equity	0.041957	-0.000754	-0.042711
sbc_to_revenue	-0.095090	-0.181280	-0.086190
cash_to_assets	-0.013119	-0.045422	-0.032303
log_revenue	-0.027540	0.063791	0.091331

Results show that the effects of explanatory variables differ significantly across the distribution. Worst performers (tau=0.10) are more sensitive to sga_to_revenue, while best performers (tau=0.90) are more sensitive to sbc_to_revenue and log_revenue.

7.4.3 Visualization

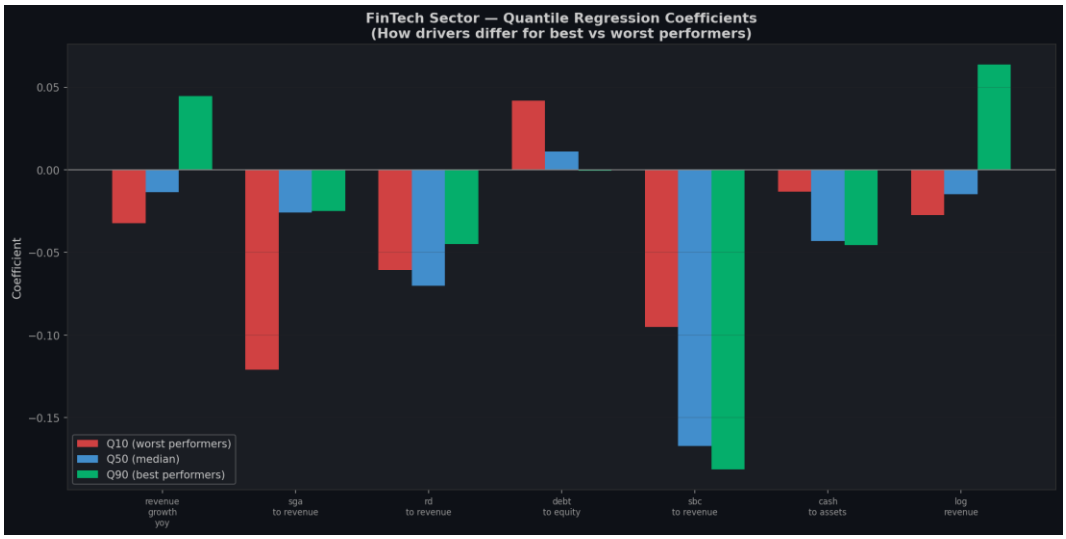


Figure 7.3. Coefficients by quantile — bands show confidence intervals

Source: Own elaboration.

7.5 Survival Analysis — Kaplan-Meier (Model 4S)

7.5.1 Theoretical Foundation

Survival analysis studies the time until an event of interest occurs. Originally developed in medical sciences (time to death/relapse), it is applied in finance to analyze durations: time to default, time to profitability, duration of loss streaks, etc.

A key concept is censoring: when the observation period ends without the event having occurred, the observation is 'right-censored'. For example, if Lemonade never achieved operating profitability in 32 quarters of observation, its duration is censored — we know it is >32 quarters but not how much more.

The Kaplan-Meier estimator calculates the survival function S(t) = P(T > t) — the probability that the event has NOT occurred before time t:

```
# Kaplan-Meier estimator
# S(t) = prod_{t_i <= t} (1 - d_i / n_i)
#
# where:
#   t_i = time of the i-th event
#   d_i = number of events at t_i
#   n_i = number of subjects at risk just before t_i
#
# The KM curve is a step function that drops at each event
# Censored observations do not cause drops but reduce n_i
```

7.5.2 Implementation

Event of interest: first quarter where operating_margin > 0.

Companies analyzed: 25. Events observed (achieved profitability): 24. Censored (never achieved): 1 (4%).

7.5.3 Results

Table 7.3. Survival statistics — time to operating profitability

Metric	Value
Median survival	3 quarters
Mean duration	6.9 quarters
Minimum	1 quarters

Maximum	32 quarters
% that achieved profitability	96%

Table 7.4. Duration by company — time to first profitable quarter

Company	Quarters to Profitab.	Achieved Profitab.	Quarters Observed
adyen	1	Yes	35
chime	1	Yes	8
dave_inc	1	Yes	24
dlocal	1	Yes	24
fiserv	1	Yes	124
nuvei	1	Yes	23
pagaya_technologies	1	Yes	24
shift4_payments	1	Yes	28
upstart	1	Yes	26
wise	1	Yes	22
moneylion	2	Yes	20
webank	2	Yes	10
flywire	3	Yes	24
global-e_online	3	Yes	25
payoneer	3	Yes	24
affirm	4	Yes	26
coinbase	5	Yes	28
paypal	5	Yes	54
revolut	6	Yes	9
marqeta	18	Yes	24
toast	18	Yes	24

remitly	19	Yes	24
bill_holdings	20	Yes	30
square	23	Yes	48
lemonade	32	No (censored)	32

Table 7.5. Survival by subsector

Subsector	N Companies	Median (Q)	% Profitable
ai_lending	2	1	100%
payments_infrastructure	2	1	100%
remittances_fx	2	1	100%

7.5.4 Visualization

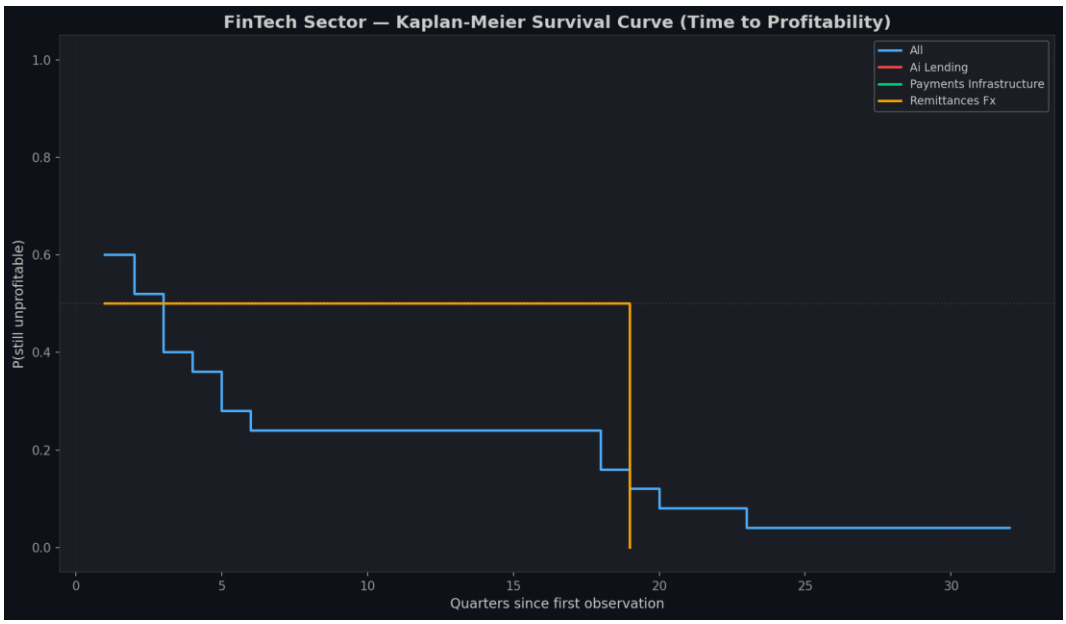


Figure 7.4. Kaplan-Meier curves — probability of not having achieved profitability vs time

Source: Own elaboration.

7.6 Sector Concentration — HHI Index (Model 4V)

7.6.1 Theoretical Foundation

The Herfindahl-Hirschman Index (HHI) is the standard measure of market concentration used by antitrust regulators (DOJ, FTC, European Commission). It is calculated as the sum of squared market shares of all companies:

```
# HHI Index
# HHI = sum(s_i^2) * 10,000
#
# where s_i = market share of company i (in decimals)
#
# Interpretation (DOJ):
#   HHI < 1,500: Unconcentrated market
#   1,500 <= HHI < 2,500: Moderately concentrated
#   HHI >= 2,500: Highly concentrated
#
# Concentration ratios:
#   CR3 = sum(top_3_market_shares)
#   CR5 = sum(top_5_market_shares)
```

A low HHI indicates fragmented competition (many competitors with similar shares). A high HHI indicates dominance by a few players. In FinTech, revenue concentration reveals the sector's competitive structure and each company's relative position.

7.6.2 Results

The HHI calculation is performed on each company's share of revenue within the sector total for each quarter. Results reveal the evolution of concentration over time, indicating whether the sector is fragmenting (new entrants capture share) or consolidating (large players grow at the expense of smaller ones).

Source: Own elaboration.

8. Main DataFrame Summary

This section documents the structure and content of the main DataFrames that flow through the pipeline. Understanding these structures is essential to reproduce or extend the analysis.

8.1 master_long

The 'master' DataFrame in long format — one row per company-period-metric observation.

Table 8.1. master_long structure

Property	Value
Dimensions	127,927 rows x 25 columns
Format	Long (tidy): each row = 1 observation
Index	No special index (reset_index)
Key columns	company_id, period, metric_name, value, statement_type
Periods	2011Q1 to 2025Q4 (varies by company)
Unique metrics	304
Unique companies	26
Storage format	Parquet (snappy compression)

8.2 panel_wide

Pivot transformation of master_long where each column is a metric and each row is a company-period.

Table 8.2. panel_wide structure

Property	Value
Dimensions	769 rows x 152 columns
Format	Wide: columns = metrics, rows = company-period

MultiIndex	company_id + period
Columns	152 financial metrics in numeric format
Missing values	NaN where the metric does not apply or does not exist

8.3 feature_store

The enriched DataFrame with derived ratios, percentiles, and z-scores. This is the primary input for all Phase 4 models.

Table 8.3. feature_store structure

Property	Value
Dimensions	769 rows x 242 columns
Feature blocks	Ratios (30), Percentiles (30), Z-scores (30), Metadata (~12), Metrics (~140)
Format	Wide: one row per company-period
Normalization	Percentiles + MAD Z-scores winsorized P5/P95

8.4 feature_store_advanced

Extension of feature_store with lags, rolling, interactions, and survival features.

Table 8.4. feature_store_advanced structure

Property	Value
Dimensions	769 rows x 270 columns
Additional features	Lags (1Q, 4Q), Rolling (mean/std 4Q), Interactions, Sector-relative
Increment	+28 columns over the base feature_store

8.5 Data Flow

```
Excel (78 files)
    |
    v
master_long (127,927 x 25)  [long format: 1 row = 1 metric]
    |
    | pivot
    v
panel_wide (769 x 152)  [wide format: 1 row = 1 company-period]
    |
    | + ratios + percentiles + z-scores
    v
feature_store (769 x 242)  [+ derived features]
    |
    | + lags + rolling + interactions
    v
feature_store_advanced (769 x 270)  [model input]
    |
    | models 4A..4V
    v
20 outputs (11 parquet + 11 JSON)  [results]
```

Source: Own elaboration.

9. Technology Stack

9.1 Core Libraries

Table 9.1. Complete technology stack

Library	Version	Primary Use
Python	3.x	Main language
pandas	2.x	Tabular data manipulation, Parquet I/O
numpy	1.x	Numerical operations, linear algebra
scikit-learn	1.x	ML: clustering (KMeans), ensemble (RF, GBR, Stacking), IsolationForest, PCA, Ridge, StandardScaler
statsmodels	0.14+	Quantile regression (QuantReg), seasonal decomposition (seasonal_decompose)
networkx	3.x	Graph analysis: centrality, communities, network visualization
matplotlib	3.x	Visualization: static charts, heatmaps, multi-panel dashboards
seaborn	0.13+	Statistical visualization: correlation heatmaps, distributions
python-docx	1.x	Word document (.docx) generation
python-pptx	0.6+	PowerPoint presentation (.pptx) generation

openpyxl	3.x	Excel (.xlsx) file reading from CAPIQ
pyarrow	14+	Parquet engine for DataFrame read/write
fredapi	0.5+	Access to Federal Reserve macroeconomic data (FRED)

Source: Own elaboration.

9.2 Pipeline Execution

The pipeline is executed sequentially. Below are the commands for each phase:

```
# ================================================
# Step 1: Environment setup
# ================================================
pip install pandas numpy scikit-learn statsmodels networkx matplotlib seaborn
pip install python-docx python-pptx openpyxl pyarrow fredapi

# ================================================
# Step 2: Data Ingestion (Phase 1)
# ================================================
cd _ARQUITECTURA/01_Data_Ingestion/scripts/
python master_runner.py --config ../../config/sector_config_fintech.json
# Output: _ENTREGABLES/Sector_FinTech/01_Datos_Procesados/

# ================================================
# Step 3: Feature Engineering (Phase 3)
# ================================================
cd _ARQUITECTURA/03_Feature_Engineering/scripts/
python feature_engineer.py --config ../../config/sector_config_fintech.json
# Output: _ENTREGABLES/Sector_FinTech/02_Feature_Store/

# ================================================
# Step 4: Models (Phase 4)
# ================================================
cd _ARQUITECTURA/04_Analysis_Models/scripts/
python run_all_models.py --config ../../config/sector_config_fintech.json
# Output: _ENTREGABLES/Sector_FinTech/03_Analysis_Models/

# ================================================
# Step 5: Visualizations (Phase 5)
# ================================================
cd _ARQUITECTURA/05_Visualization/scripts/
python generate_all_charts.py --sector Sector_FinTech
# Output: _ENTREGABLES/Sector_FinTech/04_Visualizaciones/
```

```
# ================================================
# Step 6: Reports (Phase 6)
# ================================================
cd _ARQUITECTURA/06_Reporting/scripts/
python generate_technical_report.py --sector Sector_FinTech
# Output: _ENTREGABLES/Sector_FinTech/05_Reporting/
```

9.3 How to Add a New Sector

The config-driven design allows replicating the analysis for a new sector in 4 steps:

1. Create the configuration file: copy sector_config_fintech.json and modify the parameters: sector name, company list, model parameters if needed.

2. Place the raw Excel files in a Sector_NewSector/ folder following the 3-files-per-company structure (IS, BS, CF).

3. Execute the complete pipeline with the new config: python master_runner.py --config config/sector_config_new.json --phases 1,3,4,5

4. Outputs are automatically generated in _ENTREGABLES/Sector_NewSector/.

Important note: if the new sector uses specific metrics not included in standard_mapping.json, the metric map must be extended. The parser automatically logs unmapped labels to facilitate this process.

9.4 Complete Directory Structure

```
BIG_ANALITICS_ANALISIS_SECTORIAL/
|
|-- _ARQUITECTURA/
|   |-- config/
|   |   |-- sector_config_fintech.json
|   |
|   |-- 01_Data_Ingestion/
|   |   |-- scripts/
|   |   |   |-- capiq_parser.py
|   |   |   |-- capiq_mapper.py
|   |   |   |-- capiq_normalizer.py
|   |   |   |-- capiq_assembler.py
|   |   |   |-- capiq_validator.py
|   |   |   |-- master_runner.py
|   |   |-- mappings/
|   |       |-- standard_mapping.json
|   |
```

```
|   |-- 03_Feature_Engineering/
|   |   |-- scripts/
|   |       |-- feature_engineer.py
|   |       |-- feature_engineer_advanced.py
|   |
|   |-- 04_Analysis_Models/
|   |   |-- scripts/
|   |       |-- 4A_sector_scorecard.py
|   |       |-- 4B_clustering.py
|   |       |-- 4C_ridge_drivers.py
|   |       |-- 4D_alpha_analysis.py
|   |       |-- 4E_risk_scoring.py
|   |       |-- 4F_nowcasting.py
|   |       |-- 4G_anomaly_detection.py
|   |       |-- 4H_margin_predictor.py
|   |       |-- 4I_factors_momentum.py
|   |       |-- 4J_macro_sensitivity.py
|   |       |-- 4K_regime_drivers.py
|   |       |-- 4L_macro_factor_model.py
|   |       |-- 4M_trend_decomposition.py
|   |       |-- 4N_nonlinear_drivers.py
|   |       |-- 4O_network_analysis.py
|   |       |-- 4Q_rolling_window.py
|   |       |-- 4R_quantile_regression.py
|   |       |-- 4S_survival_analysis.py
|   |       |-- 4V_hhi_concentration.py
|   |       |-- run_all_models.py
|   |
|   |-- 05_Visualization/
|   |   |-- scripts/
|   |       |-- generate_all_charts.py
|   |
|   |-- 06_Reporting/
|       |-- scripts/
|           |-- generate_report.py
|           |-- generate_technical_report.py
|           |-- generate_pptx.py
|           |-- generate_study_guide.py
|
|-- _ENTREGABLES/
|   |-- Sector_FinTech/
|       |-- 01_Datos_Procesados/
|       |   |-- master_long.parquet
|       |   |-- company_metadata.parquet
|       |   |-- quality_report.json
|       |
|       |-- 02_Feature_Store/
|       |   |-- feature_store.parquet
|       |   |-- feature_store_advanced.parquet
|       |   |-- feature_catalog.json
|       |   |-- sector_stats.json
|       |
|       |-- 03_Analysis_Models/
```

```
|       |   |-- scorecard_enhanced.parquet
|       |   |-- cluster_assignments.parquet
|       |   |-- cluster_summary.json
|       |   |-- driver_model.json
|       |   |-- risk_scores.parquet
|       |   |-- momentum_scores.parquet
|       |   |-- margin_predictor.json
|       |   |-- sector_cycle.parquet
|       |   |-- factor_scores.parquet
|       |   |-- factor_info.json
|       |   |-- anomaly_scores.parquet
|       |   |-- context_packs.json
|       |   |-- quantile_regression.json
|       |   |-- survival_analysis.json
|       |   |-- macro_sensitivity_summary.json
|       |   |-- regime_drivers.json
|       |   |-- macro_factor_model.json
|       |   |-- trend_summary.json
|       |
|       |-- 04_Visualizaciones/
|       |   |-- 01_sector_heatmap.png .. 13_macro_sensitivity.png
|       |
|       |-- 05_Reporting/
|           |-- BigAnalitics_FinTech_Informe_Tecnico.docx
|           |-- BigAnalitics_FinTech_Report.docx
|           |-- BigAnalitics_FinTech_Presentacion.pptx
```

Source: Own elaboration.

10. Future Improvements and Research Directions

The current framework covers a broad analytical spectrum, but there are multiple lines of improvement that would increase the predictive power, statistical robustness, and practical utility of the system.

10.1 Advanced Econometric Models

Panel GMM (Arellano-Bond)

The Generalized Method of Moments estimator for dynamic panels allows including the dependent variable lagged as a regressor (y_{t-1} as predictor of y_t), which is natural in financial data where margins exhibit persistence. The Arellano-Bond method (1991) uses lags as instruments to resolve the endogeneity introduced by the lagged regressor. This would allow modeling the temporal dynamics of margins more rigorously than the current static within-transformation regression.

Copulas for Joint Risk (Tail Risk)

Copulas allow modeling the dependence structure between variables beyond linear correlation. In particular, Clayton or Gumbel-type copulas capture tail dependence — the probability that multiple companies experience distress simultaneously. This would be valuable for the risk scoring module, adding a systemic risk dimension to the current individual assessments.

VECM (Cointegration)

The Vector Error Correction Model (VECM) is applied when stable long-run relationships (cointegration) exist between variables. For example, if operating_margin and sga_to_revenue are cointegrated, temporary deviations from the equilibrium relationship self-correct. This would allow distinguishing between temporary deviations (that self-correct) and structural changes (that require revaluation).

10.2 Margin Predictor Improvements

The current predictor (AUC~0.45) has significant room for improvement:

1. Target reformulation: instead of 'YoY margin improvement', define 'margin exceeds sector median in the next quarter' (a more stable and predictable target).

2. Additional features: incorporate momentum features (4Q trend), sector dispersion features, and macro variables from the extended model.

3. SMOTE for balancing: although the current dataset is reasonably balanced (53/47), sub-periods may be imbalanced. Synthetic Minority Oversampling could improve stability across folds.

4. Walk-forward validation: replace standard cross-validation with temporal validation (train on periods 1..t, test on t+1) that respects the temporal structure of the data.

5. Longer horizon target: predict improvement at 2-4 quarters instead of 1, which may be more predictable by smoothing quarterly noise.

10.3 Market Data Integration

Adding market data (stock prices, market cap, enterprise value, valuation multiples) would open new analytical dimensions:

- Multiples vs fundamentals: identify undervalued/overvalued companies.
- Investment returns: evaluate whether identified operational alphas translate into superior market returns.
- Market volatility vs fundamental volatility: compare market-perceived risk vs fundamental risk measured by our models.

10.4 Interactive Dashboard (Streamlit)

An interactive dashboard would allow analysts to explore results dynamically: filter by company/period, compare metrics, drill down into anomalies, and generate customized reports. Streamlit is the natural choice given the existing Python stack — the existing visualization functions can be reused as a foundation.

10.5 NLP on Earnings Calls

Incorporating natural language processing on earnings call transcripts would capture qualitative information that financial statements do not reflect: management sentiment, forward guidance, mentions of expansion/contraction. Pre-trained models like FinBERT are specifically designed for financial text and could extract sentiment and topic features.

10.6 Quarterly Pipeline Automation

The pipeline is currently executed manually. Automation would include: (1) automatic download of CAPIQ data via API when new quarters are published, (2) automatic execution of the complete pipeline, (3) alert generation when anomalies or regime changes are detected, (4) incremental update of the feature store without reprocessing the entire history.

These improvements would transform the system from a retrospective analysis tool into a real-time financial intelligence system, capable of generating actionable alerts automatically each quarter.

Source: Own elaboration.