UNIVERSITY of SOUTH FLORIDA

Robust Models

Spring 2020

Instructor: Ankit Shah, Ph.D.

Factors Affecting Model Performance

Class Imbalance

- Suppose you have a training data set with multiple (discrete) classes as your response variable values
- Say, the relative frequencies of these classes are unequal
- You apply one of the models to fit the training data set
- Will this impact your model's training and making predictions for the data samples in your testing data set?

- If there was a significantly low proportion of one (or more) of the classes in your training data set, then it is a class imbalance issue that may affect the performance of your model

Class Imbalance

- How did it affect the performance of the model?
  - The class imbalance will bias predictions to the majority class

Confusion Matrix

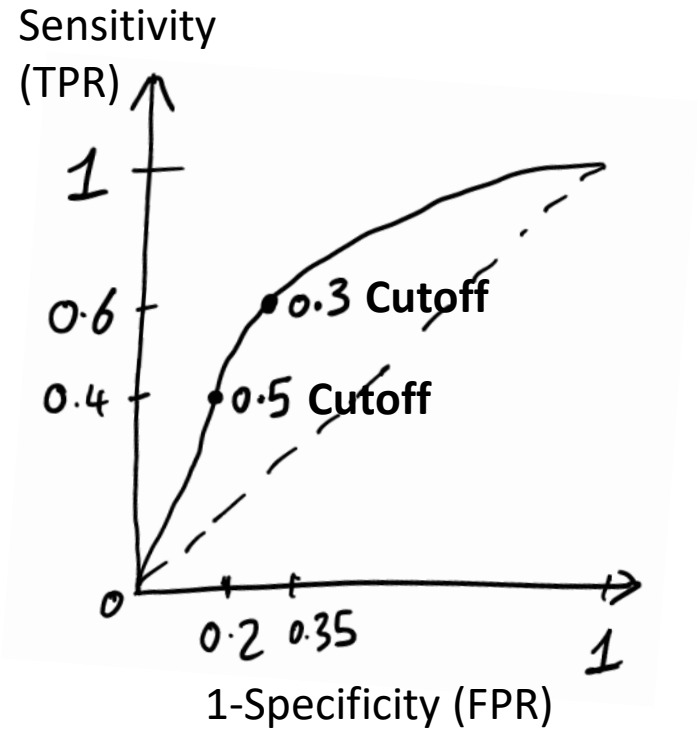
Confusion matrix is a cross-tabulation of the observed and predicted classes for the data

	Observed	
Predicted	Event	Nonevent
Event	**True Positives (TP)**	**False Positives (FP)**
Nonevent	**False Negatives (FN)**	**True Negatives (TN)**

- Top row corresponds to samples predicted to be events
- Bottom row corresponds to samples predicted to be nonevents
- Sensitivity = $\frac{TP}{TP+FN}$
- Specificity = $\frac{TN}{TN+FP}$

Receiver Operating Characteristic (ROC) Curve

- In Spam Filtering, whether to classify an email as Junk or Not, the focus is on Specificity
- Specificity = $\frac{TN}{TN+FP}$
- False Positives need to be minimized
- More important to minimize deletion of valid emails by incorrect classification as Junk
- Trade-off between reducing FP (Specificity) and FN (Sensitivity)
- Receiver Operating Characteristic (ROC) curve is one technique for evaluating this trade-off



- ROC curve is a graphical representation of how Sensitivity and Specificity vary as we change the probability threshold
- It can be thought of as a summarization of all the confusion matrices that can be obtained by varying the threshold from 0 to 1

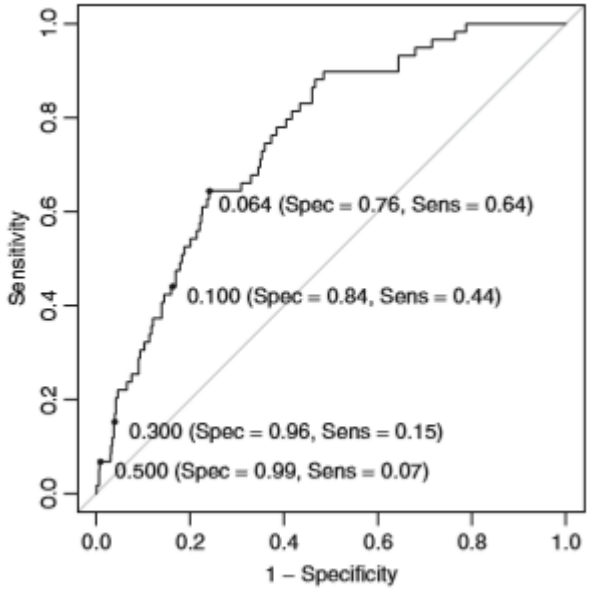
Class Imbalance: Alternate Cutoffs

Class Imbalance: Alternate Cutoffs

Improve the prediction accuracy of the minority class samples

- For a 2-class classification problem, one method for improving the performance of the model is to determine alternate cutoffs for the predicted probabilities that define a predicted event

- An approach is to use the ROC curve since it calculates the sensitivity and specificity across a continuum of cutoffs



- If we decrease the cutoff, it will increase the sensitivity at the expense of specificity
- Perhaps, 0.064 might be a good cutoff here – giving a good balance between the two
- Comparing models based on default sensitivity and specificity might be misleading in many classification problems
  - A better cutoff may be possible based on the ROC curve

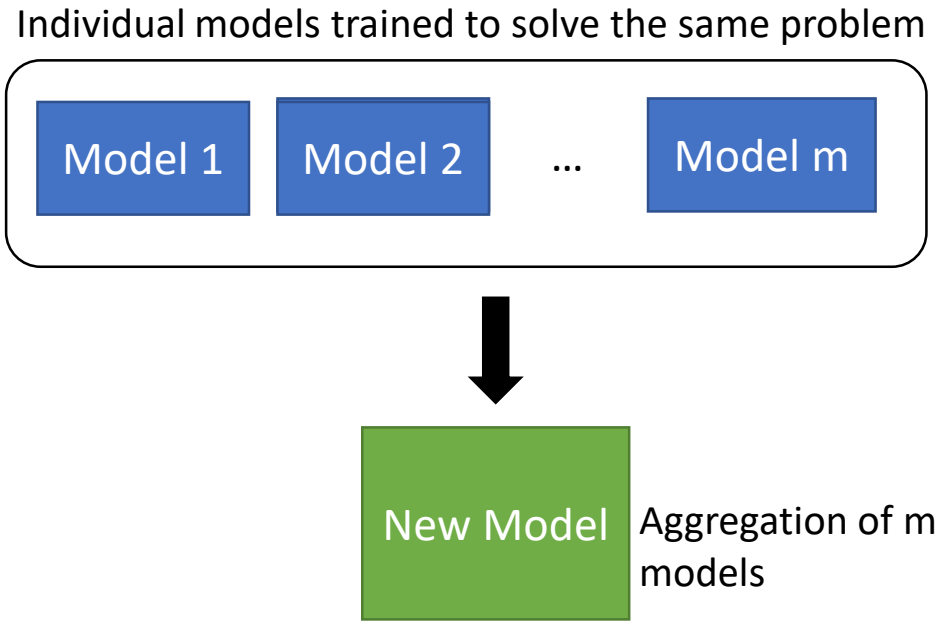
This is a post-processing method to redefine the class predictions!

Class Imbalance:
Assigning Different Weights to the Samples

Class Imbalance: Assigning Different Weights to the Samples

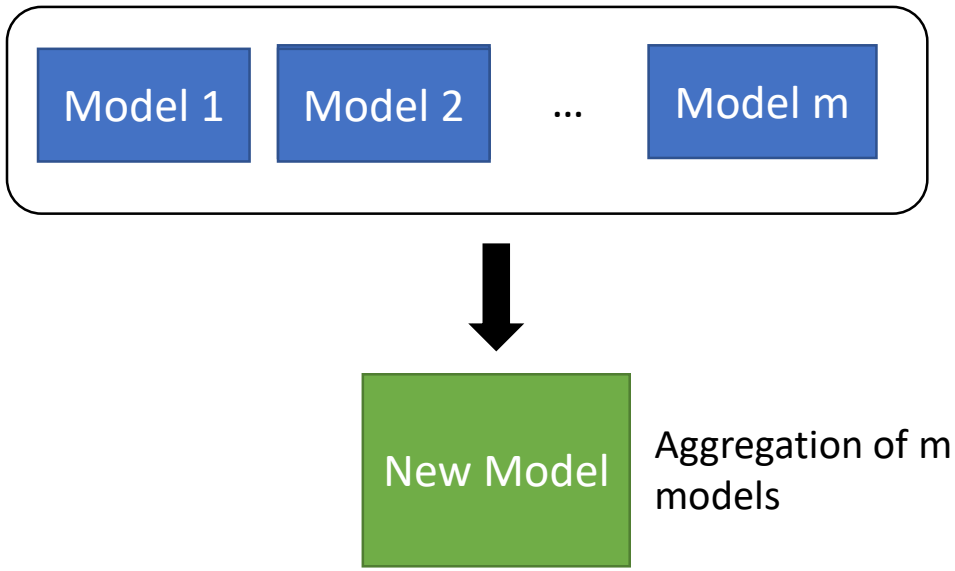
- Another method to improve the model performance due to the issue of class imbalance is by assigning more emphasis on the individual data points from the minority class (which will be incorrectly classified) during the training phase
- An example of such a method is – Boosting

- Boosting is an ensemble technique
- In boosting, the "weak" learners are fit sequentially
- Data that was not handled correctly in the previous model becomes the focus of the current model
- This process continues until we obtain a "strong" learner, which has **a low bias**

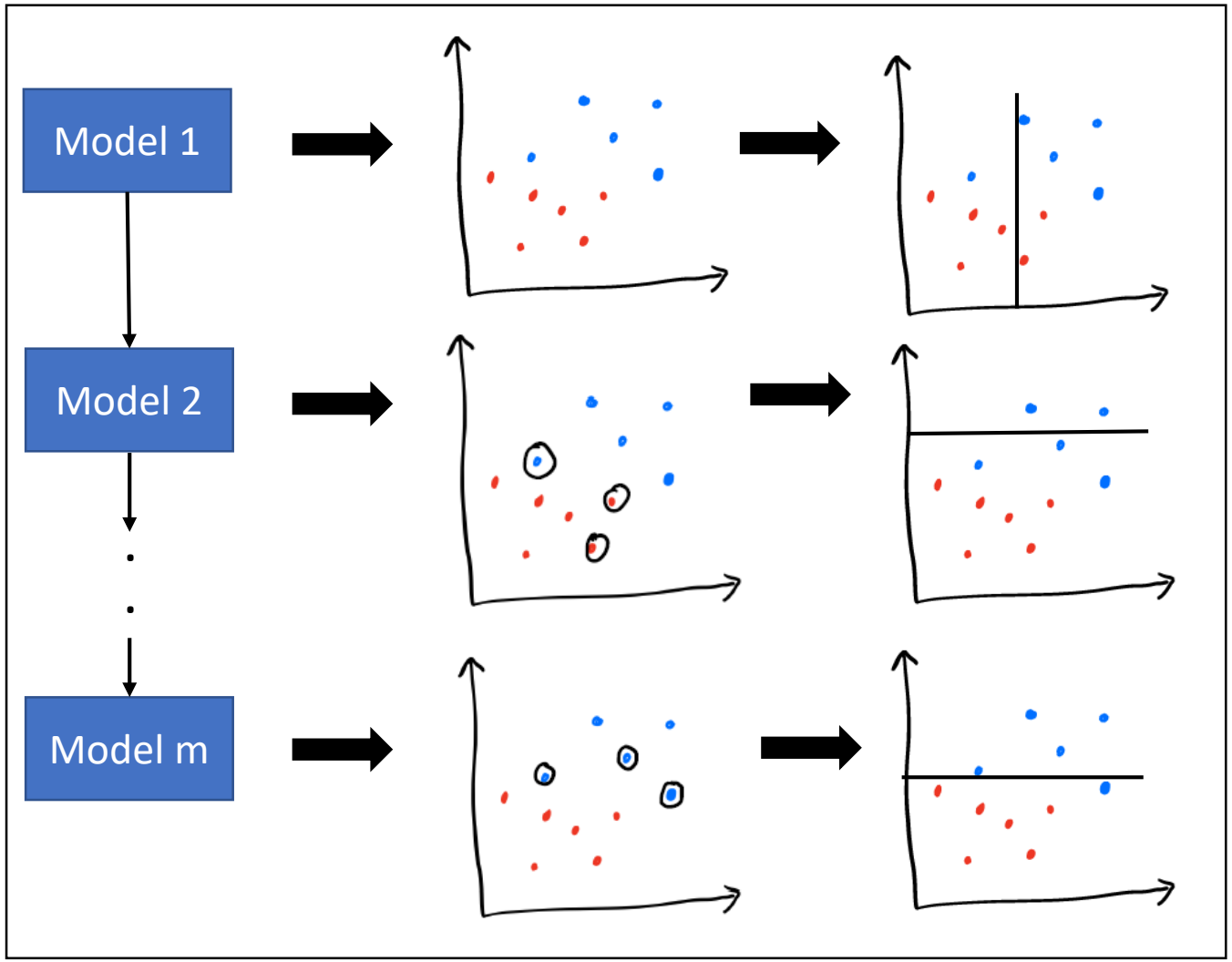


Boosting

Individual models trained to solve the same problem



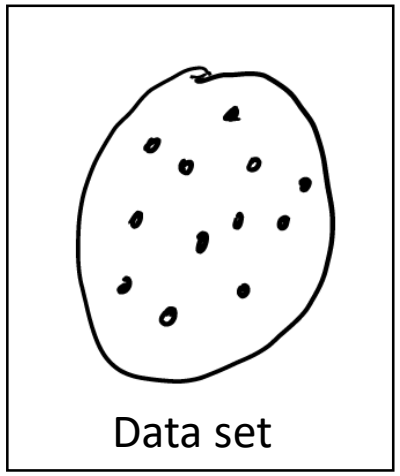
Aggregation of m models

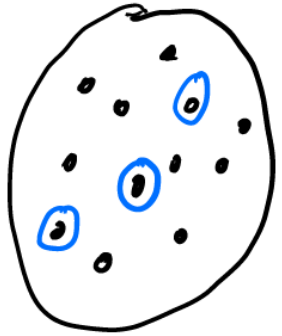


Adaptive Boosting - Classification

Model 1

Assign each sample an equal weight (1/n)

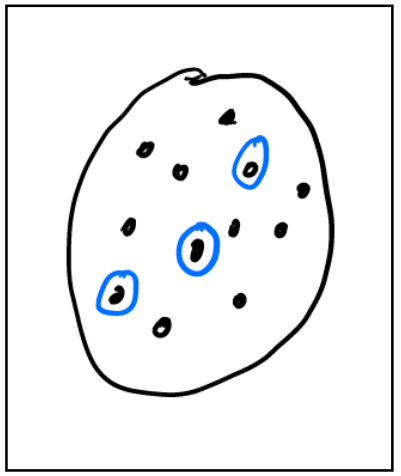


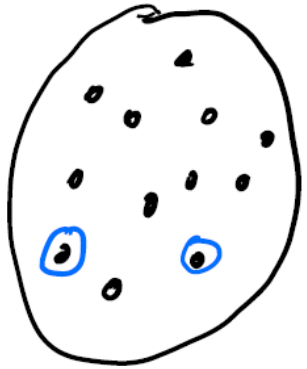


Record the misclassification rate (ε) and compute the model's value:
½ * ln((1- ε)/ε)

Model 2

Assign higher weights to the misclassified samples





Record the misclassification rate (ε) and compute the model's value:
½ * ln((1- ε)/ε)

Model m

Upon Training:

Model	ε	Value
1	0.27	0.424
2	0.65	-0.310
3	0.5	0

To classify a sample:

Predicted Class
1
-1
-1

$$\sum_{m} Predicted\ Class\_m * Value\_m$$

1*0.424 + (-1)*(-0.310) + (-1)*0

If the above sum is positive, then class is 1
Else, class -1

Class Imbalance: Sampling Methods

Class Imbalance: Sampling Methods

- One of the easiest way to sample is by selecting a training set where there are roughly equal event rates
  - In other words, instead of the model trying to balance the class frequencies, we can handpick the balanced frequencies
- This may not be possible in all cases, as there might be a significant difference between the frequencies of the classes and there may not be a good size training data set that could be obtained with this approach
- In this case, what are our options?

- There are 2 very important post hoc approaches that can help reduce the effects of an imbalanced data set
  - Up-sampling
  - Down-sampling

Class Imbalance: Sampling Methods

Up-sampling

- A technique that simulates or imputes additional data points to improve balance across classes
- One approach is to keep sampling cases from the minority class with replacement until the classes are balanced
- Issue: some data points will show up in the training data set multiple times while the majority class will have data points that are unique

Down-sampling

- A technique that reduces the number of samples from the majority class to balance the classes
- One approach is to randomly sample the majority class such that both the classes are balanced

Class Imbalance: Sampling Methods

Synthetic Minority Over-sampling TEchnique (SMOTE)

- A data sampling technique that uses up-sampling
- To up-sample for the minority class, this technique simulates new data points
  - It randomly selects a data point from the minority class
  - It finds the K-nearest neighbors of the data point
  - It creates a new synthetic data point

Python code to apply up-sampling technique:

```
from imblearn.over_sampling import SMOTE
sm = SMOTE(k_neighbors=5)
x_training_set, y_training_set = sm.fit_sample(x_training_set,y_training_set)
```

What would happen if the up-sampling procedure is done prior to the data being split into training and test (holdout) sets?

Class Imbalance: Sampling Methods

Near Miss

- A data sampling technique that performs down-sampling
- To down-sample:
  - It calculates the distances between all instances of the majority class and the instances of the minority class
  - It finds n instances of the majority class that are the closest to each of the instances of the minority class – i.e., if there are k instances in minority class, it will find up to k*n instances of the majority class
  - It uses a triage rule to select the equal number of instances from the majority class to match the number of instances of the minority class, i.e., k number of instances

Python code to apply down-sampling technique:

```
from imblearn.under_sampling import NearMiss
nm = NearMiss(n_neighbors=3)
x_training_set, y_training_set = sm.fit_sample(x_training_set,y_training_set)
```