Assignment 2: Tunning

Joshua Bainbridge

250869629

Description of the Selected Classification Problem

Classification problem: evaluate a patient's medical history in order to predict the likelihood the patient currently has or is likely to contract heart disease.

Heart disease is the most common cause of death in the world. The ability to forecast the potential of an individual have or contracting heart disease could be a valuable resource for physicians, hospitals, and patients.

This is classification forecasting problem. The final result should be able to accurately classify a patient as either having (or at risk of heart disease) or not having (or at risk of) heart disease base on a give set of independent cardiac related medical attributes.

Description of Available Data

The data set being used to build a forecasting algorithm was provided in a Kaggle competition posting [https://www.kaggle.com/fedesoriano/heart-failure-prediction]. The dataset is made of 920 individual patient medical information. Each patient in the data set is described by two background attributes, nine medical attributes with some relationship to heart failure and a single classification attribute denoting whether or not the patient has heart disease. All data is being used to complete this assignment.

Background Attributes

The background attributes for this data set are age and sex. These attributes are considered background attributes because they do not relate directly to medical conditions but are clinically relevant when determining the likelihood a patient has heart disease.

Medical Attributes

Medical attributes are attributes which describe medical conditions which have been shown to have a relationship with heart disease in patients. These attributes include chest pain type, resting blood pressure, cholesterol levels, fasting blood pressure, resting ECG, maximum heart rate, exercise angina, exercise relative to rest (old peak) and exercise induced increments in heart rate (ST slope). These attributes are used in the dataset because individually each of them is often found in patients with heart disease, however patients present with unique variations and levels of each of these conditions.

Numerical Attributes

Resting blood pressure, cholesterol levels, fasting blood pressure, maximum heart rate and old peak attributes are given as numerical values. After examining each attribute no large outliers or excessively large or small values were found. Therefore the data did not need to be normalized prior to being used in the three forecasting algorithms.

Ordinal Attributes

Sex, chest pain type, resting ECG, exercise angina and ST slope are all given in the data set as ordinal attributes. Prior to being used in a forecasting algorithm the data was vectorized. The table below details the results of the vectorizing process.

Table 1. Vectorized Data from Given Ordinal Data

Sex	Vectorized	Chest Pain Type	Vectorized	Resting ECG	Vectorized	ST Slope	Vectorized	Exercise Angina	Vectorized
M	0	ATA	0	Normal	0	Down	0	N	0
F	1	NAP	1	ST	1	Flat	1	Y	1
		ASY	2	LVH	2	Up	2		
		TA	3						

Classification Attribute

The final attribute is a heart disease attribute. This attribute represents the dependent variable in this forecasting problem. In the given data set the attribute is given as a binary value. The value is either zero, the patient does not have heart disease or one, the patient has heart disease.

Overview of Network Architecture

Selected Network: Feed Forward Neural Network

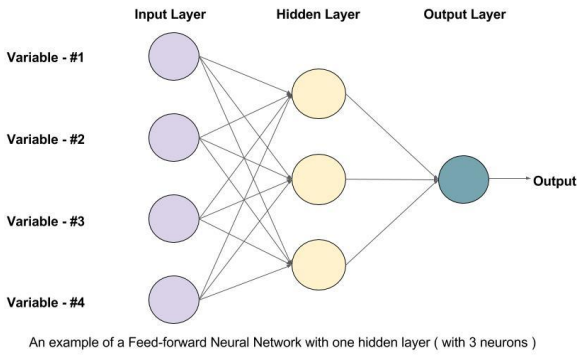


Figure 1. https://learnopencv.com/wp-content/uploads/2017/10/mlp-diagram.jpg

A feed forward neural network is a deep learning algorithm made of three different types of layers of perceptrons. An input layer, output layer and hidden layer(s). The hidden layers are made of perceptron each fulling connected to the pervious and next layer. Each perceptron connection has a associated bias. These biases are altered and adjusted during the training process.

Other Architectures

Convolutional neural networks (CNN) are deep learning algorithms derived from the feedforward neural network. However instead of being fully connected the CNN uses a convolution filter and only connected data points that are in close proximity to one another. The network functions by repeating the pattern of convolution filtering, ReLU and pooling. An CNN was not selected as the deep learning architecture for this assignment because the data set selected is not given in an image format.

Recurrent neural network (RNN) is deep learning algorithms base on the feed forward neural network that adds an extra dimension (time) into the network. The connection between neutrons spans adjacent

time steps. An RNN was not selected as the deep learning architecture for this assignment because the data set is independent of time.

Autoencoders are fully connected deep learning algorithms that have a unique hidden layer structure. With each successive layer of an autoencoder in the first half of the network the number of neurons per layer decrease. The mid point of the hidden layers in the network has the lowest numbers of neurons. This point is, the bottleneck, is the point where the inputs is compressed. The layers after the bottle neck mirrors the number of neurons in the first half. The final output of the network is the same as the input values. An auto encoder was not selected as the deep learning algorithm used in this assignment because autoencoders should be trained with normal data and data with anomalies will be detected as part of the test set. The given data in roughly an even split of patients with heart disease and without and would not be enough data to train a deep learning network.

Tuning parameters

A grid search tuning methodology was employed to tune the feed forward neural network. Grid search is a brute force approach that tests all permutations of values a hyperparameter to find the set of values that have the lowest cost. Grid search is a very thorough tuning method however it is very computationally expensive. In order to reduce the computational costs grid search was employed first by having large search boundaries to find a region(s) that produce networks with the highest precision. Each of the highest performing set of hyperparameters were comparted in the results sections.

The following hyperparameters were considered:

Number of hidden layers: The number of hidden layers was tuned for 1 to 5 hidden layers

Number of neurons in each layer: The number of neurons in each hidden layer was tuned with all the layers containing the same number of neurons.

Learning Rate: Two learning rates were tested, a constant learning rate of 0.0001 and an adaptive learning rate of the current learning rate divided by 5 after two successive epoch fails.

Activation Function: Two activation functions were tuned, hyperbolic (tanh) and rectified linear unit function (ReLU)

Solver: The solver was not tuned and was set to ‘sgd’. This solver was the only solver that allowed for learning rates to changed.

Max Iterations: is the maximum number of epochs the neural network will attempt if no convergence point if found. The maximum number of iterations was tuned however it was not tuned as part of grid search. A concern when determining the number of epochs use to build a neural network is over fitting. Over fitting can occur with the neural network fits too well to the training data and can no longer account for slight variations in test or validation data. Due to the limited amount of data for this problem, it would be impossible to determine if a good model is overfitting. Therefore, the maximum iterations hyperparameter was tested at 50, 100, 200, 400 and 800, with 400 epochs being the lowest number of iterations required for all tests to converge.

Tuning Boundaries

Hyperparameter	Boundaries
Number of layers	1,2,3,4,5
Number of neurones per layer	11,66,121
Activation function	‘tanh’, ‘ReLU’
Learning rate	0.0001, adaptive
Solver	‘sgd’
Max Iterations	400

Process

Preprocessing / Feature generation

The only preprocessing required prior to training the feed forward neural network was to convert all data types represented as ordinal data into numerical data, as described in the *Description of Available Data* section. The categories of sex and exercise angina have binary value options therefore they were vectorized as 1 or 0. The categories of chest pain type and resting ECG have independent value options, therefore they were vectorized from 0-3 and 0-2 respective, in the order they first appeared in the data set. The category ST slope contains three value options that are associated to each other by different amounts, i.e. an increasing (up) ST slope is associated closer to a flat ST slope then a decreasing (down) ST slope. Similar to how when vectorizing sales for a business, the month April and May are more closely related than April and October. Due to the association the ST slope was vectorized in the specific sequence, down, flat , up and labeled 0,1,2 respectively.

No feature generation was done to the data set due to the fact that this data set is not a time series, no new pertinent attributes could be extracted.

Training Model

The program begins by reading in the data and separating the attributes into two data frames. The independent attributes are store in the X data frame and the Y data frame contains the single dependent attribute.

Once the data set is split into dependent and independent data, it is split into training and test sets. The split used in this assignment is the hold-out method. The data split of the data used in this assignment the split of data is 80% (736 data points) are used as training data and 20% (184 data points) are used at testing data.

The feedforward neural network was applied using the *sklearn.neural_network* library using *MLPClassifier* in python.

Testing and Tuning Model

In order to tune the neural network the hypermeters being tested were stored in an array. The array and neural network where they passed to *sklearn.GridSearchCV,* which trained and tested each permutation of hyperparameter combinations, and the precision of the final model printed to the console.

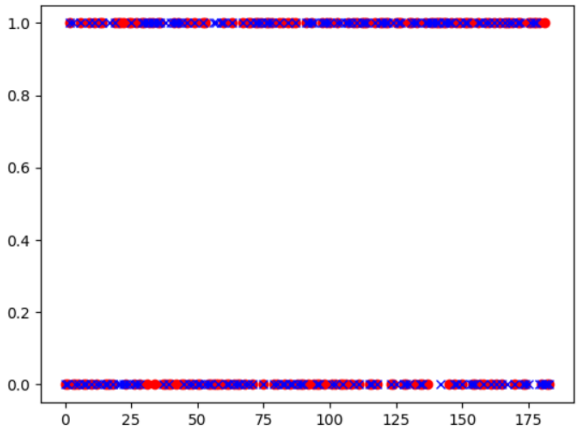
Results

The full table of results for iteration 1 can be found in the appendix.

Precision	Parameters [Number of layers, Number of neurons, Activation function, Learning rate]	Recall	F1 - Score
0.79	[121, 1, 'ReLU', adaptive]	0.81	0.79
0.78	[66, 1, 'ReLU', adaptive]	0.76	0.76
0.765	[121, 3, 'ReLU', adaptive]	0.76	0.75
0.763	[66, 2, 'ReLU', adaptive]	0.74	0.72

Testing Tuned Neural Network

Best performing hyperparameters: [121, 1, 'ReLU', adaptive]



	POSITIVE	NEGATIVE
TRUE	76	71
FALSE	20	17

Evaluations Metrics

Accuracy is the number of correctly classified instances divided by the total number of data points.

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN} = \frac{76 + 71}{20 + 17} = 79.8\%$$

Precision is the number of correctly identified positive outputs divided by the number of outputs identified as positive.

$$Precision = \frac{TP}{TP + FP} = \frac{76}{76 + 20} = 0.791$$

Recall is a measure of the sensitivity of the model.

$$Recall = \frac{TP}{TP + FP} = \frac{76}{76 + 17} = 0.817$$

Conclusion

	Assignment 2 - FFNN	Assignment 1 – Logistic Regression
Accuracy	79.8%	56.2%
Precision	0.791	0.496
Recall	0.817	0.513

After constructing a confusion matrix for the tuned feed forward neural network and computing the accuracy, precision, and recall the results were compared with the best results from assignment 1. As seen in the table above using a feed forward neural network is a significantly better solution to use when attempting to classify patients with heart disease.

Appendix

Score	Activation	Alpha	Layers Size	Neuron Number	Learning Rate	Iterations	Solver
665	tanh	0.0001	5	11	constant	800	sgd
698	tanh	0.0001	5	11	adaptive	800	sgd
610	tanh	0.0001	4	11	constant	800	sgd
585	tanh	0.0001	4	11	adaptive	800	sgd
689	tanh	0.0001	3	11	constant	800	sgd
684	tanh	0.0001	3	11	adaptive	800	sgd
691	tanh	0.0001	2	11	constant	800	sgd
674	tanh	0.0001	2	11	adaptive	800	sgd
707	tanh	0.0001	1	11	constant	800	sgd
597	tanh	0.0001	1	11	adaptive	800	sgd
703	relu	0.0001	5	11	constant	800	sgd
688	relu	0.0001	5	11	adaptive	800	sgd
663	relu	0.0001	4	11	constant	800	sgd
745	relu	0.0001	4	11	adaptive	800	sgd
721	relu	0.0001	3	11	constant	800	sgd
698	relu	0.0001	3	11	adaptive	800	sgd
721	relu	0.0001	2	11	constant	800	sgd
708	relu	0.0001	2	11	adaptive	800	sgd
729	relu	0.0001	1	11	constant	800	sgd
755	relu	0.0001	1	11	adaptive	800	sgd

Score	Activation	Alpha	Layers Size	Neuron Number	Learning Rate	Iterations	Solver
708	tanh	0.0001	5	66	constant	800	sgd
721	tanh	0.0001	5	66	adaptive	800	sgd
727	tanh	0.0001	4	66	constant	800	sgd
734	tanh	0.0001	4	66	adaptive	800	sgd

711	tanh	0.0001	3	66	constant	800	sgd
718	tanh	0.0001	3	66	adaptive	800	sgd
709	tanh	0.0001	2	66	constant	800	sgd
713	tanh	0.0001	2	66	adaptive	800	sgd
714	tanh	0.0001	1	66	constant	800	sgd
715	tanh	0.0001	1	66	adaptive	800	sgd
722	relu	0.0001	5	66	constant	800	sgd
773	relu	0.0001	5	66	adaptive	800	sgd
721	relu	0.0001	4	66	constant	800	sgd
720	relu	0.0001	4	66	adaptive	800	sgd
725	relu	0.0001	3	66	constant	800	sgd
731	relu	0.0001	3	66	adaptive	800	sgd
747	relu	0.0001	2	66	constant	800	sgd
765	relu	0.0001	2	66	adaptive	800	sgd
743	relu	0.0001	1	66	constant	800	sgd
780	relu	0.0001	1	66	adaptive	800	sgd

Score	Activation	Alpha	Layers Size	Neuron Number		Learning Rate	Iterations	Solver
712	tanh	0.0001	5		121	constant	800	sgd
738	tanh	0.0001	5		121	adaptive	800	sgd
716	tanh	0.0001	4		121	constant	800	sgd
741	tanh	0.0001	4		121	adaptive	800	sgd
720	tanh	0.0001	3		121	constant	800	sgd
702	tanh	0.0001	3		121	adaptive	800	sgd
714	tanh	0.0001	2		121	constant	800	sgd
710	tanh	0.0001	2		121	adaptive	800	sgd
725	tanh	0.0001	1		121	constant	800	sgd
717	tanh	0.0001	1		121	adaptive	800	sgd
735	relu	0.0001	5		121	constant	800	sgd
738	relu	0.0001	5		121	adaptive	800	sgd
722	relu	0.0001	4		121	constant	800	sgd
759	relu	0.0001	4		121	adaptive	800	sgd
747	relu	0.0001	3		121	constant	800	sgd
763	relu	0.0001	3		121	adaptive	800	sgd
737	relu	0.0001	2		121	constant	800	sgd
739	relu	0.0001	2		121	adaptive	800	sgd
755	relu	0.0001	1		121	constant	800	sgd
790	relu	0.0001	1		121	adaptive	800	sgd