Journal of Data Acquisition and Processing
ISSN: 1004-9037
https://sjcjycl.cn/
DOI: 10.5281/zenodo.7674239

PR V1.0, A DEEP LEARNING APPROACH FOR FEATURE SELECTION FROM MICROARRAY DATA

Vishwas Victor[1]*, Ragini Shukla[2]
[1]Department of IT & CS, C. V. Raman University, Bilaspur
[2]Department of IT & CS, C. V. Raman University, Bilaspur
* Corresponding author E-mail: victor.vishwas365@gmail.com

Abstract:
In the field of medical science, feature (or gene) selection is a burning topic. Microarray data are very important for the feature selection to diagnose any disease. It may be possible that few thousands of features are present in microarray data. That's the reason why to find out a proper subset of features using conventional algorithms, it becomes an important object. We have to reduce the whole dataset's dimension in order to produce a proper subset, keeping in mind that we don't have to miss significant features while removing redundant features. For disease diagnosis small features set with minimal features. Autoencoder technique is proved to be very powerful and efficient for reducing the dimension. Inspired by the Autoencoder technique, both of us built a model based on Folded Autoencoder (FA) for the selection of features set. After that, some deep learning classifiers are applied to check the accuracy of classifier. Performance of Support Vector Machine (SVM) is better than other classifiers after reducing the dimension of features. This model is named as Folded Autoencoder – SVM (FAS). Lastly, there is a comparison of results obtained from whole dataset (without applying FA) and reduced dataset (after applying FA).
Keywords: PR v1.0, Microarray Data Analysis, Genes, Feature Selection, Folded Auto-encoder, SVM, KNN

1. INTRODUCTION:

In the prediction system for any disease in recent years, machine learning and deep learning algorithms have been applied. Because microarray data has a large number of characteristics and few samples, analysing it might be difficult. Researchers are working hard to identify the best subset of the entire microarray data for this reason. Deoxyribonucleic Acid is harmed by ongoing cell changes (DNA). The DNA sample sequences are included in the microarray data (part III). There are thousands of characteristics in the microarray dataset because there are more samples of cancer patients, tumour patients, patients with other diseases, and healthy patients. This makes determining the accuracy of any learning classifier a challenging process. The health care prognostic system relies on feature selection (FS) as a key component to solve these kinds of issues. Many scholars have suggested various feature selection, dimension reduction, and classification strategies; the majority of them are included in section II. In addition to this, this study has a contribution from our end; we suggested the Folded Auto-encoder Support Vector Machine (FAS) method. SVM produces superior classification results and Folded Auto-encoder (FS) is utilized to minimise the dimension. We fold the network from an encoded layer in a folded auto-encoder (section IVB), which also contains some hidden layers. Actually, the selection, dimension reduction, and categorization

of features are all made possible by machine learning and deep learning algorithms. Due to many types of interfaces and open sources, implementing machine learning and deep learning models is a less difficult process. The open-source platform TensorFlow is used to represent various learning methods. Calculate numerical values and easily solve challenging issues. Since the convoluted auto-encoder is a neural network-based approach (a hard work numerically), we use TensorFlow here to simplify the computational part. In addition, it can be said to be a system that processes numerical information based on deep learning by changing the complex information structure in an artificial neural network.

This article is structured as follows- The second part (Section 2) discusses the relevant work, the third part (Section 3) presents an overview of microarray technology, the fourth part (Section 4) describes the proposed model, the fifth part (Section 5) consists implementation of analysis, the simulation analysis and results are carried out and described in the sixth part (Section 6) and in the seventh part (Section 7) is conclusion.

2. RELATED WORKS:

Different types of methods have been proposed by several researchers for the selection of microarray data, the bare minimal number of features or genes (or subsets of characteristics). Another crucial factor is feature selection. The reason behind this is that it's important to find out whether any particular patient is normal patient or abnormal patient in less time. Now, we are going to discuss briefly about existing techniques, related to our proposed work after doing detailed literature survey.

In [9], the authors have proposed an Artificial Intelligence based algorithm namely Recursive Memetics for feature selection focussed on improving the accuracy of classification. Here, Wrapper-Filter-Feature-Selection-Algorithm is proposed along with recursion technique and fitness is also modified with reduction criteria. In [19], [1], to select top ranked genes, the authors have designed and developed a max-relevance-min redundancy (mRMR) filter method. In [18], the author reduced the dimension using attribute selection & principal component analysis technique along with the combination of consistency-based subset evaluation and minimum redundancy maximum relevance. But in most of the cases, classification accuracy doesn't give good result just because of Principal Component Analysis (PCA). This is the reason why, almost all researchers prefer Deep Auto-encoder Technique to reduce the dimension, based con deep neural network. In [2], a Deep Wavelet Auto-encoder is combined with Deep Neural Network to compress the images that means, dimension of image is reduced. In [7], S. Kilicarslan proposed a hybrid method by combining ReliefF and Stacked Auto-encoder approach for reducing dimension. For classification purpose, Convolutional Neural Network and Support Vector Machine are used. These dimension reduction techniques are employed for improving the classification accuracy. In [6], authors used Deep Flexible Neural Forest (DFNForest) Network Model. It changes, multi-class problem to binary-class problem in each forest. In some of the cases Principal Component Analysis (PCA) and Auto-encoder are combined together for feature identification and selection [18]. In [14], the Convolutional Neural Network is introduced for feature selection and classification. In [15], "P. K. Ram and P. Kuila" selected employing genetic algorithms to identify the fewest characteristics with the highest accuracy from microarray data. Compared to other evolutionary algorithms, the genetic algorithm has the highest accuracy with the fewest feature evaluations [13]. A hybrid

framework is established for feature selection and for classification purpose using Support Vector Machine, by combining the genetic algorithm & tabu search [3]. In [11], the authors proposed combining "Kernalized Fuzzy Rough Set and Support Vector Machine to predict cancer biomarker". After that, to improve the accuracy of classification, they took help of Semi Supervised Support Vector Machine. In [8], the author proposed a method called Score Based Criteria Fusion Selection Method after combining the two methods – Symmetrical Uncertainty and ReliefF method. Here, it should be noted that the Symmetrical Uncertainty method is used to relate relevance after avoiding redundant features whereas the ReliefF method is employed to distinguish the feature. In [5], Folded Neural Network Auto-encoder technique is designed and developed by the author for dimensionality reduction. It is basically based on Conventional Simple Auto-encoder technique.

Our aim is to propose an efficient model to select the minimum feature set with maximum accuracy and for that purpose, we have considered Folded Auto-encoder deep learning technique. To generate the model structure, we used to fold the right side of the traditional Auto-encoder to the left side. It is more efficient as compared to PCA dimensionality reduction technique. The only reason behind this is use of several neural layers and non-linear transformation. As far as structure of Folded Auto-encoder is concerned, it is simple and is efficient enough to reduce time required for computing by enhancing the generalization performance. After that we have added SVM classifier with this that leads to better performance.

3. MICROARRAY TECHNOLOGY:

Microarray data is a throughput technology which is used in cancer research for diagnosis and prognosis of disease. To find the patterns among the normal and abnormal tissue, Microarray technology is the easiest way. Microarray is a silicon chip where presents thousands of tiny spots. A tissue or DNA sample is supposed to be placed on tiny spots of the chip. DNA stands for deoxyribonucleic acid and is hereditary material that is present in human beings and almost all other living organisms. Firstly, different genetic samples are required to be collected from unknown (normal and abnormal) persons. These are inserted carefully to the tiny spots of the silicon chip. Then DNAs are hybridized. These DNAs are attached to its compliment DNA and RNA, mRNA and tRNA are isolated using anneal primer. After labelling mRNA by transcriptase enzyme, a labelled cDNA is created. A laser emission (green or red) is affected onto the chip after the hybridization of labelled cDNA. Finally, microarray data is generated after computing the intensity.

3.1 Description of Microarray Datasets

Most of the datasets are available at "https://www.ncbi.nlm.nih.gov, http://csse.szu.edu.cn/staff/zhuzx/Datasets.html, http://www.biolab.si/supp/bi-cancer/projections/ and https://web.stanford.edu/~hastie/CASI_files/DATA/leukemia.html". In this study, experimental works were done with the help of five microarray datasets which are publicly available and concise as given in Table 1.

Table 1: Description of Microarray Datasets

Datasets	No. Of Features	No. of Samples	No. of Classes

Colon	2115	65	2
CNS	7257	63	2
Ovarian	15134	257	2
Leukaemia	7145	70	2
Prostate	12437	105	2

i. **Colon:** The Colon Cancer dataset has 27 samples from normal person and 38 samples from abnormal patient with 2115 features.

ii. **CNS:** The CNS (Central Nervous System) Cancer has 19 samples from normal person and 44 samples from Cancer patients with 7257 features.

iii. **Ovarian:** The Ovarian Cancer dataset has 95 samples from normal person and 162 samples from abnormal / cancer patients with 15134 features.

iv. **Leukaemia:** The Leukaemia Cancer dataset has 29 samples from AML (i.e. Acute Mycloid Leukaemia) patient and 41 samples from ALL (i.e. Acute Lymphoblastic Leukaemia) patients with 7145 features.

v. **Prostate:** The Prostate cancer dataset has 47 samples from normal person and 58 samples from abnormal patient with 12437 features.

4. PROPOSED METHOD:

It is proved to be a challenging task in medical research to select the minimum number of features with high accuracy. Actually, the main reason behind this is the high dimensionality of the microarray data as it has maximum features and irrelevant data with a very less number of samples. In our proposed method, we gave a solution to these kinds of problems. For this, e only utilised the Folded Autoencoder to minimise the dimension or to produce a subset of features using the same amount of samples but fewer features. Thereafter, we have used some Machine Learning algorithms to reduce dataset. A SVM classifier provides a better result as compared to other classifiers. At the end, we compared the accuracy of this classification along with whole dataset classifications' accuracy. As a result, we found that the reduced data with SVM provides higher accuracy. This is the reason why; FAS is the name of the model we propose (Folded Autoencoder - SVM) also code-named PR v1.0.

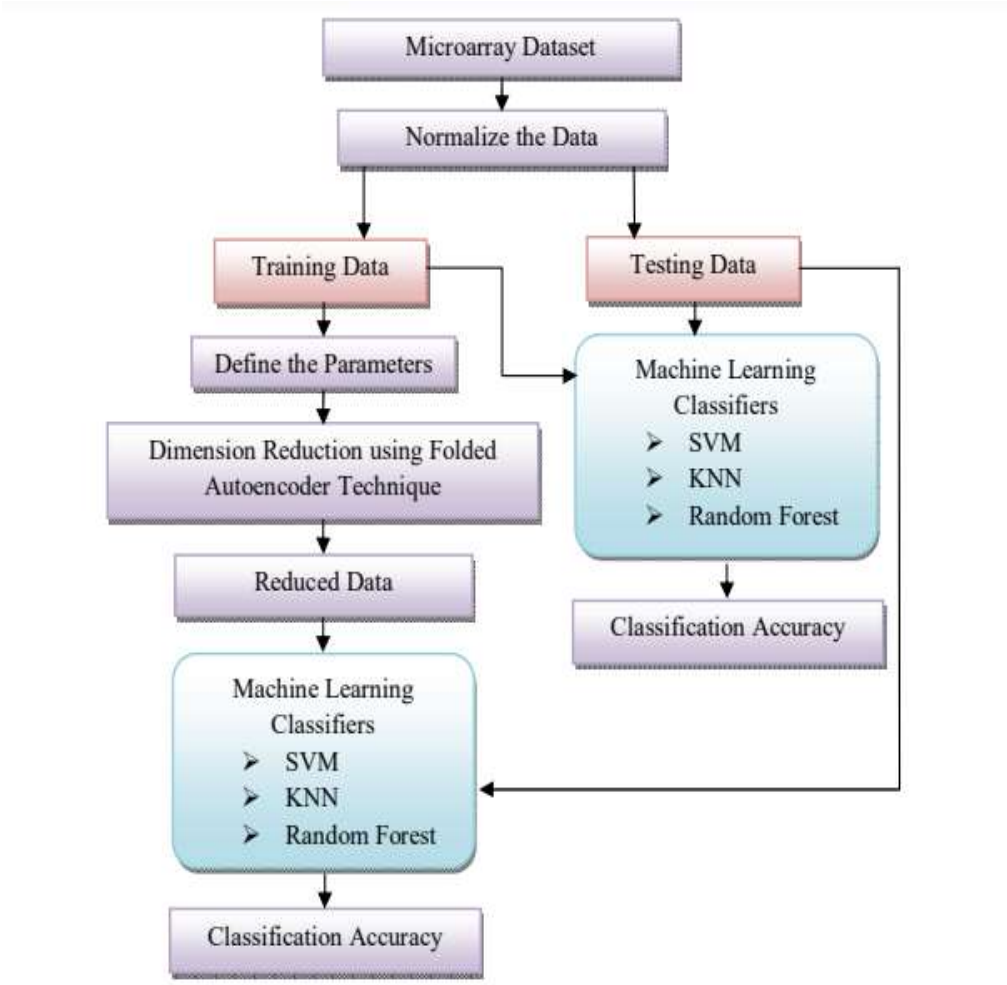

Figure 1: Project Flow Chart

A. Normalization of Datasets:

Firstly, we will gather all the datasets (Section 3.1), after which we will use the min-max normalisation approach to normalise the dataset (please refer equation 1). With this method, all numbers between 0 and 1 are rearranged. It causes our model's numerical efficiency to increase.

Equation 1

$$\hat{X}[:,i] = \frac{X[:,i] - min(X[:,i])}{max(X[:,i]) - min(X[:,i])}$$

Where, 'X'is the total dataset and 'X[:,i]' illustrate feature 'i'.

B. Folded Autoencoder

Autoencoder is nothing but the deep learning optimization technique to compress the original data from the compressed data. Actually, the Autoencoder is generally used to reduce the dimensionality. Encoders and Decoders are processed to compress the high dimensional data and to reconstruct the data simultaneously. If we talk about the Input and Output layers, both are same. In this case, hidden layers are used to compress the dimension of data and a

back-propagation technique to set the output value. Here it should be noted that to generate new feature subset, we have proposed a Folded Autoencoder. This is based upon the simple conventional Autoencoder. The Folded Autoencoder will contain (N-1)/3 hidden layers which is same as compared to Unfolded Autoencoder with N number of hidden layers. Here, the computing cost is reduced to and the actual reason behind this is the Folded Autoencoder. It may be noted that the Weights are initialized using the random initialization. The original high-dimensional data are compressed and reach the coding layer when Folded Autoencoder acts as an encoder (Encoded Representation). When it functions as a decoder, the input data are then rebuilt from the compressed data. The input/output layer, three hidden layers, one code layer, and a one-third reduction in each temporal dimension are all employed in the proposed model. The input and output layer dimensions are the same for each hidden layer, with the first hidden layer being (input dimension / 3) and the second hidden layer being (input dimension / 9) and the third hidden layer's being (input dimension / 27) and (input dimension / 81), respectively. If we take a look at the pattern, we will find out that in our proposed model, "the dimension of features is reduced by the **power of three** in every layer". If we talk about the code layer, it is the newly generated subset of features.

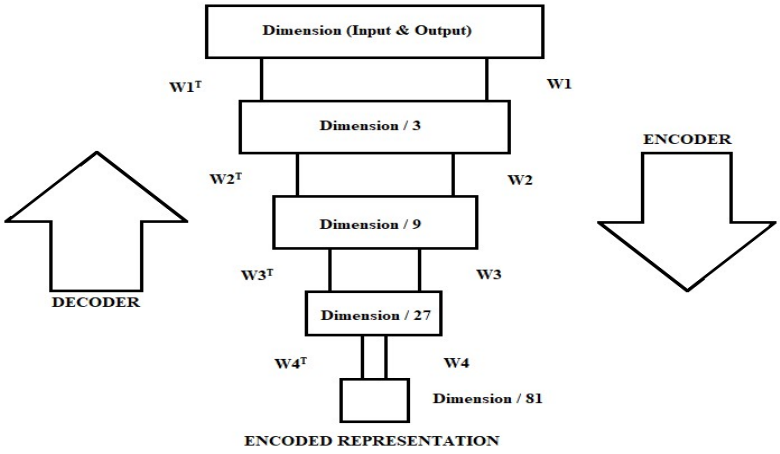

Figure 2: Framework of Folded Autoencoder

C. Parameter Selection:

The highest accuracy value is used to pick each parameter. Network parameters in the input and output layers are the same as the input dimension for dimensionality reduction. It makes up one-third of the input/output layer in the first hidden layer, one-third of the first hidden layer in the second hidden layer, one-third of the second hidden layer in the third hidden layer, and one-third of the third hidden layer in the code layer. For the Folded Autoencoder model, training epochs are set to 100 and learning rate at 0.001.

5. IMPLEMENTATION:

For the implementation purpose, we have done the experiment for the proposed method using Python 3.9 version and Jupiter Notebook. We used System with Intel i5, 2.5 GHz processor, 4GB RAM and Ubuntu Operating System. We are hereby attaching the Code Snippets, ROC-AUC Curve as well as Scatter Plotter Graphs (On the basis of which analysis is done).

Let us consider the following-

```
import pandas as pd

import numpy as np

import seaborn as sns

import os

import matplotlib.pyplot as plt
%matplotlib inline

input = pd.read_csv('e:\\leukemia_dataset.csv')

input = pd.read_csv('e:\\leukemia_dataset.csv',low_memory=False)

df = input
```

Figure 3: Pre-processing of Data

```
# Normalise the data
def dfNormalize(df):
    for feature_name in df.columns:
        df.loc[:,feature_name]= pd.to_numeric(df.loc[:,feature_name], errors='coerce').fillna(0)
        max_value = df[feature_name].max()
        min_value = df[feature_name].min()
        if (max_value - min_value) > 0:
            df.loc[:,feature_name] = (df.loc[:,feature_name] - min_value) / (max_value - min_value)
        else:
            df.loc[:,feature_name] = (df.loc[:,feature_name]- min_value)
    return df
```

```
df.shape
```

```
(72, 7071)
```

```
df.drop_duplicates(inplace=True)
df = df.replace([np.inf, -np.inf], np.nan)
df = df.dropna()

print(df.shape)
```

```
(72, 7071)
```

Figure 4: Normalizing the Data

```
# roc curve and auc
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from matplotlib import pyplot
# generate 2 class dataset
X, y = make_classification(n_samples=1000, n_classes=2, random_state=1)
# split into train/test sets
trainX, testX, trainy, testy = train_test_split(X, y, test_size=0.5, random_state=2)
# generate a no skill prediction (majority class)
ns_probs = [0 for _ in range(len(testy))]
# fit a model
model = LogisticRegression(solver='lbfgs')
model.fit(trainX, trainy)
# predict probabilities
lr_probs = model.predict_proba(testX)
# keep probabilities for the positive outcome only
lr_probs = lr_probs[:, 1]
# calculate scores
ns_auc = roc_auc_score(testy, ns_probs)
lr_auc = roc_auc_score(testy, lr_probs)
# summarize scores
print('No Skill: ROC AUC=%.3f' % (ns_auc))
print('Logistic: ROC AUC=%.3f' % (lr_auc))
# calculate roc curves
ns_fpr, ns_tpr, _ = roc_curve(testy, ns_probs)
lr_fpr, lr_tpr, _ = roc_curve(testy, lr_probs)
# plot the roc curve for the model
pyplot.plot(ns_fpr, ns_tpr, linestyle='--', label='No cancer')
pyplot.plot(lr_fpr, lr_tpr, marker='.', label='Logistic')
# axis labels
pyplot.xlabel('False Positive Rate')
pyplot.ylabel('True Positive Rate')
# show the legend
pyplot.legend()
# show the plot
pyplot.show()
```

Figure 5: Analysing the Data

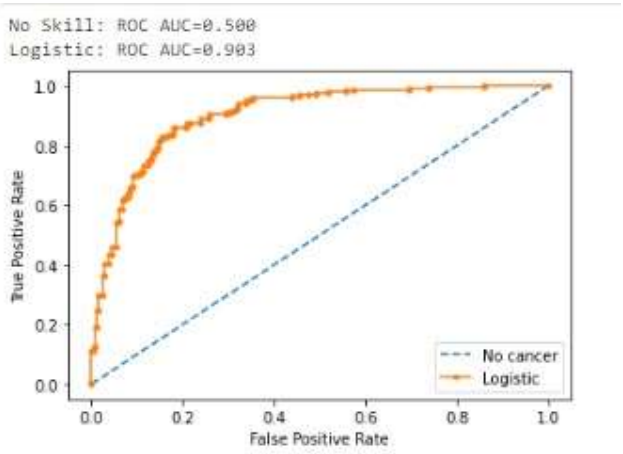

Figure 6: ROC- AUC Curve of Result Analysis

```
df = pd.DataFrame(np.random.randn(100, 12), columns=['v1', 'v2', 'v3', 'v4', 'v5', 'v6', 'v7', 'v8', 'v9', 'v10', 'v11', 'v12'])


ax1 = df.plot(kind='scatter', x='v1', y='v2', color='r')
ax2 = df.plot(kind='scatter', x='v3', y='v4', color='g', ax=ax1)
ax3 = df.plot(kind='scatter', x='v5', y='v6', color='b', ax=ax1)

print(ax1 == ax2 == ax3)  # True
```

```
True
```

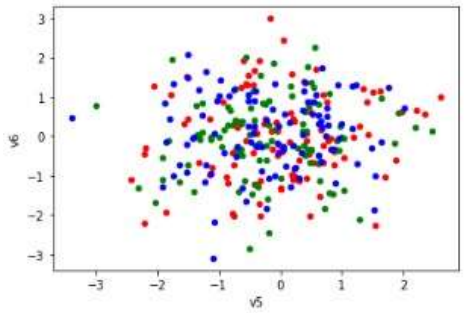

Figure 7: Result Representation of Data

```
df.describe()
```

	class	v1	v2	v3	v4	v5	v6	v7	v8	v9	...	v7061	v7062	v7063	v7064	v7065	v7066	v7067	v7068	v7069	v7070
count	72.000000	72.000000	72.000000	72.000000	72.000000	72.000000	72.000000	72.000000	72.000000	72.000000	...	72.000000	72.000000	72.000000	72.000000	72.000000	72.000000	72.000000	72.000000	72.000000	72.000000
mean	-0.305556	-0.111111	0.000000	-0.111111	-0.194444	-0.055556	-0.277778	-0.305556	-0.166667	-0.138889	...	-0.083333	-0.138889	0.194444	-0.194444	0.111111	-0.083333	-0.166667	0.138889	-0.222222	0.027778
std	0.958856	1.605936	1.424138	1.055576	1.306957	1.382882	1.512597	1.562220	1.601056	1.475677	...	1.517504	1.513373	1.507156	1.170517	1.570463	1.517504	1.414214	1.270528	1.235875	1.482026
min	-1.000000	-2.000000	-2.000000	-2.000000	-2.000000	-2.000000	-2.000000	-2.000000	-2.000000	-2.000000	...	-2.000000	-2.000000	-2.000000	-2.000000	-2.000000	-2.000000	-2.000000	-2.000000	-2.000000	-2.000000
25%	-1.000000	-2.000000	-0.500000	0.000000	-2.000000	-0.500000	-2.000000	-2.000000	-2.000000	-2.000000	...	-2.000000	-2.000000	0.000000	0.000000	-2.000000	-2.000000	-2.000000	0.000000	-0.500000	-2.000000
50%	-1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	1.000000	2.000000	0.500000	0.000000	0.000000	0.000000	0.000000	0.000000	2.000000	0.000000	...	2.000000	0.500000	2.000000	0.000000	2.000000	2.000000	0.000000	0.000000	0.000000	2.000000
max	1.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	...	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000

8 rows × 7071 columns

```
df.head(2)
```

	class	v1	v2	v3	v4	v5	v6	v7	v8	v9	...	v7061	v7062	v7063	v7064	v7065	v7066	v7067	v7068	v7069	v7070
0	-1	-2	2	0	0	-2	-2	0	0	0	...	0	-2	-2	0	0	0	0	2	0	0
1	-1	0	0	0	0	-2	0	0	0	0	...	0	2	0	0	0	0	0	0	-2	0

2 rows × 7071 columns

Figure 8: Classification of Data

6. SIMULATION ANALYSIS AND DISCUSSION:

We have done the experiment for the proposed method using Python 3.9 version and Jupiter Notebook. We used System with Intel i5, 2.5 GHz processor, 4GB RAM and Ubuntu Operating System. Different Classification algorithms are applied for the validation of performance of Folded Autoencoder technique. The result is lastly compared with classification accuracy of original features set. Tensorflow, Numpy Scikit-Learn are used in our proposed program. And for the evaluation purpose in this experiment, we divided the microarray datasets in two parts that are Training and Testing. 80:20 ratios, 70:30 ratios and 60:40 ratios in these three different ways, microarray datasets are divided for experimental purpose. Sensitivity, specificity and accuracy are checked for performance evaluation.

6.1 Result for Selected Datasets

The following describes the experimental findings for each microarray dataset. Utilizing several machine learning classifiers, we conducted three tests. The performance assessments of the complete microarray dataset and the truncated microarray dataset were then compared.

Dataset	Classifier	Test	Classification without Folded Autoencoder				Classification with Autoencoder			
			T_1	T_2	T_3	Avg.	T_1	T_2	T_3	Avg.
Colon	SVM	Sensitivity	74.54	74.26	73.94	74.25	91.68	90.87	90.24	90.93
		Specificity	72.31	71.98	71.66	71.98	88.86	88.31	87.73	88.30
		Accuracy	73.68	73.11	72.81	73.20	90.79	90.15	89.51	**90.15**
	KNN	Sensitivity	58.12	57.55	57.13	57.60	82.91	82.25	81.89	82.35
		Specificity	56.34	54.98	54.15	55.16	80.77	80.17	79.94	80.29
		Accuracy	57.36	56.97	56.54	56.96	81.81	81.09	80.68	81.19
	RF	Sensitivity	66.89	66.22	65.74	66.28	86.96	86.42	85.82	86.40
		Specificity	63.98	63.40	62.89	63.42	83.77	82.98	82.39	83.05

		Accuracy	65.29	64.77	64.09	64.71	85.62	85.07	84.63	85.11
CNS	**SVM**	**Sensitivity**	73.88	73.19	72.47	73.18	90.03	89.70	89.13	89.62
		Specificity	71.02	70.57	70.11	70.57	87.26	86.58	86.02	86.62
		Accuracy	72.22	71.77	71.21	71.73	89.66	89.07	88.71	**89.15**
	KNN	**Sensitivity**	56.21	55.68	54.97	55.62	75.11	74.65	73.98	74.58
		Specificity	53.11	52.66	52.23	52.67	72.34	71.69	71.03	71.69
		Accuracy	54.44	53.91	53.31	53.89	73.91	73.25	72.87	73.34
	RF	**Sensitivity**	65.33	64.44	63.71	64.49	84.55	83.87	83.36	83.93
		Specificity	62.25	61.76	60.96	61.66	82.11	81.72	81.28	81.70
		Accuracy	63.84	63.09	62.42	63.12	83.29	82.97	82.58	82.95
Ovarian	**SVM**	**Sensitivity**	87.63	86.88	86.12	86.88	96.49	95.52	95.03	95.68
		Specificity	85.37	84.81	84.26	84.81	93.92	93.28	92.41	93.20
		Accuracy	86.42	85.75	84.91	85.69	94.49	93.70	93.03	**93.74**
	KNN	**Sensitivity**	75.19	74.49	73.51	74.49	88.12	87.67	87.06	87.62
		Specificity	72.50	71.66	70.78	71.65	85.90	85.21	84.77	85.29
		Accuracy	73.94	72.68	72.57	73.06	87.11	86.54	86.08	86.58

	RF	**Sensitivity**	80.24	79.41	79.02	79.56	91.78	91.16	90.65	91.19
		Specificity	78.95	78.16	77.48	78.19	88.61	88.07	87.59	88.09
		Accuracy	79.84	78.97	77.83	78.88	90.93	90.26	89.47	90.22
Leukaemia	**SVM**	**Sensitivity**	85.92	85.05	84.24	85.07	94.37	93.64	93.03	93.68
		Specificity	83.31	82.71	82.01	82.68	91.58	90.73	90.11	90.81
		Accuracy	84.77	84.12	83.77	84.22	93.54	93.62	92.83	**93.33**
	KNN	**Sensitivity**	78.81	78.06	77.31	78.06	87.55	86.67	86.08	86.77
		Specificity	76.53	75.77	74.93	75.74	84.98	84.29	83.81	84.36
		Accuracy	77.72	77.13	76.65	77.17	85.89	85.24	84.72	85.28
	RF	**Sensitivity**	84.44	81.87	81.34	81.88	90.57	90.09	89.42	90.03
		Specificity	79.99	79.22	78.49	79.23	87.78	87.17	86.64	87.19
		Accuracy	81.39	80.68	80.06	80.71	89.67	89.11	88.52	89.10
Prostate	**SVM**	**Sensitivity**	65.53	65.05	64.61	65.06	86.68	86.17	85.52	86.12
		Specificity	62.99	62.23	61.78	62.33	82.91	82.33	81.82	82.35
		Accuracy	64.83	64.30	63.77	64.30	84.88	84.16	83.71	**84.25**
		Sensitivity	54.18	53.71	53.09	53.66	78.93	78.37	77.72	78.34

	KNN	**Specificity**	51.16	50.55	50.12	50.61	75.84	75.30	74.77	75.30
		Accuracy	52.71	52.10	51.63	52.15	76.99	76.48	75.91	76.46
	RF	**Sensitivity**	59.21	58.54	57.06	58.27	82.31	81.83	81.33	81.82
		Specificity	57.49	56.78	56.19	56.82	79.77	79.12	78.80	79.23
		Accuracy	58.34	57.63	57.03	57.67	80.97	80.25	79.61	80.28

Where T_1 = $Test_1$ (80:20)
T_2 = $Test_2$ (70:30)
T_3 = $Test_3$ (60:40)
Avg. = Average
RF = Random Forest

7. CONCLUSION

The prevalence of difficult diseases, the importance of generating microarray data from various tissues, and the analysis of this data have all grown in recent years. The microarray data, however, is difficult to categorise due to the large number of characteristics and sparse sample size. In this research, we provide a method called Folded Autoencoder-SVM (Code named PR v1.0), which is based on the region of the order of Folded Autoencoder and an SVM classifier (FAS). Folded Autoencoder produces a subset of adding minimal features while taking into account pertinent data. Utilizing three rotating robot learning classifiers, we evaluated the effectiveness of a freshly constructed dataset (SVM, KNN and Random Forest). The SVM classifier provides enhanced results after creating a new feature set. This result has been contrasted with classifier accuracy beyond sum dataset. FAS is competent at recognising features whose exposure to the atmosphere is erratic and whose refreshing is claimed. As a result, FAS demonstrates its effectiveness in choosing and categorising elements as soon as unstoppable exactness results. For microarray data from the colon, central nervous system, ovary, leukaemia, and prostate, the suggested FAS technique offers improved classification accuracy. As a result, we can say that FAS provides increased exactness when considering the newly created feature subset.

REFERENCES

[1] M. Ghosh, S. Begum, R. Sarkar, D. Chakraborty, U. Maulik, Recursive memetic algorithm for gene selection in microarray data, Expert Systems with Applications, vol. 116, pp. 172-185, 2019.

[2] W. H. Chan, M. S. Mohamad, S. Deris, N. Zaki, S. Kasim, S. Omatu, J. M. Corchado, H. Al Ashwal, Identification of informative genes and pathways using an improved penalized

support vector machine with a weighting scheme, Computers in biology and medicine, vol. 77, pp. 102-115, 2016.
[3] N. Pochet, F. De Smet, J. A. Suykens, B. L. De Moor, Systematic benchmarking of microarray data classification: assessing the role of non-linearity and dimensionality reduction, Bioinformatics 20, vol. 17, pp. 3185-3195, 2004.
[4] S. Ramaswamy, P. Tamayo, R. Rifkin, S. Mukherjee, C.-H. Yeang, M. Angelo, C. Ladd, M. Reich, E. Latulippe, J. P. Mesirov, et al., Multiclass cancer diagnosis using tumor gene expression signatures, Proceedings of the National Academy of Sciences 98, vol. 26, pp. 15149-15154, 2001.
[5] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., Tensorflow: A system for largescale machine learning, in: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI), vol. 16, pp. 265-283, 2016.
[6] J. Lv, Q. Peng, X. Chen, Z. Sun, A multi-objective heuristic algorithm for gene expression microarray data classification, Expert Systems with Applications, vol. 59, pp. 13-19, 2016.
[7] N. S. Mohamed, S. Zainudin, Z. A. Othman, Metaheuristic approach for an enhanced mrmr filter method for classification using drug response microarray data, Expert Systems with Applications, vol. 90, pp. 224-231, 2017.
[8] J. T. De Souza, A. C. De Francisco, D. C. De Macedo, Dimensionality reduction in gene expression data sets, IEEE Access, vol. 7, pp. 61136-61144, 2019.
[9] P. K. Mallick, S. H. Ryu, S. K. Satapathy, S. Mishra, G. N. Nguyen, P. Tiwari, Brain mri image classification for cancer detection using deep wavelet autoencoder-based deep neural network, IEEE Access, vol. 7, pp. 46278-46287, 2019.
[10] S. Kilicarslan, K. Adem, M. Celik, Diagnosis and classification of cancer using hybrid model based on relieff and convolutional neural network, Medical Hypotheses, vol. 137, pp. 109577, 2020.
[11] J. Xu, P. Wu, Y. Chen, Q. Meng, H. Dawood, M. M. Khan, A novel deep flexible neural forest model for classification of cancer subtypes based on gene expression data, IEEE Access, vol. 7, pp. 22086-22095, 2019.
[12] D. Zhang, L. Zou, X. Zhou, F. He, Integrating feature selection and feature extraction methods with deep learning to predict clinical outcome of breast cancer, IEEE Access, vol. 6, pp. 28936-28944, 2018.
[13] D. Q. Zeebaree, H. Haron, A. M. Abdulazeez, Gene selection and classification of microarray data using convolutional neural network, in: 2018 International Conference on Advanced Science and Engineering (ICOASE), IEEE, pp. 145-150, 2018.
[14] P. K. Ram, P. Kuila, Feature selection from microarray data: Genetic algorithm based approach, Journal of Information and Optimization Sciences 40, vol. 8, pp. 1599-1610, 2019.
[15] N. Almugren, H. Alshamlan, A survey on hybrid feature selection methods in microarray gene expression data for cancer classification, IEEE Access, vol. 7, pp. 78533-78548, 2019.
[16] E. Bonilla-Huerta, A. Hernandez-Montiel, R. Morales-Caporal, M. Arjona-Lopez, Hybrid framework using multiple-filters and an embedded approach for an efficient selection and classification of microarray data, IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB) 13, vol. 1, pp. 12-26, 2016.

[17] D. Chakraborty, U. Maulik, Identifying cancer biomarkers from microarray data using feature selection and semisupervised learning, IEEE journal of translational engineering in health and medicine, vol. 2, pp. 1-11, 2014.
[18] W. Ke, C. Wu, Y. Wu, N. N. Xiong, A new filter feature selection based on criteria fusion for gene microarray data, IEEE Access, vol. 6, pp. 61065-61076, 2018.
[19] J. Wang, H. He, D. V. Prokhorov, A folded neural network autoencoder for dimensionality reduction, Procedia Computer Science, vol. 13, pp. 120-127, 2012.
[20] Y. Wang, I. V. Tetko, M. A. Hall, E. Frank, A. Facius, K. F. Mayer, H. W. Mewes, Gene selection from microarray data for cancer classification - a machine learning approach, Computational biology and chemistry 29, vol. 1, pp. 37-46, 2005.
[21]Understand data normalization in machine learning - towards data science, https://towardsdatascience.com/understand-data-normalizationin-machine-learning-8ff3062101f0, Accessed on: Jan. 27, 2020.