Resampling Methods

DR. BHARGAVI
ASSOCIATE PROFESSOR
VIT CHENNAI

Introduction

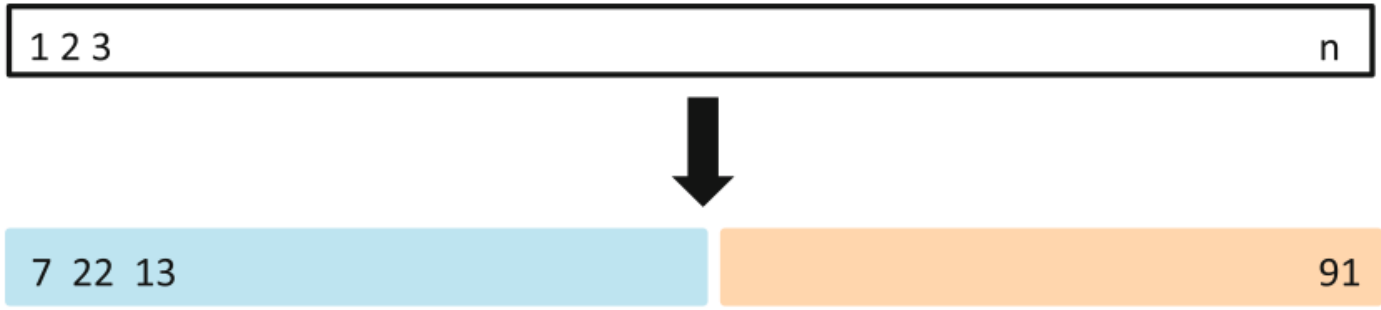
- In order to obtain additional information about the model (for example: estimated variability of Linear Regression), Resampling methods are helpful.
- Resampling methods draw samples repeatedly from a training set and refit a model of interest on each sample in order to obtain additional information about the fitted model.
- For example they provide estimate of test-set prediction error, standard deviation and bias of parameter estimates.
- Two Resampling Methods
  - Cross-Validation
  - Bootstrap

Training Error vs Test Error

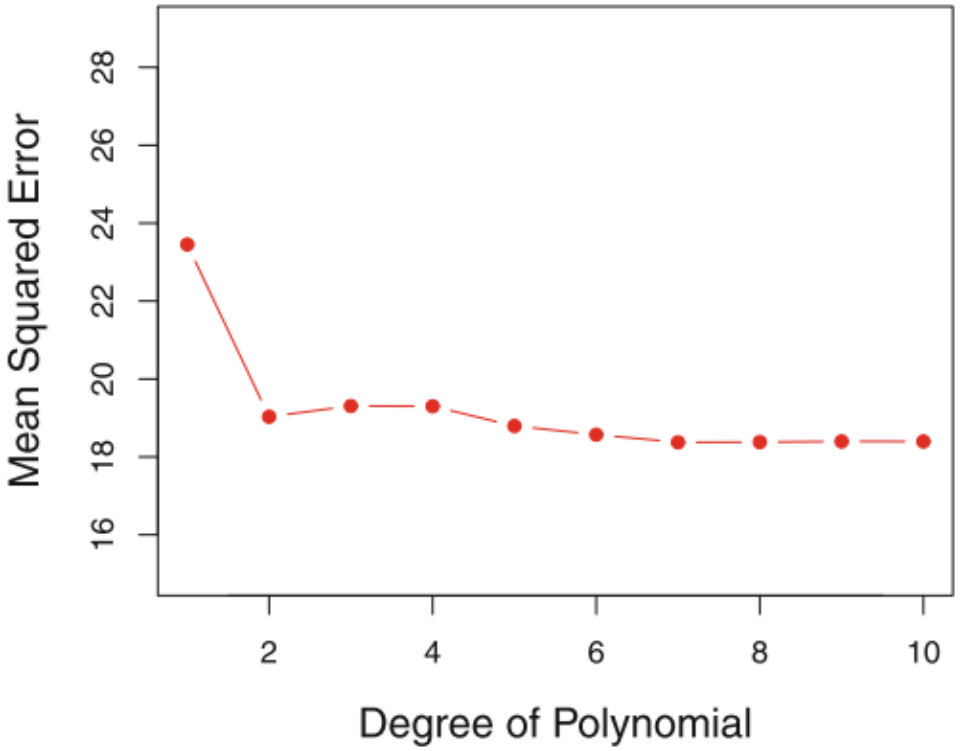
- **Test error** is the average error that results from using a Machine learning method to predict the response on a new observation, one that was not used in training the method.
- **Training error** can be easily calculated by applying the Machine learning method to the observations used in its training.
- Training error rate often is quite different from the test error rate, and in particular the training error can dramatically underestimate the test error.

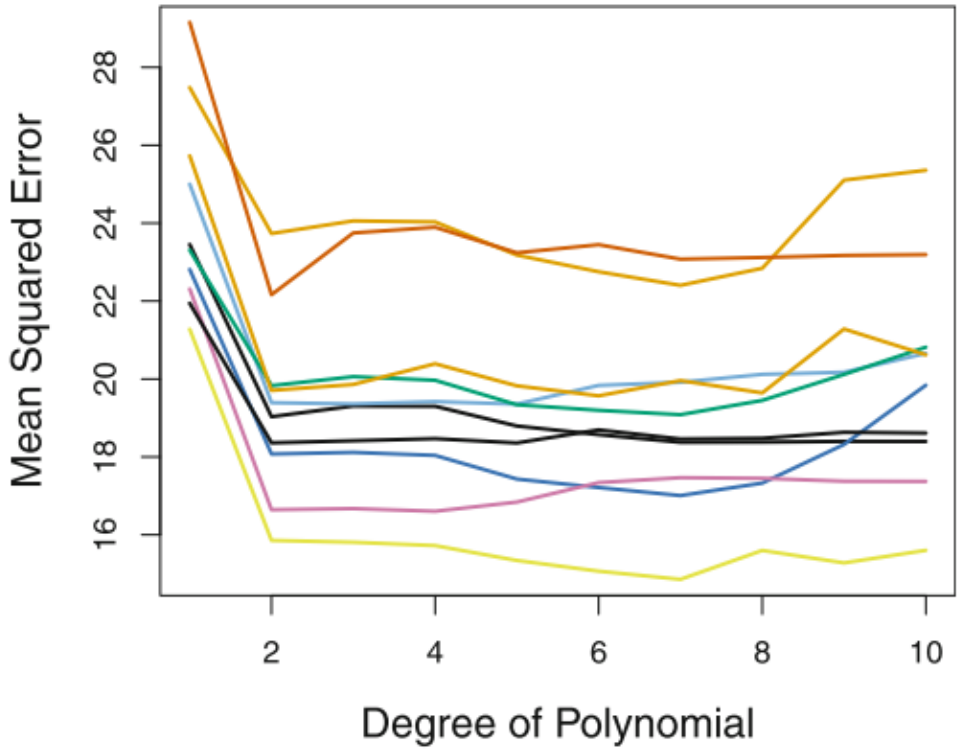
Cross Validation - Validation Set Approach (Recap)

- Available dataset is randomly divided into two parts: a training set and a validation or hold-out set.
- The model is fit on the training set, and the fitted model is used to predict the responses for the observations in the validation set.
- The resulting validation-set error provides an estimate of the test error.
- This is typically assessed using MSE in the case of a quantitative response and misclassification rate in the case of a qualitative (discrete) response.



Validation Set Approach (Cont…) (Test Error)





Validation Set Approach (cont…)

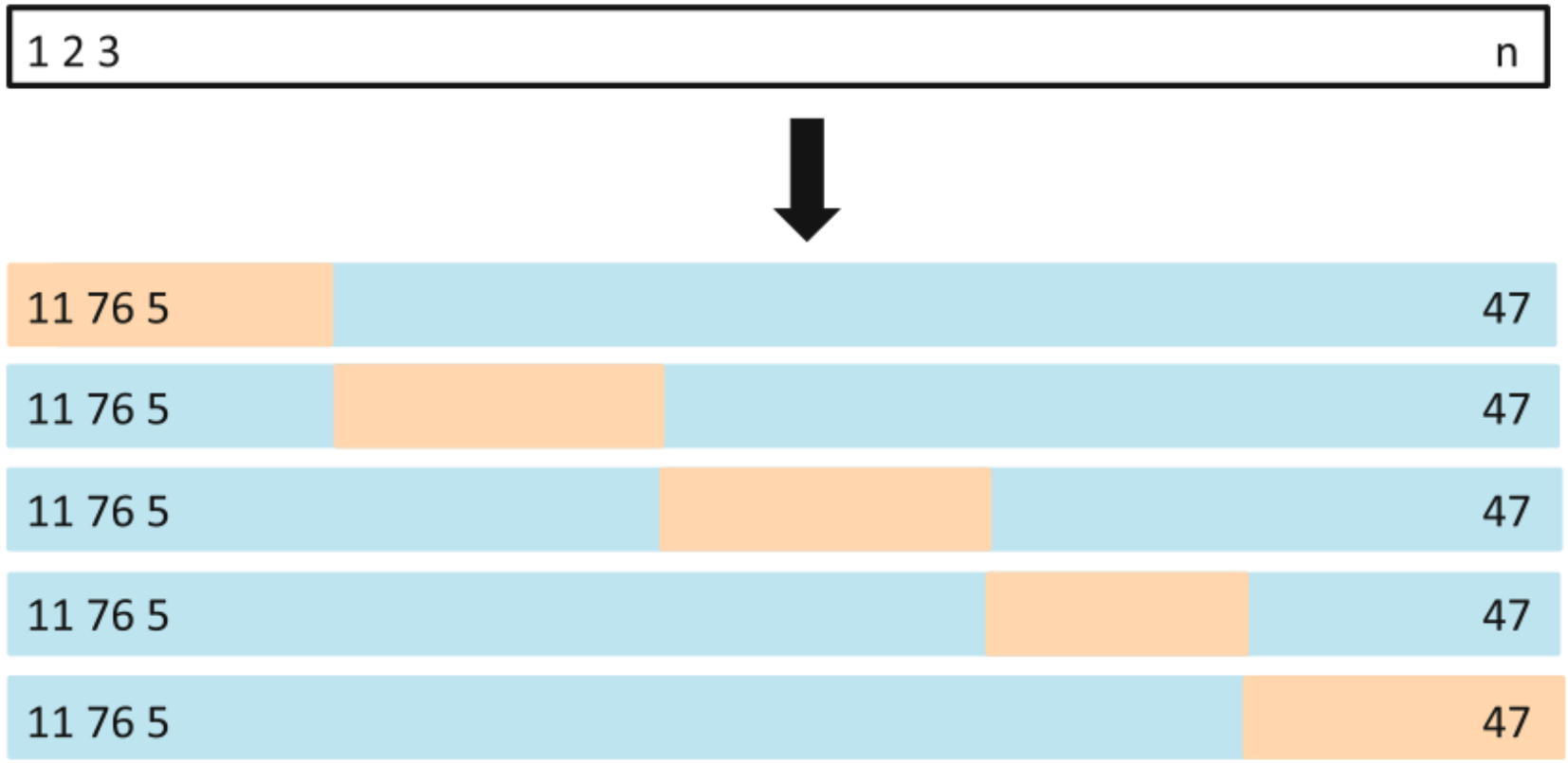
- The validation estimate of the test error can be highly variable, depending on precisely which observations are included in the training set and which observations are included in the validation set.
- In the validation approach, only a subset of the observations — those that are included in the training set rather than in the validation set — are used to fit the model.
- This suggests that the validation set error may tend to overestimate the test error for the model fit on the entire data set.

k-Fold Cross Validation

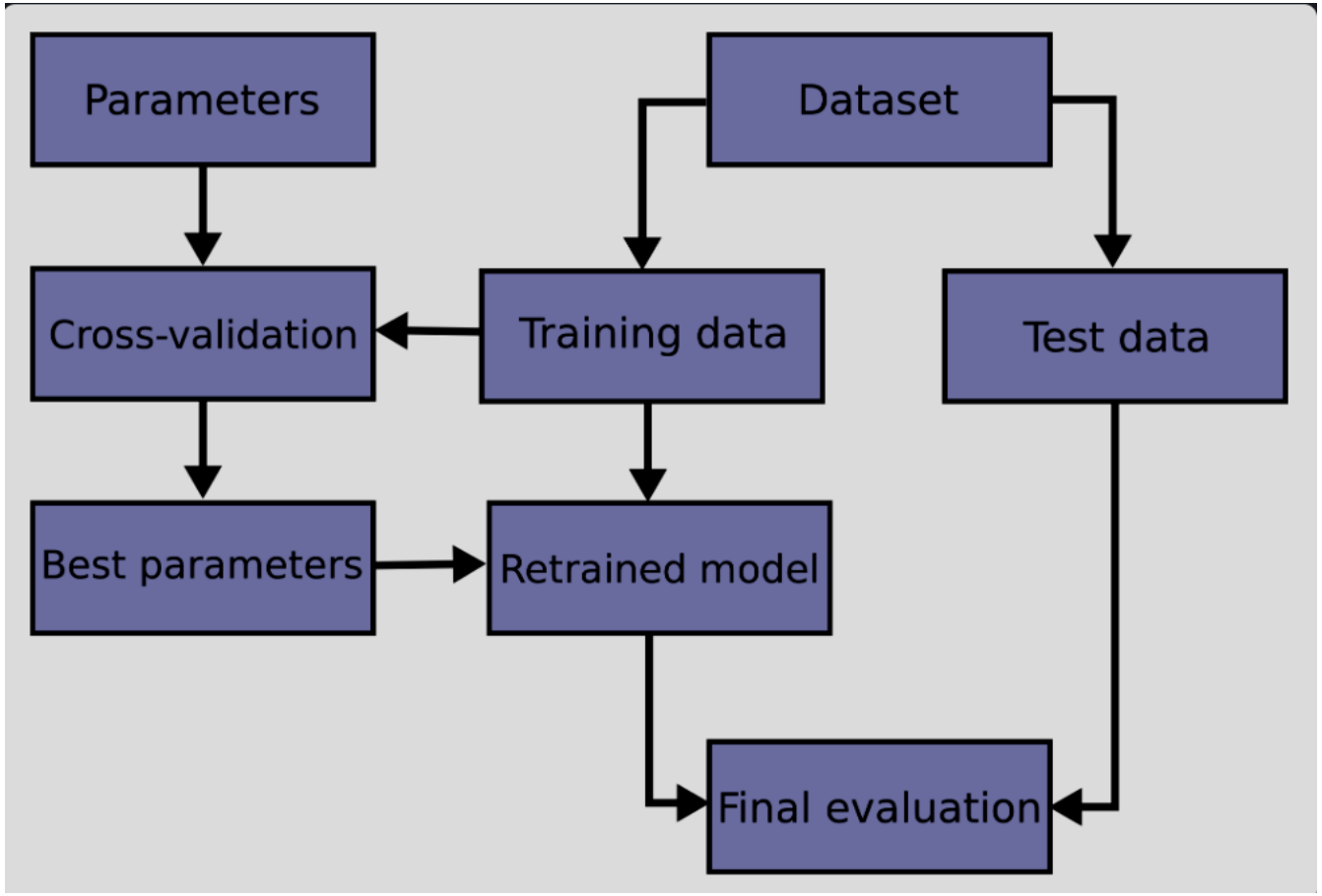
- Randomly divide the set of observations into k groups, or folds, of approximately equal size.
- The first fold is treated as a validation set, and the method is fit on the remaining $k - 1$ folds.
- The mean squared error, MSE1, is then computed on the observations in the held-out fold.
- This procedure is repeated k times; each time, a different group of observations is treated as a validation set.
- This process results in k estimates of the test error, MSE1, MSE2,...,MSEk. The k-fold CV estimate is computed by averaging these values

$$\mathrm{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k}\mathrm{MSE}_i.$$

k-Fold Cross Validation (cont…)



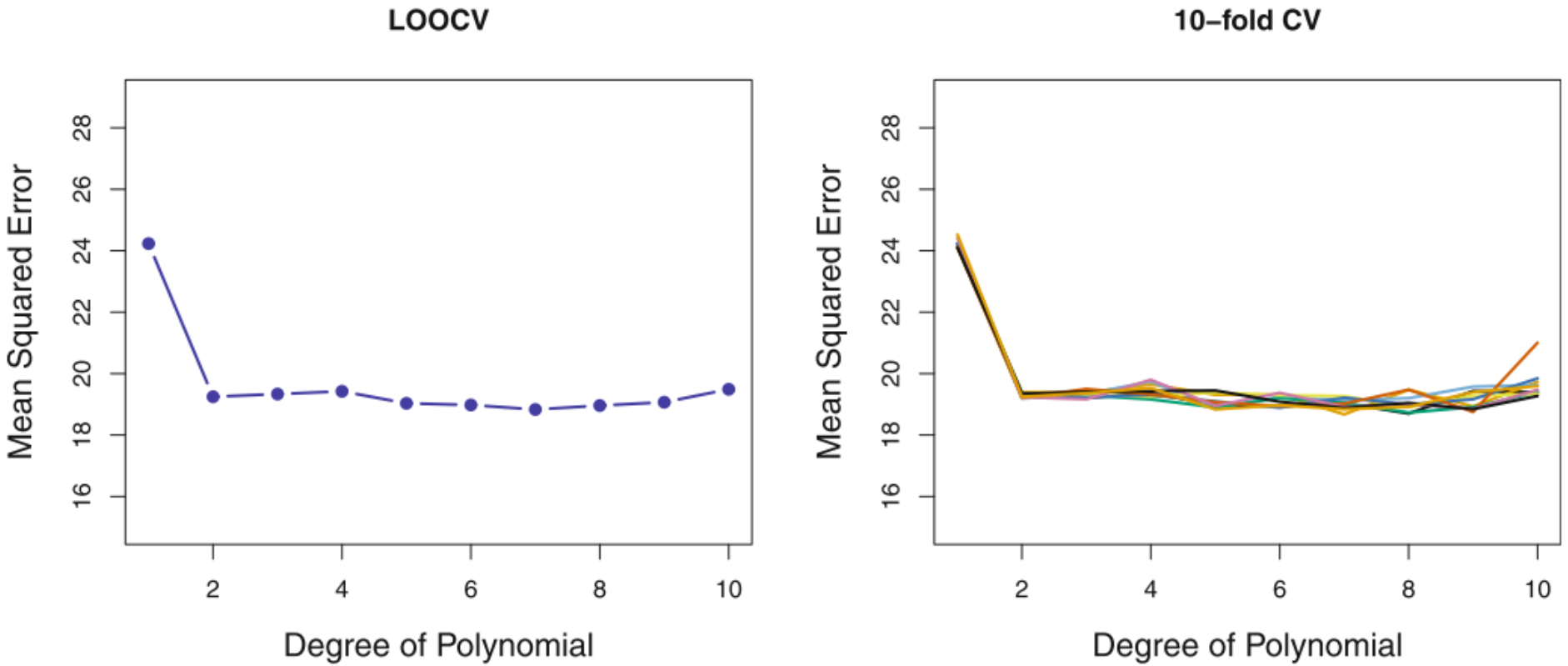
k-Fold Cross Validation (cont…)



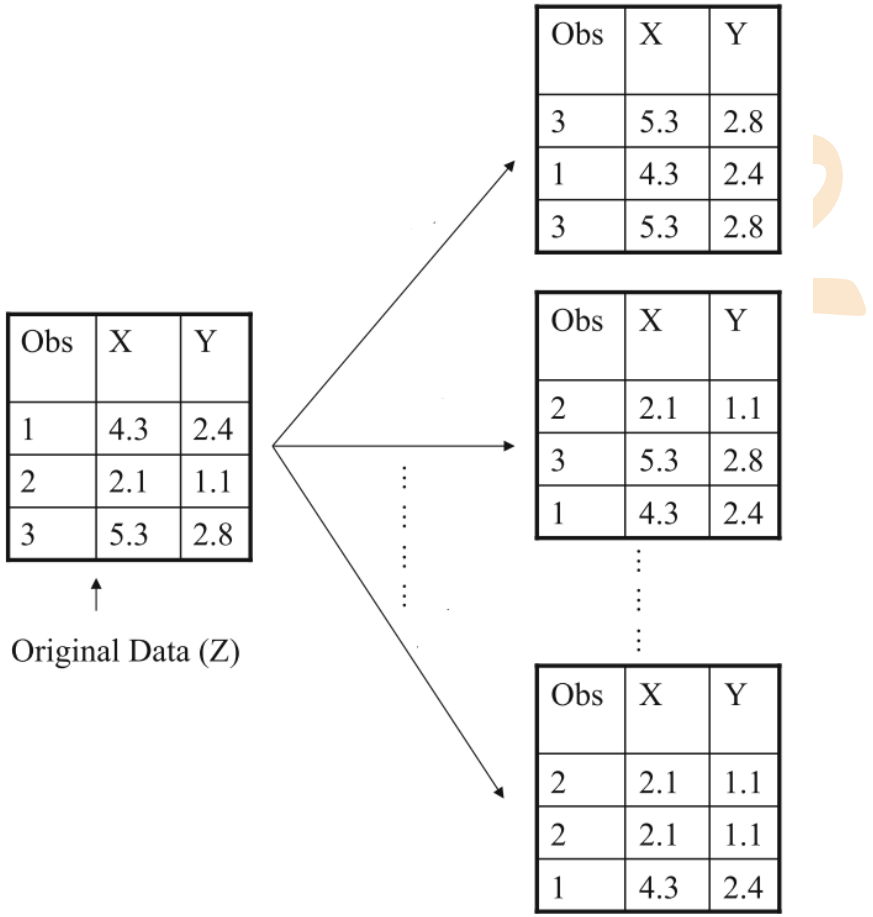
Leave-One-Out Cross-Validation

- Setting K = n yields n-fold or leave-one out cross-validation (LOOCV).
- Issue with k-Fold Cross validation : Since each training set is only (K − 1)/K as big as the original training set, the estimates of prediction error will typically be biased upward.
- Bias is minimized when K = n (i.e. LOOCV).
- LOOCV approach tends not to overestimate the test error rate as much as the validation set approach does.
- Performing LOOCV multiple times will always yield the same results as there is no randomness in the training/validation set splits.
- K = 5 or 10 provides a good compromise for this bias-variance tradeoff.

k-Fold Cross Validation (cont…)



Bootstrap Resampling Method



Handling Imbalanced Dataset

- Most of the real-world datasets are imbalanced.
- Normally we have more negative instances (majority class) of data compared to positive instances (minority class)(in the case of classification problem).
- The conventional model evaluation methods do not accurately measure model performance when faced with imbalanced datasets.
- Some of the practices for handling imbalnced data
  - Collect More Data
  - Changing Performance Metrics
  - Data Level approach (Resampling techniques)
  - Use Boosting / Bagging algorithms.

Under-Sampling

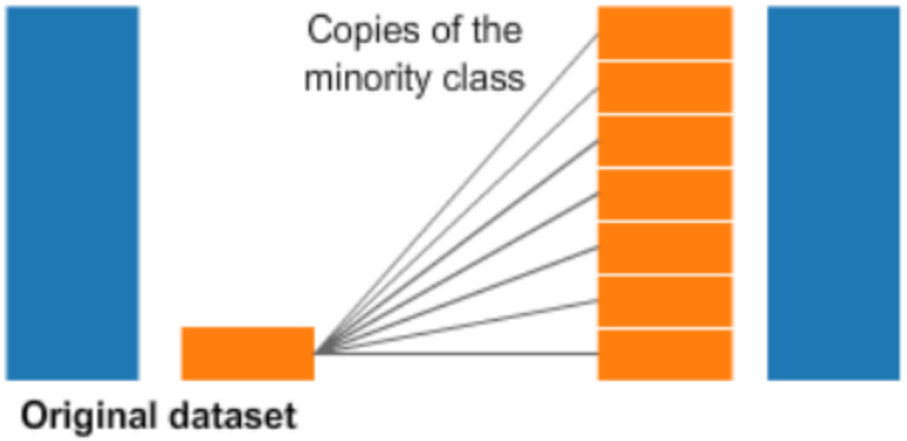
- It aims to balance class distribution by randomly eliminating majority class examples.

Disadvantages:

- It can discard potentially useful information which could be important.
- The sample chosen by random under sampling may be a biased sample and it will not be an accurate representative of the population. Thereby, resulting in inaccurate results with the actual test data set.

Over-Sampling

- It increases the number of instances in the minority class by randomly replicating them in order to present a higher representation of the minority class in the sample.
- Add copies of instances from the under-represented class. (sampling with replacement).



Over-Sampling (cont…)

Advantages

- Unlike under sampling this method leads to no information loss.
- Outperforms under sampling

Disadvantages

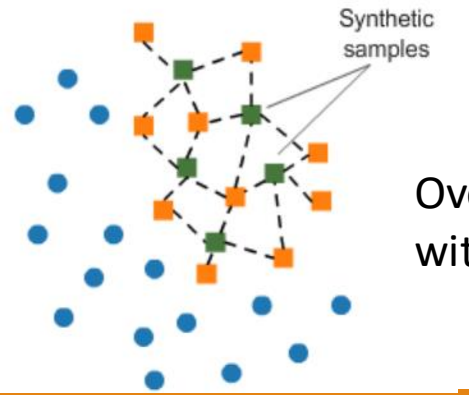
- It increases the likelihood of overfitting since it replicates the minority class instances.
- Increases the size of the training set and the time to build a classifier.

Generate Synthetic Samples (SMOTE)

Synthetic Minority Oversampling Technique (SMOTE):

- SMOTE involves synthesizing the instances for the minority class, based on those that already exist.
- SMOTE works by randomly picking a point from the minority class and computing the k-nearest neighbors for this point.
- The synthetic points are added between the chosen point and its neighbors.

Original Data



Oversampled Data
with SMOTE

Confusion Matrix

		Predicted Class	
		C_1 (+)	C_2 (-)
Actual Class	C_1 (+)	True positive (TP)	False negative(FN)
	C_2 (-)	False positive (FP)	True negative (TN)

Performance Metrics

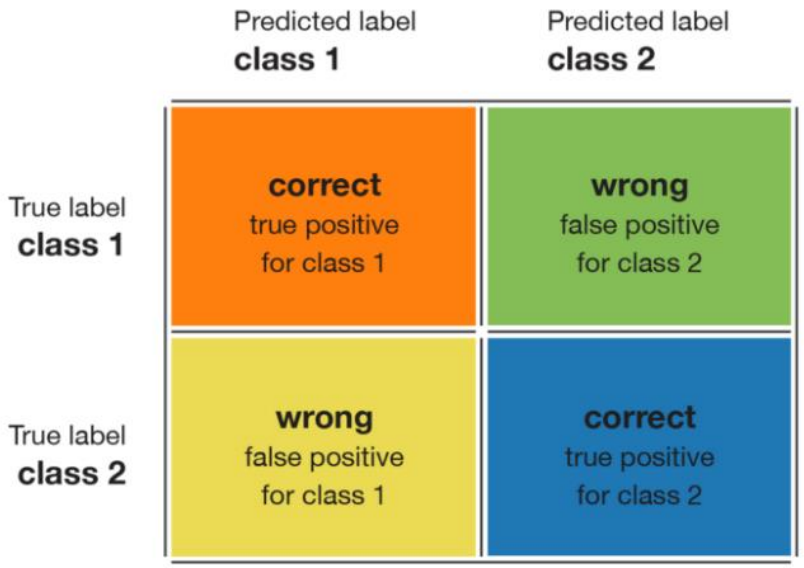
Precision = TP/(TP + FP)

Recall/ Sensitivity = TP/(TP +FN) /* true positive recognition rate(TPR) */

Specificity = t-neg/neg (i.e TN/N) /* true negative recognition rate(TNR) */
or TN/(TN +FP)

Accuracy = sensitivity * pos/(pos + neg) + specificity * neg/(pos + neg)
Here pos = TP + FN
neg = TN + FP

Confusion Matrix (cont…)



$$\text{accuracy} = \frac{\blacksquare + \blacksquare}{\blacksquare + \blacksquare + \blacksquare + \blacksquare}$$

$$\text{class 1 precision} = \frac{\blacksquare}{\blacksquare + \blacksquare}$$

$$\text{class 2 precision} = \frac{\blacksquare}{\blacksquare + \blacksquare}$$

$$\text{class 1 recall} = \frac{\blacksquare}{\blacksquare + \blacksquare}$$

$$\text{class 2 recall} = \frac{\blacksquare}{\blacksquare + \blacksquare}$$

Performance Evaluation Metrics - Classification

Confusion Matrix

	Predicted label Class 1/ Positive	Predicted label Class 2 / Negative
True label Class 1 / Positive	Correct Predictions (True positive for Class 1 / True Positives)	Incorrect Predictions (False positive for Class 2 / False Negatives)
True label Class 2 / Negative	Incorrect Predictions (False positive for Class 1 / False Positives)	Correct Predictions (True positive for Class 2 / True Negatives)

$$\text{Accuracy} = \frac{\text{TP for class 1} + \text{TP for class 2}}{\text{TP for class 1} + \text{FP for class 1} + \text{TP for class 2} + \text{FP for class 2}}$$

$$\text{Class 1 Precision} = \frac{\text{TP for class 1}}{\text{TP for class 1} + \text{FP for class 1}}$$

$$\text{Class 2 Precision} = \frac{\text{TP for class 2}}{\text{TP for class 2} + \text{FP for class 2}}$$

$$\text{Class 1 Recall} = \frac{\text{TP for class 1}}{\text{TP for class 1} + \text{FP for class 2}}$$

$$\text{Class 2 Recall} = \frac{\text{TP for class 2}}{\text{TP for class 2} + \text{FP for class 1}}$$

Numerical Example

	Predicted Labels: Class 1	Class 2	Class 3
True labels Class 1	16	0	1
Class 2	0	17	1
Class 3	0	0	11

Accuracy = 16 +17 + 11 / 16 + 17 + 11 + 1+ 1 = 44/46 = 0.956

$$\text{Precision for Class 1} = \frac{\text{TP for class 1}}{\text{TP for class 1+ FP for class 1}} = 16/16+0+0 = 1$$

$$\text{Precision for Class 2} = \frac{\text{TP for class 2}}{\text{TP for class 2+ FP for class 2}} = 17/17+0+0 = 1$$

$$\text{Precision for Class 3} = \frac{\text{TP for class 3}}{\text{TP for class 3+ FP for class 3}} \quad 11/11+1+1 = 0.84$$

$$\text{Recall for Class 1} = \frac{\text{TP for class 1}}{\text{TP for class 1+ FP for class 2+FP for class 3}} = 16/16+0+1 = 16/17 = 0.94$$

$$\text{Recall for Class 2} = \frac{\text{TP for class 2}}{\text{TP for class 2+ FP for class 1+ FP for class3}} = 17/18 = 0.94$$

$$\text{Reclass for Class 3} = \frac{\text{TP for class 3}}{\text{TP for class 3+ FP for class 1+ FP for class2}} = 11/11 = 1$$

Performance Metrics (cont…)

- **The accuracy** of the model is basically the total number of correct predictions divided by total number of predictions.
- **The precision** of a class define how trustable is the result when the model answer that a point belongs to that class.
- **The recall** of a class expresses how well the model is able to detect that class.
- **The F1 score** of a class is given by the harmonic mean of precision and recall (2×precision×recall / (precision + recall)), it combines precision and recall of a class in one metric.

Performance Metrics (cont…)

For a given class, the different combinations of recall and precision have the following meanings :

- high recall + high precision : the class is perfectly handled by the model
- low recall + high precision : the model can’t detect the class well but is highly trustable when it does
- high recall + low precision : the class is well detected but the model also include points of other classes in it
- low recall + low precision : the class is poorly handled by the model

Area Under Curve - AUC

ROC Plot: Receiver Operating Characteristic Plot

The ROC curve is a graphical representation of the trade-off between the True Positive Rate (TPR) and the False Positive Rate (FPR) at various threshold settings.

AUC (Area Under the Curve) is a performance measurement for classification models, particularly in binary classification problems.

The AUC provides a single scalar value to evaluate the model's performance across all threshold values.

The AUC value ranges from 0 to 1. AUC = 1 A perfect model, which perfectly distinguishes between positive and negative classes.

AUC = 0.5: A model that is no better than random guessing.

AUC < 0.5: A model that performs worse than random guessing.