Adult Income Prediction based on Neural Network

Shengli Zhou (12212232), CSE, SUSTech

***Abstract*—In this paper, we provide a method for predicting whether an adult's annual income exceeds 50 thousand dollars based on the Adult Census Income dataset. We implement our model using Neural Network while also comparing its performance with Support Vector Machine, Decision Tree and Naive Bayes.**

***Index Terms*—Artificial Intelligence, Machine Learning, Deep Learning, Neural Network, Adult Census Income.**

I. Introduction

ADULT Income Prediction is a classic problem in machine learning. The problem is based on the Adult Census Income dataset, which is extracted from the 1994 Census bureau database by Ronny Kohavi and Barry Becker (Data Mining and Visualization, Silicon Graphics). [1] In the problem, the model is given a series of personal information about a group of people and needs to predict whether each individual is likely to earn more than 50 thousand dollars annually.

In this project, we propose a method for predicting adult's annual income based on Neural Network. We also compare the performance on this task between different models including Neural Network (NN), Support Vector Machine (SVM), Decision Tree (DT) and Naive Bayes (NB) with different hyperparameters.

II. Preliminary

In the task of Adult Income Prediction, we train all our models in the Adult Census Income dataset, which can be downloaded on kaggle.

The dataset includes a table with each row representing the information of an individual and each column representing an attribute. The attributes provided in the dataset include:

- age: A numerical value representing the age of each data sample.
- workclass: A string representing the type of work (e.g., private, government, self-employed, etc.) of the individual.
- education: A string representing the level of education of each sample.
- education_num: A number representing the schooling year of each sample, which has one-to-one correspondence to "education".
- marital_status: The marital status of each sample in strings.
- occupation: The occupation of each sample in strings.
- relationship: The family relationship of each sample in strings.
- race: The race of each data sample.
- gender: The sex of each sample, i.e., Male or Female.
- capital_gain: The capital gain is the profit resulting from the sale of a capital asset, such as stocks, bonds, or real estate, where the amount realized from the sale exceeds the purchase price.
- capital_loss: The capital loss is the financial deficit that arises from the sale of a capital asset at a lower price than its purchase price. This results in a negative return on investment for the holder.
- hours_per_week: An integer representing the approximate number of working ours of each data sample.
- native_country: The country where the individual is from.
- fnlwgt: fnlwgt is short for final weight. The final weight in the context of the Current Population Survey (CPS) refers to the adjusted figures assigned to each survey participant's data, ensuring that the aggregated data aligns with independent population estimates for various demographic categories, such as age, sex, race, and Hispanic origin, across the US civilian non-institutional population, through a process called 'raking', which iteratively balances these controls for enhanced representativeness.
- income: a categorical data representing whether the annual income of the data sample exceeds 50 thousand dollars.

In the task of Adult Income Prediction, the model is required to predict the label for "income" with the other attributes provided in the dataset.

III. Methodology

A. General Workflow

The framework of our model is illustrated in Fig. 1 below.

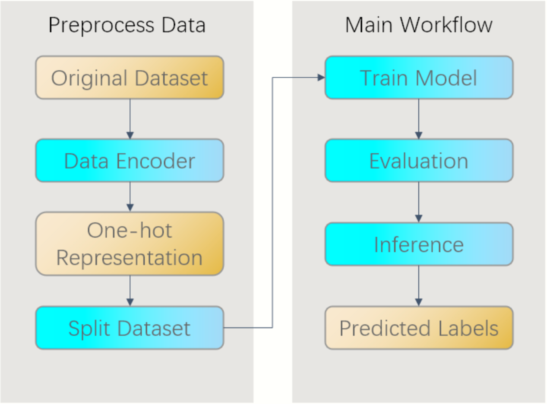

Fig. 1. General Workflow.

This project was supported by Department of Computer Science and Engineering, Southern University of Science and Technology (SUSTech). The project was posted in course CS311 Artificial Intelligence (H) as the third course project on May 10, 2024. All works of the project was finished on June 7, 2024.

S. Zhou is a sophomore student in Southern University of Science and Technology (SUSTech), Shenzhen, Guangdong Province, China. (e-mail: zhousl2022@mail.sustech.edu.cn).

In the workflow, we first build a data frame for the dataset and encode the data into one-hot representation w.r.t. each field. Then, we divide the dataset into training set, validation set and test set. Finally, we train and evaluate our model on the dataset.

B. Data Preprocessing

At the beginning of our model, we load the dataset from a .csv file with csv.DictReader and trim the white spaces to get the actual personal information for each data sample. Here we also transform all numeric data from strings to integers. Then we acquire the one-hot-encoded data by pandas.get_dummies(pandas.DataFrame(data)). Finally, we split our dataset into the training set, validation set and test set with a size ratio of approximately $8:1:1$.

C. Model Analysis

In this section, we will analyze some machine learning models that can perform binary classification on non-linearly-separable data and provide our reason for choosing models theoretically. We will also compare these models by experiments in Sec. IV.

1) Neural Network: In this paper, we build the Neural Network under the framework illustrated in Fig. 2. In the model, each block has a dense layer (i,e., fully-connected layer), an activation layer and a dropout layer. Here we use ReLU function $f(x) = \max(0, x)$ as the activation function for hidden layers and Sigmoid function $g(x) = \dfrac{1}{1+e^{-x}}$ as the activation function for output layer. By adding these activation functions, the model can have better performance when handling non-linearly-separable data. Moreover, we also add a dropout layer with 50% dropout ratio for each hidden layer to avoid over-fitting.

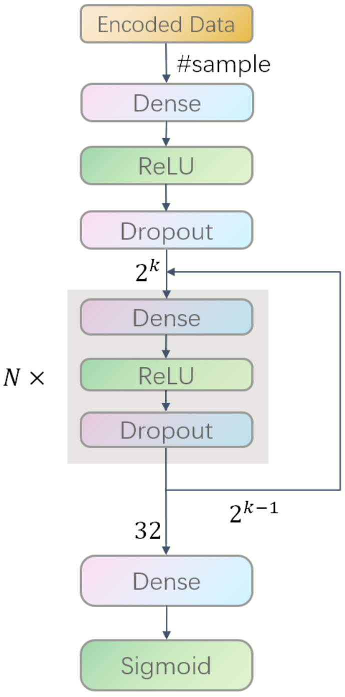

Fig. 2. Neural Network Framework.

2) Support Vector Machine (SVM): Here we use SVM with Gaussian Kernel (i.e., radial basis function network)

$$k(\boldsymbol{x}, \widetilde{\boldsymbol{x}}) = \exp\left(-\frac{||\boldsymbol{x} - \widetilde{\boldsymbol{x}}||_2^2}{2\sigma^2}\right)$$

According to Taylor series,

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$$

the Gaussian kernel can expand the dimension of the input data to infinity, making the model possible to handle non-linearly-separable data.

3) Decision Tree: Here we use sklearn.tree. DecisionTreeClassifier in python to construct a decision tree. We also set the hyperparameter ccp_alpha to perform post-pruning, which will add a penalty to the decision tree proportional to the complexity of the model.

4) Naive Bayes: We also use sklearn.naive_bayes. GaussianNB to build a model based on Naive Bayes. We also set the var_smoothing parameter to avoid the occurrence of zero probability event. However, Naive Bayes is not expected to achieve a performance as good as the models mentioned before, since some attributes in the dataset are correlated (e.g. education and education_num, capital_gain and capital_loss), which breaks the assumption of Naive Bayes.

5) Other Models: Note that there are some other common models that also satisfy the basic requirement of being able to handle non-linearly-separable data, e.g., K-nearest neighbors and logistic regression. However, these methods are generally based on the distance between different data points, which require the model to transform categorical data into numerical labels, making the distance between data points meaningless. Thus, we do not consider these models here.

D. Model Evaluation

After training the models, we save the parameters of each model and use them to predict the answer to each instance given in the validation set or test set. Finally, we can use metrics including accuracy, precision, sensitivity and specificity to evaluate the performance of models.

IV. Experiments

During this part, we conduct an experiment to compare the performance of each model mentioned in Sec. III and also evaluate the effect of hyperparameters on the models.

A. Experimental Setup

1) Environment: The code for performing the experiments is written in Python 3.8.5 and can be executed under an environment with keras==2.13.1, numpy==1.24.1, pandas==2.0.3, scikit-learn==1.3.2, scipy==1.10.1 and tensorflow. All the models are trained and tested on the dataset downloaded from kaggle and the details of the dataset can be seen in Sec. II.

For more details, please refer to the README file.

TABLE I
RESULT OF EXPERIMENTS

Model	Hyperparameter	TP	FP	FN	TN	Accuracy	Precision	Sensitivity	Specificity
NN	k=8	489	170	315	2283	85.11	74.20	60.82	93.07
NN	k=7	454	131	350	2322	85.23	**77.61**	56.47	**94.66**
NN	k=6	507	179	297	2274	**85.39**	73.91	63.06	92.70
NN	k=5	489	164	315	2289	85.29	74.89	60.82	93.31
SVM	/	448	149	356	2304	84.49	75.04	55.72	93.93
DT	ccp_alpha=0.020	328	132	476	2321	81.33	71.30	40.80	94.62
DT	ccp_alpha=0.015	360	139	444	2314	82.10	72.14	44.78	94.33
DT	ccp_alpha=0.010	417	143	387	2310	83.73	74.46	51.87	94.17
DT	ccp_alpha=0.008	417	143	387	2310	83.73	74.46	51.87	94.17
NB	var_smoothing=0.15	751	1201	53	1252	61.50	38.47	**93.41**	51.04
NB	var_smoothing=0.10	751	1172	53	1281	62.39	39.05	**93.41**	52.22
NB	var_smoothing=0.08	749	1166	55	1287	62.51	39.11	93.16	52.47

B. Experiments

Here we conduct four groups of experiments to test the performance of four models (i.e., Neural Network, Support Vector Machine, Decision Tree and Naive Bayes) on the Adult Income Prediction task.

For Neural Network, we try different values of k (which is defined in Fig. 2) to adjust the number of layers and neurons in the network. We use the best model within 50 training epochs to evaluate the performance (since all Neural Network models here reach the state of over-fitting after training 50 epochs).

For Support Vector Machine, we set gamma as:

$$\gamma = \frac{1}{\text{\# Features} \times \text{Var}(X)}$$

For Decision Tree, we set ccp_alpha as $0.020, 0.015, 0.010$ and 0.008 to evaluate the impact of complexity penalty on the performance of the model.

For Naive Bayes, we set var_smoothing as $0.15, 0.10, 0.08$.

After evaluation, we can acquire the number of true positive (TP), false positive (FP), false negative (FN) and true negative (TN) samples. Furthermore, we use the following metrics to evaluate the performance of the model:

- Accuracy = (TP + TN) / (TP + TN + FP + FN)
- Precision = TP / (TP + FP)
- Sensitivity = TP / (TP + FN)
- Specificity = TN / (TN + FP)

To make the results more intuitive, we present the results in percentage terms.

Finally, we can get the results shown in Table I.

C. Analysis

From the results shown in Table I, we can see that the performance of Neural Network, Support Vector Machine and Decision Tree are relatively close. Among the three models, Neural Network is the best as it reaches the highest score in three metrics (except sensitivity). Though Naive Bayes have much higher scores in sensitivity than other models, it performs bad when considering accuracy, precision and specificity, which also matches our prediction mentioned in Sec. III-C4.

Considering Neural Networks with different complexity, the network with $k = 7$ is preferred as it has the highest precision and specificity while its accuracy is only 0.16% less than the maximum score obtained by Neural Network with $k = 6$.

Therefore, we choose Neural Network with $k = 7$ for submission.

V. CONCLUSION

In this paper, we compare the performance of Neural Network, Support Vector Machine, Decision Tree and Naive Bayes on predicting adult's annual income (based on the Adult Census Income dataset). By conducting experiments, we find that Neural Network has the best performance among all types of models while its performance reaches maximum when it has 3 hidden layers (i.e., 3 Dense Layer - ReLU Activation Layer - Dropout Layer blocks).

Though Neural Network has the best performance in predicting adult's annual income, it also has some **limitations**:

- The interpretability of Neural Networks is far from ideal and the model is hard to be applied to realistic scene as it is unable to explain the result and can only take the result as a reference.
- Neural Network requires the longest time and the largest dataset for training among all four models, which has relatively high threshold and may not be trivial to apply it on all tasks.
- Neural Network also have the largest number of hyper-parameters among all the models (such as structure of the network and the size of each layer). In this paper, we simplified our experiments by using a fixed structure of Neural Network, which may not exhibit the best performance of Neural Network.

Different from previous projects, we are able to use much more packages in python, including powerful libraries in machine learning, such as keras and scikit-learn. These packages have greatly simplified the process of building models and training them, which make it possible for more amateurs to train their own model. As students major in computer science, we should not be satisfied with simply using those packages to build our models, I hope that I will have the chance to dig into the models and discover more details (such as algorithms and optimizations) in the encapsulated packages and compare them with the classic theories learned in class.

ACKNOWLEDGMENTS

I would like to thank Professor Bo Yuan for his informative lectures and guidance through the project.

I would also like to thank Teaching Assistants Yibo Tang, Tingjing Zhang and Xiang Yi for the tutorial sessions and patient question-answering.

REFERENCES

[1] "Adult Census Income," *www.kaggle.com.* https://www.kaggle.com /datasets/uciml/adult-census-income