Evolutionary Flexible Neural Tree: Highly Explainable Deep Learning Architecture

Yang Yan
yangyan@sas.upenn.edu

Yide Zhao
yidezhao@seas.upenn.edu

University of Pennsylvania

May 2020

Contents

1 Introduction **2**

2 Model Setup **2**
- 2.1 Tree Structure . . . 2
- 2.2 Population Initialization . . . 3
- 2.3 Genetic Algorithm . . . 3
  - 2.3.1 Selection . . . 3
  - 2.3.2 Crossover . . . 3
  - 2.3.3 Mutation . . . 3
  - 2.3.4 Reproduction . . . 4
- 2.4 Performance Measurement . . . 4

3 Results on Synthetic Dataset **4**
- 3.1 Data . . . 4
- 3.2 Training . . . 5
- 3.3 Results . . . 5

4 Case Study and Performance Comparison **6**
- 4.1 Overview . . . 6
- 4.2 Evolutionary Flexible Neural Tree . . . 6
- 4.3 Feed-Forward Neural Network . . . 7
- 4.4 Decision Tree . . . 8

5 Conclusion **8**

6 Future Direction **8**

7 Code **9**

8 Acknowledgement **9**

A Overview of Wine Quality Dataset **11**

B Sample Program Output **11**

1 Introduction

Deep learning[1] has succeed in many domains in past decades such as image recognition[2], natural language processing[3], and prediction of drug activity[4]. Convolutional neural network (CNN) has done a great job in image recognition and classification, while recurrent neural network (RNN) is specialized in processing sequential inputs like speech or language. However, most of the deep neural network seem to be a black-box and fall short in explainability. On the other hand, decision tree model like random forest[5] are widely practiced by researches for its high explainability and accuracy. Aiming to add more explainability to the deep learning model, researchers have proposed a combination of both approaches, such as deep neural decision tree[6], deep neural decision forests[7], and flexible neural tree[8], [9]. The flexible neural tree model has shown to be successful in traffic identification[10] and financial forecasting[11], [12]. In this project we propose the evolutionary flexible neural tree by adopting the flexible neural tree model and extending it with an evolutionary idea in training such as selection, crossover, mutation and reproduction. We train and evaluate the evolutionary flexible neural tree model by first using a synthetic dataset and then the UCI wine dataset[13]. We also compare the performance of the evolutionary flexible neural tree with decision tree and feed-forward neural network.

The rest of the paper is distributed as follows: section 2 introduces the theoretical design of the evolutionary flexible neural tree model with its structure, initialization, genetic algorithm and performance measurement; section 3 shows the result on the synthetic dataset; section 4 comparatively studies the result of the evolutionary flexible neural tree, decision tree, and the feed-forward neural network on the UCI wine dataset; section 5 concludes the study with discussion and section 6 lays out future direction of the study.

2 Model Setup

2.1 Tree Structure

A sample neural tree structure is shown in Fig.1:

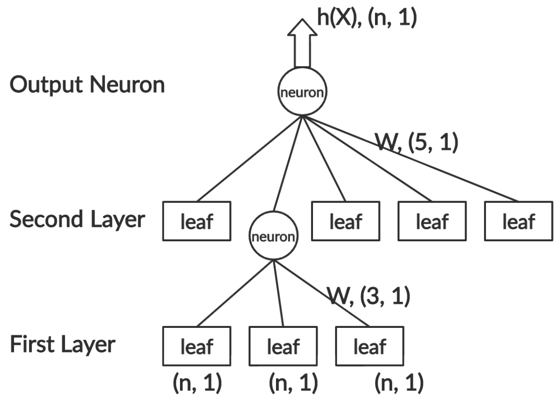


Figure 1: Typical Neural Tree Structure

Each leaf node represents one particular feature of input $\boldsymbol{X}$. Each neuron node is like a neuron in the conventional neural network. Features that are more important in classifying the output will appear in upper layer closer to the output neuron. We use a Gaussian activation function $f(x, W) = exp(-\frac{(x \cdot W - a)^2}{b^2})$, where W, a, b are all trainable parameters. They will be trained by a user-designated optimizer using back-propagation.

2.2 Population Initialization

The program will randomly generate a set number of initial flexible neural trees given parameters of maximum tree depth, percentage of neuron nodes and optimizer properties. Immediately after initialization, every tree's W, a, b is trained individually using the assigned optimizer and will be ready for the evolution process.

2.3 Genetic Algorithm

2.3.1 Selection

Each tree are scored individually based on a scoring function defined as follows:

$$f(y, \hat{y}, n_{leaf}, n_{neuron}) = \frac{\mathbb{1}(y = \hat{y})}{N_y} - \alpha(max\{n_{leaf} - t_l, 0\})^2 - \beta(max\{n_{neuron} - t_n, 0\})^2 \quad (1)$$

This function is mainly based on accuracy calculated in the first term. The second and third term are regularization terms that penalize tree complexity, but only when the number of leaves/neuron exceeds a user-defined threshold.

After calculating the score, a certain percentage of worst-performing trees will be deleted from the population. The specific percentage is a user-defined hyper-parameter. The remaining neural trees will enter the next step of the evolution.

2.3.2 Crossover

Crossover is the process where a tree's neuron exchanges place with another tree's neuron. It can potentially take a useful part of a sub-tree and transplant it to an another tree so that more useful information are inherited and passed on to future generation. This process mimics the chromosomal crossover in real life shown in Fig.2, and is thought to be one of the most important evolutionary drivers[14]. In this algorithm, a tree can crossover with others at a user-defined probability.

2.3.3 Mutation

Mutation is extremely important in evolution. It is the ultimate source of all genetic variation[16]. In our algorithm, the following four kinds of mutations are used.

1. Change one leaf. One of the leaf nodes is changed. Biologically it is equivalent to substitution mutations.

2. Change all leaf. All of the leaf nodes are changed. This operation does not have a biological counterpart, but is an integral part of genetic algorithm[17].

3. Pruning. Take one neuron of a tree, and turn it into a leaf node. This reduces the complexity of the tree and could potentially lose information. Biologically it corresponds to deletions of large chromosomal regions.

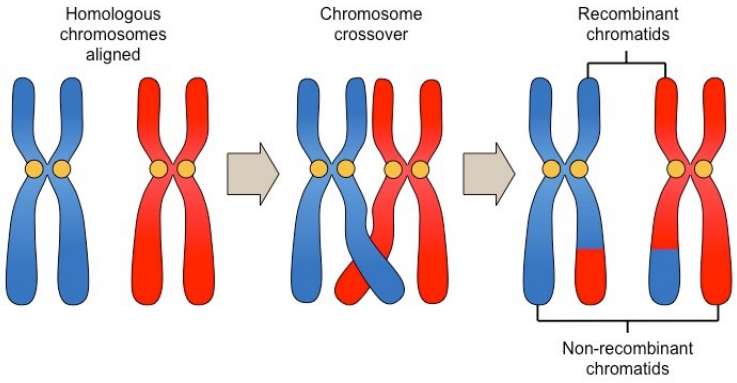


Figure 2: Simple illustration of chromosomal crossover[15]

4. Growing. Take one leaf node of a tree, and turn it into a neuron. This can gain information at the expense of higher model complexity. Biologically it is similar to amplifications.

At present, all four variants happen at the same probability. Future research or tuning could potentially change this distribution, even adaptively (*e.g.* if the tree is complex, it is more likely to prune).

2.3.4 Reproduction

Reproduction is introduced to keep the population relatively constant over time. In this particular use case, reproducing simply means copying a tree and then mutating it.

2.4 Performance Measurement

After evolution simulations, the best performing tree (based on the previously mentioned scoring function) is selected. Then, it is evaluated using test dataset based on accuracy and confusion matrix. The structure of the tree is also manually inspected to evaluate its ability to capture important features.

3 Results on Synthetic Dataset

3.1 Data

Instances are uniformly drawn from feature space $\mathcal{X}^8 = [0.0, 1.0)^8$. The label is generated based on the following process:

$$f(\boldsymbol{x}) = \sigma(x_1^2 + x_2x_4 + sin(x_3) + \sqrt{3x_5} - 5x_7)$$
$$\boldsymbol{y} = \mathbb{1}(f(\boldsymbol{x}) > \frac{1}{2})$$

where x_n denotes n-th feature of $\boldsymbol{x}$. In this way, the positive and negative labels are roughly of the same size. It is also worth noting that x_3, x_5, x_7 are particularly important features while x_0, x_6 are not included at all.

3.2 Training

The population is initialized with 10 individual, and the evolution is set for 20 generations. Other hyper-parameters are tuned and are set to be the same throughout the paper. One thing to note is that selection percentile should be set to roughly the same as the reproduction probability to ensure stable population size.

Tree Parameters	
max depth	3
neuron ratio	0.3
optimizer	Adam
learning rate	1e-2
num of iterations	1,200
Score Function	
α	0.003
t_l	number of features
β	0.001
t_n	3 - 5
Evolution Parameters	
Selection Percentile	40%
Crossover Proba.	0.6
Mutation Proba.	0.4
Reproduction Proba.	0.4

Table 1: Hyper-parameters of the Model

3.3 Results

The accuracy is between 93% - 96% after 10 generations. The following graph summarizes the result:

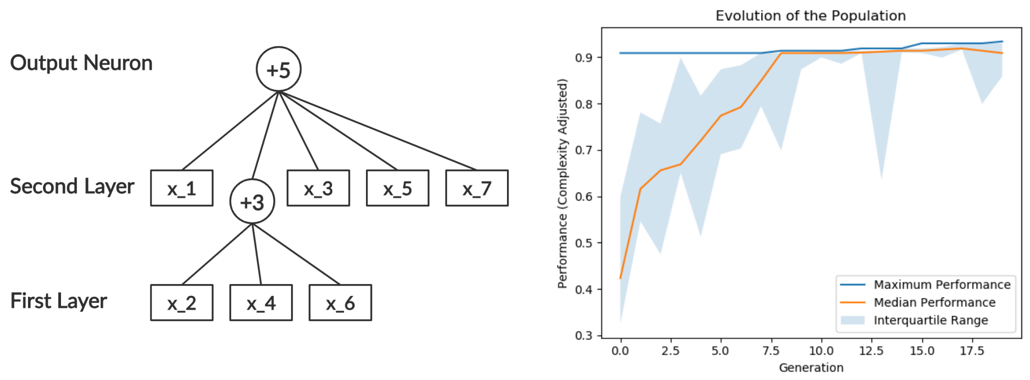


Figure 3: Tree Structure and Evolution Graph from Synthetic Dataset

From the tree structure, we can see that this tree successfully captures the most important features, namely x_1, x_3, x_5, x_7; and it is successful in identifying a relationship between x_2 and x_4. It can also filter out unimportant features, which, in this case, is x_0 (it did not filter x_6 though). This particular tree has an accuracy of 96%, and the confusion matrix is $\begin{bmatrix} 104 & 1 \\ 7 & 88 \end{bmatrix}$.

The evolution graph on the right shows that the population is constantly improving before Generation 8, but the highest-performing tree is not changing. However, as the majority of species achieved a good structure (in terms of IQR), crossover and mutations have a better chance in creating a better structure. Indeed, there is an evolution in max-performing tree in Generation 8, 11, 13 and 18.

4 Case Study and Performance Comparison

4.1 Overview

We use the UCI wine dataset[13] (see Appendix A) to do a comparative study on evolutionary flexible neural tree, decision tree, and feed-forward neural network. The quality score ranging from [3,4,5,6,7,8] is treated as label where quality score greater than or equal to 6 is treated as 1 (high quality) and other as 0 (low quality). As explained in the introduction, the evolutionary flexible neural tree can be viewed as a combination of both decision tree and neural network where it maintains the deep learning architecture while adding explainability. Therefore, we explicitly do a comparison in this section. With some randomness, the final accuracy for classifying the wine data are around 73% for evolutionary flexible neural tree, around 72% for decision tree, and around 70% for feed-forward neural network.

4.2 Evolutionary Flexible Neural Tree

The initial population size is 100, and the evolution is set for 25 generations. All hyper-parameters are the same as in Table 1. The final accuracy is between 71% to 75%, on par with decision tree. One representative structure is listed below. This particular tree has accuracy of 71.56%, and the confusion matrix is $\begin{bmatrix} 119 & 34 \\ 57 & 110 \end{bmatrix}$.

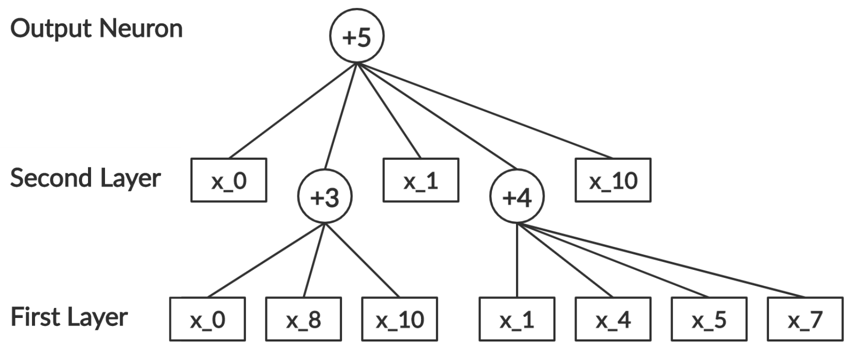


Figure 4: Tree Structure from Wine Dataset

In addition, we calculated the feature importance using the state-of-the-art XGBoost algorithm, shown in Fig.5. Comparing the tree structure with the result from XGBoost, we see an almost identical ranking of feature importance. For example, x_1, x_0, x_{10} are the most imprtant features selected by the XGBoost algorithm. Similarly, x_1, x_0, x_{10} are *exactly* the features closest to the output neuron, which represents the highest importance. Features that ranks lowest in XGBoost, like x_3 and x_6, are not even included in the neural tree.

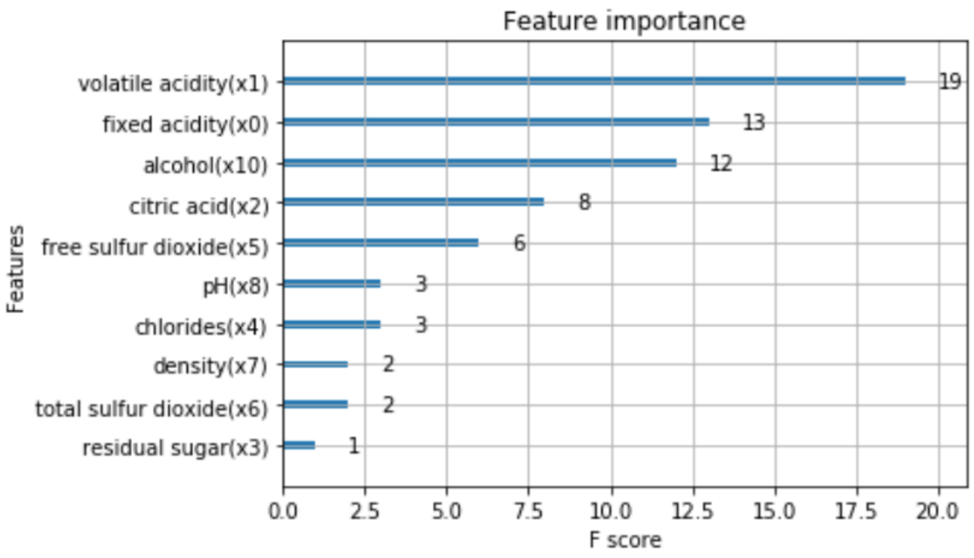
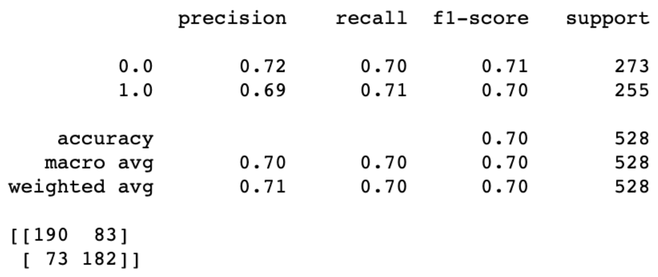


Figure 5: Feature Importance from XGBoost Algorithm

4.3 Feed-Forward Neural Network

For this feed-forward neural network, we has two hidden layer each with size of 20 and the Relu activation function at each layer. We train the neural network for 10000 epochs with learning rate 0.001. The following Fig.6 shows the accuracy result with the confusion matrix. Here, we get an accuracy around 70%.

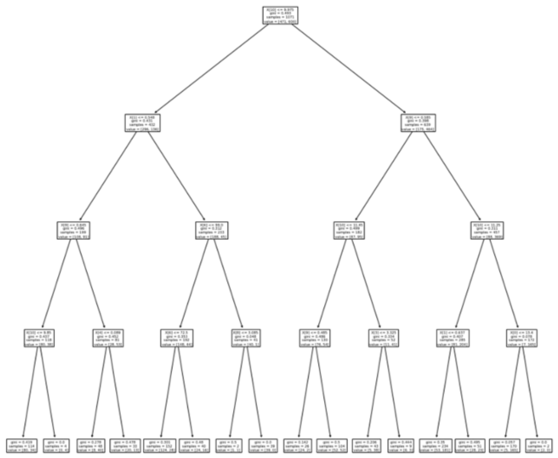


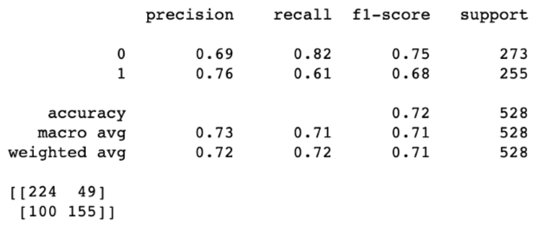
```
              precision    recall  f1-score   support

         0.0       0.72      0.70      0.71       273
         1.0       0.69      0.71      0.70       255

    accuracy                           0.70       528
   macro avg       0.70      0.70      0.70       528
weighted avg       0.71      0.70      0.70       528

[[190  83]
 [ 73 182]]
```

Figure 6: Performance of Feed-Forward Neural Network

4.4 Decision Tree

For the decision tree, we choose the max depth to be 4 after hyper-parameter tuning. In Fig.7, the left shows the structure of the decision tree and the right shows the accuracy result with the confusion matrix. Here, we get an accuracy around 72%.





```
              precision    recall  f1-score   support

           0       0.69      0.82      0.75       273
           1       0.76      0.61      0.68       255

    accuracy                           0.72       528
   macro avg       0.73      0.71      0.71       528
weighted avg       0.72      0.72      0.71       528

[[224  49]
 [100 155]]
```

Figure 7: Structure and Performance of Decision Tree

5 Conclusion

In this study, we proposed an evolutionary flexible neural tree that takes advantage of the deep learning architecture in neural network and the explainability in decision tree. The neural tree has a similar layer structure as the CNN in the way that the most important features are closer to the output neuron. Also, the model utilizes ideas in evolution like selection, crossover, mutation, and reproduction in training. Such genetic algorithm "naturally evolves and selects" the best model with the highest score. From the result of the synthetic dataset, we can see the median accuracy of the evolutionary flexible neural tree is always above 90% after 10 generations. From the real wine dataset, we can see the evolutionary flexible neural tree with accuracy 73%, on par with the decision tree and outperforming the feed-forward neural network. More importantly, the evolutionary flexible neural tree is able to extract the most important features from the data and push them to layers closer to the output. These extracted features are exactly the features chosen by XGBoost algorithm. All in all, the proposed evolutionary flexible neural tree in this study is a successful deep learning architecture that is highly explainable.

6 Future Direction

First, an auto "slimming" algorithm could be implemented. For example, it often happens that a neuron has two same nodes as input. It is clearly inefficient and could be easily fixed by conducting a sanity check after every tree generation / modification.

Second, the reproduction algorithm could be further optimized. In biology, a stronger individual has a higher chance of mating and thus its gene has a better chance of passing on to the next generation. Genetic Algorithm could also simulate this process by dynamically adjusting the reproduction probability based on individual performance. Also, reproduction needs to happen between two individual (instead of one in this paper). For example, take a *copy* of a tree and change one branch to a sub-tree of another tree. Note its difference with crossover: the structures of parent trees remain unchanged.

Finally, extension to multiclass classification could be implemented. Either one-vs-rest or one-vs-one strategy is a quick and easy extension of current program, and they can reveal which feature contributes most to each class. However, given the computational complexity of the evolutional training, both would be time-consuming. If efficiency is more important, multiclass perceptrons should be considered with a softmax activation function. As a trade-off, per-class feature importance is less clear.

7 Code

See GitHub: https://github.com/yanyang0063/Evolutionary-Flexible-Neural-Tree.

8 Acknowledgement

This paper is the final project for ENM 531 Data-driven Modelling at the University of Pennsylvania. We would like to thank Professor Paris Perdikaris for his well-taught lectures throughout the semester. Also, we would like to thank the helpful TAs, Yibo Yang and Taruna Kar, for their support along the way.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] U. Rafi, B. Leibe, J. Gall, and I. Kostrikov, "An efficient convolutional network for human pose estimation," *Procedings of the British Machine Vision Conference 2016*, 2016. DOI: `10.5244/c.30.109`.

[3] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[4] J. Ma, R. P. Sheridan, A. Liaw, G. E. Dahl, and V. Svetnik, "Deep neural nets as a method for quantitative structure–activity relationships," *Journal of Chemical Information and Modeling*, vol. 55, no. 2, pp. 263–274, 2015, PMID: 25635324. DOI: `10.1021/ci500747n`. [Online]. Available: `https://doi.org/10.1021/ci500747n`.

[5] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[6] Y. Yang, I. G. Morillo, and T. M. Hospedales, "Deep neural decision trees," *arXiv preprint arXiv:1806.06988*, 2018.

[7] P. Kontschieder, M. Fiterau, A. Criminisi, and S. Rota Bulo, "Deep neural decision forests," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1467–1475.

[8] Y. Chen, A. Abraham, and B. Yang, "Feature selection and classification using flexible neural tree," *Neurocomputing*, vol. 70, no. 1-3, pp. 305–313, 2006.

[9] X. Lei and Y. Chen, "Multiclass classification of microarray data samples with flexible neural tree," in *2012 Spring Congress on Engineering and Technology*, IEEE, 2012, pp. 1–4.

[10] L. Peng, H. Zhang, B. Yang, Y. Chen, M. T. Qassrawi, and G. Lu, "Traffic identification using flexible neural trees," in *2010 IEEE 18th International Workshop on Quality of Service (IWQoS)*, IEEE, 2010, pp. 1–5.

[11] Y. Chen, B. Yang, and A. Abraham, "Flexible neural trees ensemble for stock index modeling," *Neurocomputing*, vol. 70, no. 4-6, pp. 697–703, 2007.

[12] Y. Chen, L. Peng, and A. Abraham, "Exchange rate forecasting using flexible neural trees," in *International symposium on neural networks*, Springer, 2006, pp. 518–523.

[13] *Wine quality data set*. [Online]. Available: `https://archive.ics.uci.edu/ml/datasets/Wine%20Quality`.

[14] G. Coop and M. Przeworski, "An evolutionary view of human recombination," *Nature Reviews Genetics*, vol. 8, no. 1, pp. 23–34, May 2006. DOI: `10.1038/nrg1947`.

[15] *Crossing over*. [Online]. Available: `https://ib.bioninja.com.au/standard-level/topic-3-genetics/33-meiosis/crossing-over.html`.

[16] T. A. P. Society, *Mutation*. [Online]. Available: `https://www.apsnet.org/edcenter/disimpactmngmnt/topc/PopGenetics/Pages/Mutation.aspx`.

[17] L. M. Schmitt, "Theory of genetic algorithms," *Theoretical Computer Science*, vol. 259, no. 1, pp. 1–61, 2001, ISSN: 0304-3975. DOI: `https://doi.org/10.1016/S0304-3975(00)00406-0`. [Online]. Available: `http://www.sciencedirect.com/science/article/pii/S0304397500004060`.

Appendices

A Overview of Wine Quality Dataset

Fxd. acidity x_0	Vol. acidity x_1	citric acid x_2	resid. sugar x_3	Cl^- x_4	free SO_2 x_5	total SO_2 x_6	density x_7	pH x_8	SO_4^{2-} x_9	abv x_{10}	Quality *label*
7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	0
7.8	0.88	0	2.6	0.098	25	67	0.9968	3.2	0.68	9.8	0
7.8	0.76	0.04	2.3	0.092	15	54	0.997	3.26	0.65	9.8	0
11.2	0.28	0.56	1.9	0.075	17	60	0.998	3.16	0.58	9.8	1
7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	0
7.4	0.66	0	1.8	0.075	13	40	0.9978	3.51	0.56	9.4	0
7.9	0.6	0.06	1.6	0.069	15	59	0.9964	3.3	0.46	9.4	0
7.3	0.65	0	1.2	0.065	15	21	0.9946	3.39	0.47	10	1
7.8	0.58	0.02	2	0.073	9	18	0.9968	3.36	0.57	9.5	1
7.5	0.5	0.36	6.1	0.071	17	102	0.9978	3.35	0.8	10.5	0

B Sample Program Output

```
Initializing population...
100%|                              | 20/20 [00:14<00:00,  1.36it/s]
Finished initialization, starting evolution...
-------------------GEN 1-------------------
Mean metric is -0.4303
Best metric is 0.8388
Crossover...
Mutating...
Reproducing...
Population: 20
-------------------GEN 2-------------------
Mean metric is 0.4331
Best metric is 0.8988
Crossover...
Mutating...
Reproducing...
Population: 18
-------------------GEN 3-------------------
Mean metric is 0.5484
Best metric is 0.8988
Crossover...
Mutating...
Reproducing...
Population: 14
-------------------GEN 4-------------------
Mean metric is 0.7311
Best metric is 0.9212
```

```
Crossover...
Mutating...
Reproducing...
Population: 11
--------------------GEN 5-------------------
Mean metric is 0.8510
Best metric is 0.9212
Crossover...
Mutating...
Reproducing...
Population: 11
--------------------GEN 6-------------------
Mean metric is 0.8703
Best metric is 0.9212
Crossover...
Mutating...
Reproducing...
Population: 9
--------------------GEN 7-------------------
Mean metric is 0.9106
Best metric is 0.9212
Crossover...
Mutating...
Reproducing...
Population: 7
Population too small, reproducing more...
--------------------GEN 8-------------------
Mean metric is 0.9018
Best metric is 0.9212
Crossover...
Mutating...
Reproducing...
Population: 12
--------------------GEN 9-------------------
Mean metric is 0.9217
Best metric is 0.9262
Crossover...
Mutating...
Reproducing...
Population: 20
--------------------GEN 10-------------------
Mean metric is 0.9113
Best metric is 0.9262
Crossover...
Mutating...
Reproducing...
Population: 28
--------------------GEN 11-------------------
```

```
Mean metric is 0.8713
Best metric is 0.9262
Crossover...
Mutating...
Reproducing...
Population: 29
-------------------GEN 12-------------------
Mean metric is 0.8864
Best metric is 0.9337
Crossover...
Mutating...
Reproducing...
Population: 34
-------------------GEN 13-------------------
Mean metric is 0.8784
Best metric is 0.9337
Crossover...
Mutating...
Reproducing...
Population: 38
-------------------GEN 14-------------------
Mean metric is 0.8981
Best metric is 0.9337
Crossover...
Mutating...
Reproducing...
Population: 46
-------------------GEN 15-------------------
Mean metric is 0.8654
Best metric is 0.9337
Crossover...
Mutating...
Reproducing...
Population: 55
-------------------RESULT-------------------
Best Tree Structure is
FN+4;[FN+3;[7,5,2, depth:2]
,3,5,7, depth:3]

Best Accuracy is 0.9350
Confusion Matrix:
 [[111   6]
 [  7  76]]
```