Instituto Tecnológico
y de Estudios Superiores de Occidente

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Department of Mathematics and Physics
Master in Data Science

Cryptocurrency Forecasting Models and DeFi

Thesis to obtain the **Degree** de
MASTER IN DATA SCIENCE

A thesis presented by: **Carlos Emilio Carranza Avila**

Thesis Advisor: **Juan Francisco Muñoz Elguezábal**

Tlaquepaque, Jalisco, December, 2022

Cryptocurrency Forecasting Models and DeFi

Carlos Emilio Carranza Avila

Abstract

The nature of Cryptocurrency markets presents a challenge for Financial Time series forecasting, the regular use of time bars as a source of data to forecast can prove insufficient to predict the movements of the crypto token value. The use of additional data from DeFi sources can be used to create a more robust base in which to use different methods to perform better feature generation and feature selection to use for the prediction models. The use of the Three Barrier Method for labeling the movements of the data is suggested as a way to generate multiclass labeling in which both directions of the prices and magnitude are represented. The proposal of this work is that the use of DeFi data, the adapted use of the three-barrier method, and the use of Genetic Programming could create a dataset that has good predictor capabilities for the multiclass classification prediction of the movement and magnitude of the value of Bitcoin. In this work, a comparison between prediction models is performed using a combination of benchmark models, and the implementation of Random Forest and Multi-Layer Perceptron to construct a multiclass classifier for the price movement of the cross of Bitcoin and USDT from the Binance Exchange using historical data from Binance, Ethereum Blockchain, and symbolic data.

Contents

Page

List of Figures

List of Tables

Acknowledgments:

Dedicated to my Mother and Father, who always supported me unconditionally, without them, nothing would have been possible, to Maria Jose Cano Pagaza, who always pushed me to be better and whose unconditional love and support made this possible, to Sebastian Balart Sanchez and Angel Pereida Vega who always listened and gave me their support all these years, to Eduardo Balart Sanchez, for introducing me to the world of Machine Learning, to Juan Francisco Toral Cañedo, who struggled together with me on this journey since the beginning, To Juan Carlos Martinez Alvarado, who always pushed me to learn more and give my best, to my Brother, Daniel Eduardo Carranza Avila, for always being there for me in the difficult times.

1 Introduction

Contents

This chapter included content about the three significant aspects of this work: the problem's context, the Financial Machine learning approach to the problem, and finally, the document's structure.

1.1 Context-Problem

Introduced in 2008, bitcoin[1] was presented as a decentralized currency that would eliminate the need for central banks or an intermediary for the production of the currency. Satoshi Nakamoto's invention began small; in 2010, for example, there was a transaction of 10,000 bitcoins for only two pizzas in Florida; as of November of 2022, bitcoin has a market cap of 318.37 Billion dollars.

[1] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system

In the following years, the growth of bitcoin as an alternative to fiduciary currency gained the attention of the financial industry, with an average growth of 156.8%. Bitcoin has become a great form of investment; with such competition to the traditional markets, it was essential to improve the quality of predictions and the velocity with which the industry can predict the future values or trends of the cryptocurrencies; because of this, machine learning has become inseparable from the forecast of financial assets on the industry.

However, as of 2022, bitcoin has lost 71.76% of its value since last year, and the conditions of the market are volatile; this volatility can be measured as 1.74714% on the previous 30 days estimate as of November 2022[2], which can be explained as bitcoin is an asset with a finite market cap which is small in comparison with other assets so that every transaction will have a more significant effect on the price.

[2] yahoo finance (2022)

This volatility can be seen in price changes between 5 to 10 percent of the daily market prices. This makes this cryptocurrency a volatile asset, making the importance of an accurate forecast much more urgent.

In this work, the predictability of the movements of bitcoin was analyzed using various machine learning models to reach an optimized model that could accurately predict the cryptocurrency's movement. This uses different methods to treat the data before the training period to obtain the optimal data preprocessing techniques and the best models available to get better results.

1.1.1 Cryptocurrencies markets

As of 2022, cryptocurrencies run on the blockchain, which is a public ledger that records all transactions held by the crypto holders; the cryptos are generated through the mining process, in which computational power is used to solve complex mathematical problems and generate tokens or coins[3], later this coins can be exchanged through brokers or stored in wallets.

[3] James Martin, Jack Cunliffe, R. M. (2019). *Cryptomarkets: A Research Companion*. Emerald Publishing Limited

For this work, the exchange used to collect the data was Binance; this exchange was chosen because it is the largest one in the world in terms of daily trade of volume of crypto; Binance was founded in 2017 and is currently registered in the Cayman Islands, the determining factor in choosing it is because being the largest market in the world means that Binance has a more sophisticated market than other exchanges and that taking into account that bitcoin is susceptible to large volume movements Binance is a good representation of the value of bitcoin in the cryptocurrency market[4].

[4] Binance (2022)

1.1.2 Defi

Another innovation came with the creation of Ethereum; this cryptocurrency had different comparing it to Bitcoin; both are digital currencies, but Ethereum is programmable, which permits the deployment of applications on the Blockchain of Ethereum[5].

[5] Ethereum.org (2022). What is ethereum

This means that while Bitcoin is a coin that can be used to pay for services, Ethereum is a market for different kinds of financial services.

One of those innovations was Smart contracts, which run on the blockchain and execute functions according to their programming, making transactions; these are final and cannot be reversed.

This makes the decentralized exchange also referred to as DEX. On this important platform, much information about the behavior of cryptocurrencies is held, besides the information on other exchanges. Because the transactions in Defi using DEX are Peer to peer, the information is on the blockchain instead of the data from regular exchanges.

One of the most important DEX is Uniswap, which began in 2018, with transactions of more than half a million dollars a day, open source contracts, and over three billion in assets in the protocol; this makes Uniswap a great option to obtain more information about the behavior of cryptocurrencies.

1.1.3 Financial TimeSeries Forecasting

Time series analysis is the practice of taking information from statistical information from points arranged in chronological order, using the past to predict the future, and parts from the question of how the past influenced the future.

In the case of financial time series, with the advent of financial markets and the use of stocks in time, these became technical and advanced, which needed better techniques to predict the value of the assets in the future, so the use of financial time series began to rise in the beginning they did this using mathematical model by hand, and with the continuous sophistication of the model's used machine learning models currently being used.[6]

[6] Nielsen, A. (2020). *Practical Time Series Analysis*. O'Reilly

1.2 Financial Machine Learning

Through the years, the industry became more technical, and the advances of machine learning got the interest of the Financial world; with the advent of automation and its use in finance, it became essential to get better and more efficient methods to forecast the future value of an asset[7], The use of algorithms to predict future values of assets became one of the more critical aspects of Financial forecasting,

[7] Lopez de Prado, M. M. (2018). *Advances in Financial Machine Learning*. Wiley

Another significant advance in machine learning was the incursion of genetic programming[8], a method to generate computer programs based on biological evolution, taking unfit programs and, through stochastic transformations obtaining new ones and continuing the same process for generations until getting the expected result.

[8] Gplearn (2016). Introduction to gplearn

For machine learning, one application for genetic programming is the generation of symbolic variables; using evolution methods through transformations, one can create new variables that have a high correlation with the target variable.

1.3 *Document structure and Implementation Notes*

The composition of this work will be the following: chapter 2 will contain a more detailed problem definition for this work, in which the creation of the target feature and the problem to tackle will be explained; chapter 3 is about the methodology used for the Data ETL and the generation of the dataset used; as well as the sources of the datasets used; Chapter 4 will explain the different models and the predictive modeling process used for this work. In chapter 5; the results will be presented; furthermore in chapter 6; a discussion about the results and the future work needed based on the results.

2 *Problem Definition*

Contents

In this chapter, the definition of the problem is explained in the two fundamental parts that compose classification forecasting: the fundamentals of financial time series forecasting and the particularities of a time series of a cryptocurrency asset. The problem type is also defined, explaining the idea behind the choice of a multi-class classification for this particular case being tackled in this work.

2.1 *Financial Time Series Forecasting*

The modeling process working with time series typically begins with the premise that the time series is dependent on temporality; this means that the chronology of the data cannot change, so certain precautions have to be taken into account to maintain it. The basis of the forecast with time series is that the previous data contains information that can help the model predict the future value or label depending on the problem being tackled.[1]

[1] Nielsen, A. (2020). *Practical Time Series Analysis*. O'Reilly

In contrast with other areas of time series forecasting, for example, in image recognition, one can obtain samples of a physical object, having certainty that the model will have the same molecular composition making the representation maintained, whereas in the case of financial time series will not be, in other comparisons the uses of time series in health, for example in the use of ECG for heart monitoring, is the same case, as that is a physical object. The general behavior of the heart will not change as much, having certainty that it will behave between the horizon marked by being a human organ, making that the representation is maintained as well.

In the case of financial time series, you take one sample of a stochastic process from which no one has a hundred percent certainty of the representation, and it is impossible to take a sample from a past process. Hence, the use of data is quite different from other kinds of time series forecasting. Moreover, given that the price of an asset is a non-repeatable sample, which is taken from an unknown and ever changes stochastic process, considering the sparsity of information is of the utmost importance. This can be assessed by using techniques based on time-aware sub data sets generators, where the entire data set is divided into subsamples, each of which represents different distributions the target variable takes and thus provide an increase in sparsity in target variable representation, with which out-of-sample and even out-of-distribution generalization can be achieved under certain conditions[2].

[2] Muñoz-Elguezábal, J. F. and Sánchez-Torres, J. D. (2021). T-fold sequential-validation technique for out-of-distribution generalization with financial time series data. International Conference on Econometrics and Statistics

In time series, two methods are usually used for forecasting: Regression and Classification. The first is used to predict the asset's value using the time series information and indicates the asset's future value. The second is used for classification, which makes a forecast of the different classes assigned to the different samples used for the problem.

In this work, the methodology chosen was to frame the problem as a classification, meaning that a label had to be created in which all periods would be classified as one of the classes created for it.

In the case of classification, the $\hat{y}$ is represented as $\in \{0,1\}$ in case of binary problems and $\in \{0,1,..., N\}$ in case of multi-class cases. For this work, the multi-class classification was chosen because the objective of the target variable was to represent the magnitude as well as the direction of the value of bitcoin.[3]

[3] López de Prado, M. M. (2020). *Machine Learning for Asset Managers*. Cambridge University Press

2.2 *Cryptocurrencies Markets Dynamics*

In principle, crypto markets function with the same process that a regular stock exchange would behave, with the bonus that it is decentralized; as of November 2022, there are more than nine thousand cryptocurrencies worldwide, with the most important ones being Bitcoin and Ethereum in that order. These cryptocurrencies have different ways to be obtained; the first one is through mining, where to solve complex mathematical problems, the computer or server mines the token.

The second one is through exchanges. As of November 2022, Bitcoin's total market capitalization rests at 310 billion dollars, down from 1.26 trillion dollars in November 2021, which shows the kind of volatility the token has. There are many cryptocurrency exchanges; the most important one is Binance, having a market cap of 47 billion dollars as of November 2022.[4]

[4] yahoo finance (2022)

The way the exchanges work is through bids and asks, and has many different symbols or tokens available to exchange; one example of this is the one used for this work, which is a cross between bitcoin and USDT, a representation in the token form of U.S. dollar, having the same value as that currency. When wanting to partake in a transaction, it's organized in two ways, bidding high and asking low comes first. It's organized in rows, where the first one of the row comes first in the transaction, in a way meeting in the middle when a bid and ask are the same, the transaction happens, and all this is recorded in the order book of the token.[5]

[5] James Martin, Jack Cunliffe, R. M. (2019). *Cryptomarkets: A Research Companion*. Emerald Publishing Limited

With the arrival of Ethereum, a new way to exchange crypto tokens became available, which are the Smart Contracts, which are programs that run through the blockchain and execute transactions that cannot be reversed automatically; these Smart Contracts come programmed with information about the quantity, ask or bid for the trade and are deployed to the blockchain to wait for the transaction, for this work the data obtained was from the Uniswap from Ethereum, compiling data from the cross between BTC and USDT on the Ethereum blockchain.[6]

[6] Ethereum.org (2022). What is ethereum

2.3 *Target Variable Formulation*

For this classification problem, the use of a label was chosen; a label is a representation of one or more characteristics of the target; in this case, the target was to obtain information both on the magnitude of the price change as well as the direction of the movement, to achieve this the objective became to create a multi-class classification system which was represented with the following numbers $\hat{y} \in \{0, 1, 2, 3\}$, for the formulation of the target the values of Open and Close would be used to create the label.

$$CO_t = Close_t - Open_t \tag{2.1}$$

Using the result of CO_t, the classes were assigned using thresholds based on the quintiles of the total values of it, assigning a class to each period evaluated; the thresholds were not fixed and changed depending on the periodicity of the data, and the purpose of the moving threshold was to obtain a target variable that was balanced and had an equal representation in all the classes.

To get the price direction and magnitude, the algorithm would use the prices of $Open_t$ and $close_t$ in conjunction with the threshold calculated from the quintiles at 25% and 75% to generate the four labels that encompass both magnitude and price direction. The following is the formulation of the target variable as will be used in this work.

$$\hat{y}_t = sign\{Close_t - Open_t\} \tag{2.2}$$

The label formulation that became the target was not as effective as it will be described in chapter 4, so it became essential to generate the target with a more sophisticated formulation. The one chosen was the triple barrier method of label generation, as the last method, also known as fixed time horizon, uses fixed time intervals and fixed thresholds, which do not represent the way the market functions, so in his book, Lopez de Prado suggests a method called triple barrier method or TBM.[7]

[7] López de Prado, M. M. (2020). *Machine Learning for Asset Managers*. Cambridge University Press

TBM is a method in which dynamic thresholds are used, based on daily volatility allowing for more optimal thresholds than the other method; in his proposal, De Prado uses barriers that mark events, and rather than fixing the thresholds, they can be interpreted as profit taking or stop loss. This labeling method produces labels in binary form, with -1 being don't sell, and one as well, and all other periods without touching the barriers come with a percentage of how close it is to that barrier; in this work, this was reworked to produce four labels, with the intent to implement a labeling process to take into account magnitude and direction of the asset.

$$X_{t-n:t-1} = \phi Data \tag{2.3}$$

2.4 Features Formulation

The Features used to train the models were sourced from different places; the first one, as previously stated, is the time bars that came from the order book from Binance, which will be referred to as the OHLCV; from Binance also came the order book data; and the public trades data, that represent the transactions as well as ask and bids in the exchange, this will be the endogenous data.

On the other hand, exogenous data is the data that came from outside the exchange; this data presents more variety and is used to obtain more variables with the assumption that it will help to create better models once it is transformed. one of those exogenous datasets is the Defi data, that came from the liquidity pool which is cryptocurrency locked in smart contracts, the two most important DeFi exchanges are SushiSwap and Uniswap, with the latter being the bigger of the two, and use liquidity pools on the Ethereum blockchain, for this work the data used was sourced from Uniswap, using a cross between BTC and USDT, more of the content and transformation of this datasets will be explained on chapter 3.

Having those datasets, the other kind of data that will be used is created based on the previous data, the first of these methods is to use stochastic processes to generate more data using the time series to extract new data from the time bars—using lags, means, and differences to create new features. The second is through evolutionary computing, which is the process of creating new programs using a method that mimics evolution and is now applied to machine learning, through a symbolic transformer, which makes symbolic features using math equations and "breeding" new features[8] in a process that used multiple epochs to create new data that represents the hidden relation between the original features and the symbolic ones, more of this process will be presented in chapter3.

[8] Taber, K. (2009). Learning at the symbolic level. *Models and Modeling in Science Education*, 4:75–105

3 Data and Methods

Contents

In this chapter, the data sources will be discussed, presenting the different datasets used, their generation and composition, and an explanation of how the final dataset is compiled.

3.1 Data Sources

The process that was conducted to obtain the data from the mentioned sources is the following: First, for the case of Binance, it was obtained from the Binance Vision source and the data in particular is the Open, High, Low, Close, Traded Volume (quote), and Traded volume (base) of the selected symbol.

Also, from Binance, it was obtained the Full OrderBooks of the same symbol, along with the PublicTrades. and thus, the overall, the complete set of data that is used for this work can be stated within the following categories:

In the case of de Decentralized Finance type of data, it was obtained by the use of an external service, Bitquery.io, which provided a Graphical User Interface to construct queries for the GraphQL-based API, having in that sense, free access to the transactions history of the Ethereum Blockchain, in particular transactions To and from a specific smart contract named "Uniswap V3", a liquidity pool is operating entirely in the blockchain.

- Centralized Exchange Activity: OrderBooks, PublicTrades, OHLC.
- Decentralized Exchange Activity: Liquidity Pool.

3.1.1 Binance

The data was obtained from Binance vision[1], which permits downloading the compiled data in OHLCV format from a repertoire of indexes and different time metrics, the dataset used for this project was composed of monthly OHLCV datasets from January 1st, 2021, to October 31st, 2022, with a period frequency of 1 minute for a total of 962,367 minutes measured and nine different features, besides a timestamp as this is a financial time series this can be seen on figure 3.1.

[1] Binance (2022)

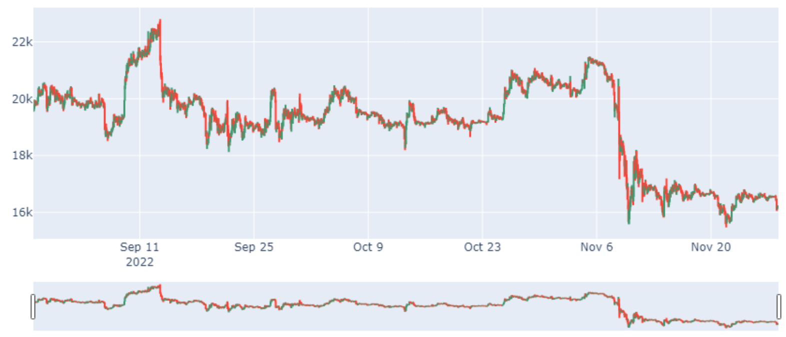
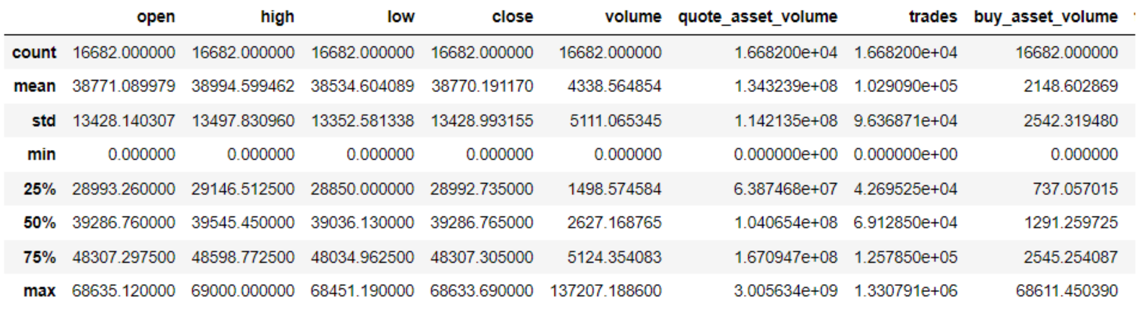


Figure 3.1: OHLCV Plot
OHLCV plot of finance data, period of September through November 2022

The data that compose the dataset are numerical, discrete, and linear. the target feature created is categorical and ordinal. later, autoregressive data was designed to generate more features, and genetic algorithms were used to develop genetically generated features. The first dataset is composed of the following features[2]:

[2] Binance (2022)

- Timestamp: Index, the periods of 1 minute.
- Open: The first price of the period.
- High: The highest value of all the prices per period.
- Low: The lowest price value of the period.
- Close: Final price of the period.
- Volume: The total sum of bitcoins operated per period.
- Quote Asset Volume: Total quantity of USDT operated during the period.
- Trades: Total number of trades per period.
- Buy Asset Volume
- Taker Buy asset volume: Total quantity of BTC operated during the period.



	open	high	low	close	volume	quote_asset_volume	trades	buy_asset_volume
count	16682.000000	16682.000000	16682.000000	16682.000000	16682.000000	1.668200e+04	1.668200e+04	16682.000000
mean	38771.089979	38994.599462	38534.604089	38770.191170	4338.564854	1.343239e+08	1.029090e+05	2148.602869
std	13428.140307	13497.830960	13352.581338	13428.993155	5111.065345	1.142135e+08	9.636871e+04	2542.319480
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000e+00	0.000000e+00	0.000000
25%	28993.260000	29146.512500	28850.000000	28992.735000	1498.574584	6.387468e+07	4.269525e+04	737.057015
50%	39286.760000	39545.450000	39036.130000	39286.765000	2627.168765	1.040654e+08	6.912850e+04	1291.259725
75%	48307.297500	48598.772500	48034.962500	48307.305000	5124.354083	1.670947e+08	1.257850e+05	2545.254087
max	68635.120000	69000.000000	68451.190000	68633.690000	137207.188600	3.005634e+09	1.330791e+06	68611.450390

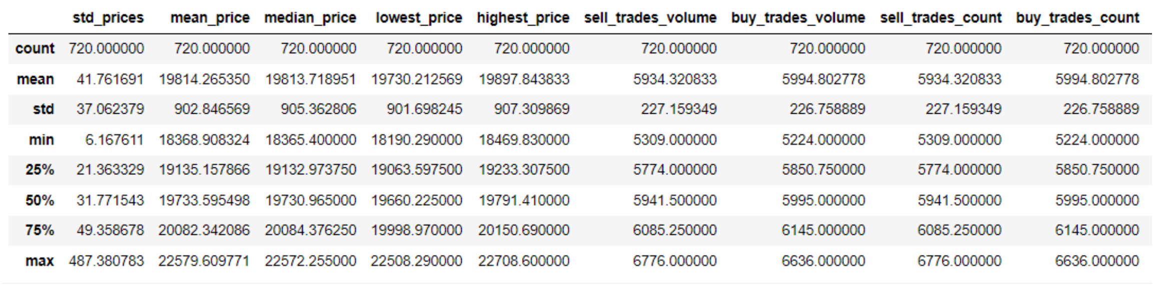
FIGURE 3.2: Data statistics
Statistics of the OHLCV data

Public Trades also was sourced from Binance; the data of the public trades during September was compiled and transformed using statistic methods; the dataset is numerical, discrete, and linear; the features are the following[3]:

[3] Binance (2022)

- std prices: This represents the standard deviation of the price.
- means price: Mean price during the period.
- median price: Median price during the hour period.
- lowest price: Lowest price during the sample.
- highest price: Highest price during the period.
- sell trades volume: Volume of trades sold.
- buy trades volume: Volume of trades bought.

- sell trades count: Number of sold events.
- buy trades count: Number of trades from the buyer side.
- quantile 25 volume: Quantile 25 of volume.
- quantile 75 volume: Quantile 75 of volume.
- Interquantile range volume: Measure of dispersion based on lower and upper quartile.
- trades count: Number of trades per period.
- trades volume: Total volume per sample.

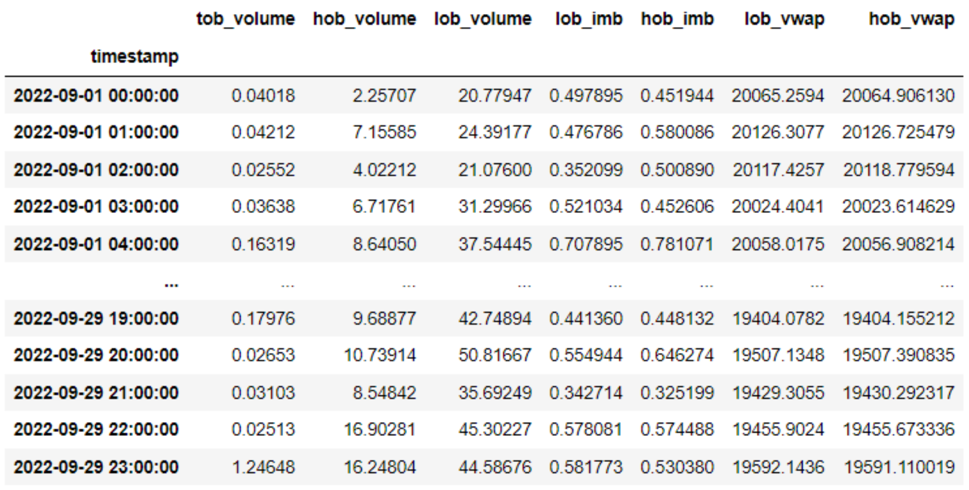


	std_prices	mean_price	median_price	lowest_price	highest_price	sell_trades_volume	buy_trades_volume	sell_trades_count	buy_trades_count
count	720.000000	720.000000	720.000000	720.000000	720.000000	720.000000	720.000000	720.000000	720.000000
mean	41.761691	19814.265350	19813.718951	19730.212569	19897.843833	5934.320833	5994.802778	5934.320833	5994.802778
std	37.062379	902.846569	905.362806	901.698245	907.309869	227.159349	226.758889	227.159349	226.758889
min	6.167611	18368.908324	18365.400000	18190.290000	18469.830000	5309.000000	5224.000000	5309.000000	5224.000000
25%	21.363329	19135.157866	19132.973750	19063.597500	19233.307500	5774.000000	5850.750000	5774.000000	5850.750000
50%	31.771543	19733.595498	19730.965000	19660.225000	19791.410000	5941.500000	5995.000000	5941.500000	5995.000000
75%	49.358678	20082.342086	20084.376250	19998.970000	20150.690000	6085.250000	6145.000000	6085.250000	6145.000000
max	487.380783	22579.609771	22572.255000	22508.290000	22708.600000	6776.000000	6636.000000	6776.000000	6636.000000

FIGURE 3.3: Data statistics
Statistics of the Public Trades data

The final dataset obtained from Binance Vision is the order book metrics. This one consists of the calculations performed on the order book data, which is composed of periods with a list of asks and bids for the particular asset.

- Top of the book volume: Volume of the first ask and bid per period.
- Head of the book volume: Volume of the first five levels of the book per period.
- Limit order book volume: volume of the entire period of the order book.
- Limit order book imbalance: Imbalance from the whole of the book per period.
- Head of the book imbalance: Imbalance between ask and bid from the top levels of the period.
- LOB volume weighted average price: mean of the volume weighted by the price of the whole book.
- Head of the book volume weighted average price: Mean of the volume weight of the top levels.



	tob_volume	hob_volume	lob_volume	lob_imb	hob_imb	lob_vwap	hob_vwap
timestamp							
2022-09-01 00:00:00	0.04018	2.25707	20.77947	0.497895	0.451944	20065.2594	20064.906130
2022-09-01 01:00:00	0.04212	7.15585	24.39177	0.476786	0.580086	20126.3077	20126.725479
2022-09-01 02:00:00	0.02552	4.02212	21.07600	0.352099	0.500890	20117.4257	20118.779594
2022-09-01 03:00:00	0.03638	6.71761	31.29966	0.521034	0.452606	20024.4041	20023.614629
2022-09-01 04:00:00	0.16319	8.64050	37.54445	0.707895	0.781071	20058.0175	20056.908214
...	...	...	...	...	...	...	...
2022-09-29 19:00:00	0.17976	9.68877	42.74894	0.441360	0.448132	19404.0782	19404.155212
2022-09-29 20:00:00	0.02653	10.73914	50.81667	0.554944	0.646274	19507.1348	19507.390835
2022-09-29 21:00:00	0.03103	8.54842	35.69249	0.342714	0.325199	19429.3055	19430.292317
2022-09-29 22:00:00	0.02513	16.90281	45.30227	0.578081	0.574488	19455.9024	19455.673336
2022-09-29 23:00:00	1.24648	16.24804	44.58676	0.581773	0.530380	19592.1436	19591.110019

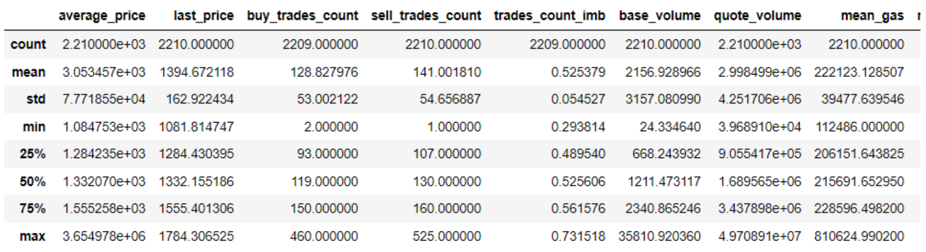
FIGURE 3.4: Data statistics
Statistics of the Public Trades data

3.1.2 *Ethereum blockchain*

To obtain information about the trading activity in a Liquidity Pool provided by the smart contract Uniswap v3, which is also considered a Decentralized Exchange that is entirely live functioning in the Ethereum network, as a smart contract.

The dataset extracted from the blockchain was composed of 4175 timestamps of one hour each, encompassing the time from June 1st, 2022, to November fifteen, 2022; the dataset is composed of the following features:

- Timestamp: index, periods of 1 hour.
- Price: the mean value of the crypto in that hour.
- Volume: the total volume traded in the timestamp.
- Base volume: volume calculated at the time based on how much one can buy with the quantity of BTC one has.
- quote volume total: volume of BTC quoted.
- mean gas value: pricing value to conduct a transaction or execute a contract on the blockchain.



	average_price	last_price	buy_trades_count	sell_trades_count	trades_count_imb	base_volume	quote_volume	mean_gas
count	2.210000e+03	2210.000000	2209.000000	2210.000000	2209.000000	2210.000000	2.210000e+03	2210.000000
mean	3.053457e+03	1394.672118	128.827976	141.001810	0.525379	2156.928966	2.998499e+06	222123.128507
std	7.771855e+04	162.922434	53.002122	54.656887	0.054527	3157.080990	4.251706e+06	39477.639546
min	1.084753e+03	1081.814747	2.000000	1.000000	0.293814	24.334640	3.968910e+04	112486.000000
25%	1.284235e+03	1284.430395	93.000000	107.000000	0.489540	668.243932	9.055417e+05	206151.643825
50%	1.332070e+03	1332.155186	119.000000	130.000000	0.525606	1211.473117	1.689565e+06	215691.652950
75%	1.555258e+03	1555.401306	150.000000	160.000000	0.561576	2340.865246	3.437898e+06	228596.498200
max	3.654978e+06	1784.306525	460.000000	525.000000	0.731518	35810.920360	4.970891e+07	810624.990200

FIGURE 3.5: Data statistics
statistics of the Public Trades data

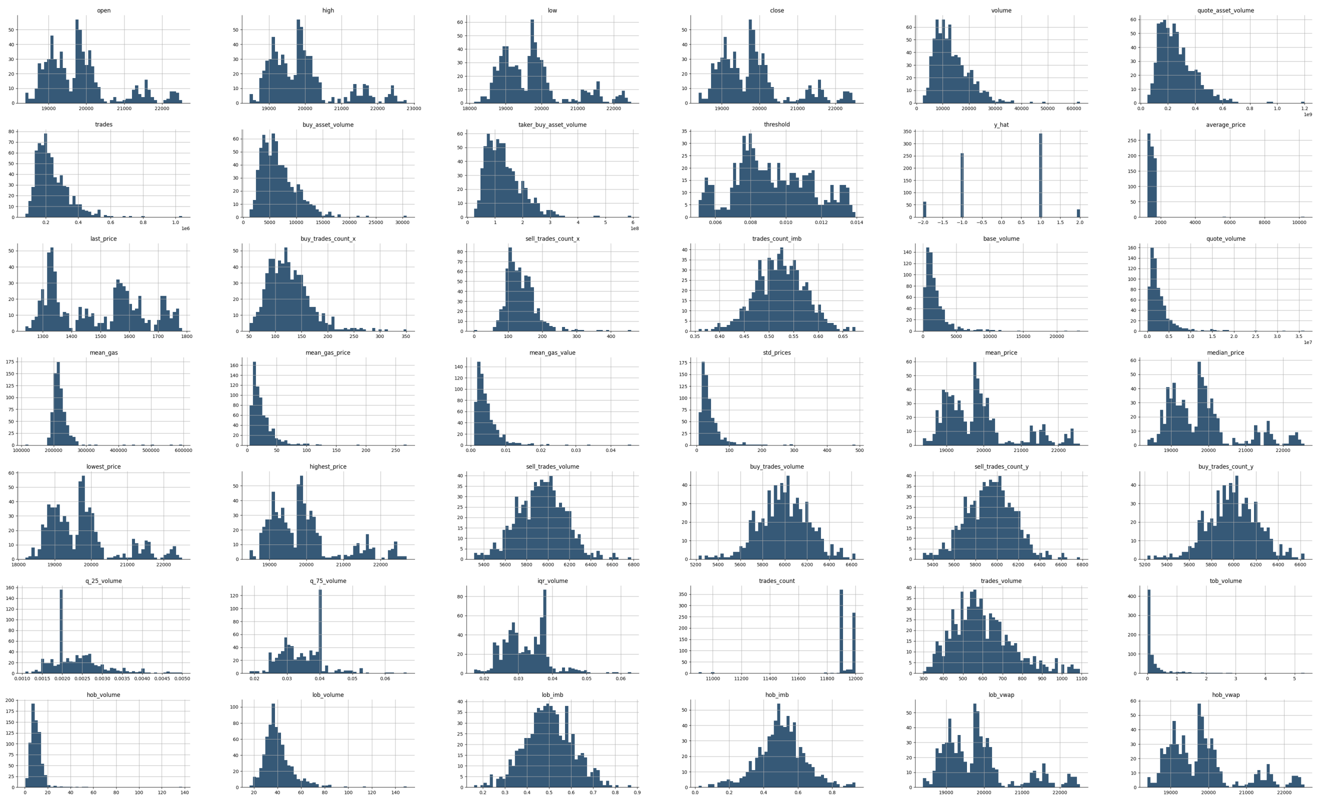


FIGURE 3.6: Data Histogram
Histogram of the features from the first dataset

The first thing that became apparent, as seen in figure 2.2, was that the OHLCV had similar distributions and that the volumes did as well; later, the data was resampled to one-hour intervals; this changed the distribution and created data that had more significant differences between features. All the parts are numerical, and there are no categorical data. First, generating more features to explain and a target feature representing the different classes was necessary.

3.2 *Data set formation*

Thea is presented in periods of one minute, encompassing the timeframe that January 1, 2021, to October 31 2022, for the experiments, the following was used:

- 4 hours per sample period. Only OHLCV
- 8 hours per sample period. Only OHLCV
- 1 hour per sample period. Only OHLCV
- 1 hour per sample, with all the datasets.

This being a time series, to split the dataset for training and testing, there was special consideration for maintaining the time series, so no random or shuffle of the data was performed, as this would have disrupted the temporality of the dataset, which will be expanded upon in chapter 4.

3.2.1 Classical

The classical way to split a time series is to take a certain percentage from the last part of the dataset and divide it into training, validation, and testing. To achieve this, the dataset was split into 70%, which is used to generate features x, and matrics, another 20% for testing, and a final 10% for validation. These are visualized in figure 2.3.3.9

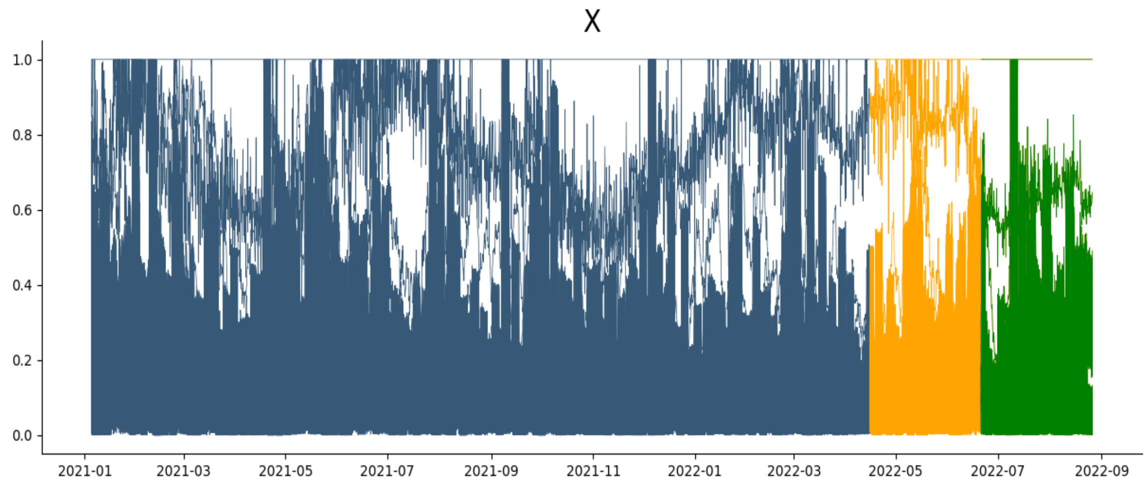


FIGURE 3.7: Split of Features
Split of the x dataset, composed of features.

3.2.2 Data Methods

3.3 Experiment 1 Data Methods

For the first experiment, the OHLCV dataset was used; the dataset was composed of 962,367 timestamps and nine different variables as expressed before; the dataset was obtained from Binance vision and encompasses from January 1, 2021, to October 30, 2022.

In the first experiment, the dataset was resampled to 4 hours. This had the objective of matching half a day of trading and could show the cryptocurrency trends in the given timeframe. This means that a low frequency was used, as this experiment starts from the simplest and continues to increase the complexity in each experiment.

Next, a new set of features were created; these features present information on the price behavior of the asset, which are created using basic statistic methods, and four features can be generated that help explain the prices and their behavior throughout the time series.

- Volatility: Difference between the highest value and the lowest.

$$V_t = H_t - L_t \tag{3.1}$$

- Uptrend: Difference between the highest value and the first value on the epoch.

$$HO_t = H_t - O_t \tag{3.2}$$

- Downtrend: The difference between the lowest value and the last value.

$$OL_t = O_t - L_t \tag{3.3}$$

- Direction: the difference observed between the close, and open value is the feature used to create the target feature $\hat{y}$.

$$CO_t = C_t - O_t \tag{3.4}$$

These features are called micro-trends and help to find more information and connections between the data, also helping explain the behavior of the price values of the BTC/USDT cross. After creating the microtrends, the following step was creating the autoregressive features.

Using the linear features, the autoregressive features were created using means, differences, standard deviation, and lag. Two hundred twenty-five new features were created using this method, and due to the nature of the autoregressive methods, the dataset was left with 3990 timestamps.

3.3.1 *Target Engineering Experiment 1*

The target, as previously discussed, is a label called signal, which takes the value of CO_t and gives it a classification based on a classification method using an empirically chosen threshold; for this particular problem, a multi-class signal was used, and this created four classes. $\hat{y}$ is based on CO_t (close-open), one of the statistic features as a signal it will help to see the direction of the trend in price per epoch; this is later shifted one epoch before, so we don't cause a leakage of the data, as it can happen when working with time series, because the model would be predicting the present with data from the same epoch. The formula for $\hat{y}$ is:

$$\hat{y} = sign[Close_t - Open_t] \tag{3.5}$$

The next step was to check the correlation between features and target, choosing to eliminate the elements with a correlation below 15% correlation with $\hat{y}$ using the spearman correlation and also to eliminate the features with a correlation between them which exceeded a threshold chosen at 80% correlation using the Pearson coefficient, to accomplish this a method was implemented where those features would be selected and eliminated from the dataset, leaving it with 107 features to train the models. The formula used for Spearman is the following one.

$$\rho = 1 - \frac{6\sum d_i^2}{n\left(n^2 - 1\right)} \tag{3.6}$$

In which: d= the distances of the ranks of the variables xi and yi n = number of samples.

Pearson was the second used because it is the most widely used method to check the correlation between features. in which 1 represents a perfect positive linear relationship and -1 is a perfect negative one. In contrast, a 0 represents no correlation at all.

$$\rho = \frac{\operatorname{cov}(X, Y)}{\sigma_x \sigma_y} \tag{3.7}$$

The estimate is done with the following formula:

$$r = \frac{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)\left(y_i - \bar{y}\right)}{\sqrt{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2\left(y_i - \bar{y}\right)^2}} \tag{3.8}$$

Following this, the dataset was split into X and Y datasets. The X dataset was transformed using a scaler and later a normalizer to achieve a scale from 0 to 1 for the data and a mean of 0 using both methods. The equations used are the following:

$$x_{scaled} = \frac{x}{max\left(|x|\right)} \tag{3.9}$$

$$x_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3.10}$$

After this, the datasets were split into three smaller datasets for Train, Test, and validation, following the metric of 70%,20%, and 10%; special care was taken to not disturb the periodicity of the dataset, as with time series, one cannot use the normal methods to split the data as seen on figure 4.2.

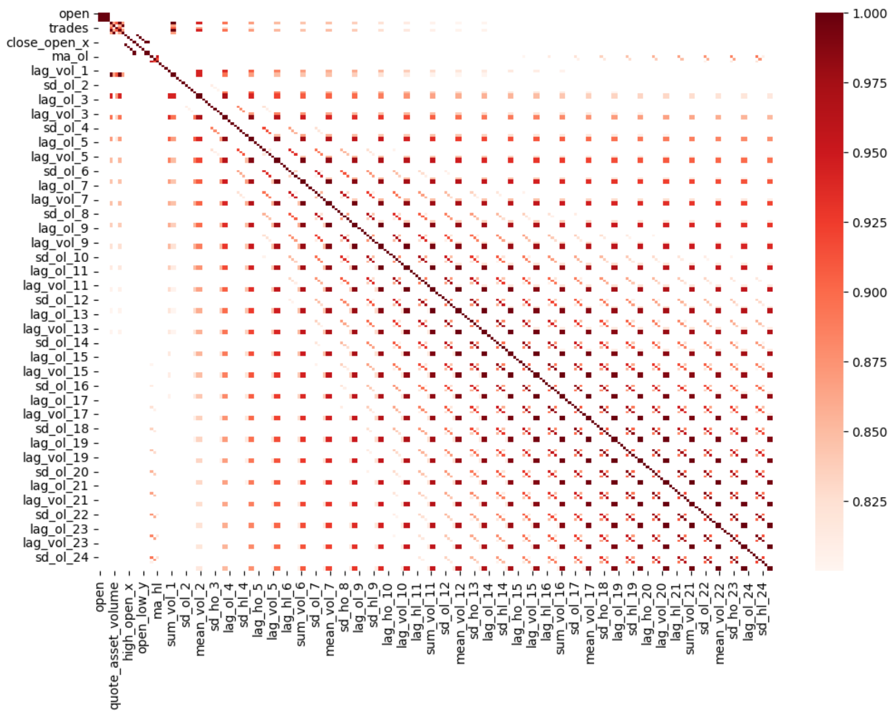


FIGURE 3.8: Heatmap of highly correlated features
Heatmap generated from the features produced having more than or equal to 80% correlation between features.

3.3.2 *Experiment 2 Data methods*

In contrast with the first experiment, the resample of this iteration was changed to eight hours; this was elected because of the time a trade session lasts, after this, the same methods were used, and the basic statistics features and the autoregressive were added to the dataset, adding to 1983 timestamps and 234 different explanatory features.

For the symbolic features, a symbolic transformer was implemented with the following parameters: functions such as subtraction, adding, inverse, multiplication, division, absolute, logarithmic, and square root; the parameters were a population of 12,000, tournaments of 3000, hall of fame of 30, for five generations, to create 20 new features.

With those parameters and the entire dataset, four different features were created and later added to the dataset. The data transformation was implemented in the same way as the first experiment, with the difference that it was done before the feature selection, with the hypothesis that it could change the metrics results.

After applying the feature selection, only 88 features remained on the dataset. The data was split using the same method as in experiment one splitting it into three datasets for train, testing, and validation.

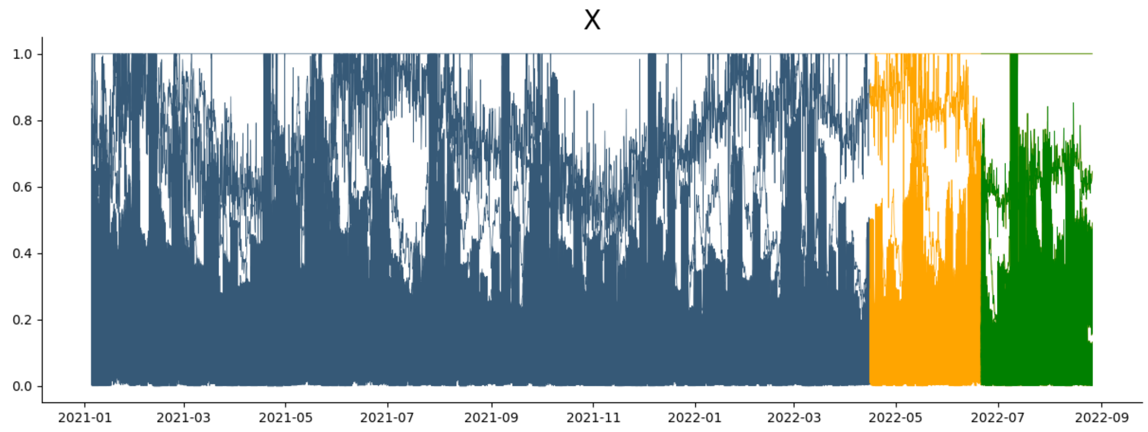


Figure 3.9: Data Split
Plot of Data split performed, using 70, 20, 10 ratios for Training, test, and validation.

3.3.3 *Target Engineering*

For the second experiment, $\hat{y}$ was created in the same way, with the only distinction being that the benchmark was changed to reflect the difference in resampling of the timestamps, rising to 300 the benchmark value to get a more balanced classification.

3.3.4 *Dataset Generation Experiment 3*

The OHLCV dataset was generated the same way as the last experiment; then, it was resampled as 1-hour timestamps to match the Uniswap dataset. The dataset was merged, and the timestamps without Uniswap or OHLCV data were removed, leaving 3824 timestamps and 14 features.

After this, the same methods were applied to create explanatory features, creating 226 autoregressive features and 17 symbolic ones, giving a total of 264 descriptive features.

The dataset was then split into X and Y and transformed through the scaling and normalization transformations; then, the feature selection was applied, leaving 124 explanatory features.

3.3.5 *Target Engineering Experiment 3*

The target variable $\hat{y}$ was created in the same way as the previous experiments, with a different threshold whose value was elected to reflect the temporality of the timestamps.

3.3.6 Dataset Generation Experiment 4

With the assumption that more features could help to create a dataset with more prediction possibilities, the dataset used in experiment 4 all the datasets was used; this means that OHLC, order books, public trades, and Uniswap data were used to generate a dataset for September 2022, this data had 697 timestamps and was composed of forty-two different features, from each of the other previous datasets. After this, the autoregressive features were created using the same methods as the other experiments, resulting in a dataset composed of 191 different features. Which would be used to create new symbolic features. This method was the same as before and used the following parameters:

- Population: 12000
- Tournament: 3000
- hall of fame: 300
- Generations: 7
- Features generated: 100
- with a depth of 2 to 16

With this, the symbolic transformer generated one hundred new features for a total of 290 with aggregated, autoregressive, symbolic, and original features, which would begin the process of choosing the feature variables to train the model.

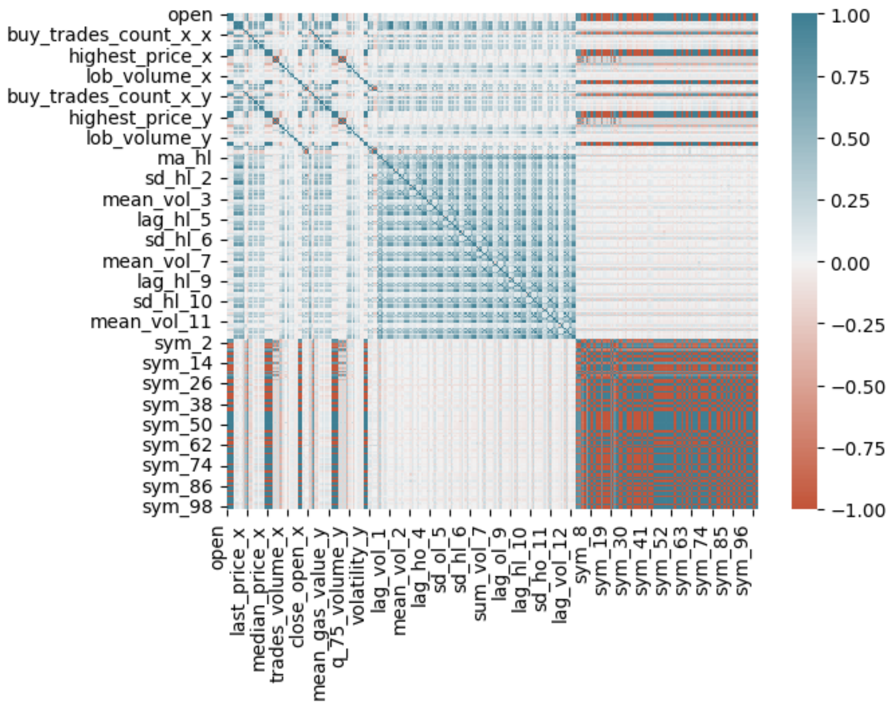


FIGURE 3.10: Data Correlation
Plot of the Matrix generated with the correlations between variables.

3.3.7 Target variable generation Experiment 4

For experiment 4, a different way to create the labels was used; this method is referred to as three barriers by Lopez De Prado; what it does is create dynamic thresholds for the labeling based on the volatility of the market, creating three barriers, upper, middle, and low. The method was adapted and changed to make four classes instead of two, where in the case presented by Lopez De Prado[4], it creates signals only when touching the barriers, and in the case presented in this work, it was reworked to assign signals to each case.

[4] Lopez de Prado, M. M. (2018). *Advances in Financial Machine Learning*. Wiley

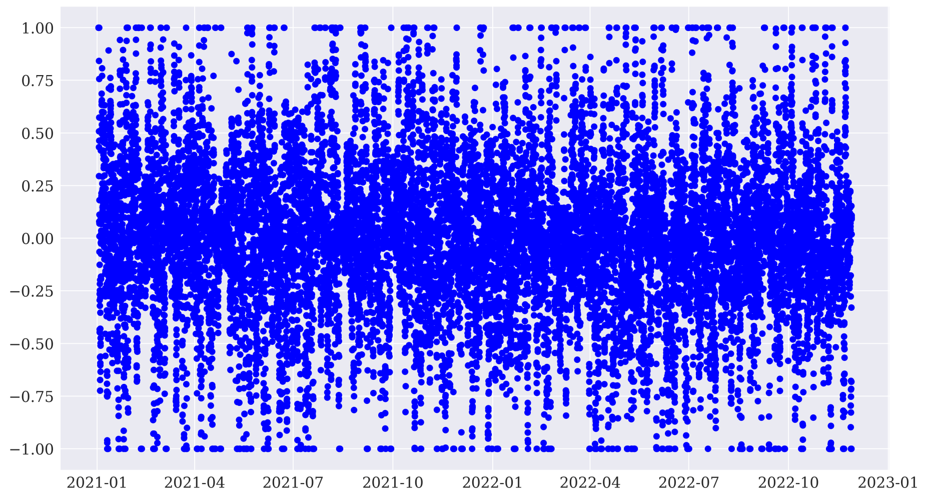


FIGURE 3.11: Plot of Labeling Method
Plot of the classification of each epoch based on their label

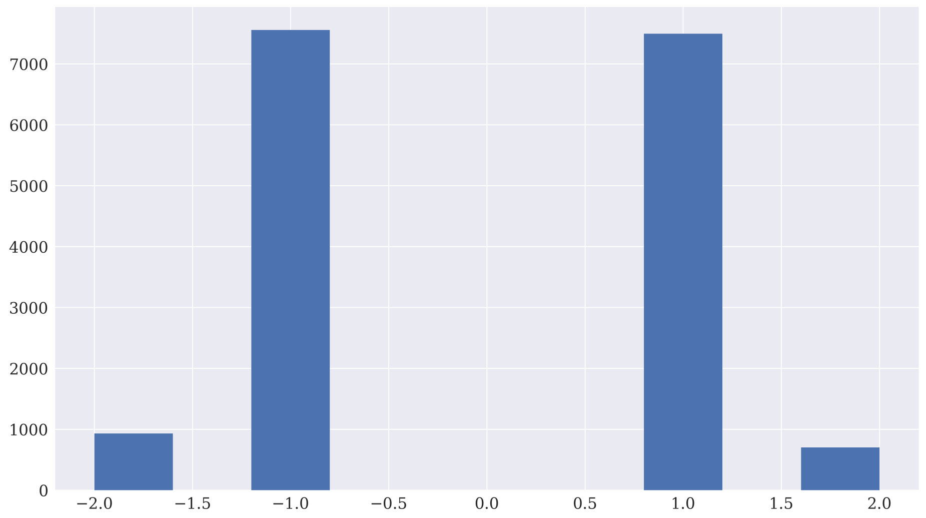


FIGURE 3.12: Histogram of $\hat{y}$
Histogram of Target Variable

3.3.8 Feature Selection Experiment 4

After generating all the variables to be used, the next step was to select the ones that would contribute the most to the final model; the method used was the same as in past experiments, and the resulting Plot for the correlation matrix with the highly correlated features exceeding 80% correlation was the following one:

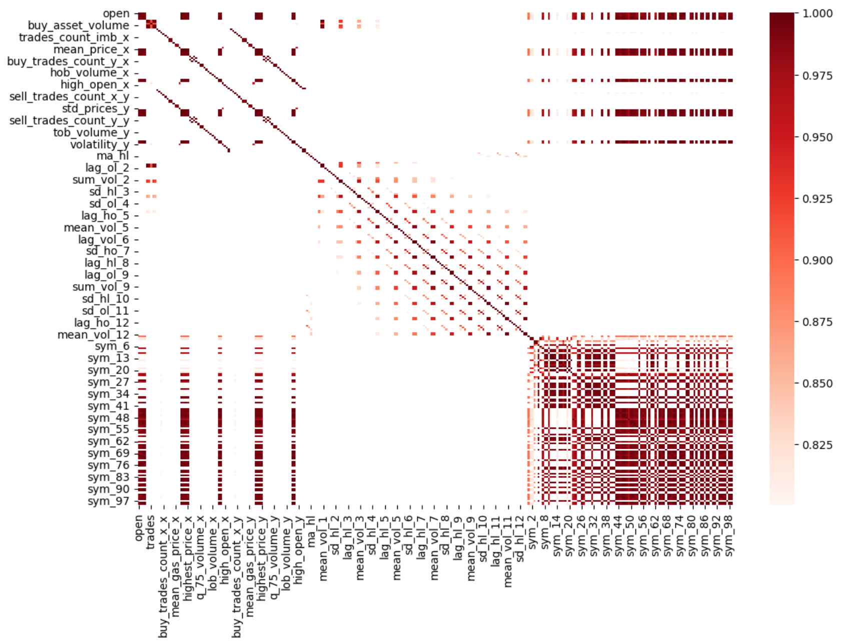


FIGURE 3.13: Highlight of the Plot with highly correlated features
Plot of the correlation matrix

This Plot shows the correlation between the different types of data used, with the different sections composing the highly correlated Plot being highlighted. This is explained because of the other methods that the data was transformed and created, which is to be expected.

Dropping the features left the dataset with eighty-five features, which would be used to correlate with the target variable, then dropping the features which do not meet the minimum correlation to the target, leaving only 70 features in the final dataset.

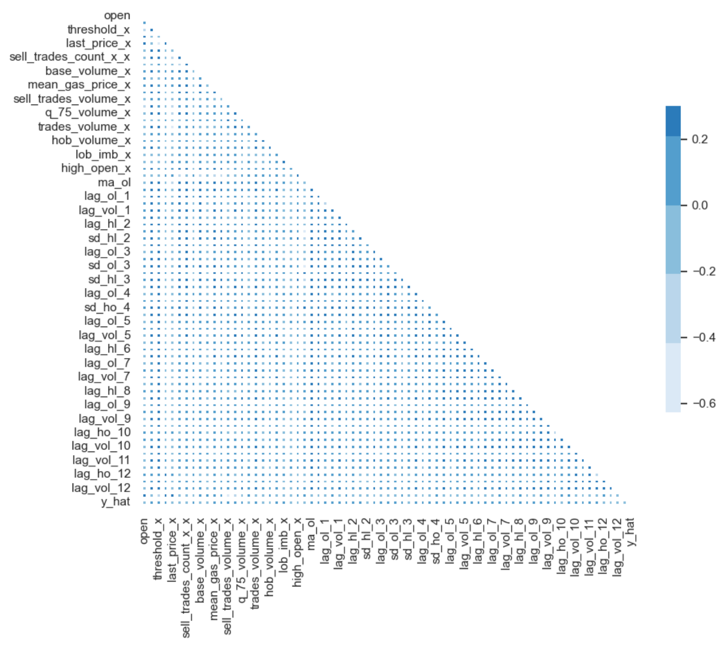


FIGURE 3.14: Final dataset correlation to $\hat{y}$
Plot of the correlation matrix

3.3.9 Data Split Experiment 4

The Data split was done the same way as in past experiments, splitting the data into 70 percent for training, 20 for testing, and ten percent for validation.

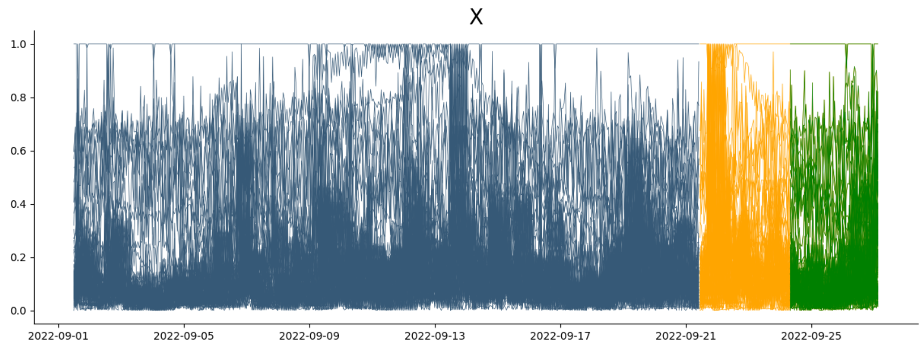


FIGURE 3.15: Data split for X dataset
Data Split for X features

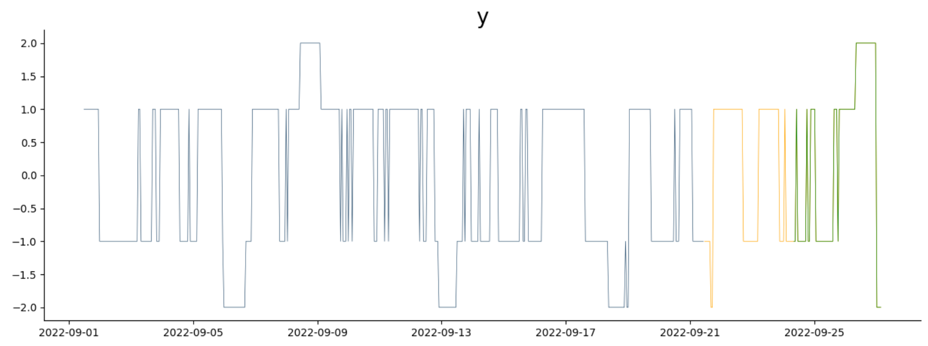


FIGURE 3.16: Data split for $\hat{y}$ dataset
Data Split for $\hat{y}$

4 *Predictive Modeling Process*

Contents

The contents of this chapter are the models used for the prediction of $\hat{y}$ and the different experiments implemented during the optimization of the models.

4.1 Data Scaling

Before applying the datasets to the models, a series of statistical techniques were applied: Transformation, standardization, and normalization help in the process of machine learning due to the original data being on different scales, these scaling methods have a vital role in the preprocessing of the data, and has an impact on the performance of the machine learning algorithms, in this work the scaling methods used were scaling and normalization.

4.2 Predictive Models

Two models were proposed as benchmarks, starting from the simplest one, a Martingale, and the second one is based on the premise of information updates given past results, which Naive Bayes does provide. Then three models are proposed for the experiments, logistic regression, random forest classifier, and a neural network in the form of the multi-layer perceptron.

4.2.1 Martingale

Martingale was the first benchmark method, chosen because it is the simplest method used in finance as well as being a staple for low computing cost forecasting methods, and it parts from the assumption that the value of the signal $\hat{y}_{t+1}$ is going to be the same as the one in $\hat{y}_t$, this can be expressed on the following way:

$$[X_t] = E\left[X_{t-1}\right] \tag{4.1}$$

4.2.2 Naive Bayes

The second benchmark method chosen was the Naive Bayes which based on Bayes Theorem assumes that the features used for the classification are independent between them, this being the naive part of the model, this method did not have optimization and was only used as a benchmark to contrast with the other more complex models.

The following equations are used on the Naive Bayes model:

- Posterior Probability:

$$P\left(\omega_j|x\right) = \frac{p\left(x|\omega_j\right) \cdot P\left(\omega_j\right)}{p\left(x\right)} \tag{4.2}$$

$$\text{posterior probability} = \frac{\text{likelihood} \cdot \text{prior probability}}{\text{evidence}} \tag{4.3}$$

- Decision Rule:

$$\text{Decide } \omega_1 \text{ if } P\left(\omega_1|x\right) > P\left(\omega_2|x\right) \text{ else decide } \omega_2. \tag{4.4}$$

$$\frac{p\left(x|\omega_1\right) \cdot P\left(\omega_1\right)}{p\left(x\right)} > \frac{p\left(x|\omega_2\right) \cdot P\left(\omega_2\right)}{p\left(x\right)} \tag{4.5}$$

- Objective function:

$$\text{Decide } \omega_1 \text{ if } P(\omega_1|x) > P(\omega_2|x) \text{ else decide } \omega_2. \frac{p(x|\omega_1)\cdot P(\omega_1)}{p(x)} > \frac{p(x|\omega_2)\cdot P(\omega_2)}{p(x)} \tag{4.6}$$

$$g_i(\boldsymbol{x}) = \boldsymbol{x}^t\left(-\frac{1}{2}\Sigma_i^{-1}\right)\boldsymbol{x} + \left(\Sigma_i^{-1}\boldsymbol{\mu}_i\right)^t\boldsymbol{x} + \left(-\frac{1}{2}\boldsymbol{\mu}_i^t\Sigma_i^{-1}\boldsymbol{\mu}_i - \frac{1}{2}ln(|\Sigma_i|)\right) \tag{4.7}$$

- And uses the following equation to represent that the features are gaussian (the naïve part):

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}}\exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right) \tag{4.8}$$

4.2.3 Logistic Regression

This model for classification is also known in the literature as logit regression and Maxent classifier, this model uses the probabilities of possible outcomes using the logistic function.

$$f(x) = \frac{L}{1+e^{-k(x-x_0)}} \tag{4.9}$$

The Logistic Regression model was used twice in each experiment the first time without any optimization, and the second using hyperparameters optimization to contrast the results of the different configurations of the models.

The configuration for the second model used the Multinomial case, as $\hat{y}$ was produced with 4 classes, on the multinomial case the objective of the optimization becomes the following equation:

$$\min_W -C\sum_{i=1}^{n}\sum_{k=0}^{K-1}[y_i = k]\log(\hat{p}_k(X_i)) + r(W). \tag{4.10}$$

where $P(y_i = k|X_i)$ represents the Iverson bracket in which p= 0 if False, and in every other case as 1, for the regularization the elastic net was used and is expressed by the following:

$$\frac{1-\rho}{2}\|W\|_F^2 + \rho\|W\|_{1,1} \tag{4.11}$$

For the optimization of the model on the second iteration of the model implementation in each experiment, a grid was used to check different hyperparameters and choose the best-performing one.

4.2.4 Random Forest

This ensemble method of randomized trees creates multiple trees for classification and combines the predictions using an average of the different models, this is used because the ensemble reduces the variance between the different trees creating a more robust model. The method ensembles multiple trees independently from each other and then average the results. Each tree is produced using the following equations:

- The node m is represented by Q with n samples, then each candidate is split then partitioning the data into the two QS below:

$$Q_m^{left}(\theta) = \{(x,y)|x_j \leq t_m\}$$
$$Q_m^{right}(\theta) = Q_m \setminus Q_m^{left}(\theta) \tag{4.12}$$

- Then the quality of the splits of node m is computed using the loss function H.

$$G(Q_m,\theta) = \frac{n_m^{left}}{n_m} H(Q_m^{left}(\theta)) + \frac{n_m^{right}}{n_m} H(Q_m^{right}(\theta)) \tag{4.13}$$

- and then the parameters used to minimize the impurity

$$\theta^* = \operatorname{argmin}_\theta G(Q_m,\theta) \tag{4.14}$$

- as the target is a classification the criteria for classification are:

$$p_{mk} = \frac{1}{n_m} \sum_{y \in Q_m} I(y = k) \tag{4.15}$$

4.2.5 *Multi Layer Perceptron*

The final method used was a multi-layer perceptron or MLP which is a feedforward neural network, which is composed of more than one perceptron, it is composed of an initial layer, a final layer, and one or more hidden layers, the initial layer receives the initial signal, the final layer makes the decision of the prediction and the hidden layers perform the computational task of the MLP, which function in two ways, as a forward pass, where the signal flow transits from the input layer, through the hidden ones into the output were the decision is made, then there is the backward pass, or backpropagation, where the partial derivatives of the error function are backpropagated through the neural network, then using the error, the parameters are adjusted to get the MLP to minimize the error until the model cannot reduce the error anymore.

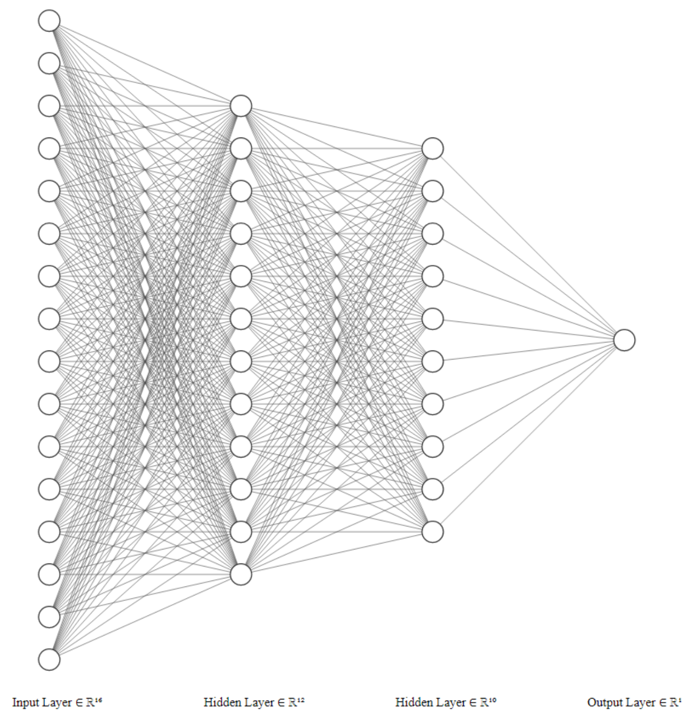


FIGURE 4.1: Diagram of an MLP
MLP consisting of 16 neurons in the input layer, 12 in hidden layer 1, 10 in hidden layer 2, and 4 in the output layer..

4.3 *Modeling Process*

The process was defined fundamentally with three elements in mid-periodicity, giving more importance to target and cost function engineering, then to be specially focused on financial time series data, and finally providing support for recursive experiment definition. Thus, the following are the general steps that are part of the predictive modeling process.

- Data set generation
- Target engineering
- Cost function engineering
- Feature engineering
- Model definition
- Model training and validation
- Performance analysis

4.4 Experiments

The criteria for the experiment definition are included in this chapter since it involved sections of the entire predictive modeling process. Also, the dynamics of the experiments followed a sequential definition and execution, e.g. the first experiment was conducted and its results were used to define the second experiment, and from the results of this last one, compared with the previous one, a third experiment was defined and executed based on the results of experiment 2, and a final experiment was formulated based on previous results.

The elements considered for experiment definition were:

- Data set generation
- Target Engineering
- Feature Engineering and Selection
- Model Optimization

4.4.1 Model definition

4.4.2 Experiment 1

The first experiment was formulated using only the OHLCV data, using the data from 2021 through November 2022, with 962367 samples, and 9 initial features, with a periodicity, changed to 4 hours per sample. The formulation for the preprocessing of the data was to use scaling and normalization after the feature selection, the target feature was generated with a threshold chosen and the only method to produce more features was autoregressive only. The final dataset had 107 explanatory features.

- Data Sources: OHLC
- period: January 1st 2021 to October 31st 2022
- Periodicity: 4 Hour intervals per sample.
- optimization of the models with a grid.
- Feature Creation methods: Autorregresive.
- Scaling and Normalization before feature selection.

In the first experiment, the following models were used as benchmarks:

- Martingale
- Naive Bayes
- Logistic Regression(no optimization)

Following the benchmarks a logistic regression was implemented using the elastic-net penalty and optimized by a grid composed of the following parameters:

- L1 Ratio [.10,.20,.30,.40,.50,.60,.70,.80,.90]

- Class weight [None, 'balanced']
- C [0.001, 0.01, 0.05, 0.1, 0.5, 1.0, 10.0]
- Solver ['newton-cg', 'sag', 'saga','lbfgs']

The Logistic Regression used accuracy as its metric.

The next model implemented was the Random Forest, which was optimized by a grid; the model used the following parameters:

- Max Features ['sqrt,' 'log2', None]
- Criterion['gini,' 'entropy,' 'log loss']
- Number of Estimators [100, 500,1000]
- Max Depth was set as none
- Min samples split set as 2

Finally, the MLP for classification was implemented using a grid for optimization. The model was created using TensorFlow and Keras modules in python. The parameters used for this configuration were:

- number of inputs
- quantity of neurons per layer
- quantity of hidden layers
- learning rate
- batch size
- epochs set as 50

The model used the loss function of categorical cross-entropy which is used for multi-class classification the optimizer chosen was Adam, and the metric was accuracy. In this configuration, there was one hidden layer and the model was set up to choose the best-performing configuration through optimization.

4.4.3 *Metrics for the models*

For the evaluation of the models, the following metrics were used:

- Accuracy, measures the model performance, the formula is the following one:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.16}$$

- Precision, how accurate the positive predictions were.

$$Precision = \frac{TP}{TP + FP} \tag{4.17}$$

$$Precision = \frac{TP}{TP + FP}$$

- Recall is the ratio of true positives to total positives in the data.

$$Recall = \frac{TP}{TP + FN} \tag{4.18}$$

- Specificity, percentage of true negatives.

$$Specificity = \frac{TN}{TN + FP} \tag{4.19}$$

- f1 score hybrid metric for unbalanced classes.

$$F1 = \frac{2TP}{2TP + FP + FN} \tag{4.20}$$

- Balanced Accuracy, the accuracy that considers the true weight of unbalanced classes.

4.5 *Experiment 2*

For the next experiment, the formulation of the implementation of the code was changed to see the effects of resampling at a different interval chosen to be 8 hours, using symbolic features, and changing the transformation of the data to before the feature selection, with the hypothesis that this can improve the metrics of the models due to more data being on each sample period.

- Data Sources: OHLC
- period: January 1st 2021 to October 31st 2022
- Periodicity: 8 Hour intervals per sample.
- Further optimization of the models.
- Feature Creation methods: Autorregresive, Symbolic.
- Scaling and Normalization after feature generation.

The final dataset used in the second experiment was composed of 235 features composed of features from the original dataset, autoregressive created ones, and symbolic features.

4.5.1 *Model definition*

The models were implemented the same way as experiment one, having three benchmarks, the martingale, naive Bayes, and logistic without optimization, and three optimized models, Logistic regression with elastic net, Random forest classifier, and MLP for classification. These last three were optimized through a grid with the chosen parameters.

- Martingale
- Naive Bayes
- Logistic Regression(no optimization)

following the benchmarks, logistic regression was implemented using the elastic-net penalty and optimized by a grid composed of the following parameters:

- L1 Ratio [.50,.60,.70,.80,.90]
- Class weight [None, 'balanced']
- C [0.001, 0.01, 0.05, 0.1, 0.5, 1.0, 10.0]
- Solver [Saga]

The Logistic Regression used accuracy as its metric.

The next model implemented was the Random Forest, which was optimized by a grid; the model used the following parameters:

- Max Features ['sqrt,' 'log2', None]
- Criterion['gini,' 'entropy,' 'log loss']
- Number of Estimators [100, 500,1000]
- Max Depth was set as none
- Min samples split set as 2

Finally, the MLP for classification was implemented using a grid for optimization. The model was created using TensorFlow and Keras modules in python. The parameters used for this configuration were:

- number of inputs
- quantity of neurons per layer: variable Number of inputs plus [2,4,8,16,32,64]
- quantity of hidden layers two sigmoid activation
- learning rate [0.01,0.1,0.5,1]
- batch size [1,8,16]
- epochs set as 50

The model used the loss function of categorical cross-entropy for multi-class classification. The optimizer chosen was Adam, and the metric was accuracy. There was one hidden layer in this configuration, and the model was set up to select the best-performing configuration through optimization.

4.6 *Experiment 3*

Based on the results of experiment 2, experiment 3 implemented more changes to improve metrics and create more robust forecasting models. The following changes were implemented in this experiment:

- Data Sources: OHLC, Uniswap
- period: June 1, 2022- October 31, 2022
- Periodicity: 1 Hour intervals per sample.
- Further optimization of the models.
- Feature Creation methods: Autorregresive, Symbolic.
- Scaling and Normalization after feature generation.

4.6.1 *Model Definition*

The models were implemented in the same formulation as experiment two, having three benchmarks, the martingale, naive Bayes, and logistic without optimization, and three optimized models, Logistic regression with elastic net, Random forest classifier, and MLP for classification. These last three were optimized through a grid with the chosen parameters.

- Martingale
- Naive Bayes
- Logistic Regression(no optimization)

following the benchmarks, logistic regression was implemented using the elastic-net penalty and optimized by a grid composed of the following parameters:

- L1 Ratio [.50,.60,.70,.80,.90]
- Class weight [None, 'balanced']
- C [0.001, 0.01, 0.05, 0.1, 0.5, 1.0, 10.0]
- Solver [Saga]

The Logistic Regression used accuracy as its metric.

The next model implemented was the Random Forest, which was optimized by a grid; the model used the following parameters:

- Max Features ['sqrt,' 'log2', None]
- Criterion['gini,' 'entropy,' 'log loss']
- Number of Estimators [100, 500,1000]
- Max Depth was set as none

- Min samples split set as 2

Finally, the MLP for classification was implemented using a grid for optimization. The model was created using TensorFlow and Keras modules in python. The parameters used for this configuration were:

- number of inputs
- quantity of neurons per layer: variable Number of inputs plus [2,4,8,16,32,64]
- quantity of hidden layers two sigmoid activation
- learning rate [0.01,0.1,0.5,1]
- batch size [1,8,16]
- epochs set as 50

The model used the loss function of categorical cross-entropy for multi-class classification. The optimizer chosen was Adam, and the metric was accuracy. There was one hidden layer in this configuration, and the model was set up to select the best-performing configuration through optimization.

4.7 *Experiment 4*

Based on the results of experiment 3, experiment 4 had significant changes in the way that the target variable was produced. Also, more sources for data were used, which impacted the period that could be used for the final dataset. in the case of the models, those were optimized through a random search to make modeling the data faster and more accurate.

- Data Sources: OHLC, Uniswap, Publictrades
- period: September 2022
- Periodicity: 1 Hour intervals per sample.
- Further optimization of the models through random search.
- Feature Creation methods: Autorregresive, Symbolic.
- Scaling and Normalization after feature generation.

4.7.1 *Model Definition*

The models were implemented in the same formulation as experiment two, having three benchmarks, the martingale, naive Bayes, and logistic without optimization, and three optimized models, Logistic regression Elastic net, Random forest, and MLP for classification. These last three were optimized through a grid with the chosen parameters.

- Martingale
- Naive Bayes

- Logistic Regression(no optimization)

following the benchmarks, logistic regression was implemented using the elastic-net penalty and optimized by a grid composed of the following parameters:

- L1 Ratio [.50,.60,.70,.80,.90]
- Class weight [None, 'balanced']
- C [0.001, 0.01, 0.05, 0.1, 0.5, 1.0, 10.0]
- Solver [Saga]

The Logistic Regression used accuracy as its metric.

The next model implemented was the Random Forest, which was optimized by a grid; the model used the following parameters:

- Max Features ['sqrt,' 'log2', None]
- Criterion['gini,' 'entropy,' 'log loss']
- Number of Estimators [100, 500,1000]
- Max Depth was set as none
- Min samples split set as 2

Finally, the MLP for classification was implemented using a grid for optimization. The model was created using TensorFlow and Keras modules in python. The parameters used for this configuration were:

- number of inputs
- quantity of neurons per layer: variable Number of inputs plus [2,4,8,16,32,64]
- quantity of hidden layers: four soft plus, tanh activations.
- learning rate [0.01,0.1,0.5,1]
- batch size [1,8,16]
- epochs set as 50
- Input Layer: Relu
- Output Layer: Softmax 4 neurons.
- Optimizer: Adam

5 Results of the Models

Contents

In this chapter the obtained results are discussed, and also a comparison between models is analyzed.

5.1 Model's Performance

5.2 Experiment 1

Experiment one presented the following metrics:

Table 5.1: Experiment 1 Model Results

Model	Accuracy	Recall	Precision	F1
Martingale	31.328321	31.328321	31.331823	31.330068
Logistic Regression	31.203008	32.581454	21.364127	25.806288
Naive Bayes	27.067669	27.067669	28.127838	24.895935
Random Forest	34.335840	34.335840	23.155341	26.611362
MLP	33.21	33.22156	28.5486	30.584

5.2.1 Martingale Experiment 1

Martingale in the first experiment performed poorly with accuracy recall and precision and f1 at 31.33%, the benchmark model did the following classification:

- -2: 1501
- -1: 486
- 1: 474
- 2: 1529

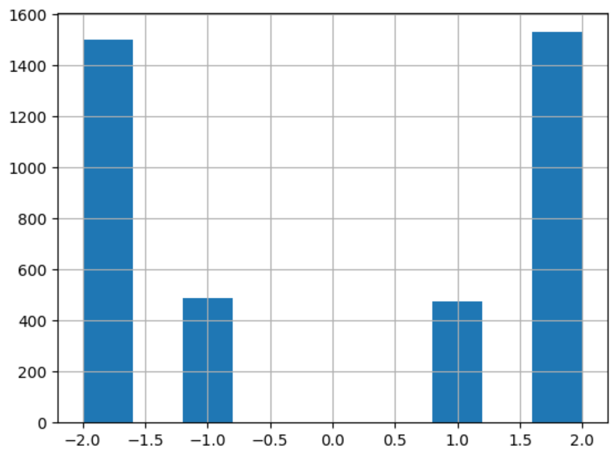


Figure 5.1: Predictions for Martingale
Martingale histogram

The confusion Matrix produced gave the following results:

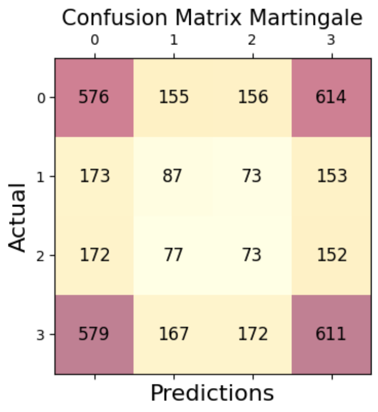


FIGURE 5.2: Plot of the Confusion Matrix
Confusion Matrix for Martingale Model

5.2.2 *Naive Bayes experiment 1*

Naive Bayes performed the worst amongst the models in experiment 1, which was expected as the features are correlated. As in the case of the martingale, the Confusion Matrix Shows, that the predictions are random, and distributed among all the classes.

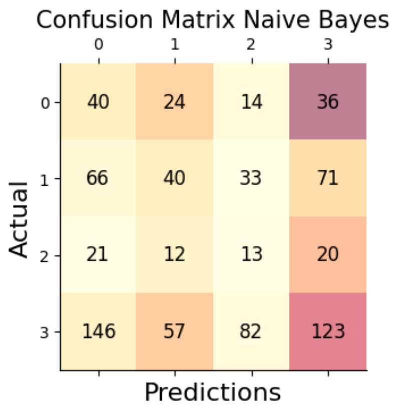


FIGURE 5.3: Predictions for Naive Bayes
Naive Bayes Confusion Matrix

5.2.3 *Logistic Regression Experiment 1*

First, the Logistic Regression was implemented without optimization.

- The Logistic Regression without optimization had an accuracy of 31.4536% for the classification of test data, while 31.5789% for validation data.
- The confusion matrix shows a bias towards the -2 and 2 classes.
- A balanced accuracy of 24.6656% confirms the bias towards the extreme classes.

With the optimization through the grid the logistic regression presented the following:

- Accuracy of 32.5814 for test and 31.5789 for validation.

- It showed the same bias as the non-optimized model.

- The confusion matrix, showed bias towards the classes -2 and 2

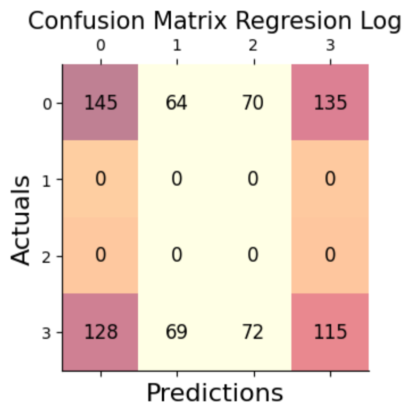


FIGURE 5.4: Predictions for Regression
Logistic Regression Confusion Matrix

5.2.4 *Random Forest Experiment 1*

The random forest was optimized through the grid and had a score of 0.4268 for the training data, the best parameters were, the Gini criterion, and max features with square root, the test dataset presented a score of 0.3433 while the validation had a 0.3358 score, this case also had the issue of bias towards the extremes of the classes, with zero true positives in the -1 and 1 classes. [99 0 0 174] [47 0 0 86] [41 0 0 101] [75 0 0 175]

5.2.5 *Multi Layer Perceptron*

The MLP was implemented using a grid optimization and had poor performance as expected in the first experiment, the best training accuracy was 0.4830 with an accuracy of 0.3321 for the testing data and validation of .3149 which was worst than the random forest metrics.

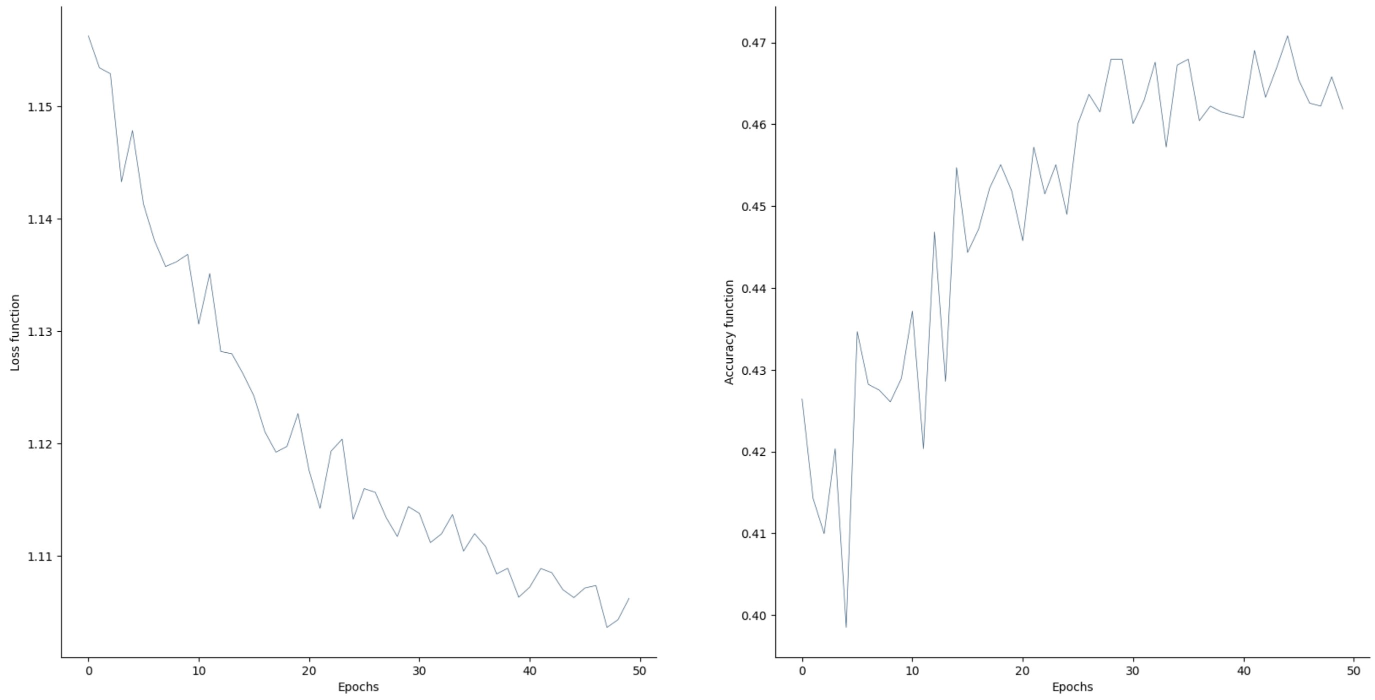


FIGURE 5.5: Training Performance for MLP
Plots of the Training data Loss and accuracy for MLP training

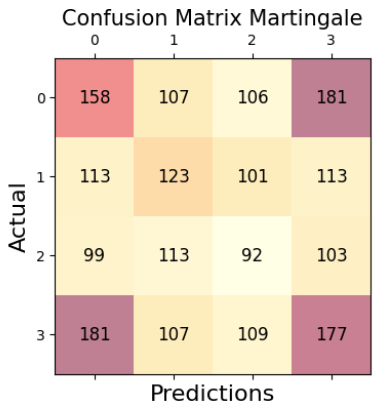
5.3 *Experiment 2*

TABLE 5.2: Experiment 2 Model Results

Model	Accuracy	Recall	Precision	F1
Martingale	29.248613	29.248613	29.247544	29.248064
Logistic Regression	20.454545	20.454545	9.888858	9.920318
Naive Bayes	31.565657	31.565657	30.331704	27.074559
Random Forest	26.010101	26.010101	32.626474	23.502318
MLP	27.27	25.581	28.5486	30.584

5.3.1 *Martingale Experiment 2*

In experiment 2, the martingale performed similarly to experiment 1.



	0	1	2	3
0	158	107	106	181
1	113	123	101	113
2	99	113	92	103
3	181	107	109	177

FIGURE 5.6: Confusion Matrix for Martingale
Plots of the Training data Loss and accuracy for MLP training

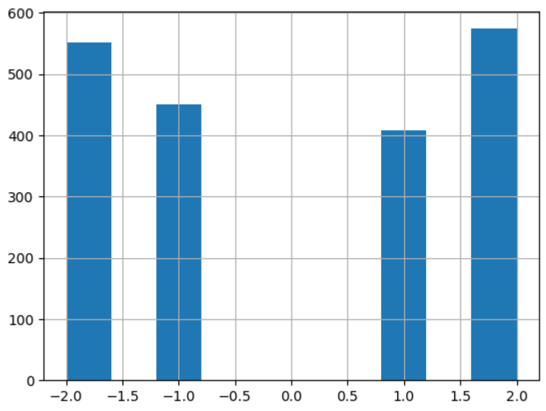


FIGURE 5.7: Histogram for Martingale classification
Histogram Plot of Martingale prediction

5.3.2 *Naive Bayes Experiment 2*

In contrast, to experiment 1 the Naive Bayes performed the second best in the second experiment, this demonstrates that it's possible that the data was not handled properly or that the periodicity of the data was too high. The validation data showed better accuracy, and recall with 0.3434 but worst precision, with only .2659

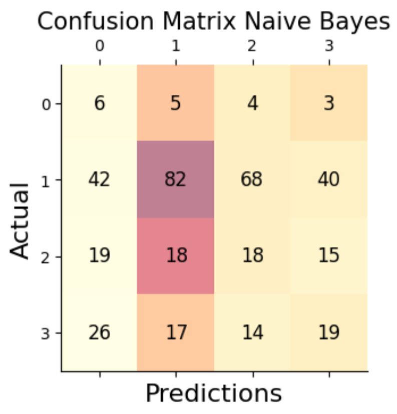


FIGURE 5.8: Predictions for Naive Bayes
Naive Bayes Confusion Matrix

5.3.3 *Logistic Regression Experiment 2*

First, the Logistic Regression was implemented without optimization.

- The Logistic Regression without optimization had an accuracy of 27.0202% for the classification of test data, while 27.7474% for validation data.
- The confusion matrix shows no bias towards the -2 and 2 classes.
- A balanced accuracy of 28.2.6656% confirms that the threshold for the signal has to be higher.

With the optimization through the grid the logistic regression presented the following:

- Accuracy of 20.4554 for test and 20.4545 for validation.
- It showed that the model was overfitted as the training accuracy was .3722
- The confusion matrix, showed bias towards the classes -2 and 2

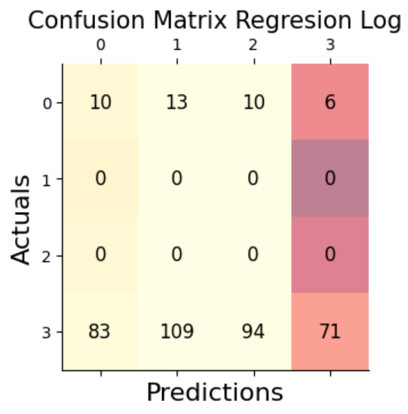


FIGURE 5.9: Predictions for Regression
Logistic Regression Confusion Matrix

5.3.4 *Random Forest Experiment 2*

- The Training accuracy was of 0.3556 which was higher than the test and validation accuracy.
- The random forest with the best estimator was one with a number of estimators of 500 and max features of log 2.
- The test accuracy was of .2626 and the validation was off.26010
- The confusion matrix had numbers in every category and seemed to be random.

5.3.5 *Multi Layer Perceptron Experiment 2*

The MLP was implemented using a grid optimization and had poor performance as expected in the first experiment, the best training accuracy was 0.4111 with an accuracy of 0.2727 for the testing data and validation of .2653 which was worst than the naive Bayes model.

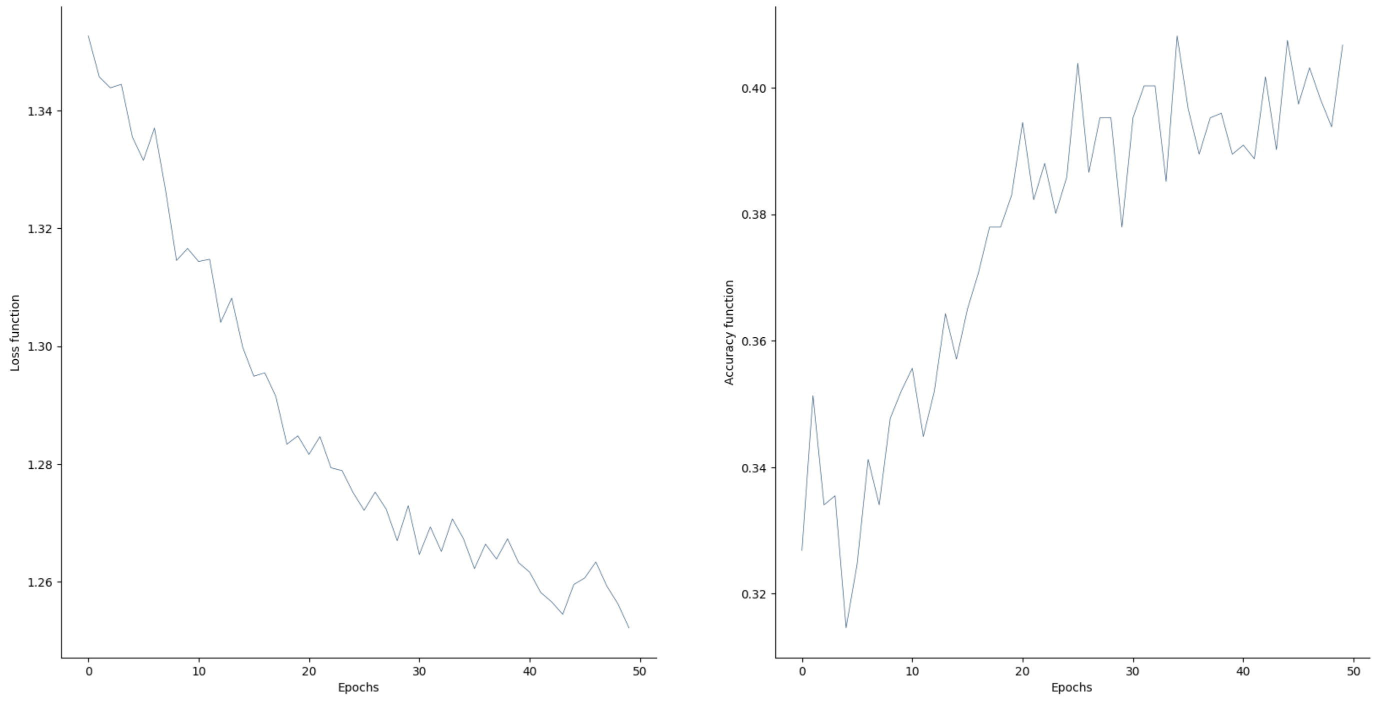


FIGURE 5.10: Training Performance for MLP
Plots of the Training data Loss and accuracy for MLP training

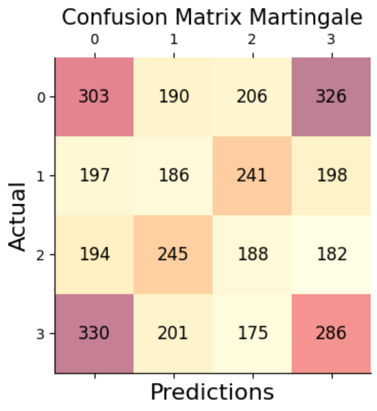
5.4 Experiment 3

TABLE 5.3: Experiment 3 Model Results

Model Balanced	Accuracy	Recall	Precision	F1
Martingale	27.850877	27.850877	27.850790	27.850829
Logistic Regression	27.945205	27.945205	50.659737	23.562354
Naive Bayes	31.232877	31.232877	33.803137	25.416627
Random Forest	32.054795	32.054795	32.509789	31.269207
MLP	33.95	32.61	30.63	30.584

5.4.1 Martingale Experiment 3

In experiment 3, the martingale performed similarly to experiment 2.



$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 303 & 190 & 206 & 326 \\ 1 & 197 & 186 & 241 & 198 \\ 2 & 194 & 245 & 188 & 182 \\ 3 & 330 & 201 & 175 & 286 \end{array}$$

FIGURE 5.11: Confusion Matrix for Martingale
Plots of the Training data Loss and accuracy for MLP training

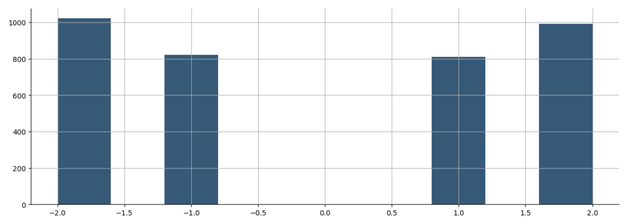


FIGURE 5.12: Histogram for Martingale classification
Histogram Plot of Martingale prediction

5.4.2 Naive Bayes Experiment 3

In contrast, to experiment 2 the Naive Bayes performed the third best in this experiment, The test data showed worse accuracy than the validation split with 31.2328 The validation data showed better accuracy, and recall with 0.34.6153 but worst precision, with only .2658 and both had bias towards the classes 1 and -1.

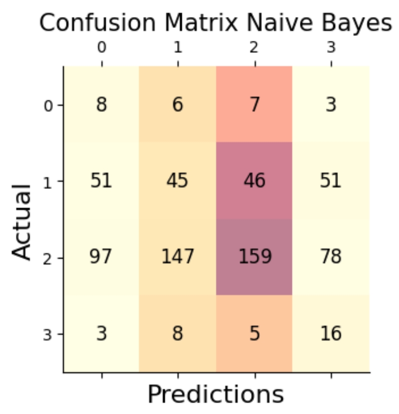


FIGURE 5.13: Predictions for Naive Bayes
Naive Bayes Confusion Matrix

5.4.3 *Logistic Regression Experiment 3*

First, the Logistic Regression was implemented without optimization.

- The Logistic Regression without optimization had an accuracy of 31.9178% for the classification of test data, while 33.5164% for validation data.

- The confusion matrix shows no bias towards the -2 and 2 classes.

- A balanced accuracy of 30.6951% confirms that the threshold for the signal was closer to reality.

With the optimization through the grid the logistic regression presented the following:

- Accuracy of 27.9452 for test and 28.5714 for validation.

- It showed that the model was overfitted as the training accuracy was 0.3339

- The confusion matrix, showed bias towards the classes -2 and 2

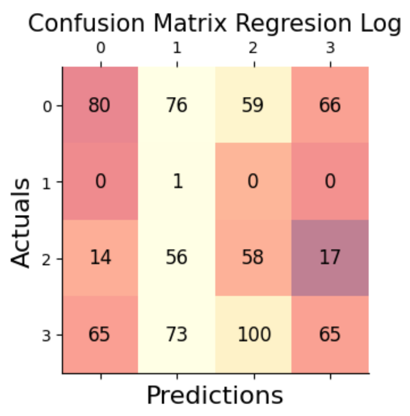


Figure 5.14: Predictions for Regression
Logistic Regression Confusion Matrix

5.4.4 *Random Forest Experiment 3*

- The Training accuracy was 0.4358 which was higher than the test and validation accuracy.

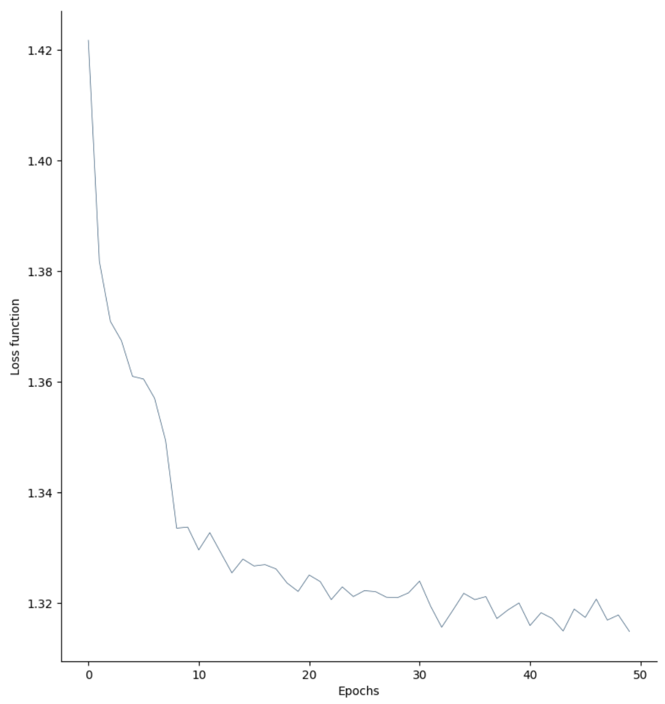
- The random forest with the best estimator was one with a number of estimators of 1000 and max features of square root and criterion gini.

- the test accuracy was 0.3273 and the validation was of 0.3131

- the confusion matrix had numbers in every category and seemed to be random.

5.4.5 *Multi Layer Perceptron Experiment 3*

The MLP was implemented using a grid optimization and had a poor performance as expected in the first experiment, the best training accuracy was 0.3646 with an accuracy of 0.3395 for the testing data and validation of .3175 which was the best performing model in this experiment.



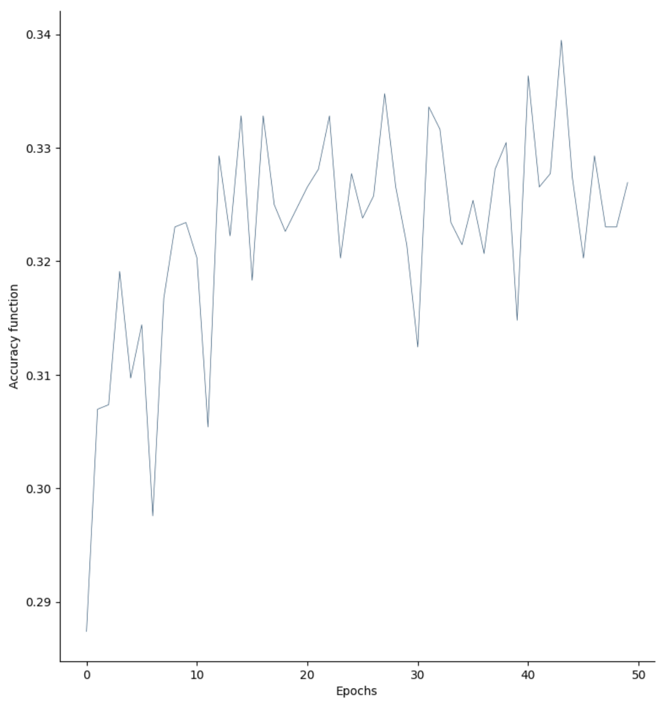


Figure 5.15: Training Performance for MLP
Plots of the Training data Loss and accuracy for MLP training

5.5 Experiment 4

TABLE 5.4: Experiment 4 Model Results

Model	Accuracy	Recall	Precision	F1
Logistic Regression	40.14598540145985	40.14598540145985	34.30413625304137	9.920318
Naive Bayes	27.73722627737226	27.73722627737226	30.3731088685151 43	36.98576592828412
Random Forest	51.82481751824818	51.82481751824818	45.1645983317145 15	45.76283855418012
MLP	42.34	42.34	41.56	43.82

5.5.1 Naive Bayes Experiment 4

The Naive Bayes performed the worst in this experiment, The test data showed better accuracy than the validation split with 27.7372 The validation data showed worse accuracy, and recall with 10.29411 with only .2558 in precision and both had a bias towards the classes 1 and -1.

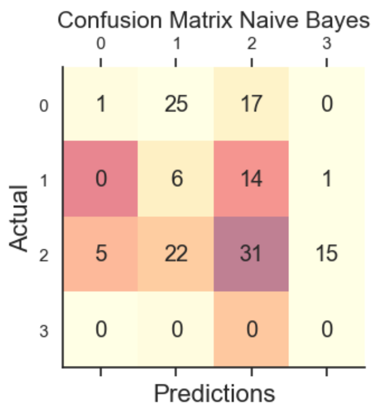


FIGURE 5.16: Predictions for Naive Bayes
Naive Bayes Confusion Matrix

5.5.2 Logistic Regression Experiment 4

First, the Logistic Regression was implemented without optimization.

- The Logistic Regression without optimization had an accuracy of 36.4963% for the classification of test data, while 38.2352% for validation data.
- The confusion matrix shows bias towards the -1 and 1 classes which can be explained for the few samples of the dataset.
- A balanced accuracy of 21.6676% confirms that there are few data and needs more samples.

With the optimization through the grid the logistic regression presented the following:

- Accuracy of 40.1459 for test and 36.76470 for validation.

- It showed that the model was not overfitted. with 0.45255 for training
- The confusion matrix, showed bias towards the classes 1 and -1

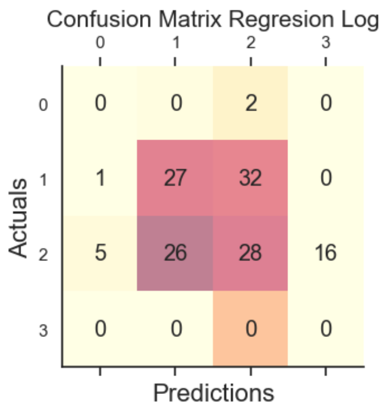


FIGURE 5.17: Predictions for Regression
Logistic Regression Confusion Matrix

5.5.3 *Random Forest Experiment 4*

- The Training accuracy was 0.5135 which was lower than the test and higher than the validation accuracy.
- The random forest with the best estimator was one with a number of estimators of 1000 and criterion log loss and no max features.
- The test accuracy was 0.5182 and the validation was 0.4117
- The confusion matrix had a visible bias towards classes 1 and -1.

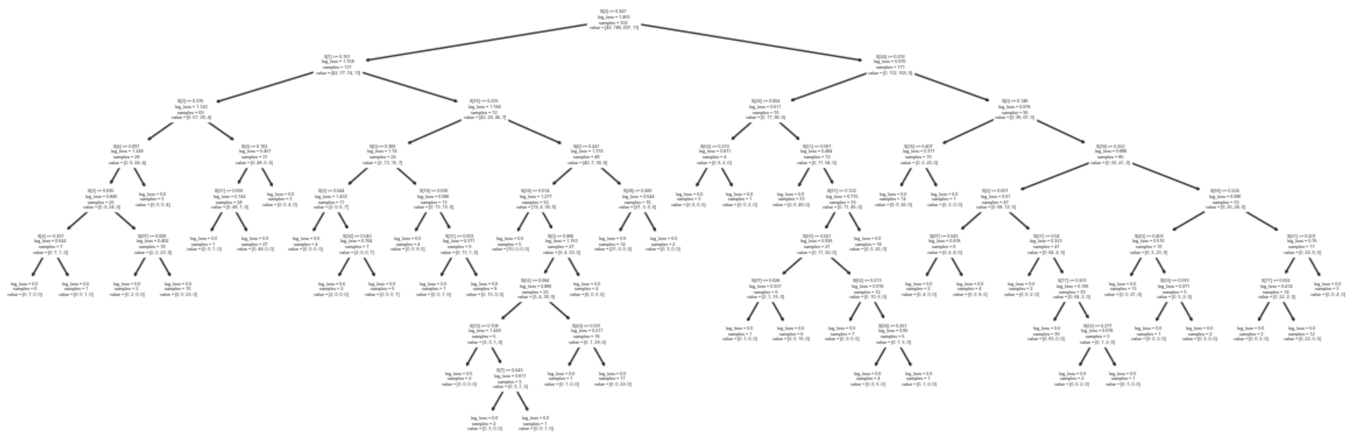


FIGURE 5.18: Tree Generated
Generated tree for the model

5.5.4 *Multi Layer Perceptron Experiment 3*

The MLP was implemented using a grid optimization and had better performance than expected in the final experiment, the best training accuracy was 0.8243 with an accuracy of 0.4234 for the testing data and validation of .5572 which was the best performing model in this experiment, the model appears to be overfitted, and it most possibly is because there were too few data for this experiment.

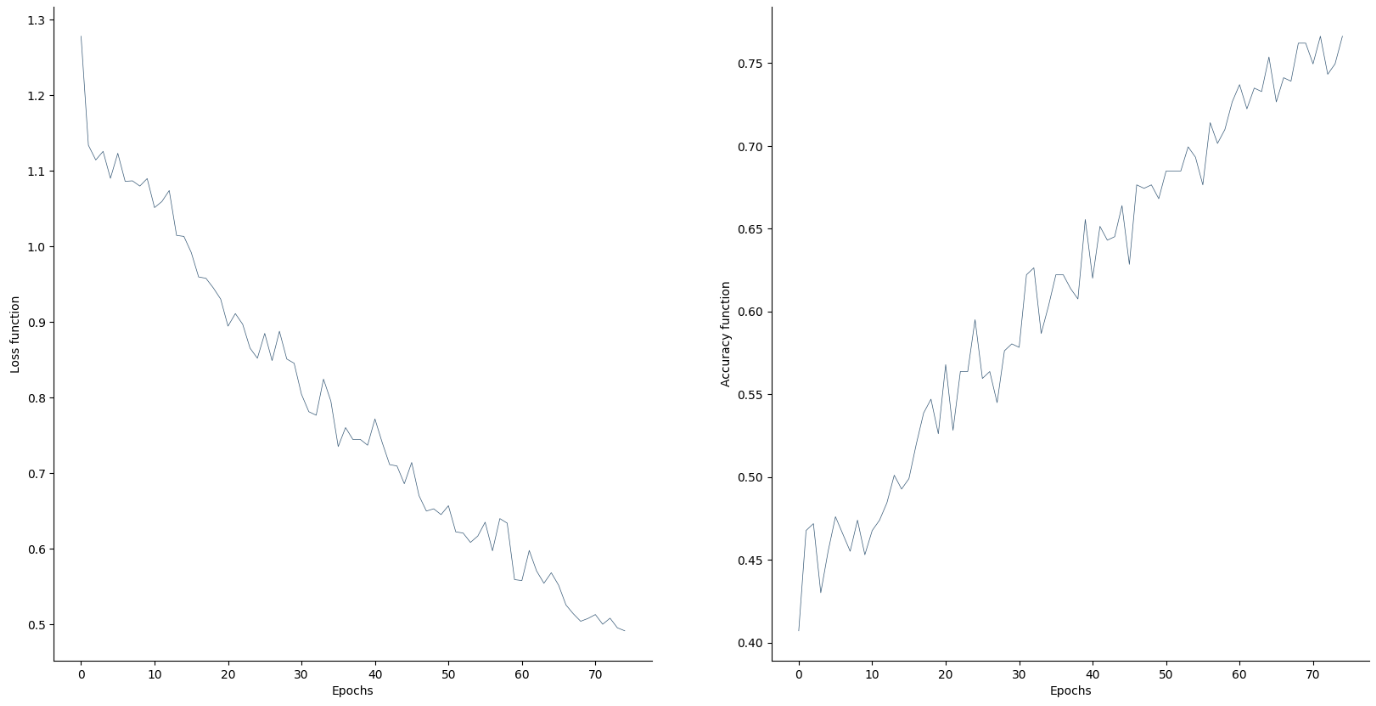


FIGURE 5.19: Training Performance for MLP
Plots of the Training data Loss and accuracy for MLP training

6 Conclusions and Future work

Contents

This chapter includes a discussion of the results shown in chapter 5 and presents the proposal for future work.

6.1 Conclusions

- Symbolic-created features using evolutionary computation have an important part in the process of feature generation for financial time series forecasting, where certain relations between features are hidden or cannot be easily seen.
- The primary objective of implementing a model with accuracy to forecast the movements of the movement of cryptocurrency through 4 labels was not achieved, and it's possible that the problem should be seen as a binary one.
- The use of the three-barrier method vastly improved the metrics for the models in experiment four; even if the data had a sample that was small and had an effect on the performance of the testing and validation phase, the correlation between the target and explanatory features was bigger and had a better performance even if the dataset was smaller.
- the feature engineering and feature selection had an impact on the performance of the models, which can be seen particularly in experiment 4; the use of Uniswap and Publictrades Data helped to improve the predictability of the model, giving more information and permitted the generation of better explanatory features throughout the autoregression and symbolic transformer.
- The best-performing models were the random forest classifier and the Multi-Layer perceptron in experiment 4, the random search grid method for optimization vastly improved the execution times of the models and had better metrics.
- The benchmarks performed the worst in all of the experiments, which is to be expected.
- The crypto market presents different challenges compared to the traditional markets on which to create a forecasting method and the application of a machine learning method.

6.2 Future Work

- The first thing that could be improved in further revisions is the use of a larger period of sample for the models, using a larger quantity of periods of time to train the model, and a larger sample for testing and validation.
- The second thing that could be implemented is the use of stochastic methods to optimize the models, which would improve the training time, as well as the performance of the models.
- Generate the symbolic data only using the test dataset to optimize the process of evolutionary computation for the symbolic transformer and generate data in a more advanced manner.
- The use of a more sophisticated model for classification, such as a Recurrent Neural Network, which could have a better learning capacity for this type of classification problem.
- The use of a cluster for training models with a larger quantity of data to circumvent the time the models require.

Bibliography

Binance (2022).

Ethereum.org (2022). What is ethereum.

Gplearn (2016). Introduction to gplearn.

James Martin, Jack Cunliffe, R. M. (2019). *Cryptomarkets: A Research Companion*. Emerald Publishing Limited.

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

López de Prado, M. M. (2020). *Machine Learning for Asset Managers*. Cambridge University Press.

Muñoz-Elguezábal, J. F. and Sánchez-Torres, J. D. (2021). T-fold sequential-validation technique for out-of-distribution generalization with financial time series data. International Conference on Econometrics and Statistics.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.

Nielsen, A. (2020). *Practical Time Series Analysis*. O'Reilly.

Taber, K. (2009). Learning at the symbolic level. *Models and Modeling in Science Education*, 4:75–105.

yahoo finance (2022).