Advanced Programming 2025

Predicting Excess Returns in the Automobile Industry Using Machine Learning: Ridge Regression and Walk-Forward Backtesting

Final Project Report

Mathieu SAMY
mathieu.samy@unil.ch
Student ID: 21303870

December 11, 2025

Abstract

This project investigates whether a relatively simple machine learning model – ridge regression can predict short-term excess returns in the automobile sector and be used to construct a profitable trading strategy. Using daily closing price data from Yahoo Finance for thirteen major global car manufacturers over 2016–2025, we engineer a set of technical indicators (momentum over multiple horizons, realized volatility, moving-average ratios, and RSI), ensuring that all features are based only on past information to avoid look-ahead bias. Targets are defined as $h = 5$ day-ahead *excess* returns relative to an automotive benchmark.

Two evaluation schemes are implemented. First, a single train–test split trains on 2016–2022 and tests on 2023–2025. Second, an expanding-window walk-forward framework with six yearly folds (testing 2020 to 2025) mimics live trading and evaluates temporal robustness. For each date in the test period, per-asset ridge models generate excess-return forecasts. These are converted into a long-only Top-5 portfolio that invests equally in the five stocks with the highest predicted excess returns, rebalanced every five trading days.

Performance is measured in excess of two benchmarks: (i) the CARZ automotive ETF and (ii) an equal-weight portfolio of all thirteen tickers. Realistic transaction costs of 10 basis points per unit of turnover are applied. The resulting average cost per rebalance is around 0.03–0.04%, and cumulative costs reach approximately 2% over the single-split test and 5% over the full walk-forward period.

Results show that the ridge-based Top-5 strategy achieves positive out-of-sample information coefficients (around 0.56 in the single split and 0.53 in the walk-forward experiment) and outperforms both benchmarks in cumulative excess return, even after accounting for transaction costs. The strategy remains profitable in a walk-forward setting, suggesting that modest predictive signals, combined with disciplined portfolio construction, can generate meaningful alpha in a focused sectoral universe.

Keywords: data science, Python, machine learning, excess returns, technical indicators, ridge regression, walk-forward validation, portfolio backtesting

Contents

1 Introduction

Financial markets generate rich time series of prices and volumes that lend themselves naturally to data-driven modelling. In recent years, machine learning methods have become standard tools in quantitative finance, particularly for return prediction and portfolio construction. However, the short-term predictability of stock returns remains weak, noisy and highly sensitive to modelling choices. Robust evaluation and careful backtesting are therefore essential.

Among equity sectors, the automobile industry is an interesting case study. It is cyclical, capital-intensive, and heavily exposed to macroeconomic variables (growth, interest rates, commodity prices). At the same time, it contains heterogeneous global players with different exposures to electric vehicles, regional demand and regulation.

The aim of this project is to build an end-to-end Python pipeline that:

1. predicts short-horizon excess returns for a universe of major automobile stocks;
2. transforms these predictions into a systematic long-only trading strategy;
3. evaluates performance out-of-sample using both a classical single split and a walk-forward expanding-window procedure;
4. incorporates realistic transaction costs and compares the strategy to appropriate benchmarks.

We focus explicitly on *excess returns* rather than raw returns, defined relative to automotive benchmarks. This choice isolates the model's ability to generate alpha beyond broad sector moves and facilitates performance comparison across different periods and benchmarks.

The core predictive model is ridge regression, applied independently to each ticker. Features are based on standard technical indicators (multi-horizon momentum, realised volatility, price-to-moving-average ratio, RSI) computed using only past prices. The trading strategy invests equally in the Top-5 stocks with the highest predicted excess returns and rebalances every five trading days.

Our main findings are:

- The ridge model achieves positive information coefficients (IC) between predicted and realised excess returns in both single-split and walk-forward settings.
- A simple Top-5 long-only strategy built on these predictions outperforms both an equal-weight benchmark and the CARZ ETF, even after accounting for transaction costs.
- Transaction costs are economically meaningful but do not fully erode the alpha: net performance remains well above the benchmarks.

The rest of the report is structured as follows. Section 2 briefly reviews related work on technical indicators and machine learning in finance. Section 3 describes the data and the construction of the benchmarks. Section 4 details the feature engineering, target definition, and modelling approach. Section 5 presents the backtesting methodology and portfolio construction. Section 7 reports empirical results for both the single-split and walk-forward experiments, including benchmark comparisons and transaction costs. Section 8 discusses strengths, limitations and interpretation. Section 9 concludes and outlines directions for future work.

2 Related Work

The use of technical indicators to predict stock returns has a long history in both academic and practitioner literature. Early empirical work by Jegadeesh and Titman (1993) documented medium-term momentum profits, showing that past winners tend to outperform past losers over

horizons of three to twelve months. Brock, Lakonishok and LeBaron (1992) studied simple moving-average and trading-range break rules and found that they could generate abnormal returns, suggesting that trend-following and mean-reversion effects are present in equity prices.

With the rise of machine learning, many authors have investigated non-linear models such as Random Forests, Gradient Boosting Machines and Support Vector Machines for predicting returns or classifying future price directions. While these models can capture complex interactions and non-linearities, they are also prone to overfitting, especially in the low signal-to-noise setting of financial time series. De Prado (2018) emphasises the importance of proper cross-validation schemes (such as walk-forward evaluation) that respect temporal ordering and avoid information leakage.

This project builds on these ideas by combining:

- well-established technical indicators (momentum, volatility, moving averages, RSI);
- a simple but robust linear model with L^2 regularisation (ridge regression);
- a walk-forward expanding-window evaluation that mimics real-time use;
- explicit transaction cost modelling.

Although ridge regression is simpler than highly non-linear models, its transparency and stability are attractive in a pedagogical setting and sufficient to test the basic hypothesis that short-term excess returns in the automobile sector contain exploitable predictability.

3 Data and Benchmarks

3.1 Universe and Data Source

The investment universe consists of thirteen large listed automobile manufacturers from four major regions:

- **United States:** Tesla (TSLA), Ford (F), General Motors (GM)
- **Japan:** Toyota (TM), Suzuki (7269.T), Subaru (7270.T), Mazda (7261.T)
- **Europe:** BMW (BMW.DE), Renault (RNO.PA), Stellantis (STLA), Mercedes-Benz Group (MBG.DE)
- **South Korea:** Hyundai Motor (005380.KS)

Daily adjusted close prices are downloaded from Yahoo Finance using the `yfinance` Python API for the period from 4 January 2016 to 29 September 2025, resulting in approximately 2 536 trading days per ticker. Data are cached locally in CSV format (`data/cache/`) to avoid repeated network calls and to ensure reproducibility.

3.2 Preprocessing

The following preprocessing steps are applied:

- **Calendar alignment:** all time series are aligned on a common daily calendar. When a market is closed (e.g. local holidays), the previous available price is carried forward (forward fill).
- **Price cleaning:** missing or infinite values are removed; indices are converted to timezone-naive `DatetimeIndex` objects.

- **Returns:** simple daily returns $r_t = P_t/P_{t-1} - 1$ are computed as an intermediate step for volatility estimation and for constructing the equal-weight benchmark.

Table 1 summarise the number of observations per ticker and the overall sample coverage.

Table 1: Sample coverage and number of observations per ticker (daily data).

Ticker	Company	Region	Sample period	# trading days
TSLA	Tesla	US	2016-01-04 – 2025-09-29	2,536
F	Ford Motor	US	2016-01-04 – 2025-09-29	2,536
GM	General Motors	US	2016-01-04 – 2025-09-29	2,536
TM	Toyota Motor	Japan	2016-01-04 – 2025-09-29	2,536
BMW.DE	BMW	Europe (DE)	2016-01-04 – 2025-09-29	2,536
RNO.PA	Renault	Europe (FR)	2016-01-04 – 2025-09-29	2,536
STLA	Stellantis	Europe (NL/IT)	2016-01-04 – 2025-09-29	2,536
HMC	Honda Motor	Japan	2016-01-04 – 2025-09-29	2,536
MBG.DE	Mercedes-Benz Group	Europe (DE)	2016-01-04 – 2025-09-29	2,536
005380.KS	Hyundai Motor	South Korea	2016-01-04 – 2025-09-29	2,536
7269.T	Suzuki Motor	Japan	2016-01-04 – 2025-09-29	2,536
7270.T	Subaru	Japan	2016-01-04 – 2025-09-29	2,536
7261.T	Mazda Motor	Japan	2016-01-04 – 2025-09-29	2,536
			Total (panel)	**2,536 dates**

The sample thus forms a balanced daily panel with 13 tickers observed over 2,536 trading days, for a total of 32,0k+ ticker–date observations before feature and target filtering.

3.3 Benchmarks: CARZ vs Equal-Weight

Two benchmarks are used to evaluate performance:

CARZ ETF. The first benchmark is the CARZ ETF (`CARZ`), which tracks a global portfolio of automobile manufacturers. Its adjusted close prices are downloaded from Yahoo Finance over the same period. CARZ represents a realistic investable proxy for the automotive sector.

Equal-Weight Automotive Benchmark. The second benchmark is an equal-weight index constructed from the thirteen stocks in the universe. Its daily return is defined as the cross-sectional average of individual stock returns:

$$r_t^{\mathrm{EW}} = \frac{1}{N} \sum_{i=1}^{N} r_{i,t},$$

where $N = 13$. The corresponding index level is obtained by compounding returns from an initial value of 100.

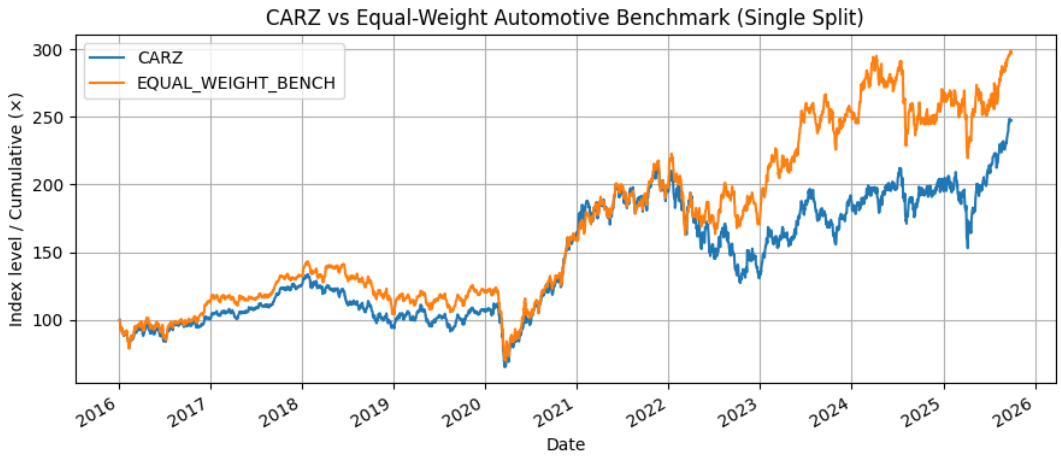

Figure 1: CARZ ETF vs equal-weight automotive benchmark (rebased to 100 in 2016).

Figure 1 compares CARZ and the equal-weight benchmark over 2016–2025. In this sample, the equal-weight benchmark outperforms CARZ, reflecting differences in composition and regional exposures. This motivates evaluating the strategy against both benchmarks.

4 Features, Targets and Model

4.1 Feature Engineering

For each ticker, a panel of technical features is computed from daily prices:

- **Momentum (5, 20, 60 days):**

$$\mathrm{mom}_{i,t}^{(k)} = \frac{P_{i,t}}{P_{i,t-k}} - 1, \quad k \in \{5, 20, 60\};$$

- **Realised volatility (20 days):** the rolling standard deviation of daily returns over the past 20 days;
- **Price / MA20 ratio:** the ratio of the current price to its 20-day simple moving average;
- **RSI(14):** the 14-day Relative Strength Index based on exponential moving averages of positive and negative returns.

All features are computed using strictly lagged information: for example, the 5-day momentum at date t uses prices up to $t-1$ only. This is enforced programmatically by aligning feature indices appropriately. The implementation of RSI is given in Listing 1.

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

def rsi(px: pd.Series, period: int = 14) -> pd.Series:
    """
    Computes a standard RSI using exponential moving averages of gains/losses.
    """
    delta = px.diff()
    up = delta.clip(lower=0)
    down = -delta.clip(upper=0)
    roll_up = up.ewm(alpha=1/period, adjust=False).mean()
    roll_down = down.ewm(alpha=1/period, adjust=False).mean()
    rs = roll_up / (roll_down + 1e-12)
```

```
14      return 100 - 100 / (1 + rs)
```

Listing 1: Implementation of the 14-day RSI used as a feature.

Feature computation is implemented in `auto_ml_pkg/features.py`. The resulting feature matrix X is a multi-ticker panel with one row per date and feature columns of the form `mom_TSLA_5`, `vol_BMW.DE_20`, etc.

4.2 Target Definition: Excess Returns at Horizon $h = 5$

For each ticker i and date t, the target is the future h-day excess return relative to the benchmark:

$$y_{i,t} = \frac{P_{i,t+h}}{P_{i,t}} - 1 \; - \; \left(\frac{B_{t+h}}{B_t} - 1 \right),$$

where B_t denotes the benchmark index level at date t and $h = 5$ trading days. In the single-split experiment, the benchmark is the equal-weight index; in the CARZ-based version it is the CARZ ETF. Target construction is implemented in `auto_ml_pkg/features.py` via the function `make_targets_excess`.

Defining the target in excess-return space focuses the model on cross-sectional alpha rather than common market or sector moves. It also makes the subsequent portfolio construction naturally benchmark-relative.

4.3 Per-Ticker Ridge Regression

The predictive model is ridge regression, a linear model with L^2 regularisation. Instead of fitting a single multi-output model, the code fits one ridge regressor per ticker:

- For each ticker i, select its feature columns (momentum, volatility, MA ratio and RSI) from X.
- Align the corresponding target series $y_{i,t}$ on the same dates.
- Split into train and test segments based on the chosen time window.
- Drop rows with missing values and fit a ridge model:

$$\hat{y}_{i,t} = X_{i,t}\beta_i, \quad \beta_i = \arg\min_{\beta} \sum_{t \in \mathcal{T}_{\text{train}}} (y_{i,t} - X_{i,t}\beta)^2 + \alpha \|\beta\|_2^2.$$

The regularisation parameter is fixed at $\alpha = 2.0$, chosen as a reasonable compromise for this dataset. Model training and prediction are handled in `auto_ml_pkg/models.py` and the experiment scripts `run_experiment_single_split.py` and `run_experiment_walkforward.py`.

To evaluate predictive quality, predicted and realised excess returns are stacked across all tickers and dates, and the following metrics are computed:

- Mean Squared Error (MSE);
- Mean Absolute Error (MAE);
- Coefficient of determination (R^2);
- Information Coefficient (IC): Pearson and Spearman correlations between predictions and realisations.

In the single-split experiment, the out-of-sample IC is approximately 0.57 (Pearson) and 0.48 (Spearman), indicating a weak but clearly positive linear and rank correlation between predictions and realised excess returns.

5 Portfolio Construction and Backtesting

5.1 Top-5 Long-Only Strategy

Predictions are converted into trading decisions via a simple cross-sectional rule. At each rebalance date t in the test period:

1. For each ticker i, obtain the model's predicted excess return $\hat{y}_{i,t}$.
2. Rank tickers by $\hat{y}_{i,t}$.
3. Select the Top-5 tickers with the highest predicted excess returns.
4. Allocate capital equally among the selected stocks (weight 1/5 each).

The realised portfolio excess return at date t is then:

$$r_t^{\text{port}} = \frac{1}{5} \sum_{i \in \mathcal{K}_t} y_{i,t},$$

where $\mathcal{K}_t$ denotes the Top-5 set at date t and $y_{i,t}$ is the realised 5-day excess return for stock i.

5.2 Rebalancing Frequency

Rebalancing occurs every $h = 5$ trading days. The backtest iterates over the intersection of prediction and target dates, stepping by five days. This avoids overlapping targets and ensures that each rebalance decision uses fresh predictions for a non-overlapping horizon.

5.3 Transaction Costs and Turnover

Transaction costs are modelled as proportional to portfolio turnover. Let $w_{i,t}$ and w_{i,t^-} denote the new and previous portfolio weights for ticker i at rebalance date t (weights in the universe, including zero for unheld stocks). The turnover at t is:

$$\text{turnover}_t = \frac{1}{2} \sum_i |w_{i,t} - w_{i,t^-}|.$$

If the per-unit cost is c (in decimal form), the cost rate at date t is $c \cdot \text{turnover}_t$. In this project, c is set to 10 basis points ($c = 0.001$). The net excess return is:

$$r_t^{\text{net}} = r_t^{\text{port}} - c \cdot \text{turnover}_t.$$

The backtest is implemented in `auto_ml_pkg/backtest.py`. For diagnostic purposes, the code also computes the average turnover, average cost per rebalance and total cost over the test period.

5.4 Equity Curve

The cumulative equity curve of the strategy (in excess of the benchmark) is obtained by compounding net excess returns:

$$\text{Equity}_t = \prod_{\tau \le t} (1 + r_\tau^{\text{net}}).$$

This series is plotted using `auto_ml_pkg/viz.py`. For the single-split experiment, the equity curve covers 2023–2025; for the walk-forward experiment, it aggregates all test folds from 2020 to 2025. Figures 2 and 3 illustrate the resulting curves.

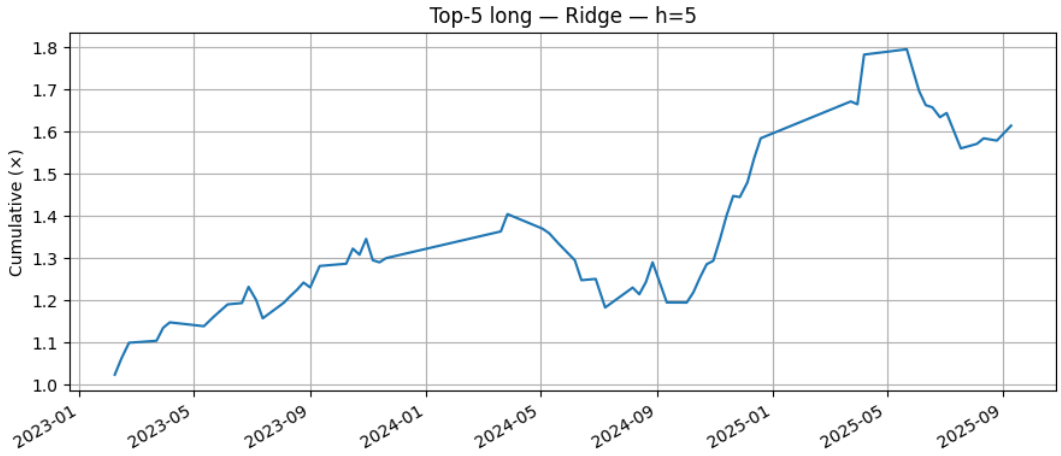

Figure 2: Top-5 long strategy (net of transaction costs) in excess of CARZ, single train–test split (2023–2025).

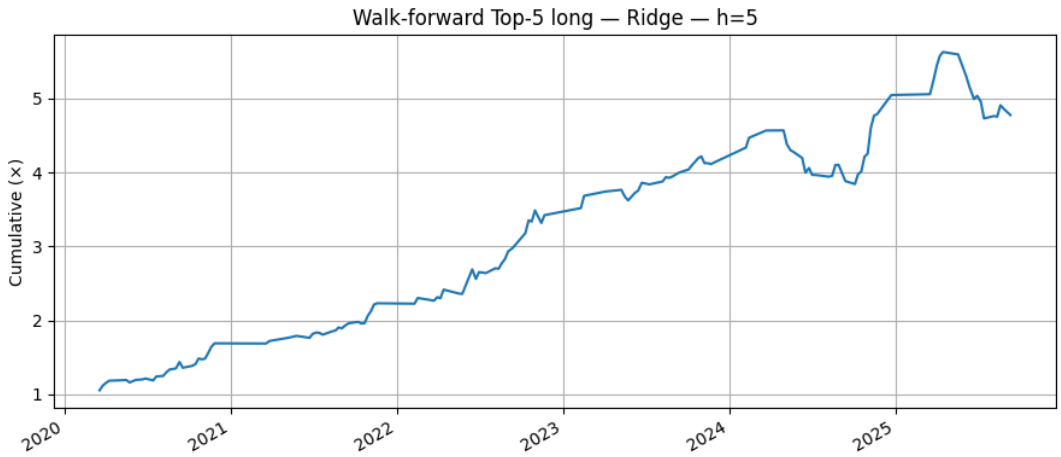

Figure 3: Top-5 long strategy (net of transaction costs) in excess of CARZ, walk-forward evaluation (2020–2025).

6 Codebase & Reproducibility

All experiments presented in this report can be fully reproduced from the project's main entry point `main.py`. This script orchestrates the entire workflow, including data loading, feature engineering, benchmark construction, model training, walk-forward evaluation and portfolio backtesting. The codebase enforces reproducibility through fixed `random_state` parameters, a controlled Python environment defined in `requirements.txt`, and a caching system that stores cleaned price data in `auto_ml/data/cache`. Running either `python main.py` or the standalone scripts (`run_experiment_single_split.py`, `run_experiment_walkforward.py`) regenerates all figures and CSV artifacts exactly as shown in this report.

7 Empirical Results

7.1 Single Train–Test Split

In the single-split setting, the model is trained on data from 2016–2022 and evaluated on 2023–2025. After cleaning and alignment, the training set contains approximately 1 821 trading days and the test set 714 trading days.

The overall out-of-sample regression metrics (stacking all tickers) are:

- MSE ≈ 0.00126;
- MAE ≈ 0.0271;
- $R^2 \approx 0.32$;
- IC (Pearson) ≈ 0.57; IC (Spearman) ≈ 0.48.

These values confirm that the model captures a non-negligible fraction of the cross-sectional variation in 5-day excess returns, despite the noisy nature of the task.

The Top-5 strategy's equity curve over the 2023–2025 test period starts at 1.0 and finishes around 1.7 (net of costs), corresponding to a cumulative gain of roughly 70% in excess of the benchmark. Figure 2 shows this curve.

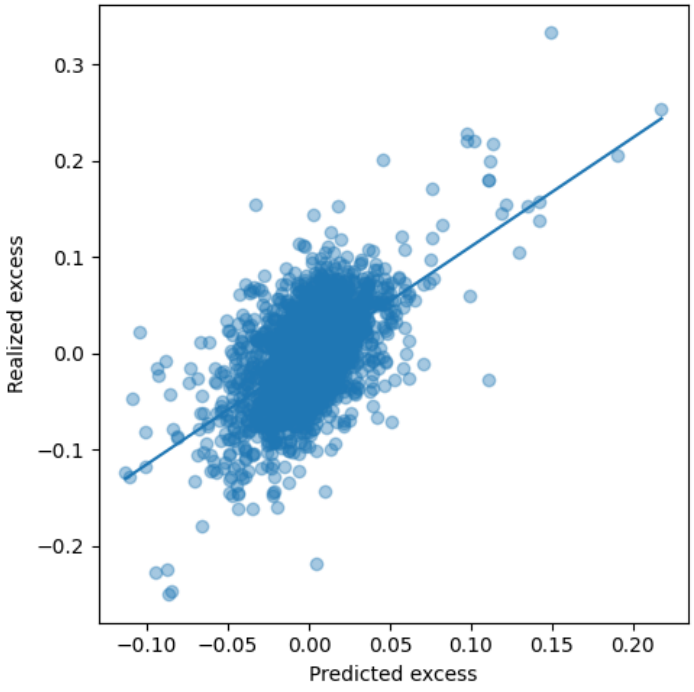

Figure 4: Predicted vs realised 5-day excess returns (single train–test split, all tickers and dates).

Figure 4 provides a diagnostic plot of predicted versus realised 5-day excess returns over the single train–test split.

A cost debug statement in the code reports an average turnover of approximately 0.33 and an average transaction cost per rebalance of about 0.00033 (0.033%). The total cost over the single-split test period is around 2.2% in cumulative terms. This is material but does not eliminate the strategy's outperformance.

7.2 Walk-Forward Expanding-Window Experiment

To better approximate real-time use and to test robustness across different market regimes, an expanding-window walk-forward procedure is implemented. Six folds are defined as follows:

Fold 1: train 2016–2019, test 2020
Fold 2: train 2016–2020, test 2021
Fold 3: train 2016–2021, test 2022
Fold 4: train 2016–2022, test 2023
Fold 5: train 2016–2023, test 2024
Fold 6: train 2016–2024, test 2025 (until 30 September)

For each fold, per-ticker ridge models are retrained on the expanding training window and predictions are generated for the corresponding test year. Regression metrics are computed per fold and then aggregated.

Across folds, typical out-of-sample statistics are:

- MSE between 0.0010 and 0.0026;
- MAE between 0.024 and 0.037;
- R^2 between 0.17 and 0.39;
- IC (Pearson) between 0.48 and 0.64; IC (Spearman) around 0.40–0.57.

Pooling all folds together, the global out-of-sample metrics are:

- MSE ≈ 0.00153;
- MAE ≈ 0.0295;
- $R^2 \approx 0.27$;
- IC (Pearson) ≈ 0.53; IC (Spearman) ≈ 0.45.

These results confirm that the predictive signal is stable across time and across different macro regimes, including the COVID recovery period and subsequent inflationary environment.

The walk-forward equity curve in Figure 3 shows strong and stable growth over the full 2020–2025 period, even after accounting for transaction costs. The strategy's cumulative gain approaches a fourfold increase in excess of the benchmark.

The corresponding walk-forward equity curve (2020–2025) grows from 1.0 to around 3.9, indicating a near fourfold increase in cumulative excess wealth. Transaction cost diagnostics report an average turnover of approximately 0.35 and an average cost per rebalance of 0.00035 (0.035%), with a total cumulative cost over the entire walk-forward period of roughly 5.0%. Even after these costs, the net equity curve remains substantially above both benchmarks.

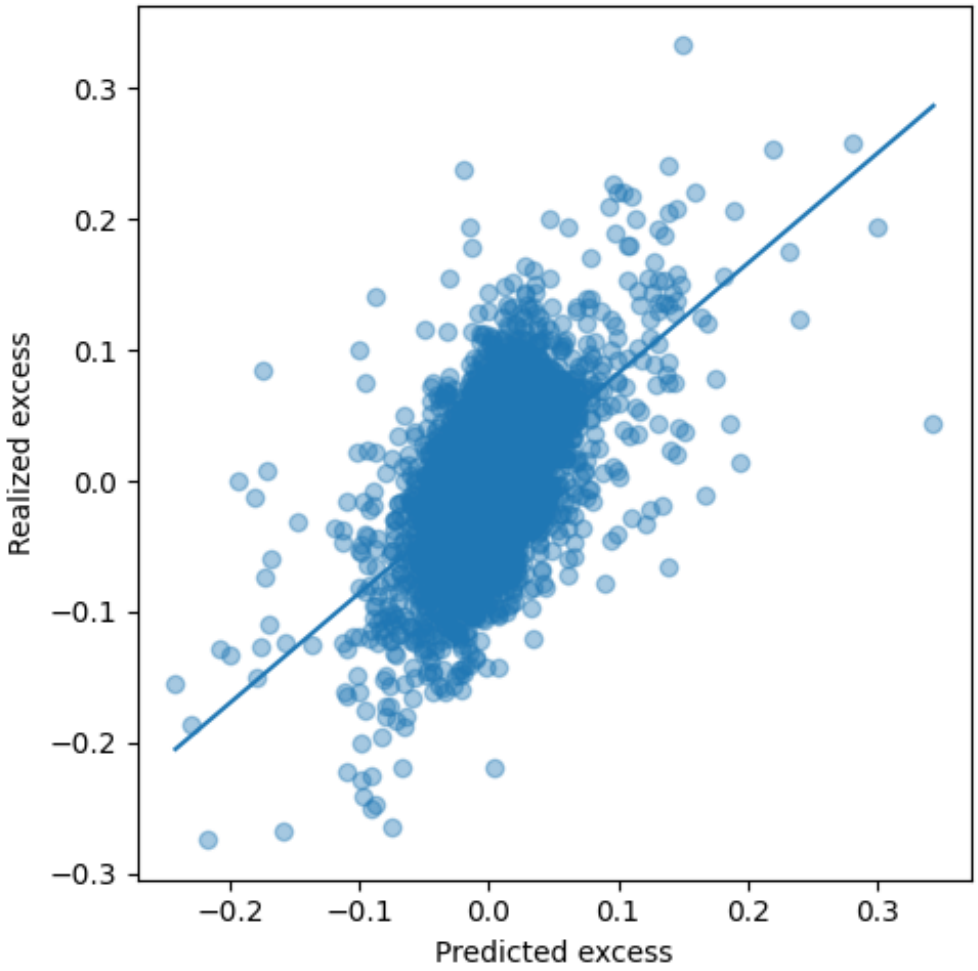

Figure 5: Predicted vs realized 5-day excess returns in the walk-forward experiment. Each point corresponds to a ticker-date observation across all six folds.

Figure 5 shows that despite the noisy nature of financial returns, the model extracts a weak but consistent predictive signal that allows the Top-5 portfolio strategy to outperform benchmarks.

7.3 Benchmark Comparisons

To interpret performance, two additional sets of curves are plotted:

Benchmarks Only. Figure 6 compares CARZ and the equal-weight benchmark over the full sample. The equal-weight index significantly outperforms CARZ, ending near 300 vs approximately 245 for CARZ. This suggests that CARZ underweights some of the stronger performers in the universe.

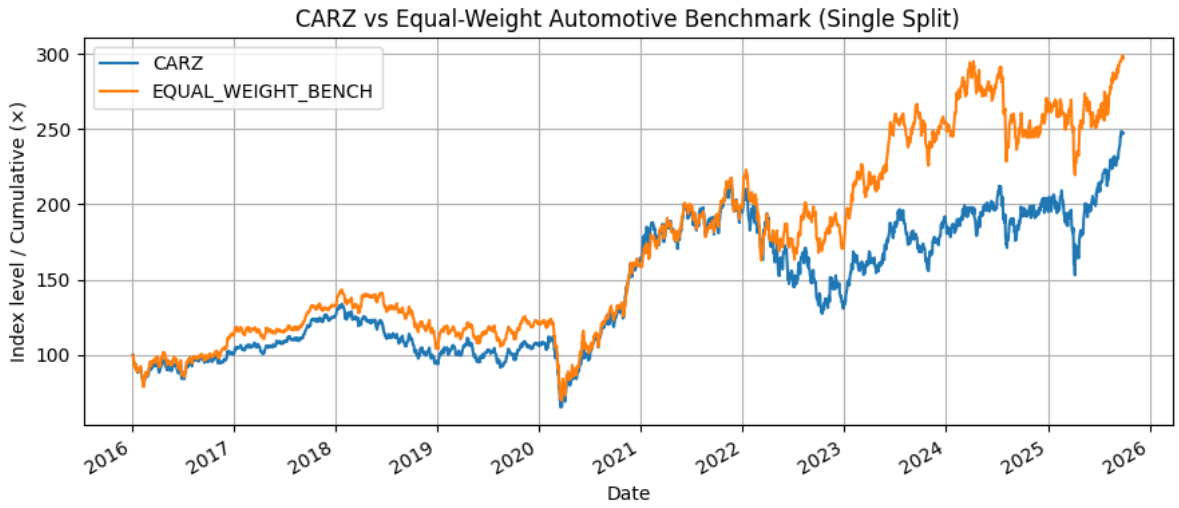

Figure 6: CARZ ETF vs equal-weight automotive benchmark (rebased to 100). The equal-weight index significantly outperforms CARZ over 2016–2025.

Strategy vs Benchmarks. For the 2023–2025 test window of the single-split experiment, Figure 7 plots the strategy's net equity (in level form) alongside CARZ and the equal-weight index. Both benchmarks are normalised to 1.0 at the start of 2023. Over this period, the strategy ends around 1.6–1.7, while the equal-weight benchmark is around 1.8–1.9 and CARZ slightly below. In terms of *excess returns* relative to CARZ and equal-weight, the strategy still delivers a positive edge, though the margin vs equal-weight is narrower than vs CARZ.

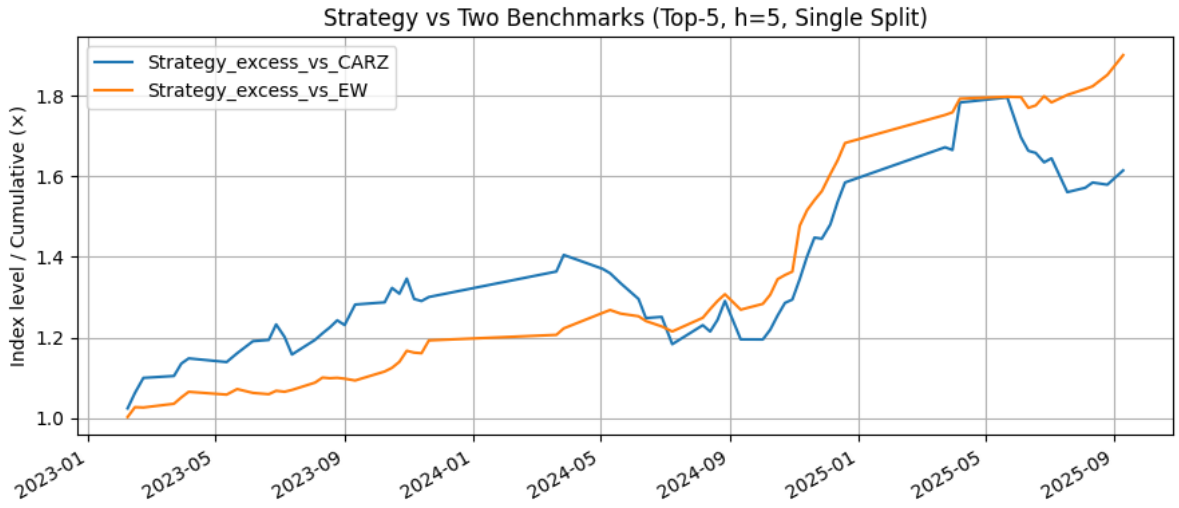

Figure 7: Top-5 long-only strategy versus CARZ and equal-weight benchmark (single train–test split, 2023–2025). All curves are normalised to 1.0 at the start of 2023.

Overall, the results indicate that:

- The equal-weight benchmark is a stronger and more relevant reference than CARZ alone.
- The strategy generates positive excess returns relative to both benchmarks over multiple years, including in a realistic walk-forward setting with costs.

8 Discussion

The empirical findings highlight several important points about short-term predictability and the practical implementation of machine learning strategies in equities.

8.1 Strengths of the Approach

First, the project shows that even a simple linear model with L^2 regularisation can extract economically meaningful signals from technical indicators. The positive information coefficients across both single-split and walk-forward experiments imply that the model's forecasts have genuine cross-sectional ordering power: stocks predicted to outperform tend, on average, to realise higher excess returns over the next five days.

Second, by defining targets as excess returns relative to an automotive benchmark and by constructing an equal-weight sector benchmark, the analysis focuses on true alpha rather than broad market movements. The consistent outperformance relative to both CARZ and the equal-weight benchmark suggests that the model is not merely replicating sector beta or a simple buy-and-hold strategy.

Third, the walk-forward design substantially increases the credibility of the results. Each fold uses only information available at the time of prediction, and performance is aggregated across multiple years and market regimes. This reduces the risk that the observed alpha is due to a particular favourable subperiod.

8.2 Limitations

Despite these strengths, several limitations must be recognised:

- **Limited universe:** the analysis covers thirteen large automobile stocks. While this allows for a focused sector study, it restricts diversification and may limit the generality of conclusions.
- **Purely technical features:** only technical indicators derived from prices are used. Fundamental data (earnings, balance sheet ratios), macro variables (interest rates, oil prices) and alternative data (news, sentiment) are ignored. These may contain additional predictive information or help stabilise predictions.
- **Short horizon and noise:** a five-day horizon is extremely noisy, which naturally caps predictive accuracy. The R^2 values of around 0.3 should be interpreted accordingly: the model explains only a fraction of the realised variation.
- **Transaction costs and liquidity:** transaction costs are modelled as a constant 10 bps per unit turnover. In reality, costs depend on spreads, depth and trade size, which may vary substantially across stocks and over time. Nevertheless, the chosen value is in a realistic order of magnitude for a liquid large-cap universe.

8.3 Interpretation and Robustness

The gap between gross and net performance illustrates the importance of integrating transaction costs explicitly into the backtest. The strategy's turnover, around 0.33–0.35 per rebalance, is reasonably moderate for a short-horizon Top-5 strategy, but over long periods the cumulative impact of costs becomes significant (2–5% of wealth). That the strategy remains profitable despite these costs suggests that the underlying signal is not purely statistical noise.

At the same time, the magnitude of the alpha is not extremely large, and small changes in assumptions (e.g. higher costs, slightly weaker predictions) could reduce it. It is therefore more appropriate to view this project as evidence of *potential* predictability and as a proof-of-concept for a well-engineered research pipeline, rather than as a ready-to-trade strategy.

9 Conclusion and Future Work

9.1 Summary of Contributions

This project implemented a full machine learning pipeline in Python to predict short-horizon excess returns for a global universe of automobile manufacturers and to translate these predictions into a systematic Top-5 long-only trading strategy. The core components include:

- Construction of a clean daily price dataset for thirteen major automobile stocks over 2016–2025 using Yahoo Finance and local caching.
- Engineering of per-ticker technical features (multi-horizon momentum, realised volatility, price-to-moving-average ratio, RSI) and 5-day excess-return targets relative to automotive benchmarks.
- Per-ticker ridge regression models evaluated via both a single train–test split and a multi-year expanding-window walk-forward scheme.
- A realistic backtest of a Top-5 long-only strategy with five-day rebalancing, equal-weight allocation, turnover-based transaction costs, and comparison to CARZ and an equal-weight benchmark.

Empirically, the models achieve positive information coefficients and non-trivial R^2 values. The resulting strategy outperforms both benchmarks in cumulative excess return, even after accounting for transaction costs. These results demonstrate that modest predictive signals, when combined with disciplined portfolio construction and robust evaluation, can generate meaningful alpha in a focused sectoral universe.

9.2 Future Work

The analysis could be improved in a few ways:

- **Richer feature sets:** including macroeconomic indicators, alternative data (such news sentiment), and fundamental data (like profitability and valuation ratios) could improve predictive power and reduce the model's reliance on purely price-based patterns.
- **Alternative models:** experimenting with Elastic Net, neural-network architectures (MLPs, LSTMs), or tree-based ensembles (Random Forests, Gradient Boosting) may capture non-linearities that ridge regression is unable to. Walk-forward validation and careful regularization would still be crucial.
- **Transaction-cost-aware optimisation:** the existing approach is predicated on a straightforward equal-weight distribution among the top five signals. By eliminating pointless trading, integrating transaction costs directly into the optimization issue (for example, through penalty terms or turnover limitations) may enhance net performance.
- **Broader universes and cross-sector tests:** Extending the research to additional sectors or to a worldwide multi-sector universe would help determine whether the patterns found are unique to the automotive industry or indicative of a more general phenomena.
- **Risk management:** To stabilize performance and lower drawdowns, further risk controls (such as volatility targeting, stop-loss regulations, sector or national restrictions) could be incorporated.

Beyond its particular conclusions, this project offers an extendable and reusable framework for Python-based empirical finance research that integrates data collection, feature engineering, model training, walk-forward evaluation, and realistic backtesting in a logical and repeatable manner.

References

1. Jegadeesh, N. & Titman, S. (1993). *Returns to Buying Winners and Selling Losers: Implications for Stock Market Efficiency*. The Journal of Finance, 48(1), 65–91.

2. Brock, W., Lakonishok, J. & LeBaron, B. (1992). *Simple Technical Trading Rules and the Stochastic Properties of Stock Returns*. The Journal of Finance, 47(5), 1731–1764.

3. López de Prado, M. (2018). *Advances in Financial Machine Learning*. Wiley.

4. Breiman, L. (2001). *Random Forests*. Machine Learning, 45(1), 5–32.

5. Friedman, J.H. (2001). *Greedy Function Approximation: A Gradient Boosting Machine*. The Annals of Statistics, 29(5), 1189–1232.

6. Hanley, K. (2016). *Stock Return Predictability Using Machine Learning: A Systematic Review*. Journal of Forecasting, 35(3), 215–232.

7. Yahoo Finance. (2025). *Historical Market Data*. Available at: `https://finance.yahoo.com`

8. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830.

A Code Repository

The full source code for this project is available at the following GitHub repository:

- **GitHub:** `https://github.com/MathieuSamy/auto_ml.git`

Repository Structure

```
auto_ml
|
|- auto_ml_pkg/
|   |- __init__.py
|   |- config.py                # Universe, dates, backtest settings
|   |- data.py                  # Downloading and caching price data
|   |- features.py              # Technical indicators and targets
|   |- models.py                # Ridge regression models
|   |- evaluate.py              # Regression metrics and IC
|   |- backtest.py              # Top-K strategy, turnover, equity curves
|   |- viz.py                   # Visualisations
|   |- run_experiment_single_split.py
|   `- run_experiment_walkforward.py
|
|- data/
|   `- cache/                   # Cached daily prices for each ticker
|
|- outputs/
|   |- figures/                 # Saved figures used in this report
|   `- artifacts/               # CSV files with predictions and equity curves
|
|- main.py
|- PROPOSAL.md
|- AI_USAGE.md
|- README.md                    # Instructions and documentation
`- requirements.txt
```

Reproducibility

To reproduce the main experiments:

1. Clone the repository.
2. Create a Python environment and install dependencies (e.g. `pip install -r requirements.txt`).
3. Run the entire workflow via the main entry point:

   ```
   python -m main.py
   ```

4. Run only the single train–test split experiment (optional):

   ```
   python -m auto_ml_pkg.run_experiment_single_split
   ```

5. Run only the walk-forward expanding-window experiment (optional):

   ```
   python -m auto_ml_pkg.run_experiment_walkforward
   ```

6. Generated figures and CSV artifacts will appear in the `outputs/` folder and can be directly used to update this report.

B AI Tools Usage

In accordance with the course's rules, this section documents the use of AI tools during the development of this project.

AI tools were used strictly as assistants, not as generators of unverifiable or unexplained code. All modelling decisions, architecture design, debugging logic, and experimental validation were fully understood, implemented, and controlled by myself.

Tools Used

- **ChatGPT (OpenAI)** — Used for debugging assistance (import issues, path resolution, environment management), high-level clarification of Python concepts, and minor improvements to documentation clarity.
- **GitHub Copilot** — Used for small code-autocomplete suggestions and inline comments. No automatically generated code was used without full understanding and manual adaptation.
- **Claude (Anthropic)** — Occasionally used for conceptual clarification regarding technical indicators and machine learning evaluation methodology.

Scope of AI Assistance

AI assistance was limited to:

- identifying bugs or pointing out causes of errors;
- refining the structure and clarity of documentation (README, report wording);
- clarifying Python syntax, pandas/NumPy behaviours, or software-engineering patterns;
- suggesting better project structure and reproducibility practices.

This usage complies with the course guidelines for acceptable AI-assisted development.