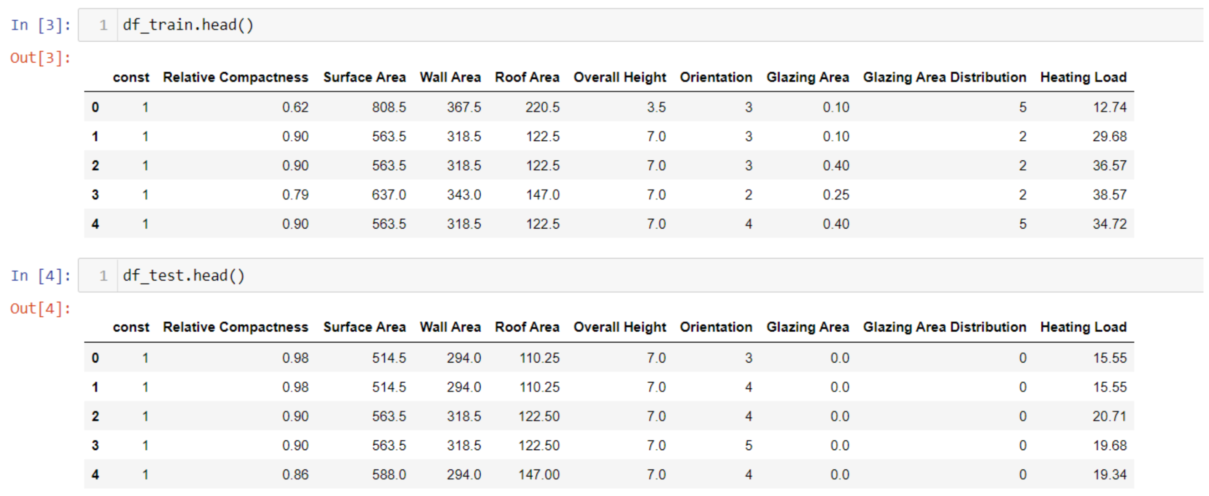
Bayesian Machine Learning

Final project report

I-Hsiu Chiang
UNIVERSITY OF BATH

1 Exploratory analysis

For the exploratory analysis, I begin it by importing the training and the testing set into the panda's data frame, which is shown below:



```
In [3]: df_train.head()
Out[3]:
```

	const	Relative Compactness	Surface Area	Wall Area	Roof Area	Overall Height	Orientation	Glazing Area	Glazing Area Distribution	Heating Load
0	1	0.62	808.5	367.5	220.5	3.5	3	0.10	5	12.74
1	1	0.90	563.5	318.5	122.5	7.0	3	0.10	2	29.68
2	1	0.90	563.5	318.5	122.5	7.0	3	0.40	2	36.57
3	1	0.79	637.0	343.0	147.0	7.0	2	0.25	2	38.57
4	1	0.90	563.5	318.5	122.5	7.0	4	0.40	5	34.72

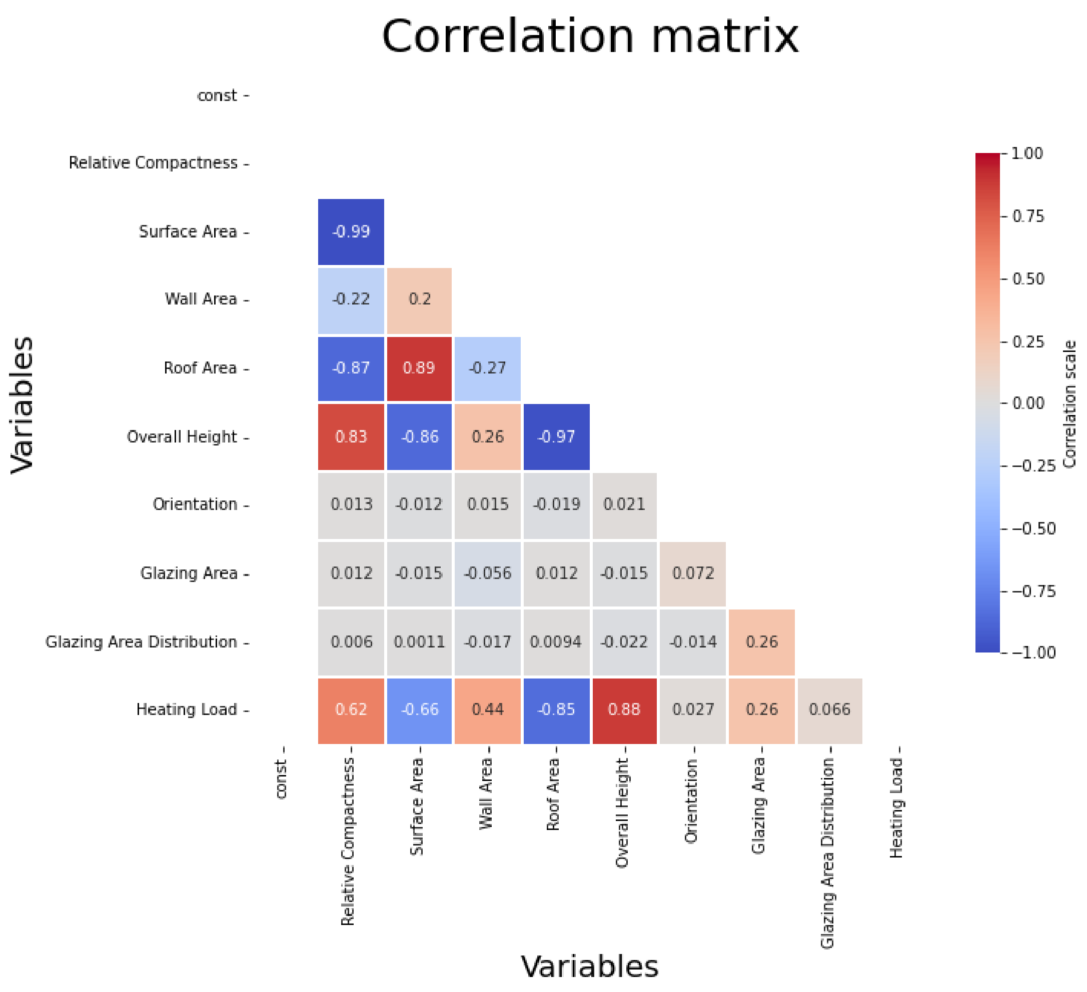
```
In [4]: df_test.head()
Out[4]:
```

	const	Relative Compactness	Surface Area	Wall Area	Roof Area	Overall Height	Orientation	Glazing Area	Glazing Area Distribution	Heating Load
0	1	0.98	514.5	294.0	110.25	7.0	3	0.0	0	15.55
1	1	0.98	514.5	294.0	110.25	7.0	4	0.0	0	15.55
2	1	0.90	563.5	318.5	122.50	7.0	4	0.0	0	20.71
3	1	0.90	563.5	318.5	122.50	7.0	5	0.0	0	19.68
4	1	0.86	588.0	294.0	147.00	7.0	4	0.0	0	19.34

Figure 1: Dataframe for training and test dataset

Due to the variety of different ranges in the values, I standardized all the values (except heat load) and split both train and test data sets to x_train, x_test, y_train, and y_test data sets. I plotted a correlation matrix between all the variables to observe the linearity of each variable. Out of all 8 variables, there are 8 correlated cases where the correlation reached above 0.80, with the other 2 reaching around 0.60 for the correlation.

5 variables were able to get a decent correlation with heat load, with roof area and overall height being the highest correlated variable compared to the others. Relative compactness scores a value of 0.62, surface area scores -0.66, wall area scores 0.44, roof area scores -0.85 and overall height scores a strong correlation of 0.88, the highest of all 5. The glazing area has a weak correlation of 0.26. In my opinion, based on the correlation matrix in figure 2, orientation and glazing area distribution can be considered irrelevant for predicting the heat load, as the values do not seem to correlate with both mentioned variables.



	const	Relative Compactness	Surface Area	Wall Area	Roof Area	Overall Height	Orientation	Glazing Area	Glazing Area Distribution	Heating Load
const										
Relative Compactness										
Surface Area		-0.99								
Wall Area		-0.22	0.2							
Roof Area		-0.87	0.89	-0.27						
Overall Height		0.83	-0.86	0.26	-0.97					
Orientation		0.013	-0.012	0.015	-0.019	0.021				
Glazing Area		0.012	-0.015	-0.056	0.012	-0.015	0.072			
Glazing Area Distribution		0.006	0.0011	-0.017	0.0094	-0.022	-0.014	0.26		
Heating Load		0.62	-0.66	0.44	-0.85	0.88	0.027	0.26	0.066	

Figure 2: Correlation matrix

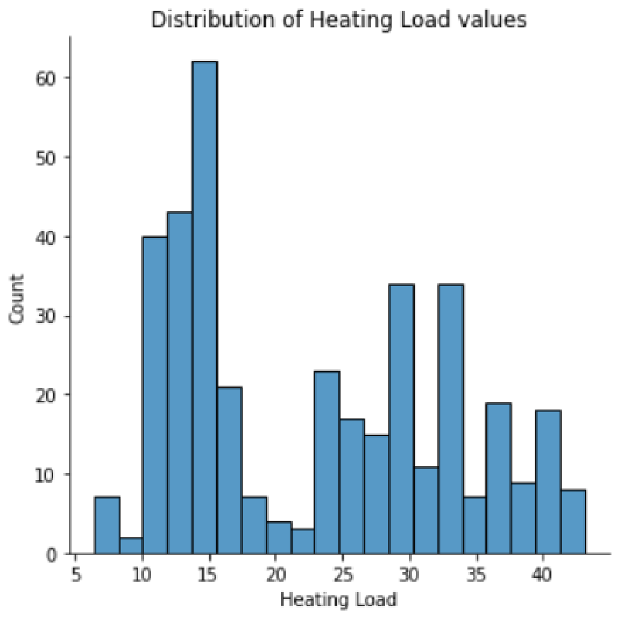


Figure 3: Heat load count distribution

A predictive baseline has also been plotted using the training set by least-squares, and its prediction accuracy has been assessed on both train and test data sets, see the plot below:

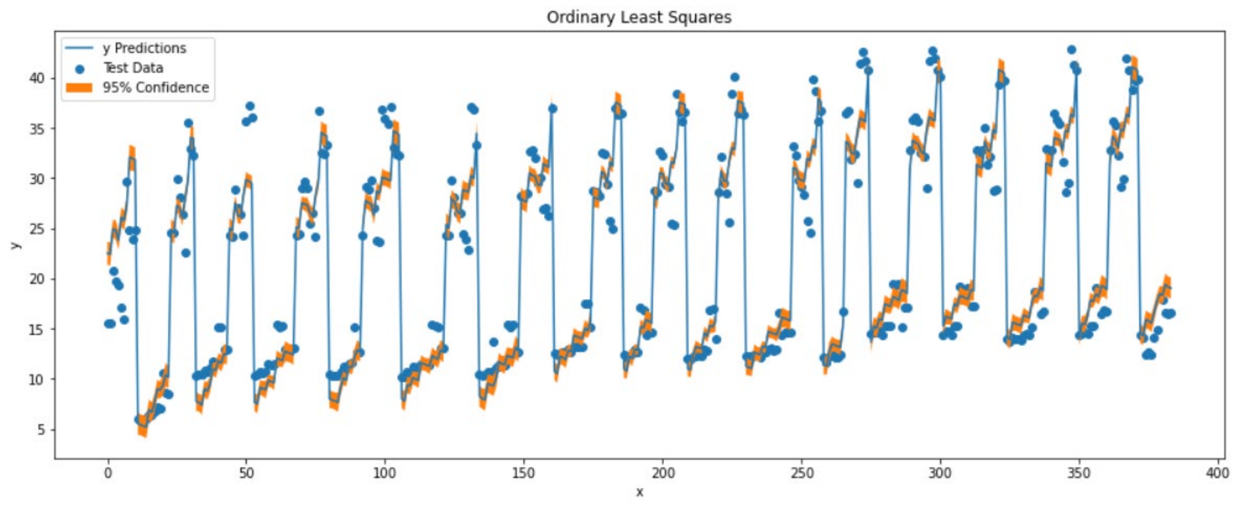


Figure 4: Ordinary least square predictive baseline

The mean absolute error (MAE) for the training data set is 2.131 and the testing set is 2.069.

2 Bayesian linear regression

2.1 Type-II maximum likelihood

Using the product rule, the posterior can be re-written as:

$$p(w, \alpha, \beta | y) \equiv p(w | y, \alpha, \beta)\, p(\alpha, \beta | y)$$

The second term of this equation cannot be computed directly, therefore, to find the most probable values of alpha and beta, it is a requirement to maximize this distribution:

$$p(\alpha, \beta | y) = \frac{p(y|\alpha,\beta)p(\alpha)p(\beta)}{p(y)}$$

Since the distribution has already been assumed as a flat uninformative prior to alpha and beta, therefore within the distribution $p(y|\alpha,\beta)$ is maximized through logarithm:

$$\alpha, \beta = argmax\ log\ p(y|\alpha,\beta)$$

Normally, the covariance matrix that is part of the hyperparameter of the distribution is defined as:

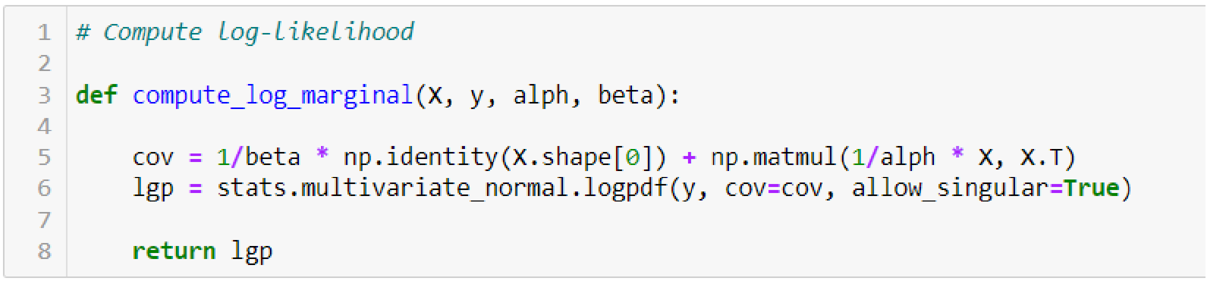
$$\Sigma = \beta I\ +\ \alpha^{-1}\Phi\Phi^T$$

But the precision is given as $\alpha\ =\ \frac{1}{\sigma_w^2}$ and $\beta\ =\ \frac{1}{\sigma_\epsilon^2}$, the covariance in the distribution would be stated as:

$$\Sigma = \frac{1}{\beta} I\ +\ \frac{1}{\alpha^{-1}}\Phi\Phi^T$$

This is then calculated through the log of multivariate normal distribution, of which the multivariate normal distribution is stated as:

$$\frac{1}{(2\pi)^{\frac{n}{2}}\Sigma^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(y-\mu)^T\Sigma^{-1}(y-\mu)\right)$$



```python
# Compute log-likelihood

def compute_log_marginal(X, y, alph, beta):

    cov = 1/beta * np.identity(X.shape[0]) + np.matmul(1/alph * X, X.T)
    lgp = stats.multivariate_normal.logpdf(y, cov=cov, allow_singular=True)

    return lgp
```

Figure 5: Log marginal likelihood

After looping through 10000 different combinations of alpha and beta values, the most probable alpha is 0.0117 with the beta being 0.1084. Figure 6 shows the log-posterior distribution with log alpha on the x-axis and log beta on the y axis. Within the contour plot, a red dot is indicated the most probable log alpha and log beta within the distribution. The MAE of the training data is 2.131 and the testing data is 2.069. The RMSE of the training data is 3.012 and the testing data is 2.844.

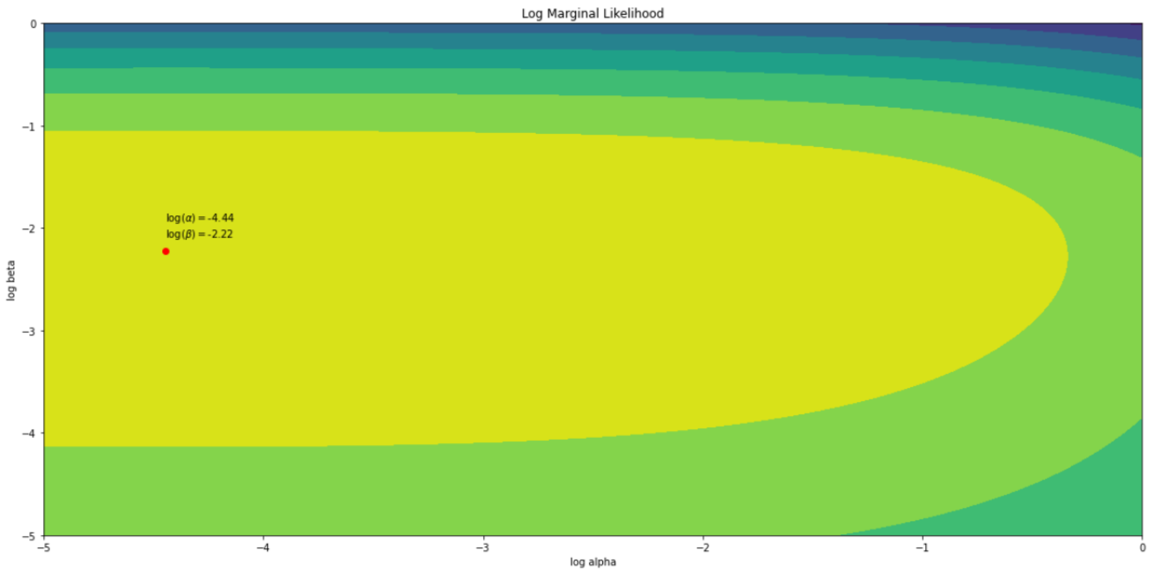


Figure 6: Most probable alpha and beta

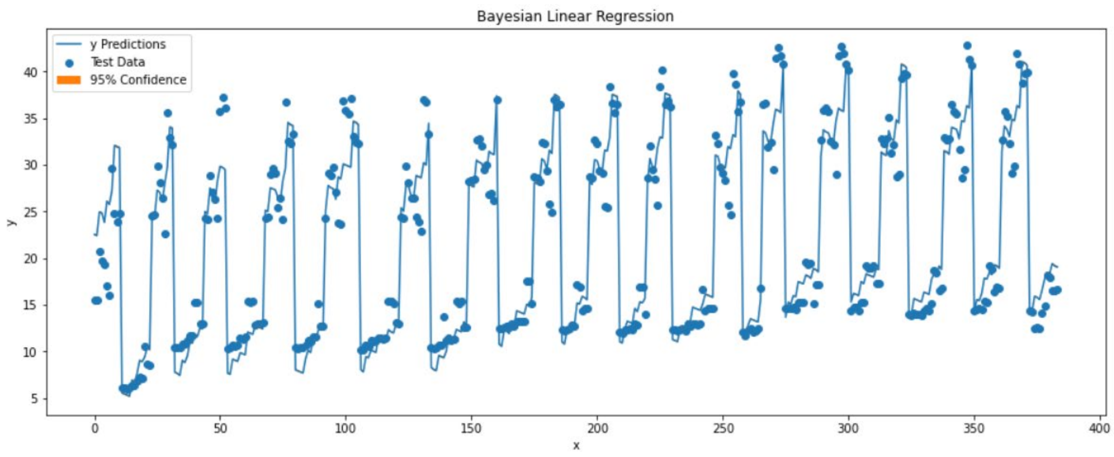


Figure 7: Baseline prediction plot for Bayesian linear regression

2.2 Variational inference

The variational method is one of the deterministic approximation methods that can be used to find the proposal distribution Q(θ) that is similar to the unknown posterior distribution P(θ). The distribution starts with:

$$q(w, \alpha, \beta) = q(w)q(\beta)q(\alpha)$$

This then can be worked out as:

$$q(w) \propto exp\langle log\, p(y, w, \alpha, \beta)\rangle_{\alpha,\beta}$$

$$q(\alpha) \propto exp\langle log\, p(y, w, \alpha, \beta)\rangle_{w,\beta}$$

$$q(\beta) \propto exp\langle log\, p(y, w, \alpha, \beta)\rangle_{w,\alpha}$$

Where the prior is stated as:

$$q(w) = N(w|\mu_N, \Sigma_N)$$

$$q(\alpha) = Gam(\alpha\, |a_0, b_0)$$

$$q(\beta) = Gam(\beta|c_0, d_0)$$

To calculate the above, the initial parameters can be deciphered into:

$$\mu_N = (X^T X + \beta\alpha I)^{-1} X^T y$$

$$\Sigma_N = \beta(X^T X + \beta\alpha I)^{-1}$$

$$a_N = a_0 + \left(\frac{M}{2}\right)$$

$$b_N = b_0 + \frac{1}{2}\langle w^T w\rangle_{q_\alpha(\alpha)} = b_0 + \frac{1}{2}(\mu_N^T \mu_N + tr(\Sigma_N))$$

$$c_N = c_0 + \left(\frac{N}{2}\right)$$

$$d_n = d_0 + \frac{1}{2}(y - wx)^T (y - wx)$$

M and N are the shape of the data given, which in this case M is 9 and N is 384. The calculation would then begin looping through b_N and d_N and through these parameters, both expected alpha and beta can be found by:

$$\alpha = \frac{a_N}{b_N}$$

$$\beta = \frac{c_N}{d_N}$$

The expected alpha is calculated as 0.0119 and the expected beta is 0.1103. This is both shown on the contour plot below in figure 8:

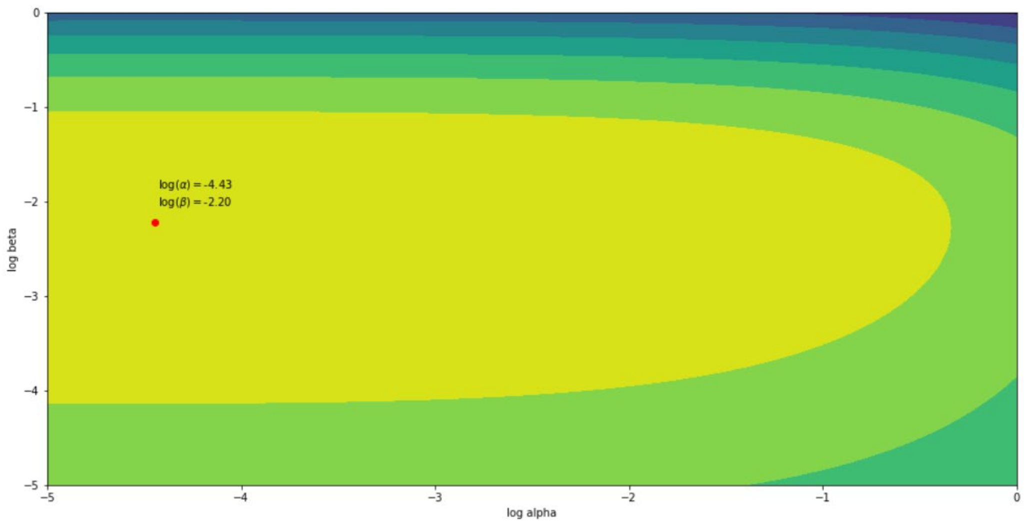


Figure 8: VI expected alpha and beta

From this, the RMSE of the training set is 3.0116 and the testing set is 2.8436. The MAE of the training set is 2.1307 and the testing set is 2.0690.

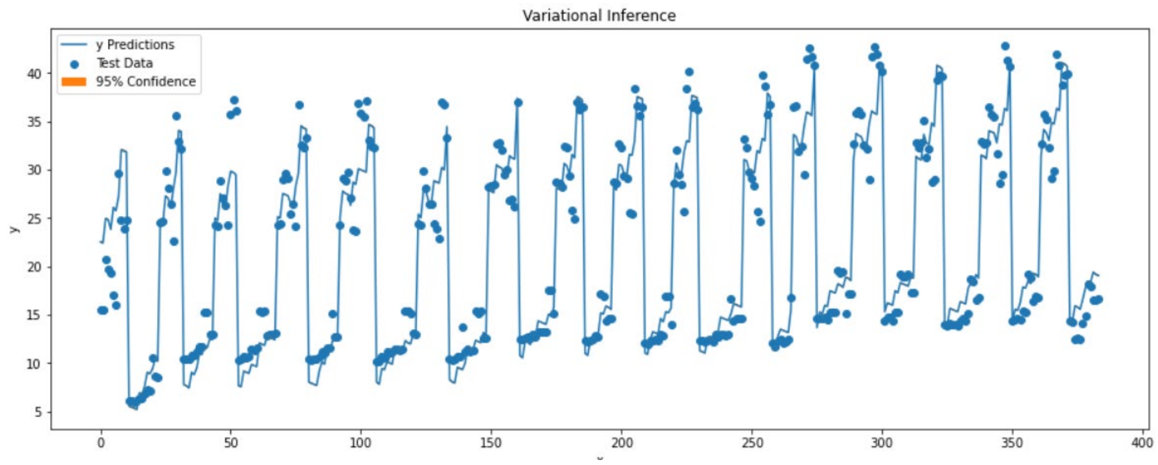


Figure 9: Baseline prediction plot for variational inference

3 Verify HMC on a standard 2D Gaussian example

At the beginning of the energy function, it takes in the x and mu values and inputs them into the negative logarithm of the multivariate gaussian function, which the multivariate gaussian distribution is defined as:

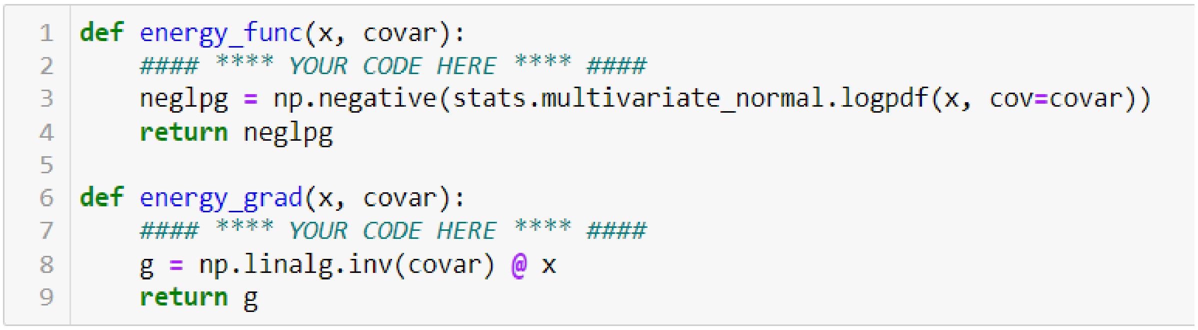
$$-\frac{1}{(2\pi)^{\frac{n}{2}}\Sigma^{\frac{1}{2}}}\exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

$$-\ln\left(\frac{1}{(2\pi)^{\frac{n}{2}}\Sigma^{\frac{1}{2}}}exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)\right)$$

$$-\ln\left(\frac{1}{(2\pi)^{\frac{n}{2}}\Sigma^{\frac{1}{2}}}\right)-\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

To get the gradient of the distribution, the above equation is differentiated with respect to x, therefore given that mu is 0, the energy function gradient would simply be:

$$g = \Sigma^{-1}x$$



```python
def energy_func(x, covar):
    #### **** YOUR CODE HERE **** ####
    neglpg = np.negative(stats.multivariate_normal.logpdf(x, cov=covar))
    return neglpg

def energy_grad(x, covar):
    #### **** YOUR CODE HERE **** ####
    g = np.linalg.inv(covar) @ x
    return g
```

Figure 10: The algorithm for the 2D Gaussian example

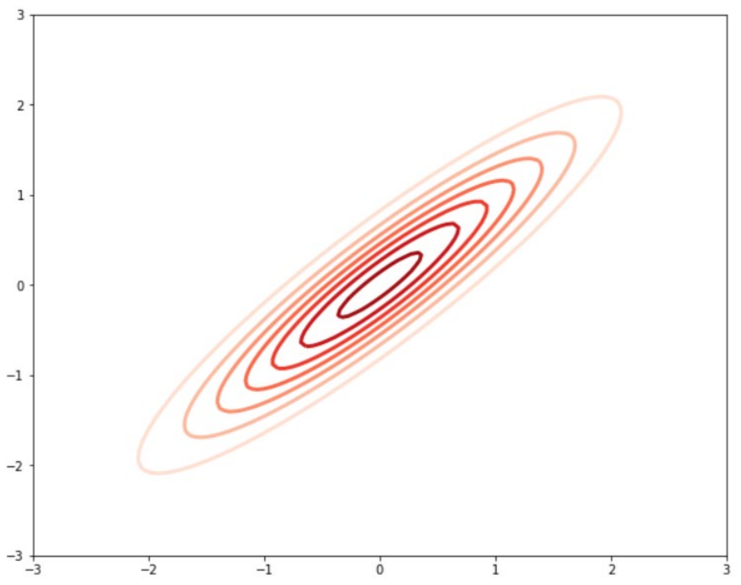
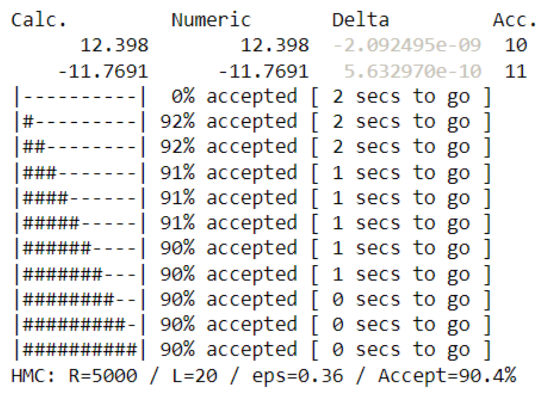


Figure 11: Standard 2D Gaussian example running on HMC



```
Calc.          Numeric        Delta          Acc.
      12.398         12.398   -2.092495e-09  10
    -11.7691       -11.7691    5.632970e-10  11
|----------|  0% accepted [ 2 secs to go ]
|#---------| 92% accepted [ 2 secs to go ]
|##--------| 92% accepted [ 2 secs to go ]
|###-------| 91% accepted [ 1 secs to go ]
|####------| 91% accepted [ 1 secs to go ]
|#####-----| 91% 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 ]
|##########| 90% accepted [ 0 secs to go ]
HMC: R=5000 / L=20 / eps=0.36 / Accept=90.4%
```

Figure 12: Validating the gradient function for 2D Gaussian example

For the hyperparameter selection, L is set as 20, as it is sufficient for this small example. R is also set at a lower value of 5000 as this example is not as complicated. For eps, it is believed that 0.36 is enough for this task. The final result accepted percentage was able to reach 90% using these parameters. 3 parameters mentioned above can be tuned slightly above 80% using higher R, higher L or higher eps, but it is observed that the parameters set for this example are good enough for this task and no further adjustment is required.

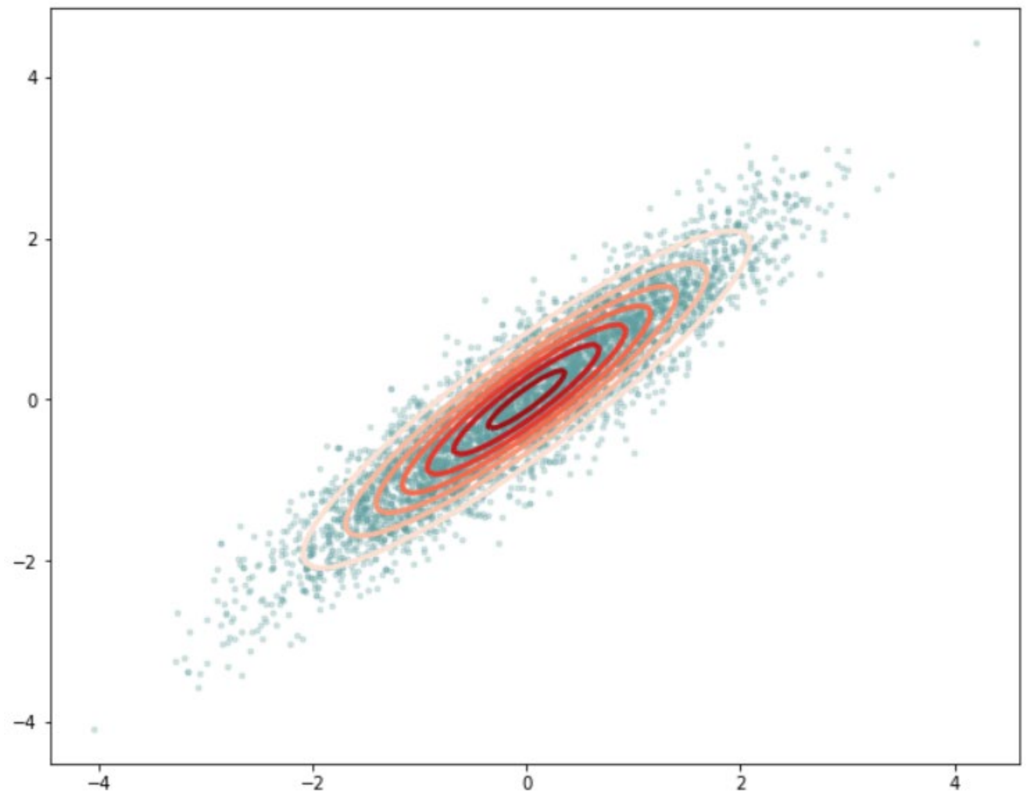


Figure 13: The HMC sampling on the 2D Gaussian example

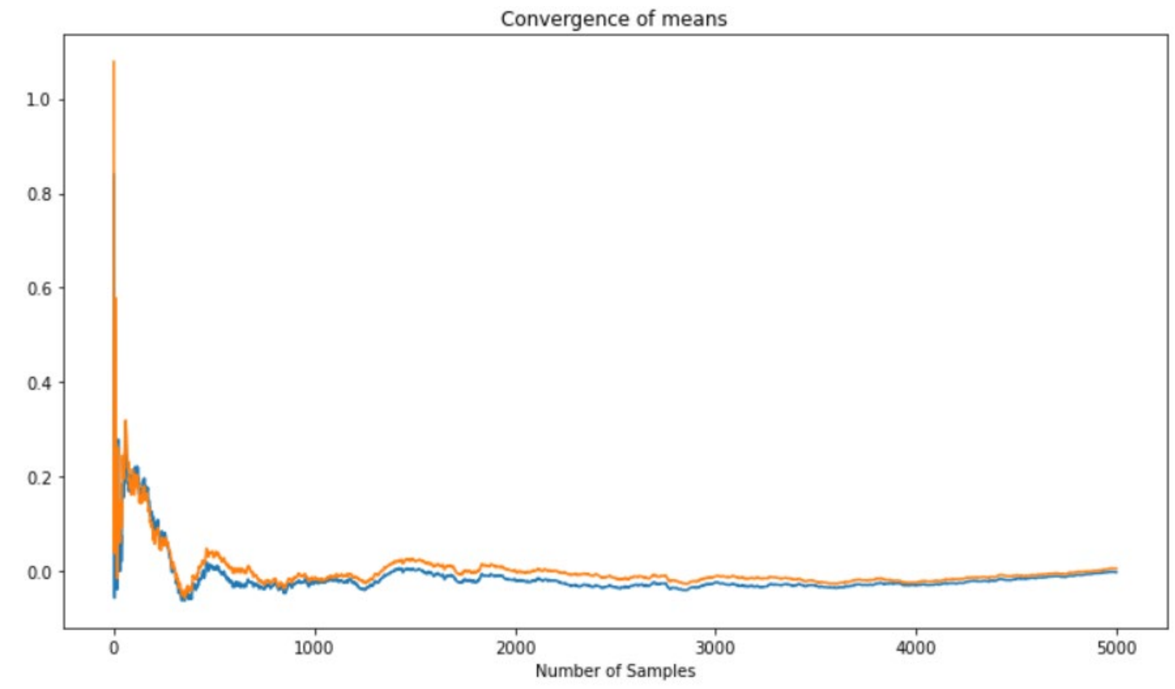


Figure 14: The convergence of the mean on the 2D Gaussian example

4 Apply HMC to the Linear Regression Model

The posterior distribution that is being sampled is the $p(w, \alpha, \beta|y)$, which can be written as:

$$p(w, \alpha, \beta|y) = p(y|w, \beta, \alpha)p(w|\alpha)$$

Where the prior can be stated as:

$$p(y|w, \beta, \alpha) = \left(\sqrt{\frac{\alpha}{2\pi}}\right)^M \exp\left(-\frac{\alpha}{2}w^T w\right)$$

$$p(w|\alpha) = \left(\sqrt{\frac{\beta}{2\pi}}\right)^N \exp\left(-\frac{\beta}{2}(y - Xw)^T(y - Xw)\right)$$

To compute this into the energy function needed, a logarithm is needed for this distribution, which is derived as:

$$\log p(y|w, \beta, \alpha) = \log\left(\sqrt{\frac{\alpha}{2\pi}}\right)^M - \frac{\alpha}{2}w^T w$$

$$\log p(w|\alpha) = \log\left(\sqrt{\frac{\beta}{2\pi}}\right)^N - \frac{\beta}{2}(y - Xw)^T(y - Xw)$$

Combining both equations, the log probability would be:

$$\log p(w, \alpha, \beta|y) = \log\left(\sqrt{\frac{\alpha}{2\pi}}\right)^M - \frac{\alpha}{2}w^T w + \log\left(\sqrt{\frac{\beta}{2\pi}}\right)^N - \frac{\beta}{2}(y - Xw)^T(y - Xw)$$

$$= \frac{\mathrm{M}}{2}\log\alpha - \frac{M}{2}\log 2\pi - \frac{\alpha}{2}w^T w + \frac{\mathrm{N}}{2}\log\beta - \frac{M}{2}\log 2\pi - \frac{\beta}{2}(y - Xw)^T(y - Xw)$$

Taking the negative of the entire equation, it would be the energy function written in the codes:

$$-\log p(w, \alpha, \beta|y) \\ = -\frac{\mathrm{M}}{2}\log\alpha + \frac{M}{2}\log 2\pi + \frac{\alpha}{2}w^T w - \frac{\mathrm{N}}{2}\log\beta + \frac{M}{2}\log 2\pi + \frac{\beta}{2}(y - Xw)^T(y - Xw)$$

This means that the gradient of the energy function with respect to w, alpha and beta would be:

$$\frac{\partial p}{\partial \alpha} = -\frac{M}{2\alpha} + \frac{1}{2}w^T w$$

$$\frac{\partial p}{\partial \beta} = -\frac{N}{2\beta} + \frac{1}{2}(y - Xw)^T(y - Xw)$$

$$\frac{\partial p}{\partial w} = -\alpha w + \beta(y - Xw)x$$

Through trial and error, the R-value is set to 20000. The reason being is that it is possible to visualise if the alpha and beta values converge and stabilized. The L is set as 25 as usual. The reason that it is not set higher is that it would hinder the performance of the sampling. The eps were set at 0.0003, and both initial alpha and beta values are set at 0.1. There have been attempts to reduce the initial value to 0.001, but error messages pop up soon after the algorithm was run. Using the mentioned parameters, 89% acceptance was achieved in the HMC sampler, with the expected alpha converging at 0.0158 and the expected beta of 0.1084. The bias converges at 0.2711. The MAE of the training data was 2.1612, and the test data was 2.1040. The RMSE training data was 3.0191 and the testing data is 2.8526.

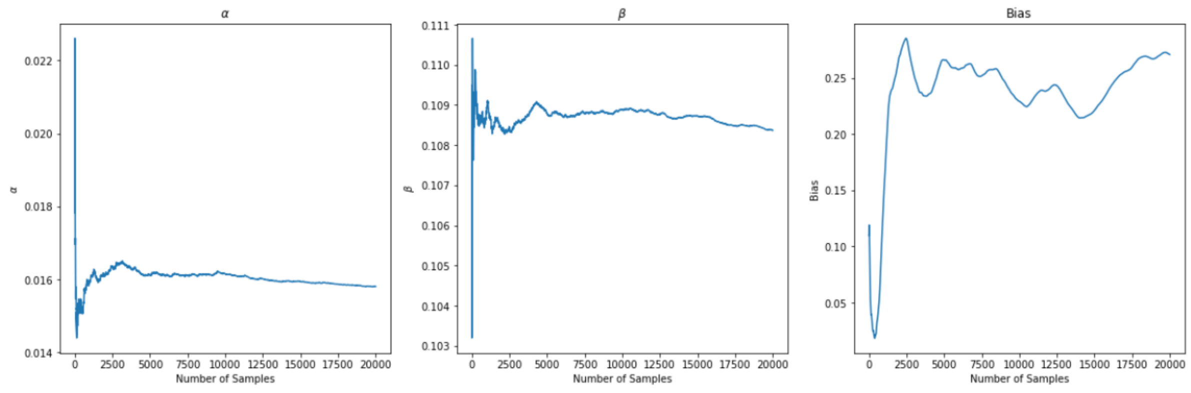
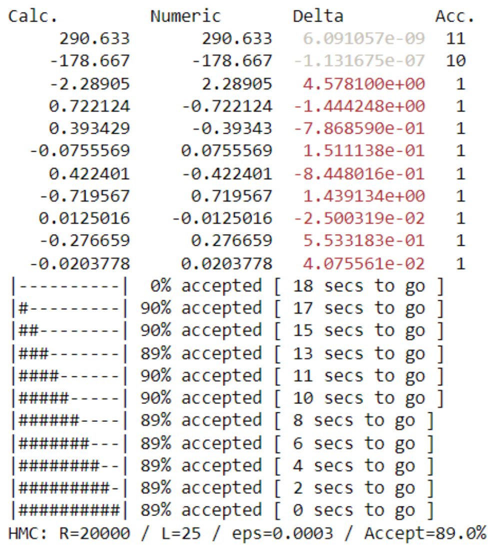
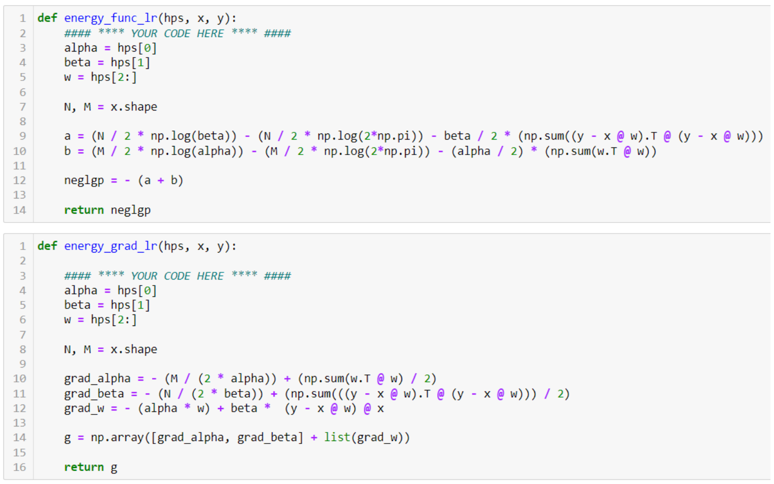


Figure 15: Alpha, beta, and bias convergence



```
Calc.          Numeric        Delta          Acc.
     290.633        290.633    6.091057e-09  11
    -178.667       -178.667   -1.131675e-07  10
    -2.28905        2.28905    4.578100e+00   1
    0.722124      -0.722124   -1.444248e+00   1
    0.393429       -0.39343   -7.868590e-01   1
  -0.0755569      0.0755569    1.511138e-01   1
    0.422401      -0.422401   -8.448016e-01   1
   -0.719567       0.719567    1.439134e+00   1
   0.0125016     -0.0125016   -2.500319e-02   1
   -0.276659       0.276659    5.533183e-01   1
  -0.0203778      0.0203778    4.075561e-02   1
|----------|  0% accepted [ 18 secs to go ]
|#---------| 90% accepted [ 17 secs to go ]
|##--------| 90% accepted [ 15 secs to go ]
|###-------| 89% accepted [ 13 secs to go ]
|####------| 90% accepted [ 11 secs to go ]
|#####-----| 90% accepted [ 10 secs to go ]
|######----| 89% accepted [ 8 secs to go ]
|#######---| 89% accepted [ 6 secs to go ]
|########--| 89% accepted [ 4 secs to go ]
|#########-| 89% accepted [ 2 secs to go ]
|##########| 89% accepted [ 0 secs to go ]
HMC: R=20000 / L=25 / eps=0.0003 / Accept=89.0%
```

Figure 16: Validating the gradient function for linear regression model in HMC



```python
def energy_func_lr(hps, x, y):
    #### **** YOUR CODE HERE **** ####
    alpha = hps[0]
    beta = hps[1]
    w = hps[2:]

    N, M = x.shape

    a = (N / 2 * np.log(beta)) - (N / 2 * np.log(2*np.pi)) - beta / 2 * (np.sum((y - x @ w).T @ (y - x @ w)))
    b = (M / 2 * np.log(alpha)) - (M / 2 * np.log(2*np.pi)) - (alpha / 2) * (np.sum(w.T @ w))

    neglgp = - (a + b)

    return neglgp
```

```python
def energy_grad_lr(hps, x, y):

    #### **** YOUR CODE HERE **** ####
    alpha = hps[0]
    beta = hps[1]
    w = hps[2:]

    N, M = x.shape

    grad_alpha = - (M / (2 * alpha)) + (np.sum(w.T @ w) / 2)
    grad_beta = - (N / (2 * beta)) + (np.sum(((y - x @ w).T @ (y - x @ w))) / 2)
    grad_w = - (alpha * w) + beta *  (y - x @ w) @ x

    g = np.array([grad_alpha, grad_beta] + list(grad_w))

    return g
```

Figure 17: Energy function and the energy gradient for linear regression

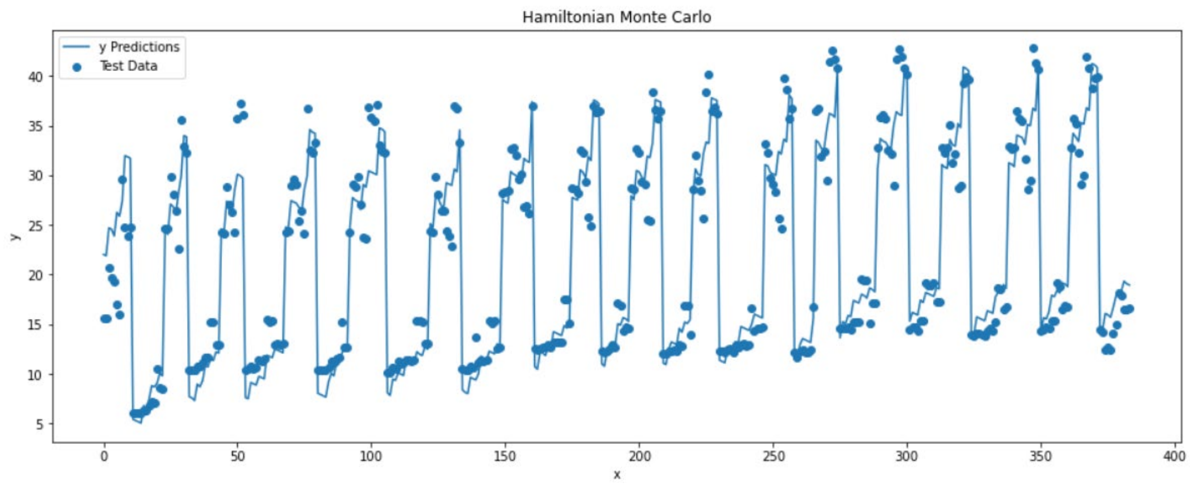


Figure 18: Baseline prediction plot for HMC

5 Apply GP to the Linear Regression Model

All the regressors in this task are all taken from sci-kit learn. The first Gaussian Process Regressor model was using the default kernel with no hyperparameter setting. In the second regressor model, RBF was added to the model and the result was unsatisfactory, as the MAE and RMSE both jumps up in value significantly. The length scale was set as 0.5 and the length scale bound was set as (0.001, 100) in the RBF kernel. In the third GP, the white kernel is added to the kernel, with a noise level of 0.0001 and a noise level bound of (1e-22, 100). This fixed the model by adding noise to it, where the MAE was reduced to 0.6719 and RMSE were reduced to 0.5022. The last model uses the Rational Quadratic kernel with a length scale of 0.5 and the same length scale bound as the previous model. It scores 0.8883 in RMSE and 0.6126 in MAE.

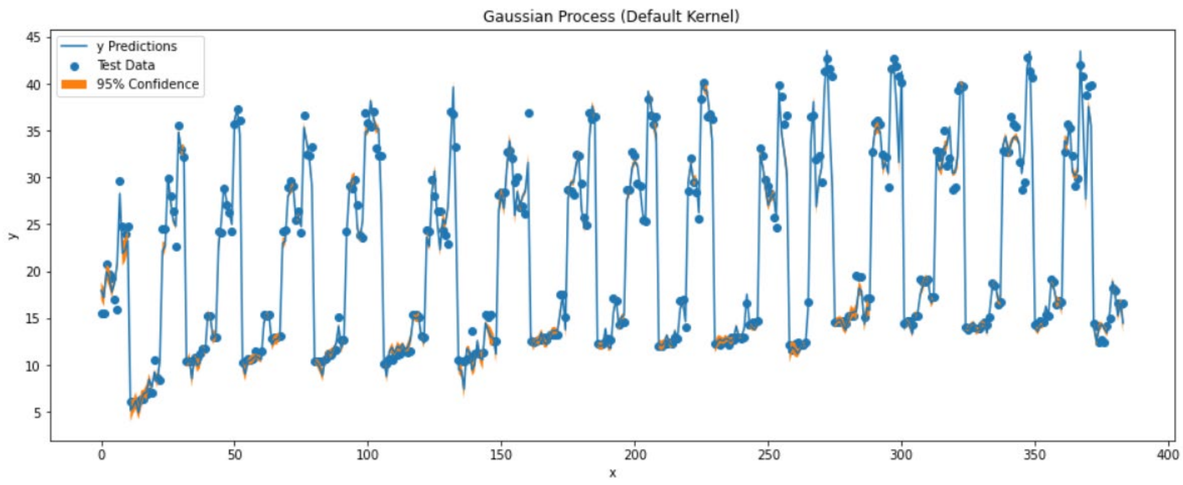
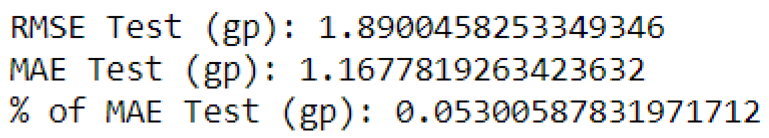


Figure 19: Gaussian Process using the default kernel



```
RMSE Test (gp): 1.8900458253349346
MAE Test (gp): 1.1677819263423632
% of MAE Test (gp): 0.05300587831971712
```

Figure 20: MAE and RMSE of the first GP

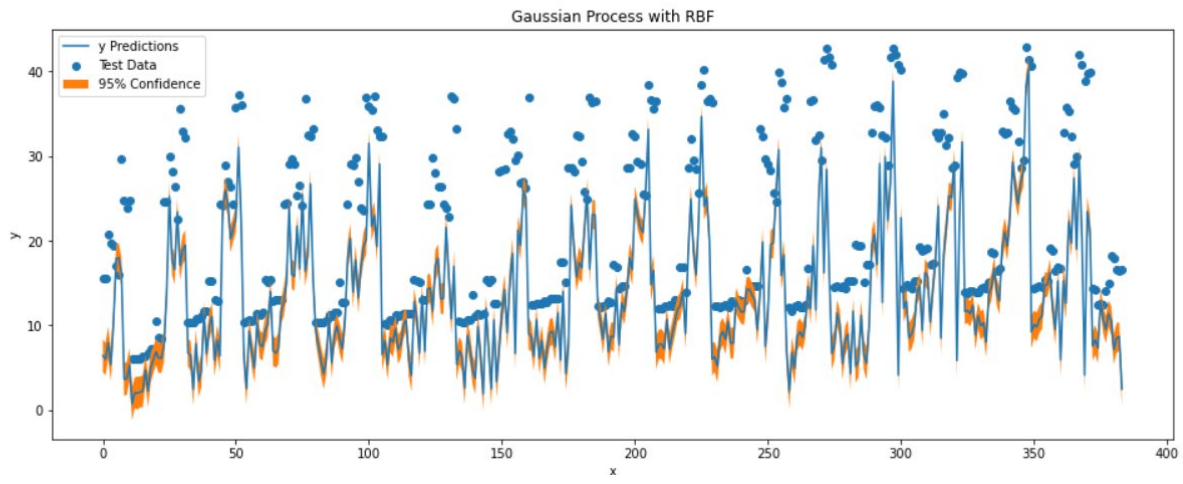


Figure 21: Gaussian Process using RBF kernel

```
RMSE Test (gp w/ rbf): 10.924232516393257
MAE Test (gp w/ rbf): 8.312543062134372
% of MAE Test (gp w/ rbf): 0.36799433312361973
```

Figure 22: MAE and RMSE of the second GP

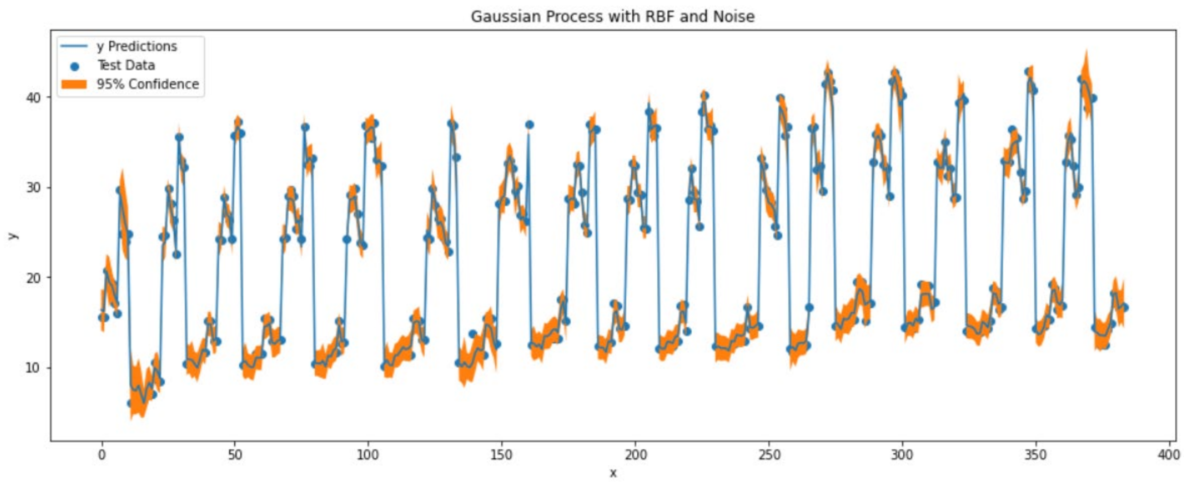


Figure 23: Gaussian Process using RBF kernel with noise

```
RMSE Test (gp w/ rbf and noise): 0.6718558557884639
MAE Test (gp w/ rbf and noise): 0.502244661688745
% of MAE Test (gp w/ rbf and noise): 0.029289792258009012
```

Figure 24: MAE and RMSE of the third GP

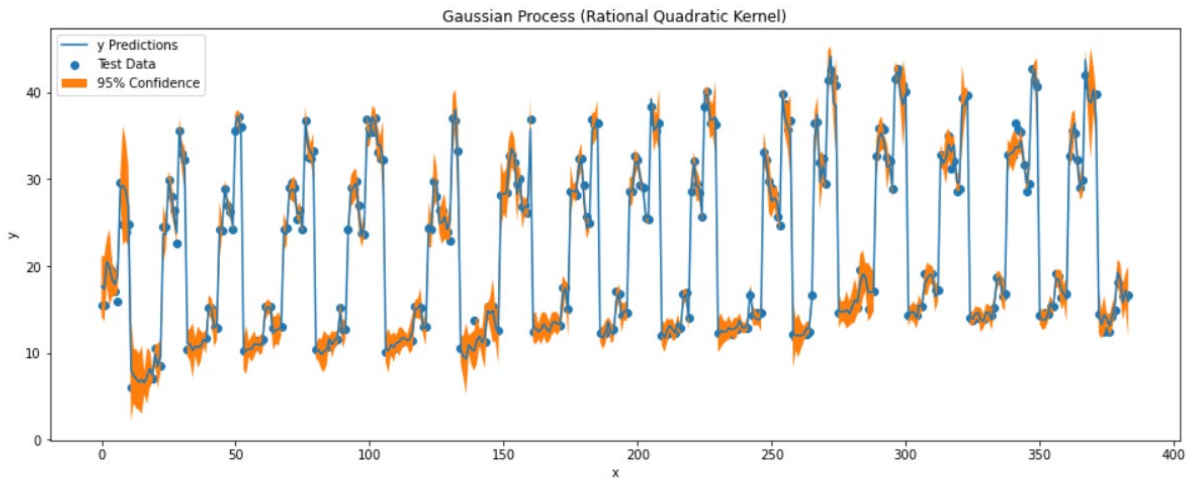


Figure 25: Gaussian Process using rational quadratic

```
RMSE Test (gp w/ RQ): 0.888316704999718
MAE Test (gp w/ RQ): 0.612581308396763
% of MAE Test (gp w/ RQ): 0.03127067613544764
```

Figure 26: MAE and RMSE of the last GP