Enhanced Feature Prediction in Time-Series Data Using a Structured Learning Approach

Hao Li
Department of Applied Mathematics
University of Washington
henrylihao@outlook.com

1 Introduction

This machine learning project aimed to establish statistical relationship between selected features $\mathcal{X} = (\mathbf{X_1}, \mathbf{X_2})$ and the target features $\mathcal{Y} = (\mathbf{Y_1}, \mathbf{Y_2}, \mathbf{Y_3})$ using a combination of different models. This learning architecture involved three main stages: initial prediction, residual training, and model refinement. In this report, we reflect on each stage, discussing the methodologies, mathematical underpins and insights gained.

2 Preliminary Stage

There are two phases in the Preliminary stage: feature selection and toy example. This is a time-series dataset with five feature columns and each row indicate a time-step with 2000 steps in total. The general requirement is that we only allowed to use two features as input features and use as less data as possible for training.

1. Feature selection: The goal is to identify the two features that best represent the entire dataset to enhance model generalization during training. After plotting the data, it was observed that features 1, 2, and 3 share similar shapes, as do features 4 and 5. Performing a Principal Component Analysis (PCA) revealed that feature 1 ($\mathbf{X_1}$) and feature 5 ($\mathbf{X_2}$) contribute the most to the first and second principal components, respectively (see Table 2 in Section B in the appendix). Hence, the remaining three features will be our target features $(\mathbf{Y_1}, \mathbf{Y_2}, \mathbf{Y_3}) = \mathcal{Y}$.
2. Toy example: In the toy example, I generated some non-linear training pairs and experimented with different types of models (linear regression, SVR, XGBoost, MLP, etc.) that can be used in the first training stage. More importantly, with these models, we also compared two training methods: a directly non-linear mapping (e.g., MLP) and parameter-learning mapping (e.g., linear regression). Although these experiments did not yield a definitive conclusion on which model and method to use, given that the real task involves a different dataset, the goal was to understand the mathematical mechanisms underlying these functional mappings across different models. The results of these toy examples will be posted in the appendix if you are interested.

Note that these five features are actual experimental data with specific physical meanings. For example, feature $\mathbf{X_1}$: `pressureBoundary_out1.vg(m/s)` represents the velocity of gas at output 1 over time. Consequently, a deeper data analysis can be performed here, particularly focusing on the relationships between these physical variables, such as the velocity of gas and fluid at an output. Such analysis could be valuable later when building the model. I will elaborate on this point in more detail in the further work section. In the next section, I will present the flowchart to illustrate the learning architecture, and in the following three sections, I will discuss the methodologies on each stage.

3 Flowchart

This section demonstrates the learning framework for this project using a flowchart. The whole process contains three main stages: **parameter recognition** (learning the parameters that best represent target features using input features); **residual training** (learning the parameters that best estimate the residuals so that they can be subtracted from initial predictions); and **model refinement** (comparing with the true value to learn the parameters that further improve accuracy).

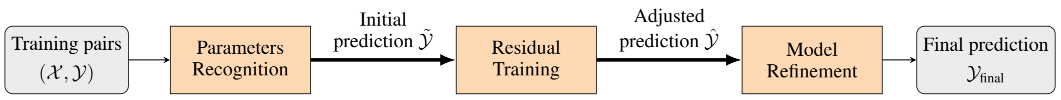
Training pairs $(\mathcal{X}, \mathcal{Y})$
Parameters Recognition
Initial prediction $\tilde{\mathcal{Y}}$
Residual Training
Adjusted prediction $\hat{\mathcal{Y}}$
Model Refinement
Final prediction $\mathcal{Y}_{\text{final}}$

Figure 1: Overall learning framework.

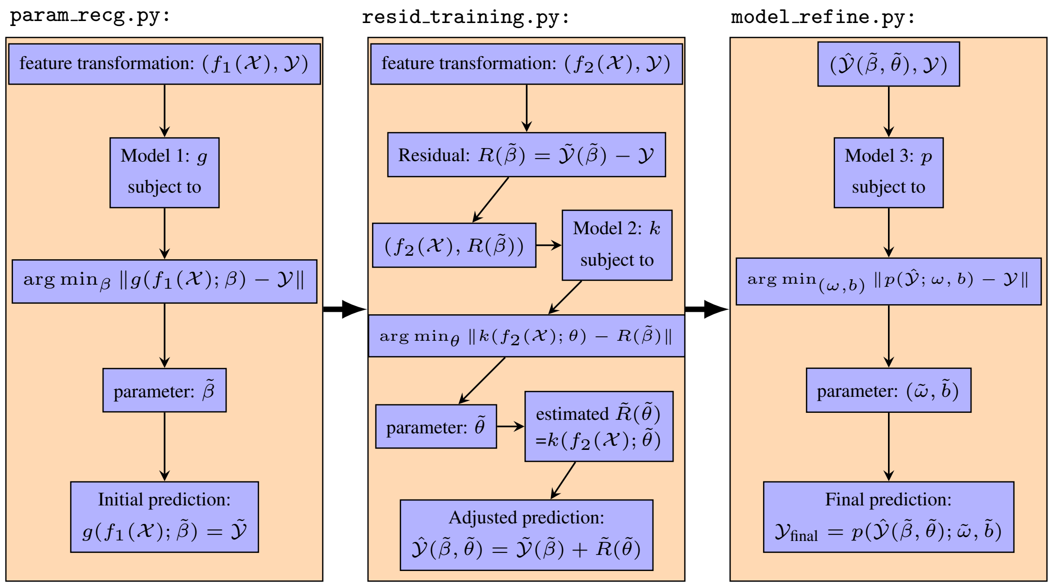
param_recg.py:
feature transformation: $(f_1(\mathcal{X}), \mathcal{Y})$
Model 1: g subject to
$\arg\min_\beta \|g(f_1(\mathcal{X}); \beta) - \mathcal{Y}\|$
parameter: $\tilde{\beta}$
Initial prediction: $g(f_1(\mathcal{X}); \tilde{\beta}) = \tilde{\mathcal{Y}}$
resid_training.py:
feature transformation: $(f_2(\mathcal{X}), \mathcal{Y})$
Residual: $R(\tilde{\beta}) = \tilde{\mathcal{Y}}(\tilde{\beta}) - \mathcal{Y}$
$(f_2(\mathcal{X}), R(\tilde{\beta}))$
Model 2: k subject to
$\arg\min_\theta \|k(f_2(\mathcal{X}); \theta) - R(\tilde{\beta})\|$
parameter: $\tilde{\theta}$
estimated $\tilde{R}(\tilde{\theta})$ $=k(f_2(\mathcal{X}); \tilde{\theta})$
Adjusted prediction: $\hat{\mathcal{Y}}(\tilde{\beta}, \tilde{\theta}) = \tilde{\mathcal{Y}}(\tilde{\beta}) + \tilde{R}(\tilde{\theta})$
model_refine.py:
$(\hat{\mathcal{Y}}(\tilde{\beta}, \tilde{\theta}), \mathcal{Y})$
Model 3: p subject to
$\arg\min_{(\omega, b)} \|p(\hat{\mathcal{Y}}; \omega, b) - \mathcal{Y}\|$
parameter: $(\tilde{\omega}, \tilde{b})$
Final prediction: $\mathcal{Y}_{\text{final}} = p(\hat{\mathcal{Y}}(\tilde{\beta}, \tilde{\theta}); \tilde{\omega}, \tilde{b})$

Figure 2: Detailed process of each stage in the learning framework.

In the next section, I will follow the second flowchart (Figure 2) to explain the process in more details.

4 Training Stage

Stage 1: Parameter Recognition

- **Feature Transformation:** After applying standard scaling, our first step is to apply a kernel transformation to the input features $\mathcal{X} \in \mathbb{R}^d$. Formally, we apply the Laplacian kernel:
$$\begin{aligned} &\mathcal{L} : \mathcal{X} \mapsto \mathcal{L}(\mathcal{X}, \gamma) \\ \text{where} \quad &\mathcal{L}(\mathcal{X}, \gamma) = \exp(-\gamma \|\mathcal{X}_i - \mathcal{X}_j\|^2) \in \mathbb{R}^D. \end{aligned}$$
Typically, $D \gg d$ because additional dimensions capture non-linear interactions among the original features.
- **Model 1:** The first model used is a simple **linear regression** model. We assume that the transformed features have a linear relationship with the target feature $\mathcal{Y}$. Let $\mathbf{X} = \mathcal{L}(\mathcal{X}) \in \mathbb{R}^D$. Mathematically, we have:
$$\begin{aligned} &g : \mathbf{X} \mapsto g(\mathbf{X}) \\ \text{where} \quad &g(\mathbf{X}) = \mathbf{X} \cdot \boldsymbol{\beta} \\ \text{such that} \quad &\arg\min_{\boldsymbol{\beta}} \|g(\mathbf{X}; \boldsymbol{\beta}) - \mathcal{Y}\| \end{aligned}$$

Using Ordinary Least Squares (OLS), the optimal parameters $\tilde{\boldsymbol{\beta}}$ are given by the normal equation:

$$\tilde{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathcal{Y}$$

Hence, the initial prediction is $\tilde{\mathcal{Y}} = \mathbf{X} \cdot \tilde{\boldsymbol{\beta}}$.

- **Results:** In feature transformation, we have a hyperparameter γ. By using a for loop to apply different values of γ to the linear regression model, we found that using only **25%** of the dataset for training, when $\gamma = \mathbf{0.06}$, resulted in the lowest Mean Squared Error (MSE) for the initial prediction: **0.2442**.
- **Reflection:** Several questions arise at this stage:
  1. *Why use the Laplacian kernel?* Although I experimented with various kernels, the Laplacian kernel provided the best results. Technically, this kernel captures the similarity between data points in a way that is robust to outliers, emphasizing the differences between points that are close to each other. Given our goal to predict target features by imitating the shape of the input features, the Laplacian kernel effectively highlighted these similarities, facilitating easier model learning.
  2. *Why use linear regression?* Despite experimenting with different models (SVR, XGBoost, MLP, etc.), the simplest model yielded the best results. One reason could be the correctness of the assumed linear relationship. Another reason might be that the transformed features effectively captured the internal relationships, allowing a simple model to generalize well and avoid overfitting, especially with the limited training data available.

Stage 2: Residual Training

In this stage, the goal is to find the parameters that best estimate the residuals: $R(\tilde{\beta}) = \tilde{\mathcal{Y}}(\tilde{\beta}) - \mathcal{Y}$ so that we can subtract the estimated residuals $\tilde{R}$ from the initial prediction to further improve the results.

- **Feature Transformation:** Similar to Stage 1, we also applied a Laplacian kernel transformation here and let $\mathbf{X} = \mathcal{L}(\mathcal{X}) \in \mathbb{R}^D$. The choice of laplacian is also made from experiments. This kernel highlighted the similarities seems very effective in this problem. Of course, a more rigors mathematical analysis can be done here to ask why this kernel is so helpful, I'll explain more this points in the later section.
- **Model 2:** The second model used is a *Bayesian Ridge Regression* model. This is also a linear model, and thus we maintain the assumption of a linear relationship. The *Ridge* term refers to the L_2 regularization applied in linear regression. Bayesian Ridge Regression combines the principles of Bayesian inference with the ridge regression model:

  $$\mathcal{R}(\tilde{\boldsymbol{\beta}}) = \mathbf{X}\,\boldsymbol{\theta} + \boldsymbol{\epsilon}$$

  The coefficients $\boldsymbol{\theta}$ are considered random variables with a prior Gaussian distribution. This prior is updated with observed data to form a posterior distribution. Mathematically, we have:

  $$\begin{aligned} & k : \mathbf{X} \mapsto k(\mathbf{X}) \\ \text{where} \quad & k(\mathbf{X}) = \mathbf{X} \cdot \boldsymbol{\theta} \\ \text{such that} \quad & \arg\max_{\boldsymbol{\theta}} \log \mathcal{N}(\mathcal{R}|\mathbf{X}\,\boldsymbol{\theta}, \sigma) + \log \mathcal{N}(0, \tau) \\ & = \arg\min_{\boldsymbol{\theta}} \{-\log \mathcal{N}(\mathcal{R}|\mathbf{X}\,\boldsymbol{\theta}, \sigma) + \log \mathcal{N}(0, \tau)\} \end{aligned}$$

  Unlike OLS, the parameters are estimated by maximizing the posterior distribution. Knowing the optimal parameters $\tilde{\boldsymbol{\theta}}$, we can have our best estimation of residuals $\tilde{\mathcal{R}}(\tilde{\boldsymbol{\theta}}) = \mathbf{X}\,\tilde{\boldsymbol{\theta}}$. Hence, our adjusted prediction is

  $$\hat{\mathcal{Y}}(\tilde{\boldsymbol{\beta}}, \tilde{\boldsymbol{\theta}}) = \tilde{\mathcal{Y}}(\tilde{\boldsymbol{\beta}}) + \tilde{\mathcal{R}}(\tilde{\boldsymbol{\theta}})$$

  For the detailed mathematical explanation, refer to the appendix (Section A).

- **Results:** Similarly, using a for loop to apply different values of γ to the Bayesian Ridge Regression, we found that using $\mathbf{60\%}$ of the dataset for training, when $\gamma = \mathbf{0.05}$, resulted in the lowest Mean Squared Error (MSE) for the adjusted prediction: **0.000511**.
- **Reflection:** Several questions arise at this stage:
  1. *Why use Bayesian Ridge Regression?* Bayesian Ridge Regression is effective in estimating residuals for my dataset because it incorporates prior information to regularize coefficients, preventing overfitting, which is crucial given the initial fluctuations and overall trends in my data. Unlike traditional methods like OLS, it provides a probability distribution for coefficients, accounting for uncertainty and leading to more stable and accurate predictions. This regularization helps in stabilizing the coefficient estimates, especially when data points are sparse or not ideally dispersed, resulting in minimal Mean Squared Error (MSE) values and accurate residual estimation.

Stage 3: Model Refinement

The goal for this stage is to build a model that returns parameters revealing the internal relationships (e.g., the temporal dynamics) of target features and uses it to adjust the prediction again to make a final prediction.

- **Training Pairs:** Before stage 3, we used 25% of the dataset to find the relationship between input features and target features, and used 60% of the input features to estimate the residuals. However, we haven't fully utilized that 25% of target features. Hence, in this stage, our training will be based on 25% of the adjusted prediction and the true values: $(\hat{\mathcal{Y}}(\tilde{\boldsymbol{\beta}}, \tilde{\boldsymbol{\theta}}), \mathcal{Y})$. Note that we did not apply any kernels, as we are mapping the predictions to actual values, so there is no need for any functional transformations.
- **Model 3:** The third model used is a *Multi-Layer Perceptron* (MLP) neural network model. This is a classical non-linear model, building upon the Universal Approximation Theorem which states that a neural network can approximate any function to any desired degree of accuracy. Mathematically, we have:

$$\begin{aligned} p :& \mathbb{R}^n \to \mathbb{R}^m \\ & \hat{\mathcal{Y}} \mapsto p(\hat{\mathcal{Y}}) \\ \text{such that} \qquad & \arg\min_{(\boldsymbol{\omega}, \boldsymbol{b})} \| p(\hat{\mathcal{Y}}; \boldsymbol{\omega}, \boldsymbol{b}) - \mathcal{Y} \| \end{aligned}$$

  Unlike the linear models we encountered before, the MLP does not learn explicit coefficients (e.g., $\boldsymbol{\beta}$). Instead, it learns the weights and biases $(\boldsymbol{\omega}, \boldsymbol{b})$ of the activation functions at each layer through the backpropagation algorithm, aiming to minimize the loss function. Although we don't have an explicit form of function p, the network itself should be viewed as function p with optimal parameters $(\tilde{\boldsymbol{\omega}}, \tilde{\boldsymbol{b}})$, such that the final prediction is $\mathcal{Y}_{\text{final}} = p(\hat{\mathcal{Y}}(\tilde{\boldsymbol{\beta}}, \tilde{\boldsymbol{\theta}}); \tilde{\boldsymbol{\omega}}, \tilde{\boldsymbol{b}})$.
- **Results:** Using an MLP with hidden layers of sizes ([64, 32, 8]) and the same $\mathbf{25\%}$ of the dataset for training, the mean squared error (MSE) of the final prediction is reduced to **0.000473**.
- **Reflection:** Several questions arise at this stage:
  1. *Why use MLP?* Initially, I assumed a linear relationship between the predictions and actual values. However, after testing different models, the MLP returned the best results. One reason could be the flexibility of the MLP. As mentioned, because it can approximate any function, the MLP can capture some non-linear relationships that might have been missed (e.g., discrepancies) without making any prior assumptions about the dataset.

5 Improvements

The plot below visualizes the final results compared with the true values.

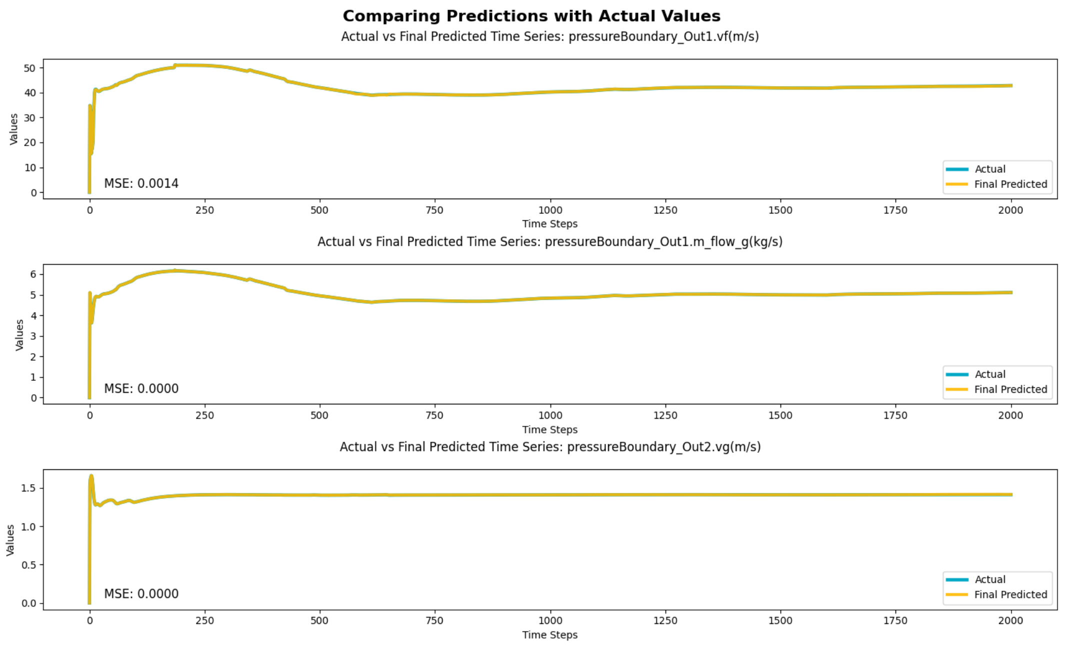

Figure 3: Final results compared with true values

The plot demonstrates the accuracy of the final model in predicting the target values. The closeness of the predicted values to the true values indicates the effectiveness of the model refinement process. The final model's predictions closely align with the true values, showcasing significant improvements over previous versions.

Comparison of Model Versions

To illustrate the improvements across different versions of the model, we present the following table:

Model Version	**Data Usage**		**MSE**
	(Input Features)	(Target Features)	
Single MLP Neural Network	100%	30%	**0.0304**
Three Learning Stages: MLP + MLP + MLP	60%	100%	**0.00425**
Linear Regression + Bayesian Regression + MLP	60%	25%	**0.000473**

Table 1: Comparison of Model Versions

The initial version involved a direct MLP mapping from input features to target features without incorporating any detailed statistical or mathematical analysis. This straightforward approach led to overfitting and insufficient accuracy due to the indiscriminate use of all input data. In the second version, a different learning architecture was employed, involving three stages: parameter recognition, residual training, and model refinement. By splitting the process into stages and fully utilizing the target features, this version managed to significantly reduce the MSE. The final version presented in this paper builds upon the previous architecture but incorporates detailed statistical analysis. Instead of blindly using MLP to approximate general functions, a parameter-traced model was used to focus on estimating the parameters accurately. By focusing on parameter estimation rather than general function approximation, the model achieved better accuracy with less data.

Summary of Improvements

The progression from the single MLP neural network to the final combined model shows a clear improvement in model accuracy. The final version's detailed statistical analysis and focused parameter estimation contributed to its superior performance. The use of a smaller subset of data in a more strategic manner resulted in more efficient training and significantly lower prediction errors.

6 Recap and Insights:

Recap:

This project focused on developing a machine learning model to predict target features from selected input features through a structured learning process. While Multi-Layer Perceptron (MLP) models are powerful, their application requires meticulous statistical analysis and parameter tuning to avoid overfitting. It was observed that simpler, parameter-traced models, which emphasize estimating specific parameters rather than learning a broad, general mapping, often resulted in more accurate and reliable predictions.

Insights and Further Work:

The above summary outlines the key stages and transformations applied in this machine learning project. The evolution of the model reflects a progression from a basic neural network approach to a more sophisticated architecture grounded in statistical and mathematical principles. This progression was driven by a deeper understanding of the mapping relationships and coefficient estimation, achieved through extensive experimentation.

Despite these improvements, the advancements raise questions about the underlying intrinsic logic and mathematical rigor. While the experiments guided us to better models and feature transformations, they often lacked a robust mathematical explanation for why certain models performed better. This highlights the need for a deeper mathematical analysis to rigorously derive why specific models or transformations are superior.

Moreover, the current results, although improved, are not necessarily optimal. The experimental nature of the approach means we did not exhaust all possible model combinations. This underscores the importance of detailed data analysis mentioned in Section 2. Understanding the actual physical relationships, such as the formula between fluid and gas velocities, could enable more targeted model selection and parameter estimation, reducing the reliance on exhaustive experimentation.

In conclusion, while this project demonstrates significant improvements through iterative experimentation and structured model refinement, it also points to the necessity of further mathematical analysis and domain-specific knowledge to achieve truly optimal results.

A Detailed Mathematical Explanation of Bayesian Ridge Regression

Model 2: Bayesian Ridge Regression Bayesian Ridge Regression combines the principles of Bayesian inference with the ridge regression model. The model is expressed as:

$$\mathcal{R}(\tilde{\boldsymbol{\beta}}) = \mathbf{X}\,\boldsymbol{\theta} + \boldsymbol{\epsilon}$$

1. *Prior Distribution:* We assume a Gaussian prior on the coefficients $\boldsymbol{\theta}$:

   $$\boldsymbol{\theta} \sim \mathcal{N}(0, \lambda^{-1}\mathbf{I})$$

   where λ is the precision (inverse of the variance) of the prior distribution.
2. *Likelihood:* The likelihood of observing $\mathcal{R}$ given $\mathbf{X}$ and $\boldsymbol{\theta}$ is:

   $$p(\mathcal{R}|\mathbf{X}, \boldsymbol{\theta}, \sigma^2) = \mathcal{N}(\mathbf{X}\,\boldsymbol{\theta}, \sigma^2\mathbf{I})$$

3. *Posterior Distribution:* By applying Bayes' theorem, we update our prior distribution using the observed data to obtain the posterior distribution of $\boldsymbol{\theta}$:

   $$p(\boldsymbol{\theta}|\mathcal{R}, \mathbf{X}, \sigma^2, \lambda) \propto p(\mathcal{R}|\mathbf{X}, \boldsymbol{\theta}, \sigma^2) \cdot p(\boldsymbol{\theta}|\lambda)$$

   Given the Gaussian forms of the likelihood and prior, the posterior distribution of $\boldsymbol{\theta}$ is also Gaussian:

   $$\boldsymbol{\theta}|\mathcal{R}, \mathbf{X}, \sigma^2, \lambda \sim \mathcal{N}(\boldsymbol{\mu}_\theta, \boldsymbol{\Sigma}_\theta)$$

   where:

   $$\boldsymbol{\Sigma}_\theta = (\mathbf{X}^T\mathbf{X} + \lambda\sigma^2\mathbf{I})^{-1}$$
   $$\boldsymbol{\mu}_\theta = \boldsymbol{\Sigma}_\theta\mathbf{X}^T\mathcal{R}$$

4. *Maximum A Posteriori Estimation:* The parameters are estimated by maximizing the posterior distribution. This can be expressed using the Maximum A Posteriori (MAP) estimation formula:

   $$\arg\max_{\boldsymbol{\theta}} \log\mathcal{N}(\mathcal{R}|\mathbf{X}\,\boldsymbol{\theta}, \sigma) + \log\mathcal{N}(0, \lambda^{-1})$$
   $$= \arg\min_{\boldsymbol{\theta}} \left\{-\log\mathcal{N}(\mathcal{R}|\mathbf{X}\,\boldsymbol{\theta}, \sigma) + \log\mathcal{N}(0, \lambda^{-1})\right\}$$

5. *Assumptions:* When using Bayesian Ridge Regression, we make the following assumptions:
   (a) The relationship between the predictors $\mathbf{X}$ and the response $\mathcal{R}$ is linear.
   (b) The noise $\boldsymbol{\epsilon}$ follows a Gaussian distribution with zero mean and constant variance σ^2.
   (c) The predictors $\mathbf{X}$ are not perfectly collinear.
   (d) The prior distribution of the coefficients $\boldsymbol{\theta}$ is Gaussian with mean zero and precision λ.
   (e) The observed data $\mathcal{R}$ provides sufficient information to update the prior distribution meaningfully.

B Principal Component Analysis

The following table presents the Principal Component Analysis (PCA) loadings for the features used in the study:

Feature	**PC1 Loading**	**PC2 Loading**
pressureBoundary_Out1.vg(m/s)	-3.56	0.25
pressureBoundary_Out1.vf(m/s)	-3.44	0.32
pressureBoundary_Out1.m_flow_g(kg/s)	-1.58	0.00
pressureBoundary_Out2.vg(m/s)	-2.34	0.01
pressureBoundary_Out2.m_flow_f(kg/s)	-0.13	0.87

Table 2: Principal Component Analysis (PCA) Loadings