جامعة محمد بوضياف - المسيلة
Université Mohamed Boudiaf - M'sila

Artificial Learning Models

Lecture 6 : Ensemble Learning (Boosting and Bagging)

By : Dr. Lamri SAYAD

2023

Agenda

- Ensemble learning
- Bagging
- Boosting
- Handling Imbalanced datasets

Objectives

- Combining similar classifiers to improve performance
- Applying the AdaBoost algorithm
- Dealing with classification imbalance

Introduction

- Many models available in machine learning for classification and regression
- Why should the problems you address with machine learning be solved by one algorithm?

- Instead of using one model in isolation improved performance can be obtained by combining different models
  - Meta-algorithms
  - What is meta-algorithms ?
    - Meta-algorithms are a way of combining other algorithms.

Enemble learning (Boosting, bagging …)

Ensemble learning or meta-algorithms

- Methods that do this are known as *ensemble methods or meta-algorithms*.
- Approaches:
  - using different algorithms applied to the same dataset
  - using the same algorithm with different settings
  - assigning different parts of the dataset to different classifiers

Basic ensemble learning techniques

- **Max voting**
  - classifier 1 – class A
  - classifier 2 – class B
  - classifier 3 – class B

  final prediction here would be class B

- **Averaging**
  - regressor 1 – 200
  - regressor 2 – 300
  - regressor 3 – 400

  final prediction would be the average of 200, 300, and 400.

- **Weighted average**
  - Weights = 0.35, 0.45, and 0.2 then final prediction : 0.35 * 200 + 0.45*300 + 0.2*400 = 285

Advanced ensemble learning techniques

- Stacking
- Bagging
- Boosting
- Blending

Ensemble learning: Summary

- Differ in training strategy, and combination method
  - Parallel training with different training sets
    1. Bagging (bootstrap aggregation) { train separate models on overlapping training sets, average their predictions
  - Sequential training, iteratively re-weighting training examples so current classier focuses on hard examples: boosting
  - Parallel training with objective encouraging division of labor: mixture of experts

- Notes:
  - Also known as meta-learning
  - Typically applied to weak models, such as decision stumps (single-node decision trees), or linear classifiers

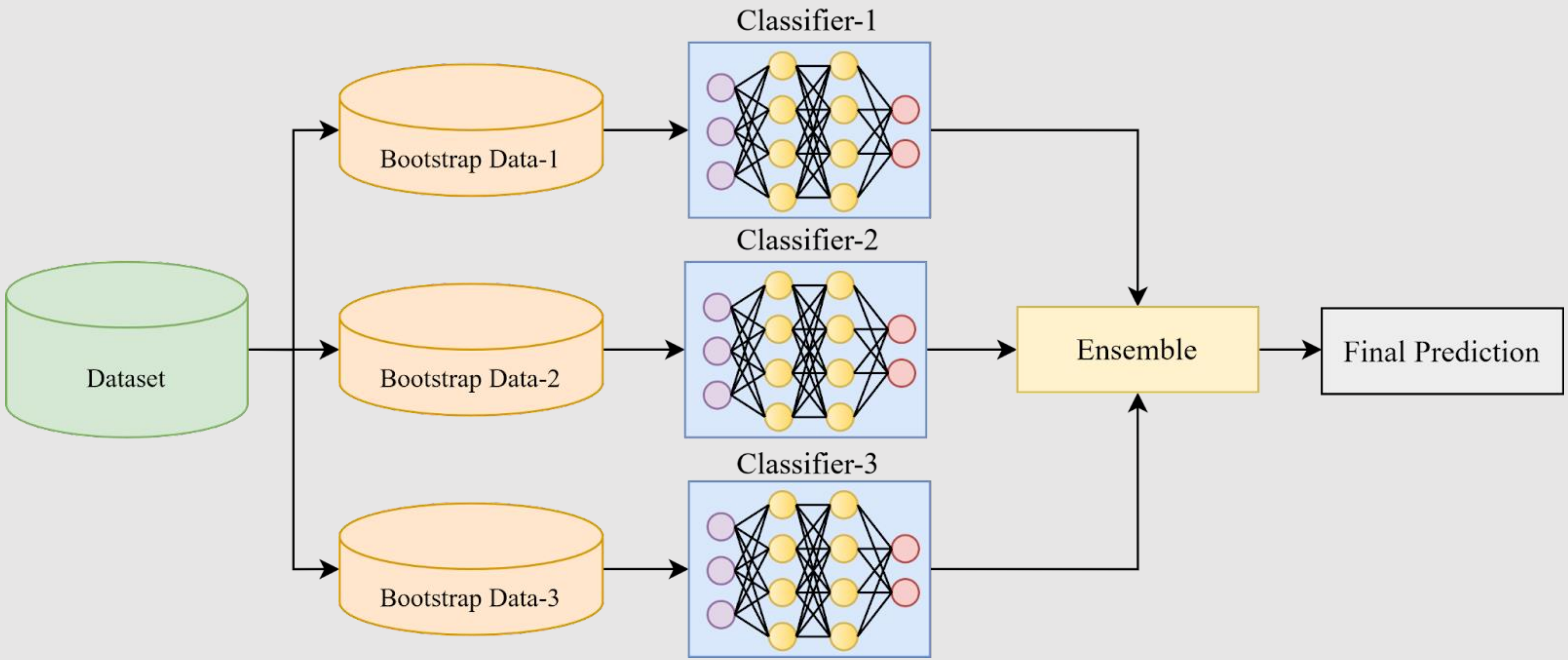
Bagging (Bootstrap aggregating)

- It is short for *Bootstrap Aggregating*
- It is a technique for reducing generalization error by combining several models
  - Idea is to train several models separately, then have all the models vote on the output for test examples
- This strategy is called *model averaging*
- Techniques employing this strategy are known as *ensemble methods*
- Model averaging works because different models will not make the same mistake
- A widely used and effective machine learning algorithm based on the idea of bagging is random forest.

Bagging (Bootstrap aggregating)

- It is a technique where the data is taken from the original dataset S times to make S new datasets.
- The datasets are the same size as the original.
- Each dataset is built by randomly selecting an example from the original with replacement.
  - “with replacement” : you can select the same example more than once.
  - Some values in the new dataset will be repeated
  - some values from the original won’t be present in the new set.
- After the S datasets are built, a learning algorithm is applied to each one individually.
- To classify a new piece of data, you’d apply our S classifiers to the new piece of data and take a majority vote.

Bagging (Bootstrap aggregating)



Bagging Algorithm

Training Phase

Initialize the parameters

D={Φ}

h=the number of classification

For k=1 to h

Take a bootstrap sample Sk from training set S

Build the classifier Dk using Sk as training set

D=DUDi

Return D

Classification Phase

Run D1,D2,………..Dk on the input k

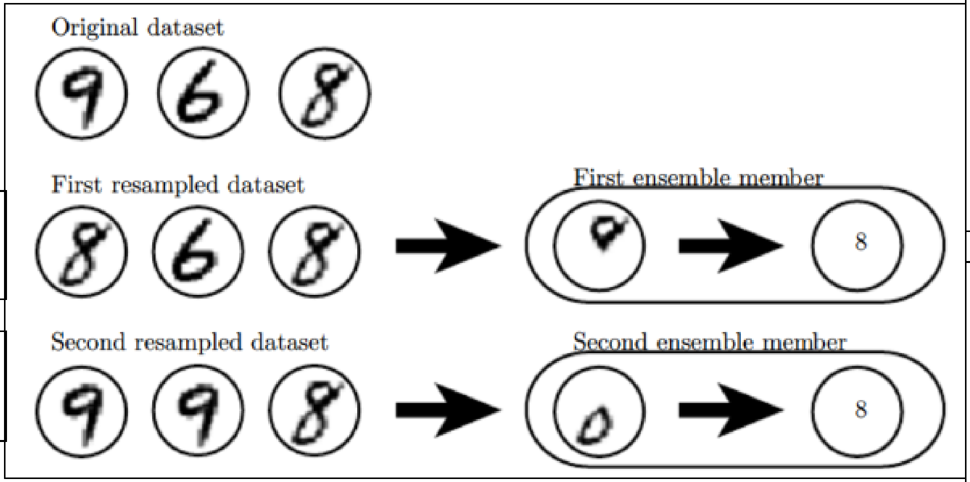
The class with maximum number of vote is choosen as the label for X.

Example of Bagging Principle

- Task of training an 8 detector
- Bagging training procedure
  - make different data sets by resampling the given data set

Omit 9
Repeat 8

Omit 6
Repeat 9



Detector learns that loop on top is an 8

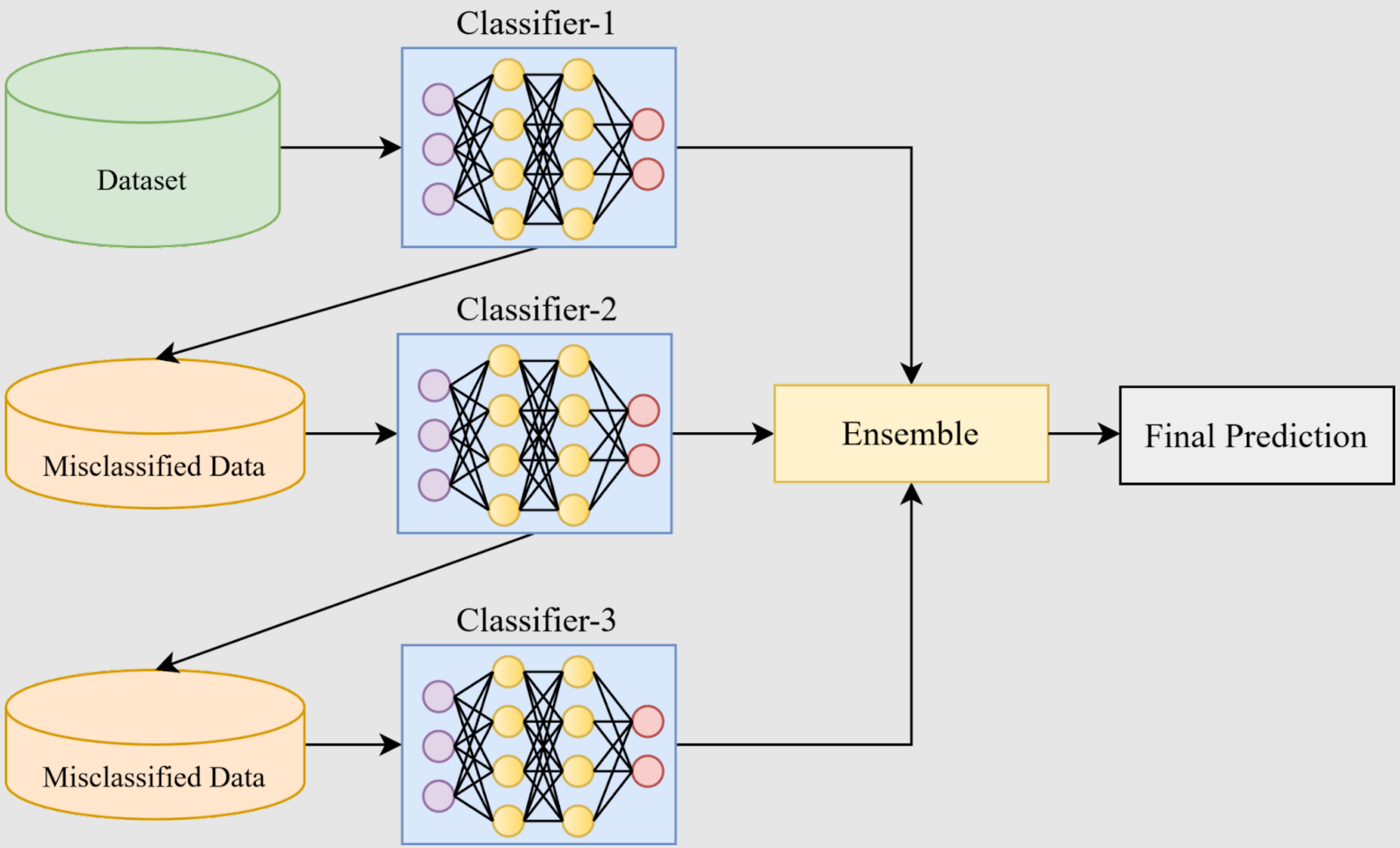
Detector learns that loop at bottom is an 8

- Each detector is brittle
- Their average is robust achieving maximum confidence when both loops are present

Boosting

- Similar to bagging
- Different classifiers are trained sequentially.
- Each new classifier is trained based on the performance of those already trained.
- New classifiers focus on data that was previously misclassified by previous classifiers.
- The most popular boosting version, called AdaBoost.
- Boosting is different from bagging because the output is calculated from a weighted sum of all classifiers.

Boosting



Classifier-1
Dataset
Classifier-2
Misclassified Data
Ensemble
Final Prediction
Classifier-3
Misclassified Data

AdaBoost (Adaptive Boosting)

- Question : can we take a weak classifier and use multiple instances of it to create a strong classifier?
  - Weak classifier : its error rate is greater than 50% in the two-class case.
  - Strong classifier : will have a much lower error rate.

AdaBoost (Adaptive Boosting)

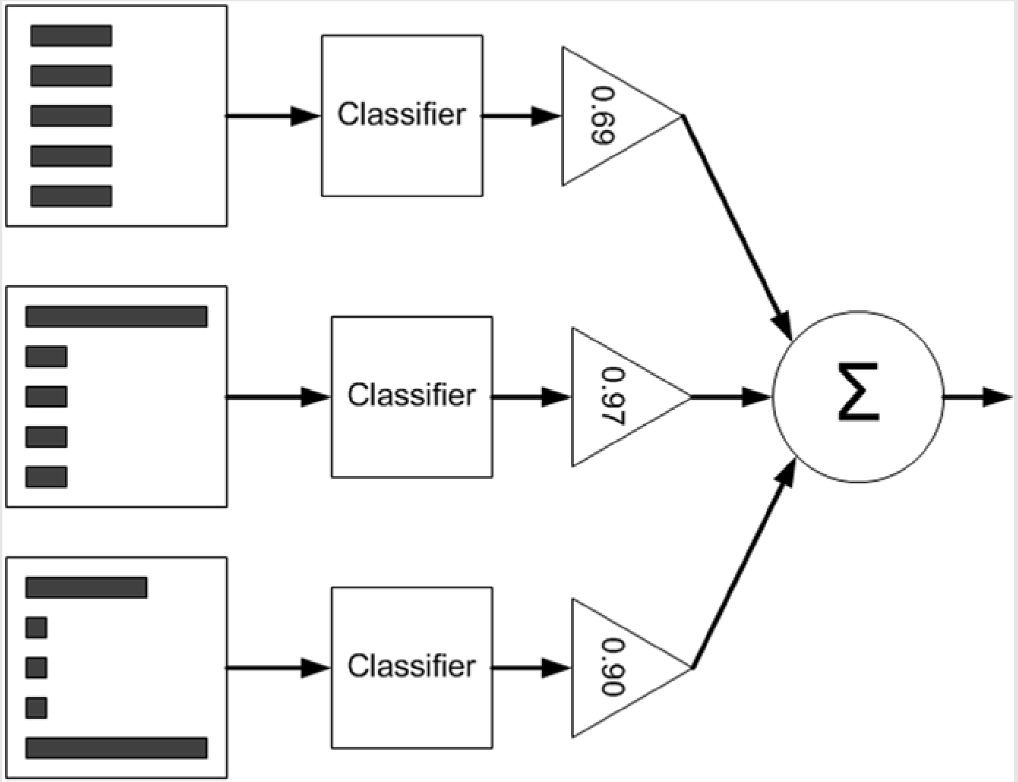
AdaBoost

- Most widely used form of boosting is the AdaBoost algorithm
- Boosting can give good results even if base classifiers have performance, only slightly better than random
  - Hence base learners are called weak learners
- Boosting can be extended to regression

AdaBoost

- A weight is applied to every example in the training data.
- Initially, these weights are all equal.
- A weak classifier is first trained on the training data.
- The errors from the weak classifier are calculated, and the weak classifier is trained a second time with the same dataset.
- This second time the weak classifier is trained, the weights of the training set are adjusted so the examples properly classified the first time are weighted less and the examples incorrectly classified in the first iteration are weighted more.
- To get one answer from all of these weak classifiers, AdaBoost assigns α values to each of the classifiers.

AdaBoost



AdaBoost

- Error:

$$\varepsilon = \frac{\textit{number of incorrectly classified examples}}{\textit{total number of examples}}$$

- Alpha is given by

$$\alpha = \frac{1}{2} \ln\left(\frac{1-\varepsilon}{\varepsilon}\right)$$

- After you calculate α, the weight vector is updated

Boosting vs Bagging

- The base classifiers are trained in sequence
- base classifier is trained using a weighted form of the data set in which the weighting coefficient associated with each data point depends on the performance of the previous classifiers
  - Misclassified points are given greater weight when used to train the next classifier
- Once all classifiers have been trained, their predictions are combined through a weighted majority weighting scheme

Handling Imbalanced Datasets

What is Imbalanced Data

- datasets where the target class has an uneven distribution of observations
- one class label has a very high number of observations and the other has a very low number of observations.
- Example
  - A bank is concerned that some fraudulent transactions are going
  - when the bank checks their data they found that for each 2000 transaction there are only 30 of fraud recorded.
  - So, the number of fraud per 100 transactions is less than 2%, or we can say more than 98% transaction is “No Fraud” in nature.
  - Here, the class “No Fraud” is called the **majority class,**
  - and the much smaller in size “Fraud” class is called the **minority class**.

What is Imbalanced Data

- More such example of imbalanced data is:
  - Disease diagnosis
  - Customer churn prediction
  - Fraud detection
  - Natural disaster

- Class imbalanced is generally normal in classification problems.
- But, in some cases, this imbalance is quite critical where the majority class’s presence is much higher than the minority class.

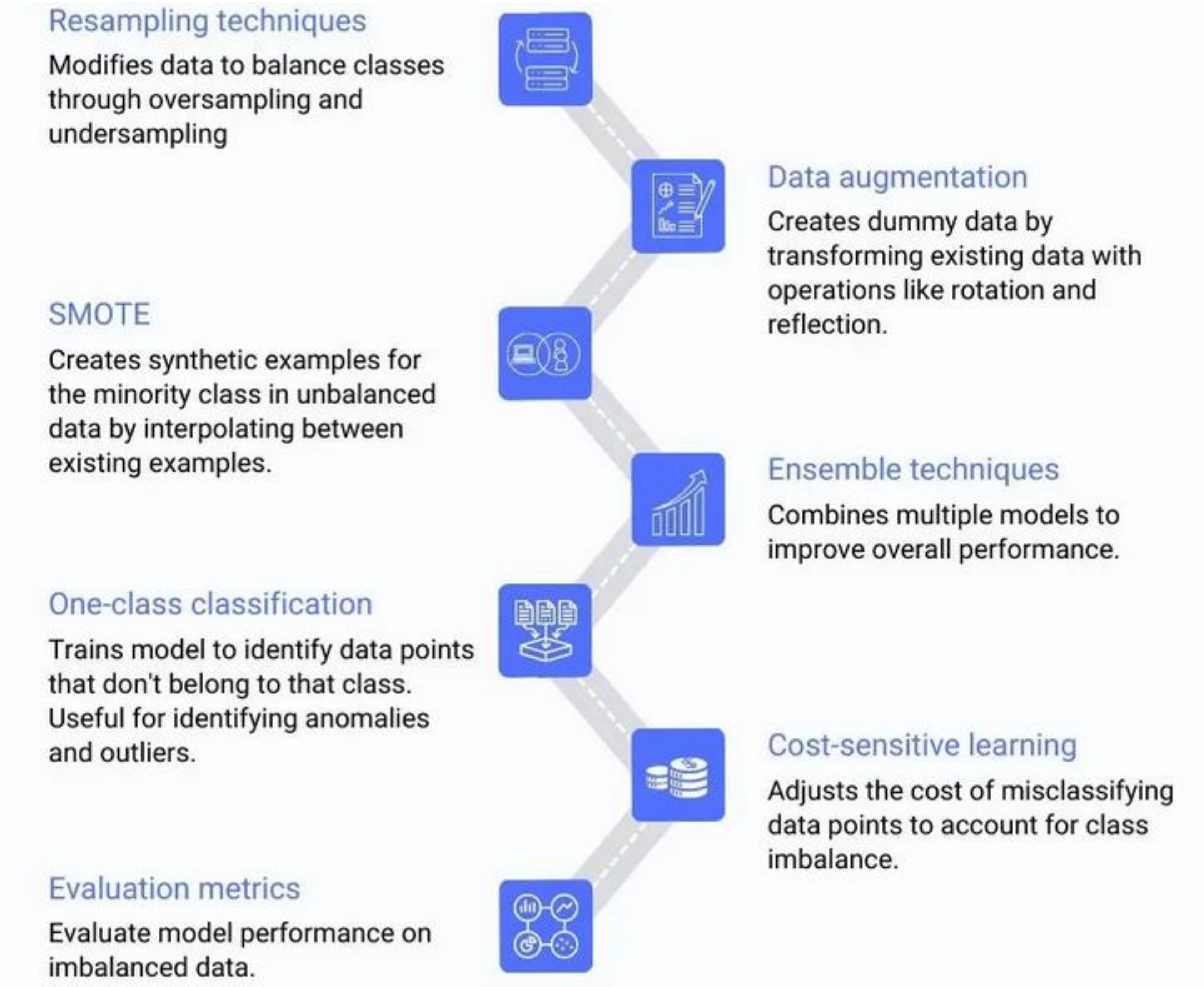
Problems with Handling Imbalanced Data Classification

- Example of disease diagnosis
  - we are going to predict disease from an existing dataset where for every 100 records only 5 patients are diagnosed with the disease.
  - the majority class is 95% with no disease and the minority class is only 5% with the disease.
  - assume our model predicts that all 100 out of 100 patients have no disease.
  - the accuracy of the model from the confusion matrix is the total no of correct predictions by the model divided by the total no of predictions.

	Original	
Prediction	Positive	Negative
Positive	True Positive	False Positive
Negative	False Negative	True Negative

ACCURACY = TP + TN / TP+TN+FP+FN

  - In the above case it is (0+95)/(0+95+0+5)=0.95 or 95%. It means that the model fails to identify the minority class yet the accuracy score of the model will be 95%.


Resampling techniques
Modifies data to balance classes through oversampling and undersampling
Data augmentation
Creates dummy data by transforming existing data with operations like rotation and reflection.
SMOTE
Creates synthetic examples for the minority class in unbalanced data by interpolating between existing examples.
Ensemble techniques
Combines multiple models to improve overall performance.
One-class classification
Trains model to identify data points that don't belong to that class. Useful for identifying anomalies and outliers.
Cost-sensitive learning
Adjusts the cost of misclassifying data points to account for class imbalance.
Evaluation metrics
Evaluate model performance on imbalanced data.

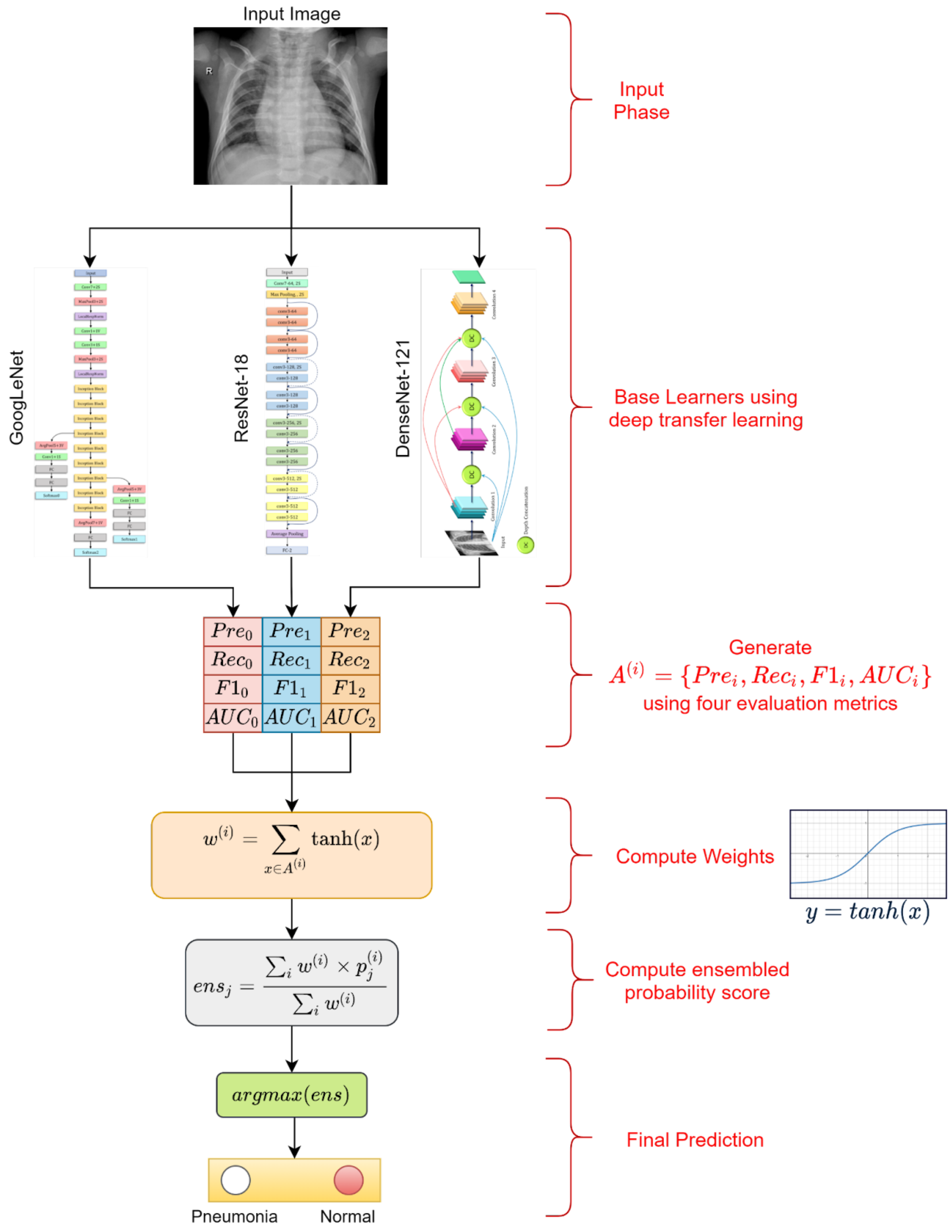
Handling Imbalanced Datasets

- Solutions:
  - Assign weights to classes
    - Assign a weight for every class.
    - The learning algorithm takes this information into account when looking for the best hyperplane.
  - Oversampling the minority class:
    - If a learning algorithm doesn't allow weighting classes, you can try the technique of over-sampling.
    - It consists of increasing the importance of examples of some class by making multiple copies of the examples of that class.
  - Undersampling the majority class
    - Randomly remove from the training set some examples of the majority class.
  - Synthetic oversampling (resampling):
    - SMOTE
    - ADASYN

Handling Imbalanced Datasets

- **Data augmentation**
  - applying various transformations such as rotations, translations, and flips to the existing data.
- **Synthetic minority over-sampling technique (SMOTE)**
  - is a type of oversampling technique
  - involves creating synthetic examples of the minority class.
- **Ensemble techniques**
  - involve combining multiple models to improve performance.
  - bagging, boosting, and stacking.

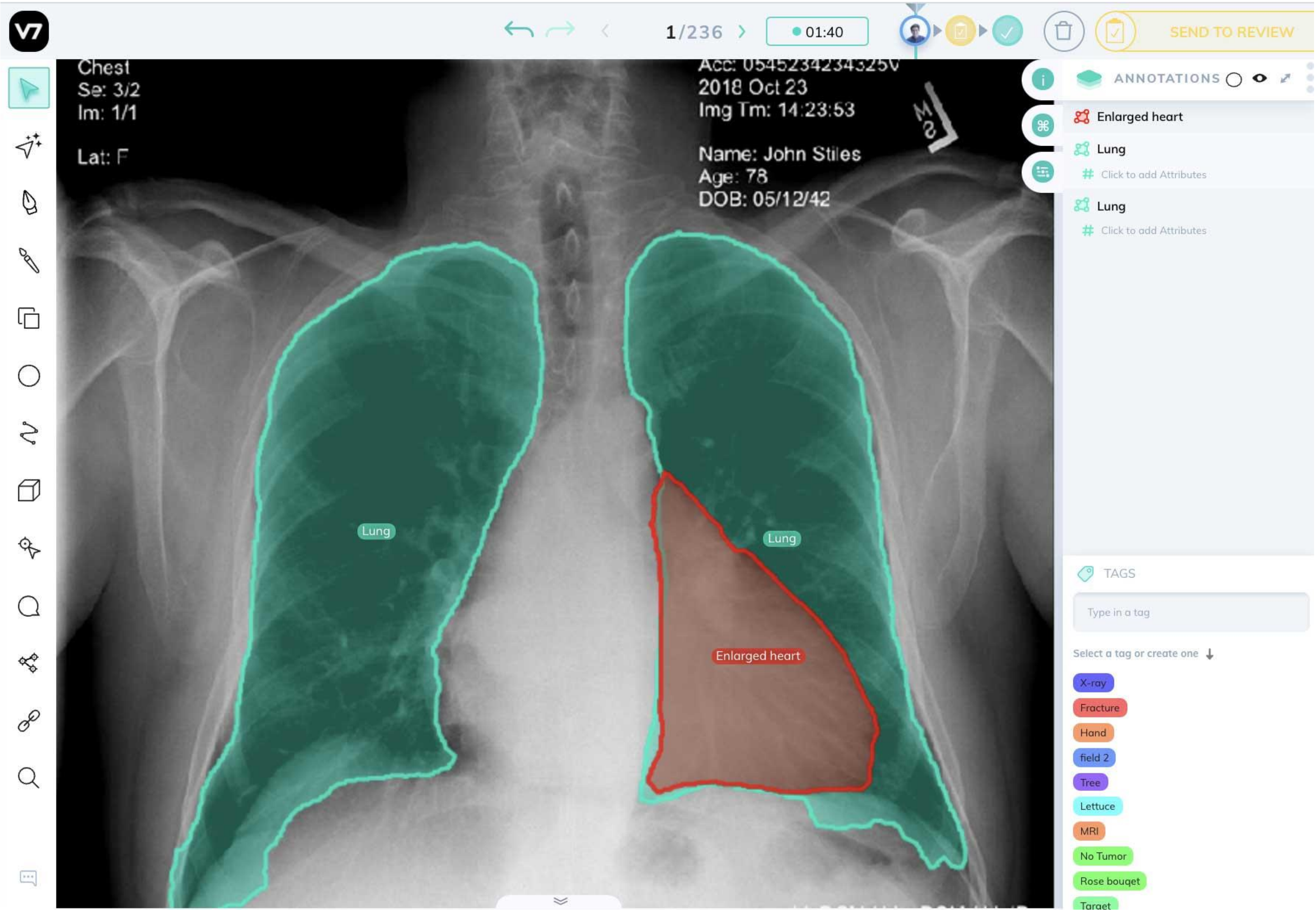
Example



Input Image
Input Phase
GoogLeNet
ResNet-18
DenseNet-121
Base Learners using deep transfer learning
Pre_0 Pre_1 Pre_2
Rec_0 Rec_1 Rec_2
$F1_0$ $F1_1$ $F1_2$
AUC_0 AUC_1 AUC_2
Generate
$A^{(i)} = \{Pre_i, Rec_i, F1_i, AUC_i\}$
using four evaluation metrics
$w^{(i)} = \sum_{x \in A^{(i)}} \tanh(x)$
Compute Weights
$y = tanh(x)$
$ens_j = \frac{\sum_i w^{(i)} \times p_j^{(i)}}{\sum_i w^{(i)}}$
Compute ensembled probability score
$argmax(ens)$
Final Prediction
Pneumonia
Normal

Applications

- **Disease detection**

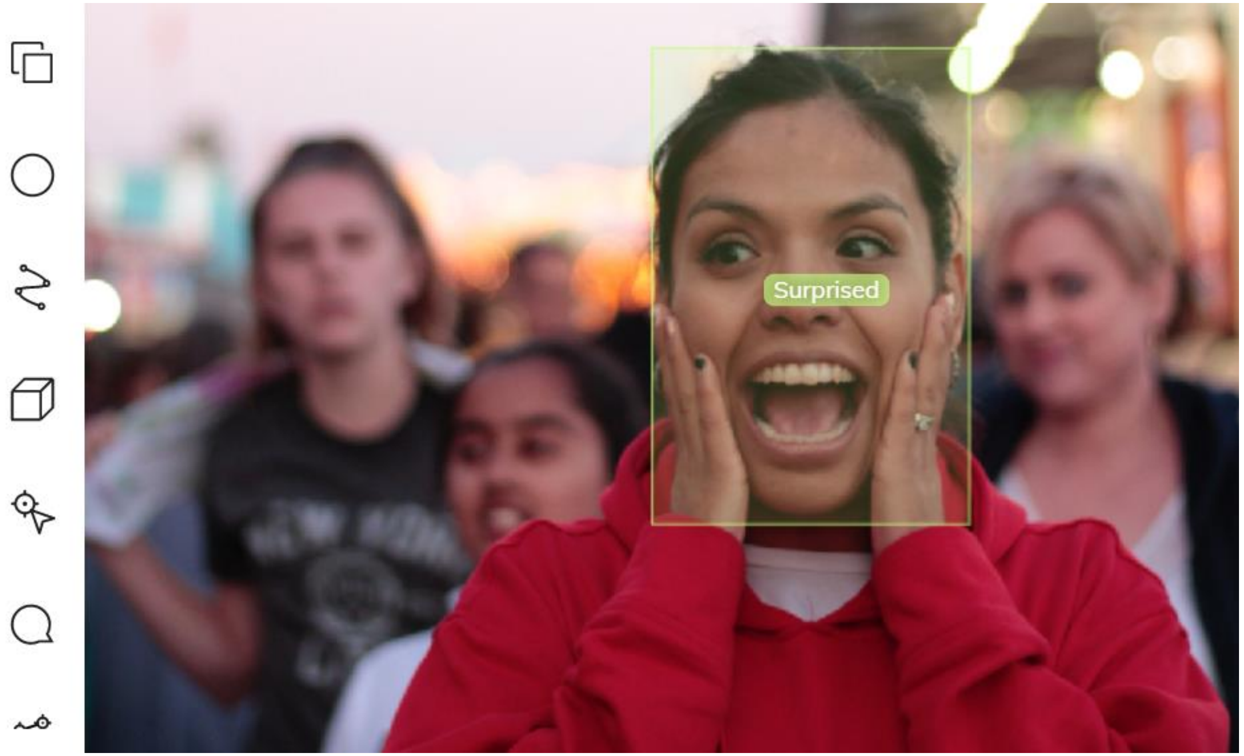


Applications

- **Remote Sensing**
- Monitoring of physical characteristics of a target area without coming in physical contact,

Applications

- **Speech emotion recognition**



Libraries for ensemble learning

- from sklearn.ensemble import BaggingClassifier
- from sklearn.tree import DecisionTreeClassifier
- bagging = BaggingClassifier(base_estimator=DecisionTreeClassifier(),n_estimators=10, max_samples=0.5, max_features=0.5)

- from sklearn.ensemble import BaggingRegressor
- bagging = BaggingRegressor(DecisionTreeRegressor())
- bagging.fit(X_train, y_train)
- model.score(X_test,y_test)

Libraries for ensemble learning

- from sklearn.ensemble import AdaBoostClassifier
- model = AdaBoostClassifier(n_estimators=100)
- model.fit(X_train, y_train)
- model.score(X_test,y_test)

- from sklearn.ensemble import GradientBoostingClassifier
- model = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0, max_depth=1, random_state=0)
- model.fit(X_train, y_train)
- model.score(X_test,y_test)