Bayesian Machine Learning

Stephen Charles

Final Project
The University of Bath
May 2022

Abstract

The final piece of assessed coursework involves the evaluation of Bayesian modelling methods on a real multivariate regression task. The guiding objectives are to derive a good predictor for data derived from an "energy efficiency" data set, and to estimate which of the input variables are relevant for prediction. In particular, the exercise focuses on approximating (and averaging over) posterior distributions using the Hamiltonian Monte Carlo (HMC), Variational Inference (VI), and Gaussian Processes.

Experiments will be based mainly on existing (or supplied) analytic code and techniques you have already learned. You will of course need to write all the relevant code in the relevant cells of the Jupyter Notebook to process the data, apply the methods appropriately, extend them in places, and ultimately calculate and output the necessary results. A key part of the assessment is to compile, present and critique all those results effectively within an "individual project report" document. For this exercise, your code in the Jupyter Notebook will also be auto-marked. Marks will be awarded based on the report, code, and (potentially) group contributions.

Contents

List of Figures

List of Tables

List of Code

Chapter 1

Exploratory Analysis

The energy efficiency data set is distributed by the University of Oxford, and available for download at the UCI Machine Learning Repository. It is a multivariate dataset containing 768 examples, comprising of 1 constant bias and 8 input variables $x_0, x_1, x_2, \cdots, x_8$, where x_0 is the constant bias, and the rest represent basic architectural parameters for buildings.

For the purposes of modeling, we chose to pre-process the data to standardise the inputs to have 0 mean and a standard deviation of 1. The *const* column was preserved to be 1. The data was split into two sets, *'ee-train.csv'* for the training stage, and *'ee-test.csv'* for the testing stage.

Previously, analysis (Tsanas and Xifara, 2012) for this dataset used both random forest and classical linear regression models, which had a mean average error of 0.51 and 1.42 respectively.

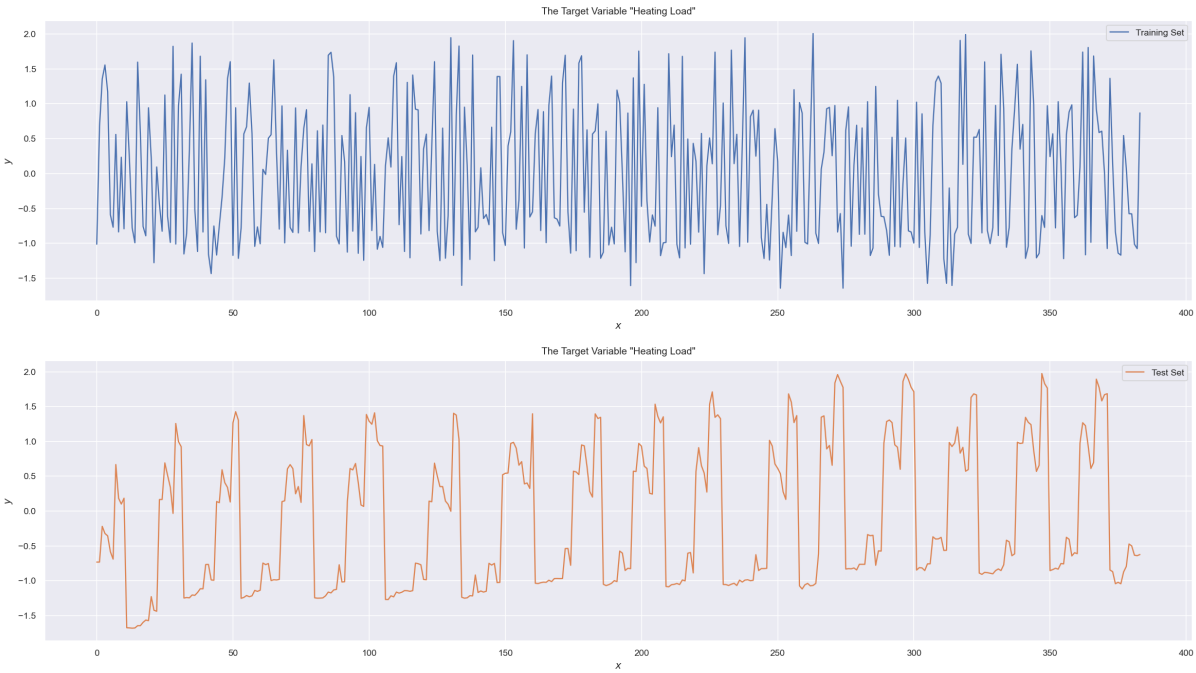

Figure 1.1: The training and test sets for the target variable *Heating Load*.

1.1 Correlation

The data appears non-Gaussian, thus we used the correlation coefficient (Spearman) as a metric to determine the association between the target variable and the dependants. In a sense we can gauge an estimation of the linearity using this method. Plots shown in Figure 1.2 were computed to illustrate the correlation between the target variable *Heating Load* and the 8 input variables.

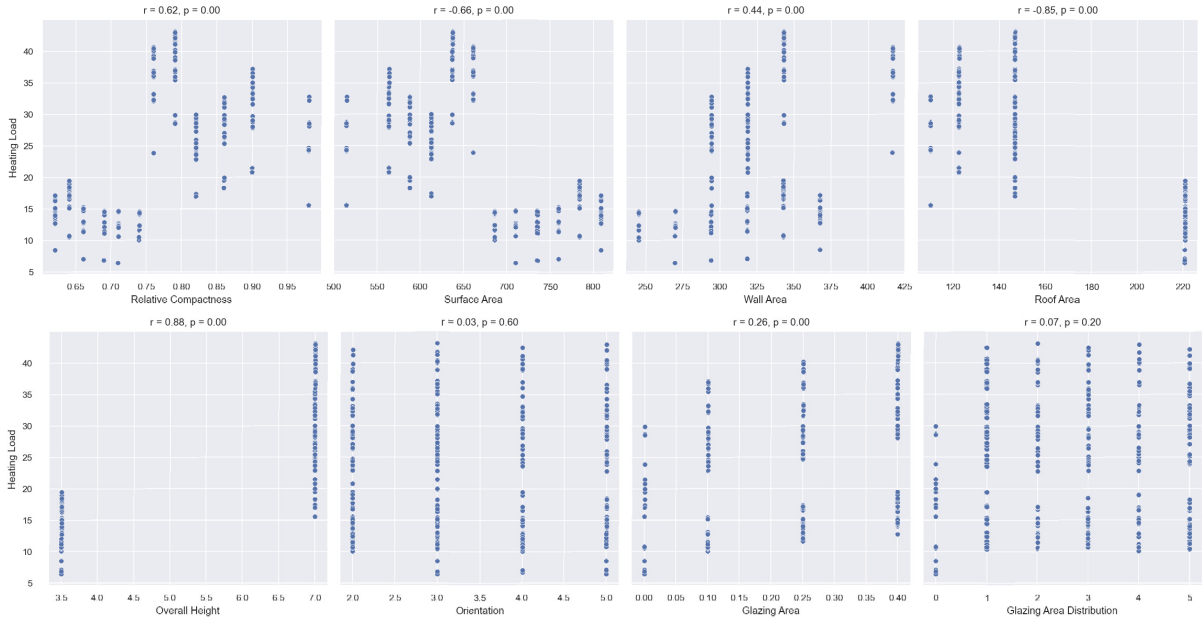

Figure 1.2: The correlation between heating load and the other dependent variables.

The least correlated and thus least important in terms of their feature importance to the model were *Orientation, Glazing Area and Glazing Area Distribution*. Conversely, the most important appear to be *Relative Compactness, Roof Area and Overall Height* in this analysis, in agreement with Tsanas and Xifara (2012).

Figure 1.3 shows a correlation heatmap between all features. Positive and negative correlation is defined the same as before, corresponding to the colormap indicated by the colorbar. The final row for *Heating Load* gives us the same information shown as before in Figure 1.2.

Figure 1.3: The correlation between all features.

1.2 Ordinary Least Squares

Ordinary Least Squares regression is a method where the coefficients of linear regression equations can be estimated which describe multiple independent variables and a dependent variable. Figure 1.4 shows the results of such an approach, and Table 1.1. From the OLS regression results we can see that the most correlated features are x_1, x_2, x_4 and x_5, corresponding to *Relative Compactness, Surface Area, Roof Area* and *Overall Height*, in agreement with our visual inspection of the correlation.

Table 1.1: The RMSE and MAE of the train and test sets for OLS.

Metric	Train Error
RMSE	0.2995679580412746
MAE	0.21194498202562453
Metric	Test Error
RMSE	0.2828601052708685
MAE	0.20581054973407428

OLS Regression Results

Dep. Variable:	y	R-squared:	0.910
Model:	OLS	Adj. R-squared:	0.909
Method:	Least Squares	F-statistic:	544.8
Date:	Thu, 14 Apr 2022	Prob (F-statistic):	1.72e-192
Time:	13:08:09	Log-Likelihood:	-81.993
No. Observations:	384	AIC:	180.0
Df Residuals:	376	BIC:	211.6
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	3.99e-17	0.015	2.58e-15	1.000	-0.030	0.030
x1	-0.7196	0.165	-4.373	0.000	-1.043	-0.396
x2	-0.3921	0.120	-3.269	0.001	-0.628	-0.156
x3	0.0752	0.031	2.420	0.016	0.014	0.136
x4	-0.4210	0.108	-3.880	0.000	-0.634	-0.208
x5	0.7166	0.085	8.401	0.000	0.549	0.884
x6	-0.0125	0.016	-0.803	0.423	-0.043	0.018
x7	0.2756	0.016	17.129	0.000	0.244	0.307
x8	0.0203	0.016	1.267	0.206	-0.011	0.052

Omnibus:	7.903	Durbin-Watson:	1.890
Prob(Omnibus):	0.019	Jarque-Bera (JB):	11.852
Skew:	0.109	Prob(JB):	0.00267
Kurtosis:	3.832	Cond. No.	8.07e+15

Figure 1.4: The correlation between heating load and the other dependent variables.

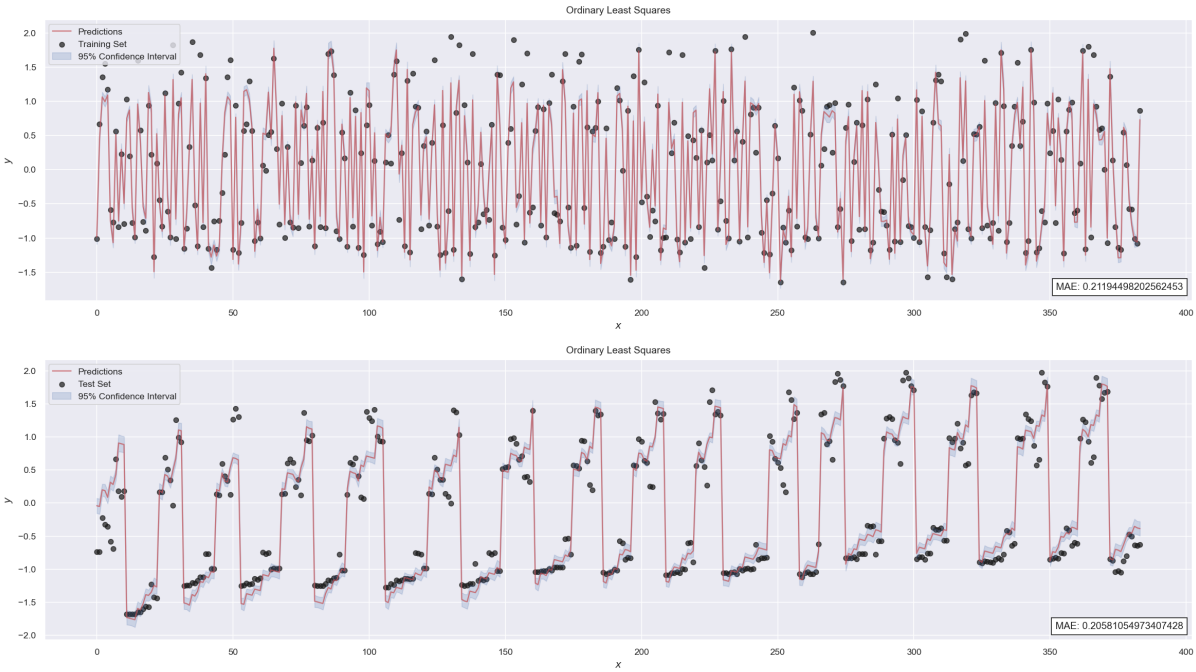

Figure 1.5: The predictions against the training and test set for OLS.

Chapter 2

Bayesian Linear Regression

Define a standard linear regression model with an unknown coefficient set $\mathbf{w}$.

- $\mathbf{w}$ is assumed to have a Gaussian prior $\mathcal{N}(0, \sigma_w^2)$, and the precision as $\alpha = \frac{1}{\sigma_w^2}$.
- The problem can be modelled using additive Gaussian noise $\mathcal{N}(0, \sigma_\epsilon^2)$, and the precision as $\beta = \frac{1}{\sigma_\epsilon^2}$.
- The unknown hyperparameter set consists of $\theta = (\sigma_w^2, \sigma_\epsilon^2) = (\alpha, \beta)$.
- With observation $\mathcal{D}$, the posterior we want to estimate can be written as $p(\mathbf{w}, \theta|\mathcal{D})$, in our case this is y, *Heating Load*.

2.1 Type-II Maximum Likelihood

The full posterior we require is $p(\mathbf{w}, \theta|\mathcal{D})$ as defined above, where $\theta = (\sigma_w^2, \sigma_\epsilon^2) = (\alpha, \beta)$. The Type-II maximum likelihood is used to infer the hyperparameters α and β, and the posterior in our case for this dataset is

$$p(\mathbf{w}, \alpha, \beta|y) = p(\mathbf{w}|\alpha, \beta, y)p(\alpha, \beta|y). \tag{2.1}$$

We wish to maximise the marginal likelihood $p(y|\alpha, \beta)$ to approximate the most probable values for α and β. For this, 100 uniformly distributed samples across the range -5 to 0 were iterated over using `compute log marginal` to determine the most likely values shown in Figure 2.1 and Table 2.1. The predictions were made via the `compute posterior` function. The errors in the train and test sets are displayed in Table 2.2.

Table 2.1: The most probable values for the Type-II Maximum Likelihood.

Priors	
α	0.1543261793585188
β	0.09313200592177617
$\log(\alpha)$	-1.868686868686869
$\log(\beta)$	-2.3737373737373737
Log-Likelihood	-108.930157577226

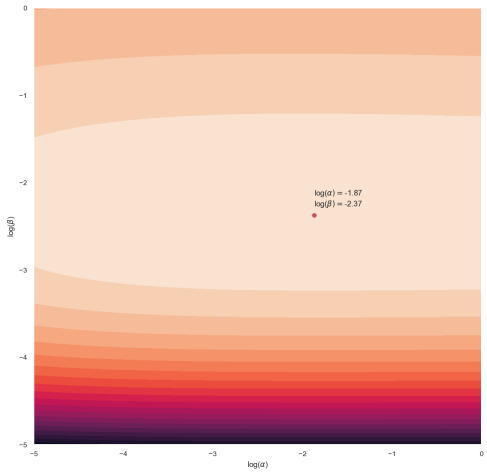

Figure 2.1: For clarity, the posterior is shown in log space. The most probable values were $\alpha = 0.15$ and $\beta = 0.093$.

Table 2.2: The RMSE and MAE of the train and test sets for BLR.

Metric	Train Error
RMSE	0.29956822144275885
MAE	0.21193096625675303
Metric	Test Error
RMSE	0.28285511216629783
MAE	0.20578061010720858

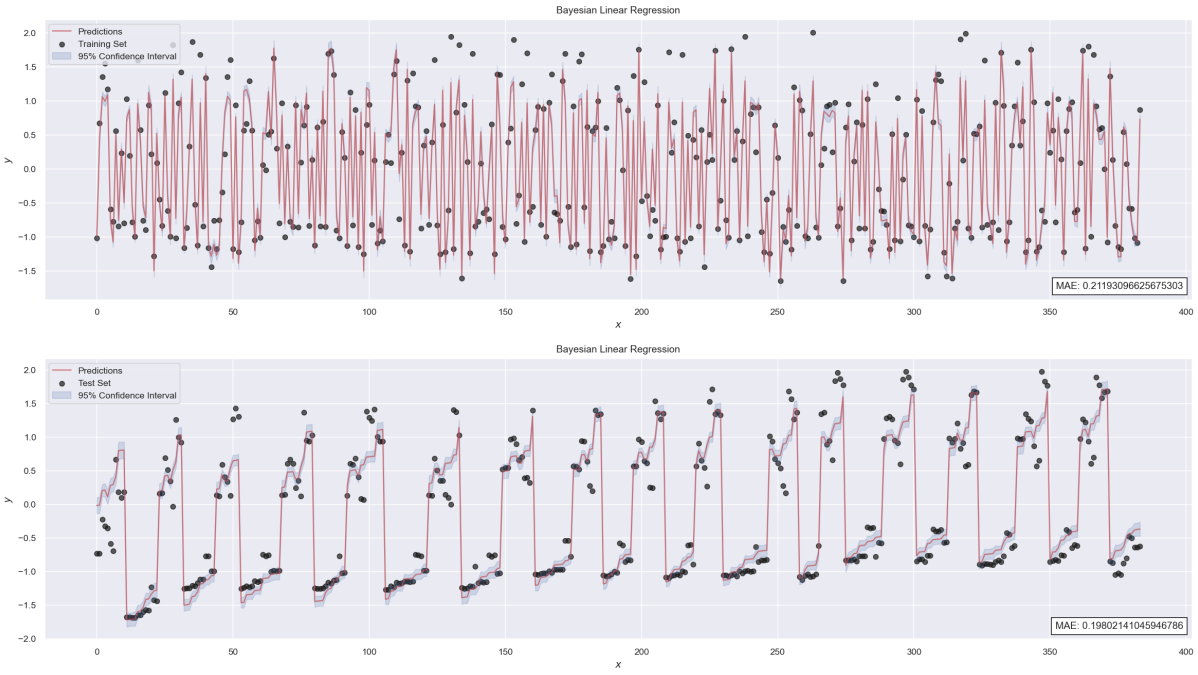

Figure 2.2: The predictions against the training and test set for BLR.

2.2 Variational Inference

Variational Inference is a method for approximating distributions, casting inference as an optimization problem. The main goal is to find a good proposal distribution $\mathcal{Q}(\theta)$ which is the best match for an unknown posterior distribution or objective $\mathcal{P}(\theta)$. The Kullback-Leibler divergence is used to analyse similarity between the proposal and posterior, to determine similarities between both.

In this project, the proposal is given by

$$\mathcal{Q}(\mathbf{w}, \alpha, \beta) = \mathcal{Q}(\mathbf{w}, \beta)\mathcal{Q}(\alpha), \tag{2.2}$$

with priors

$$\mathcal{Q}(\mathbf{w}) = \mathcal{N}(\mathbf{w}|0, (\alpha\beta)^{-1}) \tag{2.3}$$
$$\mathcal{Q}(\alpha) = \mathcal{G}(\alpha|a_0^{\alpha}, |b_0^{\alpha}) = \mathcal{G}(\alpha|a, b) \tag{2.4}$$
$$\mathcal{Q}(\beta) = \mathcal{G}(\beta|a_0^{\beta}, |b_0^{\beta}) = \mathcal{G}(\beta|c, d), \tag{2.5}$$

where

$$\mu_N = \tfrac{a_N}{b_N}\mathbf{I} + \mathbf{X}^T\mathbf{X} \tag{2.6}$$
$$\Sigma_N = V_N\mathbf{X}^T\mathbf{y} \tag{2.7}$$
$$a_N = a_0 + \tfrac{K}{2} \tag{2.8}$$
$$b_N = \tfrac{1}{2}b_0(\tfrac{c_N}{d_N})\Sigma_N^T\Sigma_N + Tr(\mu_N) \tag{2.9}$$
$$c_N = c_0 + \tfrac{N}{2} \tag{2.10}$$
$$d_N = \tfrac{1}{2}d_0(||y - x\Sigma_N||^2 + \Sigma_N^T(\tfrac{a_N}{b_N}))\Sigma_N \tag{2.11}$$
$$\tag{2.12}$$

The floats a_N, b_N, c_N, d_N give us the hyper-parameters α and β, and are defined by

$$\alpha = \tfrac{a_N}{b_N} \tag{2.13}$$
$$\beta = \tfrac{d_N}{c_N}. \tag{2.14}$$

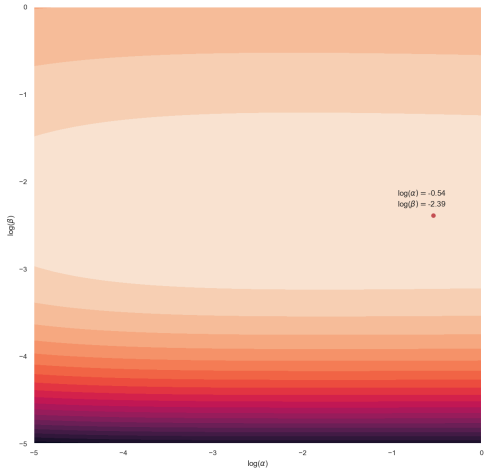

Figure 2.3: For clarity, the posterior is shown in log space. The expected values were $\alpha = 0.84$ and $\beta = 0.1$.

Table 2.3: The most probable values for the Type-II Maximum Likelihood.

Priors	
α	0.583662366586864
β	0.09193207284416256
$\log(\alpha)$	-0.5384326027469926
$\log(\beta)$	-2.3867053132897262
Log-Likelihood	-111.05977819108548

Table 2.4: The RMSE and MAE of the train and test sets for VI.

Metric	Train Error
RMSE	0.29982217954808726
MAE	0.2119250434809071
Metric	Test Error
RMSE	0.28297622986213017
MAE	0.20512841137791837

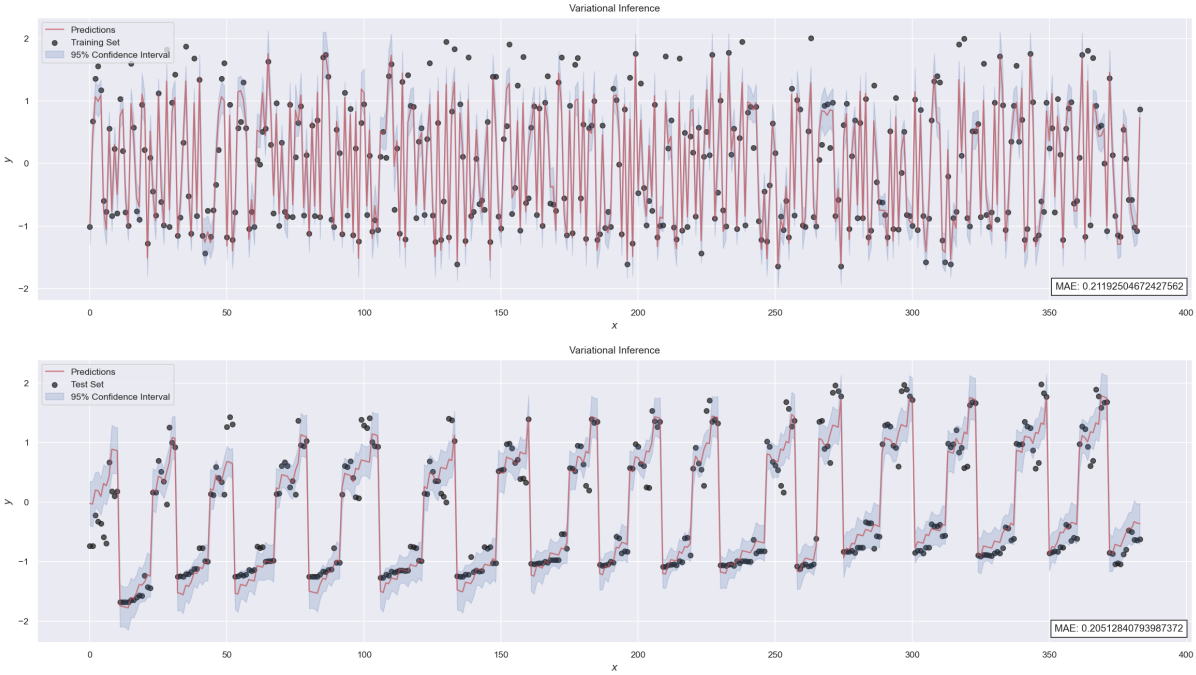

Figure 2.4: The predictions against the training and test set for VI.

Chapter 3

Verify HMC on a 2D Gaussian

The Hamiltonian Monte Carlo method aims to make the random walk in algorithms more efficient. The 2D Gaussian proposed is shown in Figure 3.1 with mean 0 and variance 1. The mathematical formula for this is

$$f(x,y) = A\exp\left(-\left(\frac{(x-x_0)^2}{2\sigma_X^2} + \frac{(y-y_0)^2}{2\sigma_Y^2}\right)\right). \tag{3.1}$$

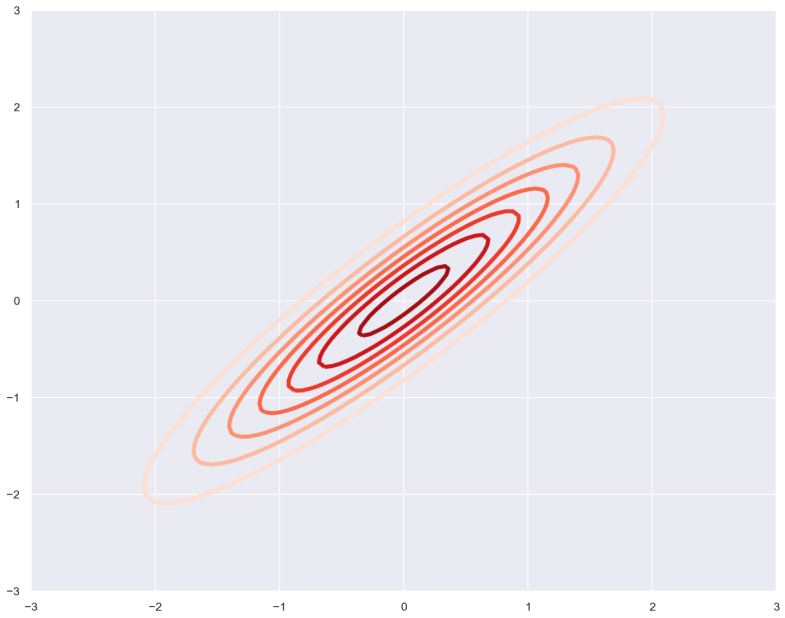

Figure 3.1: A 2D Gaussian with mean 0 and variance 1.

Table 3.1: The values used to obtain HMC results.

Hyper-Parameters	
R	5000
L	20
eps	0.36

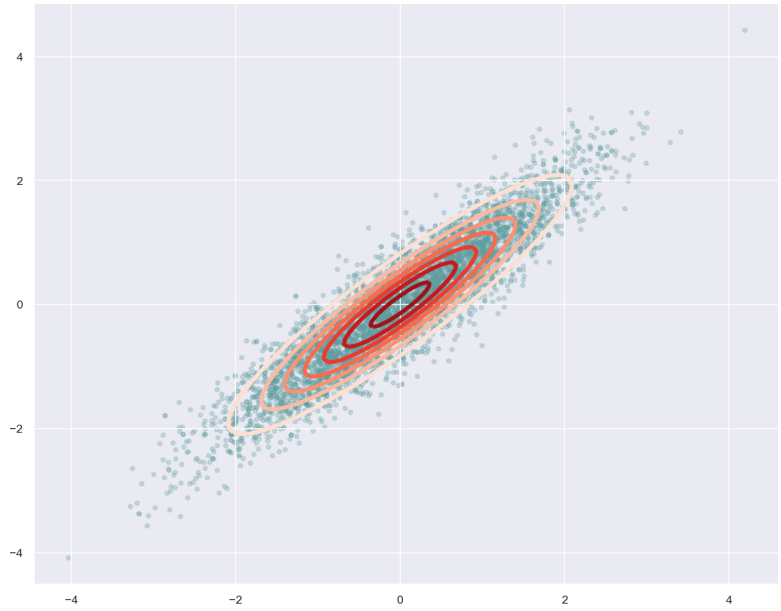

Figure 3.2: HMC Distribution fitted.

```
Calc.          Numeric       Delta          Acc.
     10.6984        10.6984  -7.451781e-10  11
    -9.82492       -9.82492  -1.095202e-09  10
|----------|  0% accepted [ 2 secs to go ]
|#---------| 92% accepted [ 2 secs to go ]
|##--------| 92% accepted [ 2 secs to go ]
|###-------| 91% accepted [ 2 secs to go ]
|####------| 91% accepted [ 2 secs to go ]
|#####-----| 91% accepted [ 1 secs to go ]
|######----| 90% accepted [ 1 secs to go ]
|#######---| 90% accepted [ 1 secs to go ]
|########--| 90% accepted [ 1 secs to go ]
|#########-| 90% accepted [ 0 secs to go ]
|##########| 90% accepted [ 0 secs to go ]
HMC: R=5000 / L=20 / eps=0.36 / Accept=90.4%
```

Figure 3.3: The acceptance rate over iterations.

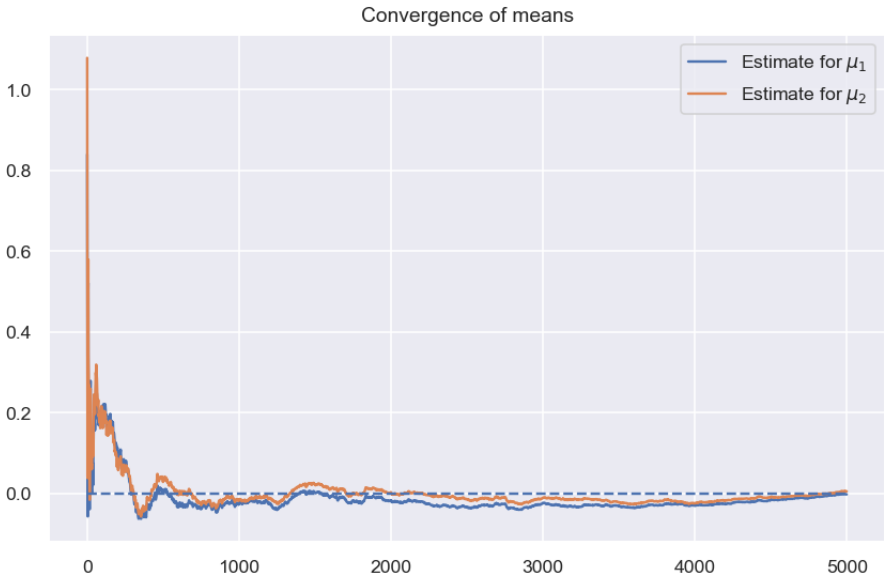

Figure 3.4: The convergence over 5000 samples.

```python
def energy_func(x, covar):
    """Energy function. Returns Neglogpdf."""
    return -stats.multivariate_normal.logpdf(x, cov=covar)

def energy_grad(x, covar):
    """Gradient function."""
    v1 = covar[0, 0]
    v2 = covar[1, 1]
    s1 = np.sqrt(v1)
    s2 = np.sqrt(v2)
    rho = covar[0, 1]/(s1 * s2)

    g = np.zeros(2)
    g[0] = 2 / v1 * (x[0]) - 2 * rho * (x[1])/(s1 * s2)
    g[1] = 2 / v2 * (x[1]) - 2 * rho * (x[0])/(s1 * s2)
    g = g / (2 * (1 - rho ** 2))
    return g
```

Listing 3.1: Designed functions `energy_func` and `energy_grad`

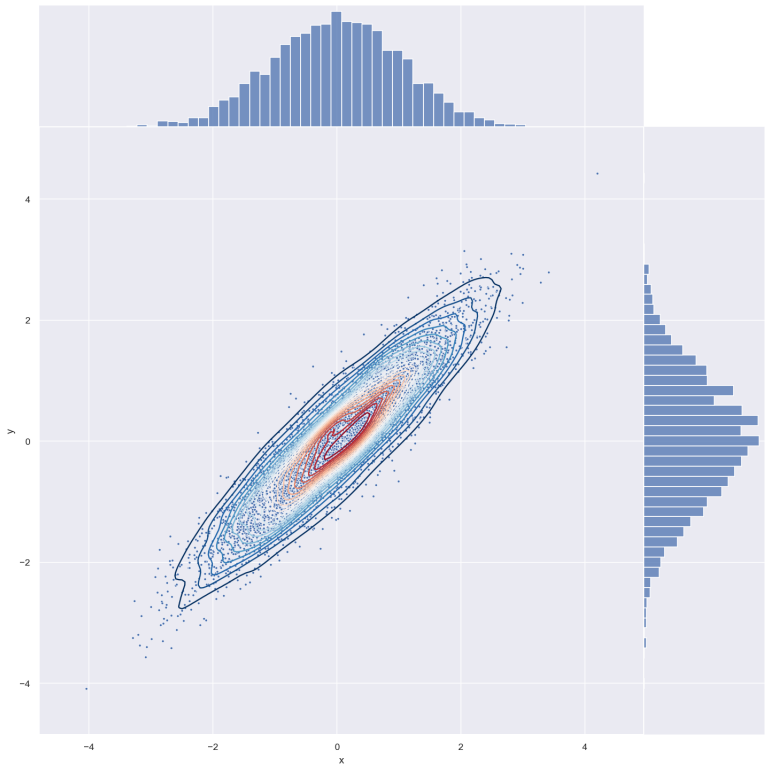

Figure 3.5: Verification of the HMC model to a simple Gaussian. Both histograms demonstrate a typical Gaussian distribution across the samples accepted.

Chapter 4

Apply HMC to the Linear Regression Model

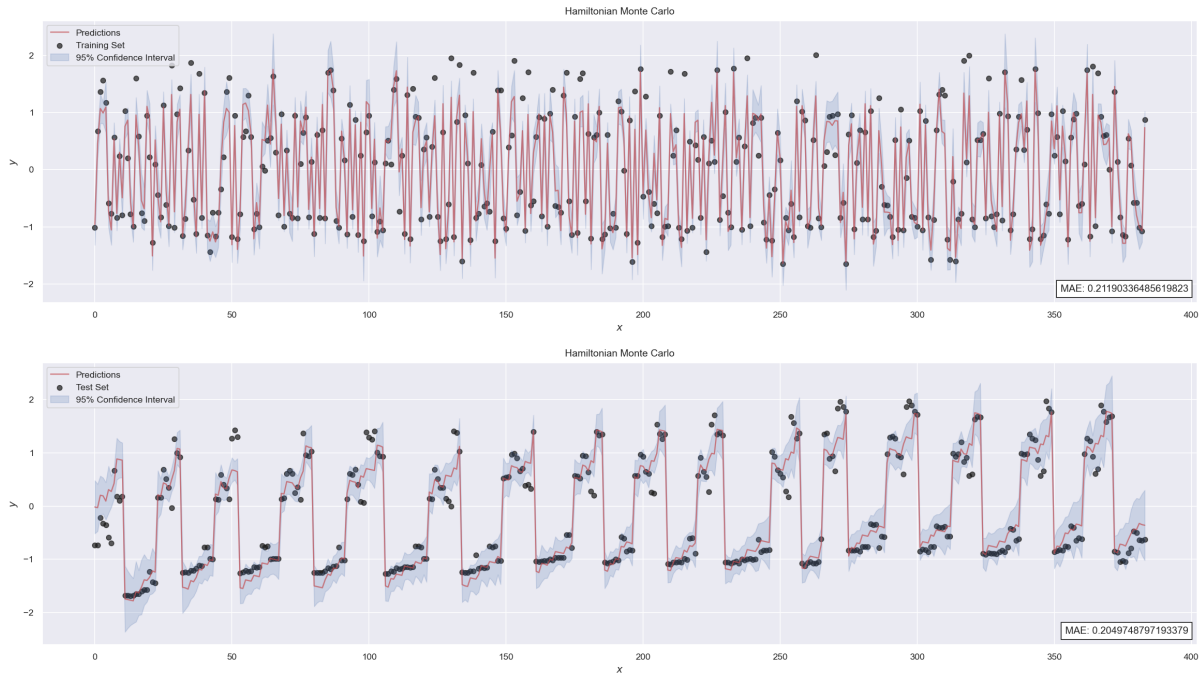

Figure 4.1: The predictions against the training and test set for HMC LR.

Table 4.1: The hyper-parameters used to obtain HMC LR results.

Hyper-Parameters	
R	20000
L	50
eps	0.0065

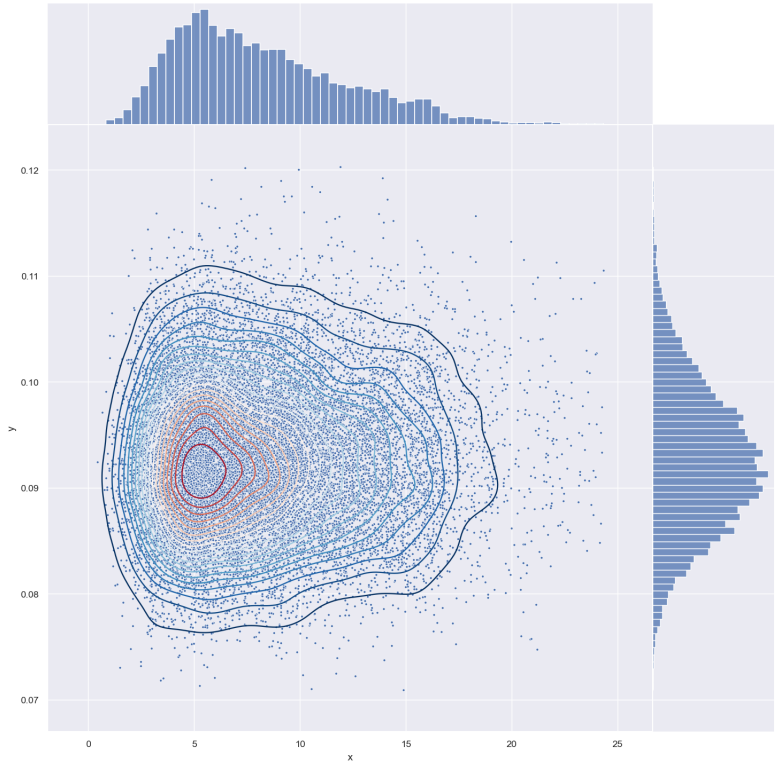

Figure 4.2: The posterior of the HMC model fitted to BLR. The data is clearly non-gaussian, verified by the accepted samples.

Table 4.2: The optimal values of the unknown terms for HMC LR.

Optimal Values	
α	8.215647481615413
β	0.09272017139466739
Bias	0.019619464516079654

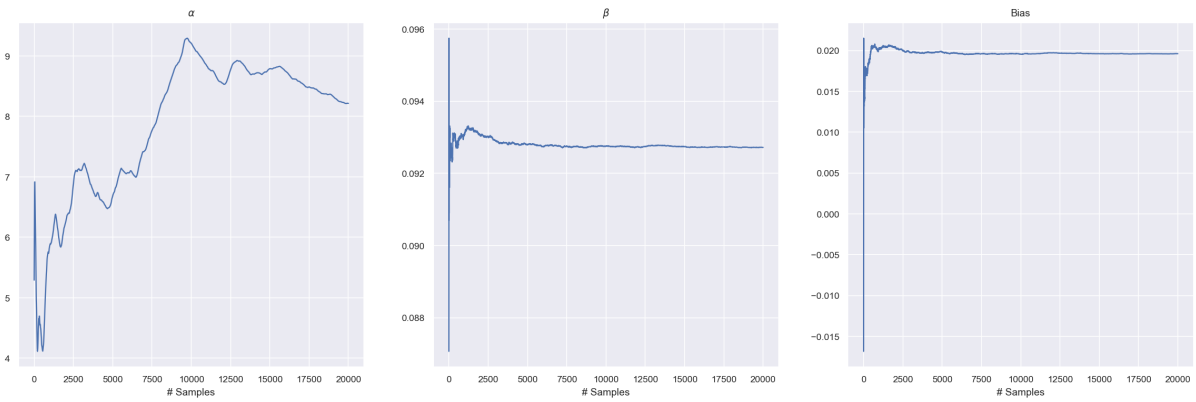

Figure 4.3: The optimal values of the unknown terms for HMC LR.

Table 4.3: The RMSE and MAE of the train and test sets for HMC LR.

Metric	Train Error
RMSE	0.29992302172147134
MAE	0.2119033648562094
Metric	Test Error
RMSE	0.28301448669994267
MAE	0.20497487971933573

Chapter 5

Apply GP to the Linear Regression Model

A Gaussian Process is stochastic (collection of random variables), where every finite linear combination of random variables is normally distributed. The Gaussian Process itself is the joint distribution of all the normally distributed random variables. The prior mean is assumed to be constant and zero, which we have achieved through our preprocessing step with `StandardScaler`.

A note about the kernel choices made for GP:

- RBF - Explains a long term, smooth rising trend.
- WhiteKernel - Explains the correlated noise components.
- ExpSineSquared - Explains periodicity. To allow for variance in periodicity, multiply with an RBF Kernel.

5.1 Default Kernel

The default kernels hyper-parameters are optimised during fitting and uses

```
1 ConstantKernel(1.0, constant_value_bounds="fixed" *
2 RBF(1.0, length_scale_bounds="fixed").
```

Listing 5.1: Default Kernel for the Gaussian Process.

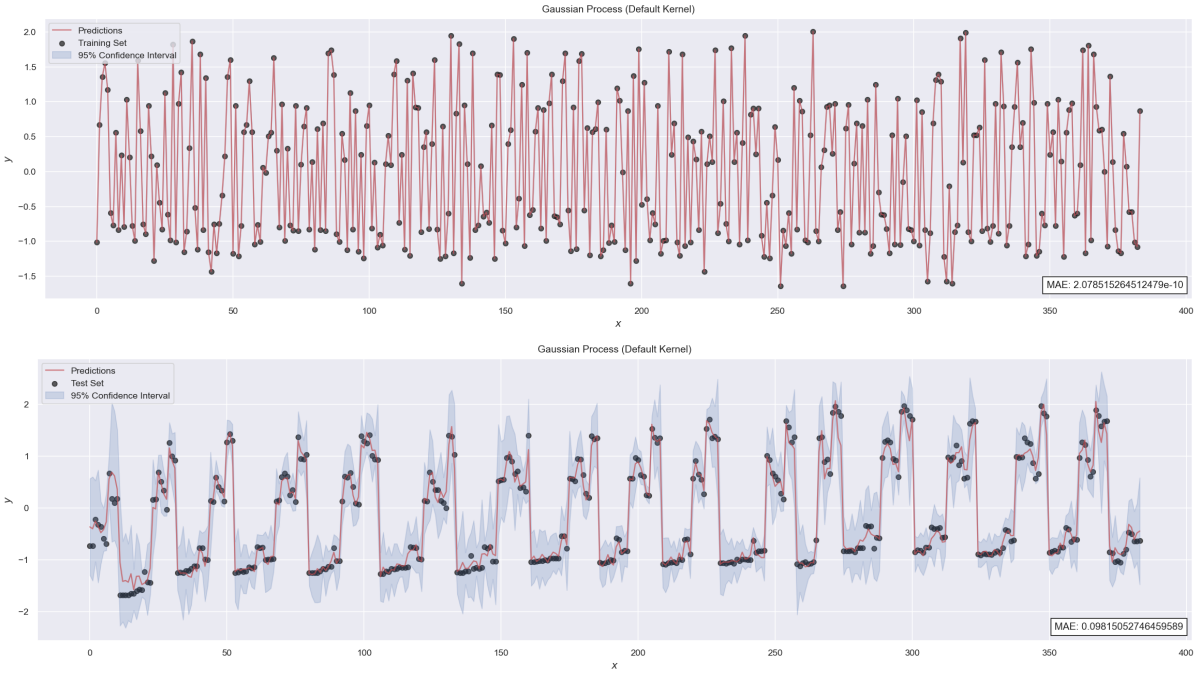

Figure 5.1: The predictions against the training and test set for GP (Default Kernel).

The hyper-parameters of the kernel are optimized during fitting of `GaussianProcessRegressor` by maximizing the log-marginal-likelihood (LML) based on the passed optimizer.

5.2 Custom RBF Kernel

The first kernel I tried was to modify the parameters of the RBF kernel

```
kernel = 1.0 * RBF(length_scale=0.5, length_scale_bounds=(1e-3, 1e2))
```

Listing 5.2: Custom RBF Kernel for the Gaussian Process.

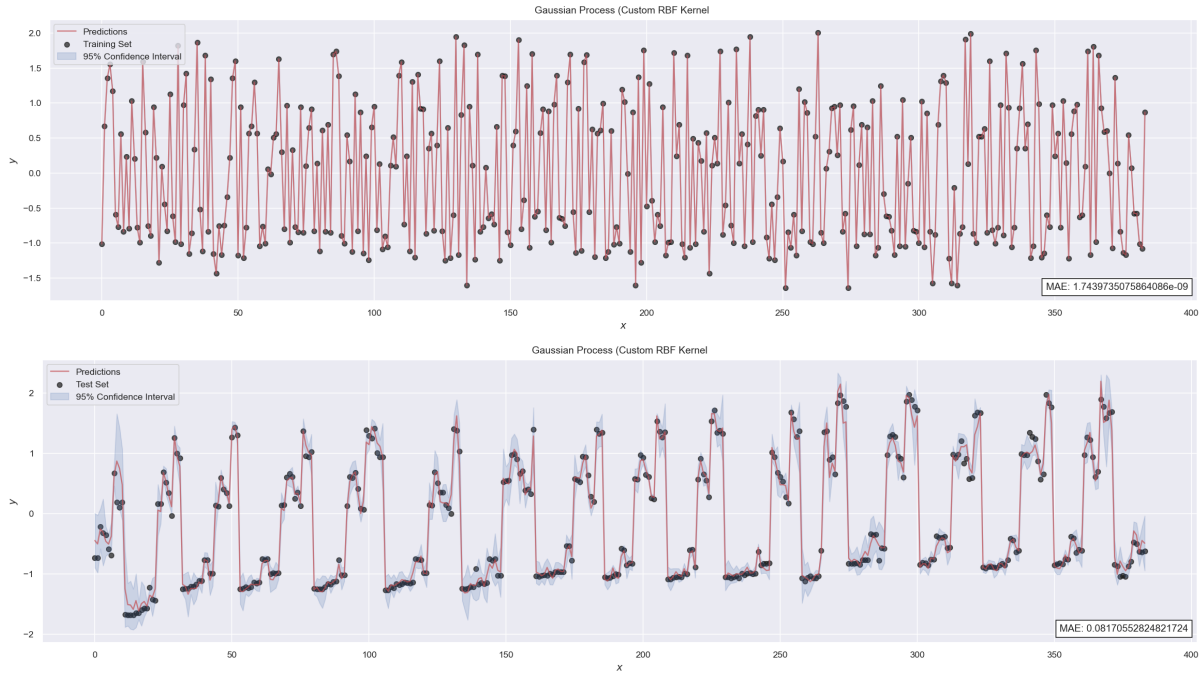

Figure 5.2: The predictions against the training and test set for GP (Custom RBF Kernel).

5.3 Custom RBF Kernel + White Noise

The `sklearn` (Pedregosa et al., 2011) documentation states that:

> "The noise level in the targets can be specified by passing it via the parameter alpha, either globally as a scalar or per datapoint. Note that a moderate noise level can also be helpful for dealing with numeric issues during fitting as it is effectively implemented as Tikhonov regularization, i.e., by adding it to the diagonal of the kernel matrix. An alternative to specifying the noise level explicitly is to include a WhiteKernel component into the kernel, which can estimate the global noise level from the data."

So i chose to use additive noise in the form of WhiteKernel, which allows the model to learn the noise level of the data.

```
kernel = 1.0 * RBF(length_scale=0.5, length_scale_bounds=(1e-3, 1e2)) +
WhiteKernel(noise_level=1e-4, noise_level_bounds=(1e-22, 1e2))
```

Listing 5.3: Custom RBF-WN Kernel for the Gaussian Process.

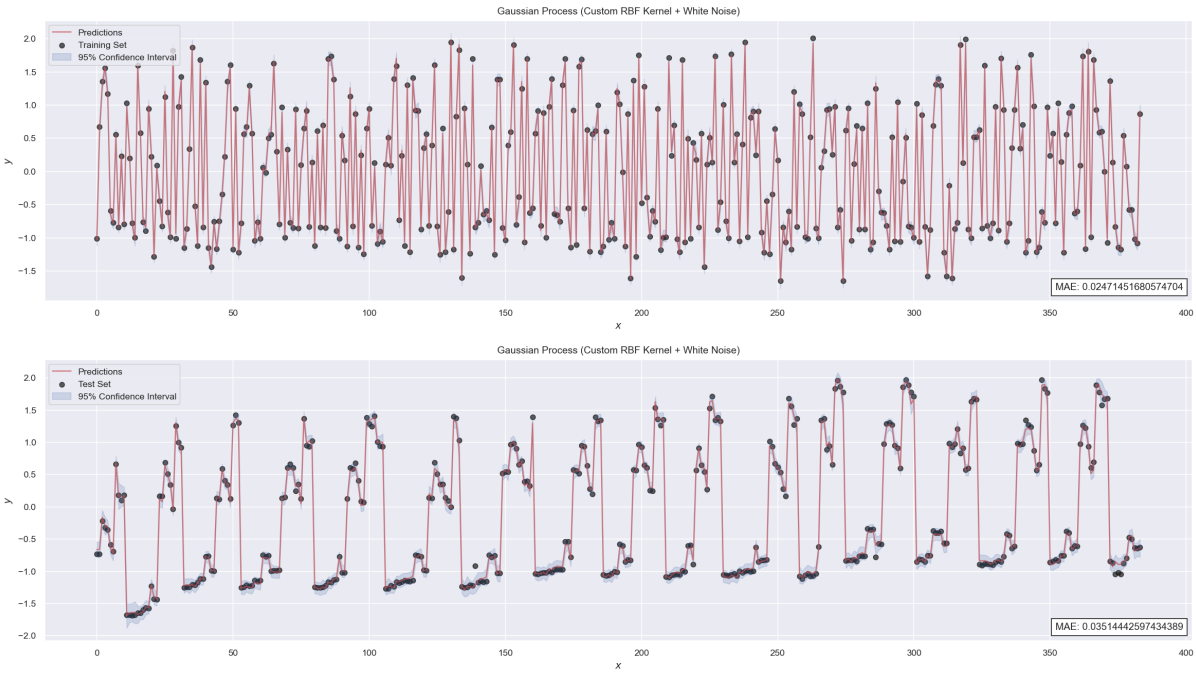

Figure 5.3: The predictions against the training and test set for GP (Custom RBF + WN Kernel).

5.4 Gaussian Process Bayesian Neural Network

For this model I used `tensorflow-probability` (Abadi et al., 2015) to implement a Bayesian Neural Network, where the linear weights are replaced with a probability distribution. The performance is notable worse than the previous method, as initialising the kernel proved to be more difficult than simple additive modes for noise in `sklearn`. I chose not to investigate this further as the model complexity seemed unnecessary, but it could be an avenue for future research.

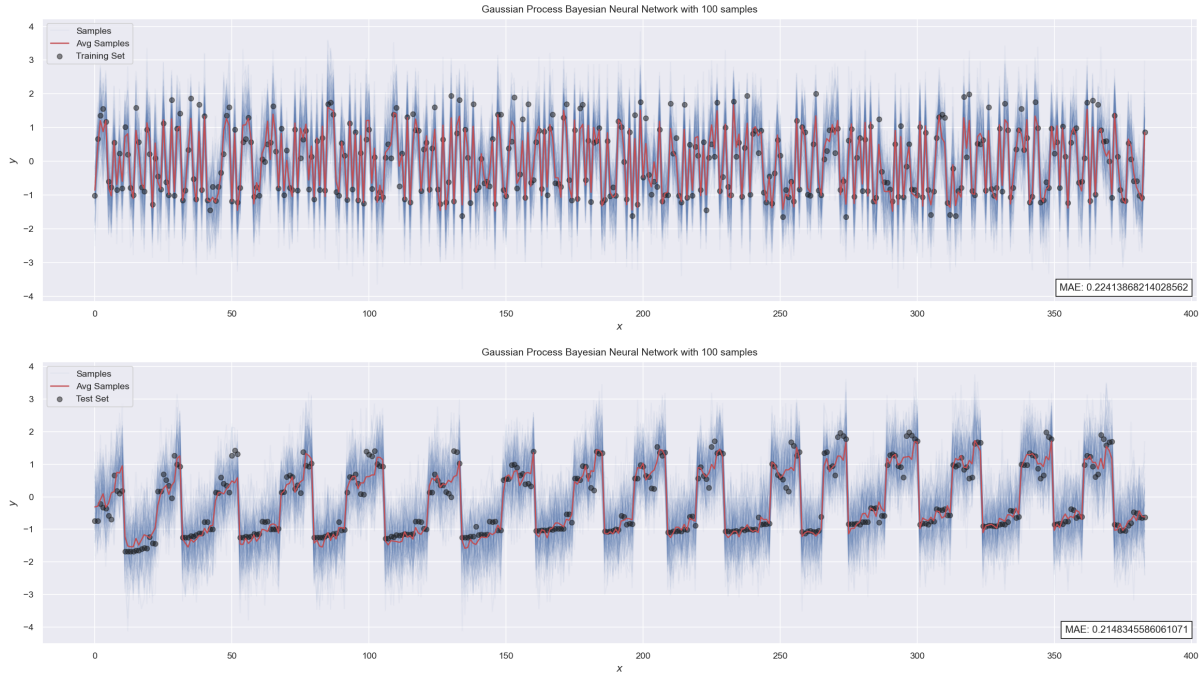

Figure 5.4: The predictions against the training and test set for the BNN.

5.5 Approaches Compared

The initial kernels showed clear signs of overfitting (Table 5.1), as the errors are orders of magnitude apart from the test errors in Table 5.2. Adding noise to the kernel allowed the model to properly generalise and avoid memorising the training set.

Table 5.1: The RMSE and MAE of the training set for GP.

Metric	Default Kernel
RMSE	3.5798617444942677e-10
MAE	2.078515264512479e-10
	Custom RBF Kernel
RMSE	3.6162774775211967e-09
MAE	1.7439735075864086e-09
	Custom RBF + WN Kernel
RMSE	0.03288741425422382
MAE	0.02471451680574704
	BNN
RMSE	1.380201525931696
MAE	0.20633710626738713

Table 5.2: The RMSE and MAE of the test set for GP.

Metric	Default Kernel
RMSE	0.13879017380992578
MAE	0.09815052746459589
	Custom RBF Kernel
RMSE	0.11937871142948968
MAE	0.08170552824821724
	Custom RBF + WN Kernel
RMSE	0.04892887557951222
MAE	0.03514442597434389
	BNN
RMSE	1.380201525931696
MAE	0.20633710626738713

Chapter 6

Results

The results across all models shown in Table 6.1 for the mean average error, suggest that the most accurate model is the Gaussian Process with an RBF + WN kernel. The others have very similar performance, with Hamiltonian Monte Carlo having the second best accuracy likely due to the sampling allowing the most optimal alpha and beta to be found.

The Bayesian Neural Network has a similar mean average error to the other models, but a much higher root mean squared error in Table 6.2, suggesting this model is not well optimised and has large outliers.

Table 6.1: The MAE for all fitted and evaluated models.

Model	Train Error (MAE)	Test Error (MAE)	Absolute Difference
OLS	0.21194498202562453	0.20581054973407428	0.006134432291550251
BLR	0.21193096625675303	0.20578061010720858	0.006150356149544456
VI	0.2119250434809071	0.20512841137791837	0.0067966321029887256
HMC	0.2119033648562094	0.20497487971933573	0.006928485136873663
GP	0.02471451680574704	0.03514442597434389	0.01042990916859685
BNN	0.22413868214028562	0.2148345586061071	0.009304123534178532

Table 6.2: The RMSE for all fitted and evaluated models.

Model	Train Error (RMSE)	Test Error (RMSE)	Absolute Difference
OLS	0.2995679580412746	0.2828601052708685	0.016707852770406095
BLR	0.29956822144275885	0.28285511216629783	0.016713109276461025
VI	0.29982218663158655	0.28297622986213017	0.016845956769456383
HMC	0.29992302172147134	0.28301448669994267	0.01690853502152867
GP	0.03288741425422382	0.04892887557951222	0.0160414613252884
BNN	1.3763220146336062	1.394754734543617	0.018432719910010942

The model with the smallest absolute difference between the train and test set for RMSE was GP, which suggests it is fairly well generalised with fewer outliers than the other models. When using the MAE however, the best performing model was OLS closely followed by BNN. The MAE is robust to outliers, thus gives us a better idea of the model generalization without noise.

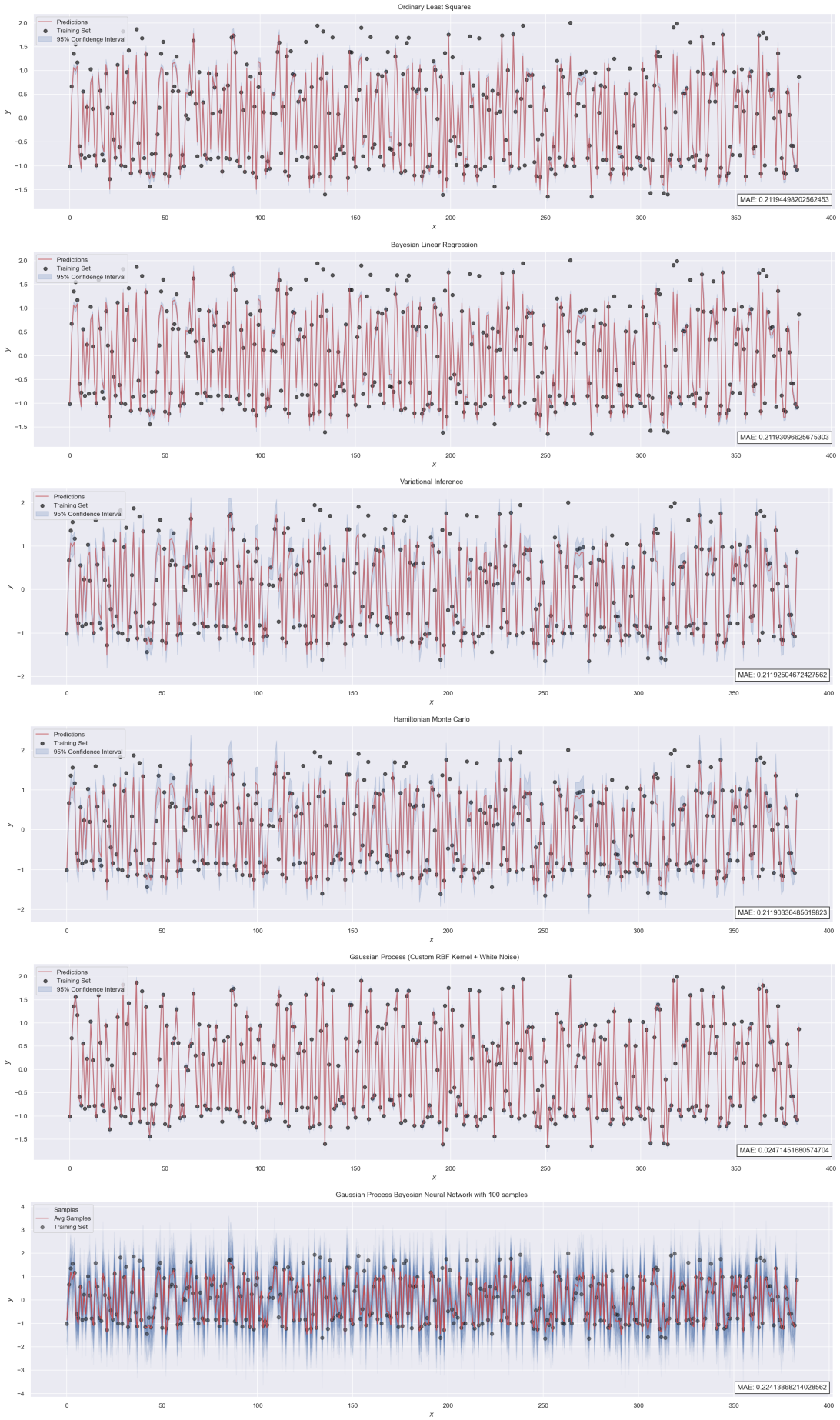

Figure 6.1: The predictions against the training set for all models.

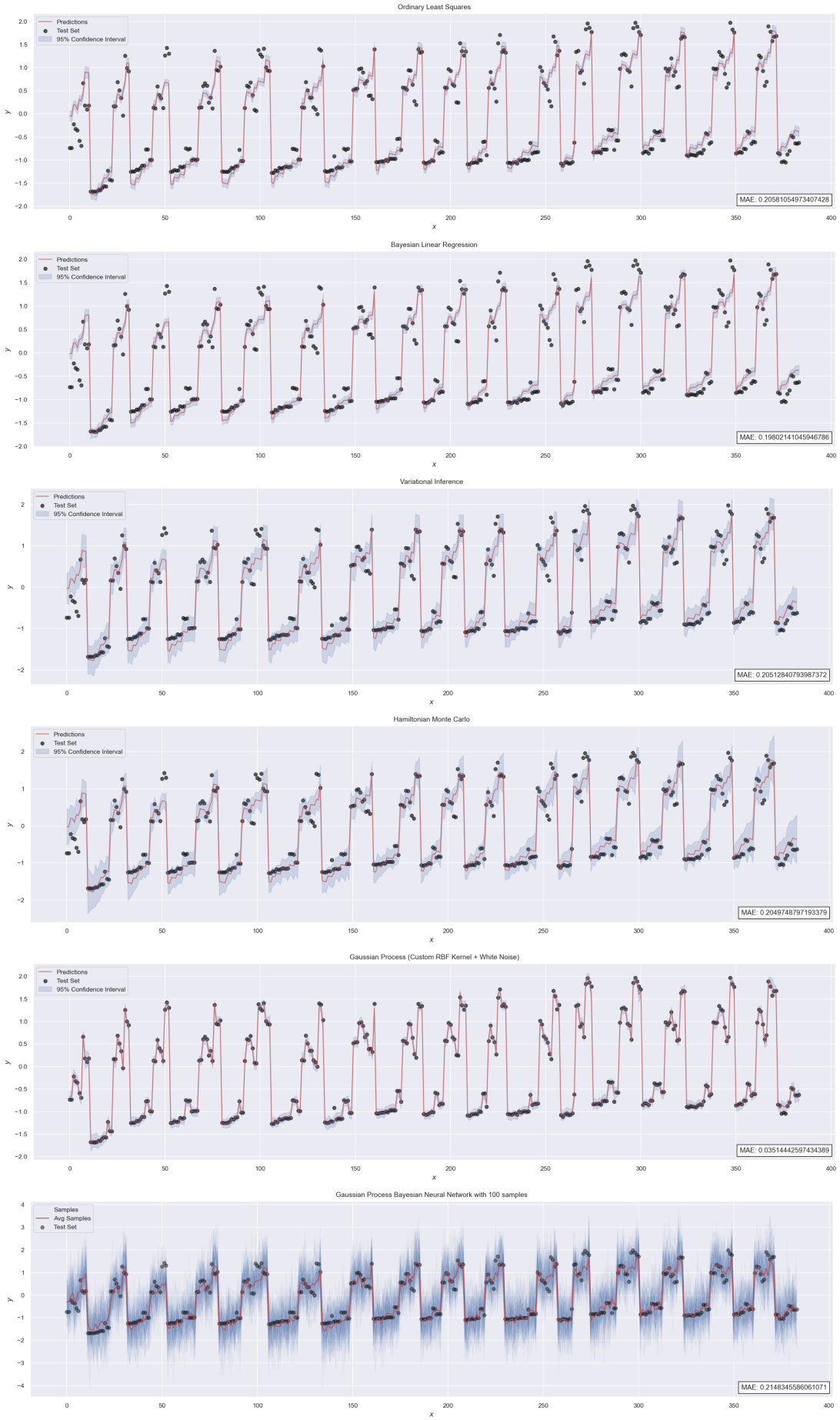

Figure 6.2: The predictions against the test set for all models.

Chapter 7

Conclusion

Figure 4.2 demonstrates that there is a bias in the distribution on the x and y axes, where the skew is higher along x. This deviation suggests that the dataset itself is non-gaussian, as opposed to Figure 3.5, and linear models are potentially suitable for prediction.

Of all the methods used, shown in Figure 6.2, the *Gaussian Process* is the clear winner, likely due to the ability to refine a custom kernel to allow insights into what the distribution might be to be given almost like a prior. Accuracy across the board was quite uniform (Table 6.1), which is to be expected given their model similarities. The advantage of *Variational Inference* was the improved computation time to find the optimal α and β. *Hamiltonian Monte Carlo* had an extra overhead of computation required to sample from the posterior, but this resulted in a slight accuracy boost compared to the previous methods.

The least performing model was the *Gaussian Process Bayesian Neural Network*, but with limited time constraints it became beyond the scope of this analysis. Future work could involve taking a review (Shridhar, Laumann and Liwicki, 2019) of this research area, in particular the kernel used for the network to try and incorporate noise, as was the case for improvement seen in the standard *Gaussian Process*.

We report a mean average error of 0.035 for our best model (Table 6.1), in contrast to 0.51 reported using *Random Forests* by Tsanas and Xifara (2012), however their predicted values were inverted back to the original values in their analysis. Advancements in machine learning algorithm implementation, and the mass adoption of python have allowed the community to create tools and packages to allow much faster prototyping of different models since the initial paper by Tsanas and Xifara (2012).

The difference in the training and test accuracy was taken to assess the level of overfitting and underfitting. The best absolute difference for the *mean average error* (robust to outliers) was found to be *ordinary least squares* and for the *mean squared error* (amplifies the value associated with outliers), the *Gaussian Process*. This suggests that GP had less outliers and thus a better spread of the data compared to the other models.

We have shown that a stochastic approach to this dataset gave us access to confidence intervals, of which a comparison is illustrated in Figure 6.2. We can see that in general the confidence interval in the *Gaussian Process* has a much smaller range, thus we can be more confident in the predicted values for this model.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. and Zheng, X., 2015. TensorFlow: Large-scale machine learning on heterogeneous systems [Online]. Software available from tensorflow.org. Available from: https://www.tensorflow.org/.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E., 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research* [Online], 12, pp.2825–2830. Available from: https://scikit-learn.org/.

Shridhar, K., Laumann, F. and Liwicki, M., 2019. A Comprehensive guide to Bayesian Convolutional Neural Network with Variational Inference [Online]. [Online], pp.1–38. 1901.02731, Available from: http://arxiv.org/abs/1901.02731.

Tsanas, A. and Xifara, A., 2012. Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. *Energy and buildings* [Online], 49, pp.560–567. Available from: https://doi.org/10.1016/j.enbuild.2012.03.003.