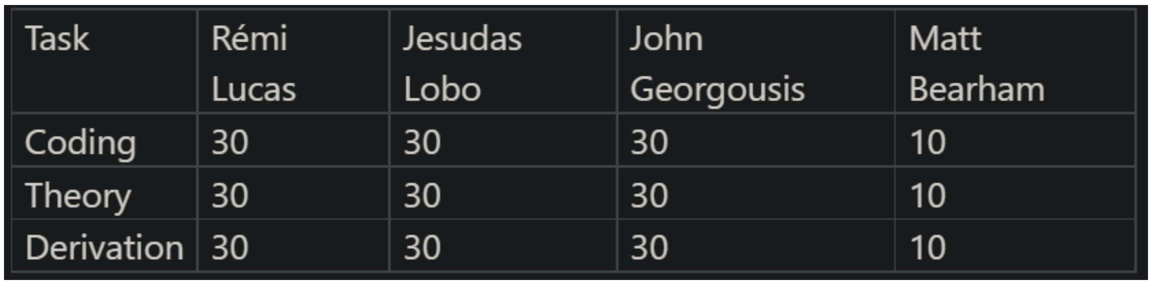
Final Project Report

Student: *John Georgousis*

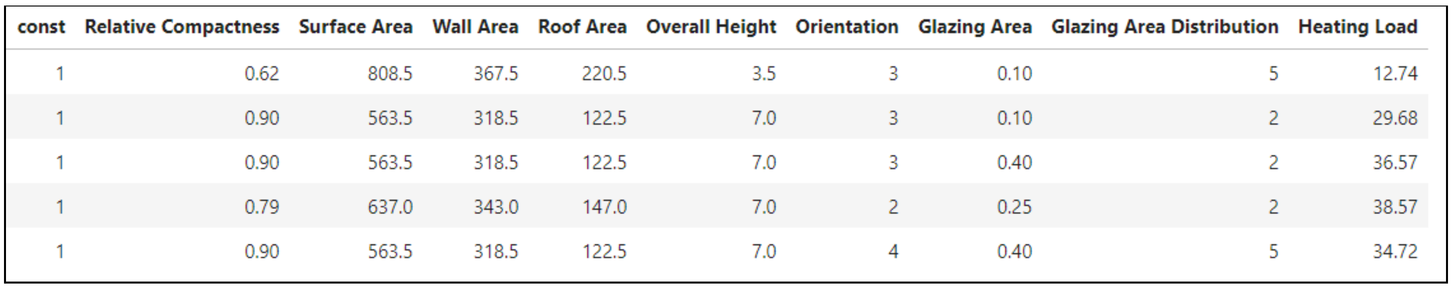
Contribution Form (Group 15)



Task	Rémi Lucas	Jesudas Lobo	John Georgousis	Matt Bearham
Coding	30	30	30	10
Theory	30	30	30	10
Derivation	30	30	30	10

Introduction

This project involves an analysis of the Energy Efficiency dataset, originally from the University of Oxford, using Bayesian modelling methods. The dataset consists of 768 examples with 9 input variables (including a constant bias) $x_0, x_1, x_2, \cdots, x_8$, representing architectural parameters for buildings which will be used to predict the target variable *y*, the “Heating Load”. The first 5 training examples are shown in Figure 1.



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

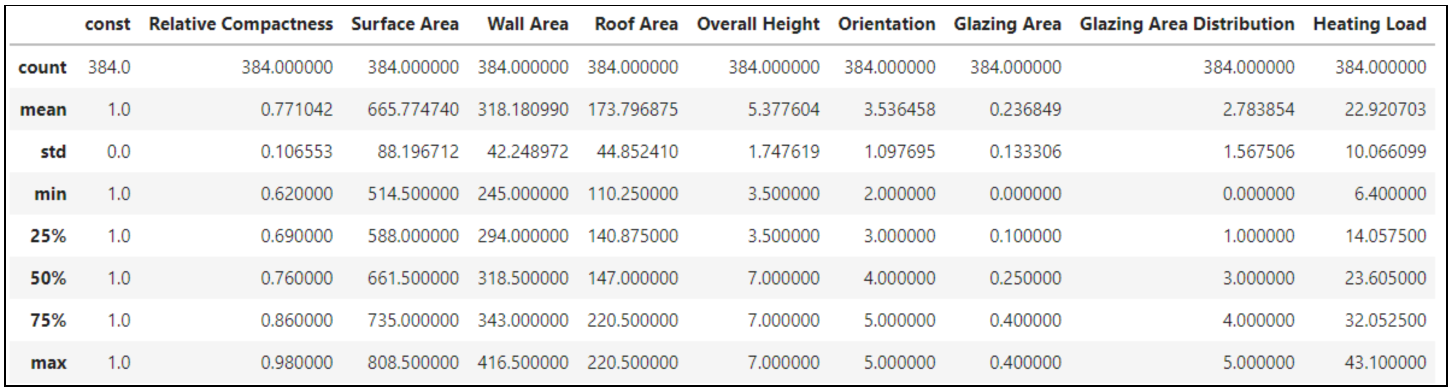
Figure 1. First 5 examples of the Energy Efficiency dataset with their values.

The data were later standardised to improve performance in modelling.

Task 1: Exploratory Data Analysis

Descriptive Statistics

Initially, basic descriptive statistics for each variable were displayed (shown in Figure 2) after making sure there were no missing values.

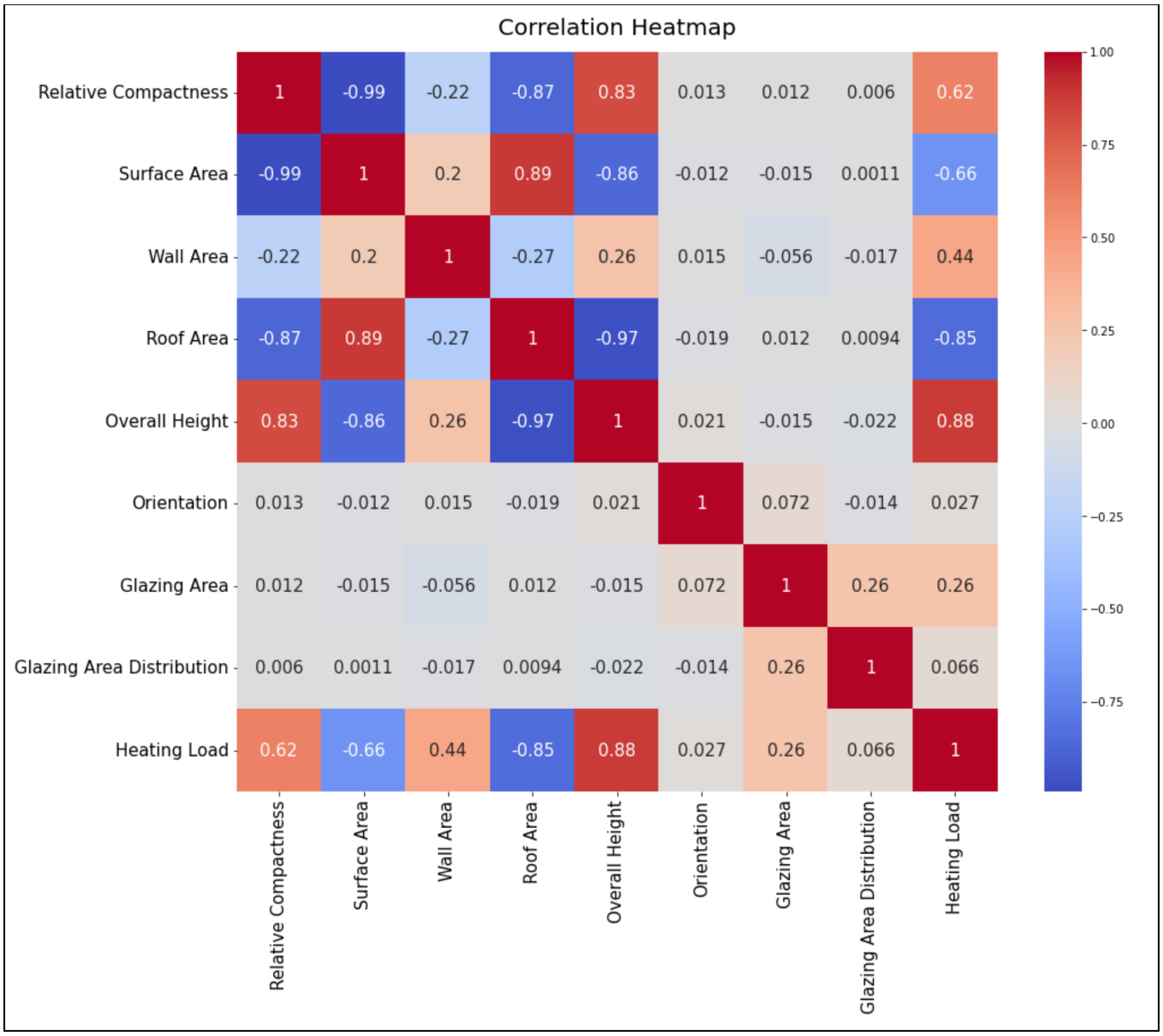


	const	Relative Compactness	Surface Area	Wall Area	Roof Area	Overall Height	Orientation	Glazing Area	Glazing Area Distribution	Heating Load
count	384.0	384.000000	384.000000	384.000000	384.000000	384.000000	384.000000	384.000000	384.000000	384.000000
mean	1.0	0.771042	665.774740	318.180990	173.796875	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

Figure 2: Descriptive statistics for each variable.

Correlations

Then, an analysis was performed to identify correlations between variables. Strong correlations make for good predictor variables for the heating load. The full heatmap of correlations is displayed in Figure 3 were -1: strong negative correlation, 0: no correlation, 1: strong positive correlation.



	Relative Compactness	Surface Area	Wall Area	Roof Area	Overall Height	Orientation	Glazing Area	Glazing Area Distribution	Heating Load
Relative Compactness	1	-0.99	-0.22	-0.87	0.83	0.013	0.012	0.006	0.62
Surface Area	-0.99	1	0.2	0.89	-0.86	-0.012	-0.015	0.0011	-0.66
Wall Area	-0.22	0.2	1	-0.27	0.26	0.015	-0.056	-0.017	0.44
Roof Area	-0.87	0.89	-0.27	1	-0.97	-0.019	0.012	0.0094	-0.85
Overall Height	0.83	-0.86	0.26	-0.97	1	0.021	-0.015	-0.022	0.88
Orientation	0.013	-0.012	0.015	-0.019	0.021	1	0.072	-0.014	0.027
Glazing Area	0.012	-0.015	-0.056	0.012	-0.015	0.072	1	0.26	0.26
Glazing Area Distribution	0.006	0.0011	-0.017	0.0094	-0.022	-0.014	0.26	1	0.066
Heating Load	0.62	-0.66	0.44	-0.85	0.88	0.027	0.26	0.066	1

Figure 3. Correlations between all variables in our dataset.

What we are most interested in however is the variables that are most useful for predicting heating load, our target variable. A table with heating load correlations was isolated and is shown in Figure 4.

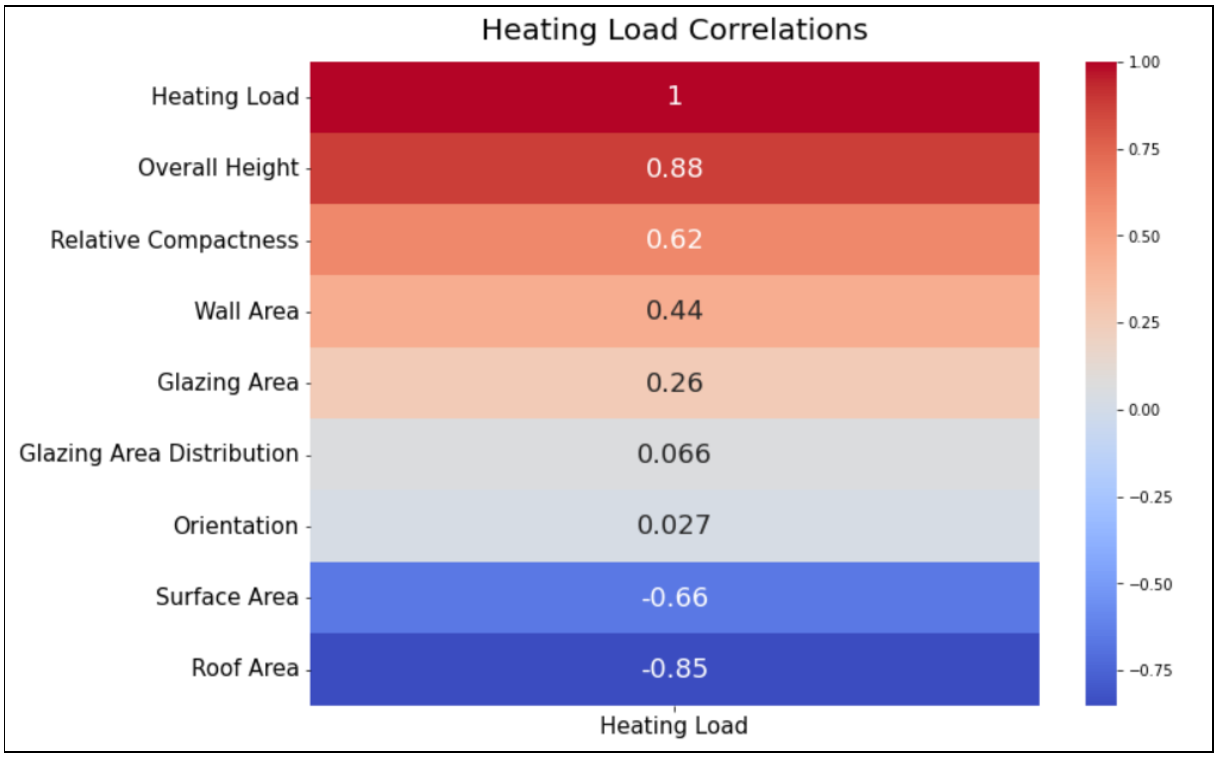
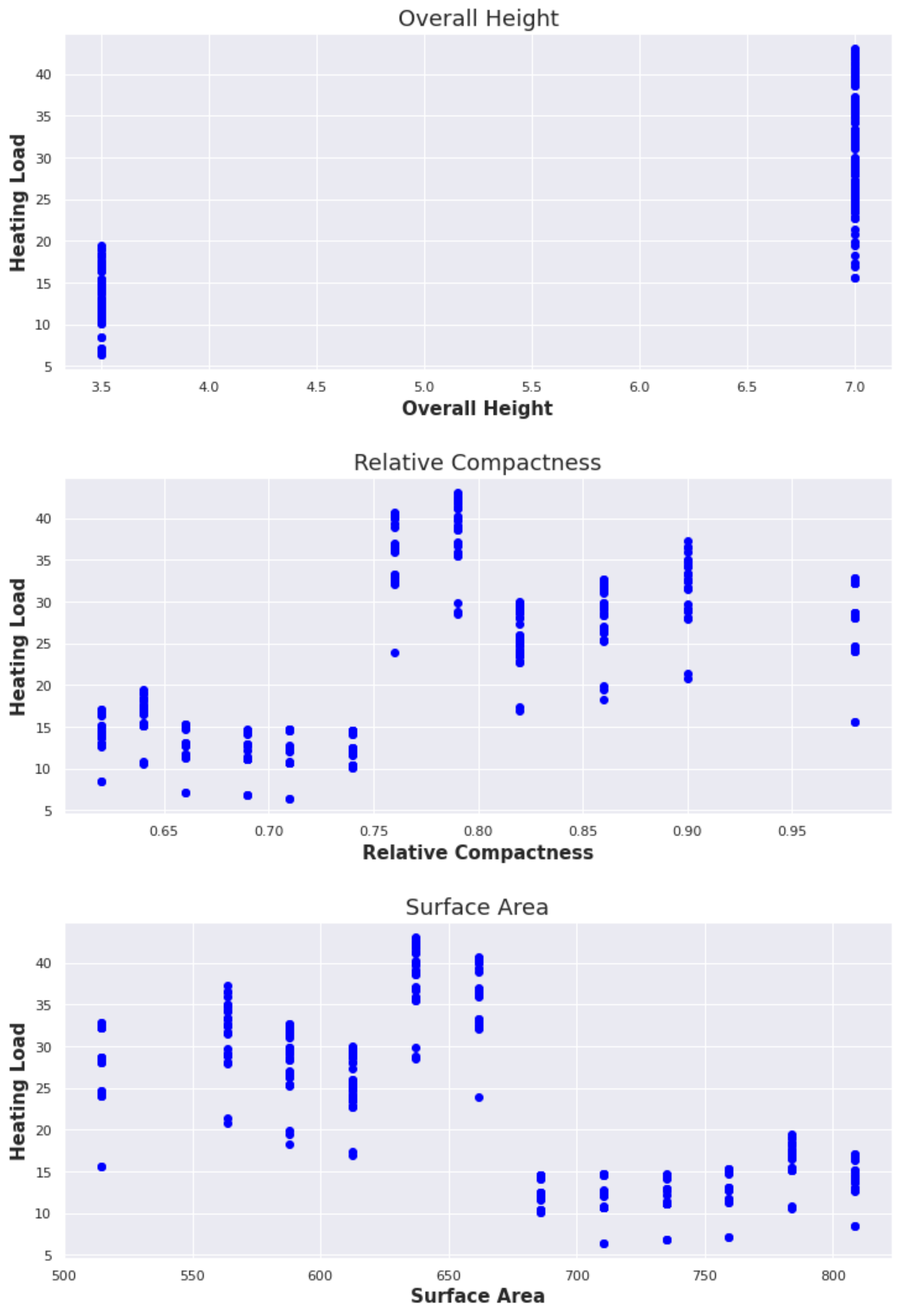


Figure 4. Correlations of variables with our target variable (Heating Load).

We can see that the overall height of a building has a strong positive relationship with the heating load. This makes sense since the heating load describes the amount of energy needed to maintain a target temperature in a space, and a larger space would mean more energy. However, the roof area of a building has a strong negative correlation which does not abide by the same logic of overall height, since we would normally expect a positive correlation there as well. After further inspection, this happens because, in the buildings that were included in this dataset, buildings with a large roof area were half the height of buildings with a smaller (but not half) roof area. Hence, this difference in correlation seems to be due to this architectural decision. Due to these inconsistencies, predicting the heating load might be more challenging using our Bayesian methods. Other variables that positively correlate with heating load are relative compactness, wall area, and glazing area (which make sense) while the surface area has a negative correlation with the heating load. All these make for good predictor variables.

Relationships (Linear or Otherwise)

Next, the relationship between the input and the target variables was examined for the most correlated variables and is shown in Figure 5.



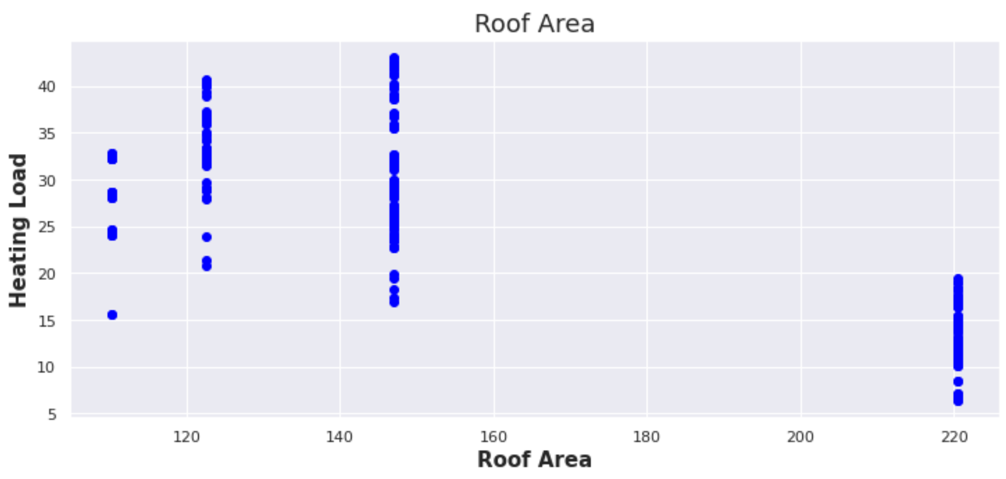


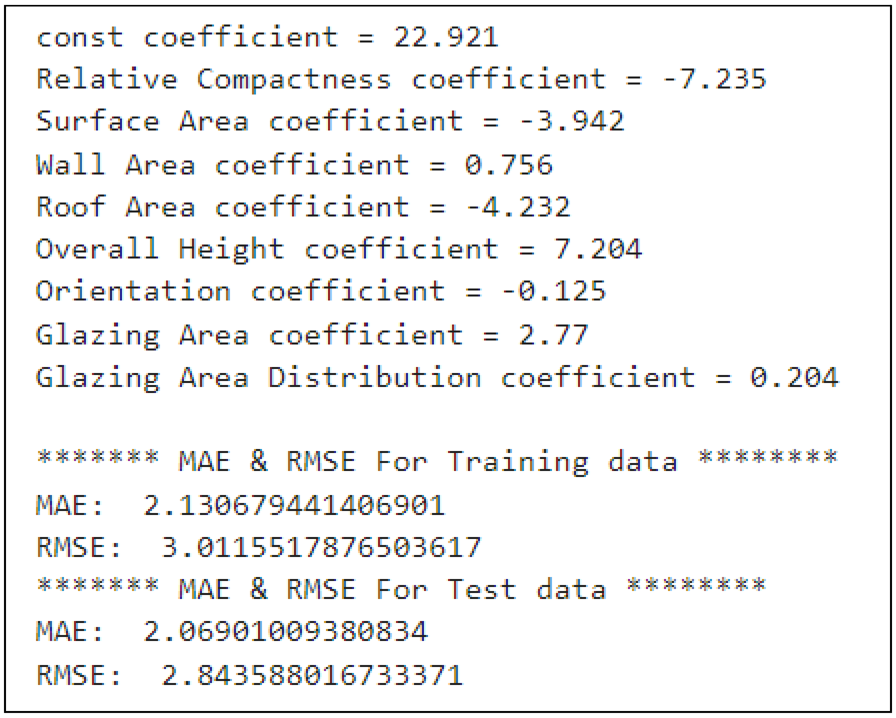
Figure 5: Linearity of input variables with respect to target variable.

We can see that relationships are not clearly linear so this can add to the difficulty of the problem.

Linear Model Fit

Finally, a baseline linear regression model using ordinary least squares was implemented and tested on the train and test sets. Its performance was measured in terms of Root Mean Square Error (RMSE) and Mean Absolute Error (MAE).

The coefficients for all input data along with the baseline model's performance on the training and test sets are shown below.



```
const coefficient = 22.921
Relative Compactness coefficient = -7.235
Surface Area coefficient = -3.942
Wall Area coefficient = 0.756
Roof Area coefficient = -4.232
Overall Height coefficient = 7.204
Orientation coefficient = -0.125
Glazing Area coefficient = 2.77
Glazing Area Distribution coefficient = 0.204

******* MAE & RMSE For Training data ********
MAE:  2.130679441406901
RMSE:  3.0115517876503617
******* MAE & RMSE For Test data ********
MAE:  2.06901009380834
RMSE:  2.843588016733371
```

OLS Model Results and Performance

In the following sections, we will see how this compares to more advanced modelling methods.

Task 2: Type-II Maximum Likelihood

In this task, Type-II maximum likelihood was implemented to estimate the most probable values for hyper-parameters of our model. Priors of hyperparameters *α* and *β* were assumed to follow flat Uniform distributions, and the likelihood function $(y \mid \alpha, \beta)$ was derived to compute the log-likelihood. Then, the marginal likelihood was computed and the and the most probably values were *0.01174* and *0.10837* for *α* and *β* respectively. For this combination of hyperparameters, the maximum marginal likelihood was found to be -*1001.5* with a contour plot for the posterior distribution being shown in Figure 6.

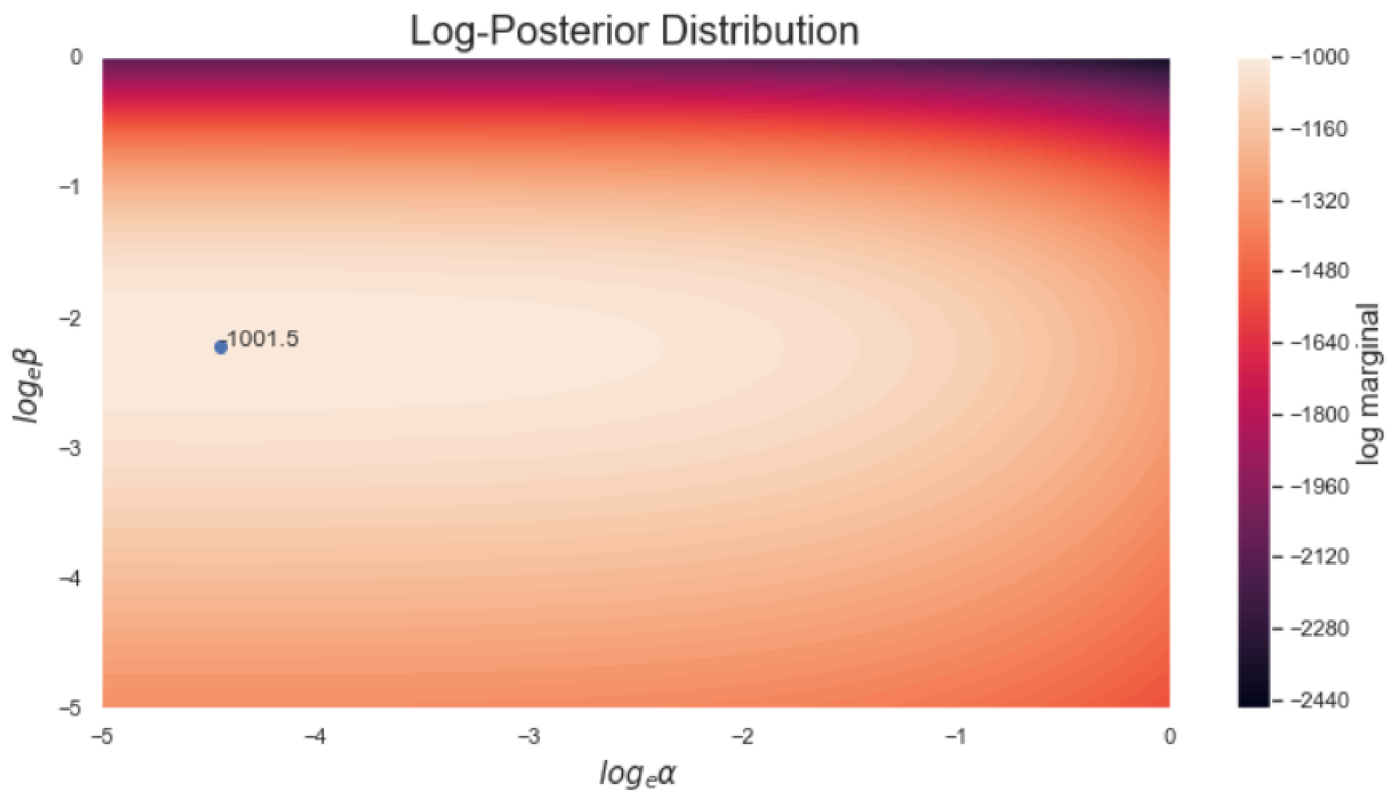
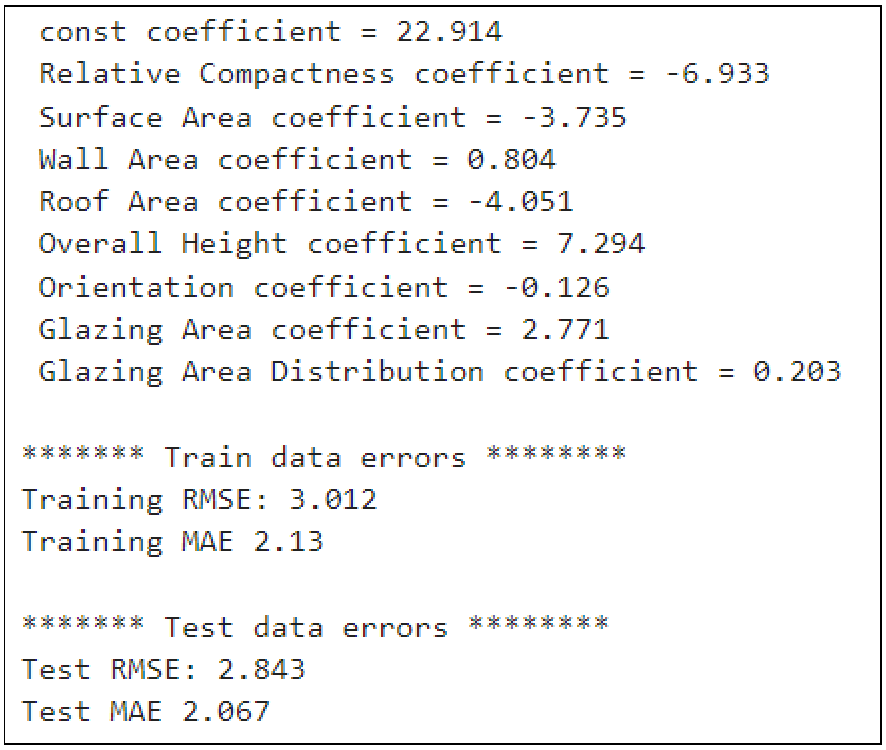


Figure 6: Contour plot of the posterior distribution, highlighting the maximum marginal likelihood.

Finally, the posterior mean *μ* and covariance **Σ** for the Bayesian linear regression model were computed and used to calculate the MAE and RMSE for the training and test sets. The results are shown below, along with the coefficients for each input variable.



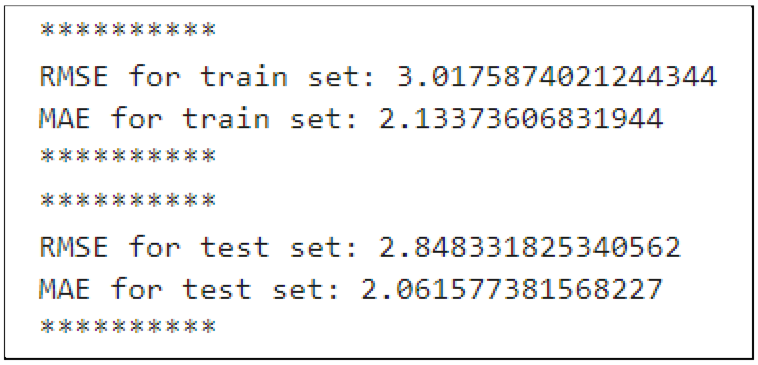
```
 const coefficient = 22.914
 Relative Compactness coefficient = -6.933
 Surface Area coefficient = -3.735
 Wall Area coefficient = 0.804
 Roof Area coefficient = -4.051
 Overall Height coefficient = 7.294
 Orientation coefficient = -0.126
 Glazing Area coefficient = 2.771
 Glazing Area Distribution coefficient = 0.203

******* Train data errors ********
Training RMSE: 3.012
Training MAE 2.13

******* Test data errors ********
Test RMSE: 2.843
Test MAE 2.067
```

Next, we will see how Variational Inference methods perform on this task.

Task 3: Variational Inference

Similarly to the previous task, the goal in Task 3 was to estimate the most probable values for α and β, but this time using Variational Inference with simple Mean-Field Theory factorisation[1].

Using variational inference the model converged after four iterations and the expectations of α and β were *0.1103* and *0.1085* respectively. The performance of this model in terms of MAE and RMSE was the following:



```
**********
RMSE for train set: 3.0175874021244344
MAE for train set: 2.13373606831944
**********
**********
RMSE for test set: 2.848331825340562
MAE for test set: 2.061577381568227
**********
```

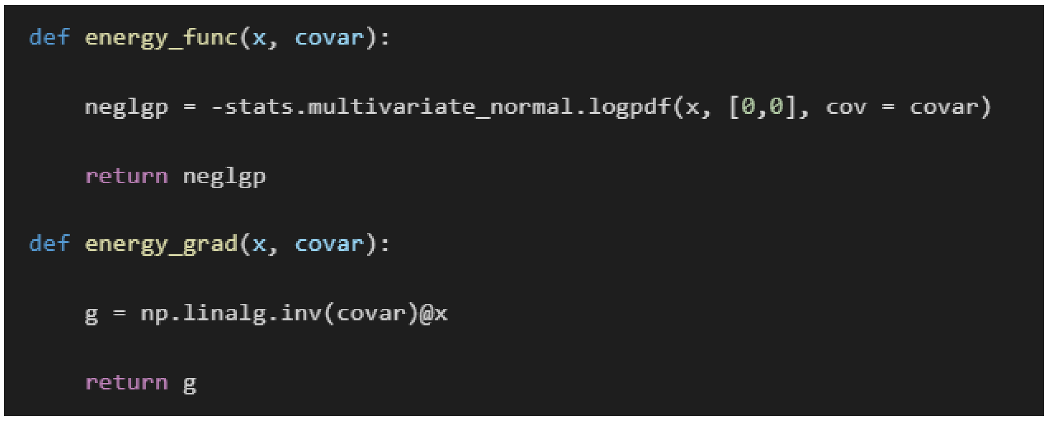
We can see that VI performance on the test set is very similar to that of Type-II Maximum Likelihood as well as to that of the baseline OLS model.

[1] https://sccn.ucsd.edu/~rapela/docs/vblr.pdf

Task 4: Verify HMC on a Standard 2D Gaussian Example

• Write down the mathematical formula of the standard 2D Gaussian example, please make sure the variables are consistent to the rest of your report
• Verify and demonstrate (with appropriate figures or numerical tables) that your HMC works as expected

In this task, the Hamiltonian Monte Carlo (HMC) sampling algorithm was tested on a standard Gaussian example in two dimensions. The energy function that was used was simply the negative log-probability of the distribution to be sampled from, while the gradient function is the gradient of the energy function with respect to the hyperparameters. The respective implementations are shown in Figure 7.



```
def energy_func(x, covar):

    neglgp = -stats.multivariate_normal.logpdf(x, [0,0], cov = covar)

    return neglgp

def energy_grad(x, covar):

    g = np.linalg.inv(covar)@x

    return g
```

Figure 7: Energy and gradient functions for HMC example.

An example mathematical representation of a 2D Gaussian is

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

where x_0, y_0 is the centre, and σ_x, σ_y are the *x* and *y* spreads. Our distribution is visualised in Figure 8.

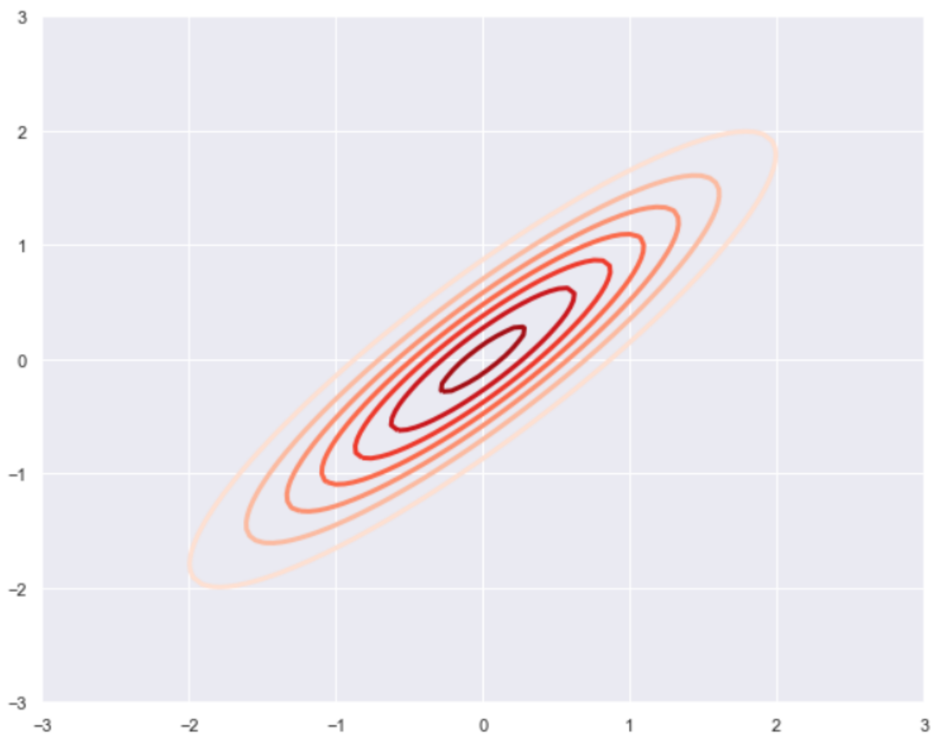
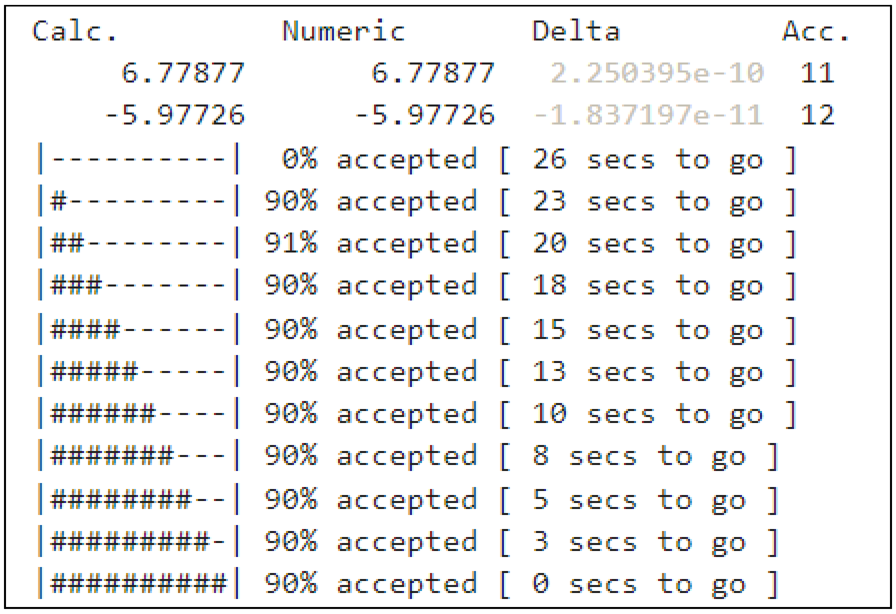


Figure 8: Standard 2D Gaussian Distribution.

Before running the final test of our HMC sampler, we optimised the following hyperparameters and chose the quoted values:

R = 10,000 (number of samples desired)
L = 20 (number of steps to run the Hamiltonian dynamics within each cycle of the sampler)
eps = 0.3925 (step size for the dynamics).

The sampling process output is the following:



```
Calc.          Numeric        Delta          Acc.
     6.77877        6.77877    2.250395e-10   11
    -5.97726       -5.97726   -1.837197e-11   12
|----------|  0% accepted [ 26 secs to go ]
|#---------| 90% accepted [ 23 secs to go ]
|##--------| 91% accepted [ 20 secs to go ]
|###-------| 90% accepted [ 18 secs to go ]
|####------| 90% accepted [ 15 secs to go ]
|#####-----| 90% accepted [ 13 secs to go ]
|######----| 90% accepted [ 10 secs to go ]
|#######---| 90% accepted [ 8 secs to go ]
|########--| 90% accepted [ 5 secs to go ]
|#########-| 90% accepted [ 3 secs to go ]
|##########| 90% accepted [ 0 secs to go ]
```

The sampler achieved an 89.8% acceptance and the sampling is visualised in Figure 9.

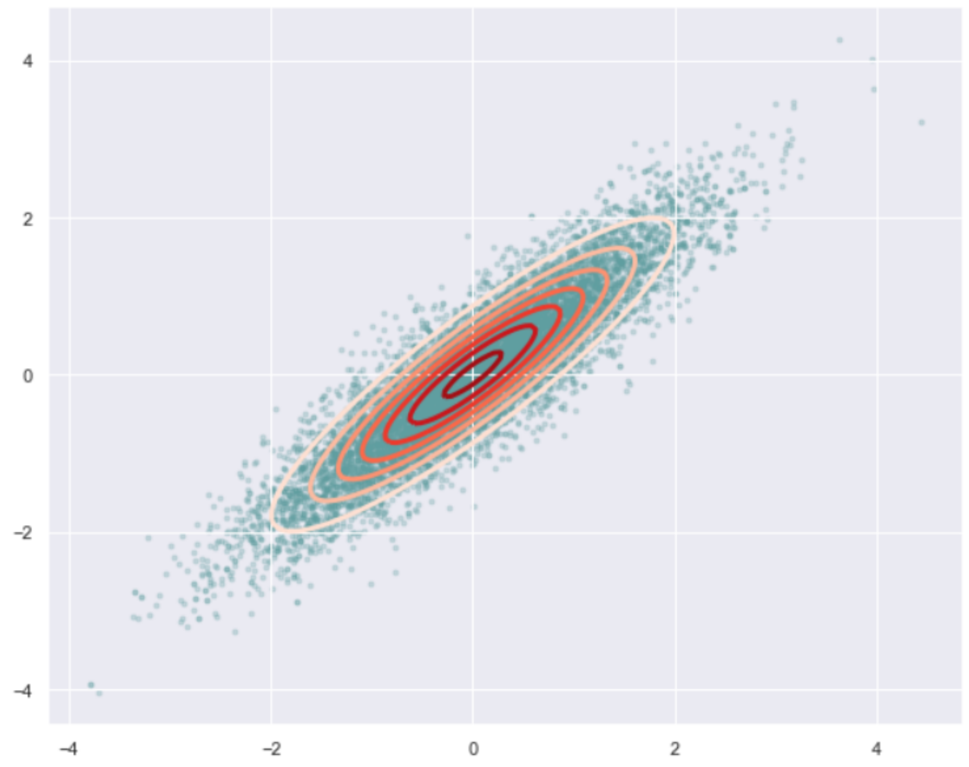


Figure 9: Visualisation of our HMC sampler sample points from a standard 2D Gaussian.

Task 5: Apply HMC to the Linear Regression Model

In this task, the goal was to apply HMC to the Bayesian model to sample weights and hyperparameters. To calculate the energy function we first take the negative log of the likelihood and the prior:

$$-\log P(w,\alpha,\beta \mid x,y) = -\log P(y \mid x,w,\beta) - \log P(w \mid \alpha),$$

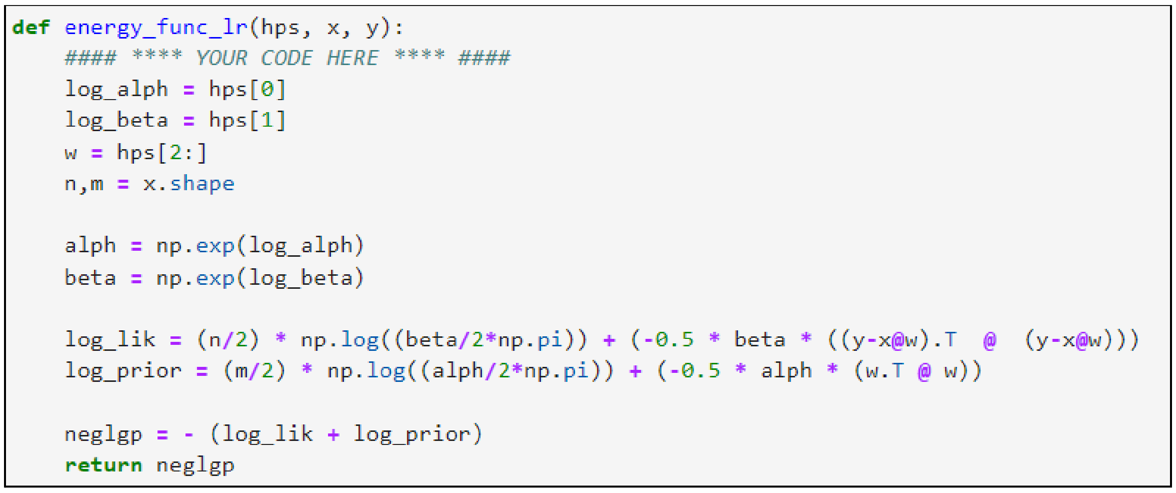
where the functions are defined as

$$P(\omega,\alpha,\beta \mid x,y) = P(y \mid x,\omega,\beta)\cdot P(\omega|\alpha)$$

$$P(y \mid x,w,\beta) = \left(\frac{\beta}{2\pi}\right)^{\frac{N}{2}} e^{-\frac{\beta\|xw-y\|^2}{2}}$$

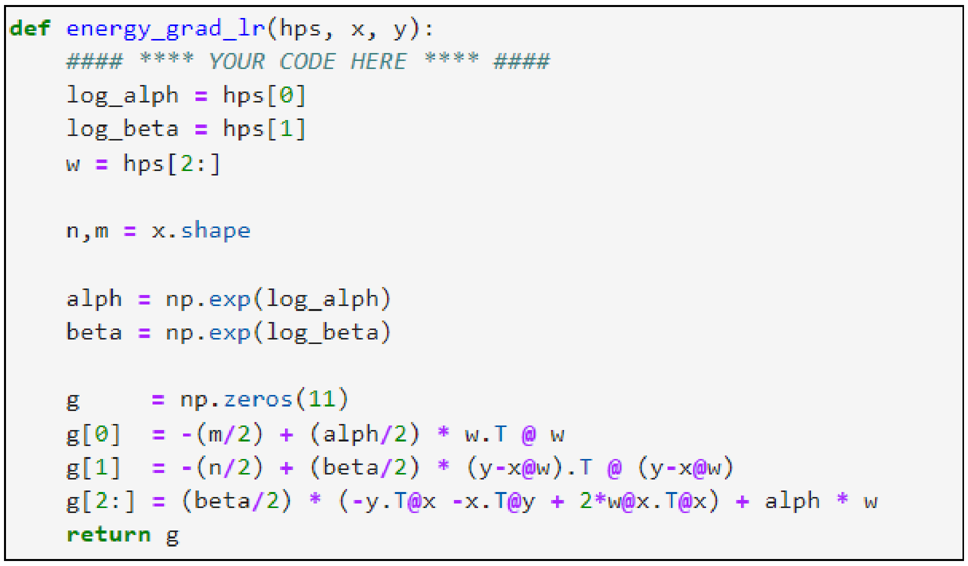
$$P(w \mid \alpha) = \left(\frac{\alpha}{2\pi}\right)^{\frac{M}{2}} e^{-\frac{\alpha}{2}}\sum_{m=1}^{M} w_m^2$$

Then, we use the partial derivatives with respect to *α* and *β* to derive our energy gradient. The final results were the following:



```
def energy_func_lr(hps, x, y):
    #### **** YOUR CODE HERE **** ####
    log_alph = hps[0]
    log_beta = hps[1]
    w = hps[2:]
    n,m = x.shape

    alph = np.exp(log_alph)
    beta = np.exp(log_beta)

    log_lik = (n/2) * np.log((beta/2*np.pi)) + (-0.5 * beta * ((y-x@w).T  @  (y-x@w)))
    log_prior = (m/2) * np.log((alph/2*np.pi)) + (-0.5 * alph * (w.T @ w))

    neglgp = - (log_lik + log_prior)
    return neglgp
```

Figure 10: Energy Function



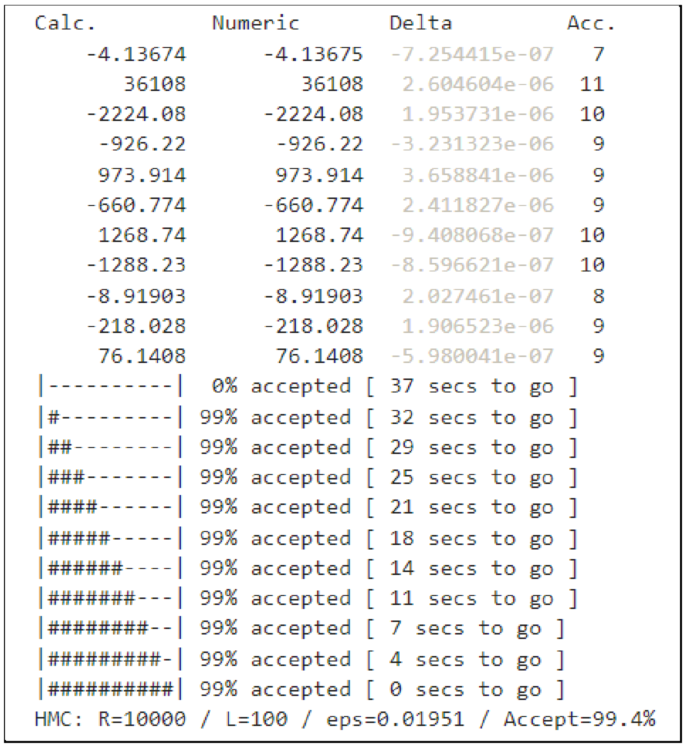
```python
def energy_grad_lr(hps, x, y):
    #### **** YOUR CODE HERE **** ####
    log_alph = hps[0]
    log_beta = hps[1]
    w = hps[2:]

    n,m = x.shape

    alph = np.exp(log_alph)
    beta = np.exp(log_beta)

    g     = np.zeros(11)
    g[0]  = -(m/2) + (alph/2) * w.T @ w
    g[1]  = -(n/2) + (beta/2) * (y-x@w).T @ (y-x@w)
    g[2:] = (beta/2) * (-y.T@x -x.T@y + 2*w@x.T@x) + alph * w
    return g
```

Figure 11: Energy Gradient Function

Using the same method as in the example in task 4, the chosen values for the hyperparameters were the following:

R = 10000
L = 100
eps = 0.01951

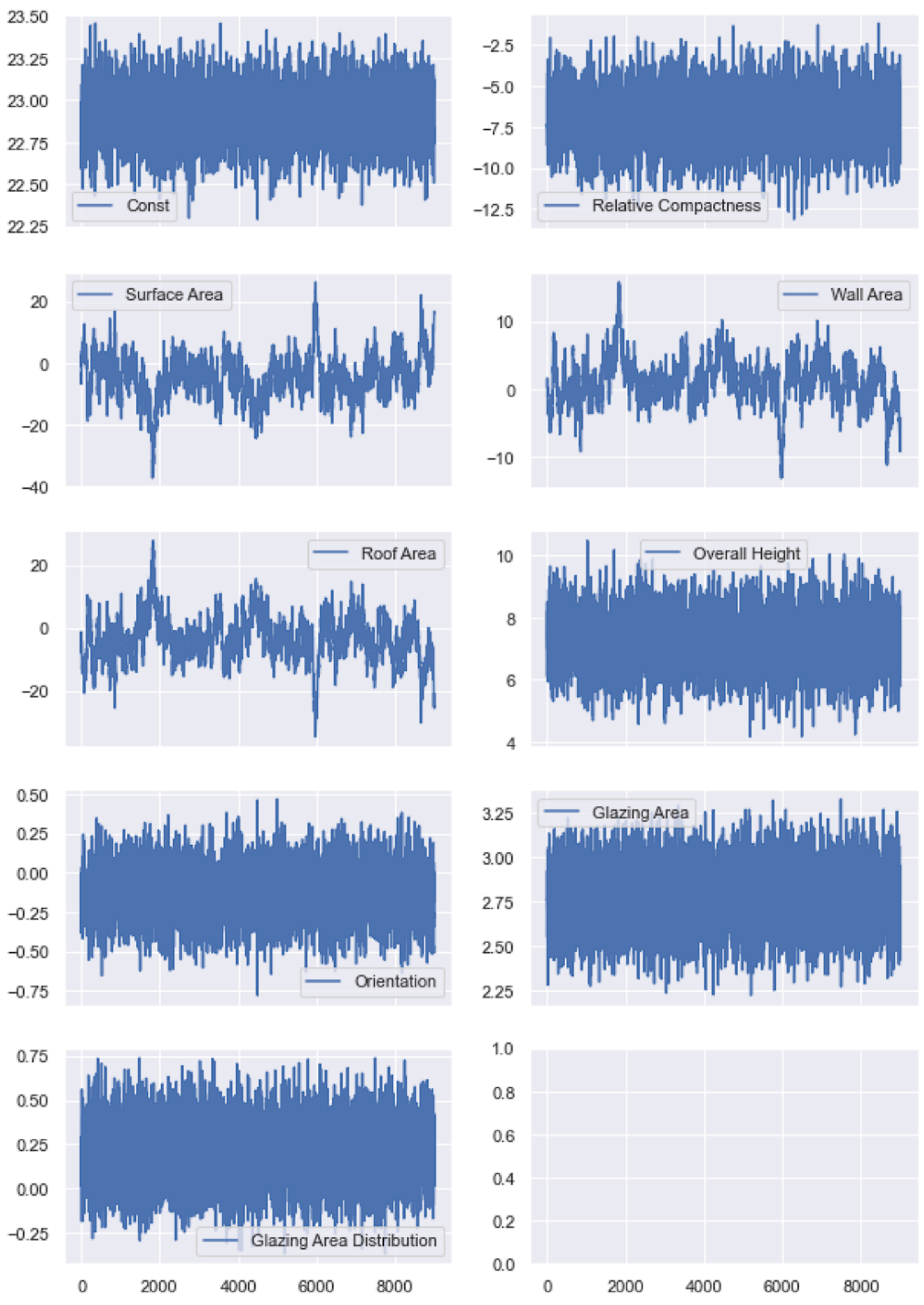
The sampling outputs are shown in Figure 12, and our HMC sampler achieved an acceptance of 99.4%.



```
Calc.           Numeric         Delta           Acc.
    -4.13674        -4.13675    -7.254415e-07    7
       36108           36108     2.604604e-06   11
    -2224.08        -2224.08     1.953731e-06   10
     -926.22         -926.22    -3.231323e-06    9
     973.914         973.914     3.658841e-06    9
    -660.774        -660.774     2.411827e-06    9
     1268.74         1268.74    -9.408068e-07   10
    -1288.23        -1288.23    -8.596621e-07   10
    -8.91903        -8.91903     2.027461e-07    8
    -218.028        -218.028     1.906523e-06    9
     76.1408         76.1408    -5.980041e-07    9
|----------|  0% accepted [ 37 secs to go ]
|#---------| 99% accepted [ 32 secs to go ]
|##--------| 99% accepted [ 29 secs to go ]
|###-------| 99% accepted [ 25 secs to go ]
|####------| 99% accepted [ 21 secs to go ]
|#####-----| 99% accepted [ 18 secs to go ]
|######----| 99% accepted [ 14 secs to go ]
|#######---| 99% accepted [ 11 secs to go ]
|########--| 99% accepted [ 7 secs to go ]
|#########-| 99% accepted [ 4 secs to go ]
|##########| 99% accepted [ 0 secs to go ]
HMC: R=10000 / L=100 / eps=0.01951 / Accept=99.4%
```

Figure 12: Sampling outputs

The plots in Figure 13 help visualise the sampling that occurred for each input variable and hyperparameter:


Const
Relative Compactness
Surface Area
Wall Area
Roof Area
Overall Height
Orientation
Glazing Area
Glazing Area Distribution
0
2000
4000
6000
8000

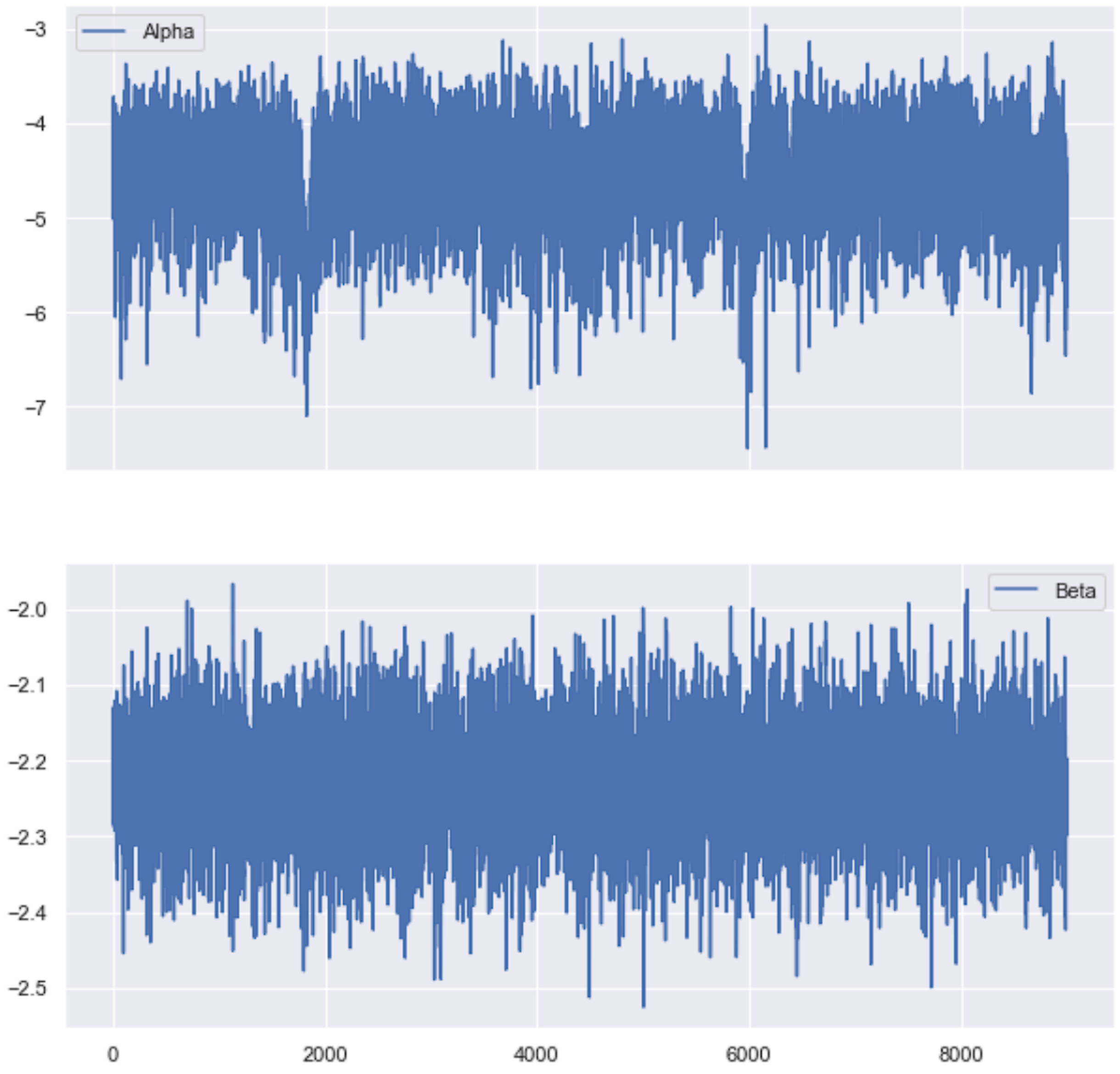
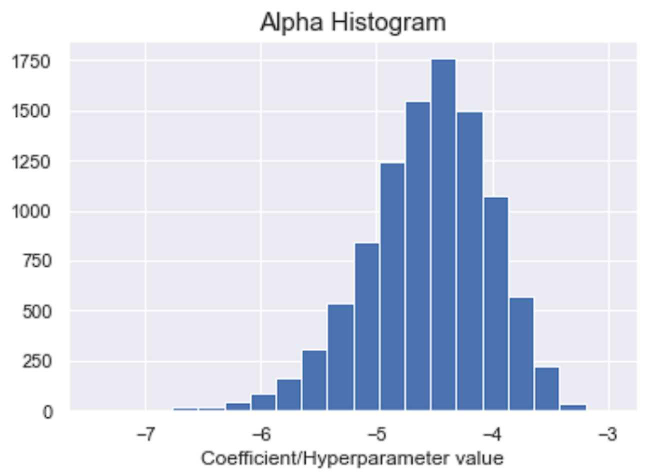
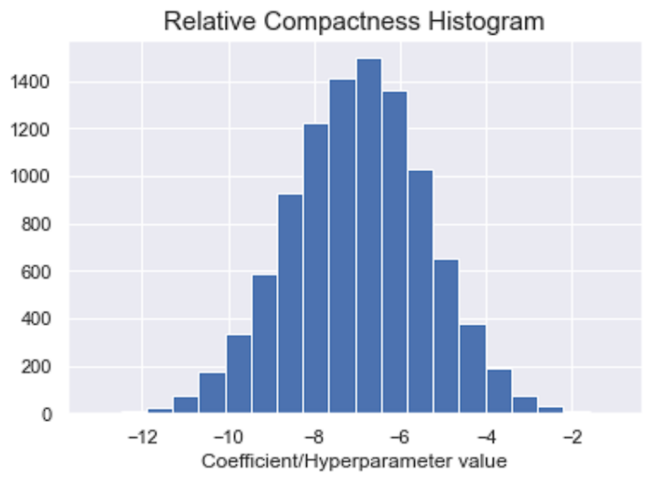
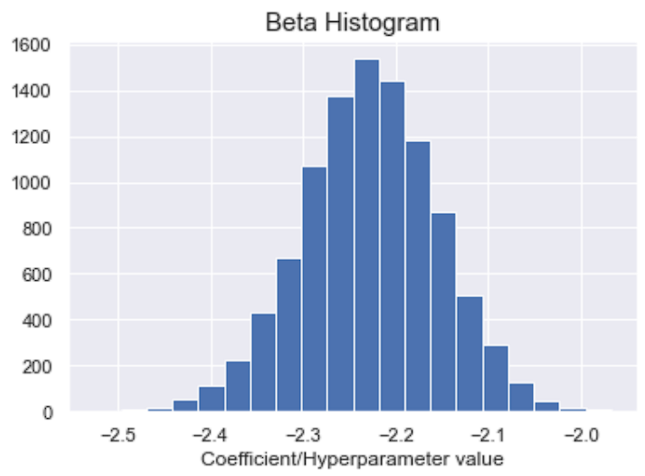


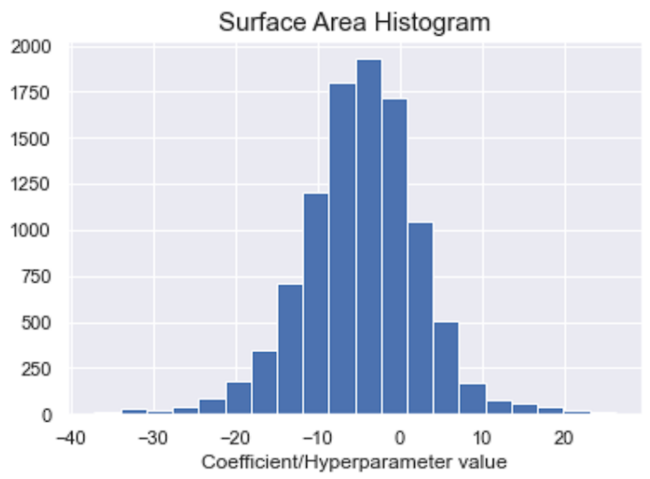
Figure 13: Visualisation of HMC Sampler

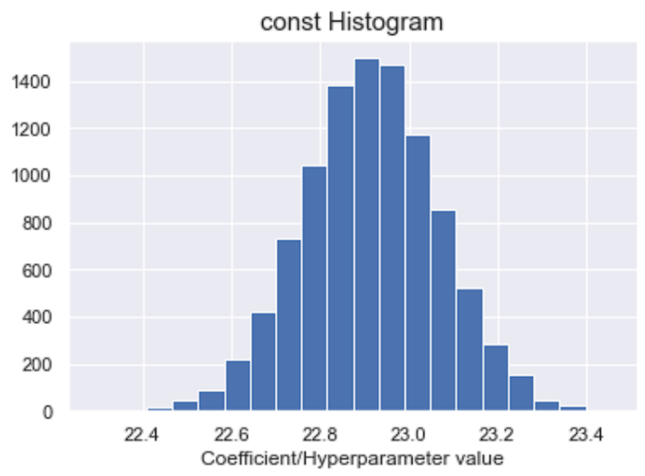
From this, we can see that the sampler is working as expected. In addition, the Gaussian distribution functionality is visualised in the following histograms, where we also notice that the values for *α* are mildly skewed:

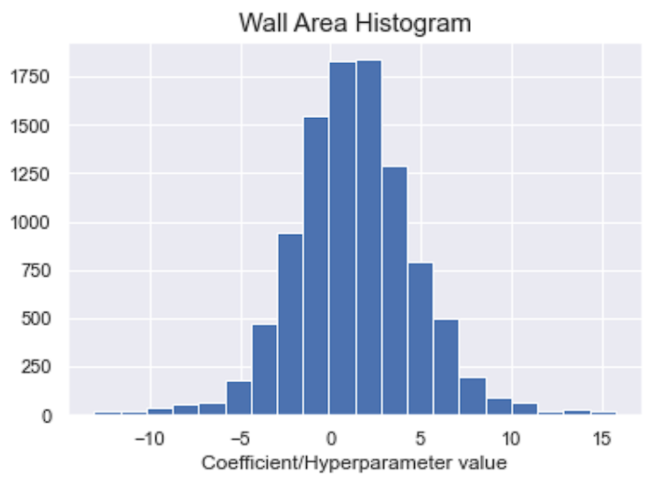

Alpha Histogram
1750
1500
1250
1000
750
500
250
0
−7
−6
−5
−4
−3
Coefficient/Hyperparameter value

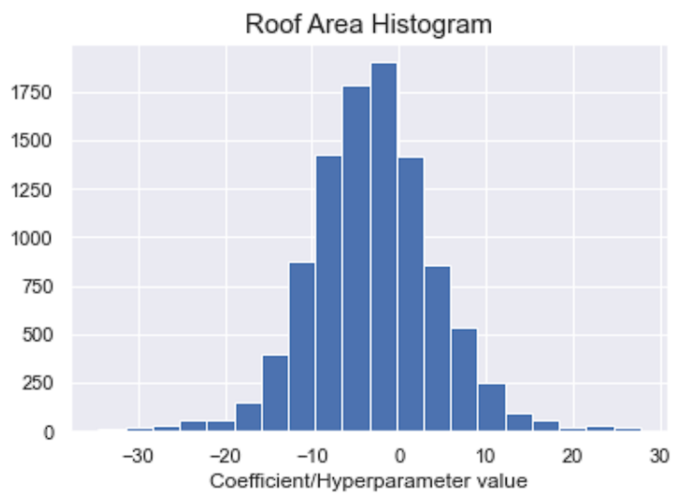

Relative Compactness Histogram
1400
1200
1000
800
600
400
200
0
−12
−10
−8
−6
−4
−2
Coefficient/Hyperparameter value

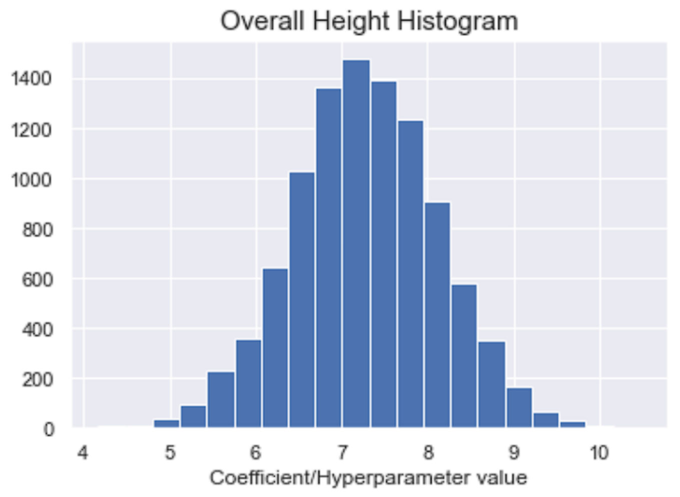

Beta Histogram
1600
1400
1200
1000
800
600
400
200
0
−2.5
−2.4
−2.3
−2.2
−2.1
−2.0
Coefficient/Hyperparameter value

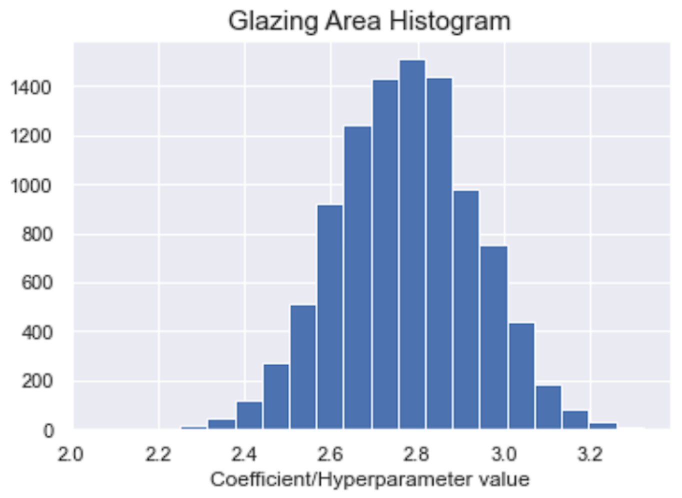

Surface Area Histogram
2000
1750
1500
1250
1000
750
500
250
0
−40
−30
−20
−10
0
10
20
Coefficient/Hyperparameter value

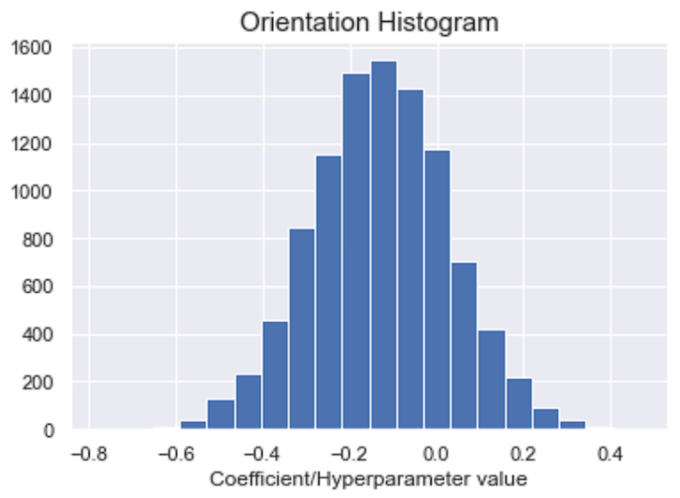

const Histogram
1400
1200
1000
800
600
400
200
0
22.4
22.6
22.8
23.0
23.2
23.4
Coefficient/Hyperparameter value

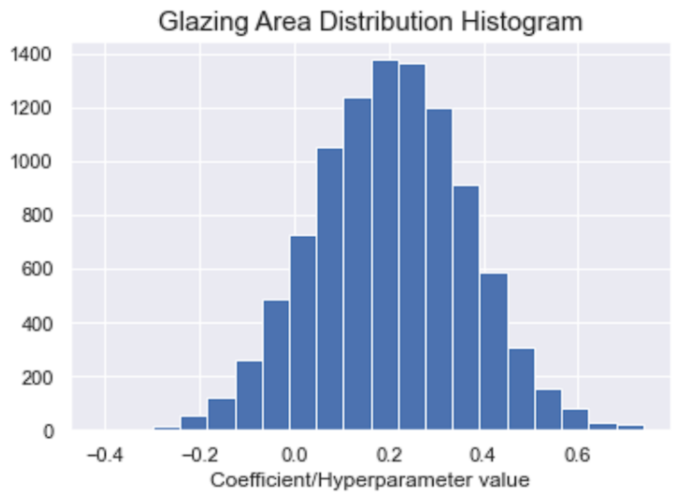

Wall Area Histogram
1750
1500
1250
1000
750
500
250
0
−10
−5
0
5
10
15
Coefficient/Hyperparameter value


Roof Area Histogram
Coefficient/Hyperparameter value

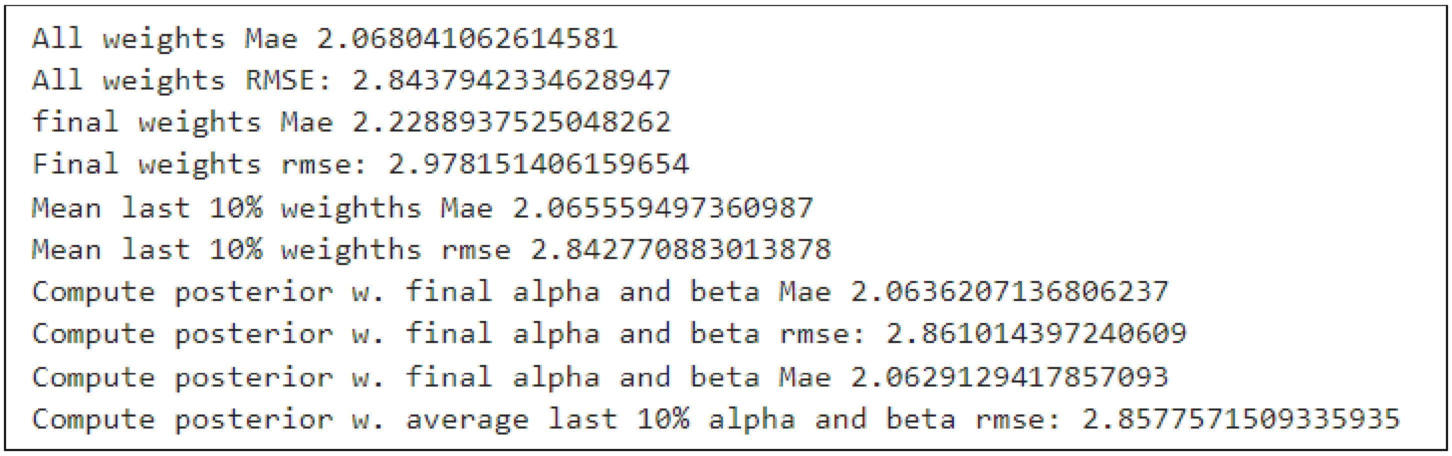

Overall Height Histogram
Coefficient/Hyperparameter value


Glazing Area Histogram
Coefficient/Hyperparameter value


Orientation Histogram
Coefficient/Hyperparameter value


Glazing Area Distribution Histogram
Coefficient/Hyperparameter value

Finally, using HMC sampling methods, the following performance metrics were yielded:



```
All weights Mae 2.068041062614581
All weights RMSE: 2.8437942334628947
final weights Mae 2.2288937525048262
Final weights rmse: 2.978151406159654
Mean last 10% weighths Mae 2.065559497360987
Mean last 10% weighths rmse 2.842770883013878
Compute posterior w. final alpha and beta Mae 2.0636207136806237
Compute posterior w. final alpha and beta rmse: 2.861014397240609
Compute posterior w. final alpha and beta Mae 2.0629129417857093
Compute posterior w. average last 10% alpha and beta rmse: 2.8577571509335935
```

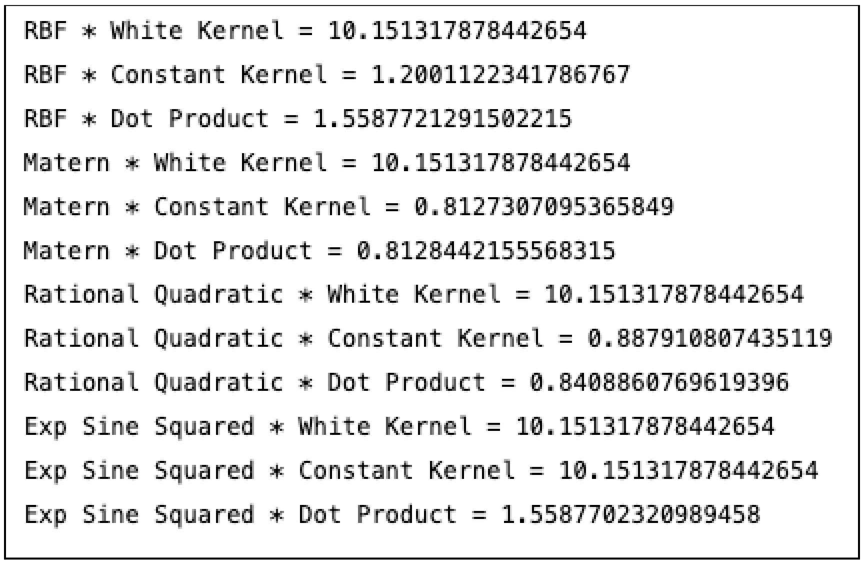
We notice that even with HMC sampling, the model performed very similarly in terms of MAE and RMSE to our previous implementations, even to the baseline OLS model.

Task 6: Apply GP to the Linear Regression Model

In this final task, we implemented Gaussian process regression (GPR). GPR generally works well on small datasets and has the ability to provide uncertainty measurements on the predictions.
GP was used to sample weights of the standard Bayesian regression model. Specifically, GP was applied to obtain the joint posterior over the LR coefficients **w**.

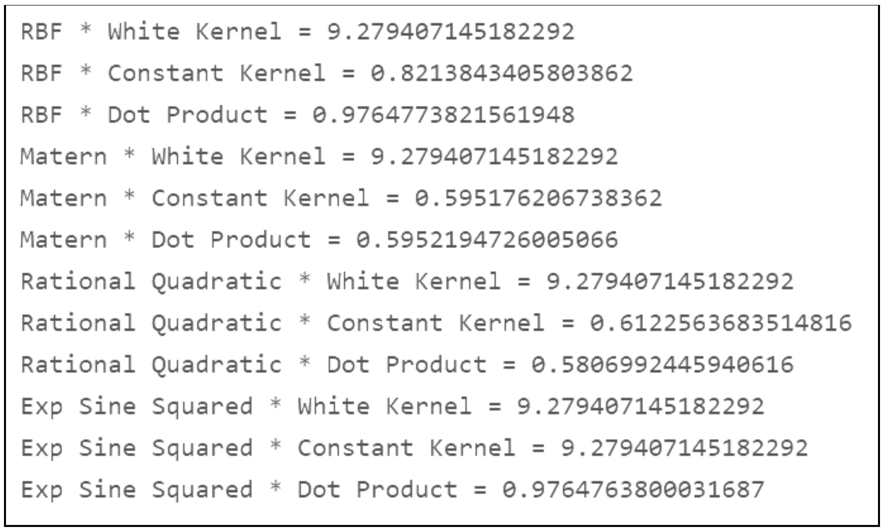
Multiple kernel functions were used and the results for each are shown below:

RMSE



```
RBF * White Kernel = 10.151317878442654
RBF * Constant Kernel = 1.2001122341786767
RBF * Dot Product = 1.5587721291502215
Matern * White Kernel = 10.151317878442654
Matern * Constant Kernel = 0.8127307095365849
Matern * Dot Product = 0.8128442155568315
Rational Quadratic * White Kernel = 10.151317878442654
Rational Quadratic * Constant Kernel = 0.887910807435119
Rational Quadratic * Dot Product = 0.8408860769619396
Exp Sine Squared * White Kernel = 10.151317878442654
Exp Sine Squared * Constant Kernel = 10.151317878442654
Exp Sine Squared * Dot Product = 1.5587702320989458
```

MAE



```
RBF * White Kernel = 9.279407145182292
RBF * Constant Kernel = 0.8213843405803862
RBF * Dot Product = 0.9764773821561948
Matern * White Kernel = 9.279407145182292
Matern * Constant Kernel = 0.595176206738362
Matern * Dot Product = 0.5952194726005066
Rational Quadratic * White Kernel = 9.279407145182292
Rational Quadratic * Constant Kernel = 0.6122563683514816
Rational Quadratic * Dot Product = 0.5806992445940616
Exp Sine Squared * White Kernel = 9.279407145182292
Exp Sine Squared * Constant Kernel = 9.279407145182292
Exp Sine Squared * Dot Product = 0.9764763800031687
```

We can see that the rational quadratic * dot product function outperformed others in terms of MAE, and GP methods did significantly better on the test set compared to all previous linear regression implementations which yielded consistently higher errors.