Cryptocurrencies Regime Prediction

Mehdi Hachimi
MSc Quantitative Finance
ETH Zürich/UZH
mhachimi@student.ethz.ch
mehdi.hachimi@uzh.ch

Weisheng Wang
MSc Quantitative Finance
ETH Zürich/UZH
weiswang@student.ethz.ch
weisheng.wang@uzh.ch

Abstract

In this paper, we do regime prediction for the five cryptocurrencies with the highest market cap using machine learning algorithms and reservoir computing. We run Logistic Regression, Artificial Neural Networks, and Random Forests with past price information and technical indicators as model features. We also use randomized signatures to build the reservoir. The average classification accuracy is above the 50% threshold for all cryptocurrencies, showing that there exists predictability of trends in prices to a certain degree in the cryptocurrency markets. The reservoir method with randomized signatures demonstrates the best and most consistent results in terms of predictive accuracy compared to the machine learning algorithms.

1 Introduction

Since Bitcoin was invented in 2008, a lot of other cryptocurrencies were born, and knew different levels of success, with most of them disappearing shortly after they were released. Some cryptocurrencies, however, had a huge success like Ethereum and BNB, and their market capitalizations currently exceed those of some of the most famous companies, for example Ethereum has a higher market cap than Boeing, Disney, Starbucks, and Nike. This, and the incredibly high returns that can be achieved - if invested correctly -, attracted many investors (both retail and institutional) to cryptocurrencies, and made it one of the most liquid markets.

The recent advances in machine learning and stochastic analysis made it more accessible to understand the price evolution and dynamics of different assets. The goal of this paper is to leverage some of these recent methods to predict the signs of the next returns of some cryptocurrencies. The choice of the cryptocurrencies is based on their market cap, volume and liquidity, as well as data availability.

Besides Bitcoin and Ethereum that are omnipresent in the litterature about crypto market prediction, the other cryptocurrencies have not been studied in depth, one of the reasons being their short life (or existence) time compared to BTC or ETH. In this paper, besides BTC and ETH, we decided to also study some of the other "trendy" and most traded cryptocurrencies of the moment like BNB and Solana.

Our work is based on what was done by Erdinc Akyildirim, Ahmet Goncu and Ahmet Sensoy in [1], we build up on the key ideas that they develop in their paper by adding some features and using models based on signatures and randomized signatures that are recent advances in stochastic analysis. We also do it on a more recent dataset to test it on a setting compatible with the current market conditions.

Since predicting the exact time series of cryptocurrencies is a very challenging task due to the high volatility of their prices, we focus here on regime prediction, in other terms whether the price will go up or down in the next period(s). This is still a very useful information for a trader since they can decide based on that to take a long position in the cryptocurrency beforehand, in the case of an

expected price increase period, while in the case of an expected price decrease period, they can take a short position in the cryptocurrency through futures, ETFs, or margin trading.

All the code used for the work and results in this paper can be found on our respective Github accounts https://github.com/MehdiHachimi and https://github.com/Upower4S.

We start by presenting our dataset and how we got it in section 2. In section 3 we describe and explain the features that we feed our models, then in section 4 we describe the models used and present the results obtained by each one of them. In section 5, we introduce the randomized signature and use it for regime prediction. We discuss our results in section 6, and talk about randomized signature for portfolio optimization in section 7. Finally, we conclude in section 8.

2 Data

We get our data from Binance, you can find the script used to fetch the data on our Github repository. The dataset consists of 1-minute frequency data for 5 cryptocurrencies: BTC (Bitcoin), ETH (Ethereum), BNB, SOL (Solana), and XRP, from 01-01-2024 at 00:00 to 30-09-2025 at 23:59. The data we got from Binance has the following features for each minute: [open, close] time, [open, close, high, low] price, volume, quote asset volume, number of trades, taker buy base volume, taker buy quote volume. For BNB, we decided to drop the first 3 months of data because traded volume was almost 0 and hence price and other features were almost constant, we observed that this choice improved our models' performances.

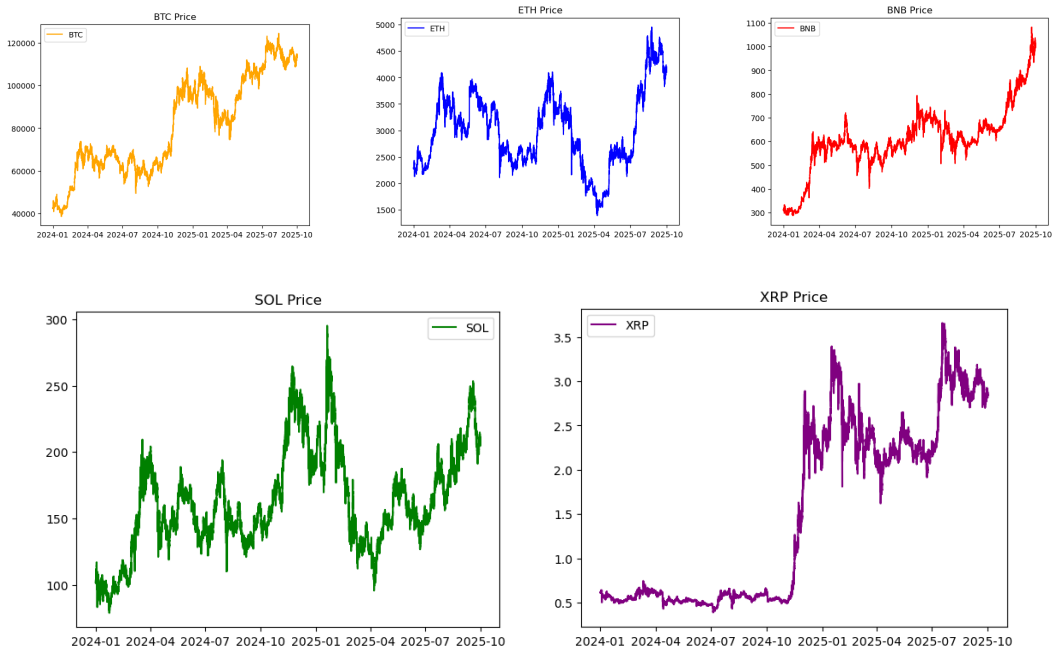


Figure 1: Minute-frequency close prices of Bitcoin (BTC), Ethereum (ETH), BNB, SOL (Solana), and XRP.

The choice of these 5 cryptocurrencies is due to data availability, since most of the other cryptocurrencies have a shorter lifetime. The other reason why we chose these 5 is that they are - excluding USDT (USD Token) which is a token - the top 5 in terms of market capitalization in the crypto market. As observed in table 1 below, these 5 cryptocurrencies represent 80.99% of the crypto market cap.

We transform our 1-minute frequency data into 15-minutes frequency data, and we split our dataset into 80% train, 10% validation and 10% test, except for logistic regression when we have a 90% train and 10% test split. The choice of the 15-minutes frequency is due to several reasons: First, it allows us to compare our results with our benchmark [1], but the most important reason is that the proportions of +1 and -1 (up and down) are balanced (approx. 50% each), while for the other frequencies, it is not. This balance is very important to build robust models, especially when working

Table 1: Market Cap of the crypto market, Bitcoin (BTC), Ethereum (ETH), BNB, SOL (Solana), and XRP. Souce: Coinbase.

Cryptocurrency	Market Cap (in $)	Percentage
BTC	1.82T	58.52%
ETH	368.07B	11.83%
BNB	122.36B	3.93%
SOL	77B	2.47%
XRP	132.07B	4.24%
Crypto Market	3.11T	100%

with financial datasets, that have a very low signal-to-noise ratio. For example, if the proportion of +1 is higher than -1, the model will tend to learn to output +1 more often in case the decision between the two is very tight, instead of exploring more deeply other caracteristics of our dataset.

3 Feature Engineering

In this section, we present the features we feed our machine learning models. We used the python package 'TA-Lib' to perform calculations.

To predict cryptocurrency returns, we construct a comprehensive set of technical features based on historical Open, High, Low, Close, and Volume (OHLCV) data. To ensure causal validity and prevent look-ahead bias, all input features are derived from data available at time $t-1$ or earlier, whereas the target variable is derived from time t.

Let C_t, H_t, L_t, and V_t denote the Close, High, Low, and Volume prices at time t, respectively.

3.1 Basic Price and Range Features

Before constructing complex indicators, we include the raw one-day lag of all input variables (Open, High, Low, Close, Volume, and Quote Volume). We also compute the daily price range:

$$\text{Range}_t = H_t - L_t \tag{1}$$

3.2 Return and Momentum Features

The primary basis for momentum analysis is the logarithmic return. We define the log-return r_t as:

$$r_t = \ln(C_t) - \ln(C_{t-1}) \tag{2}$$

We construct lagged return features for the window $i \in [1, 5]$. Additionally, we aggregate momentum over different time horizons:

- **Cumulative Returns:** Sum of returns over 3-day (S_3) and 5-day (S_5) windows, along with their divergence $\Delta_S = S_3 - S_5$.
- **Simple Momentum (MOM):** The raw price change over 5 periods: $MOM_5 = C_{t-1} - C_{t-6}$.

3.3 Trend Indicators

To capture the underlying market direction, we employ several moving average derivatives:

- **Simple Moving Average (SMA):** A 5-period rolling average of the closing price.
- **Double Exponential Moving Average (DEMA):** A 10-period DEMA of closing price.
- **Correlation of SMA with closing price:** the correlation of previously calculated SMA with closing price with rolling window of size 30. This is to assess trend consistency.
- **Exponential Weighted Average (EWA):** An exponentially weighted mean of the closing price with $\alpha = 0.9$, giving high weight to recent observations.

- **MACD:** The Moving Average Convergence Divergence indicator (Fast=5, Slow=10, Signal=5).
- **Rate of Change (ROC):** The percentage change in price over 9 and 14 periods:

$$ROC_n = \frac{C_{t-1} - C_{t-n}}{C_{t-n}} \tag{3}$$

3.4 Oscillators and Volatility

3.4.1 Relative Strength Index (RSI)

We implement the Relative Strength Index over 9 and 14 periods. The RSI is calculated using Exponential Moving Averages (EMA) of upward (U) and downward (D) price movements:

$$RSI = 100 - \frac{100}{1 + RS}, \quad \text{where } RS = \frac{\text{EMA}(U, n)}{\text{EMA}(D, n)} \tag{4}$$

and U and D are calculated as $U = \max(\text{close}_{now} - \text{close}_{previous}, 0)$ and $D = \max(\text{close}_{previous} - \text{close}_{now}, 0)$. Binary indicators are also generated to flag overbought ($RSI > 80$) and oversold ($RSI < 20$) conditions.

3.4.2 Average True Range (ATR)

Volatility is measured using the Average True Range. First, the True Range (TR) is defined as:

$$TR_t = \max(H_{t-1} - L_{t-1}, |H_{t-1} - C_{t-2}|, |L_{t-1} - C_{t-2}|) \tag{5}$$

The ATR is then calculated as the rolling mean of TR over 5 and 10 periods.

3.4.3 Williams %R Variant

We construct a normalized range feature inspired by the Williams %R indicator. Using a lookback window of 14 periods, let HH_{14} be the highest high and LL_{14} be the lowest low. The feature is defined as:

$$W_R = \frac{HH_{14} - C_{t-1}}{HH_{14} - LL_{14}} \tag{6}$$

3.4.4 Aroon and Stochastic Transformation

We also include the **Stochastic Oscillator** and **Aroon Indicator** to further capture market cycle positions and trend strength. We utilize the Aroon indicator to measure the time relative to the most recent highs and lows. Using a lookback window of $N = 14$ periods, let τ_{high} and τ_{low} be the number of periods since the rolling window's highest high and lowest low, respectively (where 0 indicates the current period). The Aroon Up and Down components are calculated as:

$$\text{Aroon}_{Up} = 100 \cdot \frac{(N-1) - \tau_{high}}{N-1} \tag{7}$$

$$\text{Aroon}_{Down} = 100 \cdot \frac{(N-1) - \tau_{low}}{N-1} \tag{8}$$

The Aroon Oscillator is defined as the difference: $AO = \text{Aroon}_{Up} - \text{Aroon}_{Down}$.

Furthermore, we apply a Stochastic transformation to the Aroon Oscillator itself (rather than price) to normalize its values. We compute the K and D on the AO series over a 14-period window:

$$K_{Aroon} = 100 \cdot \frac{AO_t - \min(AO_{14})}{\max(AO_{14}) - \min(AO_{14})} \tag{9}$$

$$D_{Aroon} = \text{SMA}(K_{Aroon}, 3) \tag{10}$$

3.5 Target Variable

The prediction task is formulated as a binary classification problem. The target variable Y_t indicates whether the log-return at time t is positive:

$$Y_t = \begin{cases} 1 & \text{if } r_t > 0 \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

3.6 Signature Features

The signature is an infinite dimensional vector that captures the shape of a path, which is in our case here a time series. More precisely, the signature is defined as follows:

Definition 3.1 (Signature) *Let $X : [0, T] \to \mathbb{R}^d$ be a continuous path. The signature of X over an interval $[s, t] \subset [0, T]$ is defined as an infinite series of tensors indexed by the signature order $n \in \mathbb{N}$,*

$$Sig_{s,t}(X) := \left(1, Sig^1_{s,t}(X), Sig^2_{s,t}(X), \dots, Sig^n_{s,t}(X), \dots\right), \tag{12}$$

where

$$Sig^n_{s,t}(X) := \int_{s<u_1<\cdots<u_n<t} dX_{u_1} \otimes \cdots \otimes dX_{u_n} \in (\mathbb{R}^d)^{\otimes n}. \tag{13}$$

We choose a lookback window of 20 minutes and we truncate the signature up to level 2 (hence we have 6 features). This choice of parameters allows faster calculations of the first six signature features that already provide some information about our time series (see table 3 in the appendix about the feature importance for the tree model). We calculate the signature features of $(t, log(R_t))$, where R_t is the cumulative log returns, because cumulative log returns are more stable than prices, given that the latter are very volatile for cryptocurrencies, and the fact that the more regular the path is, the better the truncated signature captures relevant information about it. The choice of $(t, log(R_t))$ instead of only R_t is due to the fact that the truncated signature performs better in a multidimensional setting and captures more information about the path. One justification for that is the fact that signature of a 1-dimensional path X_t is invariant to time reparametrization, so $sig(X_t) = sig(X_{\phi(t)})$ where ϕ is a time reparametrization function. However, by introducing the time component, the signature uniquely determines the 2D-path (t, X_t).

In order to calculate (approximate to be more precise) the signature features, we use the library iisignature. The function that calculates the signature features in our setting can also be found on our Github.

A small observation regarding the sig function of the iisignature library is that it does not return the first component which is always equal to 1 as stated in the definition above. Another observation concerns the features sig_0 and sig_2. These two are constant and do not provide any relevant information to train our models, therefore, we decided to remove them from the features we feed our models. Note that we have $sig_0 = 1 = \int_0^1 dt$, and $sig_2 = 0.5 = \int_0^1 tdt$.

4 Regime prediction using Machine Learning

In this section we describe the different standard machine learning models (logistic regression, neural networks, and classification trees) that we use to predict the regimes of the 5 cryptocurrencies.

4.1 Logistic Regression

Given a binary classification problem, the logistic regression will assign a probability to each row of features. Let N be the number of samples, the loss function for logistic regression is:

$$\min_{\mathbf{w},c} \frac{\mathbf{w}^T\mathbf{w}}{2} + C\sum_{i=1}^{N} \log\left(\exp\left(-y_i(\mathbf{x}_i^T\mathbf{w} + c)\right) + 1\right).$$

Here C controls the effect of the penalty term. In this paper, we select the constant C by k fold cross validation on training set. The results we get for logistic regression can be found in table 2.

4.2 Neural Networks

Instead of using multilayer perceptron as in [1], we used TabNet as in [3]. TabNet is a high-performance and interpretable canonical deep tabular data learning architecture. It uses sequential attention to choose which features to reason from at each decision step and it can give importance scores of each feature like in the tree method. It can also be used without any feature preprocessing. The results we get with this model can be found in table 3.

Table 2: Accuracy of the logistic regression model on the test set.

Cryptocurrency	Accuracy	Benchmark
BTC	0.520	0.54
ETH	0.529	0.53
BNB	0.513	-
SOL	0.517	-
XRP	0.529	0.53

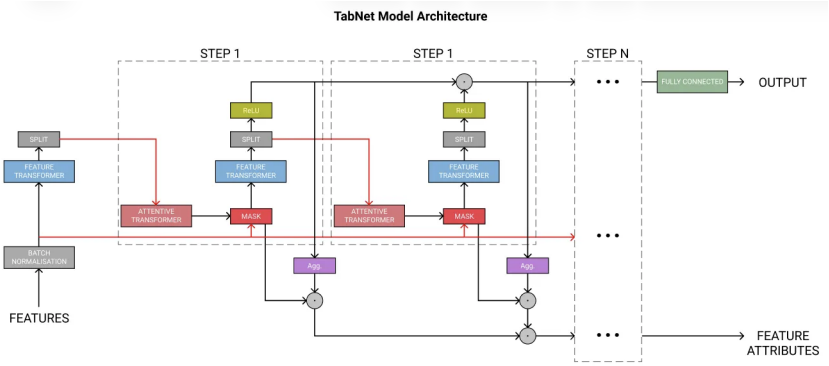


Figure 2: An illustration of the architecture of TabNet

4.3 Tree models

Instead of the famous XGBoost or Random Forest (the latter was used in [1]), we use a LightGBM model that is more efficient and is faster to train. However, since the latter is known for overfitting very easily, we use a high number of estimators that each have a small maximum depth. This avoids the model to build deep trees that end up only learning noise and overfitting on it. Although this slows down the training process, it is still faster than XGBoost and RF and achieves equivalent results. The parameters we use to train our model can be found in table 5, and Table 4 shows the accuracy obtained on the test set for each of the 5 cryptocurrencies. We also have the features importance in the training process in the appendix in figure 3.

LightGBM

LightGBM was created specifically to deal with big datasets (large number of instances and large number of features). It solves the following non-trivial problem: how to reduce the number of data instances and the number of features without loosing performance. Sampling uniformly randomly from the features might impact the performance especially when we pick features that do not contain meaningful information about our dataset. Therefore, two techniques were suggested to solve this issue:

- Gradient-based One-Side Sampling (GOSS): the idea behind this technique is that data instances with larger gradients (i.e. under-trained instances) will contribute more to the information gain. Hence, when down sampling the data instances, it is better to keep

Table 3: Accuracy of the TabNet model on the test set.

Cryptocurrency	Accuracy	Benchmark
BTC	0.524	0.52
ETH	0.524	0.50
BNB	0.529	-
SOL	0.518	-
XRP	0.516	0.52

instances with large gradients (larger than a threshold for example), and randomly drop instances with small gradients, which would allow to retain the accuracy of information gain.

- Exclusive Feature Bundling (EFB): the idea here is that when we have a large number of features, many of them are (almost) exclusive (i.e. they rarely take nonzero values simultaneously), an example of that being one-hot encoding. To this end, it is possible to reduce the optimal bundling problem to a graph coloring problem (by taking features as vertices and adding edges for every two features if they are not mutually exclusive), and solving it by a greedy algorithm with a constant approximation ratio.

More information about LightGBM (including proofs and complete algorithms) can be found in [4].

Table 4: Accuracy of the LightGBM model on the test set.

Cryptocurrency	**Accuracy**	**Benchmark**
BTC	0.5345	0.53
ETH	0.5283	0.51
BNB	0.5012	-
SOL	0.5144	-
XRP	0.5159	0.52

Table 5: Parameters used to train the LightGBM model for each cryptocurrency.

Parameters	**value**
learning rate	0.001
max depth	64 (8 for SOL and BTC)
number of jobs	-1
early stopping round	500
number of estimators	10000
feature fraction	0.5
bagging fraction	0.5
bagging frequency	2
min data in leaf	20

5 Regime prediction using Randomized Signature

We follow the randomized signature method similar to [2]. We consider our classification problem as a path space functional. Signature of path provides a infinite dimensional regression basis.

Consider a non-linear function of path $X \to f(X)$, we can approximate this function via a linear function of the signatures $\mathrm{Sig(X)}$. This is stated as the following universal approximation property:

Proposition 5.1 *Let $\mathcal{V}_1([0,T];\mathbb{R}^d)$ be the space of continuous paths from some interval $[0,T]$ to $\mathbb{R}^d$. Suppose $\mathcal{K} \subset \mathcal{V}_1([0,T];\mathbb{R}^d)$ is compact and $f : \mathcal{K} \to \mathbb{R}$ is continuous. For any $\varepsilon > 0$ there exists a truncation level $n \in \mathbb{N}$ and coefficients $\alpha_{\mathbf{J}} \in \mathbb{R}$ such that for every $X \in \mathcal{K}$, we have*

$$\left| f(X) - \sum_{i=0}^{n} \sum_{\mathbf{J} \in \{1,\dots,d\}^i} \alpha_{\mathbf{J}} Sig_{a,b}(X) \right| \leq \varepsilon. \tag{3}$$

We can use signatures to form the reservoir in Reservoir Computing which is a regression basis on path space. We will focus on training a static and memory-less readout map such as linear regression over this reservoir. However signature suffers from curse of dimension since the dimension grows exponentially with the order of signature. This can be overcome by Randomized Signatures. Assume $X(t) : \mathbb{R} \to \mathbb{R}^d$ be a path. We construct randomized signature as follows: First fix an activation function σ, a set of hyper-parameters θ and a dimension r_d. Then depending on θ, we choose random

matrices $A_0, \ldots, A_d \in \mathbb{R}^{r_d \times r_d}$ and shifts $b_0, \ldots, b_d \in \mathbb{R}^{r_d}$. For the random matrix A_i each entry follows a normal distribution with mean r_m and variance r_v, and the entries of the random biases $b_i \in R^{r_d}$ follow a standard normal distribution. We have $\theta = (r_m, r_v)$. Similarly to the universal approximation property of signature, One can tune the hyper-parameters θ and dimension r_d such that paths of

$$dR_t = \sum_{i=0}^{d} \sigma(A_i R_t + b_i)\, dX_t^i, \quad R_0 \sim N(0, I_{r_d})$$

approximate path space functionals of $(X_s)_{s \leq t}$ via a linear readout up to arbitrary precision. Let w be length of the time points in the past over which we calculate the randomized signature, and we define:

$$\mathcal{F}^{r_d, r_m, r_v} : \mathbb{R}^{w \times d} \longrightarrow \mathbb{R}^{r_d}$$

be a numerical scheme solving the equation (5).

For each coin, our algorithm can be written as:

Input: A path $\{X_t \in \mathbb{R}^5\}_{1 \leq t \leq T}$, where X_t^0 is time, $X_t^1, \ldots, X_t^4$ are calculated as the first differences of log prices after normalization by their initial value. They are calculated with respect to close, open , high and low prices respectively.
Output: Prediction for $(\mathrm{Sign}(X_{t_s+1}), \ldots, \mathrm{Sign}(X_T))$ denoted as $(\hat{X}_{t_s+1}, \ldots, \hat{X}_T)$
Set $(Y_{t_w}, \ldots, Y_T) := (\mathcal{F}_{r_d, r_m, r_v}(X_0, \ldots, X_{t_w-1}), \ldots, \mathcal{F}_{r_d, r_m, r_v}(X_{T-t_w}, \ldots, X_{T-1}))$
Set $t_s = \text{round}(9T/10)$, ensure that T is large enough such that $t_s \gg t_w$;
Train a ridge regression model:
$\mathcal{M}_t \leftarrow$ fit RidgeRegression$(\{Y_{t_w}, \ldots, Y_{t_s-1}\}, \{\mathrm{Sign}(X_{t_w+1}), \ldots, \mathrm{Sign}(X_{t_s})\})$;
Predict: $(\hat{X}_{t_s+120}, \ldots, \hat{X}_T) = (\mathcal{M}_{t+119}(Y_{t+119}), \ldots, \mathcal{M}_{T-1}(Y_{T-1}))$;
return predictions $(\hat{X}_{t_s+120}, \ldots, \hat{X}_T)$;

Table 6: Accuracy of the random signature model on the test set.

Cryptocurrency	Accuracy
BTC	0.649
ETH	0.626
BNB	0.645
SOL	0.645
XRP	0.672

6 Discussion of the results

An important note before comparing our results to those of the benchmark [1] is that they round their accuracy to the nearest percentage. Also note that BNB and SOL did not exist yet when [1] was written, which is why we do not have a benchmark for these two cryptocurrencies, and that our respective datasets cover different time periods.

For logistic regression, as we can observe on table 2, we obtain similar results for ETH and XRP (modulo the rounding), but we achieve a worse performance for BTC. An explanation for this underperformance is that they use 5 years of data for BTC while we use less than 2 years, and the more data you feed a logistic regression, the better it performs.

For the neural network model and tree model, we obtain similar accuracy for BTC and XRP, and we outperform the benchmark for ETH. An explanation for our outperformance is that ETH was created in 2015, and was not very liquid in its beginning compared to now. We can also see that some of the signature features rank high in the feature importance of the tree models (see figure 3 in the appendix).

The very surprising results were given by the randomized signature (see table 6), which beats by far all our previous models. We checked carefully our code and there is no lookahead bias, but we invite the reader to also do so.

7 Extension/Bonus: Portfolio optimization using Randomized Signature

We follow a similar framework as in [2]. The idea is to use randomized signature to predict the next returns and feed them into a portfolio optimization framework. This would allow us to use these predictions instead of the historical mean for example in our objective function (that maximizes the sharpe ratio for example). However, the issue is that randomized signature is good for short term predictions, but performs poorly in long term predictions. The main problem is that this would work well if we can predict the expected returns between two rebalancing periods, but since we can only do this in a short horizon, we would be constrained to a very frequent rebalancing, and hence high transaction costs.

We have not found a way yet to overcome this problem, which is why this section is labeled as an extension, open for future improvements. One idea is to use lower frequency data, but on the other hand, the high volatility of cryptocurrency prices could be a problem.

8 Conclusion

Cryptocurrency markets have evolved into a critical alternative asset class, attracting significant attention from the global investment community. In this study, we investigated the predictability of five major cryptocurrencies using a combination of machine learning algorithms and stochastic analysis. All employed models achieved accuracy scores exceeding 0.5, indicating that short-term cryptocurrency returns possess inherent predictability that can be captured by these methods. Notably, Reservoir Computing with randomized signatures emerged as the superior approach for $15\mathrm{min}$ data intervals, outperforming standard machine learning models by a significant margin of over 0.1 in accuracy.

References

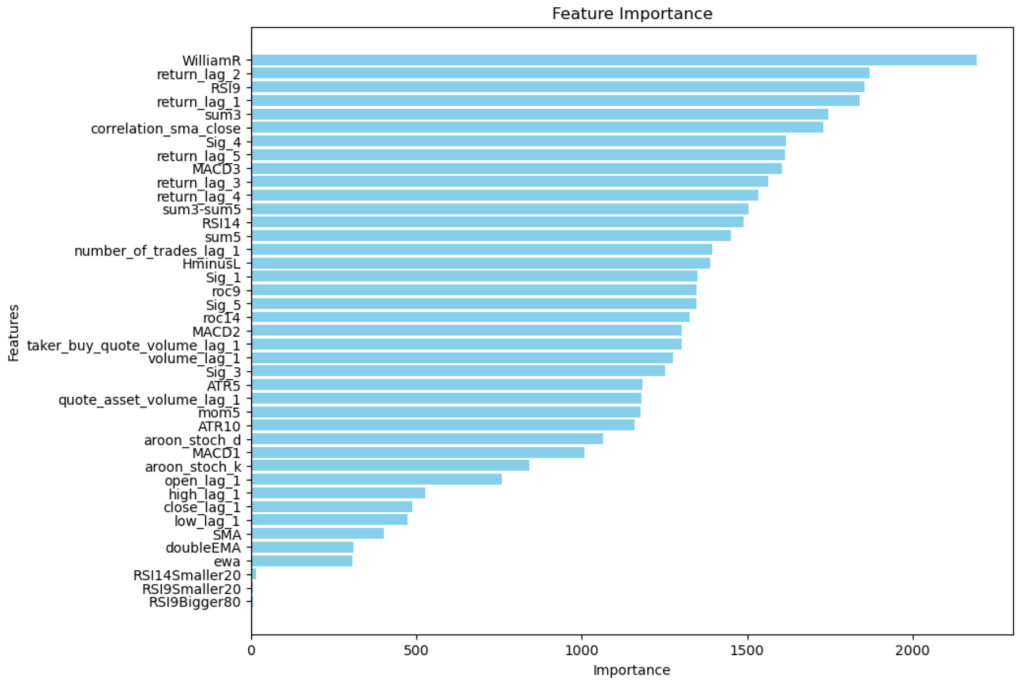
[1] Erdinc Akyildirim, Ahmet Goncu, and Ahmet Sensoy. "Prediction of cryptocurrency returns using machine learning". In: *Annals of Operations Research* 297.1 (Feb. 2021), pp. 3–36. DOI: 10.1007/s10479-020-03575-y. URL: https://ideas.repec.org/a/spr/annopr/v297y2021i1d10.1007_s10479-020-03575-y.html.

[2] Erdinç Akyildirim et al. "Randomized signature methods in optimal portfolio selection". In: *Quantitative Finance* 25.2 (2025), pp. 197–216. DOI: 10.1080/14697688.2025.2458613. eprint: https://doi.org/10.1080/14697688.2025.2458613. URL: https://doi.org/10.1080/14697688.2025.2458613.

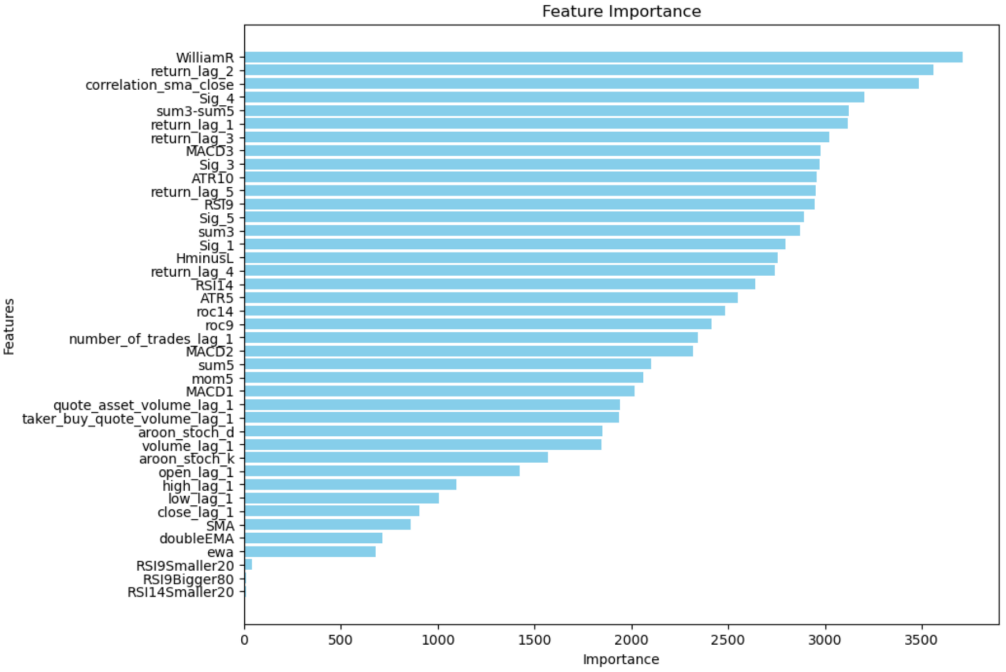
[3] Sercan O. Arik and Tomas Pfister. *TabNet: Attentive Interpretable Tabular Learning*. 2020. arXiv: 1908.07442 [cs.LG]. URL: https://arxiv.org/abs/1908.07442.

[4] Guolin Ke et al. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

9 Appendix

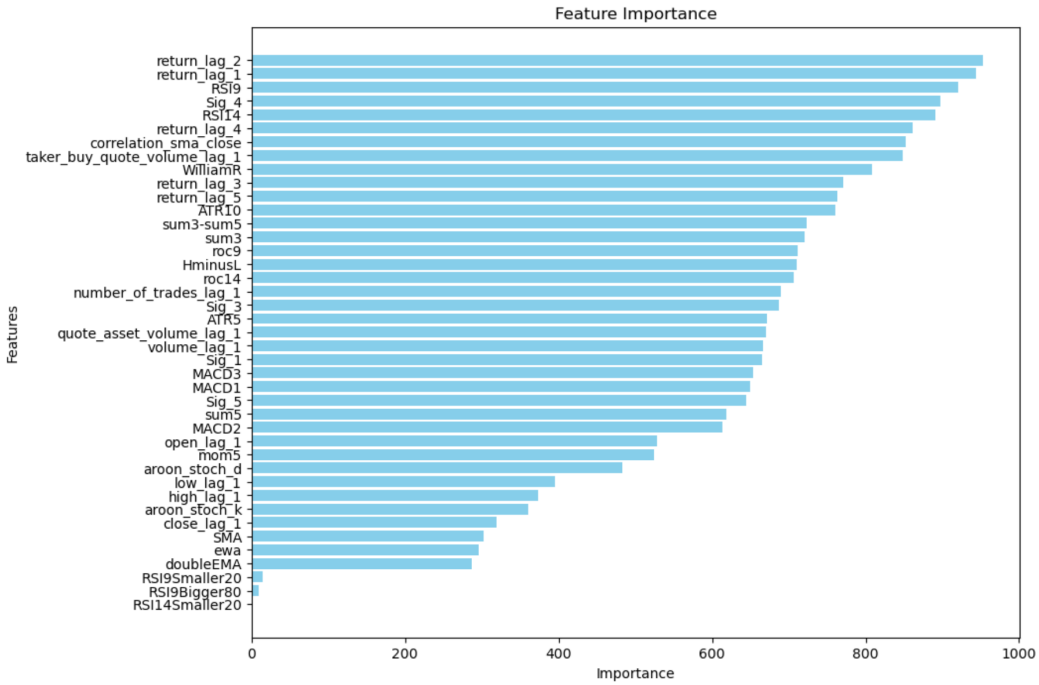


(a) BTC

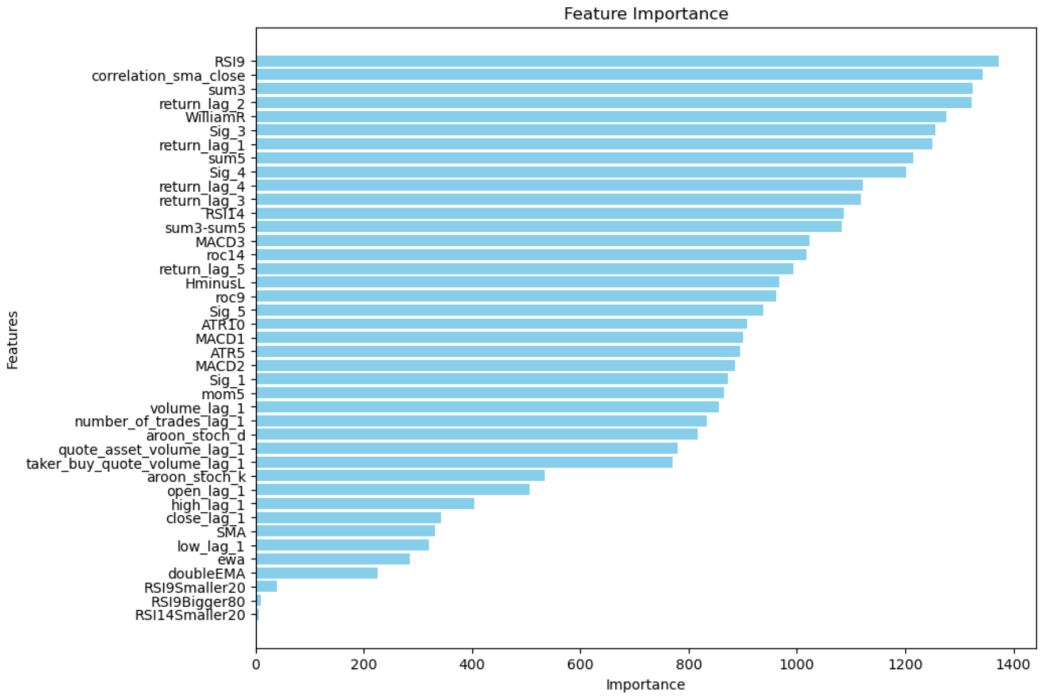


(b) ETH

Figure 3: Feature importance in the training process of the LightGBM model for the 5 cryptocurrencies.

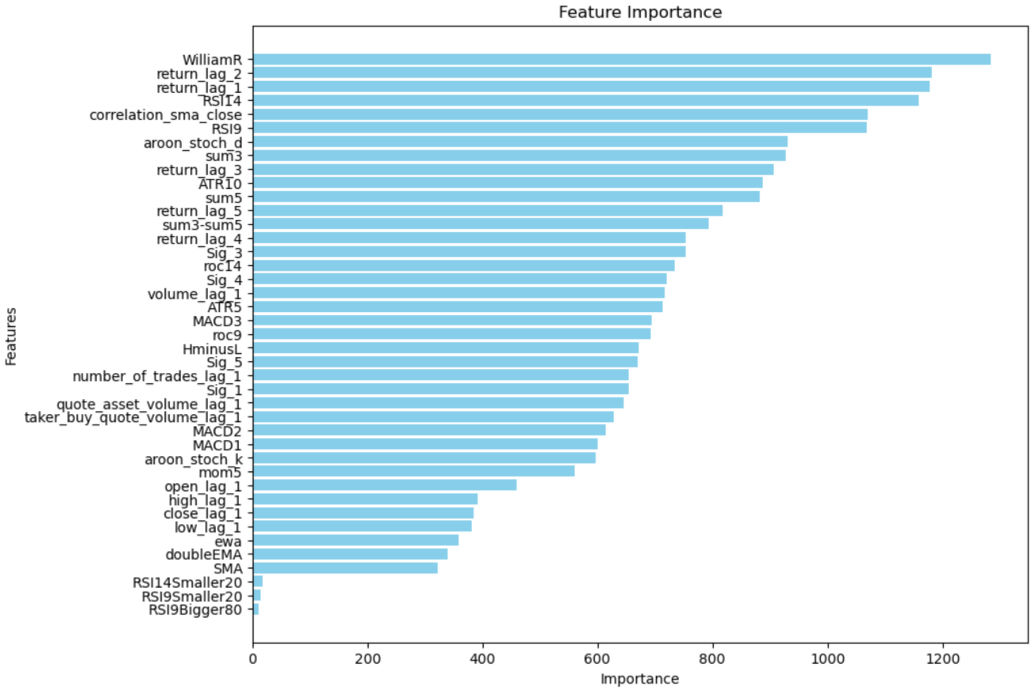


(c) BNB



(d) SOL

Figure 3: Feature importance (continued).



(e) XRP

Figure 3: Feature importance (continued).