National Research University Higher School of Economics

Faculty of Computer Science

Programme 'Master of Data Science'

MASTER'S THESIS

Electricity Spot Prices Forecasting Using Stochastic Volatility Models

Student Batyrov Andrei Renatovich

Supervisor Kasianova Ksenia Alekseevna

Moscow, 2024

ABSTRACT

There are several approaches to modeling and forecasting time series as applied to prices of commodities and financial assets. One of the approaches is to model the price as a non-stationary time series process with heteroscedastic volatility (variance of price).

The goal of the research is to generate probabilistic forecasts of day-ahead electricity prices in a spot marker employing stochastic volatility models. A typical stochastic volatility model – that treats the volatility as a latent stochastic process in discrete time – is explored first. Then the research focuses on enriching the baseline model by introducing several exogenous regressors.

A better fitting model – as compared to the baseline model – is derived as a result of the research. Out-of-sample forecasts confirm the applicability and robustness of the enriched model. This model may be used in financial derivative instruments for hedging the risk associated with electricity trading.

Keywords: Electricity spot prices forecasting, Stochastic volatility, Exogenous regressors, Autoregression, Bayesian inference, Stan

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

1. INTRODUCTION

Today electricity is traded in markets around the world using spot and derivative contracts. Electricity demand depends on individual and business activities, as well as weather and other factors, which may create various seasonality profiles: peak and off-peak hours, working days and holidays, etc.

Electricity is non-storable which makes it a unique commodity and leads to its price having high volatility with possible sudden spikes (shocks), which in turn leads to the need of hedging the risk associated with trading this commodity. This is why predicting the price at least one day ahead is so important.

In the studied scenario, the electricity spot market is divided into two price zones subject to economic and climate factors:

1. European market;
2. Siberian market.

Each market has its own consumer consumption and price profiles. In this research we will study hourly profiles, that is we will build and examine hourly (high frequency) models for:

1. Peak hour;
2. Off-peak hour.

1.1 Approaches to Forecasting

Electricity price forecasting uses mathematical, statistical and machine learning models to predict electricity prices in the future. As of today, several approaches and models have been proposed and tested [1]:

- Multi-agent (multi-agent simulation, equilibrium, game theoretic);
- Fundamental (structural);
- Reduced-form (quantitative, stochastic);
- *Statistical (econometric, technical analysis)*;

- Computational intelligence (artificial intelligence-based, non-parametric, non-linear statistical);
- Hybrid solutions.

Statistical approach includes the following models [1]:

- Similar-day and exponential smoothing;
- Regression (AR, ARMA, etc.);
- *Heteroscedastic (non-constant volatility)*.

Among heteroscedastic models there are:

- (Generalized) Autoregressive Conditional Heteroscedasticity ((G)ARCH);
- Stochastic Volatility (SV).

In (G)ARCH models, the variance of the time series is represented by an autoregressive (ARCH) or autoregressive with moving average (GARCH) process [1, p. 1055], which means that the volatility is deterministic at time t. On the contrary, in SV models, the volatility is modeled as a stochastic, that is random, process [2, p. 361].

A comprehensive comparison of GARCH vs SV models is studied in [3] with Bayesian estimation of model parameters. The authors claim that "... stochastic volatility models almost always outperform their GARCH counterparts, suggesting that stochastic volatility models might provide a better alternative to the more conventional GARCH models."

This research is devoted to building and enriching stochastic volatility models as applied to electricity prices forecasting in a spot market.

1.2 Acronyms

The following is a list of acronyms used in this thesis paper.

ADF Augmented Dickey-Fuller (test)

AR AutoRegressive (model)

CI	Confidence Interval
IDE	Integrated Development Environment
MAE	Mean Absolute Error
ML	Machine Learning
MSE	Mean Squared Error
MWU	Mann-Whitney U (test)
OOP	Object-Oriented Programming
PACF	Partial Autocorrelation Function
PPD	Posterior Predictive Distribution
RV	Random Variable
RMSE	Root Mean Squared Error
SV	Stochastic Volatility
SV X	Stochastic Volatility Exogenous (model)

2. MODEL IMPLEMENTATION

2.1 Bayesian Inference

Model parameters estimation can be a challenging task. By far not always parameters can be expressed in closed-form solutions. One of the approaches to estimate (learn) model parameters in ML is Bayesian inference which treats model parameters as RV, that is each parameter is described by its distribution with some density rather than by a single value [4, p. 139]. The general framework of Bayesian inference is the following:

1. Choose a prior distribution for the parameter $p(\theta)$. Prior distribution is a distribution before the train data is observed. In our study the train data is the price and may also include exogenous regressors, altogether denoted as y here;
2. Given the train data – the distribution of the observed data conditional on its parameters, i.e. $p(y|\theta)$, also known as the likelihood function $\mathrm{L}(\theta|y) = p(y|\theta)$, –
3. Generate the posterior distribution of the parameter with density

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{\int p(y|\theta)p(\theta)d\theta}. \tag{1}$$

4. To generate predictions for new unseen data $\tilde{y}$, compute the PPD with density

$$p(\tilde{y}|y) = \int p(\tilde{y}|\theta)p(\theta|y)d\theta. \tag{2}$$

   Thus, instead of a single point as a prediction, a distribution is generated – same concept as with the model parameter distribution.

2.2 Development Environment

Python [5] was chosen as the programming language to perform all accompanying mathematical computations to support this research. An interactive computations approach was chosen through the use of Jupyter Notebooks [6] which were developed within an IDE Visual Studio Code [7]. The code for all computations was developed,

tested and run on Linux (Ubuntu) in a Docker container [8] running on a Windows machine.

Along with Python – to ensure correctness and robustness of the computations results and to facilitate the speed of development and debugging – industry- and community-proven tools and libraries were chosen and used:

- Stan [9] with Python interface PyStan [10]: Bayesian inference;
- scikit-learn [11]: OOP approach for model building and cross-validation, clustering, metrics;
- Pandas [12]: data manipulation, matrix and vector operations, descriptive statistics;
- NumPy [13]: polynomial fitting, correlation;
- SciPy [14]: hypothesis testing;
- statsmodels [15]: hypothesis testing, PACF;
- Matplotlib [16]: plotting;
- seaborn [17]: plotting.

A Python module was developed for implementing a custom class `StanModel` which inherits from the scikit-learn's `BaseEstimator` and `RegressorMixin` classes. This allows to overload the scikit-learn's standard methods, such as `fit()`, `predict()`, etc. To correctly split the times series data into train-test folds, the `TimeSeriesSplit` class was used. To ensure reproducibility of the results, the random seed was fixed.

All computation code supporting this paper can be found at the author's GitHub repository [18].

2.3 Modeling with Stan

For model parameters estimation we will use the Stan platform for statistical modeling and high-performance statistical computation. It uses its own proprietary probabilistic language and can do Bayesian statistical inference and prediction. Other notable probabilistic programming library is PyMC. The main blocks of a Stan program are the following:

- data: declaration of variables that are read in as external data;
- parameters: declaration of model parameters that will be learned during inference;
- model: model definition and generation of model parameters posterior distributions – fitting the model;
- generated quantities: generation of PPD – predicting for new data with the fitted model.

Our custom class `StanModel` implements two main methods: `fit()` and `predict()`, which are wrappers around PyStan's `sample()` and `fixed_param()` methods and provide scikit-learn-compatible interfaces. The `sample()` method is used for generating model parameters posterior distributions (fit) and `fixed_param()` method is used for generating PPD (predict).

Generating posterior distributions may be time consuming. Taking into account the need to fit and predict many models during building, cross-validation and forecasting, we would like to optimize the running time of Stan code. A possible approach for generating PPD for predictions is to use only the expected values of the estimated parameters, instead of their full posterior distributions. Experiments show that the posterior distributions of model parameters are almost always close to normal (see Figures 4 and 16) which are symmetric and has one mode only. This approach might not be acceptable for severely non-normal distributions with long tails or several modes, though. The density of a now constant parameter is a Dirac delta function shifted by the expected value of the parameter. Thus, instead of 2, the density of PPD will be computed as

$$p(\tilde{y}|y) \approx \int p(\tilde{y}|\theta)\delta(\theta - \mathbb{E}[\theta|y])d\theta. \tag{3}$$

Practically, this means using the mean values of the estimated parameters when drawing samples from PDD with Stan. When examining predictions, we will use CI to compensate for the "loss" of full parameters distributions (estimation uncertainty). Stan example code for a basic SV model can be found at [9], [19]. Complete Stan code for SV Baseline and SV X models developed in this research can be found in the Appendix.

3. SV BASELINE MODEL

There is a common practice in ML tasks to begin the research with the so-called baseline model. The baseline model is used as a foundation for more complex models to build upon. A baseline model can be simple with low variance and/or high bias and not necessarily be a good estimator, thus enabling it to be used as a reference model to estimate the quality and improvement, if any, of all other derived models.

A typical SV model can be described with the following parameters [2, p.361], [9]:

- μ, mean log volatility;
- ϕ, persistence of volatility;
- σ, white noise shock scale;
- h_t, latent log volatility at time t.

The variable ϵ_t represents the white-noise shock (i.e., multiplicative error) on the price at time t, whereas δ_t represents the shock on volatility at time t: $\epsilon_t \sim \mathcal{N}(0,1)$; $\delta_t \sim \mathcal{N}(0,1)$.

Then the price at time t can be described as

$$y_t = e^{h_t/2}\epsilon_t, \tag{4}$$

where $h_t = \mu + \phi(h_{t-1} - \mu) + \delta_t\sigma$; $h_1 \sim \mathcal{N}\left(\mu, \frac{\sigma}{\sqrt{1-\phi^2}}\right)$.

Rearranging the equations above yields the following final model for the price:

$$y_t \sim \mathcal{N}\left(0, e^{h_t/2}\right). \tag{5}$$

The time point t is subject to desired market frequency. In our research we examine hourly profiles distributed over 24 hours (days), that is each point in time represents exactly one consumer price during the whole given hour on a given day.

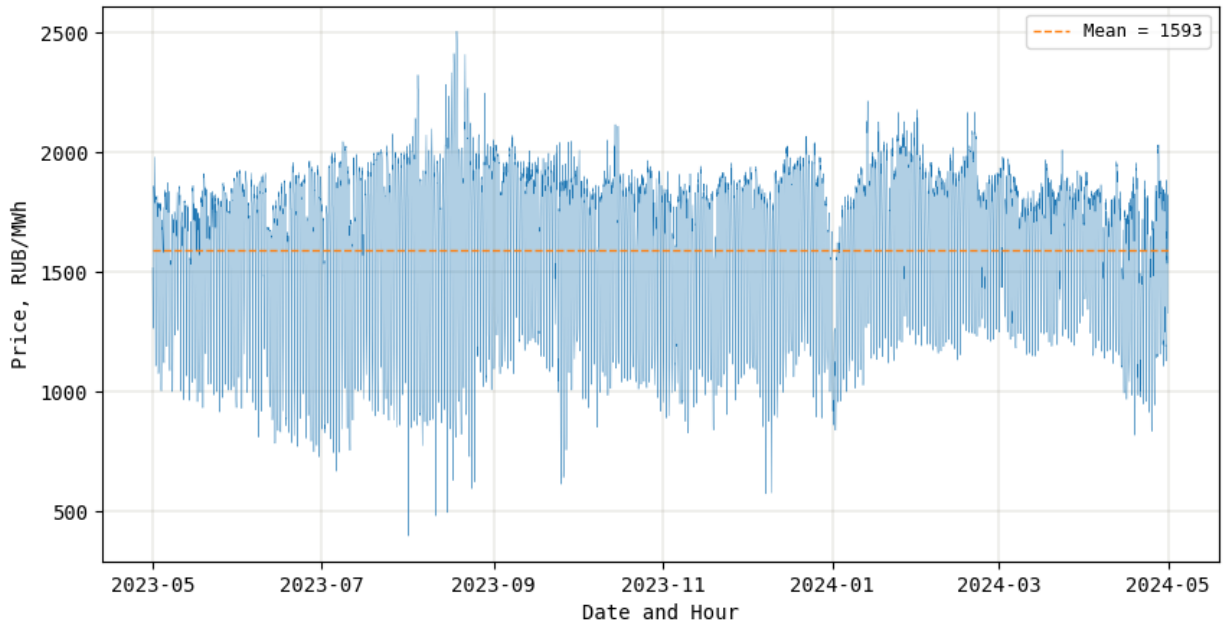

Fig. 1. Consumer prices

3.1 Consumer Prices

As explained in the Introduction, there are two price zones: 1 (European) and 2 (Siberian) in the day-ahead spot electricity markets. We will use the following datasets in our research:

1. For modeling, we will load the hourly data for the price zone 1 for the period of 01.05.2023 – 30.04.2024 (one year back from now) for the peak hour.
2. For model cross-validation, we will load and examine both price zones for both peak and off-peak hours with cross-validation over a sliding window for the period of 23.06.2014 – 30.04.2024 (approx. ten years back from now).
3. For forecasting, we will generate predictions for both price zones for both peak and off-peak hours for the period of 01.05.2024 – 07.05.2024 (one week ahead).

The Daily Indices and Volumes of The Day-ahead Market data is available on the Administrator of Trading System (ATS) [20], starting from Aug 8th, 2013 to present. The unit of prices is RUB/MWh.

A plot of all consumer prices for price zone 1 (modeling dataset) is shown in the Figure 1. The market time frame is one hour.

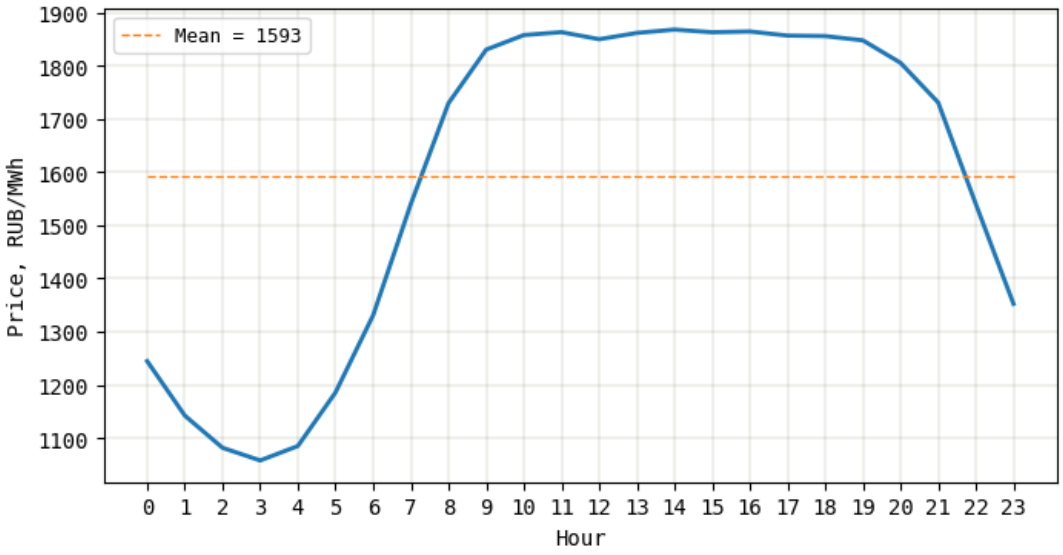

Fig. 2. Consumer prices mean hourly profile

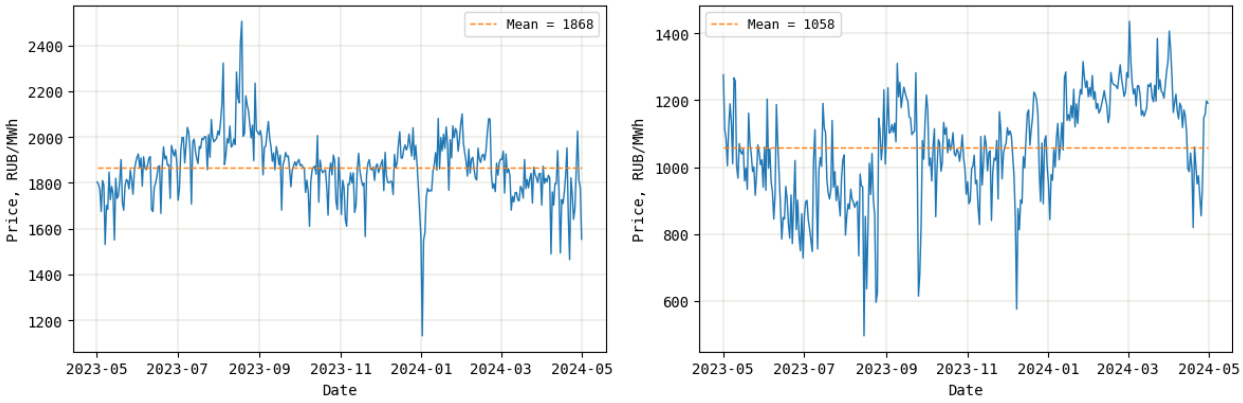

Fig. 3. Trace plots for peak hour #14 (left) and off-peak hour #3 (right)

As explained in the Introduction, we are modeling the price during each hour. The plot of mean prices for each hour is shown in the Figure 2.

The hourly profile, that is the change of the price during the 24 hours (one day) is clearly seen. We will choose one peak hour #14 and one off-peak hour #3. The consumer prices for both peak and off-peak hours over the whole modeling time frame are shown in the Figure 3.

3.2 Test For Stationarity

The plots in the Figure 3 suggest that the price is not a stationary process. We will check this hypothesis using the ADF test [21], [22, p. 2]. This test tests the null hypothesis that there is a unit root in the time series. A linear stochastic process has a

unit root if the process's characteristic equation has a root equal to 1, hence the name. Such a process is non-stationary. The alternative hypothesis is usually stationarity.

$\alpha = 0.05$

$\mathcal{H}_0$: Unit root (Non-stationary)

$\mathcal{H}_1$: Stationary

The results of the ADF tests are shown in the Table 1.

Hour	Statistic	p-value	Result at α
Peak #14	-2.56	0.10	Non-stationary
Off-peak #3	-2.71	0.07	Non-stationary

Table 1. ADF tests results

We cannot reject the null hypothesis at the significance level of 0.05. The consumer price is a non-stationary process, which means that the variance of price (volatility) is an RV with its own moments and can thus be modeled as a stochastic process itself.

3.3 Modeling

The mean of y_t is modeled as 0 in the original model 5, since this model was derived for returns on holding an asset, that is the difference between two consecutive prices, as per the market time frame. In our research we will focus on studying the price itself but not the returns, which means that we have to change the model's mean from 0 to our data real mean, which yields the following final SV Baseline model:

$$y_t \sim \mathcal{N}\left(\bar{y}, e^{h_t/2}\right), \tag{6}$$

where $\bar{y}$ is the mean price seen during estimation (learning) of the model's parameters. The consumer price at time t is a sample drawn from the normal distribution with the mean value being the train price mean (constant) and the variance being the exponent of half of the train log volatility, again at time t (stochastic).

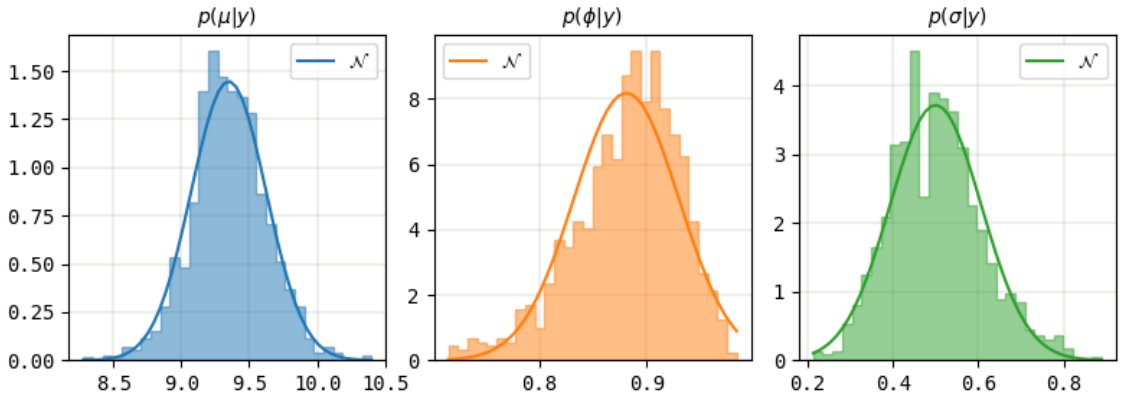

Fig. 4. SV Baseline: Posterior distributions of estimated parameters

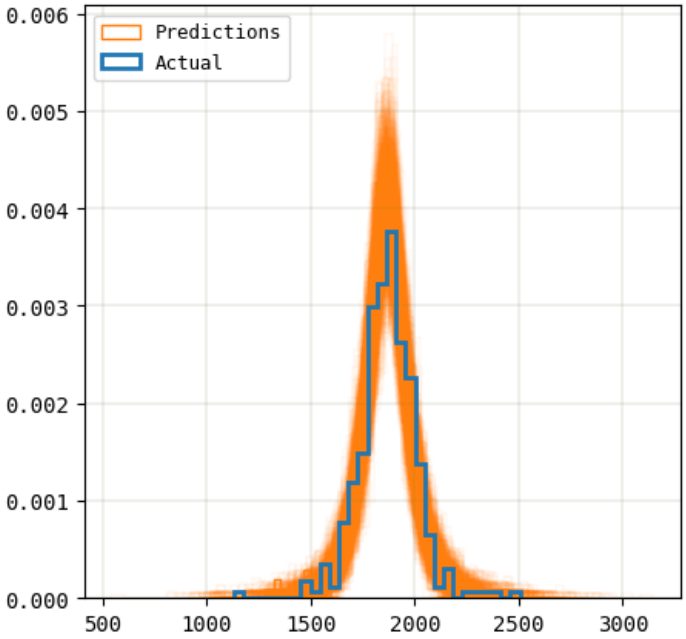

Fig. 5. SV Baseline: 1,000 predicted and actual price distributions

We have shown that the price is a non-stationary process (see 3.2), which means that both the mean and variance are not constant over time. Strictly speaking, we should not assume the constant mean $\bar{y}$ in 6. However, since this is a baseline model, we will keep the assumption of constant mean, yet stochastic volatility at the same time. We will address this issue with the mean later while developing an extended model.

After running the Stan code, we have obtained the posterior distributions of all SV Baseline model's parameters – see Figure 4.

We can now use the estimated model's parameters to compute PDD and generate in-sample predictions for the modeling time frame. 1,000 density plots sampled from

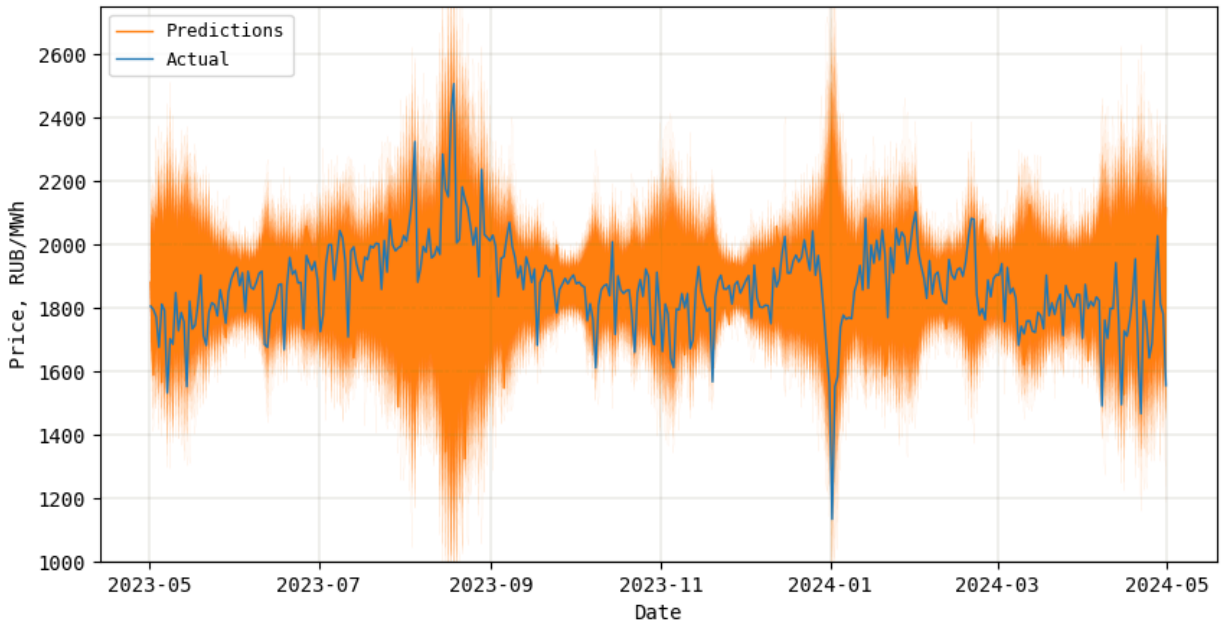

Fig. 6. SV Baseline: Actual price and 1,000 predictions

the PDD of the consumer price are shown in the Figure 5. Observe, that the actual distribution (density) of the consumer prices is well within the predicted density plots, which confirms the correctness of modeling. 1,000 trace plots sampled from the PDD of the consumer prices are shown in the Figure 6. The 95% CI for all 1,000 predictions is shown in the Figure 7.

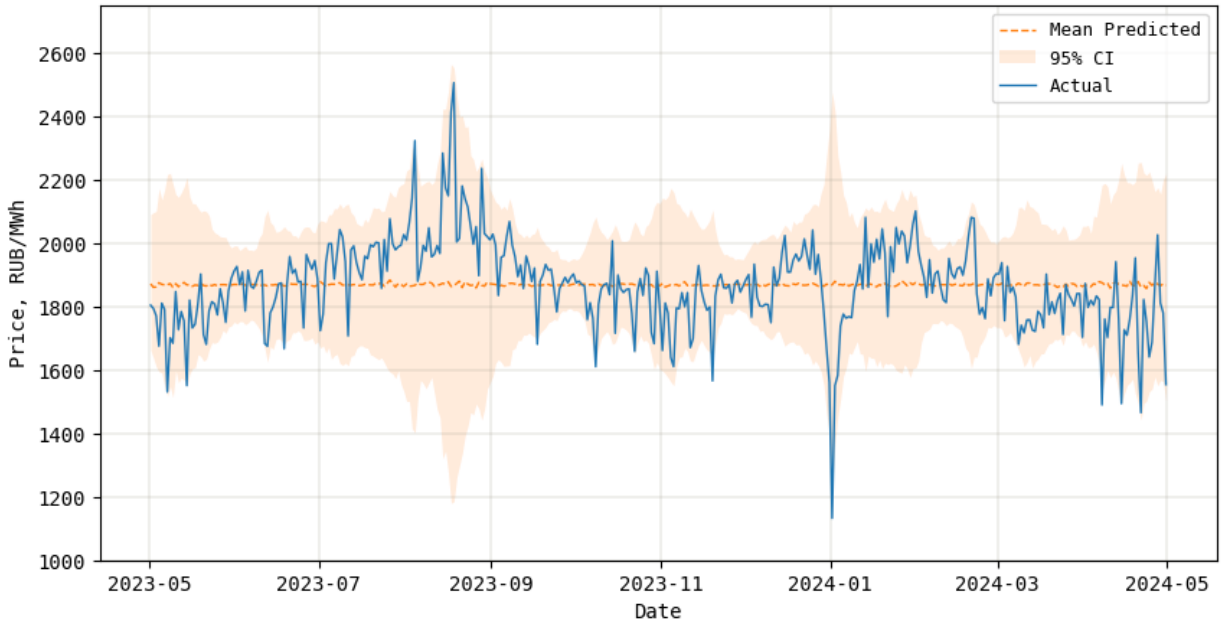

Fig. 7. SV Baseline: 95% CI for 1,000 predictions

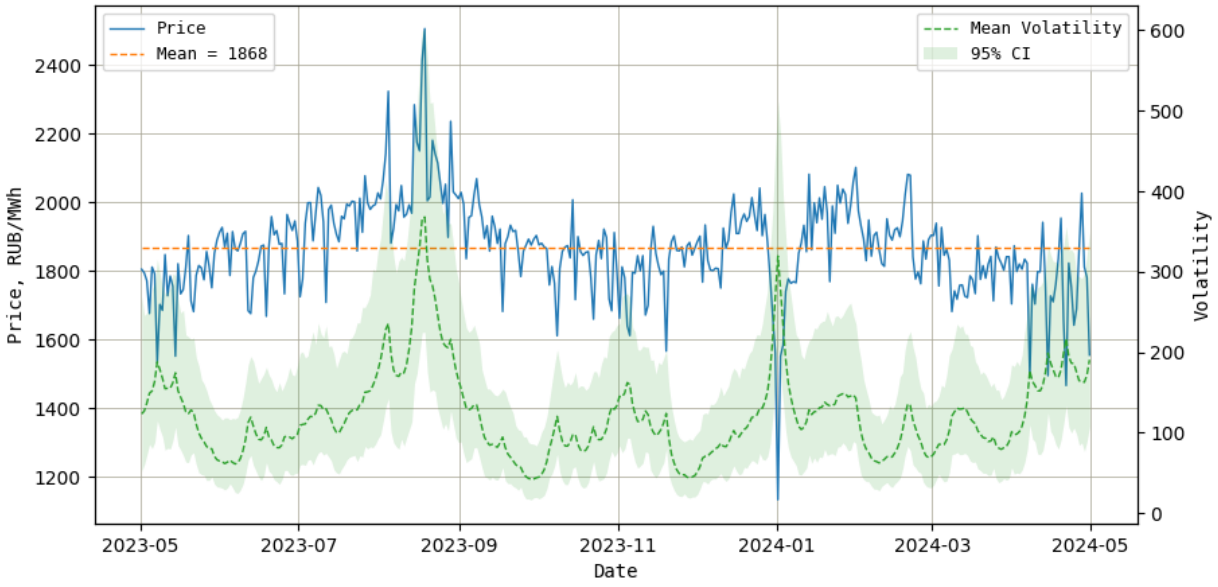

Fig. 8. SV Baseline: Consumer prices and learned volatility

The model has learned the latent stochastic volatility process quite correctly. The learned volatility $e^{h_t/2}$ (both mean and 95% CI) is shown in the Figure 8.

To understand the behavior of the volatility we will visually check the correlation between the price and volatility – see Figure 9. We can clearly see that volatility *does* depend on consumer price, having a rather V-shaped dependency: the volatility tends to increase for the prices higher and lower than the mean price, while it's minimal for the prices around the mean.

3.4 Goodness-of-fit

To evaluate the quality of the model we will compute the following metrics [23, p. 28, 437], [1, p. 1039]:

$$MAE(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|, \tag{7}$$

$$RMSE(y, \hat{y}) = \sqrt{MSE(y, \hat{y})} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}, \tag{8}$$

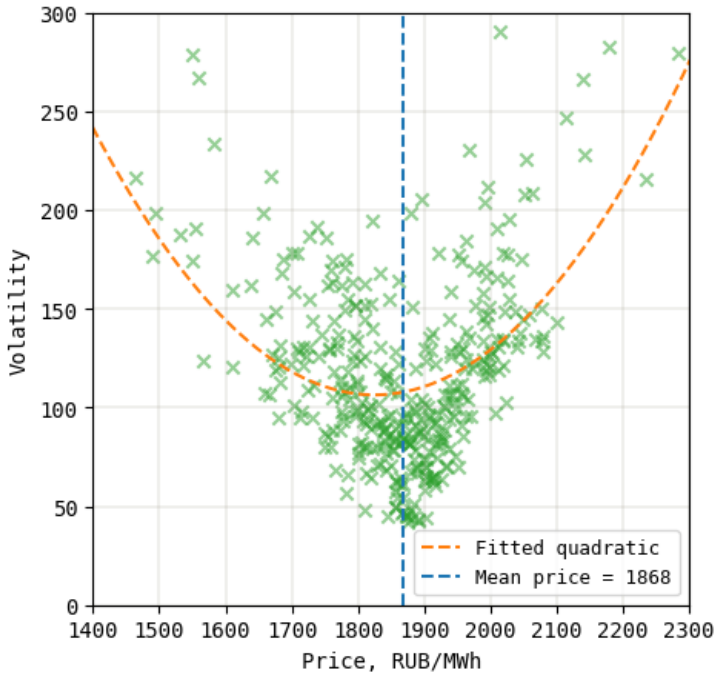

Fig. 9. SV Baseline: Learned volatility vs Consumer price

where n is the number of target points, y_i are the true target values, $\hat{y}_i$ are the model's predictions.

We will apply the metrics 7 and 8 to the consumer price. Note that both MAE and RMSE are point metrics. We are drawing 1,000 samples from PDD which are then averaged for each time t, and these averaged consumer prices are used in the metrics.

The metrics for the in-sample predictions made by the SV Baseline model are shown in the Table 2.

Metric	SV Baseline
MAE	99.18
RMSE	137.04

Table 2. SV Baseline: MAE and RMSE for predicted consumer prices

We also want to check the correlation between the predicted mean price and actual price – see Figure 10.

Though the model was able to learn heteroscedasticity of the volatility, the correlation between the mean predictions and actual prices is rather weak.

We should also check the distribution of residuals, probability plot, and correla-

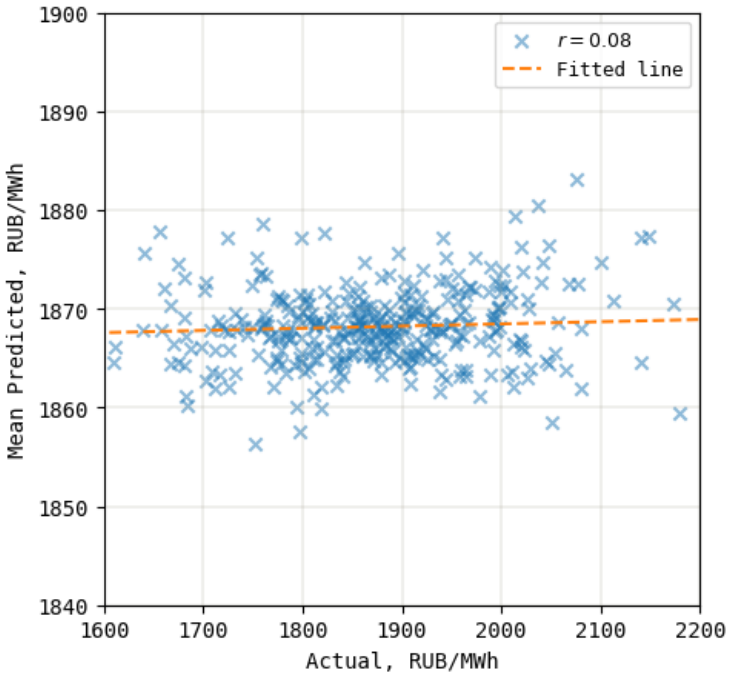

Fig. 10. SV Baseline: Predicted vs Actual price

tion between residuals and mean predicted prices – see Figure 11. Residuals look to be distributed more or less normally with several outliers. Also, residuals are almost not correlated with predictions (homoscedastic).

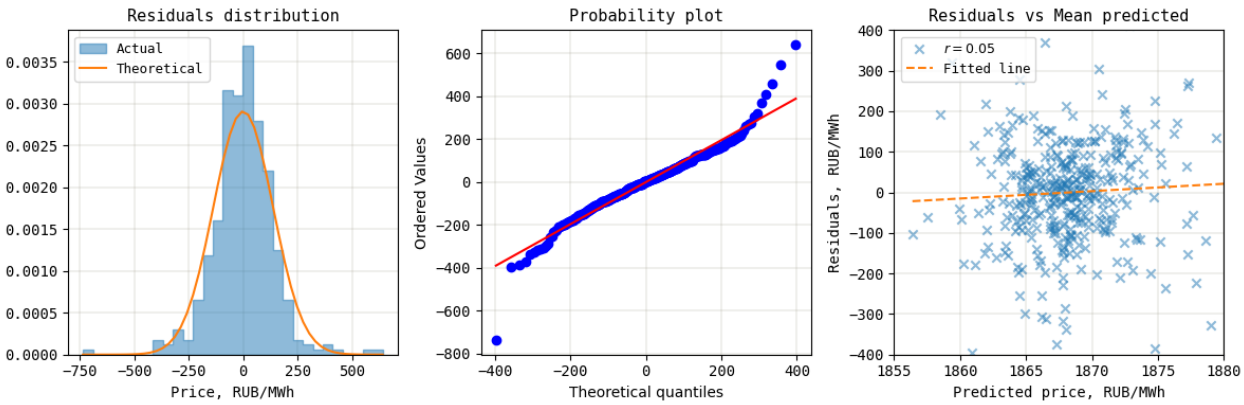

Fig. 11. SV Baseline: Residuals plots

We will *accept* the SV Baseline model.

4. SV EXOGENOUS MODEL

Having built our baseline model, we want to derive a better performing model. One of the approaches is adding independent variables that should help explain the price change over time. We will call such variables as exogenous regressors.

The idea of enriching a model by introducing exogenous factors was studied in [24]. The authors introduce the logarithm of the hourly air temperature at time t, indicator variables for days of the week, and the minimum of the previous day's 24 hourly log prices as the exogenous regressors. One of the latest subject-matter researches confirms the influence of the weather and seasonality factors on the price across price zones [25].

We will now examine similar regressors to understand if they can enrich our model to provide better quality.

4.1 Air Temperature

We will start with the air temperature variable. It may well be expected that demand for electricity, and thus its price, can be dependent on air temperature. Heating is required in cold season (winter) and cooling is required in hot season (summer). The air temperature and other weather data can be found at [26]. Archives of hourly data for different countries and places are available. The hourly air temperature profile in Moscow – price zone 1 over the modeling time frame – is shown in the Figure 12. Each point on the plot represents the air temperature for one whole hour. We are studying the same one-hour time frame as we did for the consumer prices.

We want to examine the correlation between the air temperature and consumer prices. For that we will combine the price and temperature hourly data as two features and then study their relationship. The results of the analysis are shown in the Figure 13. First, we applied the K-Means clustering algorithm [23, p.521] to see if there are meaningful clusters in the data. We can clearly see that data has a rather V-shaped form with two distinct clusters: prices for warm and cool air temperatures. For each cluster we computed the coefficient of correlation [27, p. 265]. We can see that the consumer price *does* depend on the air temperature, at least for the warm cluster:

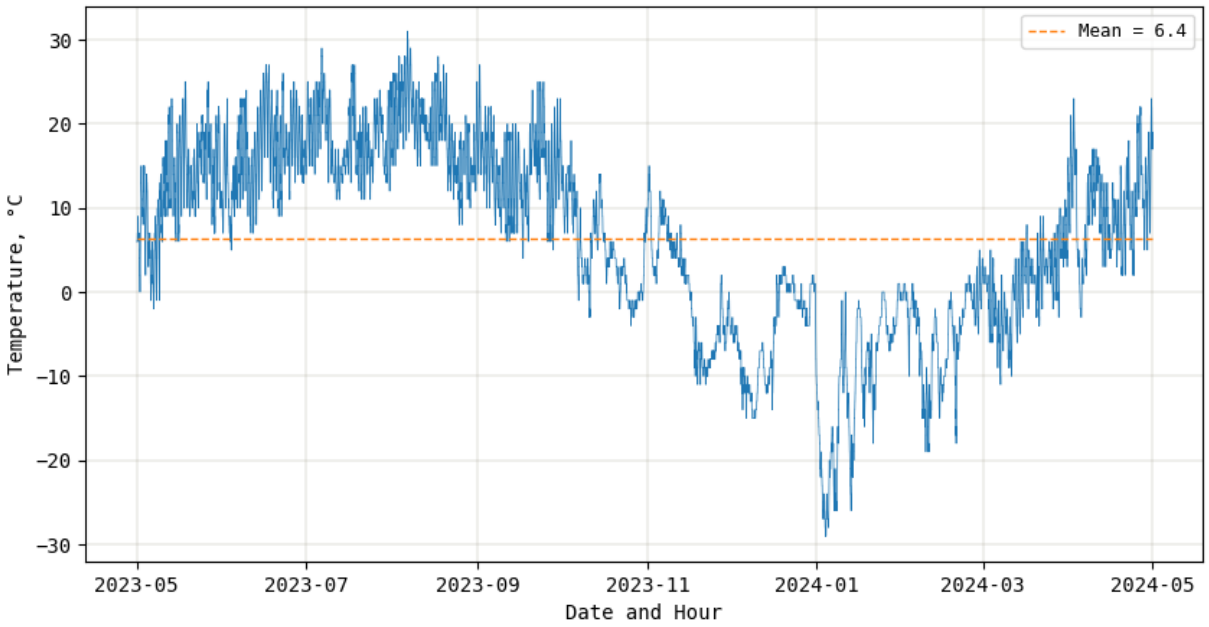

Fig. 12. Air temperature in Moscow

- High air temperatures cluster: coefficient of correlation $r_{warm} = 0.39$,
- Low air temperatures cluster: coefficient of correlation $r_{cool} = -0.01$.

The price tends to be higher for higher and lower air temperatures, which confirms the demand expectations: electricity demand is higher during the hot and cold seasons of the year (both cooling and heating are required), while the demand is minimal during semi-seasons when minimal heating and cooling are required. At the same time, the price responds more to the higher air temperatures.

Second, for the whole data we fitted a third degree polynomial:

$$y_t \propto X_t^3, \tag{9}$$

where X_t is the hourly air temperature at time t. We tested linear, parabolic and cubic functions and chose the cubic. We did not want our new model to either under- (linear, parabolic) or overfit (degrees larger than 3). Also, observe that the domain of 9 is almost surely limited within a reasonable interval of air temperatures, which means that the model should not either underfit or overfit, since it is simply not defined outside of this interval. Finally, taking into account computation complexity with Bayesian inference with Stan, the cubic polynomial seems to be a proper choice.

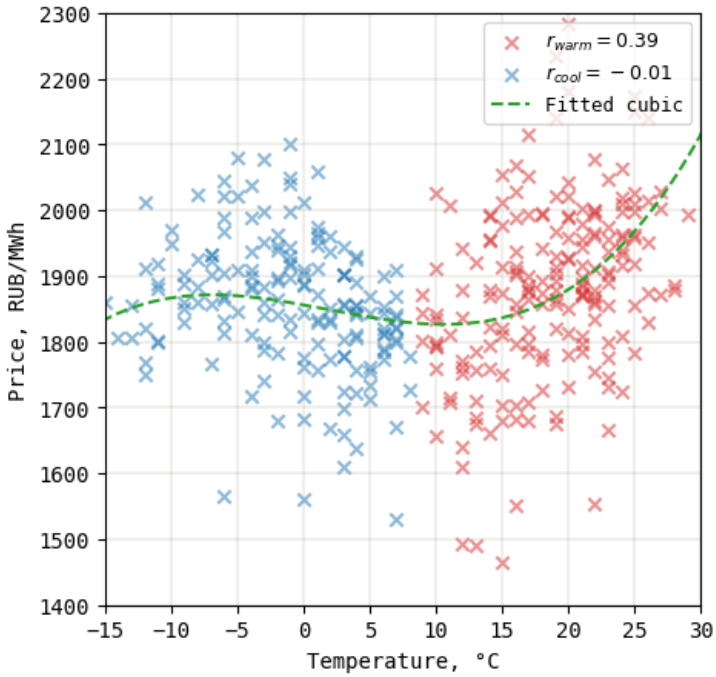

Fig. 13. Price vs Temperature

4.2 Weekday

Our next possible regressor might be the day of the week. It may well be expected that the electricity demand is not constant during the week with peak and off-peak days, just like it is not constant during any given day. Formally, both the consumer price and day of the week are discrete RV, and we might apply the χ^2-test for their independence [27, p. 714]. However, since the price's cardinality is much larger than that of the day of the week, the χ^2-test is not practically applicable, since most likely the contingency table will not satisfy the minimal count assumption of the test ($N_{Cell \neq 0} \geq 80\%$). Instead, we assess their correlation with box plots – see Figure 14.

The box plots confirms that the mean price tends to be lower on Sundays and the volatility tends to be lower on Fridays, which suggests that the consumer price depends on the day of the week:

$$y_t \propto D_t, \tag{10}$$

where D_t is the day of the week at time t.

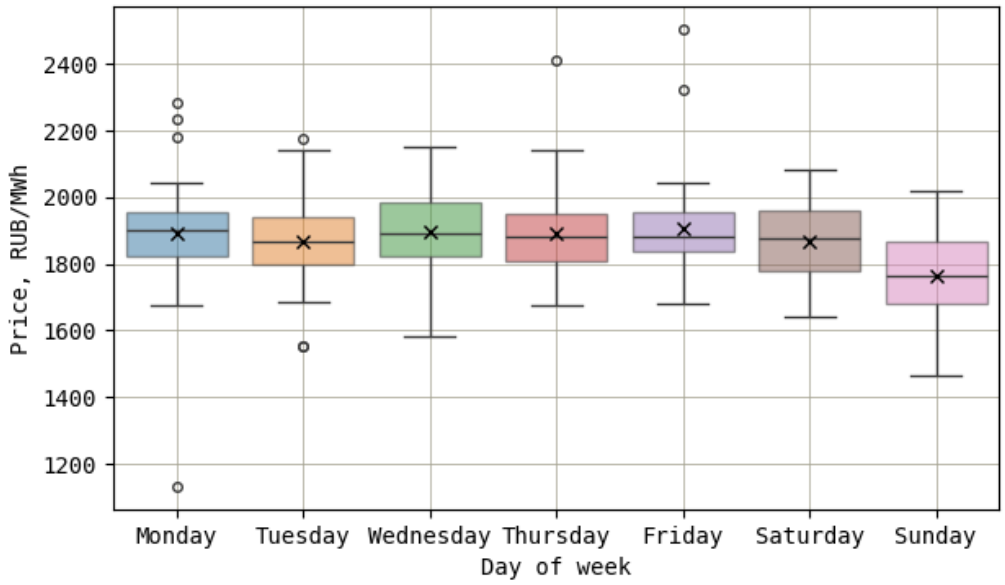

Fig. 14. Correlation between the price and day of the week

4.3 Autoregressive Component

We will examine the correlation between the prices at time t (today) and $t-1$ (yesterday). For that, we will compute the PACF [21] which is the correlation between two observations that the shorter lags between those observations do not explain, i.e., the partial correlation for each lag is the unique correlation between those two observations after removing out the intervening correlations. The plot of PACF for the price zone 1 and peak hour #14 (modeling dataset) is shown in the Figure 15.

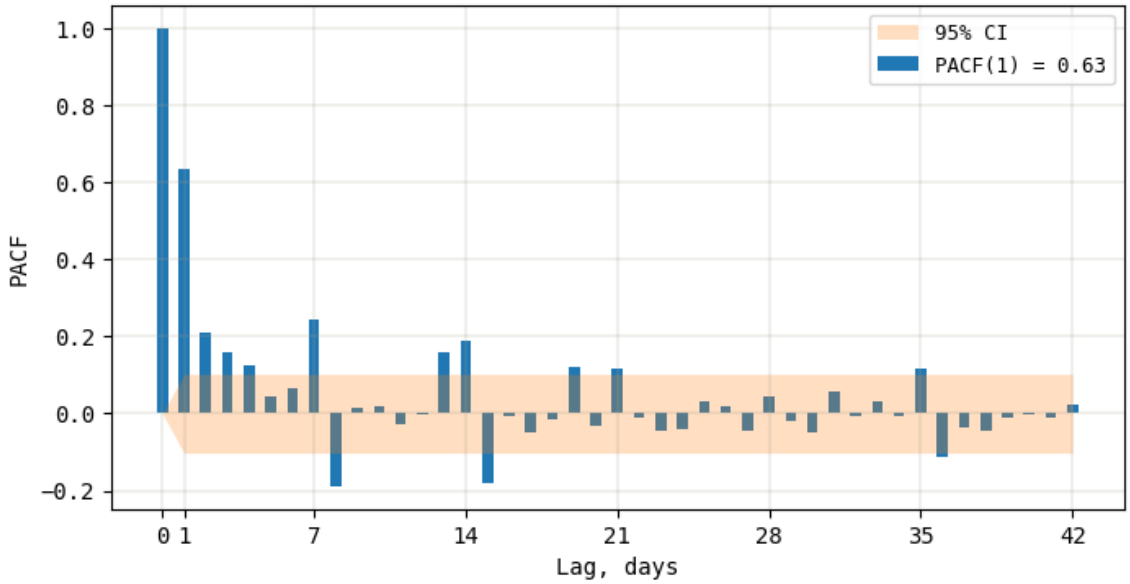

Fig. 15. PACF

Since our price is not stationary (see 3.2), the PACF decays rather slowly during the period of 42 days (6 weeks). We have shown above that the price *is* correlated with the day of the week. Observe that the PACF has spikes exactly every 7 days (1 week) which further supports the correctness of possible inclusion of the day of the week as an exogenous regressor.

PACF suggests that using at least 7 or more lags (new features) might be possible. We already decided to include the day of the week to explain the seasonality of the price. The strongest correlation is seen for the lag of one (0.63), while other lags are much weaker correlated with the price. Taking into account computational complexity of Bayesian inference with Stan, using only the lag of one day seems to be a reasonable compromise. In other words, we will consider an autoregressive model with lag of one AR(1) as the autoregressive component.

A first-order autoregressive model AR(1) is the following [9]:

$$y_t = \mathcal{N}\left(\alpha + \beta y_{t-1}, \sigma\right), \tag{11}$$

where α and β are the intercept and slope of the autoregressive component and $\sigma \sim \mathcal{N}(0, 1)$ (constant normal volatility). Since we model volatility as a stochastic process (not constant), but the AR-type models assume homoscedastic volatility [1, p. 1055], we will consider only the mean value of the AR(1) model.

4.4 Modeling

We have shown that the consumer price is correlated with 1) air temperature, 2) day of the week, and 3) with itself with at least lag of one day. We propose a new model SV X, which extends our SV Baseline model 6 with the three exogenous regressors 9, 10, 11:

$$y_t \sim \mathcal{N}\left(\bar{y} + \alpha y_{t-1} + \beta_3 X_{t-1}^3 + \beta_2 X_{t-1}^2 + \beta_1 X_{t-1} + \gamma D_t + \xi, e^{h_t/2}\right), \tag{12}$$

where X_{t-1} is the hourly air temperature at time $t-1$. To prevent target leakage, the temperature readings are lagged one day behind $(t-1)$; D_t is the day of the week at time t.

Thus, we are introducing 6 new model parameters for 3 exogenous regressors:

- α, autoregressive component;
- $\beta_{i=1...3}$, air temperature regressor. The unit of air temperature is °C;
- γ, day of the week regressor. The weekdays are numbered from 0 (Monday) to 6 (Sunday);
- ξ, constant term (intercept) for all exogenous regressors.

Recall that we assumed the constant mean $\bar{y}$ in the SV Baseline model which is not correct for our non-stationary price. SV X takes care of this drawback by ”mixing” exogenous regressors to the constant price mean in a regression-like manner.

After running the Stan code, we have obtained the posterior distributions of all SV X model's parameters – see Figure 16.

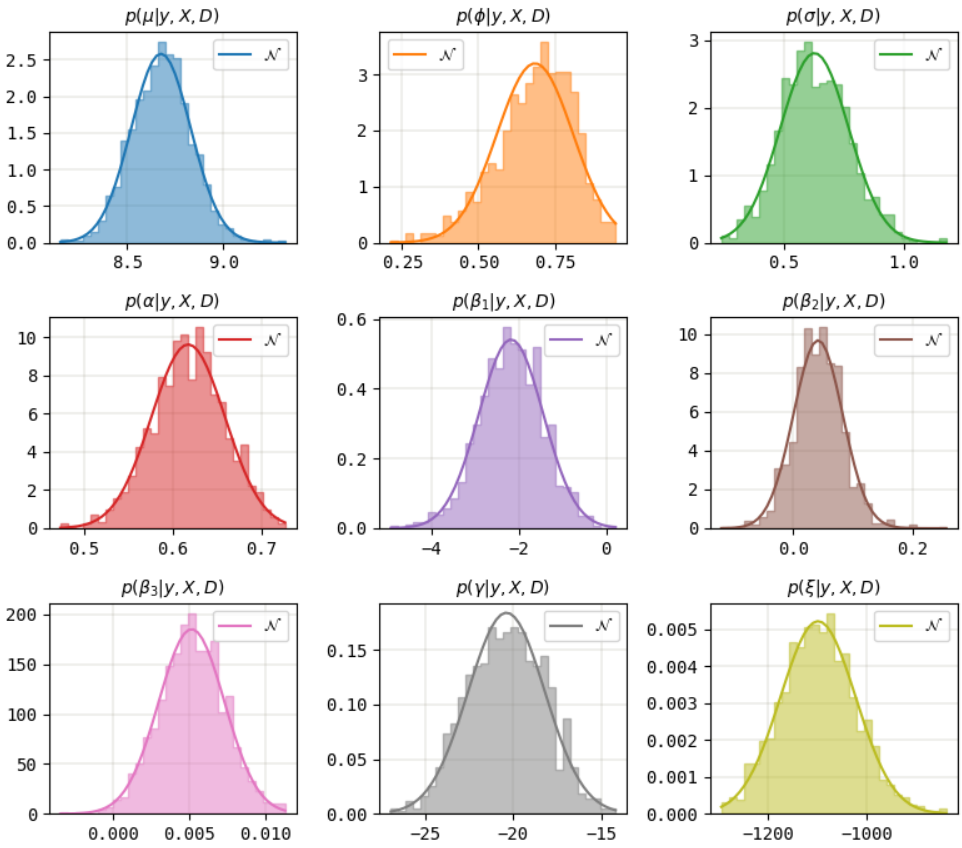

Fig. 16. SV X: Posterior distributions of estimated parameters

We can now use the estimated model's parameters to compute PDD and generate in-sample predictions for the modeling time frame. 1,000 density plots sampled from the PDD of the consumer price are shown in the Figure 17.

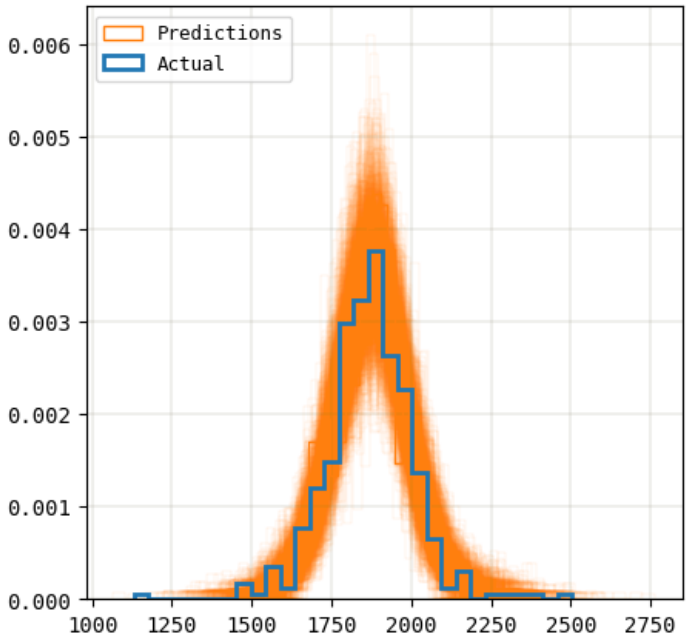

Fig. 17. SV X: 1,000 predicted and actual price distributions

Observe, that the actual distribution (density) of the consumer prices is well within the predicted density plots, which confirms the correctness of modeling. 1,000 trace plots sampled from the PDD of the consumer prices are shown in the Figure 18.

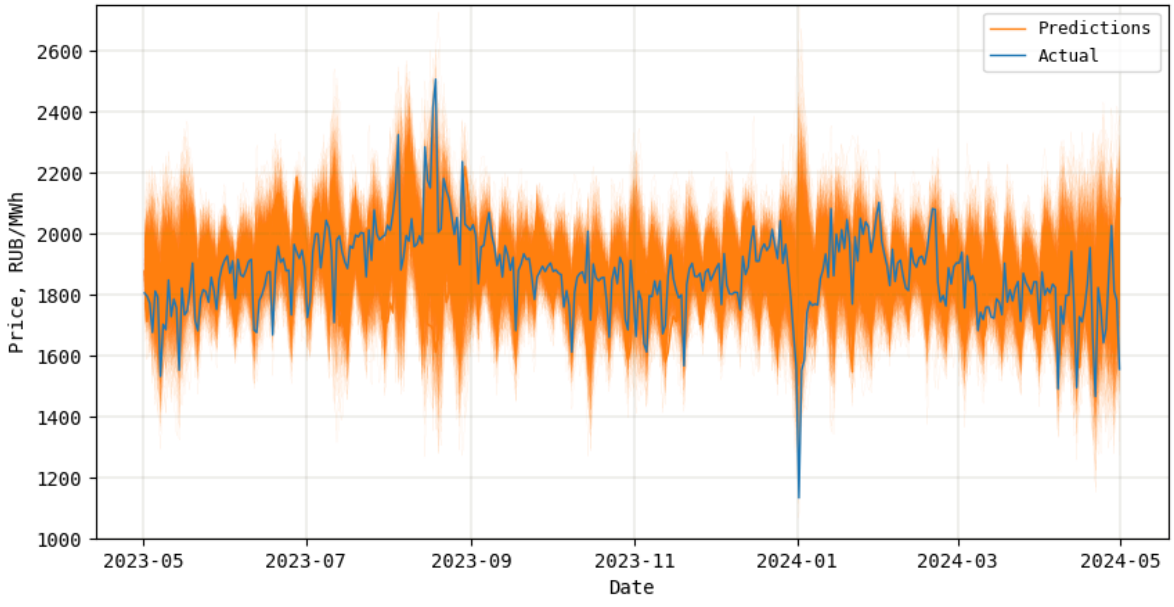

Fig. 18. SV X: Actual price and 1,000 predictions

The 95% CI for all 1,000 predictions is shown in the Figure 19.

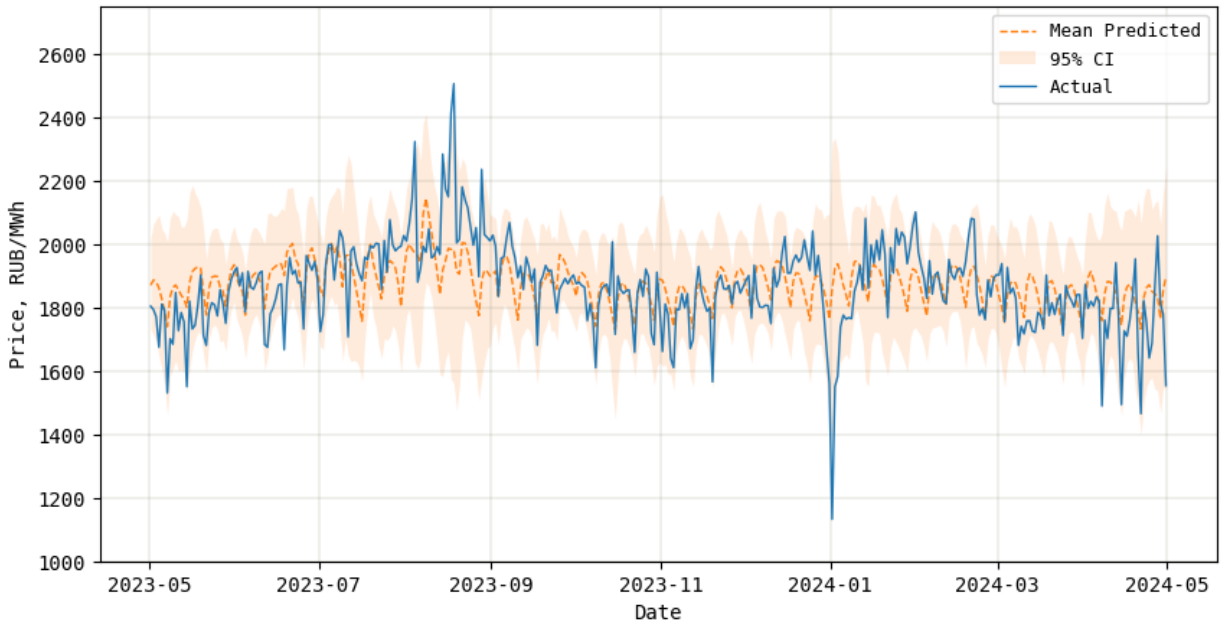

Fig. 19. SV X: 95% CI for 1,000 predictions

The model has learned the latent stochastic volatility process quite correctly. The learned volatility $e^{h_t/2}$ (both mean and 95% CI) is shown in the Figure 20.

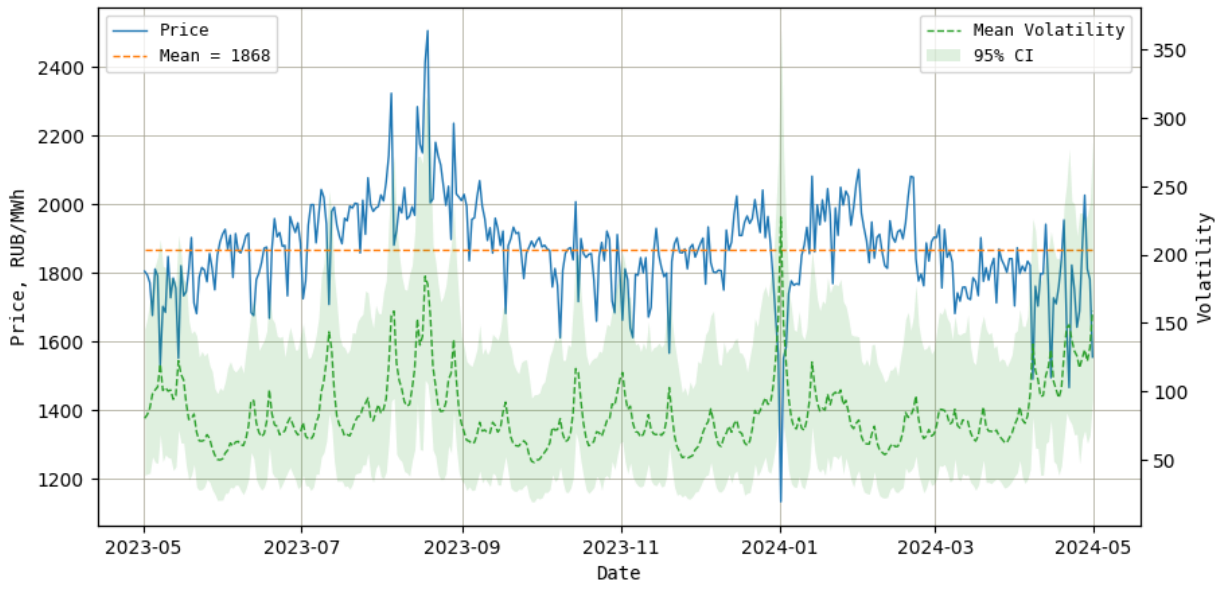

Fig. 20. SV X: Consumer prices and learned volatility

To understand the behavior of the volatility we will visually check the correlation between the price and volatility – see Figure 21.

We can clearly see that volatility *does* depend on consumer price, having a rather

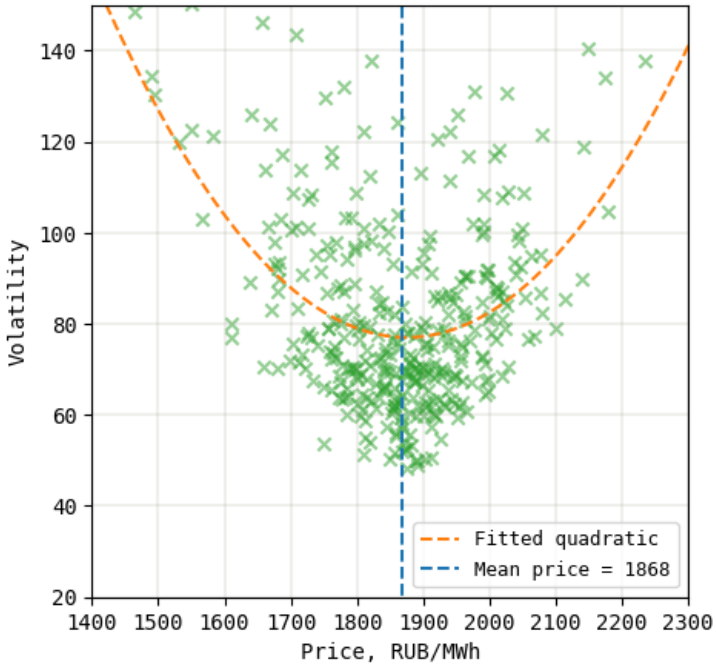

Fig. 21. SV X: Learned volatility vs Consumer price

V-shaped dependency: the volatility tends to increase for the prices higher and lower than the mean price, while it's minimal for the prices around the mean.

4.5 Goodness-of-fit

The metrics for the in-sample predictions made by the SV X model are shown in the Table 3. SV Baseline model's metrics are shown for comparison.

Metric	SV Baseline	SV X
MAE	99.18	**85.23** (-14.06%)
RMSE	137.04	**117.91** (-13.96%)

Table 3. SV X: MAE and RMSE for predicted consumer prices

We also want to check the correlation between the predicted mean price and actual price – see Figure 22. Observe, that the correlation is much stronger, as compared to the SV Baseline model.

We should also check the distribution of residuals, probability plot, and correlation between residuals and mean predicted prices – see Figure 23. Residuals look to be

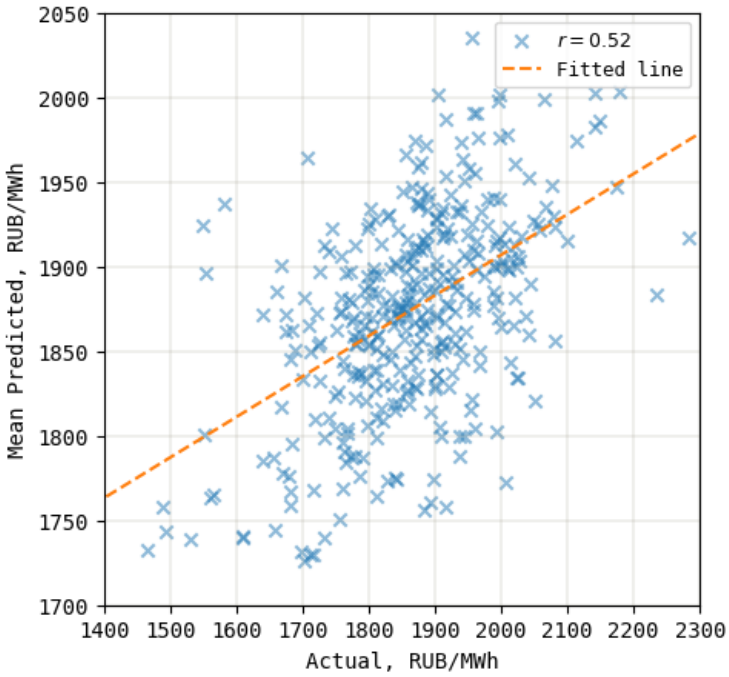

Fig. 22. SV X: Predicted vs Actual price

distributed more or less normally with several outliers. Also, residuals are almost not correlated with predictions (homoscedastic).

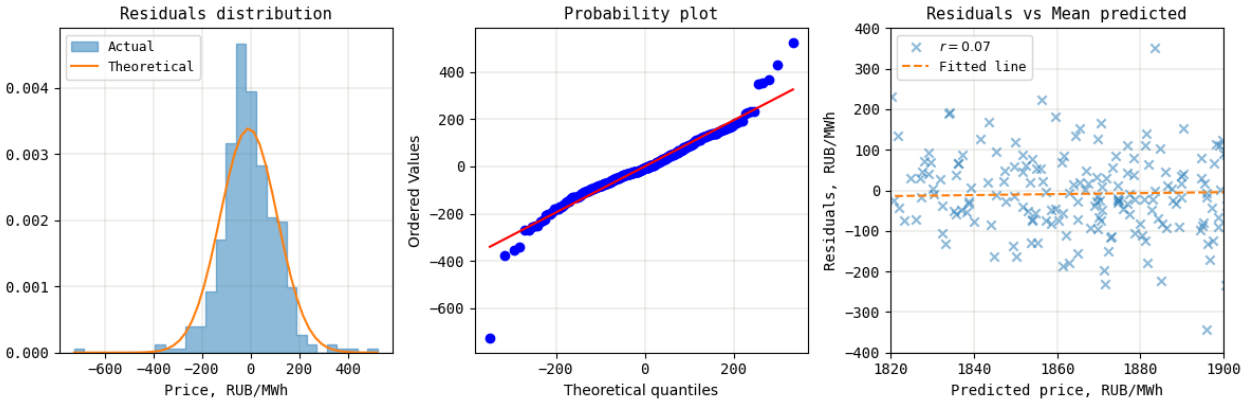

Fig. 23. SV X: Residuals plots

We will *accept* the SV X model as a *better* fit for our price data with exogenous regressors, as compared to the SV Baseline model.

5. CROSS-VALIDATION

We have trained and tested our models on the same fixed year-long modeling time frame. There is a common practice in ML to train and test models on different portions of the whole data to generate the distribution of metric(s) to ensure model robustness. This process is known as cross-validation [23, p. 202]: all data is divided into a number of non-intersecting subsets of samples (folds) some of which are used for training and the rest for testing. For data with i.i.d. samples, i.e., when the samples are independent and thus the order of samples is not important, the order of folds is also not important. This is not the case for time series data, where training folds must not precede the testing folds, otherwise the target leakage will occur and the results cannot be trusted.

5.1 Strategy

Model combinations to cross-validate are shown in the Table 4.

Combination #	Model family	Hour	Price zone
1	SV Baseline	Peak	1 (European)
2		Peak	2 (Siberian)
3		Off-peak	1
4		Off-peak	2
5	SV X	Peak	1
6		Peak	2
7		Off-peak	1
8		Off-peak	2

Table 4. Model combinations to cross-validate

Time frame: 23.06.2014 – 30.04.2024 (3600 days), approx. ten years back from now. The number of date sliding windows, and thus the number of train-test folds, can be computed as $N_w = \frac{3600-30\times12}{30\times3} = 36$. All 36 train-test fold time frames, as well as the consumer price and air temperature data descriptive statistics for both price zones over

the cross-validation time frame can be found in the author's computations notebook [18]. The cross-validation algorithm 1 is shown below.

Algorithm 1 Cross-validation

1: Split the data into train-test folds.
2: **for all** Model $\in [1, 8]$ **do** ▷ *4 SV Baseline + 4 SV X models*
3: **for all** Fold $\in [1, N_w]$ **do** ▷ $N_w = 36$
- Fit on train fold (360 days – approx. 1 year); ▷ *80/20 split*
- Predict on test fold (90 days – approx. 3 months = 1 quarter) using the most recent log volatility learned during training (90 points);
- Compute metrics: actual (test fold) vs mean predicted.

4: **end for**
5: **end for**

Hour and temperature hyperparameters used for cross-validation are shown in the Table 5.

Price zone	Peak hour	Off-peak hour	Air temperature city
1 (European)	#11	#3	Moscow
2 (Siberian)	#16	#1	Novosibirsk

Table 5. Hyperparameters used for cross-validation

Observe that the peak hour for the price zone 1 differs from the one used in modeling (#14). This can be explained by a different (10 times wider) time frame used for cross-validation as compared to the time frame used for modeling.

5.2 SV Baseline

During cross-validation, for each SV Baseline model and for each of the 36 folds, posterior distributions of all model parameters are obtained (on train fold) and then the PDD for predictions is computed from which 1,000 samples is drawn (on test fold).

The MAE and RMSE are computed, as in modeling, using the averaged samples of the predicted consumer prices for time t – see Table 6.

Model family	Hour	Price zone	MAE	RMSE
SV Baseline	Off-peak hour	1	**104.42**	137.50
	Peak hour	1	105.39	**130.53**
	Peak hour	2	119.14	146.45
	Off-peak hour	2	134.86	168.31

Table 6. SV Baseline: MAE and RMSE results for cross-validation

Both metrics show that on average the SV Baseline models for Price zone 1 (European) have higher quality than for Price zone 2 (Siberian).

5.3 SV X

Same as with SV Baseline, the MAE and RMSE are computed using the averaged samples of the predicted consumer prices for time t – see Table 7.

Model family	Hour	Price zone	MAE	RMSE
SV X	Off-peak	1	**93.92**	129.59
	Peak	1	99.45	**123.59**
	Peak	2	115.56	142.10
	Off-peak	2	118.18	150.73

Table 7. SV X: MAE and RMSE results for cross-validation

As with the SV Baseline model, both metrics for the SV X models show that on average the SV X models for Price zone 1 (European) have higher quality than for Price zone 2 (Siberian).

5.4 Model Comparison

We have generated metrics distributions during models cross-validation – see Figure 24 for MAE and Figure 25 for RMSE. Each distribution contains 36 values, as

per the number of cross-validation date sliding windows.

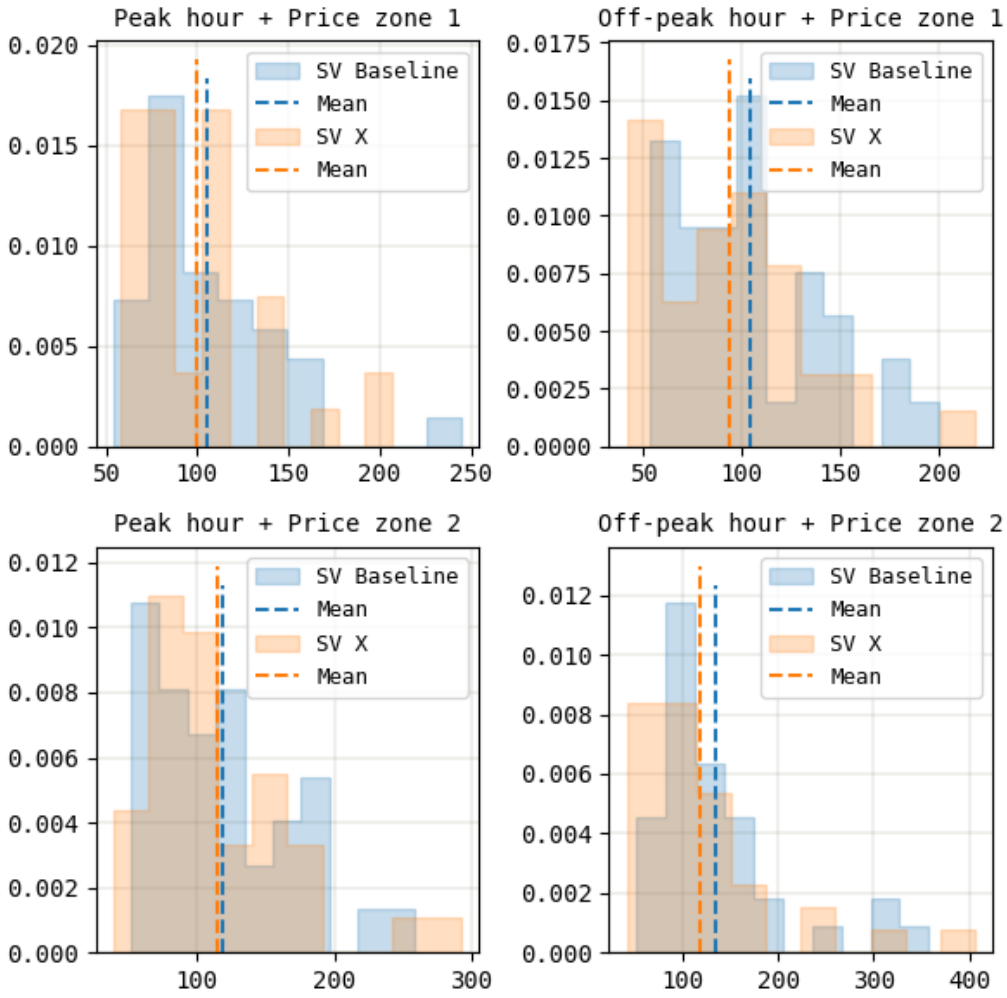

Fig. 24. MAE: Cross-validation for all models

We would also like to check the statistical significance of the difference in the metrics. We will use a non-parametric one-tailed Mann-Whitney U test for two independent samples [27, p. 758] to see if the distributions of metrics for SV Baseline and SV X models are identical or have a shift between each other.

$\alpha = 0.05$

$\mathcal{H}_0 : F_{X_1}(x) = F_{X_2}(x)$ (distributed identically)

$\mathcal{H}_1 : F_{X_1}(x) = F_{X_2}(x + \Delta x)$ (distributed with positive shift Δx), where

$F_{X_1}(x)$ is the distribution of a given metric for SV Baseline model; $F_{X_2}(x)$ is the distribution of a given metric for SV X model.

Statistic: Mann-Whitney's U

Null distribution: no assumption; generated empirically by permutation test; for

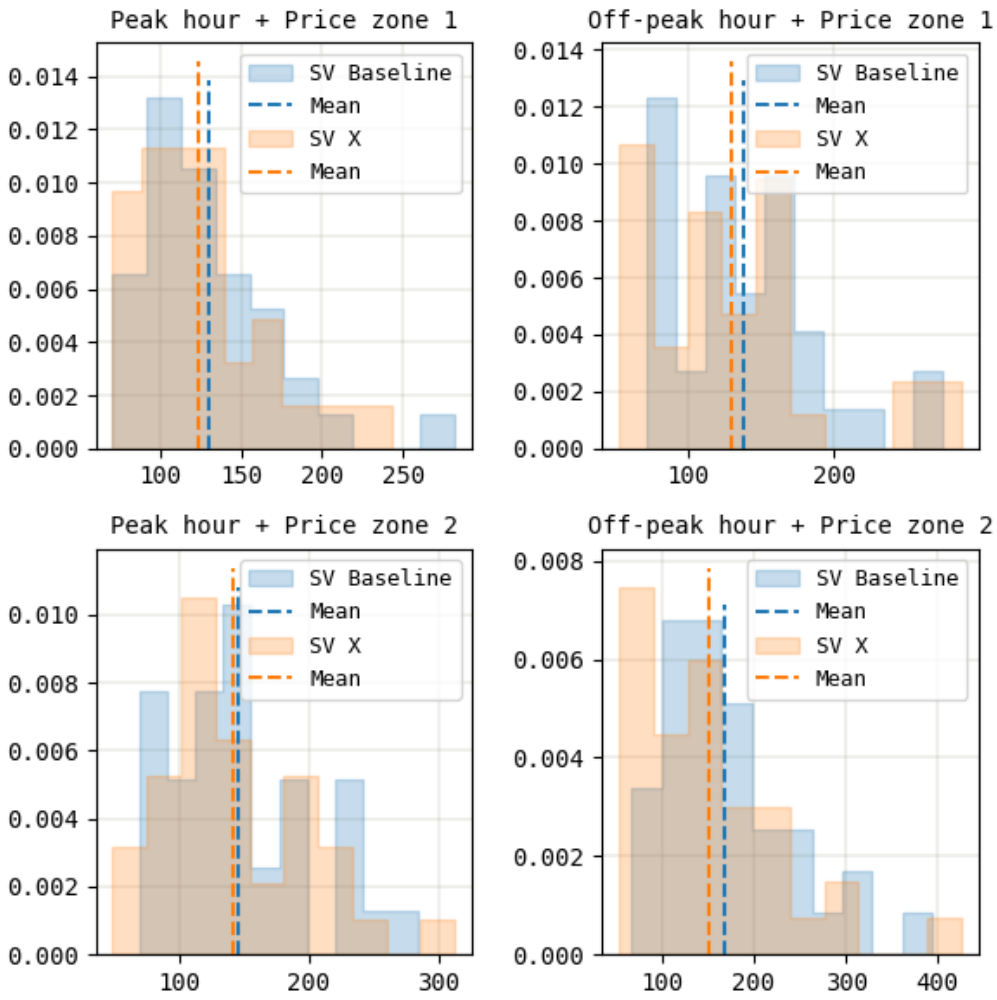

Fig. 25. RMSE: Cross-validation for all models

large samples it can be approximated by $\mathcal{N}\left(\mu = \frac{n_1 n_2}{2}, \sigma^2 = \frac{n_1 n_2 (n_1+n_2+1)}{12}\right)$.

The results of the MWU tests are shown in the Table 8.

Metric	Statistic	p-value	Result at α
MAE	11785	0.0225	$\Delta x > 0$
RMSE	11575	0.0439	$\Delta x > 0$

Table 8. MWU tests results

We have rejected the null hypothesis at the significance level of 0.05 in favor of the alternative hypothesis – SV Baseline's metrics are shifted to the right relative to SV X's metrics (= worse).

The summary of metrics for all models computed during cross-validation is shown

in the Table 9.

Metric	Hour	Price zone	SV Baseline	SV X
MAE	Peak	1	105.39	**99.45**
	Peak	2	119.14	**115.56**
	Off-peak	1	104.42	**93.92**
	Off-peak	2	134.86	**118.18**
		Average	115.95	**106.78** (-7.91%)
RMSE	Peak	1	130.53	**123.59**
	Peak	2	146.45	**142.10**
	Off-peak	1	137.50	**129.59**
	Off-peak	2	168.31	**150.73**
		Average	145.69	**136.50** (-6.31%)

Table 9. Cross-validation: Summary of metrics for all models

The results of cross-validation:

1. SV X's MAE is 7.91% less (= better) than SV Baseline's MAE on average;
2. SV X's RMSE is 6.31% less (= better) than SV Baseline's RMSE on average;
3. These differences are statistically significant at $\alpha = 0.05$;
4. Overall, the SV X model family is a *better* fit for our price data with the air temperature, day of the week, and lagged price as the exogenous regressors, as compared to the SV Baseline model family without exogenous regressors.

6. FORECASTING

Having built and cross-validated our models, we would finally like to generate forecasts for the nearest day(s) ahead with them.

6.1 Strategy

Just like with cross-validation, we will generate forecasts for 8 model combinations (see Table 4). First train time frame: 01.05.2023 – 30.04.2024 (same as used for modeling). Forecast time frame: 01.05.2024 – 07.05.2024 (1 week ahead). The forecasting algorithm 2 is shown below.

Algorithm 2 Forecasting

1: Set the train time frame to the first train time frame.
2: **for all** Model $\in [1, 8]$ **do** ▷ *4 SV Baseline + 4 SV X models*
3: **for all** Day $\in [1, 7]$ **do** ▷ *7 days = 1 week*
- Fit the model on the train time frame (1 year);
- Generate *one* day-ahead prediction (forecast) using the most recent log volatility learned during training (1 point), and populate the forecasted prices;
- Move the train time frame forward by *one* day.

4: **end for**
5: Compute metrics: actual vs mean forecasted;
6: **end for**

6.2 Results

For all model combinations and for each day ahead we have drawn 1,000 forecasts from the day's PDD. The plots of mean of each 1,000 draws and 95% CI for forecasts are shown in the Figures 26, 27, 28, 29.

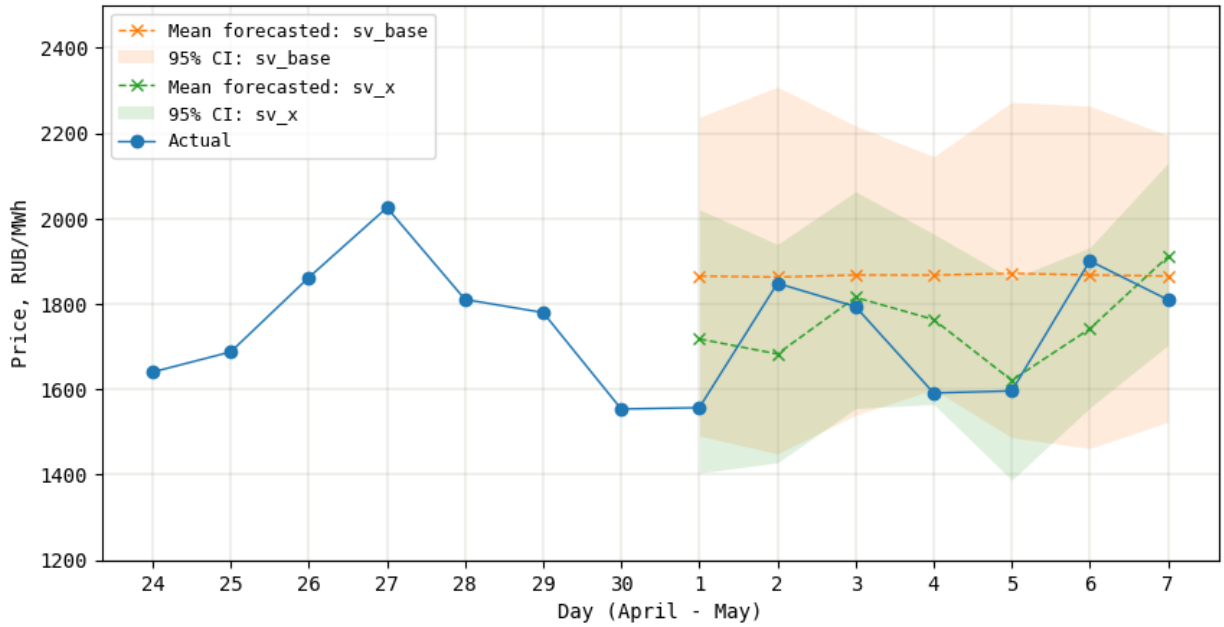

Fig. 26. Forecasts: Peak hour + Price zone 1

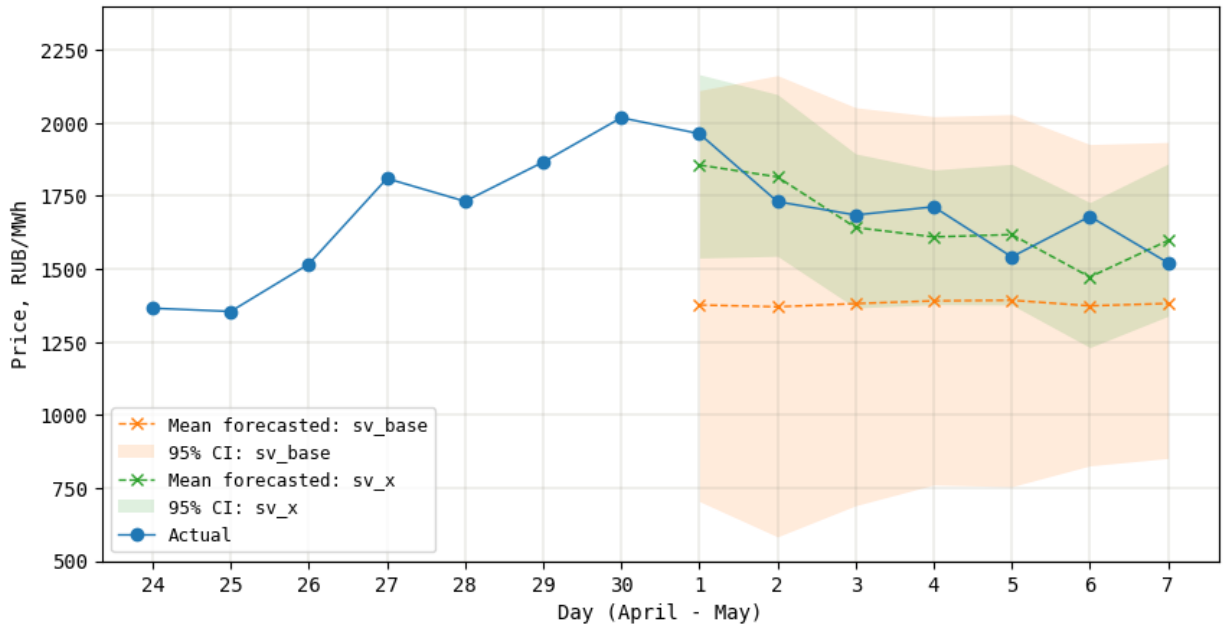

Fig. 27. Forecasts: Peak hour + Price zone 2

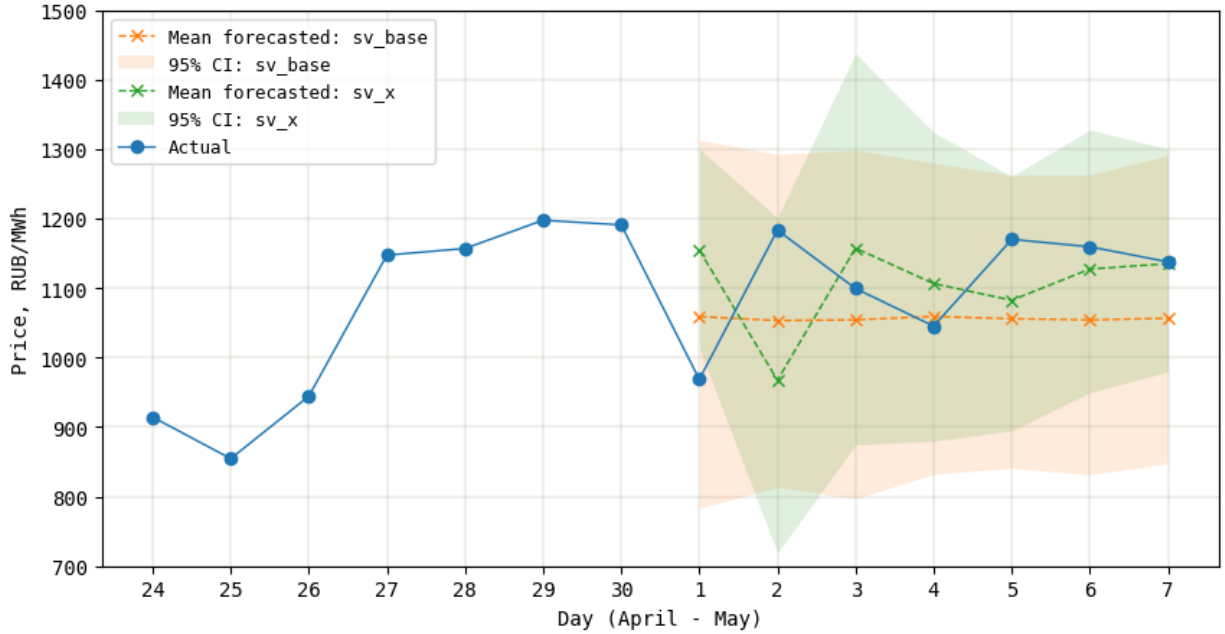

Fig. 28. Forecasts: Off-peak hour + Price zone 1

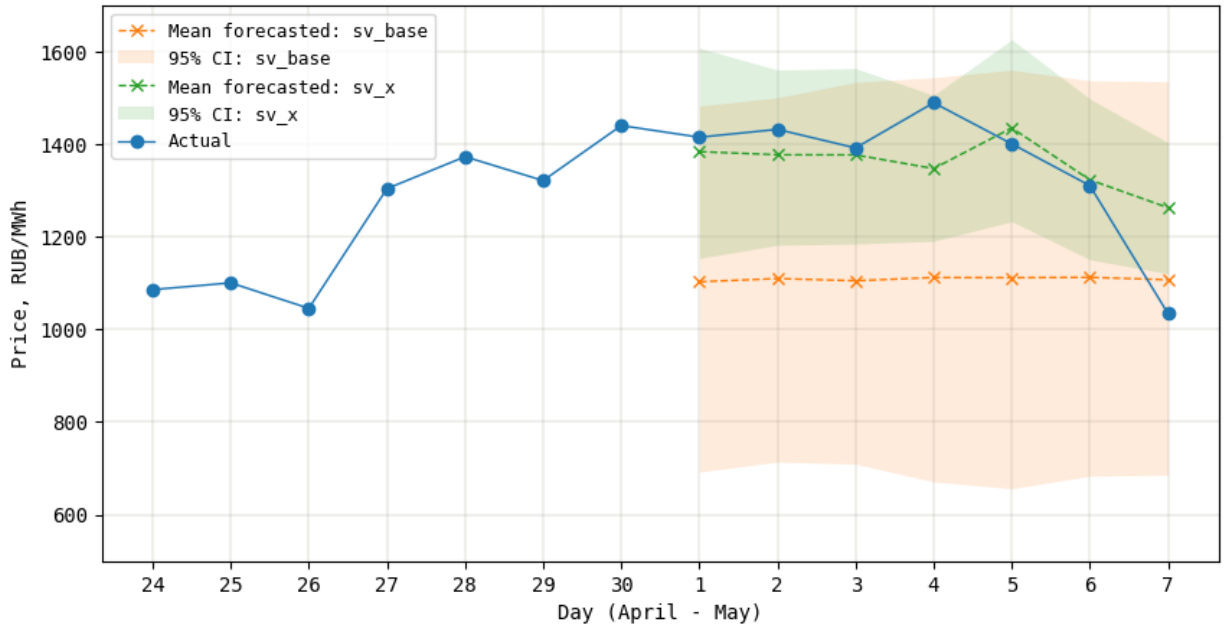

Fig. 29. Forecasts: Off-peak hour + Price zone 2

For computing the metrics, all draws were averaged for each day (same as in modeling and cross-validation). The summary of metrics is shown in the Table 10.

Metric	Hour	Price zone	SV Baseline	SV X
MAE	Peak	1	148.06	**114.95**
	Peak	2	309.28	**99.56**
	Off-peak	1	**82.84**	91.94
	Off-peak	2	265.35	**73.97**
		Average	201.38	**95.11** (-52.77%)
RMSE	Peak	1	191.49	**130.60**
	Peak	2	338.86	**110.59**
	Off-peak	1	**91.02**	117.76
	Off-peak	2	281.16	**105.44**
		Average	225.63	**116.10** (-48.54%)

Table 10. Forecasting: Summary of metrics for all models

The results of forecasting:

1. SV X's MAE is 52.77% less (= better) than SV Baseline's MAE on average;
2. SV X's RMSE is 48.54% less (= better) than SV Baseline's RMSE on average;
3. The SV X model is a *better* fit for Peak hour + Price zone 1, Peak hour + Price zone 2, Off-peak + Price zone 2.
4. The SV X model is a *worse* fit for Off-peak hour + Price zone 1.

In general, all models are able to learn the price change – 95% CI for all sampled forecasts cover the actual price. SV X model family provides a narrower forecast CI as compared to SV Baseline model family. The reason for this is that for the SV X model the latent log volatility h_t is learned w.r.t. the mean price being not simply a constant value but a function of the mean and exogenous regressors, as opposed to the SV Baseline model. As a result, h_t can "follow" the price change more precisely.

7. CONCLUSION AND FUTURE WORK

We have built and tested our models for two market hours: peak and off-peak, and two price zones: 1 (European), 2 (Siberian). The results are a good estimation and understanding of the models' behavior. On average, the family of SV X models is a *better* fit for predicting the price over time with exogenous regressors (see Tables 9 and 10). The forecasting results show that for the Price zone 1 + Off-peak hour the better performance is provided by the SV Baseline model, while for the rest three scenarios the best performer is the SV X model. Thus, we should build similar models for all 24 hours for the day-ahead for both price zones and then validate and compare all obtained models to choose the best performer for future forecasting. Ideally, we should build 48 hourly models tailored to each price zone and each hour of the day.

The cross-validation and forecasting results confirm the applicability and robustness of the enhanced SV X model. This model may be used in financial derivative instruments for hedging the risk associated with electricity trading.

The author is developing their own ML library in C++ called EasyML [28]. The main motivation is to build a compact, flexible and fast library tailored for subject matter tasks. The library already covers the main linear models: Linear Regression, Logistic Regression, AR(p). There are also data transformers already available: Standard Scaler (z-scores) and Time Series Feature Extractor (lags).

We have introduced only one weather regressor – air temperature. At the same time, it might well be expected that other climate and weather factors drive the electricity demand: humidity, precipitation, solar irradiance, wind speed, etc. The future research might include these factors as new regressors into modeling. As the number of regressors increase, the computational complexity of training and generating predictions increases too. We have studied 8 models, while in fact their number might be up to 48 (96 including SV Baseline). The need to re-train this number of models each and every day to generate the next day-ahead forecast makes the author believe that a task-tailored library will increase computation speed thus allowing to build more complex models and validate them more rigorously and promptly in production-like scenarios.

REFERENCES

[1] Rafał Weron, "Electricity price forecasting: A review of the state-of-the-art with a look into the future," *International Journal of Forecasting (30)*, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207014001083

[2] Kim Sangjoon, Neil Shephard, Siddhartha Chib, "Stochastic Volatility: Likelihood Inference and Comparison with ARCH Models," *Review of Economic Studies (65)*, 1998. [Online]. Available: https://apps.olin.wustl.edu/faculty/chib/papers/KimShephardChib98.pdf

[3] Joshua C.C. Chan, Angelia L. Grant, "Modeling Energy Price Dynamics: GARCH versus Stochastic Volatility," *Energy Economics (54)*, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0140988315003539

[4] Simon Rogers, Mark Girolami, *A First Course in Machine Learning*. CRC Press, Taylor & Francis Group, 2012.

[5] Python Software Foundation, *Python*. [Online]. Available: https://www.python.org/about/

[6] Jupyter Team, *Jupyter Notebook*. [Online]. Available: https://jupyter-notebook.readthedocs.io/en/latest/notebook.html

[7] Microsoft, *Visual Studio Code*. [Online]. Available: https://code.visualstudio.com/

[8] Docker Inc., *What is a container?* [Online]. Available: https://docs.docker.com/guides/docker-concepts/the-basics/what-is-a-container/

[9] Stan Development Team, *Time-Series Models. Stan User's Guide*. [Online]. Available: https://mc-stan.org/docs/stan-users-guide/time-series.html

[10] pystan Developers, *PyStan*. [Online]. Available: https://pystan.readthedocs.io/en/latest/

[11] scikit-learn developers, *scikit-learn User's Guide*. [Online]. Available: https://scikit-learn.org/stable/user_guide.html

[12] pandas via NumFOCUS, Inc., *pandas documentation*. [Online]. Available: https://pandas.pydata.org/docs/

[13] NumPy Developers, *NumPy documentation*. [Online]. Available: https://numpy.org/doc/stable/index.html

[14] The SciPy community, *SciPy documentation*. [Online]. Available: https://docs.scipy.org/doc/scipy/

[15] Josef Perktold, Skipper Seabold, Jonathan Taylor, statsmodels-developers, *statsmodels User's Guide*. [Online]. Available: https://www.statsmodels.org/stable/user-guide.html

[16] John Hunter, Darren Dale, Eric Firing, Michael Droettboom and the Matplotlib development team, *Matplotlib User's Guide*. [Online]. Available: https://matplotlib.org/stable/users/index

[17] Michael Waskom, *seaborn User's Guide*. [Online]. Available: https://seaborn.pydata.org/tutorial.html

[18] Andrei Batyrov. (2024) Master's Thesis Materials. GitHub Repository. National Research University Higher School of Economics. Faculty of Computer Science. [Online]. Available: https://github.com/andrewha/mds2022/tree/main/Thesis

[19] Michael Clark, *Bayesian Stochastic Volatility Model*. [Online]. Available: https://m-clark.github.io/models-by-example/bayesian-stochastic-volatility.html

[20] Administrator of Trading System (ATS). Daily indices and volumes of the day-ahead market. [Online]. Available: https://www.atsenergo.ru/results/rsv/index

[21] Prashant Banerjee, *Complete Guide on Time Series Analysis in Python*. [Online]. Available: https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python

[22] James G. MacKinnon, “Critical Values For Cointegration Tests,” *Queen’s Economics Department*, 2010. [Online]. Available: https://www.econ.queensu.ca/sites/econ.queensu.ca/files/wpaper/qed_wp_1227.pdf

[23] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, Jonathan Taylor, *An Introduction to Statistical Learning with Applications in Python*. Springer, 2023. [Online]. Available: https://www.statlearning.com/

[24] Maciej Kostrzewski, Jadwiga Kostrzewska, “Probabilistic electricity price forecasting with Bayesian stochastic volatility models,” *Energy Economics (80)*, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0140988319300544

[25] Ksenia Kasianova, “Comparison of the dynamics of spot prices for electricity in the European and Siberian price zones of Russia in a stochastic volatility model,” *Economics and mathematical methods (60)*, 2024. [Online]. Available: https://publications.hse.ru/en/articles/923455155

[26] Raspisaniye Pogodi Ltd. Reliable Prognosis. [Online]. Available: https://rp5.ru/

[27] Dennis D. Wackerly, William Mendenhall III, Richard L. Scheaffer, *Mathematical Statistics With Applications*. Thomson Brooks/Cole, 2008.

[28] Andrei Batyrov, “Sometimes you really want to re-invent the wheel: EasyML,” *Medium.com*, 2024. [Online]. Available: https://medium.com/@handrewkha/sometimes-you-really-want-to-re-invent-the-wheel-easyml-8fa3ab99aed5

APPENDIX

A.1. Stan Code for SV Baseline Model

```
/**
 * @file sv_base_fit.stan
 * @author Andrei Batyrov (arbatyrov@edu.hse.ru)
 * @brief SV Baseline model: Fit method
 * Exogenous regressors: None
 * @version 0.1
 * @date 2024-05-08
 *
 * @copyright Copyright (c) 2024
 *
 */

data
{
  int<lower=1> N; // Number of train time points (equally spaced)
  vector[N] y;    // Vector of train prices at time t
  real y_mean;    // Mean price
}

parameters
{
  real mu;                     // Mean log volatility
  real<lower=-1, upper=1> phi; // Persistence of volatility
  real<lower=0> sigma;         // White noise shock scale
  vector[N] h_std;             // Standardized log volatility at time t
}
```

```
27
28 transformed parameters
29 {
30   // Log volatility at time t; now h ~ normal(0, sigma)
31   vector[N] h = h_std * sigma;
32   h[1] = h[1] / sqrt(1 - pow(phi, 2)); // Rescale h[1]
33   h = h + mu;
34   for (t in 2:N)
35   {
36     h[t] = h[t] + phi * (h[t - 1] - mu);
37   }
38 }
39
40 model
41 {
42   // Priors recommended by Stan
43   phi ~ uniform(-1, 1);
44   sigma ~ cauchy(0, 5);
45   mu ~ cauchy(0, 10);
46   h_std ~ std_normal();
47
48   // Model
49   y ~ normal(y_mean, exp(h / 2));
50 }
```

Listing 1. SV Baseline: Fit method

```
1 /**
2  * @file sv_base_predict.stan
3  * @author Andrei Batyrov (arbatyrov@edu.hse.ru)
```

```
 * @brief SV Baseline model: Predict method
 * Exogenous regressors: None
 * @version 0.1
 * @date 2024-05-08
 *
 * @copyright Copyright (c) 2024
 *
 */

data
{
  int<lower=1> N_pred; // Number of test time points (equally spaced)
  vector[N_pred] h;    // Vector of learned log volatility at time t
  real y_mean;         // Learned mean price
}

generated quantities
{
  vector[N_pred] y_pred;

  // Generate posterior predictive distribution
  for (t in 1:N_pred)
  {
      y_pred[t] = normal_rng(y_mean, exp(h[t] / 2));
  }

}
```

Listing 2. SV Baseline: Predict method

A.2. Stan Code for SV X Model

```
/**
 * @file sv_x_fit.stan
 * @author Andrei Batyrov (arbatyrov@edu.hse.ru)
 * @brief SV Exogenous model: Fit method
 * Exogenous regressors:
 * - Temperature
 * - Weekday
 * - AR(1)
 * @version 0.1
 * @date 2024-05-08
 *
 * @copyright Copyright (c) 2024
 *
 */

data
{
  int<lower=1> N;         // Number of train time points (equally spaced)
  vector[N] y;            // Vector of train prices at time t
  real y_mean;            // Mean price
  vector[N] Temperature; // Vector of train Temperature at time t
  vector[N] Weekday;     // Vector of train Weekday at time t
}

parameters
{
  real mu;                      // Mean log volatility
```

```
  real<lower=-1, upper=1> phi; // Persistence of volatility
  real<lower=0> sigma;         // White noise shock scale
  vector[N] h_std;             // Standardized log volatility at time t
  real alpha;                  // Param of AR(1) component
  real beta_1;                 // Params of Temperature regressor
  real beta_2;
  real beta_3;
  real gamma;                  // Param of Weekday regressor
  real xi;                     // Intercept for all exogenous regressors

}

transformed parameters
{
  // Log volatility at time t; now h ~ normal(0, sigma)
  vector[N] h = h_std * sigma;
  h[1] = h[1] / sqrt(1 - pow(phi, 2)); // Rescale h[1]
  h = h + mu;
  for (t in 2:N)
  {
    h[t] = h[t] + phi * (h[t - 1] - mu);
  }
}

model
{
  // Priors recommended by Stan
  phi ~ uniform(-1, 1);
  sigma ~ cauchy(0, 5);
```

```
  mu ~ cauchy(0, 10);
  h_std ~ std_normal();

  // Model
  // We do not model y[t = 1],
  // since we do not have y[t - 1 = 0] and Temperature[t - 1 = 0]
  // So assume the first value without Temperature and AR(1)
  // and continue from index 2
  y[1] ~ normal(y_mean + gamma * Weekday[1],
                exp(h[1] / 2));
  for (t in 2:N)
  {
    y[t] ~ normal(y_mean +
                  alpha * y[t - 1] +
                  beta_3 * pow(Temperature[t - 1], 3) +
                  beta_2 * pow(Temperature[t - 1], 2) +
                  beta_1 * Temperature[t - 1] +
                  gamma * Weekday[t] +
                  xi,
                  exp(h[t] / 2));
  }
}
```

Listing 3. SV X: Fit method

```
/**
 * @file sv_x_predict.stan
 * @author Andrei Batyrov (arbatyrov@edu.hse.ru)
 * @brief SV Exogenous model: Predict method
 * Exogenous regressors:
```

```
 * - Temperature
 * - Weekday
 * - AR(1)
 * @version 0.1
 * @date 2024-05-08
 *
 * @copyright Copyright (c) 2024
 *
 */

data
{
  int<lower=1> N_pred; // Number of test time points (equally spaced)
  vector[N_pred] h;    // Vector of learned log volatility at time t
  real y_mean;         // Learned mean price
  real alpha;          // Learned params of exogenous regressors
  real beta_1;
  real beta_2;
  real beta_3;
  real gamma;
  real xi;
  vector[N_pred] Temperature_pred; // Vector of test Temperature at time
    t
  vector[N_pred] Weekday_pred;     // Vector of test Weekday at time t
}

generated quantities
{
  vector[N_pred] y_pred;
```

```
34
35  // Generate posterior predictive distribution
36  // We do not predict y_pred[t = 1],
37  // since we do not have y_pred[t - 1 = 0] and Temperature[t - 1 = 0]
38  // So predict the first value without Temperature and AR(1)
39  // and continue from index 2
40  y_pred[1] = normal_rng(y_mean + gamma * Weekday_pred[1],
41                         exp(h[1] / 2));
42  for (t in 2:N_pred)
43  {
44    y_pred[t] = normal_rng(y_mean +
45                           alpha * y_pred[t - 1] +
46                           beta_3 * pow(Temperature_pred[t - 1], 3) +
47                           beta_2 * pow(Temperature_pred[t - 1], 2) +
48                           beta_1 * Temperature_pred[t - 1] +
49                           gamma * Weekday_pred[t] +
50                           xi,
51                           exp(h[t] / 2));
52  }
53 }
```

Listing 4. SV X: Predict method