Exploratory analysis

Describing the initial steps taken in exploring and analysing the structure of the given dataset.

The data provided are originally from the University of Oxford, made available at the UCI Machine Learning Repository and provide recordings to help analyse Energy Efficiency. The whole dataset containing 768 examples comprises of 8 input values (Relative Compactness, Surface Area, Wall Area, Roof Area, Overall Height, Orientation, Glazing Area, Glazing Area Distribution) that show some architectural building parameters which can be used to predict the 9th target variable which is the Heating Load. These are then split between a training and a test set. An example of the structure of the data with the first 10 examples of the training set are as shown by Figure 1.

	Relative Compactness	Surface Area	Wall Area	Roof Area	Overall Height	Orientation	Glazing Area	Glazing Area Distribution	Heating Load
0	0.62	808.5	367.5	220.50	3.5	3	0.10	5	12.74
1	0.90	563.5	318.5	122.50	7.0	3	0.10	2	29.68
2	0.90	563.5	318.5	122.50	7.0	3	0.40	2	36.57
3	0.79	637.0	343.0	147.00	7.0	2	0.25	2	38.57
4	0.90	563.5	318.5	122.50	7.0	4	0.40	5	34.72
5	0.64	784.0	343.0	220.50	3.5	2	0.25	4	16.99
6	0.64	784.0	343.0	220.50	3.5	3	0.10	4	15.19
7	0.98	514.5	294.0	110.25	7.0	3	0.25	3	28.56
8	0.71	710.5	269.5	220.50	3.5	2	0.40	1	14.51
9	0.86	588.0	294.0	147.00	7.0	5	0.10	1	25.27

Figure 1. Dataset Structure Examples

The data has been checked that there are no missing values in any of the fields and some basic statistics are analysed and are as shown by Figure 2.

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

Figure 2. Training Set Basic Statistics

A pre-processing step that had been made for the purpose of aiding in modelling was standardizing each parameter. Standardizing the data column is achieved by finding its Z-score, which is subtracting the mean of the column and diving each value with the standard deviation of the column (Figure 3). This causes each column to have a mean value of 0 with standard deviation 1 and was performed for every feature except the target variable.

The purpose of this step is to be able to properly measure the standardized coefficients of each parameter for measuring and comparing the importance and commonality of each parameter in respect to the target value, Heating Load.

$$z = \frac{value - mean}{standard\ deviation}$$

Figure 3. Z-score formula

The initial observation that was made, was to identify the linearity between the data and assess the difficulty of the task. Comparing each feature column with the target variable, as shown in Figure 5, has exhibited the linearity between each piece of data in correlation to the target value, either showing positive, negative, or negligible linearity such as in the example of the Orientation.

To achieve the results shown by Figure 5, an iteration of each feature column was made a scatter plot of the feature values and the target values was plotted, these are represented by the colour blue. Then to identify and plot the orange line we use a polyfit function that uses the x and y values to find calculate the slope and y-intercept and draw the line that best fit the data. The code use to find these results is shown in the following Figure 4.

```
# Plotting all features against the Target Column
for feature in feature_columns:

    plt.figure(figsize=(8, 3))
    sns.scatterplot(data=train_stand, x='Heating Load', y=feature, color='blue', alpha=0.4)

    # https://www.kite.com/python/answers/how-to-plot-a-linear-regression-line-on-a-scatter-pl
    # Plotting line of best fit on data in the scatter plot
    m, b = np.polyfit(train_stand['Heating Load'], train_stand[feature], 1)
    plt.plot(train_stand['Heating Load'], m*train_stand['Heating Load'] + b, color='orange')
    plt.xlabel('Heating Load')
    plt.ylabel(feature)
    plt.show()
```

Figure 4. Code for plotting graphs showing linearity.

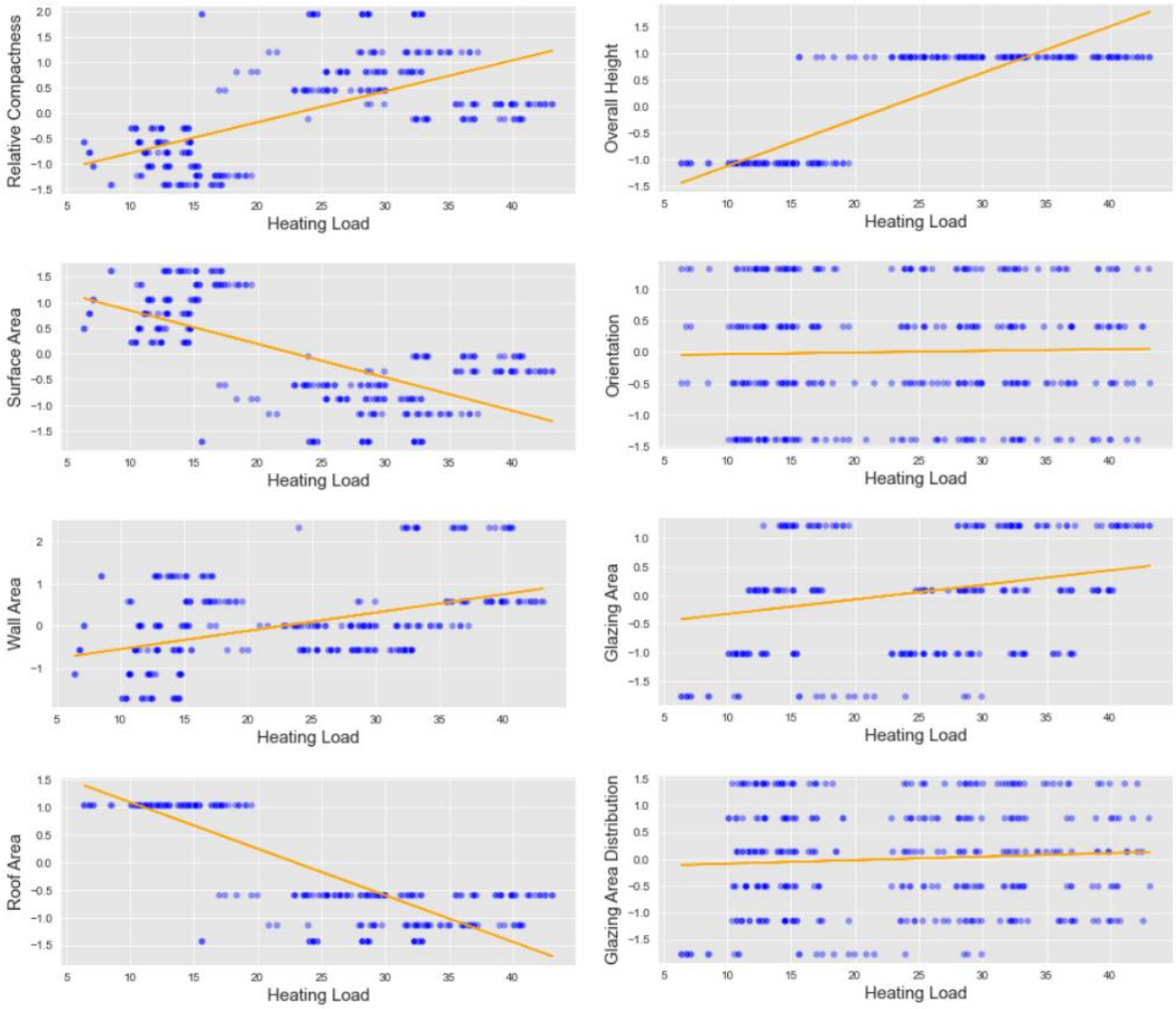

Figure 5. Graphs showing linearity amongst features and the target value.

A further approach that was made in helping to identify the correlation amongst the different features of the dataset was to plot a correlation heatmap. As shown by Figure 6, this heatmap clearly shows the positive, negative or no correlation between the features. An example is the correlation between the Roof Area and the Surface Area, shown in bright red that has a 0.89 correlation. A mask was applied to hide the upper right diagonal as it provides no additional information in the graph.

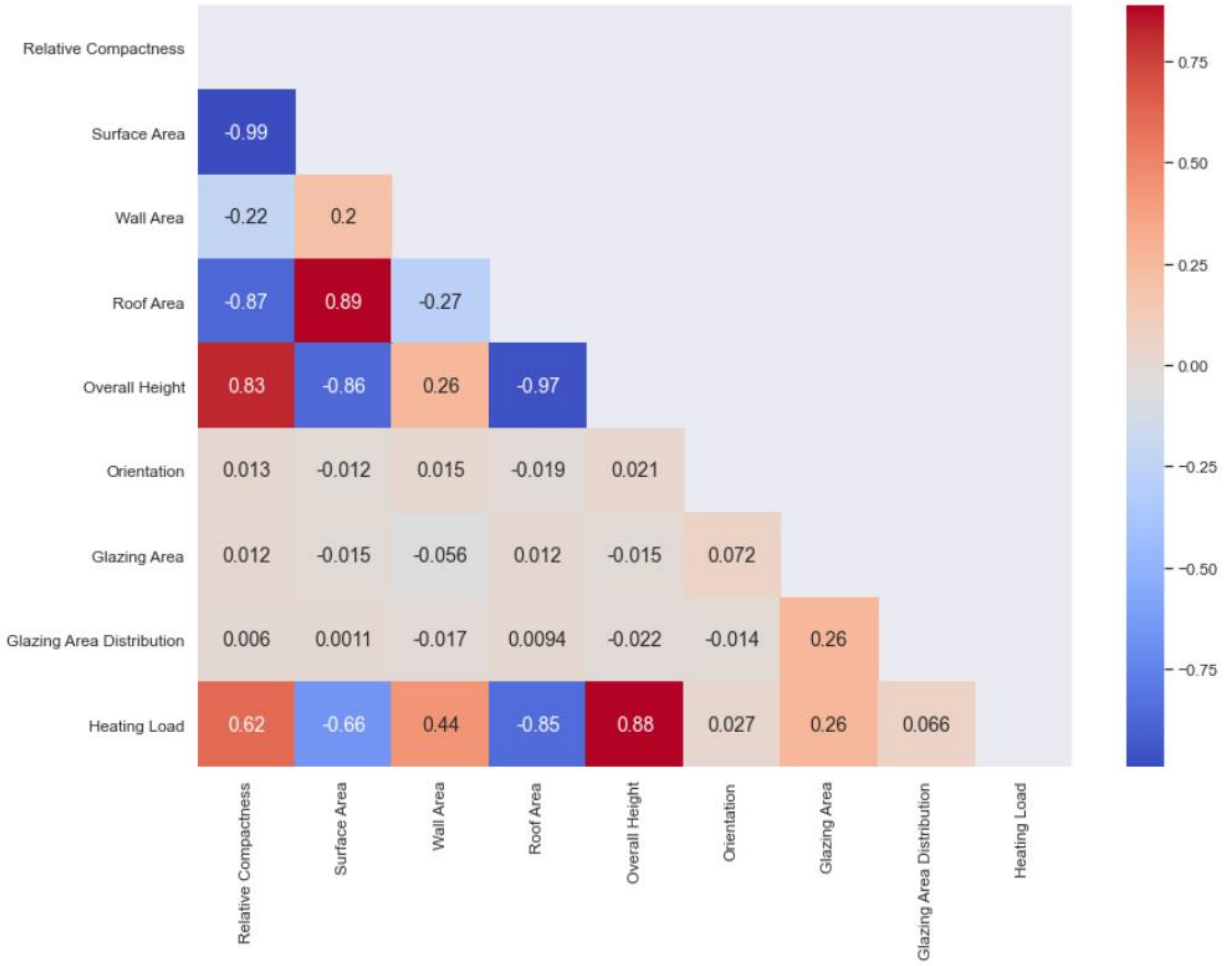

Figure 6. Correlation Heatmap between features

Then another heatmap was created that only shows the correlation or the likely relevance of the variables for predicting the target value, Heating Load. These where then sorted in descending order and presented by Figure 7. With these results it is clear to identify which features have the highest relevance with Heating Load, with the first one being Overall Height with a 0.88 correlation.

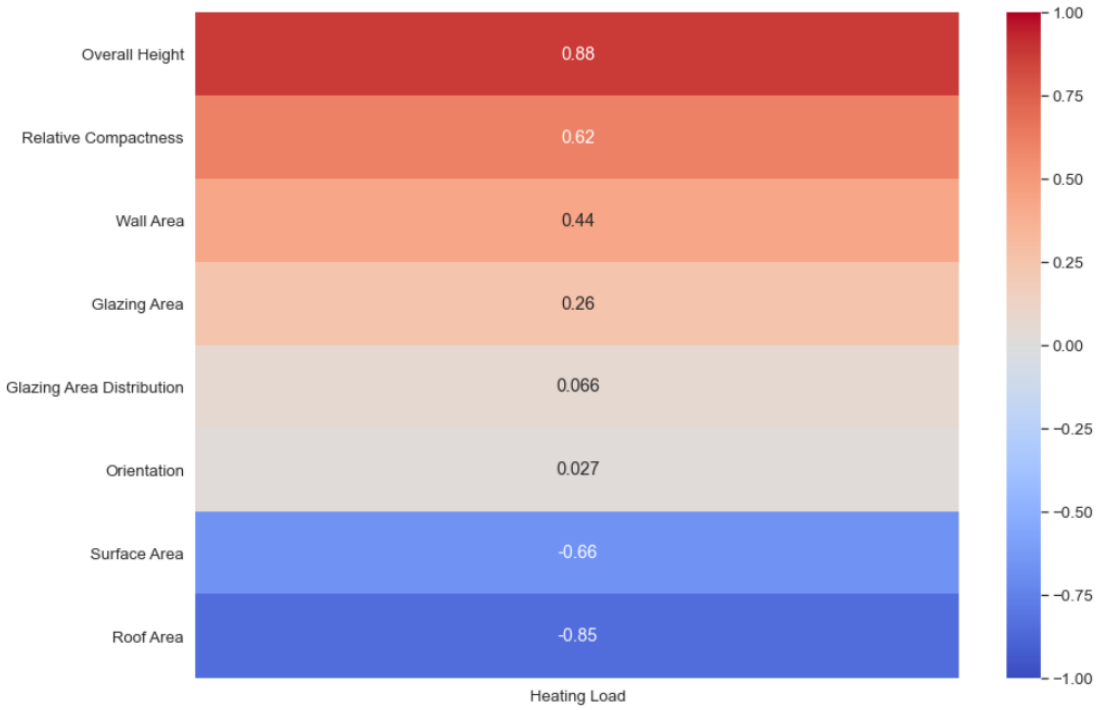

Figure 7. Sorted Heating Load correlation heatmap.

Finally, after exploring and summarizing the dataset an ordinary least squares Linear Regression method was applied using the standardized training and testing data that helped detail the accuracy of the model by using the Root Mean Square Error metric. This metric is the root mean of the squares of the errors. Using the formula as shown by Figure 8 the difference between the predicted values with the actual values is found that helps identify the dissimilarity and show the error between all the data used.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

Figure 8. Root Mean Square Error Formula

As shown by Figure 9, a LinearRegression object was created using functionalities from the library scikit-learn, which was then fitted on the training data. The fitted model was the used to make prediction on both the test set and the training set and the results obtained are as shown in Table 1. In both sets the model had a Root Mean Square Error just over 3 (rounded to 3 decimal places) and with the similarity between these results it shows that the trained Regression model was properly applied in making appropriate predictions on the Test set.

```
# Initializing, fitting and predicting on both Sets
model = LinearRegression()
model.fit(x_train, y_train)
y_test_pred = model.predict(x_test)
y_train_pred = model.predict(x_train)
```

Figure 9. Applying Linear Regression (code)

Table 1. Least-Squares Linear Model RMSE results

Set	Root Mean Square Error (RMSE) (5 d.p.)
Train	3.01155
Test	3.09589

Bayesian Linear Regression

The second task was aimed in applying a standard Bayesian linear regression model approach.

Type-II maximum likelihood

For the first part of the task, it was focused on using Type-II maximum likelihood to estimate the most probable values for hyper-parameters to be used. By testing on a relatively large range of different hyperparameter values, by assuming a flat prior over all the potentially unknown hyper-parameters it turns the problem to a Maximum Likelihood approach. Using a combination of the different variables the ones that maximize the posterior can then be identified.

This likelihood term of using the empirical Bayes is shown by Figure 10 and can be written as:

- $\mathcal{D}$: the data pair including $\mathbf{y}$ and $\mathbf{X}$.
- $\mathbf{w}$: the key unknown parameters we want to infer.
- $\mathcal{M}$: model information.
- α: potentially unknown hyper-parameters σ_ϵ and $\sigma_{\mathbf{w}}$.

$$p(\mathcal{D}|\alpha, \mathcal{M}_i) = \int p(\mathcal{D}|\alpha, \mathbf{w}, \mathcal{M}_i) p(\mathbf{w}|\alpha, \mathcal{M}_i) d\mathbf{w}$$

Figure 10. Likelihood term with the information on the corresponding symbols used.

The two hyperparameters used where alpha which is the inverse variance of w and s2 which is the noise variance. The logarithmic ranges chosen for each hyperparameter with the number of steps are as shown by the following table 2.

Table 2. Hyperparameter's Logarithmic Range

Parameter	Min	Max	Steps
alpha	-5	30	25
s2	-22	-5	25

Then iterating through all the combination of these hyperparameter values, their log marginal probability is calculated and returned. The function for calculating these log marginal probabilities is as shown below in Figure 11.

```
def compute_log_marginal_scipy(PHI, y, alph, s2):
    N, M = PHI.shape
    C = s2 * np.eye(N) + (PHI @ PHI.T) / alph
    lgp = stats.multivariate_normal.logpdf(y.T, mean=None, cov=C, allow_singular=True)
    return lgp
```

Figure 11. Function for computing log marginal probability

The most probable hyperparameter values are the ones that maximize the log marginal likelihood, thus after iterating through all the combinations and storing their likelihood result, the maximum can be identified that showed to the corresponding combination. Out of the 625 combinations the most probable values for the hyperparameters are the ones shown in table 3 and a posterior distribution contour graph showing all the results is represented by Figure 12 with a green dot marking the most probable values.

Table 3. Most probable hyperparameter values.

Parameter	Most Probable Value (5 d.p.)
alpha	0.02897
s2	1e-05

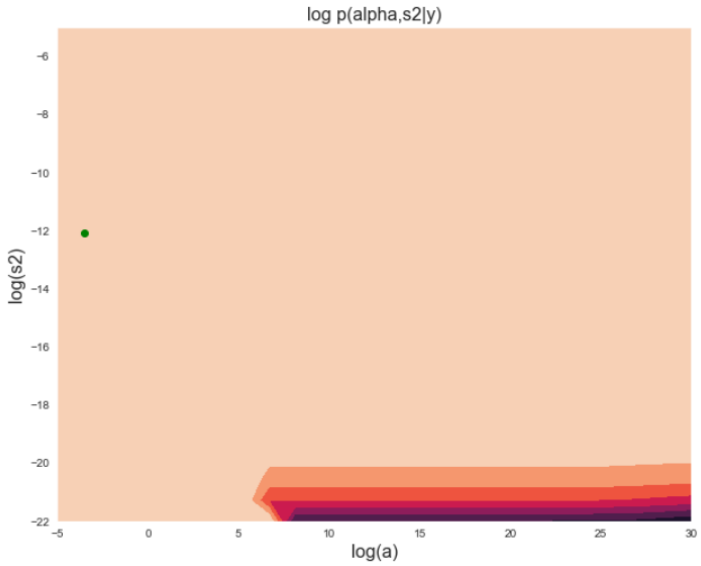

Figure 12. Posterior Distribution contour graph

Hamiltonian Monte Carlo (HMC)

Verifying HMC on a simple Gaussian Example

This section focuses on applying the Hamiltonian Monte Carlo approach on a simple bivariate gaussian example, represented by Figure 13 on a visualisation of the distribution.

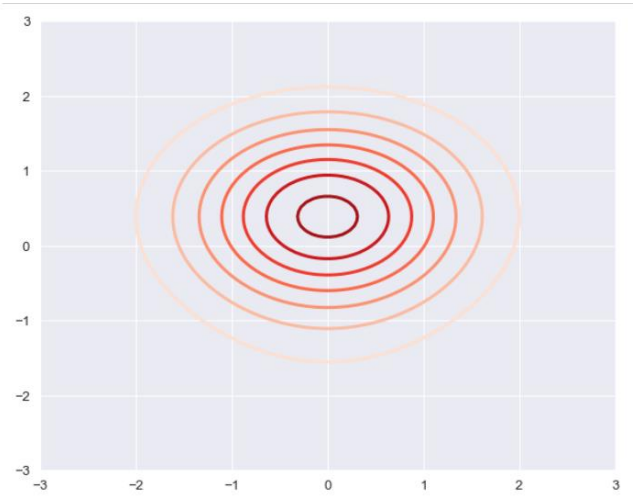

Figure 13. Bivariate Gaussian Example

The designed functions energy_func and energy_grad, are represented accordingly on the below Figures 14 and 15. The energy function provides a negative log probability of the un-normalised distribution and the gradient function contains the partial derivatives of the energy function which are in respect to x1 and x2.

```
def energy_func(x, f):
    # Simulating some unknown log-probability
    p0 = -x[0]**2/2
    p1 = -x[1]**2/2 + np.log(2+np.sin(f*x[1]))
    lgp = p0 + p1

    return -lgp
```

Figure 14. Energy Function code

```
def energy_grad(x, f):
    g = np.empty(2)
    g[0] = x[0]
    g[1] = x[1] + f*np.cos(f*x[1]) / (2+np.sin(f*x[1]))
    return g
```

Figure 15. Energy Gradient code

Passing the energy and gradient function on the sampler provided by the demo with some chosen parameter values we are provided with the visualisation represented by Figure 16 that shows the convergence performance of the sampler on the unknown parameters of the mean and covariance of the bivariate model, with an 80.0% acceptance rate.

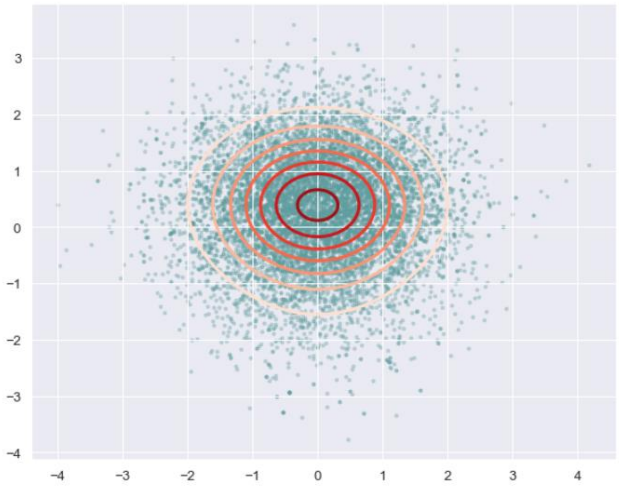

Figure 16. Bivariate Model Sampler Visualization

The parameters used to achieve the above results are represented below:

Table 4. Hyperparameter Values used for the results of Figure 16.

Parameter	Value
R	10000
L	25
epsilon0	0.03

Report Summary

To sum up, what has been learned on the provided data set under study is that the features provided for being used in identifying energy efficiency are a good representation on actual data given and tested on. The linear models performed for prediction the testing data have been proven successful and the proper identification of the relevance of the different variable was successfully recognised.

Through the observations on the effectiveness of the various methods applied in this project have been proven relatively ineffective. Some very generic representation of the applied model was shown and a lot more can be further investigated in all the approaches and methods used. Although with the observations shown the least-squares linear regression approach seemed relatively effective for properly predicting the target variable but further filtering on the most important features can be adapted to better improve its performance. Additionally, the Type-II Maximum Likelihood for identifying the probable hyperparameter values was thoroughly investigated between several range combination and steps so the values obtained should be the overall optimal ones. Lastly, applying the HMC on a simple gaussian model seemed much harder than expected but with the results obtained on the chosen hyperparameter values it showed that its application was effective.