Feature Engineering Automation for Time Series Analysis

Hélio Domingos
Instituto Superior Técnico.
Lisbon, Portugal
helio.domingos@tecnico.ulisboa.pt

***Abstract*—In recent years, time series data have been one of the most growing types of data since people and business usually measure their performance over a period of time. Temporal data is usually associated with the task of forecast. To forecast, a data scientist needs to create features that help describe the behavior of data. To help data scientists, there have been proposed frameworks that automate the data science pipeline in other fields of research, for example for classification of tabular data, but not for time series. In this work, we describe the main Automate Machine Learning frameworks and propose a new framework that automates the process of creating a pipeline of analysis of time series. The focus is on creating descriptive features, that adapt to the data. Those features are aggregated in sets and they are evaluated and selected for the model. With this work, we aim to deliver a tool that will reduce data scientists' work developing pipelines of analysis, and help them concentrate on the analysis of results.**

***Index Terms*—AutoML, Machine Learning, Time Series, Feature Engineering, XGBoost**

I. Introduction

Data science is a combination of fields from databases to data visualization passing through statistics, data mining, data analysis, and machine learning. In recent years, industry understood the importance of analyzing and extracting value from data in order to improve their business model, decision making and even products.

The data science pipeline includes data collecting, cleaning, exploring and visualizing to understand their structure, creating models and interpreting results. The data scientist needs to know his target problem and be aware of the data limitations. Also, he needs to choose the best algorithm(s) to fit the data and choose the hyperparameters of the algorithms that maximizes accuracy. All these choices need to be well performed by the data scientist and each case is different from the previous one which needs study before applying any technique. This process can consume a lot of time and human resources until achieving good results.

Automated machine learning (AutoML) aims to develop tools that build machine learning models without human intervention. This includes automating each step of the machine learning pipeline that was previously presented. The goal is that users provide data and the AutoML system automatically determines the approach that performs the best for their dataset [1].

Steps of the pipeline such as model selection or hyperparameter optimization were being studied by decades and they are well-defined and optimized. On the other hand, feature engineering needs more attention. There are frameworks that automate this step but with huge needs of computational and time resources.

In this work we purpose a framework that could automate feature engineering applied to time series data. Our proposal is based on the composition of a set of operators which pick the original data series and adds features, enriching the series. Our framework explore different models and different techniques of feature engineering. Those features will be used to forecast the series. To evaluate the forecast, we will compare the results of a neural network model without features against the results of a decision tree model based with feature engineering. With this work, we expect to forecast time series with low error.

II. Related Work

Industry have been focusing on developing frameworks to substitute data scientists. Some frameworks are AUTO-Weka [2], AUTO-Sklearn [3], TPOT [4], Hyperopt-Sklearn [5]. Also, companies such as Google, H2O.ai, Tazi.ai, DataRobot, Feedzai, are developing their own frameworks as their business.

In the following sections present a survey of the most relevant AutoML techniques proposed to date, categorized in accordance with each data science pipeline phase: data cleaning, feature engineering, model generation and hyperparameter optimization.

A. Data Cleaning

Data cleaning is an important step that depending on how is performed can create a big difference in model performance leading to different results. Existing AutoML frameworks understood this importance and included some steps to deal with this problem. These steps usually deal with imputation of missing values, removing of outliers and scaling features to a normalized range. For example, the Data Science Machine [6] only cleans the data by removing the null values, convert categorical variables using one-hot encoding, and normalize features.

B. Automated Model Selection

Model selection is the task of selecting a proper model from a set of candidate models and setting its hyperparameters in order to achieve a good learning performance [7]. There are many classification algorithms and for each one different hyperparameters are associated.

The first work that allowed to select models automatically was AUTO-WEKA [2]. This framework was implemented in the standard WEKA package [8], which is a machine learning framework and combine Bayesian optimization [9] [5] methods to define a good instantiation of WEKA for a given dataset. A new version of AUTO-WEKA [10] supports regression algorithms, optimization of all performance WEKA's metrics, parallel runs to boost computation and completed integration with WEKA. Other popular framework is AUTO-SKLEARN [3] [11], which is built-in Scikit-Learn package.

C. Automated Hyperparameter optimization

The first techniques used to solve hyperparameter optimization were grid and random search. Later works showed that is possible to tune hyperparameter by using a bayesian optimization or genetic algorithms.

1) Grid Search: The first approach proposed to explore combinations of values in the search space was grid search and is the most commonly used method. After setting the search space, this method creates an exhaustive searching through every possible configuration in the search space and evaluate the model for each combination.

2) Random Search: Random Search is an alternative to grid search [12]. It tries to find the optimal hyperparameters randomly, which do not guarantee that finds the optimal values.

3) Sequential Model-Based Optimization(SMBO): SMBO is the state of the art in hyperparameter optimization. In order to run SMBO, it is needed a search space, a metric to optimize (usually accuracy), the surrogate model of the objective function, criteria to select the next promising configuration and history of information about the previous configurations explored.

D. Feature Generation

Feature generation is the process of creation features from a given dataset. The purpose is to explore hidden relationships between features creating new features in order to maximize the model performance. It is hard to generalize because is a task that requires domain knowledge. The number of possible generated features can be limitless since it is possible to perform operations on existed features. This process is the part that most heavily involves humans since it is driven by intuition and knowledge about the domain.

1) Feature Generation without domain knowledge: As part of the Data Science Machine(DSM), it was developed the Deep Feature Synthesis(DFS) algorithm [6] that demonstrated its efficacy in online data science competitions beating human teams. DFS explores the schema of entity-relation datasets. The feature space cardinally grows very quickly due to applying all operators in all features. Most of the generated features are irrelevant for the use case. In order to reduce the feature set, DFS reduces the size of the feature space by employ Truncated SVD transformation. Then, they calculate its $f - value$ according to the target and choose high ranked features.

2) Feature Generation with domain knowledge: In the work of Friedman and Markovitch [13], they developed an algorithm that generates features by using relational expansions. Additionally to the labeled set, the algorithm receives a body of external knowledge represented in relational form. It uses the input features as objects to construct new learning problems with the information provided by the knowledge base. A knowledge base is a library usually made from contributors specialized in some area that contains structure or unstructured data about a specific topic. Examples of knowledge bases are WordNet [14], Wikipedia, TextRunner [15], and many others.

E. Feature Selection

Feature Selection is the process that finds a feature subset based on the original feature set. It can have a huge impact on model performance: memory, computational cost and time. Load the model with too many features can lead to overfit if they are not the proper ones and could take too much time (the curse of dimensionality) to learn the model. These might be enough reasons to spend some time choosing what features better describe data.

Feature selection includes dimension reduction and feature ranking. When the feature set has high dimensionality or great redundancy might be necessary to apply dimension reduction techniques. The truth is some features encode irrelevant information in a specific context (e.g. the first name of people with cancer). These techniques can be divided into feature selection or feature projection. In the first type, the most common methods are lasso and greedy search. Feature projection transforms the data in high-dimensional space to a lower one, where the most used techniques are principal component analysis (PCA), non-negative matrix factorization (NMF) and linear discriminant analysis (LDA).

F. Time Series

According to Esling [16], time series are sequences of measurements for a single entity over time, which are collected at equally spaced points in time, describing the behavior of a process, system or event.

We define a time series, ts, as a vector of timestamps, t_i, with length N, and associated measurement x_t. Each measurement can be a combination of D variables. Therefore, a time series is defined as:

$$ts : X = (x_1, x_2, ..., x_N) \in \mathbb{R}^{N \times D} \tag{1}$$

where, for each $t \in \{1, 2, ...N\}$, $x_t \in \mathbb{R}^D$ represents the t-th measurement of all variables D. Also, the N measurements have been collected at equally spaced time intervals.

A stationary time series is one whose properties do not depend on the time at which the series is observed. Thus, time series with trends, or with seasonality, are not stationary — the trend and seasonality will affect the value of the time series at different times. On the other hand, a white noise series is stationary — it does not matter when you observe it, it should look much the same at any point in time.

In order to automatically know if a time series is stationary and what to do if so, we can apply the Augmented Dickey-Fuller test.

1) Augmented Dickey-Fuller test: The Augmented Dickey-Fuller test is a type of statistical test called a unit root test. There are a number of unit root tests and the Augmented Dickey-Fuller may be one of the more widely used. It uses an autoregressive model and optimizes an information criterion across multiple different lag values

- Null Hypothesis (H0): It suggests the time series has a unit root, meaning it is non-stationary. It has some time dependent structure.
- Alternate Hypothesis (H1): It suggests the time series does not have a unit root, meaning it is stationary. It does not have time-dependent structure.

If the series is non-stationary, we need to manipulate it in order to have a stationary time series. One way to make a non-stationary time series stationary is to compute the differences between consecutive observations. This is known as *differencing*. Differencing can help stabilise the mean of a time series by removing changes in the level of a time series, and therefore eliminating (or reducing) trend and seasonality.

2) Time series Forecast: Hyndman [17] defines the time series forecast task as "predicting the future as accurately as possible, given all of the information available, including historical data and knowledge of any future events that might impact the forecasts".

3) Ensemble-based Approach: Ensemble methods is a machine learning technique that combines several base models in order to produce one optimal predictive model. XGBoost [18] is short for Extreme Gradient Boosting and is an efficient implementation of the stochastic gradient boosting machine learning algorithm. The stochastic gradient boosting algorithm, also called gradient boosting machines or tree boosting, is a powerful machine learning technique that performs well or even best on a wide range of challenging machine learning problems. Tree boosting has been shown to give state-of-the-art results on many standard classification benchmarks. It is an ensemble of decision trees algorithm where new trees fix errors of those trees that are already part of the model. Trees are added until no further improvements can be made to the model.

4) Neural Networks Approach: Artificial neural networks draw inspiration from computational biology. The advantage of neural networks is the ability to learn highly nonlinear patterns in the data. As in the scope of machine learning, neural networks are function approximators that not only learn a mapping from X to Y , or Y given X, but are able to learn novel representations of the data. Artificial neural networks do not have the memory to understand sequential data. The idea behind Recurrent Neural Networks(RNNs) is to make use of sequential information. RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being depended on the previous computations. They are networks with loops in them, allowing information to persist, [19]. RNN's have troubles about the short-term memory. If a sequence is long enough, they have a hard time carrying information from earlier time steps to later ones. Therefore, these causes the need of Long Short Term Memory (LSTM) which is a special kind of RNN's, capable of learning long-term dependencies. LSTM's have skills to remember the information for a long periods of time.

G. Python Libraries

The most common approach to classify time series focuses on manually calculate its properties by applying basic statics like mean, variance and others, and explore other measures like signal processing. Then these properties are used as features.

For feature engineering, libraries such as `pandas` [20] or `sickit-learn` [21] provide a lot of useful methods. Recently, were developed two `python` libraries to automatically extract features from time series which enable automated calculation of important features.

`tsfresh` [22] implements interfaces like the ones mentioned above which allow integrating with traditional pipelines. It provides 63 time series characterization methods, which compute a total of 794 time series features. Each time series is represented as a feature vector with size $[n_{samples}, n_{features}]$ rather than the raw dataset with size $[n_{samples}, n_{timestamps}]$. This output can be used as input to any machine learning algorithm. Features are ranked and selected by the `FRESH` algorithm implemented by this library. The algorithm performs hypothesis tests to measure the dependency between the target labels and each feature's values, and selects a subset of the features based on the p-values computed by these tests. Its widespread adoption shows that the market and recent needs due high production of temporal data require a way to automatize feature engineering [23] [24].

H. Open Issues

The frameworks present at the beginning of this section focus on supervised learning. Those frameworks enable domain experts building reasonable well-performing machine learning pipelines without the need to understand how to do it [25]. Most of them are specialized in model selection and hyperparameter optimization, while feature engineering is not so much advanced. Besides, feature engineering with domain knowledge is not very developed, and what is done requires external knowledge sources.

III. Framework

We designed a system with a set of modules that represent the data science pipeline. This way each module can be changed by other implementation, it is especially important when we want to compare our results with other alternatives for feature engineering.

A. Load Data Module

The Load Data Module is responsible for loading data from a given file, in one of several formats, such as `sql`, `csv` and `xls` files. It receives as input a string path to the data source and returns time series structure populated with given data as output. This module is also responsible for analysing the index and calculate which granularities can be explored (for example: if data corresponds to a window of one year it is not possible to forecast if the framework aggregate all the data by year).

B. Data Profiling Module

We included a step of exploration of the data to better interpret and model the series. We explore stationarity, level, trend, seasonality and noise. This data can be used to help fill missing values and later help build a model. Also, these insights are shown in the visualization tool.

1) Test of Stationarity: Augmented Dickey-Fuller test: As present in section II-F, this test is the most widely used when we want to test stationary. Here we define the threshold of the p-value and its meaning.

- $p - value > 0.05$: Accept the null hypothesis (H0), the data has a unit root and is non-stationary.
- $p - value <= 0.05$: Reject the null hypothesis (H0), the data does not have a unit root and is stationary.

If the test suggest the time series is non-stationary, we will compute the differences of first order. Then, we apply the test again. If the result is negative, we repeat the process with the second order difference, until the test suggests that the time series is stationary.

C. Data Cleaning Module

There are several alternatives to cleaning or curating a dataset. Some methods are correlated to the domain of the problem, or some insight that the user knows about the data and he knows which method the framework should use. We did not perform all of them neither tried to automatize this step.

Usually, in time series problems data scientists start by creating a new range of dates between the first timestamp and the last one. This may originate a lot of missing values. In our case, we understand that by doing that we might ending create data points that do not exist in the real world application. Duplicate values with the same timestamp are removed. Detection of outliers is also an important task to identify data points that do not correspond to the seasonal pattern or its value is far from the trend. Smoothing a time series remove the influence of outliers, reduce ups and downs, and the time series get close to its trend. This can improve the model performance and it is better for larger datasets.

We developed methods for missing values and methods to smooth time series. They are: Moving average - this method is responsible to smooth the time series using a sliding window; interpolate missing values - this method is responsible for imputation of missing values with one of the following techniques: delete the missing values, imputation with the mean or imputation with the seasonal value.

D. Feature Engineering Module

In this module, we explore different ways to generate features. We define two types of operators: extraction-based and granularity-based. The extraction-based are operators that exploit hidden features on the target instances. The granularity-based operators will exploit different time granularities on data. They are responsible to create a new time series with a smaller time interval by aggregating data points. To help the analysis and the testing of each operator we will separate the operators in groups. The groups are:

- **Lag Features** This group of features represent the past observations. The operator Add the data point at time $t - n$ to the tuple of observations at time t.
- **Time Features** This group of features represent the timestamp unities. It can be the ”second of minute”, ”minute of hour”, ”hour of day”, ”day of year”, ”day of month”, ”day of week”, ”week of year”, ”month”, ”quarter” and ”year”.
- **Aggregation Features** This group of features represent transformations of the behaviour of the trend or aggregate data point with the same time granularity by one of five aggregate functions, namely: sum, maximum, minimum, average or median.
  - **Up or down?** Looks to the result of the Difference operator and evaluate if the difference is positive or negative. If is positive, the operator returns `True`, is is negative, return `False`.
  - **Big Up** Looks to the result of the Difference operator and evaluate if the difference is above or below the standard deviation(calculated on difference operator). If is positive, the operator returns `True`, is is negative, return `False`.
  - **Big Down** Looks to the result of the Difference operator and evaluate if the difference is below or above the negative standard deviation(calculated on difference operator). If is positive, the operator returns `True`, is is negative, return `False`.
  - **Difference** Compute a new data point given by the difference between one point and its successor.
  - **Derivative**: Compute a new data point by the n derivative.
  - **Time-granularity**: Aggregate all data points by the same time-granularity and uses an aggregation function from the set {sum, maximum, minimum, average, median}: ”second-granularity”, ”Minute-granularity”, ”Hour-granularity”, ”Day-granularity”, ”Week-granularity”, ”Month-granularity”, ”Year-granularity”.
- **Smooth Features** This group of features represent different types of moving averages that transform the series in a series closer to the trend:
  - **Moving Average** Add a new value to the tuple of observations at time t that is the n moving average calculating by the average of different subsets of size n of the full data set.
  - **Exponential Moving Average** Add a new value to the tuple of observations at time t that is the n exponential moving average calculating by the

average of different subsets of size n of the full data set and places a greater weight and significance on the most recent data points.
  - **Discrete Wavelet Transform**: Compute a new data point given by the Fourier Transform.
- **Composition Features** After test each feature, we select the best features from each category above, with the methods described in the next section, and the framework calculates a new feature that is the composition of two features selected. For example, if a feature selected from the group Smooth Features is ”moving average of window 5”, and a feature selected from the group Aggregation features is ”Up or down?”, we calculate the feature ”Up or down?” with the data from feature ”moving average window=5”. This way we create another feature that is the composition of two well performed features.

1) Feature Selection: As explained before, Feature Selection is an important task to reduce the model complexity and overfit. When a feature is generated we observe its statics. If it has more than 30%of missing values or a standard deviation below 1, the feature is deleted.

After the model computation in the Forecasting Module, it is calculated a score, a the five top ranked features will be used in the Final model. The rest are deleted.

2) Baseline Approach: In our implementation we use four LSTM layers with 50 neurons, each one with a dropout of 20%, to avoid overfitting. The last layer is a simple layer with one neuron that will indicate the value of the forecast. This means that our baselines needs to learn 6050 edges.

The training takes at most 10 epochs and we have a early stop condition that is if the error became steady in at least 3 epochs the train stops. Here, we do not use generated features to enrich data.

3) Regression Ensemble Approach: The ensemble that we use in this framework is the Xgboost algorithm. It is divided into 5 stages. It starts by using the features from the Lag Features group; then uses the Time Features group; the Aggregation Features group; the Smooth Features group; the Composition Features group; and final builds a Final Model with a selection of features of each group of features.

E. Visualization Module

Visualization is a key factor when we talk about process automatization. The user needs to understand the steps performed, important decisions that were taken, and more important, the results.

We use a open-source framework - `streamlit` [1] - that allows us to integrate with our code and create a simple and interactive app. The user can explore the data, features and line charts.

IV. Case studies

Each test, or case study analysis, follows the same structure. The data from the case study is used as input to the framework. In each step we will discuss the results provided by the visualization tool.

[1]https://www.streamlit.io/

1) Metrics: Measuring the accuracy of the overall predictions with a single metric is not simple as there is no metric that could describe the error behavior. In our framework, we display a set of different measures to help the user making decisions about the predictions.

The **mean absolute error**, or MAE, is calculated as the average of the forecast error values, where all of the forecast values are forced to be positive.

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|e_t| \tag{2}$$

The **mean absolute percentage error**, or MAPE, is the average of the percentage errors. Percentage errors have the advantage of being unit-free, and so are frequently used to compare forecast performances between data sets. The most commonly used measure is:

$$MAPE = \frac{100}{n}\sum_{t=1}^{n}\left|\frac{e_t}{y_t}\right| \tag{3}$$

Measures based on percentage errors have the disadvantage of being infinite or undefined if $y_t = 0$ for any t in the period of interest, and having extreme values if any y_t is close to zero.

The **root mean squared error**, or RMSE, is a quadratic scoring rule that also measures the average magnitude of the error. It’s the square root of the average of squared differences between prediction and the observed value.

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(e_t)^2} \tag{4}$$

The **Pearson correlation coefficient**, for short correlation, is a measure of the strength of a linear association between two variables and is denoted by r. It can take a range of values from +1 to -1. A value of 0 indicates that there is no association between the two variables. A value greater than 0 indicates a positive association.

Finally, the **accuracy**. We evaluate the forecast problem as a classification problem, where the output means if the value will increase or decrease relative to the day before. The positive variable is ”increase” and the negative is ”decrease”. Accuracy is given by the sum of instances well predicted as ”increase” or ”decreased” divided by the total of predictions.

2) Data: The data selected consist in a set of financial series with different characteristics. We examined the daily change of closing prices of Nio and the SP500 index. The stock prices are downloaded from Yahoo finance. The main reason for choosing each dataset it is their variation along time. The Nio Stock looks like a valley. It starts with a high value, decreases, and then increases. Also, it was chosen by the number of data points, which are much less than the other case studies. The SP500 index is the most linear trend. This was chosen mostly by its uptrend and as a good representation of all financial time series.

A. *Nio Stock*

The Nio stock is made up of 505 data points, with a low standard deviation of about 3.7 and is not stationary. On Figure 1 is shown the daily values at the closing market time, from Sep 12, 2018 to Sep 14, 2020. Visually, the stock value behaviour can be split into three periods of time.

The first period goes from Sep 12, 2018 to Mar 7, 2019. The stock opens negotiating at 6.6$, and quickly goes up to 11.6$, a 75% appreciation, in the second day on the market. This value will be the maximum until Jul 7, 2020 when it reaches the price of 13.22$. This period is marked by a high volatility with significantly changes day for day. The second period ends on May 22, 2020. During this period, the price decreases, devaluing 81%, when it reaches 1.32$ on Oct 1, 2019. This is the historical minimum. After that, the trend is positive, and the stock appreciate 166%.

The third period has a notable upward trend. The stock appreciate 1050%, reaching 37.7$. From Figure 1, it is visible a pattern that can explain future values. That pattern consists in a appreciation of 7$ in one or two days, followed by a small period of devaluation.

This dataset is relative small, with only 505 data points. This is an important aspect to take in consideration in the analysis that follows. Also, the test period corresponds to the one with higher changes that aren't observed before.

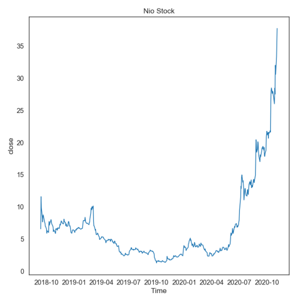

Fig. 1. Daily close value for Nio stock between Sep 12, 2018 and Sep 14, 2020.

1) Data profiling: The framework starts by analyzing the stationary property. The Augmented Dickey-Fuller test informed that the data is non stationary, so it performs a first-order differencing. The time series after data profiling is shown in Figure 2, where we can see a different time series. The new values are more bonded between -10 and 10, mean is close to zero, and the values outside the range -10 to 10, correspond to the first and last period, specially the last one when the stock grew very fast.

2) Baseline: The MAE is about 12.86, RMSE of 16.48 and MAPE is 435.07, accuracy is near 0.46, which means that the model only could predict correctly less than half of the positive variations. The predicted values are relative well bounded in the range of observed values. In the observed time series we have a high value followed by a low value, resulting in consecutive ups and downs. However, in the predicted time

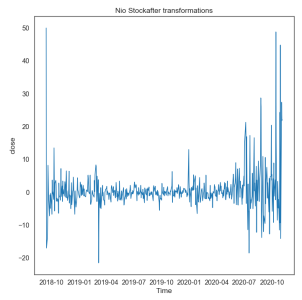

Fig. 2. Nio stock data after data profiling.

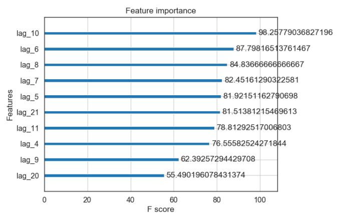

Fig. 3. Lag Feature Importance.

series we have two ups or two downs consecutive before changing the trend. This result in a low accuracy and high errors. The results are lacking correlation with the observed values, and the correlation coefficient is near zero.

3) Lag Features: The MAE is about 10.09, RMSE is about 14.09, MAPE is 163.78, accuracy 0.47, and correlation 0.08. Although very poor results, they are slightly better than the baseline. The predictions are very disperse without a clear correlation between observed and predicted. Analysing the features used that corresponds to previous observations, we can't name a feature or a set of features that can describe the data. The score of each feature shown in Figure 3 indicate that lag_2 was the most used feature to split the data. Since XGBoost built different trees each time we run the algorithm, these feature scores can have different values. We ran the procedure several time and the top features, namely lag_2, lag_6 and lag_5, always scored high.

In the baseline analysis, we saw that the observed time series has consecutive ups and downs. So, an even number of previous observations will help to notice that pattern. And since in the last period we have a very high upwards trend, recently previous observations will help to notice the upward trend, than the older observation, when the value was lower. Finally, lag_11 and lag_20 play and important role to help discover the patterns in the last period.

4) Time Features: The MAE is about 9.43, RMSE is about 13.44, MAPE is 120.10, accuracy 0.56 and correlation 0.19. The results are better than baseline and Lag Features. This means that the time of buying the Nio Stock is more important that previous observations. Again, we can apply the analysis made about Lag Feature and apply here. These features are related to the day of the year, day of the week, as explained

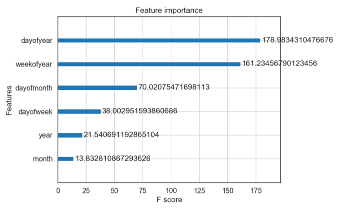

Fig. 4. Time Features Importance.

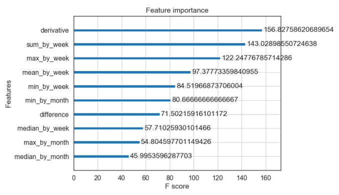

Fig. 5. Aggregation Features Importance.

in section III-D. The feature with the highest score is "week of the year". Since we have a small dataset, with only 3 years of data, it wouldn't be clear that this feature can split the most data points.

5) Aggregation Features: Similar to Lag Features, aggregation features are correlated to past values, however they are transformations of those past values. MAE is about 8.11, RMSE is about 11.22, MAPE is 113.62, accuracy is 0.67 and correlation is 0.60. These results show a better performance than previous approaches and a significant increase in both accuracy and correlation when compared against our baseline. In Tesla analysis similar results were obtained. Now we have a clear correlation between actual data and predicted values. "Derivative" is the feature with highest score. It is a very representative feature, it always scores higher if we ran the procedure several times. Features aggregated by weeks follow the top. "Mean by week" and "max by week" can help describe the patterns mentioned before. The rest of features are aggregations to the values of weeks and months. Days are not present since we have only one observation for each day.

6) Smooth Features: The results of smooth features are the best ones. MAE is about 2.84, RMSE is about 5.65, MAPE is 31.25, accuracy is 0.93, and correlation 0.93. The data points have a high correlation and small errors. In Figure 6, we have the top scoring features. The three Exponential smoothing features used rank in the top, meaning that the most recent values describe better than previous ones. This is in line with

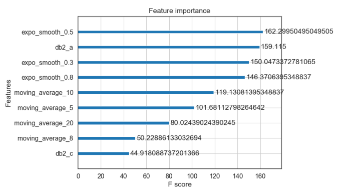

Fig. 6. Smooth Features Importance.

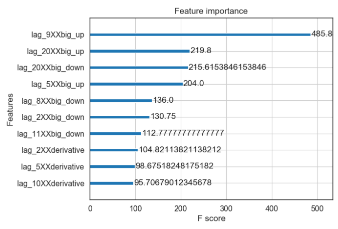

Fig. 7. Composition Features Importance.

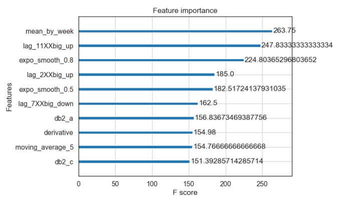

Fig. 8. Final Model Features Importance.

the results from Aggregation features, where the top ones are related to small unities of time. Also, it is according to Lag Features, where the most recent observations scored higher in feature importance. Followed by Exponential moving average features, we have normal moving averages. The first one to appear in the rank is the one with lowest range, of 8. Finally, features extracted from wavelet transformations have a lower score.

7) Composition Features: The results of composition features are far from the results expected. MAE is about 10.32, RMSE is about 14.22, MAPE is 129.40, accuracy is 0.40, and correlation -0.04. The observed values and predicted values do not have correlation. The range of the predicted values are between -6 and 6, although the observed values are in a range between -10 and 50. The top features are all related to lag features. This can explain the bad results, since Lags features obtained similar errors. Also, composition features have more features than the other sets of analysis. This could lead to overfit and increase the complexity of the model.

8) Final Model: The final model joins all features selected from each approach. The final result has the best performance above all. MAE is about 1.87, RMSE is about 2.93, MAPE is 19.02, accuracy is 0.98, and correlation 0.92. The best features chosen, shown in Figure 8, are picked from all sets of study. Meaning that in spite of Smooth Features results have an high accuracy, high correlation and low errors, the other features could be important too. "lag_7 composed with big down" is the most descriptive feature, followed by "mean by week" and "exponential smoothing with alpha of 0.8". In the analysis of smooth features, we said that an exponential moving average with a lower alpha describes the data better than an higher alpha. Here we see the opposite, meaning that is the set of features that matter and not only one feature by itself.

Table I resumes the results from each approach. The final model has a superior performance compared to the baseline, the observed values are correlated to predicted values, which

TABLE I
RESULTS FROM NIO DATASET.

	MAE	RMSE	MAPE	Accuracy	r
Baseline	12.86	16.48	435.07	0.46	-0.03
Lags Features	10.09	14.09	163.78	0.63	0.17
Timestamp Features	9.43	13.44	120.10	0.56	0.19
Aggregated Features	8.11	11.22	113.62	0.67	0.60
Smooth Features	2.84	5.65	31.25	0.93	0.93
Composition Features	10.32	14.22	129.40	0.40	-0.04
Final Model	1.87	2.93	19.02	0.98	0.92

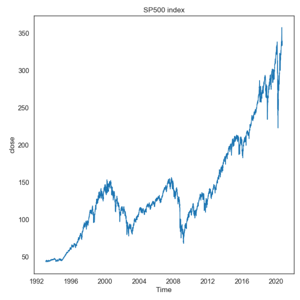

Fig. 9. Daily close value for SP500 index between Jan 29, 1993 and Sep 14, 2020.

do not happen on the baseline, and the final model is able to predict more correctly if the next data point will be higher or lower than the actual one. This can be very important in the context of stock forecast, where the user could know if the stock value will increase or decrease and that way do a more informative action.

V. SP500 INDEX

The SP500 index is made up of 6957 data points, with a high standard deviation of about 68.23 and is not stationary. On Figure 9 is shown the daily values at the closing market time, from Jan 29, 1993 to Sep 14, 2020. Visually, the stock value behaviour can be split into two periods of time.

The first period goes from Jan 29, 1993 to Mar 9, 2009. The index value opens negotiating at 44, and gradually goes up to 154, a 250% appreciation, after 7 years. Then it depreciates 55% along 2 years. The pattern repeats once more. The second period has a notable upward trend. The index value appreciate 500%, reaching 360.

This dataset is big, with 6957 data points. This will be more informative than other dataset with a smaller size. This way, the model can learn better the trend and patterns. Also, the last period is the most interesting and it corresponds to one third of the all dataset. In the Nio stock analysis, we saw that the most interesting period only corresponded to 1/10 of the all dataset. Of course, this factor will impact the model performance.

A. Data profiling

The framework starts by analyzing the stationary property. The Augmented Dickey-Fuller test informed that the data is non stationary, so it performs a first-order differencing. The new values are more bonded between -100 and 100, with some

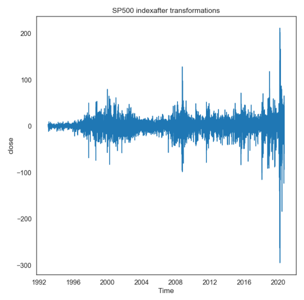

Fig. 10. SP500 index data after data profiling.

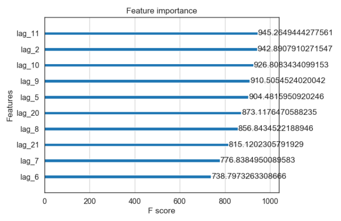

Fig. 11. SPY500 index: Lag Feature Importance.

exceptions. Mean is close to zero, and the distribution fit a normal distribution.

B. Baseline

The baseline results, or the LSTM results, show poor predictions. The MAE is about 25.92, RMSE of 40.50 and MAPE is 435.07. Accuracy is near 0.48, which means that the model only could predict correctly less than half of the positive variations. The predicted values are relative well bounded in the range -10 to 0 and between 5 and 15. We can say that the model was not able to predict value outside this range and because of that it has a bad performance. Results are lacking correlation with the observed values, and the correlation coefficient is near zero.

C. Lag Features

These lags features results show poor predictions The MAE is about 27.49, RMSE is about 44.14, MAPE is 400.10, accuracy 0.48, and correlation -0.08.

Predictions are very disperse without a clear correlation between observed and predicted values. Analysing the features used that corresponds to previous observations, we can't name a feature or a set of features that can describe the data. The score of each feature shown in Figure 11 indicate that lag_11 was the most used feature to split the data, tied with lag_5.

D. Time Features

Time features show poor predictions. The MAE is about 26.29, RMSE is about 41.90, MAPE is 362.67, accuracy 0.52 and correlation -0.02. Again, we can apply the analysis made about Lag Feature and apply here. The data points are too disperse to say that there is a correlation between observed values and predicted values. These features are related to the

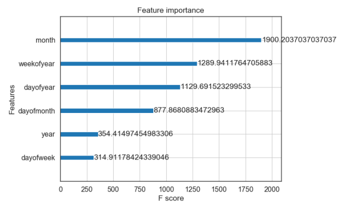

Fig. 12. SPY500 index: Time Features Importance.

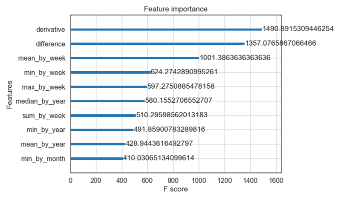

Fig. 13. Aggregation Features Importance.

day of the year, day of the week, etc, as explained in section III-D. The feature with the highest score is ”week of the year”.

E. Aggregation Features

Similar to Lag Features, aggregation features are correlated to past values, however they are transformations of those past values. MAE is about 19.14, RMSE is about 30.63, MAPE is 270.64, accuracy is 0.73 and correlation is 0.65. These results show a better performance than previous approaches and a significant increase in both accuracy and correlation when compared against our baseline.

”Derivative” is again (as in the Tesla and Nio case study) the feature with highest score. It is a very representative feature, because it double the score of the second most used feature, ”difference”. Features aggregated by weeks follow the top. ”Mean by week” and ”max by week” can help describe the patterns mentioned before. The rest of features are aggregations to the values of weeks and months. Days are not present since we have only one observation for each day.

F. Smooth Features

The results of smooth features are the best ones. MAE is about 4.95, RMSE is about 15.28, MAPE is 32.48, accuracy is 0.96, and correlation 0.93.

The data points are over the regression line. Resulting in a high correlation and small errors. In Figure 14, we have the top scoring features. The three Exponential smoothing features used rank in the top, meaning that the most recent

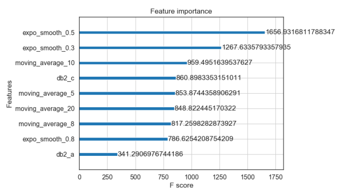

Fig. 14. Smooth Features Importance.

values describe better than previous ones. This is in line with the results from Aggregation features, where the top ones are related to small unities of time.

Followed by Exponential moving average features, we have normal moving averages. The first one to appear in the rank is the one with lowest range, of 5. Finally, features extracted from wavelet transformations have a tied score with the rest of moving average features.

G. Composition Features

The results of composition features are far from the results expected. MAE is about 27.40, RMSE is about 43.22, MAPE is 98.65, accuracy is 0.49, and correlation -0.05.

The observed values and predicted values do not have correlation. The top features are all related to lag features. This can explain the bad results, since Lag features obtained similar errors. Also, composition features have more features than the other sets of analysis. This could lead to overfit and increase the complexity of the model.

H. Final Model

The final model joins all features selected from each approach. The final result has the best performance above all. MAE is about 3.37, RMSE is about 14.46, MAPE is 21.05, accuracy is 0.98, and correlation 0.94. The best features chosen, shown in Figure 16, are picked from all sets of study. Meaning that in spite of Smooth Features results have an high accuracy, high correlation and low errors, the other features could be important too. ”lag_20 composed with big down” is the most descriptive feature, followed by another composed feature, ”lag_10 composed with big down”. In the analysis of smooth features, we said that an exponential moving average with a lower alpha describes the data better than an higher alpha. Here we see the opposite, meaning that is the set of features that matter and not only one feature by itself.

Table II resumes the results from each approach. The final model has a superior performance compared to the baseline,

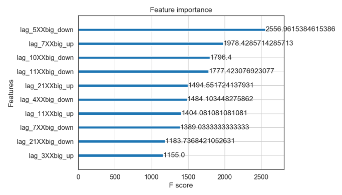

Fig. 15. Composition Features Importance.

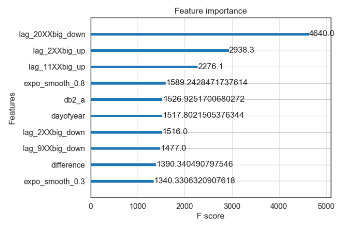

Fig. 16. Final Model Features Importance.

TABLE II
RESULTS FROM SP500 INDEX DATASET.

	MAE	RMSE	MAPE	Accuracy	r
Baseline	25.92	40.50	435.07	0.48	0.03
Lag Features	27.49	44.14	400.10	0.48	-0.08
Time Features	26.29	41.90	362.67	0.52	-0.02
Aggregated Features	19.14	30.63	270.64	0.73	0.65
Smooth Features	4.95	15.28	32.48	0.96	0.93
Composition Features	27.40	43.22	98.65	0.49	-0.05
Final Model	3.37	14.46	21.05	0.98	0.94

the observed values are correlated to predicted values, which do not happen on the baseline, and the final model is able to predict more correctly if the next data point will be higher or lower than the actual one. This can be very important in the context of stock forecast, where the user could know if the stock value will increase or decrease and that way do a more informative action.

VI. CONCLUSIONS

In this work we proposed a framework that automatize the process of analysis of a time series. The framework starts by cleaning the data, generate features, analyse those features, select the best features and create a model that can predict values with low error.

The framework was able to process a set of different datasets and deal with their differences and create different models with different sets of features. Through the visualization tool, the user can follow the process and be informed of each decision of the framework, for example what is the transformation applied if the time series is not stationary, or if it has missing values. Then it shows the sets of features computed and the results of forecast with the XGBoost model. The user can see the Mean Absolute Error, the Root Mean Squared Error, and the Mean Absolute Percentage Error, accuracy and a plot of the correlation between the observed data and predicted values.

With our cases studies we observe that the Final Model of each dataset has always a better performance than the chosen baseline.

To improve this framework, we suggest automate the optimization of parameters of the XGBoost model; test other methods to select features; explore Wavelet transformations to generate features.

REFERENCES

[1] Frank Hutter, Lars Kotthoff, and J. Vanschoren. *Automatic machine learning: methods, systems, challenges*. Challenges in Machine Learning. Springer, Germany, 2019.

[2] Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Auto-weka: Automated selection and hyper-parameter optimization of classification algorithms. *CoRR*, abs/1208.3719, 2012.

[3] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in neural information processing systems*, pages 2962–2970, 2015.

[4] Randal S Olson and Jason H Moore. Tpot: A tree-based pipeline optimization tool for automating machine learning. In *Automated Machine Learning*, pages 151–160. Springer, 2019.

[5] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, pages 2546–2554, 2011.

[6] James Max Kanter and Kalyan Veeramachaneni. Deep feature synthesis: Towards automating data science endeavors. *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–10, 2015.

[7] Quanming Yao, Mengshuo Wang, Hugo Jair Escalante, Isabelle Guyon, Yi-Qi Hu, Yu-Feng Li, Wei-Wei Tu, Qiang Yang, and Yang Yu. Taking human out of learning applications: A survey on automated machine learning. *CoRR*, abs/1810.13306, 2018.

[8] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

[9] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization*, pages 507–523. Springer, 2011.

[10] Lars Kotthoff, Chris Thornton, Holger H Hoos, Frank Hutter, and Kevin Leyton-Brown. Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka. *The Journal of Machine Learning Research*, 18(1):826–830, 2017.

[11] Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. Practical automated machine learning for the automl challenge 2018. In *International Workshop on Automatic Machine Learning at ICML*, pages 1189–1232, 2018.

[12] Richard Loree Anderson. Recent advances in finding best operating conditions. *Journal of the American Statistical Association*, 48(264):789–798, 1953.

[13] Lior Friedman and Shaul Markovitch. Recursive feature generation for knowledge-based learning. *CoRR*, abs/1802.00050, 2018.

[14] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. Language, Speech, and Communication. MIT Press, Cambridge, MA, 1998.

[15] Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. Open information extraction from the web. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 2670–2676, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[16] Philippe Esling and Carlos Agon. Time-series data mining. *ACM Computing Surveys (CSUR)*, 45(1):12, 2012.

[17] Robin John Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, Australia, 2nd edition, 2018.

[18] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754, 2016.

[19] Andrej Karpathy, Justin Johnson, and Fei-Fei Li. Visualizing and understanding recurrent networks. *CoRR*, abs/1506.02078, 2015.

[20] Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[22] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W Kempa-Liehr. Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package). *Neurocomputing*, 307:72–77, 2018.

[23] D Ignatov, Pavel Spesivtsev, and Vladimir Zyuzin. Multilabel classification for inflow profile monitoring. *MACSPro'2019, Vienna*, pages 21–23, 2019.

[24] Ivan Lazarevich, Ilya Prokin, and Boris Gutkin. Neural activity classification with machine learning models trained on interspike interval series data. *arXiv preprint arXiv:1810.03855*, 2018.

[25] Marc-André Zöller and Marco F. Huber. Survey on automated machine learning. *CoRR*, abs/1904.12054, 2019.