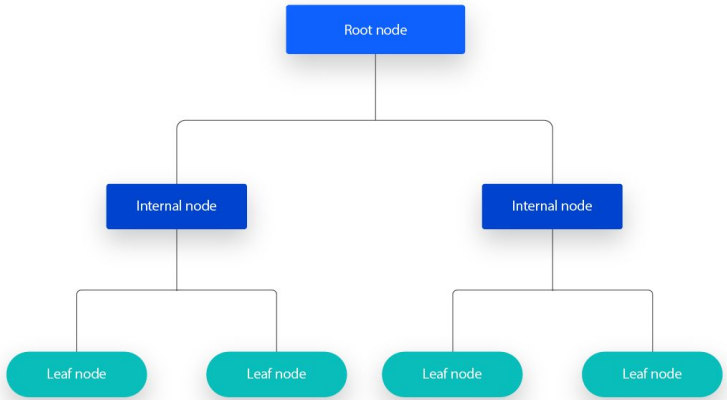
Random KNN

Arya Bharath & James Wright

Introduction

- Drawn to ensemble learning method
- Random Forest uses bagging with Decision Trees
- Were Decision Trees the best model for RF?
- Random KNN pros
    - Clustered, nonlinear data
    - Reducing Decision Tree instability



Related Work

- Limited Random KNN research
- Shengqiao Li introduced RKNN and its hyperparameters
  - r: Number of KNNs used in prediction
  - k: Number of nearest neighbors, 1-15
  - m: Features per RKNN, sqrt(dimensionality)

Datasets & Features

```
configurations = [
    {"n_samples": 100, "n_features": 5, "n_clusters": 3, "cluster_std": 1.0},
    {"n_samples": 200, "n_features": 5, "n_clusters": 4, "cluster_std": 1.5},
    {"n_samples": 300, "n_features": 6, "n_clusters": 5, "cluster_std": 0.8},
    {"n_samples": 150, "n_features": 4, "n_clusters": 2, "cluster_std": 2.0},
    {"n_samples": 250, "n_features": 3, "n_clusters": 3, "cluster_std": 0.5},
    {"n_samples": 400, "n_features": 7, "n_clusters": 6, "cluster_std": 1.2},
    {"n_samples": 180, "n_features": 4, "n_clusters": 2, "cluster_std": 1.8},
    {"n_samples": 220, "n_features": 5, "n_clusters": 4, "cluster_std": 0.7},
    {"n_samples": 350, "n_features": 6, "n_clusters": 5, "cluster_std": 1.1},
    {"n_samples": 100, "n_features": 3, "n_clusters": 2, "cluster_std": 1.5},
]
```

- Generated artificial datasets using sklearn
  - Varied samples, features, clusters, standard devs
- No need for discretization or normalization
- Examples
  - (100 samples, 5 features, 3 clusters, std 1.0)
  - (300 samples, 6 features, 5 clusters, std 0.8)

Methods

- Random Forest (RF)
  - Uses bagging to create DTs
  - Features selected randomly per tree
  - Predictions made by aggregating votes
- K-Nearest-Neighbors (KNN)
  - Classifies based on distance to neighbors
  - Different distance metrics (Euclidean, Manhattan)
- Random KNN (RKNN)
  - RF but with KNN instead of DT on sub-samples

Experimental Setup

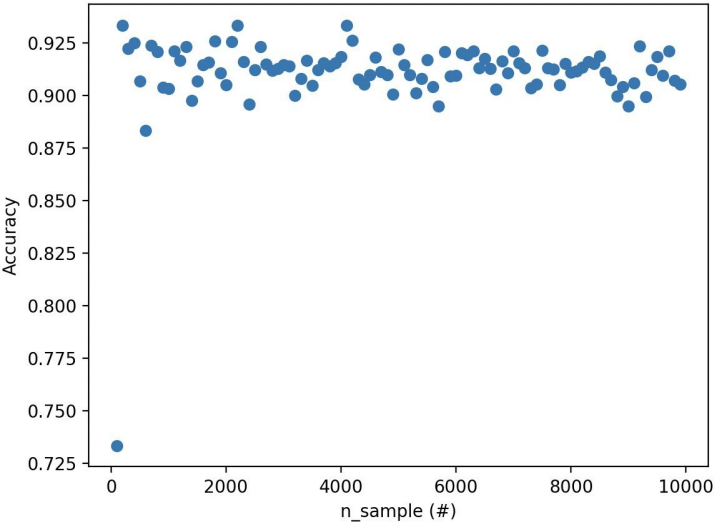
- Generated 10 synthetic datasets
- Compared RKNN and RF accuracy
- Used square root of sample size for k in RKNN

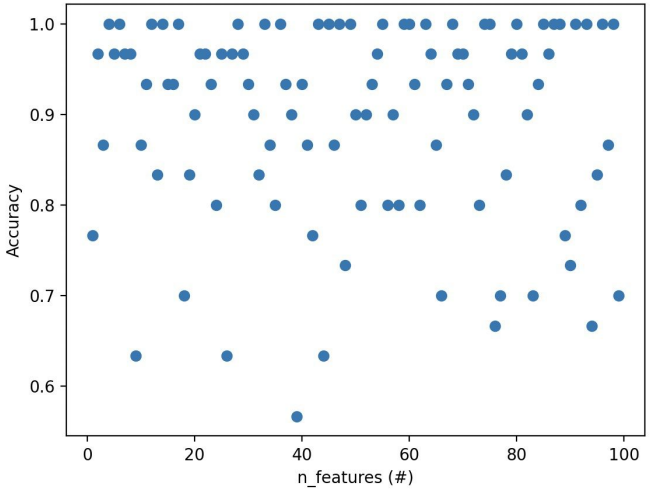
Results

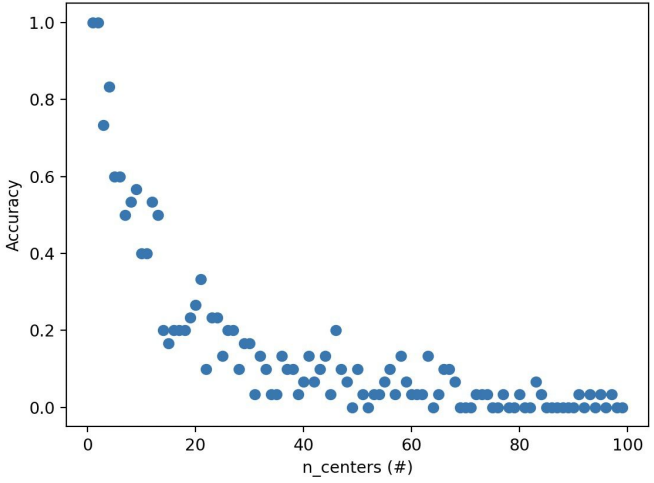
Sample #	n_samples	n_features	n_clusters	cluster_std	RKNN	RF
1	100	5	3	1.00	0.63	1.00
2	200	5	4	1.5	1.00	1.00
3	300	6	5	0.8	0.99	1.00
4	150	4	2	2.0	1.00	1.00
5	250	3	3	0.5	1.00	0.97
6	400	7	6	1.2	0.99	1.00
7	180	4	2	1.8	1.00	1.00
8	220	5	4	0.7	1.00	1.00
9	350	6	5	1.1	1.00	1.00
10	100	3	2	1.5	1.00	1.00

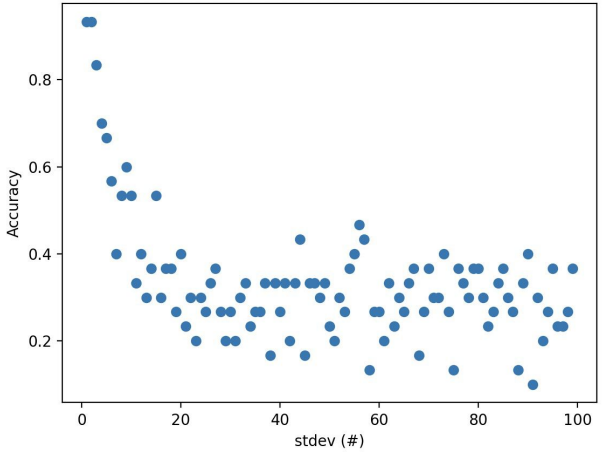
Results (cont.)

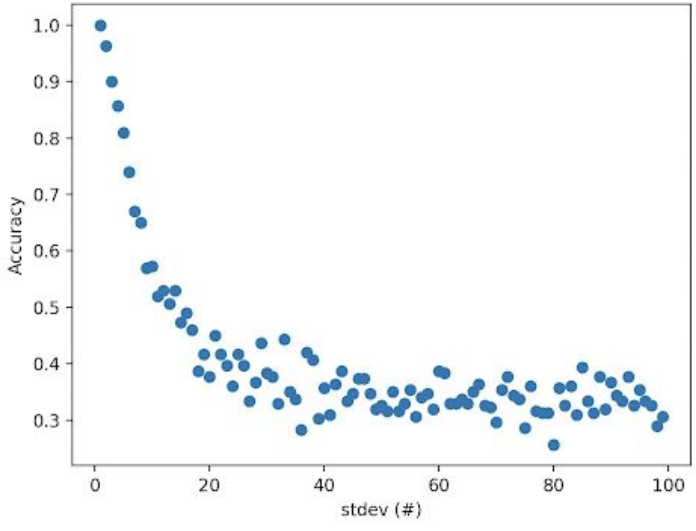
- RKNN performs similarly to RF but has inconsistencies.
- Most datasets showed RKNN accuracy close to RF.
- One dataset (Sample #1) showed significant underperformance.

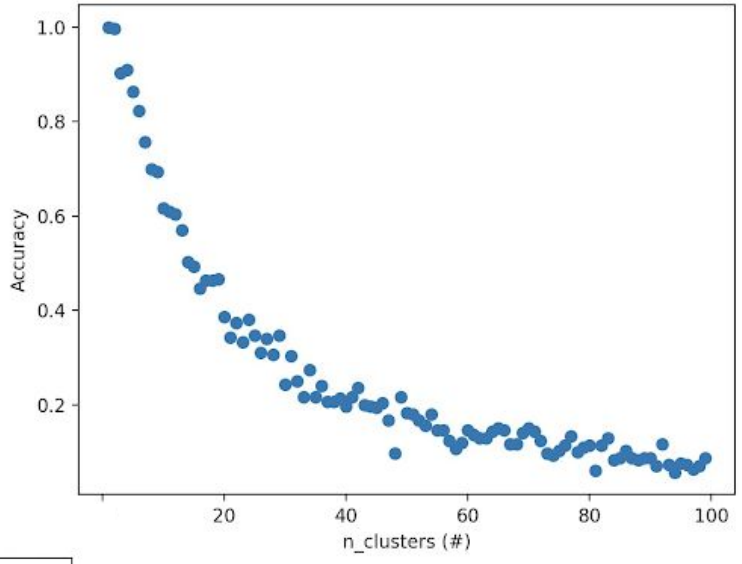

Accuracy
n_sample (#)


Accuracy
n_features (#)

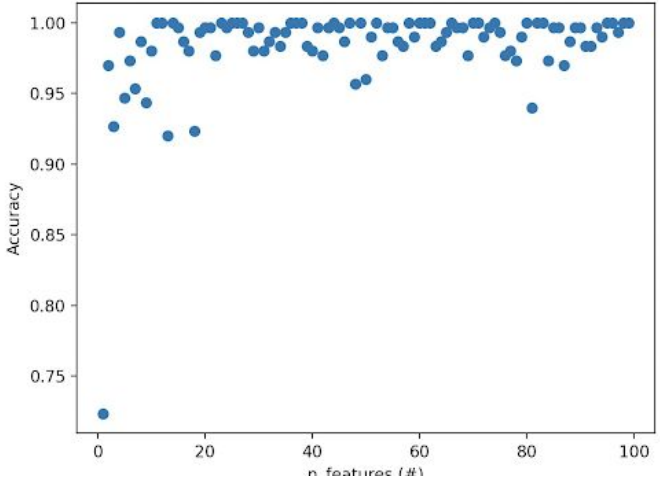

Accuracy
n_centers (#)

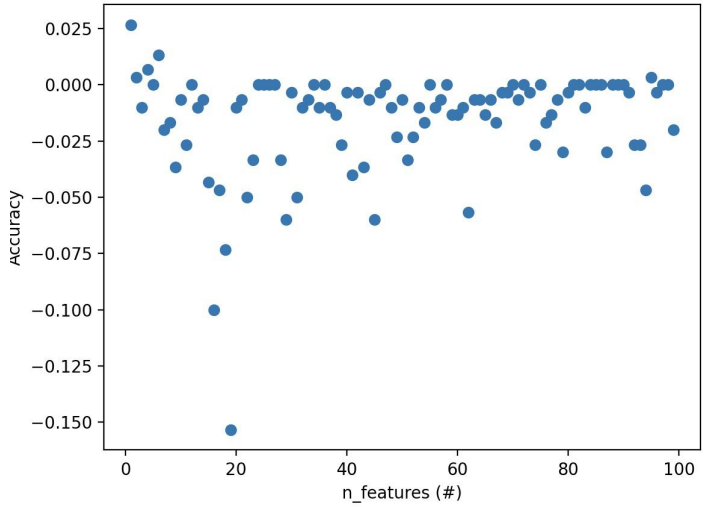

Accuracy
stdev (#)

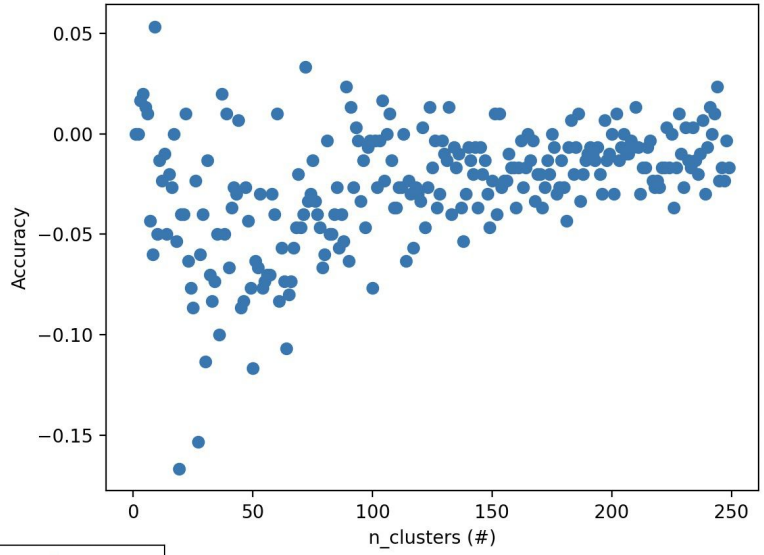




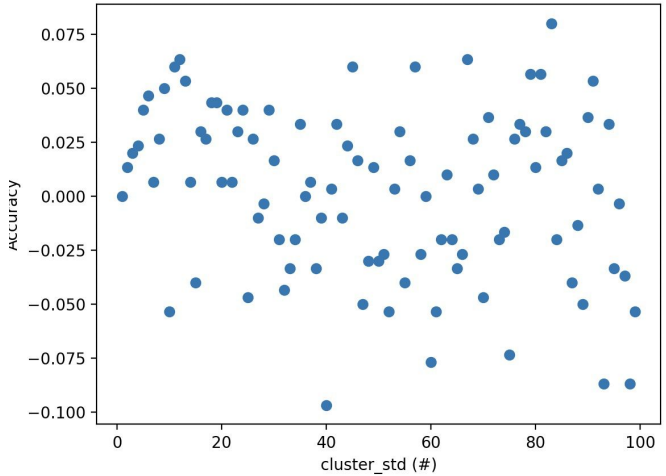
n_sample = 1000







- RKNN - RF accuracy
- Positive means RKNN >
- S = 0.21333333333337



Discussion

- **Key Findings**:
  - RKNN generally performs well but is sensitive to k selection.
  - RF is more stable across datasets.
  - RKNN can underperform if k is poorly chosen.
- **Potential Improvements**
  - Better hyperparameter tuning (r, k, m).
  - Analyzing RKNN performance on real-world datasets.

Conclusions & Future Work

- RKNN is a viable alternative to RF in certain datasets
- Performance variability suggests need for future testing
- Extensive testing on diverse datasets
- Investigating optimal k selection strategies
- Exploring hybrid models combining RF and RKNN