Generating Representative Artificial Datasets for Algorithm Evaluation

Dan-Florin Berbec
Supervisor: Peter Bloodsworth

Kellogg College

University of Oxford

Project Proposal
MSc in Software Engineering

Abstract

Proprietary complex algorithms are one of the main assets for many IT companies and their correctness is fundamental in fields such as finance and medicine, or simply when using an app that provides directions to a user or makes suggestions based on personal information. We may expect in the near future that some multi-million companies will reside mostly in the cloud with only a handful of employees. For regulators, investors or companies that have an interest in buying or investing in such a company, the task of validating the IP becomes difficult, especially when all that is accessible to a buyer is a black box with a limited amount of test data. We aim to assess the tools available to reliably and independently evaluate algorithms and present a process that can provide confidence to potential buyers that an algorithm works indeed as specified.

1 Introduction

1.1 Area of study

Algorithms are present in almost every electronic device we buy. From a simple light switch to running an OS and applications on a smartphone, their influence reaches beyond our daily decisions. For example, algorithms in a smartphone may direct a taxi driver on an alternative route based on congestion indicators set by a real-time data feed. Similarly, the prices of products in a supermarket may be influenced by low latency foreign exchange algorithms trading in an investment bank.

While simple algorithms have been around for many years aiding our decisions, complex ones now become ubiquitous in our decision-making process, many times with a significant outcome. Almost all industries including finance and healthcare or governmental institutions use algorithms to process large amounts of data to optimise spending resources. For example, a review[3] from the New Zealand government shows how crucial algorithms are in ensuring the wellbeing of the population. Ministries of justice, education and healthcare, customs and police, all use algorithms and data to improve their efficiency. Algorithms allow for data to be gathered and analyzed, identifying key areas where young people need support or helping to aggregate data and address risks at border control. Passport and visa applications can also be automated to reduce waiting times and allow staff to manually process the ones that reach a risk threshold. However, the review clearly states that significant decisions are still currently made by humans.

Another example shows the US justice system relying much more heavily on algorithms - these are used to provide a likelihood score of a convict to re-offend and is one of the main factors for a judge when deciding a sentencing[2]. The algorithm used is trained on historical data and seems to have biases correlated to a convict financial status - it transforms these biases into causations. Importantly, the underlying algorithm is private and not made available to the public for assessment.

Finance also relies heavily on automation and algorithmic trading is rising very fast. Robot advisors are slowly replacing or at least aiding financial advisors to reduce the commissions charged as well as improve the performance of a fund[6]. For most financial institutions, especially large investment banks, regulators require backtesting to be executed and results sent for analysis[5]. This aims to ensure that trading strategies performing live today can also handle data in extreme environments such as the financial crisis in 2008.

While most of the algorithms used in decision making are currently crunching data to produce a deterministic output, newer types of algorithms within Machine Learning (ML) and Artificial Intelligence (AI) aim to train on historical data to find patterns that are applicable on new test data. From systems that recommend similar items to a virtual Go player that can beat a world champion[4], ML algorithms become increasingly trusted to performs functions that may significantly affect people's life. Huge investments are poured into self driving cars and a large number of big tech companies as well as many start ups[19] claim a share of this funding. These smart algorithms powering the cars may be expected to take the right decisions in noisy environments that have not been part of the training data, or ethical ones with no clear solutions, such as the trolley problem[17]. As they are designed by humans, biases are also likely to influence the algorithm design.

One of the main assets of ML companies lies in developing such algorithms and preserving their secrecy. For big tech giants, such as Google, Uber or Waymo, theft of trade secrets is a serious concern and lawsuits are commonplace[7, 18]. Algorithmic trade secrets, such as trading software, are also crucial in the financial industry, and any attempt for industrial espionage is thoroughly chased[16].

Provided that secrecy is of utmost importance for ML companies across all industries, many questions arise from stakeholders: how can regulators verify the fairness/lack of bias of a financial algorithms? Will an algorithmic robot trained on worst case scenario data from the 2008 financial crises perform as well during a new financial crises with different root causes? How can a judge have confidence that no bias is part of the score given to a convict if the underlying algorithm is only accessible to a handful of people? How can investors evaluate that the product of an ML company does indeed what is says it does?

Most of the time, a blackbox is provided to an external audit or potential investor with limited test data. Furthermore, this data could easily be proprietary as well and cleaned to fit the claims of the product. Currently, it seems that stakeholders do not have any means to independently verify that an algorithm does indeed what it says it does.

With the increasing rise of tech startups, we may assume that in the near future, multi billion dollar companies may have only a handful of employees with a codebase living in the cloud and their main asset being IP. Their algorithms may have significant decisions for the public, yet understanding why a decision has been taken will be challenging to explain.

1.2 Proposed work

The aim of this project is to investigate the decision making process of auditing proprietary algorithms and propose methods to realistically improve the process. However, these general guidelines will aim to be applicable to any organisation independent of the size or industry.

Given the rise of machine learning algorithms and in particular deep learning, the project aims to primarily assess their explainability especially for cases where these are provided as a black box with a limited amount of proprietary test data. Deep learning mainly provides two types of algorithms: supervised and unsupervised learning. Supervised learning is used for classification purposes while unsupervised learning aims to find patterns within a data sets that are not mapped to specific labels.

One of the most commonly used supervised classification algorithm is powered by artificial neural networks (ANN). These are trained on test data sets and make predictions on new inputs. Their efficiency can be measured on how accurate their predictions are. For example, given a dataset of customers leaving a bank, a ANN can make future predictions based on independent variables on whether a customer is likely or not to leave the bank as well. A ANN could also decide based on income and age, whether a customer should be granted a loan or not. Another example could be a ANN making stock price predictions. While the algorithm is not public, it would be crucial for customers to have the ability to measure its performance before placing any money of their savings.

But ANN can easily be trained to perform well on some test data while new independent data would decrease their performance significantly or rendering the multi-million dollars algorithm worthless.

The scope of this project will mainly focus on assessing ANNs with a special interest in analyzing the training and test data available to evaluate the algorithm. We will look whether statistical inferences can be derived from the provided data and tools to expand or generate new data with similar parameters. The aim is also to test the behaviour of the algorithm under worst case scenarios or perform other modifications to the dataset to identify potential poor performance that impacts the validity of testing and its efficiency. This may include discussing big data architectures, working out patterns in data, generating random numeric datasets or creating altered datasets to test extremes. Unsupervised deep learning algorithms such as self organising maps could provide great help. Blackbox testing strategies applicable to deep learning will also be investigated.

Provided that one of the main focuses of the dissertation will be on analysing the visible test data, we may argue that the underlying algorithm within the black box is not relevant. However, each algorithm requires a specific format for the data to be provided in. Thus, understanding the underlying algorithms and being able to modify these is just as important as the dataset analysis: it will provide a deeper understanding of potential issues from the perspective of IP owner and help understand the accuracy of the algorithm performance.

The project will focus on the investigation of ANN test data, which is mostly numeric and human readable. As a stretch goal, it will be useful to apply any findings to test other types of supervised learning such as convolutional neural networks used for example to classify images, or recurrent neural networks which make predictions based on time series, for example the stock price of a company. Another stretch goal would also consider analysing reinforcement learning algorithms in a black box and comparing these with evaluation of deep learning ones. Can the steps used to evaluate a deep learning algorithm also be applied to evaluate a reinforcement learning algorithm? However, this project does not intend to analyse all the validity of test data for each type of supervised deep learning algorithm: it will strictly commit to analysing ANNs and depending on the time and relevance, extend the scope to convolutional and recurrent NN as stretch goals only.

The project will evaluate a number of open source ANNs used in a variety of fields such as healthcare, finance, or law. The algorithms may be packaged as runnable code with training and test data or as a black box with test data only. Using proprietary algorithms will be avoided where possible, but may be used if the case study provides useful information.

The expected outcome will present the reader with a with a process to validate a company IP by using a set of tools and techniques such as statistical analysis, big data, applicability of generative adversarial networks, and preparation of test data to validate the algorithm. These examples do not represent a commitment but rather a sample of what might be useful in developing the process, which is subject to further background literature and the rapid growth of new ideas being published in the field. New solutions may be developed if the literature lacks answers to some of the problems presented, but only within the realistic timeline and scope of this project.

1.3 Artificial Neural Networks

Artificial Neural Networks have become very popular in recent years for creating Artificial Intelligence algorithms. The aim is to teach a machine how to extract and learn features of a data set to make new predictions by replicating processes happening in the biological brain. Using a simplified neural representation: the dendrites, nucleus and axon of a neuron are respectively abstracted for a theoretical algorithm as the input, processing node, and output. This idea dates back from 1957 when Frank Rosenblatt demonstrated the first perception: ”the first machine which is capable of having an original idea" [14], but only recently has computational power increased to develop the ability to make predictions.

The ANN is built using visible input nodes (data you can see - ex. age, address, education, job status) that feed into several hidden layers of neural nodes further feeding into output nodes, the visible binary or categorical output, for example a loan application being granted or declined. A node in any layer can be connected to one or many nodes in the next layer. Due to the complexity of the connections in the middle layers and the fact that the data is not easily readable, the middle layers are called hidden layers (fig 1).

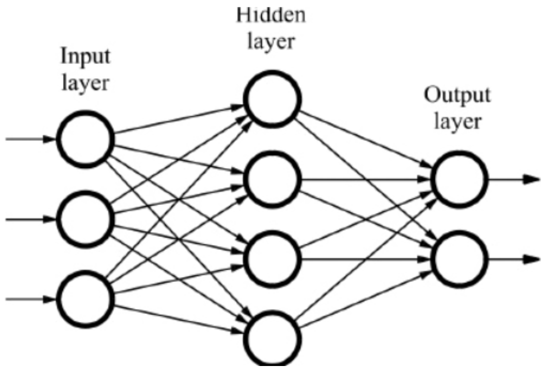


Figure 1: Simple neural network with one hidden layer [12]

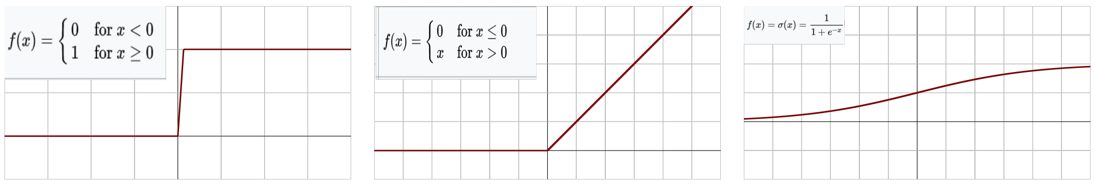
Each neuron in the network receives input from other nodes and computes this into a single output that feeds into other neurons or output nodes. A simple approach for a neuron's prediction could be to multiply an adjustable weight to each incoming input. To avoid producing zero values for zero inputs a bias can be added further as shown in eq 1

$$\sum_{i=1}^{n} x_i w_i + b_i \tag{1}$$

When the input $x_i w_i$ is significantly less than the bias b_i this implies that input has little effect. But over the training period the weight will be adjusted accordingly to overcome the effect of the bias. The final sum is not bounded and thus can be largely different than the output of other nodes. Activation functions are used to contain this value between a specific range, most commonly between -1 and 1. For example, a simple activation function, binary step, can solve this issue. But this cutoff function however will drop many small details and may only be used in binary classification problems. While there are many activation functions available, the most commonly used alternatives are the *sigmoid* function and *rectified linear unit* (ReLu) which preserve and propagate more information as shown in figure 2.

In multi-class classification problems, the output layer will have one neuron per class. The categories may or may not be mutually exclusive and a common way to pick the best prediction is to choose the one with the highest probability when using for example a sigmoid function. As neurons in the last hidden layer are not interconnected these probabilities are independent and need to be normalized such that their sum equals to 1. A simply function such as *softmax* divides each probability by the total sum. When categories are non exclusive, a probability threshold for each output neuron such as 0.5 can be used to pick the final outcomes.

The ANN can be trained using an existing dataset of examples to make future predictions and the weights of each neuron are adjusted depending on the performance of the prediction vs



(a) Binary Step (b) ReLu (c) Sigmoid

Figure 2: Activation functions [1]

the actual value. Equations such as the quadratic cost function (see eq 2) or cross entropy are commonly used to make this measurement for both binary or multi-class classification. The equation provides a positive measurement of the distance from the real value. By defining a^L as the activation function applied on $x_i w_i$, at the last layer, the cost function takes into account a large amount of information as vectors: the weights, biases, activation function and true outcome across a number of predictions called the batch size.

$$C = \frac{1}{2n} \sum_x \|y(x) - a^L(x)\|^2 \tag{2}$$

To find the correct values for the weights and biases, the cost function needs to be minimal. By plotting a simple version of this exponential cost function against the weight of one neuron (see fig 3), one could theoretically simply calculate the derivative for 0, but in reality the complexity is directly proportional to the number of neurons and will have many local minimum values making such computations infeasible.

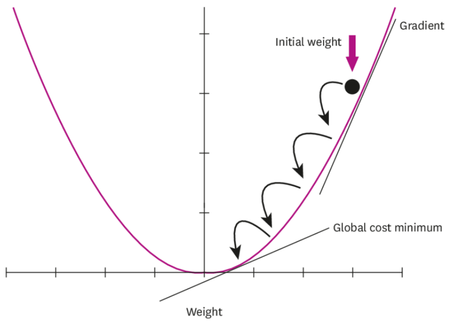


Figure 3: Simple cost function graph[8]

Instead, the slope can be calculated and moved downwards to find the minimum value that optimises *w*, known as gradient descent. After running a fraction or the entire dataset also known as the batch size, weights can be adjusted based on minimising the slope. Large step sizes may overshoot the minimum while small ones will take a long time to converge. Thus, a good approach is to adapt this learning rate and Adam[20] is one of the best performing algorithms.

When running on the entire batch size however, the updates happen less frequently and a local minimum is more likely to be chosen instead of the global one. An alternative is to optimize weights after each input in the training data, known as stochastic gradient descent. Due to the increased update frequency, this is more likely to find a global minimum, but is dependent on the order of the rows in the training set and is not deterministic when data is picked randomly. Finding the global minimum has many other challenges depending on the dataset, such as very big slopes or many small ones. Trusk 2015 [13] provides possible solution to these issues.

Backpropagation uses the cost function to apply changes on the weights and biases of each neuron in every layer. Taking a multi-class classification problem, such as labeling dogs, cats, and fish, the aim is to make adjustments starting from the output layer and propagating these back to the first hidden layer. If the training example has a true value of a cat and the output of the activation function has a high probability for dog, then the aim is to decrease $a^L(x)$ for the dog label and increase the the weight for cat label using the cost function. The in depth mathematical details of the algorithm are beyond the scope of this project but the main equations including formal proofs are discussed by Nielsen 2018 [21] in chapter 2.

2 Building an ANN

2.1 Choosing a Dataset

Kaggle[10] in a popular online platform for data scientists to share knowledge: this includes uploading real datasets and making them available for public use. Each dataset is open to a discussion thread and possible sample solution are discussed for different types of classification, including deep learning algorithms. Many top machine learning experts who work in the industry or academia will take part in competitions and their solutions are made public. This makes such datasets an excellent choice for the purpose of this project as a simple ANN implementation can be assessed against other solutions which arguably would end up powering real consumer products.

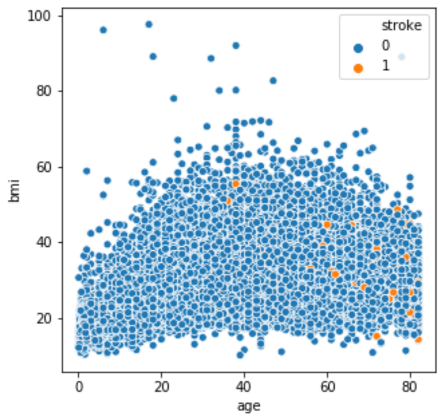
The Stroke dataset[15] contains lifestyle and medical data about patients, some of which suffered a stroke. This is an anonymous dataset the was first published be McKinsey as part of an online Hackathon on Healthcare although the source of this information cannot be verified online. The dataset has no personal identifiable information. The 14 columns contain attributes such as gender, age, hypertension, heart disease, smoking status, or the work type of a patient. More importantly, it also specifies if a patient had a stroke which allows for classifier to be trained and make future predictions.

2.2 Exploratory Data Analysis

When building predictive models, a crucial part is understanding the data before building a predictive model. For example, what columns are significant in predicting a stroke? Can we derive insights based on correlations? Can some information be discarded or can missing data be filled?

The stroke dataset contains 43400 entries of which only 783 (1.8%) are patients who suffered a stroke. Intuitively, we can assume that strokes will occur more often in adults and elderly (see fig 5). A heat-map using a balanced subset (equal numbers of stroke and non-stroke patient) of the available data shows correlations among all the data columns (fig 6). Missed or less intuitive correlations for a non medically trained viewer become clearer.

For example, it is expected that age is highly correlated with a person's lifestyle choices and health such as being married, work type, having a heart disease or hypertension. It may not be immediately obvious that heart diseases and strokes follow an almost identical distribution with a higher likelihood at older ages (fig 5)). Hypertension starts at earlier ages and may be a



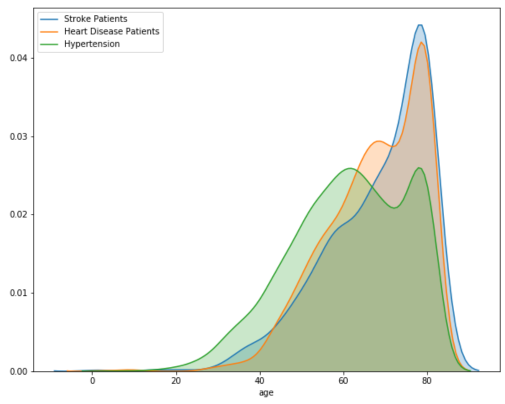


Figure 4: Age vs body mass index

Figure 5: Age distribution of patients who suffered a stroke

key indicator for later strokes, both having a high correlation. Another less clear correlation is the age vs the body mass index (bmi): fig 4 shows how the bmi is increasing from childhood through to adulthood and decreasing again for elderly people. Interestingly, the bmi is also correlated with the work type - this could likely be explained by the fact that those who work are also more active.

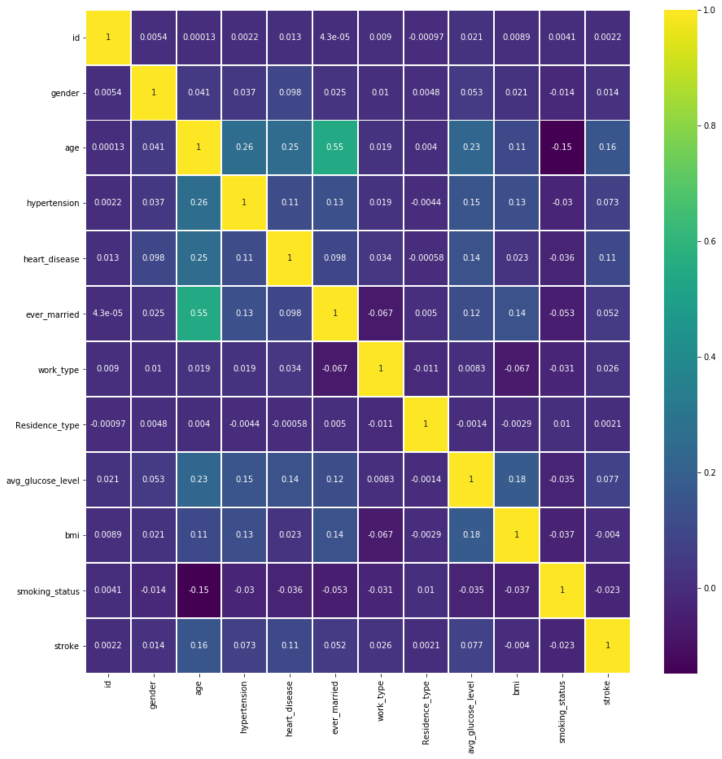


Figure 6: Correlations among all data columns using a balanced subset - smoking status is mapped to binary values

Surprisingly, a strong correlation is seen among smoking status and gender (fig 7) than among smoking vs strokes or hypertension. There are more males who smoke or formerly smoked than females. The number of females who never smoked is also significantly higher. Only 1.7% of those who never smoked suffered a stroke, only slightly better than 2% of those who regularly smoke. Unexpectedly, the highest rate of strokes is for former smokers at 3%.

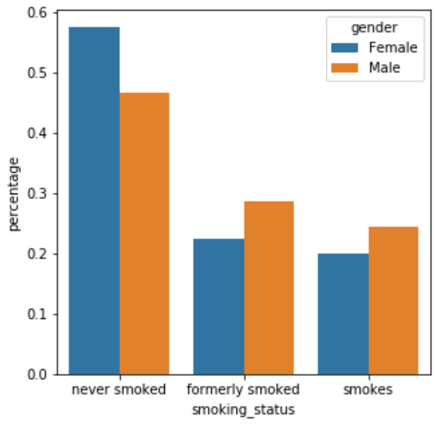


Figure 7: Percentage of male/female smokers

2.3 Categorical and Missing Data

The correlations and insights derived from the previous section allow us to fill in missing data in a manner that is consistent with the dataset's features. Two columns, the *bmi* and *smoking status* are missing information.

There are 1462 (3.3%) rows missing the bmi value. To ensure the dataset has no *nan* values, we could simply delete these rows or replace the missing bmi with an average over the entire dataset. However, the strong correlations with age and work type allow us to be more precise: replacing the bmi with the average bmi for that age group would produce a larger distribution, but the bmi has a significantly stronger correlation (0.42) with the work type, which in turn will distribute bmi average values over five types of work. Averaging the bmi of each work type to replace missing bmi is the better choice.

The smoking column is missing in 30% of the data. Surprisingly, this is not strongly correlated with other attributes, and has a particularly low correlation with the stroke column. A reasonable approach could be to delete the column entirely, but in a realistic scenario, stakeholders would loose confidence if they learned that smoking information has been dropped in a stroke prediction product. As such, we can fill this information using an empirical approach: if the work type is children or age is less than 18, we set this values to *never smoked*. Otherwise, we can make use of the fact that males are significantly heavier smokers and thus decide the status based on gender.

The least significant column in the dataset is the id of patients. This is a numerical value that replace personal identifiable data. This has no correlations with any attributes and should be dropped.

Categorical data needs to be mapped to numerical values for an ANN to work. One hot encoding is used to avoid wrongly increasing the importance of a category based on it numerical mapping. For example, the column residence type will translate to columns *urban* and *rural* with values of 0 or 1 only. As this is duplicating the data, only one column is need and the other is dropped. For attributes with more categories, columns are created for each value with mappings of 0 or 1. This approach ensures that no categorical value has a higher rank than others.

By creating these mapping to numerical values, the ANN will need more neurons and more processing resources. Such mappings are usually evenly distributed, but in the case of gender which has three categories (Male, Female, Other) , only 11 entries are labelled as Other out of 43400 rows. One hot encoding would add unnecessary overhead on the dataset and a good approach would be to merge such values in one of the other categories. In this case however, the eleven rows can be dropped. Finally, the number of columns after the data translation increased from 11 to 15 columns.

2.4 Architecture of the ANN with Keras

Keras[11] is the official api library for implementing high-level ANNs in Python using tensorflow (2.0). Hidden and output layers can be easily added with parameters such as the number of neurons or activation function to be used. A model can be trained using a large number of parameters that abstract away from the granular implementation details that would otherwise require an in depth understanding of complex mathematical formulas.

The first step is to split the dataset into a training and test set. We use 80% to train and 20% to test. The validation data will only be used to measure the performance and not for any training purposes during backpropagation. The data is then sclaed and normalized using the *std* and *min/max* values of the dataset.

The number of epochs is set to 200 with an early stopping callback to avoid over fitting after the valuation loss stops improving. Given this is an imbalanced dataset, the batch size is set to 2048 to ensure the model trains on enough rows from the minority class. The loss function is binary cross entropy with Adam as the optimizer.

On the first training attempt of the model, this achieved a 98% accuracy. However, this metric is misleading due to the fact that the dataset is highly imbalanced. The precision for class 0 (no stroke) is nearly 99% while 0 for the minority class 1 (suffered a stroke). The algorithm learns to always predict 0 regardless of the input. This is a common problem for Deep Learning. There are several approaches worth discussing to overcome this.

One solution is to attach weights to each class[9][22]. Thus, the ANN becomes sensitive to the importance of the class: the cost function is punishing the network proportionally higher when it miscalssfies a row from the under-represented class.

The networks significantly improves when the weights are set proportionally to the class distributions {class 0:0.51 and class 1:27.71}. Training is stopped early after 32 epochs (fig:**??**). The classification report in figure 8 show how label 1 now has an 83% recall score (TP/(TP +FN)) but only 4% precision (TP/(TP + FP)). If this was a real medical test, it would correctly identify most of those patients who have a stroke, but would also wrongly misdiagnose 1/3 of healthy patients.

	precision	recall	f1-score	support
0	0.995230	0.735834	0.846097	8506.000000
1	0.059439	0.825581	0.110894	172.000000
accuracy	0.737612	0.737612	0.737612	0.737612
macro avg	0.527334	0.780707	0.478495	8678.000000
weighted avg	0.976682	0.737612	0.831525	8678.000000

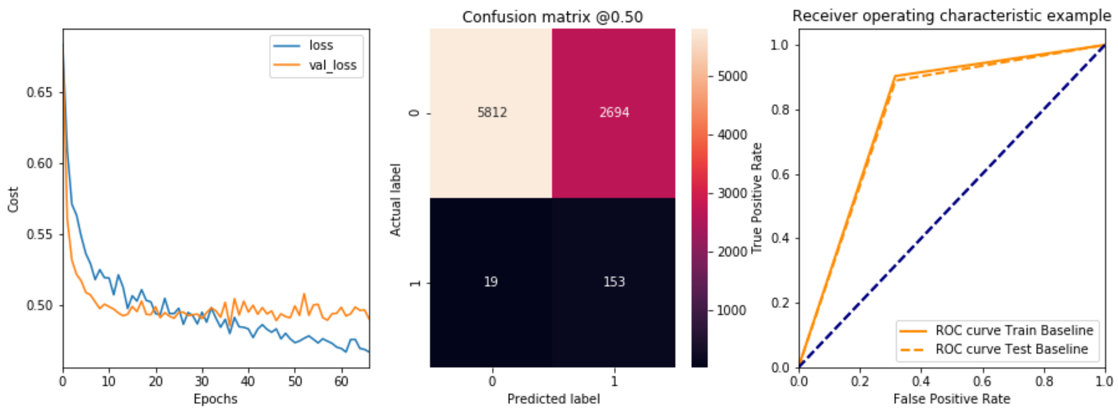


Figure 8: Weighted network

Bibliography

[1] Activation function - wikipedia. https://en.wikipedia.org/wiki/Activation_function.

[2] Ai is sending people to jail—and getting it wrong - mit. https://www.technologyreview.com/s/612775/algorithms-criminal-justice-ai/.

[3] Algorithm assessment report. https://data.govt.nz/use-data/analyse-data/government-algorithm-transparency-and-accountability/algorithm-assessment-report/.

[4] Alphago — deepmind. https://deepmind.com/research/alphago/.

[5] Backtesting - investopedia. https://www.investopedia.com/terms/b/backtesting.asp.

[6] Coding your own algo-trading robot - investopedia. https://www.investopedia.com/articles/active-trading/081315/how-code-your-own-algo-trading-robot.asp.

[7] Did uber steal google's intellectual property? — the new yorker. https://www.newyorker.com/magazine/2018/10/22/did-uber-steal-googles-intellectual-property.

[8] A graph of a cost function - researchgate. https://www.researchgate.net/figure/A-graph-of-a-cost-function-modified-from_fig1_329920042.

[9] How to develop a cost-sensitive neural network. https://machinelearningmastery.com/cost-sensitive-neural-network-for-imbalanced-classification.

[10] Kaggle. https://www.kaggle.com//.

[11] Keras. https://keras.io/.

[12] Neural network - databricks. https://databricks.com/glossary/neural-network.

[13] A neural network in 13 lines of python (part 2). https://iamtrask.github.io/2015/07/27/python-network-part2/.

[14] Professor's perceptron paved the way for ai – 60 years too soon. https://news.cornell.edu/stories/2019/09/professors-perceptron-paved-way-ai-60-years-too-soon/.

[15] Stroke dataset. https://www.kaggle.com/asaumya/healthcare-dataset-stroke-data/.

[16] The triple jeopardy of ke xu, a chinese hedge fund quant .... https://www.bloomberg.com/news/features/2018-11-19/the-triple-jeopardy-of-ke-xu-a-chinese-hedge-fund-quant.

[17] Trolley problem - wikipedia. https://en.wikipedia.org/wiki/Trolley_problem.

[18] Uber and waymo settle autonomous driving tech lawsuit. https://www.wired.com/story/uber-waymo-lawsuit-settlement/.

[19] Well-funded autonomous vehicle startups — crunchbase. https://www.crunchbase.com/lists/well-funded-autonomous-vehicle-startups/f2214864-27b0-47cf-b596-257602ab8145/organization.companies.

[20] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[21] Michael A. Nielsen. Neural networks and deep learning, 2018.

[22] S. Wang, W. Liu, J. Wu, L. Cao, Q. Meng, and P.J. Kennedy. Training deep neural networks on imbalanced data sets. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2016-, pages 4368–4374. Institute of Electrical and Electronics Engineers Inc., 2016.