



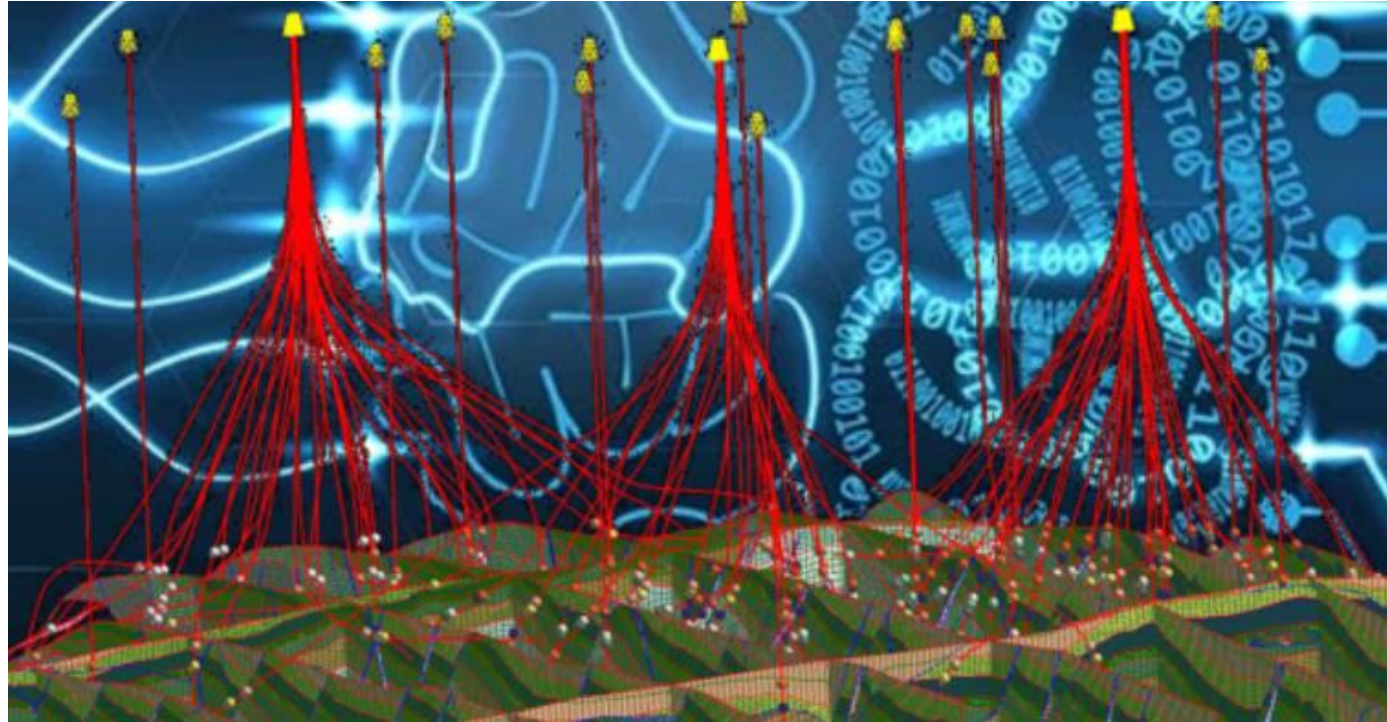
Covenant University

Raising a new Generation of Leaders

PDA_SIG Learn

HOW WELL HAS A MACHINE LEARNT?

HOW WELL HAS A MACHINE LEARNT?



Olatunde O. Mosobalaje (PhD)




Department of Petroleum Engineering,
Covenant University, Ota
Nigeria

OUTLINE





The Input-Output Relationship

-  The Conceptual Model: Unknown and Imperfect
-  The Prediction Model: Estimated and Imperfect
-  Error Reduction: Reducible versus Irreducible

Measures of Accuracy

-  Regression: Mean Squared-Error
-  Classification: Error Rate
-  Classification: Confusion Matrix




Issues in Model Performance

-  Model: Flexibility versus Simplicity
-  Model: Accuracy versus Interpretability
-  Model: Under-fitting versus Over-fitting
-  Bias Variance Trade-off

Cross-Validation

-  Leave-One-Out Cross-Validation
-  K-fold Cross-Validation

Improving Model Performance

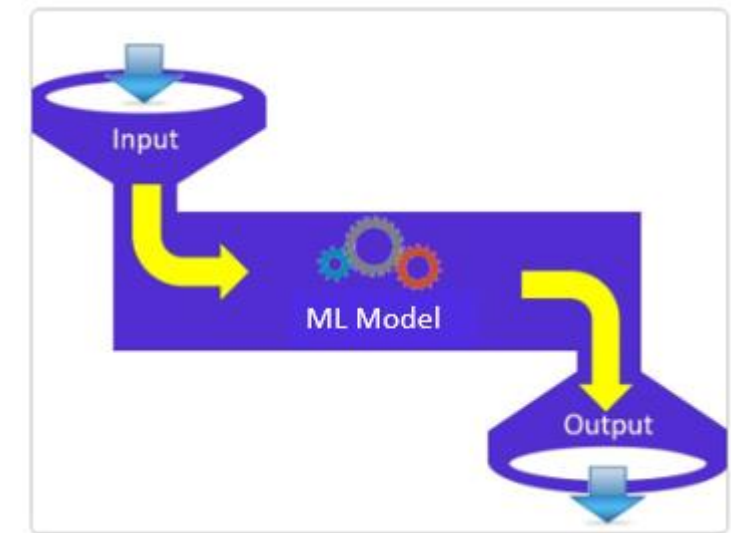
-  Hyper-parameter Tuning
-  Dimensionality-Reduction
-  Feature Selection



The Input-Output Relationship

The Conceptual Model: Unknown and Imperfect

- Recall: a machine learning model has been described as a data-driven **model** representation of a physical system/phenomenon.
- Such physical systems/phenomena typically involve input and output variables
 - A relationship between the input(s) and the output variables underlies the system/phenomenon
 - Example: Recovery Factor, RF (output) of a water-flooded reservoir system is related to Dykstra-Parson Permeability Variation Coefficient, Mobility Ratio, Permeability Anisotropy, Production Water-cut and Wettability (inputs)
- A machine learning (supervised) model is built to capture this input-output relationship, for prediction purposes.



The Input-Output Relationship

The Conceptual Model: Unknown and Imperfect

█ Ideally, this input-output relationship can be expressed thus:

$$o = f(i) \text{ ----- } (1)$$

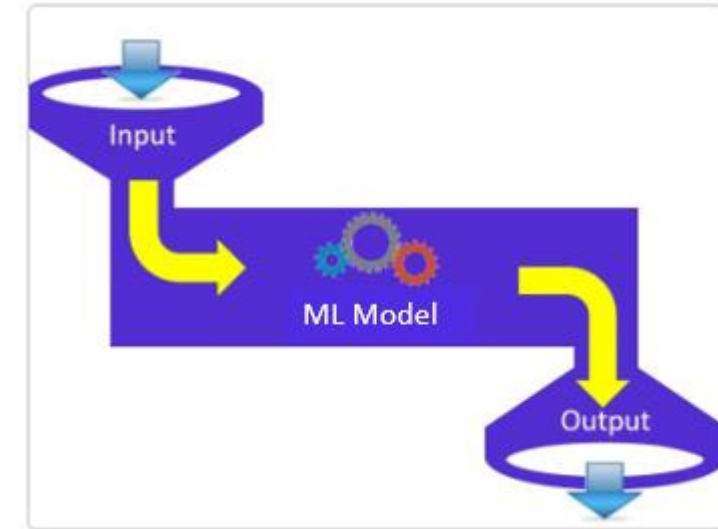
█ where:

█ f is some function of the input variable(s)

█ o is an observed/measured output value corresponding to a given input value i

█ First, in reality, this function f is NOT known – so, it is only a **Conceptual** (mental) **Model**.

█ Even if known, the function f would still not give a perfect prediction of output value that would exactly match the measured output value o . Why?



The Input-Output Relationship

The Conceptual Model: Unknown and Imperfect

Reason for the imperfection of the function f , (even if known):

It is an uncertain world, the reservoir system not excluded!

For example, our inability to understand and deal with the complexity of the geologic phenomenon introduces uncertainty into reservoir characterization.

Measurement error – the process of observing and measuring the output is NOT free of error.

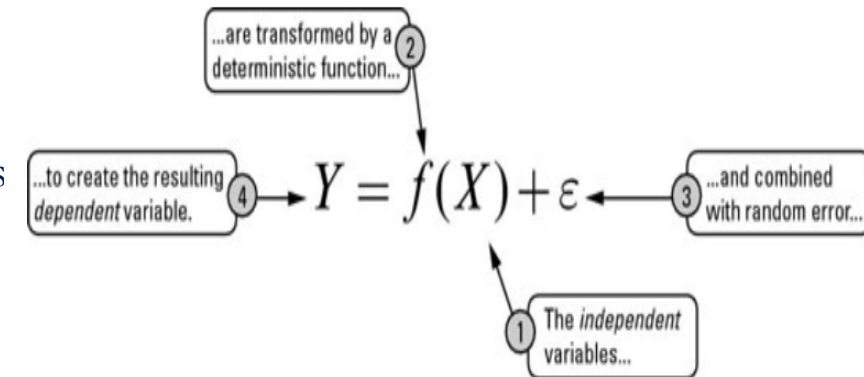
Noisy data

Unmeasured/unmeasurable input variables that also affects the output variable o

So, in reality, the **conceptual model** is better expressed as:

$$o = f(i) + \epsilon \text{ ----- } (2a)$$

where ϵ is the **random error** term added to account for the imperfection of the function f .



The Input-Output Relationship

The Prediction Model: Estimated and Imperfect

- If output must be predicted, there is a need to **estimate** the unknown function f in the conceptual model, using supervised machine learning algorithms.
- With the availability of data (a set of $[i, o]$ values), machine learning algorithms are deployed to train a machine learning model, for prediction purposes.
 - Such models are referred to as **Prediction Model**, and are generally expressed thus:

$$\hat{o} = \hat{f}(i) \text{ ----- } (3)$$

■ where:

■ \hat{o} is the **predicted** output

■ \hat{f} is the **estimated function** – i.e. an estimate of f

- Whereas f is the truth but unknown, \hat{f} is the estimate and known
- Another way to express the dichotomy between f and \hat{f} is to refer to f as the **Expected** function and \hat{f} as the **Fitted** function.



- The so-called Function \hat{f} does not have to be an explicit mathematical equation; it could be a decision structure

The Input-Output Relationship

The Prediction Model: Estimated and Imperfect

 Also note the following nomenclature as per the output:

 o --- Measured/Observed output

 \hat{o} --- Predicted output – obtained using the estimated function, $\hat{f}(i)$

 \bar{o} --- Expected output – obtainable if function $f(i)$ were to be known

 Taken as the long-run average (expected value) of several \hat{o} values for a given input value i

 Different from average of o values measured for different input values

 Hence, Equation 2a could be written as:

$$o = \bar{o} + \epsilon \text{ ----- } (2b)$$

 Or, equivalently as:

$$\epsilon = o - \bar{o} \text{ ----- } (2c)$$

The Input-Output Relationship

The Prediction Model: Estimated and Imperfect

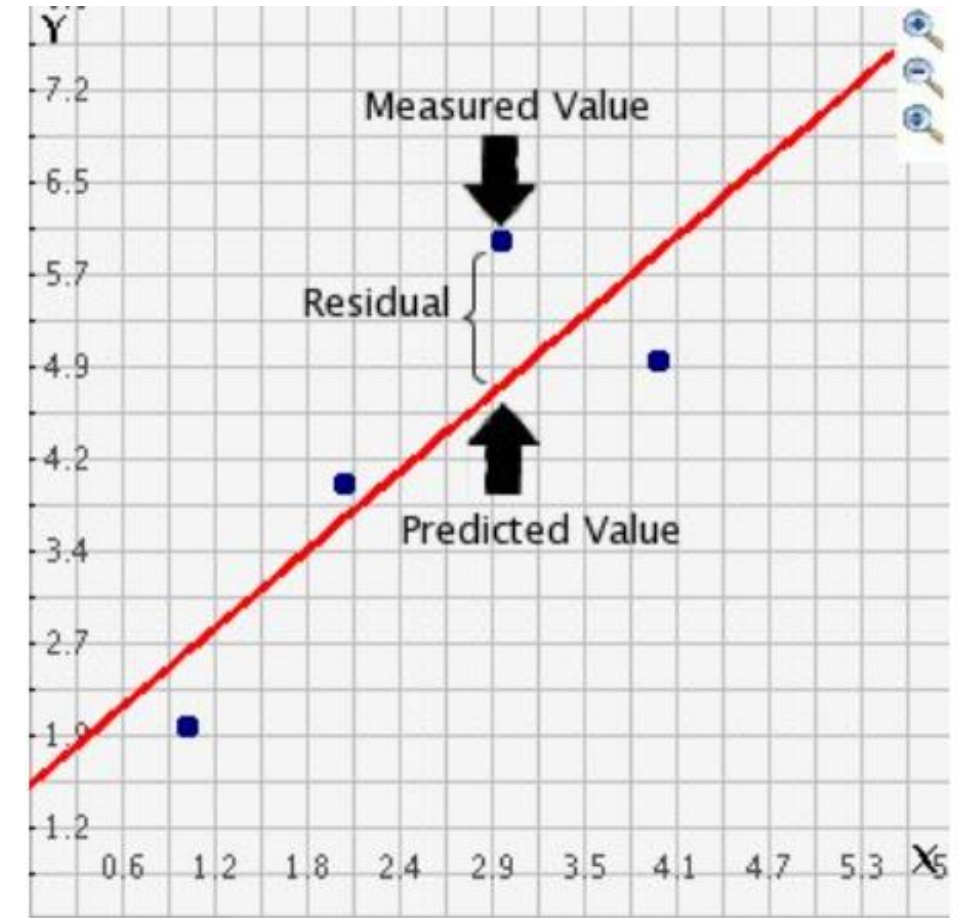
- Expectedly, the prediction output values would not exactly match the measured output values in the data.
- The mathematical difference between the observed output o and the predicted output \hat{o} is known as the **Residual** of the prediction model, e .

$$e = o - \hat{o} \text{ ----- (4a)}$$

Equivalently,

$$o = \hat{o} + e \text{ ----- (4b)}$$

$$o = \hat{f}(i) + e \text{ ----- (4c)}$$



The Input-Output Relationship

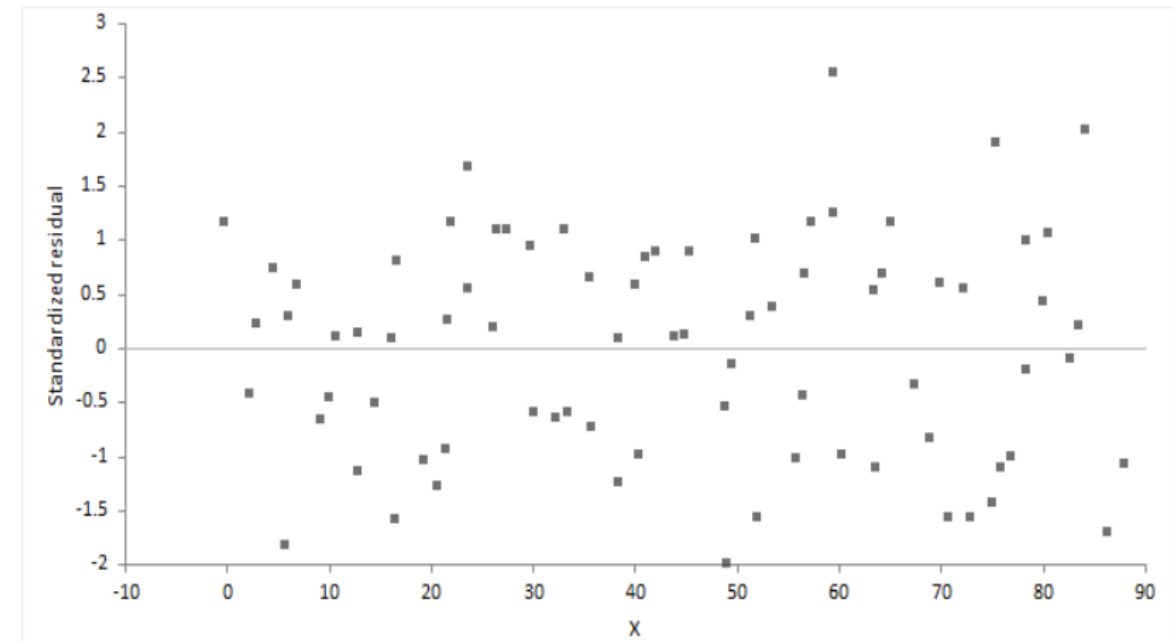
The Prediction Model: Estimated and Imperfect

In order to be acceptable, the values of the residuals must be:

- Independent of the predictor's values
- Have a mean of zero
- Constant variance

Random Error (ϵ) versus Residuals, (e)

- ϵ is conceptual (theoretical) and unobtainable
- e is real and obtainable
- The obtainable e is somewhat an estimate of the unobtainable ϵ .
 - Since the fitted line (that produces e) is an estimate of the expected line (that produces ϵ)



The Input-Output Relationship

Error Reduction: Reducible versus Irreducible

Under the hood, machine learning algorithms work by **reducing** the prediction residuals to its **barest minimum**, subject to the quantity and quality of the training data available.

This barest minimum is NOT zero!

Because, the residual contains the random error plus some extra, as shown thus:

From Equation 2c:

$$\epsilon = o - \bar{o} \text{ ----- } \text{---(2c)}$$

From Equation 4a:

$$e = o - \hat{o} \text{ ----- } \text{---(4a)}$$

The difference on the RHS of Equation 4a could be expanded thus:

$$e = (o - \bar{o}) + (\bar{o} - \hat{o}) \text{ ----- } \text{---(5)}$$

Recognizing the first term of the RHS of Equation 5 as being same as LHS of Equation 2c,

$$e = \epsilon + (\bar{o} - \hat{o}) \text{ ----- } \text{---(6)}$$

The Input-Output Relationship

Error Reduction: Reducible versus Irreducible

$$e = \epsilon + (\bar{o} - \hat{o}) \text{-----} \text{---}(6)$$

Now, a machine learning algorithm minimizes the residual e .

That minimization is achieved by ‘fitting’ an estimated function \hat{f} (that yields \hat{o}) as close as possible to the expected function f (that should yield \bar{o})

That is, minimizing $\bar{o} - \hat{o}$, the second term of Equation 6.

Hence, $\bar{o} - \hat{o}$ is the **reducible** part of the residual.

Even if (in rare cases) the algorithm did a ‘perfect’ job of fitting ‘fitting’ an estimated function \hat{f} that is exactly equal to the expected function f , the residual would still NOT be zero!

Because the random error (the first term of Equation 6) is **irreducible**!

Uncertainty: complexity and insufficient understanding of the physical system, noise, random variations

Measurement error

Unmeasured/unmeasurable factors

Measures of Accuracy

- None of the machine learning algorithms is a 'one-size-fits-all'.
 - That is, no one method is best across all datasets for all problems.
- Hence, in choosing the right algorithm for a given problem, various algorithms are used and evaluated for accuracy.
- The measures of model accuracy for a regression problem is different from that of a classification problem.

Measures of Accuracy

Regression: Mean-Squared Error

Recall the prediction residual.

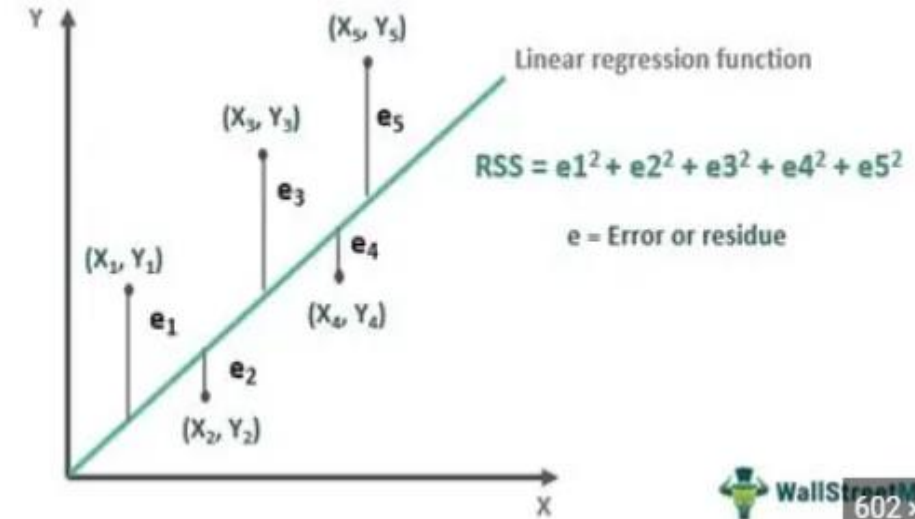
The prediction residual (simply known as **error**) is the most basic measure of accuracy (inaccuracy, actually).

The more the **residual/error**, the less accurate is the prediction model.

However, a **residual/error** value only corresponds to the prediction of just one output of a given input value.

Typically, the data (training and testing) for a machine learning project contains several values (or vectors of values) of the input variable(s) and their corresponding output values.

Hence, there would be several values of **residuals/errors**



Measures of Accuracy

Regression: Mean-Squared Error

With multiple values of **residual/error**, some sort of aggregation is needed to obtain a single measure of accuracy.

Summation and Averaging

Summing the raw **residual/error** values could be misleading:

Positive errors get cancelled by negative errors; thereby, accuracy is over-estimated (inaccuracy under-estimated)

Alternatives to raw error values:

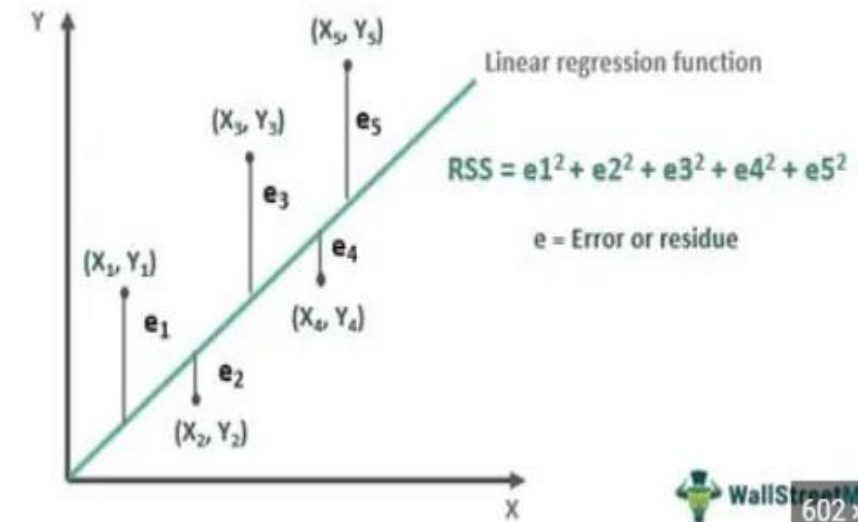
Absolute error values, or

Squared error values

Mathematically, the Squared Error is a better option.

Hence, **Sum of Squared Error (SSE)** is a very popular measure of model accuracy

Also known as **Residual Sum of Squares (RSS)**



Measures of Accuracy

Regression: Mean-Squared Error

For a dataset containing n sample points, the Sum of Squared Error (SSE) is thus:

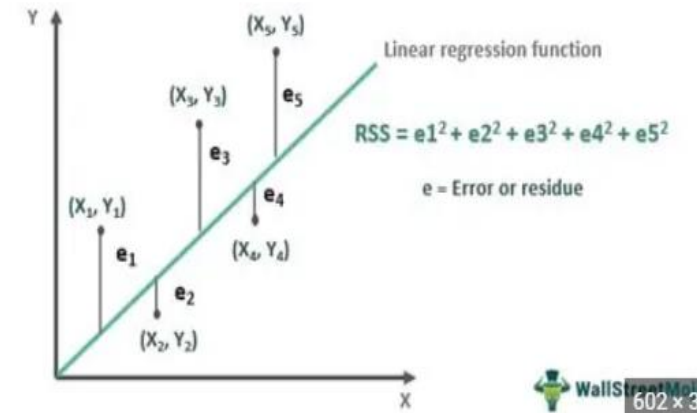
$$SSE = e_1^2 + e_2^2 + \dots + e_{j-1}^2 + e_j^2$$

$$SSE = \sum_{j=1}^j e_j^2 \text{ --- (7a)}$$

Or,


$$SSE = (o_1 - \hat{o}_1)^2 + (o_2 - \hat{o}_2)^2 + \dots + (o_{j-1} - \hat{o}_{j-1})^2 + (o_j - \hat{o}_j)^2$$

$$SSE = \sum_{j=1}^j (o_j - \hat{o}_j)^2 \text{ --- (7b)}$$



Measures of Accuracy

Regression: Mean-Squared Error

 Further still, the Squared Error could be averaged, giving rise to the **Mean-Squared Error (MSE)** - a more popular measure of accuracy.

 For a dataset (training or test) containing n sample points, the Mean-Squared Error (MSE) is thus:

$$MSE = \frac{e_1^2 + e_2^2 + \dots + e_{j-1}^2 + e_j^2}{n}$$

$$MSE = \frac{1}{n} \sum_{j=1}^j e_j^2 \text{ --- (8a)}$$

Or,


$$MSE = \frac{(o_1 - \hat{o}_1)^2 + (o_2 - \hat{o}_2)^2 + \dots + (o_{j-1} - \hat{o}_{j-1})^2 + (o_j - \hat{o}_j)^2}{n}$$

$$MSE = \frac{1}{n} \sum_{j=1}^j (o_j - \hat{o}_j)^2 \text{ --- (8b)}$$

Measures of Accuracy

Regression: Mean-Squared Error

Training MSE versus Test MSE

 When a **test dataset** is available, such should be used in evaluating the model that has been trained with the **training dataset**.

 The MSE obtained from such evaluation (with test dataset) is known as **Test MSE**

 The lower the Test MSE, the more accurate the model.

 The algorithm with the lowest Test MSE is chosen for the job

 However, sometimes, a test dataset set is not available for evaluation.

 Should the training dataset be used???

Measures of Accuracy

Regression: Mean-Squared Error

Training MSE versus Test MSE

 The training dataset may NOT be used for evaluation!

 A fundamental issue (to be discussed) in machine learning makes this approach prohibitive.

 When the same dataset used in training a model is still used to evaluate the accuracy of the model, the MSE thus obtained is known as **Training MSE**.

 The expectation that the model that gives the minimum Training MSE also gives the minimum Test MSE is NOT always true.

 Naturally, the Training MSE will always be low – since algorithms work by minimizing residuals of the training dataset.

 Curiously, the model with the lowest Training MSE may give a high Test MSE.

 Again, the issues underlying this unexpected behavior is to be discussed soon.

 In the event there is no test dataset, what is the way out?

 **Cross-validation** is it!

 To be discussed.

Measures of Accuracy

Classification: Error Rate

 In classification machine learning setting, the **Error Rate** is a common measure of accuracy.

 The **error rate** is simply the proportion of incorrect classification made by the model, during evaluation.

 The lower the error rate, the more accurate the model is.

 The algorithm with the lowest error rate is chosen for the job

 For a dataset (training or test) containing n sample points, the Error Rate E_r , is thus:


$$E_r = \frac{1}{n} \sum_{j=1}^j I(o \neq \hat{o}) \text{ --- (9)}$$


Where $I(o \neq \hat{o})$ is a binary variable that is equal to 1 if the sample is wrongly classified (i.e., $o \neq \hat{o}$) and is equal to zero if the sample is correctly classified (i.e., $o = \hat{o}$).


Measures of Accuracy

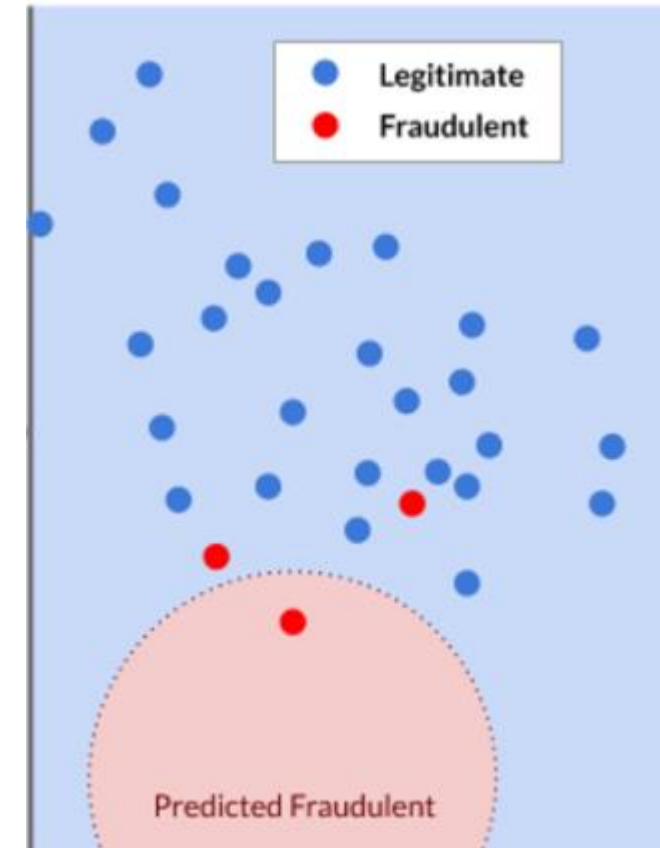
Classification: Confusion Matrix

 For an imbalanced data of a binary classification, the error rate might be a misleading measure of accuracy.

 Imbalanced data: most of the sample data (training or test) belongs to one category (out of two)

 A low error rate (high accuracy) might be because majority of the samples from the dominant category are correctly classified while a significant proportion of the samples from the non-dominant category is wrongly classified.

 Yet, the non-dominant category might be the objective of the project; e.g., classifying manufactured objects into defective (non-dominant category) and non-defective (dominant category).



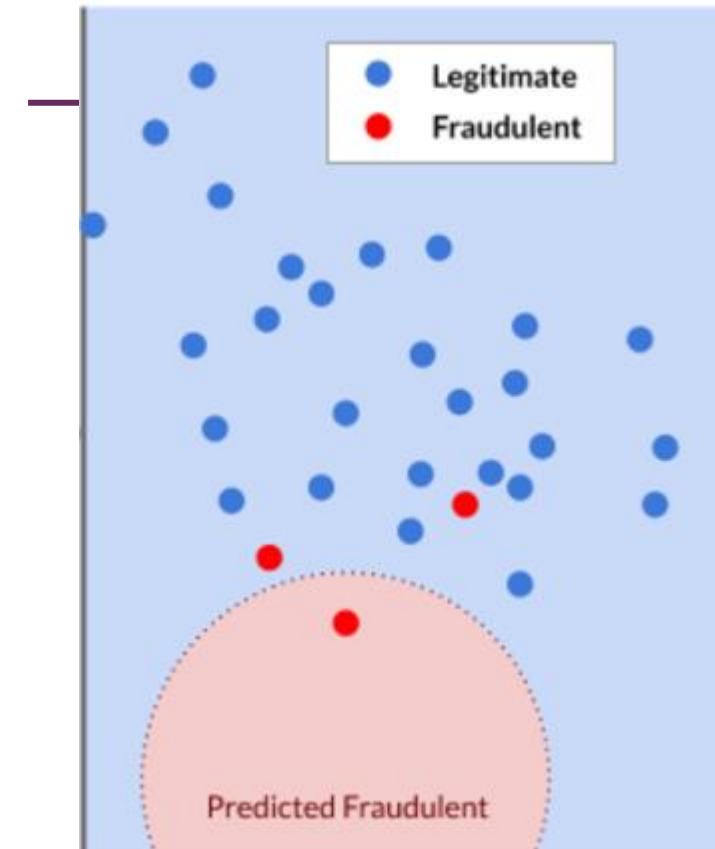
Measures of Accuracy

Classification: Confusion Matrix

In such imbalanced data cases, the **Confusion Matrix** is a better measure of accuracy.

The confusion matrix gives more details of the accuracy by showcasing the comparison between the actual and predicted output, in four terms.


		Actual values	
		Fraudulent	Not Fraudulent
Predicted	Fraudulent	1 true positives	0 false positives
	Not Fraudulent	2 false negatives	27 true negatives




Measures of Accuracy

Classification: Confusion Matrix


True Positives:

 Number of samples that actually belong to the 'positive' class and truly predicted to belong to the 'positive' class.


True Negatives:

 Number of samples that actually belong to the 'negative' class and truly predicted to belong to the 'negative' class.

False Positives:

 Number of samples that actually belong to the 'negative' class but falsely predicted to belong to the 'positive' class.

False Negatives:

 Number of samples that actually belong to the 'positive' class but falsely predicted to belong to the 'negative' class.

		Actual values	
		Fraudulent	Not Fraudulent
Predicted	Fraudulent	1 true positives	0 false positives
	Not Fraudulent	2 false negatives	27 true negatives

Measures of Accuracy

Classification: Confusion Matrix

From the quadrants of the Confusion Matrix, two measures of accuracy are defined thus:

Sensitivity:

The proportion of correct classification in the 'positive' class.

$$\text{Sensitivity} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \text{ --- (11)}$$

Specificity:

The proportion of correct classification in the 'negative' class.

$$\text{Sensitivity} = \frac{\text{True Negatives}}{\text{True Negatives} + \text{False Positives}} \text{ --- (12)}$$


Choosing between sensitivity and specificity depends on the objective at hand. If you can't afford to misclassify a 'positive' class sample, but its not a big deal if a 'negative' class got misclassified; then, you evaluate with Sensitivity

		Actual values	
		Fraudulent	Not Fraudulent
Predicted	Fraudulent	1 true positives	0 false positives
	Not Fraudulent	2 false negatives	27 true negatives


Issues in Model Performance


Model: Flexibility versus Simplicity

 Sometimes, the choice of which ML algorithm to use is a choice between **flexibility** and **simplicity**.

 A ML algorithm is said to be **flexible** if it can capture the input-output relationship by fitting several different potential forms of the estimated function \hat{f} , as the complexity of the relationship demands.

 That is, the algorithm can handle complex phenomena and complex dataset

 It could fit something as complex as a polynomial or a thin-plate spline, to capture complex variations in the data

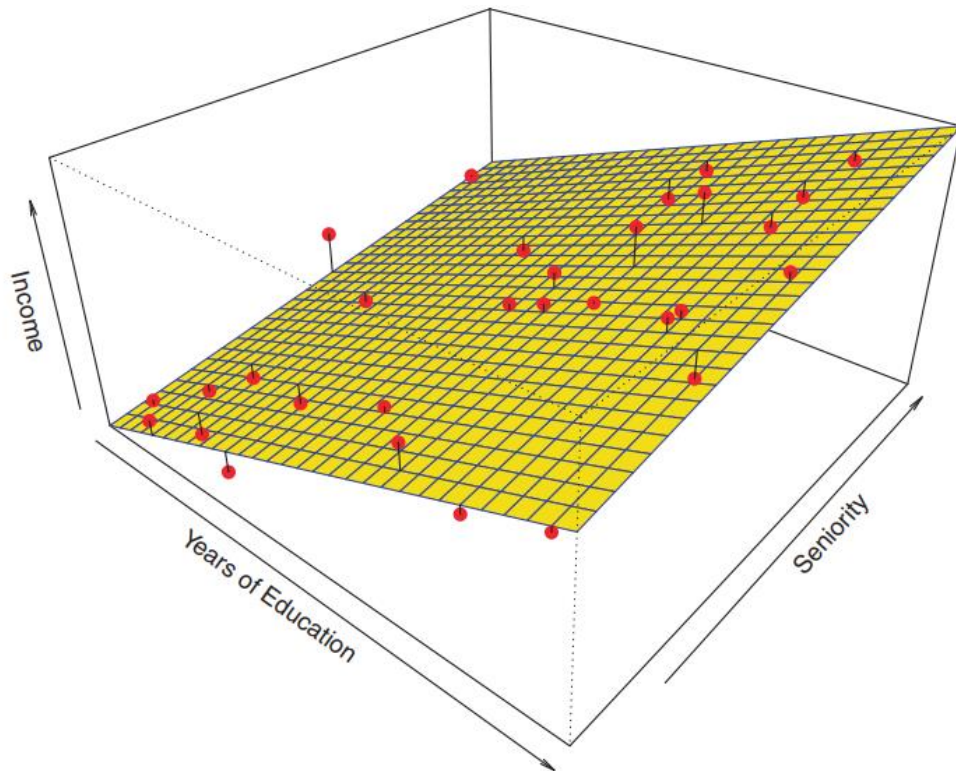
 A ML algorithm is said to be **simple** if it is limited in the forms of functions it could fit.

 That is, when the algorithm handles a complex data/phenomenon, it approximates the input-output relationship with a simple function.

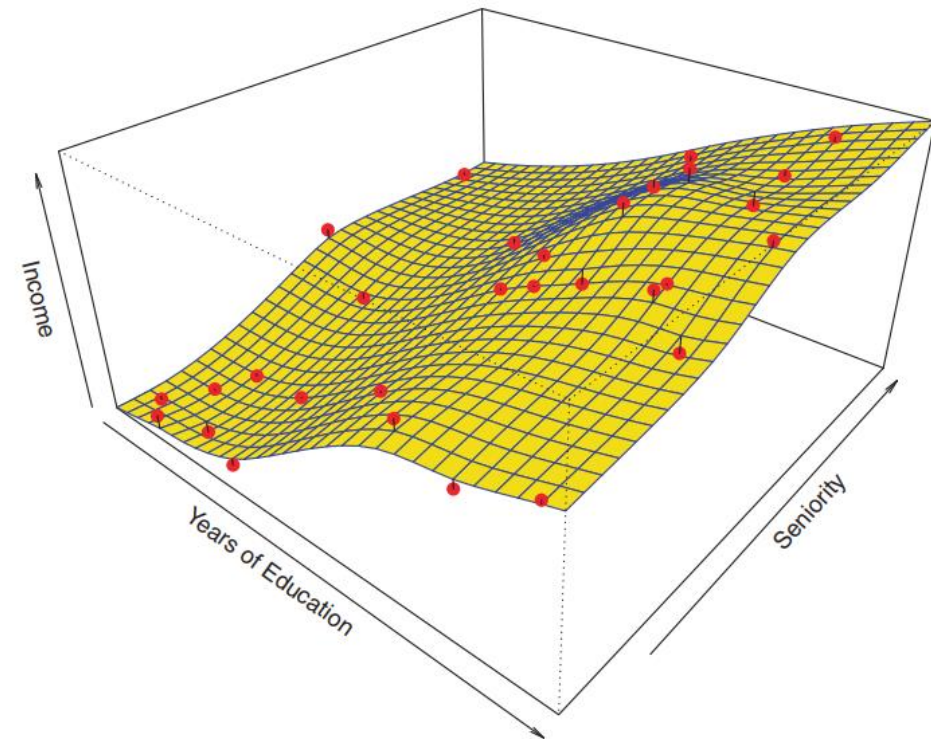
 Linear regression is a common example of this

Issues in Model Performance

Model: Flexibility versus Simplicity



A simple model: linear regression



A flexible model: thin-plate spline

Issues in Model Performance


Model: Accuracy versus Interpretability

 Obviously, a flexible model would give more prediction **accuracy**.

 But with less **interpretability**.

 And, of course, a simple model would be more **interpretable**.

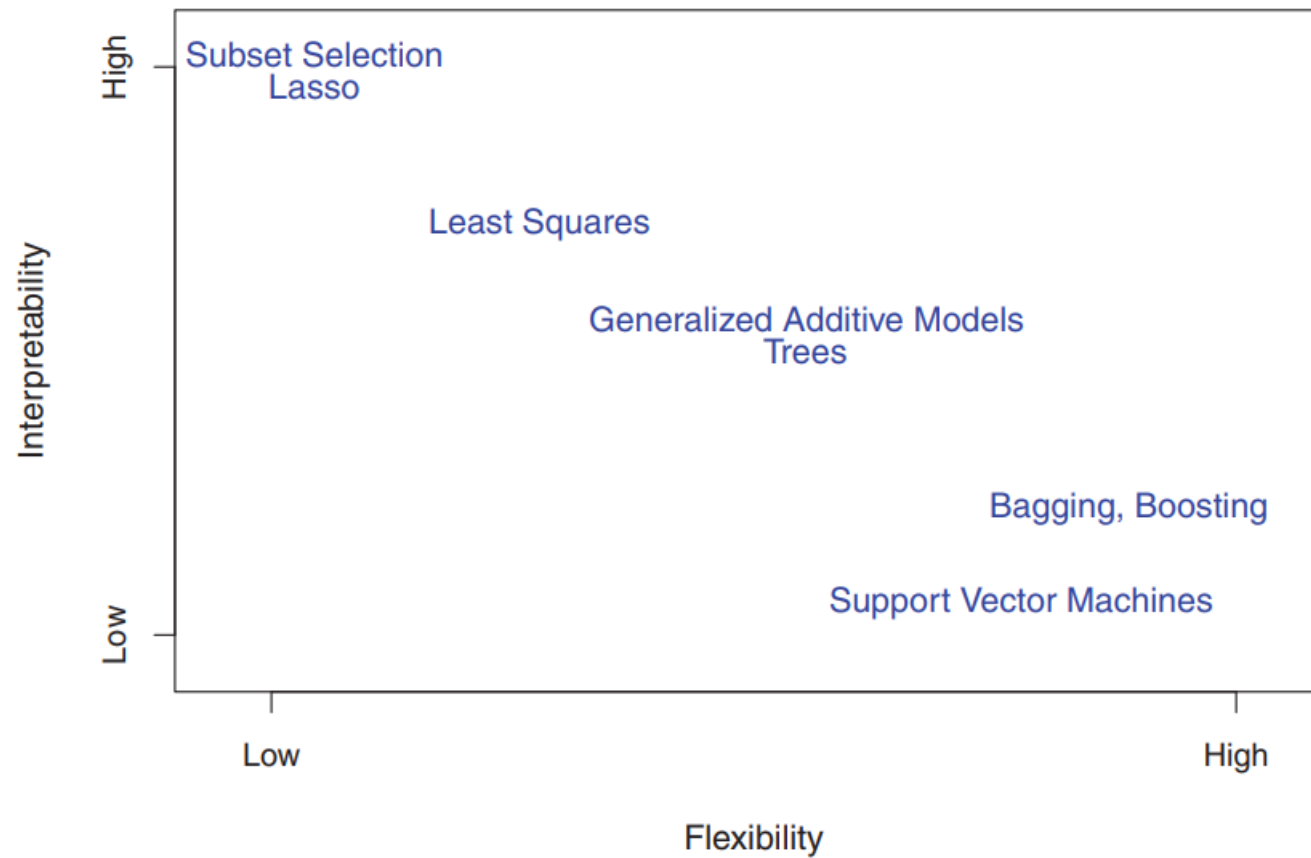
 But with less **accuracy**, if the true (expected) function is highly complex and non-linear.

 Will still have a high accuracy if the true (expected) function is linear or near-linear.

 So, choosing the right model is always a trade-off between accuracy and interpretability

Issues in Model Performance

Model: Accuracy versus Interpretability



Issues in Model Performance


Model: Accuracy versus Interpretability

 Obviously, a flexible model would give more prediction **accuracy**.

 But with less **interpretability**.

 And, of course, a simple model would be more **interpretable**.

 But with less **accuracy**, if the true (expected) function is highly complex and non-linear.

 Will still have a high accuracy if the true (expected) function is linear or near-linear.

 So, choosing the right model is always a trade-off between accuracy and interpretability.

Issues in Model Performance

Model: Over-fitting versus Under-fitting

Even if interpretability is NOT of interest, the choice between **flexible** (potentially complex) and **simple** models still has practical implications on prediction accuracy (particularly, for test and unseen dataset)

That is, it is **NOT ALWAYS** the more complex, the more accurate.

There is a point of diminishing return

Behind this diminishing return is the issue of **Over-fitting**.

For any data, there exist a Function f expected to capture the input-output relationship; but it is NOT known.

A machine learning project is an attempt to obtain Function \hat{f} as an estimate of the unknown function.

In such attempt, there are two possible extreme outcomes:

Choosing a very flexible model that result into an estimated Function \hat{f} being too complex than the true expected Function f - **Overfitting**

Choosing a rather simple model that result into an estimated Function \hat{f} that fails to capture the complexity of the true expected Function f - **Under-fitting**

Issues in Model Performance

Model: Over-fitting versus Under-fitting

Both over-fitting and under-fitting reduces the prediction accuracy

Over-fitting:

Increases prediction accuracy, if training data is used for prediction (but, who cares?!)

Training MSE is low

Reduces prediction accuracy, if test data (unseen during training) is used (everyone cares!)

Test MSE is high

Why???

The training MSE is low essentially because the model, being so complex, simply 'follows' all the data points (both **signal** and **noise**)

Signal: the desired meaningful information about the expected Function f

Noise: random unwanted fluctuation around the signal

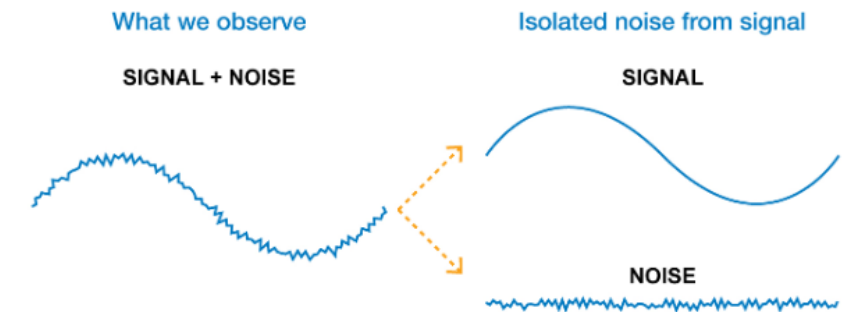



FIGURE 2. Isolated Noise from Signal


Issues in Model Performance


Model: Over-fitting versus Under-fitting


 Both over-fitting and under-fitting reduces the prediction accuracy

Over-fitting:

 The model simply ‘memorized’ the output values during training, and ‘recited’ them during evaluation.

 The test MSE is high because the model did a bad job predicting the output of unseen (not memorized) data.

 That is, a bad job **generalizing** the model to cases other than seen during the training.


 The noise that misled the model during training is NOT there in the test data. A different noise is there and the model cannot follow that, leading to significant difference in the model’s prediction and the measured output – high test MSE.


Issues in Model Performance

Model: Over-fitting versus Under-fitting

 Both over-fitting and under-fitting reduces the prediction accuracy

Under-fitting:

 To start with, training MSE is high because the model is restrictive and limited in capturing complexity.

 Did a bad job predicting the outcome values for data it has seen during training because it was restricted in following the trends (even the signals) in the data.

 Of course, test MSE is very high. Expectedly so.

 Since the model misses part of the signal during training.

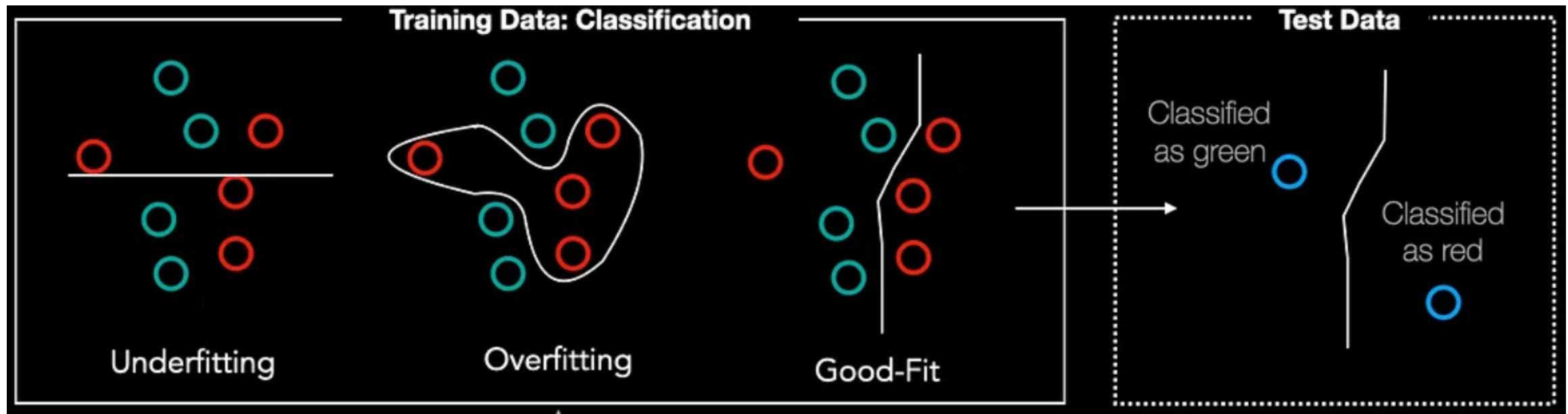
Issues in Model Performance

Model: Over-fitting versus Under-fitting



Issues in Model Performance

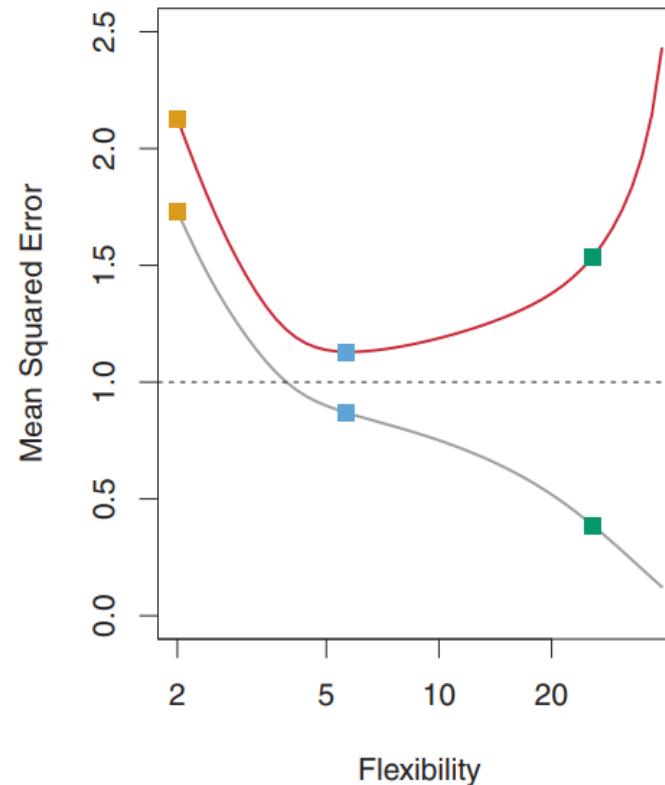
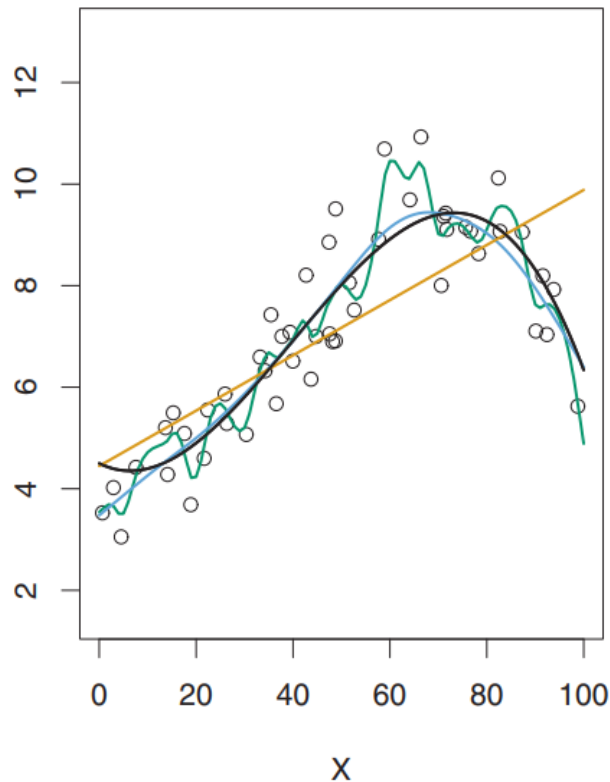
Model: Over-fitting versus Under-fitting



Issues in Model Performance

Model: Over-fitting versus Under-fitting


Synthetic Examples




 True (expected) function shown in black curve
– clearly non-linear

 Over-fitted function shown in green wiggly curve – follows signal+noise, most flexible

 Low training MSE, high test MSE

 Under-fitted function shown in yellow – a linear model, misses the non-linearity of the signal – least flexible

 High training MSE, high test MSE

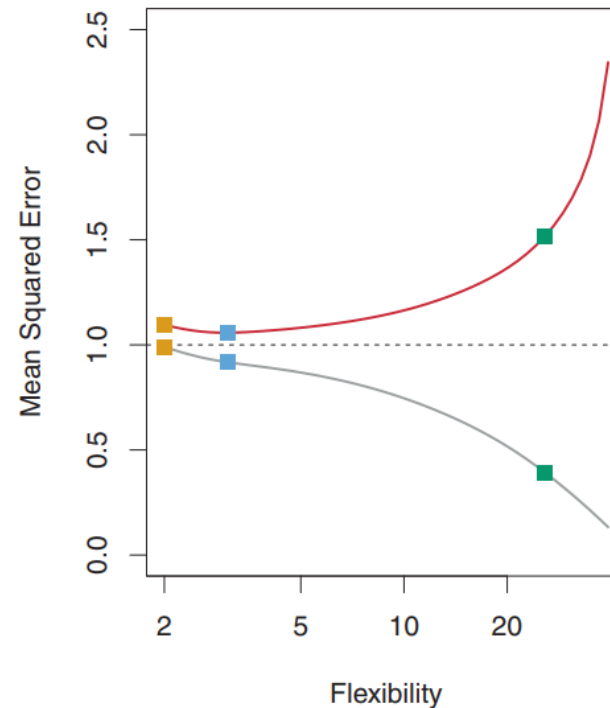
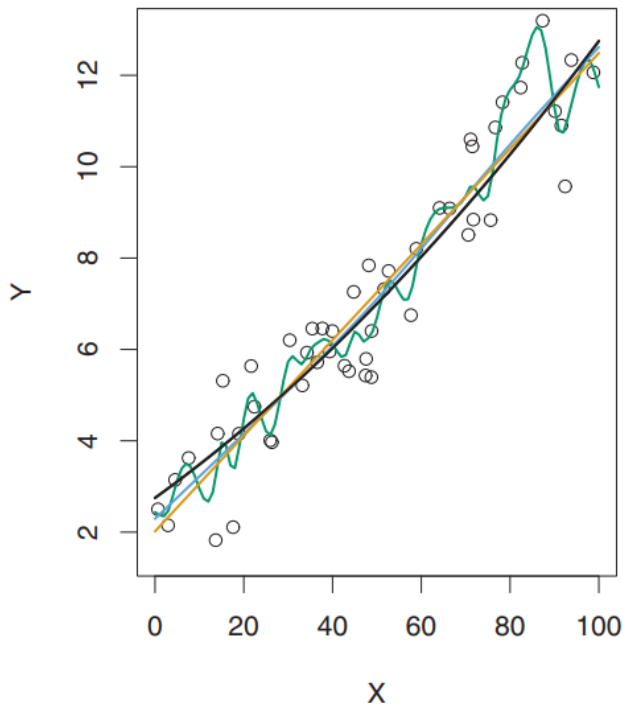
 Best model shown in blue – stays as close as possible to the expected function – moderately flexible

 Low training MSE, low test MSE


Issues in Model Performance

Model: Over-fitting versus Under-fitting


Synthetic Examples



 True (expected) function shown in black curve – almost linear

 Over-fitted function shown in green wiggly curve – follows signal+noise, most flexible, unnecessarily non-linear

 Low training MSE, high test MSE

 Under-fitted function shown in yellow – a linear model, did not miss much of the signal – least flexible

 Low training MSE, low test MSE, because truth is close to linear.

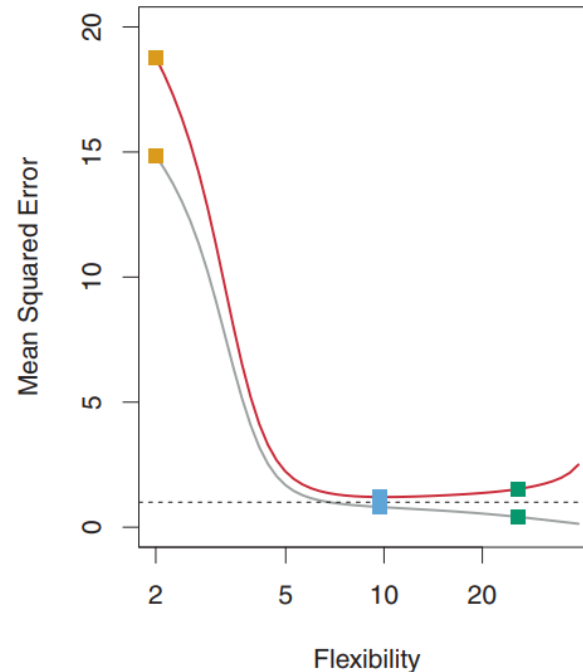
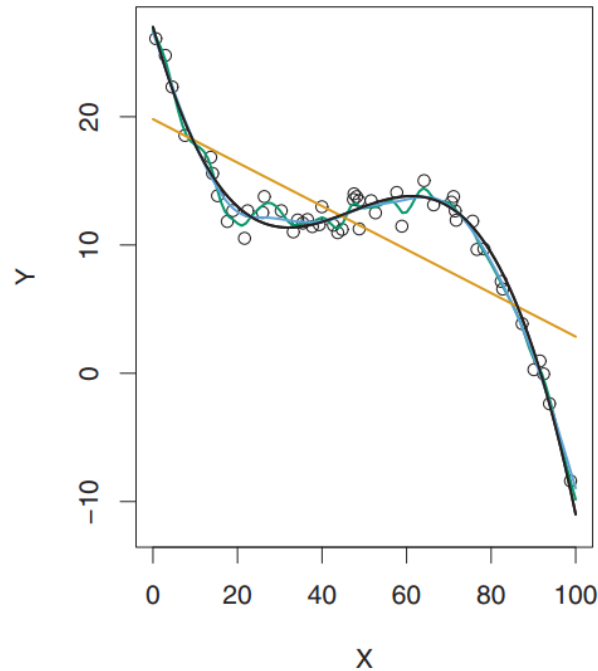
 Best model shown in blue – stays as close as possible to the expected function – moderately flexible


 Low training MSE, low test MSE


Issues in Model Performance

Model: Over-fitting versus Under-fitting


Synthetic Examples




 True (expected) function shown in black curve – clearly non-linear, not so much noise!

 Over-fitted function shown in green wiggly curve – follows signal closely, not much noise to follow, most flexible, necessarily non-linear

 Low training MSE, low test MSE

 Under-fitted function shown in yellow – a linear model, clearly miss much of the signal – least flexible

 high training MSE because of restriction following signal, high test MSE,

 Best model shown in blue – stays as close as possible to the expected function – moderately flexible

 Low training MSE, low test MSE


Issues in Model Performance

Model: Bias versus Variance


 The test MSE typically experiences a decreasing trend with increasing model flexibility up to a point; beyond that point, the test MSE increases with model flexibility.

 This diminishing return effect is due to the inter-play between two opposing phenomena – **Bias** and **Variance**; closely linked to the concepts of flexibility and simplicity.

Variance:

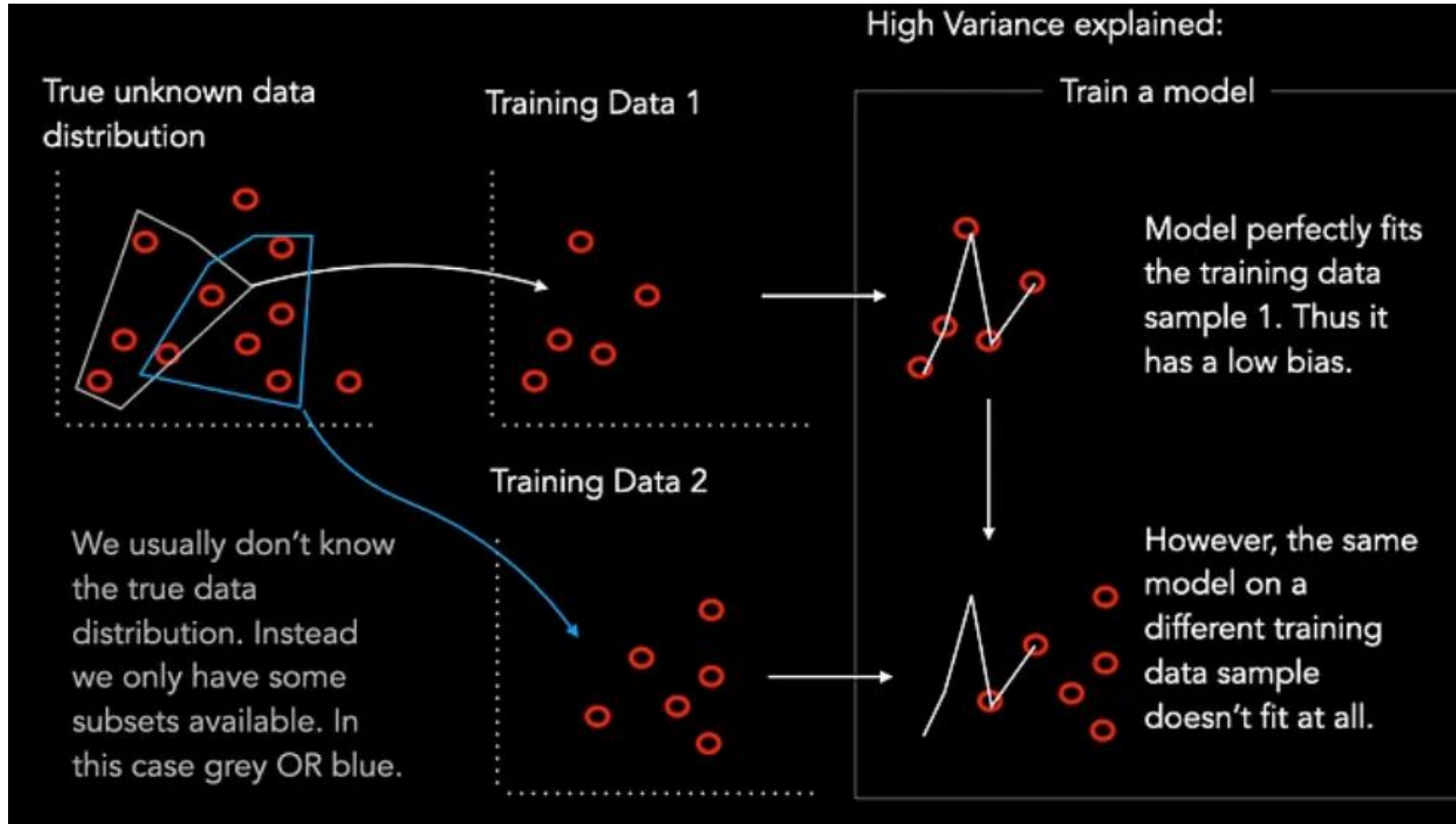
 a measure of the sensitivity of the predictions to changes in training data. Flexible models (that could fit complex f) are very sensitive

 a measure of the variability of the parameters of the predicted Function \hat{f} , if different training datasets were used.

 High variance implies that a small change in the training dataset may cause the value of parameters in \hat{f} to change remarkably.

Issues in Model Performance




Model: Bias versus Variance





Issues in Model Performance

Model: Bias versus Variance

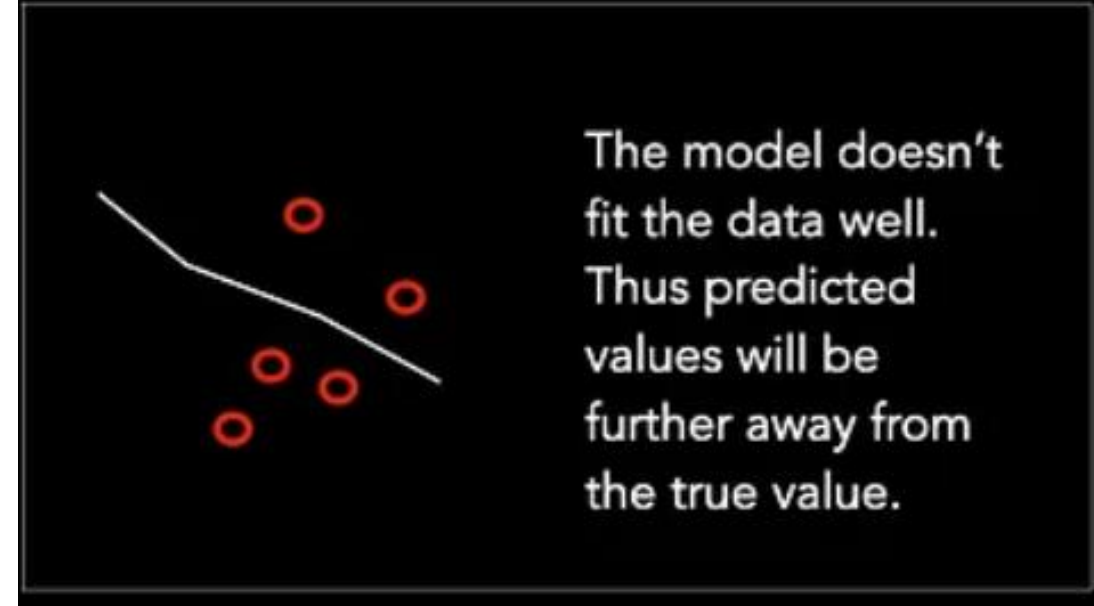
Bias:

-  A measure of the error due to approximating a complex input-output relationship with a simple model.
-  Over-simplification error.
-  Approximating a complex phenomenon with a too-simple model leads to high Bias.

So:

-  Flexible models typically give low Bias but high Variance, and
-  Simple models typically give low variance and high Bias

High Bias explained:



Issues in Model Performance

Model: Bias versus Variance

Mathematically, the expected test MSE, the Bias, the Variance are all related thus:

$$\text{Test MSE} = E(o_i - \hat{o}_i)^2 = \text{Var}(\hat{f}(i)) + [\text{Bias}(\hat{f}(i))]^2 + \text{Var}(\epsilon) \text{-----} -13$$

Notes about Equation 13:

$\text{Var}(\epsilon)$ is non-negative and irreducible

Both $\text{Var}(\hat{f}(i))$ and $[\text{Bias}(\hat{f}(i))]^2$ are non-negative;

both are reducible by the choice (of model)

variance is also reducible by increasing the quantity of training data

Hence, to minimize the test MSE, one of either $\text{Var}(\hat{f}(i))$ or $[\text{Bias}(\hat{f}(i))]^2$ must be reduced

That is, one of Variance or Bias must be reduced

The challenge:

a reduction in variance (by choosing less flexible model) leads to increase in bias

a reduction in bias (by choosing a more flexible model) leads to increase in variance


Issues in Model Performance

Model: Bias versus Variance

 To overcome the bias-variance challenge:

-  try out different models of varying flexibility

-  different algorithms: e.g. support vector machine and regression tree

-  same algorithm with different hyper-parameter: e.g. different number of trees in random forest

-  for each trial, estimate the Test MSE

-  choose the model with the lowest Test MSE

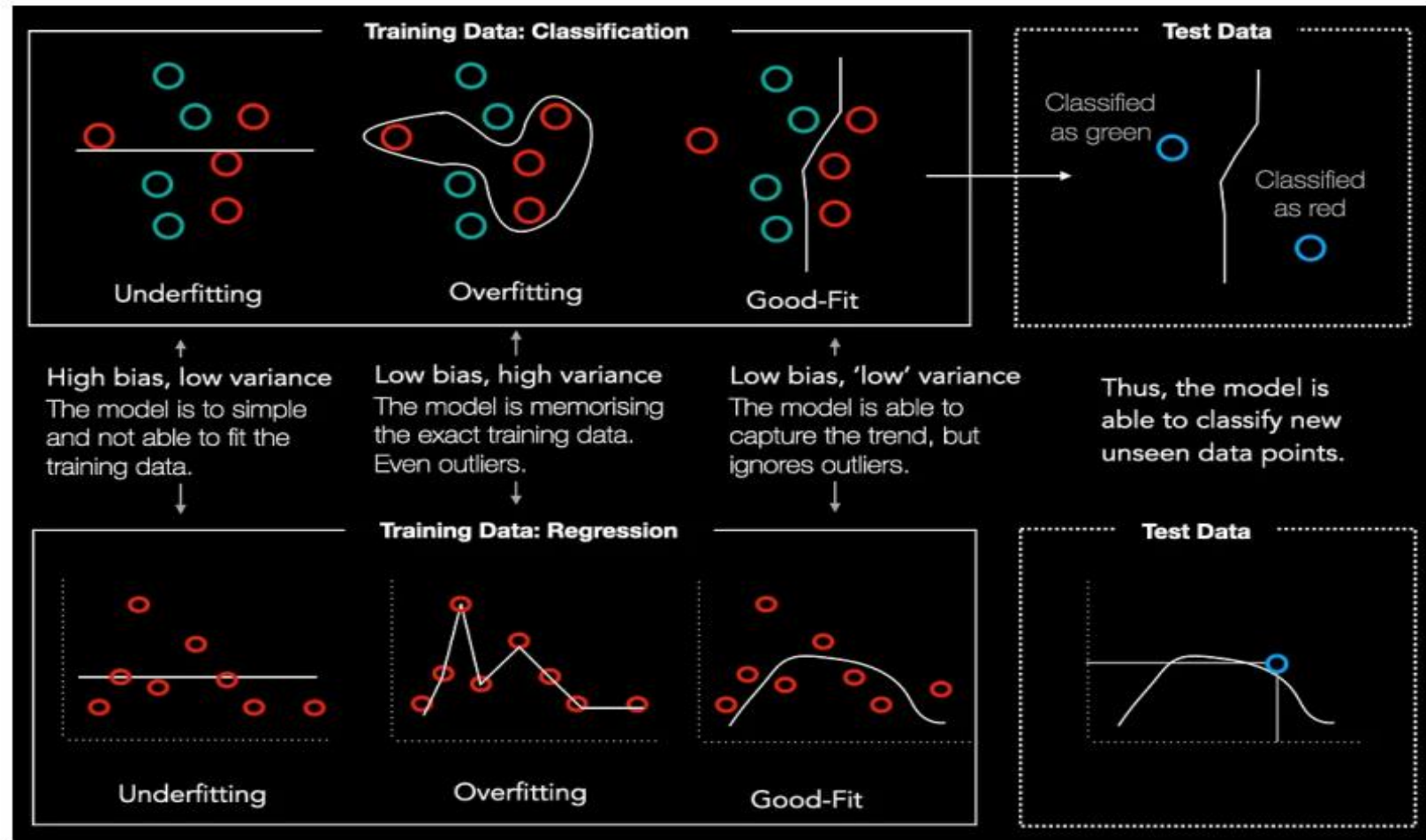
-  That is the model that struck a balance between bias and variance

 The idea of striking a balance between bias and variance is popularly known as **Bias-Variance Trade-off**.

Issues in Model Performance

Model: Bias versus Variance

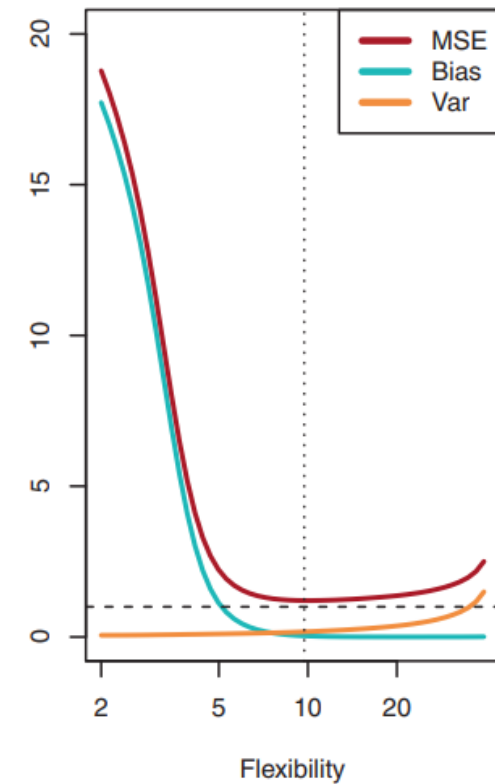
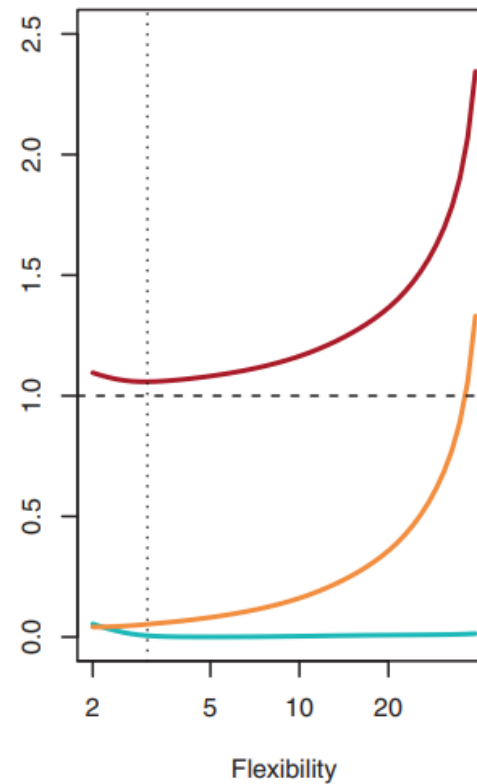
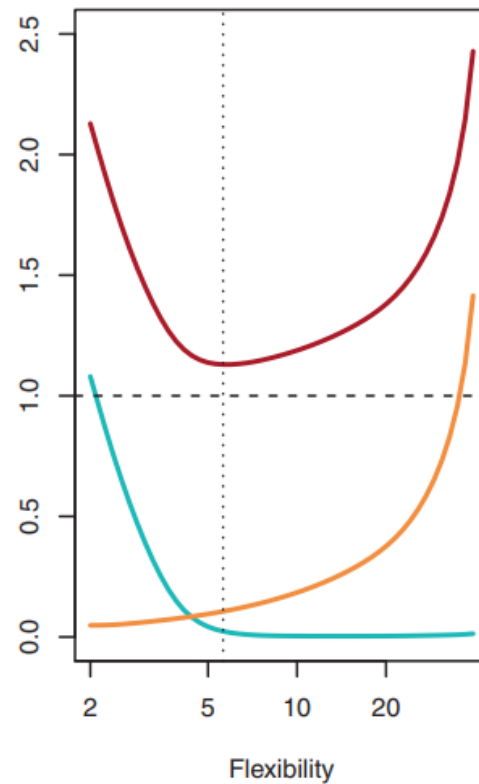
Bias-Variance Trade-off.








Issues in Model Performance

Model: Bias versus Variance








 Bias-Variance Trade-off.



Cross-validation

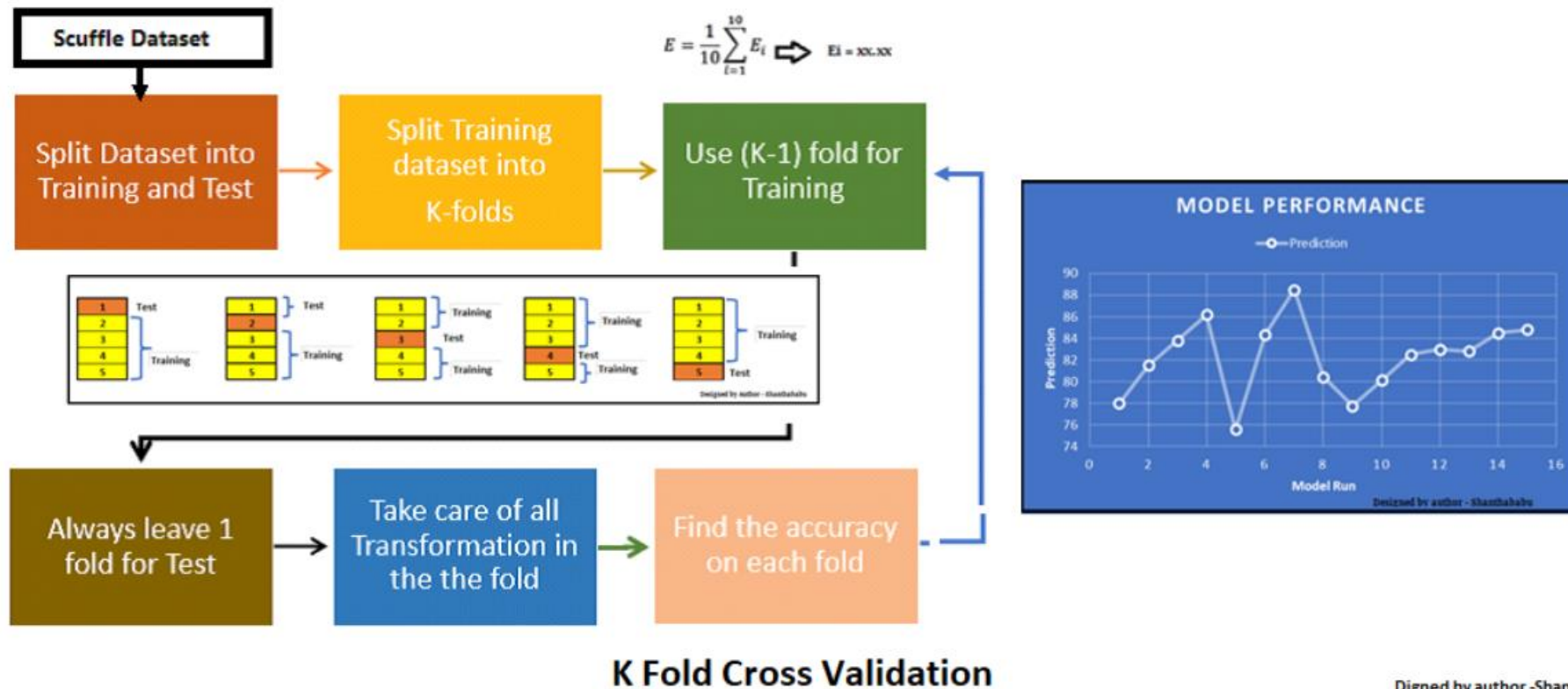
-  Availability of adequate quantity of data is typically an issue in machine learning.
-  Often, the quantity of data points available for a project is barely sufficient for training; such it is not possible to keep back a portion of data for testing.
-  As already established, it is not advisable to use training data to estimate MSE for model evaluation purposes.
 -  Because minimal Training MSE is not a guarantee for minimal Test MSE.
-  In such cases where a test data is not available, cross-validation is the way out!

Cross-validation

-  The basic idea in Cross-validation is to iteratively leave out a small portion of the training dataset during training, and then deploy the left-out portion for testing after training.
-  For each iteration, a different portion of the training data is left-out.
-  Ultimately, every data point in the training data get left-out at a particular iteration.
-  At the end of all iterations, the Test MSE values (one for each iteration) are averaged to obtain a single measure of Test MSE for the model.
-  Two types of Cross-validation methods:
 -  Leave-one-out cross validation
 -  K-fold cross validation.

Cross-validation

Model: Over-fitting versus Under-fitting



Issues in Model Performance

Model: Over-fitting versus Under-fitting

