Advanced Techniques in Data Analysis

Machine Learning in Stock Price Prediction

João Simões – 2021219025

Guilherme Alves - 2022171246

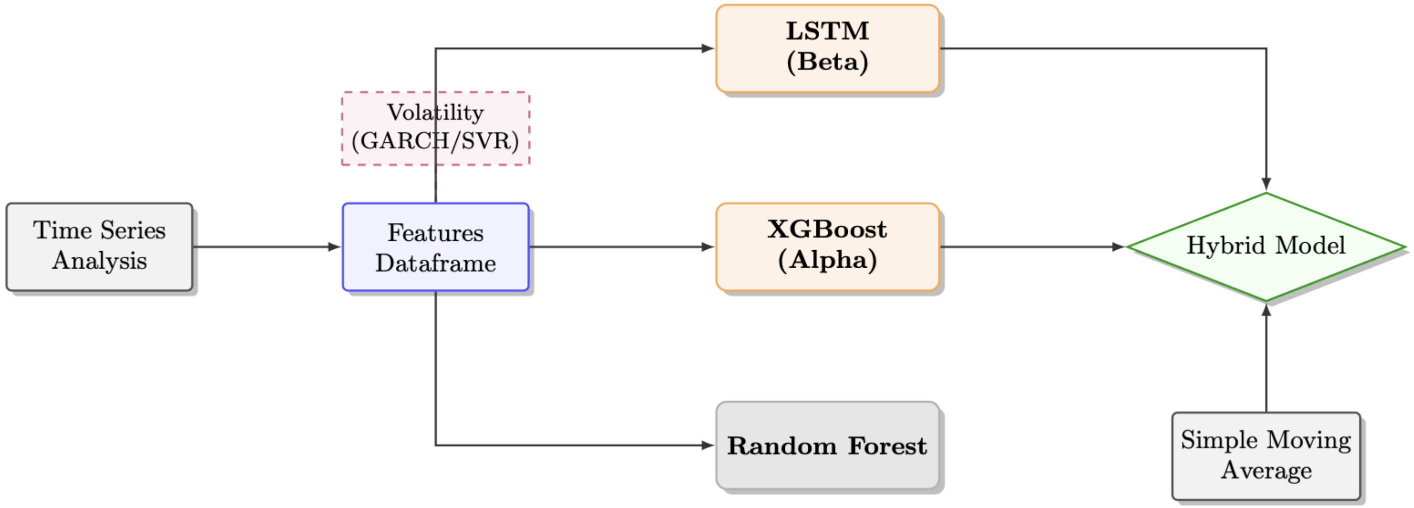
Bárbara Baptista – 2021218759

Tomás Gonçalves – 2021220159

Contents

Introduction

- **Context:** highly volatile financial market
- **Problem:** predicting PayPal price direction
- **Objective:** build a reliable direction-prediction model
- **Data:** daily historical data of the S&P 500 updated dataset



Time Series Analysis - Stationarity

Stationarity Principles

- Average
- Standard deviation
- Autocovariance

Decision Rules

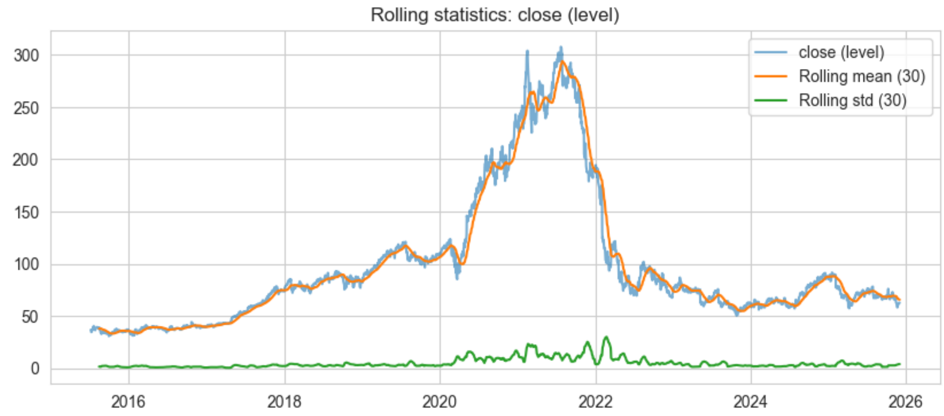
- **ADF:** small p (<0.05) → stationary | large p → non-stationary
- **KPSS:** small p (<0.05) → non-stationary | large p → stationary

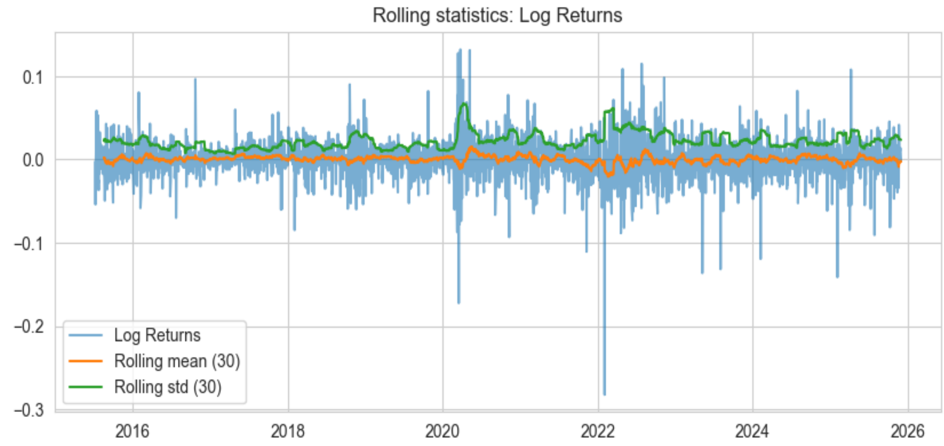
Close Prices

- **ADF:** statistic –1.4414 | p = 0.5623 → **non-stationary**
- **KPSS:** statistic 1.3707 | p = 0.0100 → **non-stationary**

Log Returns

- **ADF:** statistic –11.1814 | p = 0.0000 → **stationary**
- **KPSS:** statistic 0.4312 | p = 0.0637 → **stationary**





Time Series Analysis – Returns Analysis

Shapiro–Wilk Test

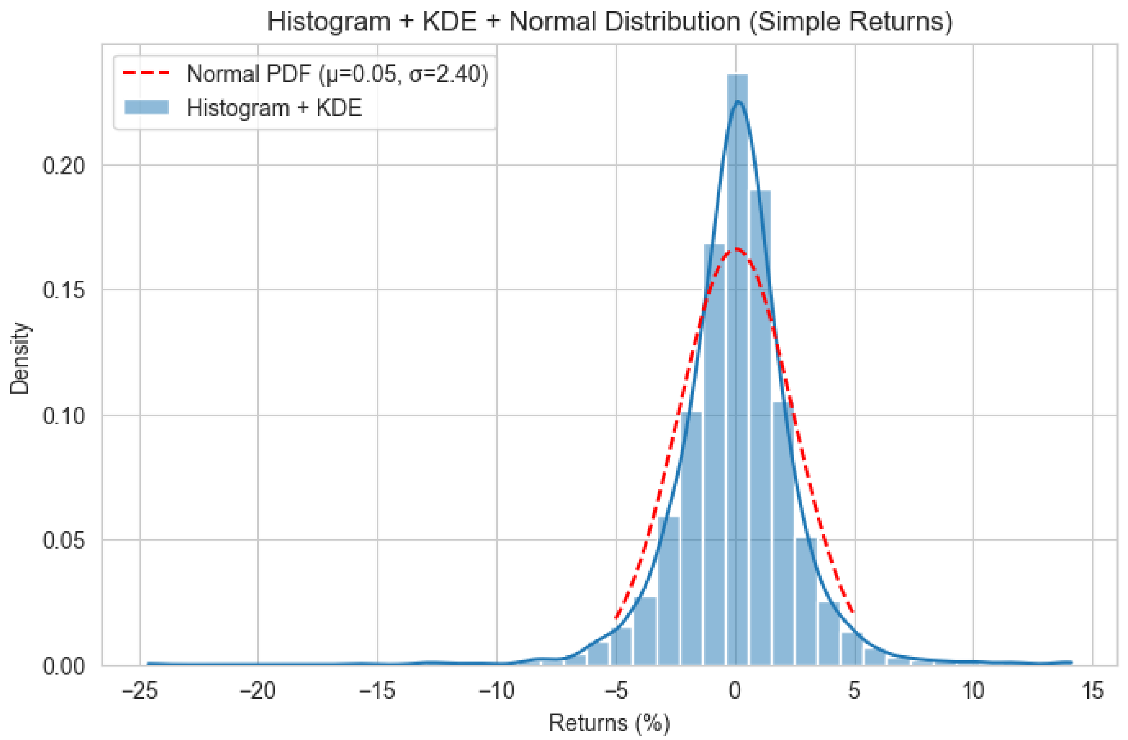
- Tests whether the sample follows a normal distribution.
- Statistics value: 0.9205
- p-value: ≈ 0.0000
- Conclusion: The null hypothesis of normality is rejected. The sample is not normally distributed.

Skewness

- –0.8027 (left-skewed distribution)

Kurtosis

- 14.3885



Feature Engineering

Features used:

```
log_ret
RSI_Norm
MACD_Norm
Dist_SMA50
BB_Width
Log_Vol_Change
RVOL
Vol_Lag_5
Ret_Sq_Mean
SVR_Vol_Pred
```

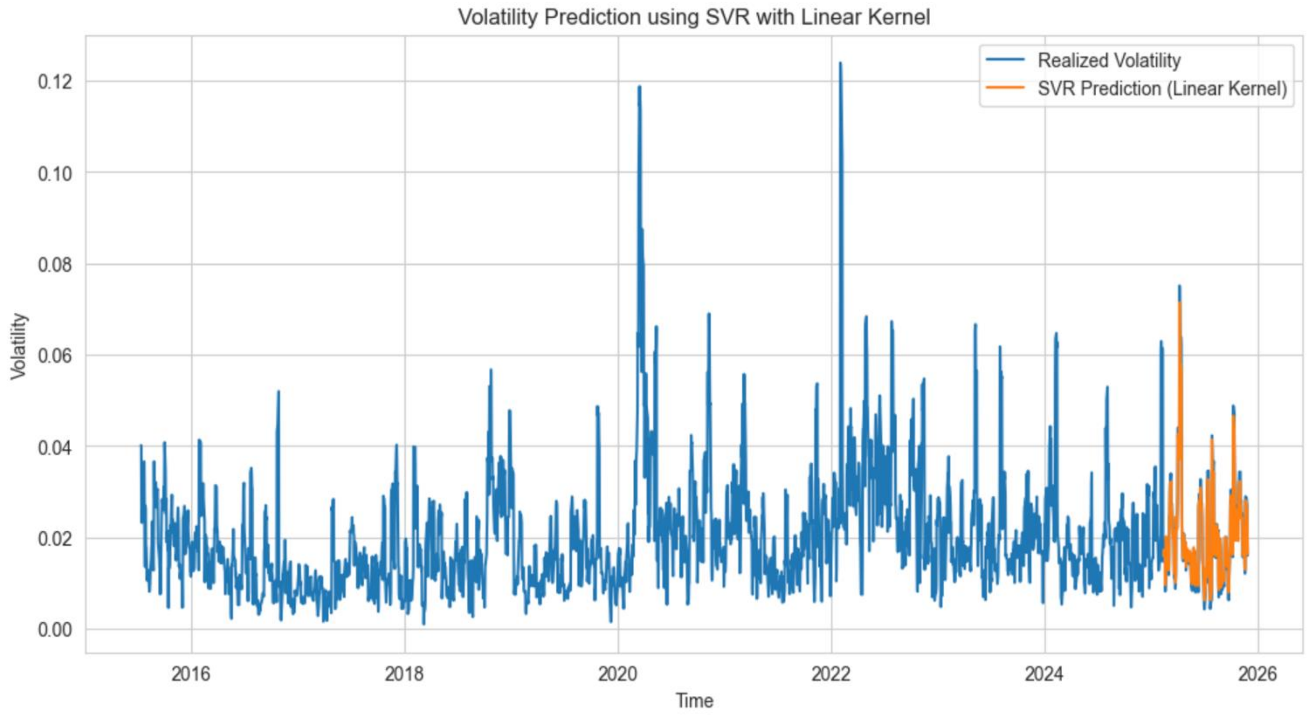
There are 3 types of technical features in the context of financial markets:

- Momentum & Trend
- Volatility
- Volume & Liquidity

Risk Management

We did a throughout exploration on this broad topic and we decided to test different models to predict volatility.

Model	RMSE
ARCH	0.2413
GARCH	0.2435
GARCH SVR (Linear Kernel)	**0.0027**
GARCH SVR (RBF Kernel)	0.0096
GARCH Neural Network	0.0096
GARCH Deep Learning	0.0038



Predicting Volatility using the GARCH model with a Linear SVR gave us the best RMSE!

Implemented Models – Random Forest

Why this model is appropriate in the financial context:

- Handles noisy data (averaging across many trees)
- Avoids overfitting (bagging + feature randomness)
- Robust to outliers and extreme events (trees isolate extreme events)

Results:

	precision	recall	f1-score	support
0	0.49	0.23	0.31	234
1	0.53	0.79	0.63	261
accuracy			0.52	495
macro avg	0.51	0.51	0.47	495
weighted avg	0.51	0.52	0.48	495

Implemented Models – LSTM

Long Short-Term Memory is a recurrent neural network. It learns time-dependent trends using memory cells.

To obtain the optimized LSTM, we:
- Determined the best Threshold value (0.57);
- Identified the best combination of Dimensions and Window size;
- Penalized choosing UP using the *BCEWithLogitsLoss()* function.

=== Table of Different Time Windows and Precision ===

	Window Size	Hidden Dim	Best Accuracy	Input Features
8	20	64	0.555789	10
4	10	32	0.552577	10
0	5	16	0.551020	10
1	5	32	0.551020	10
3	10	16	0.548454	10
6	20	16	0.547368	10
2	5	64	0.546939	10
9	60	16	0.544828	10
11	60	64	0.544828	10
5	10	64	0.544330	10
7	20	32	0.543158	10
10	60	32	0.537931	10

Results:

--- Final Report (Optimized LSTM) ---

	precision	recall	f1-score	support
Down (0)	0.48	0.85	0.61	208
Up (1)	0.52	0.15	0.23	227
accuracy			0.48	435
macro avg	0.50	0.50	0.42	435
weighted avg	0.50	0.48	0.41	435

We achieved a recall of 85% on the Down class.
A solid model for finding Beta!

Implemented Models – XGBoost

First, we did a hyperparameter optimization to get the best values with this method.

Extreme Gradient Booster is the industry golden standard to finding Alpha.

Results:

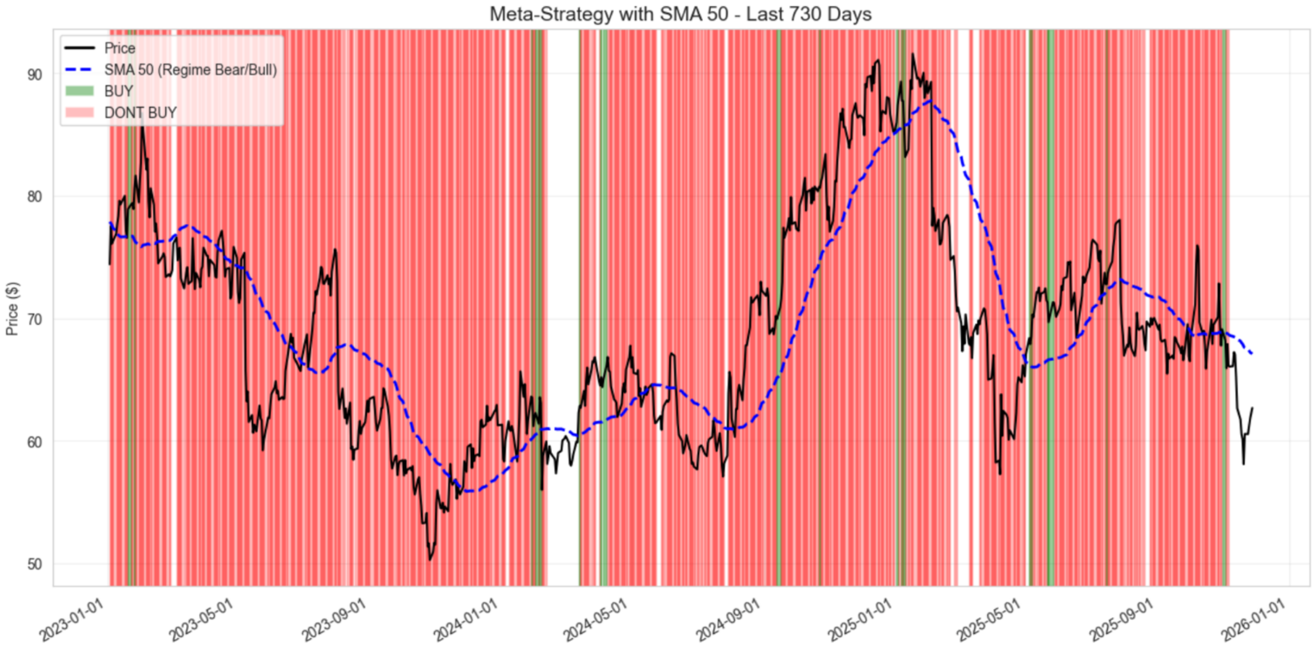
--- Final Report (XGBoost) ---

	precision	recall	f1-score	support
Down (0)	0.50	0.44	0.47	234
UP (1)	0.54	0.60	0.57	261
accuracy			0.52	495
macro avg	0.52	0.52	0.52	495
weighted avg	0.52	0.52	0.52	495

Limitations

- High volatility: stock price dominated by noise
- Direction prediction is inherently limited
- Models plateau without heavy fine-tuning
- No realistic back-testing (model tied to one asset)
- Only technical indicators used (no sentiment/news data)

Algorithmic Trading using Machine Learning



Metrics:

Metric	Buy & Hold	Our Model
Return	2.24%	0.10%
Sharpe	0.03	0.01
Max DD	-37.47%	-7.53%
Volatility	36.91%	5.66%

Future Work

- Extend model to multiple stocks
- Use broader market indices
- Integrate sentiment/news features

Conclusions

- Hybrid approach: LSTM + XGBoost (risk/beta model + alpha model)
- LSTM strong at predicting downtrends
- Instability in the last month
- Added moving average filter
- More reliable trading decisions
- Even with imperfect models, we built an effective trading function

References

[1] Abdullah Karasan, *Machine learning for financial risk management with Python : algorithms for modeling risk*. Cambridge: O'reilly, 2021.

[2] J. D. Cryer and K. Chan, *Time series analysis with applications in R*. New York: Springer, 2008.