FYS-STK4155: Project 3

Camilla Dalby Borger

December 2023

Abstract

This study aims to investigate the mechanisms underlying human visual memory recall within the realm of cognitive neuroscience. Utilizing machine learning techniques, specifically logistic regression, classification trees, and neural networks, our goal is to predict whether a patient has encountered a particular image previously. Our methodology involves employing basic feature extraction methods such as Fourier transformations alongside essential statistical measures like mean values and standard deviation derived from the data.

In our pursuit of precise predictions, decision trees are adopted for their capacity to handle complex datasets effectively, thereby enhancing prediction accuracy. Through the integration of these diverse methodologies, our objective is to unravel the intricate relationship between extracted features and the brain's capacity to recollect visual stimuli. The study's outcomes hold promise for advancing our comprehension of neural mechanisms involved in visual memory recall, laying the groundwork for the development of robust predictive models in clinical settings.

Introduction

Visual memory recall mechanisms in the human brain pose intriguing challenges in neuroscience. In this project, our aim is to employ diverse supervised machine-learning methods to predict whether a patient has encountered an image previously. This investigation stems from a recent patient study conducted at Oslo University Hospital, yielding an exceptional dataset conducive to probing direct measurements of neural activity within the human brain.

The study involves monitoring neural activity in patients diagnosed with epilepsy, utilizing an array of hundreds of implanted electrodes. These electrodes offer unparalleled advantages, presenting superior signal-to-noise ratios and an impressive temporal resolution (kHz) in comparison to conventional methods such as functional Magnetic Resonance Imaging (fMRI). During the study, patients were exposed to thousands of natural images as part of a memory task while their neural activity was recorded, capturing rapid changes in brain activity.

Our primary objective is to leverage these neural recordings to predict if a patient has previously encountered a specific image. It's important to note that the response obtained from the patient may not align with the "true" response, defined as whether the image was genuinely shown to the patient before or not. Instead, we will utilize the patient's response as the response variable in our predictive modeling.

This project aims to explore the complexities of neural activity and its relationship with visual memory recall. By harnessing advanced machine learning techniques and leveraging the unique dataset obtained from direct neural measurements, we strive to uncover the intricate mechanisms underlying the brain's ability to recall visual stimuli. This scientific report presents our methodology, analysis, and findings in pursuit of understanding and predicting visual memory recall in patients based on neural recordings.

Methods

This project primarily deals with a classification task aimed at predicting the patient's response. Our approach involves extracting features from the data to be utilized in machine learning methods. Additionally, we'll be directly using the raw data to evaluate the performance of various models.

The project centers around a specific patient named "2022-11." This patient viewed 842 out of 1000 images, with 339 images viewed thrice and 631 viewed twice. Our focus will be solely on images viewed two or three times. To analyze the data, we'll employ different design matrices.

Design Matrix

The design matrix utilized in this project comprises raw data. Specifically, the design matrix solely comprised of raw data possesses dimensions of $(296940, 2048)$ for the images shown twice and $(235935, 2048)$ for those shown three times. This matrix is denoted as:

$$\begin{bmatrix} x_{0,0} & x_{0,1} & x_{0,2} & \dots & x_{0,s} & \dots & x_{0,S} \\ x_{1,0} & x_{1,1} & x_{1,2} & \dots & x_{1,s} & \dots & x_{1,S} \\ x_{2,0} & x_{2,1} & x_{2,2} & \dots & x_{2,s} & \dots & x_{2,S} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ x_{c,0} & x_{c,1} & x_{c,2} & \dots & x_{c,s} & \dots & x_{c,S} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ x_{C,0} & x_{C,1} & x_{C,2} & \dots & x_{C,s} & \dots & x_{C,S} \end{bmatrix} \tag{1}$$

In this context, S represents the total number of samples, while C stands for the overall number of channels. As we consolidate all trials into a single design matrix, C equates to the initial number of channels (245) multiplied by the count of image IDs displayed either two or three times.

The feature matrices take on distinct shapes for images viewed two and three times, having dimensions $(296940, 8)$ and $(235935, 8)$ respectively. These matrices can be represented as:

$$\begin{bmatrix} z_{0,0} & z_{0,1} & \dots & z_{0,f} & \dots & z_{0,F} \\ z_{1,0} & z_{1,1} & \dots & z_{1,f} & \dots & z_{1,F} \\ z_{2,0} & z_{2,1} & \dots & z_{2,f} & \dots & z_{2,F} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ z_{c,0} & z_{c,1} & \dots & z_{c,f} & \dots & z_{c,F} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ z_{C,0} & z_{C,1} & \dots & z_{C,f} & \dots & z_{C,F} \end{bmatrix} \tag{2}$$

Here, F represents the number of features extracted from the raw data, and C corresponds to the total number of channels, as previously described.

In the subsequent section, we'll elaborate on the specific features we've extracted from the data.

Feature extraction

We will extract some features of the data to use in the different methods. This is to reduce the dimension of the data. The raw data for each trial (being the neural activity for each image shown, four seconds) is an array of shape $(245, 2048)$. Where 245 is the number of channels (or electrodes on probes in the brain), and 2048 is the sample. Each trial are done in four seconds, where the first three seconds is where the patient are shown the image and the last second is when the patient respond "new" or "old". There are 512 samples in each second, therefore we have 2048 samples for each trial in total.

We will use the different features describes below to reduce the dimension of the data from $(245, 2048)$ to $(245, 1)$ for each trial.

- **Min:** Taking the minimum value of each channel given one trial.

$$\min \{x_i\}_{i=1}^{2048} \tag{3}$$

- **Max:** Taking the maximum value of each channel given one trial.

$$\max \{x_i\}_{i=1}^{2048} \tag{4}$$

  where x_i is the value at sample i.

- **Mean:** Taking the mean of each channel given one trial.

$$\mu_{\text{trial},c} = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{5}$$

  where N is the sample number (time), and x_i is the brain signal at sample (time) i.

- **Standard deviation:** Taking the standard deviation of each channel given one trial.

$$\sigma_{\text{trial},c} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (x_i - \mu_{\text{trial},c})} \tag{6}$$

  where N is the sample number (time), and x_i is the brain signal at sample (time) i, and $\mu_{\text{trial},c}$ is the mean of the given trial trial and channel c.

- **Fourier Transform[1]:** The Fourier Transform, denoted by $\mathcal{F}$, of a discrete signal x_i of length N is defined as:

$$X_k = \sum_{i=0}^{N} x_i \cdot e^{-\frac{2\pi i}{N} ki} \tag{7}$$

  where N is the sample number (time), and x_i is the brain signal at sample (time) i.
  We are interesting in:

– The three first Fourier coefficients, X_1, X_2, X_3 of each channel given one trial.

– The largest Fourier coefficient, $\max[\{X_k\}_{k=1}^{K}]$, of each channel given one trial.

Logistic Regression[3]

Logistic regression was utilized as a predictive model. Logistic regression is a linear model that uses the sigmoid function defined as

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{8}$$

as its activation function and is well-suited for binary classification tasks.

The logistic regression model predicts the probability of an instance belonging to a particular class. The model's output is computed using the sigmoid function (8), where x is the linear combination of features weighted by coefficients.

The prediction for a sample with features $\mathbf{x}$ and coefficients $\mathbf{w}$ in logistic regression is given by:

$$\hat{y}(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x}) \tag{9}$$

where $\hat{y}(\mathbf{x})$ represents the predicted probability of the sample belonging to the positive class.

The activation function used in logistic regression is the sigmoid function (8) as motioned above. The function transforms the linear combination of features into a range between 0 and 1, interpreting the output as a probability.

Classification Trees[4]

A classification tree, akin to a regression tree[4], predicts a qualitative response rather than a quantitative one. In contrast to a regression tree, which predicts the mean response for observations within a terminal node, a classification tree predicts that each observation belongs to the most frequently occurring class among training observations in its corresponding region. When interpreting classification tree results, we're often interested in both the predicted class and the class proportions among the training observations falling into that region.

Growing a classification tree parallels growing a regression tree, the whole algorithm is presented below. Recursive binary splitting is utilized, but in classification, the Mean Squared Error (MSE) cannot serve as a splitting criterion. Instead, the **classification error rate** is a natural alternative. This error rate represents the fraction of training observations in a region not belonging to the most common class.

For evaluating split quality in a classification tree, the Gini index or entropy are commonly used since they are more sensitive to node purity than the classification error rate.

Suppose our targets involve $k = 1, 2, \ldots, K$ classes. The splitting criteria for each node are crucial.

We define a Probability Density Function p_{mk}, denoting the number of observations of class k in region R_m with N_m observations. This likelihood function, based on the proportion $I(y_i = k)$ of class observations in R_m, is represented as:

$$p_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k). \tag{10}$$

Here, p_{mk} signifies the majority class observations in region m. The three common splitting methods are:

- Misclassification error:

$$p_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i \neq k) = 1 - p_{mk}. \tag{11}$$

- Gini index g:

$$g = \sum_{k=1}^{K} p_{mk}(1 - p_{mk}). \tag{12}$$

- Information entropy or entropy s:

$$s = -\sum_{k=1}^{K} p_{mk} \log p_{mk}. \tag{13}$$

Algorithm

1. **Initialization:**
   - Start with the entire dataset D consisting of N observations and K classes.
   - Define the set of features $\{X_1, X_2, \ldots, X_p\}$ where p is the number of features.
2. **Splitting Criteria:**
   - Define a splitting criterion, such as the Gini index G or information entropy S, to evaluate the quality of splits at each node.
3. **Recursive Binary Splitting:**
   - Begin with the root node R_0 representing the entire dataset D.
   - At each node R_m, select the best feature X_j and its corresponding splitting value α that maximizes the splitting criterion C (e.g., Gini index or entropy).

$$C(R_m, X_j, \alpha) = \text{Criterion}(R_m, X_j, \alpha)$$

4. **Partitioning the Data:**
    - Partition the data based on the selected feature and splitting value:

$$R_{\text{left}} = \{(x, y) \in R_m \mid x_j \leq \alpha\}$$

$$R_{\text{right}} = \{(x, y) \in R_m \mid x_j > \alpha\}$$

5. **Stopping Criterion:**
    - Define a stopping criterion to halt the tree-growing process.
    - Common stopping criteria include reaching a maximum depth, minimum samples per node, or minimum samples for a split.
6. **Recursive Splitting:**
    - For each resulting subset R_{left} and R_{right}:
        - If the stopping criterion is not met, recursively apply steps 3 to 6 to split the node further.
        - If the stopping criterion is met, declare the node as a leaf node and assign the majority class or make further decisions based on specific rules.

Neural Network [2]

A feedforward neural network was employed as a predictive model in this project. The network architecture consisted of an input layer, one hidden layers with 50 hidden nodes, and an output layer. The cost function used was therefore the cross-entropy loss. The activation function used in the hidden layers was the sigmoid function as we are dealing with a simple neural network. The output layer employed also the sigmoid function.

Architecture

We used a feedforward neural network with three layers including input and output layer. The input layer consists of the size of the design matrix X (one trial given all channels) the hidden layer of fifty hidden nodes and the output layer consists of one node. The connections between layers were governed by weights $W^{(l)}$ and biases $b^{(l)}$ where l represent each layer. The input data was fed into the input layer, and the network's forward pass computed activations layer by layer until the final output was obtained.

Activation Functions

The activation function used within the hidden layers was the sigmoid function, defined as (8). The sigmoid function introduced non-linearity into the network and enabled the modeling of complex relationships within the data.

Cost Functions

When using the network as a classifier/predictior the cost funcion for a binary classification task is the cross-entropy cost function. This function is defined as:

$$\text{Binary Cross-Entropy} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \tag{14}$$

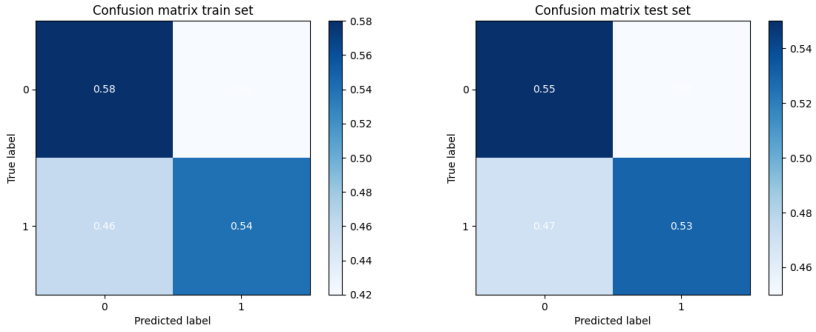
where N is the number of samples, y_i is the true label, and $\hat{y}_i$ is the predicted probability for the i-th sample.

Results

In the following section, we will present the results of our models used on brain data. The code used in this section is provided in:
`https://github.com/camilldb/FYS-STK4155/tree/master/Project3`

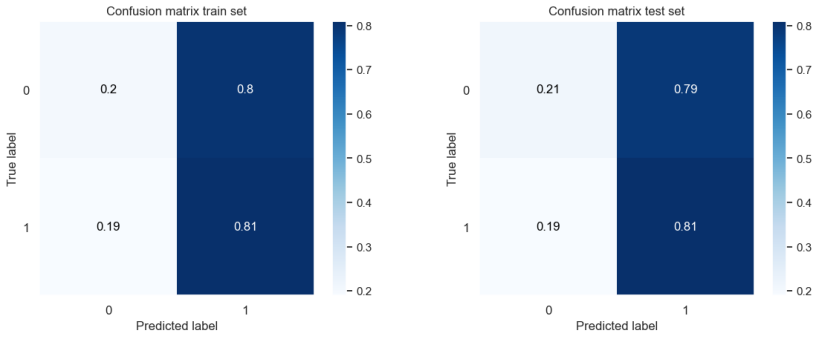
Let X_2 be the design matrix for the raw data where all images are shown two times, and X_3 where all images are shown three times. Let Z_2 be the feature matrix based on the data from X_2 and Z_3 be the feature matrix based on the data from X_3.

The figures below are confusion matrices for the different design matrices when using logistic regression.



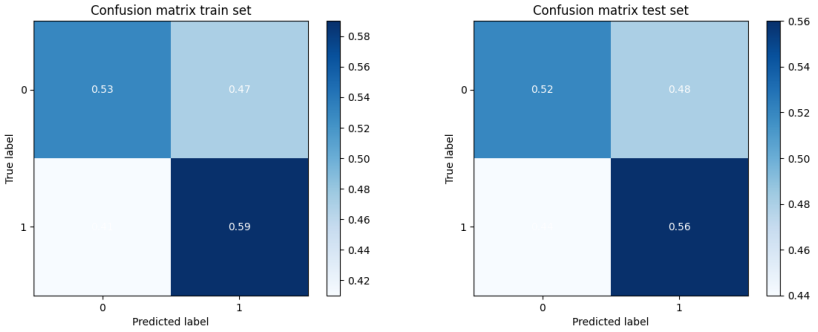
(a) Train accuracy (b) Test accuracy

Figure 1: Train and test accuracy for logistic regression using X_2



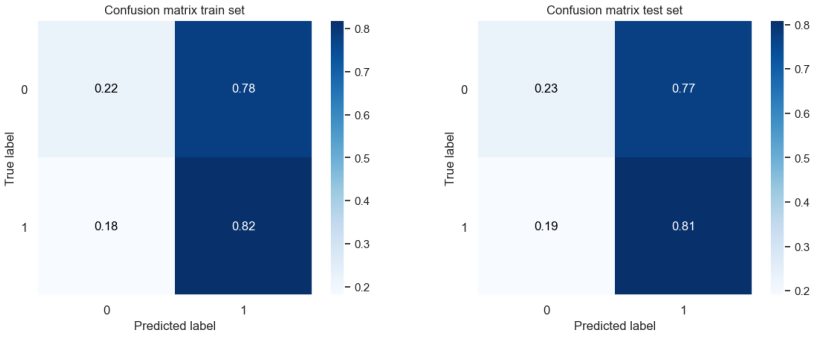
(a) Train accuracy (b) Test accuracy

Figure 2: Train and test accuracy for logistic regression using Z_2



(a) Train accuracy (b) Test accuracy

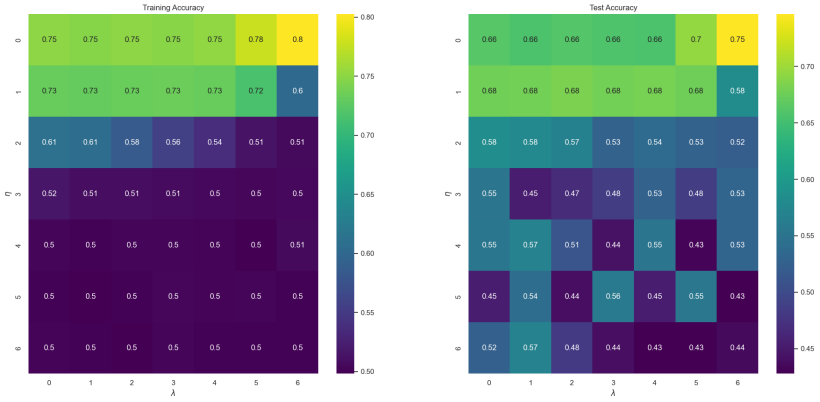
Figure 3: Train and test accuracy for logistic regression using X_3



(a) Train accuracy (b) Test accuracy

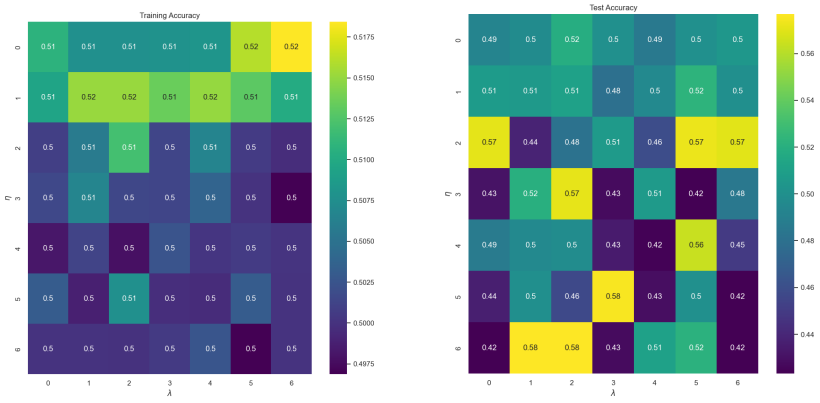
Figure 4: Train and test accuracy for logistic regression using Z_3

The next four figures 5, 7, 6 and 8 represents the training and test accuracy for the different matrices using different η's and $\lambda's$ ranging from 0.00001 to 10.



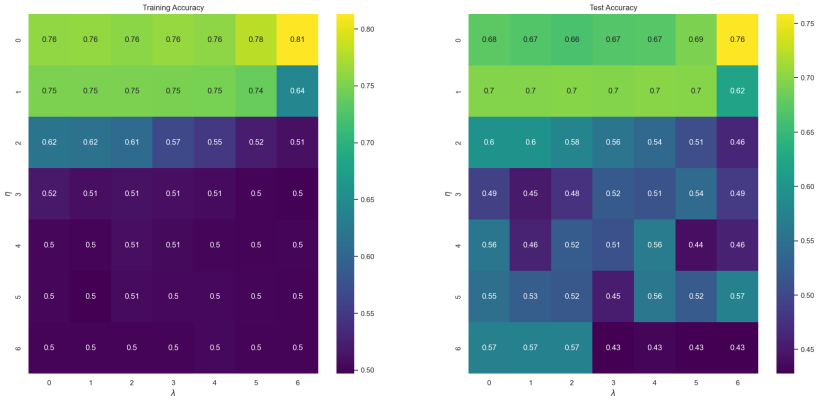
(a) Train accuracy (b) Test accuracy

Figure 5: Train and test accuracy for neural networks using X_2



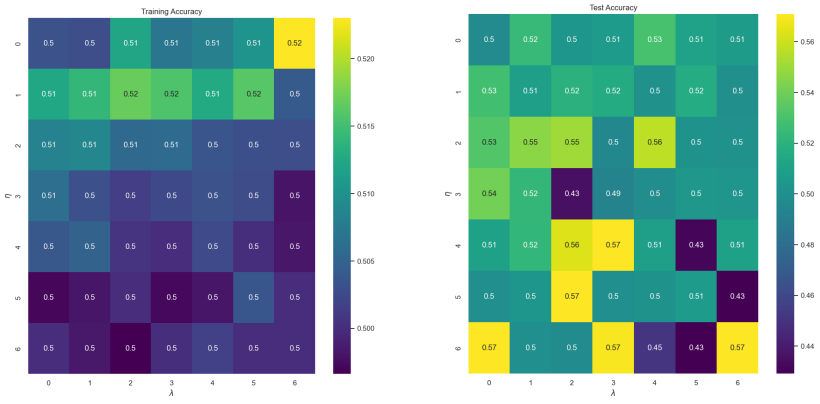
(a) Train accuracy (b) Test accuracy

Figure 6: Train and test accuracy for neural networks using Z_2



(a) Train accuracy (b) Test accuracy

Figure 7: Train and test accuracy for neural networks using X_3



(a) Train accuracy (b) Test accuracy

Figure 8: Train and test accuracy for neural networks using Z_3

Summary

The tables 1 and 2 shows how well different models work on various types of data from the same set. This helps us compare the performance of each model on different design matrices.

	Logistic Regression	Classification Tree	Neural Network
X_2	0.56	0.54	0.80
X_3	0.56	0.53	0.81
Z_2	0.50	0.57	0.52
Z_3	0.52	0.57	0.52

Table 1: Train set Accuracy

	Logistic Regression	Classification Tree	Neural Network
X_2	0.53	0.49	0.75
X_3	0.53	0.58	0.76
Z_2	0.56	0.55	0.58
Z_3	0.48	0.55	0.57

Table 2: Test set Accuracy

Discussion

The tables provided show how different models performed on various types of data. We looked at Logistic Regression, Classification Tree, and Neural Network models on different datasets derived from the same information. We checked how well these models worked on both training and test sets.

The Logistic Regression and Classification Tree models were consistent in training, scoring around 0.50 to 0.60. The Neural Network did even better, consistently scoring above 0.75, especially on datasets labeled X_2 and X_3.

When we compared these datasets, X_3 seemed a bit more accurate than X_2, suggesting it might have better information for the task. The differences between Z_2 and Z_3 were smaller, but Z_2 seemed slightly better overall.

However, when we split the data into training and test sets, we noticed something important. In datasets X_3 and Z_3, there were more 'old' images (labeled as 0) than 'new' ones (labeled as 1). This imbalance might make the models better at predicting the majority ('old') images and not so good at predicting the 'new' ones. It could make the predictions biased towards the majority group, affecting how well the models work overall.

Conclusion

The results highlight how picking the right features significantly impacts how well models perform. Among the different setups we tested, it's clear that using X_3 gives the best info for predicting outcomes. This tells us that the original data, like the images we used, provides the most valuable details, and it's been consistent each time we checked.

Even though the Neural Network works the best overall, we need to watch out for overfitting, where the model gets too caught up in the training data. To avoid this, we might need to adjust the model settings to make sure it can work well with new data too.

In the future, we could try blending different models or dig deeper into the features to make predictions even better without getting too tied up with the training data. We could also consider using techniques to handles the imbalance in the training sets.

Understanding which features matter is crucial, especially if we want to use similar data from other patients in this research project. We need to make sure the features we choose match the data well, so our models can perform even better than they did in this study. This knowledge will help us build better predictive models for future studies in the same area.

Bibliography

[1] Vahid Shahrezaei. Calculus and Applications - Part II: Chapter 1 Fourier Transforms. 2021.

[2] Morten Hjorth-Jensen. Lecture Notes in FYS-STK4155. Applied data analysis and Machine Learning: 13. Neural networks. 2023. https://compphysics.github.io/MachineLearning/doc/LectureNotes/_build/html/chapter9.html

[3] Morten Hjorth-Jensen. Lecture Notes in FYS-STK4155. Applied data analysis and Machine Learning: 6. Logistic Regression. 2023. https://compphysics.github.io/MachineLearning/doc/LectureNotes/_build/html/chapter4.html

[4] Morten Hjort-Jensen. Lecture Notes in FYS-STK4155. Applied data analysis and Machine Learning: 9. Decision trees, overarching aims. 2023. https://compphysics.github.io/MachineLearning/doc/LectureNotes/_build/html/chapter6.html

[5] The Programming Foundation. Module 4 - Logistic Regression. https://learn.theprogrammingfoundation.org/getting_started/intro_data_science/module4/?gclid=Cj0KCQiAsburBhCIARIsAExmsu5R10NrtZ5QiDcXbZC2o9ms8Et03_Dz01Aa_4ABldZiOWbIXE1Atc8aArQnEALw_wcB