ENSEMBLE NOISE FILTERING FOR STREAMING DATA USING POISSON BOOTSTRAP MODEL FILTERING

Dr. Ashwin Satyanarayana,
Rosemary Chinchilla
Computer Systems Technology
New York City College of Technology (CUNY),
Brooklyn, NY - 11201

13th International Conference on
Information Technology : New Generations
ITNG 2016
April 11-13, 2016, Las Vegas, Nevada, USA

What is an outlier (anomaly / noise)?

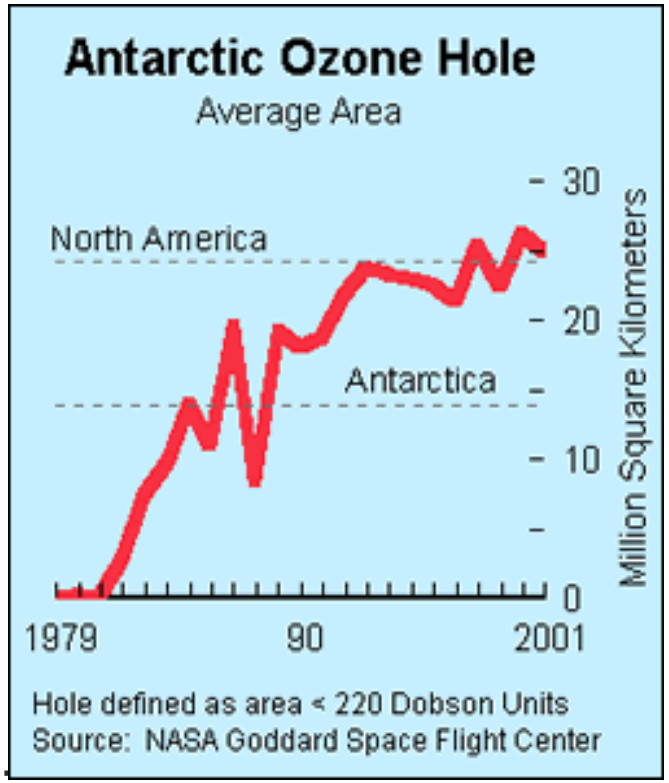
Outlier is an observation that deviates too much from other observations so that it arouses suspicions that it was generated by a different mechanism.

Outliers

Importance of Anomaly Detection

Ozone Depletion History

- In 1985 three researchers (Farman, Gardinar and Shanklin) shocked the world by data gathered by the British Antarctic Survey showing that ozone levels for Antarctica had dropped 10% below normal levels
- Why did the Nimbus 7 satellite, which had instruments for recording ozone levels, not record similarly low ozone concentrations?
- The ozone concentrations recorded by the satellite were so low they were being treated as outliers by a computer program and discarded!



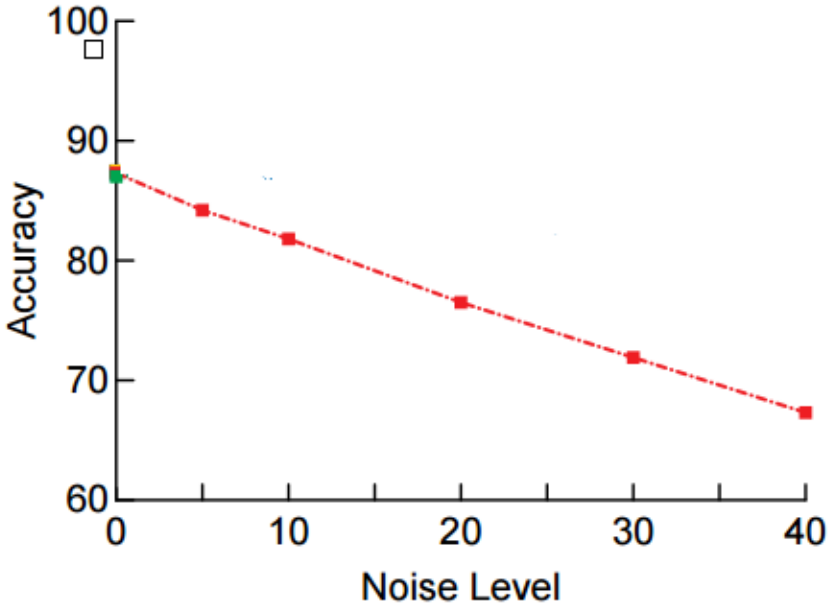
Sources:
http://exploringdata.cqu.edu.au/ozone.html
http://www.epa.gov/ozone/science/hole/size.html

Anomaly/Outlier Detection

- What are anomalies/outliers?
  - The set of data points that are considerably different than the remainder of the data
- Variants of Anomaly/Outlier Detection Problems
  - Given a database D, find all the data points $\mathbf{x} \in D$ with anomaly scores greater than some threshold t
  - Given a database D, find all the data points $\mathbf{x} \in D$ having the top-n largest anomaly scores $f(\mathbf{x})$
- Applications:
  - Credit card fraud detection, telecommunication fraud detection, network intrusion detection, fault detection

Noise Removal in Training Data

- Quinlan level increase mislabeled training eases the predictiv classier.

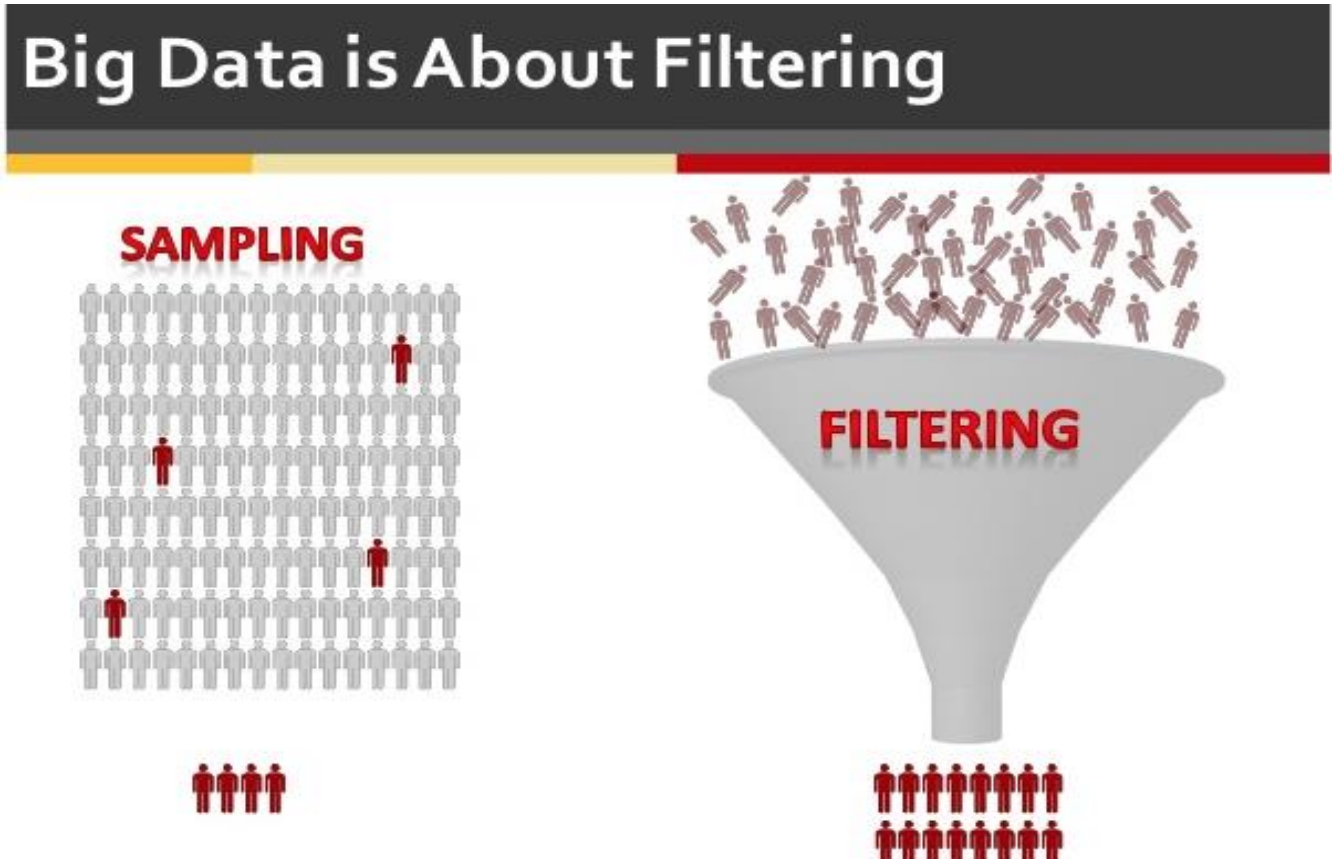


- Brodley and Friedl illustrate that for class noise levels of less than 40%, removing mislabeled instances from the training set resulted in higher predictive accuracy relative to classification achieved without “cleaning" the training data.

TNG

NEW YORK CITY COLLEGE OF TECHNOLOGY CUNY

- This paper focuses on improving the quality of streaming data by ***identifying*** and ***eliminating*** mislabeled instances.



Limitations of Prior Work

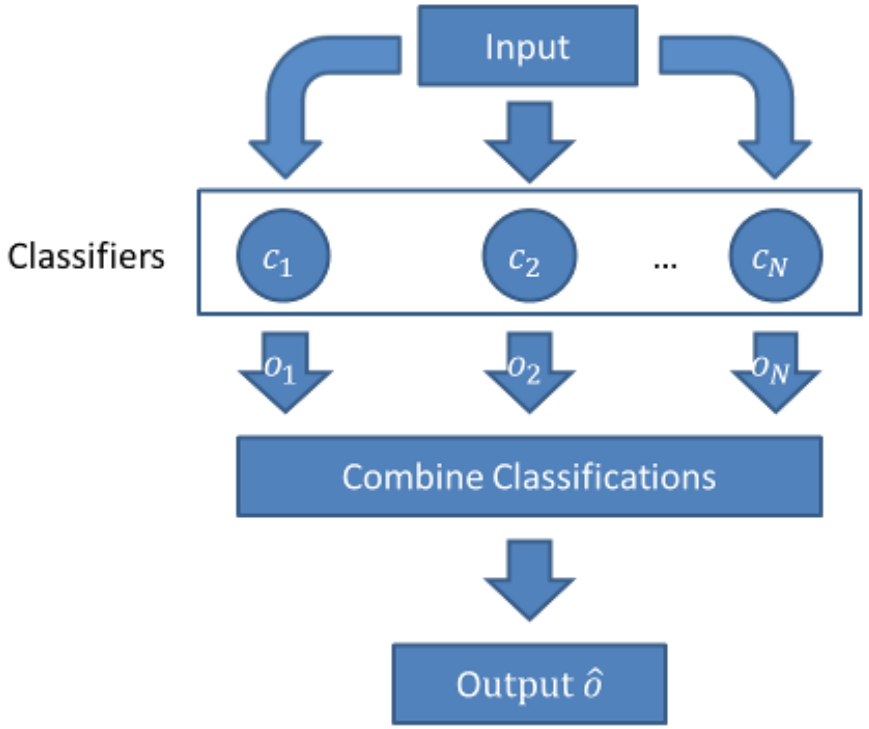
- Does not work on streaming data
- No analysis performed to determine the underlying *noise model distribution*
- No sampling done along with filtering

Our contributions in this area are as follows:

1. We use ensemble classifiers with voting for higher predictive accuracy
2. We perform Bayesian analysis to understand the underlying *noise model distribution* and show it tends to Poisson(1) distribution as $n \rightarrow \infty$.
3. Bootstrap sampling reduces to Poisson(1) distribution as $n \rightarrow \infty$

Background: Ensemble Classifiers

- Instead of using one base classifier, we use multiple classifiers with voting between them to identify bad records (instances).
- An **ensemble classifier** detects noisy instances by constructing a set of classifiers (base level detectors).
- A majority vote filter tags an instance as mislabeled if more than half of the *m* classifiers classify it incorrectly.

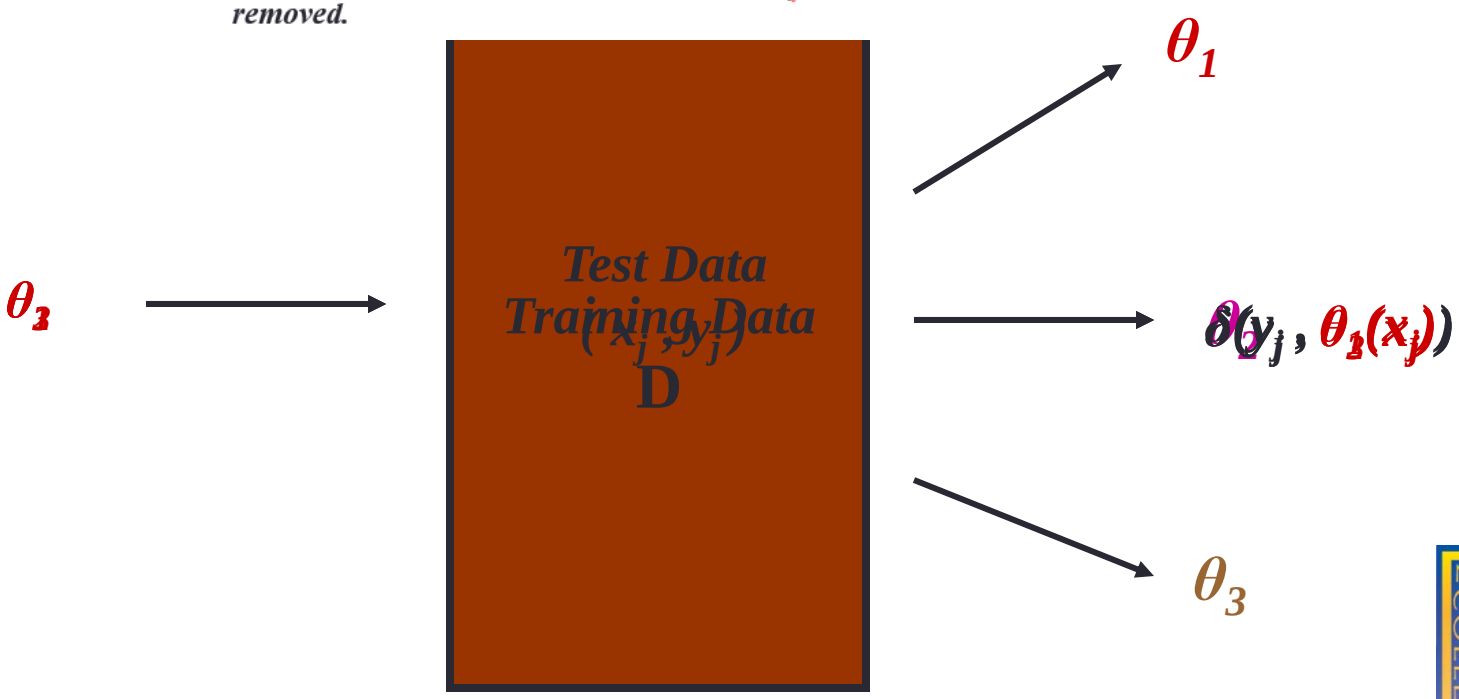


Actual Value y_j	Predicted $\theta_1(x_j)$	Predicted $\theta_2(x_j)$	Predicted $\theta_3(x_j)$
A			
B			

Noise

Example: Say $\delta(y_j, \theta_1(x_j)) = 1$ *(non-noisy)*
$\delta(y_j, \theta_2(x_j)) = 0$ *(noisy)*
$\delta(y_j, \theta_3(x_j)) = 0$ *(noisy)*

$\Pr(y|x,\Theta^*) = 1/3 = 0.33 < ½$ *, The instance* x_j *is treated as noisy and is removed.*



Why do ensemble classifiers work?

- Ensemble techniques can be considered as performing **model averaging (they extend the model space)**.

$$\Pr(y = t \mid x, \Theta^*) = \frac{1}{|\Theta^*|} \sum_{\theta \in \Theta^*} \delta(t, \theta(x))$$

  - If $\Pr(y = t \mid x, \Theta^*) < ½$ then the instance is noisy and is removed
  - Example: Say $\delta(y_j, \theta_1(x_j)) = 1$ ***(non-noisy)***
    $\delta(y_j, \theta_2(x_j)) = 0$ ***(noisy)***
    $\delta(y_j, \theta_3(x_j)) = 0$ ***(noisy)***

$\Pr(y|x,\Theta^*) = 1/3 = 0.33 < ½$, *The instance x_j is treated as noisy and is removed.*

Ensemble Classifiers will not work on Large Streaming data

- The main challenge in applying prior filtering techniques to data streams is that a stream is theoretically infinite and that means that data has to be processed in a single pass using little memory.

- We need to understand the underlying noise distribution and use bootstrapping.

Bayesian Analysis (1 of 2)

- Let $\vec{x}$ represent the original training set, $\vec{y}$ the corresponding class labels and a model (or hypothesis) in the model space Θ . Mathematically, an unseen test instances x, is assigned to a class y that maximizes the following equation:

$$Pr(y|\vec{x},\Theta) = \sum_{\theta \in \Theta} Pr(y|x,\theta).Pr(\theta|\vec{x},\vec{y})$$

Prior Probability

- By Bayes' theorem, and assuming the instances are drawn independently, the posterior probability of is given by:

$$Pr(\theta|\vec{x},\vec{y}) = \frac{Pr(\theta)}{Pr(\vec{x},\vec{y})} \prod_{i=1}^{n} Pr(x_i,y_i|\theta)$$

Likelihood

Noise Model

$$Pr(x_i,y_i|\theta) = Pr(x_i,\theta).Pr(y_i|x_i,\theta)$$

Bayesian Analysis (2 of 2)

- The probability $Pr(y_i|x_i,\theta)$ is called the ***noise model***.
- We assume a uniform class model in which each instance's class is corrupted with probability ϵ, and thus:

$$Pr(y_i|x_i,\theta) = 1-\epsilon \text{ if } \theta \text{ predicts the correct class } y_i, \text{ and}$$
$$Pr(y_i|x_i,\theta) = \epsilon \text{ if } \theta \text{ predicts an incorrect class}$$

$$Pr(\theta|\vec{x},\vec{y}) \propto Pr(\theta)(1-\epsilon)^s \epsilon^{n-s}$$

where ***s*** is the number of instances correctly classified by θ. As $n \rightarrow \infty$ the distribution of the noise model tends to **Poisson(1) distribution**.

Introduction to Bootstrapping

- Consider the typical view of mining where a single training set of size *n* is available from the underlying distribution that generated the data F.
- However this view masks the underlying uncertainty in the data, namely that the training data we have is one of many that could have been chosen. If we were to build a model for each possible data set we would have a probability distribution over the model space.
- To approximate this distribution using a single dataset, Efron [8] created the **bootstrapping approaches**.

Poisson Bootstrap Model Filtering

- The standard bootstrap procedure creates each simulated data set by drawing observations from *x* with replacement. In any given resample, each observation may occur 0, 1 or more times according to:

$$Binomial(n, \frac{1}{n})$$

- Estimate population mean given the random sample:

{1.4, 2.6, 2.9, 4.2}

Bootstrapped Resamples:

{4.2, 2.9, 2.6, 4.2}
{2.6, 2.6, 1.4, 2.6}
{2.9, 4.2, 2.6, 4.2}

Counts:

{ 0, 1, 1, 2 } # Counts for Resample 0
{ 1, 3, 0, 0 } # Counts for Resample 1
{ 0, 1, 1, 2 } # Counts for Resample 2

The total number of observations is constrained to be n, the counts are jointly: Multinom(4, 1/4, 1/4, 1/4, 1/4).

$$Multinomial(n, \frac{1}{n}, \ldots, \frac{1}{n})$$

TNG

NEW YORK CITY COLLEGE OF TECHNOLOGY CUNY

Poisson distribution

- But when it comes to big data, the multinomial bootstrap is computationally problematic. But we note that :

$$\lim_{n\to\infty} Binomial(n, \frac{1}{n}) = Poisson(1)$$

- This way there is no need to know *n*, the total count of observations, ahead of time. When applied to our tiny example, we generate *B* boostrapped samples from Poisson(1) for each observation independently.

Bootstrap Model Filtering Algorithm

Algorithm BootstrapModelFiltering (E, n, k)

Input: E (training set),
n (number of Poisson bootstrap samples),
k (number of inductive learning algorithms)
Output: A (a detected noisy subset of E)

1: Form n Poisson bootstrap samples $E_{1,\ldots,n}$ of E
2: $A \leftarrow \emptyset$
3: **for** $i = 1, \ldots, n$ **do**
4: **for** $j = 1, \ldots, k$ **do**
5: $\theta_{i,j} \leftarrow$ model built from bootstrap sample E_i and inductive algorithm j
6: **end for**
7: $E_t \leftarrow E \backslash E_i$
8: **for** every $e \in E_t$ **do**
9: If e is misclassified by more than half of the $\theta_{i,j}$ models built, then it is noisy and needs to be eliminated.
10: $A \leftarrow A \cup \{e\}$
11: **end for**
12: **end for**

Tool used: WEKA

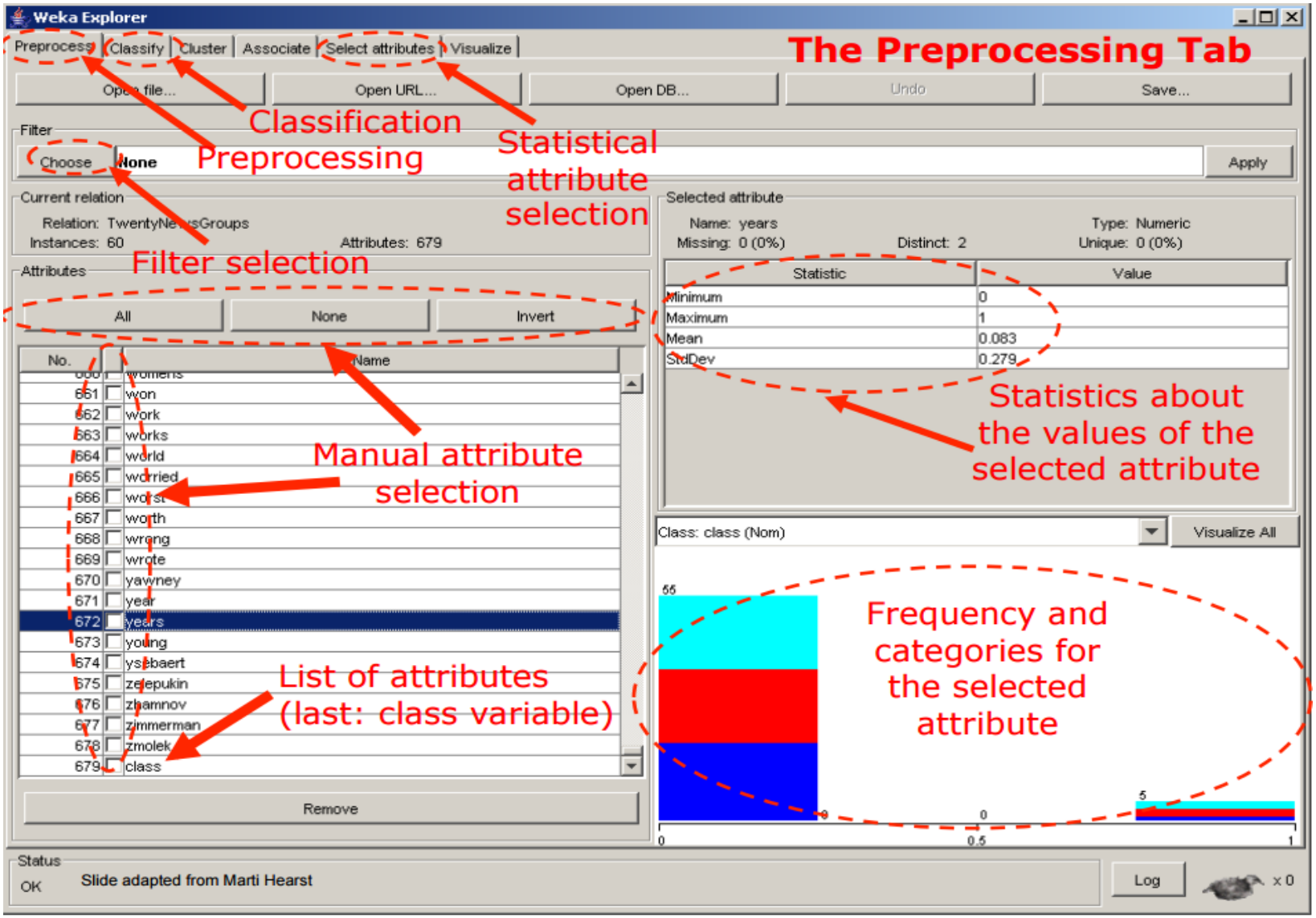
Weka: Data Mining Software

- Collection of machine learning algorithms
  - open-source package written in Java
- Used for research, education and application
- Main features:
  - data pre-processing tools
  - learning algorithms
  - evaluation methods
  - graphical inference
  - environment for comparing learning algorithms

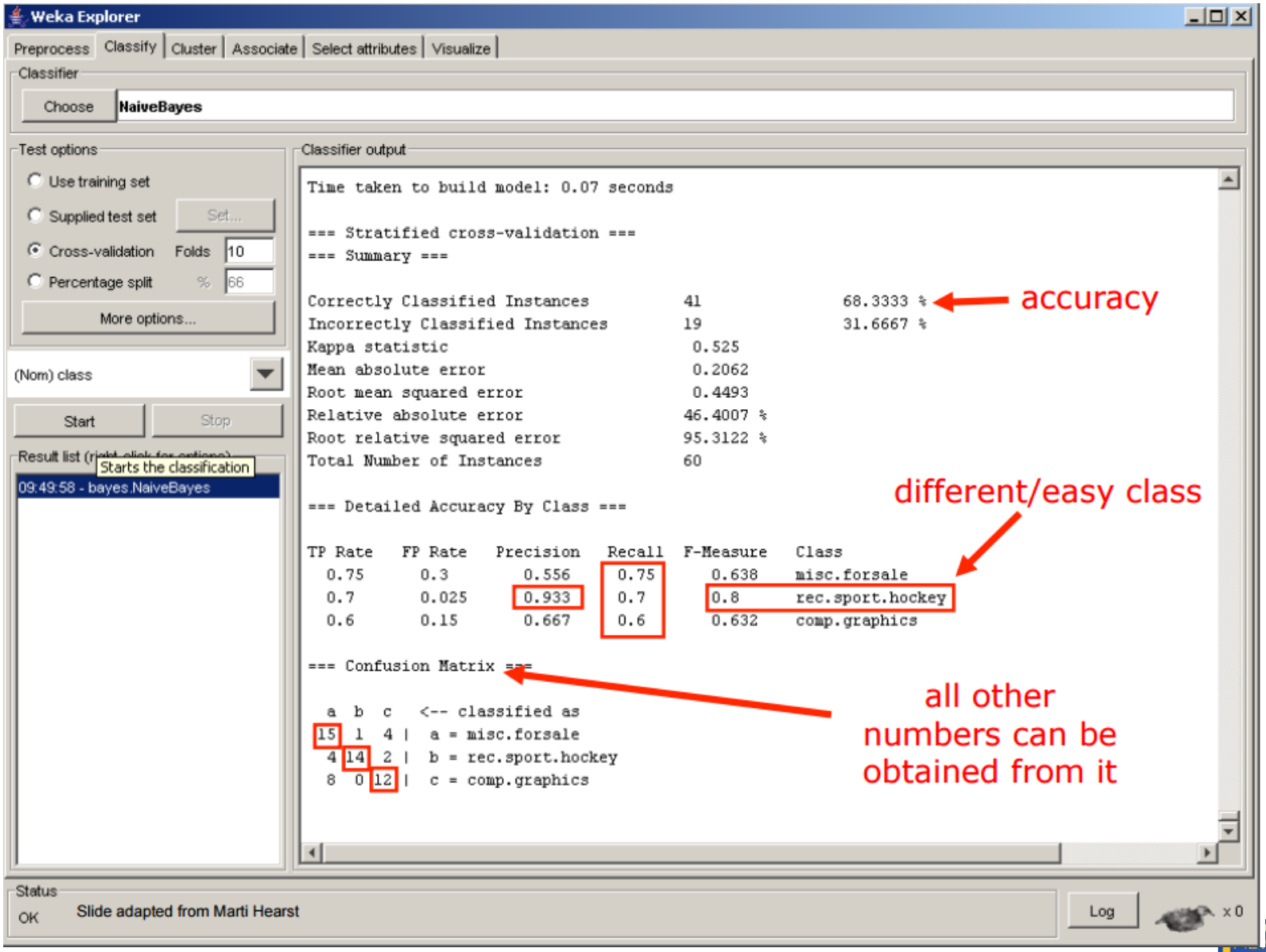
Weka: Data Mining Software

- Classification algorithms:
  - decision trees, kNN, SVM, Naive-bayes
- Prediction algorithms:
  - regression (linear/SVM) , perceptron
- Meta-algorithms:
  - bagging, boosting (AdaBoost)

among others


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
The Preprocessing Tab
Open file...
Open URL...
Open DB...
Undo
Save...
Filter
Choose
None
Apply
Classification
Preprocessing
Statistical
attribute
selection
Filter selection
Current relation
Relation: TwentyNewsGroups
Instances: 60
Attributes: 679
Selected attribute
Name: years
Missing: 0 (0%)
Distinct: 2
Type: Numeric
Unique: 0 (0%)
Statistic
Value
Minimum 0
Maximum 1
Mean 0.083
StdDev 0.279
Attributes
All
None
Invert
No.
Name
661 won
662 work
663 works
664 world
665 worried
666 worst
667 worth
668 wrong
669 wrote
670 yawney
671 year
672 years
673 young
674 yssebaert
675 zelepukin
676 zhamnov
677 zimmerman
678 zmolek
679 class
Manual attribute
selection
Statistics about
the values of the
selected attribute
Class: class (Nom)
Visualize All
55
0
5
0
0.5
1
Frequency and
categories for
the selected
attribute
List of attributes
(last: class variable)
Remove
Status
OK
Slide adapted from Marti Hearst
Log
x 0


TNG


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
NaiveBayes
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) class
Start
Stop
Result list (right-click for options)
Starts the classification
09:49:58 - bayes.NaiveBayes
Classifier output
Time taken to build model: 0.07 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 41 68.3333 %
Incorrectly Classified Instances 19 31.6667 %
Kappa statistic 0.525
Mean absolute error 0.2062
Root mean squared error 0.4493
Relative absolute error 46.4007 %
Root relative squared error 95.3122 %
Total Number of Instances 60
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure Class
0.75 0.3 0.556 0.75 0.638 misc.forsale
0.7 0.025 0.933 0.7 0.8 rec.sport.hockey
0.6 0.15 0.667 0.6 0.632 comp.graphics
=== Confusion Matrix ===
a b c <-- classified as
15 1 4 | a = misc.forsale
4 14 2 | b = rec.sport.hockey
8 0 12 | c = comp.graphics
accuracy
different/easy class
all other numbers can be obtained from it
Status
OK
Slide adapted from Marti Hearst
Log
x 0

WEKA Code for Ensemble Filtering

```
//Check
System.out.println("\n10 Remove predicted classes before the 2nd run of classification:");
System.out.println("   Hit \"Enter\" key to continue");
System.in.read(new byte[System.in.available()]);//Flush/Clear System.in (stdin) before reading
System.in.read();       //PAUSE

remove.setAttributeIndices(Utils.getOption("x", args));         //i indexing from 1

// w - inversion state
remove.setInvertSelection(new Boolean(Utils.getOption("w", args)).booleanValue());
remove.setInputFormat(mergedData);
mergedData = Filter.useFilter(mergedData, remove);          // OBJECT mergedData



//DISPLAY DATA FOR THE 2nd RUN OF CLASSIFICATION
System.out.println(mergedData);

//WRITE OBJECT mergedData TO an ARFF FILE
DataSink.write(Utils.getOption("n", args), mergedData);     //-n write to filtered.arff
```

```
// DELETE NOISE RECORDS
n = 0;
for (int i = mergedData.numInstances() - 1; i >= 0; i--)
{
    //INDEXES COUNTED FROM 0
    String act = mergedData.instance(i).stringValue(nact);
    String c11 = mergedData.instance(i).stringValue(nc11);
    String c12 = mergedData.instance(i).stringValue(nc12);
    String c13 = mergedData.instance(i).stringValue(nc13);

    //!(filtering good records) = noise records do delete

    //Regular Rule: two predictions == actual then accepted
    if(!(act.equals(c11) && c11.equals(c12)
            || act.equals(c11) && c11.equals(c13)
            || act.equals(c12) && c12.equals(c13)))

    //Strict Rule: All tree predictions == actual
    //if(!(act.equals(c11) && c11.equals(c12) && c12.equals(c13)))
    {
        // OBJECT mergedData with deleted noise
        mergedData.delete(i);
        n++ ;
    }
}
```

```
//Merging objects:
   //The "mergedData" contains the following attributes:
                              //all regular attributes + act, class1, class2, class3
//-------------------------------------------------------------------------------------348 input
//READ 1-st ARFF FILE                                                  //input - run1348.arff
                                                                       //output - inst1 :  Instances
//INPUT: train.arff
Instances inst1 = DataSource.read(Utils.getOption("j", args));      // -j read run1348.arff
inst1.renameAttribute(nc11, "class1");          //indexing from 0       rename predicted class
//-------------------------------------------------------------------------------NAIVE BAYS input
//READ 2-nd ARFF FILE AND EXTRACT PREDICTED CLASS ATTRIBUTE            //input - run1RT.arff
                                                                     //output - instTemp : Instances
Instances inst = DataSource.read(Utils.getOption("t", args));          // -t read run1RT.arff
Remove remove = new Remove();
remove.setAttributeIndices(Utils.getOption("i", args));                        //i indexing from 1
remove.setInvertSelection(new Boolean(Utils.getOption("v", args)).booleanValue());// v - inversion state
remove.setInputFormat(inst);
inst = Filter.useFilter(inst, remove);                        //extract predicted class in run1RT.arff
inst.renameAttribute(0, "class2");   //indexing from 0          rename predicted class
//-------------------------------------------------------------------------------------OBJECT mergedData
//1-st MERGE CLASSIFICATIONS                                           //input - (inst1 + inst) : Instances
                                                                       //output - mergedData   : Instances

   Instances mergedData = Instances.mergeInstances(inst1 ,inst);   //merge to the 1-st file
                                                                    //the predicted class from the 2-nd file

//-------------------------------------------------------------------------------RANDOM FOREST input
//READ 3-rd ARFF FILE AND EXTRACT PREDICTED CLASS ATTRIBUTE                 //input - run1RF.arff
                                                                        //output - instTemp : Instances
inst = DataSource.read(Utils.getOption("f", args));         // -f read run1RF.arff
remove.setInputFormat(inst);
inst = Filter.useFilter(inst, remove);                      // extract predicted class in run1RF.arff
inst.renameAttribute(0, "class3");   //indexing from 0          rename predicted class
//-------------------------------------------------------------------------------------OBJECT mergedData
//2-nd MERGE CLASSIFICATIONS                                //input - (mergedData +  instTemp): Instances
                                                                //output - mergedData   : Instances
mergedData = Instances.mergeInstances(mergedData ,inst);    //merge to the two already merged files
                                                                //the predicted class from the 3-rd file
```

Empirical Results

- We tested our approach on small, medium and massive datasets (from UCI repository)
- For each dataset, we compare the our technique with the following techniques:

1. **Single Model:** We used decision trees (J48) as our single filtering base model.

2.**Partition Filtering:** Partition Filtering is chosen as it outperforms Classification and Ensemble Filtering techniques [6]. We use Decision Tree as the base model.

3. Online Bagging: We implemented online bagging as illustrated by Oza [12] using Naïve Bayes as the base model.

Small Datasets

- We use 30 Poisson bootstrap samples with 3 different classfiers (J48, RandomForest and RandomTree)

Table 1. Sizes of Small Datasets

Dataset	# of instances	# of attributes	# of classes
Diabetes	768	9	2
Liver Disorder	345	7	2
Hepatitis	155	20	2
Dengue	846	18	2

Table 2. Classification Accuracy after eliminating noisy instances

Dataset	Decision Tree (J48)	Partition Filtering	Online Bagging	Bootstrap Model Filtering
Diabetes	0.73	0.75	0.79	**0.88**
Liver Disorder	0.68	**0.74**	**0.76**	**0.79**
Hepatitis	0.83	0.84	0.88	0.89
Dengue	0.74	0.77	0.81	**0.92**

Medium and Large Datsets

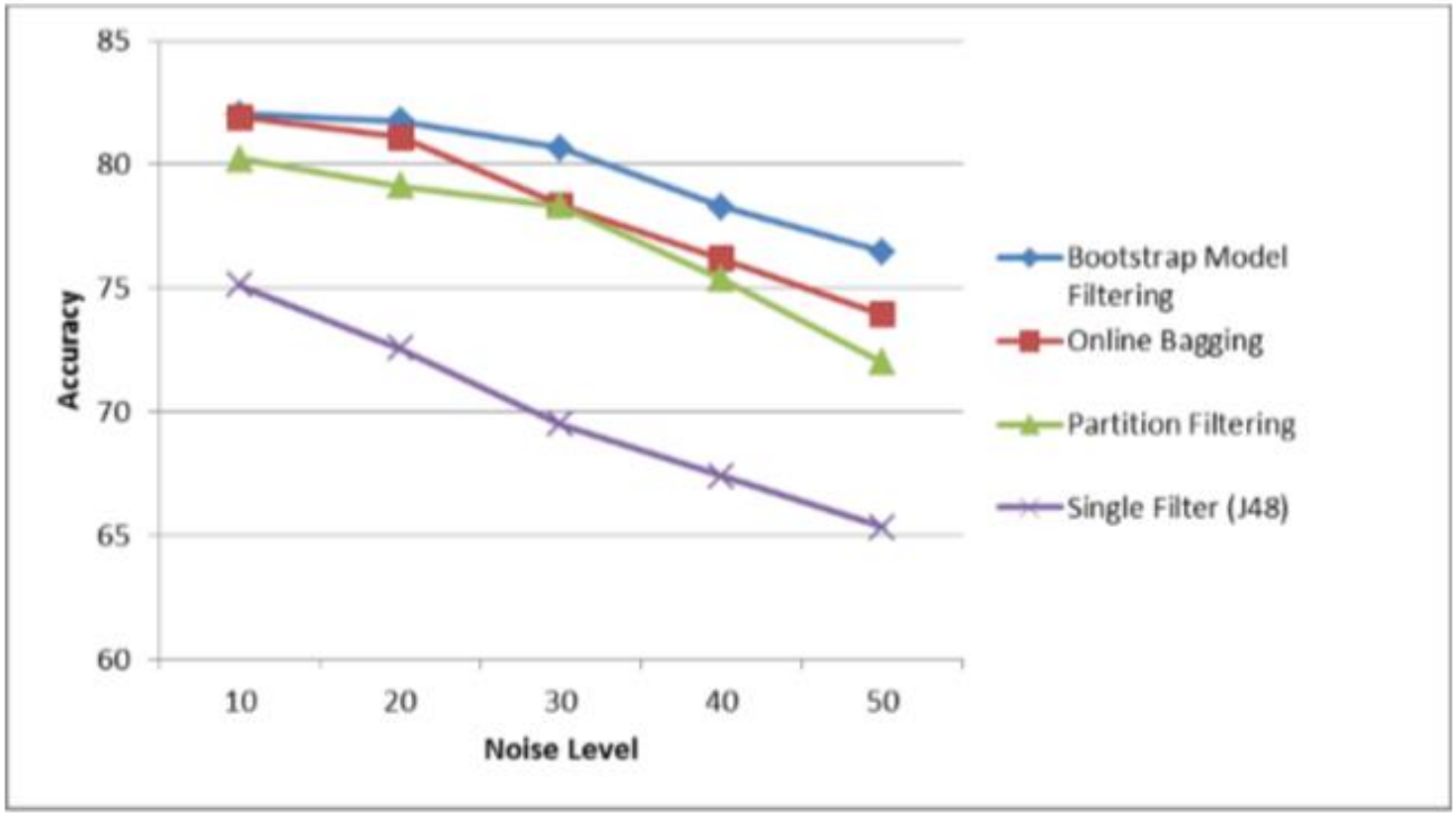
- Medium: 50 Poisson bootstrap samples
- Large: 100 Poisson bootstrap samples

Dataset	# of instances	# of attributes	# of classes
CMC	1473	10	3
Car Evaluation	1728	7	4
Multiple Features	2000	48	10
Abalone	2924	9	28
Waveform	3500	41	3
Wine Quality	3429	12	11
Nursery	12960	9	5
Letter Recognition	20000	17	26
Electricity Board	45781	5	31
Localization	164860	8	11

Dataset	Decision Tree (J48)	Partition Filtering	Online Bagging	Bootstrap Model Filtering
CMC	0.52	0.65	**0.69**	**0.75**
Car Evaluation	0.92	0.94	0.95	0.97
Multiple Features	0.72	0.74	0.76	**0.78**
Abalone	0.20	0.34	**0.38**	**0.48**
Waveform	0.75	0.76	0.77	**0.82**
Wine Quality	0.56	0.58	0.61	**0.69**
Nursery	0.97	0.97	0.98	0.98
Letter Recognition	0.87	0.88	0.88	0.89
Electricity Board	0.65	0.66	0.68	**0.73**
Localization	0.76	0.73	0.79	**0.89**

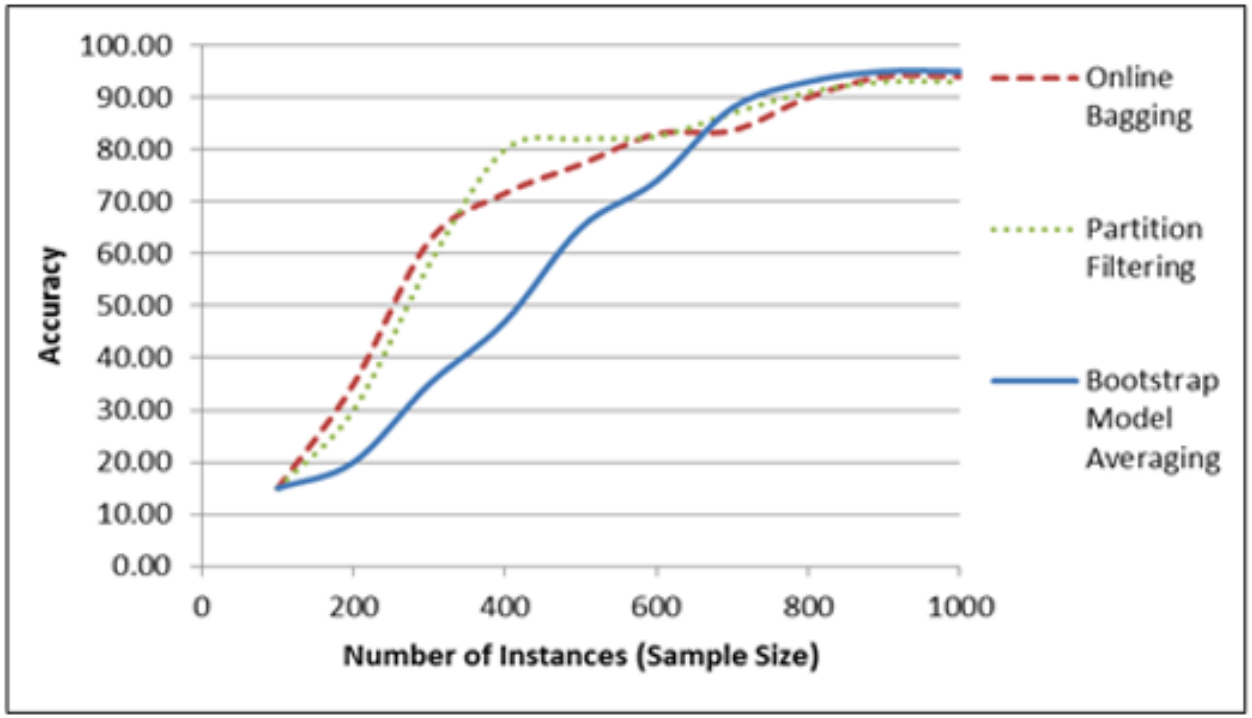
Classification accuracy vs noise

- We introduce noise levels from 10% to 40% on each of the datasets by adding mislabeled instances in the test set.



Learning Curve Phenomenon

Car evaluation dataset: 1728 instances. We used 528 as the test set. Streams of 200, 400, 600, 800 and 1000 instances as streams of training data.



Conclusion

- Resolving data quality issues in stream mining is often one of the biggest efforts in data mining.
- The main challenge in applying prior filtering techniques to data streams is that a stream is theoretically infinite and that means that data has to be processed in a single pass using little memory.
- Also, using all the available data is prohibitive due to memory and time constraints. In this paper, we attempt to address this problem for different sized datasets by using Poisson bootstrap samples with multiple base classifiers.
- Although we have used decision trees, random forest and random trees, other base classifier models can also be used.

Questions?