Predicting Adult Census Income Using XGBoost Gradient Boosted Trees System

Haibin Lai, 12211612, *SUSTech*

***Abstract*—Tree boosting is a highly effective and widely used machine learning method. This project focuses on predicting the Adult Census Income Dataset using the XGBoost Gradient Boosted Trees System from [1]. Beginning with a brief overview of the dataset and its objectives, we conducted a review of relevant literature, explored dataset analysis and visualization techniques, and ultimately selected the XGBoost model after careful consideration of the data's characteristics and with the assistance of Autogluon. Subsequently, we delved into the principles of XGBoost and implemented it in Python. Through evaluation on testing and prediction phases, we performed experiments involving data pruning and hyperparameter tuning, resulting in a significant enhancement of our model's accuracy to 87.85%.**

***Index Terms*—Artificial Intelligence, Machine Learning, Data Mining, Gradient Tree Boosting, Autogluon, XGBoost**

I. Introduction of the Problem

THE prominent inequality of wealth and income is a huge concern in the whole world. So, learning and predicting adult Census Income is an essential job nowadays. In this project, we need to implement a model analyze a classical dataset Adult Census Income from [6]. We need to predict whether income exceeds $50K/yr based on census data like people's age, gender and education level etc.

II. Literature Review

As the rise of boosting in machine learning as [4] said, using machine learning models to help analyze and predict data power many aspects of modern society. While we are choosing the predict model for the dataset, we consider the model of Decision Tree, NeuralNet, Random Forest, K Neighbors Distance. Also, some the data scientists had analyzed the dataset. Vidya, Sejal and Ronit [6] had implemented Extra Trees Classifier in the dataset and had reached 88.16% Validation Accuracy. Also they pointed out in conclusion that running XGBoost could reach the score of 87.53%. Leo's Random Forest model [5] worked well in some dataset like zip-code data with 5.1% training errors. Ali [8] used decision Tree model in Sklearn library on the dataset got 84% Validation Accuracy. And Chakrabarty designed a new Tree Classifier which holds the best classifier that beats PCA, XGBoost with 88.16% accuracy.

According to previous studies, we can have a rough understanding of the basic characteristics of this dataset and the suitability of models. At the same time, through literature review, we understand that the decision tree algorithm has shown good performance in predicting this dataset.

CS311 Project 3 Haibin Lai 12211612
Manuscript received June 9 2024

III. Dataset Inspection

A. Dataset Introduction

The dataset is extracted from the 1994 Census bureau database by Ronny Kohavi and Barry Becker [3]. The prediction task is to determine whether a person makes over $50K a year.

The dataset has several columns: age, workclass, final weight, education, education year, marital status, occupation, relationship, race, sex, capital gain, capital loss, working hours per week and their native country.

B. Data Visualization

To find out which methodology suits the data, we need to take a bird eye view on our data set.

We first look at the relationship between age and income. We can find that the age around 40 to 60 may get a higher income ratio in figureIII-B(b), which is nearly a quarter of the population we count as figure III-B(a) shows.

Then we take a glimpse on the number of observational representatives of a sample in a state. We found that the samples with a final weight of 0.2 are the most numerous, and at the same time, they have the highest proportion of samples on income as figureIII-B (e) shows.

From figure III-B(c) we can observe that people from High school and collage maintain primary, while people in Bachelors, Master and higher education level has much higher income ratio. They have more than 13 education year and figure III-B(d) just display this trend.

From figure 2, for Occupation, jobs like Craft-repair, Sales, Prof-specify and Exec-managerial seems to maintain higher income .

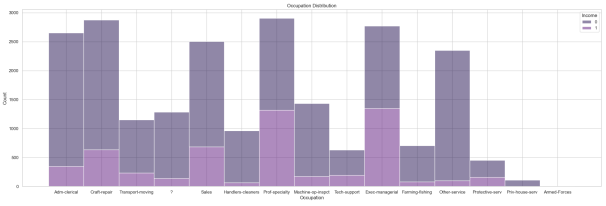
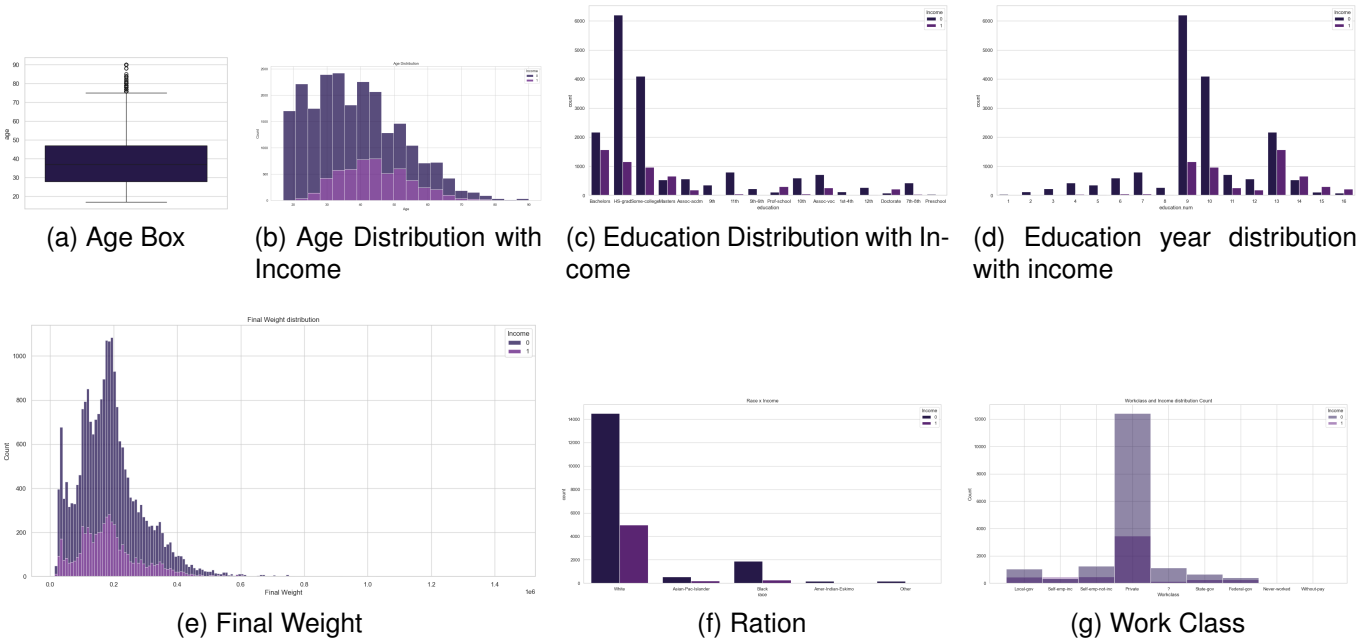


Fig. 2. Occupation

We continue on analyzing race and work class impact on income. We can find that ration of white takes majority of dataset with highest ratio of incomeIII-B(f). As for work classIII-B(g), the ratio of Local-gov, Self-emp-inc, Federal gov for high income maintain high level but are in few amount.

By implementing Linear Regression on every two feature of these 14 feature, we draw out the Correspondence Matrix



(a) Age Box (b) Age Distribution with Income (c) Education Distribution with Income (d) Education year distribution with income

(e) Final Weight (f) Ration (g) Work Class

Fig. 1. Data Visualization of Age and Education

as figure 3 displays. Between each of these feature, we can find that the ***education year, capital gain, working hours per week*** are **highly Positively correlated** to the income comparing with other feature. ***Race, occupation*** has very **low statistical significance** with income. ***Relationship*** are **highly negatively correlated** with income comparing to other feature. This visualization helps guide us deciding features of data and show some direction of pruning to raise the accuracy of the model.

Due to the importance of data, although some features do not mention here but more visualization and analysis can be found at GithubVIII-C.

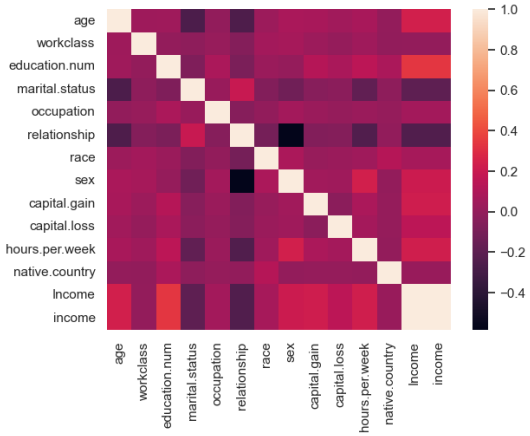


Fig. 3. Correspondence Matrix

C. Data Preprocessing

In Data Preprocessing part, we try the following 3 steps to raise our training effect.

1. Reordering Data

Reordering data with merging income label into training data file. This involves arranging the dataset in a structured manner, facilitating our training of models.

2. Drop empty line and duplicate data

By dropping empty lines and duplicate data, we eliminate potential sources of bias and prevent inaccurate conclusions. Moreover, excluding incomplete instances and same samples reduces computational overhead and enhances the efficiency of data processing pipelines.

3. Scaling Data:

After that, by scaling features to a common range [0, 1], we ensure fair comparisons between variables and facilitate the convergence of optimization algorithms. Additionally, scaling enhances the interpretability of results by removing the influence of measurement units.

IV. Proposed Methodology: XGBoost

A. Methodology Selection

The next problem come to us is that what model we should use. Several mainstream machine learning models were considered.

First model is Logistic regression. It performs well on linearly separable. **However**, Logistic regression may perform poorly since we are handling nonlinear relationships with income and might struggle to capture complex patterns in the data.

Secondly, **Neural Networks**. It can learn complex nonlinear relationships and perform well on large datasets. However, it requires large amounts of data and computational resources, which our dataset doesn't provide. Also interpretability counts but it's poor.

Thirdly, **Decision Trees**. It can handle nonlinear relationships and complex feature interactions, with good inter-

pretability. They are robust to data distribution and capable of handling mixed data types.

Last but not least, **Random forests**. It ensembles learning methods based on decision trees, reducing overfitting risk by randomly selecting feature subsets and sample subsets.

Here my one of my Senior, Canming Ye, advise me that maybe I can try AutoGluon first and test which model runs best in the dataset.

AutoGluon is Amazon's AutoML library. The library was open-sourced by Amazon Web Services (AWS) at re:Invent 2019. With AutoGluon, we can train state-of-the-art machine learning models for tasks like image classification, object detection, text classification, and tabular data prediction.

So here we implement Autogluon on the dataset. It tested 14 models like LightGBMXT, a Decision Tree Model, Random ForestGini, Neural Network on PyTorch, KNN on sci-kit learn. And it output the fitting rank by their precision score as figure 4 shows.

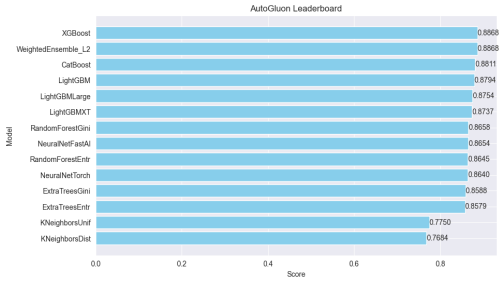


Fig. 4. Autogluon Model Learning Accuracy

From the result we can find that **XGBoost, a decision tree model with Extreme Gradient Boosting**, fit the data set best as a ratio of 0.8868. Weighted Ensemble_L2 holds the same score as XGBoost but with longer time consume. Also, Vidya, Sejal and Ronit [6] report that XGBoost works well with its well classification on data feature like education year, which we find on data visualization part. Above all We come to the conclusion that using XGBoost may best suitable for Adult Census Income Dataset.

B. Introduction to XGBoost

XGBoost stands for "Extreme Gradient Boosting", where the term "Gradient Boosting" originates from the paper *Greedy Function Approximation: A Gradient Boosting Machine* [9], by Friedman. XGBoost is used for supervised learning problems, where we use the training data (with multiple features) x_i to predict a target variable y_i (4.1):

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in \boldsymbol{F}$$

XGBoost belongs to the family of gradient boosting algorithms, which sequentially combine weak learners (usually decision trees) to create a strong learner. It builds models in an additive manner, where each new model corrects errors made by the previous ones.

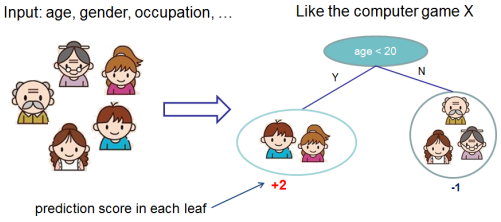


Fig. 5. Decision Tree Ensembles

C. Regularized Learning Objective

XGBoost employs an ensemble of **decision trees** just like 5, it accept the input data and predict them into trees, where each tree learns to predict the residuals (errors) of the previous trees using **objective function** $obj(\theta)$ (4.2):

$$obj(\theta) = \sum_{i}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \omega(f_k)$$

Where ω is the penalty parater and l is the loss function. In XGBoost, we define ω as following(4.3):

$$\omega(f) = \Upsilon T + \frac{\lambda \sum_{j=1}^{T} \omega_j^2}{2}$$

D. Objective Validation Function of XGBoost

For the definition of incremental modeling, we utilize the prediction results of each tree to fit the residuals of the previous tree's predictions. This approach ensures that the overall tree model performance improves progressively.

1) Tree Boosting: Here we use **Tree Boosting** model proposed in passage [1] (4.4):

$$obj(\theta) = \sum_{i}^{n} l\left(y_i, \hat{y}_i^{(t)}\right) + \sum_{k=1}^{K} \omega(f_k)$$

This is the Extreme Gradient Boosting. By using this function, we can find a effective spilt tree in boosting way.

2) Additive Training: We take **Additive Training** to find the parameters of trees. This iterative process continues until a predefined number of trees is reached, or until no further improvement can be made:

$$\begin{aligned}
\hat{y}_i^{(0)} &= 0 \\
\hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\
\hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i)
\end{aligned}$$

$$\cdots\cdots$$

$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)$$

So in the general case, we combine (4.2),(4.3) into the function (4.4) and take the *Taylor expansion of the loss function up to the second order*(4.5):

$$obj^{(t)} = \sum_{i}^{n} \left[l\left(y_i, \hat{y}_i^{(t-1)}\right) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \omega(f_t) + c$$

In this equation (4.5), parameters are independent with respect to each other, the form is quadratic and the best for a given structure and the best objective reduction we can get is the derivative:

$$g_i = \partial_{\hat{y}_i^{t-1}} l\left(y_i, \hat{y}_i^{t-1}\right)$$

$$h_i = \partial^2_{\hat{y}_i^{t-1}} l\left(y_i, \hat{y}_i^{t-1}\right)$$

Usually, a single tree is not strong enough to be used in practice [7]. What is actually used in XGBoost is the ensemble model, which sums the prediction of **multiple trees** together just like 6. We build and tune a tree's parameter by learning their predict score. After that, We use objective value with the i-th tree as formula $obj_{(t)}$ (4.5) shows. For a given tree structure $q(x)$, we can derive the objective function for **whole model objective**:

$$I_j = \{i|q\left(x_i\right) = j\}, \; G_j = \sum_{i \in I_j} g_i, \; H_j = \sum_{i \in I_j} h_i$$

$$obj^{(t)} = \sum_{j=1}^{T} \left[G_j \omega_j + \frac{1}{2}\left(H_j + \lambda\right)\omega_j^2\right] + \Upsilon T$$

With the objective function determined, the evaluation criteria for our tree model are also established.

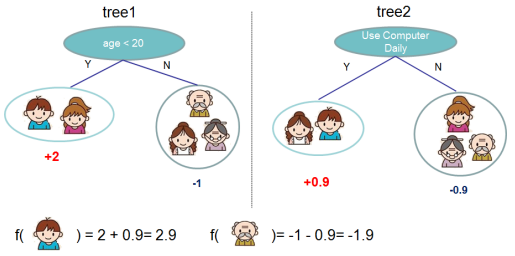


Fig. 6. Multiple Trees used in XGBoost

Here we differentiate the funtion and get objective function in practice7:

$$\omega_j^* = -\frac{G_j}{H_j + \lambda}$$

$$obj^* = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j + \lambda} + \Upsilon T$$

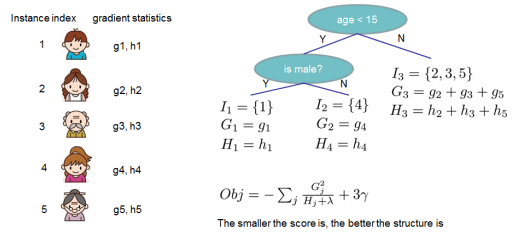


Fig. 7. Decision Tree Model in XGBoost

We define our objective function $Gain$ (4.6) from (4.5) to evaluate a subtree if specifically we try to split a leaf into two leaves , which involves subtracting the objective value after the split from the subtree before the split:

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{\left(G_L + G_R\right)^2}{H_L + H_R + \lambda}\right] - \Upsilon$$

E. Key Boosting Algorithm in XGBoost

1) Spilt Finding Algorithm used by XGB: The key issue in equation (4.6) is to find appropriate splits. The precise greedy algorithmIV-E1 finds the optimal split solution by enumerating all possible partitions of features.

Algorithm 1: Exact Greedy Algorithm for Split Finding

Input: I, instance set of current node
Input: d, feature dimension
$gain \leftarrow 0$
$G \leftarrow \sum_{i \in I} g_i$, $H \leftarrow \sum_{i \in I} h_i$
for $k = 1$ ***to*** m **do**
  $G_L \leftarrow 0$, $H_L \leftarrow 0$
  for j *in sorted(I, by* $\mathbf{x}_{jk}$*)* **do**
    $G_L \leftarrow G_L + g_j$, $H_L \leftarrow H_L + h_j$
    $G_R \leftarrow G - G_L$, $H_R \leftarrow H - H_L$
    $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  end
end
Output: Split with max score

Similarly, other nodes repeat this process recursively until reaching the maximum tree depth, or stop growing when the sum of sample weights is less than a set threshold to prevent overfitting. This completes the training process for a tree. The training process for the next tree involves computing gradients and determining the tree structure in the same manner.

2) Approximate algorithms: Approximate algorithms alleviate the computational burden by eschewing exhaustive enumeration of all possible feature splits. As datasets grow larger, it becomes nearly impossible to load all data into memory, rendering precise partitioning impractical. Even in distributed settings, precise partitioning can present challenges. We have thus devised approximate strategies, as illustrated in Algorithm IV-E2.

Algorithm 2: Approximate Algorithm for Split Finding

for $k = 1$ ***to*** m **do**
  Propose $S_k = \{s_{k1}, s_{k2}, \cdots s_{kl}\}$ by percentiles on feature k.
  Proposal can be done per tree (global), or per split(local).
end
for $k = 1$ ***to*** m **do**
  $G_{kv} \leftarrow= \sum_{j \in \{j | s_{k,v} \geq \mathbf{x}_{jk} > s_{k,v-1}\}} g_j$
  $H_{kv} \leftarrow= \sum_{j \in \{j | s_{k,v} \geq \mathbf{x}_{jk} > s_{k,v-1}\}} h_j$
end
Follow same step as in previous section to find max score only among proposed splits.

This algorithm initially proposes candidate split points based on the distribution percentages of features. Subsequently, it maps features to slots according to these candidate split points and identifies the optimal split percentage.

V. Training the Model

A. Implement detail

Here we train the XGBoost model using python 3.11 with package *sci-kit learn* and *xgboost* from official.

And our software and hardware description is as following Table I. More details are in github website provided in VIII-C

TABLE I
SOFTWARE AND HARDWARE DESCRIPTION

Software / hardware	Name
CPU	11th Gen Intel(R) Core(TM) i7-11370H @ 3.30GHz
Python Version	3.11
sci-kit learn	1.4.0
XGBoost	2.0.3

Class	Precision	Recall	F1-score	Support
0	0.90	0.93	0.91	2238
1	0.76	0.66	0.71	725
Accuracy			0.87	2963
Macro avg	0.83	0.80	0.81	2963
Weighted avg	0.86	0.87	0.86	2963

TABLE II
CLASSIFICATION REPORT

B. Model Evaluation

Out of a total of 22792 instances present in the dataset, 19829 instances have been used for training XBGoost while the rest 2963 instances(13% of the dataset) have been reserved for testing. After complete evaluation, the model output a Confusion Matrix for the train dataset.

- The top-left value (2083) represents True Positives (TP)
- The top-right value (155) represents False Negatives (FN)
- The bottom-left value(243) represents False Positives (FP)
- The bottom-right value(482) represents True Negatives (TN)

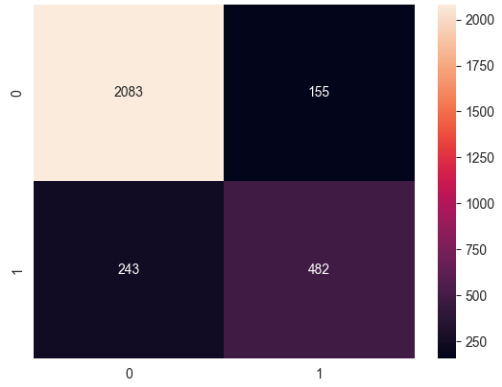


Fig. 8. Confusion Matrix

Here we evaluate the model performance on the following metrics:

- Training Accuracy $= \frac{\text{NP}}{\text{Total Number of Samples}} \times 100\%$, From the model, a Training Accuracy of 86.57% is achieved.
- Recall $= \frac{TP}{TP+FN} \times 100\%$, We use recall to measure the ability of the model to correctly identify positive samples.
- Precision $= \frac{TP}{TP+FP} \times 100\%$, Precision measures the accuracy of the positive predictions made by the model. It's the ratio of correctly predicted positive observations to the total predicted positives.
- $F_1 = \frac{2}{\frac{1}{Recall}+\frac{1}{Precision}}$ F1Score is the Harmonic Mean of Recall and Precision.
- **Macro average** calculates the average of the precision, recall, and F1-score for each class without considering class imbalance.
- **Weighted average** calculates the average of precision, recall, and F1-score for each class while considering the number of instances for each class. It gives more weight to classes with more instances.

From table II we can see that the accuracy reach to 87%(86.57%) in total. And the precision of 1 is 0.76, which shows out that the model have lower prediction power on predicting high income people. However, precision of 0 is in high level to 90%. Also, the recall of class 1 is also in low level of 66% while class 0 contains level of 93%. These are mainly because the majority of income class is 0, then the decision tree failed to fit them since not enough 1 data may be trained.

So, we have a nice prediction ratio and a good for predicting the income of 1. But we need improvemnt on predicting the high income community.

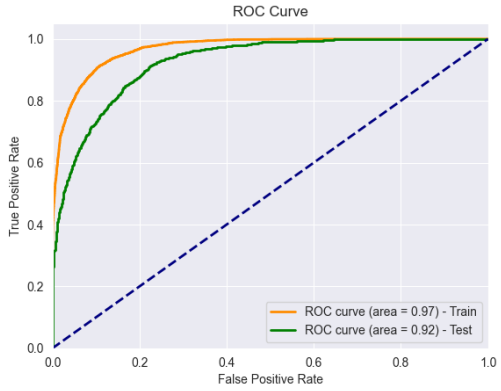


Fig. 9. ROC Curve showing the Area Under the Curve

The Receiver Operator Characteristic Curve (ROC Curve) is implemented using the decision function attribute of Gradient Boosting Classifier which returns values that give a measure of how far a data-point is away from the Decision Boundary from either side. It's shown as figure 9.

The ROC curve is plotted with FPR on the x-axis and TPR on the y-axis. The area under the ROC curve (AUC) is used as a metric to evaluate the model's performance, with values ranging from 0 to 1. A higher AUC indicates better model performance.

In an ideal scenario, the TPR should be as high as possible while keeping the FPR as low as possible. Therefore, the ROC curve closer to the upper left corner signifies better model performance. Here we have ROC curve of area 0.97 for train data and 0.92 for test data, which shows that we perform the performance of the model well in the dataset and **our model is well trained**.

Fig. 10. Accuracy and ROC area of XGBoost by droping some feature

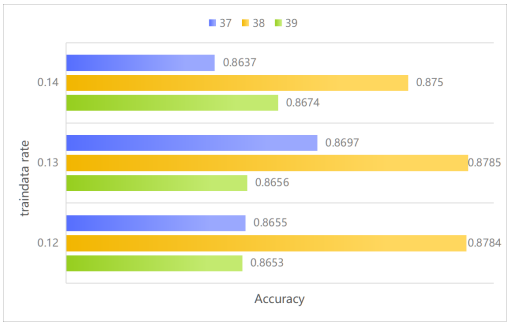


Fig. 12. Tuning hyperviser of test data size ratio of train data and random state

VI. Experiment and Result

A. Changing the classified input features

As we mention in Data Visuilization part, some of the data may not be very essential. So we can try to train the model without taking them. Here we drop some of the data feature to see if XGBoost can maintain or raise up its accuracy.

We drop one feature at each time and count its accuracy and ROC area. **Surprisingly, the accuracy maintain at a steady level**. What's more, dropping ***"Sex"*** feature even make the prediction better with prediction accuracy rate 0.87.

We next try to drop the data feature and train the XGBoost. We drop 'sex', 'age', 'workclass', etc as figure 11 displays. Surprisingly, the accuracy and ROC area drop slowly. After dropping 8 features, our pridiction accuracy comes to 0.8555. According to the testdata we try on dataset, about $(2083 + 155)/2963 \times 100\% = 75.53\%$ of the data is 0. That means the actual predict rate on class 1 may drop down up to 12.63% , since the prediction rate of 1 originally is $(87\% - 75.53\%)/24.47\% = 46.87\%$, but the new one is $(85.55\% - 75.53\%)/24.47\% = 40.95\%$, the rate drop down to $1 - 40.95\%/46.87\% = 12.63\%$ as before.

So that proves the robustness of XGBoost. Even with cutting half of the feature down from the dataset, the model can still perform and also the ROC area maintain greater than 90%.

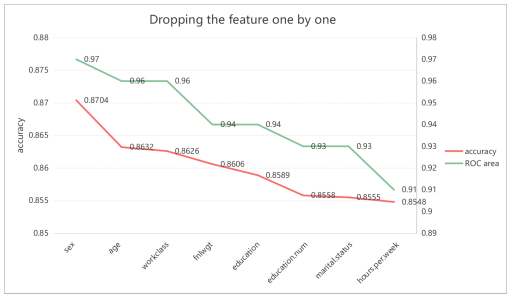


Fig. 11. Drop the feature one by one

B. Tuning the hypervisor

We now try to tune the hypervisor like testdata size in training, random state for splitting. Here is the result 12. We can find that when testdata size = 0.13 of traindata, random state comes to 38, then we come to a local maximum of the tuning. Since we work well in this tunning, I submit the data with this one.

VII. Comparing the Result with Autogluon

We compare our predicting result with Autogluon. And we can find the distribution of them. In tableIII, we find that the

TABLE III
Line Diffusion between our XGBoost and Autogluon

Model Prediction	Numeric
Autogluon Expected line to be 1	1990
XGBoost Expected line to be 1	1902
Line Diffusion Count	476

numeric count of predicting line of 1 is near with Autogluon 1990 and XGBoost 1929. But Line Diffusion Count is 451 which means our data predict 451 instance different from Autogluon.

We try to find different variable to get close to Autogluon's Model since it has more tuning experience. We try with some of the method like dropping, adding the ratio of training dataset. However eventually we still can't beat Autogluon. After reading paper, we find that Autogluon will run XGBoost decision tree to predict with genetic algorithms13. It used the framework of TPOT which will cope with the iiegular space, which is not included in our fitting model.

> **TPOT.** The Tree-based Pipeline Optimization Tool (TPOT) of Olson & Moore (2019) employs genetic algorithms to optimize ML pipelines. Each candidate consists of a choice data processing operations, hyperparameters, models, and the option to stack ensembling with other models. While their evolutionary strategy can cope with this irregular search space, many of the randomly-assembled candidate pipelines evaluated by TPOT end up invalid, thus, wasting valuable time that could be spent training valid models.

Fig. 13. Optimization of Autogluon on XGBoost

VIII. Conclusion

A. Limit and Prospective

We find our XGBoost classifer contains some of the **limit**:

Instability: XGBoost are sensitive to small variations in the training data. Even minor changes in the data can result in completely different tree structures just like **education year**. Somehow this is not very good to our dataset since a slightly change in people's feature may not cause tremendous change on income.

Local Optima: XGBoost is still in a greedy algorithms that make locally optimal splits to build the tree. This greedy decision-making may lead to suboptimal trees that are not globally optimal, potentially missing better splits in some cases.

Hard to tune hypervisor: XGBoost contains the feature of decision tree, some of its feature are not tunning while in Autogluon they tune it with Generic Algorithm.

B. Prospects for Our Model's improvement

Ensemble a better Gradient Boosted Trees Functions just as Chakrabarty's new function in [2]. So that we can combine multiple decision trees to reduce overfitting and improve generalization.

Employing Randomize algorithms' idea like Random Forests that build multiple trees on random subsets of the data and aggregate their predictions can improve stability. By doing that, we may reach out the local maximum to find the maximum accuracy. Also, Generic Algorithm like in [7] is important on tuning.

A reconsideration on Data Preprocessing. Utilize techniques like binning or discretization from [1] to transform continuous features into categorical ones, making them more suitable for decision trees. Decision tree variants like Regression Trees or Conditional Inference Trees are specifically designed to handle continuous target variables.

C. Summary

Overall, in this project we predict Adult Census Income Dataset using XGBoost Gradient Boosted Trees System. We take short introduction to the dataset and its goal, get a review on the formal coreleated works, doing dataset analysis and visulization, choosing out XGBoost model by analyzing the style of the data and with the aid of Autogluon. We next introduce the principle of XGBoost and implenment it on python. And we have done evaluation on its testing and predicting. We do some experiment on tuning with dropping data and tuning hypervisor, which eventually brings our model's accuracy to 87.85%.

ACKNOWLEDGMENTS

Thanks to Prof.Yuan for teaching CS311 and giving me advice on the algorithm. Thanks to our class TA help me on ordering the report and improving my source code. Thanks to Canming Ye on advising Autogluon, hope you enjoy your Master on Qinghua University! Having a bit of taste on the area of Machine learning, the joy of exploration and research is what reading and implementing them brings to me.

RELEVANT WEBSITES

This project has been on line in github: Laihb1106205841/CS311-AI-Peoject3-XGBoost: Predicting Adult Census Income Using XGBoost Tree Boosting System (github.com)

REFERENCES

[1] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16). Association for Computing Machinery, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785 XGBoost | Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining

[2] Chakrabarty, N., & Biswas, S. (2018). A Statistical Approach to Adult Census Income Level Prediction. *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, 207-212.

[3] Vidya Chockalingam, Sejal Shah and Ronit Shaw: *"Income Classification using Adult Census Data"*,

[4] LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* 521, 436–444 (2015). https://doi.org/10.1038/nature14539 Deep learning | Nature

[5] Leo Breiman. 2001. Random Forests. Mach. Learn. 45, 1 (October 1 2001), 5–32. https://doi.org/10.1023/A:1010933404324 Random Forests | Machine Learning (springer.com)

[6] 1994 Census bureau database by Ronny Kohavi and Barry Becker (Data Mining and Visualization, Silicon Graphics

[7] Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M., amp; Smola, A. (2020). AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data. arXiv preprint arXiv:2003.06505.

[8] Ali, Ahmed. (2020). Machine-learning analysis for adult census income dataset. Machine-learning analysis for adult census income dataset (researchgate.net)

[9] Friedman, J.H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics, 29*, 1189-1232.