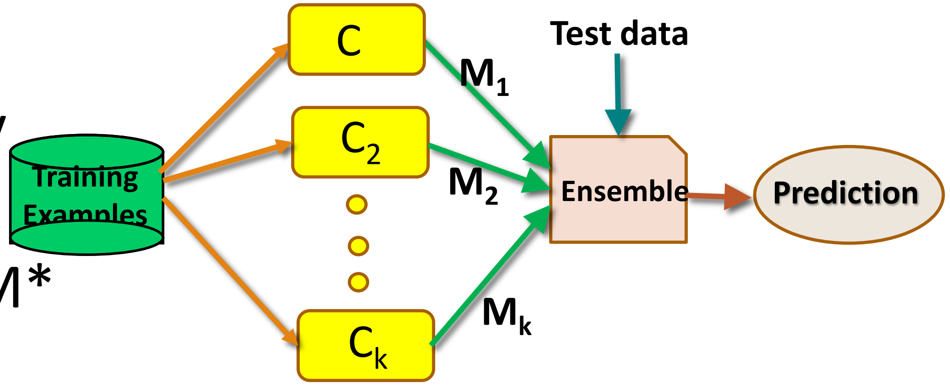
# Techniques to Improve Classification Accuracy: Ensemble Methods

# Outline

- Techniques to Improve Classification Accuracy: Ensemble Methods
- Bagging: Bootstrap Aggregation
- Random Forest: Basic Concepts and Methods
- Boosting and AdaBoost
- Classification of Class-Imbalanced Data Sets
- Classifying Data Streams with Skewed Distribution

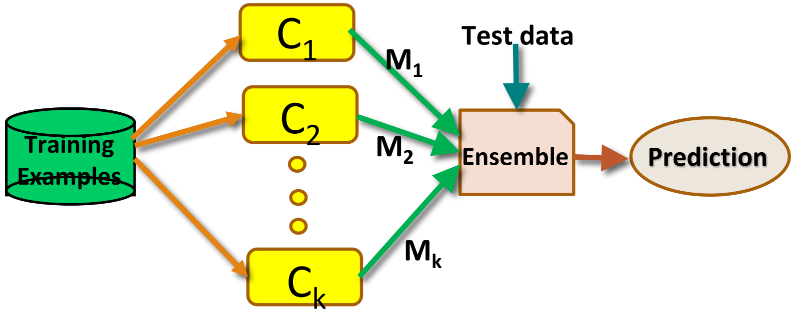
# Ensemble Methods: Increasing the Accuracy




- Ensemble methods
  - Use a combination of models to increase accuracy
  - Combine a series of *k* learned models, $M_1$, $M_2$, …, $M_k$, with the aim of creating an improved model M*
- Popular ensemble methods
  - Bagging: Trains each model using a subset of the training set
    - Models learned independently, in parallel
    - Simple voting: Outcome is determined by the majority of the parallel models
  - Boosting: Trains each new model instance to emphasize the training instances that previous models misclassified (correcting the "errors" of previous model)
    - Models learned in order (a sequential ensemble)
    - Weighted voting: Outcome is determined by the majority - but the sequential models were built by assigning greater weights to misclassified instances of the previous models

# Bagging: Bootstrap Aggregation

- Analogy: Diagnosis based on multiple doctors' majority vote
- Training
  - Given a set D of tuples, at each iteration *i*, a training set $D_i$ of *d* tuples is sampled with replacement from D (i.e., bootstrap)
  - A classifier model $M_i$ is learned for each training set $D_i$

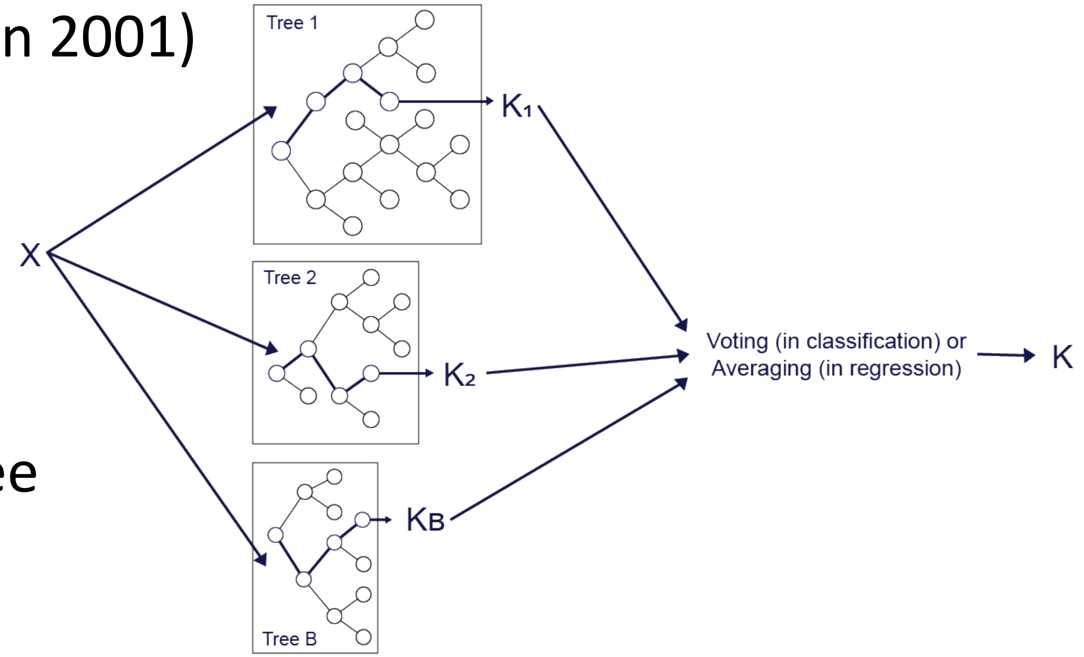



- Classification: Classify an unknown sample **X**
  - Each classifier $M_i$ returns its class prediction
  - The bagged classifier M* counts the votes and assigns the class with the most votes to **X**
- Prediction: It can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple
- Accuracy: Improved accuracy in prediction
  - Often significantly better than a single classifier derived from D
  - For noise data: Not considerably worse, more robust

# Random Forest: Basic Concepts

- Random Forest (first proposed by L. Breiman in 2001)
  - A variation of bagging for *decision trees*
  - *Data bagging*
    - Use a subset of training data

      by sampling with replacement for each tree
  - *Feature bagging*
    - At each node use a random selection of attributes as candidates and split by the best attribute among them
  - Comparing with original bagging, the *random forests* method increases the diversity among generated trees
  - During classification, each tree votes and the most popular class is returned



# Methods for Constructing Random Forest

- Two Methods to construct Random Forest:
  - Forest-RI (*random input selection*)
    - Randomly select, at each node, *F* attributes as candidates for the split at the node
      - The CART methodology is used to grow the trees to maximum size
  - Forest-RC (*random linear combinations*)
    - Creates new attributes (or features) that are a linear combination of the existing attributes (reduces the correlation between individual classifiers)
- Comparable in accuracy to Adaboost, but more robust to errors and outliers
- Insensitive to the number of attributes selected for consideration at each split, and faster than typical bagging or boosting

# Boosting

- Analogy: Consult several doctors, based on a combination of weighted diagnoses - weight assigned based on the previous diagnosis accuracy
- How boosting works?
  - **Weights** are assigned to each training tuple
  - A series of *k* classifiers is iteratively learned
  - After a classifier $M_i$ is learned, the weights are updated to allow the subsequent classifier, $M_{i+1}$, to **pay more attention to the training tuples that were misclassified** by $M_i$
  - The final **M* combines the votes** of each individual classifier, where the weight of each classifier's vote is a function of its accuracy
- Boosting algorithm can be extended for numeric prediction
- Comparing with bagging: Boosting tends to have greater accuracy, but it also risks overfitting the model to misclassified data

# Adaboost (Freund and Schapire, 1997)

- Given a set of *d* class-labeled tuples, $(\mathbf{X_1}, y_1), \ldots, (\mathbf{X_d}, y_d)$
- Initially, all the weights of tuples are set the same (i.e., 1/d)
- Generate *k* classifiers in *k* rounds. At round i,
  - Tuples from D are sampled (with replacement) to form a training set $D_i$ of the same size
    - Each tuple's chance of being selected is based on its weight
  - A classification model $M_i$ is derived from $D_i$
    - Its error rate is calculated using $D_i$ as a test set
  - **If a tuple is misclassified, its weight is increased; otherwise, it is decreased**
- Error rate: $err(\mathbf{X_j})$ is the misclassification error of tuple $\mathbf{X_j}$
- Classifier $M_i$ error rate is the sum of the weights of the misclassified tuples:

$$error(M_i) = \sum_{j}^{d} w_j \times err(\mathbf{X_j})$$

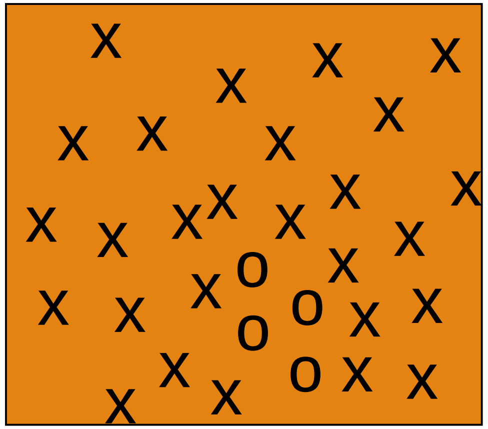
- The weight of classifier $M_i$'s vote is $\log \frac{1-error(M_i)}{error(M_i)}$



8

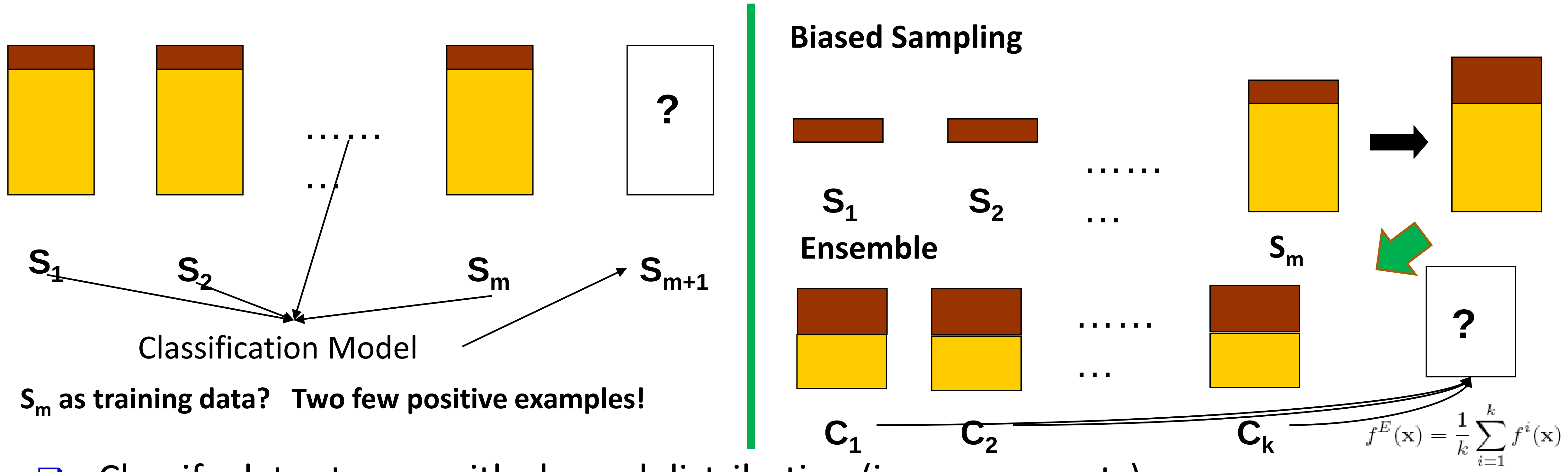
# Classification of Class-Imbalanced Data Sets

- Class-imbalance problem: Rare positive examples but numerous negative ones
  - E.g., medical diagnosis, fraud transaction, accident (oil-spill), and product fault
- Traditional methods assume a balanced distribution of classes and equal error costs: Not suitable for class-imbalanced data



- Typical methods on imbalanced data in two-class classification
  - **Oversampling**: Re-sampling of data from positive class
  - **Under-sampling**: Randomly eliminate tuples from negative class
  - **Threshold-moving**: Move the decision threshold, t, so that the rare class tuples are easier to classify, and hence, less chance of costly false negative errors
  - **Ensemble techniques**: Ensemble multiple classifiers introduced above
- Still difficult for class imbalance problem on multiclass tasks

# Classifying Data Streams with Skewed Distribution




- Classify data stream with skewed distribution (i.e., rare events)
  - **Biased sampling:** Save only the positive examples in the streams
  - **Ensemble:** Partition negative examples of $S_m$ into k portions to build k classifiers
  - Effectively reduce classification errors on the minority class

(Gao, Fan, & Han, 2007)