ENSEMBLES OF MODELS

BAGGING & BOOSTING

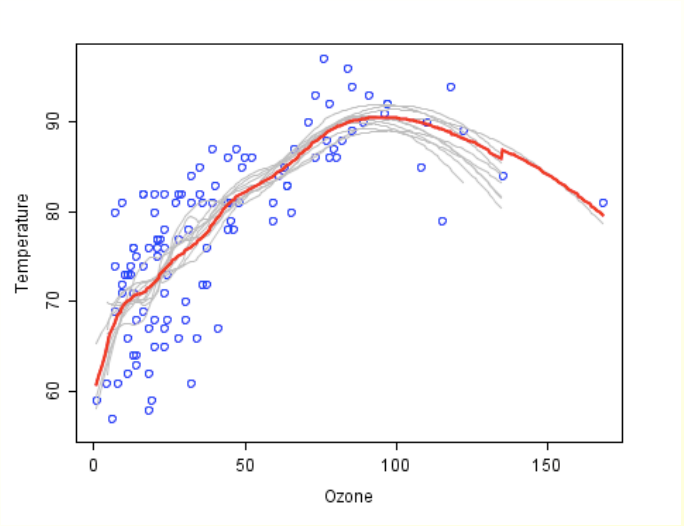
Ensembles of models

- **Ensemble** = collection of models (collection of **base models**)
- Main types:
  - Bagging: base models are trained in parallel
    - Random Forests: base model = tree
  - Boosting: base models are trained sequentially
    - Gradient boosting: base model = regression tree
  - Stacking: base models are organized hierarchically

BAGGING

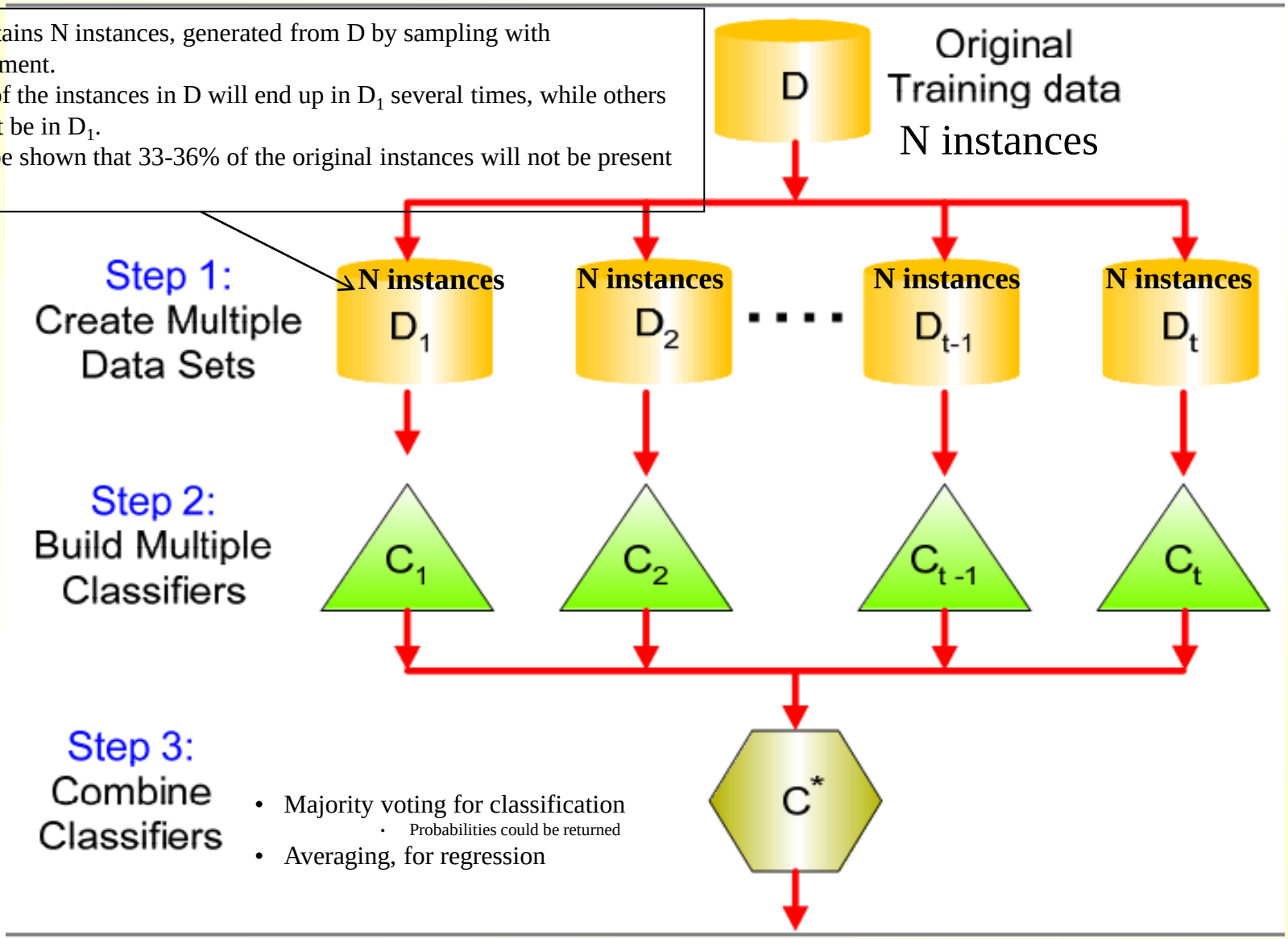
Bagging (Bootstrap aggregating)

- Motivation: some training methods tend to overfit the training set (unstable methods, high variance methods)
  - Unstable: neural networks, trees (with large max depth), ...
  - Stable: linear, KNN, Support Vector Machines, ...
- Bagging strategy: given that overfitting to instances with noise/biases in the training set happens randomly, do:
  - train several models with different training samples and average them



Bagging (Bootstrap aggregating)

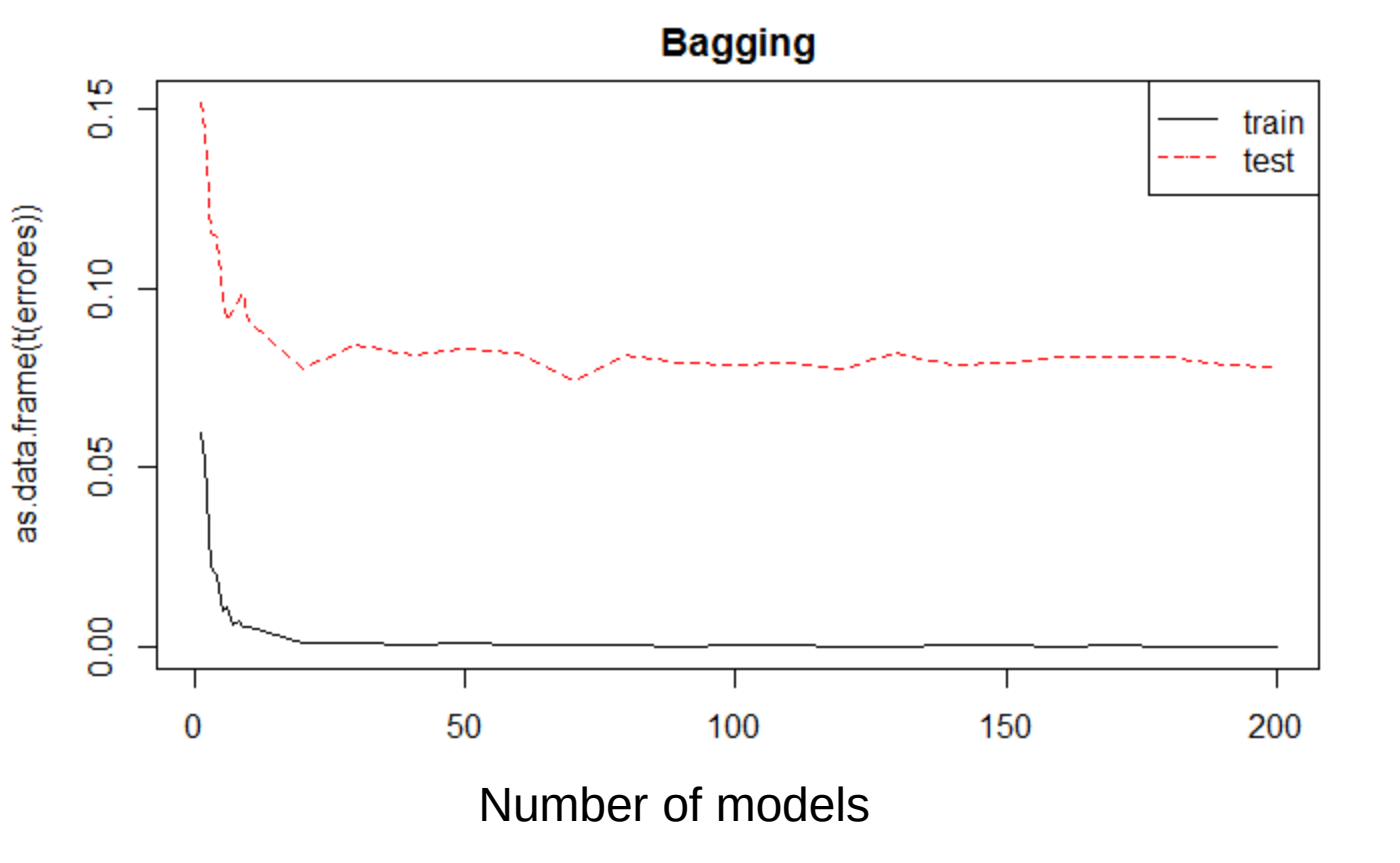
- D_1 contains N instances, generated from D by sampling with replacement.
- Some of the instances in D will end up in D_1 several times, while others will not be in D_1.
- It can be shown that 33-36% of the original instances will not be present in D_1



- Majority voting for classification
  - Probabilities could be returned
- Averaging, for regression

Bagging and error

Typically, the more models, the better (lower) the error. Bagging does not overfit by adding more base models.



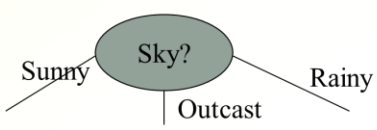
(Random Forest)

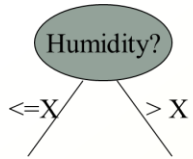
Randomization

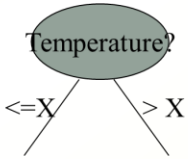
- Randomization is another way of creating an ensemble of models (different to creating several training sets).
- Some training methods are stochastic: everytime they are run, they generate a different model (even if the training set is the same)
  - E.g. : neural networks, because initial weights / parameters are random.
- But deterministic methods (such as trees) can be modified so that they become stochastic.
- Random Forests modify the tree training algorithm so that it becomes stochastic. RF use both several bootstrap samples and randomization.

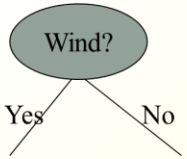
Random Forests (RF)

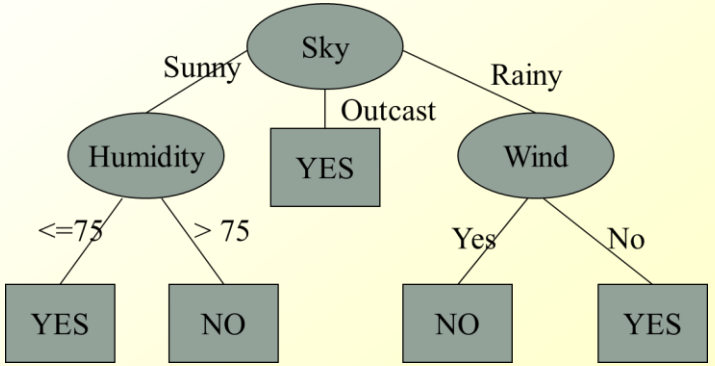
- RF = **Bagging** with decision trees. It uses:
  - Random sampling with replacement
  - Randomization: when choosing an attribute for a node, the best is not selected. Rather, the best is selected from a random subset of m attributes. For instance, if there are M=16 attributes and m=4, then 4 attributes are randomly chosen, and then the best of the four is selected (m is usually called *mtry*).
    - Typically m=sqrt(M) for classification and m = M/3 for regression.
    - If m == M, then RF = Bagging











Random Forests

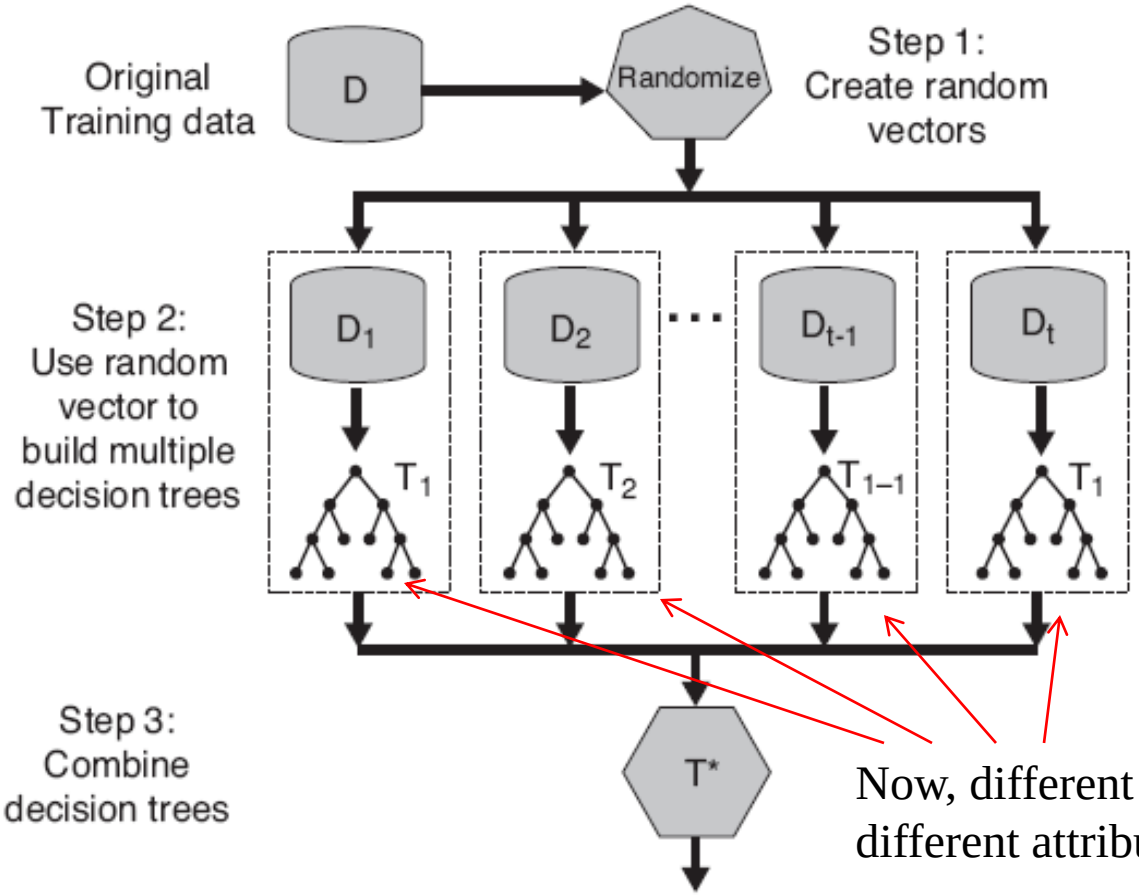


Figure 5.40. Random forests.

Results of Random Forests

- It is quite common that RF outperform single decision trees

Test set misclassification error (%)

Data set	**Forest**	**Single tree**
Breast cancer	2.9	5.9
Ionosphere	5.5	11.2
Diabetes	24.2	25.3
Glass	22.0	30.4
Soybean	5.7	8.6
Letters	3.4	12.4
Satellite	8.6	14.8
Shuttle $\times 10^3$	7.0	62.0
DNA	3.9	6.2
Digit	6.2	17.1

Random Forests. Out of bag estimation (OOB)

- We already know that in order to estimate the performance, it is necessary to use a different test set than the one used for training
- This is typically done by means of train/test (also called hold-out) and also by means of crossvalidation
- But RF are able to provide a success rate estimation without setting aside a test set
- It takes advantage that some instances are not present in some of the training sets D_i. Given that they have not been used for training, they can be used for testing the associated tree T_i
- The error of instance *x* will be computed by using the trees where *x* was not used for training

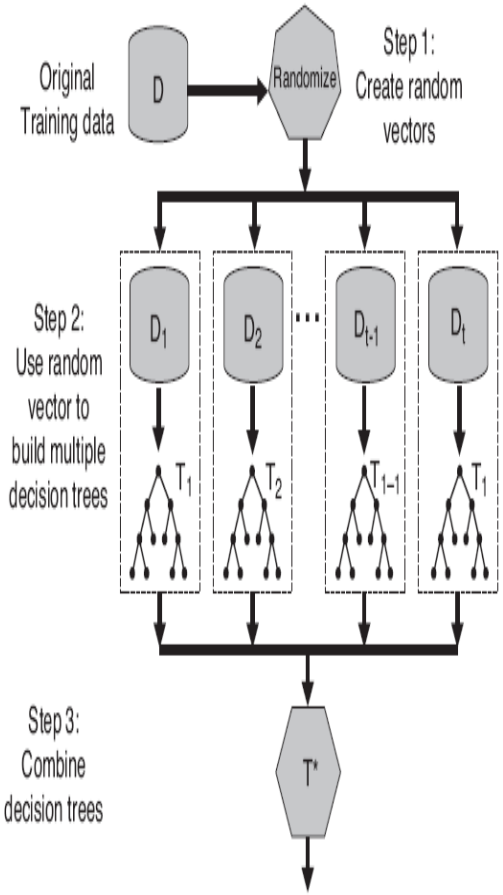


Figure 5.40. Random forests.

https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

Random Forests. Summary

- Only two new (hyper-)parameters: number of trees in the ensemble (*k*) and size of the attribute subset (*m or mtry*)
- Usually it outperforms single decision trees
- Faster than standard Bagging (because only *mtry* << M attributes are considered for each node)
- We get Out-of-bag estimation (estimation of future performance) and attribute ranking, for free
- It is able to return probabilistic predictions: if 90 trees say “+” and 10 say “-”, probability of “+” is 0.9

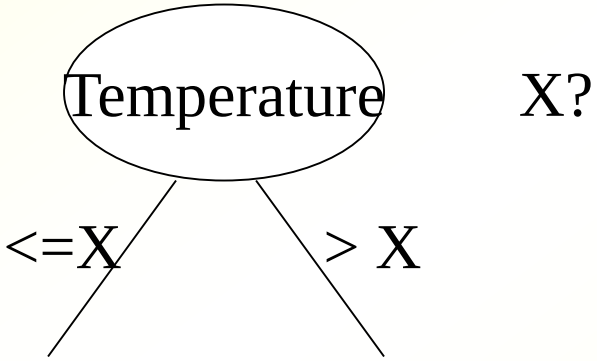
Extremely randomized trees (Extra-trees)

- Motivation:
  - Increase randomization of base models
  - Reduce training time: most of the training time in trees is spent when selecting the threshold for numeric attributes
- Extra-trees work similarly to RF, but:
  - the whole training set is used for all base models (sampling with replacement is NOT used)
  - For every decision tree node, *mtry* attributes are chosen randomly (out of M), but:
    - If the attribute is numeric, the threshold is chosen randomly.
    - If the attribute is categorical, a binary decision is selected randomly:
      - E.g. : if attribute A has values {a,b,c,d,e}, a binary decision could be:
        - if(A ∈ {c,e}) left else right.

Reminder: what to do for continuous attributes?

A binary (two-valued) attribute is created by computing a threshold X

Nota: only some thresholds are shown. The best one is X=84 with average entropy = 0.83



64 – Yes, 65-No, 68 – Yes, 69 – Yes, 70 – Yes, 71 – No, 72 – YesNo, 75 – YesYes, 80 – No, 81 – Yes, 83 – Yes, 85 - No

HP = 0.83

X=84

4 No, 9 Yes

1 No, 0 Yes

64 – Yes, 65-No, 68 – Yes, 69 – Yes, 70 – Yes, 71 – No, 72 – YesNo, 75 – YesYes, 80 – No, 81 – Yes, 83 – Yes, 85 - No

HP = 0.93

2 No, 4 Yes

X=71.5

3 No, 5 Yes

64 – Yes, 65-No, 68 – Yes, 69 – Yes, 70 – Yes, 71 – No, 72 – YesNo, 75 – YesYes, 80 – No, 81 – Yes, 83 – Yes, 85 - No

HP = 0.89

1 No, 4 Yes

X=70.5

4 No, 5 Yes

Extra-trees results

- In 12 classification and 12 regression problems

Table 2 Win/Draw/Loss records (corrected *t*-tests) comparing the algorithm in the column versus the algorithm in the row

	Classification problems					Regression problems				
	PST	TB	RS*	RF*	ET^d	PST	TB	RS*	RF*	ET^d
PST	–	8/4/0	11/1/0	11/1/0	10/2/0	–	10/2/0	8/4/0	10/2/0	10/2/0
TB	0/4/8	–	7/5/0	7/5/0	7/5/0	0/2/10	–	0/9/3	1/11/0	4/8/0
RS*	0/1/11	0/5/7	–	0/8/4	2/10/0	0/4/8	3/9/0	–	4/8/0	4/7/1
RF*	0/1/11	0/5/7	4/8/0	–	5/7/0	0/2/10	0/11/1	0/8/4	–	3/7/2
ET^d	0/2/10	0/5/7	0/10/2	0/7/5	–	0/2/10	0/8/4	1/7/4	2/7/3	–

- PST = single trees (trained with CART)
- TB = Tree Bagging
- RF = Random Forests

Extra-trees results

- ET takes much less time

Table 4 Computing times (msec) of training (ensembles of $M = 100$ trees)

Classification problems					Regression problems			
Dataset	ST	TB	RF^d	ET^d	Dataset	ST	TB/RF^d	ET^d
Waveform	68	4022	1106	277	Friedman1	7	372	284
Two-Norm	66	3680	830	196	Housing	12	685	601
Ring-Norm	101	4977	1219	251	Hwang-f5	16	917	742
Vehicle	80	5126	1500	685	Hwang-f5n	15	948	748
Vowel	236	14445	4904	694	Pumadyn-32fh	251	13734	9046
Segment	291	18793	5099	1053	Pumadyn-32nm	221	11850	9318
Spambase	822	55604	9887	8484	Abalone	73	4237	3961
Satellite	687	45096	11035	5021	Ailerons	495	29677	26572
Pendigits	516	34449	12080	5183	Elevators	289	15958	13289
Dig44	4111	259776	67286	9494	Poletelecomm	497	28342	26576
Letter	665	44222	17041	14923	Bank-32nh	613	34402	20178
Isolet	37706	2201246	126480	11469	Census-16H	597	35207	27900