The utility of merging k-nearest neighbors and classification tree

Tor Nitayanont

Joint work with Dorit Hochbaum

Department of Industrial Engineering and Operations Research, UC Berkeley

September 2021

Berkeley
UNIVERSITY OF CALIFORNIA

Outline

- **Introduction: CART and kNN classifier**
- **Motivation**
- **kNN-under-tree**
- **Experimentation**
  - models
  - datasets
  - results
- **Future Directions**

Introduction: CART and kNN classifier

CART

- easy to interpret, can handle both continuous, categorical and even missing variables without a lot of data preprocessing
- does not rely on the assumption about the data
- automatically perform feature selection (or compute feature importances)
- can overfit, sensitive to outliers/missing data
- does not capture non-linear relationship/ non-linear boundary between samples with different labels well

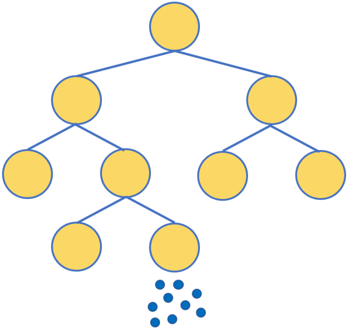
kNN classifier

- higher computational cost than CART
- deal with non-linear boundary well
- also does not rely on the assumption about the data

Motivation

- **CART** separates the data well at a higher level (the first few depths of the tree).
- However, as the tree grows deeper and the number of samples in each branch becomes smaller, the boundaries between samples with different labels become more intricate (more non-linear), which is difficult for CART to handle.
- **kNN classifier** can help draw the boundaries at the local level.
- **kNN classifier** is a lazy learning method that incurs high computational cost, especially when the dataset grows large. Its performance also depends largely on the choice of distance metrics.
- **CART** can reduce the computational cost by limiting the number of neighboring labeled samples to be considered for each unlabeled samples. It also evaluates *the feature importance*, which can be utilized to make the distance metric reflect the (dis)similarity between samples more accurately.

kNN-under-tree



- build a classification tree using the labeled samples
  (parameter: the minimum number of samples required to be at a leaf node)
- for each leaf node, we build a kNN classifer using only labeled samples in that leaf
- we use the kNN classifier in each leaf node to predict the labels for the unlabeled queries that are put in the same leaf node

Berkeley
UNIVERSITY OF CALIFORNIA

Experimentation

We test our proposed model and its variants, along with baseline models on a **binary classification** problem on multiple synthetic datasets. Parameters in the models are selected by performing 4-fold cross validation.

Two baseline models

- **CART**
  - impurity function: Gini impurity
  - fraction of minimum number of samples required to be at leaf node: $[0.001, 0.002, 0.005, 0.01]$
- **kNN classifier**
  - k or the number of neighbors: $[4, 8, 12, 16, 20]$
  - weight: *inverse of distance*
    $label(i) = \mathbb{1}(\sum_{j \in N(i), label(j)=1} \frac{1}{dist(x_i, x_j)} \geq \sum_{j \in N(i), label(j)=0} \frac{1}{dist(x_i, x_j)})$ where $N(i)$ is the set of k nearest samples of the sample i
  - distance: Euclidean distance

Berkeley
UNIVERSITY OF CALIFORNIA

Proposed model

- **kNN-under-tree**
  - fraction of minimum number of samples required to be at leaf node: $[0.001, 0.002, 0.005, 0.01]$
  - k or the number of neighbors: $[4, 8, 12, 16, 20]$
  - weight: *inverse of distance*
  - distance: Euclidean distance

Two variants using CART's feature importance (FI)

- **kNN classifier with FI**
- **kNN-under-tree with FI**

Berkeley
UNIVERSITY OF CALIFORNIA

Two variants using CART's Feature Importance (FI)

- **kNN classifier with FI**
- **kNN-under-tree with FI**

Feature Importance (FI)

- The feature importance of a feature is the total impurity reduction brought by that feature.
- $FI(m) = \sum_{i \in V(m)} ImpRed(i)$ where $V(m)$ is the set of nodes that split on feature m
- $ImpRed(i) = |node(i)|Imp(i) - |left(i)|Imp(left(i)) - |right(i)|Imp(right(i))$
- In this work, we use the Gini Impurity.

FI-weighted distance

- normalized $FI(m)$ or $nFI(m) = FI(m) / \sum_{l=1}^{M} FI(l)$
- $dist(x_i, x_j) = (\sum_{m=1}^{M} nFI(m) \cdot (x_{im} - x_{jm})^2)^{\frac{1}{2}}$

Berkeley
UNIVERSITY OF CALIFORNIA

Two variants using CART’s Feature Importance (FI)

- **kNN classifier with FI**
  - similar to the standard kNN classifier except that the distance is computed using the FI-weighted distance
  - a CART needs to be built so that the feature importance can be comouted
  - we use the same set of parameters introduced earlier to build a CART
- **kNN-under-tree with FI**
  - similar to the kNN under tree method except that the distance in each kNN classifier is computed using the FI-weighted distance

Datasets

Synthetic datasets In each experiment, we test our models on 20 datasets, which are generated using each combination of the following parameters.

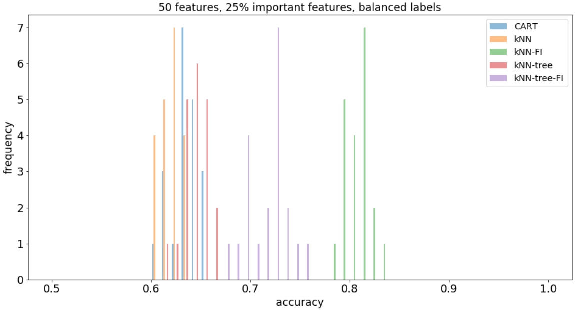
- number of features: 50 and 200
- fraction of relevant features: $0.25, 0.5$ and 0.75
- fraction of samples with label 0 and label 1: $0.5 : 0.5$ and $0.7 : 0.3$

Each dataset contains 10000 samples, which are split into 7000 labeled samples and 3000 unlabeled samples.

Berkeley
UNIVERSITY OF CALIFORNIA

Results

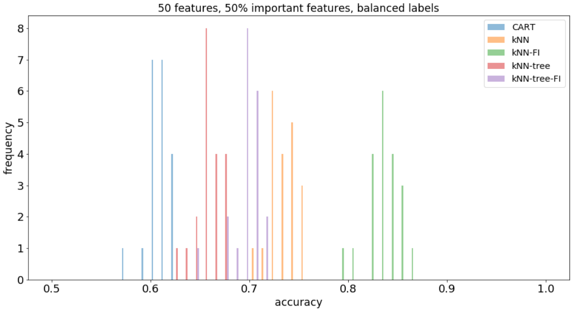
Histogram of the accuracy of all models on 20 datasets where 13 features out of 50 features are important features and the labels in the dataset are balanced.



Berkeley
UNIVERSITY OF CALIFORNIA

Results

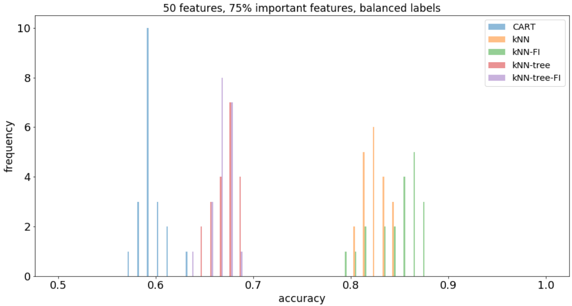
Histogram of the accuracy of all models on 20 datasets where 25 features out of 50 features are important features and the labels in the dataset are balanced.



Berkeley
UNIVERSITY OF CALIFORNIA

Results

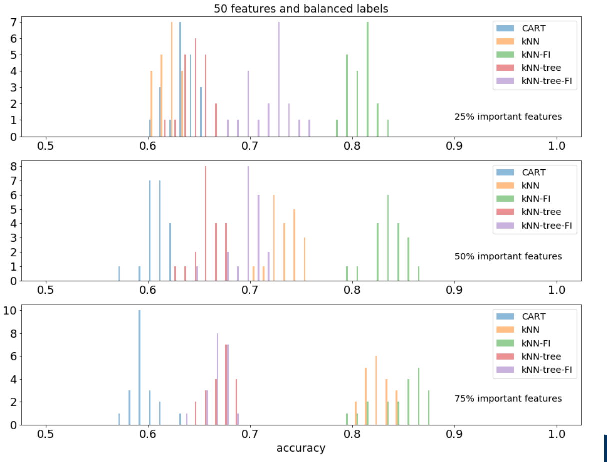
Histogram of the accuracy of all models on 20 datasets where 38 features out of 50 features are important features and the labels in the dataset are balanced.



Berkeley
UNIVERSITY OF CALIFORNIA

Results

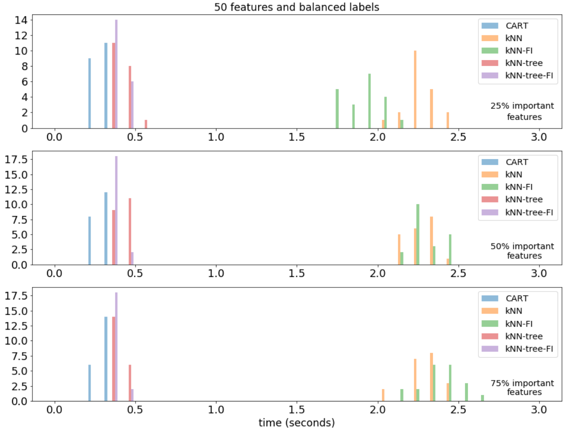
Histograms of the models' accuracy on datasets with 10000 samples, 50 features and balanced labels.



Berkeley
UNIVERSITY OF CALIFORNIA

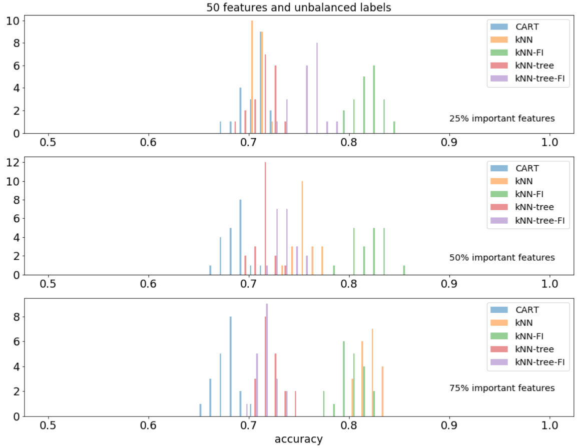
Results

Histograms of the computation time of the models on datasets with 10000 samples, 50 features and balanced labels.



Results

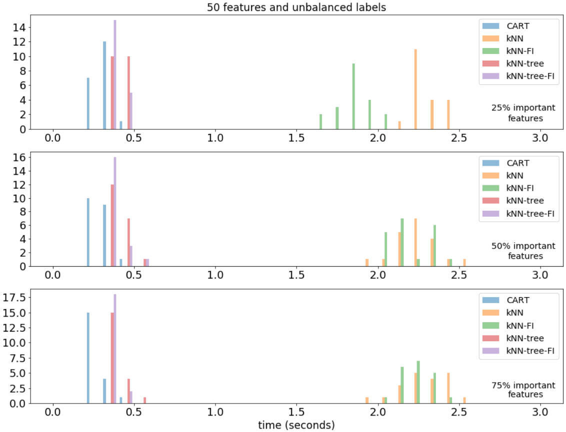
Histograms of the models' accuracy on datasets with 10000 samples, 50 features and **unbalanced** labels.



Berkeley
UNIVERSITY OF CALIFORNIA

Results

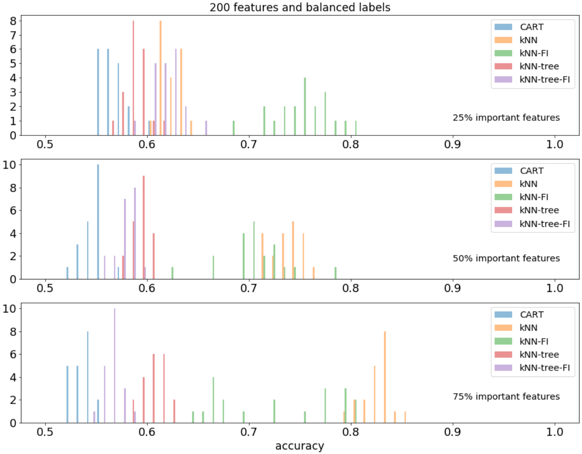
Histograms of the computation time of the models on datasets with 10000 samples, 50 features and **unbalanced** labels.



Berkeley
UNIVERSITY OF CALIFORNIA

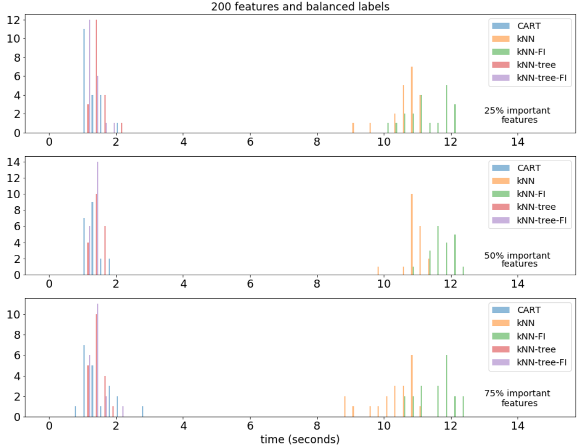
Results

Histograms of the models' accuracy on datasets with 10000 samples, **200 features** and balanced labels.



Results

Histograms of the computation time of the models on datasets with 10000 samples, **200 features** and balanced labels.



Berkeley
UNIVERSITY OF CALIFORNIA

Conclusion

- **kNN-under-tree with FI** provides varying accuracy improvement over **CART**. It gives an impressive improvement when the data is noisy. As the data becomes less noisy, the enhancement of **knn-under-tree with FI** is less noticeable, but it still achieves higher accuracy than CART.
- In terms of computation time, the computation time of **kNN-under-tree** and **kNN-under-tree with FI** are consistently low regardless of the noise level.
- **kNN under tree with FI** is an interesting option when we have limited time budget and we know that the data contains many noisy features.
- **kNN with FI** achieves the highest accuracy in most experiments. However, the computational cost is significantly expensive.
- The use of **CART's feature importance** in kNN enhances the accuracy of the model. However, when the data becomes more high-dimensional, the improvement decreases. This is partly due to the sparseness of the feature importance.

Future directions

- Random Forest
  - improves the accuracy
  - provides a more accurate / less sparse feature importance, which might be able to handle the cases where data is less noisy or more high-dimensional
- Sparse Computation framework
  - finds pairs of samples that are similar enough and should be considered as potential neighbors
  - reduces the computation time significantly
- Hyperparameter selection
  - finds the appropriate values of tree parameters so that it works well with kNN

Results on real data

Dataset 1

m	CART	kNN	kNN-FI	kNN-tree	kNN-tree-FI
1	0.8194 (1)	0.8059 (5)	0.8136 (2)	0.8116 (4)	0.8132 (3)
2	0.8194 (1)	0.8059 (5)	0.8138 (2)	0.8116 (4)	0.8132 (3)
3	0.8238 (1)	0.8148 (2)	0.8133 (5)	0.8134 (4)	0.8148 (2)
4	0.8172 (1)	0.8106 (5)	0.8149 (2)	0.8123 (3)	0.8122 (4)
5	0.8173 (1)	0.8088 (5)	0.8106 (4)	0.8116 (3)	0.8128 (2)
avg rank	1	4.4	3	3.6	2.8

Berkeley
UNIVERSITY OF CALIFORNIA

Results on real data

Dataset 2

m	CART	kNN	kNN-FI	kNN-tree	kNN-tree-FI
1	0.8539 (1)	0.8260 (5)	0.8463 (4)	0.8505 (2)	0.8501 (3)
2	0.8580 (1)	0.8332 (5)	0.8494 (4)	0.8561 (2)	0.8543 (3)
3	0.8517 (1)	0.8337 (5)	0.8460 (3)	0.8453 (4)	0.8503 (2)
4	0.8525 (1)	0.8354 (5)	0.8509 (2)	0.8462 (4)	0.8508 (3)
5	0.8561 (1)	0.8324 (5)	0.8513 (3)	0.8515 (2)	0.8505 (4)
avg rank	1	5	3.2	2.8	3

Results on real data

Dataset 3

m	CART	kNN	kNN-FI	kNN-tree	kNN-tree-FI
1	0.8837 (5)	0.9707 (2)	0.9772 (1)	0.9506 (3)	0.9473 (4)
2	0.8873 (5)	0.9738 (2)	0.9782 (1)	0.9467 (3)	0.945 (4)
3	0.8777 (5)	0.9717 (2)	0.9755 (1)	0.9503 (3)	0.9503 (3)
4	0.884 (5)	0.9675 (2)	0.9753 (1)	0.9467 (4)	0.9513 (3)
5	0.8887 (5)	0.9677 (2)	0.9757 (1)	0.9473 (4)	0.9512 (3)
avg rank	5	2	1	3.4	3.4

Berkeley
UNIVERSITY OF CALIFORNIA

Results on real data

Dataset 4

m	CART	kNN	kNN-FI	kNN-tree	kNN-tree-FI
1	0.8209 (5)	0.8649 (4)	0.9193 (1)	0.8809 (2)	0.8767 (3)
2	0.8107 (5)	0.8634 (4)	0.92 (1)	0.8713 (2)	0.8834 (2)
3	0.8164 (5)	0.8700 (4)	0.9205 (1)	0.8914 (2)	0.8914 (2)
4	0.8145 (5)	0.8691 (4)	0.9193 (1)	0.8784 (3)	0.8948 (2)
5	0.8148 (5)	0.8727 (4)	0.9207 (1)	0.8848 (3)	0.8944 (2)
avg rank	5	4	1	2.4	2.2

Results on real data

Rank of the models from all datasets

m	CART	kNN	kNN-FI	kNN-tree	kNN-tree-FI
1	1	4.4	3	3.6	2.8
2	1	5	3.2	2.8	3
3	5	2	1	3.4	3.4
4	5	4	1	2.4	2.2
avg	3	3.85	2.05	3.05	2.85

Berkeley
UNIVERSITY OF CALIFORNIA