Post-COVID Stock Forecasting: An Investigation of Bayesian Methods in Machine Learning

Renyi Qu

Keio University

May 31, 2021

Table of Contents

Table of Contents

Background

- COVID-19: 2020/01-now
- deep interest in Bayesian methods during winter holiday
- investigate Bayesian integration in different (uncommon) forecasting methods:
  - **Regression**: GPR, SVR
  - **Neural networks**: BNN, LSTM
  - **Decision Tree Ensembles**: Random Forest, XGBoost
  - **New Bayesian**: BPS

Table of Contents

Data

4 datasets of major stock indices in the world from Yahoo Finance: Nikkei 225, NASDAQ, SSE Composite Index, HANG SENG Index.

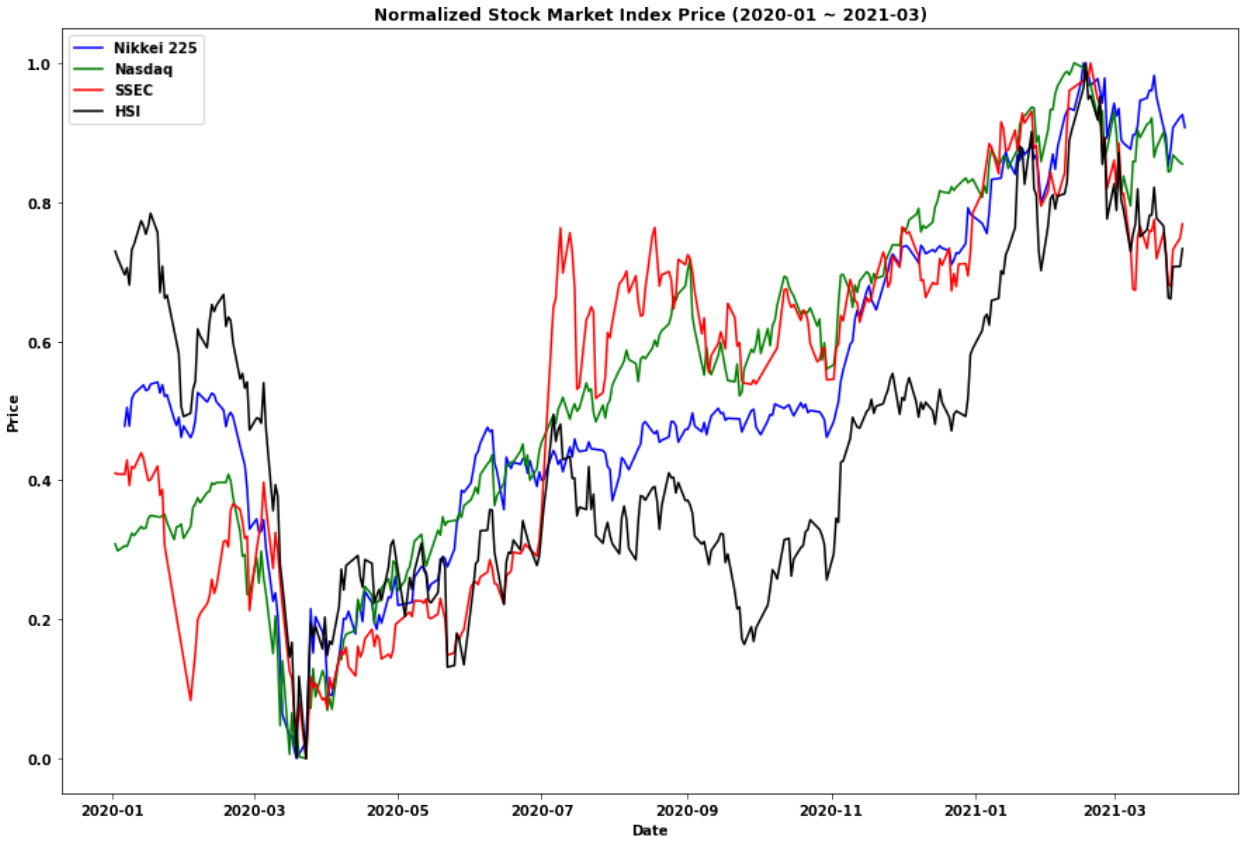


Figure 1: Normalized Stock Market Index Price (2020/01-2021/03)

Exploratory Data Analysis

Index	Original	1st-order difference	2nd-order difference
Nikkei225	-0.483	-10.457	-8.759
	(0.895)	(0.000)	(0.000)
Nasdaq	-0.746	-5.049	-12.555
	(0.834)	(0.000)	(0.000)
SSEC	-1.197	-16.587	-10.204
	(0.675)	(0.000)	(0.000)
HSI	-1.525	-11.783	-8.499
	(0.521)	(0.000)	(0.000)

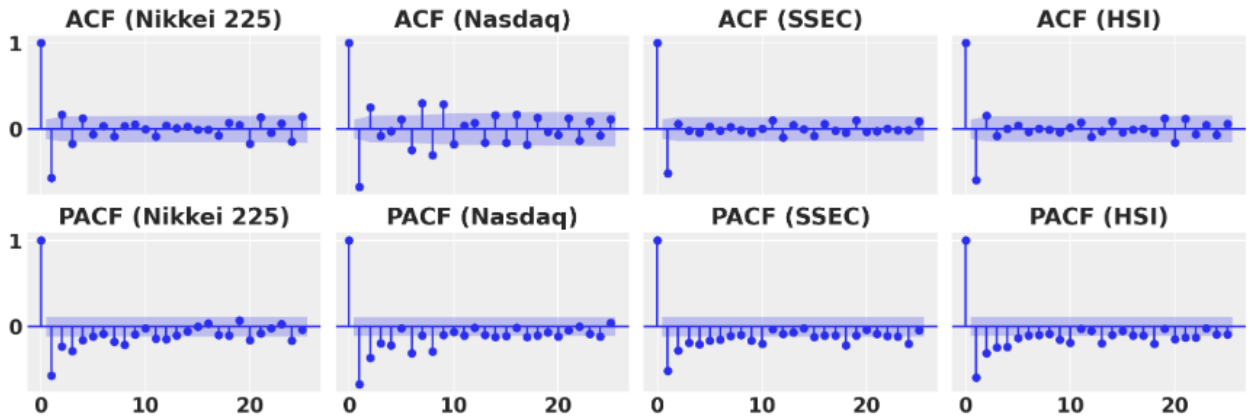


Figure 2: Exploratory Data Analysis

Table of Contents

Bayesian Inference

Bayes' Theorem:

$$p(\theta|x) = \frac{p(\theta)p(x|\theta)}{p(x)}$$

Bayesian inference:

$$\text{Posterior} \propto \text{Prior} \times \text{Likelihood}$$

1. Modeling: define prior & formulate likelihood
2. Computing: generate posterior

$$\text{MCMC} = \text{Markov Chain Sampling} + \text{Monte Carlo Integration}$$

Metropolis-Hastings Algorithm

Algorithm 1: Metropolis-Hastings algorithm

Initialize: $t = 0$, x_0, $g(x|y)$;

for t in $[1, t_{\max}]$ **do**

  Generate a random candidate $x \sim g(x|x_t)$;

  Calculate acceptance probability $a = \min\left(1, \frac{P(x)}{P(x_t)} \frac{g(x_t|x)}{g(x|x_t)}\right)$;

  Generate a Bernoulli indicator $u \sim \text{Bernoulli}(a)$;

  if u **then**

    $x_{t+1} = x$;

  else

    $x_{t+1} = x_t$;

  end

end

Hamiltonian Monte Carlo (HMC)

Replace the original posterior distribution with the Hamiltonian of the system:

$$H(x, v) = \ln P(x) + \ln Q(v)$$

- $\ln P(x)$: the potential energy of the system (i.e. underlying likelihood)
- $\ln Q(v)$: the kinetic energy of the system (i.e. random noise introduced by velocity v)

Establish a new acceptance probability:

$$a = \min\left(1, \frac{\exp\left(H(x_T, v_T)\right)}{\exp\left(H(x_0, v_0)\right)}\right)$$

No-U-Turn Sampler: an improvement of HMC that automatically optimizes integration time T, widely used in probabilistic programming packages.

Variational Inference

Approximate the posterior distribution $p(H|D)$ with $q(H)$ and make them as similar as possible. Such dissimilarity is measured by Kullback–Leibler (KL) divergence:

$$D_{\mathrm{KL}}(q\|p) = \int_H q(H') \ln \frac{q(H')}{p(H'|D)} dH'$$

The posterior $p(H'|D)$ is still in the equation $\rightarrow$ further derivation:

$$\begin{aligned} D_{\mathrm{KL}}(q\|p) &= \int_H q(H') \Big(\ln \frac{q(H')}{p(H', D)} + \ln p(D) \Big) dH' \\ &= \int_H q(H') \ln \frac{q(H')}{p(H', D)} dH' + \int_H q(H') \ln p(D) dH' \\ &= \mathbb{E}_q[\ln q(H) - \ln q(H, D)] + \ln p(D) \\ &= \ln p(D) - \mathcal{L}(q) \end{aligned}$$

Bayesian Optimization for Hyperparameter Tuning

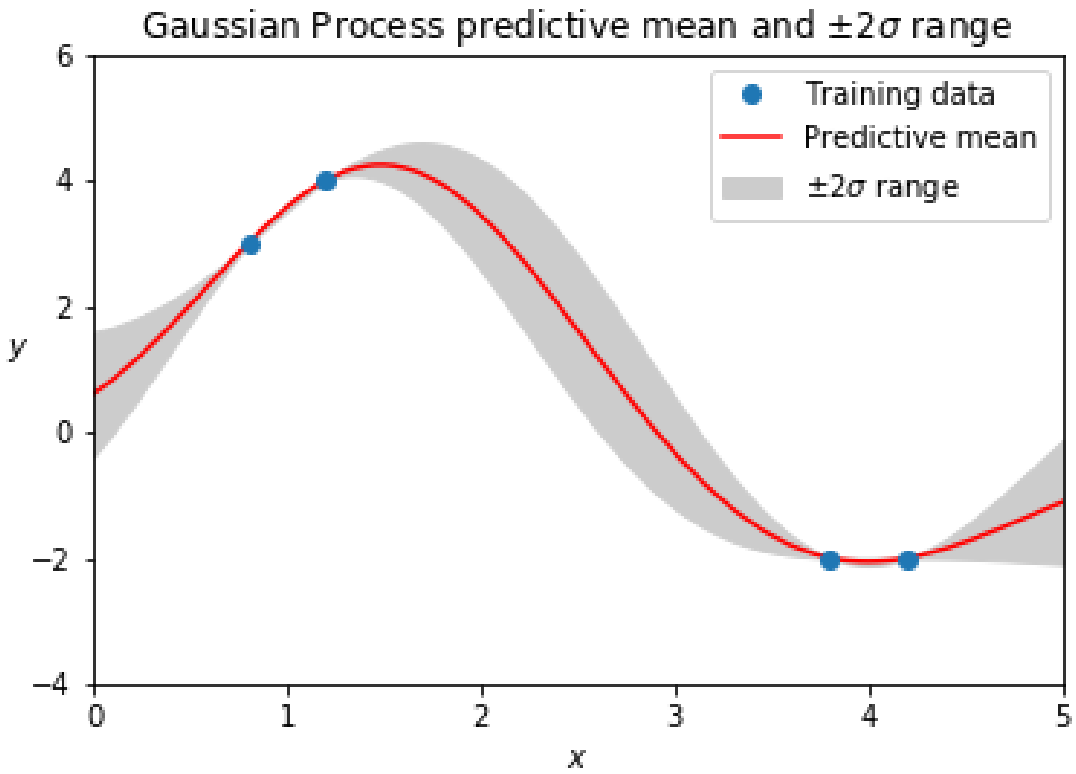
1. Define a hyperparameter space $\mathcal{H}$ for all hyperparameters of interest.
2. Initialize a sample set $\{\mathbf{h}_j, l_j\}_{j=1}^{N}$
3. Train a GPR based on the sample set, with a Gaussian prior $f(\mathbf{h})$ and the output $l_j \sim N(f(\mathbf{h}_j), v)$.
4. Identify the hyperparameter that minimizes the acquisition function:

$$a_{\text{LCB}}(\mathbf{h}; \{\mathbf{h}_j, l_j\}) = \mu(\mathbf{h}; \{\mathbf{h}_j, l_j\}) - \kappa\sigma(\mathbf{h}; \{\mathbf{h}_j, l_j\})$$

5. Add $\mathbf{h}_{\text{opt}} = \arg\min_{\mathbf{h}} a_{\text{LCB}}(\mathbf{h}; \{\mathbf{h}_j, l_j\})$ back to the sample set.
6. Repeat 3-5 until convergence.

Gaussian Process Regression

Gaussian Process Regression: non-parametric, no fixed functional form but a collection of normally distributed random functions.



Credit: https://planspace.org/20181226-gaussian_processes_are_not_so_fancy/

Gaussian Process Regression

Model:

$$f(x) \sim \mathcal{GP}(\mu(x), k(x, x'))$$
$$y \sim \mathcal{GP}(\mu(x), k(x, x') + \sigma_y^2 I)$$

Kernel choice: radial basis function (RBF):

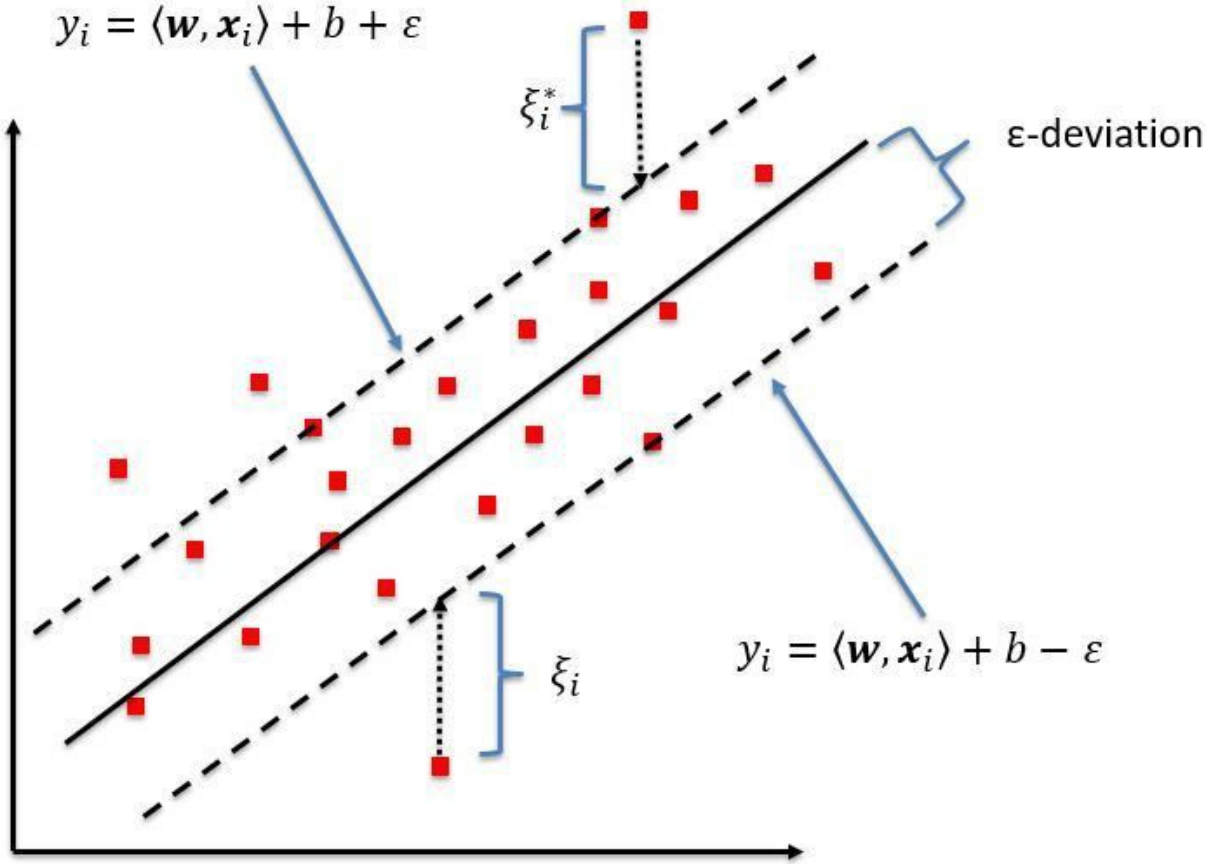
$$k(x, x') = \sigma_f^2 e^{-\frac{\|x - x'\|^2}{2l^2}}$$

Posterior Inference:

$$\begin{bmatrix} f(x) \\ f(x^*) \end{bmatrix} \sim N\left(\begin{bmatrix} \mu(x) \\ \mu(x^*) \end{bmatrix}, \begin{bmatrix} k(x, x') & k(x^*, x) \\ k(x^*, x) & k(x^*, x^{*'}) \end{bmatrix} \right)$$

Support Vector Regression

Support Vector Machine (SVM): construct a decision boundary with maximal geometric margin.
Support Vector Regression (SVR): SVM for regression.



Credit: `https://www.pinterest.es/pin/787285578600674679/`

Support Vector Regression

Objective function:

$$\arg\min_{\mathbf{w}} \left\{ \left[\frac{1}{2}\|\mathbf{w}\|^2 + C \sum_i |\xi_i| \right] + \sum_i \mathcal{L}(y_i - f(\mathbf{w}, x_i)) \right\}$$

Posterior Inference with constant hyperparameters:

$$p(f|D) \propto p(f)p(D|f)$$

$$p(f|D) \propto \exp\left\{ - C \sum_i \mathcal{L}(y_i - f(\mathbf{w}, x_i)) - \frac{1}{2} f^T k^{-1}(x, x') f \right\}$$

Bayesian Neural Network

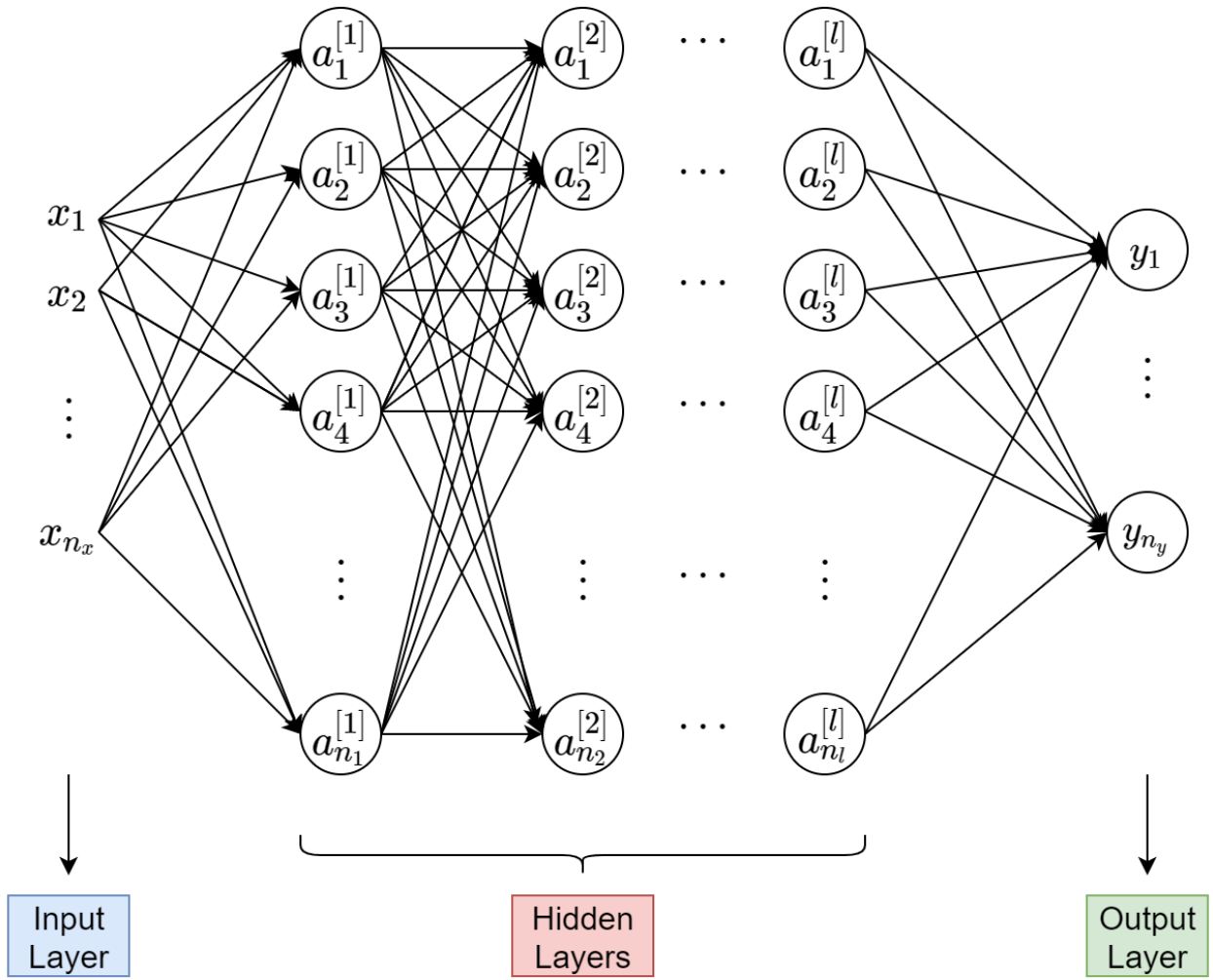


Figure 3: Neural Network

Bayesian Neural Network

Variational Posterior:

$$\mathbf{w} = \mu + \ln\left(1 + e^{\rho}\right) \circ \epsilon$$

ELBO:

$$f(\mathbf{w}, \theta) = \ln q(\mathbf{w}|\theta) - \ln p(\mathbf{w}p(D|\mathbf{w}))$$

Gradients:

$$\Delta_{\mu} = \frac{\partial f}{\partial \mathbf{w}} + \frac{\partial f}{\partial \mu}$$

$$\Delta_{\rho} = \frac{\partial f}{\partial \mathbf{w}} \frac{\epsilon}{1 + e^{-\rho}} + \frac{\partial f}{\partial \rho}$$

Gradient descent on the variational parameters:

$$\mu \leftarrow \mu - \alpha \Delta_{\mu}$$

$$\rho \leftarrow \rho - \alpha \Delta_{\rho}$$

LSTM

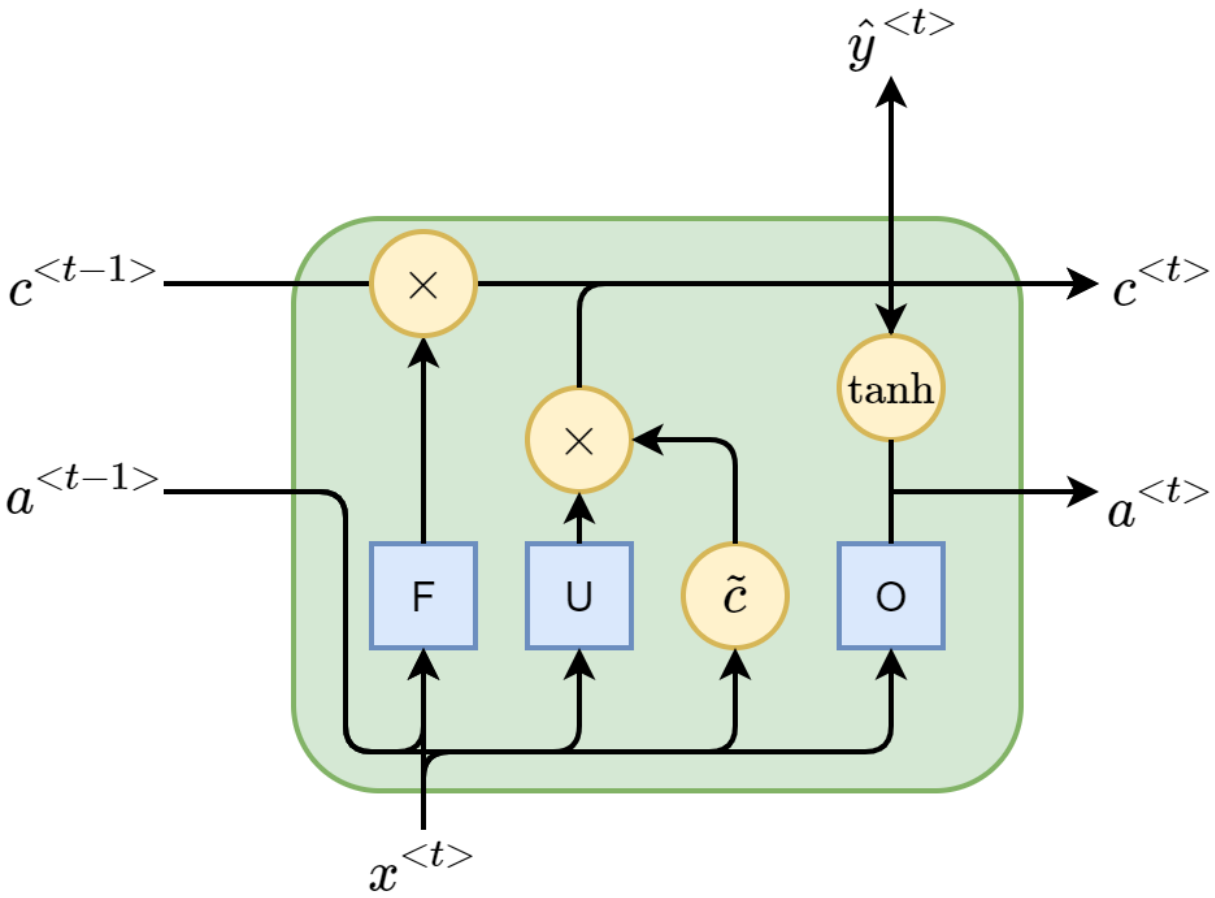


Figure 4: Long Short-Term Memory

LSTM

1. Compute F gate: $\Gamma_f = \sigma(W_f[a^{<t-1>}; x^{<t>}] + b_f)$
2. Compute U gate: $\Gamma_u = \sigma(W_u[a^{<t-1>}; x^{<t>}] + b_u)$
3. Compute O gate: $\Gamma_o = \sigma(W_o[a^{<t-1>}; x^{<t>}] + b_o)$
4. Compute candidate: $\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}; x^{<t>}] + b_c)$
5. Compute memory cell: $c^{<t>} = \Gamma_u \tilde{c}^{<t>} + \Gamma_f c^{<t-1>}$
6. Output activation for the next unit: $a^{<t>} = \Gamma_o \cdot \tanh c^{<t>}$

Random Forest

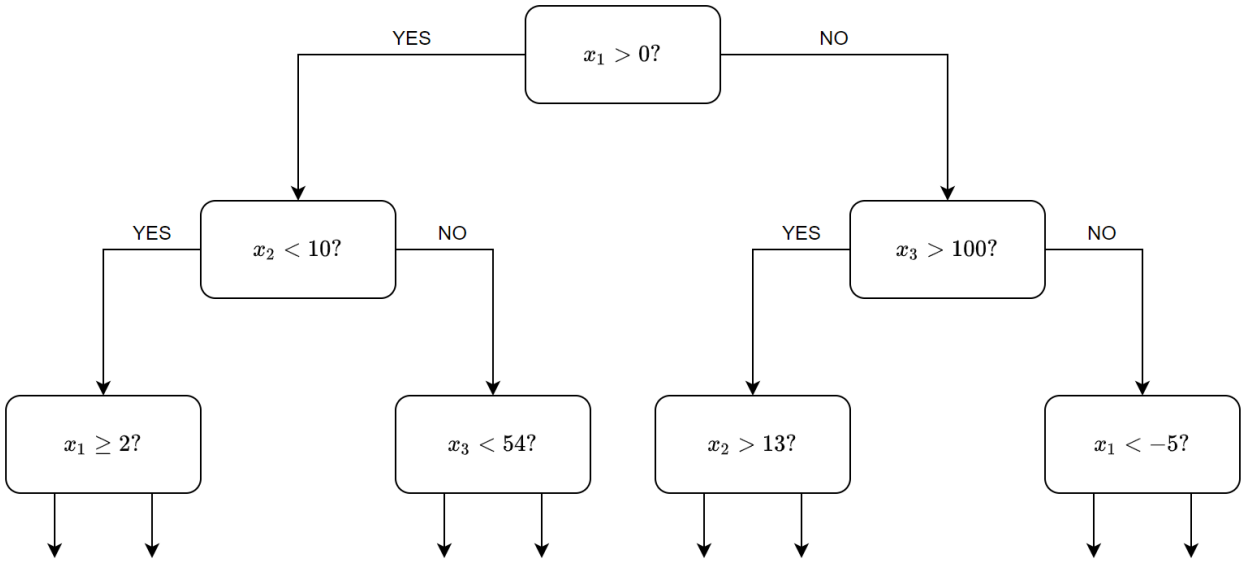


Figure 5: Example of a decision tree

Random Forest = a bag of multiple decision trees

$$\hat{f} = \frac{1}{B} \sum_{b=1}^{B} f_b(x^*)$$

XGBoost

Idea:

$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)$$

Objective function to be optimized:

$$\sum_{i=1}^{m} \left(g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right) + \Omega(f_t)$$

- $g_i = \frac{\partial \mathcal{L}}{\partial \hat{y}_i^{(t-1)}}$: first-order gradient ($\mathcal{L}(y_i, \hat{y}_i^{(t-1)})$: original loss function)
- $h_i = \frac{\partial^2 \mathcal{L}}{\partial \hat{y}_i^{(t-1)^2}}$: second-order gradient
- $\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$: regularization term

Bayesian Predictive Synthesis

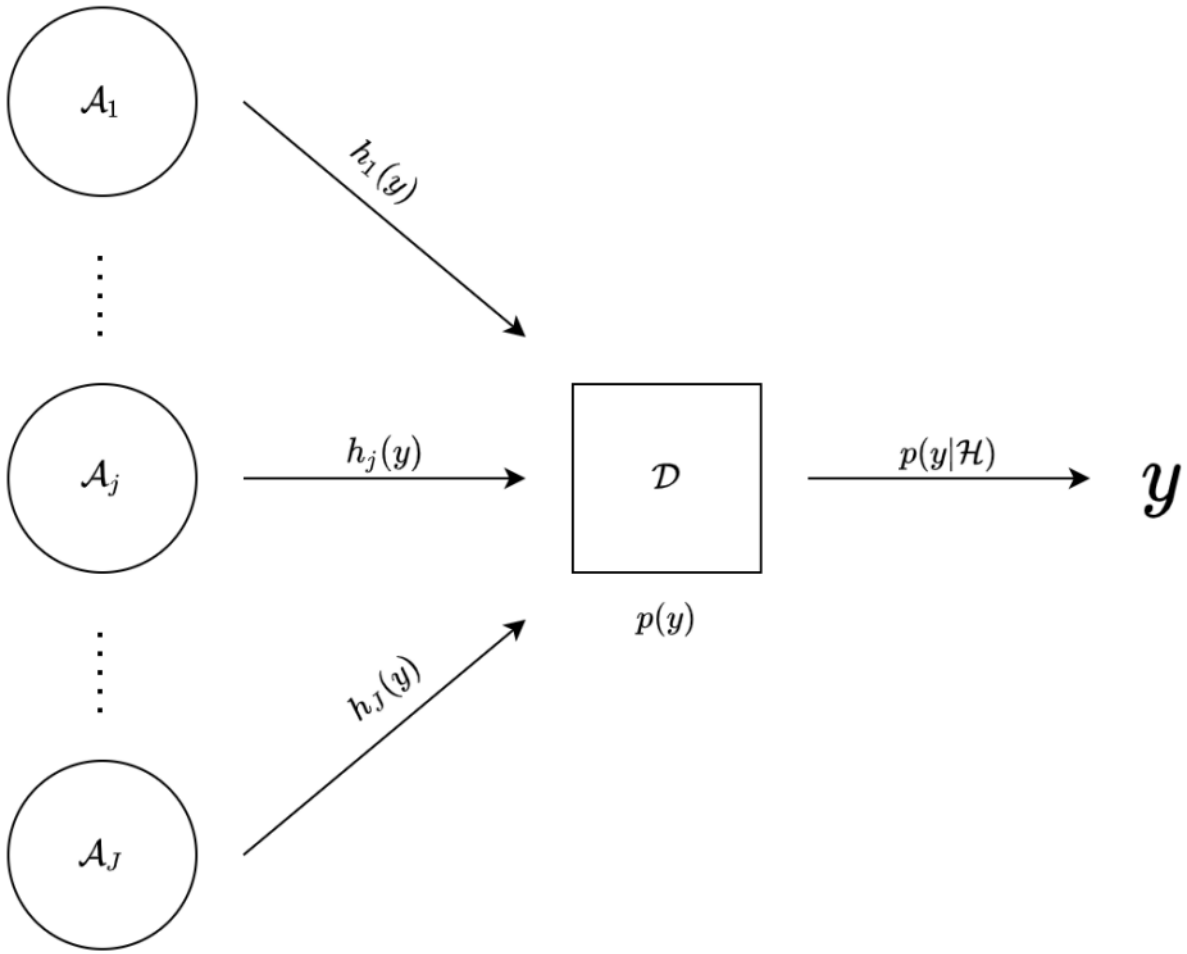


Figure 6: Bayesian Predictive Synthesis

Bayesian Predictive Synthesis

Original BPS:

$$p(y|\mathcal{H}) = \int_{\mathbf{x}} \alpha(y|\mathbf{x}) \prod_{j=1}^{J} h_j(x_j) d\mathbf{x}$$

Temporal BPS:

$$p(y|\mathbf{\Phi}_t, \mathcal{H}_t) = \int \alpha_t(y_t|\mathbf{x}_t, \mathbf{\Phi}_t) \prod_{j=1}^{J} h_{tj}(x_{tj}) d\mathbf{x}_t$$

Prior:

$$p(\mathbf{x}_t|\mathbf{\Phi}_t, y_{1:t-1}, \mathcal{H}_{1:t}) \equiv p(\mathbf{x}_t|\mathcal{H}_t) = \prod_{j=1}^{J} h_{tj}(x_{tj})$$

Likelihood:

$$\alpha_t(y_t|\mathbf{x}_t, \mathbf{\Phi}_t) = N(y_t|\mathbf{F}_t^T \boldsymbol{\theta}_t, v_t)$$

Bayesian Predictive Synthesis

Likelihood - DLM:

$$y_t = \mathbf{F}_t^T \boldsymbol{\theta}_t + \nu_t, \quad \nu_t \sim N(0, v_t)$$
$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \boldsymbol{\omega}_t, \quad \boldsymbol{\omega}_t \sim N(0, v_t \mathbf{W}_t)$$

- $\mathbf{F}_t = (1, \mathbf{x}_t^T)^T$
- $\boldsymbol{\theta}_t = (\theta_{t0}, \cdots, \theta_{tJ})^T$
- $\boldsymbol{\Phi}_t = (\boldsymbol{\theta}_t, v_t)$

Posterior inference:

1. Draw parameters from $p(\boldsymbol{\Phi}_{1:t} | \mathbf{x}_{1:t}, y_{1:t})$ by FFBS.
2. Draw agent states from $p(\mathbf{x}_{1:t} | \boldsymbol{\Phi}_{1:t}, y_{1:t}, \mathcal{H}_{1:t})$ by MH.

Table of Contents

Experiment

- Python:
  - GPR: $k_{\text{GPR}} = k_{\text{noise}} + k_{\text{const}} \times k_{\text{RBF}}$
  - BNN: 8×16×8, ReLU, MSE, Adam, epoch 10, batch size 32
  - LSTM: 1→16, ReLU, MSE, Adam, epoch 10, batch size 32
  - Random Forest: max_depth (in [3,10]), min_samples_leaf (in [1,4]), min_samples_split (in [2,10]), n_estimators (in [100,200])
  - XGBoost: max_depth (in [3,10]), gamma (in [0,1]), colsample_bytree (in [0.1,0.9]), eta (in [0.1,0.5]), and subsample (in [0.5,1])
- Julia:
  - SVR: RBF with scaled l, $C = 1000, \epsilon = 10^{-5}$
  - BPS: monthly average prices instead of daily prices, $J = 4$, Gaussian priors, burn-in $= 1000$, MCMC $= 2000$.

Table of Contents

Results

Table 3: CI Inclusion Accuracy Results (in %)

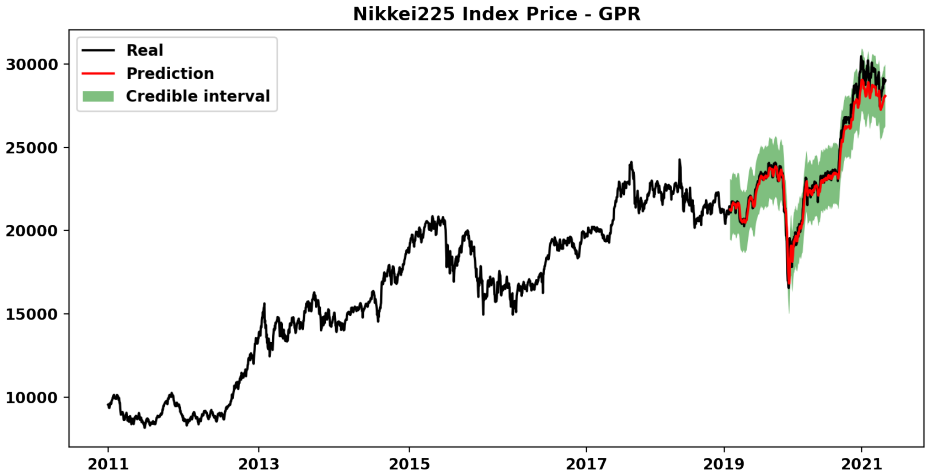
	GPR	SVR	BNN	LSTM	BPS
Nikkei225	**99.58**	69.05	29.07	67.01	**100.00**
NASDAQ	**63.62**	28.23	17.84	48.69	**100.00**
SSEC	**99.81**	99.40	94.19	97.30	**99.46**
HSI	**99.79**	99.38	77.00	77.82	**99.89**

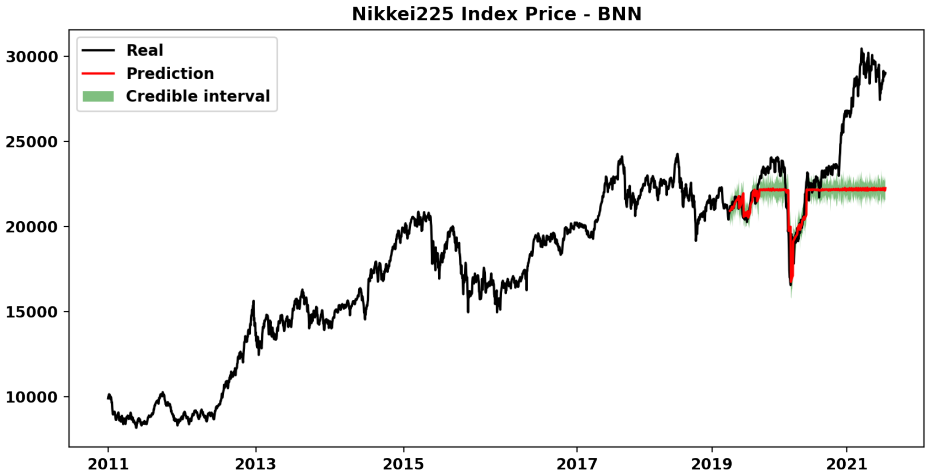
Results

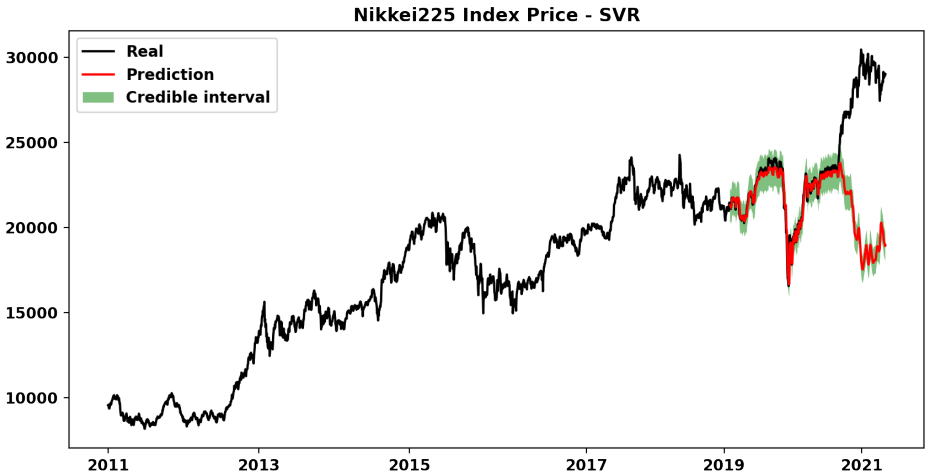
Table 4: RMSE Results

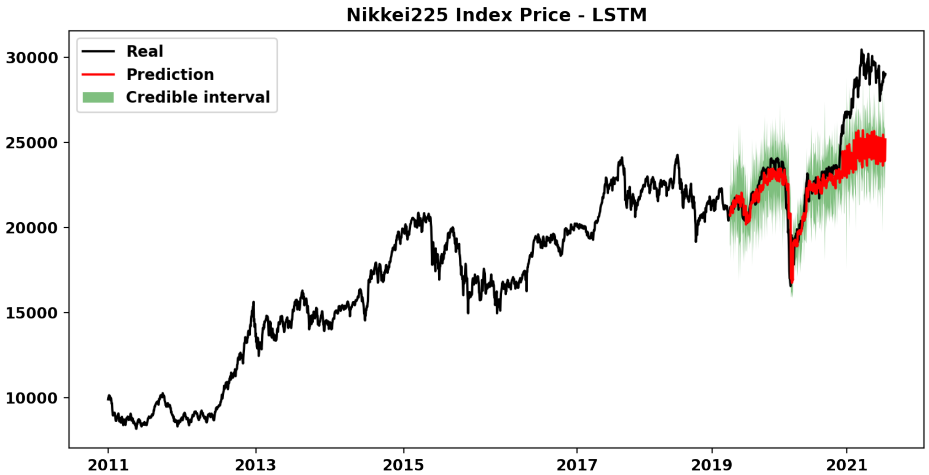
	GPR	SVR	BNN	LSTM	RF	XGBoost	BPS
Nikkei225	**599.22**	4887.60	3506.62	2368.35	2464.90	2445.03	**109.01**
NASDAQ	**872.53**	4932.07	3673.23	2890.25	3107.07	3118.09	**54.82**
SSEC	38.29	**35.63**	108.30	92.66	44.43	43.58	**18.56**
HSI	353.16	**341.44**	1001.96	778.42	388.46	385.76	**108.53**

Nikkei225 Forecast

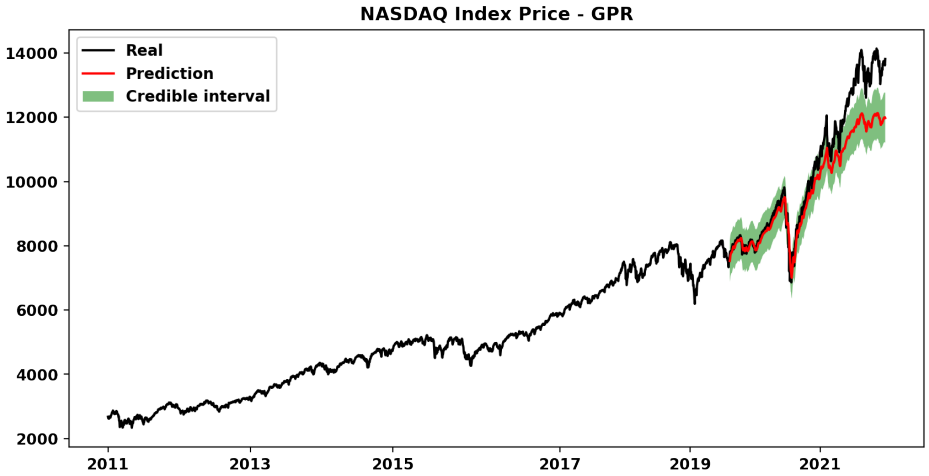


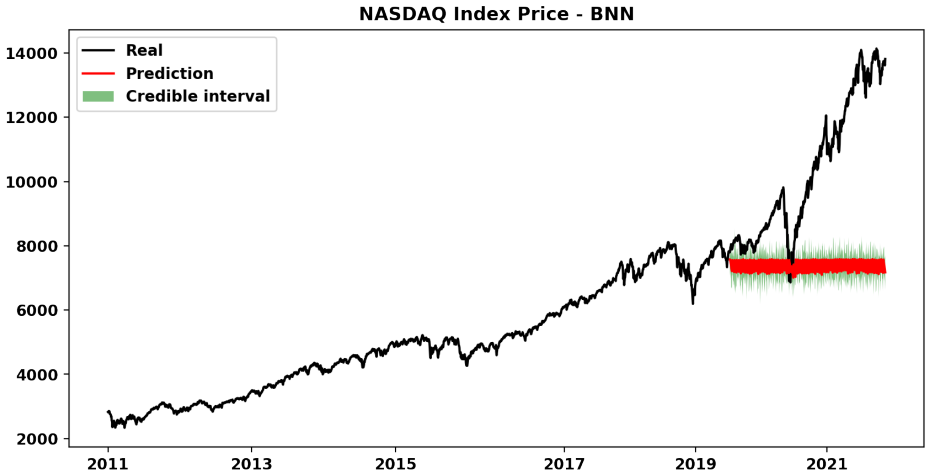


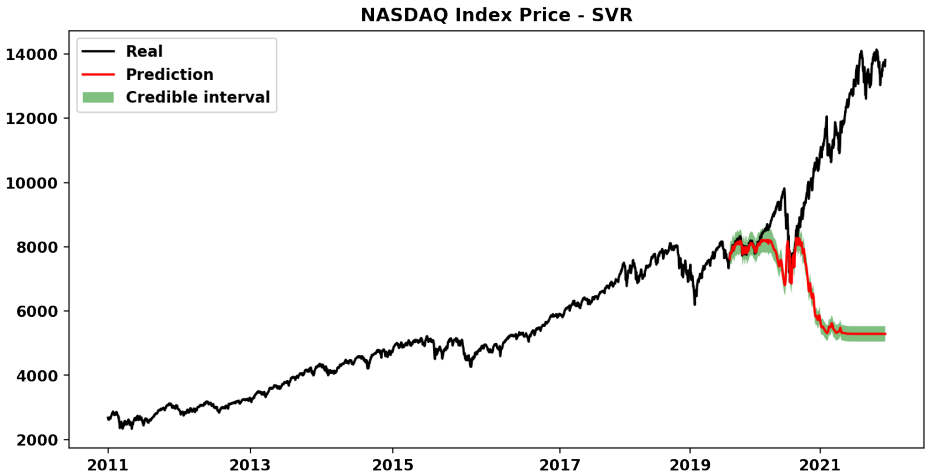


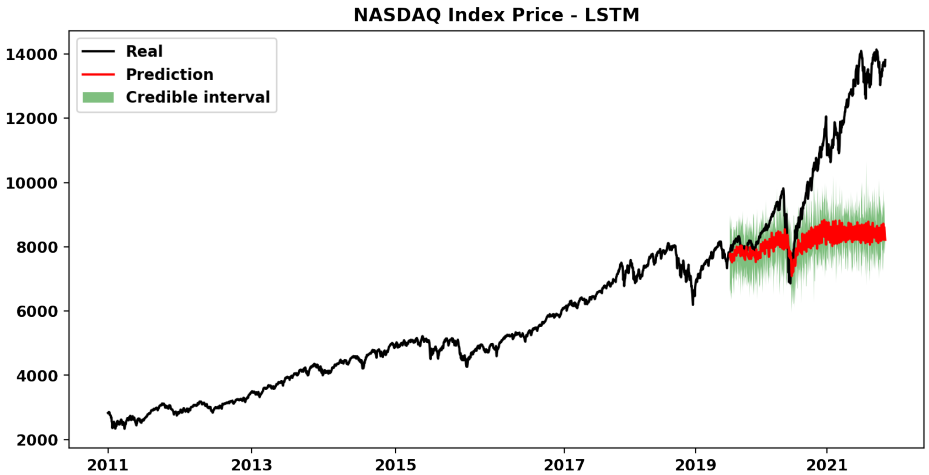


NASDAQ Forecast

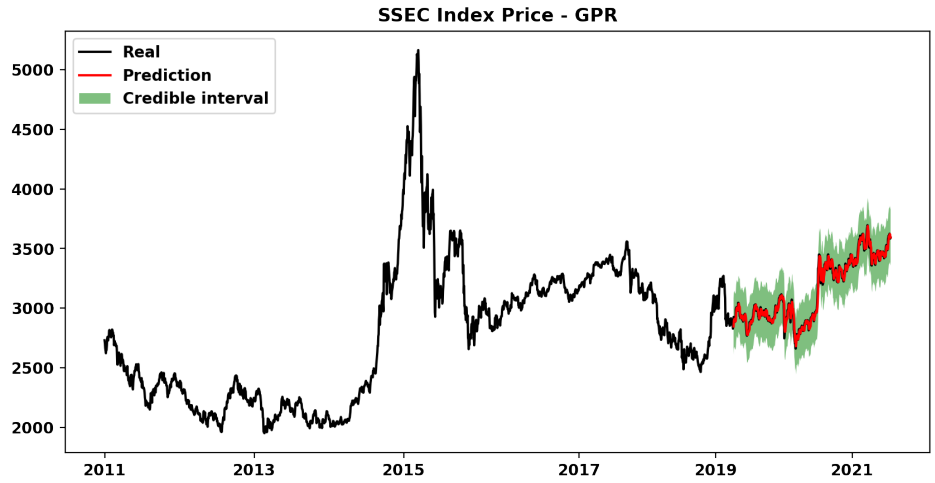


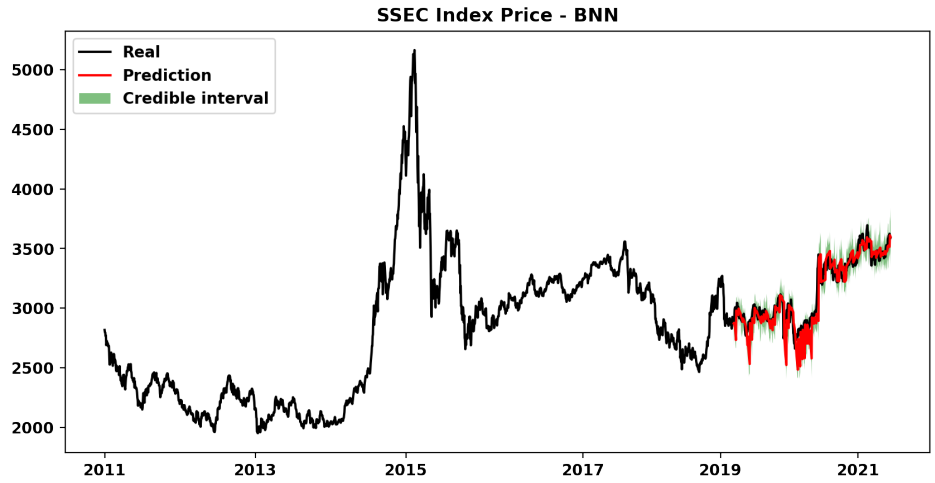


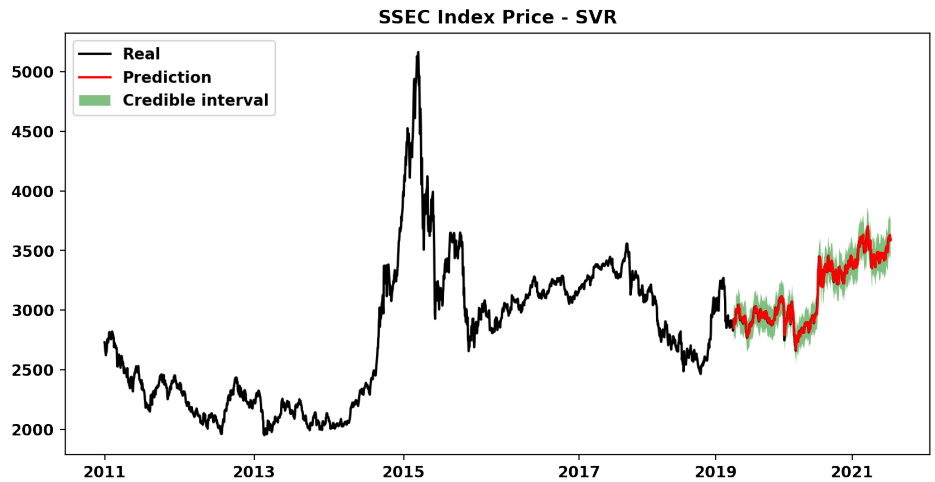




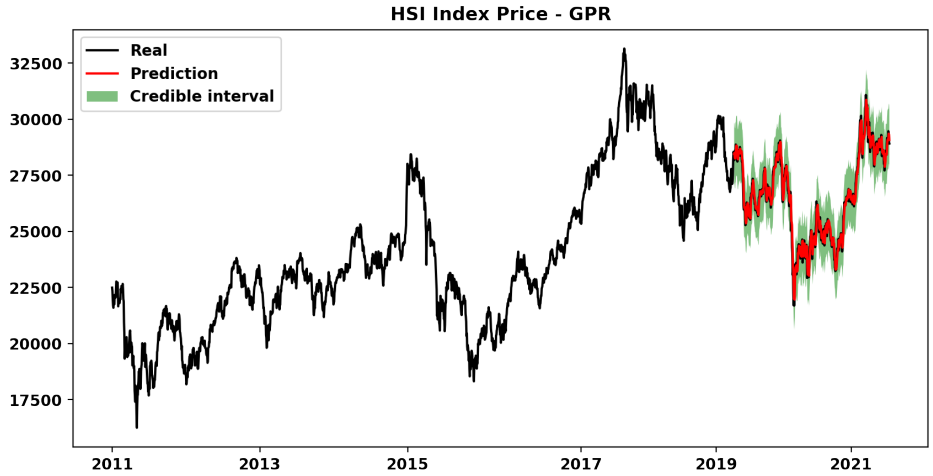
SSEC Forecast

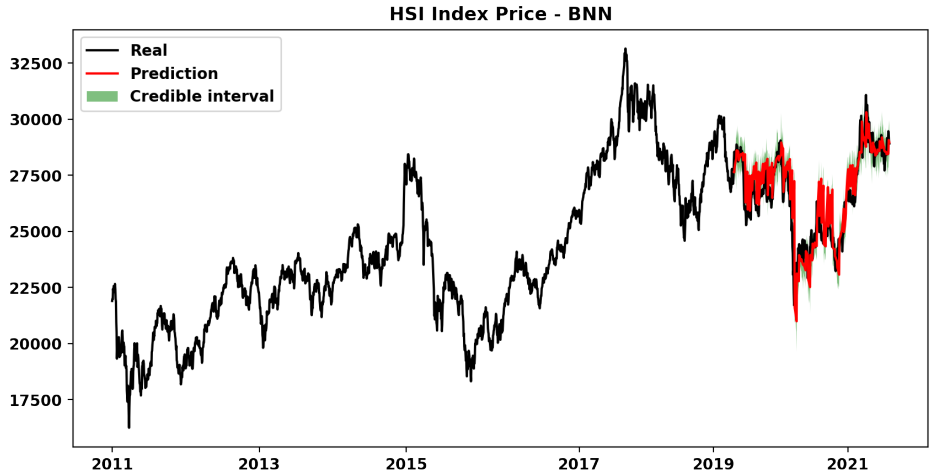


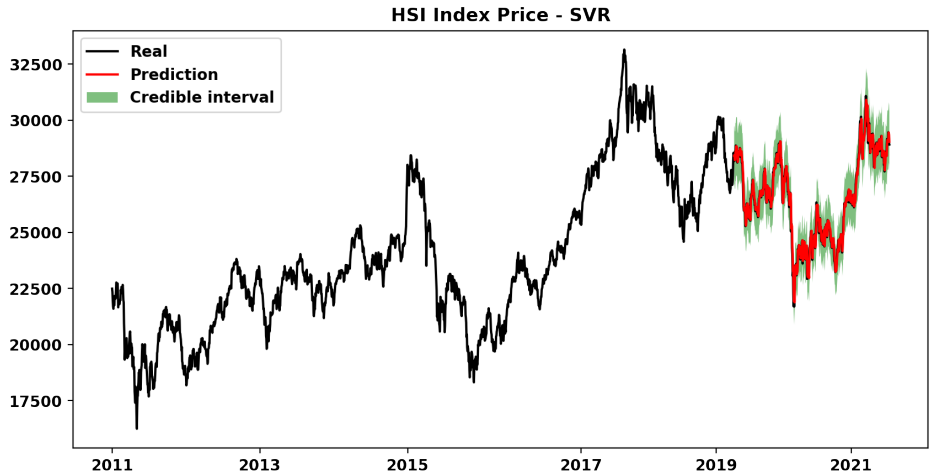


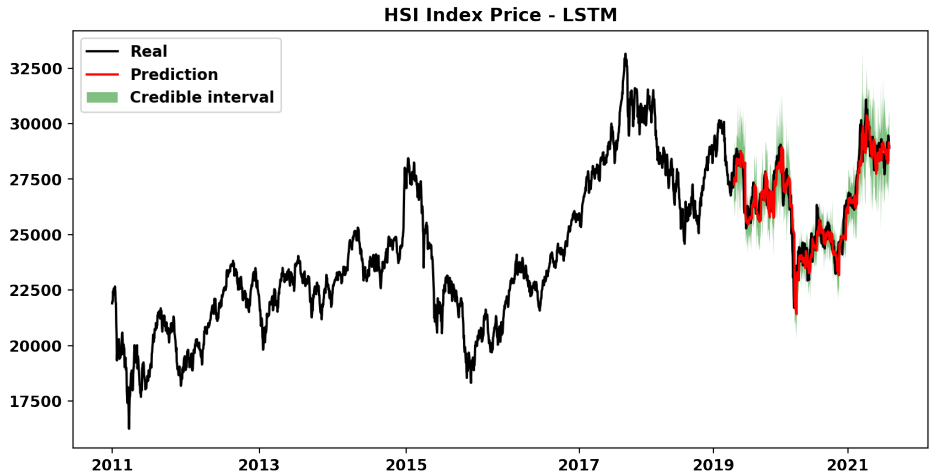


HSI Forecast









Optimal Hyperparameters

Table 5: Optimal Hyperparameters for Random Forest

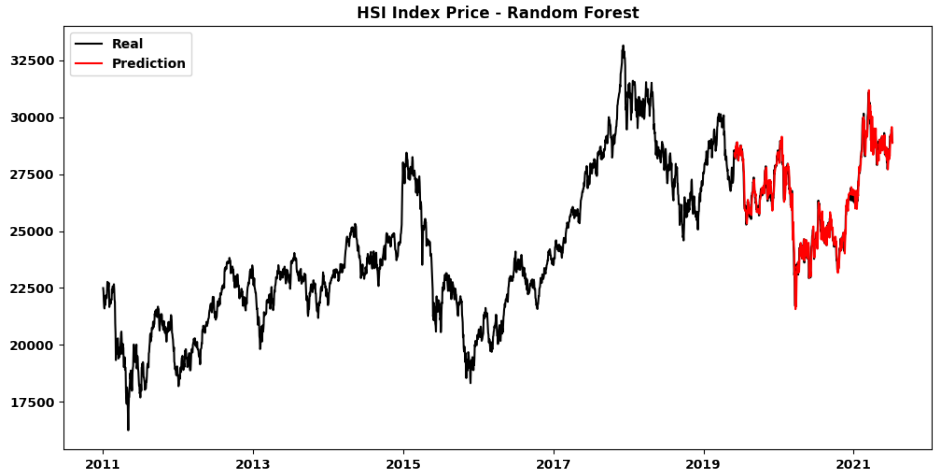
	max_depth	min_sample_leaf	min_sample_split	n_estimators
Nikkei225	10	1	2	100
NASDAQ	10	1	2	100
SSEC	10	1	5	190
HSI	10	1	5	139

in Nonlinear Dynamics and Econometrics

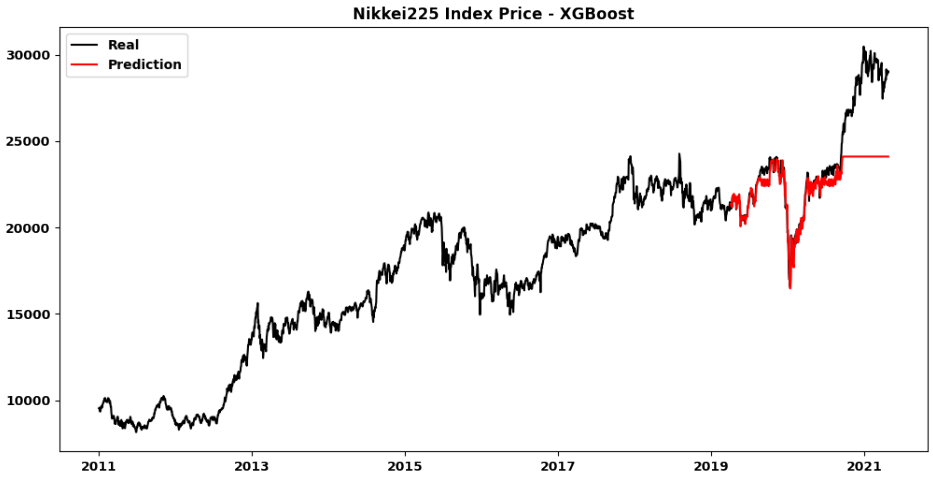
Table 6: Optimal Hyperparameters for XGBoost

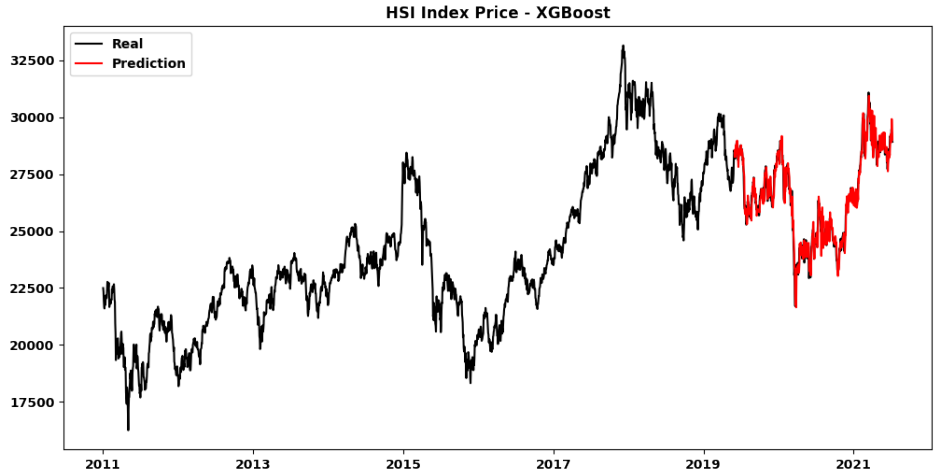
	max_depth	gamma	colsample_bytree	eta	subsample
Nikkei225	3	0.37	0.90	0.1	0.8
NASDAQ	6	0.06	0.90	0.1	0.8
SSEC	3	1.00	0.90	0.1	0.8
HSI	3	0.64	0.83	0.1	0.8

Forecasts from Random Forest

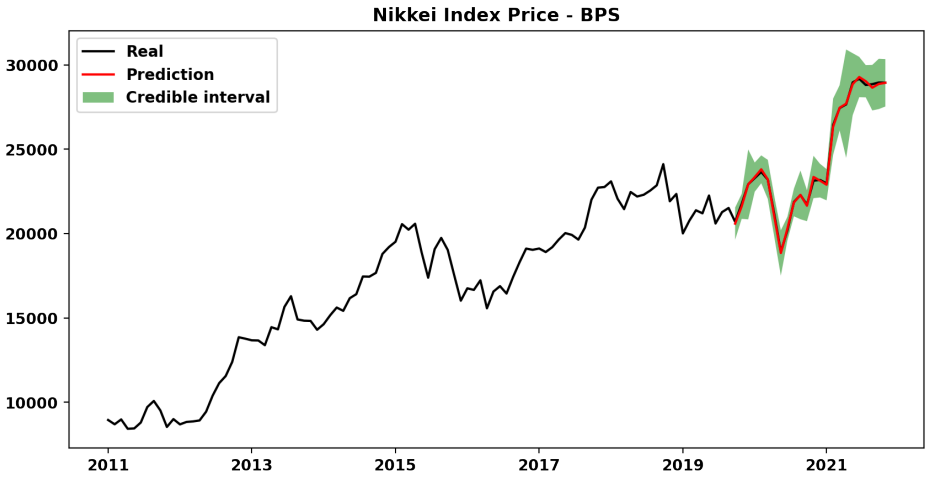


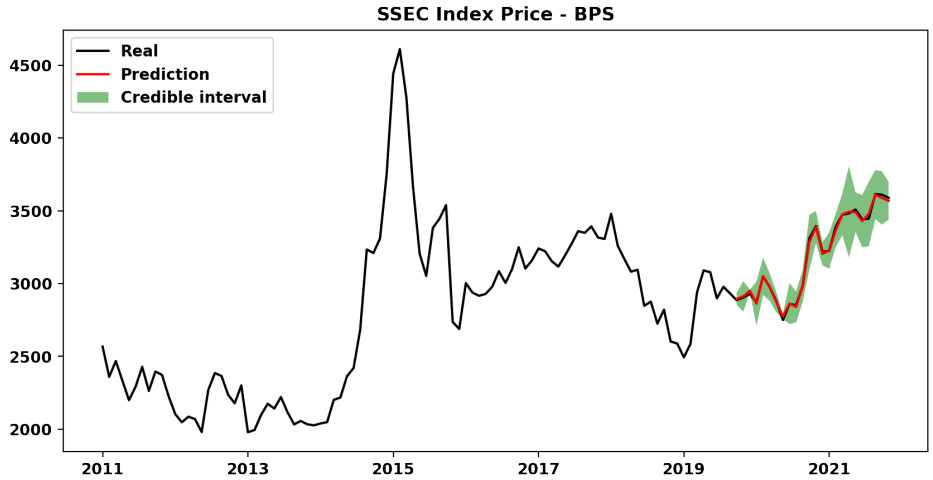
Forecasts from XGBoost

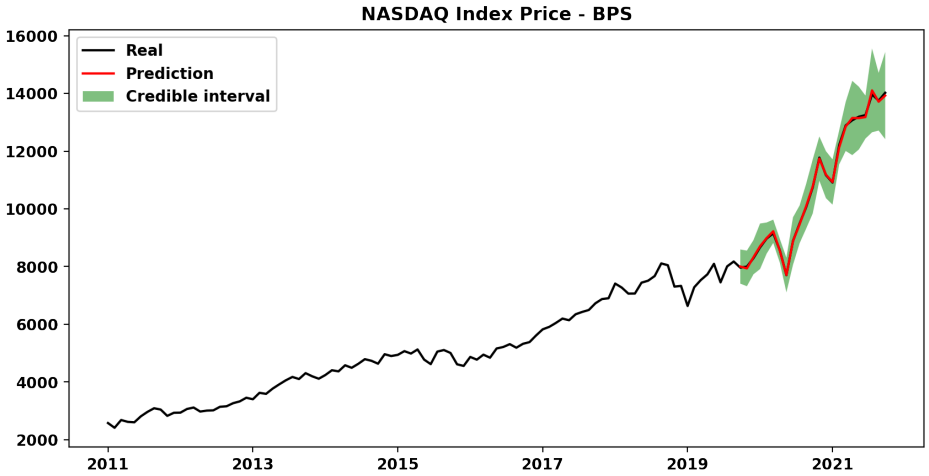




Forecasts from BPS







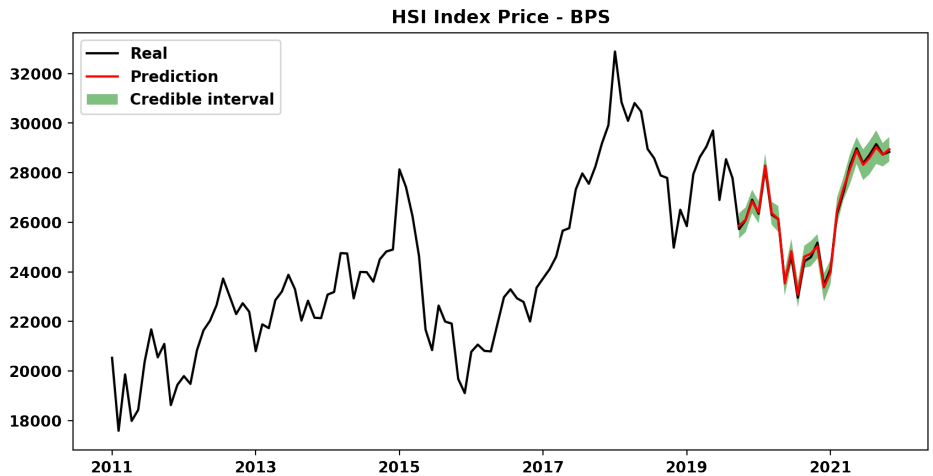


Table of Contents

Conclusion

1. A good hardware condition is necessary for a successful implementation of advanced Bayesian methods.
2. Pure Bayesian methods seem to be the best fit for time series forecasting tasks.
3. Complexity in the model structure does not necessarily improve the performance in time series forecasting.
4. Bayesian optimization for hyperparameter tuning is beneficial for time series forecasting models.