Genetic Programming and Neural Networks for Financial Predictive Modeling

J.F. Munoz-Elguezabal
franciscome@iteso.mx - 0000-0003-0143-5797
Mathematics and Physics Department
Western Institute of Technology and Higher Education (ITESO)
Tlaquepaque, Mexico

***Abstract*—Financial time series forecasting certainly is the case of a predictive modeling process with many challenges, mainly because the temporal structure of the data. Genetic programming, as a particular variation of genetic algorithms, can be used to as a feature engineering, importance and selection process all at once, it can provide highly interpretable symbolic features that have low colinearity among them and yet high correlation with a target variable. We present the use of such method for generating symbolic features from endogenous linear and autoregressive variables, along with a Multi-Layer Perceptron, to construct a binary predictor for the price of Ethereum cryptocurrency.**

***Index Terms*—Genetic Programming, Neural Networks, Financial Machine Learning**

I. Introduction

The predictive modelling process can be considered as one of the most frequent type of Machine Learning implementations, and although there is no clear and formal definition of its components, such process can be defined in 3 sections, data related process, model related process and performance-generalization related process. This three componentes will be present, along with some variations, in almost every type of application and industry, for financial time series forecasting is certainly the case.

Financial Time Series Forecasting poses many challenges on such process, mainly due to the stochastic nature of the data generating financial processes. When using a regressive approach to forecasting, one common approximation when dealing with financial time series is to generate endogenous variables as candidate features, this often times is problematic because of the temporal structure of the data and the colinearity presented among features, in addition to the well recognized consideration on the efficiency of the markets, which states no information about the future can be derived efficiently from the past in a systematic fashion. An even that financial data is highly available but with a well known low signal-to-noise ratio, meaning that there is no guarantee that more data means more information.

One key aspect of financial time series in general is that eventhough it is frequent to treat it as a univariate variable, one observation at a time, it is at ever time t the only realization of an ever unknown process often times considered as one with stochastic nature. Because of this, There is a clear need of a model that can represent highly non linear phenomena, one strong candidate is an Artificial Neural Network. It is well identified that other challenging process are needed to be address, from model architecture, to hyperparameter optimization, to model performance. In addition, convergence to global minimum of cost function, reproducibility and parametric stability in variations of subsamples are some of the potentially problematic aspects of the implementation of such type of model.

Whether is a classification or a regression perspective, performance metrics will vary and express particularities of their interpretations because of the financial nature of the data. One case of such interpretation is that the generalization capabilities of the model will be, at the very least, questionable at all cases, simply because the one generating process of such data is unknown, there can be only be measured once and such measurement is one-sized, meaning that no matter how well a model fits to the presented data, by definition, it will be always insuficient.

The proposition of this work is threre fold, first we formulate the classical regression problem of forecasting a price change as a classification one, in the sense that the labeling criteria for the target variable will be of a binary class consisting on the sign operation for the open and close price difference within a time period. Then, in order to address the feature engineering step, we propose the use of Genetic Programming to produce non linear variables highly correlated with a target and highly uncorrelated with each other. Finally, we present this work as an implementation of lucidmode, a Python programming framework for transparent and interpretable Machine Learning models, and the implemented model is a Multi-Layer Perceptron with an elasticnet regularization. Training evolution and performance evaluation will be included.

The organization of the document is as follows: In the first chapter a description on the financial data and the key aspects of the formulation of the classification problem. Next a description of the implementation process of the genetic programming method as a feature engineering, importance and selection processes all at once. Next the predictive modeling process, which includes the data description, model definition and optimization and finally the performance measurment. The last section will be for result discussion.

Fig. 1. *Historical OHLC prices data*

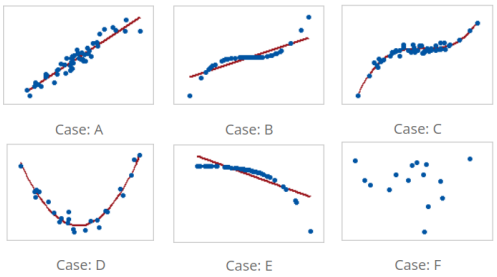

Fig. 2. *Utility cases for Pearson and Spearman Correlations*

II. Financial Time Series

The value of a financial asset can be considered as a continuous variable but the price of it as a discretization of such value, and the frequency of sampling can be defined according to a time criteria but there are other *labeling* considerations [4]. For this work, each time based sample, has 6 values that are always known:

- **Timestamp**: The date and time for each interval.
- **Open**: The first price of the interval.
- **High**: The highest price registered during the interval.
- **Low**: The lowest price registered during the interval.
- **Close**: The last price of the interval.
- **Volume**: The total amount of contracts or transactions during the interval.

A. The OHLC characterization

Let V_t be the value of a financial asset at any given time t in a continuous matter, from which it can be formulated a discrete representation S_t if there is an observable transaction Ts_t. Similarly, a set of discrete prices observed during an interval of time T of $n = 1, 2, ..., n$ units of time, will define $\{S_T\}_{T=1}^{n}$, which in turn will be characterized by the prices $OHLC_T : \{Open_t, High_t, Low_t, Close_t\}$.

B. The OHLC plot

A common visualization for such type of discrete prices representation is the *candlestick* plot, as shown in "Fig. 1" it represents 4 time series, $OHLC_T : \{Open_t, High_t, Low_t, Close_t\}_{t=1}^{T}$. Such frequent representation also provides a visual component to empircally detect missing values, though, is infrequent to encounter such situations for publicly traded financial assets, and even less infrequent for cryptocurrencies since by definition its price fluctuate 24 hours a day, 365 days a year.

III. Linear and Autoregressive Variables

The following operations are used to define four aspects of the dynamic of the discrete values representing a continuous variable, such transformations are frequently used in time series analysis [1]. For this work, in an interval of time, discrete prices will be used as representing the continuous value of a financial asset:

A. Linear Variables

- **micro-volatility:** $\rightarrow HL_t$
  As a measurement of volatility, $HL_t = High_t - Low_t$, represents the complete range of prices observed during the time interval T
- **micro-trend:** $\rightarrow CO_t$
  As a formulation for the directional movement, $CO_t = Close_t - Open_t$, represents both the magnitud and the direction of the value V_T, represente as prices, during the T interval of time.
- **micro-uptrend:** $\rightarrow HO_t$
  Starting at $t = 0$, the greatest positive difference in discrete prices can be characterized with $HO_t = High_t - Open_t$.
- **micro-downtrend:** $\rightarrow OL_t$
  Starting at $t = 0$, the greatest negative difference in discrete prices can be characterized with $OL_t = Open_t - Low_t$.

B. Autoregressive Variables

For the autoregressive features, fundamental operations used were moving average: MA_t, lag: LAG_t, standard deviation: SD_t and cumulative sumation: $CSUM_t$. This operations where applied to the past linear features. For values of $k = 1, 2, ...K$, with K as the *memory* proposed as a parameter. Therefore, we define the following autoregressive transformations to apply to a number of linear variables $\{OL\}_{t-k}$, $\{HO\}_{t-k}$, $\{HL\}_{t-k}$, $\{HLV\}_{t-k}$, $\{COV\}_{t-k}$, $\{VOL\}_{t-k}$

$$MA_t \rightarrow \mu_t^{t+k} = \sum_{i=1}^{n} \frac{x_i}{n} \quad , \quad LAG_t \rightarrow \Delta_t^{t+k} = \Delta_t$$

$$SD_t \rightarrow s_t^{t+k} = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n-1}} \quad , \quad CSUM_t \rightarrow \sum_{t}^{t+k} \overline{x}_{t:k}$$

The target variable is the sign operation of the difference between opening price and closing price. Since this is a discretization of a continuous variable, such transformation results in formulating a regression problem as a classification one, for this particular work, a binary classification.

$$\hat{y}_t = sign\{Close_t - Open_t\} \tag{1}$$

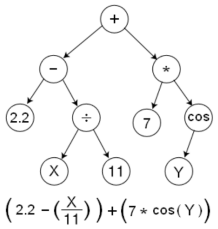

Fig. 3. *A Tree Representation of a Genetic Programm*

IV. Genetic Programming

Genetic programming, as a particular variation of genetic algorithms, can be used as a feature engineering, importance and selection process all at once, it can provide highly interpretable symbolic features that have low colinearity among them and yet high correlation with atarget variable.

It is considered as a derivation or special case of *Genetic Algorithms*, one difference of particular interest for FE is: The output of the process are equations generated from a combination of symbolic operations that can be represented as trees, as shown in "Fig. 3", choosen from a set of possible symbolic operations i.e. summation, exponentiation (wich would have an arity of two), other examples are inverse, logarithm operations (arity of 1). Such programs will be evaluated with a fitness metric, which for this work two were used, the pearson correlation:

$$\rho(x,y) = \frac{E\left[(x-\mu_x)(y-\mu_y)\right]}{\sqrt{E\left[(x-\mu_x)^2\right]E\left[(y-\mu_y)^2\right]}} \tag{2}$$

Another important aspect is that such programs have a tree-like representation, which means that the mathematical complexity of the programm can be characterized by the dept and the breadth of its three representation. And although it does represent the relationship between two variables, it fails when such relationship is non-linear because probability distributions among variables could differ on important statistical aspects such the simmetry, skeness and kurtosis of the distribution. Because of such cases do exist, it was used the *Spearman* correlation, which is a special case of *Pearson* but for $ranked$ variables:

$$r_s = \rho_{rg_X, rg_Y} = \frac{cov(rg_X, rg_Y)}{\sigma_{rg_X}\sigma_{rg_Y}} \tag{3}$$

Spearman is a metric to capture the long term and monotonic correlation, and Pearson for the short term linear correlation. As shown in "Fig. 2", there are different cases of correlation between a candidate explanatory variable and the target variable, in the variety of such cases is the use of using *Pearson* or *Spearman* as fitness metric. In case A is presented a *perfect* linear correlation, B and E is a close to linear but with outliers and monotonically increasing relationship, C is non linear monotonically increasing but les outliers, D and F are the hardest cases since represent non linear and no correlation, respectively.

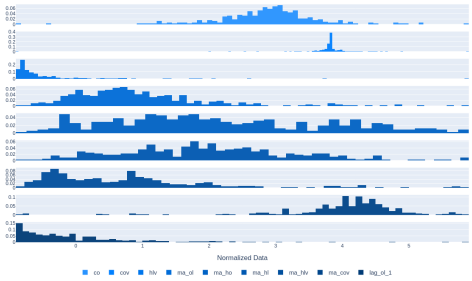

Fig. 4. *Linear, Autoregressive and Symbolic Features*

A. Implementation

The implementation was through the *SymbolicTransformer* method of the *gplearn* python package. It looks to maximize the absolute correlation between the generated feature with the target, it can be chosen a correlation metric like Pearson product-moment correlation coefficient or the Spearman rank-order correlation coefficient.

B. Key aspects of Genetic Programms

Programs do not have a boundary in complexity, that is, as a tree representation, they can be grown indefinitely only to reaching two conditions:

1) Number of generations reached.
2) Metric goal reached.

In order to produce somewhat explainable features a parsimony coefficient will be useful. The library offers the specification of such parsimony coefficient by adding a multiplier to the chosen final metric of every program, a number between $[0, \infty)$, so the more evolved a program the more penalized will be its fitness metric and the less likely to be chosen as part of the final "Hall Of Fame" set of programs.

In "Fig. 4" are shown 8 examples in their histogram form, though the output can be of *n_features*, which in principle will be the most correlated programs to the target variable and the least correlated amongst them, such correlation metric is the same and is specified as the metric parameter. This kind of output is useful since it provides a set of Genetic Programmed Features that captures an indirect relationship, whether linear or monotonic, that can then be exploited by a predictive model.

- Fitness is a demean value transformation in gplearn, calculate .corr() to have original pearson value.
- Is possible to have repeated elements (symbolic features) because: low generations mostly.
- *warm_start* = True for continuing evolution and not loose previous generation.

V. Predictive Modeling

A. Cost function

The binary cross-entropy or logloss cost function was utilized for the implemented model.

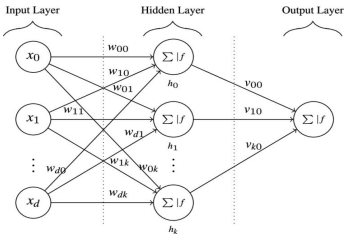

Fig. 5. *FeedForward Multi-Layer Perceptron*

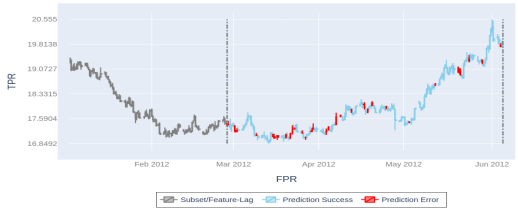

Fig. 6. *OHLC plot for classification results*

$$J(w) = -\frac{1}{m}\sum_{i=1}^{m}\left[y_i\ log(p_i) + (1-y_i)\ log(1-p_i)\right] \quad (4)$$

m: Number of samples.
w: Model weights.
y_i: The *i-th* ground truth (observed) output.
p_i: The *i-th* probabilistically forecasted output.

B. Experiment Definition

A model that complies with the above 3 conditions, for the case of this work, it will be defined as one capable of generalizing sufficiently well for the *Out-Of-Sample* case.

1) **Condition 1**: Accuracy in train set is above 70%
2) **Condition 2**: Accuracy in validation set is above 70%
3) **Condition 3**: Difference between 1), 2) is below 10%

C. Model definition

The predictive model implemented was a Feedforward Multi-Layer Perceptron with Backpropagation, as its represented in its general form in "Fig. 5".

- *Architecture:* 2 hidden layers, tanh, ReLU and Sigmoid as output layer, 32 Minibatch, Binary CrossEntropy cost function, 500 epochs
- *Hyperparameters:* learning rate for optimizer, regularization factors.
- *Weights initialization:* Xavier-uniform criteria as published in this work [3]
- *Regularization:* Weight activity, Bias activity with Elasticnet regularization.

D. Regularization

One core component of the predictive mopdeling prrocess was to add the regularization criteria *elasticnet* [2], for both the Cost Function and the Weight updating in the model:

$$J(w) = J(w) + C\frac{\lambda}{m}\sum_{j=1}^{n}\|w_j\|_1 + (1-C)\frac{\lambda}{2m}\sum_{j=1}^{n}\|w_j\|_2^2 \quad (5)$$

L_1: Also known as $Lasso$
L_2: Also known as $Ridge$
C: A coefficient to regulate the effect between L_1 and L_2

E. Hyperparameter search

It was sufficient to apply a Random Grid Search with memory, by implementing random permutations without repetition of the following values:

- $alpha = \left\{1^{-5} : 1^{-2}, 1^{-3}\right\}$
- $L1\_L2\_ratio = \left\{1^{-2} : 1, 1^{-2}\right\}$

Eventhough a 6 month Fold sub-sampling process was conducted, as it is stated in this work [6], expected prediction error can be estimated, there is no theoretical guarantee that the variance can be defined but only empirically estimated.

VI. Results

A. Feature Engineering

Feature Engineering of Endogenous Variables from simple variables transformed with PG can be a useful process, both for inferring and forecasting, but it will depend on the following process within predictive modeling. Genetic Programming provides a tool to make linear combinations of non-linear variables. Being a heuristic method, ambiguity in design and implementation must be tolerated, which will be based on "context" and which is a process that needs attention on its own. In financial time series there is a high degree of collinearity between endogenous variables, and the definition of the target variable plays a decisive role.

One of the most identifyable characteristic is that, since the heuristic nature of this process, there is no assurance on its generality across samples, more specifically, across samples with the same probability distribution and between samples with different probability distribution.

B. Classification metrics

In "Fig. 6" is shown two aspects of the results found in this work. First, as takingn into consideration the *Leakage of information* stated in [4], an *embargo* operation was performed to each 6-month fold tested for all the years of historical prices. Such *embargo* criteria means to take out n data points between every testing-training (in that order) intersections during Time Series Cross-Validation techniques. The second element is the color indication of success (blue) and failure (red) of the model classifying data during training.

C. Model Performance

Engineering of Endogenous Variables from linear variables transformed with PG can be a useful process, both for inferring and forecasting, but it will depend on the predictive modeling process. Genetic Programming provides a tool to make linear combinations of non-linear variables, which is useful to build a model from a convex approach. Being a heuristic method, ambiguity in design and implementation must be tolerated, which will be based on "context" and which is a process that needs attention on its own. In financial time series there is a high degree of collinearity between endogenous variables, and the definition of the target variable plays a decisive role. In "Table: I" the main metrics of the winning model are presented.

acc-train	0.81
acc-val	0.78
logloss-train	220.12
logloss-val	605.56

TABLE I
Performance metrics of best model

D. Open Source Project

In order to generate the presented results, $+3,000$ lines of python code were created specially for this work. From complementary functions, to data ingestion and formating, parallel processing, logging, results extraction and data visualization. In the next section only the most relevant code snippets are included, the rest of the programmatic functionalities can be consulted in the official Github repository, at: https://github.com/lucidmode/lucidmode, which is an open-source, low-code and lightweight Python framework for transparent and interpretable machine learning models. It has built in machine learning methods optimized for visual interpretation of some of the most relevant calculations.

REFERENCES

[1] Box, Jenkins, Reinsel and Ljung, "Time Series Analysis, Forecasting and Control", Wiley, vol. 5th, 2015.

[2] Zou, Hui and Hastie, Trevor, "Regularization and variable selection via the elastic net", Journal of the Royal Statistical Society, 04/301-320-2005

[3] Glorot and Bengio, "Understanding the difficulty of training deep feedforward neural network", Journal of Machine Learning Research, 09/249-256, 2010.

[4] Lopez de Prado, "Advances in Financial Machine Learning", Wiley, 2018.

[5] Bengio and Grandvalet, "No Unbiased Estimator of the Variance of K-Fold Cross-Validation", Journal of Machine Learning Research, 2004.

[6] Goodfellow, Bengio and Courville, "Deep Learning", MIT Press, 2016.