Decision Tree Classification Model with Map Reduce

Yuting Mei

April 2023

1 Project Description

This is a project for designing and implementing Decision Tree Classification(DTC) model by building from scratch and use map reduce strategy for a prediction task on ‘Adult’dataset. Possible scenario for model implementation is assumed and model performance under certain setting is measured.

2 Methodology

2.1 Decision Tree Classification(DTC)

2.1.1 Tree Framework

The decision tree classification algorithm works by recursively splitting the training data based on different attributes or features. The decision-making process starts at the root node, which represents the entire dataset. At each internal node, a decision rule is applied to determine which branch to follow based on the value of a specific attribute. The process continues until stopping criteria is reached, which represents the predicted class or category.

Goal: The overall goal is to find best splits at each node which get informative and effective tree in unseen dataset, with both maximizing infomation gain to control overall reduction in uncertainty and minimizing impurity of each node with accurate label for controlling homogeneity of labels in subsets.

Stopping Criteria: either one of the following is met:

1. Tree Depth $<$ maximum depth
2. Samples in Node $\geq$ minimum number of samples

Impurity measure:

1. Impurity measure:
    (a) Gini impurity: $1 - \sum_{i=1}^{k} p_i^2$, k is the number of labels in target variable.

(b) Variance: $\boldsymbol{E}[Y^2] - \boldsymbol{E}[Y]^2$

Information gain:
$\text{Impurity}_{\text{parent}} - \sum_{i=1}^{n} w_{\text{child}} \cdot \text{Impurity}_{\text{child}}$, n is the number of child node in a parent node, $w_{\text{child}} = \frac{|child|}{|\sum_{i=1}^{n} child|}$.

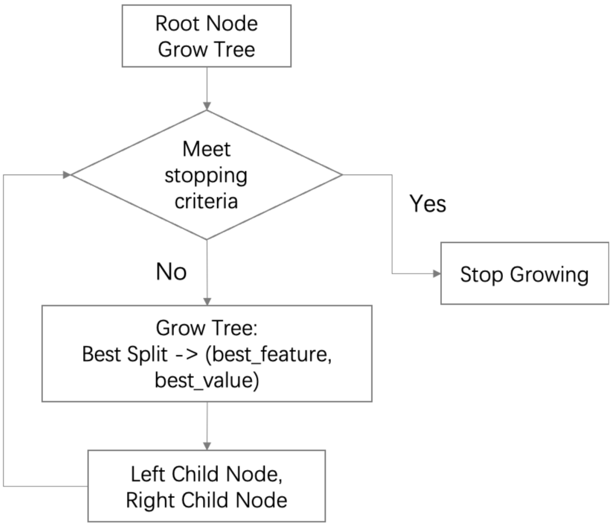


Fig1: Tree flow chart

2.1.2 Missing Value Imputation

In this project we implemented three imputation strategy for missing values: (i)recursive imputation: suitable for all data types; (ii) mean, median value imputation: suitable for numerical values in feature.

1. Recursive imputation: method inspired from $C4.5$ which recursively bracketing values based on value of feature with corresponding highest probability inside target variable until only one point in bracket. In Fig2 we show the general procedure for getting imputation in feature X_i:
   (a) Initial iteration $k = 0$. Determine the feature X_i that has missing value, removing all NAs.
   (b) For each value v_j in traverse points bracket b^k, get left and right cut, compute the probability of all labels in both left and right cut.
   (c) Get split point of value with the maximum probability of Y in all cutting points. Use the rest cutting points from the direction of maximum probability in(either left or right). $k = k + 1$.
   (d) Go back to step (b), compare the maximum probability p^k with p^{k-1}. If $p^k < p^{k-1}$, impute NA as v^{k-1}, else if $|b^k| = 1$, impute NA as v^k, else go back to step (b).

2. Mean value imputation: using mean of v in X_i to fill NA after implementing step (a), (b), (c) from first method one time.

3. Median value imputation: using median of v in X_i to fill NA after implementing step (a), (b), (c) from first method one time.

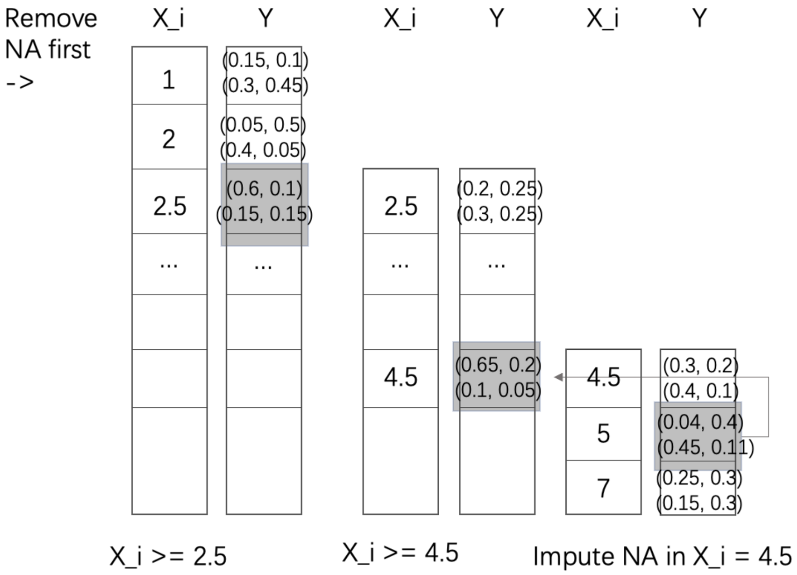


Fig2: Missing Value Imputation procedure

2.1.3 Best Split

DTC model performance is determined by the partition point for splitting. For different data types the partition rule is designed as follows:

1. Categorical variable: simply treated as Bool type for partition due to small size of distinct values.

2. Discrete variable: moving average with two time windows $M(2)$ traversing all distinct values $set(v)_f$ if under threshold else fix $|set(v)_f| =$ threshold.

3. Continuous variable: bin method B(bin number N, outlier handing).
   Parameter bin number N is designed for the following strategy:

   (a) User specified: it's useful when user is confident with prior knwoledge of suitable number for partition, otherwise it would be sensitive to the parameter setting.

   (b) $|n|$ based: Sturges' Rule ($N = log_2^n + 1$) takes information from the number of observation n in feature to decide the suitable bin partition number.

   (c) X based: Scott's Rule($N = \frac{3.5\sigma}{n^{\frac{1}{3}}}$), where σ is the standard deviation of feature f.

(d) Y based: activation function ($N = \delta[\text{Impurity}_f] \cdot |M(2)|$), popularly used activation function e.g. $tanh, logistic$ can be specified.
This method can provide shrinkage of number of bins when impurity is low, which implies that the tree is almostly complete partition of samples. It might increase robustness when the tree is overfitting.
When the tree is underfitting, certain threshold can still be set for preventing rapid shrinkage of partition in avoid of preventing model learn few information from dataset. But this method is supposed to help speed up partition since even when impurity is high, since N is still smaller than $|M(2)|$.

Parameter outlier handing decides if removing outlier before partition.
Whether removing outlier or not affects the distribution of sample numbers fell into each partition points: if we have outliers in raw feature, the distribution is unevenly distributed, otherwise we will have approximately even numbers in each partition bracket. Whether implementing outlier removing is based on if outliers are useful for modeling. It still remains discussion for whether we consider almost even samples partition for bin method.

2.2 Map Reduce

2.2.1 Idea of Map Reduce

The MapReduce strategy divides a complex task into smaller sub-tasks that can be processed independently in parallel across a cluster of computers. The process consists of two main stages: the map stage and the reduce stage.
Map Stage The input dataset is divided into chunks, and each chunk is processed independently by multiple mapper tasks. Mappers perform a specific computation on their assigned data, producing intermediate key-value pairs. The intermediate results are then sorted to group together values with the same key.
Reduce Stage The intermediate results are passed to reducer tasks, which perform an aggregation or computation on the grouped data. The reducers combine and summarize the values associated with each key, generating the final output or result. The MapReduce strategy is fault-tolerant, scalable, and efficient for processing large amounts of data by distributing the workload across multiple machines.

2.2.2 Implementation under DTC

Pipeline The general pipeline of DTC using map reduce is designed as follows(Fig 3):

1. Data partition into s groups by subsampling: $D \to D_i$, $i = 1, 2, \cdots s$.

2. Getting traverse candidate points with map reduce:
Map stage: counter frequency of each value v by feature f in each data subgroups to get: $\{D_i : \{f_j : (v_n, c_m)\}\}$.
Reduce stage: combine the statistics information by summation of counter values

with corresponding value v as keys with respect to feature f as key of v from each data subgroups into original dataset size: $\{D : \{f_j : (v_N, c_M)\}\}$.

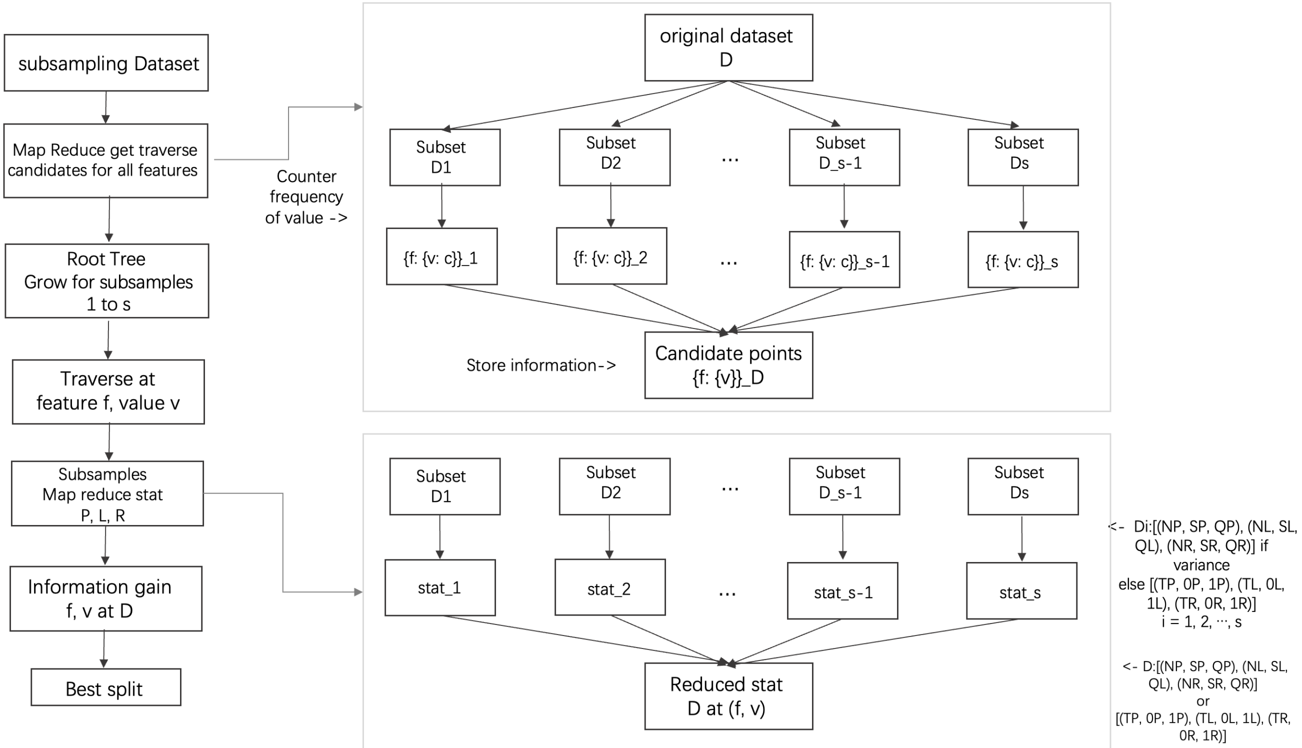
3. Choosing candidate points as traverse based on output from previous step, store key value pairs in memory.

4. Implementing tree structure to grow as follows stopping criteria.

5. Traversing feature f at value v to get (best feature, best value) with map reduce:
   Map Stage: Collect statistics value for impurity function at parent, left child, right child node with respect to f and v in each subgroup: $\{D_i : (P_n, P_s, P_q), (L_n, L_s, L_q), (R_n, R_s, R_q)\}$ if impurity function is variance; $\{D_i : (P_t, P_0, P_1), (L_t, L_0, L_1), (R_t, R_0, R_1)\}$ if impurity function is gini.
   Reduce Stage: Combine all statistic value in corresponding parent, left child, right child node together into original dataset size(assume using variance):
   $\{D : (P_N, P_S, P_Q), (L_N, L_S, L_Q), (R_N, R_S, R_Q)\}$.

6. Get corresponding impurity measure using impurity function $I(N, S, Q)$ in parent, left child, right child node respectively. Compute information gain at f, v.

7. Implementing DTC to find best split.



subsampling Dataset
Map Reduce get traverse candidates for all features
Root Tree Grow for subsamples 1 to s
Traverse at feature f, value v
Subsamples Map reduce stat P, L, R
Information gain f, v at D
Best split
Counter frequency of value ->
original dataset D
Subset D1
Subset D2
...
Subset D_s-1
Subset Ds
{f: {v: c}}_1
{f: {v: c}}_2
{f: {v: c}}_s-1
{f: {v: c}}_s
Store information->
Candidate points {f: {v}}_D
stat_1
stat_2
stat_s-1
stat_s
Reduced stat D at (f, v)
<- Di:[(NP, SP, QP), (NL, SL, QL), (NR, SR, QR)] if variance else [(TP, 0P, 1P), (TL, 0L, 1L), (TR, 0R, 1R)] i = 1, 2, ···, s
<- D:[(NP, SP, QP), (NL, SL, QL), (NR, SR, QR)] or [(TP, 0P, 1P), (TL, 0L, 1L), (TR, 0R, 1R)]

Fig3: Pipeline of map reduce procedure in DTC

2.2.3 Possible scenario for usage

The above strategy is especially suitable when there's limited memory data storage(e.g. 5 GB) in modeling big dataset(e.g. 25 GB). The number of multiprocessor specified(e.g. 5 pools) is to parallel subdataset and only store small number of determinant information statistics(e.g. 5 pairs of statistics value each time) for model prediction when each computing task is independent with each other without information share.

3 Result

3.1 Data explanation

In Fig4 and Fig5 we shows the distribution information of all features. From box plot of numerical variable we notice that there's significant amount of outliers in certain features. From distribution of categorical variable we notice there's imbalanced classes among most features.
We implemented quantile transformation for numerical variable for mitigating impact from outliers. From Fig6 we notice that there's still some amount of outliers even after transformation.

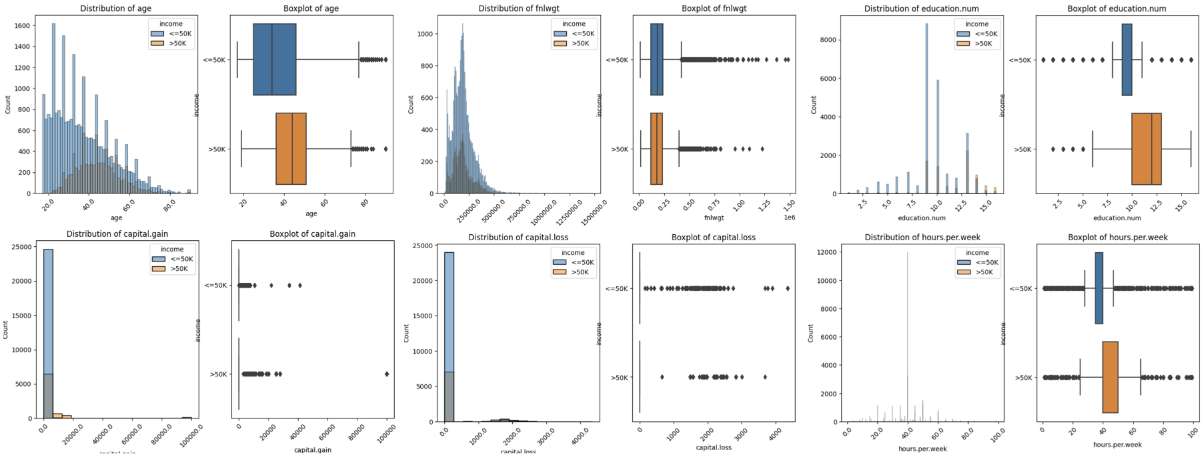


Fig4: Data visualization of numerical variable in Adult dataset

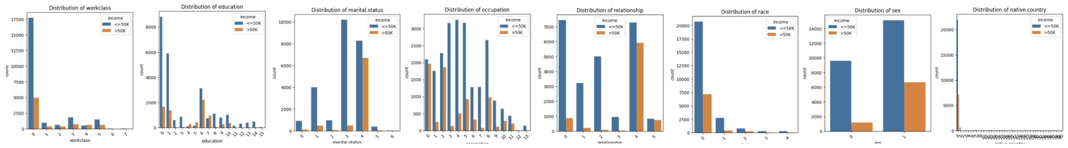


Fig5: Data visualization of categorical variable in Adult dataset

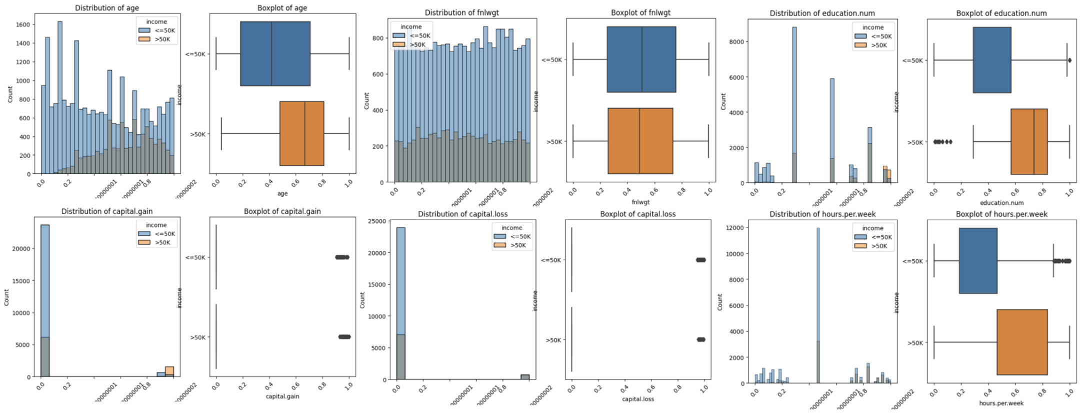


Fig6: Data visualization of numerical variable after quantile transformation

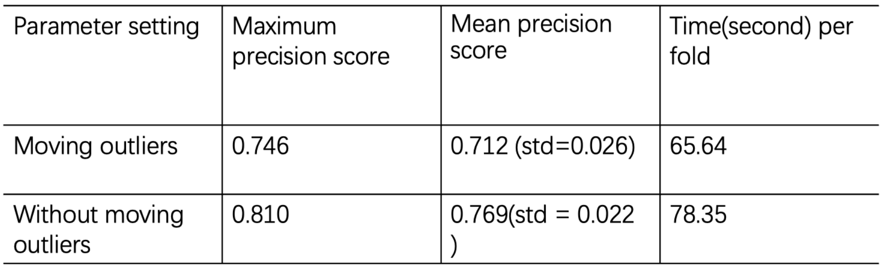
3.2 Comparison of model performance

In this project we specified Repeated stratified K fold as cross validation for DTC, taking categorical data as bool type for splitting.

Comparison of outlier handling

- model parameter setting:

$n_{\text{splits}} = 10, n_{\text{repeats}} = 2, \text{traverse threshold} = 25, \text{min samples split} = 1850, \text{max depth} = 11, \text{info method} = \text{'variance'}, \text{na method} = \text{'recursive'}, \text{bins} = \text{'tanh'}$



Parameter setting	Maximum precision score	Mean precision score	Time(second) per fold
Moving outliers	0.746	0.712 (std=0.026)	65.64
Without moving outliers	0.810	0.769(std = 0.022)	78.35

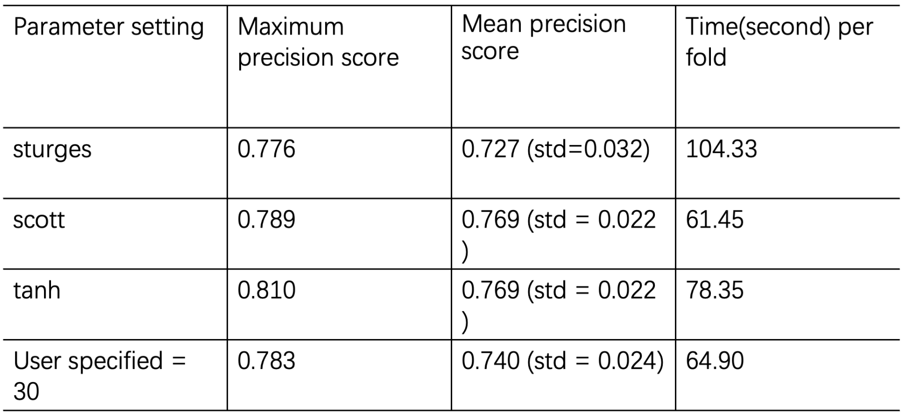
Fig7: Comparison of outlier handling

Comparison of bin number generation criteria

3.2.1 DTC model performance

- model parameter setting:

$n_{\text{splits}} = 10, n_{\text{repeats}} = 1, \text{traverse threshold} = 25, \text{min samples split} = 1850, \text{max depth} = 11, \text{info method} = \text{'variance'}, \text{na method} = \text{'recursive'}, \text{outlier} = \text{False}$



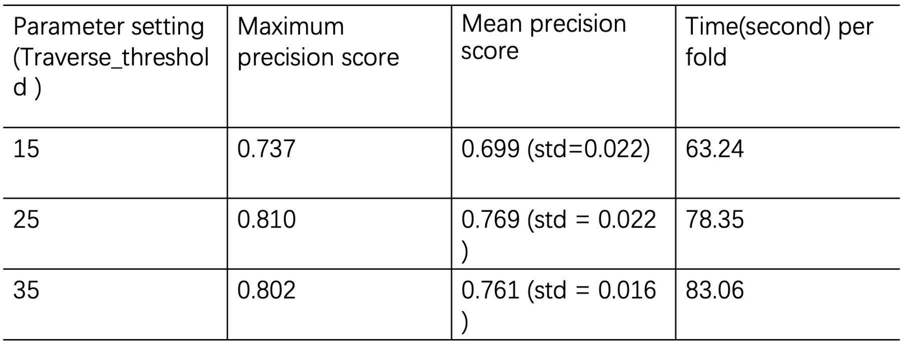
Parameter setting	Maximum precision score	Mean precision score	Time(second) per fold
sturges	0.776	0.727 (std=0.032)	104.33
scott	0.789	0.769 (std = 0.022)	61.45
tanh	0.810	0.769 (std = 0.022)	78.35
User specified = 30	0.783	0.740 (std = 0.024)	64.90

Fig8: Comparison of bin number generation criteria

Comparison of traverse threshold

- model parameter setting:

$n_{\text{splits}} = 10, n_{\text{repeats}} = 1, \text{min samples split} = 1850, \text{max depth} = 11, \text{info method} = \text{'variance'}, \text{na method} = \text{'recursive'}, \text{bins} = \text{'tanh'}, \text{outlier} = \text{False}$



Parameter setting (Traverse_threshol d)	Maximum precision score	Mean precision score	Time(second) per fold
15	0.737	0.699 (std=0.022)	63.24
25	0.810	0.769 (std = 0.022)	78.35
35	0.802	0.761 (std = 0.016)	83.06

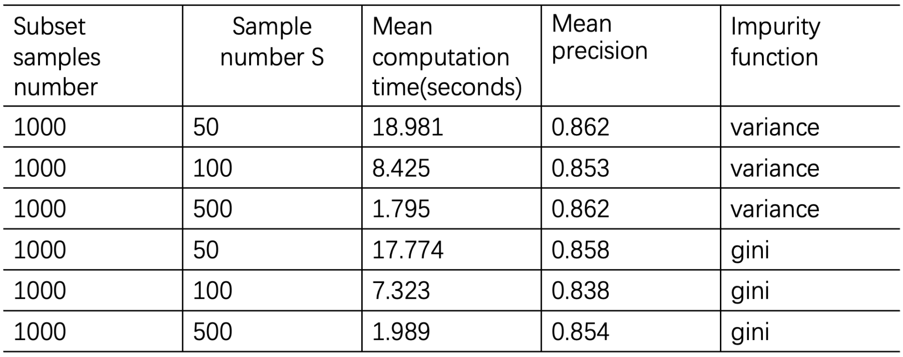
Fig9: Comparison of traverse threshold

3.2.2 DTCMR model performance

We used train test split method in DTC with map reduce setting. The general parameter setting is as follows:

$\text{min samples split} = 600, \text{max depth} = 11, \text{na method} = \text{'recursive'}, \text{bins} = 15, \text{traverse threshold} = 10, \text{outlier} = \text{False}$

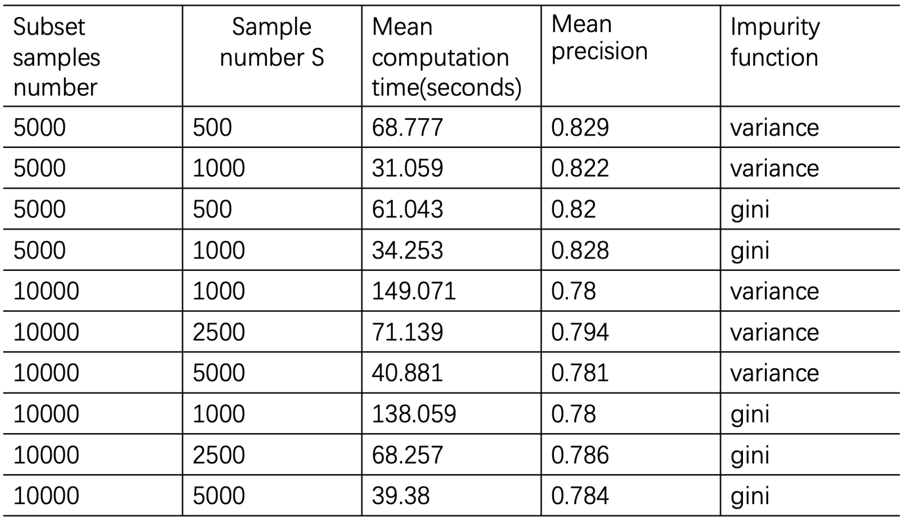
Comparison of DTCMR in 1000 subsamples



Subset samples number	Sample number S	Mean computation time(seconds)	Mean precision	Impurity function
1000	50	18.981	0.862	variance
1000	100	8.425	0.853	variance
1000	500	1.795	0.862	variance
1000	50	17.774	0.858	gini
1000	100	7.323	0.838	gini
1000	500	1.989	0.854	gini

Fig10: Comparison of DTCMR in 1000 subsamples

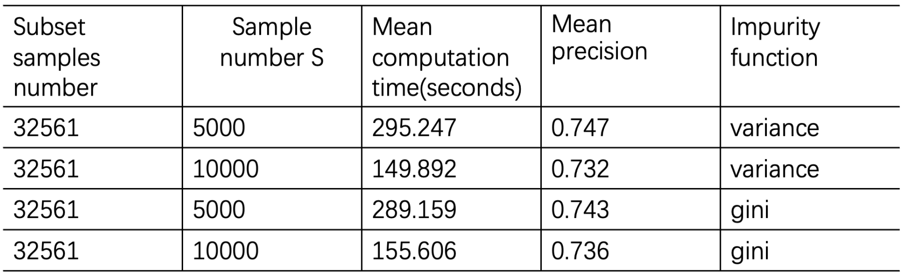
Comparison of DTCMR in 5000 and 10000 subsamples



Subset samples number	Sample number S	Mean computation time(seconds)	Mean precision	Impurity function
5000	500	68.777	0.829	variance
5000	1000	31.059	0.822	variance
5000	500	61.043	0.82	gini
5000	1000	34.253	0.828	gini
10000	1000	149.071	0.78	variance
10000	2500	71.139	0.794	variance
10000	5000	40.881	0.781	variance
10000	1000	138.059	0.78	gini
10000	2500	68.257	0.786	gini
10000	5000	39.38	0.784	gini

Fig11: Comparison of DTCMR in 5000 and 10000 subsamples

Comparison of DTCMR in raw dataset



Subset samples number	Sample number S	Mean computation time(seconds)	Mean precision	Impurity function
32561	5000	295.247	0.747	variance
32561	10000	149.892	0.732	variance
32561	5000	289.159	0.743	gini
32561	10000	155.606	0.736	gini

Fig11: Comparison of DTCMR in raw dataset

4 Conclusion

If the code is appropriately implemented, we can draw conclusions from the following:

1. For DTC, outlier gives useful information when considering into data partition due to significant amount size in Adult dataset.
2. For DTC splitting candidate partition in continuous variable, impurity information is informative for bin method compared with others.
3. For DTC partitioning continuous variable, increasing partition of candidate points to traverse lead more accurate precision if tree is underfitting, but there's trade off between computational time.
4. For DTCMR, map reduce strategy helps solving limited memory issue by compromising model precision due to local optimal result from subsample information and increase computational time.

5 Online Resources

The sources for the work are available via

- GitHub Link