# Comparing Decision Trees and Logistic Regression in Breast Cancer Diagnosis

## Introduction:

Breast cancer is one of the most common types of cancer diagnosed in women. When breast cancer is suspected, the examination begins with medical imaging, such as mammography and ultrasound. However, the definitive diagnosis can only be done via biopsy. In biopsy, a small amount of tissue is extracted from the suspected tumor. Then the biopsy sample is analyzed in a laboratory, and experts determine if the cells in the sample are cancerous. In fine needle aspiration (FNA) biopsy, the sample is extracted with a thin, hollow needle.

In this report I am comparing two different machine learning methods in breast cancer diagnosis. Section two discusses the problem formulation. Section three discusses the used methods. Then in sections 4 and 5 I present the results and conclusions of the tests performed. The references are listed in the last section.

## Problem Formulation:

The diagnosis of breast cancer is based on the characteristics of the cells in the biopsy sample. The process relies on the expertise and skill of the pathologist examining the sample, and there are many things to consider when determining if the cell is from a cancer tumor. Machine learning could be used to make the diagnosing process faster, more efficient, and more reliable.

## Method:

I used the Breast Cancer Wisconsin (diagnostic) dataset, which is a famous dataset commonly used for classification problems and studying different machine learning methods. The dataset is available in the Python machine learning library scikit-learn, in the sklearn.datasets package. The dataset in the package is a copy of UCI ML Repository Breast Cancer Wisconsin (Diagnostic) datasets (https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)). As the Breast Cancer Wisconsin dataset is a well-known dataset for machine learning, many different implementations can be found on the internet. For this project, I browsed some of them to get an overall idea of what methods could be useful for this kind of data.

In the dataset the datapoints represent FNA biopsy samples of breast masses. The features are numerical and have been analyzed from images of the biopsy samples. They represent different characteristics of the nuclei of the cells in the image. The characteristics are:

1) radius (mean of distances from center to points on the perimeter)
2) texture (standard deviation of gray-scale values)
3) perimeter
4) area
5) smoothness (local variation in radius lengths)
6) compactness (perimeter^2 / area - 1.0)
7) concavity (severity of concave portions of the contour)
8) concave points (number of concave portions of the contour)

9) symmetry
10) fractal dimension ("coastline approximation" - 1)

(Source: scikit-learn.org)

For each image, the mean, standard error, and worst/largest (mean of the three worst or largest values in the image) of these 10 characteristics were computed, so for each datapoint there are 30 features. In the scikit-learn dataset, the datapoint labels are binary, with 0 representing malignant (cancerous) and 1 benign (not cancerous) tissue. For clarity, I changed the labels so that 1 represents positive cancer diagnosis. **In this project, label 1 represents cancerous and label 0 represents non-cancerous tissue.** The total number of datapoints in this dataset is 569: 357 of which are labeled benign and the remaining 212 malignant. There are no missing values.

As the dataset is embedded in the scikit-learn library and I already had some experience with Python, I decided to make a Python notebook. The dataset is multivariate with 0/1 binary labels, and therefore it is reasonable to treat it as a classification problem. I decided to compare two classification models: decision trees (chapter 3.10 of mlbook.cs.aalto.fi) and logistic regression (ch. 3.6 of mlbook.cs.aalto.fi).

For decision tree models, I decided to use the 0-1 loss. In the zero_one_loss function in sklearn.metrics module, the loss is defined as

$$L_{0\text{-}1}(y_{\mathrm{i}},\, \hat{y}_{\mathrm{i}}) \;=\; 1(\,\hat{y}_{\mathrm{i}} \neq y_{\mathrm{i}})$$

Where $1(\mathrm{x})$ equals 1 if the predicted value is the same as the actual value, and 0 if the predicted value is not the same as the actual value. The function returns the percentage of values that are predicted wrong. I used 0-1 loss for both training and validation sets.

Logistic regression uses logistic loss. I used the function log_loss in scikit-learn, which is defined as

$$L_{\log}(y,p) \;=\; -(y \log{(p)} \;+\; (1 \;-\; y) \log{(1 \;-\; p)})$$

for a sample with a true label $\mathrm{y} = 0{,}1$. I used this loss function for both training and validation errors.

In addition to training and validation error, I also used the F1-score to evaluate model performance. F1-score is defined as $2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$, where $\text{Precision} \;=\; \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$ and $\text{Recall} \;=\; \frac{\text{True Positive}}{\text{True Positive} + \text{False negative}}$. I chose the F1-score instead of accuracy (number of correct predictions/total number of predictions) because in the case of cancer diagnosis, F1-score can be a better measurement than accuracy, because it minimizes false negatives and false positives better. We don’t want to fail to diagnose a patient who has cancer, nor give a false cancer diagnosis to a patient who is actually healthy.

I began the project with loading the data from sklearn.datasets. As I looked at the data, I noticed that the feature values range from almost zero to thousands. To achieve better performance on the models, I normalized the data with min-max scaling. After the normalization, the feature values range between 0 and 1. Then I split the data into training, validation, and test sets. I used the train_test_split function, which shuffles the data before splitting. I used the ratio of 60%, 20% and 20% for the training, validation, and test sets respectively. I used this ratio, because the training set should be larger than validation set, and this ratio is commonly used in machine learning problems. I left the test set to wait for the final evaluation of the chosen model.

Decision trees:

I used a scikit-learn decision tree function with Gini impurity as the function which measures the quality of a split. I used 7 models with different maximum tree depths d = 1, 2, 3, 4, 5, 6 and 7.

| d value | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Training | 0.0645 | 0.0352 | 0.0235 | 0.00293 | 0 | 0 | 0 |
| Validation | 0.0789 | 0.0614 | 0.0614 | 0.0702 | 0.0526 | 0.0526 | 0.0526 |
| F1 | 0.883 | 0.916 | 0.914 | 0.905 | 0.930 | 0.930 | 0.930 |

Table 1: the training errors, validation errors and f1 scores for the decision tree models.

After the tree depth (longest path from the root to a leaf) is 5 or bigger, the training error becomes 0. This would imply that the models are overfitting. The model should be chosen from the first four trees, based on the smallest validation error. The validation error is the same for d = 2 and d = 3. I chose d = 2, because the training error value is closer to the validation error value, and the value of F1 is larger.

Logistic regression:

For the logistic regression part, I used the scikit-learn logistic regression function. The results can be seen in the following table.

| Training error | Validation error | F1-score |
|---|---|---|
| 1.2154 | 1.2119 | 0.951 |

Table 2: training error, validation error and F1-score for logistic regression.

The training error is slightly larger than the validation error. This would suggest that the model is underfitting. To improve the model, I decided to try k-fold cross validation to find better parameters for the logistic regression function. I used k-fold cross validation with k=10 (as defined in chapter 6.2 of mlbook.cs.aalto.fi) and obtained the following results:

| Training error | Validation error | F1-score |
|---|---|---|
| 0.2026 | 0.9089 | 0.964 |

Table 2: training error, validation error and F1-score for logistic regression with k-fold cross validation.

The training error is smaller, and the F1-score is better than for the previous logistic regression model. However, the validation error is still quite large. When compared to the training error, the training error is approximately 70 % smaller than the validation error. This would suggest that the model might be overfitting.

**Results**:

As the final model, I chose the decision tree with maximum depth = 2, because the validation error was quite small (for 0-1 loss, the biggest possible loss is 1). I evaluated the performance of this chosen model with the test set and calculated the 0-1 loss. The resulting test error is 0.105, which is a little larger than the validation error 0.0614. The F1-score was 0.846, which was lower than the score measured with validation set. The chosen depth of 2 prevented serious overfitting, but the model might have missed some important features.

The models I used tended to overfit. To prevent this phenomenon, I could have done feature selection. For each datapoint there were a total of 30 features, which may have had correlation with each other, or little to no correlation at all to the label values. The dataset is also rather small, and there were more datapoints labeled “benign” than “malignant”. Although I split the training, testing and validation sets randomly, it is possible that there was an imbalance between malignant and benign datapoints in some of the sets.

**Conclusion:**

Based on the test error and F1-score, there could be room for improvement. When diagnosing cancer, we want the accuracy of the predictions to be close to 100%. As there are already several problems on the internet (for example Kaggle.com) about this dataset, I would consider collecting more data. However, I think that this dataset is very suitable for beginner-level machine learning projects like this. The datapoints in this dataset are numerical values calculated from images, so the next step could be image classification.

**References**

W.N. Street, W.H. Wolberg and O.L. Mangasarian
Nuclear feature extraction for breast tumor diagnosis.
IS&T/SPIE 1993 International Symposium on Electronic Imaging: Science
and Technology, volume 1905, pages 861-870, San Jose, CA, 1993.

F. Pedregosa, et.al, “Scikit-learn: Machine Learning in Python”, Journal of Machine Learning Research, vol. 12, p. 2825--2830, 2011

Cancer.org, Fine Needle Aspiration (FNA) Biopsy of the Breast (read 20.3.2021)

Kaggle.com/uciml/breast-cancer-wisconsin-data (read 20.3.2021)

Mayoclinic.org, Breast cancer (read 27.3.2021)