Measuring the accuracy of Neural Network on Epileptic Seizures

A study carried out in the context of the Machine Learning Principles and Concepts course

Vasileios Nikiforidis
MSc, School of Science and Technology
International Hellenic University
Thessaloniki, Greece
v_nikiforidis@outlook.com

***Abstract*—This paper presents an application of Machine Learning principles, specifically Neural Networks (Deep Learning), on the act of predicting Epileptic Seizures. This act has relied on human-brain power for many years, but with the rise of Machine Learning in the recent years we can now replicate the procedure, sometimes with higher accuracy, using computers. The main advantage of using Machine Learning is that, once an algorithm is taught what to do with the data given to it, it can do its work automatically.**

Keywords—Machine Learning; Deep Learning; Neural Networks; PCA; algorithm; automation

I. INTRODUCTION

Sometimes after viewing the data, we cannot interpret the pattern or extract information from the it. In such a case, we apply machine learning. With the abundance of datasets available, the demand for machine learning is in rise. Many industries from medicine to military apply machine learning to extract relevant information [1].

The purpose of machine learning is to learn from the data. Many studies have been done on how to make machines learn by themselves [2] [3]. In this paper we will demonstrate the application of a Multi Layer Perceptron Neural Network on a dataset containing features derived from brain wave activity scans. The dataset that was used had a total of 11500 entries and 179 attributes for each one.

II. CONCEPTS

In this section we provide a definition and briefly discuss the core concepts of this paper.

A. Machine Learning

Machine learning is the scientific study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead [4]. As a branch of artificial intelligence, Machine learning is the ability for computer programs to analyze big data, extract information automatically, and learn from it.

Data is being generated faster than at any other time in history. We have reached a point where data analysis cannot be done manually due to the sheer amount of daily produced data.

Many still think of Machine Learning as ‘just another algorithm’. The key difference between a regular algorithm and a Machine Learning Model is that a human has not written any code for the model to be able to differentiate between, for example, a dog and a cat. Instead the model has been taught how to reliably discriminate between the two by training on a huge amount of labelled data, also called training set, containing features describing the two animals

B. Supervised Learning

Supervised learning algorithms construct a mathematical model of a set of data that includes both the inputs and the expected outputs [5]. This set is called training data and consists of a set of training examples. Each example has one or more inputs and the expected output, also known as a supervisory signal. In the mathematical formula, each example is portrayed by an array or vector, sometimes called a feature vector, and the training data is represented by a matrix. Through repetitive optimization of an objective function, supervised learning algorithms learn a function that can be used to predict the output associated with new unlabelled inputs.[4] An optimal function allows the algorithm to correctly determine the output for these unlabelled inputs that were not a part of the training data.

Supervised learning algorithms include classification and regression. Classification algorithms are used when the outputs are restricted to a limited set of values, typically binary, and regression algorithms are used when the outputs may have any numerical value within a range. For the purposes of this study, we will use a classification algorithm, since we want to predict whether an individual could have an Epileptic Episode or not, and thus our problem falls under the Supervised Learning spectrum. There are also other approaches like, Unsupervised Learning, Transfer Learning, Self Learning, Reinforcement Learning, Feature Learning and Semi-supervised Learning but since they are not inside the scope of our study, they will not be further explained here.

C. Neural Networks

A Neural Network is a type of machine learning which models itself after the human brain, creating an artificial neural network, commonly known as ANN, that via an algorithm allows the computer to learn by incorporating new data.

Even though there are plenty of artificial intelligence algorithms these days, neural networks can perform what has been called **deep learning**. While the basic unit of the brain is the neuron, the core building block of an artificial neural network is a perceptron which accomplishes simple signal processing, and these are then connected into a large mesh network.

The computer with the neural network is taught to do a task by having it analyze training examples, which have been previously labelled in advance. A common example of a task for a neural network using deep learning is an object recognition task, where the neural network is presented with a large number of objects of a certain type, such as a cat, or a street sign, and the computer, by analyzing the recurring patterns in the presented images, learns to categorize new images.

The general structure of a Neural Network consists of three types of layers. First, the Input Layer, where the data to be classified are passed. Then, we have any number of what we call 'Hidden' Layers. Hidden layers are set up in many ways. In some cases, weighted inputs are randomly assigned. In others, they are tuned and calibrated through a process called backpropagation. Either way, the artificial neuron in the hidden layer works similarly to its counterpart, a biological neuron in the brain – it takes in its probabilistic input signals, works on them and converts them into an output corresponding to the biological neuron's axon. Finally, there is the Output Layer, where the model provides the result it arrived to after a multitude of calculations. For example, in a binary classification problem, the result would be either positive or negative, which would then be interpreted by the user accordingly.

III. TECHNOLOGIES

In this section, we present the technologies used for this study along with a brief discussion on how they were chosen.

A. Programming Language

For the purposes of this study, **Python** was the language of choice. In most cases, when one wants to play around with machine learning models, Python is the go-to language. This is even more true when delving deeper, into the area of Deep Learning.

There are many reasons as to why Python is such a big deal in machine learning. To start with, it is one of the easiest to understand languages. In very few words, machine learning is recognizing patterns in data and making intelligent decisions based on those patters. Due to Pythons' readability and non-complexity, one can easily understand how the model fulfils its purpose and improve on it. Furthermore, Python has a huge number of supporting libraries, some of which are specialized in machine learning. Among those, the most prevalent are scikit-learn [14], TensorFlow [15] (more of a framework but still functions as a library) and pylearn2 [16]. Later, in this paper, TensorFlow will be further explained, as it was the framework of choice.

Finally, while with other programming languages one would need to familiarize themselves with the language before being able to use it for machine learning, this is not the case for Python. Even with basic knowledge of the language, one can directly use it for machine learning due to the huge number of libraries and community support available.

B. Framework

In layman terms, a framework provides a standard way to build and deploy an application and is a universal, reusable software environment that provides functionality as part of a larger software platform to facilitate development of software applications, products and solutions. Frameworks may include support programs, compilers, code libraries, tool sets, and application programming interfaces (APIs) that bring together all the different components to enable development of a project or system.

Frameworks have key distinguishing features that separate them from normal libraries [6]:

- **inversion of control:** In a framework, unlike in libraries or in standard user applications, the overall program's flow of control is not dictated by the caller, but by the framework.
- **extensibility:** A user can extend the framework – usually by selective overriding – or programmers can add specialized user code to provide specific functionality.
- **non-modifiable framework code:** The framework code, in general, is not supposed to be modified, while accepting user-implemented extensions. In other words, users can extend the framework, but cannot modify its code.

For the purposes of this study, TensorFlow was chosen, as it provided more specialization for Neural Networks, and the library that was used also runs on top of TensorFlow. More on the library in the next subsection.

C. Library

In computer science, a library is a collection of non-volatile resources used by computer programs, often for software development. These may include configuration data, documentation, help data, message templates, pre-written code and subroutines, classes, values or type specifications. Python, in particular, has a large variety of libraries as mentioned before. The library that was used in this study is Keras [17].

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow. It was developed with a focus on enabling fast experimentation.

D. Platform/IDE

When coding in any language, most developers tend to use an IDE which best suits their needs. An IDE provides functionalities and easiness for some features of the language while also automating some processes. Thankfully there are many easy to use and powerful IDEs out there for Python, but what is the core need for machine learning, is computational power, especially when dealing with Big Data.

For this reason, Google Colaboratory [18] (also known as Colab) was chosen to develop the Neural Network presented in this paper. It is a part of the Project Jupyter [19], thus it provides the user with a free Jupyter notebook environment that runs in the **cloud.** This means that all the computational power is provided by the cloud and even low-end computers can now run complex and taxing models with no problem, which makes it an ideal environment for any Machine Learning project.

IV. Data & Problem Description

The Epileptic Seizure Recognition Data Set [20] was used for this study, which resides in the UCI Machine Learning Repository [21]. Originally the dataset was comprised of brain wave activity reading from 500 different people. Each subject's brain activity was scanned for 23.6 seconds, and for every second, there were 178 attributes extracted, so in total, the dataset consists of:

$$23 seconds \; x \; 500 subjects = 11500 \; entries$$

The data set had no missing values, thus there was no need to apply any missing values handling technique.

The target variable in the 179th column (data sets start at column position 0) was an integer with value spanning between 1 and 5. All subject falling in classes 2,3,4 and 5 are subjects who did not have an epileptic seizure. These values have another meaning which is not relevant for the study. Only subject in class 1 have epileptic seizure, thus the problem falls in the binary classification area. In this paper, a way to build a prediction model using an MLP (multi-layer perceptron) Neural Network, will be presented.

Furthermore, we will check the models' integrity and stability by performing a 10-fold cross validation, which is a standard statistical technique for testing the performance of a machine learning model.

Finally, we will perform a dimensionality reduction technique called Principal Component Analysis (PCA) [23]. In simple terms, PCA is a technique to reduce the dimensions of the feature space (178 for this dataset), by feature extraction. This is done through the creation of new independent variables by combining the existing features. By creating new variables, it might seem as if more dimensions are introduced, but we select only a few variables from the newly created variables in the order of importance. Then the number of those selected variables is less than what we started with and that's how we reduce the dimensionality.

$$PC1 = w_{1,1}(\text{Feature A}) + w_{2,1}(\text{Feature B}) + w_{3,1}(\text{Feature C}).... + w_{n,1}(\text{Feature N})$$

$$PC2 = w_{1,2}(\text{Feature A}) + w_{2,2}(\text{Feature B}) + w_{3,2}(\text{Feature C}).... + w_{n,2}(\text{Feature N})$$

$$PC3 = w_{1,3}(\text{Feature A}) + w_{2,3}(\text{Feature B}) + w_{3,3}(\text{Feature C}).... + w_{n,3}(\text{Feature N})$$

Figure 1 Creating new variables from existing features by taking feature weights into account.

V. Description of Model

The current section is devoted to introducing the reader into the methodology and the model used in this study.

To start with, a standardization of the data is required preceding the PCA. In particular, we perform a typical standardization on our data, with a mean equal to 0 and a standard deviation equal to 1, which results in having all our attribute values belong in the area between 0 and 1.

Concerning the Neural Network itself, a dense sequential Keras network was created to receive the data. As the name suggests, a dense network is one comprised by fully connected layers, meaning that all the neurons in a layer are connected to those in the next layer. Typically, the term 'dense' is applied on the layer itself, so when all layers of a network are dense, it is referred to as a Dense Network. A densely connected layer provides learning features from all the combinations of the features of the previous layer. Also, the term 'sequential' means that the model is a linear stack of layers.

In order to add a layer to the network, some parameters must be provided. First, the number of nodes, which is also the dimensionality of the output space, must be provided.

Then the developer must provide the activation function. The activation function of a node defines the output of that node given an input or set of inputs. There is a variety of choices for this, such as Tanh, Sigmoid, ReLU etc. For this study, the ReLU (rectified linear unit) function was chosen. Both Sigmoid and ReLU are very commonly used activation function but ReLU is more computationally efficient since it just needs to pick $max(0, x)$ and not perform expensive exponential operations as in Sigmoid [7][8]. In practice, networks with ReLU tend to show better convergence performance than sigmoid [9]. For the output layer only, the hard_sigmoid [22] activation function was used, which assures that the result will be a value between 0 and 1 and is also faster to compute than regular Sigmoid.

Moving on, the regularizers need to be set for each layer. Regularizers allow the network to apply penalties on layer parameters or layer activity during optimization, thus also indirectly helping with overfitting prevention. These penalties are incorporated in the loss function that the network optimizes [10]. In the network of this study, the *kernel_regularizer* was used. This regularizer has three options to choose from, which determine the way of penalization, l1, l2, and l1_l2. After a grid

search [24], l2 was found to be the most effective of the three for this dataset. There is no absolute rule that can tell one which to use so only through trial and error can the best regularizer be found. The difference between l1 and l2 is that l1 penalizes the sum of absolute value of weights while l2 penalizes the sum of square weights.

In order to combat overfitting, Dropout layers were introduced to the model. Dropouts consists in randomly setting a fraction rate of input units to 0 at each update during training time, which helps prevent overfitting [11]. Here, a Dropout layer was added after the input layer and each hidden layer.

Finally, since the goal is to maximize the accuracy of the model, the optimal values for the parameters set needed to be found. This exploration was done in a separate notebook on Colab. In that notebook, a grid search for regularizer values, layers, and nodes per layer was performed, each one separately. After finding the optimal values for each one from their respective grid search, a general, and much more time consuming (3.7 hours for 252 iterations of the network) grid search made to find the best combination of these parameters and see if they coincide with the individual parameter values findings.

VI. Comparative Experiments and Results

This section is devoted to displaying the results of this study.

To begin with, a plot depicting a regular brain wave pattern versus an epileptic one, will be displayed, mostly for educational purposes.

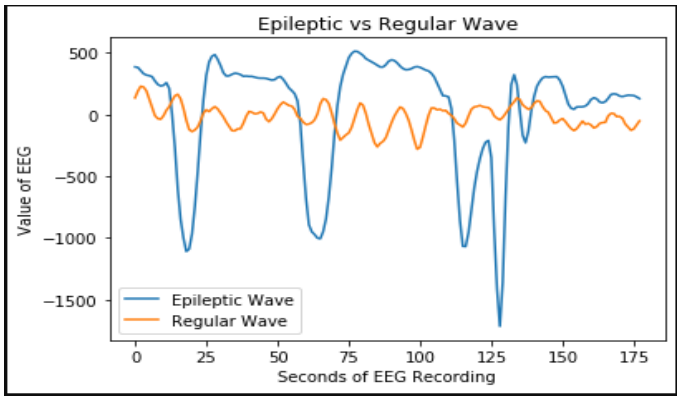

Figure 2 Brain waves comparison.

We can clearly see the irregularities in the readings of an epileptic brain wave compared to a standard one, even with just a quick glance at the plot.

Moving on to the training, the model was created with the parameters mention earlier and only a single hidden layer as it was deemed to be the optimal number of layers after the grid search, the created neural network had a total of **15,681** parameters and after 200 epochs, it achieved around **98.9%** accuracy for the training set and **97.8%** for the test set.

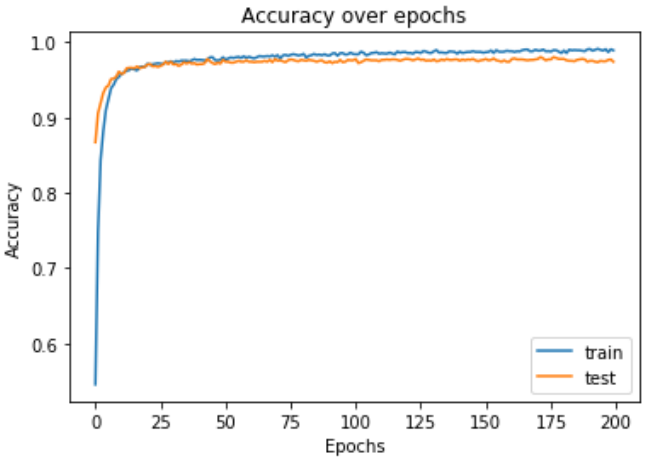

Figure 3 Accuracy of the model

As we can see, 100 epochs would have been enough since the accuracy is mostly stabilized after that, especially for the test set, but since 100 more epochs had little computational cost, the model was allowed to train a bit more.

Even though the results seem satisfactory at first glance, the ROC curve should be studied to determine the ability of the model to not only correctly predict a positive as a positive, but also a negative as a negative. The ROC curve does this by plotting sensitivity, the probability of predicting a real positive will be a positive, against 1-specificity, the probability of predicting a real negative will be a positive. The best decision rule is high on sensitivity and low on 1-specificity. It's a rule that predicts most true positives will be a positive and few true negatives will be a positive.

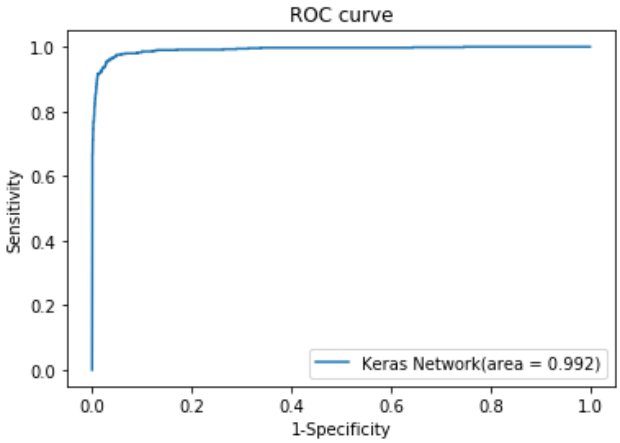

Figure 4 ROC curve.

The plot above proves that the model is quite capable of identifying false positives and false negatives. The same test was run without using the Dropout layers, which affected the model's ability to manage overfitting and resulted in an area of 0.988 which means that, even though it's a slight difference, the model would have more false positives and/or false negatives without introducing the Dropout layers to further prevent overfitting.

On the subject of overfitting, a good way to test a model and see if it's dependable on any data given to it, provided the data are subject to the same standard as the data it was trained on, is cross validation. Cross validation is done through repeatedly training and validating the model on a dataset. Each such repetition is called a 'fold', and the most common number of folds for a cross validation is 10, otherwise called 10-fold cross validation. One such fold, involves partitioning a sample of data into complementary subsets, performing the analysis on one subset, and validating the analysis on the other subset. To reduce variability, in most methods multiple rounds of cross-validation are performed using different partitions, and the validation results are combined (e.g. averaged) over the rounds to give an estimate of the model's predictive performance. In summary, cross-validation combines (averages) measures of fitness in prediction to derive a more accurate estimate of model prediction performance [12].

```
fold 1 : accuracy: 98.09%
fold 2 : accuracy: 98.70%
fold 3 : accuracy: 98.96%
fold 4 : accuracy: 97.83%
fold 5 : accuracy: 99.30%
fold 6 : accuracy: 98.70%
fold 7 : accuracy: 98.78%
fold 8 : accuracy: 99.22%
fold 9 : accuracy: 99.22%
fold 10 : accuracy: 99.13%
Average accuracy: 98.79% (+/- 0.47%)
```

Figure 5 Cross Validation results

As we can see, the model scored close to 1% more than on the data it was initially trained on (Figure 3). This can be attributed to the fact that the data it was first trained on, was just one of the possible subsets of the dataset. The true accuracy is the one provided by the cross validation since it was measured by getting the average value of all 10 folds it was trained in. Furthermore, we can see that the model is quite stable, having less than 0.5 deviation from its average accuracy. Taking these results into account, we could assume that the network safe from overfitting.

Finally, we arrive at the PCA application. There are three steps to take before actually doing PCA on our model.

First off, we need to normalize the data. PCA is used to identify the components with the maximum variance, and the contribution of each variable to a component is based on its magnitude of variance. It is best practice to normalize the data before conducting a PCA as unscaled data with different measurement units can distort the relative comparison of variance across features [13].

The second step is to create a covariance matrix for Eigen decomposition. A useful way to get all the possible relationship between all the different dimensions is to calculate the covariance among them all and put them in a covariance matrix which represents these relationships in the data. Understanding the cumulative percentage of variance captured by each principal component is an integral part of reducing the feature set [13].

Last but not least, we need to select the optimal number of principal components. The optimal number of principal components is determined by looking at the cumulative explained variance ratio as a function of the number of components. The choice of principal components is entirely dependent on the tradeoff between dimensionality reduction and information loss. Initially we set the number of principal components equal to the number of attributes of the dataset in order to figure out how many are actually needed.

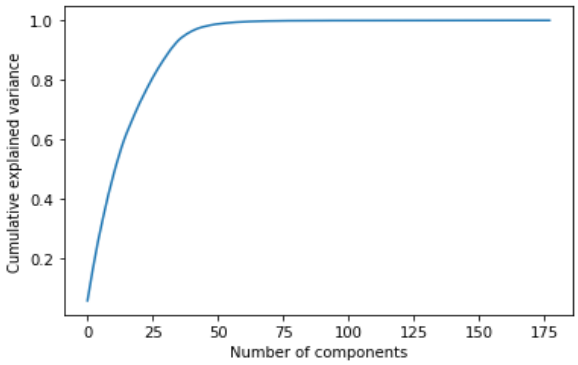

Figure 6 PCA components

In the plot above, it is clearly visible that the variance peaks close to 50 components. That means that data describing changes is mostly contained in 50 components. Thus, a new network was created using the PCA created variables, for the network to train on. The newly created network had a total of **7,489** parameters, which is less than half of the original one. The only thing left to do now, is perform one more 10-fold cross validation to see how this network will perform.

```
Performing cross validation on the pca_network.
fold 1 : accuracy: 97.76%
fold 2 : accuracy: 97.76%
fold 3 : accuracy: 98.51%
fold 4 : accuracy: 98.88%
fold 5 : accuracy: 98.63%
fold 6 : accuracy: 99.25%
fold 7 : accuracy: 98.51%
fold 8 : accuracy: 99.13%
fold 9 : accuracy: 99.13%
fold 10 : accuracy: 99.13%
Average accuracy: 98.67% (+/- 0.52%)
```

Figure 7 Cross validation on network after PCA

VII. Conclusions

The aim of this study was to develop and evaluate a neural network with the purpose of identifying epileptic seizures. The network managed to achieve a high accuracy while also being stable (less than 0.5% deviation from average accuracy).

It was also observed that Dropout layers played a role in preventing overfitting and helping the model improve its sensitivity and specificity.

The generalized grid search resulted in the best result coming from a network with 2 hidden layers with 512 nodes each, and a regularizer value of 10^{-4}. That accuracy though, was only 0.001 higher than the one achieved by a network with 1 hidden layer with 64 nodes and a regulirizer value of 10^{-6} and required a lot more time to train and compile the network. So, taking into account the tradeoff between the increase in accuracy and the computational speed, the second set of parameters were used for the final model.

Finally, after studying the results of the cross validation on the original network and comparing them with the results of the cross validation on the network created after PCA, the average accuracy of the latter was found to be slightly lower (-0.08%) and also with higher deviation (+0.05%). What was gained from the PCA though was speed. The PCA network compiled, trained and predicted much faster the original due to its reduced dimensionality. In the end, the tradeoff of a low decrease in accuracy for faster predictions lies with what the developer wants to accomplish.

References

[1] Ayon Dey / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 7 (3) , 2016, 1174-1179

[2] M. Welling, "A First Encounter with Machine Learning"

[3] M. Bowles, "Machine Learning in Python: Essential Techniques for Predictive Analytics", John Wiley & Sons Inc., ISBN: 978-1-118- 96174-2

[4] https://en.wikipedia.org/wiki/Machine_learning

[5] Russell, Stuart J.; Norvig, Peter (2010). Artificial Intelligence: A Modern Approach (Third ed.). Prentice Hall. ISBN 9780136042594.

[6] Riehle, Dirk (2000), Framework Design: A Role Modeling Approach (PDF), Swiss Federal Institute of Technology

[7] Han, Jun; Morag, Claudio (1995). "The influence of the sigmoid function parameters on the speed of backpropagation learning". In Mira, José; Sandoval, Francisco (eds.). From Natural to Artificial Neural Computation. Lecture Notes in Computer Science. 930. pp. 195–201. doi:10.1007/3-540-59497-3_175. ISBN 978-3-540-59497-0.

[8] Vinod Nair and Geoffrey Hinton (2010). Rectified Linear Units Improve Restricted Boltzmann Machines (PDF). ICML.

[9] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton ImageNet; University of Toronto; Classification with Deep Convolutional Neural Networks; http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf

[10] https://keras.io/regularizers/

[11] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov (2014), "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", Journal of Machine Learning Research 15 (2014) 1929-1958

[12] Grossman, Robert; Seni, Giovanni; Elder, John; Agarwal, Nitin; Liu, Huan (2010). "Ensemble Methods in Data Mining: Improving Accuracy Through Combining Predictions". Synthesis Lectures on Data Mining and Knowledge Discovery. Morgan & Claypool. 2: 1–126. doi:10.2200/S00240ED1V01Y200912DMK002.

[13] https://medium.com/apprentice-journal/pca-application-in-machine-learning-4827c07a61db

[14] https://scikit-learn.org/stable/index.html

[15] https://www.tensorflow.org/

[16] http://deeplearning.net/software/pylearn2/

[17] https://keras.io/

[18] https://colab.research.google.com/

[19] https://en.wikipedia.org/wiki/Project_Jupyter#Colaboratory

[20] https://archive.ics.uci.edu/ml/datasets/Epileptic+Seizure+Recognition

[21] https://archive.ics.uci.edu/ml/index.php

[22] https://keras.io/activations/#hard_sigmoid

[23] https://en.wikipedia.org/wiki/Principal_component_analysis

[24] https://en.wikipedia.org/wiki/Hyperparameter_optimization#Grid_search