

To what extent can we benefit from machine learning in terms of prognosis of diseases?

Shuntian Liu

Dulwich College, London

Word count: 4835

Declaration: I confirm that the work here is my own and is submitted on 4/7/2018 in completion of the CREST Gold Award.

To what extent can we benefit from machine learning in terms of prognosis of diseases?

Shuntian Liu

Dulwich College, London

# ABSTRACT:

I downloaded a dataset from the Internet and applied several machine learning algorithms on it in order to see if the program was able to make predictions, which are potential symptoms that a user might be suffering from in the future, based upon the given input. I estimated the performance by reshuffling the dataset and trained the algorithm again before testing it on the test set. The final result obtained was 56%. Although the result is not ideal, the improvements of the program can be seen along the way when I am making adjustments in different versions of my algorithm. The implication of this project is that if more medical research effort can be put into this area, which implies that more reliable and professional data can be collected, machines would be able to give a reasonably good prediction to users as a gentle warning before an actual symptom occurs.

To what extent can we benefit from machine learning in terms of prognosis of diseases?

# INTRODUCTION:

The question being investigated in my Crest Project is the application of artificial intelligence in the field of medical prediction and treatment. In this project, efforts will be focused on building a program that can facilitate families with little professional medical knowledge.

A well-posed definition by Tom Mitchell for machine learning is that, “a computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.” (Mitchell, 1997) In other words, this is the process that the machine is training itself through a large amount of data and adjusting parameters according to the measurement P to improve the accuracy of its predictions.

Since the middle of the last century, researchers have explored the potential applications of intelligent techniques in every field of medicine. (Lusted, 1955). The application of AI technology in the field of surgery was first successively investigated by Gunn in 1976, when he explored the possibility of diagnosing acute abdominal pain with computer analysis. (Gunn, 1976) So far, several systems have been developed to help analyse and process data, such as Artificial Neural Networks, Fuzzy Expert Systems Evolutionary Computation and Hybrid Intelligent Systems. Moreover, some projects carried out by universities and big companies are also being developed. A team at Stanford University in California recently unveiled a machine-learning algorithm trained to scrutinise slides of cancerous lung tissue. The computer learned to pick out specific features about each slide, like the cells’ size, shape and texture. It could also distinguish between samples from people who had only lived for a short time after diagnosis – say, a few months – and ones from those who survived much longer. The study verified the algorithm’s results by testing it on historical data, so now the AI could in principle be used with patients. There are also examples where a technology company is working with a national authority to achieve better medical analysis. Google DeepMind, based in London, is using masses of anonymised data from the UK’s National Health Service to train an AI that will help ophthalmologists. The aim here is to spot looming eye disease earlier than a human can. (Aviva, 2016)

Apart from academic research done by universities, applications that are designed for people to diagnose their health conditions have also been made possible by scientists and doctors in recent years. A range of choices including Ada, Dr. AI and YourMD, all aim to make credible medical advice more accessible and lead you through a clinical-grade triage process by using the principle of AI. (Douglas , 2017) These applications are trained by experts in the fields of both medicine and AI to make predictions as accurate as possible and increasingly precise by learning themselves from multiple real-life situations. There are various benefits of using these apps: (1) users are able to have a brief understanding of the seriousness and receive suggestions for further steps to take; (2) reducing the workload on doctors by offering patients the choice of consulting AI; (3) machines’ performance will not decrease over time but humans do need rest for better efficiency. Overall, the principle idea of this program is that it can act as an assistant for people who do not have professional medical knowledge to track and prevent potential diseases.

Despite all the progress that has been made so far in medicine using machine learning, there remain some problems. For example, although there are lots of software that can help people diagnose their existing diseases, there seems to be few tools that can tell people in advance what symptoms they might have and what corresponding precautions need to be taken based upon users’ daily activities and what medical conditions they have. However, the actual situation is that a large number of people who live in areas that are not well-developed can have few accesses to good medical resources, nor do they have good medical knowledge to prevent certain diseases. The consequence of that is lots of people are suffering from serious diseases that could have been prevented if they had a local GP to consult with or some gentle warnings beforehand. In order to resolve this issue, a program can be built which is publicly available to everyone so even people with little help can have access to some sources that can help them medically. Output given by such a program might be imprecise at the beginning but with increasingly more interactions with users, the advice could become more and more accurate. Therefore, improvements need to be made on how to give people early warnings before the actual symptom occurs.

# METHOD:

It is difficult to achieve this target, and the only way is to analyse a large dataset which includes lots of records about the daily activities of people, although one issue is that information in the dataset can vary a lot. Despite a small number of choices of data, there are many machine learning algorithms available, for example, logistic regression, support vector machine and neural networks. There does not exist a criterion to choose the best algorithm, however, neural network is used in this case considering its ability to recognise very complex patterns between different variables.

Since I am doing an individual project, I need to make a plan for myself on how many steps and what I need to accomplish at each stage. My initial plan is that I need to spend some time learning machine learning algorithms from various resources to obtain a basic understanding of principles behind those algorithms. The next step would be to find a suitable dataset online. The reason that I put this step after having learned the machine learning knowledge is that it would probably not be easy to find the appropriate dataset if I knew nothing about this area at all. After the suitable dataset is found, I will see what processing I can do on it to make it suit better into the algorithm. Finally, I can apply the algorithm to the processed dataset which would consist of lots of steps, the details of which will be in the later part of the report.

To begin with, I looked for datasets from different sources, aiming to find a dataset that includes people’s everyday activities and any diseases that they are suffering. After some thorough checking, I decided to use this dataset obtained from Kaggle (Chronic illness: symptoms, treatments and triggers, 2018), and the original provider of this dataset is Flaredown (FLAREDOWN, n.d.). For the full dataset, please refer to the website stated above (full URLs in bibliography). Here is an extract of the dataset: 

A brief explanation from the website Kaggle about the data is the following: (Chronic illness: symptoms, treatments and triggers, 2018)

**Instead of coupling symptoms to a particular illness, Flaredown asks users to create their unique set of conditions, symptoms and treatments (“trackables”). They can then “check-in” each day and record the severity of symptoms and conditions, the doses of treatments, and “tag” the day with any unexpected environmental factors.**

**User: includes an ID, age, sex, and country.**

**Condition: an illness or diagnosis, for example Rheumatoid Arthritis, rated on a scale of 0 (not active) to 4 (extremely active).**

**Symptom: self-explanatory, also rated on a 0–4 scale.**

**Treatment: anything a patient uses to improve their symptoms, along with an optional dose, which is a string that describes how much they took during the day. For instance, “3 x 5mg”.**

**Tag: a string representing an environmental factor that does not occur every day, for example “ate dairy” or “rainy day”.**

**Food: food items were seeded from the publicly-available USDA food database. Users have also added many food items manually.**

**Weather: weather is pulled automatically for the user's postal code from the Dark Sky API. Weather parameters include a description, precipitation intensity, humidity, pressure, and min/max temperatures for the day.**

**If users do not see a symptom, treatment, tag, or food in our database (for instance “Abdominal Pain” as a symptom) they may add it by simply naming it. This means that the data requires some cleaning, but it is patient-centred and indicates their primary concerns.**

There are some problems regarding this dataset, which need to be handled before machine learning algorithms can be applied. For example, some bits of information are missing from certain users, and most of the information is in the form of a string, which is not suitable for machine learning as numbers are usually the preference. Therefore, it is essential to pre-process the data so that the efficiency of the algorithm can be ensured.

After proper processing is carried out on the raw dataset, machine learning algorithms will be used on the reformatted dataset. There are lots of options available in terms of what machine learning algorithm to use, and it is hard to tell in advance which algorithm is the best one for this dataset. I decided to use neural network due to the following two facts:

* firstly, with increasing computing power, neural network is now able to learn more complex patterns from the data in a smaller amount of time;
* secondly, as the digital age arrives, a lot more data is produced every day than in the past, which is exactly what neural network needs.

The actual code used in this project will both be implemented by myself and downloaded from the Python library Scikit-learn (Pedregosa et al, 2011).

After algorithms are applied to the data, results can be analysed and performance of the algorithm may be improved using several methods, such as tuning network, namely, improving the neural network performance by changing hyper-parameters of the algorithm, and more data can be included to obtain a better performance as well. More details of how to tune the network can be found in results and analysis. For the source code, please refer to the appendix.

There are few hazards or safety issues as all data collected from users by Flaredown is anonymous. For possibly wrong predictions of diseases, we do not need to take this into consideration until the program is published and used by customers.

# RESULTS AND ANALYSIS:

As machine learning cannot have significant improvements without trial and error, I have altogether six versions of the program to demonstrate the process of analysing results and making adjustments accordingly. Because there are two main steps involved in my program, which are processing data and applying the algorithm. I will mainly be considering these two steps and the differences from one version to another.

## Version1:

I decided to keep the first version of my algorithm as simple as possible. Considering there are too many different types of conditions, tags and symptoms, I analysed the frequency of those “trackable types”, in other words, the input data, and selected those types with the highest frequencies only. For example, if a condition only appears once in the whole dataset, then it will be discarded and not used by the algorithm. Data that are incomplete will be removed as well.

Because the objective of the program is to make predictions on symptoms by taking conditions and tags as input, the raw data is consequently grouped when a symptom appears. Concretely, a symptom is used as the y-value and all tags and conditions preceding it will be considered as x-value. (x and y correspond to input and output). For the actual result, please refer to Version1/rearrangement.txt in the appendix.

Conventionally, neural network will have the same number of inputs for every sample. Due to this restriction, some data are further discarded. Only the two most frequent conditions and tags are left in a sample and if there are fewer than two conditions or tags, then the whole sample will not be used. Here is a snapshot that shows the final result of the processed dataset:



After all of these reformatting, the data are ready to be converted into numbers for the algorithm to learn from them. The way I converted them to numbers is that every group will become one sample, and other variables, such as age, gender or countries will all be mapped to a corresponding number, which will either be a cardinal number that stands for how many of something there are, such as how severe a condition is or an ordinal number which identifies the presence of a certain status, such as being tired. Here is a look at the result:



Finally, the neural network algorithm is applied to the data. This algorithm is adapted from a Coursera Jupiter notebook (Planar data classification with one hidden layer v5, n.d.). As it is the first neural network, it does not have many hyperparameters and there is only one hidden layer with the size of 4. After 10000 iterations, the algorithm’s performance on the train and dev set are:



A few things to specify: the cost is computed by the Cross-entropy loss function, which is a way of measuring the loss of a classification algorithm. And the performance is calculated based on how many correct predictions there are in all cases.

It can easily be seen that the accuracy is not satisfactory, apart from this, there are lots of drawbacks in version1, for example, lots of information is ignored and many parameters in the neural network are not properly adjusted.

## Version2:

Before tackling the problem that not enough information is included in the dataset, I also found out that the original dataset is not ordered chronologically, however it is clear that an activity will normally contribute to a later symptom, therefore it is important that data is arranged by time.

I also noted that there are lots of symptoms that are consecutive, namely, there are multiple y-values that the program needs to output. In order to keep the algorithm simple at the beginning, I eliminate all symptoms that have lower severity among consecutive y-values and keep the one with the highest severity as it might be easier for the algorithm to predict.

Next, I tried to include all “trackable types” in the raw dataset. There are many ways of doing this. The method I used, taking condition as an example, is to produce a list that consists of all conditions in the dataset, and then assign every condition a unique ordinal number. In this way, a condition vector can be created. If we have 200 conditions altogether, then a vector of size 200 is defined. If a particular condition is present in a sample, then the appropriate index of the vector will be assigned to the severity of that condition. For example, if the condition “hip pain when getting up” corresponds to the 1st condition and the severity of “hip pain when getting up” is 3, then the first item of the vector will be equal to 3. If a condition is not present, then it will remain zero.

The operation is carried out on tags and symptoms, though the vector of them will only consist of zeros and ones to identify whether something exists or not.



In this way, every single variable will be included in the processed data, leading to a dataset that is too large to be trained in a reasonable amount of time. I have to reduce then number of samples in order to speed up the machine learning algorithm.



Final results obtained from this version are even worse than the first version (shown below). Possible causes might include: there are too many factors that are affecting the algorithm, some of these might be negatively affecting the algorithm since some of these factors will appear only several times or even just once so that they should really have been ignored. Additionally, the algorithm is also not complex enough to learn from such a massive dataset.



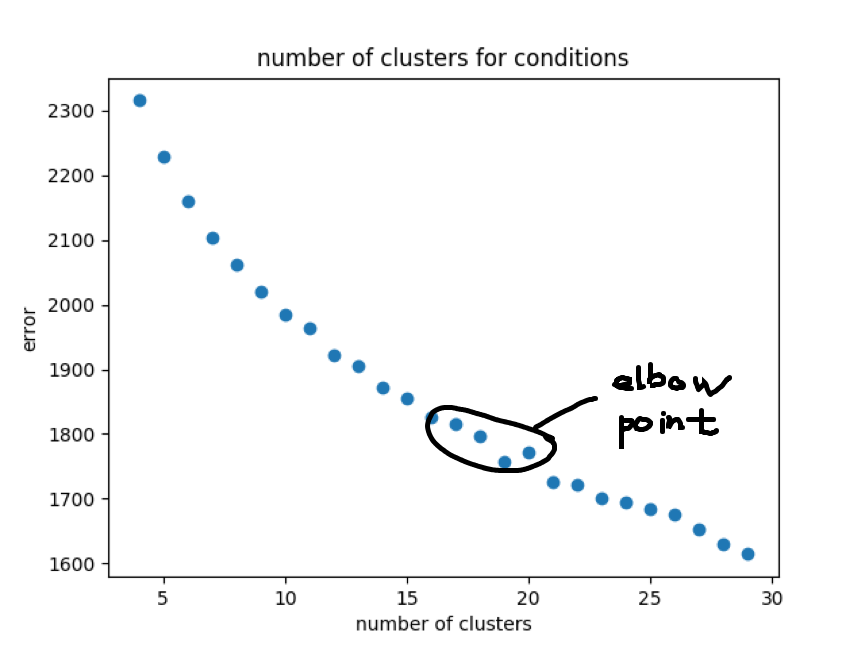
## Version3:

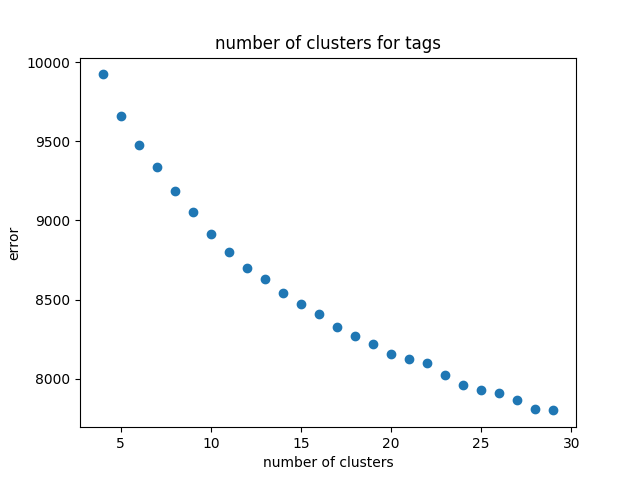
(From this version on, I started to use code from the Scikit-learn library)

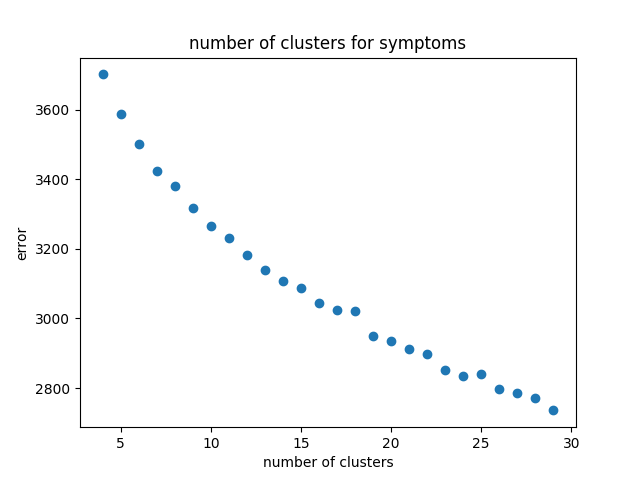
Having seen the effects of a too large or too small dataset, I realised it is crucial to ensure that the size of the training set is chosen carefully. Considering that most of these data are typed by users of the app Flaredown themselves, it would be worthwhile to use natural language processing as part of the data processing. The pre-trained word vectors, which are numerical representations of different words, used in NLP is downloaded from a Biomedical natural language processing website (Pyysalo et al, 2013) which provides various language resources created from the entire available biomedical scientific literature, a text corpus of over five billion words. Please refer to PMC-w2v website stated above for full details of the word vector.

After word vectors are obtained, every “trackable-name”, represented as a string, will be averaged into a single word vector. For instance, every word in the condition “pain in face and jaw” will have its own word vector, and the final word vector used in the algorithm will be the average value of these five vectors. This process turns all the natural language typed by the user into numbers, which are better understood by computers. However, this still doesn’t solve the problem that there are too many inputs.

Clustering algorithms can be used here to tackle this issue. Specifically, another machine learning algorithm called ‘k-means clustering’ is used here in order to group different “trackable-names”. The point of doing this is that, for example, the condition “insomnia” and “sleep disorder” could possibly be grouped together and treated the same by the algorithm. However, a new problem caused by using k-means is that the number of clusters has to be determined by the programmer. This is not a straightforward task to accomplish since it requires lots of medical knowledge to see whether the clustering is reasonable or not and it does add more complexity to the program which is beyond the scope of this project. But by looking at the graph of error against number of clusters, I chose the “elbow point” provisionally where the tangent starts to slow down, as this is how the choice is made conventionally. However, I used a more systematic method to tackle this problem later on, details of which can be found in the extra version.







When it comes to measuring the performance of this new method, I adopted a new measurement to estimate the accuracy. As the y-value is a vector that comprises of only one 1 and all others are 0s, a function that compares every element in the output and records every correct prediction made by the algorithm and the percentage of correct predictions can be calculated. The result is unexpectedly high, which is:



Despite this high performance on both the training and test set, it is still worth considering the number of 0s and 1s in the actual y-value. Since the actual y-value is dominated by zeros, we can easily obtain a very high performance even if the output is always zero! Due to this fact, another method to properly measure the performance needs to be used to deal with this biased data.

## Version4:

The most common way to handle biased data is to introduce precision and recall to give an indication of how well the algorithm is doing. The definitions of precision and recall are: (Precision and recall, n.d.)



Apart from using a new estimation of the performance, there are also two other improvements in this version. In version2, I only kept the symptom with the highest severity among consecutive symptoms. In fact, a neural network algorithm is able to do multi-task learning, which can identify more than one symptom. This can include more information from the raw dataset rather than simply choosing the one with the highest value, which can potentially make the model more accurate. The second improvement is that some samples in the dataset are not good, for example:



These two samples only have output but no input, therefore it is reasonable to discard them to improve the performance of the algorithm.

After all of the improvements above have been made, I obtained the following result:

With two numbers in the bracket standing for precision and recall respectively.

Performance on the dev set decreased a lot compared with the last version, but it is a better estimation. However, further improvements can still be made.

## Version5:

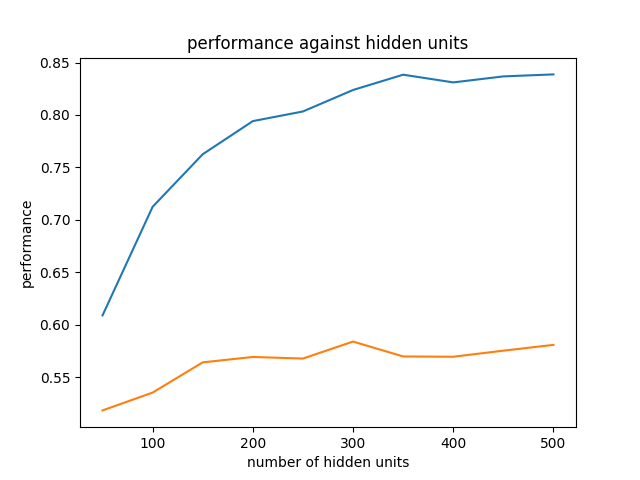
Before tuning the neural network, a useful technique is applied in order to allow the algorithm to “see” more data before being tested on a new set. I shuffled the whole dataset before splitting it into training and dev set so that we do not need to worry that the algorithm will under-perform due to the lack of comprehensive data we feed into it. And it does prove to be useful according to the new result:(three numbers following train are precision, recall and harmonic mean of the previous two numbers respectively)



Once shuffling is done, I started to modify the hyperparameters of my neural network. The main hyperparameters that need modifications are: regularisation term (parameter that penalises the overfitting of the algorithm), hidden layer size (number of units in a hidden layer) and learning rate of the algorithm (how fast the algorithm is learning). Here is the process of tuning these three parameters:

Hidden layer size:

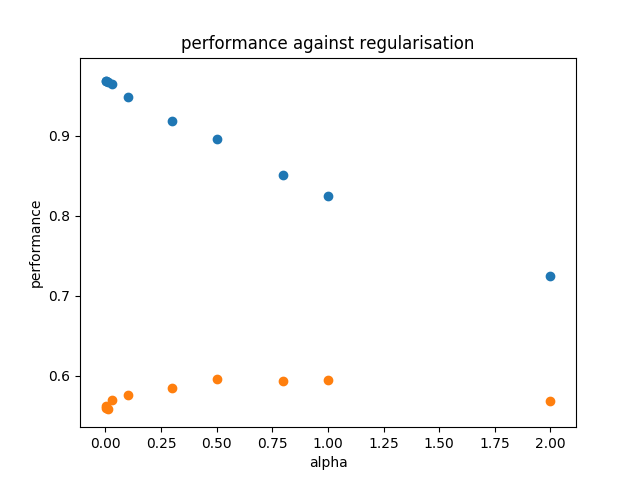
By looking at the result in version4, it can be clearly seen that the algorithm is overfitting the dataset, i.e. it is over-describing the data rather than learning the general trend. Therefore, a smaller number of layers needs to be used. I will stick to only one hidden layer since the number of samples in my dataset is only about 6000, which is not huge. After making this decision, I need to adjust the number of hidden units to achieve the best performance. The graph is plotted: (with the blue one being training and orange one being dev)



It is not difficult to see that when the number of hidden units is equal to 300, the algorithm has the best performance on the dev test, therefore the number of hidden units for the first and the only layer is chosen to be 300.

Regularisation term alpha:

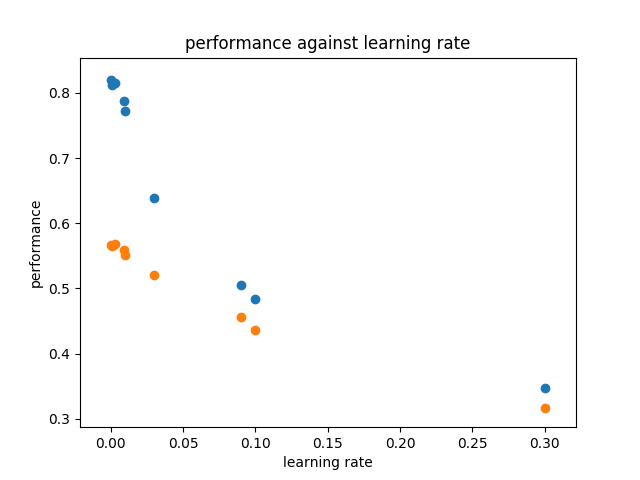
By applying a similar approach, the following graph is obtained:



Again, the scatter graph shows that, on the dev set (orange points), the algorithm is best performed when alpha=1.00, so the 1.00 is used as the regularisation parameter.

Learning rate:

The last parameter to be tuned is the learning rate and the graph is as shown:



The performance of the program on the dev set declines as the learning rate increases. However, the first few points are very close together. Therefore, considering the running time of the program, a small value of the learning rate is chosen, which is 0.001.

After the tuning process is done, new values for the performance on the training and dev test is obtained again:



The improvement on the overall performance is around 4%, which is not huge. However, a major improvement on the precision on the dev set can be seen: from 63% to 73%.

## Extra Version:

This is an extra version for deciding the number of clusters for all conditions, tags and symptoms. In k-means algorithm, “elbow point” is conventionally chosen to be the number of clusters to be used. However, in this particular case, it might be worthwhile to consider whether the clustering result makes medical sense, i.e. will a human doctor do the clustering in a similar way if given a certain number of conditions? Unfortunately, I do not have access to such medical resources to help me with my clustering algorithm, therefore, I carried out a test on the performance of the algorithm with different numbers of clusters.

The range of values for the clusters I tried is [40,100]. The impact of the number of clusters on the performance is not so significant, as shown by the result of plugging different values into the algorithm (performance ranging from 54% to 66%). Details of this tuning can be found in the file esult.txt in the appendix, here is an extract.



This test is not particularly relevant to the original aim of my project, but it is considered to be an extension to other versions in order to make the project more practical. It is also an important improvement that could be made in the future which will be discussed in more detail in the Discussion section.

A brief summary of all six versions of my program: the program starts from simple by removing some features with less frequency and builds up the complexity by processing natural language as well as clustering similar features and discarding data that are not helpful for the algorithm to learn the pattern. The final achievement is tested on a test set by reshuffling the whole dataset in a different way to prevent overfitting, and the performance achieved is:



# DISCUSSION:

It can be seen that, from version to version, different improvements have been made, some were very useful in terms of increasing the performance of the program, some were not. The overall result is that the final performance is 43% higher than the initial one.

As summarised above, the final performance on a test set achieved by the algorithm is approximately 56%, which is not an ideal value. However, when estimating the performance of a machine learning algorithm, it is crucial to compare it with the so-called human level performance, namely, how well a human expert, or a panel of experts can do on this task. Additionally, there is a fundamental concern about this dataset: is it correct to use conditions and tags as input to predict symptoms? Again, it would require a lot more research efforts to estimate how suitable the input is chosen, which is not accessible to my project. Despite this limitation, 56% is a respectable result to give people some reference.

Size of data

Machine learning algorithm usually requires a very large amount of data to achieve a good performance. Generally, having more data will always result in better accuracy. The dataset I used is not particularly large considering the huge amount of data collected by large IT companies from people’s daily online activities nowadays. The quality of the data is not guaranteed either due to the need to modify them in order to make them more suitable for the algorithm to run on. Moreover, some samples in the dataset with very low quality are also discarded to prevent the algorithm from learning from “outliers”.

What other improvements could have been made?

There are some other methods in the deep learning domain that could have been used, such as using the recurrent neural network to deal with the natural language processing which enables the program to take the order of words into consideration.

Different models of machine learning algorithms could also be adopted such as the SVM algorithm and so on, which cannot learn such complex patterns as neural networks do, but they could perform well on this particular problem.

Last but not least, some features have always been ignored throughout the processing, such as “food” or “weather” due to their small number. If the data collected has a larger proportion of these features, they could be included to give algorithms more information, which might also give a positive feedback.

Implications

Although the result of my algorithm is not very satisfying, it is undoubtable that machine learning will thrive in medicine. Future research efforts could be put into collecting professional medical data by large organisations as well as seeking assistance from medical experts in what parameters to include in the algorithm to make its predictions more reliable and accurate. With all that being said, though it is not practical to produce an algorithm that is very medically beneficial individually, it is still very possible for large companies to accomplish this task due to the fact that improvements can be seen over all versions of a very simple program with some data that are not collected specifically for this purpose.

# REFERENCES

Aviva, R. (2016, August 31). *Smart medicine is coming of age, but will doctors bite?* Retrieved from New Scientist: https://www.newscientist.com/article/mg23130892-500-smart-medicine-is-coming-of-age-but-will-doctors-bite/

*Chronic illness: symptoms, treatments and triggers*. (2018). Retrieved from kaggle: https://www.kaggle.com/flaredown/flaredown-autoimmune-symptom-tracker/data

Douglas , H. (2017, July 19). *An app a day keeps the doctor away*. Retrieved from New Scientist: https://www.newscientist.com/article/2141247-an-app-a-day-keeps-the-doctor-away/

*FLAREDOWN*. (n.d.). Retrieved from FLAREDOWN: http://flaredown.com/

Gunn, A. (1976). The diagnosis of acute abdominal pain with computer analysis. *J R Coll Surg Edinb, 21*, 170–2.

Lusted, L. (1955). Medical progress – medical electronics. *N Engl J Med, 252*, 580–5.

Mitchell, T. (1997). Machine Learning. McGraw Hill.

Pedregosa et al. (2011). Scikit-learn: Machine Learning in Python. *JMLR 12*, 2825-2830.

*Planar data classification with one hidden layer v5*. (n.d.). Retrieved from Coursera: https://hub.coursera-notebooks.org/user/jyvluiohpnatetiyufkfkp/notebooks/Week%203/Planar%20data%20classification%20with%20one%20hidden%20layer/Planar%20data%20classification%20with%20one%20hidden%20layer%20v5.ipynb

*Precision and recall*. (n.d.). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Precision\_and\_recall

Pyysalo et al. (2013). *Biomedical natural language processing*. Retrieved from Biomedical natural language processing: http://bio.nlplab.org/

# APPENDIX

For all codes and relevant files mentioned in the report, refer to the website <https://github.com/Vincent-lau/flare-down>