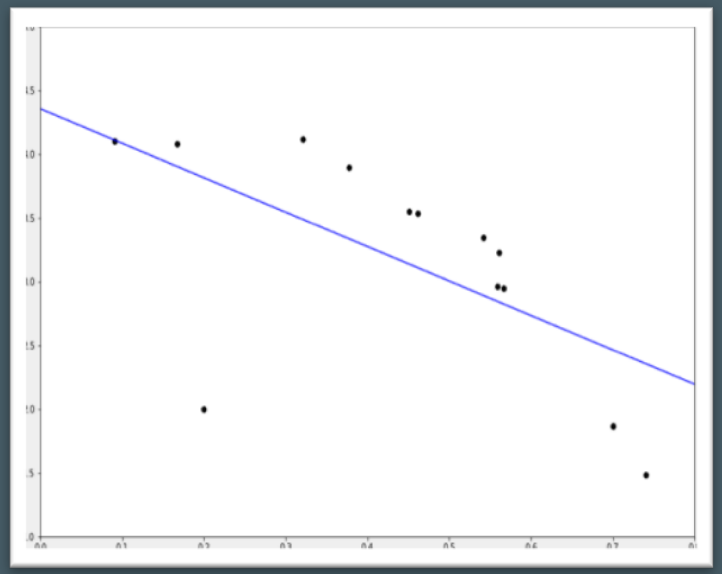
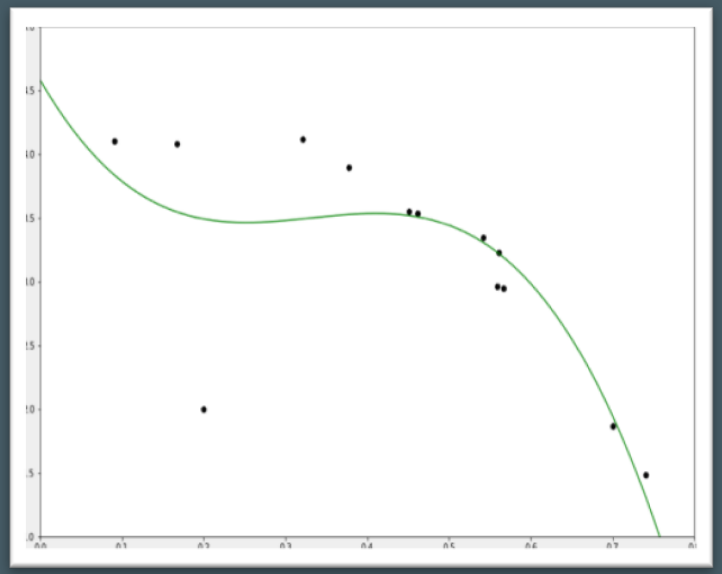
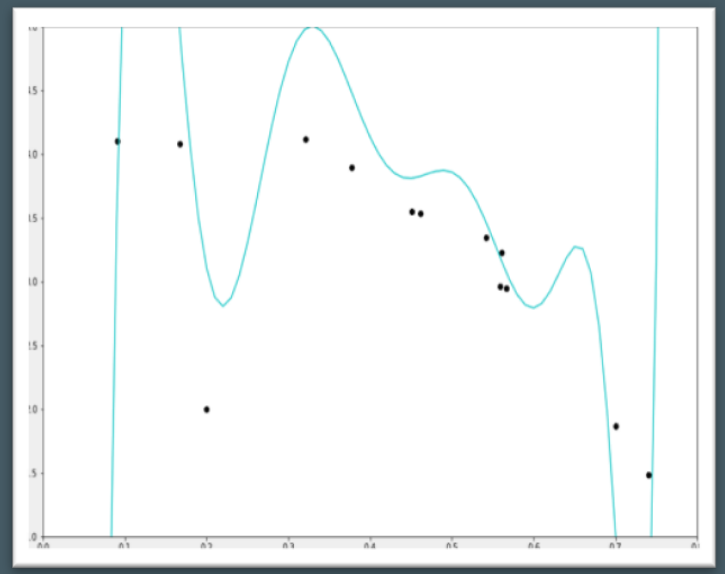
Validation

One of the main objective of cross validation is to detect overfitting and underfitting.







Overfitting occurs when the classifier/regressor, models the noise and outliers of the dataset instead of the actual relationship between the input X and the output y. This means that the classifier/regressor is **over**stating the influence its features.

A regressor is said to be **underfit** to the data, if the influence of the features to the output variable is too mild. This results to a model which describes a too simple relationship.

Underfit

- assumes generalizations
- unsusceptible to noise, outliers and complexity
- models are too simple to describe relationships
- an underfit model is said to have high bias

Overfit

- does not generalize at all
- oversensitve to noise, outliers and sensitivity
- models are too complex to create generaliztions
- an overfit model is said to have high variance

An underfit model is like a person that says:

“all muslims are terrorists”,

assuming the oversimple relationship:
(if muslim, then terrorist)
when in fact human nature is more complex than that.

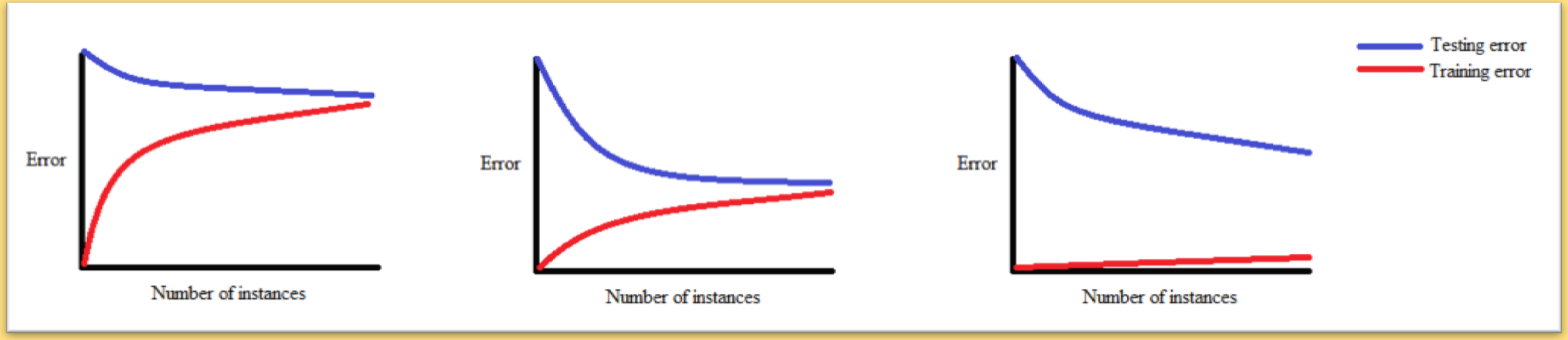
an overfit model is useless as well,
its like a person that says:

"terrorist are Bin Laden, Omar Maute, Dylan Klebold"

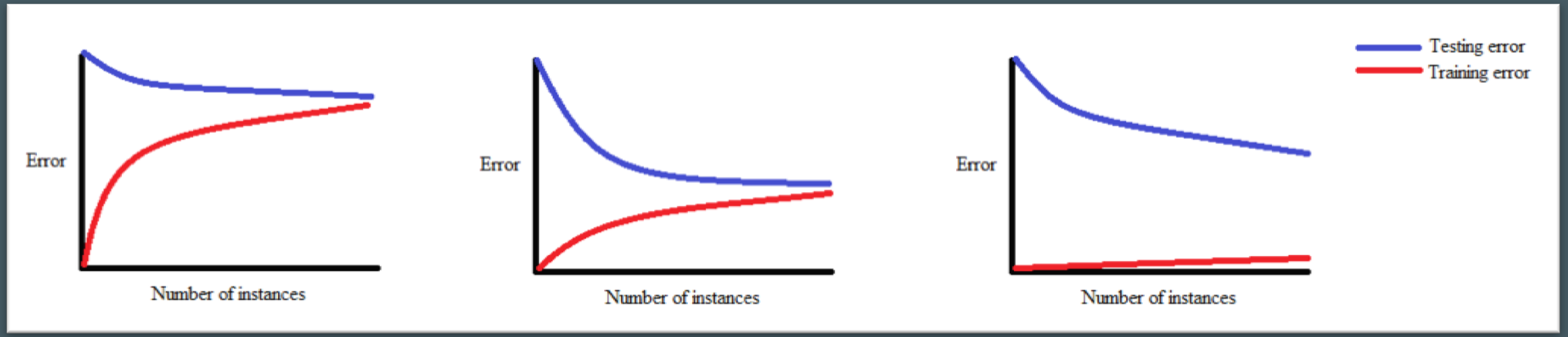
It is a useless model since it just memorizes, associations based on the training data it cannot be used to identify potential terrorists

BIAS and VARIANCE always exists with tradeoff. Controlling the bias will increase variance and controlling the variance will increase the bias

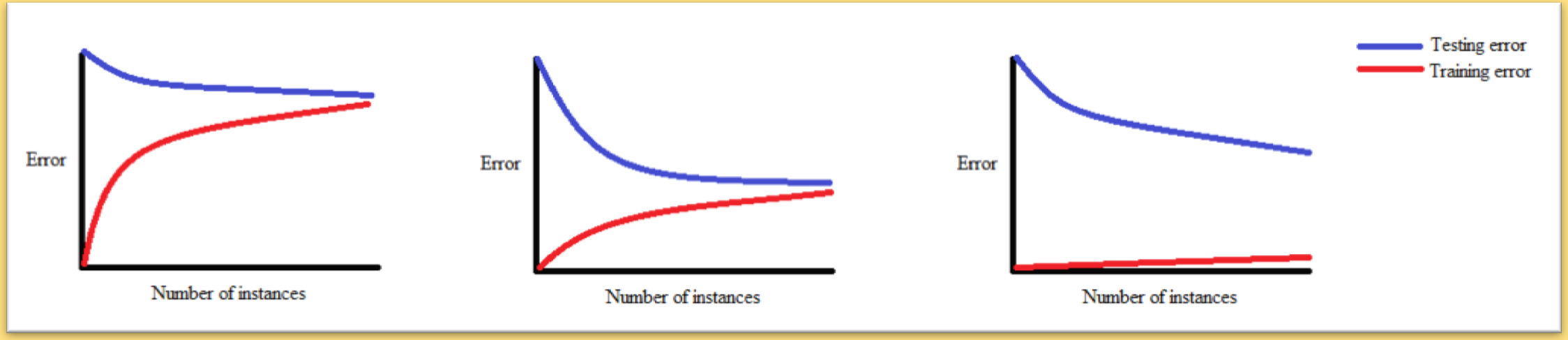
One way to detect overfitting in your model is by splitting the dataset into **train** and **test** partitions. The model is trained using the train partition but is tested using the test partition.



It is natural for the model to score well in the train partition since the model will tend to overfit the dataset to some degree.



But if the dataset scores extremely well in the training set but scores poorly in the test set, this means that the model is overfit to the training data



If both the train score and test score have poor scores, this means that the model is underfit to the dataset.

Size of Partitions

Amount of Data in Training Set

The objective to reduce overfitting in data is to provide enough data in the training set so that the model is able to ignore the noise and outliers in the data. If there is too little data in the training set, the model will not be able to distinguish between real data and noise.

Amount of Data in the Test Set

There should also be enough examples in the test set so that your tests produce reliable results. If there is too little data in the test set, the test set may be an inaccurate sample representation of the population. Thus the scores derived from said test may be an inaccurate representation as well.

K-Fold Cross Validation

K-Fold cross validation is a generalized train-test validation. It creates k equal-sized partitions from your dataset. One partition is used for training the model and $k-1$ partitions is used for testing. Each partition is tested exactly once. The overall score for the model is calculated as the average score for $k-1$ test partitions.

Evaluating Classification Models

The Problem with Accuracy

Accuracy is a common measure, calculated as:

$$A = \frac{T_p + T_n}{T_p + T_n + X_p + X_n}$$

	Tested as Positive, V_p	Tested as Negative, V_n
Actually Positive, U_p	Accurate Prediction (True Positive, T_p)	Type II Error (False Negative/Miss, X_n)
Actually Negative, U_n	Type I Error (False Positive/False Alarm, X_p)	Accurate Prediction (True Negatives, T_n)

	$V_p = 0$	$V_n = 1000$
$U_p = 10$	$T_p = 0$	$X_n = 10$
$U_n = 990$	$X_p = 0$	$T_n = 990$

The accuracy of the model described in the previous slide is 0.99:

$$A = \frac{990 + 0}{1000}$$

But it is obvious that this model is terrible. It predicts negative for any given example.

To avoid this problem, instead of accuracy, the measure F_1 score is generally used

$$F_1 = 2\frac{PR}{P+R}$$

where P is the measure of precision, calculated as, $P = \frac{T_p}{T_p+F_p}$, and R is the measure of recall (or sensitivity), calculated as: $R = \frac{T_p}{T_p+F_n}$

Using F_1 score on the same example, the model will score 0 since both the precision and recall are zero.

False Alarm vs Miss

Sometimes a model may be built to value false positives and false negatives differently. For example, consider, a model which is used on a biometrics scanner which predicts positive if the thumb placed in the scanner belongs to a registered user or negative if otherwise.

False Alarm vs Miss

This model is should be more averse to false positives than false negatives. This is because false positives on this model is more dangerous than false negatives. False positives will allow unregistered users while false negatives will just prompt the registered user to use the scanner again.

False Alarm vs Miss

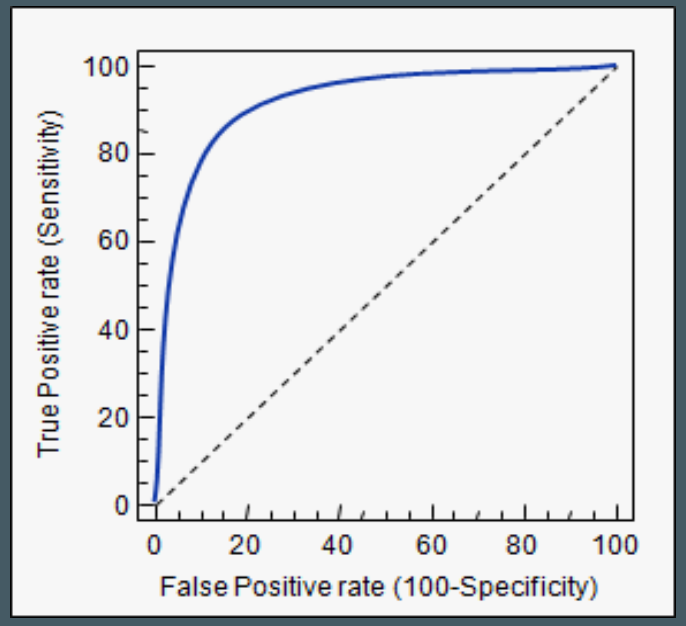
This behavior can be enforced on a model artificially. A logistic regression model's cost function can be changed to:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)}\log(h_\theta(x^{(i)})) + 2(1 - y^{(i)})\log(1 - h_\theta(x^{(i)}))$$

This cost function will penalize false positives twice more than false negatives.

False Alarm vs Miss

Another way to value different types of errors differently is by using ROC (Receiver operating characteristic) analysis and choosing the appropriate model based on the analysis.



Evaluating Regression Models

Evaluating regression models is similar to the mean squared error cost of a linear regression model. These are two examples of regression scoring

- **Explained Variance.** Regression models can be scored using the metric, explained variance, this value is the measure of variation of predictions from actual values.
- **Coefficient of determination.** R^2, or coefficient of determination is calculated as the proportion of variance in the dependent varaible that is predictable from the independent variable.