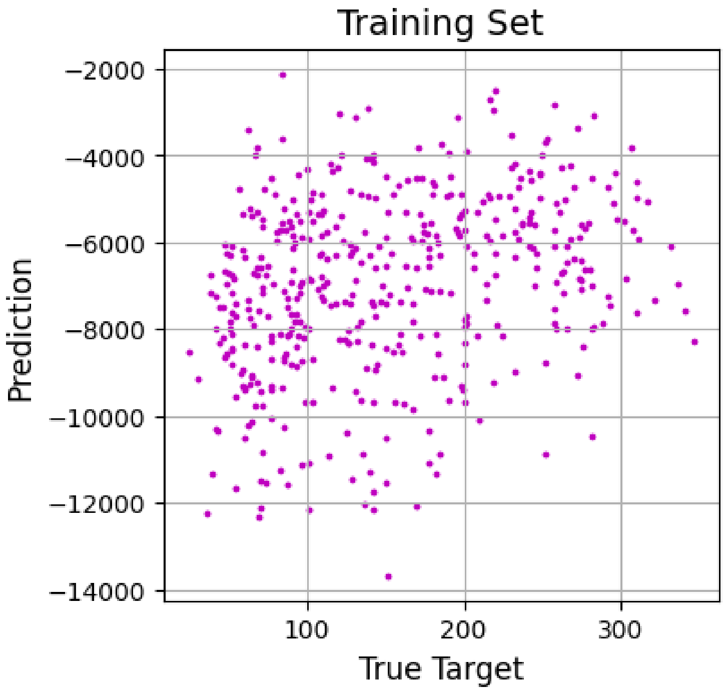
CO544: Machine Learning and Data Mining
Machine Learning Lab Five

Ranage R.D.P.R. - E/19/310

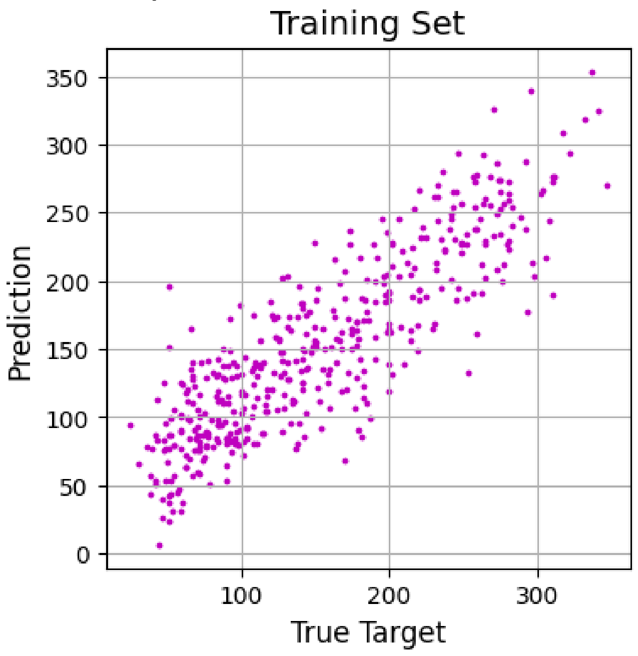
1. Radial Basis Functions (RBF)

1. Study the skeleton implementation of a Gaussian RBF model given in the Appendix.

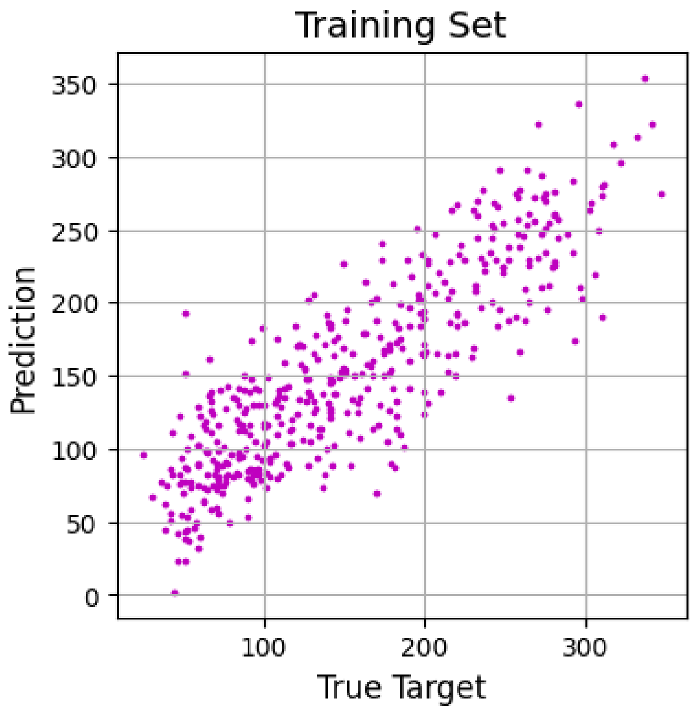


2. Make the following improvements to the given implementation:

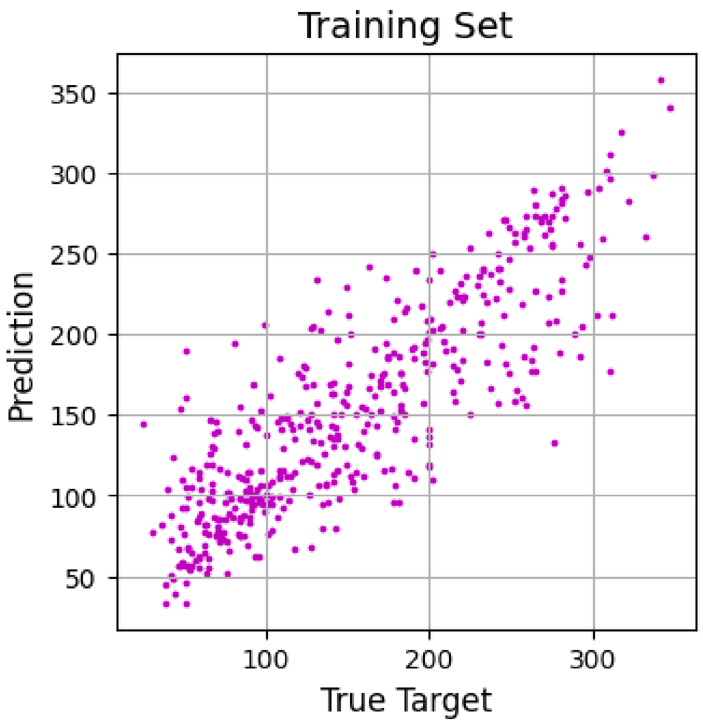
- **Normalize each feature of the input data to have a mean of 0 and standard deviation of 1.**



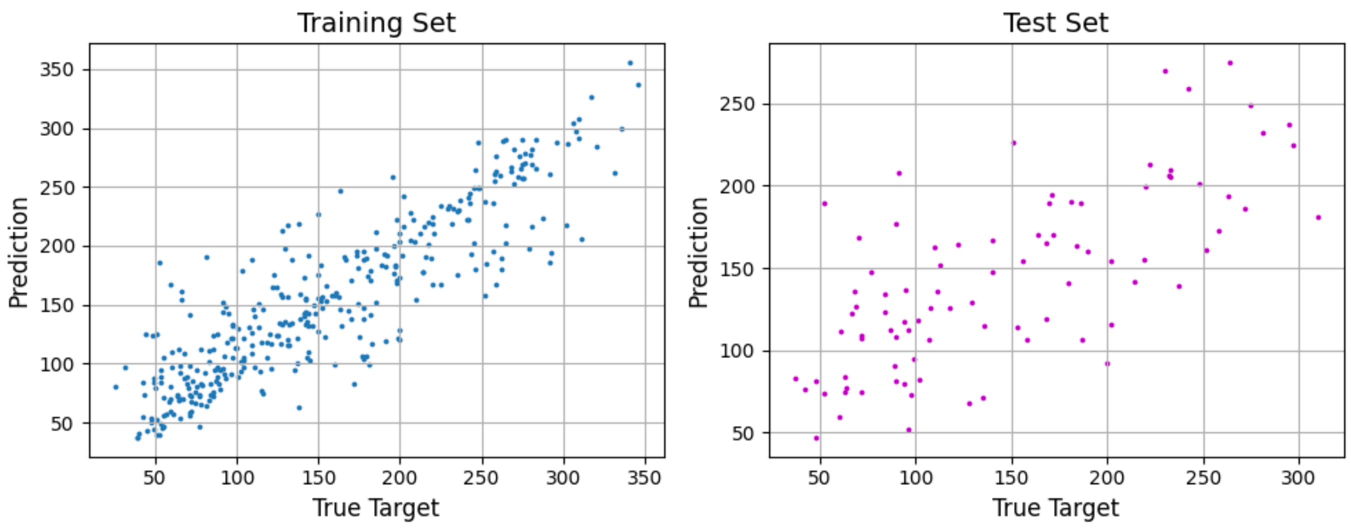
- **The width parameter of the basis functions σ is set to be the distance between two randomly chosen points. Could this sometimes cause an error? Change it to be the average of several pairwise distances.**



- **The locations of the M basis functions mj are set at random points in the input space. Cluster the data using K-means clustering (with K = M) and set the basis function locations to the cluster centres.**

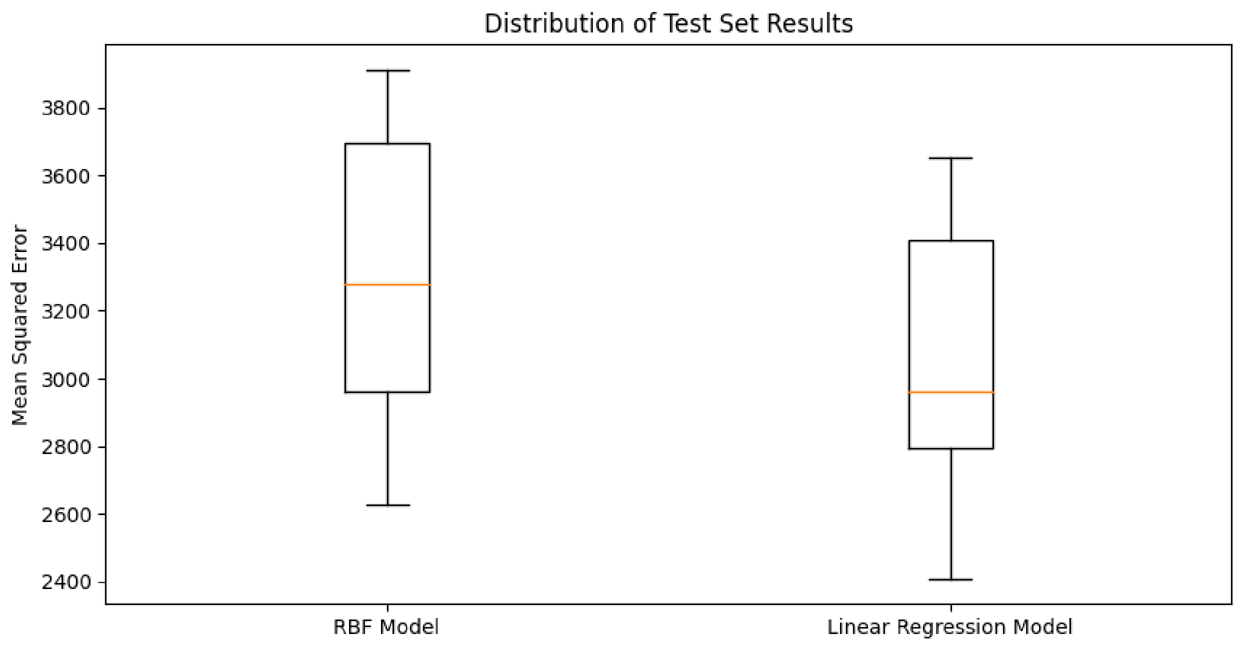


- **Split the data into training and test sets, estimate the model on the training set and note the test set performance.**

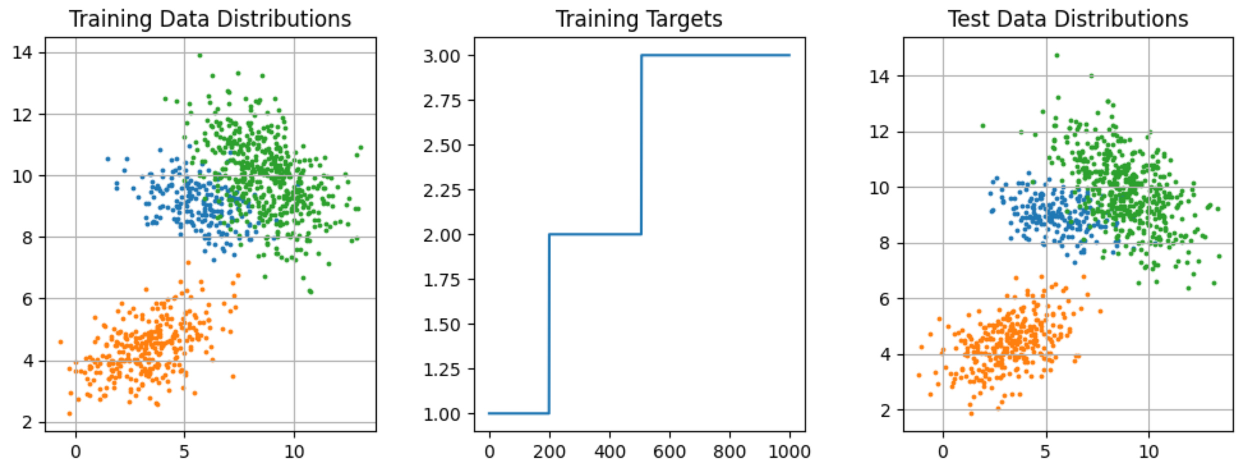


3. Implement ten-fold cross validation, where you split the data into ten parts, train on nine tenths of the data and test on the held out tenth set, repeating the process ten times.
Mean Squared Error on Test Set: 2574.99080225848

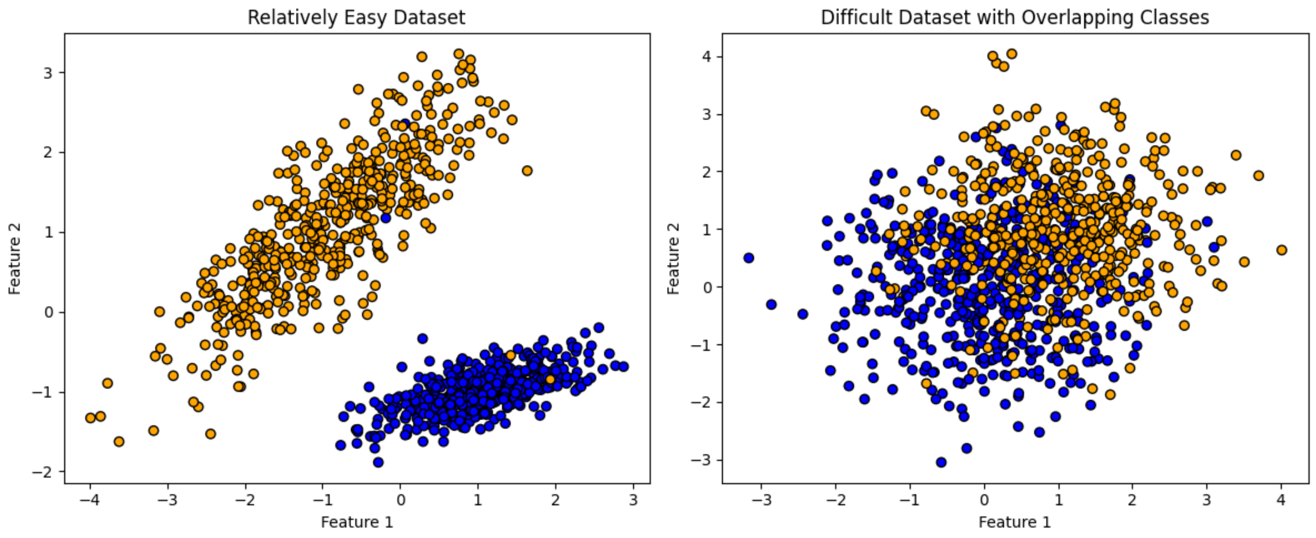
4. Display the distributions of test set results for the RBF and Linear Regression Models as boxplots side by side.



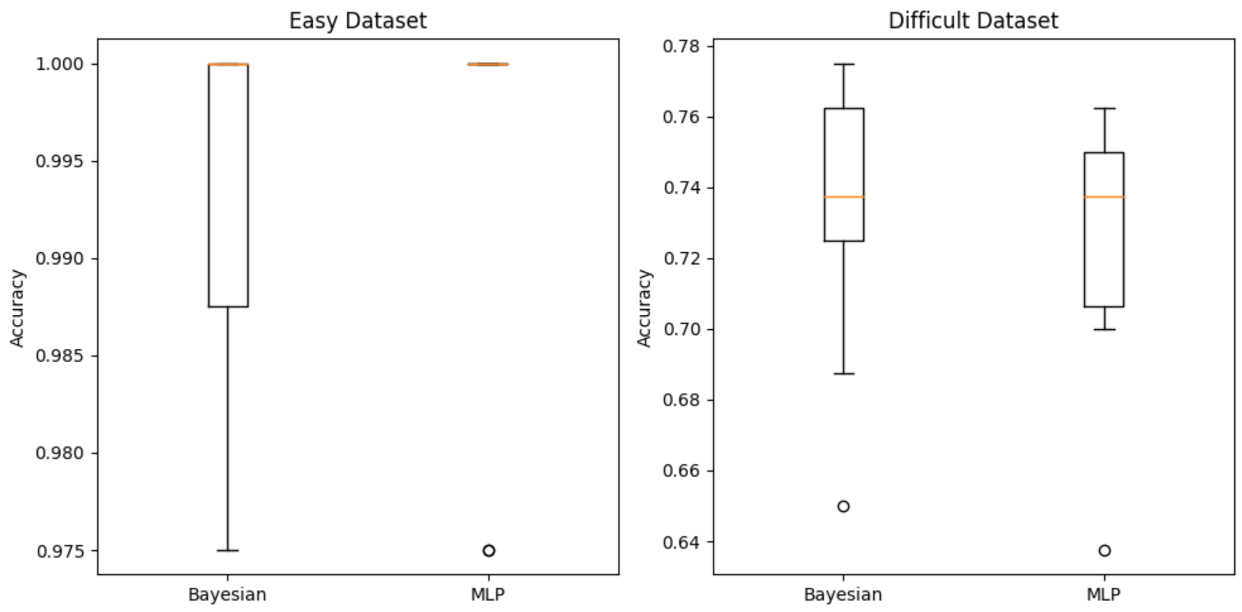
2. Multi-Layer Perceptron (MLP)



1. Set up two classification problems, one relatively easy to learn (i.e.) the classes are far apart and another in which the classe overlap (say, approximately 20% of the data overlap) and difficult to learn. Each class may be either a Gaussian or a Mixture of Gaussians.

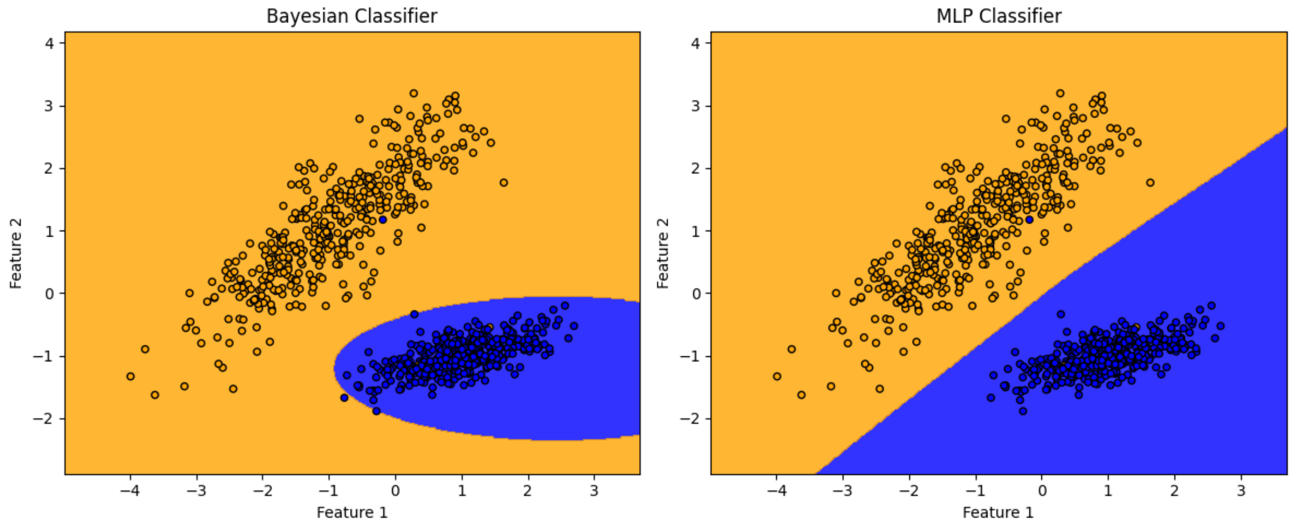


2. Split the data into training and test sets and implement a Bayesian classifier and an MLP classifier and compare their performances. Your answer should be in the form of two boxplots obtained by ten-fold cross validation.

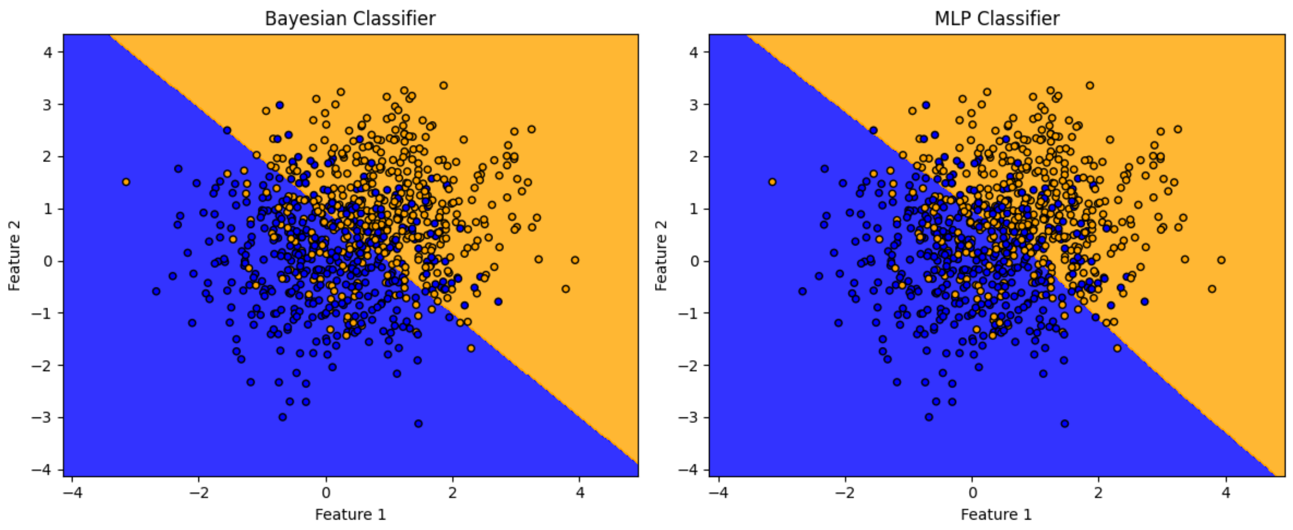


3. For one of the partitions (of training - test split), plot the class boundaries from the Gaussian and MLP classifiers. Compare a very simple MLP (with a very small number of hidden nodes) and a complex one.

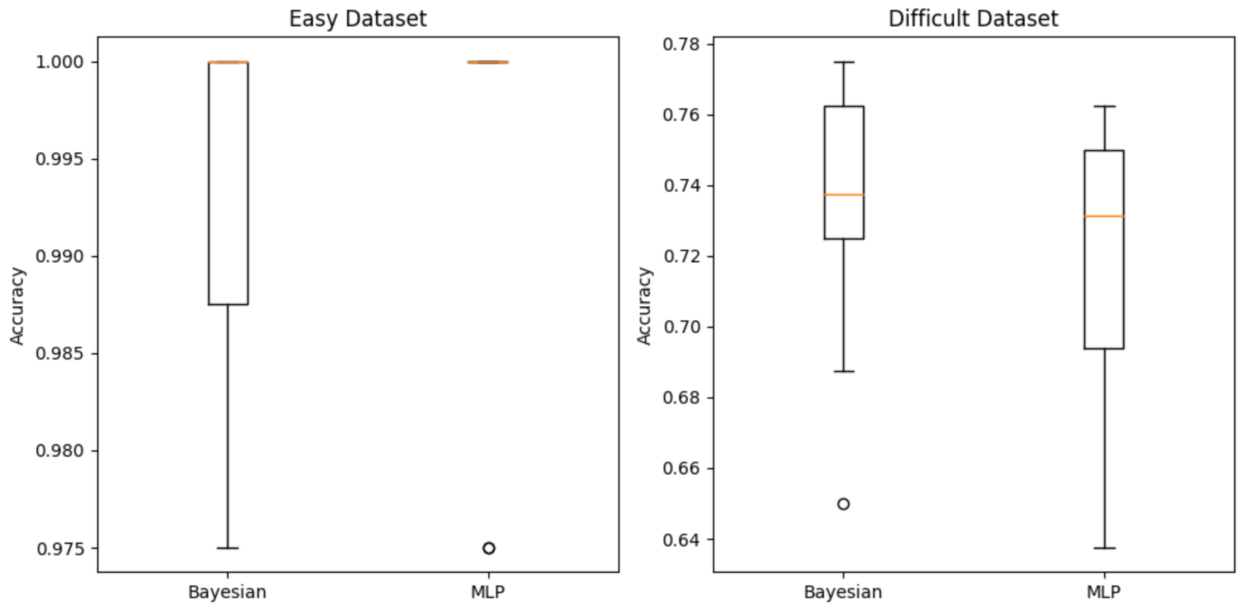
Easy Dataset



Difficult Dataset



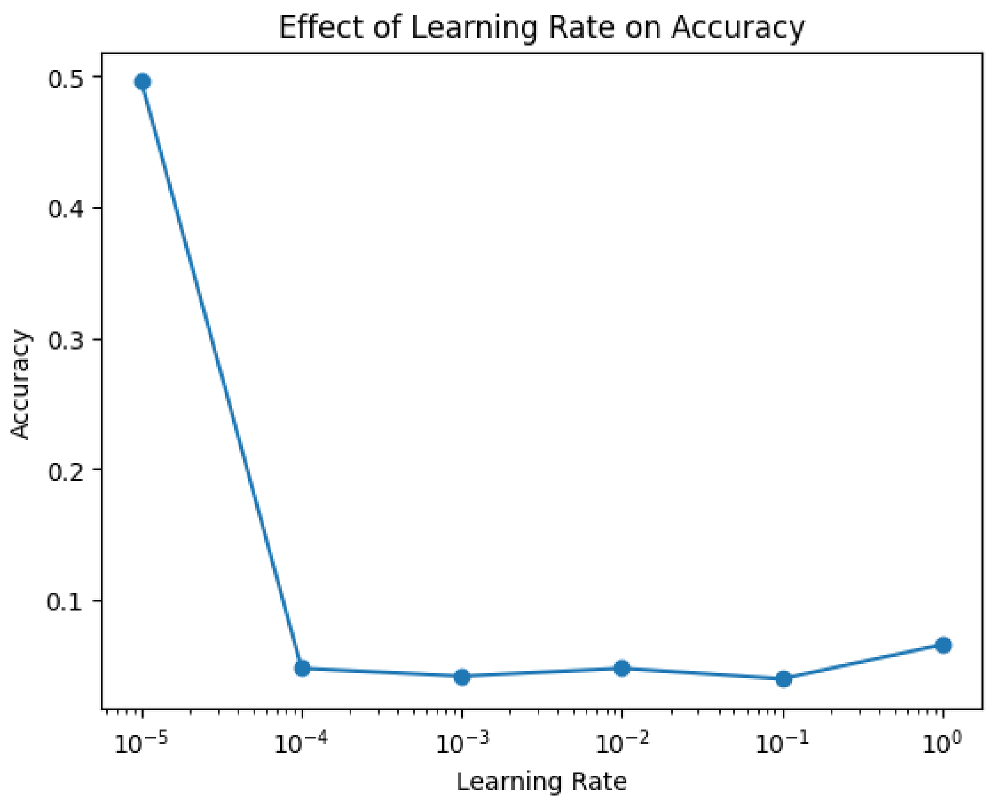
4. Read in the documentation what the role of each of the parameters is, and show the effect of changing any two of them in your report (e.g. How does convergence change of you choose a different value for the learning rate in it parameter?).

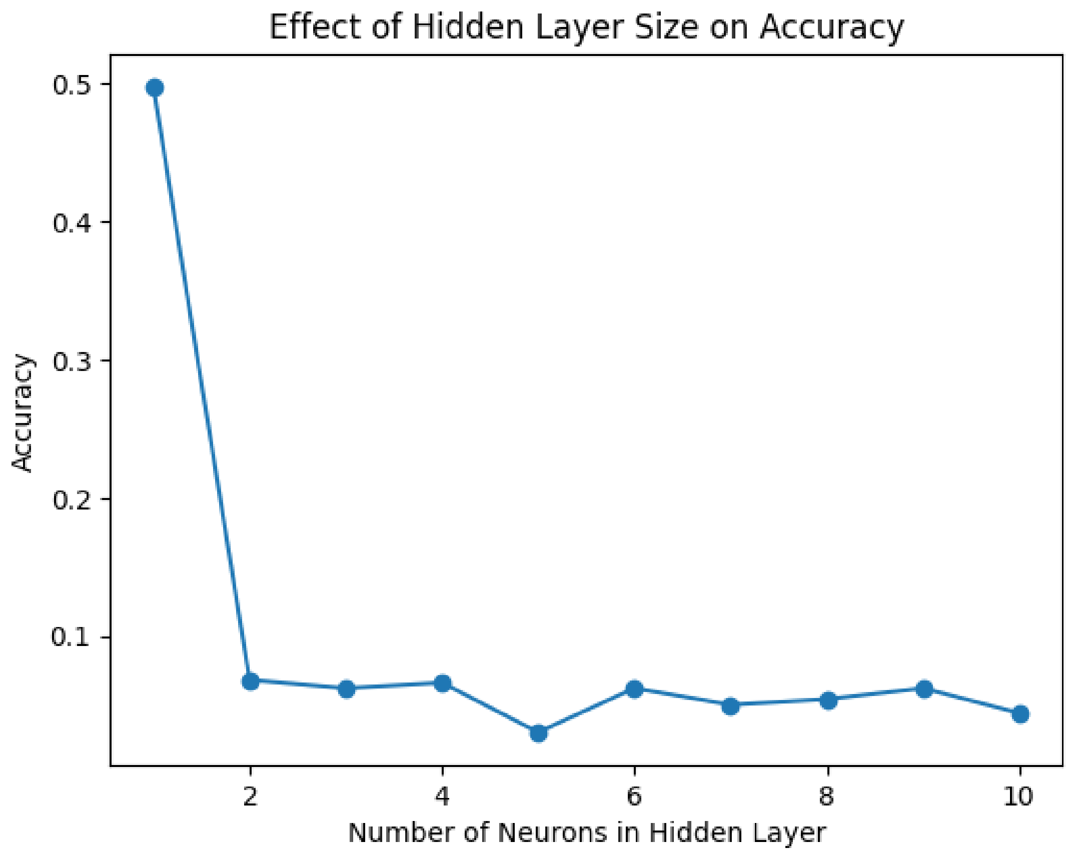


- activation: The activation function for the hidden layer. It could be 'identity', 'logistic', 'tanh', or 'relu' (Rectified Linear Unit).
- alpha: L2 penalty (regularization term) parameter.
- batch_size: Number of samples per batch during training. It can be 'auto' (default) where the batch size is min(200, n_samples).
- beta_1, beta_2: Exponential decay rates for the first and second moments estimates in Adam optimizer.
- early_stopping: If set to True, training will stop when validation score doesn't improve.

- epsilon: Value for numerical stability in Adam optimizer.
- hidden_layer_sizes: Tuple representing the number of neurons in each hidden layer.
- learning_rate, learning_rate_init: The learning rate schedule for weight updates. 'constant', 'invscaling', or 'adaptive'. The initial learning rate.
- max_iter: Maximum number of iterations.
- momentum: Momentum for gradient descent update.
- nesterovs_momentum: Whether to use Nesterov's momentum.
- power_t: The exponent for inverse scaling learning rate.
- random_state: Seed for random number generator.
- shuffle: Whether to shuffle samples in each iteration.
- solver: The optimization algorithm to use. It could be 'adam', 'lbfgs', or 'sgd'.
- tol: Tolerance for optimization.
- validation_fraction: Fraction of training data to set aside as validation set.
- verbose: Whether to print progress messages.
- warm_start: If set to True, it will reuse the solution of the previous call to fit as initialization.

Learning Rate: While slower convergence may result from lower learning rates, the model may be able to produce more accurate and stable solutions. Smaller learning rates are too sluggish to be of any benefit because they take a lot longer to converge. During training, models with higher learning rates typically converge more quickly; nevertheless, this convergence may not last forever if LR is too great.





Max Iterations: The model may train for a longer period of time when the Max iteration is increased, and this improves accuracy as the heat map illustrates. However, if the maximum iteration is relatively high, the model can be overfitting to the same data without making any sense of the new data.