CS F376 End-Semester Project Report

Student: Sachita Nishal
ID Number: 2016A7PS0719G

Supervising Faculty: Dr. Sukanta Mondal

4 December, 2018

Objective:

The aim of this semester’s undertaking was to apply disease classification algorithms to gene-expression datasets, in order to obtain good classification accuracy and understand the effects of the chosen algorithms.

Our starting point last semester was to conduct a survey of currently existing methods that adopt different approaches to work with massive amounts of molecular data. Understanding how these methods implement data preprocessing, feature selection, dimension reduction etc. was vital to proposing efficient alternatives for the same. Based on these ideas, we constructed an early stage pipeline in order to classify a multiclass Lung Cancer dataset[1] (4 disease classes + 1 normal case).

[1] "Classification of human lung carcinomas by mRNA expression ... - PNAS." Accessed December 6, 2018. http://www.pnas.org/content/98/24/13790.

Pre-Processing:

Last semester's readings on EllipsoidFN[2] algorithm shed light on the usage of entropy to filter out genes from the dataset if they exhibited less variation in their expression values across classes. It is used to identify the genes which are differentially expressed across separate samples and classes i.e. have high *population sparsity.*

Following this, the entropy calculations were carried out, and the entropies of individual genes were plotted to obtain a more comprehensive picture of the distribution. The plots are given below:

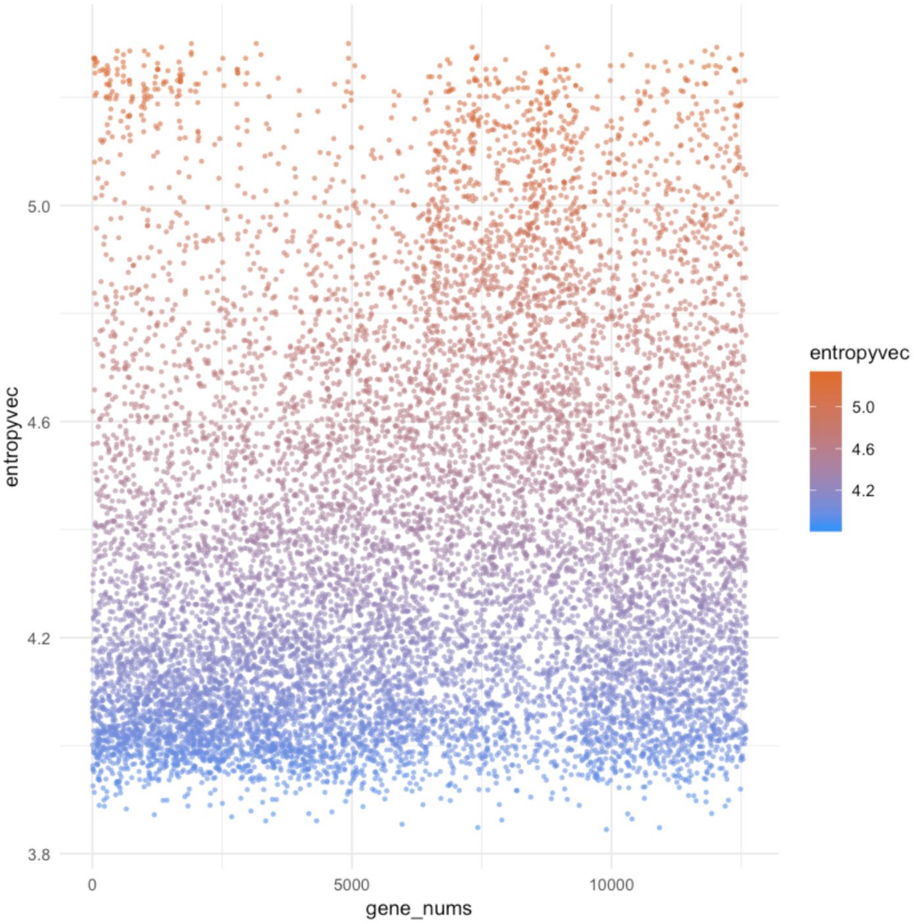

Fig 1. Gene vs. Entropy graph

[2] "ellipsoidFN: a tool for identifying a heterogeneous set of cancer ... - NCBI." Accessed December 4, 2018. https://www.ncbi.nlm.nih.gov/pubmed/23262226.

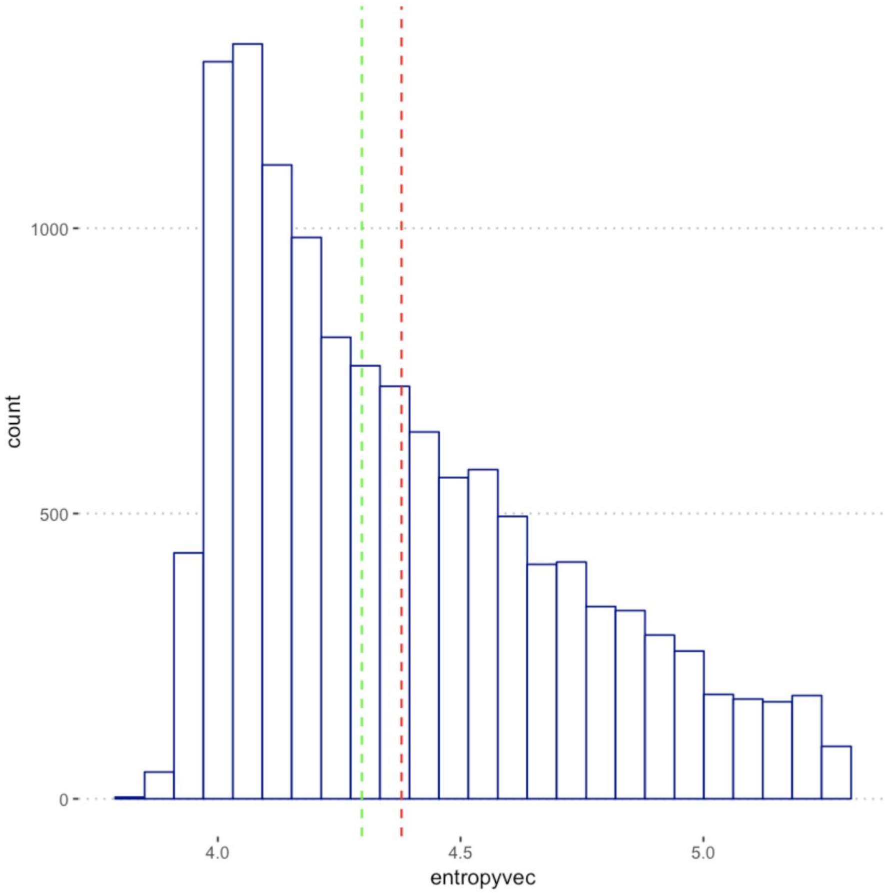

Fig 2. Histogram for ranges of entropy values vs count of genes in that range. Mean of dataset as red dotted line, and median as green dotted line

After this, the entropy filter technique was applied, and various subsets of the gene expression values, based on varying the threshold entropy value for the filter.

After this process, Principal Component Analysis[3] was used to project the dataset onto a lower dimension space. To carry this out, the cumulative summation of the explained variance was plotted, to obtain the necessary amount of principal components:

[3] "Principal Component Analysis explained visually - Setosa.IO." Accessed December 4, 2018. http://setosa.io/ev/principal-component-analysis/.

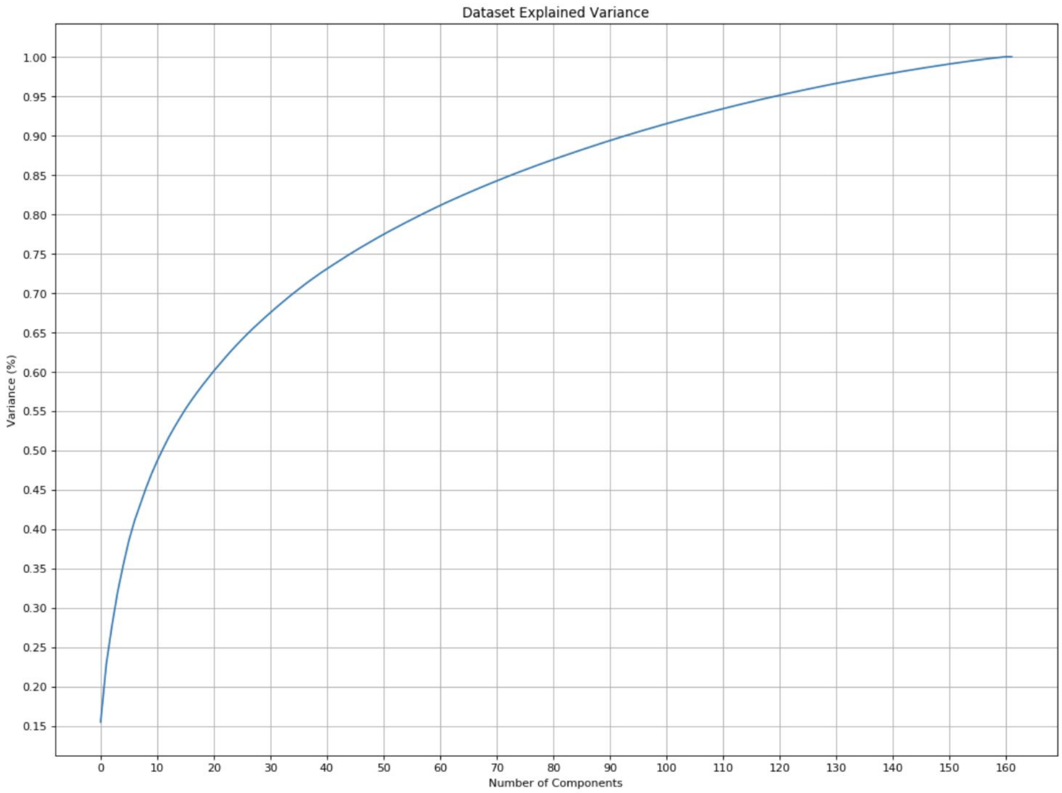

Fig 3. *Cumulative summation of the explained variance (in %) vs number of components necessary*

This determined the number of components required to retain 95% of the variance of the dataset (95% is the threshold in popular practice of PCA), which was 121 components. These components were selected and PCA was completed.

Classification:

A preliminary neural network model was implemented using the Keras library in Python, in order to get a foundation for work on classification. Here is the model summary for the same:

```
_________________________________________________________________

Layer (type)            Output Shape          Param #

=================================================================

dense_4 (Dense)           (None, 12)           1464

_________________________________________________________________

dense_5 (Dense)           (None, 6)            78

_________________________________________________________________

dense_6 (Dense)           (None, 5)            35

=================================================================

Total params: 1,577

Trainable params: 1,577

Non-trainable params: 0

_________________________________________________________________
```

The first 2 layers employed ReLU activations, whereas the third one used Softmax activation for multiclass classification. This is a relatively small network, on which optimisation work is ongoing. The best accuracy it obtained was on a subset of data with an entropy filter of 4.8, and it gained an accuracy of 89%. Work is ongoing to make evaluation of the neural network more effective by converting it into Stratified K-Fold Model[4], which evaluates a model more rigorously, and in multiple iterations.

[4] "sklearn.model_selection.StratifiedKFold — scikit-learn 0.20.1 ...." Accessed December 4, 2018. http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html.

Future Work:

This project will be continued in its last leg over the next semester, and the following tasks will be carried out:

- Further optimisation of neural network
- Conversion of neural network to Stratified K-Fold Model
- Using K-Means Clustering[5] on dataset and comparing results with the results of neural network
- Mapping the chosen PCA components back to the original genes
- Potentially using the results to map the causal genes to Protein-Protein Interaction Networks

[5] "sklearn.cluster.KMeans — scikit-learn 0.20.1 documentation." Accessed December 4, 2018. http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html.

Appendix

Google Drive link with the code implemented and the dataset used.