**Please describe the background, context, and importance of the data in light of related literature. Show a sound interpretation of the medical problems presented in the data. Outline the selected dataset (including features and class labels) and provide descriptive statistics of the contained variables. Visualise the feature space in a plot and explore the underlying characteristics.**

* **Describe the data cleansing, feature selection, feature construction, and feature preprocessing of the chosen dataset.**
* **Select two supervised models of the course. Give a high-level description of both algorithms including their pseudo-code. Describe and demonstrate for both classification algorithms:**
  + - **supervised learning on training data**
    - **optimization of hyper-parameters**
    - **model evaluation including but not limited to criteria such as confusion matrix,**
    - **precision, recall, F1, and AUC**
* **Demonstrate your solution with an attached iPython notebook. Ensure reproducibility and transparency by using an URL with the original dataset.**

**Present optimized hyper-parameters and reasonable evaluation criteria such as a confusion matrix, precision, recall, F1, AUC and a ROC-plot. Provide a sensitivity analysis for both algorithms with different parameters and give a textual description of the results.**

**Discussion and Conclusion Words Compare and discuss your findings with other scientific publications that used the same medical dataset. Discuss how you would improve your methodology. Outline the potential usage of the trained algorithms in healthcare and health service delivery. What benefits might be anticipated from their deployment?**

Cancer is the second leading cause of deaths in the world. Breast cancer is the most commonly occurring cancer in women (WHO, 2018). In 2013 breast cancer had the highest incidence among women, with an estimated number of 1.8 million new cases, leading to around 471,000 deaths in the same year (Thompson, 2015).

The exact cause of the breast cancer remains unknown, early detection and diagnoses are considered highly important in determining the success of treatment and improving survival from breast cancer (ACS, 2015). Mammograms produce images of the inside of the breasts using low-dose X-rays and detect cancer in the preclinical phase when it is small and impalpable (NICE, 2017). Around 5-10% of the mammograms do not produce conclusive results (Kopans, 1992). Biopsy is the gold standard for pathological diagnosis of breast cancer (Zhang, 2013). However, only 15–30% of biopsies performed on suspicious masses found on mammogram prove to be malignant ([Hall, 1988](https://www-sciencedirect-com.libproxy.ucl.ac.uk/science/article/pii/S0957417410012054#b0040); Kopans, 1992). In order to reduce the number of unnecessary biopsies and the associated cost and furthermore, the mental and physical discomfort for experienced by many patients.

Research is needed to find more accurate ways of identifying patients that should be referred for a biopsy. Data mining algorithms have been successfully applied to predicting breast cancer and can be utilised in assisting physicians in making decisions about whether to perform a breast biopsy on a suspicious lesions seen on a mammogram image or whether to offer a follow up examination instead (Mendelson, 2019).

The mammographic dataset investigated in this study is from the UCI Machine Learning Repository. The dataset includes BIRADS attributes for 961 full field digital mammograms that were collected at the Institute of Radiology of the University Erlangen-Nuremberg between 2003 and 2006. Each instance has got a target label that offers information about whether the mass sample was classified as benign (0) or malignant (1). 516 or the reported cases were benign and 445 were malignant. Information The BIRADS assessment rating is non-predictive and will be removed during the data-pre-processing stage (ACR, 2013). The descriptive statistics of the variables are shown in Table 1. The correlation matrix reveals that an irregular mass shape has the highest correlation to the class label; age is also highly correlated to the class. The correlation matrix further indicates that a round mass shape is also highly correlated to the mass having a circumscribed margin. The 3D plot of the features reveals that malignant cases tend to have a more lobular or irregular the mass shape, and an obscured, ill-defined or speculated mass margin as well as an increased density.

The mammographic mass dataset gives us information about the predictive BI-RADS attributes and the associated response represented as a class label for every patient. Having an associated response for every observation means that we can approach this as a supervised learning problem. The aim is to fit a model that accurately determines the relationship between the BI-RADS attributes and whether the mass is malignant or benign.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Attribute | Attribute type | Min value | Max  value | Mean value | Mode value | Original data type | Number of missing values |
| BI-RADS Assessment | Ordinal/ Non- predictive | 1 | 5 | 4.35 | 4 | Object | 2 |
| Age | Ordinal continuous | 18 | 96 | 55 | 59 | Integer | 5 |
| Mass shape | Nominal:   * Round - 1 * Oval - 2 * Lobular - 3 * Irregular - 4 | 1 | 4 | n/a as nominal feature | 4 | Object | 31 |
| Mass margin | Nominal:   * Circumscribed - 1 * Microlobulated - 2 * Obscured - 3 * ill-defined - 4 * speculated - 5 | 1 | 5 | n/a as nominal feature | 1 | Object | 48 |
| Mass density | Ordinal:   * Extremely dense -1 * Heterogeneously dense -2 * Scattered areas of fibroglandular density - 3 * Almost entirely fatty -4 | 1 | 4 | 2.91 | 3 | Object | 76 |

**Table 1.** Overview of the descriptive statistics of the predictive and non-predictive features.

Before applying a supervised learning model to the dataset, we need to ensure that any data quality issues are identified and addressed.

The mammographic mass dataset has 160 attribute values are missing, this is about 16% of the data. Removing all rows with missing values will lead to losing 125 rows of valuable information. Imputation would allow any missing values to be replaced with plausible estimated values, however imputing can also introduce bias and reduce the power of the prediction, therefore, the missing values will be removed (Azur, 2011).

Both Support Vector Classifier (SVC) and Multilayer Perceptron (MLP) are algebraic machine learning models; hence the input vector needs to be numerical. Categorical nominal variables have been assigned numerical labels from 1-4, hence the LabelEncoder does not need to be used. However, classifiers believe these labels to be ordinal, meaning that the classifier considers mass shape irregular (4) to be greater than mass shape round (1). One Hot Encoding using dummy variables converts each category value into a new column and assigns 0 or 1 (False/True) value to the column. This allows the values to be assigned appropriate weighting by the classifier.

Selecting only meaningful features reduces the complexity of the model and overfitting and improves the generalization capabilities of the classifier. This dataset has only 4 predictive features and all of which will be used to build the models. Feature extraction is another way to reduce the complexity of a model. Principal component analysis (PCA) is a feature extraction technique that finds the direction of the maximum variance by projecting the data onto a lower dimensional feature space (Raschka, 2017).

Plotting explained variance against the number of principal components indicates that the first and second principal components account for around 26% and 14 % of the variance, respectively. Plotting the accuracy against the number of principal components shows that choosing either 1 or 3 principal components produces the best accuracy with the SVC model, selecting 2 or 3 principal components procures best accuracy with MLP.

All input values are standardized to have a mean of zero and standard deviation of one. This allows all inputs to be treated equally in the regularization process, and appropriate starting weights to be selected by the classifier(Hastie et al., 2009; Raschka, 2017).

Class distribution is balanced with around 51% of the tumours in the dataset benign and 49 % malignant, if the distribution was skewed, undersampling could be used to remove some of the overrepresented instances.

The data split with 80% of the data used for building the model and selecting the optimal hyperparameters and 20% of the data saved for independently assessing the individual models.

The Support Vector Classifier (SVC) is a supervised classification algorithm. Support Vector Machine (SVM) is an extension of the SVC and uses kernels to enlarge the feature space (James 2013; Fung, 2015). For non-linearly separable data, the SVM maps the data to a higher dimensional feature space using a non-linear mapping kernel function. SVM algorithm finds the optimal hyperplane that can separate all objects of one class from those in the class with the largest margin. The optimal hyperplane is found by solving a convex quadratic optimization problem (Hastie et al., 2009). The training samples that fall on the margins of the hyperplane are called support vectors (Mokhtar, 2013). The hyperplane is built by a set of these support vectors, when the algorithm is applied on the test data, it determines which side of the hyperplane they fall on and subsequently suggests a class label (Fung, 2015; Jain, 2017).

**Pseudocode 1.** Pseudocode for training an SVM (Pedersen, 2006).

**Require:** X and y loaded with training labelled data, α ⇐ 0 or α ⇐ partially trained SVM

1: C ⇐ some value (10 for example)

2: **repeat**

3: **for all** {xi, yi}, {xj, yj} **do**

4: Optimize αi and αj

5: **end for**

6: **until** no changes in α or other resource constraint criteria met

**Ensure:** Retain only the support vectors (αi > 0)

### Both SVM and MLP can be tuned by using GridSearchCV. In nested cross-validation, the “inner” cross-validation uses GridSearchCV to perform an exhaustive search over the specified hyperparameter values to find the best combination of hyperparameters for the classifier. Subsequently, the “outer” cross-validation independently evaluates the performance of the model (Halder & Ozdemir, 2018).

The following parameters can be tuned to optimise SVM: C is the regularization parameter and is usually set between 1 and 10, increasing the value improves the prediction accuracy for the training data, but this can also lead to over-fitting. γ is the kernel coefficient by increasing γ the flexibility of the model increases and this can help improve accuracy; C and γ a regulate the trade-off between training errors and misclassifications. Looking at the validation curve with differing C and γ values shows that increasing the C value from 1 to 10 does improve the accuracy on the training set, but not on the test set. Furthermore, increasing the size of γ from 0.1 to 1 also decreases the accuracy on the test set. The best accuracy on the test set is achieved with a C value of 6 and γ value of 0. 1. Linear kernel is used for linearly separable data and radial basis kernel (rbf) for non-linearly separable data.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Results optimised for: | Best relevant score  on training set | Best relevant score on test set | C | kernel | gamma |
| Accuracy | Accuracy  0.823 +/- 0.03 | Accuracy  0.75595 | 0.001 | linear | n/a |
| Precision | Precision  0.797 +/- 0.049 | Precision  0.72024 | 0.01 | rbf | 1.0 |
| Recall | Recall  0.917 +/- 0.044 | Recall  0.75000 | 0.01 | rbf | 0.1 |
| F1 score | F1 score  0.835 +/- 0.029 | F1 score  0.75595 | 0.1 | rbf | 0.01 |
| AUC | AUC  0.877 +/- 0.043 | AUC  0.75000 | 10 | rbf | 0.001 |

**Table 2**. SVM hyperparameter values optimised for the best accuracy, precision, recall, f1 score and AUC.

Confusion matrix (Image 1) helps us understand the performance of a learning algorithm, it reports the number of True positive (TP), True negative (TN), False positive (FP) and False negative (FN) predictions of a classifier. The aim is to improve the classifier so that the number of TP and TN predictions increase and number of false predictions are decreased.



**Image 1.** Confusion Matrix (Narkhede, 2018).

Accuracy (ACC) offers information about how many samples were correctly labelled. ACC is the sum of all correct predictions (TP & TN) to all predictions. The best accuracy is achieved with hyperparameter values shown in Table 2.

Precision (PRE) shows the number positively predicted cases that were correct. Optimising for precision means tuning the parameters so that there the number of True Positive predictions is maximised.

Recall (REC) allows us to determine the sensitivity of the classification model and indicates the total number of malignant cases that were correctly identified. This is a very important measure when it comes to the Mammographic Mass dataset. As the cost of False Negatives is very high, because it would mean that someone who potentially has a cancerous mass is not referred for further investigation and treatment. The hyperparametres producing giving the best Recall score are shown in Table 2.

F1 - score is the combination of Precision and Recall and is calculated as follows:

Artificial Neural Network (ANN) is a powerful predictive two stage model that can be used for both regression and classification problems. Multilayer perceptron is a feedforward artificial neural network with an input layer, one or more hidden layers and an output layer. As data moves through the network in a feed-forward fashion, it is influenced by the connection weights and the activation function type. There are several activation functions that are in use including linear, sigmoid, tanh and Rectified Linear Unit Function (ReLu) (Mahmood, 2018). The choice of activation function depends on the model used, the hyperparameters used for optimisation and the properties of the feature vector. The values for weights are changed progressively as the algorithm finds the optimal solution that best fits the data with the backpropagation learning algorithm (Patterson, 2018). The bigger the weight the higher the correlation between the input signal and the outcome. Backpropagation uses gradient descent on the weights of the connections between layers to minimize the error on the output of the network (Patterson, 2018; Hastie et al., 2009). The weights are adjusted in the direction that most decreases the error. Output errors are propagated backwards to come up with the errors for the hidden layers. The gradient is computed by a forward and backward sweep over the whole network (Grus, 2005; Hastie et al., 2009).

##### **Pseudocode 2. General neural network training pseudocode (Patterson, 2018).**

function neural-network-learning (training-records) returns network

network <- initialize weights (randomly)

start loop

**for** each example in training-records **do**

network-output = neural-network-output (network, example)

actual-output = observed outcome associated with example

update weights in network based on

{example, network-output, actual-output}

end **for**

end loop when all examples correctly predicted or hit stopping conditions

**return** network

|  |  |
| --- | --- |
| Table 4. Neural network notation (Patterson, 2018). | |
| Notation | **Meaning** |
| *i* | Index of artificial neuron |
| n i | Neuron at index *i* |
| *j* | Index of neuron in previous layer connecting to neuron *i* |
| a i | Activation value of neuron *i* (output of neuron *i*) |
| A*i* | Vector of activation values for the inputs into neuron *i* |
| *g* | Activation function |
| *g’* | Derivative of the activation function |
| E r r i | Difference between the network output and the actual output value for the training example |
| W*i* | Vector of weights leading into neuron *i* |
| W j, i | Weight on the incoming connection from previous layer neuron *j* to neuron *i* |
| i n p u t \_ sum i | Weighted sum of inputs to neuron *i* |
| i n p u t \_ sum j | Weighted sum of inputs for neuron *j* in previous layer (used in backpropagation) |
| α | Learning rate |
| Δ j | Error term for connected neuron *j* in previous layer |
| Δ i | Error term for neuron *i*;   = E r r i × g ' ( i n p u t \_ s u m i ) |

**Pseudocode 2.** Backpropagation algorithm for updating weights (Patterson, 2018).

function backpropagation-algorithm

(network, training-records, learning-rate) returns network

network <- initialize weights (randomly)

start loop

for each example in training-records do

  // compute the output for this input example

network-output <- neural-network-output (network, example)

// compute the error and the [delta] for neurons in the output layer

example\_err <- target-output - network-output

  // update the weights leading to the output layer

Wj,i ← Wj,i + α x aj × E r r i × g’ (input\_sum i)

for each subsequent-layer in network do

  // compute the error at each node

Δ j ← g’ (input\_sum j) Σ i W j,i Δ i

  // update the weights leading into the layer

W k, j ← W k, j + α × ak × Δ j

end for

end for

end loop when network has converged

return network

**Pseudocode 3.** Pseudocode for neural network training (Grus, 2005).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| def feed\_forward( neural\_network, input\_vector): | | | | | | | |
|  | #takes in a neural network (represented as a list of lists of weights) | | |
|  | and returns the output from forward-propagating the input | | |
|  |  | | |
|  | outputs = [] | | |
|  |  | | |
|  | #process one layer at a time  for layer in neural\_network: | | |
|  | | |  | | | | |
|  | | | | | | | |
| |  | | --- | | Inpu with\_bias = input\_vector + [1] # add a bias input | | output = [neuron\_output (neuron, input\_with\_bias) for neuron in layer] | | # compute the output for this layer | | outputs.append(output) # and remember it | |  | | # then the input to the next layer is the output of this one | | input\_vector = output | |  | | return outputs |   def backpropagate (network, input\_vector, target): | | | | | | | |
|  | | | | | | | |
|  | hidden\_outputs, outputs = feed\_forward (network, input\_vector) | | | |
|  |  | | | |
|  | # the output \* (1 - output) is from the derivative of sigmoid | | | |
|  | output\_deltas = [output \* (1 - output) \* (output - target[i]) | | | |
|  | for i, output in enumerate(outputs)] | | | |
|  |  | | | |
|  | # adjust weights for output layer (network[-1]) one neuron at a time | | | |
|  | for i, output\_neuron in enumerate(network[-1]):  #focus on the ith output layer neuron | | | |
|  | for j, hidden\_output in enumerate(hidden\_outputs + [1]): | | | |
|  | | | #adjusts the jth weight based on both this neuron’s delta and  its jth output  output\_neuron[j] -= output\_deltas[i] \* hidden\_output | | | | |
|  | | |  | | | | |
|  | # back-propagate errors to hidden layer | | | | |
|  | hidden\_deltas = [hidden\_output \* (1 - hidden\_output) \* | | | | |
|  | | | dot(output\_deltas, [n[i] for n in network[-1]]) | | | | |
|  | | | for i, hidden\_output in enumerate(hidden\_outputs)] | | | | |
|  | | |  | | | | |
|  | | # adjust weights for hidden layer (network[0]) one neuron at a time | | | | |
|  | | for i, hidden\_neuron in enumerate(network[0]): | | | | |
|  | | | for j, input in enumerate(input\_vector + [1]): | | | | |
|  | | | hidden\_neuron[j] -= hidden\_deltas[i] \* input | | | | |

The following hyperparameters can be tuned to improve the performance of the MLP Classifier:

Hidden layers capture the nonlinear properties of the data and increase the flexibility of the model. The number of hidden units usually falls between 5 and 100. More layers can be used for bigger datasets with a greater number of training examples. Cross-validation and GridSearchCV can be used to estimate the optimal number of hidden layers at different levels of hidden layers. The hidden layer sizes of 100, 100, 100 gave the best results for accuracy.

Momentum for gradient descent prevents the learning algorithm from getting stuck thereby helping to continue the path towards local minimum. Increasing the momentum size helps improve accuracy. The default value for momentum is 0.9.

Learning rate controls the extent by which weights are adjusted during optimisation. A large learning rate coefficient (e.g. 1) makes big adjustments to the weights, whereas a small learning rate (e.g. 0.00001) ensures that smaller adjustments are made. Large leaps can save time initially but the may also overshoot the local minimum. A smaller learning rate is more likely to eventually reach the local minimum, but takes a very long time to run. Validation curve with differing values of initial learning rate and momentum show that the best accuracy is achieved with an initial learning rate value of 0.001 and momentum value of 0.9. Increasing the learning initial learning rate. Smaller momentum value

Learning rate can be kept as a constant at the rate that was initially specified. Adaptive learning rate keeps the learning rate constant as long as the training loss is reducing. Inv-scaling can be used to gradually decrease the learning rate. Constant learning rate produced the best accuracy score.

The solver refers to the method by which the weights are optimised, ‘sgd’ refers to stochastic gradient descent and ‘adam’ refers to a stochastic gradient-based optimizer proposed by Kingma, Diederik, and Jimmy Ba.

Alpha is the L2 regularization coefficients; regularization helps prevent over fitting by regulating the extent to which the hyperparameters are adjusted over time. Alpha value of 1 produced the best results in terms of accuracy.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Results optimised for: | Best relevant score  on training set | Best relevant score on test set | Activation function | alpha | Hidden layer sizes | Learning rate | Solver |
| Accuracy | Accuracy  0.812 +/- 0.040 | Accuracy  0.74405 | tanh | 1 | (100,100,100) | constant | adam |
| Precision | Precision  0.827 +/- 0.048 | Precision  0.74405 | ReLu | 0.0001 | (50,100,50) | constant | adam |
| Recall | Recall  0.827 +/- 0.048 | Recall  0.74405 | tanh | 0.0001 | (50,100,50) | constant | sgd |
| F1 score | F1 score  0.812 +/- 0.040 | F1 score  0.74405 | tanh | 1 | (100,100,100) | constant | adam |
| AUC | AUC  0.786 +/- 0.050 | AUC  0.73810 | ReLu | 1 | (100,50,100) | constant | adam |

**Table 5**. MLP hyperparameter values optimised for the best accuracy, precision, recall, f1 score and AUC.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Classifier** |  | | **Actual values** | |
| **SVM** | **Predicted values** |  | **Benign (0)** | **Malignant (1)** |
| **Benign (0)** | 65 | 21 |
| **Malignant (1)** | 20 | 62 |
| **MLP** |  | **Benign (0)** | **Malignant (1)** |
| **Benign (0)** | 65 | 21 |
| **Malignant (1)** | 22 | 60 |

**Table 3**. Confusion matrixes for MLP Classifier and SVM Classifier.

The area under the ROC curve shows the overall performance of a classifier. The greater the area under curve (AUC) measure the better the classifier, ideally the ROC curve should fall as close as possible to the top left corner (James et al., 2013). The ROC curve takes into account the entire range of different thresholds and hence is a considered a good measure of the classifier’s performance. In this instance both the SVC and MLP both perform with AUC values of 0.81 and 0.79 respectively. The SVM Classifier is slightly superior.

There are several studies that have applied various machine learning algorithms to this mammographic mass dataset. Mokhtar et al. (2013) applied Decision Tree, Artificial Neural Network, and Support Vector Machine algorithms. Keles et al. (2011) used neuro-fuzzy rules in developing an expert system for predicting breast cancer using the same dataset. Elsayad (2010) evaluated the performance of two Bayesian Networks, the tree augmented Naïve Bayes (TAN) and Markov blanket estimation (MBE) as a classifier on this network and compared the outcome to multi-layered neural network classifier.

Could use convoluten neural network and a computer with GPU, would also look at the effect of using different learning rates and momentum. And using the poly kernel fro SVM

**References:**

American Cancer Society. 2015. Breast Cancer Facts & Figures 2015-2016. [Online]. [Accessed 29 March 2019]. Available from: <https://www.cancer.org/content/dam/cancer-org/research/cancer-facts-and-statistics/breast-cancer-facts-and-figures/breast-cancer-facts-and-figures-2015-2016.pdf>

American College of Radiology (ACR). 2013. BI-RADS Atlas — Reporting System. [Online]. [Accessed 29 March 2019]. Available from: <https://www.acr.org/-/media/ACR/Files/RADS/BI-RADS/Mammography-Reporting.pdf>

Azur, M.J. et al., 2011. Multiple imputation by chained equations: what is it and how does it work? *International journal of methods in psychiatric research,* 20(1), pp.40–9.

Elsayad, A.M., 2010. Predicting the severity of breast masses using Bayesian networks. *2010 The 7th International Conference on Informatics and Systems (INFOS)*, pp.1–9.

Fung, Germaine et al., 2015. Distinguishing bipolar and major depressive disorders by brain structural morphometry: a pilot study. *BMC Psychiatry*, 15(298), p.298.

Grus, J. 2015. *Data science from scratch : first principles with python*, Sebastopol: O'Reilly Media.

Halder, S. & Ozdemir, S. 2018. *Hands-On Machine Learning for Cybersecurity.* Birmingham: Packt Publishing

Hastie, T. et al. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* Second. New York: Springer.

Keleş, A., Keleş, A., Yavuz, U. 2011. Expert system based on neuro-fuzzy rules for diagnosis breast cancer. *Expert Systems With Applications*, 38(5), pp.5719–5726.

Kopans, D.B., 1992. The positive predictive value of mammography. *AJR. American journal of roentgenology*, 158(3), pp.521–6.

Mahmood, H. 2018. Activation Functions in Neural Networks. [Online]. [Accessed 04 April 2019]. Available from: <https://towardsdatascience.com/activation-functions-in-neural-networks-83ff7f46a6bd>

Narkhede, S. 2018. *Understanding Confusion Matrix.* [Online]. [Accessed 04 April 2019]. Available from: <https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62>

James, G., Witten, D., Hastie, T., Tibshirani, R. 2013. *An introduction to statistical learning : with applications in R.* London: Springer.

Jain, R. 2017. *Simple Tutorial on SVM and Parameter Tuning in Python and R*. [Online]. [Accessed 04 April 2019]. Available from: <https://www.hackerearth.com/blog/machine-learning/simple-tutorial-svm-parameter-tuning-python-r/>

Mendelson, E.B. 2019. Artificial Intelligence in Breast Imaging: Potentials and Limitations. *American journal of roentgenology*, 212(2), pp.293–299.

Mokhtar, S. & Elsayad, A. 2013. Predicting the Severity of Breast Masses with Data Mining Methods. *International Journal of Computer Science Issues (IJCSI)*, 10(2 Part 2), pp.160–168.

National Institute for Health and Care Excellence (NICE). 2017. Familial breast cancer: classification, care and managing breast cancer and related risks in people with a family history of breast cancer (update). [CG164]. London: National Institute for Health and Care Excellence.

Patterson, J. & Gibson, A. 2018. *Getting started with deep learning*. Sebastopol: O'Reilly Media, Inc.

Pedersen, R. & Schoeberl, M., 2006. An Embedded Support Vector Machine. 2006 International Workshop on Intelligent Solutions in Embedded Systems, pp.1–11.

Public Health England (PHE). 2016. NHS Breast Screening Programme. Clinical guidance for breast cancer screening assessment. [NHSBSP publication number 49]. London: Public Health England.

Qiao, M. et al. 2018. Breast Tumor Classification Based on a Computerized Breast Imaging Reporting and Data System Feature System. *Journal of Ultrasound in Medicine*, 37(2), pp.403–415.

Raschka, S. & Mirjalili, V. 2017. *Python machine learning : machine learning and deep learning with Python, scikit-learn, and TensorFlow*. 2nd ed. Birmingham: Packt Publishing Ltd.

Thompson, A. 2015. The Global Burden of Cancer 2013. *JAMA Oncology*, 314(10), p.1078.

WHO. 2018. Cancer: Key facts. [Online]. [Accessed 29 March 2019]. Available from: <https://www.who.int/news-room/fact-sheets/detail/cancer>

Zhang, Yan-Jun et al. 2013. Status quo and development trend of breast biopsy technology. *Gland surgery*, 2(1), pp.15–24.

**Data Set Information:**

Mammography is the most effective method for breast cancer screening   
available today. However, the low positive predictive value of breast   
biopsy resulting from mammogram interpretation leads to approximately   
70% unnecessary biopsies with benign outcomes. To reduce the high   
number of unnecessary breast biopsies, several computer-aided diagnosis   
(CAD) systems have been proposed in the last years.These systems   
help physicians in their decision to perform a breast biopsy on a suspicious   
lesion seen in a mammogram or to perform a short term follow-up   
examination instead.   
This data set can be used to predict the severity (benign or malignant)   
of a mammographic mass lesion from BI-RADS attributes and the patient's age.   
It contains a BI-RADS assessment, the patient's age and three BI-RADS attributes   
together with the ground truth (the severity field) for 516 benign and   
445 malignant masses that have been identified on full field digital mammograms   
collected at the Institute of Radiology of the   
University Erlangen-Nuremberg between 2003 and 2006.   
Each instance has an associated BI-RADS assessment ranging from 1 (definitely benign)   
to 5 (highly suggestive of malignancy) assigned in a double-review process by   
physicians. Assuming that all cases with BI-RADS assessments greater or equal   
a given value (varying from 1 to 5), are malignant and the other cases benign,   
sensitivities and associated specificities can be calculated. These can be an   
indication of how well a CAD system performs compared to the radiologists.   
  
Class Distribution: benign: 516; malignant: 445

**Attribute Information:**

6 Attributes in total (1 goal field, 1 non-predictive, 4 predictive attributes)   
  
1. BI-RADS assessment: 1 to 5 (ordinal, non-predictive!)   
2. Age: patient's age in years (integer)   
3. Shape: mass shape: round=1 oval=2 lobular=3 irregular=4 (nominal)   
4. Margin: mass margin: circumscribed=1 microlobulated=2 obscured=3 ill-defined=4 spiculated=5 (nominal)   
5. Density: mass density high=1 iso=2 low=3 fat-containing=4 (ordinal)   
6. Severity: benign=0 or malignant=1 (binominal, goal field!)   
  
  
Missing Attribute Values:   
- BI-RADS assessment: 2   
- Age: 5   
- Shape: 31   
- Margin: 48   
- Density: 76   
- Severity: 0

Histogram, boxplot, calc std mean, quantiles, outliers.

How do other studies compare.

<https://towardsdatascience.com/handling-missing-values-in-machine-learning-part-2-222154b4b58e>

The values of ordinal attribute represent categories with some intrinsic ranking while they nominal attribute represent categories with no intrinsic ranking in nominal type.

Outliers, no invalid outliers are identified.

Would have needed to use the labelencoder, however, the lablels have already been assigned in the dataset. Lable encoder assisgn

One hot encoder performs binarization of the categorical feature values;This is why we use one hot encoder to perform “binarization” of the category and include it as a feature to train the model.

One hot encoder converts the categorical features to different features which are all binary ‘is\_daffodil’, ‘is\_lily’, and ‘is\_rose’ which all are binary.

<https://towardsdatascience.com/grid-search-for-model-tuning-3319b259367e>

<https://github.com/rohanjoseph93/Python-for-data-science/blob/master/Grid%20Search%20-%20Breast%20Cancer.ipynb>

<http://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf>

SVM - kernel needs to be polynomial as we are dealing with several features.

<https://github.com/matthewbrems/ODSC-missing-data-may-18/blob/master/Analysis%20with%20Missing%20Data.pdf>

<https://notebooks.azure.com/holgerkunz/projects/UCLMHLBFOUR/html/MLHB_week4_data_preprocessing_with_solution.ipynb>

<http://www.wildml.com/2015/09/implementing-a-neural-network-from-scratch/>

Performing one-hot encoding on nominal features

One hot encoder <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html>

Lable encoder assisgn

One hot encoder performs binarization of the categorical feature values;This is why we use one hot encoder to perform “binarization” of the category and include it as a feature to train the model.

One hot encoder converts the categorical features to different features which are all binary ‘is\_daffodil’, ‘is\_lily’, and ‘is\_rose’ which all are binary.

<https://hackernoon.com/what-is-one-hot-encoding-why-and-when-do-you-have-to-use-it-e3c6186d008f>

<https://towardsdatascience.com/how-to-handle-missing-data-8646b18db0d4>

You can use [https://pypi.python.org/pypi/fancyimpute/0.0.4 1.2k](https://pypi.python.org/pypi/fancyimpute/0.0.4) and <http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Imputer.html>

<https://stackoverflow.com/questions/25239958/impute-categorical-missing-values-in-scikit-learn>

<https://www.statsmodels.org/dev/generated/statsmodels.imputation.mice.MICEData.html>

from fancyimpute import MICE

#We use the train dataframe from Titanic dataset

*#fancy impute removes column names.*  
train\_cols = list(train)

# Use MICE to fill in each row's missing features

train = pd.DataFrame(MICE(verbose=False).complete(train))  
train.columns = train\_cols

Installing fancyimpute

<https://stackoverflow.com/questions/50217492/unable-to-install-fancyimpute-for-use-in-jupyter>

<https://stackoverflow.com/questions/44239269/fancyimpute-installation-in-anaconda>

Zhang, Y. and Ren, H. 2017. Meta-analysis of diagnostic accuracy of magnetic resonance imaging and mammography for breast cancer. *Journal of cancer research and therapeutics*, 13(5), pp.862–868.

Shetty, M. 2010. Screening for Breast Cancer with Mammography: Current Status and An Overview. *Indian Journal of Surgical Oncology*, 1(3), pp.218–223.

Gard, C.C. et al. 2015. Misclassification of Breast Imaging Reporting and Data System (BI‐RADS) Mammographic Density and Implications for Breast Density Reporting Legislation. *Breast Journal*, 21(5), pp.481–489.

<https://ieeexplore-ieee-org.libproxy.ucl.ac.uk/document/5461768>

<https://www.acr.org/-/media/ACR/Files/RADS/BI-RADS/Mammography-Reporting.pdf>

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3244246/>

<https://onlinelibrary-wiley-com.libproxy.ucl.ac.uk/doi/full/10.1111/j.1524-4741.2010.00992.x>

<https://jamanetwork-com.libproxy.ucl.ac.uk/journals/jama/fullarticle/181879>

**https://arxiv.org/ftp/arxiv/papers/1305/1305.7057.pdf**

[**https://www.cochrane.org/CD001877/BREASTCA\_screening-for-breast-cancer-with-mammography**](https://www.cochrane.org/CD001877/BREASTCA_screening-for-breast-cancer-with-mammography)

Løberg, M. et al., 2015. Benefits and harms of mammography screening. *Breast Cancer Research : BCR*, 17(1), p.63.

<https://www.sciencedirect.com/science/article/abs/pii/S0950061811001942>

<https://core.ac.uk/download/pdf/26831826.pdf>

<https://github.com/janishar/mit-deep-learning-book-pdf/blob/master/complete-book-pdf/deeplearningbook.pdf>

<https://stackoverflow.com/questions/50670080/mice-implementation-in-python>

MICE operates under the assumption that given the variables used in the imputation procedure, the missing data are Missing At Random (MAR), which means that the probability that a value is missing depends only on observed values and not on unobserved values ([Schafer & Graham, 2002](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3074241/#R30)). In other words, after controlling for all of the available data (i.e., the variables included in the imputation model) “any remaining missingness is completely random” ([Graham, 2009](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3074241/#R8)).  (Azur)

<https://github.com/matthewbrems/ODSC-missing-data-may-18/blob/master/Analysis%20with%20Missing%20Data.pdf>

Neural networks <http://cs231n.github.io/>

# 30 years of adaptive neural networks: perceptron, Madaline, and backpropagation

Pseudocode svm

Neural networks

SVM

Assessment

Artificial Neural network

Backpropagation

Gradient decent

Learning rate

Weights to start with

Good article on ANN

gamma to specify a value of γ for the radial basis kernel. By increasing γ we can produce a more flexible fit and generate further improvements in accuracy. γ is a positive constant.

Hyperparameters C and gamma, which controls a tradeoff between allowed training errors and misclassifications, and the width of the radial basis function, were tuned using a 10-fold cross-validation approach. The optimized parameters that provide the best accuracy would be selected for the final model.

In the present study, the classifier’s performance is evaluated using the common leave-one-out-each-group cross-validation approach. This validation procedure provides robust parameter estimates particularly for smaller samples [[22](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4655080/#CR22)]. In each trial observation, one patient per group was left out from the data to train the classifier, but then used to determine the detection rate of this trained classifier (testing). The procedure was repeated until every participant had been used for testing a classifier. The overall accuracy of the classifier was the averaged detection rate. The sensitivity and specificity of the classifier were also quantified. Specifically, sensitivity was calculated by the number of true BD dividing by the total number of true BD and those misclassified BD as MDD. Specificity was calculated by the number of true MDD dividing by the total number of true MDD and those misclassified MDD as BD. To evaluate the probability of obtaining the overall accuracy by chance, statistical significance was verified by means of permutation tests [[24](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4655080/#CR24)]. We randomly assigned a class label to each patient and repeated the same cross-validation procedures for 1000 times. Then we counted the total number of times that the detection rates from the permutation tests were higher than or equal to the actual value obtained from the real test. A p-value for classification is derived from dividing this number by 1000. The classifications were performed using R version

It follows a technique called the kernel trick to transform the data and based on these transformations, it finds an optimal boundary between the possible outputs. identify the optimal separating hyperplane which maximizes the margin of the training data. Hyperplane is a n-1 dimensional subspace of an n-dimensional Euclidean space. We have said that the objective of an SVM is to find the optimal separating hyperplane. Therefore, maximizing the distance between the nearest points of each class and the hyperplane would result in an optimal separating hyperplane. This distance is called the margin.

The goal of SVMs is to find the optimal hyperplane because it not only classifies the existing dataset but also helps predict the class of the unseen data. And the optimal hyperplane is the one which has the biggest margin

Here, the parameter C is the **regularization parameter** that controls the trade-off between the slack variable penalty (misclassifications) and width of the margin.

* Small C makes the constraints easy to ignore which leads to a large margin.
* Large C allows the constraints hard to be ignored which leads to a small margin.
* For C=inf, all the constraints are enforced.

Support Vector Machines handle such situations by using a kernel function which maps the data to a different space where a linear hyperplane can be used to separate classes. This is known as the **kernel trick** where the kernel function transforms the data into the higher dimensional feature space so that a linear separation is possible.

Parameters are as follows:

* C: It is the regularization parameter, C, of the error term.
* kernel: It specifies the kernel type to be used in the algorithm. It can be ‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’, or a callable. The default value is ‘rbf’.
* degree: It is the degree of the polynomial kernel function (‘poly’) and is ignored by all other kernels. The default value is 3.
* gamma: It is the kernel coefficient for ‘rbf’, ‘poly’, and ‘sigmoid’. If gamma is ‘auto’, then 1/n\_features will be used instead.

SVM classifier node is used to train SVM with polynomial kernel. There are four parameters that need to be optimized; C, γ , r and d. The value of the regularization parameter C should be set between 1 and 10 inclusive; increasing the value improves the prediction accuracy for the training data, but this can also lead to overfitting. Using trial and error we found that the best values for these parameters are 10, 1, 0.1 and 4 for C, γ , r and d respectively. These values result in 83.66% and 81.25% prediction accuracies for training and test subsets respectively. It takes only 35 seconds to build the model.

SVM is a supervised multivariate classification algorithm (Fung, 2015). SVM is trained on a subset of the dataset, it works by follows a technique called the kernel trick to transform the data and based on these transformations, it finds an optimal decision boundary between the possible outputs of benign and malignant. Kernel function can be used to compute the dot products in the higher-dimensional space and use those to find a hyperplane (Grus).

Support Vector Machines handle such situations by using a kernel function which maps the data to a different space where a linear hyperplane can be used to separate classes. This is known as the kernel trick where the kernel function transforms the data into the higher dimensional feature space so that a linear separation is possible Join, 2017). The goal of SVMs is to find the optimal hyperplane because it not only classifies the existing dataset but also helps predict the class of the unseen data. Hyperplane is an n-1 dimensional subspace of an n-dimensional Euclidean space(Jain,2017). The optimal separating hyperplane separates the two classes and maximizes the distance to the closest point from either class, the hyperplane would result in an optimal separating hyperplane this leads to better classification performance on test data.. The hyperplane is built by a set of support vectors, the algorithm is applied on the test data and using the predicted values it determines which side of the hyperplane they fall and suggesting a class label (Fung, 2015, Jain, 2017).

Four parameters can regulated in order to optimize the SVM algorithm. C is the regularization parameter and is usually set between 1 and 10, increasing the value improves the prediction accuracy for the training data, but this can also lead to over-fitting. Small C value ignores majority of the constraints and has a very large margin, larger C value makes the constraints harder to ignore thereby the margin is also smaller. γ is a positive constant, it is the kernel coefficient by increasing γ the flexibility of the model increases and this can help improve accuracy, however, C and γ a regulate the trade-off between allowed training errors and misclassifications. The optimized parameters that provide the best accuracy would be selected for the final model. d is a positive integer, it is the degree of the polynomial kernel function (‘poly’) and is ignored by all other kernels.. Using such a kernel with d > 1, instead of the standard linear kernel, in the support vector classifier algorithm leads to a much more flexible decision boundary. It essentially amounts to fitting a support vector classifier in a higher-dimensional space involving polynomials of degree d, rather than in the original feature space .

**Algorithm 2 (Pedersen, 2006)**

Training an SVM

**Require:** X and y loaded with training labelled data, α ⇐ 0 or α ⇐ partially trained SVM

1: C ⇐ some value (10 for example)

2: **repeat**

3: **for all** {xi, yi}, {xj, yj} **do**

4: Optimize αi and αj

5: **end for**

6: **until** no changes in α or other resource constraint criteria met

**Ensure:** Retain only the support vectors (αi > 0)

SVM can be tuned by using GridSearch that finds the most optimal hyperparameters

#### Support vector machine (SVM)

SVM is a binary linear classifier that is commonly used to solve classification problems. More detail description of SVM can be found elsewhere [[23](https://link-springer-com.libproxy.ucl.ac.uk/article/10.1007%2Fs11255-016-1346-4#CR23), [24](https://link-springer-com.libproxy.ucl.ac.uk/article/10.1007%2Fs11255-016-1346-4#CR24)]. In brief, SVM algorithm is to find the optimal hyperplane that can separate all objects of one class from those of the other class with the largest margin between the two classes. The largest margin can reach the maximal width of the slab parallel to the hyperplane while there is no interior data point in this region. The objects located far from the boundary are removed from the calculation while those data points on the boundary of the slab will be maintained and determined as the so-called support vectors to obtain satisfactory computational efficiency.

#### Soft independent modeling of class analogy (SIMCA)

The SIMCA classifier is a well-known pattern recognition method and a classical quadratic discriminant analysis, which focuses on the similarity within a class and describes each class separately in a principal components model [[21](https://link-springer-com.libproxy.ucl.ac.uk/article/10.1007%2Fs11255-016-1346-4#CR21), [25](https://link-springer-com.libproxy.ucl.ac.uk/article/10.1007%2Fs11255-016-1346-4#CR25)]. When a blind unknown is presented, it will be projected into each principal components space by applying principal components analysis (PCA). Then, SIMCA determines membership based on the distance of a new object from eigenvectors of each class of training set. The closest set of eigenvectors assigns the class that the object belongs to.

### Cross-validation

Reliable, robust model is important for applications. In this work, the performance of each multivariate classifier was evaluated reliably and unbiasedly based on bootstraps Latin partition cross-validation, which provides a systematic approach to classifier evaluation and measurement of precision [[26](https://link-springer-com.libproxy.ucl.ac.uk/article/10.1007%2Fs11255-016-1346-4#CR26)].

When using Latin partition, data are randomly divided by sample into training and prediction sets so that the replicates from same sample will not be included in the training and prediction sets. The proportions of the number of samples for each class are maintained between the prediction set and training set, and thus the distribution of objects in training and prediction set is similar to the un-partitioned data set. Several training-prediction set pairs are given so that each sample is used once and only once for prediction. Because every object is used for prediction, Latin partition makes efficient use of the data, eliminates the bias of using only a subset of well-behaved prediction objects, and the results can be averaged across the bootstraps and the precision of the average obtained by its standard deviation.

In this work, all generations of composite data sets, multivariate models building, and bootstrapped evaluations were implemented by homebuilt codes in MATLAB software (MathWorks, Natick, MA).

Close

S. Charbonnier, L. Zoubek, S.Lesecq, F. Chapotot**Self-evaluated automatic classifier as a decision-support tool for sleep/wake staging**

Comput. Biol. Med., 41 (6) (2011), pp. 380-389

[Article](https://www-sciencedirect-com.libproxy.ucl.ac.uk/science/article/pii/S0010482511000606)[Download PDF](https://www-sciencedirect-com.libproxy.ucl.ac.uk/science/article/pii/S0010482511000606/pdfft?md5=c0243753ff34bd9ef3bb467f550d98e1&pid=1-s2.0-S0010482511000606-main.pdf)[View Record in Scopus](https://www.scopus.com/inward/record.url?eid=2-s2.0-79956334572&partnerID=10&rel=R3.0.0)[Google Scholar](https://scholar.google.com/scholar_lookup?title=Self-evaluated%20automatic%20classifier%20as%20a%20decision-support%20tool%20for%20sleep%2Fwake%20staging&publication_year=2011&author=S.%20Charbonnier&author=L.%20Zoubek&author=S.%20Lesecq&author=F.%20Chapotot)

<http://web.mit.edu/6.034/wwwbob/svm-notes-long-08.pdf>

<https://www.sciencedirect.com/science/article/pii/S0966636208002026#app1>

Neural network explanation/pseudocode

<https://www.cs.sfu.ca/~mori/courses/cmpt882/fall05/slides/chapter20b.pdf>

good overview of SVM

very good overview of svm

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2850872/>

Goodfellow deep learning

<https://onlinelibrary-wiley-com.libproxy.ucl.ac.uk/doi/pdf/10.1002/9780470503065.ch7>