|  |
| --- |
| **Understanding of Bioinformatics**  **using AI and Big Data** |
| **Final report** |



|  |  |
| --- | --- |
| Submission Date | December 4th, 2018 |
| Subject | Understanding of Bioinformatics  using AI and Big Data |
| Professor | Jaegyoon Ahn Prof |
| department | Industrial Management Engineering |
| Student ID | 201401210 |
| name | Hyeongwon Kang |

Contents

1. **Introduction**
2. Set goals for project
3. Data description
4. **Midterm report summary**
5. KNN
6. Naive Bayesian Classification
7. Information Gain
8. SVM
9. Deep Neural Network
10. **Method**
11. preprocessing
12. Develop DNN
13. **Result**
14. **Discussion**

* Appendix (Code)

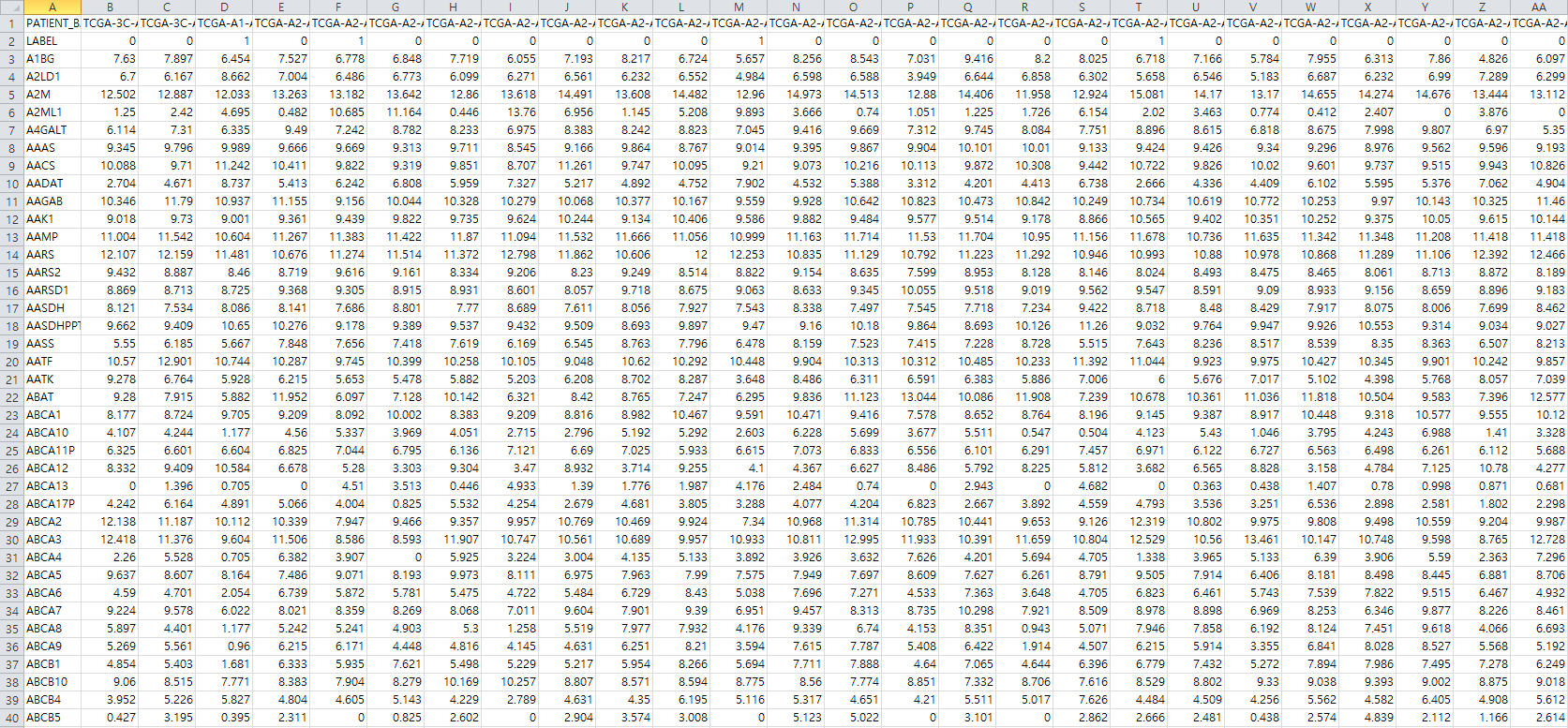
1. **Introduction**
2. **Set goals for project**

* Use BRCA\_prognosis data to create a program that can detect gene anomalies and predict breast cancer.

1. Execute test data and training data separately.
2. Using gene data from patients, distinguishes between good and risky genes.
3. As a result, the program will be makes good and risky predictions of the gene.
4. **Data description**

* Data structure

(I converted the data to a csv file for easy viewing.)

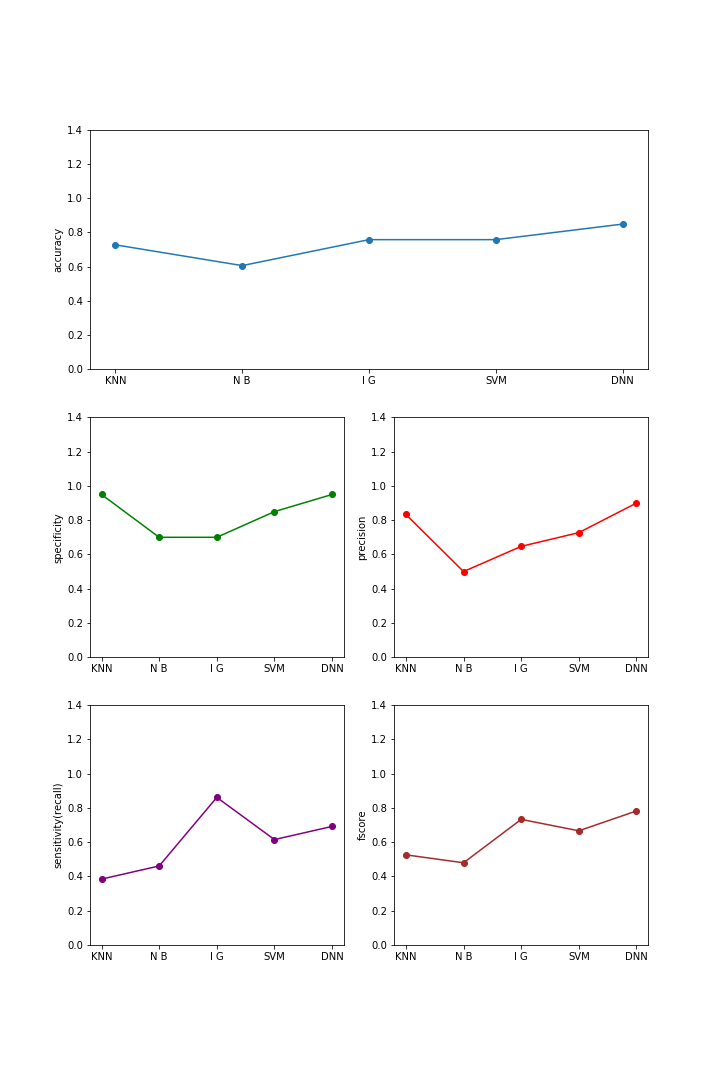


|  |  |
| --- | --- |
| Gene | Gene (16603) ex) A1BG |
| PATIENT\_BARCODE | Patient ID (165) ex) TCGA-3C-AAAU |
| LABEL | Good(0): 98, Risky(1): 67 |

1. **Midterm report summary**

**Without preprocessing, the results of unsupervised learning**

1. KNN (k=[3, 5, 7, 9, 11, 13, 15])
2. Naive Bayesian Classification
3. Information gain ( max\_depth=[3, 5, 7, 9, 11, 13, 15] )
4. SVM (kernel = [linear, poly, rbf, sigmoid])
5. DNN (solver=[adam, sgd, lbfgs], activation= [identity, logistic, tanh, relu])

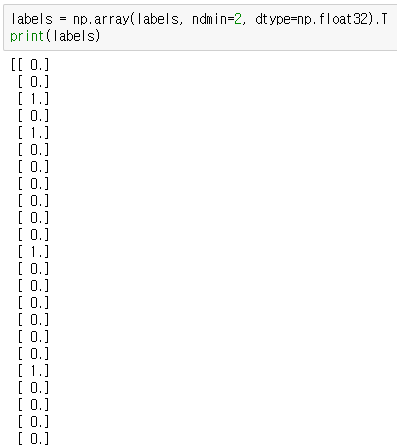
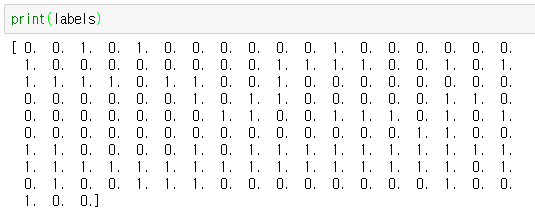


The accuracy of DNN was the highest at about 0.85, and DNN had the highest value except the sensitivity. As a result, DNN (solver = ibfgs, activation = logistic) is the best classification.

When SVM kernel is sigmoid and DNN solver is adman and sgd, it is not classified properly.

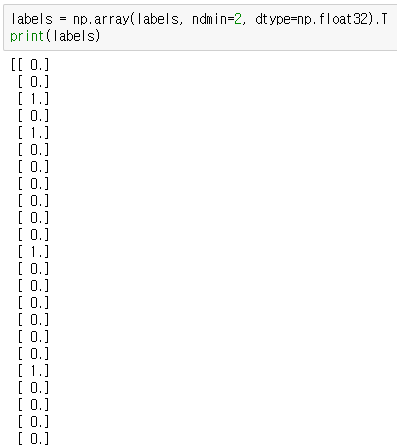
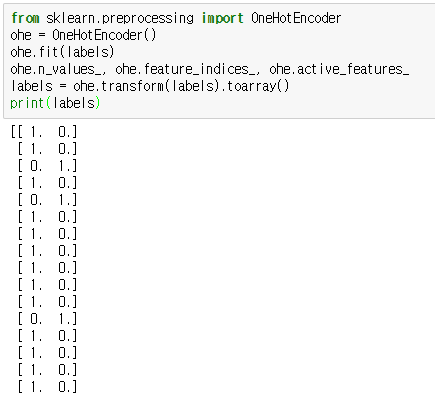
1. **Method**
2. Preprocessing
3. Edit labels array

For use in the DNN model, the Labels array is changed to a two-dimensional array, and the column vectors are replaced by row vectors.



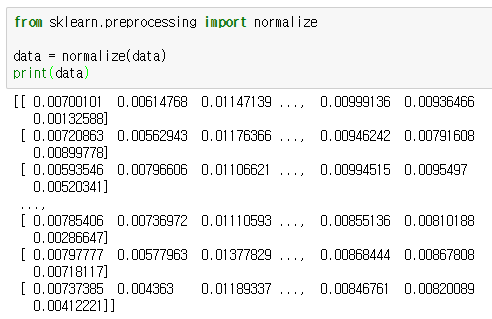
1. One-Hot-Encoding

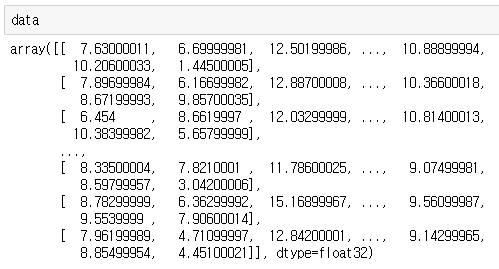
One-Hot-Encoding is used to change the values ​​of labels. One-Hot-Encoder is also referred to as One-of-K encoding and converts an integer scalar value having a value of 0 to K-1 into a K-dimensional vector having a value of 0 or 1.



1. Normalization

Normalize the values of the data. Normalization is a transformation to make all of the individual data the same size.





1. Data grouping

Divide into two groups with similar characteristics to get better results.

|  |  |
| --- | --- |
| 1. Principal component analysis (PCA) | 1. K-means |
| The dimension of the data is reduced to two dimensions. | Use K-means to divide into two groups. |
| pca | Figure_1 |

1. Grouping

data\_train

Cluster[0]

data\_ train \_gp1

Cluster[1]

data\_ train \_gp2

labels\_ train

Cluster[0]

labels \_ train \_gp1

Cluster[1]

labels \_ train \_gp2

data\_test

Cluster[0]

data\_test\_gp1

Cluster[1]

data\_test\_gp2

labels\_test

Cluster[0]

labels \_test\_gp1

Cluster[1]

labels \_test\_gp2

1. Training
2. DNN (Deep Neural Network)

Input

(16603)

Hidden\_1

(4096)

Hidden\_2

(1024)

Hidden\_3

(256)

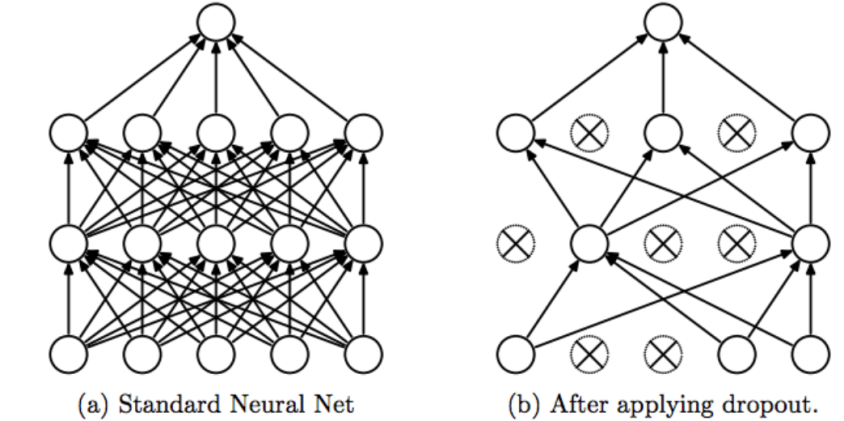
Hidden\_4

(32)

Output

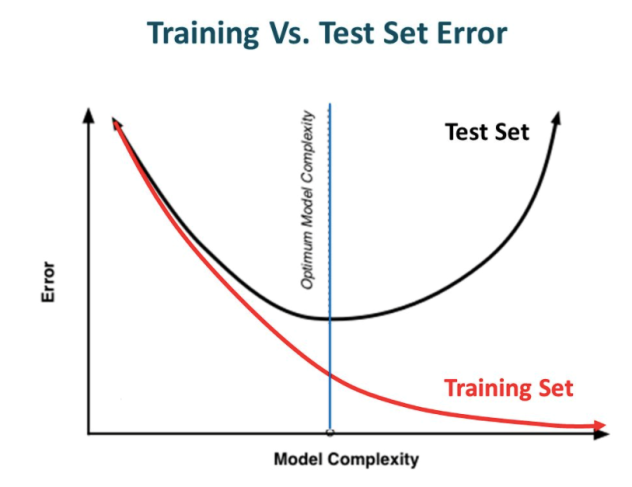
(2)

The hidden layer is composed of four layers (4096, 1024, 256, 32) and the Learning\_rate is set to 0.0001. I used the solver as the adam optimizer function and the activate function as relu. Train step were set to 500.

1. Dropout

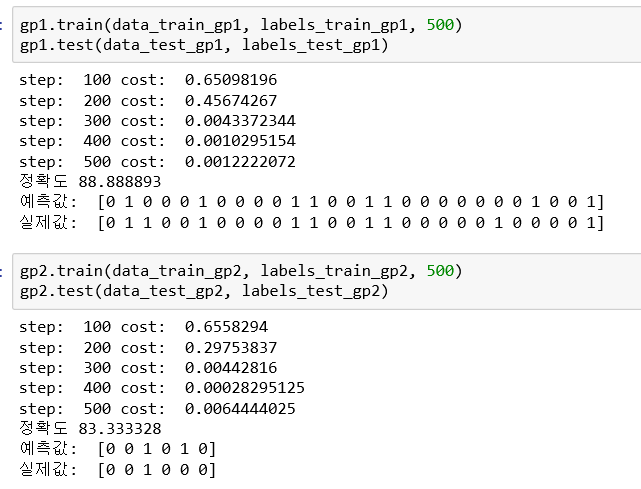
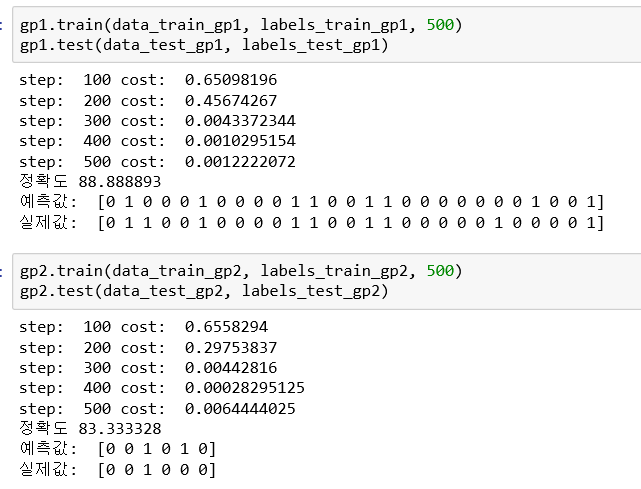
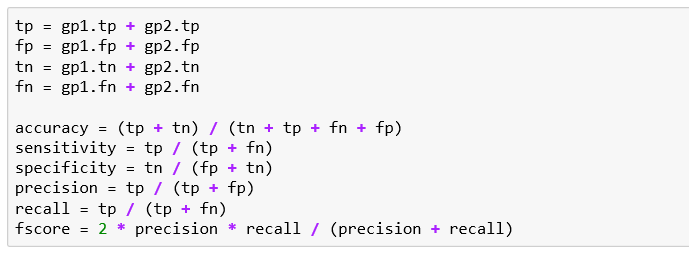
Avoid using some of the neurons at each learning step to prevent some features from sticking to specific neurons, balancing the weights to prevent overfitting.

Dropouts were set to 0.8.

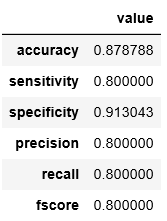
1. Regularization

Let’s not have too big numbers in the weight. And, prevent overfitting.

Reularization was set to 0.001.

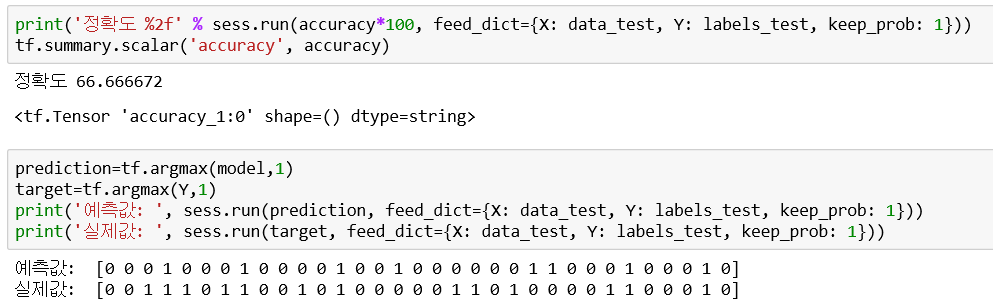
1. **Result**
2. Result
3. First Group & Second Group
4. Sum first group, seconde group result

|  |  |  |  |
| --- | --- | --- | --- |
| Predicted  Actual | Good(0) | Risky(1) | Total |
| Good(0) | 21 | 2 | 23 |
| Risky(1) | 2 | 8 | 10 |
| Total | 23 | 10 | 33 |

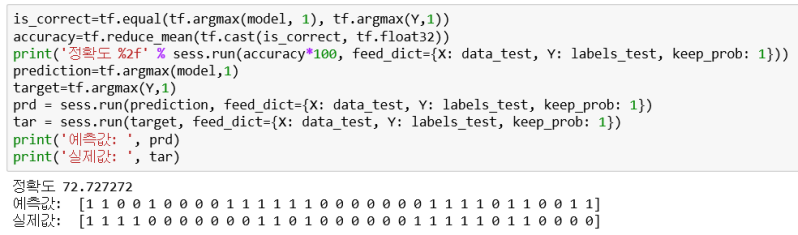


29 of the 33 data were matched. Accuracy was 0.88.

1. Compare with other methods
2. Not grouping, Not regularization, node(1024,256,32)



1. Not grouping, Not regularization, node(4096,1024,256,32)



1. Not grouping, Use regularization, node(4096,1024,256,32)



1. **Discussion**
2. I wanted to get better results but I was sorry that I could not.
3. I realized that it takes a lot of effort and time to create a predictive model.