Project Report
DSAA2011 Machine Learning

Hua XU
HKUST(GZ)
hxu401@connect.hkust-gz.edu.cn

Jianhao RUAN
HKUST(GZ)
jruan189@connect.hkust-gz.edu.cn

Leyi WU
HKUST(GZ)
lwu398@connect.hkust-gz.edu.cn

1 Introduction

don't forget to finish this part

2 Methods and results

add an introductory sentence here

2.1 Data preprocessing

- observed patterns of dataset or insights

The Adult Income datasets is a binary classification dataset with 14 usable variables, six of which are integer types while the rest eight variables are categorical. In this project, we are using different strategy to process the data.

- As for the categorical variables, we firstly use the most frequent value to impute the missing values in the data, then we use one-hot encoding to convert the values into one-hot vectors. This step is implemented in Python with `OneHotEncoder` in scikit-learn.
- As for the numeric variables, we use median values of the variables to replace the missing values, and after dealing with the missing values, we then normalize the data to 0 mean and 1 variance with `StandardScaler` in scikit-learn.

After imputing missing values and converting all features to numeric ones, we then get a wide dataframe with 105 features, making the computation costs high and taking much time in training and testing. For efficiency and visualization purposes, we then use t-SNE to reduce the dimensionality of the original data and get both the 2D and 3D embedding for future usages.

Table 1: The relationship between performance of t-SNE (measured with KL divergence) and the target dimension and perplexity. The lower the KL divergence value is, the better the performace is.

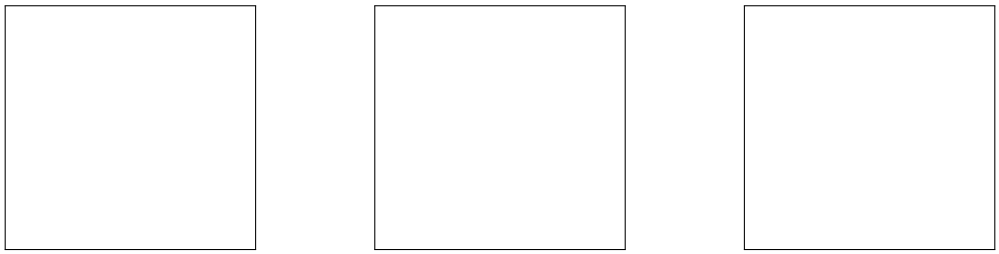
		Perplexity		
		10	30	50
Dimensionality	2	ValueA	ValueB	ValueC
	3	ValueD	ValueE	ValueF

justify why we should use t-sne in 3D: comparing the time and performance

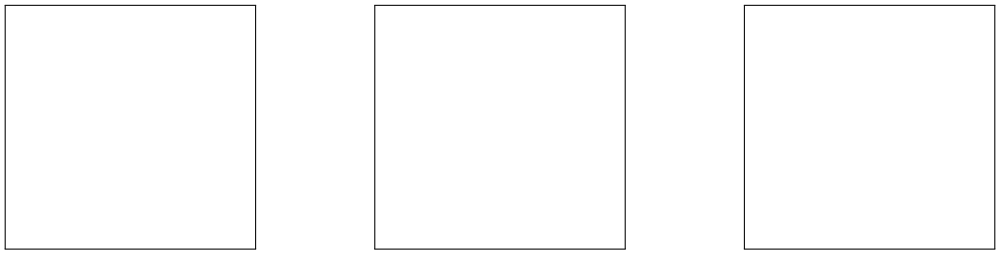
Preprint. Under review.

2.2 Data visualization

explicitly show the mis assigned labels and try to make the 3D graph more readable



(a) A (b) B (c) C



(d) D (e) E (f) F

Figure 1: This figure shows the visualization results using t-SNE with different dimensionality and different perplexity. The corresponding parameters are showed on the figure.

With 2D and 3D t-SNE embedding, we would like to visualize the dataset and explore more information within the dataset. For the 2D visualization, the first thing we notice is that samples from two different classes are not very distinguishable as they fall in the same clusters with different labels. As for samples with income less than 50K, they are spread over almost all the figure while the samples with income greater than 50K tend to cluster on only one side of the figure, which indicates that the class $\leq 50K$ is a more general, diverse class with samples from different distribution while for the class $> 50K$.

If we take a closer look at the labels inside, we may see that there are different sub clusters.

Moreover, this also indicates that most of the features in the original dataset may not be useful in terms of separating the samples apart and these features make the samples close to each other in the original feature space, and we may need to find the important ones for better performance.

Moreover, this also indicates that if we just directly use the t-SNE embedding to perform future tasks, we may fail since we cannot separate the samples based on the embedding good enough.

2.3 Clustering analysis

Table 2: Comparison of Clustering Algorithm Performance Metrics

Algorithm	Silhouette Score	Calinski-Harabasz Score	Davies-Bouldin Score	Adjusted Rand Score
K-means	0.123	12345.67	1.234	0.123
Agglomerative Clustering	0.456	67890.12	0.987	0.456

Methods

Based on the t-SNE visualization, we have already observed the clustering characteristic within the data and we can perform clustering analysis to gain more insights. Here we are using K-means and agglomerative hierarchical clustering.

(a) A (b) B

Figure 2: This figure shows the clustering results for K-means and agglomerative hierachical clustering.

K-means K-means is a classic clustering algorithm, which iteratively calculates the distance between sample points and the centroids of the cluster. The algorithm can be divided into **assignment step** and **update step**, as the assignment step assigns each data point x_i from the dataset X to the closest cluster centroid μ_i using the formula $r_i = \underset{k\in\{1\dots K\}}{\arg\min} \|x_i - \mu_k\|$. The centroids will then be updated during the update step, as each of the centroid of the cluster will be updated with the formula $\mu_k = \frac{\sum_{i:r_i=k} x_i}{\sum_{i:r_i=k} 1}$. In this method, the choose of the initial centroids is very important and therefore here we uses the **k-means++** initialization, which choose the first center randomly from the data points and for each remaining point, compute its squared distances to the nearest existing center. After that we choose another centroid with probability proportional to the squared distance. Then we repeat the process until K centers are chosen. This can reduce sensitivity to poor initialization, theoretically guarantees $O(\log K)$-competitive solutions. We implement this method in Python with `KMeans` in scikit-learn.

Agglomerative clustering Agglomerative clustering is a bottom-up approach which starts with each data point as a single cluster and merges the closest pairs of clusters iteratively. It will continue the process until one cluster remains. In this method, the way to measure the distance between the clusters will affect the performance of clustering methods, and we may have for different ways: 1) single linkage, which uses minimum pairwise distance; 2) complete linkage, which uses maximum pairwise distance; 3) average linkage, which uses average of all pairwise distances; 4) wardd's method, which aims at minimizing within-cluster sum of squares measured by the costs of merging cluster A and B: $\Delta(A,B) = \frac{2n_A n_b}{n_A+n_B}\|\mu_A - \mu_B\|^2$ where n_A, n_B means the number of elements in the corresponding clusters. We implement this method in Python with `AgglomerativeClustering` in scikit-learn.

For these two methods, k-means is the most classic clustering methods which can act as a baseline and since it is fast, it can quickly bring us some insights on the clustering analysis. And since we have already see sub-clusters in the t-SNE analysis, we may try to use hierarchical clustering for more precisely capturing the locality of small sub-clusters.

Results

The clustering results is visualized in Fig and we have used some methods to evaluate the performance of the clustering.

need to analyze the results and tell which one is better in terms of performance

need to gain more insights from the clustering

2.4 Prediction: training and testing

need to describe the chosen classification target, model classes and why they were selected

need to describe the training process

Methods

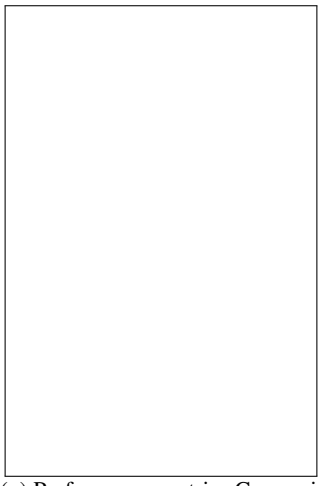
Based on the t-SNE visualization we may see that it is very hard to find a continuous line or surface to separate the samples from the two different classes apart. Therefore, as for prediction, we would like to use decision tree based models and logistic regressions, as the decision tree based models may create discrete decision boundaries while logistic regression may bring us a simple decision boundary, making the classification precise.

Logistic regression Logistic regressions 123456.

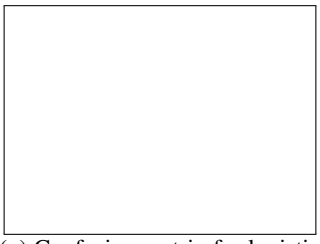
Decision trees Decision trees 123456.

Results

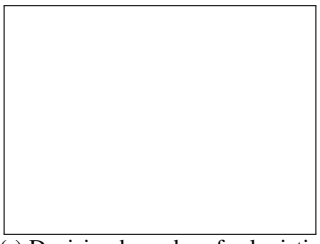
Figure 3: This is the performance comparison for the two vanilla models.



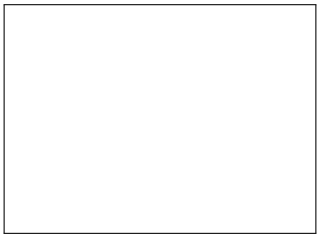
(p) Performance metrics Comparison (the heat map)



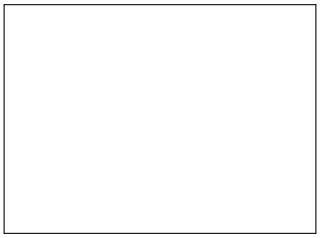
(q) Confusion matrix for logistic regression



(r) Decision boundary for logistic regression



(s) Confusion matrix for decision tree



(t) Decision boundary for decision tree

After fitting the model with the train data, we then get the trained model and the corresponding decision boundary is also visualized in the figure. From the figure we can see that the decision tree model is more expressive, as it can bring us multiple decision areas and decision boundaries while the logistic regression model cannot handle the complex data distribution and its decision boundary and the decision boundary just go cross some of the samples.

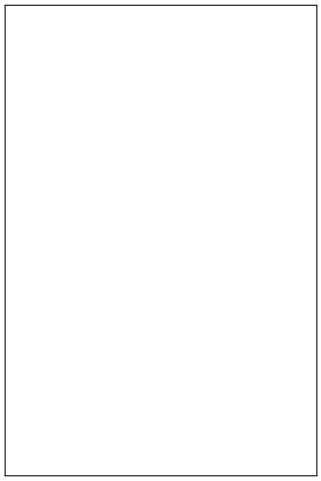
This intuition implied by the visualizatoin is also supported by the metrics. From the table below we can see that as for the overall performance.

interpretion of results

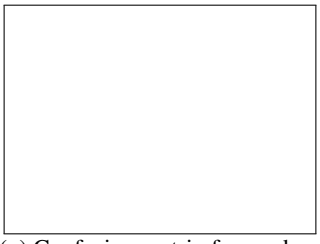
However, it is quite clear that the performance is still not amazingly great, as the accuracy is just around 1234 which is far below the legend of tree based models. Since we think the power of tree models is from its ability to create flexible decision areas, we think that adding the depth of the trees or increasing the number of trees in the process of fitting may be a good idea. We then increase the depth of a single decision tree to 1245 and did more experiments, while we used the ensemble models to do more experiments. Their decision boundary are visualized in figure xxx and the performance metrics are in table xxx. Based on the results above, we may see that the ability of classification does increase as the tree goes deeper, (but as the tree goes too deep, the performance on the test set will decrease as the model start to overfit).

And for the decision boundaries, we can see that the two models can bring us more complex but robust decision boundaries.

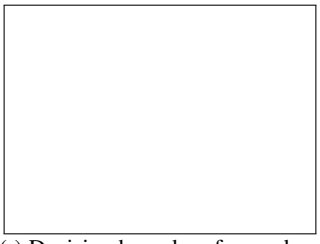
Figure 4: This is the performance comparison for the two ensemble models.



(p) Performance metrics Comparison (the heat map)



(q) Confusion matrix for random forest



(r) Decision boundary for random forest

(s) Confusion matrix for XGBoost

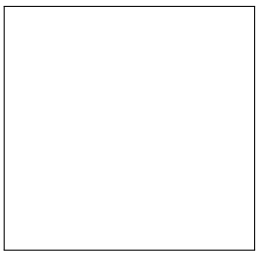
(t) Decision boundary for XGBoost

Also, we may see the importance of features for further support on our conclusion and the results can be referred to in fig xxx. For the best performance model XGBoost, we can see that the importance of features is significantly imbalanced as the xxxxx matters most while the model pays little attention on the other features. For the vanilla decision tree model and the random forest model, they pay attention to more features but result in worse performance. This interesting discovery align with the mechanism of XGBoost and the rest, as the gradient boosting model will try to learn from its failures and may be more able to learn the features which can make it away from error, while the random forest or the vanilla decision models will pay equal attention on all of the features and will be more likely affected by noises. This fact also aligns to our t-SNE visualization in figure as it clearly shows the existence of noisy features and only a few of features really matters.

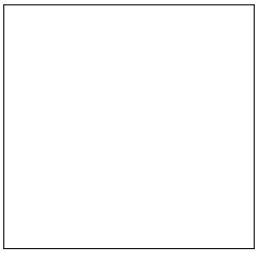
Meanwhile, the performance on one class is significantly worse than another class, which may due to the not balanced distribution in the train data.

check whether we have already explicitly compare the model performance

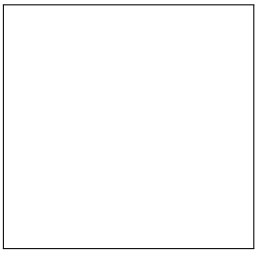
Model selection



(a) ROC of the two models



(b) top k important features of random forest



(c) top k important features of xgoost

Figure 5: This figure shows the clustering results for K-means and agglomerative hierachical clustering.

After fitting the models, we can select the model to use by analyzing their performance. The classification results are illustrated in table xxxxxxx and based on that, we can calculate the ROC and AUC in fig xxx. It shows that model xxxx has the greatest AUC and it can balance well on the data

3 Conclusion

In this project, we try to explore the intrinsic characteristics of the Adult Income dataset and try to perform classification tasks with different models on it. Through our analysis, we find that the dataset contains a lot of redundant features which do not help the task too much but still will be involved in the classification process, making the costs big and the performance bad. Only a few features matter. For better time efficiency, we firstly perform dimensionality reduction with t-SNE and found that the classes are hard to separate and there are some sub-clusters in the dataset.

then we visualized the dataset with clustering methods

Then we start to perform classification task on it. For the complex data distribution, logistic regression perform badly on the dataset as it cannot create flexible decision boundary. Tree based models performs good on the dataset, as the performance will increase as the trees get deeper. Ensemble models is the king on this task, among those the XGBoost is the good good winner, as which can learn from its failures and find the most important feature. However, the model still suffer from the imbalanced distribution of the two labels and in the future, more advanced techniques should be applied for better performance.

we may need to find one or two papers to support our conclusion that only the specific feature is important on the dataset

References

[1] Alexander, J.A. & Mozer, M.C. (1995) Template-based algorithms for connectionist rule extraction. In G. Tesauro, D.S. Touretzky and T.K. Leen (eds.), *Advances in Neural Information Processing Systems 7*, pp. 609–616. Cambridge, MA: MIT Press.

[2] Bower, J.M. & Beeman, D. (1995) *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System.* New York: TELOS/Springer–Verlag.

[3] Hasselmo, M.E., Schnell, E. & Barkai, E. (1995) Dynamics of learning and recall at excitatory recurrent synapses and cholinergic modulation in rat hippocampal region CA3. *Journal of Neuroscience* **15**(7):5249-5262.

Credit

- **Leyi WU**:
- **Hua XU**:
- **Jianhao RUAN**:
- **AI**: Formatting latex tables and figures.