FINAL PROJECT REPORT

CM50268 – BAYESIAN MACHINE LEARNING

SKYLAR YAU

Contribution:

- Code: 40%
- Theoretical/mathematical derivation: 40%
- Useful thoughts/opinions leading to group solutions: 40%

Table of Contents

1. Introduction

In this coursework, a regression problem of heating load estimation on the UCI “Energy efficiency” dataset will be solved with several Bayesian approaches.

2. Task 1: Exploratory Data Analysis (EDA)

a. Descriptive Statistics

Before modelling, EDA is performed to understand the characteristics of the dataset. There are 384 rows of data in both the training and testing set. The descriptive statistics of the training and testing sets are listed in Table 2-1 and Table 2-2. The mean and standard deviations of variables in both datasets appear to be very similar.

Table 2-1: EDA – Statistics of the training data

	const	Relative Compactness	Surface Area	Wall Area	Roof Area	Overall Height	Orientation	Glazing Area	Glazing Area Distribution	Heating Load
count	384.0	384.000000	384.000000	384.000000	384.000000	384.000000	384.000000	384.000000	384.000000	384.000000
mean	1.0	0.771042	665.774740	318.180990	173.796875	5.377604	3.536458	0.236849	2.783854	22.920703
std	0.0	0.106553	88.196712	42.248972	44.852410	1.747619	1.097695	0.133306	1.567506	10.066099
min	1.0	0.620000	514.500000	245.000000	110.250000	3.500000	2.000000	0.000000	0.000000	6.400000
25%	1.0	0.690000	588.000000	294.000000	140.875000	3.500000	3.000000	0.100000	1.000000	14.057500
50%	1.0	0.760000	661.500000	318.500000	147.000000	7.000000	4.000000	0.250000	3.000000	23.605000
75%	1.0	0.860000	735.000000	343.000000	220.500000	7.000000	5.000000	0.400000	4.000000	32.052500
max	1.0	0.980000	808.500000	416.500000	220.500000	7.000000	5.000000	0.400000	5.000000	43.100000

Table 2-2: EDA – Statistics of the training data

	const	Relative Compactness	Surface Area	Wall Area	Roof Area	Overall Height	Orientation	Glazing Area	Glazing Area Distribution	Heating Load
count	384.00	384.00	384.00	384.00	384.00	384.00	384.00	384.00	384.00	384.00
mean	1.00	0.76	677.64	318.82	179.41	5.12	3.46	0.23	2.84	21.69
std	0.00	0.10	87.69	45.01	45.36	1.75	1.14	0.13	1.54	10.09
min	1.00	0.62	514.50	245.00	110.25	3.50	2.00	0.00	0.00	6.01
25%	1.00	0.66	612.50	294.00	140.88	3.50	2.00	0.10	2.00	12.86
50%	1.00	0.74	686.00	318.50	220.50	3.50	3.00	0.25	3.00	17.16
75%	1.00	0.82	759.50	343.00	220.50	7.00	4.00	0.40	4.00	29.88
max	1.00	0.98	808.50	416.50	220.50	7.00	5.00	0.40	5.00	42.77

b. Correlation Between Variables

From Figure 2-1 and Figure 2-2, it can be observed that multicollinearity exists as some features have high correlation with one another, for instance, “Relative Compactness” and “Overall Height”, which might impact the interpretation of the final regression coefficients. However, interpretation of the results is not the focus of this coursework, the problem is neglected.

It can also be observed that “Orientation”, “Glazing Area” and “Glazing Area Distribution” have little correlation with the target variable “Heating Load”. Hence it is expected that the coefficients for these variables will be much lower than the other features.

Figure 2-1: EDA - Correlation matrix

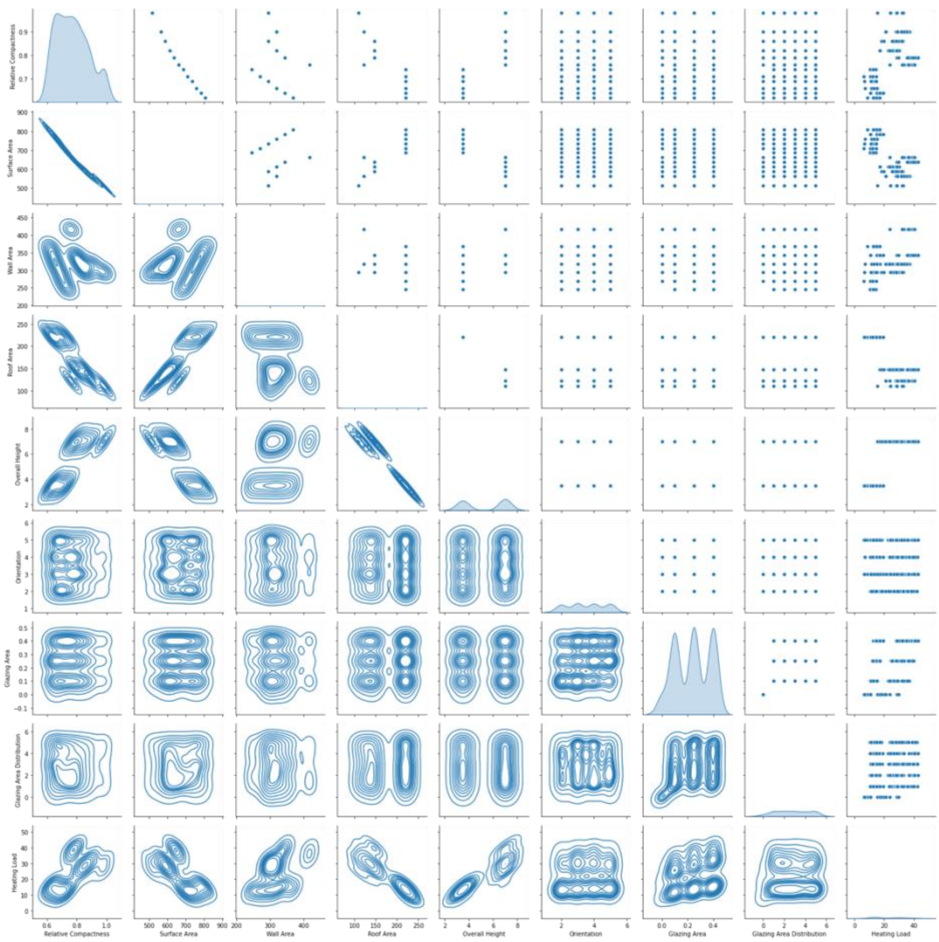
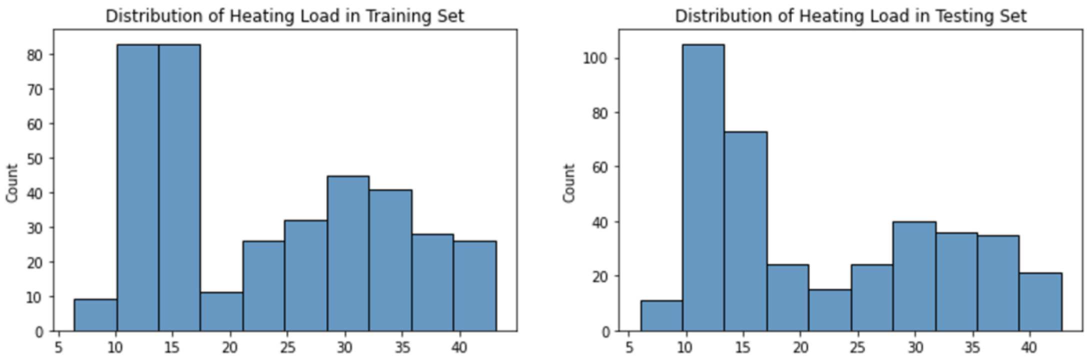


Figure 2-2: EDA - Correlation pair plot

c. Distribution of the Target Variable

The distributions of "Heating Load" in both training and testing sets are visualised in Figure 2-3, which shows a bimodal distribution that could be difficult to solve with simple linear regression.



(a) Training set (b) Testing set

Figure 2-3: EDA – Distribution of target variable

d. Standardisation of features

As the features have different ranges of values, standarisation is performed to convert all features to zero mean and unit variance before modelling.

```
transformer = StandardScaler()

# normalise X except constant bias
norm_X_train = transformer.fit_transform(pre_X_train.iloc[:, 1:])
norm_X_test = transformer.fit_transform(pre_X_test.iloc[:, 1:])
```

Figure 2-4: EDA – Code for standardising features

Table 2-3: EDA – Statistics of the standardised training data

	Relative Compactness	Surface Area	Wall Area	Roof Area	Overall Height	Orientation	Glazing Area	Glazing Area Distribution
count	384.00	384.00	384.00	384.00	384.00	384.00	384.00	384.00
mean	-0.13	0.13	0.02	0.13	-0.15	-0.07	-0.04	0.04
std	0.98	1.00	1.07	1.01	1.00	1.04	1.00	0.98
min	-1.42	-1.72	-1.73	-1.42	-1.08	-1.40	-1.78	-1.78
25%	-1.04	-0.60	-0.57	-0.73	-1.08	-1.40	-1.03	-0.50
50%	-0.29	0.23	0.01	1.04	-1.08	-0.49	0.10	0.14
75%	0.46	1.06	0.59	1.04	0.93	0.42	1.23	0.78
max	1.96	1.62	2.33	1.04	0.93	1.34	1.23	1.42

Table 2-4: EDA – Statistics of the standardised testing data

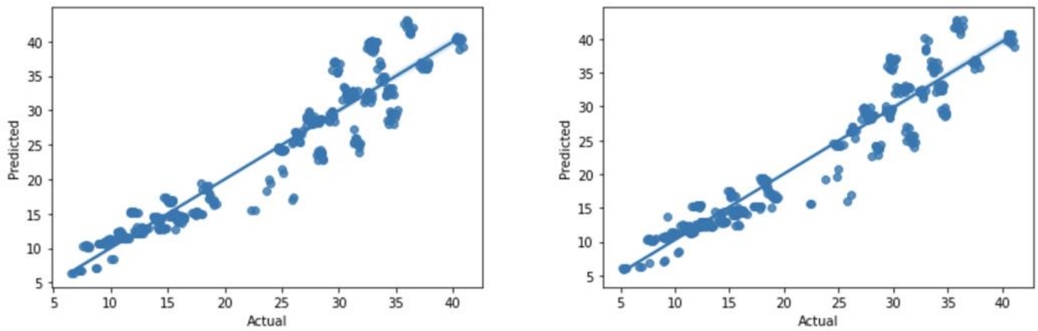
	Relative Compactness	Surface Area	Wall Area	Roof Area	Overall Height	Orientation	Glazing Area	Glazing Area Distribution
count	384.00	384.00	384.00	384.00	384.00	384.00	384.00	384.00
mean	0.00	-0.00	-0.00	-0.00	0.00	0.00	0.00	-0.00
std	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
min	-1.31	-1.86	-1.64	-1.53	-0.93	-1.29	-1.74	-1.85
25%	-0.93	-0.74	-0.55	-0.85	-0.93	-1.29	-0.99	-0.55
50%	-0.17	0.10	-0.01	0.91	-0.93	-0.41	0.14	0.10
75%	0.60	0.93	0.54	0.91	1.08	0.47	1.26	0.76
max	2.13	1.49	2.17	0.91	1.08	1.35	1.26	1.41

e. Baseline: Ordinary Least-Squares (OLS)

An OLS model is used as the benchmark. Figure 2-6 shows that there are generally two groups of predictions (heating load <= 20 and heating load > 20). This can also be evidenced by Figure 2-7, where two small peaks are observed at the two ends of the residuals plot.

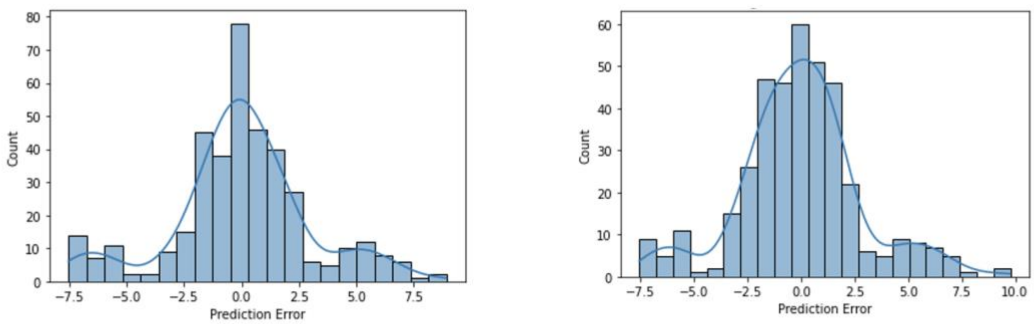
```
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
lr_y_test_pred = lr_model.predict(X_test)
lr_y_train_pred = lr_model.predict(X_train)
```

Figure 2-5: OLS – Code for the model



(a) Training set (b) Testing set

Figure 2-6: OLS – Actual vs Predicted



(a) Training set (b) Testing set

Figure 2-7: OLS - Residuals distribution

Table 2-5: OLS – Predicted weights of features

Feature	Coefficient weights
	OLS
Constant	22.92
Relative Compactness	-7.23
Surface Area	-3.94
Wall Area	0.76
Roof Area	-4.23
Overall Height	7.20
Orientation	-0.13
Glazing Area	2.77
Glazing Area Distribution	0.20

Table 2-6: OLS – Errors of model prediction on training and test sets

Measure	OLS	
	Train	Test
RMSE	3.01	2.84
MSE	9.07	8.09
MAE	2.13	2.07

3. Task 2: Bayesian Linear Regression (BLR)

In the following, some Bayesian approaches will be explored to derive the probability estimates, instead of point estimates in the frequentist OLS approach.

The BLR setup for the problem is defined as follows:

$$\begin{aligned} P\left(\vec{w} \middle| \vec{Y}, \alpha, \beta\right) &= \frac{P\left(\vec{Y} \middle| \vec{w}, \alpha, \beta\right) P(\vec{w} \mid \alpha) P(\alpha) P(\beta)}{P(\vec{Y})} \\ &\propto P\left(\vec{Y} \middle| \vec{w}, \alpha, \beta\right) P(\vec{w} \mid \alpha) P(\alpha) P(\beta) && (1) \\ &= N(\vec{m}_N, \vec{S}_N) && (2) \end{aligned}$$

where

- $P\left(\vec{Y} \middle| \vec{w}, \alpha, \beta\right) = N(\vec{w}^T \vec{X}, \beta^{-1} I)$
- $P(\vec{w} \mid \alpha) = N(0, \alpha^{-1} I)$
- $\vec{m}_N = \vec{S}_N \, (\vec{S}_0{}^{-1} \vec{m}_0 + \beta \vec{X}^T \vec{y})$
- $\vec{S}_N{}^{-1} = \vec{S}_0{}^{-1} + \beta \vec{X}^T \vec{X}$

In this problem, it is assumed that $\vec{w}$, α and β are all latent variables to be estimated.

a. Type-II Maximum Likelihood (ML)

To compute the posterior over $\vec{w}$, information about the unknown hyperparameters α and β are required. Type-II ML achieves such purpose by maximising the hyperparameter posterior defined below:

$$P(\alpha, \beta \mid \vec{Y}) = \frac{P(\vec{Y} \mid \alpha, \beta)\, P(\alpha)\, P(\beta)}{P(\vec{Y})} \tag{3}$$

$$\propto P(\vec{Y} \mid \alpha, \beta) \tag{4}$$

As flat priori are assumed for the hyperparameters, maximising the posterior becomes maximising the likelihood $P(\vec{Y} \mid \alpha, \beta)$, which is equation (4).

The expanded form of equation (3) can be written as equation (4) below:

$$P(\vec{Y} \mid \alpha, \beta) = \int P(\vec{Y} \mid \vec{w}, \alpha, \beta)\, P(\vec{w} \mid \alpha)\, d\vec{w} \tag{5}$$

This results in a zero-mean Gaussian distribution over $\vec{Y}$ with tractable covariance matrix:

$$P(\vec{Y} \mid \alpha, \beta) = N(0,\, \beta^{-1} I \,+\, \alpha^{-1} \vec{X} \vec{X}^{T}) \tag{6}$$

```python
def compute_log_marginal(X, y, alph, beta):
    #### **** YOUR CODE HERE **** ####

    N, M = X.shape
    s2 = 1/beta
    C = s2 * np.eye(N) + (X @ X.T) / alph

    lgp = stats.multivariate_normal.logpdf(y.T, mean=None, cov=C, allow_singular=True)

    return lgp
```

Figure 3-1: Type-II ML – Code for computing the log marginal likelihood

The maximum hyperparameter posterior is found by searching a grid of 100 values within the range of e^{-5} and e^{0} for α and β values. The results of the search are listed in Table 3-1. The posterior distribution is visualised in Figure 3-2.

Table 3-1: Type-II ML – Most probable hyperparameter values

Hyperparameter	Most probable value
	Type-II ML
ln (α)	-4.44
ln (β)	-2.22
α	0.01
β	0.11
Log probability	-1001.46

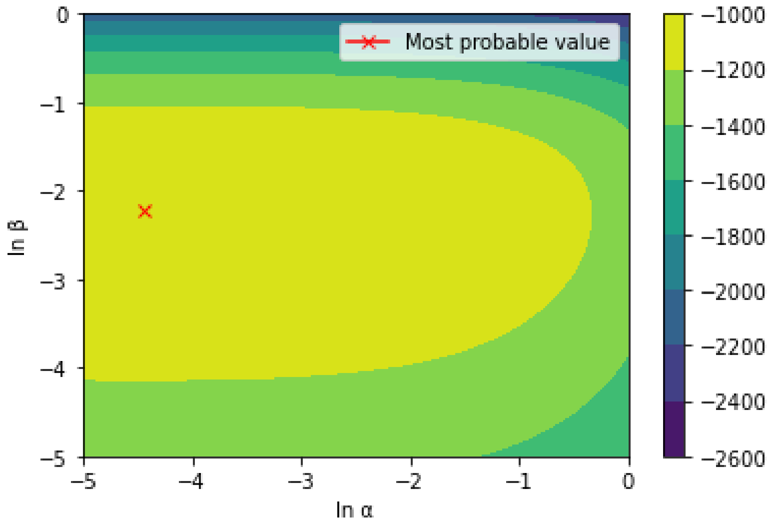


Figure 3-2: Type-II ML – Visualisation of the hyperparameter posterior distribution

```python
def compute_posterior(X, y, alph, beta):
    #### **** YOUR CODE HERE **** ####

    M = X.shape[1]
    H = beta*(X.T @ X) + alph*np.eye(M)
    SIGMA = np.linalg.inv(H)
    Mu = beta * (SIGMA @ (X.T @ y))

    return Mu, SIGMA
```

Figure 3-3: Type-II ML – Code for computing the posterior

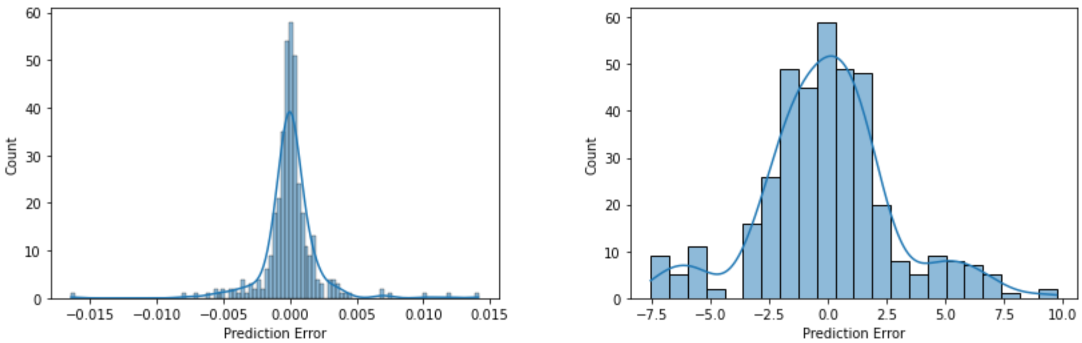
Table 3-2: Type-II ML – Predicted weights of features

Feature	Coefficient weights	
	OLS	Type-II ML
Constant	22.92	22.91
Relative Compactness	-7.23	-6.93
Surface Area	-3.94	-3.74
Wall Area	0.76	0.80
Roof Area	-4.23	-4.05
Overall Height	7.20	7.29
Orientation	-0.13	-0.13
Glazing Area	2.77	2.77
Glazing Area Distribution	0.20	0.20

The error rates of the prediction computed the posterior mean over $\vec{w}$ are listed in Table 3-3, which are very close to the OLS solution.

Table 3-3: Type-II ML – Errors of model prediction on training and test sets

Measure	OLS		Type-II ML	
	Train	Test	Train	Test
RMSE	3.01	2.84	3.01	2.84
MSE	9.07	8.09	9.07	8.09
MAE	2.13	2.07	2.13	2.07



(a) Training: Residuals distribution (b) Testing: Residuals distribution

Figure 3-4: Type-II ML - Residuals distribution

b. Variational Inference (VI)

VI is another deterministic approach for approximating the latent variables. It tries to seek a good approximation of a proposal distribution $Q(\vec{w}, \alpha, \beta)$ to the posterior $P^*(\vec{w}, \alpha, \beta\ |\vec{Y})$. The measurement used is KL divergence (Equation 7), and the aim of VI is to minimise this divergence, which is equivalent to maximising ELBO in equation 8.

$$KL(Q\ ||\ P) = -E_Q\left[\frac{P(\vec{w}, \alpha, \beta)}{Q(\vec{w}, \alpha, \beta, |\phi)}\right] + \ln P(\vec{Y}) \quad (7)$$

$$KL(Q\ ||\ P) = -ELBO + Constant \quad (8)$$

where

- ϕ is the set of hyperparameters that controls the distribution of Q

To simplify the variational calculation, conjugate Gamma hyperpriori are chosen for α and β as follows, with uninformative parameters:

- $P(\alpha) = Gamma(a_0, b_0)$, where $a_0 = b_0 = 10^{-4}$
- $P(\beta) = Gamma(c_0, d_0)$, where $C_0 = d_0 = 10^{-4}$

Instead of computing the complex joint distributions, mean-field theory is applied to assume that Q can be factorized across parameters, as shown in equation 9.

$$Q(\vec{w}, \alpha, \beta) = Q_{\vec{w}}(\vec{w})\, Q_\alpha(\alpha)\, Q_\beta(\beta) \quad (9)$$

The individual components of $Q(\vec{w}, \alpha, \beta)$ are elaborated below:

1. Posterior over $\vec{w}$

$$\boldsymbol{Q_{\vec{w}}(\vec{w})} = \exp\{\langle \ln(P(\vec{Y}, \vec{w}, \alpha, \beta))\rangle_{Q_\alpha(\alpha)\, Q_\beta(\beta)}\}$$

$$\begin{aligned}
\ln(Q_{\vec{w}}(\vec{w})) &= E_\beta[\ln P(\vec{Y} \mid \vec{w}, \beta)] + E_\alpha[\ln P(\vec{w} \mid \alpha)] + constant \\
&= \frac{N}{2}\ln\langle\beta\rangle - \frac{\langle\beta\rangle}{2}\sum_{n=1}^{N}(y_n - \vec{w}^T x_n)^2 + \frac{M}{2}\ln\langle\alpha\rangle - \frac{\langle\alpha\rangle}{2}\vec{w}^T\vec{w} + constant \\
&= -\frac{\langle\beta\rangle}{2}\sum_{n=1}^{N}(y_n - \vec{w}^T x_n)^2 - \frac{\langle\alpha\rangle}{2}\vec{w}^T\vec{w} + constant \\
&= -\frac{1}{2}\vec{w}^T\left(\langle\alpha\rangle I + \langle\beta\rangle\vec{X}^T\vec{X}\right)\vec{w} + \langle\beta\rangle\vec{w}^T\vec{X}^T\vec{Y} + constant \\
&= N(\vec{m}_N, \vec{S}_N)
\end{aligned}$$

where

- $\vec{m}_N = \langle\beta\rangle\overrightarrow{S_N}\vec{X}^T\vec{Y}$
- $\vec{S}_N = (\langle\alpha\rangle I + \langle\beta\rangle\vec{X}^T\vec{X})^{-1}$

2. Posterior over α

$$\boldsymbol{Q_\alpha(\alpha)} = \exp\{\langle \ln(P(\vec{Y}, \vec{w}, \alpha, \beta))\rangle_{Q_{\vec{w}}(\vec{w})\, Q_\beta(\beta)}\}$$

$$\begin{aligned}
\ln\left(Q_\alpha(\alpha)\right) &= \ln\left(P(\alpha)\right) + E_{\vec{w}}[\ln P(\vec{w} \mid \alpha)] + constant \\
&= (a_0 - 1)\ln(\alpha) - b_0\alpha + \frac{M}{2}\ln(\alpha) - \frac{\alpha}{2}\langle\vec{w}^T\vec{w}\rangle + constant \\
&= \left(\frac{M}{2} + a_0 - 1\right)\ln(\alpha) - (b_0 + \frac{1}{2}\langle\vec{w}^T\vec{w}\rangle)\alpha + constant \\
&= Gamma\,(a_N, b_N)
\end{aligned}$$

where

- $a_N = \frac{M}{2} + a_0$
- $b_N = b_0 + \frac{1}{2}\langle\vec{w}^T\vec{w}\rangle$

3. Posterior over β

$$\boldsymbol{Q_\beta(\beta)} = \exp\{\langle \ln(P(\vec{Y}, \vec{w}, \alpha, \beta))\rangle_{Q_{\vec{w}}(\vec{w})\, Q_\alpha(\alpha)}\}$$

$$\begin{aligned}
\ln\left(Q_\beta(\beta)\right) &= \ln\left(P(\beta)\right) + E_{\vec{w}}[\ln P(\vec{Y} \mid \vec{w}, \beta)] + constant \\
&= (c_0 - 1)\ln(\beta) - d_0\beta + \frac{N}{2}\ln\beta - \frac{\beta}{2}\sum_{n=1}^{N}(y_n - {x_n}^T\langle\vec{w}\rangle)^2 + constant \\
&= \left(\frac{N}{2} + c_0 - 1\right)\ln(\beta) - (d_0 + \frac{1}{2}\sum_{n=1}^{N}(y_n - {x_n}^T\langle\vec{w}\rangle)^2)\beta + constant \\
&= Gamma\,(c_N, d_N)
\end{aligned}$$

where

- $c_N = \frac{N}{2} + c_0$
- $d_N = d_0 + \frac{1}{2}\sum_{n=1}^{N}(y_n - {x_n}^T\langle\vec{w}\rangle)^2$

The expectations of the latent variables are computed using equations below:

$$E[\alpha] = \frac{a_N}{b_N}$$

$$E[\beta] = \frac{c_N}{d_N}$$

$$E[\vec{w}] = \vec{m}_N$$

$$E[\vec{w}^T\vec{w}] = \vec{m}_N{}^T\vec{m}_N + Tr(\vec{S}_N)$$

```
def update_w_params(e_alph, e_beta, X, y):
    N, M = X.shape
    XtX = np.dot(X.T, X)
    sig_n = np.linalg.inv(e_alph * np.eye(M) + e_beta * XtX)
    mu_n = e_beta * (sig_n @ (X.T @ y))
    return mu_n, sig_n

def update_alph_params(M, a0, b0, e_WtW):
    an = M/2 + a0
    bn = b0 + 0.5 * e_WtW
    return an, bn

def update_beta_params(c0, d0, e_mu_n, X, y):
    N, M = X.shape
    cn = N/2 + c0
    dn = d0 + 0.5 * np.sum((y-(X @ e_mu_n))**2)
    return cn, dn
```

```
old_mu_n = mu_n
old_sig_n = sig_n
mu_n, sig_n =  update_w_params(e_alph, e_beta, X, y)

e_WtW = np.dot(mu_n.T, mu_n) + np.trace(sig_n)

old_e_alph = e_alph
an, bn = update_alph_params(M, a0, b0, e_WtW)
e_alph = an/bn

old_e_beta = e_beta
cn, dn = update_beta_params(c0, d0, mu_n, X, y)
e_beta = cn/dn
```

Figure 3-5: VI – Code for updating the parameters and expectations of $\vec{w}$, α and β

During VI, the above are updated iteratively until there is no change in the expectations of the latent variables. The convergence can be evidenced by the cumulative average plot of ELBO (Figure 3-6). From Figure 3-6 and Figure 3-7, it is shown that locating the maximum posterior with VI took only 8 updates, which is much more efficient than Type-II ML.

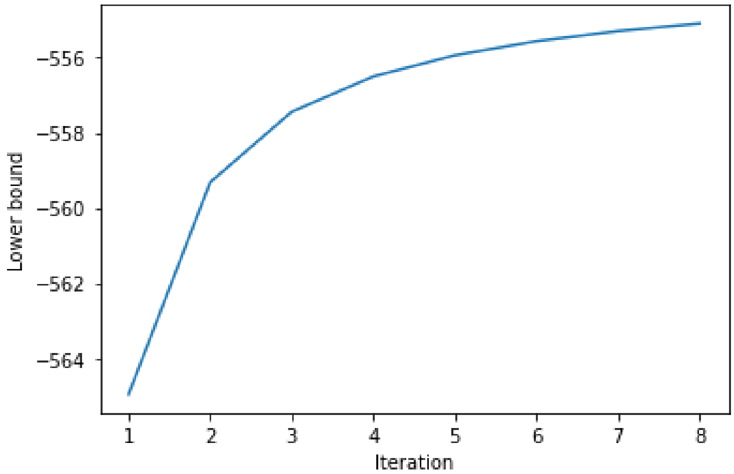


Figure 3-6: VI – Cumulative average of ELBO

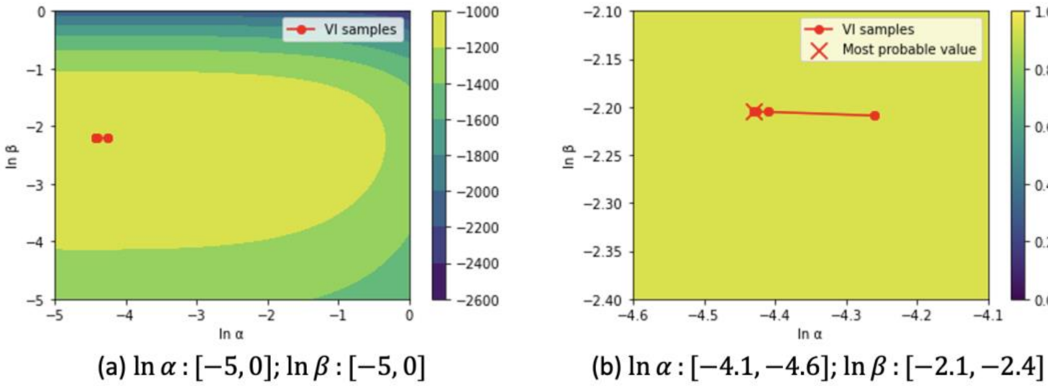


Figure 3-7: VI – Contour plots illustrating sampled hyperparameter values and the corresponding posterior values

Table 3-4: VI – Most probable hyperparameter values

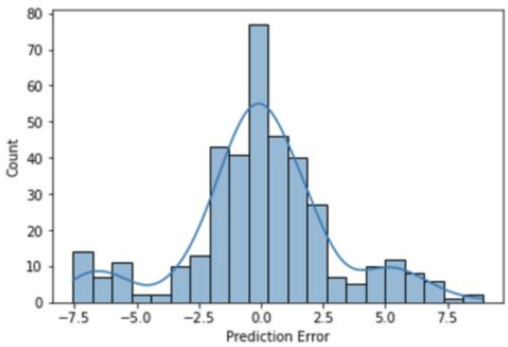
Hyperparameter	Most probable value	
	Type-II ML	VI
$\ln(\alpha)$	-4.44	-4.43
$\ln(\beta)$	-2.22	-2.21
α	0.01	0.01
β	0.11	0.11

Table 3-5: VI – Predicted weights of features

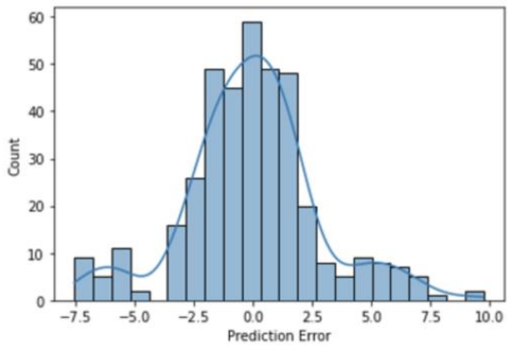
Feature	Coefficient weights		
	OLS	Type-II ML	VI
Constant	22.92	22.91	22.91
Relative Compactness	-7.23	-6.93	-6.93
Surface Area	-3.94	-3.74	-3.74
Wall Area	0.76	0.80	0.80
Roof Area	-4.23	-4.05	-4.05
Overall Height	7.20	7.29	7.29
Orientation	-0.13	-0.13	-0.13
Glazing Area	2.77	2.77	2.77
Glazing Area Distribution	0.20	0.20	0.20

Table 3-6: VI – Errors of model prediction on training and test sets

Measure	OLS		Type-II ML		VI	
	Train	Test	Train	Test	Train	Test
RMSE	3.01	2.84	3.01	2.84	3.01	2.84
MSE	9.07	8.09	9.07	8.09	9.07	8.09
MAE	2.13	2.07	2.13	2.07	2.13	2.07



(a) Training: Residuals distribution



(b) Testing: Residuals distribution

Figure 3-8: VI - Residuals distribution

4. Task 3: Verify Hamiltonian Monte Carlo (HMC) on a Standard 2D Gaussian

In Bayesian analytics, many integrals are intractable, hence stochastic methods like HMC are developed to approximate the distributions with random sampling. HMC leverages the Hamiltonian dynamics concepts of energy and gradient to direct the sampling, so that the process is more efficient than other algorithms such as Gibbs and Metropolis-Hasting sampling.

In this section, HMC is applied to approximate a target distribution of a standard 2D Gaussian with zero mean and covariance matrix of $\begin{pmatrix} 1.5 & 1 \\ 1 & 2 \end{pmatrix}$. The distribution is visualised in Figure 4-1.

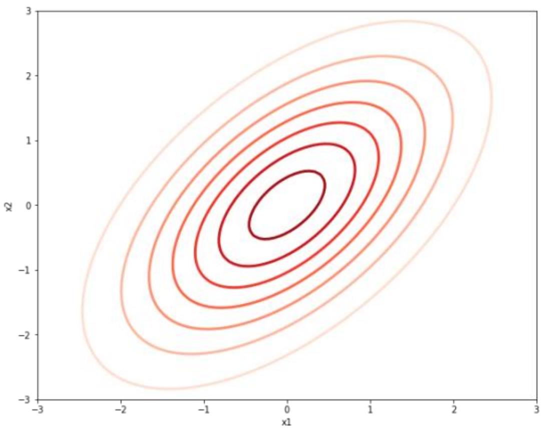


Figure 4-1: HMC – Target 2D Gaussian distribution

a. Energy function

The energy function in this problem is defined as the negative log pdf of a zero-mean 2D Gaussian (Equation 10).

$$-\log(P^*(x_1, x_2)) = \frac{1}{2}\vec{X}^T\vec{V}^{-1}\vec{X} \tag{10}$$

where

- $\vec{V}$ is the covariance matrix of the distribution

```
def energy_func(x, covar):
    #### **** YOUR CODE HERE **** ####

    neglgp = -1*stats.multivariate_normal.logpdf(x=x, mean=None, cov=covar, allow_singular=True)

    return neglgp
```

Figure 4-2: HMC – Code for computing the energy function of the standard 2D Gaussian

b. Gradient function

The gradient of the energy function with respect to $\vec{X}$ is:

$$\frac{1}{2}\vec{X}^T(\vec{V}^{-1})^T \tag{11}$$

```python
def energy_grad(x, covar):
    #### **** YOUR CODE HERE **** ####

    V_inv = np.linalg.inv(covar)
    g = 0.5 * np.dot(x.T, V_inv + V_inv.T)

    return g
```

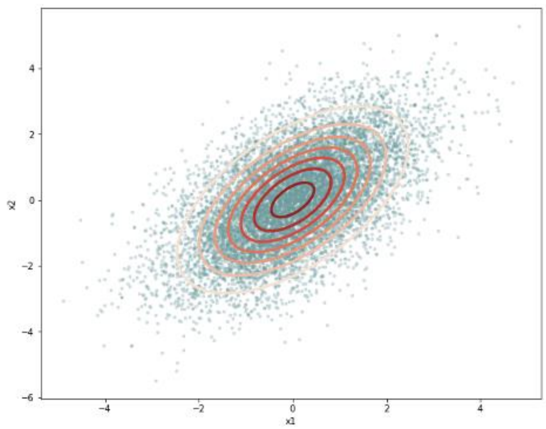
Figure 4-3: HMC – Code for computing the energy function gradient of the standard 2D Gaussian

Calc.	Numeric	Delta	Acc.
1.78452	1.78452	-1.342071e-10	11
-0.938964	-0.938964	2.158310e-10	10

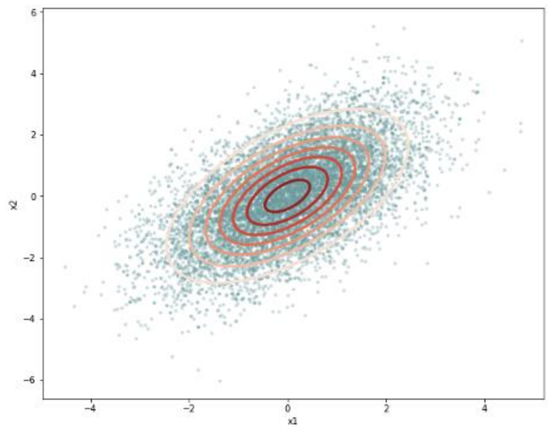
Figure 4-4: HMC – Validation on gradient equation for the standard 2D Gaussian example

c. Sampling

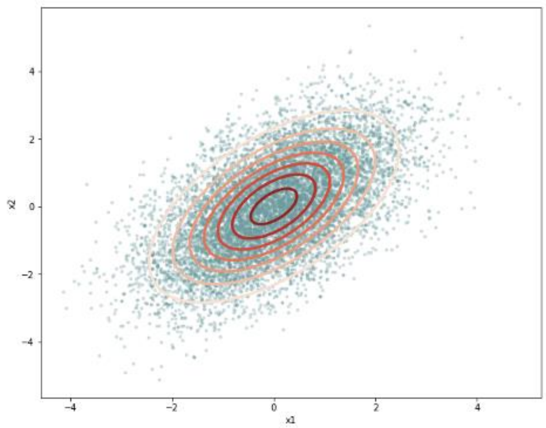
The HMC sampler is run to approximate the target distribution. The initial state is a covariance matrix with values randomly sampled from a normal distribution of zero mean and unit variance. In every iteration, 10,000 samples are obtained and 100 steps are taken within each cycle. The value of eps, which indicates the step size, is tuned by trial-and-error. An eps too large will lead to low acceptance rate, while an eps too small will lead to inefficient sampling. The final eps value (1.2) is chosen based on the general guidance of choosing an acceptance rate closest to 80%.



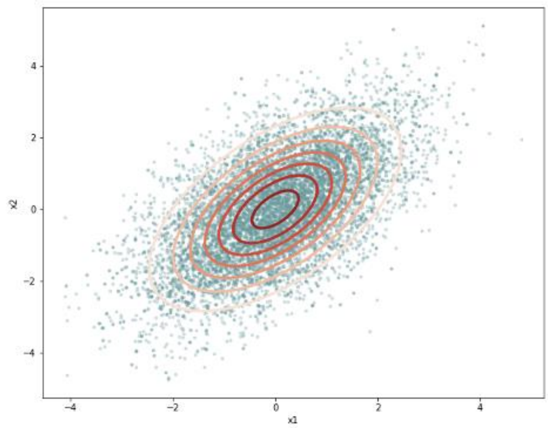
(a) Eps = 0.8;
Acceptance rate = 94.7%



(b) Eps = 1.0;
Acceptance rate = 90.5%



(c) Eps = 1.2;
Acceptance rate = 84.5%

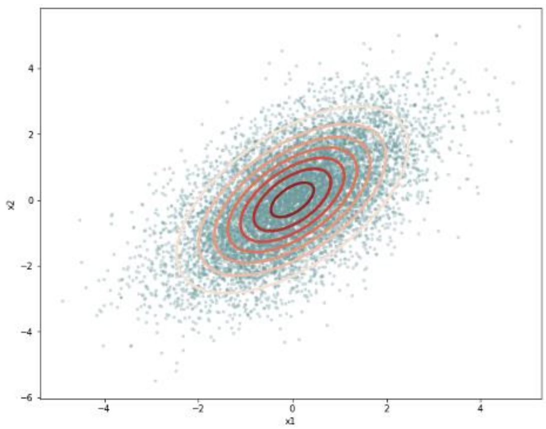


(d) Eps = 1.4;
Acceptance rate = 72.7%

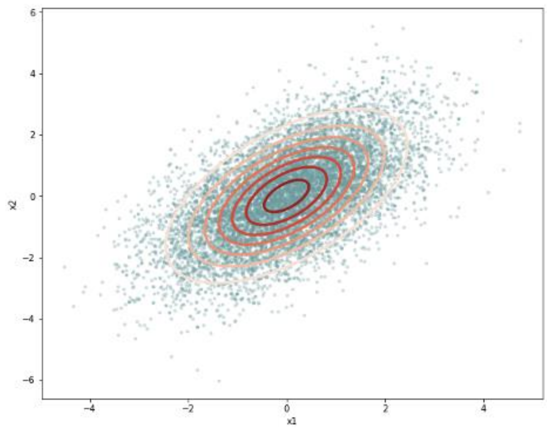
Figure 4-6 shows the results of the sampled distribution and acceptance rate with different values of eps.

```
|----------|  0% accepted [ 21 secs to go ]
|#---------| 84% accepted [ 19 secs to go ]
|##--------| 84% accepted [ 18 secs to go ]
|###-------| 84% accepted [ 17 secs to go ]
|####------| 84% accepted [ 16 secs to go ]
|#####-----| 84% accepted [ 15 secs to go ]
|######----| 84% accepted [ 12 secs to go ]
|#######---| 84% accepted [ 9 secs to go ]
|########--| 84% accepted [ 6 secs to go ]
|#########-| 84% accepted [ 3 secs to go ]
|##########| 84% accepted [ 0 secs to go ]
HMC: R=10000 / L=100 / eps=1.2 / Accept=84.5%
```

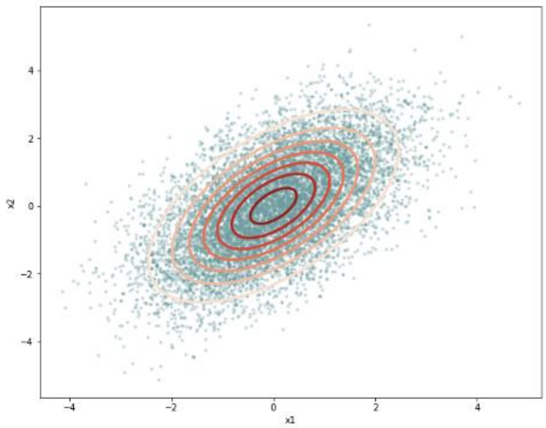
Figure 4-5: HMC – Acceptance rate with eps = 1.2



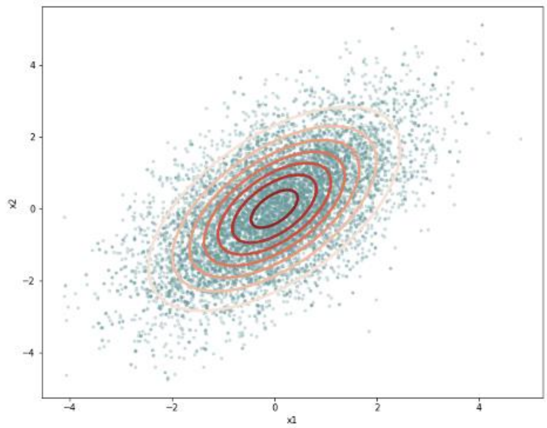
(a) Eps = 0.8;
Acceptance rate = 94.7%



(b) Eps = 1.0;
Acceptance rate = 90.5%



(c) Eps = 1.2;
Acceptance rate = 84.5%



(d) Eps = 1.4;
Acceptance rate = 72.7%

Figure 4-6: HMC – Sampling results with different values of eps

Table 4-1 lists the final approximated covariances, which is very close to that of the target distribution's.

Table 4-1: HMC – Approximations for the standard 2D Gaussian sample

Latent variable	Value
x_1	1.43
x_2	1.95

5. Task 4: Apply HMC on Linear Regression Model

The HMC sampler is to be applied on the BLR problem in this section.

a. Energy function

The energy function for the BLR problem is the negative log of the joint posterior $P(\vec{w}, \alpha, \beta|\vec{y})$. Uninformative priors are assumed for α and β. The following list the steps for deriving the function.

Step 1: Derive the joint posterior $P(\vec{w}, \alpha, \beta|\vec{y})$

$$\propto P(\vec{y}|\vec{w}, \alpha, \beta)\ P(\vec{w}|\alpha) P(\alpha) P(\beta)$$

$$= \prod_{n=1}^{N} N(y_n|{x_n}^T\vec{w}, \beta^{-1}) \prod_{m=1}^{M} N(w_m|0, \alpha^{-1})$$

$$= \prod_{n=1}^{N} (\frac{\beta}{2\pi})^{\frac{1}{2}}\, e^{-\frac{1}{2}(\frac{y_n - {x_n}^T\vec{w}}{\sqrt{\beta^{-1}}})^2} \prod_{m=1}^{M} (\frac{\alpha}{2\pi})^{\frac{1}{2}}\, e^{-\frac{1}{2}(\frac{\vec{w}}{\sqrt{\alpha^{-1}}})^2}$$

$$= (\frac{\beta}{2\pi})^{\frac{N}{2}} \prod_{n=1}^{N} e^{-\frac{\beta}{2}(y_n - {x_n}^T\vec{w})^2}\ (\frac{\alpha}{2\pi})^{\frac{M}{2}} \prod_{m=1}^{M} e^{-\frac{\alpha}{2}(\vec{w})^2}$$

Step 2: Take log of the joint posterior $\ln\left(P(\vec{w}, \alpha, \beta|\vec{y})\right)$

$$\propto \frac{N}{2}\ln\left(\frac{\beta}{2\pi}\right) - \frac{\beta}{2}\sum_{n=1}^{N}(y_n - {x_n}^T\vec{w})^2 + \frac{M}{2}\ln\left(\frac{\alpha}{2\pi}\right) - \frac{\alpha}{2}\vec{w}^T\vec{w}$$

$$= \frac{N}{2}\ln(\beta) - \frac{\beta}{2}\sum_{n=1}^{N}(y_n - {x_n}^T\vec{w})^2 + \frac{M}{2}\ln(\alpha) - \frac{\alpha}{2}\vec{w}^T\vec{w} + constant$$

Step 3: Derive the negative log joint posterior $-ln(P(\vec{w}, \alpha, \beta|\vec{y}))$

$$\propto -\frac{N}{2}\ln(\beta) + \frac{\beta}{2}\sum_{n=1}^{N}(y_n - {x_n}^T\vec{w})^2 - \frac{M}{2}\ln(\alpha) + \frac{\alpha}{2}\vec{w}^T\vec{w} + constant$$

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

    alpha = hps[0]
    beta = hps[1]

    w = hps[2:]
    N, M = x.shape

    WtX = np.dot(x, w)
    WtW = np.dot(w.T, w)

    lgp = (N/2) * np.log(beta / 2*math.pi) - (beta/2)  * np.sum((y-WtX)**2) + \
          (M/2) * np.log(alpha / 2*math.pi) - (alpha/2) * WtW

    return -lgp
```

Figure 5-1: HMC – Code for computing the energy function of the BLR problem

b. Gradient function

The gradient of the function comprises of the gradients of the energy function with respect to α, β and $\vec{w}$ respectively.

<u>**Gradient w.r.t. α**</u>

$$= \frac{d}{d\alpha}\left\{-\frac{M}{2}\ln(\alpha) + \frac{\alpha}{2}\vec{w}^T\vec{w}\right\}$$
$$= -\frac{M}{2\alpha} + \frac{1}{2}\vec{w}^T\vec{w}$$

<u>**Gradient w.r.t. β**</u>

$$= \frac{d}{d\beta}\left\{-\frac{N}{2}\ln(\beta) + \frac{\beta}{2}\sum_{n=1}^{N}(y_n - {x_n}^T\vec{w})^2\right\}$$
$$= -\frac{N}{2\beta} + \frac{1}{2}\left(\vec{Y} - \vec{X}^T\vec{w}\right)^T\left(\vec{Y} - \vec{X}^T\vec{w}\right)$$

<u>**Gradient w.r.t. $\vec{w}$**</u>

$$= \frac{d}{d\vec{w}}\left\{\frac{\beta}{2}\sum_{n=1}^{N}(y_n - {x_n}^T\vec{w})^2 + \frac{\alpha}{2}\vec{w}^T\vec{w}\right\}$$
$$= \frac{d}{d\vec{w}}\left\{\frac{\beta}{2}\left(\vec{Y} - \vec{X}^T\vec{w}\right)^T\left(\vec{Y} - \vec{X}^T\vec{w}\right) + \frac{\alpha}{2}\vec{w}^T\vec{w}\right\}$$
$$= -\beta\vec{X}^T\left(\vec{Y} - \vec{X}^T\vec{w}\right) + \alpha\vec{w}$$

<u>**Gradient of the energy function**</u>

= [Gradient w.r.t. α, Gradient w.r.t. β, Gradient w.r.t. $\vec{w}$]

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

    alpha = hps[0]
    beta = hps[1]

    w = hps[2:]
    N, M = x.shape

    WtW = np.dot(w.T, w)
    WtX = np.dot(x, w)
    XtY = np.dot(x.T, y)
    XtX = np.dot(x.T, x)
    XW = np.matmul(x, w)

    # 1. Gradient w.r.t. alpha
    grad_al = -(M/(2*alpha)) + 0.5*WtW

    # 2. Gradient w.r.t. beta
    grad_beta = -(N/(2*beta)) + 0.5 * np.sum((y-WtX)**2)

    # 3. Gradient w.r.t. w
    grad_w = -beta * XtY + beta * np.dot(XtX, w) + alpha * w

    # Combine the gradients
    grad = np.concatenate(([grad_al], [grad_beta], grad_w), axis=None)

    return grad
```

Figure 5-2: HMC – Code for computing the energy function gradient of the BLR problem

```
Calc.          Numeric        Delta          Acc.
    -665.886       -665.886   -4.866265e-06    9
     83471.6        83471.6   -2.092166e-04    9
    -57.7407       -57.7407    1.070885e-07    9
    -15.0687       -15.0687    2.924464e-08    9
     16.1026        16.1026   -4.811512e-08    9
     -9.5219        -9.5219    8.484069e-08    9
     20.3187        20.3187    7.083293e-08    9
    -21.0904       -21.0904    6.286347e-08    9
     1.91004        1.91004   -1.649822e-08    9
    -5.09471       -5.09471   -9.656807e-08    8
    0.508103       0.508103   -1.483655e-07    7
```

Figure 5-3: HMC – Validation on gradient equation for the standard 2D Gaussian example

c. Sampling

The initial state for the sampler is a size 11 arrays containing initial values of α, β and $\vec{w}$. The initial values of α and β are defined as e^{-5}, while the initial values of $\vec{w}$ are randomly sampled from uniform distribution of zero mean and unit variance.

As the BLR problem involves much more latent variables than the previous example, it is found that convergence cannot be reached using the previous parameters. Hence, the number of steps is increased to 400 instead. Once again, the optimal eps value (0.00012) is set by trial-and-error.

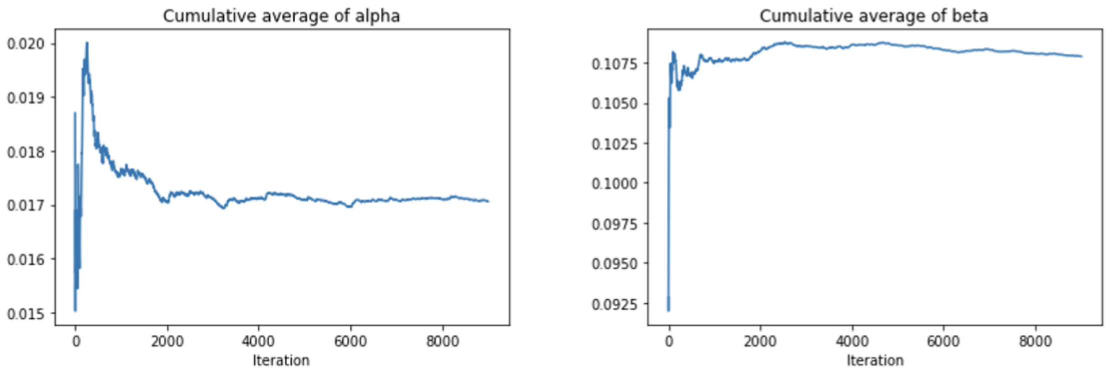
```
|----------|  0% accepted [ 273 secs to go ]
|#---------| 100% accepted [ 251 secs to go ]
|##--------| 100% accepted [ 230 secs to go ]
|###-------| 100% accepted [ 202 secs to go ]
|####------| 100% accepted [ 190 secs to go ]
|#####-----| 100% accepted [ 159 secs to go ]
|######----| 100% accepted [ 128 secs to go ]
|#######---| 100% accepted [ 95 secs to go ]
|########--| 100% accepted [ 63 secs to go ]
|#########-| 100% accepted [ 31 secs to go ]
|##########| 100% accepted [ 0 secs to go ]
HMC: R=10000 / L=400 / eps=0.00012 / Accept=100.0%
```

Figure 5-4: HMC – Acceptance rate with eps = 0.00012

Figure 5-5 and Figure 5-6 shows cumulative average of α, β and $\vec{w}$ of the drawn samples. α and β and some of the weights demonstrates convergence at values very close to that of the previous methods. Theoretically, given infinite time, HMC will be able to arrive at the exact solution, however, this is impractical in most cases. Despite having different values of $\vec{w}$, the final performance is very similar to the deterministic solutions'.

Table 5-1: HMC – Most probable hyperparameter values

Hyperparameter	Most probable value		
	Type-II ML	**VI**	**HMC**
$\ln(\alpha)$	-4.44	-4.43	-4.07
$\ln(\beta)$	-2.22	-2.21	-2.23
α	0.01	0.01	0.02
β	0.11	0.11	0.11



(a) α (b) β

Figure 5-5: HMC – Cumulative average of hyperparameters

Table 5-2: HMC – Predicted weights of features

Feature	**Coefficient weights**			
	OLS	**Type-II ML**	**VI**	**HMC**
Constant	22.92	22.91	22.91	22.91
Relative Compactness	-7.23	-6.93	-6.93	-4.48
Surface Area	-3.94	-3.74	-3.74	-4.14
Wall Area	0.76	0.80	0.80	2.20
Roof Area	-4.23	-4.05	-4.05	-0.24
Overall Height	7.20	7.29	7.29	8.25
Orientation	-0.13	-0.13	-0.13	-0.14
Glazing Area	2.77	2.77	2.77	2.79
Glazing Area Distribution	0.20	0.20	0.20	0.19

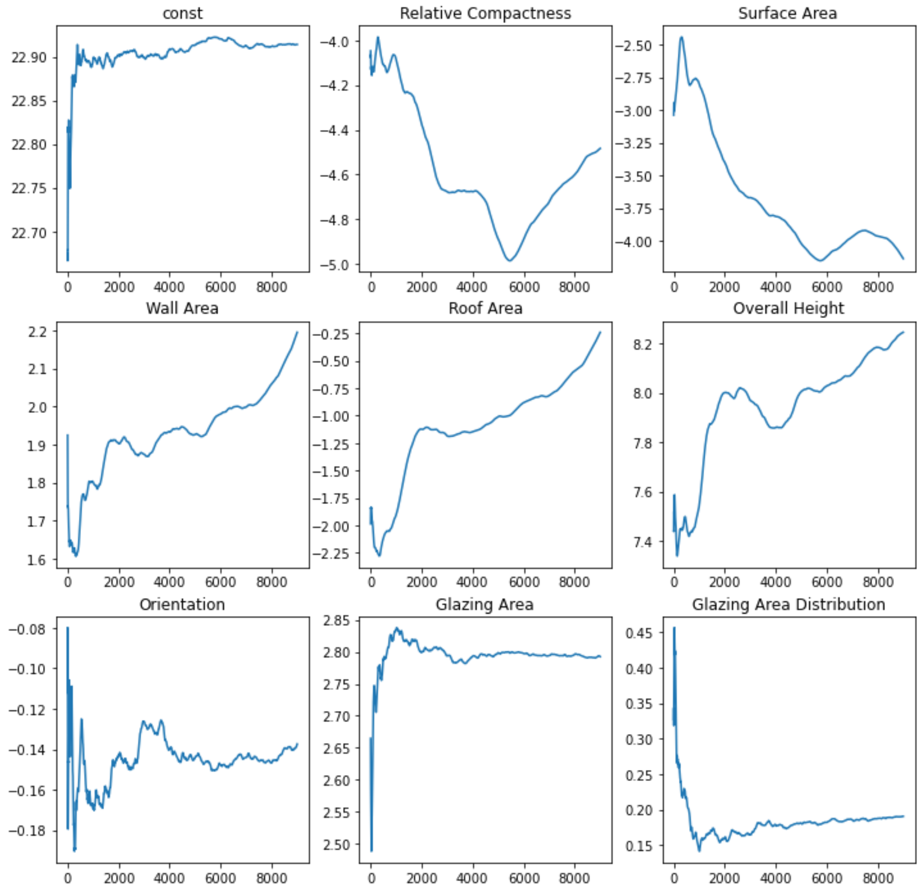
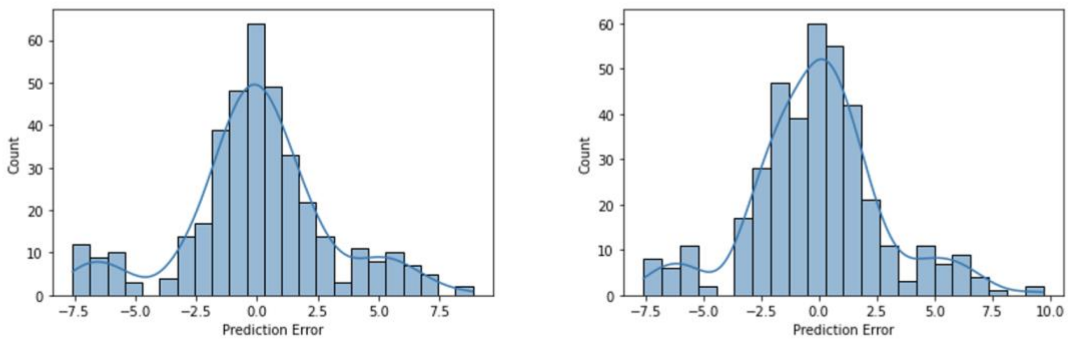


Figure 5-6: HMC – Cumulative average of hyperparameters $\overrightarrow{w}$

Table 5-3: HMC – Errors of model prediction on training and test sets

Measure	OLS		Type-II ML		VI		HMC	
	Train	Test	Train	Test	Train	Test	Train	Test
RMSE	3.01	2.84	3.01	2.84	3.01	2.84	3.02	2.86
MSE	9.07	8.09	9.07	8.09	9.07	8.09	9.14	8.16
MAE	2.13	2.07	2.13	2.07	2.13	2.07	2.15	2.08



(a) Training: Residuals distribution

(b) Testing: Residuals distribution

Figure 5-7: HMC - Residuals distribution

6. Task 5: Apply Gaussian Process (GP) to the Linear Regression Model

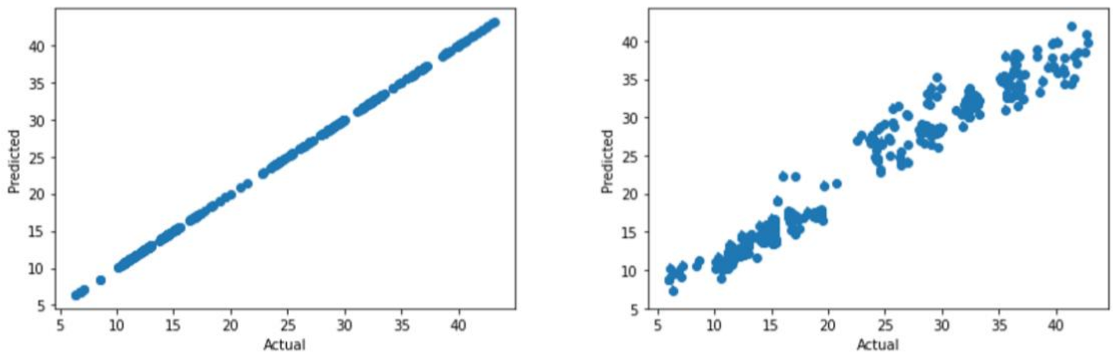
Unlike the previous methods, GP is a non-parametric method. The prior over $\vec{w}$ is assumed to follow a Gaussian distribution of mean zero, and a variance specified by a kernel. The hyperparameters of the kernel are optimised by maxmising the log marginal-likelihood, and the optimization is restarted repeatedly to avoid being stuck at local optima. GP provides a Gaussian-distributed prediction at each test point instead of point estimate, as shown in the "Actual vs Predicted" plots.

Four GP models with different combinations of kernels are implemented using sklearn's GaussianProcessRegressor. The combinations tested are listed in Table 6-1.

The most popular kernels used in GP, namely RBF and RationalQuadratic, are first being tested as the choice of kernel. To allow fair comparison with other approaches, the target values are not normalised. All GP models achieve significantly lower error rates than the previous parametric models. However, signs of overfitting are observed. From the residual distribution plots, it can be observed that the peaks at the two ends observed in previous methods are smoothed out in the first three kernels, which could indicate that the use of non-linear kernels in model help better fits the true distribution of data than linear functions used previously. Kernel #2 achieves the lowest error rate.

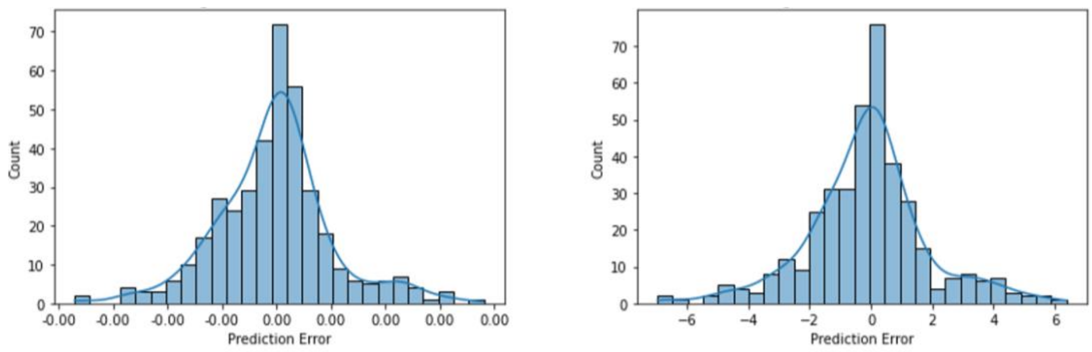
Table 6-1: GP - Combinations of kernels tested

#	Kernel Combinations	Optimised hyperparameters
1	RationalQuadratic	alpha=0.326, length_scale=0.573
2	RBF	length_scale=1.09
3	DotProduct + RationalQuadratic	sigma=23.1; alpha=8.77, length_scale=0.529)
4	DotProduct + RBF	sigma=22.9; length_scale=0.101



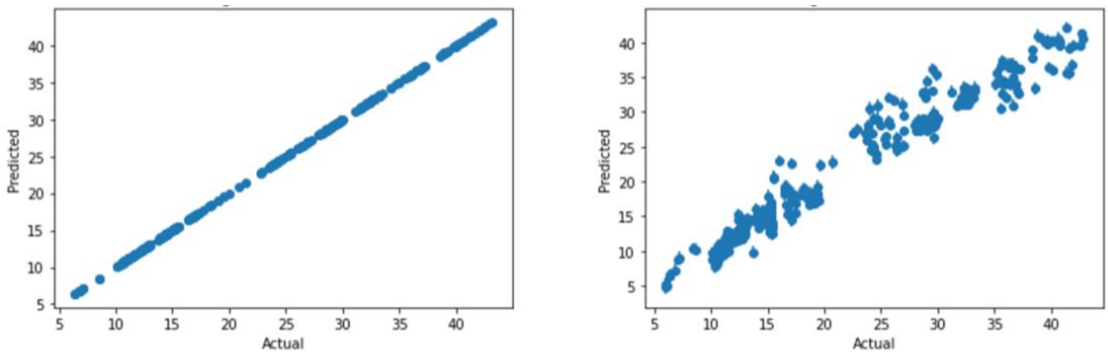
(a) Training: Actual vs Predicted (b) Testing: Actual vs Predicted

Figure 6-1: GP – (Kernel #1) Actual vs Predicted



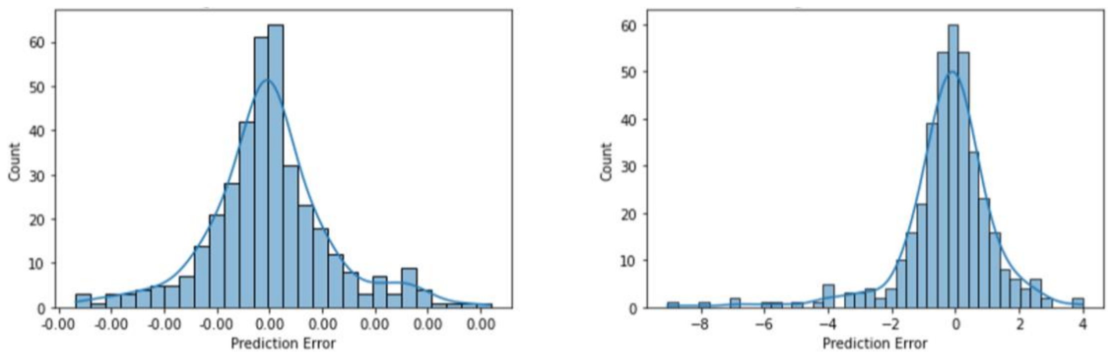
(a) Training: Residuals distribution (b) Testing: Residuals distribution

Figure 6-2: GP - (Kernel #1) Residuals distribution



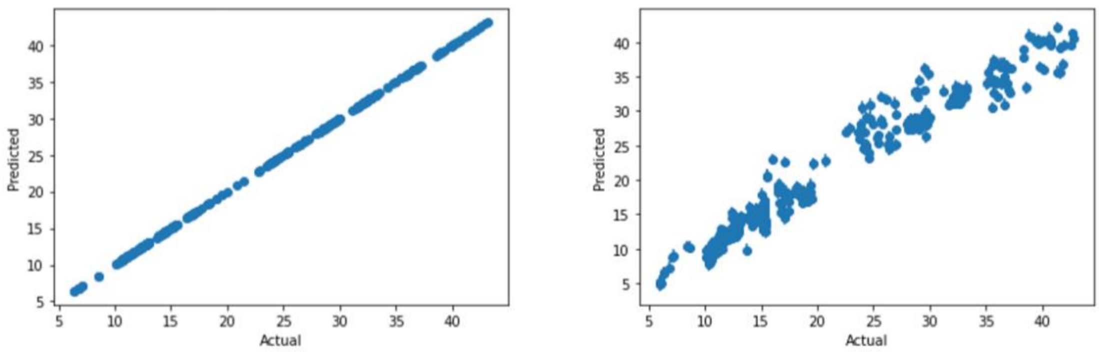
(a) Training: Actual vs Predicted (b) Testing: Actual vs Predicted

Figure 6-3: GP – (Kernel #2) Actual vs Predicted



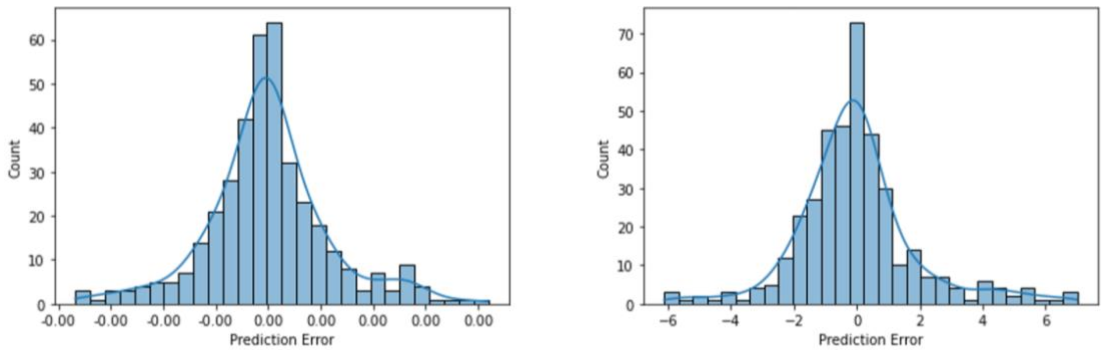
(a) Training: Residuals distribution (b) Testing: Residuals distribution

Figure 6-4: GP - (Kernel #2) Residuals distribution



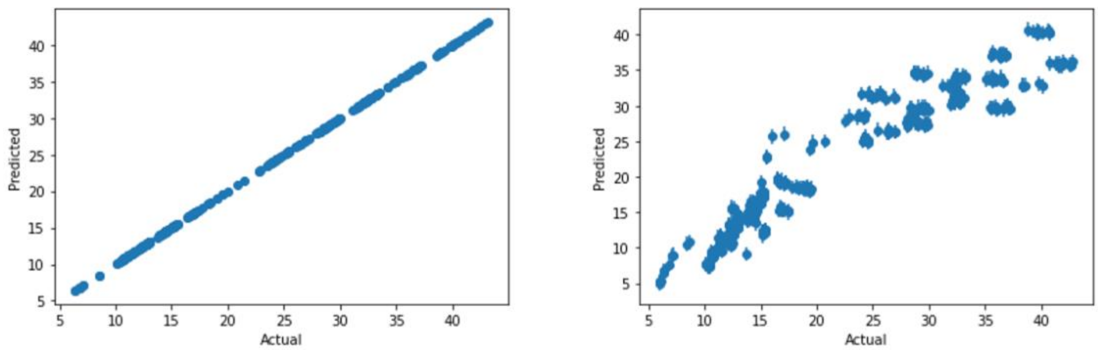
(a) Training: Actual vs Predicted (b) Testing: Actual vs Predicted

Figure 6-5: GP – (Kernel #3) Actual vs Predicted



(a) Training: Residuals distribution (b) Testing: Residuals distribution

Figure 6-6: GP - (Kernel #3) Residuals distribution



(a) Training: Actual vs Predicted (b) Testing: Actual vs Predicted

Figure 6-7: GP – (Kernel #4) Actual vs Predicted

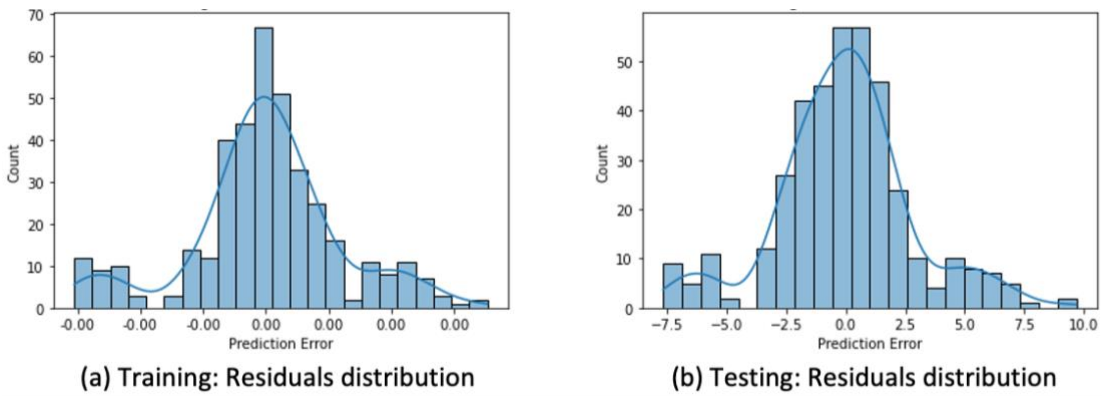


Figure 6-8: GP - (Kernel #4) Residuals distribution

Table 6-2: GP – Errors of model prediction on training and test sets

Measure	Kernel #1		Kernel #2		Kernel #3		Kernel #4	
	Train	Test	Train	Test	Train	Test	Train	Test
RMSE	$5.27e^{-5}$	1.95	0.0007	1.50	$2.91e^{-5}$	1.89	$3.02e^{-5}$	2.84
MSE	$2.78e^{-7}$	3.81	$5.48e^{-7}$	2.24	$8.45e^{-10}$	3.58	$9.10e^{-10}$	8.11
MAE	$3.75e^{-5}$	1.37	0.0004	0.95	2.04	1.28	$2.13e^{-5}$	2.06

7. Conclusion

Table 7-1 summarises the error rates of all methods on the BLR explored in this coursework. As this is a relatively simple problem, both Type-II ML and VI are able to arrive at the analytical solution. However, VI with mean-field theory found the solution much faster than Type-II ML. HMC solves the problem with stochastic random sampling approach, which results in an approximation that is close to the solution. Among all methods, Gaussian process achieves the best performance as non-linear kernels are incorporated into the model which better fit the characteristics of the dataset, however, it is more prone to overfitting.

Table 7-1: Summary of errors of all methods

Measure	OLS		Type-II ML		VI		HMC		GP	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
RMSE	3.01	2.84	3.01	2.84	3.01	2.84	3.02	2.86	0.00	1.50
MSE	9.07	8.09	9.07	8.09	9.07	8.09	9.14	8.16	$5.48e^{-7}$	2.24
MAE	2.13	2.07	2.13	2.07	2.13	2.07	2.15	2.08	0.00	0.95