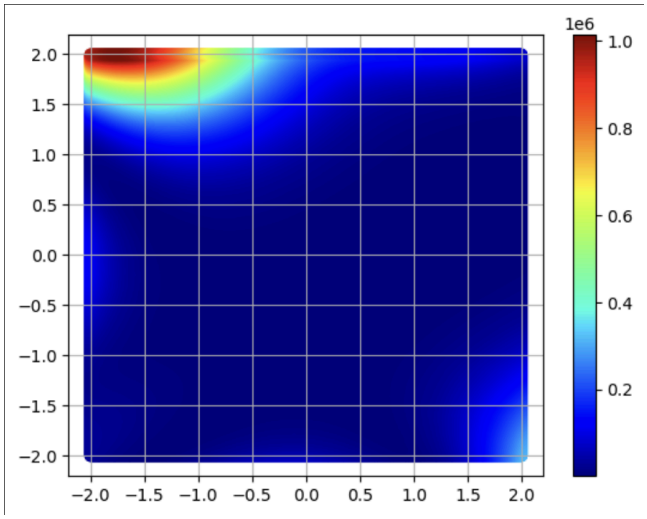
Applying Bayesian Optimization to Hyperparameter Tuning
CSE 515T: Bayesian Methods in Machine Learning Final
Spencer Armon, Sanketh Gudapati, Ethan Weilheimer

Abstract
For this project, we investigate Bayesian optimization and its application to hyperparameter tuning for machine learning models. We first investigate how Bayesian optimization can be applied to a synthetic dataset with a known minimum, the Goldstein-Price function. We then apply Bayesian optimization to two other hyperparameter tuning datasets, SVM and online LDA. All three of these problems are minimization problems.

Section 1: Data Visualization
Data visualization is important to do before running any Gaussian processes and regressions on the data so we can fully understand the data and make sure the tools we are using on the data make sense.

1.1: Heatmap of GP function

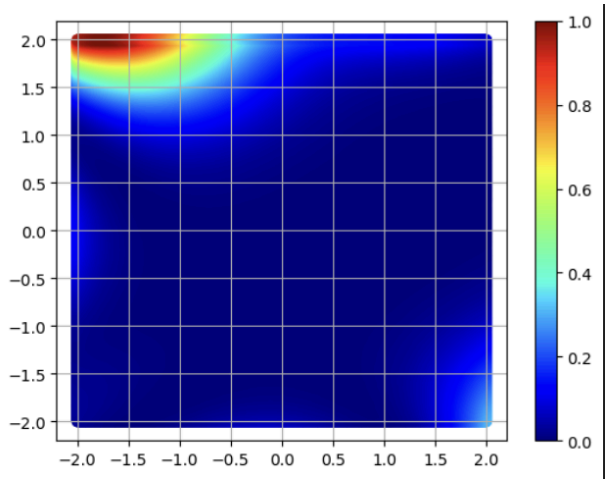


1.2: Behavior of function
It appears to remain relatively stationary throughout the domain except for a large spike near (-2,2).

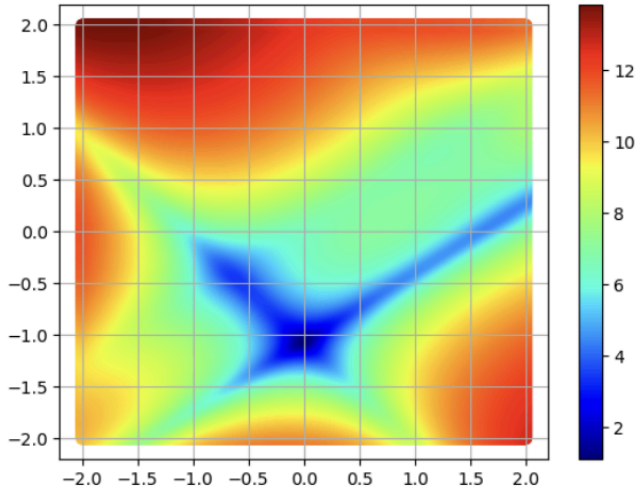
We now try to transform the data to try to get a more stable function. Doing so will allow us to fit it with models much easier. For this we used two general transformations (normalization and logarithmic). It is clear that the normalization does not do a lot to stabilize the function. However, while the logarithmic transformation makes it appear to

be less stable, if you look at the scale, it drastically reduces the amount between the minimum and maximums in the function.

1.3: Transformation of the GP function
Normalization

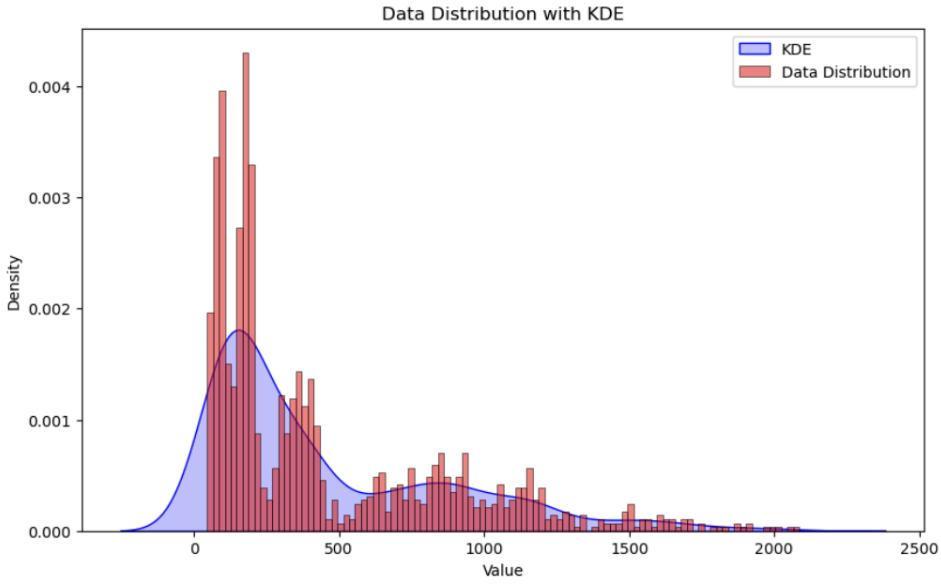


Logarithmic

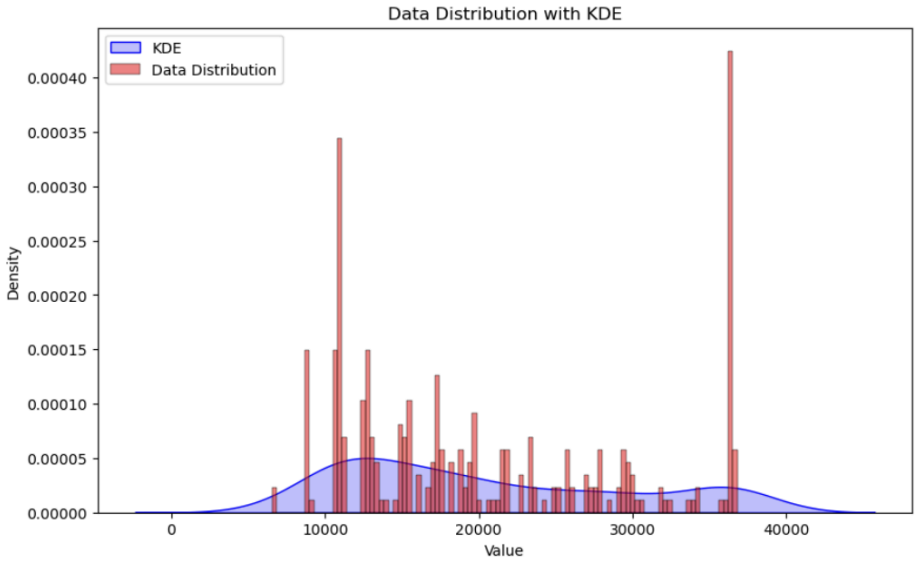


Next we took the kernel density estimates of the LDA and SVM datasets to see how well it can approximate the distribution of the data.

1.4: Kernel density estimate of LDA / SVM outputs
SVM



LDA

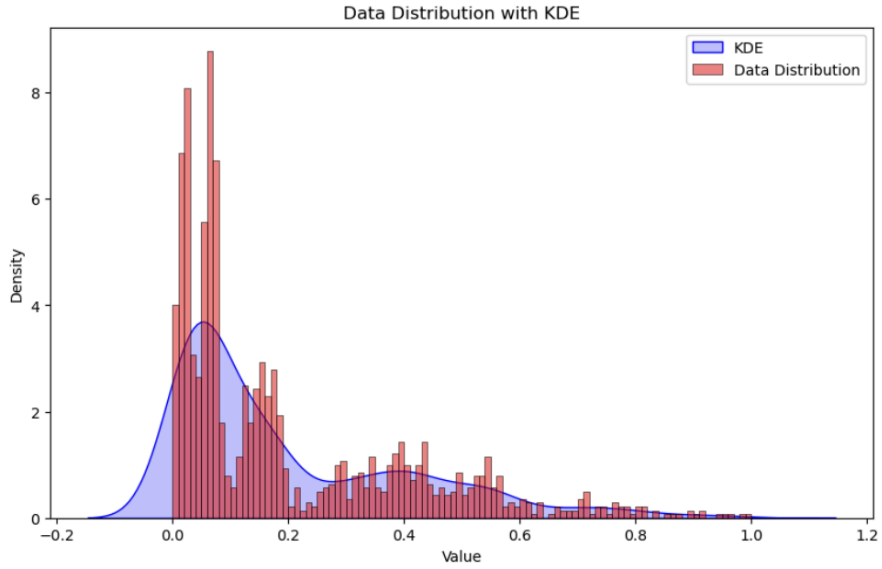


1.5: Interpret distribution
SVM:The distribution appears to be heavily right-skewed.
LDA:The distribution appears to be bimodal with a large peak around 10,000 and a smaller peak around 38,000.

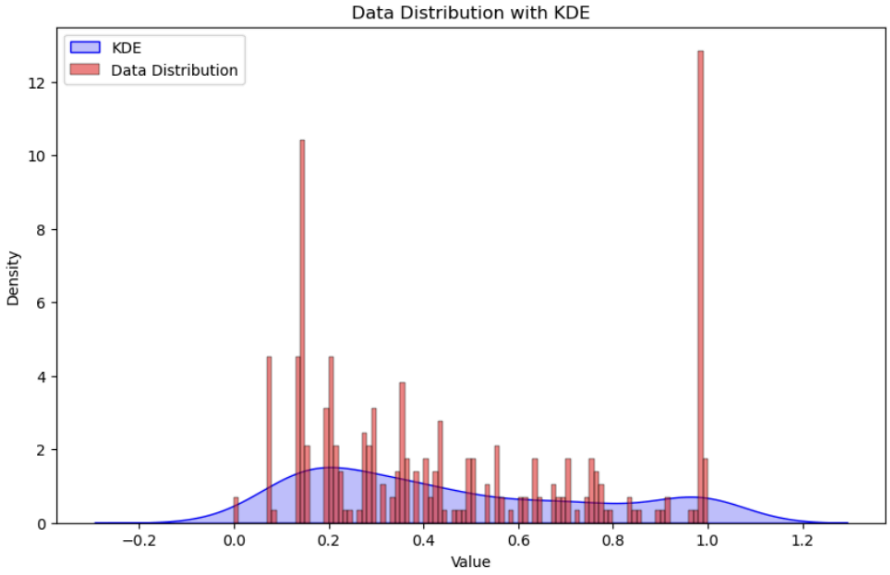
We again transform the data to see if it will allow a kernel density estimate to better fit it. The results can be seen below

1.6: Transformation of LDA, SVM

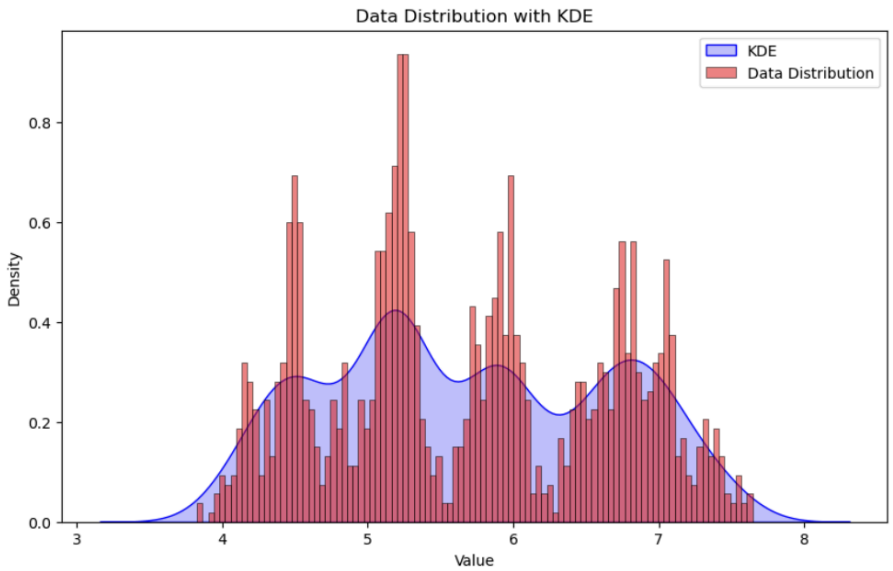
SVM Normalization



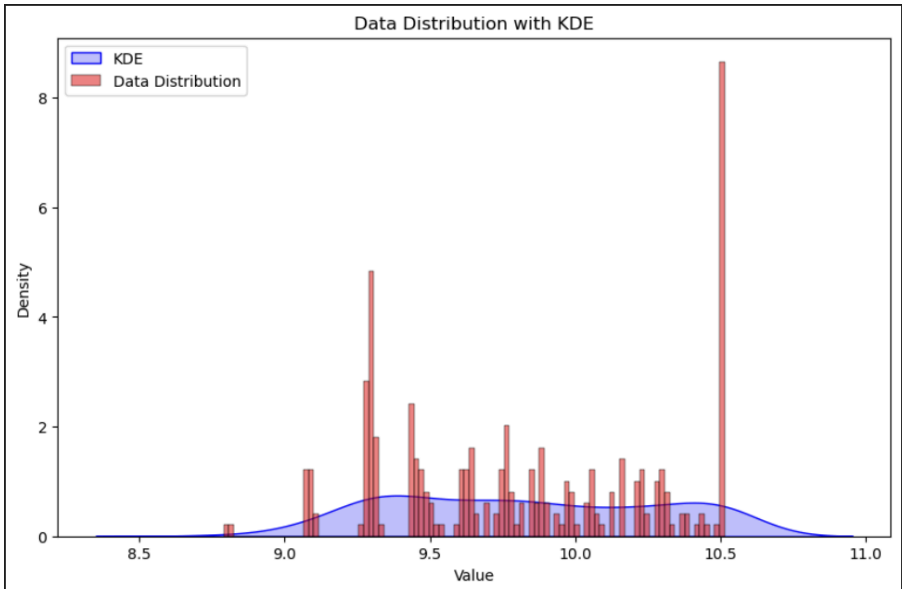
LDA Normalization



SVM Log Transform



LDA Log Transform



Section 2: Model Fitting

Having explored the data visually, we can now try to fit a model to it. We try to fit a Gaussian process to Goldstein-Price data using a Sobol sequence of points using a squared exponential covariance. The hyperparameters found below make sense given that although points together are highly correlated, the mean and output are scaled to match the data.

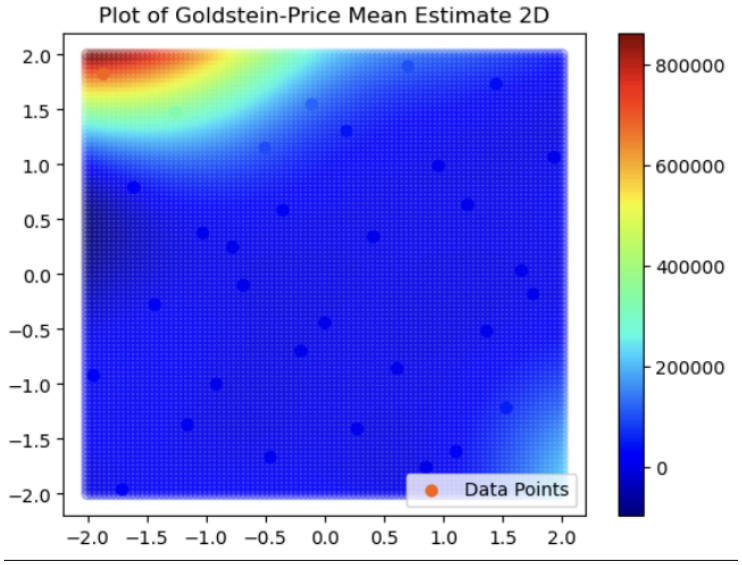
2.1: Fit GP to Sobol sequence data - hyperparameter tuning

Mean: 255161.35262161284

Length Scale: 1.4461502698706994
Output Scale: 426500.7194464453

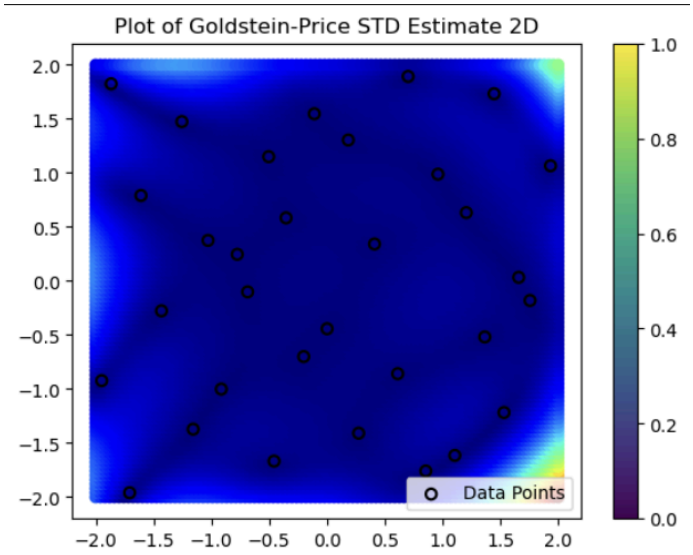
This GP fits the data well for the areas where it has data which can be seen by the colors of the data points lining up to the heat map of the estimate.

2.2: Heatmap of GP posterior mean



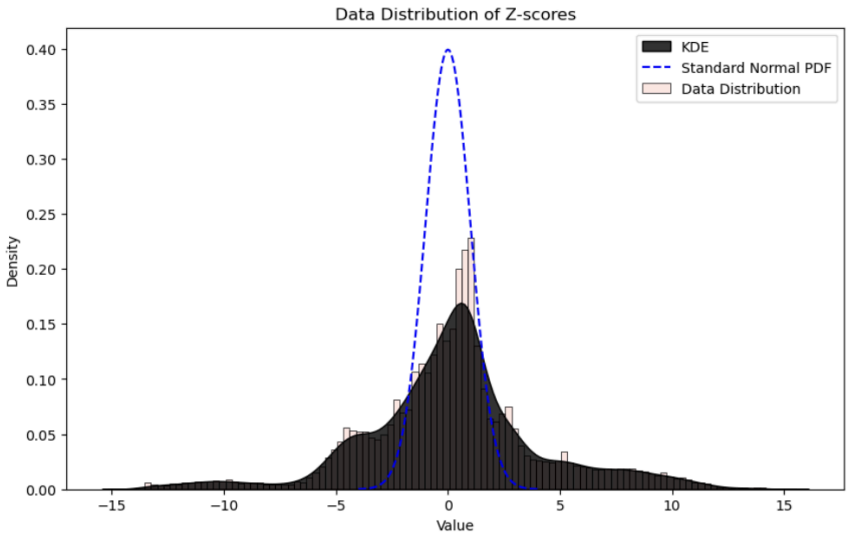
As seen below, the model has a standard deviation of close to zero near the data points which makes sense as it is training them with a noise parameter of 0.001.

2.3: Heatmap of GP posterior standard deviation

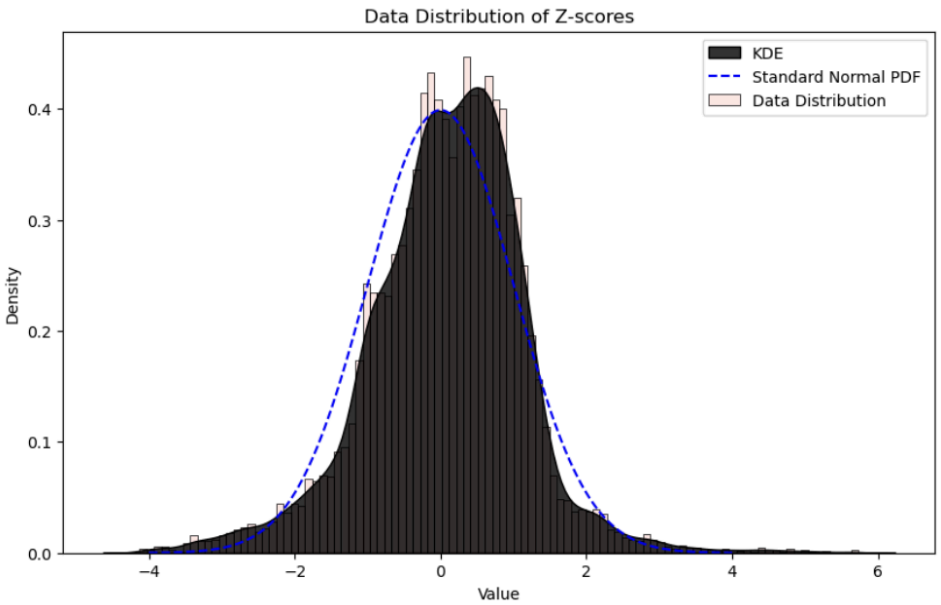


Generally a kernel density estimate of z-scores of residuals that resembles a normal distribution is good. As can be seen here while the raw data fits decently well, the logarithmic transformed data gets a much better fit with the GP. This makes sense as the data is more stationary with the logarithmic transformation.

2.4: Kernel density estimate of z-scores of residuals (raw)



2.5: Kernel density estimate of z-scores of residuals (log)



We then found the Bayesian Information Criterion for each model; the difference in BIC of 7 between the log transformed GP-data to that of the search over models indicates that the GP found with the kernel grammar search on the logarithmically transformed data is much better at fitting the data then the one used on the raw data. It is important

to note that the actual value of the BIC does not matter, it is the difference between models that use the same training data that indicates performance differences.

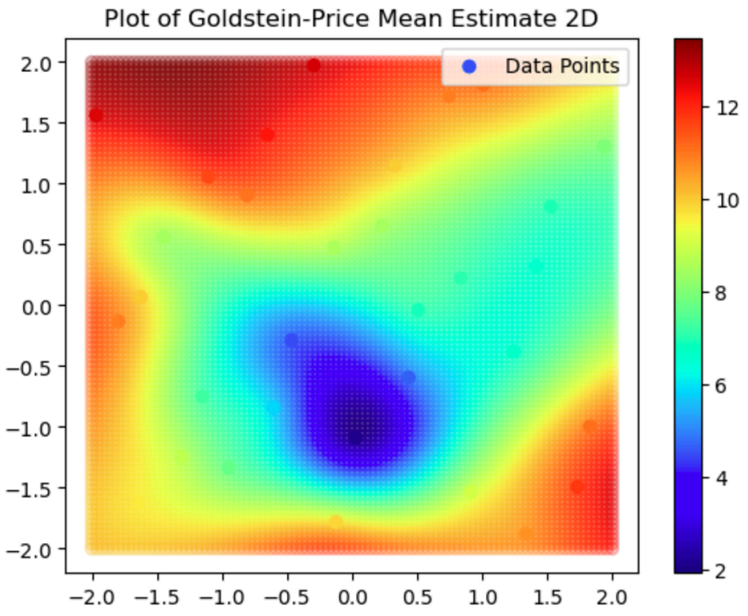
2.6: BIC for log transformed GP data
BIC: 125.60 with constant mean and Matern32 kernel

2.7: BIC search over GP
BIC=134.42

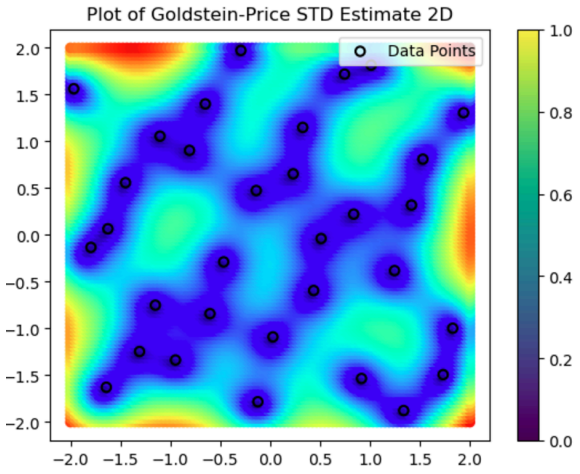
2.8: BIC search over LDA, SVM
SVM:BIC=-100.77 with linear mean and RBF kernel
LDA:BIC=504.05 with constant mean and RBF kernel

Section 3: Bayesian Optimization
Now that we successfully fit accurate Gaussian process models with optimal hyperparameters, we will investigate a Bayesian optimization procedure and observe its performance. For this procedure, we use the expected improvement acquisition function as outlined in the paper by Jasper Snoek found here https://arxiv.org/pdf/1206.2944. In Figure 3.1, we use the 32 Goldstein-Price Sobol sequence data points from above and plot the posterior mean.

3.1: Heatmap of GP posterior mean

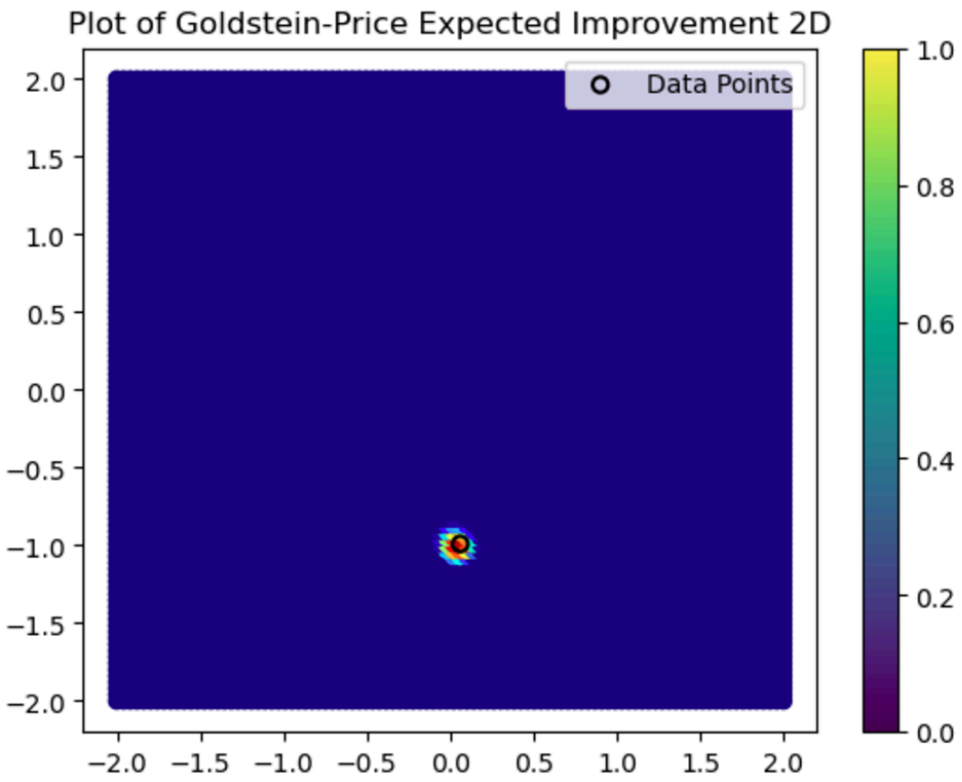


3.2: Heatmap of posterior standard deviation



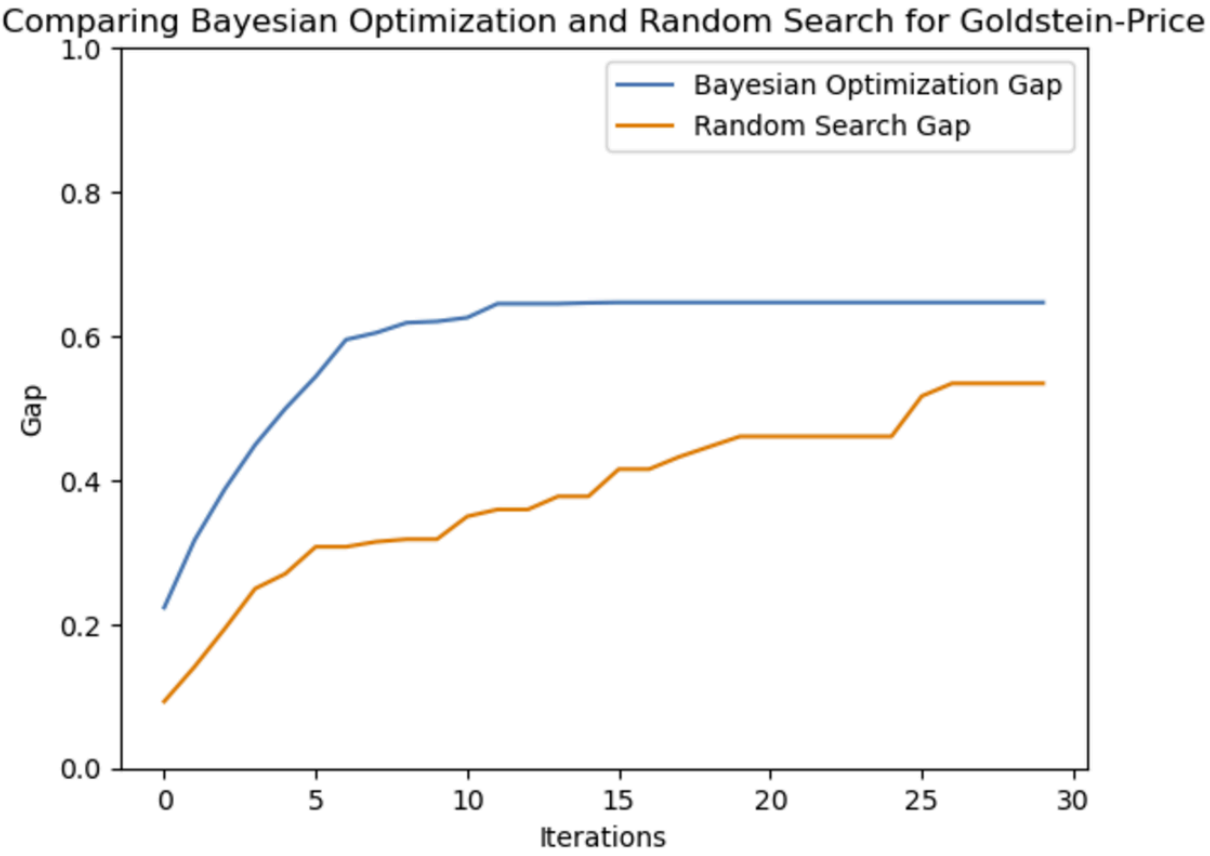
In Figure 3.3, we plot the heatmap for the expected improvement value and mark where it is maximized. The point suggested to observe next is (0.07, -0.99), which is near the true minimum at (0,-1). This shows that the expected improvement metric is effective.

3.3: Heatmap of Expected Improvement



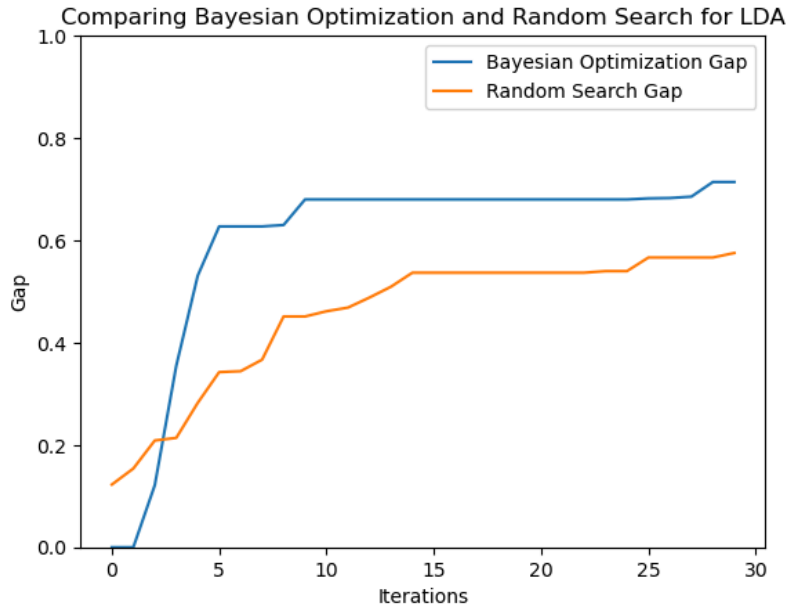
We now select 5 randomly located initial observations for Goldstein-Price, then iteratively select the point that maximizes the expected improvement acquisition function given the current data using the selected Gaussian process model. We do this until we have a dataset of 35 points (selecting a new point 30 times). To evaluate the performance, we measure the gap which is a ratio between the difference of the best found point and the best initial point vs the difference between the minimum and the best initial point. We created this dataset 20 times and averaged the gap at each new datapoint. For comparison, we also ran 20 random searches on the dataset and measured the mean gap. The averaged results are shown below.

3.4: Goldstein-Price Learning Curve

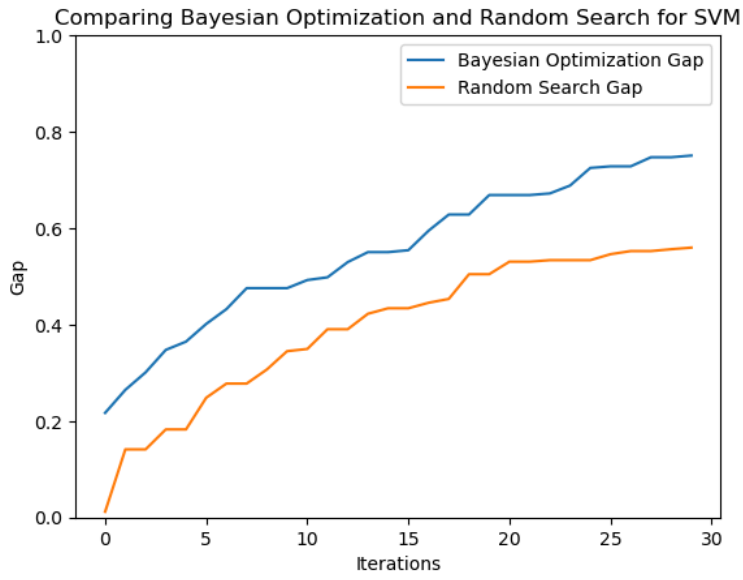


We follow a similar process for the SVM and LDA datasets, but before predicting the means and variances from the Gaussian process, we retune the hyperparameters to maximize marginal likelihood. This helps the gaussian process better adjust to the more complex data. In each of the models, the bayesian optimization outperforms the random search. The results are shown below.

3.5: LDA Learning Curve



3.6: SVM Learning Curve



3.7 GP t-test mean gaps

The difference in means of the gaps for the two approaches is 0.112 after 30 iterations, 0.065 after 60 iterations (while still 30 for the Bayesian optimization), 0.054 after 90, 0.033 after 120, and 0.008 after 150 iterations. When performing a 2-sample t-test for

the mean gaps, it has a p-value of 0.239 after 30 iterations, so it isn't statistically significant. The p-value first crosses 0.05 after 25 iterations.

3.8 LDA mean gaps

The difference in means of the gaps for the two approaches is 0.139 after 30 iterations, -0.036 after 60, -0.108 after 90, -0.131 after 120, and -0.185 after 150.

3.9 SVM mean gaps

The difference in means of the gaps for the two approaches is 0.191 after 30 iterations and 0.066 after 60 iterations (while still 30 for the Bayesian optimization). The gap becomes negative after 82 iterations, is -0.004 after 90 iterations, -0.05 after 120 iterations, and -0.06 after 150 iterations.