Umeå University
Department of Physics

19 september 2021

Evaluating machine learning models for predicting glioma from single nucleotide polymorphism data

Author: Tim Anthony

Bachelor thesis in Physics(15 ECTS)

Supervisor: Martin Rosvall

Sammanfattning

Tidig upptäkt av cancer är nödvändigt för att minimera psykiskt och fysiskt lidande. Därför undersöktes det i denna rapport om möjligheterna att använda maskininlärning för att upptäcka gliom i ett tidigt stadie. Detta genom att kolla på genetiskt data från verkliga patienter. Dessa data består av över 14 miljoner genetiska variabler (eng: features) av så kallade SNP:s, och anses därför vara högdimensionellt. Frågan är dock om detta data är möjligt att använda för prediktion av gliom?

Tillvägagångssättet som tillämpades var att först reducera dimensionen med metoder som weighted cosine similarities, PCA, undercomplete autoencoder, t-SNE och sum pooling. För att göra prediktion så användes k-means, naive bayes, kNN och neurala nätverk.

Resultat från denna studie visar på att man med metoderna som nämnts ovan kan ha svårt att bestämma cancerrisken. Detta kan däremot bero på dem hyperparametrar som använts, då dessa spelar stor roll för hur välfungerande metoderna är. Några viktiga hyperparametrar var antalet noder och lager i undercomplete autoencoder och det neurala nätverket. Att använda för många noder eller lager skulle inebära att dessa modeller blir övertränade. Å andra sidan, att använda för få noder eller lager skulle inebära att dessa modeller istället blev undertränade. Perplexiteten i t-SNE och antalet block i sum-poolingen var också nyckelparametrar, dessa två hyperparametrar var svåra att justera på lämpligt vis eftersom rutnäts-sökningen var väldigt kostsam tidsmässigt sett.

Abstract

Early detection of cancer is necessary to minimize mental and physical distress. Therefore, this report investigated the possibilities of using machine learning methods to detect glioma in an early stage. This by looking at genetic data from real patients. This data consists of more than 14 million genetic features called SNP:s, and is therefore considered highly dimensional. However, the question is if these genetic data can be used for prediction of glioma?

The approach used was to first reduce the dimension by methods such as weighted cosine similarities, PCA, undercomplete autoencoder, t-SNE and sum pooling. To make prediction, k-means, naive bayes, kNN and neural networks were used.

The results of this study show that the methods mentioned above can be difficult to use in determining the risk of cancer. However, this may depend on the hyperparameters used in the models as these play a major role in performance. Some important hyperparameters were the number of nodes and layers in the undercomplete autoencoder and the neural network. Using too many nodes or layers may cause these models to overfit. Contrary, using too few nodes or layers may instead cause them to underfit. The perplexity in the t-SNE and the number of blocks in the sum pooling were also key parameters, these two hyperparameters were hard to tune as well since the grid search was very costly time-wise.

Acknowledgements

First of all I want to thank my supervisor Martin Rosvall for his support during all phases of this thesis. I also want to mention Andrea Lancichinetti who was involved in the project. He did not only came up with many interesting ideas, but also engagingly explained them to me. He is also the one responsible for some code used in this thesis, especially the cosine similarity code. Moreover, I want to thank Anna Dahlin for helping us out with the data. I also want to thank my classmate Anton who took his time discussing ideas with me in the beginning of this project, and for giving me some well-used insights about PCA. My gratitude also goes to my girlfriend Iylia for proofreading this report.

Finally, I am sincerely grateful to all the people in Icelab. You welcomed me warmly to the office and made me feel included from the very first day.

Contents

1 Introduction

Glioma is a type of cancer responsible for approximately 30% of all tumors in the brain and central nervous system (CNS) and 80% of all malignant brain tumors[1]. This corresponds to around 19,500 glioma diagnoses each year, in the US only [2]!

To the minimize psychological and physical suffering, early detection of the cancerous cells is essential[7]. Due to medical examinations being expensive and time consuming, glioma detection is not performed unless the disease already is suspected in the patient. Therefore, we wish to use computational methods on already available genetic data. However, it remains unclear whether these data can be used to determine if the patient poses a high risk for glioma?

The aim of this report is to investigate various machine learning methods for predicting glioma diagnosis, which also is know as the patients phenotype. This by using the nucleotide mutations, so called SNP (Single-nucleotide polymorphism) data, in the patients genotype. Today this is done by using simpler models, such as logistic regression models[5, 6], which unfortunately does not perform all too well. There have also been predictions on the risk of different types of glioma, but this is based on only 25 known glioma risk SNP:s [5]. What I hope to find is a more efficient prediction method that can be used for detecting the disease. But based on a larger part of the genome, and by using more sophisticated machine learning models. This could then lead to early detection of glioma, which in turn could help doctors make the right decision in critical situations. Using larger part of the genome could also result in new insights about which SNP:s, and also combination of these, that result in a high glioma risk.

The methods used in this report can be divided into two groups; dimensional reducing methods and prediction methods. For the first group weighted cosine similarities, PCA (Principal component analysis), undercomplete deep autoencoder, t-SNE (t-distributed stochastic neighbor embedding) and sum pooling method were used. For the second group, this report will cover the k-means algorithm, naive bayes, fully connected neural networks and the KNN (k-nearest neighbours) algorithm.

For the outline of this report chapter 2 contains a description of the data and the underlying methods used. In chapter 3 the result is presented along with a discussion how the result should be interpreted. Finally, in chapter 4 thoughts and insights about this project are summarized and also suggestion about what future development that could be possible.

2 Method

2.1 Data

The phenotype data consists of a vector with 1313 elements, representing the diagnosis of the patients. A '0' represent a healthy individual and a '1' represent a person with a glioma diagnosis. In our data two-thirds (876 persons) were healthy while one-third (437 persons) had tested positive for glioma.

For the genotype data, there is 22 files where each file correspond to a chromosome pair, disregarding the sex chromosomes. In all these files the rows correspond to the same 1313 persons as in the phenotype data, while the columns correspond to the different SNP:s of that chromosome pair. These columns can be seen as the features of our data. The individual elements consist of '0', '1' and '2' which is the number of lesser common gene variants of that SNP position. So a '1' mean that one of the chromosomes in the chromosome pair has a mutation, a '2' means that both chromosomes has a mutation at that position and a '0' means that neither of the chromosomes have a mutation at that position. Missing values are solved by inserting zeroes, since the zero indicate that the most probable nucleotide variation is present at that position for both chromosomes in the pair.

One limiting factor is the data size since each of the 22 chromosome files have between 191,898 features and 1,240,996 features, while the total number of features for all chromosome files are 14,458,459 features. This makes it hard fitting all the data into the RAM, and even harder to make computations. Therefore the methods described below (except the sum pooling, kNN and neural networks) will be used on each file separately to investigate if the chromosome pair corresponding to that file can be used for the prediction.

For all the prediction methods used below, a train/test-split of 80%/20% was used. Here the 80% stands for the fraction of the data used in the training phase, while the 20% stands for the fractionthe data used in the testing phase.

Due to the sensitivity of the patient data an agreement was written, stating that the data data at all times had to be located on a physical machine in Icelab.

2.2 Reducing dimension and making predictions

Many of the methods used in this report are feature reduction methods. The reason for this is the high dimensionality of the data (see 2.1) which could also lead to overfitting, a a common problem for prediction on genetic data [3]. Also, to be able to use prediction methods with constraints of finite time and memory it is therefore necessary to reduce the data. This reduction could in turn also help with the overfitting problem, and make the methods generalize better.

The order of applying the different methods to the original data is visualized in figure 1. Here the green boxes are dimension reduction methods while the blue boxes contain prediction methods. The reason for combining multiple dimension reduction techniques, in the order we did for the top two paths, was because of performance. This since we weren't able to run PCA, the undercomplete autoencoder, or the t-SNE algorithms on the raw data. Therefore, we first used the weighted cosine similarities to reduce the data. After this, we then used the PCA algorithm or the autoencoder, to decrease the dimension further. As our final dimension reduction method for the top two paths, we used t-SNE to reduce our data into 2 dimensions which enabled us to plot it. For the upmost path, we didn't include any prediction method. The reason for this was that the output data from the t-SNE here had no separation. Trying to find a separation boundary with a prediction algorithm was therefore meaningless.

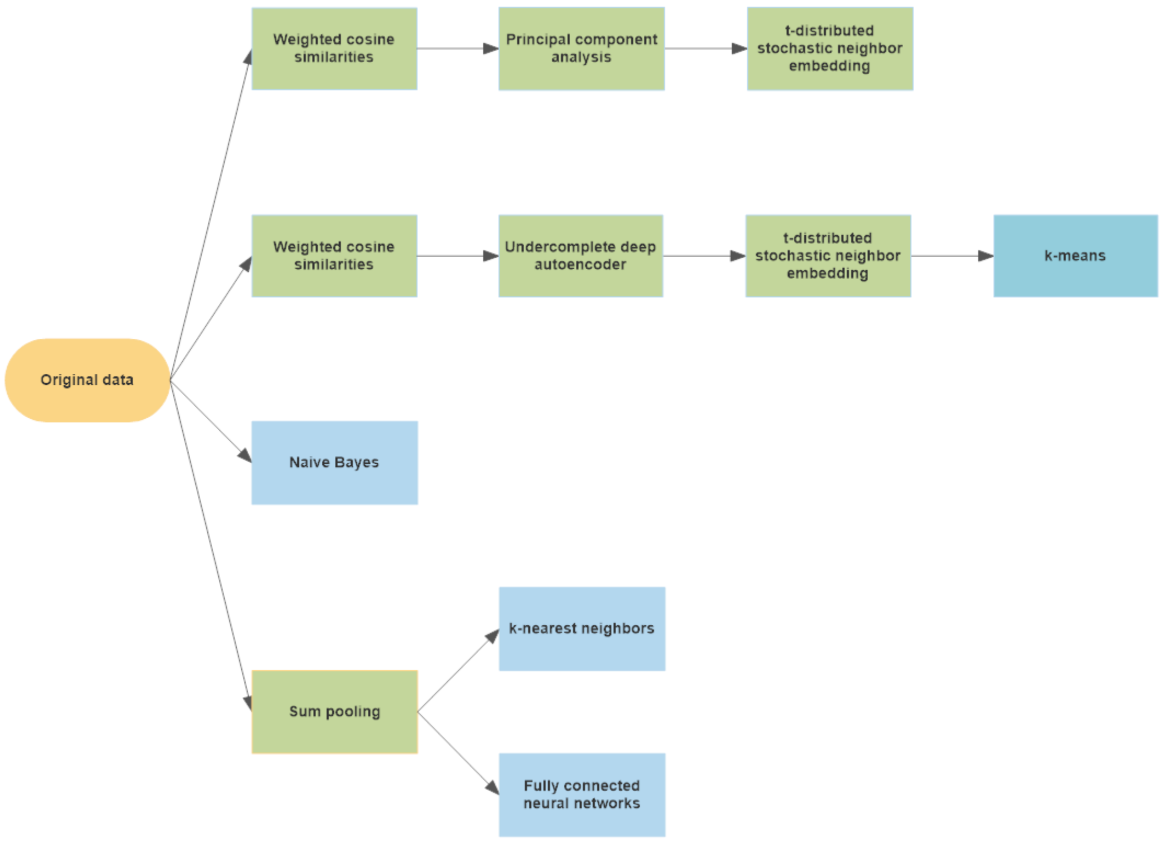

Original data
Weighted cosine similarities
Principal component analysis
t-distributed stochastic neighbor embedding
Weighted cosine similarities
Undercomplete deep autoencoder
t-distributed stochastic neighbor embedding
k-means
Naive Bayes
Sum pooling
k-nearest neighbors
Fully connected neural networks

Figure 1: Flowchart of when different methods were applied to the data. The green boxes contain dimension reduction methods and the blue boxes contain prediction methods.

2.3 Software

For this study we used python3 in Jupyter notebook . Also, we included the packages **numpy**, **pandas**, **keras**, **pickle** (for reading/writing to file), **sklearn** and **matplotlib** (for plotting).

2.4 Weighted cosine similarities

We used cosine similarities to pre-process the data and reduce the dimensionality since much of the SNP-data are correlated and have high redundancy. This is especially true for SNP:s who's position are close to each other in the chromosome. In this method we also consider all 2:s as 1:s in the genotype data, which can be seen as assuming cancerous SNP:s are dominant features.

The idea here is to group these similar SNP:s together, which we do by using graphs, and then use the groups instead of the individual SNP:s as our features. Thus reducing dimension. This grouping is done by putting links between SNP:s which are similar enough, which will give us separated groups of similar SNP:s. We will then use these groups as our features, where the belonging of observation patient i to group j is given by equation (2). In practice this is implemented by first looking at the similarity between pairs of the SNP data within 1000 positions of each other via the cosine similarity

$$similarity = \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{||\mathbf{v}_1||||\mathbf{v}_2||} \tag{1}$$

where $\mathbf{v}_1$ and $\mathbf{v}_2$ are the SNP-vectors to be considered. Here each vector $\mathbf{v}_i$ is containing all observations of SNP i. If the similarity is above 0.8, an undirected link is created between the two SNP positions. This process can thus be seen as forming a graph, where the linked subgraphs are called connected components.

From this point, the features of the data is the different connected components. The value for each individual person and feature is the number of SNP mutations the person has in the connected component, divided by the number of SNP:s making up the connected component. Considering a person with index i, and connected component j, we have

$$WeightedCosineSimilarities[i][j] = \frac{\text{SNP:s of i in connected component j}}{\text{SNP:s in connected component j}} \tag{2}$$

2.5 Principal component analysis

We used PCA as a dimension reduction method, to reduce the number of features while keeping as much of the information as possible. The idea is to transform the system linearly so that the information is kept in a few components, which then can be used as features while disregarding the less informative components.

The first step of PCA was done by first normalizing the chromosome data such that each feature, or column vector, gets mean 0 and variance 1. This by using the method 'StandardScaler' in numpy. The data was then transformed into a coordinate system where as much variation as possible were contained with respect to the first coordinate axis, which then was our first principal component. The same process was then repeated to create the following principal components, where each consecutive component would hold less and less variation.

A measurement for the effectiveness of a PCA transformation is the explained variance. This is the fraction of how much variance is retained in each principal component compared to the variance of the original data. Since the principal components are ordered in decreasing explained variance, one may decide how many principal components that is needed based on the degree of total explained variance one wants to achieve. In this report, this cutoff was set to be 80% of the original variance.

2.6 Undercomplete deep autoencoder

We used the undercomplete autoencoder as a way of reducing dimension. This by utilizing neural network in an unsupervised manner.

The idea is to use a deep neural network, which we train to predict the input data as accurately as possible in the reconstruction layer. If the output from the reconstruction layer is similar to the input we know that the structure of the data exists in all layers of our network. Hence we can use the lowest dimensional layer, also called the bottleneck layer, as our feature space. This lower dimensional data, or encoding, can now be used as our compressed data since it contains most information of the input. A structure diagram of an autoencoder can be seen in figure 2.

For this report I used an autoencoder with 4 hidden layers, similar to the one in figure 2. The hidden layers in the encoder consisted by a layer 200 neurons, followed by a layer of 50 neurons. The bottleneck layer had 20 neurons. The hidden layers in the decoder consisted by a layer of 50 neurons, followed by a layer of 200 neurons.

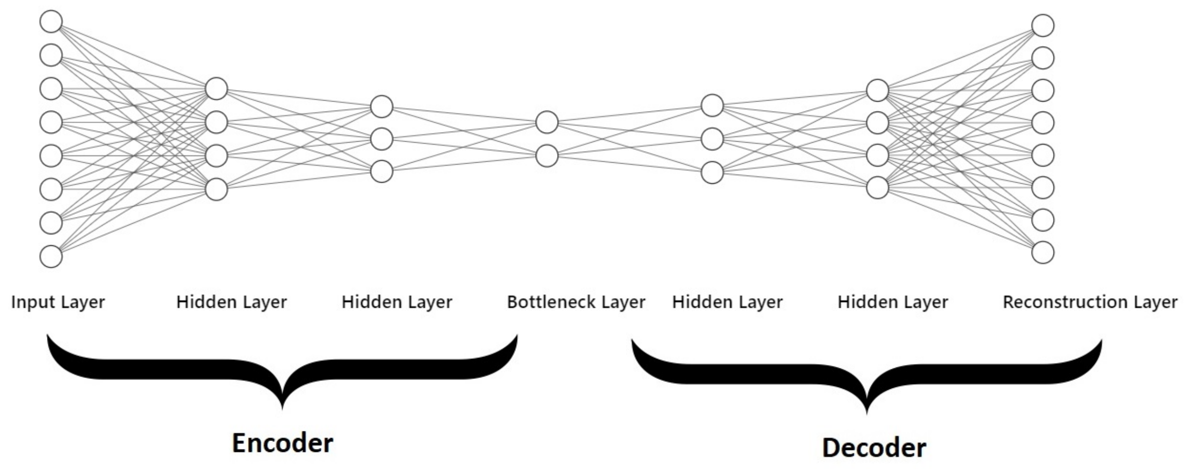


Figure 2: Diagram of a deep undercomplete autoencoder with 4 hidden layers.

2.7 t-distributed stochastic neighbour embedding

t-SNE is a method which is used to reduce the dimension of the data to two dimensions so that the data can be visualized. The algorithm does that by using a similarity measure, which is effective in preserving clustering from the high dimensional data to the low dimensional transform.

The only hyper-parameter for this algorithm is the perplexity, which affect the numbers of neighbours in the similarity measure [4]. The values I used for perplexities in this study are $1, 3, 5, 10, 20, 30$ and 100.

2.8 k-means

k-Means is an unsupervised learning algorithm which was used to identify clusters. This was done implementing Algorithm 1 described below. This algorithm can also be used for unsupervised learning by using the training data for identification of the two clusters, which are to represent the classes of healthy people and people with glioma. Then, for the prediction, all testing persons are assigned to the cluster centre that is closest to their feature vector.

Algorithm 1 k-Means

```
1: Let $dataPoints = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ be the $n$ data points
2: Let $clusters \in \mathbb{Z}^+$ be the number of cluster centres
3: Let $centresNew = \{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_{clusters}\}$ be the randomly initialized cluster centres
4: do
5:     Set $centresOld = centresNew$
6:     for $i = 1 : n$ do
7:         Assign $\mathbf{x}_i$ in $dataPoints$ to closest cluster centre in $centres$
8:     for $j = 1 : clusters$ do
9:         Set $\mathbf{c}_j$ in $centresNew$ as the mean of all $\mathbf{x}$:s assigned to that cluster centre
10: while $centresOld \neq centresNew$
```

2.9 Naive Bayes

I used the Naive Bayes algorithm as a supervised classifier due to its simplicity and efficiency. This method is based on conditional probabilities with the assumption that the different features have zero correlation, which is the reason it is considered naive.

Consider an observed sample $\mathbf{x} = (x_1, x_2, ..., x_n)$ with n features, which all are assumed to be independent. Furthermore, let C_j be one of the K possible outcomes, or classes, that our sample could belong to. Then the probability of the sample belonging to C_j can be described as

$$P(C_j|\mathbf{x}) = \frac{P(C_j)\prod_{i=1}^{n} P(x_i|C_j)}{Z} \tag{3}$$

where Z is a normalizing factor called evidence, given by

$$Z = p(\mathbf{x}) = \sum_{j=1}^{K} P(C_j)P(\mathbf{x}|C_j). \tag{4}$$

To estimate the conditional probabilities $P(x_i|C_j)$ in practice, we simply counted the fraction of occurrences of a specific SNP, i.e. the fraction of '0':s,'1':s or '2':s in a specific column, while conditioning both C_j='healthy' and C_j ='glioma'. Predicting the class of the sample $\mathbf{x}$ is then done by assigning it to the most probable class as

$$\hat{y} = \underset{j\in\{1,2,...,K\}}{\operatorname{argmax}} P(C_j)\prod_{i=1}^{n} P(x_i|C_j). \tag{5}$$

Note that we don't need the constant value Z since it is equal among all classes, and hence can be removed from the maximization expression.

2.9.1 Derivation of equation (3)

Starting from the expression

$$P(C_j, \mathbf{x}) = P(C_j, x_1, x_2, ..., x_n) \tag{6}$$

we use the definition of conditional probability, $P(A|B) = \frac{P(A,B)}{P(B)}$, repeatedly to obtain

$$\begin{aligned}
&P(x_1, x_2, ..., x_n, C_j) = \\
&P(x_1|x_2, ..., x_n, C_j)P(x_2, ..., x_n, C_j) = \\
&P(x_1|x_2, ..., x_n, C_j)P(x_2|x_3.., x_n, C_j)P(x_3, ...x_n, C_j) = \\
&\cdots = \\
&P(x_1|x_2, ..., x_n, C_j)P(x_2|x_3.., x_n, C_j)\dots P(x_{n-1}|x_n, C_j)P(x_n|C_j)P(C_j).
\end{aligned} \tag{7}$$

Now, applying the naive assumption that all x_i:s are mutually independent, which is equivalent to taking away the conditioning as $P(x_i|x_{i+1}, \dots, x_n, C_j) = P(x_i|C_j)$ we can rewrite equation (7) as

$$\begin{aligned}
&P(x_1, x_2, ..., x_n, C_j) = \\
&P(x_1|C_j)P(x_2|C_j)\dots P(x_{n-1}|C_j)P(x_n|C_j)P(C_j) = P(C_j)\prod_{i=1}^{n} P(x_i|C_j).
\end{aligned} \tag{8}$$

Looking at equation (6) again and conditioning using the definition of conditional probability, but this time by conditioning on $\mathbf{x}$, we get

$$P(C_j, \mathbf{x}) = P(\mathbf{x})P(C_j|\mathbf{x}) = P(C_j|\mathbf{x}) \sum_{j=1}^{K} P(\mathbf{x}|C_j)P(C_j) \tag{9}$$

where we in the last step used the law of total probability. Now, if we let Z be defined as above and equate equation (8) with equation (9) we retrieve equation (3), as wanted.

2.10 Sum pooling

Sum pooling is a dimension reducing technique used when suspecting that the number of mutations on a whole region of the data can be used for prediction, while disregarding the individual elements in that region. I applied this method by dividing the features of each file into 10, 100 or 1000 blocks. The number of mutations within each block was then counted, and the total sum were used as the new feature. Here we took into account that a mutation could be present at the same position for both chromosomes in the chromosome pair. Then the pooling results for all chromosome pairs were concatenated such that each person got a feature vector with length 22 times the number of blocks per file.

2.11 k-nearest neighbours

kNN is a simple prediction method. For the testing phase, the k training vectors closest to the each test vector in feature space were noted. The predicted label of each test sample was then decided by the dominant label of the neighbouring k training vectors. In this report k-values between 1 and 40 were used.

2.12 Fully connected neural networks

Neural networks is a supervised prediction method. It consists of neurons ordered in different layers, with weights between the neurons in adjacent layers. During the training phase, training data is presented to the network that calculates the predicted output. This is called the forward pass. When the network predicts the wrong label of the training data during the forward pass, the weights are updated with an algorithm called backward propagation, that propagates the error backwards through the different layers and updates the weights accordingly.

In this report 2 layers with 1000 nodes each was used.

3 Result and Discussion

3.1 Naive Bayes

For the Naive Bayes method, the result can be seen in figure 3 (for exact values, see table 4 in Appendix A). There the evaluation accuracies are plotted for each file. A reference accuracy is also included, which is the accuracy one would obtain if always guessing the patients were healthy. As all the chromosome evaluations were lower than the reference, Naive Bayes is not a viable way of predicting glioma.

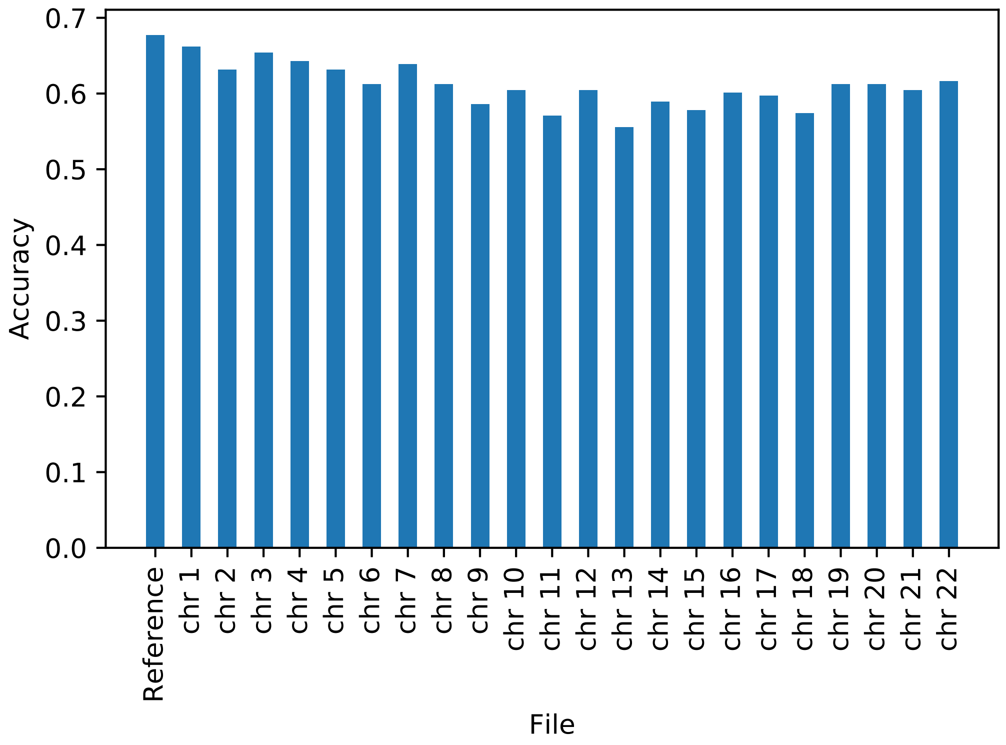
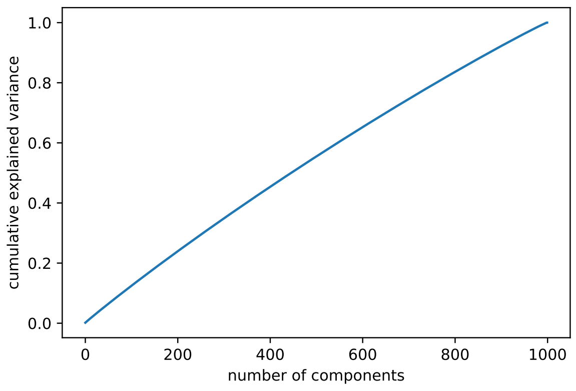


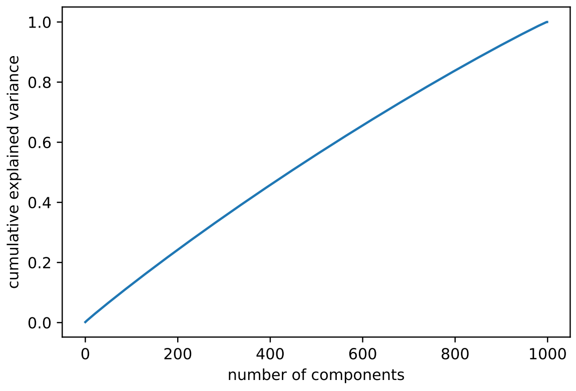
Figure 3: Accuracies of Naive Bayes for the different chromosome files, including the reference accuracy if one always would guess the patients were healthy.

3.2 Weighted cosine similarities + PCA + t-SNE

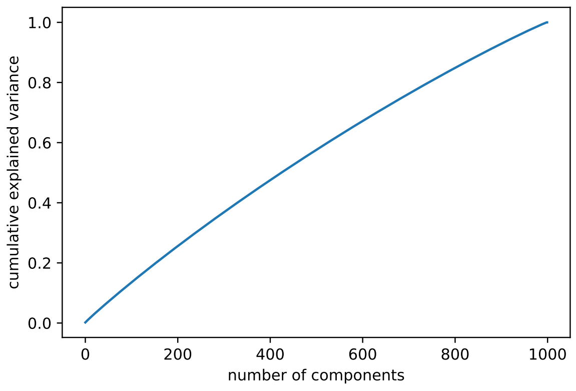
In this section we applied PCA to the output of the weighted cosine similarities. To get an understanding for how the accumulated explained variance correlates to the number of principal components used, this was plotted for three different chromosomes, as can be seen in figures 4a-4c. Here we can see that the number of principal components needed to get over 80% were many since each component contained little variance. Hence each component also contains little information gain. Ideally, we would want a sharp increase in the cummulative explained variance for the first few principal components, since this would mean these components explained much about the original data. To see the numerical values of the number of principal components needed to get over 80% cummulated explained variance, see table 5 in Appendix A.



(a) Chromosome 3



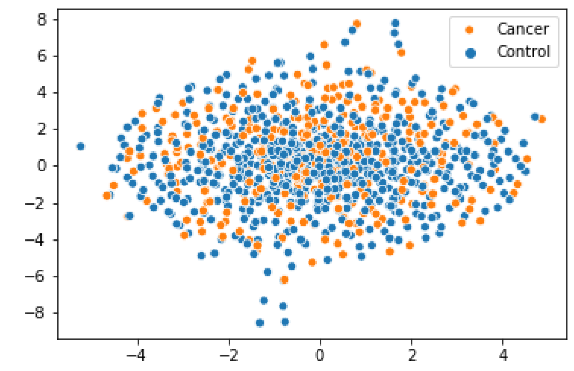
(b) Chromosome 5



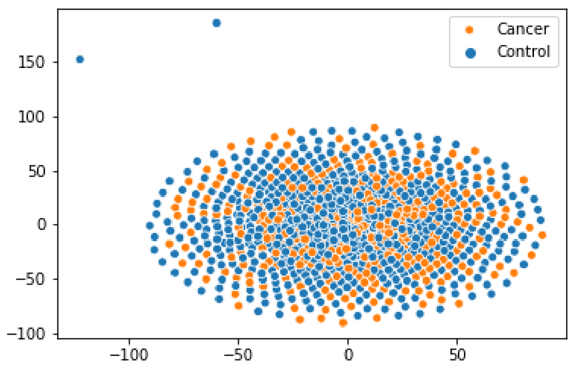
(c) Chromosome 15

Figure 4: Cumulative explained variance versus the number of PCA components used

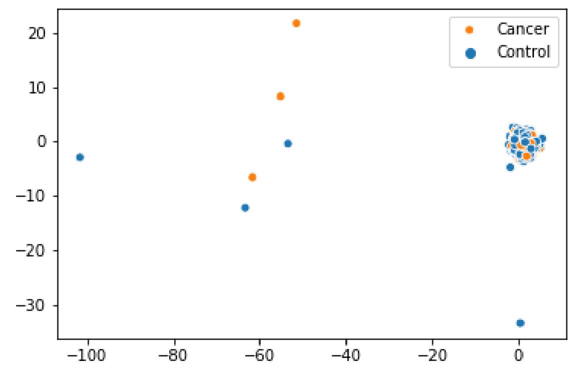
The principal components for each file were then used as input to the t-SNE algorithm. Example of results after the algorithm can be seen in figure 5a-5c, where also the phenotype of each separate data point were color coded. Here we can see that the two different phenotypes show no separation at all. This means that there is no difference between the datapoints corresponding to the people with cancer and the healthy people. Hence, no prediction algorithms will be able to make accurate predictions.



(a) Chromosome 6, perplexity 1



(b) Chromosome 10, perplexity 10

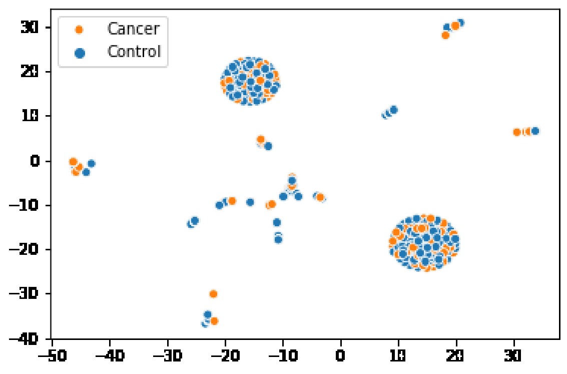


(c) Chromosome 12, perplexity 100

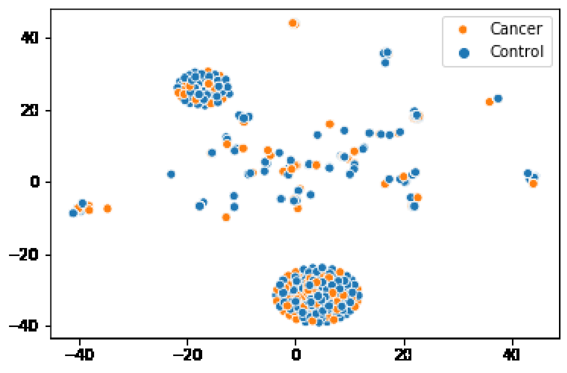
Figure 5: Data points after cosine similarity reduction, PCA and t-SNE transformation.

3.3 Weighted cosine similarities + Undercomplete deep autoencoder + t-SNE + k-means

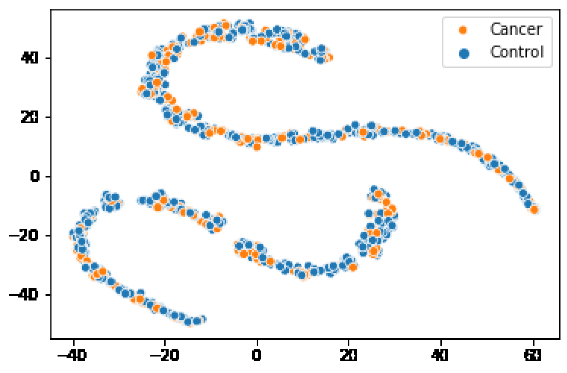
Using the weighted cosine similarity files as input to the autoencoder, I received compressed data with 100 features. This data was then used in the t-SNE algorithm to get the 2-dimensional transforms. Example of outputs of this is visualized in figure 7a-7c.



(a) Chromosome 17, perplexity 10



(b) Chromosome 18, perplexity 10

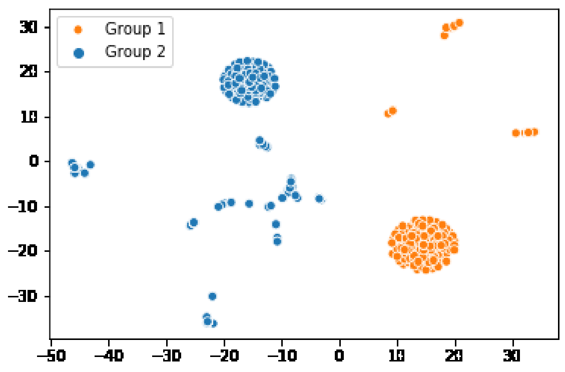


(c) Chromosome 22, perplexity 20

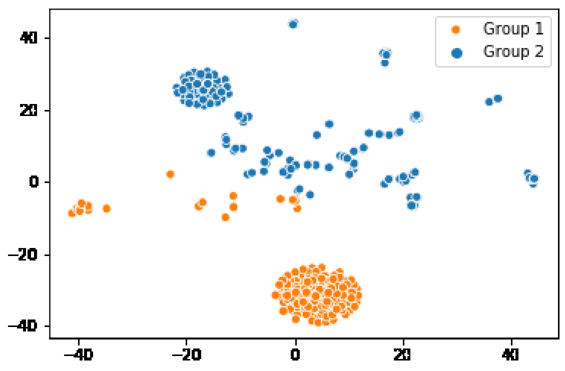
Figure 6: Data points after cosine similarity reduction, undercomplete deep autocoder encoding and t-SNE transformation.

The data in figure 6 was then divided in a training and test set. I used the training set to find the two cluster centres with the k-means algorithm. For the training data this yielded in a division of the plane as can be seen in figure 7. After that I assigned each testing point to the closest cluster centre. Which gave 57.5% accuracy for chromosome 17 with perplexity 10, 57% accuracy for chromosome 18 with perplexity 10 and 58.5% accuracy for chromosome 22 with perplexity 20. The reference value here is 50%, since this algorithm can not learn to always guess that the cells have glioma.

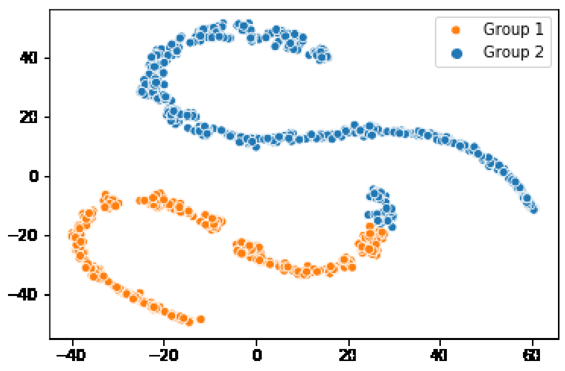
So even if there is a separation after t-SNE, and the accuracies look promising, we also have to take into account that 154 combinations of different files and perplexities were tested. In each combination, if the data had no significance, the obtained accuracy would be roughly the same as flipping a coin for each test point and noting the fraction of heads or tails, whichever is higher. I did a simulation of this for 150 iterations, and it turns out that one can expect a highest value around 0.6. Hence, we cannot say that our obtained accuracies are significant.



(a) Chromosome 17, perplexity 10



(b) Chromosome 18, perplexity 10

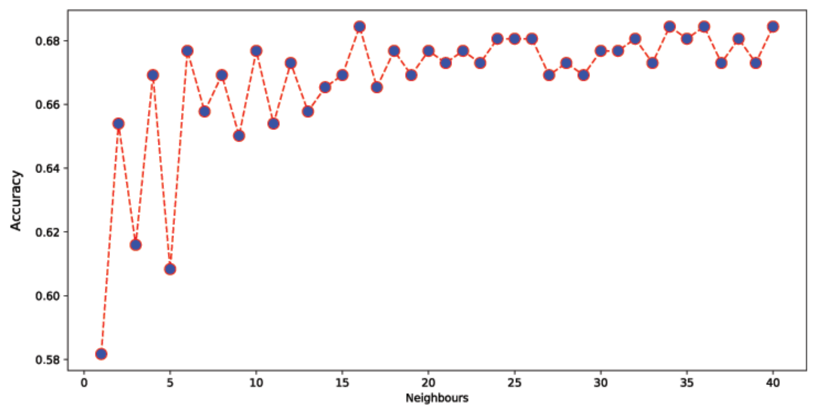


(c) Chromosome 22, perplexity 20

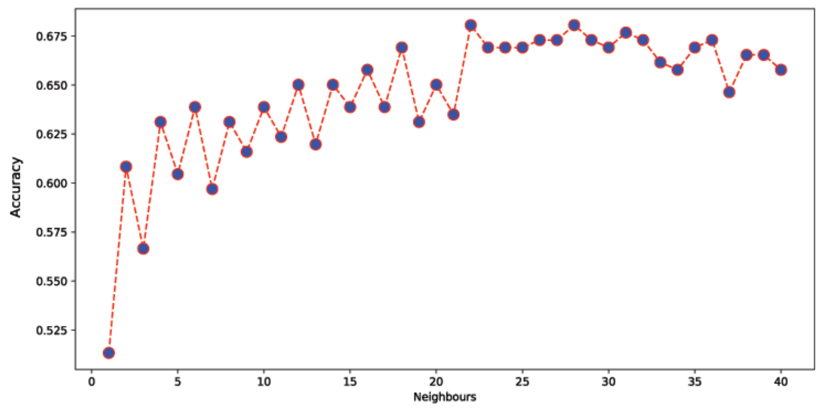
Figure 7: Division of training data after k-Means have been applied.

3.4 Sum pooling + kNN

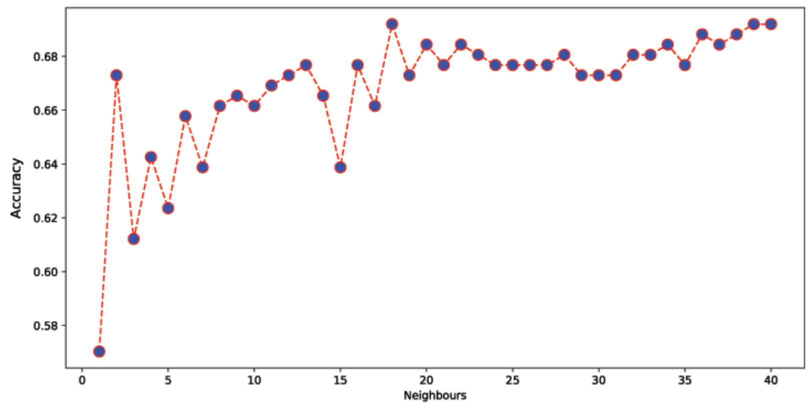
Applying kNN to the sum pooled feature files, we got the accuracies presented in figures 8a to 8c. The numerical result can be seen in table 6 in appendix A. The reference score, which one would obtain if guessing all people were healthy, was 0.688. In the figures we can see that the scores for an increasing number of neighbours seems to be better, but this is probably only due to the fact that the score converges to the reference score for more neighbours. Hence, using kNN on the sum pooled values is not a good prediction method of glioma.



(a) 10 blocks per chromosome



(b) 100 blocks per chromosome



(c) 1000 blocks per chromosome

Figure 8: Accuracy as a function as a function of neighbours using the kNN algorithm for three different block sizes of the sum pooling.

3.5 Sum pooling + NN

Applying NN to the sum pooled feature files, we got the accuracies presented in tables 1 to 3. The reference score, which one would obtain if guessing all people were healthy, is again 0.688. From the tables we see that we get slightly better accuracies with more blocks in the sum pooling. But we never reach an accuracy over the reference value. We can also see from the tables that all networks are overfitting, since the accuracies from the training were significantly higher than the testing accuracies.

	Loss:	**Accuracy:**
Test:	1.4	0.59
Training epoch 1:	0.65	0.658
Training epoch 2:	0.43	0.766
Training epoch 3:	0.028	0.998
Training epoch 4:	3.1e-4	1.00
Training epoch 5:	1.4e-4	1.00
Training epoch 6:	9.7e-5	1.00
Training epoch 7:	7.5e-5	1.00
Training epoch 8:	5.9e-5	1.00
Training epoch 9:	4.8e-5	1.00
Training epoch 10:	4.0e-5	1.00

Table 1: Loss values and accuracies from the neural network, using the sum pooled file with 10 blocks per chromosome.

	Loss:	**Accuracy:**
Test:	2.9e-5	0.627
Training epoch 1:	0.65	0.657
Training epoch 2:	0.19	0.937
Training epoch 3:	4.5e-4	1.00
Training epoch 4:	1.0e-4	1.00
Training epoch 5:	7.3e-5	1.00
Training epoch 6:	5.7e-5	1.00
Training epoch 7:	4.7e-5	1.00
Training epoch 8:	3.9e-5	1.00
Training epoch 9:	3.4e-5	1.00
Training epoch 10:	2.9e-5	1.00

Table 2: Loss values and accuracies from the neural network, using the sum pooled file with 100 blocks per chromosome.

	Loss:	**Accuracy:**
Test:	1.64	0.673
Training epoch 1:	0.65	0.659
Training epoch 2:	0.091	0.968
Training epoch 3:	1.2e-4	1.00
Training epoch 4:	7.9e-6	1.00
Training epoch 5:	5.9e-6	1.00
Training epoch 6:	5.2e-6	1.00
Training epoch 7:	4.6e-6	1.00
Training epoch 8:	4.1e-6	1.00
Training epoch 9:	3.7e-6	1.00
Training epoch 10:	3.4e-6	1.00

Table 3: Loss values and accuracies from the neural network, using the sum pooled file with 1000 blocks per chromosome.

4 Conclusion

With the methods presented in this report, we have not found a way to predict patients with high risk of having glioma. Since the main focus of this report have been to overcome the problem of the high dimension, less effort have been put into the prediction. This due to the huge amount of features, and also the small amount of patients

Looking at the different methods we first compare PCA with the autoencoders. Even though both methods yielded bad results, the autoencoder gave much more separation in the data compared to the PCA. This makes the autoencoder an better option when trying to make predictions. A hint for PCA:s bad performance was given by looking at the explained variance graph since it took very many components to get up to 80% of the original explained variance. But this could also suggest that the weighted cosine similarity method had taken away some information, since this method had been applied before PCA. Or maybe it is just the linearity of PCA that prohibit it from working well.

Going into the sum pooling, we could also see that no predictions could be made via the neural network or by the kNN algorithm. This could be because the assumption, that glioma occurs depending on how many mutations that are present on a fraction of the chromosome, might be faulty.

A different angle that might be fruitful, could be to use already know SNP:s that give a high glioma risk. For example those SNP:s mentioned in [5]. Then use this information to identify other correlated, but undiscovered, high-risk SNP:s. This set of new and old high-risk SNP:s could then be used as the starting point for the methods used in this report.

Another interesting idea to try would be to use an ensemble technique, such as bagging or boosting. This since the ensemble techniques are a way of combating the large portion of noise in the genetic data.

Exploring the interactions between chromosomes is also something that possible could improve the results of this report. This could be done by concatenate the chromosome data, before the weighted cosine similarities. Another way of looking at interactions could be to use prediction algorithms on the clustering of different chromosomes. This by investigating if the belonging to a specific combination of clusters could imply higher cancer risk. Also, doing the k-means clustering before instead of after the t-SNE algorithm could improve the prediction accuracy.

Finally, I want to stress that much is yet to be explored. The methods in this report is to be seen only as an initial step of exploring the possibilities of predicting glioma. The use of different parameters or other methods might give better results, which only the future can tell.

References

[1] M. L. Goodenberger and R. B. Jenkins, "Genetics of adult glioma", ScienceDirect, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S2210776212002608. [Accessed: 13- Feb- 2020].

[2] P. Kelly, "Gliomas: Survival, origin and early detection", Surgical Neurology International, vol. 1, no. 1, p. 96, 2010. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3019361/pdf/SNI-1-96.pdf. [Accessed: 13- Feb- 2020].

[3] K. Krawiec, Genetic programming. Heidelberg: Springer, 2013, pp. 73-74. Available: https://link.springer.com/chapter/10.1007/978-3-642-37207-0_7. [Accessed: 13- Feb- 2020].

[4] L. van der Maaten and G. Hinton, "Visualizing Data using t-SNE", Journal of Machine Learning Research, vol. 9, p. 2582, 2008. Available: http://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf. [Accessed 13 February 2020].

[5] J. Eckel-Passow et al., "Using germline variants to estimate glioma and subtype risks", Neuro-Oncology, vol. 21, no. 4, pp. 451-461, 2019. Available: https://academic.oup.com/neuro-oncology/article/21/4/451/5280678. [Accessed 13 February 2020].

[6] J. Eckel-Passow et al., "Glioma Groups Based on 1p/19q, IDH, and TERT Promoter Mutations in Tumors", New England Journal of Medicine, vol. 372, no. 26, pp. 2499-2508, 2015. Available: https://www.nejm.org/doi/full/10.1056/nejmoa1407279. [Accessed 13 February 2020].

[7] B. Kim and S. Kim, "Prediction of inherited genomic susceptibility to 20 common cancer types by a supervised machine-learning method", Proceedings of the National Academy of Sciences, vol. 115, no. 6, pp. 1322-1327, 2018. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5819441/. [Accessed 13 February 2020].

A Numerical results

File	**Accuracy**
Reference	0.68
Chromosome 1	0.66
Chromosome 2	0.63
Chromosome 3	0.65
Chromosome 4	0.64
Chromosome 5	0.63
Chromosome 6	0.61
Chromosome 7	0.64
Chromosome 8	0.61
Chromosome 9	0.59
Chromosome 10	0.60
Chromosome 11	0.57
Chromosome 12	0.60
Chromosome 13	0.56
Chromosome 14	0.59
Chromosome 15	0.58
Chromosome 16	0.60
Chromosome 17	0.60
Chromosome 18	0.57
Chromosome 19	0.61
Chromosome 20	0.61
Chromosome 21	0.60
Chromosome 22	0.62

Table 4: Numerical accuracies of Naive Bayes for the different chromosome files, including the reference accuracy if one always would have guessed the patients were healthy.

File	**Number of components**
Chromosome 1	763
Chromosome 2	763
Chromosome 3	761
Chromosome 4	758
Chromosome 5	758
Chromosome 6	756
Chromosome 7	757
Chromosome 8	754
Chromosome 9	753
Chromosome 10	755
Chromosome 11	752
Chromosome 12	754
Chromosome 13	746
Chromosome 14	743
Chromosome 15	744
Chromosome 16	748
Chromosome 17	748
Chromosome 18	745
Chromosome 19	741
Chromosome 20	742
Chromosome 21	726
Chromosome 22	728

Table 5: The number of principal components needed for the cummulated explained variance to exceed 80%.

Neighbours\Blocks per file	**10:**	**100:**	**1000:**
k=1:	0.58	0.57	0.57
k=2:	0.65	0.63	0.67
k=3:	0.62	0.6	0.61
k=4:	0.67	0.64	0.64
k=5:	0.61	0.6	0.62
k=6:	0.68	0.63	0.66
k=7:	0.66	0.62	0.64
k=8:	0.67	0.64	0.66
k=9:	0.65	0.62	0.67
k=10:	0.68	0.65	0.66
k=11:	0.65	0.62	0.67
k=12:	0.67	0.51	0.67
k=13:	0.66	0.61	0.68
k=14:	0.67	0.57	0.67
k=15:	0.67	0.63	0.64
k=16:	0.68	0.60	0.68
k=17:	0.67	0.64	0.66
k=18:	0.68	0.60	0.69
k=19:	0.67	0.63	0.67
k=20:	0.68	0.62	0.68
k=21:	0.67	0.64	0.68
k=22:	0.68	0.62	0.68
k=23:	0.67	0.65	0.68
k=24:	0.68	0.62	0.68
k=25:	0.68	0.65	0.68
k=26:	0.68	0.64	0.68
k=27:	0.67	0.66	0.68
k=28:	0.67	0.64	0.68
k=29:	0.67	0.67	0.67
k=30:	0.68	0.63	0.67
k=31:	0.68	0.65	0.67
k=32:	0.68	0.63	0.68
k=33:	0.67	0.68	0.68
k=34:	0.68	0.67	0.68
k=35:	0.68	0.67	0.68
k=36:	0.68	0.67	0.69
k=37:	0.67	0.67	0.68
k=38:	0.68	0.67	0.69
k=39:	0.67	0.68	0.69
k=40:	0.68	0.67	0.69

Table 6: Numerical accuracies of kNN depending on number of blocks used per sum pooled chromosome and the number of neighbours used.