Classification and Sequence Labelling

Task 1: Neural Networks

A multilayered perceptron (MLP) is a type of neural network composed of an input layer, one or more hidden layers, and an output layer. Neurons in each layer are connected to neurons in the adjacent layers with weights that are adjusted during training. The input layer receives data, which is then processed through the hidden layers using activation functions, and the final output is generated. MLPs are capable of learning complex relationships in data and are commonly used for tasks such as classification and regression in machine learning.

Figure 1: ***MLP classifier loss and validation***

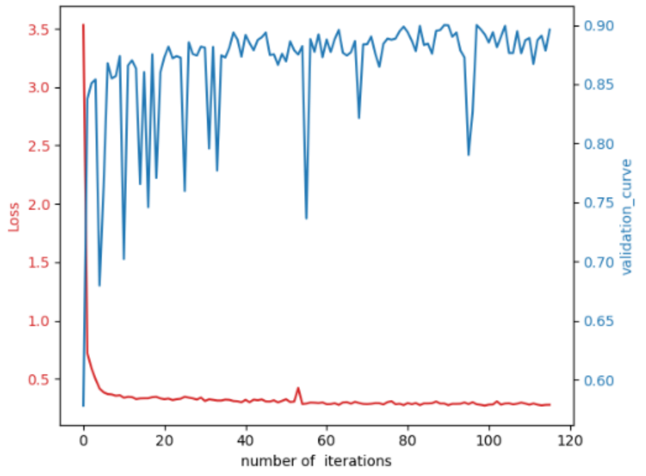
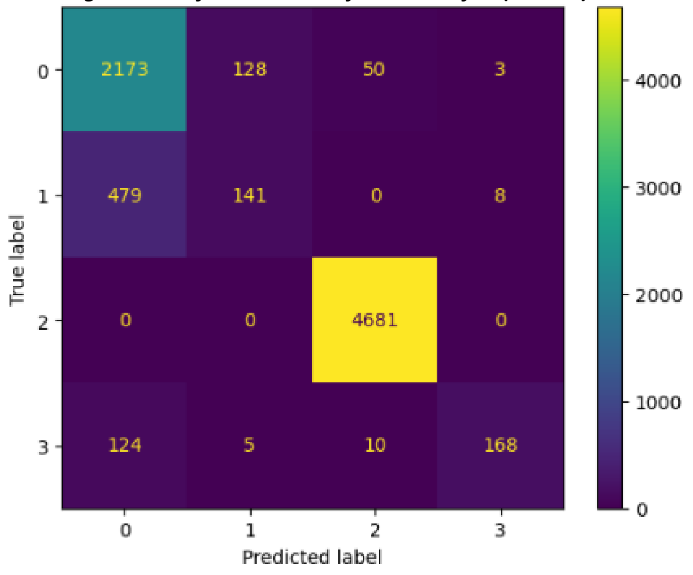


Figure 1 depicts the training loss of a multi-layered perceptron (MLP) in red and the validation score at each iteration in blue. The loss drops, then plateaus at around 7 iterations and only slightly decreases during further iterations. Whilst the validation score tends towards 90% accuracy, the rate of increase slows. As the loss decreases the validation scores increase at each iteration, this indicating successful learning. After training, the accuracy of the MLP classifier on the training set was 90.29% and the accuracy on the testing set was 89.87%. This indicates that the MLP classifier was able to generalise and classify unseen data with roughly 90% accuracy.

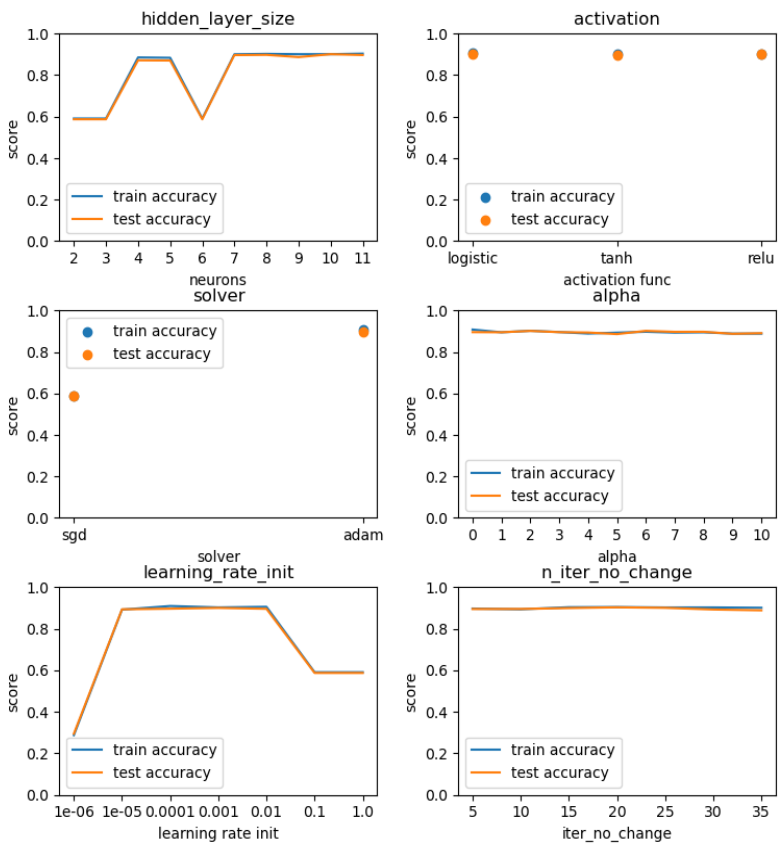
Figure 2: Confusion matrix of MLP classifier (test set)



True label \ Predicted label	0	1	2	3
0	2173	128	50	3
1	479	141	0	8
2	0	0	4681	0
3	124	5	10	168

From Figure 2, f1-scores are calculated for each class. Due to the imbalance of classes, accuracy may not be an accurate measure of the model's performance at classification as the distribution of each class isn't considered. Class 0, 1, 2 and 3 have f1-scores of 0.85, 0.31, 0.99 and 0.69 respectively. This indicates that the model is good at classifying classes 0 and 2 but poor at classifying class 1, this can be observed in the confusion matrix. This may be due to the underrepresentation of class 1 in the training set, leading to poor performance when classifying new data of that class.

Figure 3: MLP classifier scores against varying hyperparameters.



To determine which hyperparameters have the strongest effect on model parameters, the 'One At a Time' (OAT) method was deployed. At each attempt, a hyperparameter was set at varying values and a model was trained and scored on the training and testing sets, the other hyperparameters were kept at default settings provided by SKlearn. As seen in Figure 3, the solver, hidden layer size and initial learning rate had the strongest effect on model performance. Adam consistently produced a better accuracy score on both the testing and training sets than SGD. This may be due to Adam having an adaptive learning rate whereas SGD does not. Adam is also regarded as more robust to choices of hyperparameters [1]. The initial learning rates also have a consistent strong effect on model accuracy with the best range of learning rates being 0.00001 to 0.01. Anything outside of this range negatively affects the model's accuracy. The hidden layer size has a less strong effect as the accuracy scores for each value fluctuate over multiple iterations but after hidden layer size 8, it consistently produced high accuracy scores. The choices of activation function, alpha value (L2 regularisation strength) and 'iterations of no change before stopping' had no observable effect on the model's score when varied, hence, do not provide any strong effect on model performance with this dataset. Batch size was also tested at varying values from 50 to 250 in intervals of 5 but did not produce any observable effect on the model's accuracy. After running a randomised search of hyperparameters with cross-validation, the optimal parameters found were, solver Adam, learning rate 0.00029, hidden-layer size 26 and alpha of 0.07.

Task 2: Ensemble Methods (Random Forest)

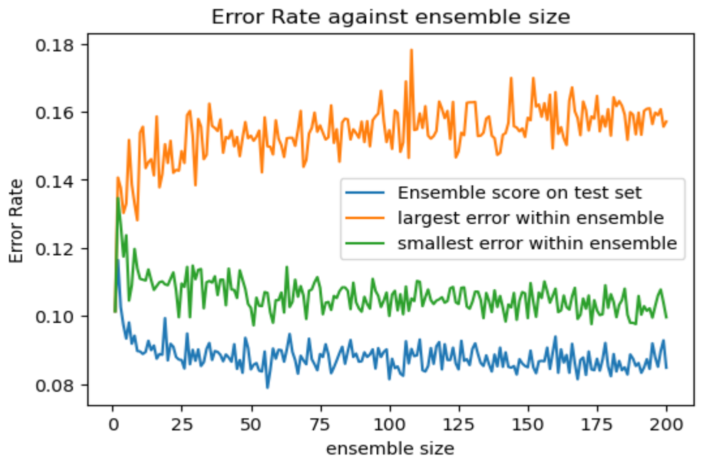
A random forest is an ensemble learning method that combines multiple decision trees to improve predictive accuracy and control overfitting. Each model in the ensemble is trained on a subset of the data with randomly selected features. During prediction, each model votes on which class is most likely for each data point, and then the class with the majority vote is assigned to that data point. Random Forests use the 'wisdom of the crowd' theory applied to decision tree models. Individual decision trees are prone to overfitting, using a Random Forest prevents the risk of overfitting and is more robust, generalising better on new data. Random forest limits the number of features considered by each decision tree, hence, increasing the diversity of models and preventing errors from being correlated.

Random Forest	Test Set	Training Set
Accuracy	0.910	0.952

The Random Forest ensemble demonstrated strong performance on the training set and testing set indicating good generalisation with no clear indications of underfitting or overfitting.

As depicted in Figure 4, increasing ensemble size exhibits a positive impact on accuracy scores in classification. Notably, the error rates of ensemble scores stabilise around 30 base models, reaching an error of 0.09. Concurrently, the increase in ensemble size extends the upper limit of individual model error rates, plateauing at approximately 0.16. Conversely, it reduces the lower limit of individual model error rates plateauing at approximately 0.1. These observations make evident the positive influence of increasing ensemble size on classification accuracy.

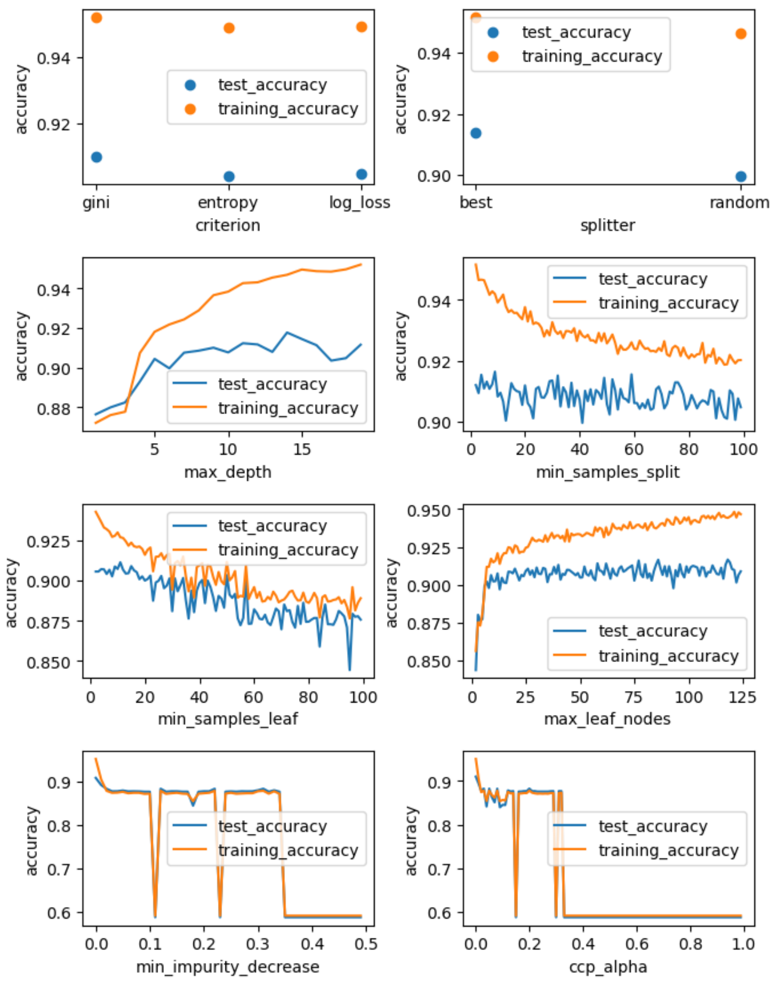
Figure 4: Random Forest error against ensemble



Hyperparameter sensitivity: Random Forest

As seen in Figure 5, the hyperparameters that the Random Forest is most sensitive to are as follows:

Figure 5: Varying Random Forest hyperparameters against score.



- **min_impurity_decrease**. This is the threshold of impurity decrease required to induce a split. The model is highly sensitive to this showing a sudden large decrease in accuracy at the threshold of 0.35, with accuracy decreasing from 0.89 to 0.6.
- **ccp_alpha**. This is the complexity parameter for minimal cost-complexity pruning. The model is highly sensitive to this parameter, exhibiting a sudden decrease in accuracy at 0.3 ccp_alpha, dropping from 0.89 to 0.6.
- **max_depth**. This determines the maximum depth of decision trees in the ensemble. The ensemble is moderately to highly sensitive to this parameter. Gradual changes in maximum depth cause gradual increases in accuracy. There is a fairly steep increase at maximum depths of 3 to 4.
- **max_leaf_nodes**. This is the maximum number of leaves trees can have in an ensemble. The model is moderately to highly sensitive to this parameter. From values of 0 to 10, there is a large increase in accuracy from 0.84 to 0.91 in the testing set and 0.925 in the training set. From there the increase plateaus in the testing set whilst the accuracy in the training set continues to gradually increase.
- **min_samples_leaf**. This is the minimum number of samples required in each leaf. The ensemble exhibits moderate sensitivity to this hyperparameter. As the minimum samples required in each leaf increase the accuracy of the ensemble gradually decreases.
- **min_samples_split**. This is the minimum number of samples required for an internal node to induce splitting. The model shows low sensitivity to this hyperparameter.

The testing accuracy doesn't vary much, while the training accuracy decreases very gradually from 0.945 to 0.92 when varying the minimum samples required from 0 to 100.

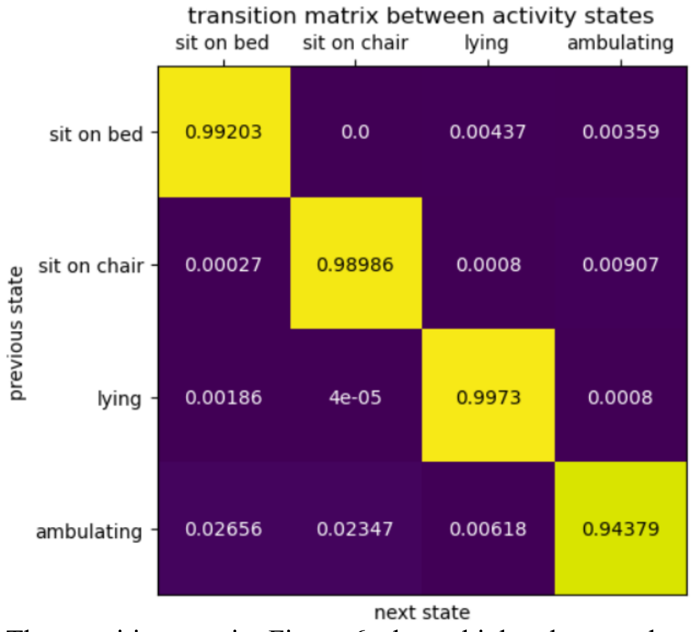
- **Criterion**. This is the function which measures the quality of splits. Between the options of criterion gini had a slight advantage in accuracy score. The ensemble is only weakly sensitive to this hyperparameter.
- **Splitter**. This is the strategy of choosing how to split each node. "Best" is slightly better than "random" in terms of accuracy score in both testing and training sets. The model was very weakly sensitive to this hyperparameter.
- **Task 3: Sequence Labelling**

A Gaussian Hidden Markov Model (GHMM) is a statistical model used for time-series data analysis, particularly in speech and pattern recognition. The model includes transition probabilities between hidden states, emission probabilities for observations given a state, and an initial state distribution. A GHMM was trained on an activity recognition dataset and the following results were produced.

Gaussian HMM	Training Sequence	Test Sequence
Accuracy	0.908	0.88

The high training accuracy of 0.908 indicates that the model has effectively learned the patterns in the training sequences. The testing accuracy of 88% reflects the model's ability to generalise to new unseen sequences. Whilst this is a strong performance, the slight drop compared to the training accuracy may indicate slight overfitting and suggests that the model may struggle to generalise to patterns beyond the ones present in this dataset.

Figure 6: transition matrix



The transition matrix, Figure 6, shows high values on the diagonals. This indicates that the system often stays in the same state of activity, whilst the off-diagonal elements in the matrix show a lower probability of transitioning between different states of activity. Ambulating has a slightly lower probability of remaining in the same state at 94%, contrary to the roughly 99% probability of the other states remaining in the same state. From ambulating there is a 2.6% chance of transitioning to sitting on the bed and a 2.3% chance of transitioning to sitting on a chair.

Figure 7: Means of features in each state

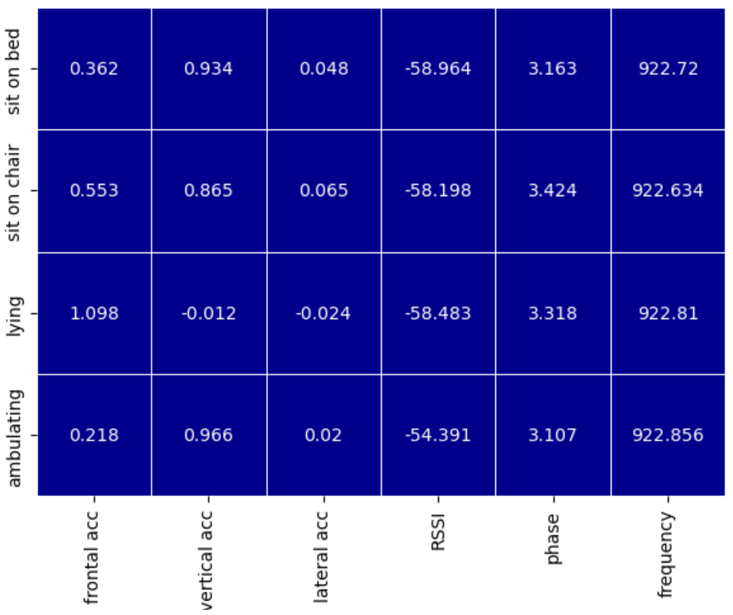
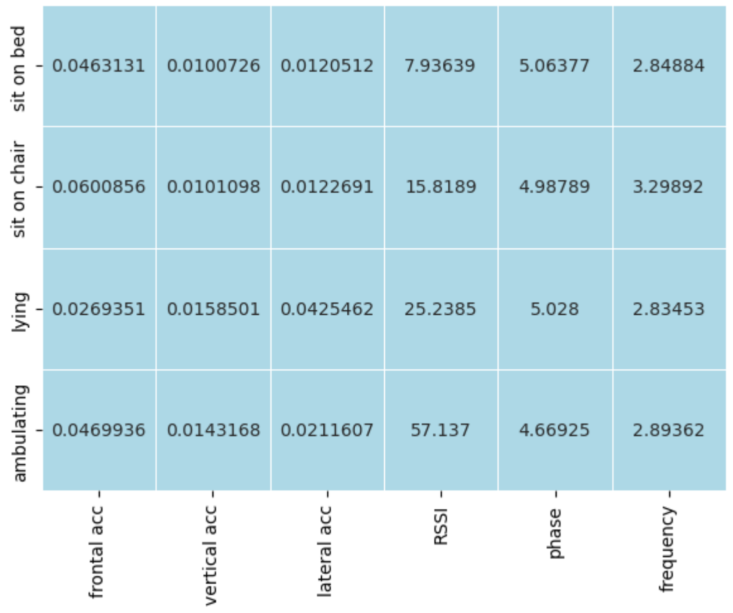


Figure 8: Variance of features in within each



Informative features

Figure 7 depicts the means of each feature in each state and Figure 8 depicts the variances of features within each state. The features that are the most informative are frontal and vertical acceleration. Frontal accuracy is the most informative as the variance within classes is very low approximately 0.04 and the means between classes are different enough to distinguish between all states.

Vertical acceleration is also informative as there is very low variance within each state for each feature and the means are different enough to distinguish activity states.

Lateral accuracy is less informative. The variance between states is low however the difference between means is smaller compared to frontal and vertical accuracy making distinguishing between activity states more difficult in comparison.

RSSI, Phase and Frequency are the least informative features within this data set. The observed variances of these features across different activity state is sufficiently large such that the differences between means of these features in each activity state is not enough to distinguish between activity states.

Comparing Multi-layered Perceptrons, Random Forests, Gaussian hidden Markov model sequence labeller

Averaging 20 training times on default settings and the same training data, the Random Forest ensemble's training time was consistently the fastest followed by the Gaussian Hidden Markov Model (Gaussian HMM) and MLP. The MLP was by far the slowest in training by over 8 seconds. As seen in the table below.

	MLP	Random Forest	Gaussian HMM
Training Time (secs)	8.4717	0.0746	0.0936

In terms of average model performance, Random Forest was the best, followed by Gaussian HMM and MLP. MLP was the worst in terms of model performance at 0.88 accuracy in classification. However, both random forest and Gaussian HMM show slight signs of overfitting to data as the training set accuracies are better than the testing set accuracies. As seen in the table below. MLP's training and testing accuracies were roughly the same.

	Model Accuracies	
	Training Set	Testing Set
MLP	0.88	0.87
Random Forest	0.95	0.91
Gaussian HMM	0.91	0.88

In terms of interpretability, the sequential labeller (Gaussian HMM) is good for interpretability as it provides a transition matrix allowing you to visualise how likely states are to follow each other. The transition probabilities and states make it easier to interpret the likelihood of a sequence of states.

Random Forests are slightly less interpretable in comparison to Gaussian HMM or singular decision trees. Random Forests are an aggregation of multiple decision trees making it difficult to visualise, however, you can still gain insights as you can extract 'feature importance' from Random Forests and can plot proximity plots to see how close data points are from the perspective of a Random Forest.

Multi-layered perceptrons are not very interpretable. Despite successfully solving many problems, they are generally used as "black boxes" [2]. While you can extract some insights, explaining individual decisions, weights and bias may not have a clear intuitive reason.

Clustering and dimensionality reduction
Task 4: Principle Component Analysis (PCA)

PCA is a method of dimensionality reduction, it looks for projections with maximum variance which are defined by principle component vectors $\mathbf{u}_i$.

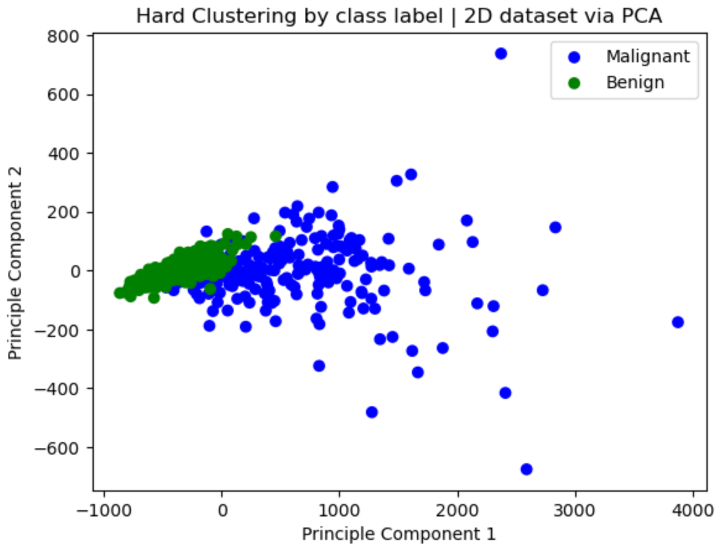
(Eq. 1) $\mathbf{u}_1^T \mathbf{S} \mathbf{u}_1 = \lambda_1$ *where S is the covariance matrix of data*

The principle component $\mathbf{u}_i$ which produces the largest eigenvalue λ_i explains the largest variance and is computed using Eq.1, the following principle components are orthogonal and together define a new coordinate system. PCA also helps minimise the curse of dimensionality. The curse of dimensionality is where data of higher dimensions need exponentially more data to be represented, this can lead to sparsity of data and increased computational demands thus making it difficult to produce reliable predictions or inferecnes.

PCA was used to reduce a data set of 30 features down to 2 features and plotted as seen in Figure 10.

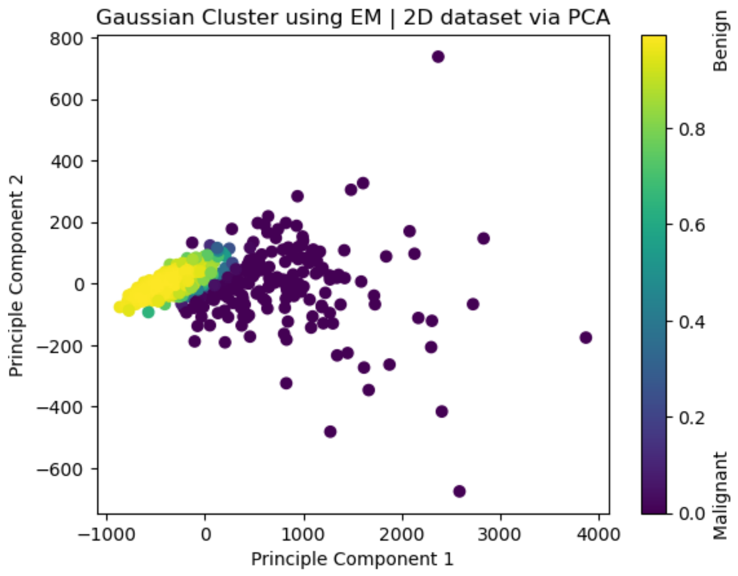
Principle component 1 contributed 98% of the variance. Principle component 2 contributed 1.6% of the variance.

Figure 9: hard clustering on reduced dataset



Task 5: Gaussian Mixture model soft clustering

Figure 10: Gaussian mixture model | coloured by responsibility



A Gaussian mixture model assumes data is generated from multiple Gaussian distributions, each characterised by its mean and covariance. The model is comprised of a weighted sum of its Gaussian components, defined in Eq.2.

(Eq.2) $$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)$$

A Gaussian mixture model (GMM) was fitted to the 2 dimension data produced via PCA. The process of fitting a Gaussian mixture model involves the EM algorithm algorithm which is an iterative algorithm that uses an "Expectation" step to compute the responsibilities each Gaussian component has over a data point and a "Maximisation" step which uses the responsibilities to compute the parameters of each Gaussian component. These steps are repeated until convergence. The EM algorithm is only guaranteed to find local maxima.

Task 6: Differences between Figure 9 and Figure 10

Figure 9 shows 2 distinct colours while Figure 10 shows a range of colours. This is due to Figure 9 using hard clustering in which each data point is assigned the label of either 'Benign' or 'Malignant' which corresponds to the colours in the plot. Figure 10's range of colours shows soft clustering, in which each data point is assigned a 'responsibility' predicted by the fitted GMM. The responsibility is a probability representing how likely a data point is to be from each Gaussian component.

Soft clustering may be more appropriate than hard clustering in this case as some data points may exhibit properties that are ambiguous and can be in either class.

Classification with Support Vector Machines
Task 7: SVM approach

Support Vector Machines (SVM) are a supervised machine learning algorithm designed for classification and regression. Its core purpose is to identify a hyperplane that maximises a margin and best separates different classes in a high-dimensional space.

The regularisation parameter C of the SVM determines the trade-off between margin size and misclassifications. A large C value will lead to opting for a smaller margin but better classifications. A smaller C value will opt for a larger margin even if there are more miss classifications. In the context of an SVM, a kernel is a function that computes the similarity between pairs of data points in higher dimensional spaces.

By using an exhaustive search on the types of kernel and regularisation parameters of an SVM, it was found that a regularisation value of 4 and the Poly kernel produced the best accuracy of 97% on the testing set after being fitted on a training set of 30 features. The parameter combinations used are as follows:

Kernels: Linear, Poly, RBF, Sigmoid
C: 1.0, 2.0, 3.0, 4.0, 5.0

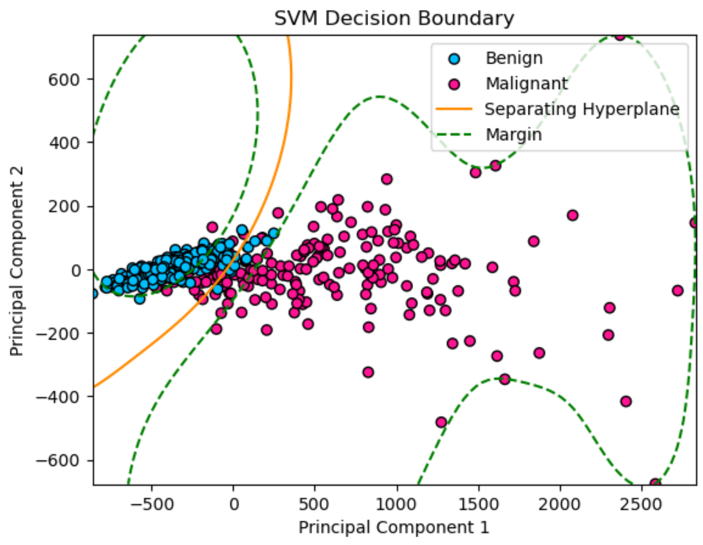
Task 8: SVM approach on 2D data set

Repeating the same process as above for the 2-dimensional data set produced by PCA, the best parameters were the RBF kernel and a regularisation parameter value of 1. This produced an accuracy of 95%, the decision boundary drawn by the SVM with the RBF kernel can be seen in Figure 11.

Task 9: Effect of the 2D approximation on accuracy

The 2D approximation via PCA did reduce the accuracy of SVMs.Without tuning new optimal parameters for the 2D approximation, the accuracy drops from 97% to 92% with C as 4 and the kernel as Poly. After re-adjusting the parameters for the 2D approximation of data, the accuracy rose to 95% with C as 1 and the RBF kernel. In conclusion, the accuracy is still good, however, some information was lost when removing 28 features of the data resulting in a

Figure 11: SVM decision boundary for 2D dataset



slightly worse performance in classification.

Bayesian Linear Regression
Task 10: Preprocessing Justifications

Bayesian Linear Regression (BLR) is a statistical modelling technique that extends linear regression by incorporating Bayesian principles. The regression coefficients are treated as probabilities. The goal of BLR is to generate a posterior distribution over these parameters. This model combines prior information about the parameters with likelihood information from the observed data to produce this posterior distribution. Mathematically, the model can be expressed as the following equation:

(Eq. 3) $P(\theta|D = d) \propto P(\theta)P(D = d|\theta)$

BLR uses Monte Chain Monte Carlo (MCMC) as usually $P(\theta|D = d)$ is not easy to represent or even compute. MCMC is used to sample from complex distributions, especially when direct sampling from these distributions is impractical. Samples are repeatedly drawn from a sequence of distributions which eventually reaches the desired distribution. In the sequence of distributions, an acceptance probability is used to determine whether to transition onto the new proposed state by Markov Chain, this eventually leads to the sequence of distributions converging onto the desired distribution, from which samples can be drawn as if sampling from $P(\theta|D = d)$.

The Metropolis-Hastings algorithm is a type of MCMC. In each iteration, a proposal state is generated. Accepting this new state is determined by the acceptance probability equation:

(Eq. 4) $$A(\mathbf{z}^*, \mathbf{z}^{(\tau)}) = \min\left(1, \frac{p(\mathbf{z}^*)q(\mathbf{z}^{(\tau)}|\mathbf{z}^*)}{p(\mathbf{z}^{(\tau)})q(\mathbf{z}^*|\mathbf{z}^{(\tau)})}\right)$$

Where $\mathbf{z}^*$ and $\mathbf{z}^{(\tau)}$ are the proposed state and previous state. 'p(z)' is the target distribution and 'q($\mathbf{z}^*|\mathbf{z}^{(\tau)}$)' is the proposal distribution.

These methods were improved upon by the NUTS sampler which is used in the 'pymc' library from which the BLR model was generated [4]. NUTS uses the same principles as Metropolis hastings but is more efficient.

Data modifications
Date:
To make the data suitable for linear regression, some preprocessing must be applied to the data. The 'Date' feature was kept and ordinally encoded. This was to retain any information not captured by the 'seasons', 'holiday' or 'functioning day' features. For example, over time people in Seoul may prefer electric scooters, therefore the possible overall downward trend of renting bikes over time may not be captured in the other features in the dataset. The 'Seasons' feature will most likely capture the cyclical nature of renting bikes throughout the year.

Seasons:
Seasons were ordinally encoded, from 0 to 3 corresponding to winter through to spring and then cyclically encoded into 2 features 'Seasons_sin' and 'Seasons_cos' as opposed to one-hot encoded to reduce the risk of the curse of dimensionality. The cyclical encoding applied was $\sin(\pi * \frac{x}{2})$, $\cos(\pi * \frac{x}{2})$ to preserve any cyclical patterns not encoded by the Date feature which has been ordinally encoded. Cyclical encoding can be thought of as mapping seasons onto a unit circle using the sin and cos functions. From there, sin and cos can determine which quadrant of the circle, in other words, what season.

Hour:
A cyclical encoding was also applied to hours to show the cyclical nature of hours. After applying this transformation hour will be split into 2 features Hour_sin and Hour_cos. The cyclical encoding applied was $\sin(\pi * \frac{x}{12})$ and $\cos(\pi * \frac{x}{12})$.

Holiday and Functioning Day:
Initially, these were binary encoded, however, the variance of both of these features was found to be close to zero and therefore will not improve the performance of the model much as they are nearly constant. These features were removed.

Dew point tempurature and Tempurature:
Temperature was dropped as a feature due to the high correlation with Dew point temperature. The reason why the Dew point temperature was retained is due to its higher variance. Additionally, Temperature was slightly more correlated with other features.

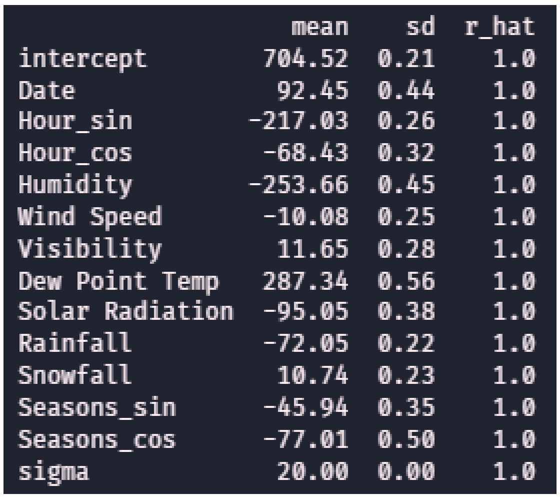
Humidity, Wind Speed, Visibility, Solar Radiation, Rainfall, Snowfall :
These features were all retained with no encoding. Despite some of these features having a moderate correlation with each other, around 0.5, I have decided to keep them to retain as much valuable information as possible. These features in particular seem relevant to the number of bikes rented out.

Standardisation:
After the steps listed above, the features were standardised. This was to bring features that may have disparate ranges into a 'standard' range. Standardising features is usually done to improve the efficiency of MCMC sampling. It isn't necessary but without standardisation, it would take longer for the chains to produce an effective sample size [3].

Task 11: Picking Priors
Flat uninformative normal priors were selected. The priors all have means of 0 and a high variance of 20. This was done due to the lack of domain-specific knowledge. Choosing uninformative priors will lead to the results being largely driven by the observed evidence rather than any prior assumptions.

Task 12: Building and running

Figure 12: MCMC run summary.



```
                    mean    sd  r_hat
intercept         704.52  0.21    1.0
Date               92.45  0.44    1.0
Hour_sin         -217.03  0.26    1.0
Hour_cos          -68.43  0.32    1.0
Humidity         -253.66  0.45    1.0
Wind Speed        -10.08  0.25    1.0
Visibility         11.65  0.28    1.0
Dew Point Temp    287.34  0.56    1.0
Solar Radiation   -95.05  0.38    1.0
Rainfall          -72.05  0.22    1.0
Snowfall           10.74  0.23    1.0
Seasons_sin       -45.94  0.35    1.0
Seasons_cos       -77.01  0.50    1.0
sigma              20.00  0.00    1.0
```

Task 14: Posterior mean values

The posterior mean values in Bayesian linear regression represent the expected values of the model parameters given both the prior information and the observed data. A negative mean implies that, on average, the target variable tends to decrease as the associated parameter increases, while a positive mean suggests the opposite relationship, on average, the target variable tends to increase as the parameter grows larger.

The majority of mean values presented in Figure 13 aren't surprising. Humidity, Rainfall, Solar Radiation, and Wind Speed all show negative values ranging from -10.08 for wind speed down to -253.66 for Humidity. This makes logical sense as people are less likely to want to ride bikes in harsh weather conditions. A surprising value is +10.74 for snowfall as I would assume that people would not like to ride bikes in the snow. This positive posterior mean for snowfall value may be due to the modifications applied to data such as standardisation. Solar Radiation being negative is also surprising but it can be reasoned that too harsh sunlight may deter people from cycling.

The posterior mean for Hour_sin, Hour_cos and Seasons_sin and Seasons_cos are both negative for each, this suggests that on average, more bikes are rented between summer and spring and just after midday but before night which is also makes logical sense.

Task 15: BLR suitability

BLR is not suitable for this data set for several reasons. The data contains features like Date, indicating that data points are not independent, BLR assumes each data point is independent. BLR does not inherently handle time series dependencies or temporal patterns well. BLR assumes a linear relationship between features and the target, however, this data may not exhibit a linear relationship. Additionally, this data contains a seasonality feature for which BLR cannot easily capture repeating trends without modifications to the data. A more suitable model for this data would be a time series model.

References

[1] Goodfellow-et-al. (2016). Deep Learning, pp.308

[2] T. Furukawa and Q. Zhao, "Interpreting Multilayer Perceptrons Using 3-Valued Activation Function". (2017). 3rd IEEE International Conference on Cybernetics (CYBCONF), Exeter, UK, 2017, pp. 1-6, doi: 10.1109/CYBConf.2017.7985786.

[3] Kruschke, J. K. (2016). Doing Bayesian Data Analysis, 2nd Edition (DBDA2E) Section 17.2.1.1 pp. 485

[4] Matthew D. Hoffman and Andrew Gelman. (2011). The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo