Task 1a + b)

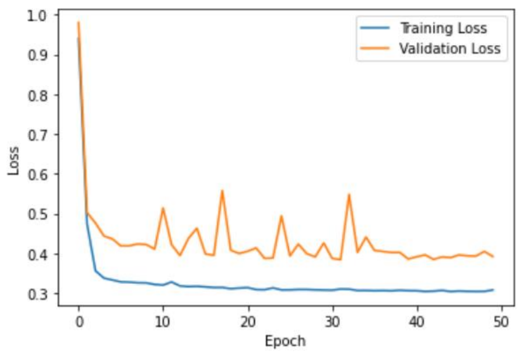


Figure 1 - Validation and training loss of neural network.

I trained a neural network to classify activity type. My model achieves a validation accuracy of 84%. The model's training loss flatlines after about 5 epochs, demonstrating quick convergence. The validation loss follows a similar trend initially. However, the validation loss is much less smooth and only really settles after about 40 epochs. The final validation loss is about 30% larger than the training loss, which indicates that the model is overfitting to the training set slightly. However, given that both training and validation loss follow similar trends and achieve similar loss, I would say the model has trained effectively.

When training the model, I noticed that the neural network will often get stuck at a local minimum of about 60% classification accuracy. Thus, it typically requires multiple training attempts with different weight initialisations to avoid this local optimum and achieve a higher classification accuracy. On the test dataset, the neural network achieves a loss of 0.319 and an accuracy of 89.9%. This would indicate that the model generalises well as this is marginally higher than both the training and validation classification accuracies.

Task 1c)

There are wide variety of model hyperparameters to tune when training a neural network. I have decided to analyse the learning rate, neural network architecture, activation function, batch size and optimiser. For each feature, I have selected a small number of values which are commonly used and cover a broad range of options. To compare them, I define a base neural network architecture and then investigate the relative performance as each variable is changed individually. Each model is trained for the same number of epochs to ensure effective comparison. For the number of training epochs, we can see from *Figure 1* that 5 epochs tend to be sufficient for a model to achieve strong classification. To select the best overall parameters, a randomised grid search would be a suitable search strategy.

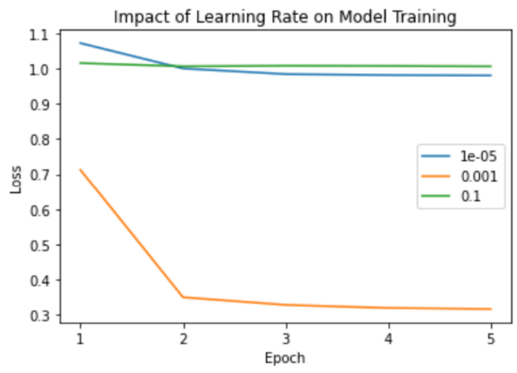
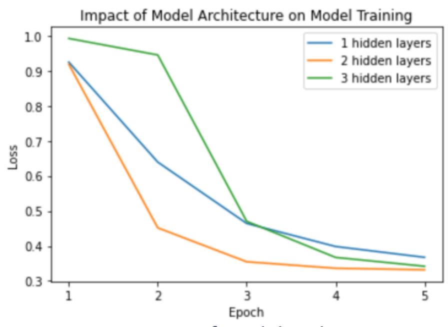


Figure 2 - Impact of learning rate on ANN performance.

Figure 2 depicts the impact learning rate has on performance. From this graph, we can see that the model gets stuck at a local optimum for learning rates of 1e-5 and 1e-1, whilst it converges for a learning rate of 1e-3. This suggests that that 1e-3 is a stronger choice of learning rate. Indeed, this is conclusion is widely adopted as this is typically the default learning rate for the ADAM optimiser [1] and is viewed to be a value that yields effective performance [2]. A greater learning rate means that step sizes are larger, leading to faster initial convergence but typically struggling to converge at the exact optimum point. Conversely, smaller step sizes converge more slowly but more accurately, even if they are more prone to getting stuck at local optima. However, in the case of ADAM, an adaptive learning rate algorithm, these problems are typically less apparent.



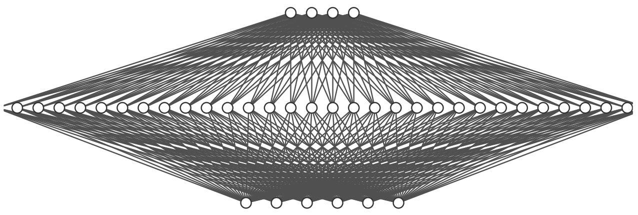


Figure 4 – Neural network architecture with one hidden layer.

Figure 2 – Impact of model architecture on training performance.

In order to assess the impact of the number of hidden layers, I created three networks with differing numbers of hidden layers. Each hidden layer has 32 neurons. It appears that larger networks converge more slowly initially but end up achieving a lower loss. This makes sense as larger networks have more capacity to model complex data relationships. However, they also have more tuneable parameters which means that they will likely learn more slowly. In the case of this instance, the difference in performance between 2 and 3 hidden layers was negligible, suggesting that both have learned the same function. Thus, in this case I would suggest that 2 hidden layers is the optimal configuration. Particularly as larger networks are more prone to overfitting and take longer to train.

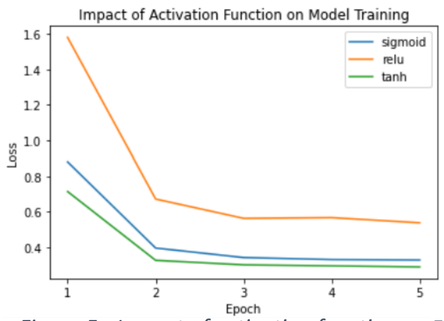


Figure 5 - Impact of activation function.

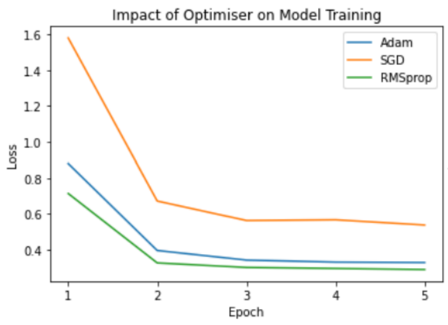


Figure 6 - Impact of model optimiser on training.

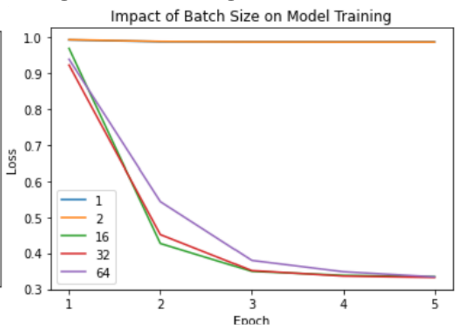


Figure 7 - Impact of batch size on training.

I experimented with three commonly used activation functions for neural networks: ReLu, hyperbolic tangent (tanh) and the sigmoid function. These functions have shown to perform well on a number of tasks. Both the sigmoid and hyperbolic tangent functions perform similarly and better than ReLu. This makes sense as the sigmoid function and tanh are fundamentally similar functions, involving exponential functions, and are commonly used for classification tasks.

I explored commonly used optimisers for training artificial neural networks: ADAM, stochastic gradient descent (SGD) and RMSprop. Both ADAM and RMSprop are adaptive learning functions where each weights value is altered by a different rate that decays over time. These approaches are considered state-of-the-art and are widely used. Stochastic gradient descent simply updates each weight by its error gradient multiplied by the learning rate. This approach performs noticeably worse than the adaptive optimisers.

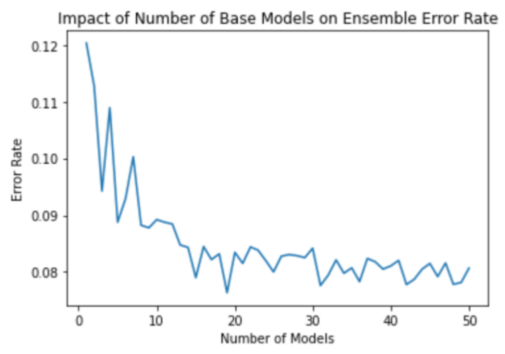
From *Figure* 7, it is clear that the model struggles to train when the batch size is too small. Batch sizes of 1 and 2 prove too small for the model to converge, whilst batch sizes greater than 16 do appear to work. This is likely because smaller batch sizes make the overall step direction too inaccurate when updating model weights.

Whilst model hyperparameters are interdependent (e.g., the optimal activation function may differ depending on the network architecture), it is apparent that some parameters are more important than others. For instance, for this task, differing network architecture seems to yield similar results. However, using an adaptive optimiser and a larger batch size do seem to improve network performance significantly.

<u>Task 2a + b)</u>

A single decision tree achieves a test accuracy of 83%. Training an ensemble of 200 trees using Adaboost achieves an accuracy of 92%, which is a significant improvement. The ensemble improves performance as each model is trained on a high proportion of the data samples that previous models classified incorrectly. This means that models address the flaws in prior models and that individual model error is related less closely. The models are also weighted according to their classification strength. Thus, the ensemble improves performance over a single tree.

<u>Task 2c + d)</u>



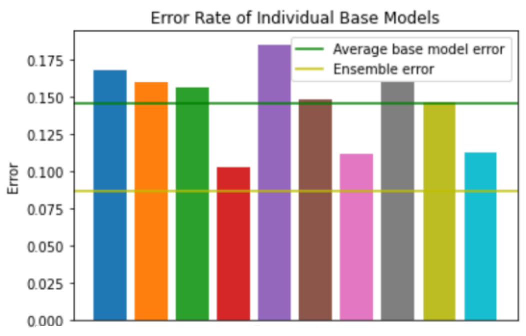
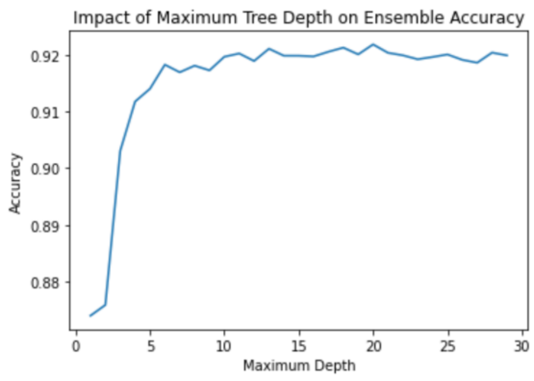


Figure 9 – Individual model errors.

Figure 8 - Error rate of ensemble with different number of models.

The ensemble error rate decreases with the number of models until it plateaus at a stable error rate. Twenty individual models appear to be the optimum amount with a low error rate and reduced inference/training time. The individual model error rate appears to be randomly distributed. The ensemble does outperform the average error of individual models. This is expected as we know from the application of Jensen's inequality that ensemble error will always be at most the average error of individual base models. The trained ensemble error is bound as follows: $\frac{1}{M}E_{AV} < E_{COM} < E_{AV}$ where M = number of models, E_{AV} = average error rate of base models and E_{COM} = ensemble error rate.



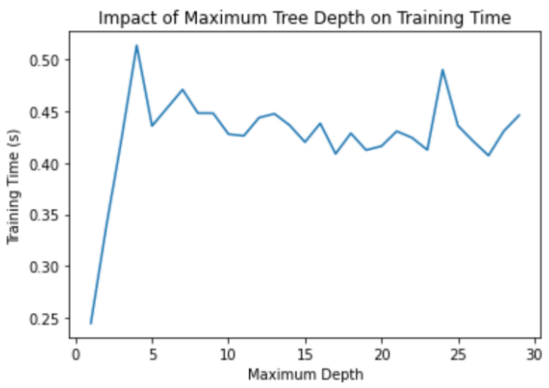


Figure 10 - Impact of tree depth on performance.

Figure 11 – Training runtime with as maximum depth varies.

The ensemble accuracy increases with maximum tree depth. This is likely because each tree is capable of making more complex splits on the data, leading to better classification. This holds till about a depth of 10. This is because whilst the trees could become deeper, they don't as the data has been split sufficiently. As the maximum tree depth increases, so does the runtime. As in *Figure 10*, *Figure 11* plateaus at a depth of around 10, indicating that model complexity stagnates at this depth. As decision trees become larger, they are capable of making more advanced (and more accurate classifications). However, they risk overfitting to the training set. Additionally, such trees quickly become hard to interpret as depth grows – negating a key advantage they have over other classification techniques. Altering the splitter and criterion hyperparameters seems to have very little impact, suggesting that the model is robust and not overly sensitive to these parameters.

<u>Task 3a + b)</u>

I trained a Gaussian Hidden Markov Model to label sequences of data. On the test set, it achieves an accuracy of 87%. That is to say that for each of the test sequences, when predicting the states given the sequence of observations using the Viterbi algorithm, 87% of the states are correctly labelled. This shows the HMM's are a strong choice for this task and can accurately predict state sequences given corresponding observation sequences.

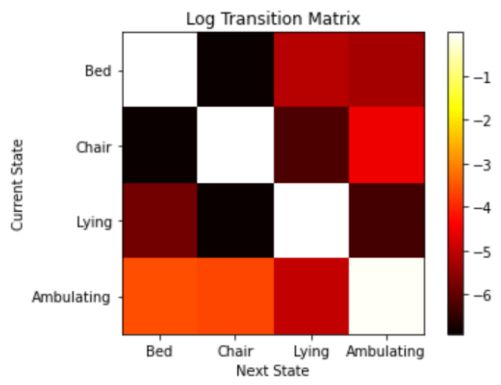


Figure 12 – Logarithm of transition matrix visualised.

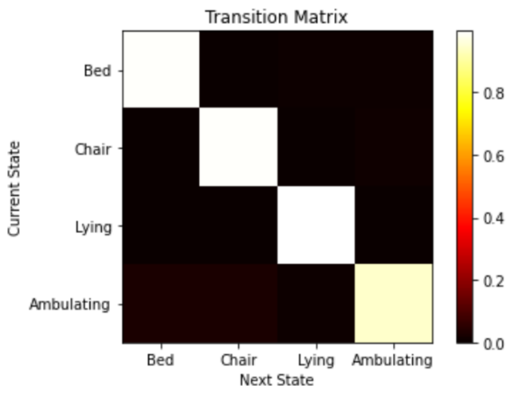


Figure 13 – Transition matrix visualised.

The transition matrix shows that people are most likely to continue doing whatever activity they are doing. This makes sense and represents people doing an activity over a period of time. Those who are ambulating are less likely to remain so compared to the other 3 states, which is likely because it is a more strenuous task. Those who are ambulating will likely sit on the bed or on the chair once they have finished but are unlikely to lie down on the bed immediately. People who are sitting on the bed are relatively likely to ambulate or lie down next but are unlikely to sit down on the chair. This makes sense as they are already sitting in a comfortable environment and would be unlikely to simply sit on a different object. Similarly, those who are lying down may sit back up on the bed or start ambulating but are unlikely to sit down on the chair immediately afterwards. Finally, those sitting on a chair may lie down or start ambulating but are unlikely to sit on the bed as their next state, as they are already sitting down.

<u>Task 3c)</u>

Analysing the distributions for the HMM model, I conclude that frequency, phase and RSSI are unimportant features. This is because they have similar mean values and high variances for their emission distributions. Lateral accuracy is also a relatively poor feature as all the labels have similar and small means for their emission distributions. Their variances are relatively high (in comparison to their means), and thus it is hard to determine which label is most likely given the observed feature.

Frontal accuracy is the most important factor as the emission means are well separated for different labels and as the corresponding variances are low. Vertical accuracy is less important, but still a good feature. It is particularly useful for identifying lying down people, as this has a mean value that differs significantly from the other three label means.

Task 3d)

Classification Accuracy	
Neural Network	Ensemble
89.9%	92.5%

Table 1 - Classification accuracy on test set of different models.

Sequence Labelling Accuracy
Hidden Markov Model
86.8%

Table 2 – HMM performance.

Tables 1 and 2 summarises the performances of the three models. The ensemble classifier achieves the best performance on the test set, followed by the neural network. Both achieve a strong classification accuracy of around 90%. The HMM classifies 87% of labels correctly when given a sequence of observations. Given that the HMM requires a sequence of observations to make accurate label predictions but still performs marginally worse than the classification techniques (which only require a single observation), I conclude that the HMM has the worst performance of the three models. Conversely, the ensemble technique performs best.

Model Type	Training Time(s)	Relative Speed
Neural Network	159.75	1x
Ensemble	0.52	307x
Hidden Markov Model	0.13	1229x

Table 3 – Model training time.

In terms of training time, the HMM model is the fastest. This is because training involves a fixed series of basic calculations. The ensemble is 4 times slower to train than the HMM, but still very fast with less than a second of training time. The neural network is the slowest model to train, taking over 2 minutes to train. Both the neural network and ensemble have variable training times depending on model hyperparameters (e.g., network architecture or number of base models), whilst the HMM training time is only dependent on the size of training data. Additionally, with the neural network, given its tendency to get stuck at a local minimum for this specific task, the actual training time is even longer. As a result, training a neural network is significantly more arduous for this problem.

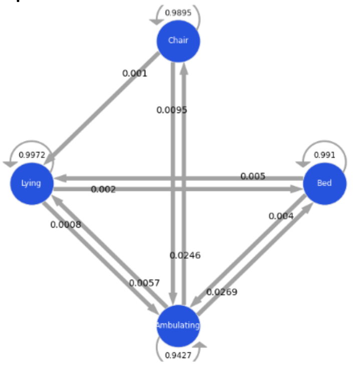


Figure 14 – Transition matrix visualisation.

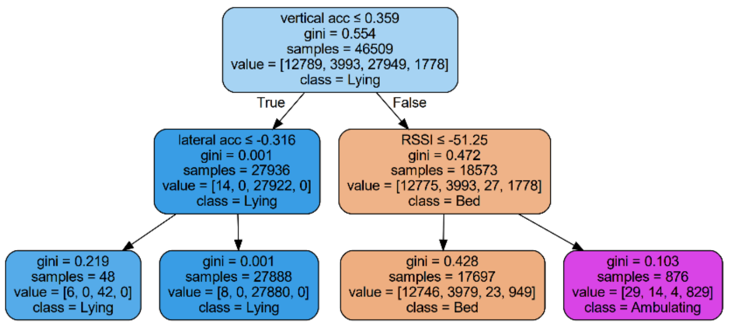


Figure 15 – Decision tree visualised.

The HMM benefits from being easy to interpret, with its transition matrix providing a strong intuition of modelling. Similarly, a decision tree is easy to interpret and can be visualised quite effectively. As the maximum depth of the tree increases, this interpretability is lost somewhat. And within the context of an ensemble model that has 200 trees, the model does prove to be quite difficult to interpret. A neural network is very hard to interpret as it contains lots of weights and biases that work together in an extremely complicated way to produce a classification (see *Figure 4*). Thus, within the context of this specific task, the HMM is the easiest to interpret. As the data is fundamentally sequentially, one could argue that the HMM is more suitable for modelling the task than the other two approaches.

Each approach has various merits, however I would say that the ensemble approach is the most suitable for this task as it is quick to train, achieves a strong classification accuracy and is somewhat interpretable.

<u>Task 4 + 5 + 6)</u>

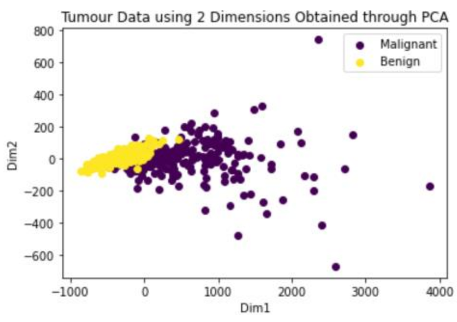


Figure 16 - PCA results.

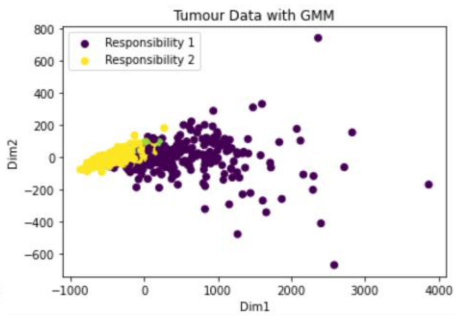


Figure 17 – Plot of responsibility composition for each data point for the GMM.

Figure 16 displays the results of using PCA to reduce the dataset dimensionality to 2. The first principal component accounts for 98.2% of the original data variance and 1.6% of the original data variance is explained by the second principal component, accounting for over 99.8% of the original variance in combination. *Figure 17* depicts the results of a Gaussian Mixture Model that has two components. The yellowness of a point indicates the responsibility strength from the first distribution. The purpleness of a point indicates the responsibility strength from the second gaussian distribution.

At a first glance, both of these scatter plots appear very similar. This demonstrates the strength of the GMM. We can intuitively think of the first responsibility as denoting the likelihood that the tumour is malignant and the second responsibility as denoting the likelihood of a benign tumour. If we use this to classify data, the model classifies 95% of the data correctly (though of course this essentially amounts to evaluation on the training set). The GMM is most uncertain at regions where the two distributions meet in *Figure 17*, with points being green (showing that the point is drawn from a mixture of the two distributions). Conversely, *Figure 16* doesn't have this colour mixture as data items are either malignant or benign (binary).

<u>Task 7 + 8 + 9)</u>

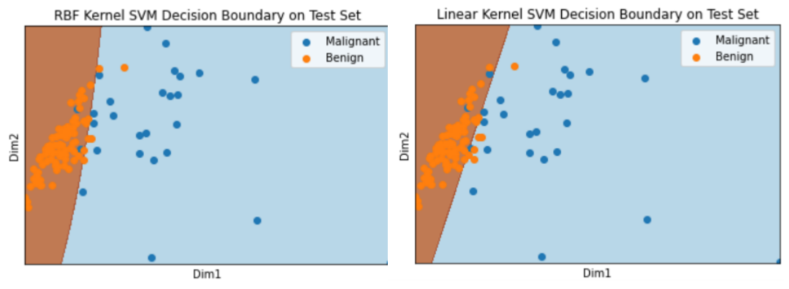


Figure 18 - Decision boundary of linear and RBF kernel.

I trained an SVM on the 30-feature dataset, achieving a classification accuracy of 95.8% on the test dataset. I utilised grid search with cross-validation to determine which kernel and regularisation parameter performs best on the training set. This suggested using a linear kernel, however I found that an RBF kernel performed slightly better on the test set, so I used this. Additionally, when visualising the SVM decision boundaries, I found that the RBF kernel produced a smoother decision boundary.

I also trained an RBF kernel on the PCA reduced 2-feature dataset, achieving a 95.0% test accuracy. This suggests that the PCA reduction is extremely effectively and has a negligible impact on test set accuracy.

Task 10)

In order to convert the *Seoul Bike Sharing Demand* dataset to a format suitable for linear regression, I had to convert all features to numerical values. This largely involved encoding categorical data to numerical values. For instance, I encoded days of the year to numbers ranging from 1 to 365. I decided to remove all records where the day was not a functioning day. This is because on non-functioning days, the demand is recorded to be 0. This is misleading as the demand is likely not actually zero, it's just that no bikes could be rented as the bike sharing system was not functioning.

Additionally, I removed the date feature. Whilst this is a useful variable when predicting bike demand, it is hard to use it in a linear regression model as there is unlikely to be a simple linear relationship between day number and bike demand. The true relationship between the two is likely to be much more complicated (e.g., it may be that weekdays have greater demand, and that each month has varying demands etc.). Additionally, most of the important information from this variable can be represented by a combination of other features such as weather and whether it is a holiday. I considered removing the hour variable as a similar argument can be applied about there being a non-linear relationship between hour number and bike demand. However, I decided to keep this in the dataset as I predict it to be the most important feature when making a prediction. For instance, bike demand at 8 in the morning during rush hour is likely to be much higher than at 3 in the morning.

I also removed seasonal information as I felt that meteorological data was likely to provide more insight. For instance, temperature in combination with hour of the day is likely a very strong indicator of season. Hence, I concluded that the seasonal information was likely redundant.

Task 11)

I model the bikes sold linear regression system as follows:

$y \sim N(\mu, \sigma^2 I)$ where $\sigma \sim |N(0,\ 20^2)|$ and $\mu = WX^T + c$ where $c \sim N(0, 20^2)$

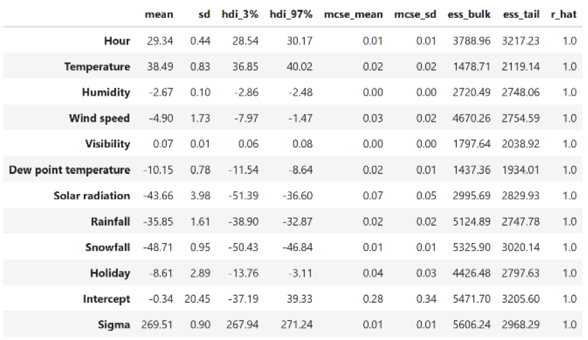
c is the y intercept for the linear regression model. W is a 1 x d matrix storing the regression coefficients (d = number of features in dataset). X is a n x d matrix (n = number of data points) containing the data points. Each regression coefficient in W has a prior distribution associated with it, which is outlined in *Table 4*.

Table 4 - Prior distributions for Bayesian linear regression.

Variable Coefficient	**Prior Distribution**
Hour of Day	$N(0, 20^2)$
Temperature	$N(10, 2^2)$
Humidity	$N(-4, 20^2)$
Wind Speed	$N(-10, 2^2)$
Visibility	$N(5, 10^2)$
Dew Point Temperature	$N(-4, 20^2)$
Solar Radiation	$N(10, 10^2)$
Rainfall	$N(-20, 2^2)$
Snowfall	$N(-50, 1^2)$
Holiday	$N(-1, 3^2)$

For hour of the day, I selected a prior with a high variance and a mean of zero. This reflects my uncertainty for how the hour of the day relates to the bike demand linearly. For temperature, I chose a large positive mean and a small variance. This is because I'm relatively confident that the hotter the day, the greater the demand there will be for bikes. For humidity, I suspect that more humidity will discourage cycling as physical exertion will be harder. However, I am not entirely sure of this as humidity may be related to temperature, and I have hence chosen a larger prior variance. I hypothesise that windy days will discourage cycling and I have chosen a prior that reflects this. Conversely, I believe that days with strong visibility are more likely to increase demand in biking, even if I think that the relative effect visibility has is perhaps smaller than something like temperature. I have selected my dew point temperature prior by following a similar line of reasoning to the one I used to determine my humidity prior. I believe that solar radiation is positively correlated to cycling demand as cycling on a sunny day is typically more pleasant. I believe that both rainfall and snowfall discourage cycling, with snowfall being likely to reduce the cycling demand to very low numbers as roads are likely to be unusable or dangerous. I believe that holiday days will have little impact on cycling demand as fewer people will be cycling to work but more people may be cycling for leisure. However, holiday days will have a slightly reduced bicycle demand as I think that more people probably cycle to work than would look to hire a bike for pleasure.

Task 12 + 13 + 14)



	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
Hour	29.34	0.44	28.54	30.17	0.01	0.01	3788.96	3217.23	1.0
Temperature	38.49	0.83	36.85	40.02	0.02	0.02	1478.71	2119.14	1.0
Humidity	-2.67	0.10	-2.86	-2.48	0.00	0.00	2720.49	2748.06	1.0
Wind speed	-4.90	1.73	-7.97	-1.47	0.03	0.02	4670.26	2754.59	1.0
Visibility	0.07	0.01	0.06	0.08	0.00	0.00	1797.64	2038.92	1.0
Dew point temperature	-10.15	0.78	-11.54	-8.64	0.02	0.01	1437.36	1934.01	1.0
Solar radiation	-43.66	3.98	-51.39	-36.60	0.07	0.05	2995.69	2829.93	1.0
Rainfall	-35.85	1.61	-38.90	-32.87	0.02	0.02	5124.89	2747.78	1.0
Snowfall	-48.71	0.95	-50.43	-46.84	0.01	0.01	5325.90	3020.14	1.0
Holiday	-8.61	2.89	-13.76	-3.11	0.04	0.03	4426.48	2797.63	1.0
Intercept	-0.34	20.45	-37.19	39.33	0.28	0.34	5471.70	3205.60	1.0
Sigma	269.51	0.90	267.94	271.24	0.01	0.01	5606.24	2968.29	1.0

Figure 19 - Posterior distribution summary.

All parameters achieve an $\hat{R}$ value of 1.0, which suggests that the model has converged and that the posteriors may be close to the true distributions. This means that the 4 independent chains used for sampling converged on the same posterior distribution. Additionally, inspecting posterior distribution plots, the values seem broadly reasonable.

Most of the values align with my assumptions made in the prior distributions, although the relative importance of certain factors does differ. For example, rainfall is even more of a deterring factor for bicycle hire than I had suspected. Visibility appears to factor in very little to the likelihood of somebody hiring a bike, which is a somewhat surprising result. I hypothesise that this is because people are only deterred from cycling for visibility reasons in extreme low visibility settings like fog, which occur rarely.

The most surprising result is that more solar radiation leads to less demand for cycling. My assumption was that sunny days would likely be more pleasant to cycle in, however this is clearly not entirely true. This is even more surprising when considered in conjunction with the fact that hotter days do lead to an increase in demand for cycling. It is possible that South Korea has a warm climate during summer and thus people may feel more inclined to cycle on hot days that are cloudy as the solar radiation is less. I was also surprised to see that the hour of the day has such a strong impact on the model and that people seem to hire more bikes later in the day. I suspect that the actual relationship between hour of the day and bikes hired is much more complicated (perhaps following a multimodal normal distribution or an even more complex function), however it would seem that, on average, more bikes are rented later in the day.

Task 15

Training a linear regression model on the dataset achieves an R^2 score of 0.47, suggesting that less than half of the data variance can be explained by a linear regression model. This implies that linear regression models are not suitable for this dataset. A more advanced, non-linear model would be more suitable to model the data. For instance, fitting a cubic regression model achieves an R^2 score of 0.65.

References

[1] *Adam*. Keras. https://keras.io/api/optimizers/adam/ (accessed on 01/12/2023).

[2] *ADAM: A Method for Stochastic Optimization*. Kingma and Ba. *22nd December 2014.* https://arxiv.org/pdf/1412.6980.pdf (accessed on 01/12/2023).