University of Novi Sad

Faculty of Science

Department of mathematics and informatics

Mirjana Samardžić

Dimensionality reduction of the Human Microbiome data

Master thesis

Supervisor: dr Sanja Brdar

2019, Novi Sad

Table of Contents

ABSTRACT

Human microbiome studies generate huge amount of data that can bring important findings related to health care and disease diagnosis. The main characteristics of such data are larger number of features than samples and high sparsity that impose challenges to the analytics steps. This thesis aims to explore dimensionality reduction on the human microbiome data and to evaluate classification of samples in reduced, latent space, compare to original features. Before analysis raw DNA sequence data are pre-processed using QIIME2 microbiome analysis package and grouped into Operational Taxonomic Units (OTUs). After clustering sequences into OTUs, obtained data set contains >30000 features. To uncover structure in high dimensional data of OTU features we project it down to a subspace by using well known Singular Value Decomposition (SVD)-matrix decomposition and a novel approach based on Autoencoder Neural Network, performing in this way dimensionality reduction of original data. After dimensionality reduction we apply Random Forest Classifier on both - original data and data with reduced set of features. Different measures were performed for evaluation of the algorithms such as: 1) accuracy classification score, 2) confusion matrix which provide detailed classification across all classes, and 3) classification report derived from confusion matrix showing the other important classification metrics. We provide results in reduced space with respect to different number of dimensions and evaluate how this parameter impacts the classification task.

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS LIST

DNA – Deoxyribonucleic Acid

QIIME2- Quantitative Insights Into Microbial Ecology

SVD – Singular Value Decomposition

HMP - Human Microbiome project

EMP - Earth Microbiome Project

RNA - Ribonucleic acid

MG-RAST - Metagenomics Service for Analysis of Microbial Community Structure and Function

OTU - Operational Taxonomic Units

ReLU - Rectified Linear Unit

1. INTRODUCTION

Human microbiome studies generate huge amount of data that can bring important findings related to the health care and disease diagnosis. The importance of this topic is highly recognized by scientific community, proven by numerous projects dealing with identifying and characterizing of microbiome data such as: Human Microbiome project (HMP) [1], Earth Microbiome Project (EMP) [2], American Gut Project [3] and MetaSUB: Metagenomics & Metadesign of Subways & Urban Biomes [4]. Research studies can be focused on the contribution of microbial components and genes on normal human physiology and disease predisposition [1]. Additional initiatives are directed towards construction of a global catalogue of the microbial diversity, which is a topic of EMP project [2]. Standardization of protocols and sample contribution are included as part of the research in the American Gut Project, [3] in order to compare human microbiome specimen from different countries. Development and testing metagenomic methods and standards, DNA isolation etc. can be used to improve design of cities considering public health as part of multidisciplinary engineering action [4].

The Human microbiome is collection of microorganisms that lives on the surface and inside the human host such as skin, gut, surface of the mouth, tooth surface, respiratory tract, and a many other sites [5]. Analysis of the human microbiome data is very important issue in human health maintenance and diagnosis. Understanding the normal temporal variation in the human microbiome is critical to developing treatments for putative microbiome-related afflictions such as obesity, Cohn’s disease, inflammatory bowel disease, and malnutrition [6].

Larger number of features than samples is a characteristic of the microbiome data which makes learning of classifiers on such data a slow and difficult problem. Therefore, it is common to apply algorithms for dimensionality reduction on data to be able to perform efficient classification.

Prior to the data analysis, the raw sequence data had to be pre-processed. We used QIIME2 (Quantitative Insights Into Microbial Ecology [7]) microbiome analysis package which helped us to analyze DNA sequence. QIIME2 requires Linux environment, therefore all the commands are

performed via Docker [8] and can be found in Appendix. Final result of preprocessing is Feature Table, which represents main starting point for further data analysis.

The machine learning algorithms were applied after preprocessing step transformed sequence data into Feature Table. We used following algorithms Random Forest for classification, SVD (Singular Value Decomposition) and Autoencoder Neural Network for dimensionality reduction. We applied Random Forest Classifier on the original data, as well as on the reduced data set in order to compare obtained results.

In the following section we will give detail explanations of data preprocessing in QIIME2, theoretical background of algorithms, algorithms implementation and discussion of obtained results.

2. MICROBIOME DATA

The data set used in these theses was taken from the study "Moving pictures of the human microbiome" [6] that is a high-throughput study of thousands of samples. Data contain two groups of examinee male and female and four body sites: the tongue, the left palm of hand, the right palm of hand, and the gut, sampled over 396 timepoints.

16S ribosomal RNA (16S rRNA) marker gene gives possibility to improve technology of gene sequencing, decreasing the cost of the sequence process [9]. 16S allows sequence conservation and domain structure with variable evolutionary rates. Profiling by 16S is performed prior to metagenomic analysis which directs the selection of sequence technology and amount of sequencing. Example of small part of data set used in this thesis is shown in Figure 2.1.

```
>L1S1.275106_346 HWI-EAS440_0386:1:25:2733:1495#0
TACGTATGGAGCGAGCGTTGTCCGGAATTATTGGGCGTAAAGGGTACGCAGGCGGTTTAATAAGTCGAATGTTAAAGATCGGGGCTCAACCCCGTAAAGCATTGGAAACTG
>L1S1.275106_1137 HWI-EAS440_0386:1:25:6202:1750#0
TACGGAAGGTTCGGGCGTTATCCGGATTTATTGGGTTTAAAGGGAGCGTAGGCCGTTTGGTAAGCGTGTTGTGAAATGTAGTAGCTCAACTTCTAGATTGCAGCGCGAACT
>L1S1.275106_1968 HWI-EAS440_0386:1:25:2889:1932#0
TACGGAAGGTTCGGGCGTTATCCGGATTTATTGGGTTTAAAGGGAGTGTAGGCGGCCTGTTAAGCGTGTTGTGAAATGTAGATGCTCAACATCTGACTTGCAGCGCGAACT
>L1S1.275106_2452 HWI-EAS440_0386:1:25:5707:2019#0
TACGTAGGGTGCGAGCGTTGTCCGGAATTACTGGGCGTAAAGAGCTCGTAGGTGGTTTGTCGCGTCGTCTGTGAAATTCCGGGGCTTAACTCCGGGCGTGCAGGCGATAGG
>L1S1.275106_2551 HWI-EAS440_0386:1:25:8068:2023#0
TACGGAGGATCCGAGCGTTATCCGGATTTATTGGGTTTAAAGGGAGCGTAGGTGGATTGTTAAGTCAGTTGTGAAAGTTTGCGGCTCAACCGTAAAATTGCAGTTGAAACT
>L1S1.275106_3232 HWI-EAS440_0386:1:25:9679:2132#0
TACGTAGGGTGCGAGCGTTGTCCGGAATTACTGGGCGTAAAGGGCTCGTAGGTGGTTTGTCGCGTCGTCTGTGAAATTCCGGGGCTTAACTCCGGGCGTGCAGGCGATACG
>L1S1.275106_3904 HWI-EAS440_0386:1:25:6027:2225#0
TACGTAGGTGGCAAGCGTTGTCCGGATTTATTGGGCGTAAAGCGAGCGCAGGCGGAAGAATAAGTCTGATGTGAAAGCCCTCGGCTCAACCGAGGAACTGCATCGGAAACT
>L1S1.275106_4091 HWI-EAS440_0386:1:25:9581:2251#0
TACGGAGGATCCGAGCGTTATCCGGATTTATTGGGTTTAAAGGGAGCGTAGATGGATGTTTAAGTCAGTTGTGAAAGTTTGCGGCTCAACCGTAAAATTGCAGTTGATACT
>L1S1.275106_4353 HWI-EAS440_0386:1:25:9037:2284#0
TACGTAGGTGGCAAGCGTTATCCGGAATTATTGGGCGTAAAGCGCGCGTAGGCGGTTTTTTAAGTCTGATGTGAAAGCCCACGGCTCAACCGTGGAGGGTCATTGGAAACT
>L1S1.275106_5521 HWI-EAS440_0386:1:25:6825:2430#0
TGCGGAAGATCCGAGCGTTATCCGGATTTATTGGGTTTAAAGGGAGCGTAGATGGATGTTTAAGTCAGTTGTGAAAGTTTGCGGCTCAACCGTAAAATTGCAGTTGATACT
```

Figure 2.1. Small part of the 16S rRNA microbiome data set.

The data and metadata were taken from MG-RAST which is a Metagenomics Service for Analysis of Microbial Community Structure and Function [10]. Data go through MG-RAST provides modular analysis pipeline, enabling the user to retrieve the data after/before different processing stages. In this work, the data set was retrieved after the quality filtering step [11].

Metadata are very important for data analysis as they provide additional details about the data. Specifically, for the used microbiome data set, metadata contain descriptions of the samples, such as subject identifier, gender, sampling time, and body site i.e. the location of sampling [12].

In the Figure 2.2 simple analysis on the used data was shown to illustrate the counts of observations across different body sites and genders using bars.

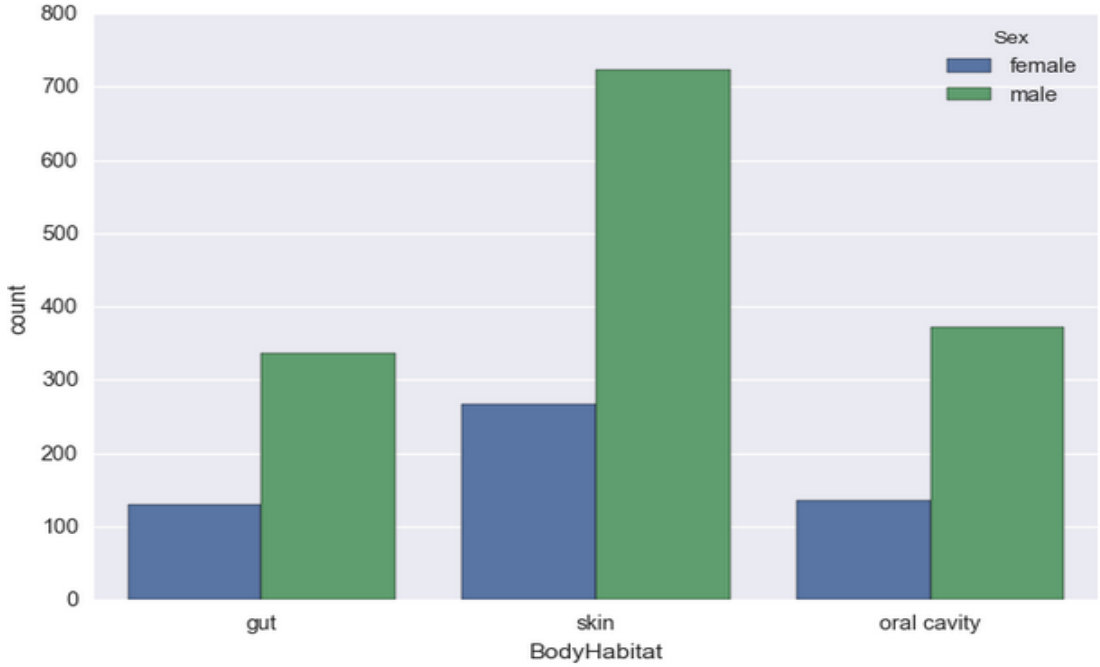

Figure 2.2. The countplot of body habitat of both individals female and male.

Data set contains 1967 files each representing one sample. Files were merged to single file with the whole size of 11.3 GB.

2.1. Preprocessing of Data

For data preprocessing we used QIIME2. It is an open source microbiome analysis package for bioinformatics scientists, which enable realization of the significant microbiome research projects. QIIME 2 pipeline is modular, enabling processing from different starting points, depending on available data format and type of analysis needed [7]. In the Figure 2.3, workflow of QIIME2 for examination of sequence data is represented.

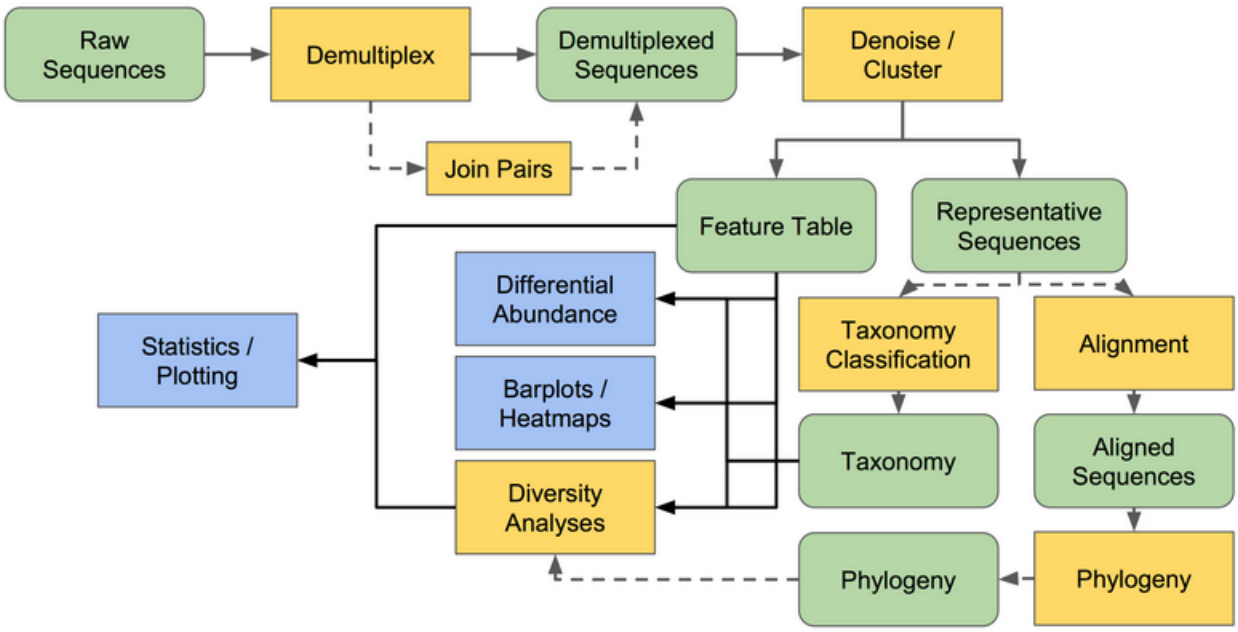

Figure 2.3. Workflow of QIIME 2 for examination of sequence data [7].

In this work we started with demultiplexed sequence data implying that barcoded reads were assigned to the samples they are derived from.

Our task was to perform clustering of sequences into OTUs (Operational Taxonomic Units) in order to obtain Feature Table which is our main result of preprocessing step. Operational taxonomic units are groups or clusters of closely related sequences, representing taxonomic unit [13].

The first step of preprocessing with QIIME2 is to import data in the special format named artifact file with extension **.qza* [7]. Set of commands for importing data in artifact trough QIIME2 via Docker is given in Appendix 1.

The clustering analysis steps are introduced in Figure 2.4. Two steps were performed in our experimental analysis: dereplication and OTU clustering. We provide here more details on these steps.

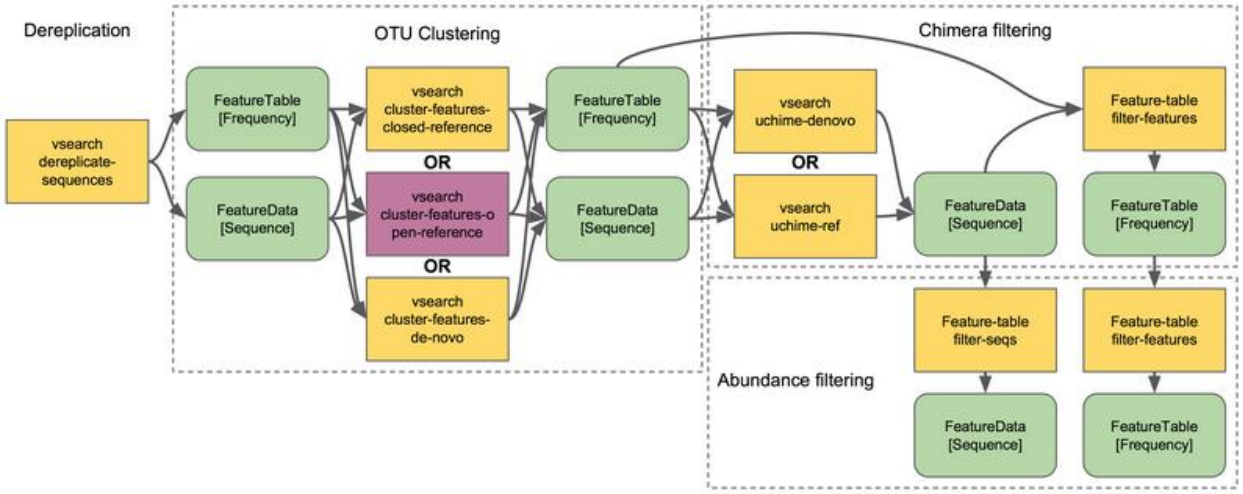

Figure 2.4. Workflow of clustering process [7].

2.1.1. Dereplication

Dereplication of sequences is a process of reducing repetition while we keep count of each replicate. It is the necessary starting point to all clustering methods in QIIME2 [7]. Set of commands for dereplication trough QIIME2 via Docker is given in Appendix 2. The obtained results from dereplicated sequences is the Feature Table [Frequency] which indicates the number of times each Operational Taxonomic Unit (OTU) is observed in each of our samples and the Feature Data [Sequence] which contain the mapping of each feature identifier to the sequence variant that defines that feature.

2.1.2. Clustering sequences into OTUs

In our work we applied closed-reference clustering. Closed-reference clustering of a Feature Table is performed at 97% identity against the Greengenes [12] OTUs reference database. Set of commands for closed-reference clustering trough QIIME2 via Docker is given in Appendix 3 [7]. The reference database is provided as a Feature Data [Sequence] artifact. The sequences in the Feature Data [Sequence] artifact are clustered against a reference database. The features in the Feature Table obtained in the previous step are changed with new features that are clusters of the input features. The obtained results from closed-reference clustering are:

1. **The Feature Table [Frequency] artifact** which indicates the number of times each OTU is observed in each of our samples.

2. The Feature Data [Sequence] artifact which in this case is not the sequences defining the features in the Feature Table, but rather the collection of feature ids and their sequences that didn't match the reference database at 97% identity.

After closed-reference clustering resulting in Feature Table [Frequency] we were able to visually summarize the data. Set of commands for visual summaries of the data through QIIME2 via Docker [7] is given in Appendix 4.

Example of small part of Feature Table obtained in this thesis is shown in Figure 2.5, where rows correspond to samples and columns to OTUs.

#OTU ID	52053.0	1620998.0	4412485.0	4396490.0	4323607.0	192282.0	338980.0
L1S208	0.0	0.0	0.0	0.0	0.0	1.0	0.0
L1S218	0.0	0.0	0.0	0.0	0.0	1.0	0.0
L1S334	0.0	0.0	0.0	0.0	0.0	0.0	0.0
L2S112	0.0	0.0	0.0	0.0	4.0	0.0	1.0
L2S114	0.0	0.0	0.0	0.0	0.0	0.0	0.0
L2S116	0.0	0.0	0.0	0.0	1.0	0.0	6.0
L2S117	0.0	0.0	0.0	0.0	0.0	0.0	0.0
L2S118	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figure 2.5. Small part of Feature Table.

The visual summaries of data are presented in Table 2.1, Table 2.2 and Table 2.3. Table 2.1 presents how many samples, features and frequencies contains our dataset. Table 2.2 and Table 2.3 present frequencies per sample and frequency per feature, respectively. Those are the statistics which QIIME 2 provides.

Table 2.1. The total number of samples, total number of features and total frequency in our Feature table.

TABLE SUMMARY	
Metrics	**Sample**
Number of samples	1 967
Number of features	37 315
Total frequency	63 563 467

Table 2.2. Related statistics summary of frequency per sample.

FREQUENCY PER SAMPLE	
	Frequency
Minimum frequency	112,0
First quartile	10 205,0
Median frequency	35 508,0
Third quartile	44 204,5
Maximum frequency	114 917,0
Mean frequency	32 314,93

Table 2.3. Related statistics summary of frequency per feature.

FREQUENCY PER FEATURE	
	Frequency
Minimum frequency	1,0
First quartile	3,0
Median frequency	12,0
Third quartile	81,0
Maximum frequency	2 699 423,0
Mean frequency	1 703,43

The Feature Table was our main result in preprocessing step. The data set contains 37315 features and 1967 samples. The number of features is much larger than the number of samples. The next goal of this thesis is to reduce the dimensionality of data to perform efficient classification on original data and on the reduced data as well.

In the following section we will introduce algorithms for dimensionality reduction: one well known based on SVD (Singular Value Decomposition) matrix decomposition and another novel that utilizes Autoencoder Neural Network for dimensionality reduction.

3. DIMENSIONALITY REDUCTION

Dimensionality reduction deals with the issue of large number of features compared to the number of samples, the characteristics of data sets, which may lead to difficulties in training the effective model [14]. When the number of features is very large compared to the number of samples in dataset, algorithms may have problem in learning models or recognizing patterns in data. Such problem is known as the curse of dimensionality.

Application of learning classifiers on high dimensional data is a slow and difficult task and may result in overfitting. In that case model is too close to particular data set and it can not be applied to the future data, as it does not generalize well and the results are unreliable [15].

In order to improve accuracy and accelerate the learning phase in classification, it is common to apply algorithms for dimensionality reduction on data to facilitate efficient classification. In this thesis we performed two dimensionality reduction techniques: Singular Value Decomposition (SVD) matrix factorization and Autoencoder Neural Network techniques.

3.1. Singular-Value-Decomposition (SVD)

The Singular value decomposition (SVD) is one of the most commonly used dimensionality reduction techniques, which is a type of matrix factorization whose computation is a step in many algorithms [16]. SVD makes it easy to eliminate the less important parts of that representation to produce an approximate representation with any desired number of dimensions [17].

Using SVD algorithm, a matrix (A) can be represent as a product of three matrices (U, Σ and V).

$$A = U\Sigma V^T$$

Σ is an $n\,x\,n$ diagonal matrix with non-negative real entries, the diagonal entries of Σ are the singular values of A.

U is $m\,x\,n$ matrix with orthonormal columns which means that each of its columns is a unit vector and the dot product of any two columns is zero.

V is an $n\,x\,n$ matrix with orthonormal columns what means that each of its columns is a unit vector and the dot product of any two columns is zero.

In the following Figure 3.1 we can see schematic representation of SVD decomposition.

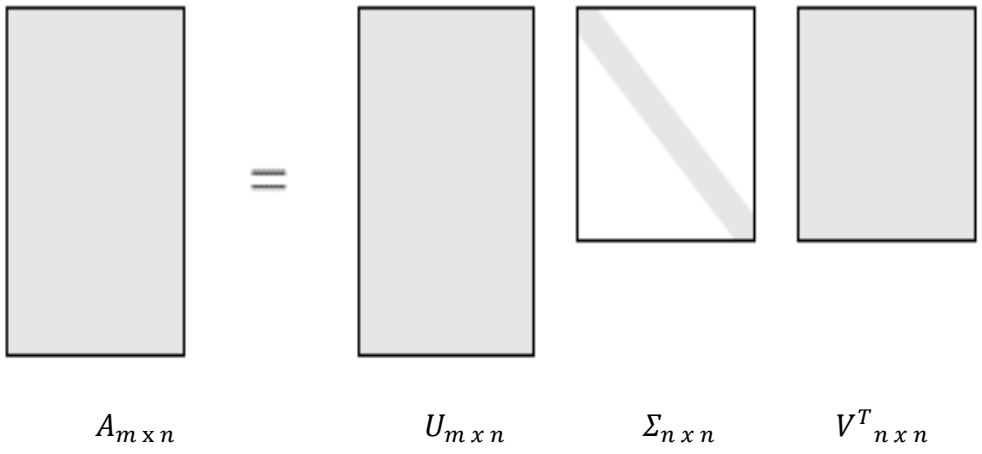

Figure 3.1. Schematic of SVD decomposition [17].

3.1.1. Procedure of a dimensionality reduction through SVD

The SVD represents a very large matrix A by its components U, $\sum$, and V. It starts with setting the smallest of the singular value to zero, which eliminated the corresponding rows of U and V (Figure 3.1) [17].

A useful rule on how to retain enough singular values is to save up 90% of the energy in $\sum$. That is, the sum of the squares of the retained singular values should be at least 90% of the sum of the squares of all the singular values [17].

$$\sum_{i=1}^{k} \sigma_i^2 \geq 90\% \sum_{i=1}^{n} \sigma_i^2,$$

where *n* is the number of all singular values and *k* is the minimum number of retained singular values, which saves the 90% of variance.

3.2. Autoencoder Neural Network

3.2.1. Brief introduction of Neural Networks

Neural networks are represented as a collection of neurons that are connected in a fully connected acyclic graph [18]. They are organized in layers, and layers are further built from neurons, and neurons between adjacent layers are connected. Figure 3.2 shows Neural Network where the output of one layer is input to the next layer.

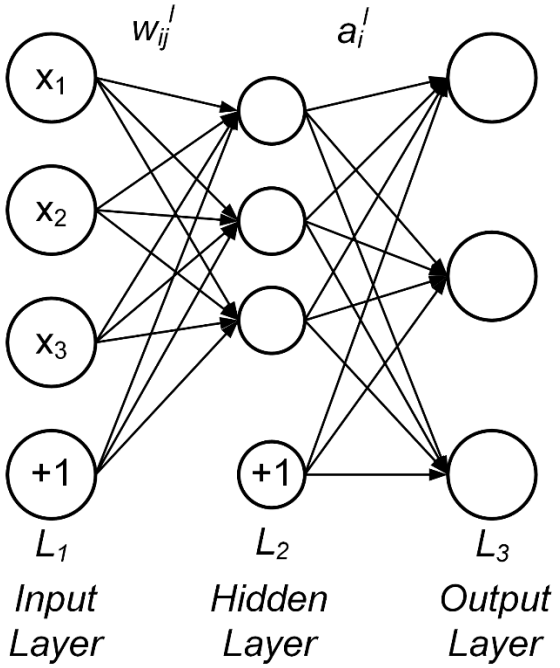

Figure 3.2. Neural Network graphical representation [19].

Input Layer – The input layer sends the information to the hidden layer without computation.

Hidden Layer – The hidden layer is placed between the input layer and output layer; it performs computations and transfers information from the input layer to the output layer.

Output Layer – The output layer is the last computational layer and it sends the information to the outside world [19].

Underlying process can be represented in the following manner:

For the sake of simplicity, we suppose our network looks like in Figure 3.2. It contains three layers L_l, $l = 1,2,3$, where: L_1 denotes input layer, L_2 denotes hidden layer and L_3 denotes output layer. Each layer contains M_l neurons. In our example from Figure 3.2 M_l equals 3 for each l. Neurons are connected between adjacent layers. Those connections between neurons are called weights. Weight connecting $j-th$ neuron at layer l and $i-th$ neuron at layer $l+1$ is denoted as $W_{ij}^{(l)}$.

Let us further define with $b_i^{(l)}$ bias associated with neuron i in layer $l+1$. Bias does not have inputs and they have fixed output of the value +1. We also define an activation function $a_i^{(l)}\left(x_i, W_{ij}^{(l)}, b_i^{(l)}\right)$, $i = 1, \dots, M_l$, $l = 2,3$. Activation function is a scalar product of weights plus bias $a(x; W, b) = f(W^T x + b)$. It gives us output of neuron $i, i = 1, \dots, M_l$ in layer $l = 2,3$. The role of the bias is to let on the activation function to be transferred to the left or right, to better fit the data [20]. It is useful to note that input layer does not have activation function.

For the fixed parameters W, b neural network defines function $h_{W,b}(x)$ that outputs the real number [20]. To train this network we need training set $\{\left(x^{(1)}, y^{(1)}\right), \dots, \left(x^{(m)}, y^{(m)}\right)\}$, where $x^{(k)}$ are inputs and $y^{(k)}$ are labels, $k = 1, \dots, m$. Our goal is to approximate the mapping of inputs $x^{(k)}$ to the outputs $y^{(k)}, k = 1, \dots, m$ using the neural network.

First, we need to define cost function $J(W, b; x, y)$ four our training set (x, y). Our goal is to minimize $J(W, b)$ in terms of W and b. Gradient descent is used for minimization of the cost function. One iteration of gradient descent updates the W, b parameters as follows:

$$W_{ij(new)}^{(l)} = W_{ij(old)}^{(l)} - \alpha \frac{\partial}{\partial W_{ij}^{(l)}} J(W, b)$$

$$b_{i(new)}^{(l)} = b_{i(old)}^{(l)} - \alpha \frac{\partial}{\partial b_i^{(l)}} J(W, b)$$

where α denote a learning rate. The learning rate is a parameter that has a small positive value, usually between 0 and 1. It is used in the training of neural networks with the purpose to control the speed at which the model learns [21].

The main problem in the optimization procedure is the calculation of the partial derivatives for weights and biases in hidden layers. The backpropagation algorithm provides a way to obtain those derivatives, based on the assumption that neurons in hidden layers contribute to the errors at the output layer. The backpropagation algorithm follows the procedure below:

1. Take a training set (x, y).
2. Run a forward propagation to compute all the activations throughout the layers including $h_{W,b}(x)$ output.
3. For each neuron $i, i = 1, \dots M_l$ in each layer l the error $\delta_i^{(l)}$ is computed which measures the contribution of that neuron to the error in output.
    a. For output neuron we can directly measure the difference between the network's activation and the true target value and use it to define $\delta_i^{(L)}$, where L is the numeration of the output layer, in our simple case it is 3.
    b. For hidden neurons we compute $\delta_i^{(l)}$ based on a weighted average of the error terms of the neurons that uses $a_i^{(l)}$ as input.

Now we can introduce the whole pseudo code for gradient descent algorithm:

1. Set $W^{(l)} = 0,\ b^{(l)} = 0, l = 1,2,3$
2. For $i = 1, \dots, m$
    a. Use backpropagation to compute $\nabla_{W^{(l)}} J(W, b; x, y)$ and $\nabla_{b^{(l)}} J(W, b; x, y)$
    b. Set $W^{(l)} = W^{(l)} + \nabla_{W^{(l)}} J(W, b; x, y)$
    c. Set $b^{(l)} = b^{(l)} + \nabla_{b^{(l)}} J(W, b; x, y)$
3. Update the parameters:

$$W^{(l)} = W^{(l)} - \alpha(\frac{1}{m} W^{(l)} + \lambda W^{(l)})$$

$$b^{(l)} = b^{(l)} - \alpha(\frac{1}{m} b^{(l)})$$

To train our neural network, we can now repeatedly take steps of gradient descent to reduce our cost function $J(W, b)$ [20].

There are many types and structures of neural networks with different architecture. In this Chapter we just gave the basic introduction and concept to make intuition how neural networks work. In this thesis we focused on special type of neural network – an autoencoder. In next section we will present in more detail autoencoder neural network.

3.2.2. Autoencoder

Autoencoder is feedforward neural network with a purpose to copy its input to its output, producing a compressed data representation in a hidden layer [22]. The most popular application of autoencoders is to reduce the dimensionality of data into a smaller representation [22]. Autoencoder is built from Encoder Layer, Hidden Layer and Decoder Layer [19]. Figure 3.3 illustrates the autoencoder archicteture.

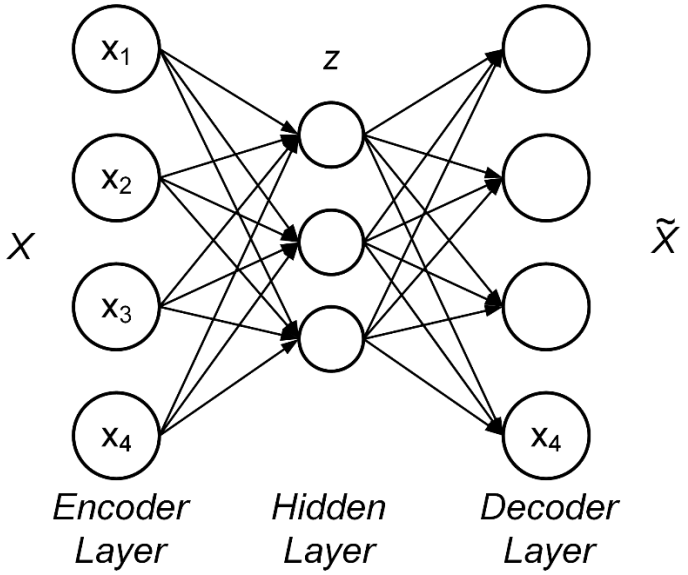

Figure 3.3. Autoencoder architecture [19].

Suppose we have a set of data points$\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$. We want to map them to another set of data points $\{z^{(1)}, z^{(2)}, \ldots, z^{(m)}\}$, where z's data points have a lower dimensionality than x's and z's can faithfully reconstruct x's.

The roles of autoencoder are following:

Encoding: Map data points $x^{(k)}, k = 1, \ldots, m$ to data $z^{(k)}, k = 1, \ldots, m$. This step reduces the input data by mapping an input into hidden layer.

Hidden Layer (Code): Hidden layer describes a code used to represent the input.

Decoding: Map from reduced data $z^{(k)}, k = 1, \ldots, m$ back to $\widetilde{x^{(k)}}, k = 1, \ldots, m$ which approximates the original data. In other words, decoder layer returns approximation of original data using hidden layer.

Let z and $\tilde{x}$ be functions of their inputs (activation function). For autoencoder that is a three layers net, i.e. a neural net with one hidden layer (as in Figure 3.3) mapping is organized in following manner:

$$z^{(k)} = W_1 x^{(k)} + b_1$$

$$\widetilde{x^{(k)}} = W_2 z^{(k)} + b_2$$

Our goal is to approximate $x^{(k)}$ to obtain $\widetilde{x^{(k)}}$. Formally, we want to minimize objective function:

$$J = (W_1, b_1, W_2, b_2),$$

by applying gradient descent [19], [20].

In summary autoencoder neural networks consists of an encoding function, a decoding function, and a distance function between the amount of information loss between the reduced

representation of data as well as the approximation representation (i.e. a loss function). The encoder and decoder are differentiable with respect to the distance function, so the parameters of the encoding and decoding functions can be optimized to minimize the reconstruction loss, using Stochastic Gradient Descent [19].

This chapter introduced theoretical background of autoencoders. More details about how we implemented autoencoder are given in Chapter 5.

4. CLASSIFICATION ALGORITHM AND EVALUATION

Machine learning was used to train classifier on original data, as well as on data with reduced dimensionality in order to evaluate how well algorithms for dimensionality reduction keep enough information for classification task. This is particularly interesting to evaluate on our dataset that is highly dimensional and sparse. From broad range of machine learning algorithms, we selected Random Forest algorithm [23], that is top performing algorithm on data structured as samples × features table. Another advantage of a Random Forest algorithm is robustness with respect to overfitting.

4.1. Random Forest Classifier

The Random Forest Classifier is well known machine learning technique. It is a connected collection of multiple decision trees which are merged together in order to jointly make prediction and thus get more accurate results. The procedure of Random Forest algorithm is given below:

For a given number of training samples N and variables M, the procedure for construction of B decision trees in the forest is following [24]:

1. Choose a training set for each tree by choosing N_s samples with replacement from N available training cases.
2. For each node of the tree, randomly choose m variables (where m is much less then M) on which to base the decision at that node. Calculate the best split based on these m variables in training set,
3. Each tree if fully grown and not pruned (as may be done in constructing a normal decision tree classifier).

Instead of using all M variables in a decision node, Random Forest algorithm uses only a small subset of them to calculate the best split of the data. When the new instance enters the model, it passes through all trees, and the result is obtained by means of majority voting. In this way it considers the output of all weak learners. The final result of the model is a set of decision trees.

In our work the main parameter to be selected was the number of trees in order to obtain the most accurate result and to prevent the overfitting. This parameter was selected in cross-validation procedure.

4.2. Classification Evaluation

For the evaluation of used algorithms, we performed different measures such as:

1. **Accuracy Classification Score:**

Accuracy measures how often the classifier makes the correct prediction. It is obtain using following formula [25], [26]:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$$

2. **Confusion Matrix**

Confusion Matrix provides detailed classification across all classes. The most important indicators are elements on the diagonal. They represent the number of samples for which the prediction label is equal to true label. Higher values demonstrate that prediction was appropriate. The elements which are not on diagonal are samples mislabeled by the classifier [27].

3. **Classification Report**

Classification Report is composed of classification metrics such as: Precision, Recall and F1-score.

For explanation of the classification metrics we need to introduce four parameters [26]:

True positives (TP): The classifier predicts positive for positive case;

True negatives (TN): The classifier predicts negative for negative case;

False positives (FP): The classifier predicts positive for negative case;

False negatives (FN): The classifier predicts negative for positive case.

In above defined parameters *True* stands for correct prediction of classifier, *False* stands for incorrect prediction of classifier. The samples are considered to be either positive or negative with respect to some condition. [25], [26].

Precision denotes what percent of our positive predictions were correct. It is the ability of a classifier not to label an instance positive that is actually a negative. Precision measures what percent was correct for all instances classified positive [26]. For each class precision is obtained using following equation:

$$Precision = \frac{True\ positives}{True\ positives + \ False\ positives}$$

Recall denotes what percent of the positive cases we were able to detect. It is the ability of a classifier to find all positive instances. Recall measures what percent was classified correctly for all instances that were actually positive [25]. For each class recall is obtained using following equation:

$$Recall = \frac{True\ positives}{True\ positives + False\ negatives}$$

F1-score is a weighted harmonic mean of precision and recall such that the best score is one and the worst is zero [25]. F1 scores are lower than accuracy measures as they embed both, precision and recall, into their computation. The weighted average of F1 should be used to compare classifier models, not global accuracy. F1 score is obtained using following equation:

$$F1\ score = 2 * \frac{Recall * Precision}{Recall + Precision}$$

5. IMPLEMENTATION OF ALGORITHMS

In this chapter we will introduce the process of implementation of algorithms. Also, we will explain several parameters which were the crucial for implementation. All algorithms were performed in Python programing language. The main used libraries are: *Scikit Learn package* for machine learning [28] and *Keras* which is the Python Deep Learning library [29].

5.1. Implementation of Singular-Value-Decomposition (SVD)

For SVD implementation we used the built-in function from *Scikit Learn package* in Python. Several parameters can be adjusted in accordance with SVD algorithm. Crucial parameter is of course the matrix which we want to decompose. In our case Feature Table obtained in preprocessing step is represented as a matrix.

The second main parameter which we want to set is the number of singular values in order to reduce the dimensionality. Desired number of components is chosen according to the rule from Chapter 3: The sum of the squares of the retained singular values should be at least 90% of the sum of the squares of all the singular values [17]. From that calculation we obtain that the smallest number is 24 retained singular values.

Furthermore, we tried different numbers of singular values to retain more than 90% variance. The evaluation how this parameter impacts classification task will be presented in next section.

5.2. Implementing of Autoencoder

For Autoencoder implementation we used *Keras* library, which is the Python Deep Learning library [29]. Several parameters can be adjusted in accordance with building Autoencoder

algorithm with *Keras*. For building the encoder layer the most important parameter to be chosen is the activation function which needs to be applied on the output [30]. The chosen activation function was Rectified Linear Unit (ReLU) function $f(x) = \max(0, x)$, Figure 5.1.

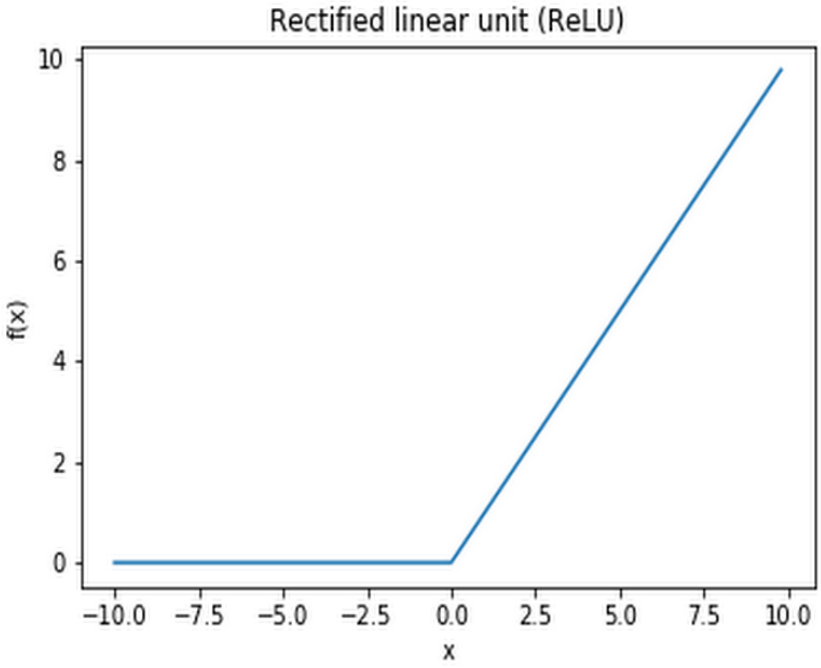

Figure 5.1. Rectified Linear Unit (ReLU) function.

The function returns 0 if it receives any negative input, but for any positive value x it returns that value back [30]. For building the decoder layer chosen activation function was sigmoid function, Figure 5.2:

$$f(x) = \frac{1}{1+e^{-x}},$$

Sigmoid functions have finite limits at negative infinity and positive infinity, most often going either from 0 to 1 or from −1 to 1, depending on convention [31].

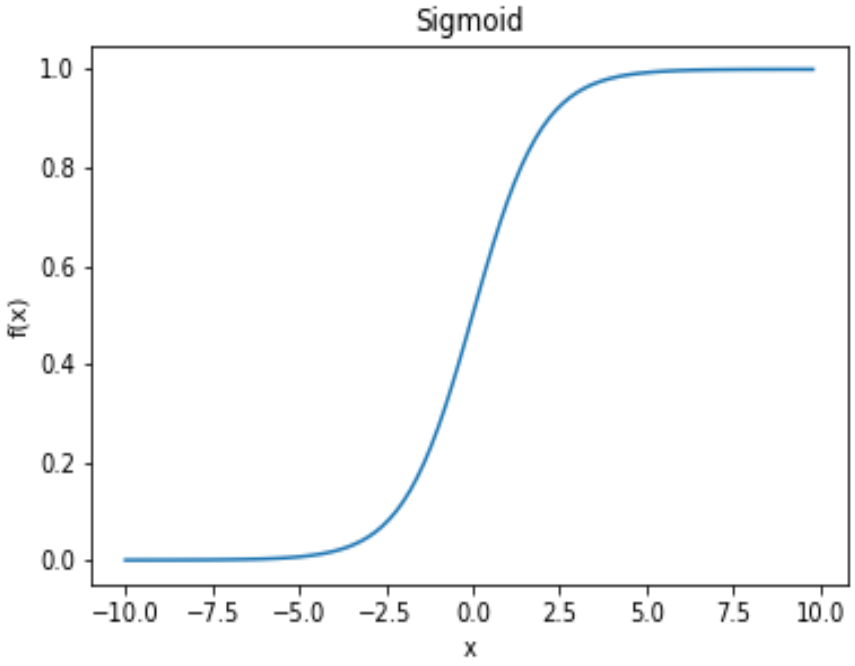

Figure 5.2. Sigmoid function.

Once we define our model, we have to compile it. In this step the cost (loss) function and the suitable optimization algorithm have to be selected.

Adam optimization algorithm [32] was used for binary cross-entropy function [33]. Adam is an optimization algorithm for updating network weights iteratively based on training data [32]. Adam was the best choice for our work because it is very efficient, requires little memory [21] and it fits very well with our type of data. For the training of the Autoencoder another important parameters to be defined are the number of epochs, which is the number of times that the learning algorithm will work through the entire training dataset and the batch size which represent the number of samples processed before the model is updated.

The Autoencoder was trained with different number of epochs. The best results according for measures that were used for the evaluation of algorithms (Accuracy Score, Confusion Matrix and Classification Report) were obtained with 150 epochs, batch size of 256 and for the size of encoded representations equal to 500. The evaluation of the impact of parameters on classification task will be presented in next Chapter.

6. RESULTS

In this Chapter we will introduce obtained results for the data which were reduced with SVD decomposition and Autoencoder Neural Network. After dimensionality reduction we apply Random Forest Classifier on both original data and data with reduced set of features. Different measures were performed for the final evaluation of the algorithms such as:

1) Accuracy Classification Score
2) Confusion Matrix
3) Classification Report

We provide results in reduced space with respect to different number of dimensions and evaluate how this parameter impacts the classification task.

Random Forest Classifier is used on original data and data with reduced dimensionality. The aim of classifier is to predict the true class. The labels are determined by host gender and its body site from which microbiome samples were obtained and are given by:

1) Female - Oral
2) Female- Skin
3) Female- Gut
4) Male- Oral
5) Male- Skin
6) Male- Gut

Algorithm tries to predict from which body site (habitat) sample came from and does it belong to female or a male individual.

6.1. Accuracy Classification Scores

To obtain results we empirically set different parameters for each of our algorithm, and evaluate it influence to our result. For the Autoencoder the crucial parameters were the number of epochs and the batch sizes. Autoencoder network for our dataset was the most accurate with 150 epochs and with batch site of 256. The SVD algorithm has saved the 90% of variance with just 24 features which is a huge reduction from our original space. For the Random Forest classifier, the crucial parameter was the number of trees in the forest, we have experimented with different values and evaluated the performance. In the evaluated parameter range, the optimal results were obtained with 800 trees in forest.

We run classification experiments with data set reduced to 24, 300, 500, 1000 features and also evaluated performance without dimensionality reduction.

In the Table 6.1 we present the best scores obtain after applying Random Forest Classifier for different dimensions after dimensionality reduction with SVD and Autoencoder algorithms. For accuracy measure in the Table 6.1 the best score is one and the worst is zero.

Table 6.1. Accuracy results for different dimensions.

Reduced dimension	Random Forest Classifier on SVD reduction	Random Forest Classifier Autoencoder reduction	Random Forest Classifier on original data
24	0.92	0.88	/
300	0.94	0.94	/
500	0.94	0.95	/
1000	0.93	0.94	/
Original data	/	/	0.96

SVD decomposition provided robust results. Highly reduced set of features (24) already provided enough information for training Random Forest classifier with accuracy larger than 90%. When performing Random Forest classifier after reducing set of features to 500 with autoencoder we obtain the accuracy of 0.95 which is the closest value to the accuracy of 0.96 obtain on original data.

6.2. Normalized Confusion Matrices

To obtain further insights in classification results we provide here results in the form of normalized confusion matrices. Matrices show in detail which classes are harder to detect and where are the largest confusions between the classes. We are interested in elements on diagonal. They represent the number of points for which the prediction label is equal to true label. The higher values on diagonal indicate the better result, i.e. better classification. We are dealing with normalized matrix, so values are between zero and one.

For each experiment 10 times cross validation was used to inspect stability of results. Obtained classification results were highly stable and only slightly changed on the second decimal.

Figure 6.1, Figure 6.2 and Figure 6.3 present the best scores for confusion matrices for experiments on original data, reduced with SVD, reduced with Autoencoder respectively. There are all presented as heatmaps.

Figure 6.1 represents the heatmap of Normalized Confusion Matrix on original data where overall accuracy was 0.95. From the main diagonal we can see that the best classification was achieved for Female-Gut that are perfectly separated from all other samples. Female-Oral, Male-Skin and Male-Gut samples are classified with 0.99 precision, followed by Male-Oral samples that are classified with 0.97 precision.

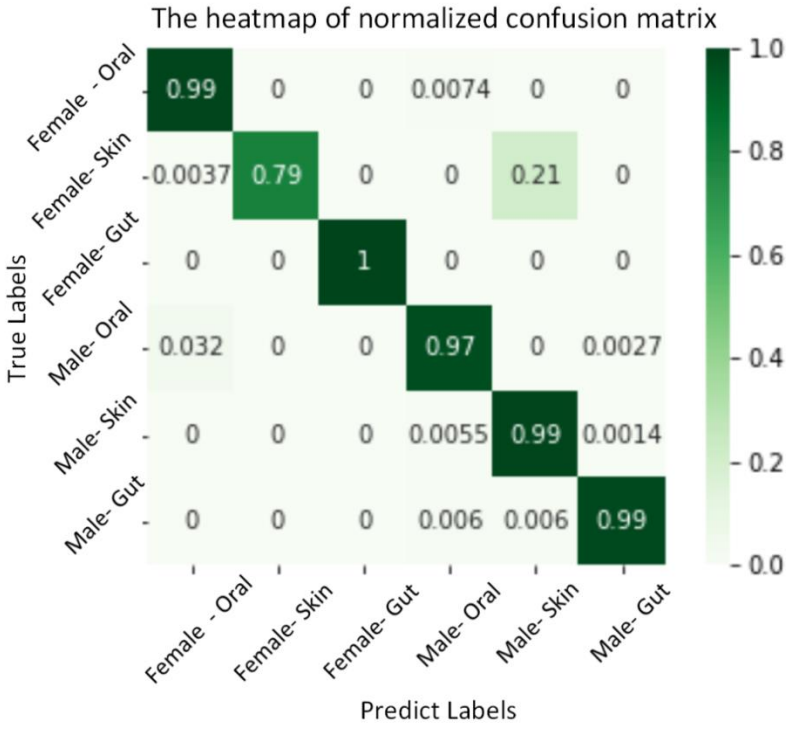

Figure 6.1. Heatmap of confusion matrix of original data.

Figure 6.2 presents the heatmap of normalized confusion matrix after SVD decomposition for the number of features n=500. Similarly, as in Figure 6.1, from the main diagonal of heatmap we can see that Female-Gut, Male-Skin and Male-Gut have the best classified with 0.99 precision, and right after there are Male-Oral and Female-Oral with classification precision of 0.98 and 0.97 respectively. Female-Skin has the 0.75 classification precision.

.

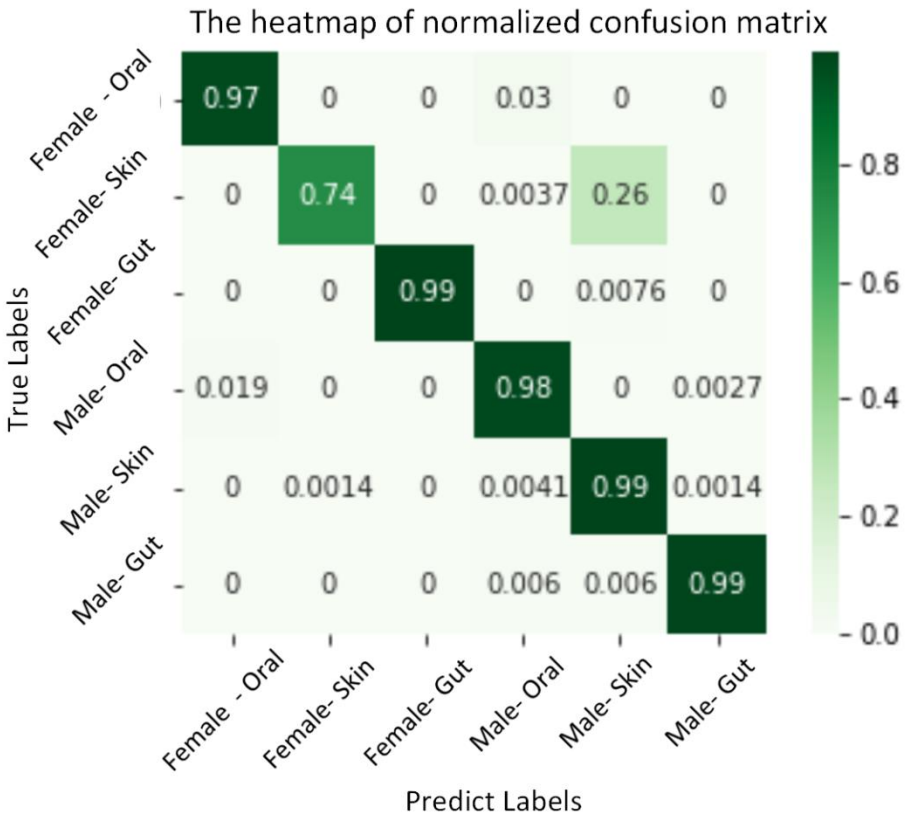

Figure 6.2. Heatmap of Normalized Confusion Matrix after SVD decomposition for n=500.

Figure 6.3 presents the heatmap of normalized confusion matrix after reducing set of features with Autoencoder on n=500 features. Again, by observing the main diagonal from the heatmap we can see that Female-Gut has the best separation from all other samples. Right after there is Male-Gut with classification of 0.99 precision. Male-Skin and Male-Oral are classified with 0.97 precision. While the Female-Oral and Female-Skin is classified with 0.95 and 0.73 precision, respectively.

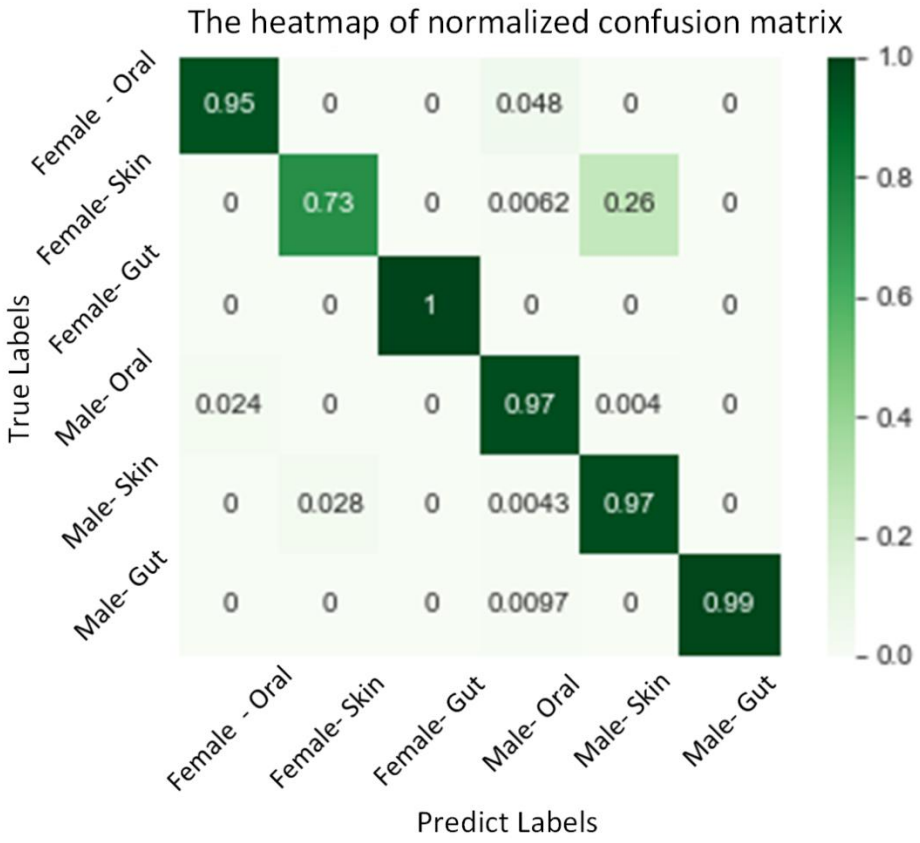

Figure 6.3. Heatmap of normalized confusion matrix after Autoencoder reduced data when n=500.

6.3. Classification report scores

To further summarize results we run generated classification reports that provide insights into precision, recall and f1-score for each class. Figure 6.4, Figure 6.5 and Figure 6.6 present the best scores for Classification Report. All classification report figures are represented in the form of heatmaps.

Figure 6.4 represent the heatmap of classification report on original data. From the figure we can see that the best score for precision was achieved for Male-Oral, Female-Oral, Female-Gut and Female-Skin. Precision for Male-Gut and Male-Skin amount 0.970 and 0.923, respectively. Recall was perfectly achieved for Male-Skin, Female-Gut and Female-Oral, right after there are

Male-Oral and Male-Gut with the recall of 0.973 and 0.970 respectively. The recall for the Female-Skin is the smallest with the value of 0.808. F1-score has the highest value for Female-Gut and Female-Oral. For Male-Gut, Male-Skin and Male-Oral F1-score amount 0.970, 0.960 and 0.986, respectively. Further for Female-Skin F1-score is 0.894.

Figure 6.4. Heatmap of classification report on original data.

Figure 6.5 presents the heatmap of classification report after SVD decomposition when number of features equals to n=500. Observing the heatmap the highest values for precision were obtained for Male-Oral, Female-Gut, Female-Skin and Female-Oral. For Male-Gut precision is 0.970 and for Male-Skin it is 0.889. Recall has the perfect value for Male-Skin, Female-Gut and Female-Oral. The smallest value is for Female-Skin with recall of 0.692. For the Male-Gut, Male-Oral the recall is 0.970 and 0.973 respectively. Finally, the F1-score for Female-Gut and

Female-Oral has the highest value, while for the Male-Gut, Male-Oral and Male-Skin F1-score is respectively 0.970, 0.941 and 0.986. The worst F1-score is for Female-Skin and it is 0.818.

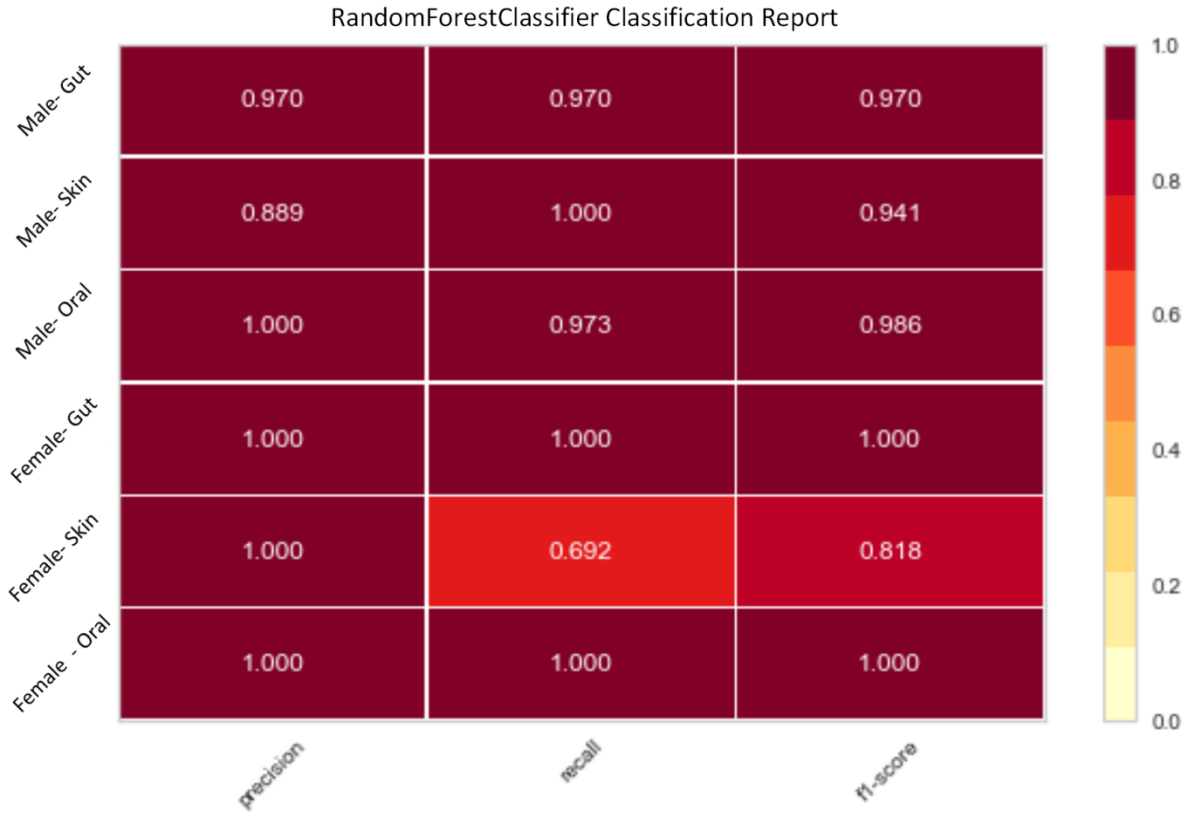

Figure 6.5. Heatmap of classification report after SVD decomposition for n=500.

Figure 6.6 represent the heatmap of Normalized Classification Report after application of autoencoder to reduce data into lower dimensional space of n=500 features. Precision values form heatmap has the highest value for Male-Gut, Male-Oral, Female-Gut, Female-Skin and Female-Oral. For the Male-Skin precision is 0.920. Recall is the worst for the Female-Skin with the value of 0.750, while for all others it has the perfect value of one. F1-score is the best for Male-Gut, Male-Oral, Female-Gut, and Female-Oral. While for Male-Skin and Female-Skin it has amount of 0.958 and 0.857, respectively.

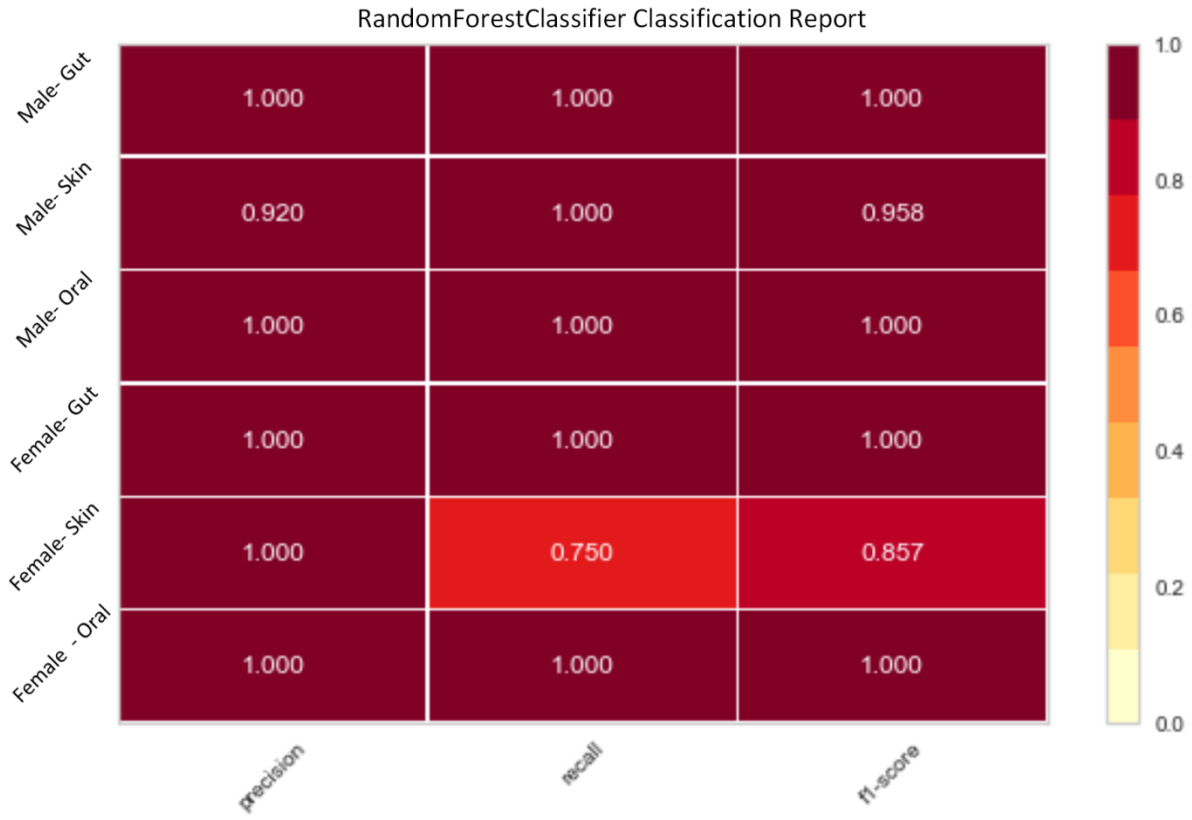

Figure 6.6. Heatmap of Classification Report after Autoencoder on reduced data for n=500 umber of features.

7. CONCLUSION

The modern technology for studying the DNA sequences gives us the new way for improving human health including novel methods for analyzing microbiome, which is a topic of this thesis. Dimensionality reduction is important step in working with high-dimensional data and adaption of such data for further analysis. In this work we started with microbiome dataset, which was preprocessed using QIIME2 package prior to dimensionality reduction. QIIME2 enables us to performed important steps to preprocess data to be ready for machine learning algorithms and deep learning analysis.

Process of dimensionality reduction was performed using both SVD decomposition and Autoencoder Neural Network. Random Forest Classifier algorithm was applied and evaluated on data in original and reduced space of features. Finally, we measured effects of dimensionality reduction and compared two approaches, SVD and Autoencoder. The classifier's performance was estimated using the accuracy score, confusion matrix, and classification report scores. The highest accuracy for this type of data was obtained by using autoencoder neural network for dimensionality reduction and Random Forest classifier, which reached accuracy of 0.95 while the accuracy on original data was 0.96. For the data set of over 37000 features we were able to reduce its dimension to 500 features and obtain results very close to original dataset. Results remained good even with larger reduction to 300 features. These results indicate that the is a very powerful technique for dimensionality reduction. Advantage of SVD over autoencoder approach for dimensionality reduction was observed in the experiment with the smallest number of features (n=24 that keep 90% of the variance). In this setting SVD outperformed the autoencoder (accuracy of 0.92 vs 0.88), but the overall results are below those obtained on original dataset, indicating that number of features should be larger.

Understanding the domain of human microbiome data is a crucial to further analysis that may lead to very important medical discoveries bearing in mind that human microbiome data become very important in human health maintenance and novel medical treatments including personalized medicine. It is expected that machine learning will play important role in the processing of microbiome data and unveiling the patterns in those valuable data. The work in this thesis provided examples of bioinformatics workflows coupled with machine learning

algorithms. Future work should be extended with more machine learning algorithms and testing on different microbiome data sets. Research work on autoencoders is progressing very fast and new variations are emerging, such as Variational, Sparse autoencoders, Convolutional autoencoder and others. Advanced versions of autoencoders should be also investigated in the future work [22].

8. REFERENCES

[1] Turnbaugh, Peter J., et al., “The human microbiome project“, Nature, vol. 449, pp. 804-810, 2017.

[2] Gilbert, Jack A., Janet K. Jansson and Rob Knight, “The Earth Microbiome project: successes and aspirations*“*, BMC biology, vol. 12, no. 69, 2014.

[3] McDonald, Daniel, et al., “American gut: an open platform for citizen science microbiome research“, mSystems, vol. 3, no. 3, e00031, 2018.

[4] Bahcall, Orli G. , “Metagenomics: urban microbiome“ ,Nature Reviews Genetics, vol. 16, no. 4, pp. 194, 2015.

[5] Valeria D’Argenio, “Human Microbiome Acquisition and Bioinformatic Challenges in Metagenomic Studies“, Int J Mol Sci., vol. 19, no. 2, pp. 383, 2018.

[6] Caporaso et al., ″Moving pictures of the human microbiome″, Genome Biology, vol. 12, R50, 2011.

[7] QIIME 2 - next-generation microbiome bioinformatics platform that is extensible, free, open source, and community developed, https://qiime2.org/

Accessed at 15. 12. 2018.

[8] Docker: Enterprise Container Platform for High Velocity Innovation https://www.docker.com/

Accessed at 3. 12. 2018.

[9] Tringe, Susannah G., and Philip Hugenholtz, ″A renaissance for the pioneering 16S rRNA gene″, Current opinion in microbiology, vol. 11, no. 5, pp. 442-446, 2008.

[10] MG-RAST: metagenomics analysis service https://www.mg-rast.org/

Accessed at 22. 12. 2018.

[11] MG-RAST:Processing Information https://www.mg-rast.org/mgmain.html?mgpage=download&metagenome=mgm4457768.3 Accessed at 22. 12. 2018.

[12] McDonald, Daniel, et al., "An improved Greengenes taxonomy with explicit ranks for ecological and evolutionary analyses of bacteria and archaea", The ISME journal, vol. 6, no. 3, pp. 610, 2012.

[13] Schloss, Patrick D., and Sarah L. Westcott, "Assessing and improving methods used in operational taxonomic unit-based approaches for 16S rRNA gene sequence analysis", Appl. Environ. Microbiol., vol. 77, no. 10, pp. 3219-3226, 2011.

[14] Dimensionality Reduction https://www.osgdigitallabs.com/blogs/2018/4/3/dimensionality-reduction

Accessed at 25. 1. 2019.

[15] Dimensionality Reduction https://medium.com/@cxu24/why-dimensionality-reduction-is-important-dd60b5611543

Accessed at 25. 1. 2019.

[16] Lloyd N. Trefethen, David Bau, "Numerical Linear Algebra", Society for Indusry and Applied Mathematics, Society for Industrial & Applied , Philadelphia, USA, 2000.

[17] Jure Leskovec, Anand Rajaraman, Jeffrey D. Ullman, "Mining of Massive Dataset", University Printing House, Cambridge, United Kingston, 2014.

[18] Notes accompany the Stanford CS class CS231n: Convolutional Neural Networks for Visual Recognition, USA, 2019.

[19] Quoc V. Le, "A Tutorial on Deep Learning, "Part 2: Autoencoders, Convolutional Neural Networks and Recurrent Neural Networks", Google Inc, 2015.

[20] Andrew Ng, Sparse autoencoder, CS294 A, Lecture Notes, Stanford, USA, 2011.

[21] Jason Brownlee, Better Deep Learning, "Develop Deep Learning Models on Theano And Tensor Flow Using Keras", ebook, 2016.

[22] Ian Goodfellow, Yoshua Bengio, Aaron Courville, "Deep Learning", MIT Press, Cambridge, United Kingdom, 2016.

[23] Misha Denil at al., "Narrowing the Gap: Random Forests In Theory and In Practice", University of Oxford, United Kingdom, 2014.

[24] Mauro Castelli, Leonardo Vanneschi and Álvaro Rubio Largo,"Supervised Learning:Classification", Encyclopedia of Bioinformatics and Computational Biology, Elsevier, Netherlands, 2019.

[25] Classification Metrics https://apple.github.io/turicreate/docs/userguide/evaluation/classification.html

Accessed at 3. 2. 2019.

[26] Understanding the Classification report in sklearn https://muthu.co/understanding-the-classification-report-in-sklearn/

Accessed at 3. 2. 2019.

[27] Sokolova, M., & Lapalme, G., "A systematic analysis of performance measures for classification tasks", Information Processing & Management, vol. 45, no. 4, pp. 427-437, 2009.

[28] Scikit-Learn Machine Learning in Python https://scikit-learn.org/stable/index.html

Accessed at 5. 2. 2019.

[29] Keras: The Python Deep Learning library https://keras.io/

Accessed at 11. 2. 2019.

[30] Rectified Linear Units (ReLU) in Deep Learning https://www.kaggle.com/dansbecker/rectified-linear-units-relu-in-deep-learning

Accessed at 12. 2. 2019.

[31] Sigmoid function https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco /wiki/ Accessed at 12. 2. 2019.

[32] Diederik P. Kingma, Jimmy Lei Ba, Adam, "A method for stochastics optimization", 3rd International Conference for Learning Representations, ICLR 2015, San Diego, USA.

APPENDIX

Docker allows applications to use the same Linux kernel as the system that they are running on. It enables users to isolate program execution. Docker can not directly start on Windows, but on the background it creates a virtual machine where the Docker is running in such a way that we should not notice it at all [8].

APPENDIX 1

```
docker run -t -i -v D:\qiime2-vm\microbes-20181115T140850Z-
001\microbes\data_download:/data qiime2/core:2018.11

qiime tools import \

  --input-path mphm_test.fna \
  --output-path mphm_test.qza \
  --type 'SampleData[Sequences]'
```

APPENDIX 2

```
docker run -t -i -v D:\qiime2-vm\microbes-20181115T140850Z-
001\microbes\data_download:/data qiime2/core:2018.11

 qiime vsearch dereplicate-sequences \

  --i-sequences mphm_test.qza \
  --o-dereplicated-table table.qza \
  --o-dereplicated-sequences rep-seqs.qza
```

APPENDIX 3

```
docker run -t -i -v D:\qiime2-vm\microbes-20181115T140850Z-
001\microbes\data_download:/data qiime2/core:2018.11
```

```
qiime vsearch cluster-features-closed-reference \

  --i-table table.qza \
  --i-sequences rep-seqs.qza \
  --i-reference-sequences 99_otus.qza \
  --p-perc-identity 0.99 \
  --o-clustered-table table-cr-99.qza \
  --o-clustered-sequences rep-seqs-cr-99.qza \
  --o-unmatched-sequences unmatched-cr-99.qza
```

APPENDIX 4

```
docker run -t -i -v D:\qiime2-vm\microbes-20181115T140850Z-
001\microbes\data_download:/data qiime2/core:2018.11

qiime feature-table summarize \

  --i-table table-cr-99.qza \
  --o-visualization table-cr-99vizNEW.qzv \
--m-sample-metadata-file SampleMetadataNEW.tsv
```

BIOGRAPHY

Mirjana Samardžić was born on the 26th of April 1990 in Novi Sad, Serbia. She received her BSc degree in Applied Mathematics in 2016 from the Faculty of Sciences, University of Novi Sad, Serbia. She continued her Master studies in the field of Applied Mathematics: Data Science at the Faculty of Science, University of Novi Sad. During her studies Mirjana attender several Workshors such as ECMI SIG Workshop on Mathematics for Big Data, Faculty of Science, University of Novi Sad, May 31 – June 1, 2017, where she presented poster "Exploring Users Mobility From Telecom Operator Data", and DataDo Workshop - Use of Data Science tools for analyzing shopping carts and loyalty programs, September 17, ICT hub Belgrade. She also attended Training Schools on Optimization and Data Science, (Faculty of Technical Science, University of Novi Sad, March 2017) and Training School on Big Data Processing and Analytics in the Internet of Everything Era, (Faculty of Science, University of Novi Sad, September 2017) as well as Data Science Conferences (DSC 3.0, Belgrade 2017 and DSC 4.0, Belgrade 2018). Mirjana is working as Data Scientist at VizLore.

UNIVERSITY OF NOVI SAD

FACULTY OF SCIENCES

KEY WORD DOCUMENTATION

Accession number:

ANO

Identification number:

INO

Document type: monograph type

DT

Type of record: printed text

TR

Contents code: Master thesis

CC

Author: Mirjana Samardžić

AU

Mentor: Sanja Brdar, PhD

MN

Title: Dimensionality reduction of the Human Microbiome data

XI

Language of text: English

LT

Language of abstract: s/e

LA

Country of publicatin: Republic of Serbia

CP

Locality of publication: Vojvodina

LP

Publication year: 2019.

PY

Publisher: author’s reprint

PU

Publ. place: Novi Sad, Trg Dositeja Obradovića 4

PP

Physical description: text

PD

Scientific field: mathematics

SF

Scientific discipline: applied mathematics

SD

Key words: Human microbime, dimensionality reduction

UC

Holding data: Department of Mathematics and Informatics’ Library,Faculty of Sciences,

Novi Sad

HD

Note:

N

Abstract: Human microbiome studies generate huge amount of data that can bring important findings related to health care and disease diagnosis. The main characteristics of such data are

larger number of features than samples and high sparsity that impose challenges to the analytics steps. This thesis aims to explore dimensionality reduction on the human microbiome data and to evaluate classification of samples in reduced, latent space, compare to original features. Before analysis raw DNA sequence data are pre-processed using QIIME2 microbiome analysis package and grouped into Operational Taxonomic Units (OTUs). After clustering sequences into OTUs, obtained data set contains >30000 features. To uncover structure in high dimensional data of OTU features we project it down to a subspace by using well known Singular Value Decomposition (SVD)-matrix decomposition and a novel approach based on Autoencoder Neural Network, performing in this way dimensionality reduction of original data. After dimensionality reduction we apply Random Forest Classifier on both - original data and data with reduced set of features. Different measures were performed for evaluation of the algorithms such as: 1) accuracy classification score, 2) confusion matrix which provide detailed classification across all classes 3) classification report derived from confusion matrix showing the other important classification metrics. We provide results in reduced space with respect to different number of dimensions and evaluate how this parameter impacts the classification task.

AB

Accepted by the Scientific Board on: 09.05.2019.

ASB

Defended:

DE

Thesis defend board:

DB

President: Dušan Jakovetić, PhD Assistant Professor

Member: Tatjana Lončar Turukalo, PhD Associated Professor

Member: Sanja Brdar, PhD Research Assistant Professor

UNIVERZITET U NOVOM SADU

PRIRODNO-MATEMATIČKI FAKULTET

KLJUČNA DOKUMENTACIJSKA INFORMACIJA

Redni broj:

RBR

Identifikacioni broj:

IBR

Tip dokumentacije: monografska dokumentacija

BF

Tip zapisa: tekstualni štampani materijal

TZ

Vrsta rada: Master rad

VR

Autor: Mirjana Samardžić

AU

Mentor: dr Sanja Brdar

MN

Naslov rada: Redukcija dimenzionalnosti nad podacima ljudkog mikrobioma

NR

Jezik publikacije: engleski

JP

Jezik izvoda:s/e

JI

Zemlja publikovanja: Republika Srbija

ZP

Uže geografsko područje: Vojvodina

UGP

Godina: 2019.

GO

Izdavač: autorski reprint

IZ Mesto i adresa: Novi Sad, Trg Dositeja Obradovića 4

MA

Fizički opis rada: text

FO

Naučna oblast: matematika

NO

Naučna disciplina: primenjena matematika

ND

Ključne reči: Ljudski mikrobiom, redukcija dimenzionalnosti

PO

UDK

Čuva se: u biblioteci Departmana za matematiku i informatiku,

Prirodno-matematičkog fakulteta, u Novom Sadu

ČU

Važna napomena:

VN

Izvod: Studije ljudskog mikrobioma generišu veliku količinu podataka koji doprinose značajnim otkrićima vezanim za zdrastevu zastitu i dijagnozu bolesti kod ljudi. Glavne karakteristike ovog tipa podatake jeste velika dimenzionalnost koja stvara poteškoće pri daljoj analizi. Ovj rad ima za cilj da istraži smanjenje dimenzionalnosti nad podacima mikrobioma u poređenju sa originalnim karakreristikama. Pre analize, podaci o sekvencama DNK su predhodno obrađeni korišćenjem paketa za analizu mikrobioma QIIME2 i grupisani u Operativne taksonomke

jedinice, dobijeni skup podataka sadrži >30000 karakteristika. Da bismo otkrili strukturu u visoko dimenzionalnim podacima OTU karakteristika projektujemo je do potprostora koristeći dobro poznatu dekompoziciju singularnih vrednosti SVD i novi pristup zasnovan na Autoencoder neuronskoj mreži, na taj način izvršavajući smanjenje dimenzionalnosti originalnih podataka. Nakon smanjenja dimenzionalnosti primenjujemo random forest klasifikator kako na originalne tako i na redukovane podatke. Izvršene su različite mere za procenu algoritama kao što su: 1) ocean tačnosti klasifikacije 2) matrica konfuzije koja obezbeđuje detaljnu klasifikaciju svih kals 3) izveštaj o klasifikaciji izvedene matrice konfuzije koji pokazuje ostale važne metrike klasifikacije. U ovon radu pokazali smo rezultate u redukovanom prostoru u odnosu na različiti broj dimenzija i procenjujemo kako ovaj parameter utiče na zadatak klasifikacije.

IZ

Datum prihvatanja teme od strane NN veća: 09.05.2019.

DP

Datum odbrane:

DO

Članovi komisije:

KO

Predsednik: dr Dušan Jakovetić, docent

Član: dr Tatjana Lončar Turukalo, vanredni profesor

Član: dr Sanja Brdar, naučni saradnik