Meta-learning for time series forecasting in application to ATM load time series

Altair Toleugazinov [1] Aibek Akhmetkazy [1] Pavel Muravskii [1] Artem Erkhov [1] Batyr Khabibullin [1]

Abstract

Forecasting time series data is a common challenge in practical scenarios, requiring significant resources and expertise to select the best model for large datasets. AutoML techniques offer potential solutions, with meta-learning being a promising strategy for model selection. The goal is to develop a meta-learning approach that surpasses brute-force methods, improving computational efficiency without compromising prediction accuracy. This problem will be addressed by extracting features from time series data and implementing meta-learning techniques on specific ATM load datasets.

1. Introduction

At the moment, Automatic Teller Machines (ATMs) are quite common devices that help people withdraw cash or conduct financial transactions. For banks, it is important to predict the amount of money that will be in the ATM because accumulating too much cash leads to potential loss of profit from investments, while having too little can result in dissatisfied customers. Therefore, this task boils down to predicting ATM time series data.

One method for model selection for forecasting involves comparing models on time series data using mean squared error (MSE). However, this approach demands considerable computational time due to the need for testing numerous models on multiple time series. The computation time of this method increases proportionally with the number of time series multiplied by the number of models to be tested.

Currently, a method known as meta-classifiers is being developed, utilizing meta-learning to select the most suitable time series forecasting method. This approach employs a variety of forecasting algorithms, and based on time series features and forecasting errors, meta-learning models determine which algorithm would be most effective for a specific dataset. This mitigates the issue of selecting a single universal model and enhances flexibility in choosing an appropriate forecasting method.

The meta-classifiers method comprises two parts: an "offline" component, involving only the computation of time series features and the use of a pre-trained classifier to determine the best forecasting model. Thus, generating forecasts involves evaluating only one forecasting model. This approach is known as FFORMS (Feature-based FORecast-Model Selection). However, methods based on this approach often fall short as it's not adequately understood how meta-learners make decisions and what happens within these complex model structures.

Therefore, our focus will be on creating features that would initially aid the meta-classifier in better understanding which class of models to consider and subsequently, which model to choose for predicting the time series. These features should be tailored specifically to the classes and models they encompass so that, given the type of input series, the meta-classifier can determine which model is best suited for the given time series type.

2. Literature review

2.1. Time series features

Reid (1972) and other researchers have highlighted that the effectiveness of forecasting methods can vary depending on the specific characteristics of the time series data being analyzed. Understanding these variations can aid in selecting the most suitable forecasting model.

A time series feature is any measurable characteristic of the time series. Examples of time series features include autocorrelation, spectral entropy and measures of self-similarity and nonlinearity. Fulcher and Jones (2014) identified 9000 operations to extract features from time series.

Different studies have identified thousands of features that can be extracted from time series data, but selecting a smaller set of features that demonstrate strong predictive power and minimal redundancy is crucial for efficient analysis. Lubba and Sethi (2019) introduced a method to identify a concise set of 22 Canonical Time-series Characteristics

[1]Skolkovo Institute of Science and Technology, Moscow, Russia. Correspondence to: Altair Toleugazinov <toleugazinov.a@phystech.edu>.

Final Projects of the Machine Learning 2024 Course, Skoltech, Moscow, Russian Federation, 2024.

(catch22) from a vast library of features, resulting in significant computational efficiency without sacrificing much in terms of classification accuracy. This approach offers a more streamlined and practical way to analyze time series data for forecasting purposes.

Therefore, In order to select the most appropriate features of a time series, it is necessary to evaluate not only its nature, but also the purpose for which it is analyzed. But another question is how to select proper model for forecasting using selected features. One of the most common techniques is to use meta-classifier.

2.2. Meta-learning for algorithm selection

John Rice was an early advocate for the concept of meta-learning, known as the algorithm selection problem (ASP) (1976). The term meta-learning started gaining traction with the rise of machine-learning literature. Rice's framework for ASP includes four key components: the problem space (P) represents the data sets, the feature space (F) includes measures characterizing the problem space, the algorithm space (A) lists suitable candidate algorithms, and the performance metric (Y) measures algorithm performance. The key challenge in ASP is identifying the mapping from the feature space to the algorithm space. Smith-Miles provided a formal definition of the algorithm selection problem in (2009).

Formal definition for ASP: for a given problem instance $x \in P$, with features $f(x) \in F$, find the selection mapping $S(f(x))$ into algorithm space A, such that the selected algorithm $\alpha \in A$ maximizes the performance mapping $y(\alpha(x)) \in Y$.

There are some existing methods for forecast-model selection problem. They differ with respect to the way they define the problem space (A), the features (F), the forecasting accuracy measure (Y) and the selection mapping (S).

2.3. Forecast-model selection using meta-learning

Prudêncio and Ludermir (2004) were the first to use the term ”meta-learning” in the context of time series model selection. They investigated how meta-learning approaches could be applied for forecast model selection using specific features and algorithms.

The existing meta-learning algorithms for forecasting are divided into two categories: (1) selecting the “best forecast model” (the model with the smallest forecast error) out of a pool of forecast models, and (2) identifying suitable weights to combine forecasts from all available models in the pool. Also meta-learning algorithms can be further classified into two extremes in terms of forecasting accuracy and computational cost: (1) low computational cost and low accuracy, and (2) high computational cost and high accuracy.

Example of selecting the “best forecast model” can be presented by work of Talkhi, Fatemi and Nooghabi (2024). They have developed a meta-model that recommend an appropriate time-series forecasting model with an accuracy of 82,5%. In their study, they used 30 hand-selected meta-features, ARIMA and TBATS as forecasting models. Among decision tree (DT), support vector machines (SVM), artificial neural networks (ANN), and random forest (RF), the decision tree showed the best result as a meta-classifier.

Example of combining the forecasts can be presented by work of Talagala, Li and Kang (2022). They have developed a new meta-learning algorithm that predicts the performance of time series forecast models. By considering various time series features derived from historical data, they use a Bayesian regression method to estimate forecast errors. This allows the forecast models to be ranked with respect to their forecast errors and the evaluation of their relative forecast performance without calculating forecasts from all available individual models in the pool. The algorithm then selects the model or combination of models that are likely to produce the most accurate forecasts based on the minimum predicted error.

There were also attempts to understand how the meta-learners make their decisions and what is really happening inside these complex model structures. Talagala, Hyndman and Athanasopoulos (2023) made a first step towards providing a comprehensive analysis of the relationship between time series features and forecast model selection using machine learning interpretability techniques.

2.4. Conclusion of literature review

As we can see, most researchers, when choosing features, did not take into account the set of the forecasting models, using the same features for different forecasting models and metaclassifiers. Also, It should be noted it is difficult to gain a good balance between accuracy and computational cost.

3. Project Plan

Initially, we will study existing models and classes of models, and then determine the direction in which the features should be developed to allow the model to be selected most optimally. Brief overview of models: The two most frequently mentioned models are ARIMA and ETS. Therefore, we will start by considering them.

SARIMA (Seasonal Autoregressive Integrated Moving Average): This is a statistical model that uses autoregression, integration, and moving averages to model time series data. ARIMA assumes that a time series has certain statistical properties, such as stationarity and autocorrelation. It considers a set of standard structures in time series data and provides a simple and powerful method for forecasting (2001).

ETS (Exponential Smoothing State Space Model): The idea of exponential smoothing is that forecasts made using it are the weighted average of past observations, with the weight decreasing exponentially over time. This model can account for seasonality and trend. It includes various variants, such as ETS (M, A, N), where M, A, and N denote the type of seasonality, trend, and residual variance, respectively. There are also extensions of this model, such as Damped Local Trend (DLT) and Holt-Winters.

Prophet: This model is used for processing data with trends, seasonality, and outliers. It is particularly well-suited for forecasting data that exhibit nonlinear trends, seasonal patterns, and regular outliers. Prophet uses deep learning for forecasting. There is also its next version called NeuralProphet, which utilizes even deeper learning.

DeepAR: It is a deep recurrent neural network that uses autoregression for time series forecasting. It is suitable for analyzing time series with complex dependencies and is capable of incorporating external variables.

In addition, there are Boosting models from classical machine learning, such as AdaBoost, Gradient Boosting, and XGBoost, which are used on more complex data to describe complex dependencies.

Comparison of the described models and discussion about features:

ARIMA assumes data stationarity and can account for seasonality, but selecting the optimal model can be challenging (2017). ETS can handle seasonality without differencing as well; however, optimal models need to be selected within ETS since models may include parameters such as level, trend, and seasonality. Therefore, the features we select should reflect the patterns of the models, based on which the meta-classifier would choose the optimal model. In addition, in our set there are models which use deep learning, and which are respectively applied when it comes to complex time series, but the problem of this model is that in this case the time series should not contain gaps and have enough data. Hence, it follows that fiches checking the time series on sparse are also needed.

To begin with, we will analyze existing feature libraries. One of the most popular libraries containing features is **Catch 22** (In addition to this library we will use **FreshPRINCE, TSFresh, Signatures** and this libraries will also be analysed). Analysis of the features contained in it has shown that there are features that would be suitable for our purposes. examples include:

- `DN_OutlierInclude_p_001_mdrmd` and `DN_OutlierInclude_n_001_mdrmd` - These features indicates the presence of outliers in the data, which may be associated with seasonality or other irregular patterns.
- `CO_f1ecac` - This feature is associated with the autocorrelation function (ACF), which can reflect seasonal patterns in the data.

Thus, the necessary features include:

- Statistical characteristics: Mean, median, standard deviation, maximum and minimum values, variation, moving averages, fluctuations, etc.
- Correlation analysis: Autocorrelation function (ACF) and partial autocorrelation function (PACF), which help determine the stationarity of the time series.
- Temporal characteristics: Seasonality periodicity, such as daily, weekly, monthly, or yearly.
- Autoregression function: Autoregression indicators, showing how the current value of the time series depends on its previous values.
- Sparsity function: The number of missing values in the data.
- Model complexity: The number of parameters required to describe the time series.

4. Main Part

4.1. Pipeline

Our dataset initially contains validation data and test data of the penultimate 30 days and the last 30 days, respectively. We divide this dataset into two parts TRAIN and "hold-out", which also contain both valid and train data respectively. Then we create our base line by selecting the model with the lowest mean error (in our case SMAPE), and this model is selected on validation data from the TRAIN part, and then we get the model on validation data of hold-out part, and on test data of the hold-out part. To create a metaclassifier, we first create fixtures for training data of the whole dataset, i.e. both TRAIN part and hold-out part, then we select fixtures and train the model on validation data from TRAIN part. Passing validation on hold-out validation data. And at the very end we already validate our model on the TRAIN part (Figure 1).

4.2. Description of our data

Our data are time series of ATMs and based on this we can identify some specific features. Financial companies believe that the demand for ATM use is driven by such factors as:

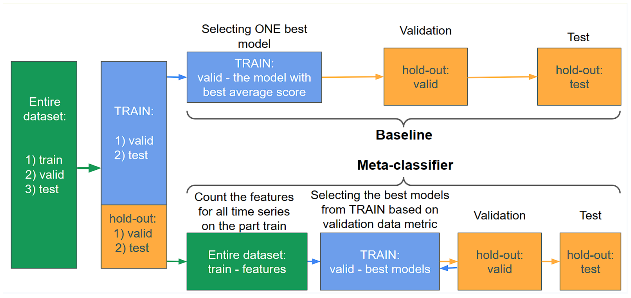

Figure 1. Pipline

- ATM location,
- Seasonal factors,
- Historical patterns from users.

From this, we can conclude that we will need seasonal features for further analysis, as well as studying previous steps for prediction. Unfortunately, ATM locations are missing in our data.

Our dataset was initially represented by 871 time series divided into parts: validation, test, training and also with corresponding models applied on this series and a list of estimates corresponding to this series ['MAE', 'MSE', 'RMSE', 'MASE', 'RMSSE', 'MAPE', 'SMAPE'], besides these columns there were also columns responsible for training time and forecast time. The total number of models used in the dataset is 24, not all datasets were run with the same number of models. In addition, the original dataset contained completely rows with NaN which corresponded to certain time series.

4.3. Preprocessing and Base line creation

First of all, we deleted two time series nn5_111 nn5_112 on which only 3 models were trained, and also deleted the rows containing NaN. In addition, we deleted the rows related to TFTTuningObjective, DeepARTuningObjecyive models, because the data for these models were obtained for about only 1/7 of all time series.

SMAPE was chosen as the further metric characterising the model, as this metric takes into account symmetric deviations and is easy to interpret and convenient for comparison between models, it is a safer metric to use when there is a lot of sparsity in the data.

$$SMAPE = \frac{100\%}{n}\sum_{t=1}^{n}\frac{|F_t - A_t|}{(|F_t| + |A_t|)/2}$$

- n - number of observations,
- A_t - actual value,
- F_t - forecast value.

As a result, all other columns with metrics were not taken into account.

After selecting the metrics, the dataset was ranked by models with the lowest average SMAPE. The best model according to this selection principle was LightAutoML with an average SMAPE - 20.268%. In addition to this model, the model that most often scored the lowest SMAPE on the time series was TFTTuningObjective_gl. This model became the best 159 times, but the average score of this model was 28.582%. Prophet was also selected as a comparison model as the most popular model and its average score was 20.716%.

4.4. Feature Extraction

4.4.1. TSFRESH FEATURE EXTRACTION

Overview of tsfresh: tsfresh facilitates the extraction of an extensive array of statistical features from time series data. It is capable of calculating a multitude of characteristics, from fundamental statistics like mean and standard deviation to more intricate ones such as Fourier coefficients and time reversal asymmetry statistics.

Features in tsfresh: The library focuses on statistical features, which are extensive and diverse, including:

- Basic statistics (mean, median, standard deviation)
- Time series properties (autocorrelation, cointegration)
- Non-linear dynamics (entropy measures, longest strike above mean)

The feature selection process is vital due to the sheer volume of features, aiming to retain only those with significant predictive power for forecasting models like *CatBoostAutoRegressivePipelineEtna* and *LightAutoML*.

4.4.2. CATCH22 FEATURE EXTRACTION

How catch22 Selects Features: catch22's feature set is the result of a comprehensive analysis that evaluated over 7000 features. The process focused on identifying features that display varied behavior across different datasets and selecting the top performers based on classification efficacy, ensuring that each feature provides unique and informative insights.

4.4.3. CUSTOM FEATURE DEVELOPMENT

Our Custom Features and Their Relation to Models: We developed custom features that complement the strengths of our forecasting models, including:

- **Temporal Dynamics:** Features like `trend_slope` and `seasonality_strength` inform models sensitive to trends and seasonal changes such as *Prophet*.
- **Spectral Content:** Features like `dominant_freq` aid models in identifying periodicities in the frequency domain.
- **Volatility and Change:** Features such as `num_peaks` and `mean_peak_height` are beneficial for models that detect volatility.
- **Complexity Measures:** `signal_entropy` and `mean_curvature` quantify unpredictability and non-linear structure, which are key for complex temporal dependency models.

This tailored feature set should enhance our meta-classifier's performance, aligning with the specific characteristics of the ATM load time series data and the nuanced behaviors our models are designed to predict.

Conclusion of Feature Extraction: By combining the comprehensive feature base of tsfresh with the interpretability of catch22, and supplementing with our custom-tailored features, we achieved a feature set that should significantly enhance the meta-classifier's performance. Our approach highlights the importance of a profound understanding of both the models and the data involved in the analysis.

4.5. Feature selection

After creating a large pool of features, it was necessary to select only those that would give the least SMAPE error for classification of the best forecasting model. First attempt was to obtain feature importances of all created features. To issue this task the random forest model was used. After that we wanted to understand how many best features should we consider. For this purpose we trained several random forest models, but this time we used only fixed number of best features. For better precision cross-validation technique was used. Due to high variance of the obtained score, each classifier was trained at the same splits, so this allowed as to see difference between number of features without negative affect of the randomness of the splits.

Figure 2 shows how SMAPE score depends on the number of the best features used. The greater the number of features used, the worse the score becomes. the beast result was obtained for 10 features (SMAPE = 21.67). The mian problem of this approach is that when so few features are used, there is a high probability that such a result is obtained only for the dataset used for training.

Another approach was to determine the best features for each feature extraction method separately and then to find which combination of this feature pools gives the best results.

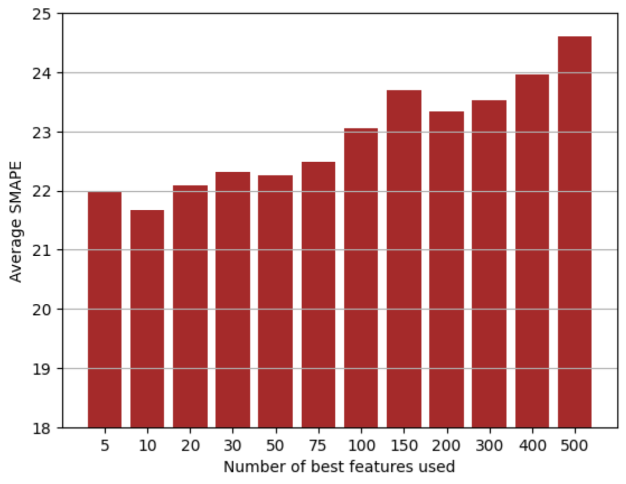

Figure 2. Average SMAPE with different number of best features

Firstly, we performed forward search to our custom features. To do this, we used the random forest model again. Every iteration classifier calculated the average SMAPE adding one feature from list of unused and then decided which feature gave the best result. After several iteration only 2 features were collected for further usage:

- `num_peaks`,
- `signal_entropy`.

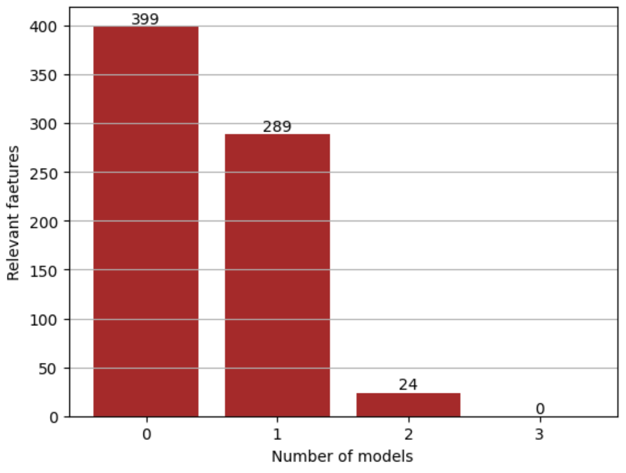

Figure 3. Number of features that are relevant for prediction of certain number of best models

Secondly, we used build-in tsrfesh function `feature_selection` to find features that are relevant for prediction of the best model. Figure 3 shows that for prediction of more than two model 24 features are

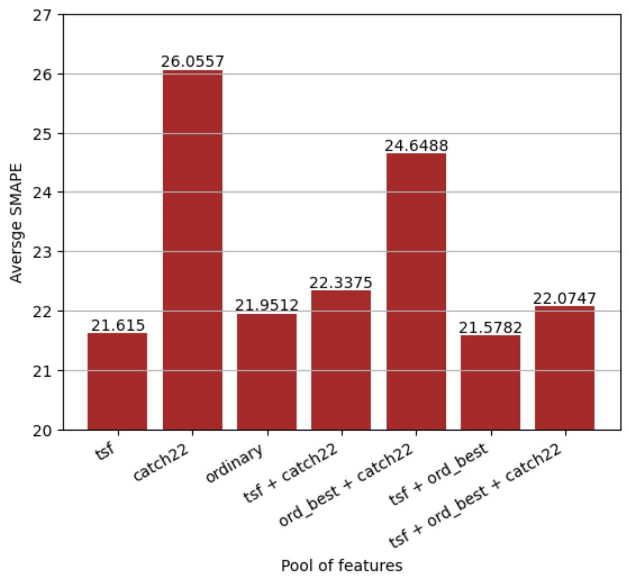

Figure 4. Average SMAPE in relation to certain feature pool

relevant, therefore this is optimal choice due to trade-off between number of features and number of models that these features are relevant for predicting.

The last step was to find the most effective pool of features. Trfesh, catch22, self-made and combinations of them have been considered. Figure 4 shows which score was obtained with different feature pools. The best result was received with combination of tsfresh fesatures and best of our custom features. The final score (SMAPE = 21.58) is slightly better than the score with top 10 features obtained by random forest feature importance.

4.6. Meta-model selection

4.6.1. Random Forest Classifier

The task of the classification algorithm is to identify the "best" forecasting method for a given time series. To train the meta-classifier, for each time series in train set we found the model with the lowest forecast error measure over the validation period and deemed it as "best".

Essentially, any classification algorithm could be used as meta-learner. We have chosen a random forest algorithm for several reasons: i) it can effectively model complex feature interactions; ii) it can capture both linear and non-linear effects through the use of many decision trees; iii) it is resistant to overfitting; iv) it can handle imbalanced classes well; v) it is faster than certain boosting algorithms; vi) it is simple to implement and work with using existing software.

Our aim is different from most classification problems in that we are not interested in accurately predicting the class,

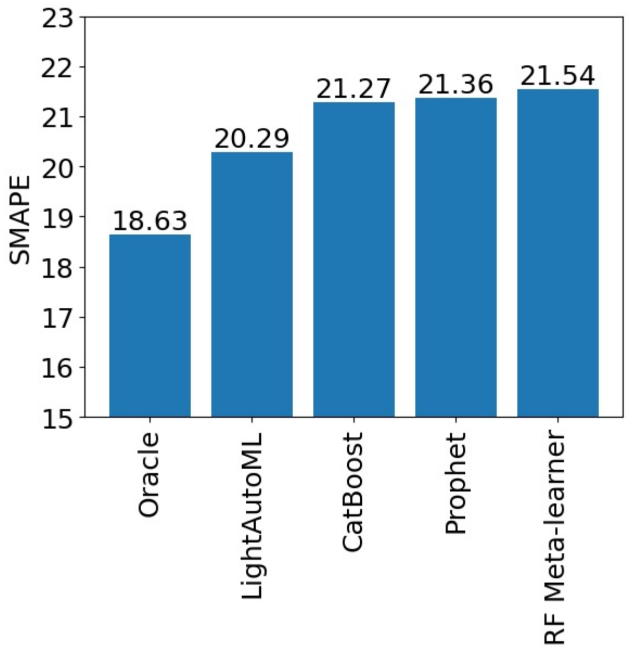

Figure 5. Mean SMAPE on test set by best forecasting models and RF meta-learner

but in finding the best possible forecast model. It is possible that two models produce almost equally accurate forecasts, and therefore it is not important whether the classifier picks one model over the other. Therefore, we report the forecast accuracy obtained from the models, recommended by meta-classifier, rather than the classification accuracy. As a baseline, the results of the best forecasting models are presented, as well as the result of the Oracle model implementing the brute force method. (Figure 5).

Figure 5 shows, that the meta-learner loses to the best forecasting models. To explain this result, we output a list of meta-model predictions (Table 1).

It can be seen that the meta-classifier most often recommends the TFTTuningObjective-gl model (only 13-th

Model name	Count
TFTTuningObjective-gl	168
Prophet	40
CatBoostAutoRegressivePipelineEtna-horizon	23
LightAutoML-mult-gl	19
LightAutoML	8
CatBoostDirectPipelineEtna-horizonlags-gl	2
CatBoostDirectPipelineEtna-horizonlags	1
NeuralProphet-gl	1

Table 1. Forecasting models recommended by Random Forest meta-learner

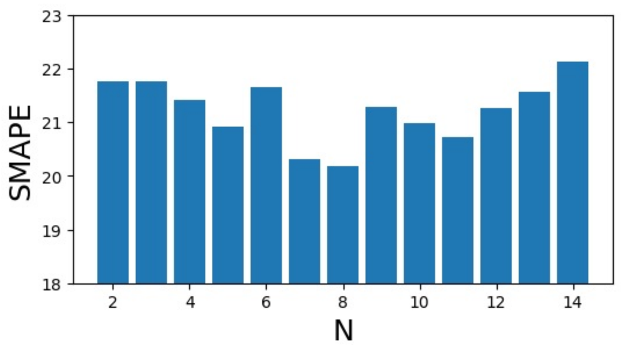

Figure 6. Dependence of average SMAPE on parameter N

place in average SMAPE), ignoring the forecasting models with the better average SMAPE. This is because the classifier minimizes some classification error metric, rather than the forecasting error as needed. This leads to high cost of misclassification - when meta-learner recommends TFTTuningObjective-gl model for an unsuitable time series, this immediately causes a significant increase in the average SMAPE. It would be more logical to choose not exactly best model, but the one that more often gets into top-N models. This is not possible in the current meta-learner configuration, because when assigning classifier labels, only the top 1 model for each time series is taken into account. This leads to poor generalization ability of the meta-learner, because it ignores a large part of the dataset.

4.6.2. OneVSRest Classifier

In order to increase the generalization ability of the meta-learner and implement the improvements described above, we applied OneVSRest approach.

Model name	Count
LightAutoML-mult-gl	69
TFTTuningObjective-gl	68
LightAutoML	67
CatBoostAutoRegressivePipelineEtna-30lags	22
CatBoostAutoRegressivePipelineEtna-horizonlags	10
Prophet	10
CatBoostDirectPipeline-Etna-horizonlags-gl	5
NeuralProphet-gl	4
CatBoostDirectPipeline-Etna-30lags-gl	4
CatBoostAutoRegressivePipeline-Etna-horizonlags-gl	2
CatBoostAutoRegressivePipeline-Etna-30lags-gl	1

Table 2. Forecasting models recommended by OneVSRest meta-learner

With this approach, a separate classifier is trained for each forecasting model, which predicts the probability that the model is suitable for this time series. After all the classifiers have calculated the probabilities, the model with the highest probability is selected. When training classifiers, the following principle is used - a model is considered suitable for a given time series if it is included in the Top-N forecasting models by SMAPE. The optimal value of the hyper-parameter N has been found with grid search (Figure 6).

In comparison with the Random Forest meta-learner, the classes predicted by OneVSRest are more balanced, and the total number of recommended models is larger (Table 2). This suggests that the meta-learner takes into account the characteristics of time series and has a high generalization ability.

4.7. Obtained results

Figure 7 shows obtained results. On the left histogram there is comparison in average SMAPE for different pure model, Oracle and our meta-classifier (OneVsRest). Our model outperform all of the pure models, this means that if we want to find the best model to forecast time series, our meta-classifier would be a good choice, because it gave the better result than just using one certain model. Another comparison is about time consumption (right histogram). Meta-classifier predicted model whose forecasting times less than if using only LightAutoML model. This means that our the experiment has a positive result, because using meta-classifier is better choice than using of one pure model or just find the best model by brute-force.

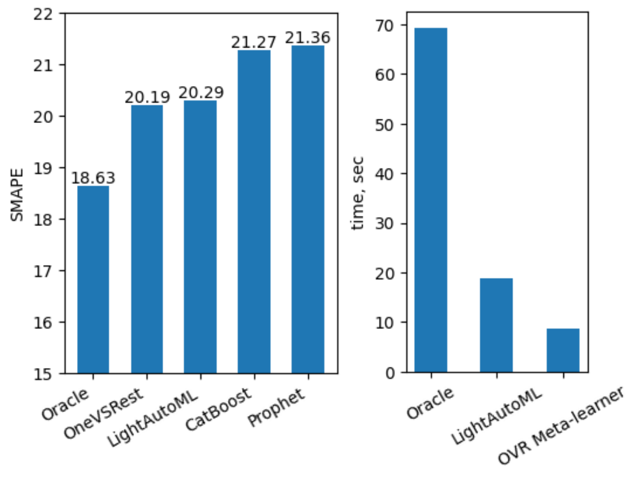

Figure 7. Average SMAPE score and time consumption of the best meta-classifier

5. Conclusion

Our exploration into feature extraction for time series forecasting has led us to a nuanced understanding of the strengths and weaknesses of various methods. While **tsfresh** offered an extensive array of statistical features, the abundance of these features required us to conduct a meticulous feature selection process to avoid the pitfall of including non-informative or redundant features that could potentially degrade the performance of our forecasting models.

On the other hand, **catch22** promised a simplified and more interpretable set of features. However, it fell short of delivering the performance we needed, highlighting the importance of not just the quantity but the quality and relevance of the selected features to the task at hand.

Our journey culminated in the development of **our custom features**, which were meticulously crafted to resonate with the specific models used in our dataset. This approach paid off, as these features were not only interpretable but also finely tuned to the particularities of our forecasting tasks, leading to a marked improvement in performance.

In summary, the greatest contribution to the meta-classifier's success came from our own features complemented by selected tsfresh features. We managed to outperform the best individual model in terms of average SMAPE by a slight margin of 0.1% and substantially by 8.4% compared to the most frequently best-performing model. Our findings underscore the potential of meta-classifiers as an excellent choice for situations where time is of the essence, and a more accurate forecasting result is required than what could be achieved by randomly selecting a model. This project has not only advanced our understanding of feature importance in time series forecasting but also has set a precedent for the thoughtful integration of machine learning techniques in the field of financial forecasting. (Table 2.)

6. Aknowledgment

https://github.com/altairje/Meta-Algorithm-ATM

References

Eamonn Keogh, Kaushik Chakrabarti, M. P. and Mehrotra., S. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and information Systems*, 3(3):263–286, 2001.

Fulcher, B. D. and Jones, N. S. Highly comparative feature-based time-series classification. *IEEE Transactions on Knowledge and Data Engineering*, 26(12):3026–3037, 2014.

Jain, G. and Singh, S. A study of time series models arima and ets. *SSRN Electronic Journal*, 01 2017. doi: 10.2139/ssrn.2898968.

Lubba, C. H., Sethi, S. S., Knaute, P., Schultz, S. R., Fulcher, B. D., and Jones, N. S. Canonical time-series characteristic. *arXiv:1901.10200*, 2019.

Prudêncio, R. B. C. and Ludermir, T. B. Meta-learning approaches to selecting time series models. *Neurocomputing*, 61:121–137, 2004.

Reid, D. J. A comparison of forecasting techniques on economic time series. *Forecasting in Action. Operational Research Society and the Society for Long Range Planning*, 1972.

Rice, J. R. The algorithm selection problem. *Advances in Computers*, 15:65–118, 1976.

Smith-Miles, K. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys (CSUR)*, 41(1):6, 2009.

Talagala, T. S., Li, F., and Kang, Y. Fformpp: Feature-based forecast model performance prediction. *International Journal of Forecasting*, 38(3):920–943, 2022.

Talagala, T. S., Hyndman, R. J., and Athanasopoulos, G. Meta-learning how to forecast time series. *J Forecasting*, 42(6):1476–1501, 2023.

Talkhi, N., Akhavan Fatemi, N., Jabbari Nooghabi, M., Soltani, E., and Jabbari Nooghabi, A. Using meta-learning to recommend an appropriate time-series forecasting model. *BMC Public Health*, 24(1):148, 2024.

A. Team member's contributions

Explicitly stated contributions of each team member to the final project.

Altair Toleugazinov (20% of work)

- Reviewing literate on the topic (10 papers)
- Preparing the GitHub Repo
- Coding the meta algorithm
- Responsible for features and their coordination

Aibek Akhmetkazy (20% of work)

- Data preprocessing
- Reviewing literate on the topic (10 papers)
- Preparing the GitHub Repo
- Lord of GitHub, performance analytics

Pavel Muravskii (20% of work)

- Reviewing literate on the topic (10 papers)
- Analyzing the feature importance based on literature
- Literature review, reports, coordination, strategy and attempts that ended in disappointment

Artem Erkhov (20% of work)

- Reviewing literate on the topic (10 papers)
- Preparing the Section 2 of this report
- Meta-learner selection and training

Batyr Khabibullin (20% of work)

- Reviewing literate on the topic (10 papers)
- Preparing the Section 2 of this report
- Feature selection, testing different feature pools