Evaluating the Performance of Bayesian Treed Gaussian Process Models for Predicting the Velocity of Ice in the Arctic Sea-Ice Computer Model

Nirupama Tamvada

April 21, 2022

Abstract

Stationary Gaussian Processes (GP's) have become a popular standard in a computer model framework due to their many desirable properties. However, for many real-life applications, the stationarity assumption can often be restrictive, particularly when it comes to modelling discontinuities, as well as accounting for differing regional variability in the parameter space. Thus, Gramacy and Lee (2008) proposed relaxing the stationarity assumption in a computationally scalable manner, by partitioning the input space through a tree-based approach, and subsequently fitting stationary Bayesian Gaussian Processes within each partition region. In this report, we apply four such Bayesian tree-based approaches: the classification and regression tree, random forests, treed Gaussian Processes and treed Gaussian Processes with a limiting linear model comparison to a stationary Gaussian Process. We do so to the end of predicting the ice velocity in the Arctic Sea-Ice Computer Model, which is speculated to be difficult to predict due to the numerical model displaying differential variability in some regions of the parameter space. Overall, we found that while the RMSEs of the linear treed approaches (random forests and treed limiting linear model GP's) were marginally lower than that of the stationary GP, the treed GP approaches did not result in a dramatic increase in prediction accuracy. The treed limiting linear model GP had the lowest RMSE compared to both the stationary GP and the treed GP. The Random forests algorithm also had a relatively low RMSE for predicting ice-velocity. Thus, while ice-velocity remains difficult to predict, the treed limiting linear model GP may be a better and less-computationally intensive model choice to use for sensitivity analysis compared to a stationary GP.

Introduction

In a computer model framework, using a Gaussian Process (GP) is a popular choice. This is because GPs are conceptually straightforward and satisfy several desirable properties: predictive uncertainty can be quantified, prior knowledge can be easily accounted for in the form of covariance functions and Bayesian approaches, a GP conditionally to given data (equality constraints) is also a GP, and the partial derivatives of a GP are also GPs [8, 5]. A standard approach in the literature for emulation is the use of a stationary smooth Gaussian Process [5]. However, there are some particular disadvantages of using this standard form of a GP. In particular, the stationarity assumption, which leads to the usage of the same covariance structure being used throughout the space, often does not scale well to a wide range of applications. This assumption can be too restrictive when it comes to modelling discontinuities, and sharp changes of the function surface [5]. Importantly, the estimated predictive error of a stationary GP only depends on a global measure of error that uses all of the discrepancies between observations and predictions [5]. However, it is often the case that some regions of the space exhibit larger variability compared to others [5]. In many applications, it is more desirable to model the uncertainty over the output space in a non-uniform manner, so that the predictive error also depends on the locally observed response values [5].

Thus many real-world applications require the use of more flexible, non-stationary Gaussian processes. However, fully non-stationary Bayesian Gaussian Processes can be difficult to fit and do not scale well computationally to a large number of data-points [5]. Gramacy and Lee (2008) address this problem by proposing partitioning the input space into regions. A stationary Gaussian Process can then fit within each region. Such a partitioning is attractive, as if offers a straightforward mechanism for creating a less computationally-demanding non-stationary Gaussian Process [5]. Under a Bayesian framework, the predictive uncertainty can also be estimated by applying a Bayesian model averaging approach. This approach is capable of flexibly modelling both continuous functions, as well as discontinuities [5]. It additionally also resolves the variability issue, as the estimated predictive error estimated can now account for differing variability between regions of the space [5].

In particular, Gramacy and Lee (2008) apply a treed partitioning approach. A Classification and Regression Tree (CART), which which fits a constant surface in each leaf, is used as it is easy to implement and interpret. Their approach involves applying a tree-generating prior to a CART. A Gaussian Process (or any suitable model) is then applied to each partition/leaf of this tree, with hierarchical priors specified for the hyper-parameters [5]. Bayesian model-averaging is then finally applied to average over in the posterior in order to make predictions and estimate predictive uncertainty [5]. This approach was applied to two applications: a computer model simulating the launch of NASA's Langley Glide-Back rocket booster, as well to a real-life application for predicting the acceleration of the head of a motorcycle rider as a function of time [5]. In both cases, the treed Bayesian Gaussian Process showed a superior performance compared to a stationary Gaussian Process, as it was able to model discontinuities (particularly those those by numerical instability of the simulation as well) and account for for the differing levels of uncertainty in the space [5].

On the usefulness of computer models, sea-ice models have become increasingly useful in the study of the high-latitude climate system [2]. The Arctic Sea-Ice Computer Model in particular, offers valuable information on the effect of ice physics parameters such as drag coefficients, and snowfall rate on outputs such as ice-thickness that are difficult to quantify [2]. While stationary Gaussian Processes have shown good performance in modelling specific outputs such as ice area, they showed a lesser predictive accuracy in modelling ice velocity in particular [2]. Chapman, Welch, Bowman, Sacks and Walsh (1994) speculated that this may be due to the numerical model being erratic in some regions of the parameter space, which may have reduced predictive power. Given the flexibility of the treed Bayesian Gaussian Process in modelling discontinuities and in accounting for differing variability in the parameter space, it was of interest to see if tree-based approaches can significantly improve the prediction accuracy of ice velocity in the Arctic Sea-Ice Computer Model.

Specifically, the objective of this report is to compare the performance of a Bayesian stationary Gaussian Process to four tree-based, Bayesian methods: Classification and Regression Trees, Random Forests, Treed Gaussian Processes, and Treed Gaussian Processes with Limiting Linear Regression Models, to the end of potentially improving the predictive accuracy of ice velocity in the Arctic Sea-Ice Computer Model.

Methods

Computer Model

The Arctic Sea-Ice computer model is a dynamic formulation based on a momentum balance for a mass of ice within a grid cell. The specific details of the thermodynamic formulation can be found in [2]. The purpose of this computer model is to estimate sensitivities of ice physics input parameters on outputs that are key sea ice properties [2]. The model was run with a daily time-step from the period of January 1, 1960-1988. It was run on a 110kn polar grid, which covers the Arctic Ocean and nearby bodies of water. Specifically, the data-set has 6 output parameters: Ice mass, Ice area, Ice velocity and Range of Area. For this project, we focus on the Ice velocity variable, which has been shown to be comparatively harder to predict accurately using a stationary GP. The model has 13 ice physics input parameters. These are as follows:

- **Drag coefficients:** These describe the drag forces exerted on ice by air (`AtmosDrag`) and ocean currents (`OceanicDrag`).
- **Ice Strength:** This measure is a function of the grid cell ice thickness and compactness (`LogIceStr`). It has been log-transformed in the data
- **Minimum lead fraction:** Permits the inclusion of the effect of small-scale motions within a grid cell. These motions tend to maintain a small fraction (several percent) of open water or very thin ice within the pack even during winter, which as most of the ocean-atmosphere exchanges of sensible and latent heat, as well as new ice growth and salt rejection ocur in these regions (`MinLead`)
- **Albedos:** Measure of the diffuse reflection of solar radiation for snow (`SnowAlbedo`), ice (`IceAlbedo`) and open-water (`OpenAlbedo`)
- **Exchange coefficient, surface sensible heat:** Bulk transfer coefficient for sensible heat (`SensHeat`)
- **Exchange coefficient, surface latent heat:** Bulk transfer coefficient for latent heat (`LatentHeat`)
- **Snowfall rate:** Snowfall at prescribed rates that vary seasonally (`Snowfall`)
- **Cloud depletion of solar flux:** . Downcoming fluxes of solar radiation which vary from the winter to the summer (`Shortwave`)

- **Cloud enhancement of longwave flux:** Downcoming fluxes of longwave radiation which vary from the winter to the summer (`Longwave`)
- **Oceanic heat flux:** prescribed vertical flux of oceanic heat into the mixed layer, which can determine the rate of ice growth and melt (`OceanicHeat`)

Stationary Gaussian Processes

A real-valued stochastic process $\{X_t, t \in T\}$ where X_t is a set of random variables indexed by T is a Gaussian Process having a jointly Gaussian distribution for a finite subset of indices if all the finite-dimensional distributions have a multivariate normal distribution [5]. A Gaussian Process is specified by a mean function $\mu(\mathbf{x}) = \mathbb{E}(Z(\mathbf{x}))$, and a correlation function $R(\mathbf{x}, \mathbf{x}')$ [5].

We assume the correlation function can be written as follows:

$$R(\boldsymbol{x_j}, \boldsymbol{x_k}|g) = R^*(\boldsymbol{x_j}, \boldsymbol{x_k}) + g\delta_{j,k} \tag{1}$$

where $R^*(\boldsymbol{x_j}, \boldsymbol{x_k})$ represents a parametric correlation function family, $\delta_{j,k}$ represents the Kronecker delta function and g is the nugget term [5].

For this project, we consider $R^*(\boldsymbol{x_j}, \boldsymbol{x_k})$ to represent the power family of correlation functions.

$$R^*(\boldsymbol{x_j}, \boldsymbol{x_k}|d) = \exp\left\{\frac{\|\boldsymbol{x_j} - \boldsymbol{x_k}\|^{p_0}}{d}\right\} \tag{2}$$

where $d > 0$ represents a single range parameter. This constant correlation structure is used for a stationary Gaussian Process. Thus, the correlation between two points only depends on the Euclidean distance between them [5].

The linear predictive distribution of this Gaussian Process is univariate normal, with mean

$$f^T(\boldsymbol{x}^*)\boldsymbol{\beta} + r^T(\boldsymbol{x}^*)\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{F}\boldsymbol{\beta}) \tag{3}$$

and variance

$$\sigma^2(1 - \boldsymbol{r}^T(\boldsymbol{x}^*)\boldsymbol{R}^{-1}\boldsymbol{r}(\boldsymbol{x}^*)) \tag{4}$$

Thus, the predictive uncertainty does not depend directly on the locally observed values [5]. Rather, the predictive error at a point depends only a global measure of error that uses the distance between observations and predictions, without regard for their distance from their location in the parameter space [5].

We implement this stationary Gaussian Process in a Bayesian framework (the prior specification is described below), as a baseline to compare our treed methods to.

Extending the Power Correlation Function for Treed Partitioning

The power correlation function specified in 2 can be trivially modified for the partitioned applications to be implemented in this project by using a separate range parameter d_i for each dimension $(i = 1, ..., m_x)$ as follows:

$$R^*(\boldsymbol{x_j}, \boldsymbol{x_k}|d) = \exp\left\{\frac{\|\boldsymbol{x_j} - \boldsymbol{x_k}\|^{p_0}}{d_i}\right\} \tag{5}$$

This allows the covariance structure to vary over the parameter space by varying between the partition region [5]. One can therefore model correlations in some input variables as stronger than others [5].This function is known as the *separable* power family [4]. This correlation structure was used for all the treed Gaussian Process approaches implemented for this project. The power exponential function with a constant range d was used for the stationary Gaussian Process.

Treed Partitioning using a Regression Tree

Treed partitioning using a regression tree (since our output is continuous) is the approach that is used here to partition the input space. Classification and regression tree models are binary decision trees that divide up the predictor space repeatedly into partitions (or nodes) based on the value of a single splitting variable [7]. Partitioning the space in this manner progressively increases the homogeneity of the output variable within each node [7]. For our continuous response variable here, a regression tree is built, predicts the average response within a node [7]. The splitting variable is chosen uniformly at each node out of all the input variables available [3]. The splitting point is also chosen randomly from all the points available within the space of the splitting variable [3].

Hierarchical Priors Specification for the Gaussian Process

Each partitioned region contains data, $D_v = \{\boldsymbol{X}_v, \boldsymbol{Z}_v\}$, each consisting of n_v observations [5]. Let $m = m_X + 1$ be the number of covariates in the design matrix X plus an intercept [5]. The heirarchical generative model is as follows:

$$\begin{aligned}
\boldsymbol{Z_v}|\boldsymbol{\beta_v}, \sigma_v^2, \boldsymbol{K_v} &\sim N_{n_v}(\boldsymbol{F_v}\boldsymbol{\beta_v}, \sigma_v^2, \boldsymbol{K_v}) \\
\boldsymbol{\beta_v}|\sigma_v^2, \tau_v^2, \boldsymbol{W}, \boldsymbol{\beta_0} &\sim N_m(\boldsymbol{\beta_0}, \sigma_v^2\tau_v^2\boldsymbol{W}) \\
\boldsymbol{\beta_0} &\sim N_m(\boldsymbol{\mu}, \boldsymbol{B}) \\
\sigma_v^2 &\sim IG(\alpha_\sigma/2, q_\sigma/2) \\
\tau_v^2 &\sim IG(\alpha_T/2, q_T/2) \\
\boldsymbol{W}^{-1} &\sim W((\rho\boldsymbol{V})^{-1}, \rho)
\end{aligned} \tag{6}$$

The constants $\boldsymbol{\mu}$, ρ, α_σ, q_σ, α_T , q_T are treated as known [5]. This model specifies a multivariate normal likelihood with linear trend coefficients $\boldsymbol{\beta_v}$, variance σ_v^2, and correlation matrix K_v [5]. The coefficients $\boldsymbol{\beta_v}$ are believed to have come from a common unknown mean $\boldsymbol{\beta_0}$ and region-specific variance $\sigma_v^2\tau_v^2$. The range d_i and nugget g are considered to be random in the correlation function [5]. An exponential prior is used for the nugget g.

Bayesian CART

The Bayesian CART model essentially represents a Classification and Regression model with the addition of a tree-generating prior. This tree-generating prior ideally guides the stochastic search within the tree space towards more promising regression models. Here, we use the prior specified in [3], which is parameterized as follows:

$$p_{SPLIT}(\eta, \mathcal{T}) = \alpha(1 + q_\eta)^{-b} \tag{7}$$

Here, we use the default values of $\alpha = 0.5$ and $b = 2$, which have been recommended by Gramacy and Lee (2008) for showing good empirical performance [5]. In this parameterization, the decreasing function of q_η making deeper nodes less likely to split and supports "bushy" trees whose terminal nodes do not vary too much in depth [5]. Zero probability is also automatically given here for this dataset to partitions containing less than 10 data-points [4].

Bayesian Treed GP

The Bayesian Treed Gaussian Process first involves the partitioning of the input region using the CART approach. Typical runs of the treed GP on this data generally found two to three partitions in trees [5]. A stationary Gaussian Process is then fitted to each of the terminal nodes (leafs) of the generated trees [5].

Bayesian Treed LLM GP

Here we additionally consider a special limiting case of the Gaussian Process, which is the standard linear model. Specifically, this model can be parameterized by changing the hierarchical specification of $\boldsymbol{Z_v}$ in 6 to the following:

$$\boldsymbol{Z_v}|\boldsymbol{\beta_v}, \sigma_v^2, \boldsymbol{K_v} \sim N_{n_v}(\boldsymbol{F_v}\boldsymbol{\beta_v}, \sigma_v^2, \boldsymbol{I}) \tag{8}$$

Specifically, while some regions of the partition space can only be modelled well with a Gaussian Process, it is possible that a GP may be unnecessarily computationally expensive for other regions [5]. Thus, a limiting linear model (LLM) can be implemented through a model-switching "mixture" prior. In particular, the mixture prior is encoded so that GPs with larger estimated range parameters will be more likely to "jump" to the LLM [4]. A mixture-of-gamma priors is thus used for d [4]. This prior gives roughly equal mass to small d representing a population of GP parameterizations for more wavy surfaces as well as for surfaces that are quite smooth or approximately linear [4]. The details of this prior can be found in [4].

Predictions and Bayesian Model Averaging

As a stationary GP is fit to each partition/node of the tree, the linear predictive distribution of each GP in each of the v partition regions resembles 3. This posterior predictive surface is conditional on a particular tree $\mathcal{T}$. Bayesian model averaging is thus applied to marginalize over trees and derive the posterior distribution of the GP parameters as follows:

$$P(\theta) = \sum_{v=1}^{R} P(\theta_v, \mathcal{T}_v) = \sum_{v=1}^{R} P(\mathcal{T}_v)P(\theta_v|\mathcal{T}_v) \tag{9}$$

where θ_v represents the set of the GP parameters [5].

Random Forests

Random Forests, an ensemble approach, is another tree-based approach, which shares the idea of averaging predictions from many classification and regression trees. In a random forests algorithm, the number of predictors is limited at each split or node of a tree to a different random subset of predictor variables (`mtry`) [6]. This is done to prevent one or a few predictors from dominating the trees and reduces the correlation between trees as well, which works particularly well to reduce the variance of bagging [6].

Essentially, M bootstrap ensembles are generated from the training data-set [6]. A learning method, $\hat{f}^m(\mathbf{x})$ is then trained on each of the m bootstrap ensembles, in order to produce predictions for each random-forest tree, which are then averaged and combined to obtain average predictions for the data, where $\hat{f}_{bag}(\mathbf{x})$ is:

$$\hat{f}_{bag}(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M}\hat{f}^m(\mathbf{x}) \tag{10}$$

Specifically, for a regression random forests model, the average of the predictions made by each of the individual trees generated is used to generate the "final" predictions.

Cross-Validation to evaluate model performance

K-fold Cross-validation is a manifestion of a training/validation/testing framework which is used here for comparatively evaluating model performance, where the data is split into K roughly equal-sized parts [6]. For each part, the model is fit to the other $K-1$ parts of the data. The root-mean squared error (RMSE) of the fitted model is then calculated for each of these k cases [6]. The K estimates of RMSE are then combined as the average. Specifically 10-fold cross-validation is used here. Reasons to use 10-folds include a relatively balanced bias-variance trade-off [6].

Analysis Set-Up

All analyses were run using `R` (version 4.1.2). All the Bayesian CART and GP models, i.e the Bayesian CART, stationary GP, the treed GP and the treed LLM GP were run using the `R` package `tgp` [4]. The package uses Gibbs sampling for the prior hyperparameters, Metropolis-Hastings for the correlation prior hyperparameters, and Reversible Jump (RJ)-MCMC for the Bayesian Model Averaging.All the Bayesian models were run with two restarts in order to better explore the marginal posterior for $\mathcal{T}$ [3]. The *separable* correlation function was used for all the Gaussian Processes run as specified in 5. The random forest model was run using the `ranger` package [10]. Two hyper-parameters: the number of random variables available at each split (`mtry`) and the number of trees (`ntrees`) were tuned using 5-fold cross-validation via the `e1071` package [9]. The hyperparameters were tuned over a grid of 1-13 for `mtry` (which encompasses $M = \sqrt{p}$, a typical empirical rule of thumb) and 100-800 trees in increments of 100 for `ntree` (which encompasses $p \times 10$, also a typical empirical rule of thumb) [1]. The performance of all five models were then compared using the averaged RMSE generated from 10-fold cross-validation.

Results and Discussion

Tuning the random forest hyper-parameters according to our specified grid yielded `mtry` $= 13$ and `ntree` $= 300$ to have the lowest mean squared error. Thus, these hyper-parameter values were used in the final model compared to the Gaussian Processes. Figure 1 depicts the mean root mean squared errors obtained by the five models, while Table 1 presents a five-number summary of the 10 root mean squared errors obtained via 10-fold cross-validation. Overall, we see that the Treed LLM GP shows the lowest RMSE, while the random forests algorithm shows the next lowest RMSE. It should be noted that none of the RMSEs of the GP fits and the random forests model have an extremely significant discrepancy. However, the Bayesian CART model seems to have a dramatically large RMSE compared to the other models.

Figure 3 shows the measure of uncertainty of the joint posterior predictive mean surface of `MinLead` and `OpenAlbedo` for both the stationary Gaussian Process and for the Treed LLM Gaussian Process. It is quite interesting to see that for the stationary GP, the one of the regions with the highest predictive uncertainty coincides with having the lowest relative sampling. The treed GP on the other hand was able to partition the input space to separate out the larger "uninteresting" parts of the input space, as well as obtain sharper regional estimates of predictive uncertainty in this case. While this improves the sampling of the regions with the highest predictive uncertainty, they do still seem to remain relatively undersampled in either model.

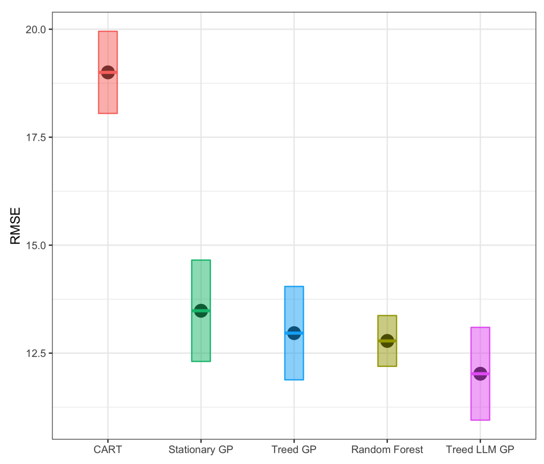


Figure 1: Summarizing mean root mean squared errors (RMSE) obtained with various models used to predict the Ice Velocity over 10-fold cross-validations (Mean ± SE)

Table 1: Five Number RMSE Summary for various models used to predict the Ice Velocity over 10-fold cross-validations

	CART	GP	Treed GP	Treed LLM GP	Random Forests
Mean	18.43	13.12	13.08	12.72	12.78
Median	19.01	12.78	12.65	11.31	13.15
5%	14.93	8.83	8.63	7.26	9.94
95%	22.75	18.07	17.40	15.94	15.14

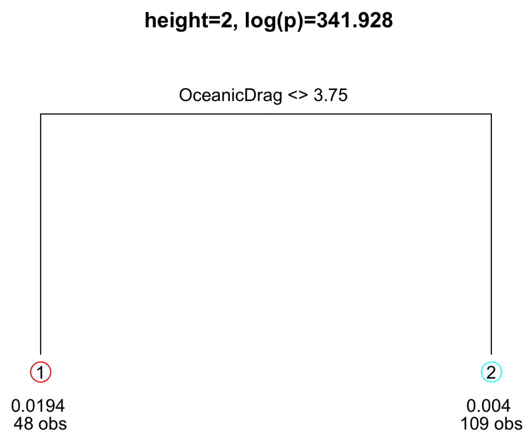
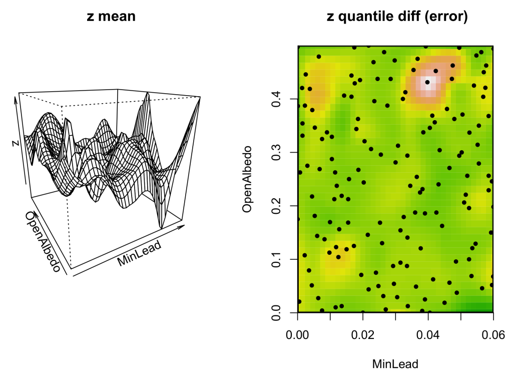
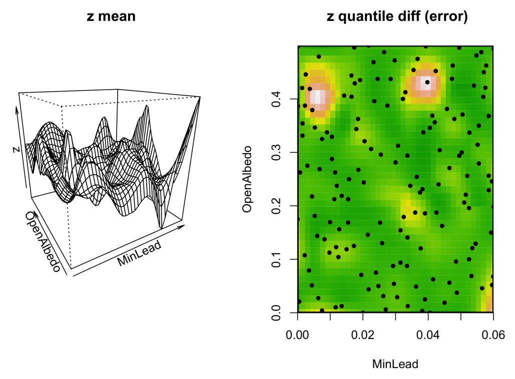


Figure 2: *Maximum a' posteriori* (`MAP`) height tree from the Bayesian Treed LLM GP. The `MAP` tree is of height 2 and its log posterior probability is 341.928. The splitting variable here is `OceanicDrag`. The text at the leaves of the tree show the number of input data points that fall into the partitions with an estimate of the variability within.



(a) Stationary Gaussian Process



(b) Treed LLM Gaussian Process

Figure 3: Measure of uncertainty of the joint posterior predictive mean surface of `MinLead` and `OpenAlbedo` given by the difference in 95% and 5% quantiles of samples from the posterior predictive distribution. The locations sampled by the MCMC algorithm are shown as dots

It is quite interesting that the treed LLM GP seemed to perform marginally better compared to the treed GP. Thinking about the `MAP` tree from the Bayesian Treed LLM GP presented in 2, the two terminal partition tree seems to have the highest posterior probability, indicating that the inputs in one subset of the data are likely quite linear with respect to ice velocity.

Conclusions and Future Directions

Overall, we see that in terms of the RMSE, the treed GP (and specifically the treed GP with a limiting linear model) only marginally outperforms the stationary GP in terms of predicting Ice velocity, with the random forests model a close second. However, looking at the measure of uncertainty in one of the joint posterior distributions between two variables, we do see that the treed GP seems to be able to quantify sharper regional estimates of uncertainty. This model may thus be the better choice compared to a stationary GP, particularly for further use in sensitivity analyses of the parameters. In terms of prediction however, the relatively more parsimonious random forests model also presents an RMSE that is quite on par with that of the treed GP. Overall, with the exception of the Bayesian CART, the treed approaches explored in this project do relatively well in predicting the Ice velocity.

It may be interesting to explore a reduced random forests model with only the "important variables", which can be identified by the change in accuracy by randomly permuting variables using out-of-bag samples. Different values of the hyper-parameters (α and b) for the tree-generating prior should also be explored, ideally through looking at the consequent prior marginal distribution of some characteristic of interest, such as the number of terminal nodes [**chipman˙george˙mcculloch˙1998**]. This could possibly improve the performance of the Bayesian CART model as well. On this note, model calibration should also be performed to evaluate the tree-generating prior, the hyper-parameters and the hierarchical model generative process for this dataset. On the basis of the treed LLM GP performing marginally better than the treed GP, it might be interesting to add a linear model fit to the full data to this comparison. Lastly, in terms of additionally improving the performance of the treed GP's, the `tgp` package additionally offers other MCMC sampling schemes, such as adaptive sampling. Implementing this might help to increase the sampling coverage in the largest areas of predictive uncertainty, such as the instance discussed above, which could potentially improve predictive performance.

References

[1] Bradley Boehmke and Brandon Greenwell. “Chapter 11 Random Forests”. In: *Hands-on machine learning with R*. Chapman and Hall/CRC, 2020.

[2] William L. Chapman et al. “Arctic sea ice variability: Model sensitivities and a multidecadal simulation”. In: *Journal of Geophysical Research* 99.C1 (1994), p. 919. DOI: 10.1029/93jc02564.

[3] Hugh A. Chipman, Edward I. George, and Robert E. McCulloch. “Bayesian Cart Model Search”. In: *Journal of the American Statistical Association* 93.443 (1998), 935–948. DOI: 10.1080/01621459.1998.10473750.

[4] Robert B. Gramacy. “tgp: A package for Bayesian nonstationary, semiparametric nonlinear regression and design by treed gaussian process models”. In: *Journal of Statistical Software* 19.9 (2007). DOI: 10.18637/jss.v019.i09.

[5] Robert B Gramacy and Herbert K. Lee. “Bayesian treed Gaussian Process models with an application to computer modeling”. In: *Journal of the American Statistical Association* 103.483 (2008), 1119–1130. DOI: 10.1198/016214508000000689.

[6] Trevor Hastie, Jerome Friedman, and Robert Tisbshirani. *The elements of Statistical Learning: Data Mining, Inference, and prediction.* Springer, 2017.

[7] Wenbiao Hu et al. “Bayesian classification and regression trees for predicting incidence of cryptosporidiosis”. In: *PLoS ONE* 6.8 (2011). DOI: 10.1371/journal.pone.0023903.

[8] Hassan Maatouk and Xavier Bay. “Gaussian process emulators for computer experiments with inequality constraints”. In: *Mathematical Geosciences* 49.5 (2017), 557–582. DOI: 10.1007/s11004-017-9673-2.

[9] David Meyer et al. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien.* R package version 1.7-9. 2021. URL: https://CRAN.R-project.org/package=e1071.

[10] Marvin N. Wright and Andreas Ziegler. “ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R”. In: *Journal of Statistical Software* 77.1 (2017), pp. 1–17. DOI: 10.18637/jss.v077.i01.

Appendix

0.1 Code

```
# Libraries
library(tgp)
library(randomForest)
library(MASS)
library(e1071)
library(tidymodels)
library(ggplot2)
library(forcats)

# Reading in Wonderland data
input <- read.csv("ice_x.csv")
output <- read.csv("ice_y.csv")

# Keeping only ice velocity
x <- input
y <- output$IceVelocity

# Combining into one dataset
dat <- data.frame(cbind(x, y))

# Create cross-validation folds
set.seed(112)
folds  <- vfold_cv(dat, v = 10)

# For Bayesian CART model
cv_fit1 <- function(splits,...) {
  ZZ <- assessment(splits)$y
 fit1 <- bcart(X = analysis(splits)[,-14],
              Z = analysis(splits)$y, XX = assessment(splits)[,-14])
 rmse <- sqrt(mean((fit1$ZZ.mean - ZZ)^2))
  return(rmse)
}

# Fitting entire training set
fit_cart <- bcart(X = x, Z = y)

plot(fit_cart)


# For Bayesian stationary model
cv_fit2 <- function(splits,...) {
  ZZ <- assessment(splits)$y
 fit2 <- bgp(X = analysis(splits)[,-14],
              Z = analysis(splits)$y, XX = assessment(splits)[,-14],
              bprior = "b0", R = 2, corr = "exp")
 rmse <- sqrt(mean((fit2$ZZ.mean - ZZ)^2))
  return(rmse)
}

# Fitting stationary GP
fit_bgp <-  bgp(X = x, Z = y,  bprior = "b0", corr = "exp", R = 2, trace = TRUE)
```

```r
# For Bayesian treed GP model
cv_fit3 <- function(splits,...) {
  ZZ <- assessment(splits)$y
 fit3 <- btgp(X = analysis(splits)[,-14], Z = analysis(splits)$y, XX = assessment(splits)[,-14],
              bprior = "b0", R = 2)
  rmse <- sqrt(mean((fit3$ZZ.mean - ZZ)^2))
  return(rmse)
}

# For Bayesian treed GP with limiting linear model
cv_fit4 <- function(splits,...) {
  ZZ <- assessment(splits)$y
 fit4 <- btgpllm(X = analysis(splits)[,-14], Z = analysis(splits)$y,
                 XX = assessment(splits)[,-14], bprior = "b0", R = 2)
 rmse <- sqrt(mean((fit4$ZZ.mean - ZZ)^2))
  return(rmse)
}

# Running GP LLM model
# d[0/] means LLM is active
fit_llm <-  btgpllm(X = x, Z = y,  bprior = "b0", R = 2, trace = TRUE)

# MAP tree
png(filename = "map-tree.png", width = 16, height = 14, units = "cm", res = 300)
tgp.trees(fit_llm)
dev.off()

# A few plots
png(filename = "statgp.png", width = 20, height = 14, units = "cm", res = 300)
plot(fit_bgp, proj = c(3, 12))

dev.off()
png(filename = "treedgp.png", width = 20, height = 14, units = "cm", res = 300)
plot(fit_llm, proj = c(3, 12))
dev.off()

# Extracting RMSE's
res_cv_train <-
  folds %>%
  mutate(res_rmse1 = map(splits, .f = cv_fit1) ,
    res_rmse2 = map(splits, .f = cv_fit2),
         res_rmse3 = map(splits, .f = cv_fit3),
    res_rmse4 = map(splits, .f = cv_fit4))

# Mean of RMSE's
rmse_cart <- sqrt(mean((unlist(res_cv_train$res_rmse1))^2))
rmse_gp <- sqrt(mean((unlist(res_cv_train$res_rmse2))^2))
rmse_tgp <- sqrt(mean((unlist(res_cv_train$res_rmse3))^2))
rmse_tgpllm <- sqrt(mean((unlist(res_cv_train$res_rmse4))^2))
var_cart <- sqrt(var((unlist(res_cv_train$res_rmse1))))/sqrt(10)
var_gp <- sqrt(var((unlist(res_cv_train$res_rmse2))))/sqrt(10)
var_tgp <- sqrt(var((unlist(res_cv_train$res_rmse3))))/sqrt(10)
var_tgpllm <- sqrt(var((unlist(res_cv_train$res_rmse4))))/sqrt(10)


# CV varying mtry and ntree
ice.tune <-  tune.randomForest(y ~., data = dat, mtry = 1:13, ntree=100*1:8,
                                   tunecontrol = tune.control(sampling = "cross",cross=5))
```

```r
# Random forests
# Setting up model with mtry = 13 and ntree = 300
rf_mod <-
  rand_forest(mtry = 13, ntree = 300) %>%
  set_engine("ranger") %>%
  set_mode("regression")

# Running random forests
rf_fit <-
  rf_mod %>%
  fit(y ~ ., data = dat)

# Workflow for cross-validation
rf_wf <-
  workflow() %>%
  add_model(rf_mod) %>%
  add_formula(y ~ .)

# Fit on 10-folds
rf_fit_rs <-
  rf_wf %>%
  fit_resamples(folds)

# Use tune package to collect metrics
collect_metrics(rf_fit_rs)

# Plotting RMSE's
rmses <- c(rmse_cart, rmse_gp, rmse_tgp, collect_metrics(rf_fit_rs)$mean[1], rmse_tgpllm)
sds <- c(var_cart, var_gp, var_tgp, collect_metrics(rf_fit_rs)$std_err[1], var_tgpllm)
mod <- c("CART", "Stationary GP", "Treed GP", "Random Forest", "Treed LLM GP")
# Data framing
dat_plot <- data.frame(cbind(rmses, sds, mod))
# Convert class
dat_plot[, 1:2] <- lapply(dat_plot[, 1:2], as.numeric)
png(filename = "RMSE.png", width = 16, height = 14, units = "cm", res = 300)
ggplot(dat_plot, aes(x= mod, y= rmses)) +
  geom_dotplot(binaxis='y', stackdir='center') +
  geom_crossbar(aes(ymin=rmses-sds, ymax=rmses+sds, col = mod, fill = mod, alpha = 0.01), width=.2,
                position=position_dodge(.9), alpha = 0.5) + theme_bw() +
  theme(legend.position="none") + aes(x = fct_inorder(mod)) + labs(col = "", x = "", y = "RMSE")
dev.off()
#
```