McMaster University ENGINEERING

W BOOTH SCHOOL OF ENGINEERING PRACTICE AND TECHNOLOGY

SEP 785: Machine Learning

Lecture 7: Model Evaluation

Instructor: Dr. Dalia Mahmoud, PhD
(Mechanical Engineering, McMaster University)
Email: mahmoudd@mcmaster.ca

Recap

ILOs

1. Explain bias and variance, identify the trade-off, and mitigate high bias/variance.
2. Understand cross-validation, perform k-fold, and compare techniques.
3. Identify and tune hyperparameters using grid/random search, and evaluate their impact on performance.
4. Differentiate between common loss functions (MSE, Cross-Entropy) and choose the appropriate one for regression vs. classification.
5. Define, calculate, and interpret regression and classification metrics.
6. Define and explain Big O notation, calculate algorithm complexity, and compare efficiency.
7. Distinguish between deterministic and probabilistic models, and apply both to solve uncertainty-related problems.

Contents

Model Evaluation

- To know whether we can trust our method or system, we need to evaluate it.
- Model selection: choose between different models in a data-driven way.
- If you cannot measure it, you cannot improve it.
- Convince others that your work is meaningful
  - Peers, leadership, clients, yourself(!)
- When possible, try to interpret what your model has learned
- The signal your model found may just be an artifact of your biased data

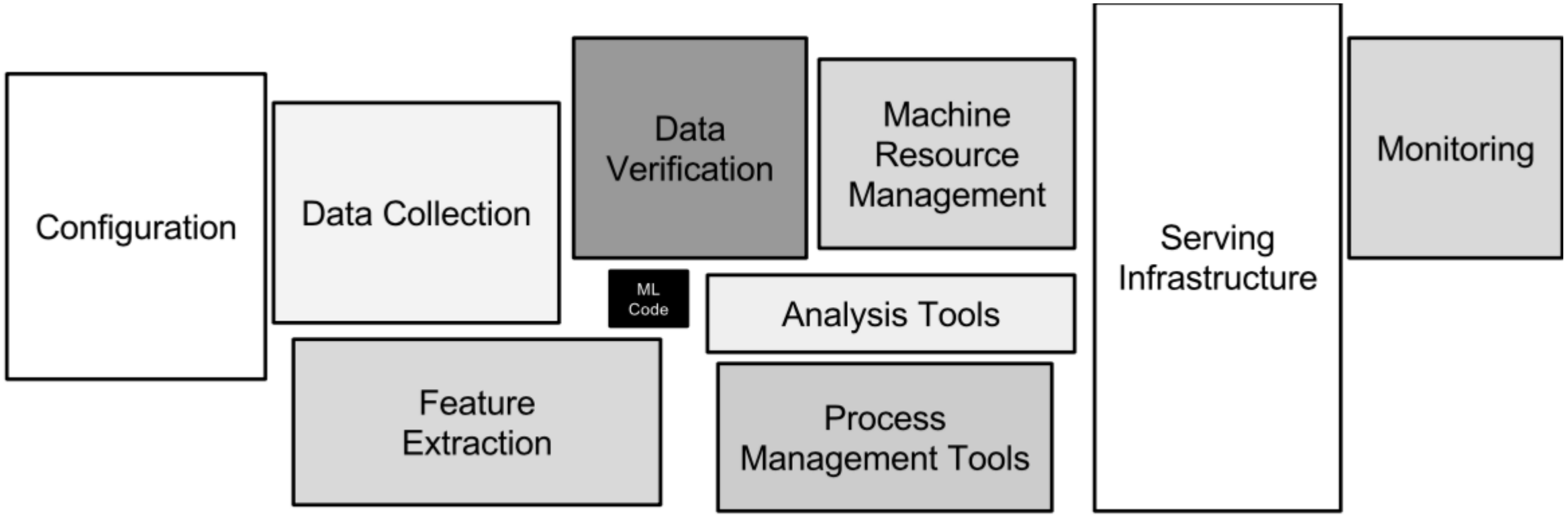
See 'Why Should I Trust You?' by Marco Ribeiro et al.

Designing Machine Learning systems

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

- Just running your favourite algorithm is usually not a great way to start
- Consider the problem: How to measure success? Are there costs involved?
- Do you want to understand phenomena or do black box modelling?
- Analyze your model's mistakes. Don't just finetune endlessly.
- Build early prototypes. Should you collect more, or additional data?
- Should the task be reformulated?
- Overly complex machine learning systems are hard to maintain

Bigger Picture

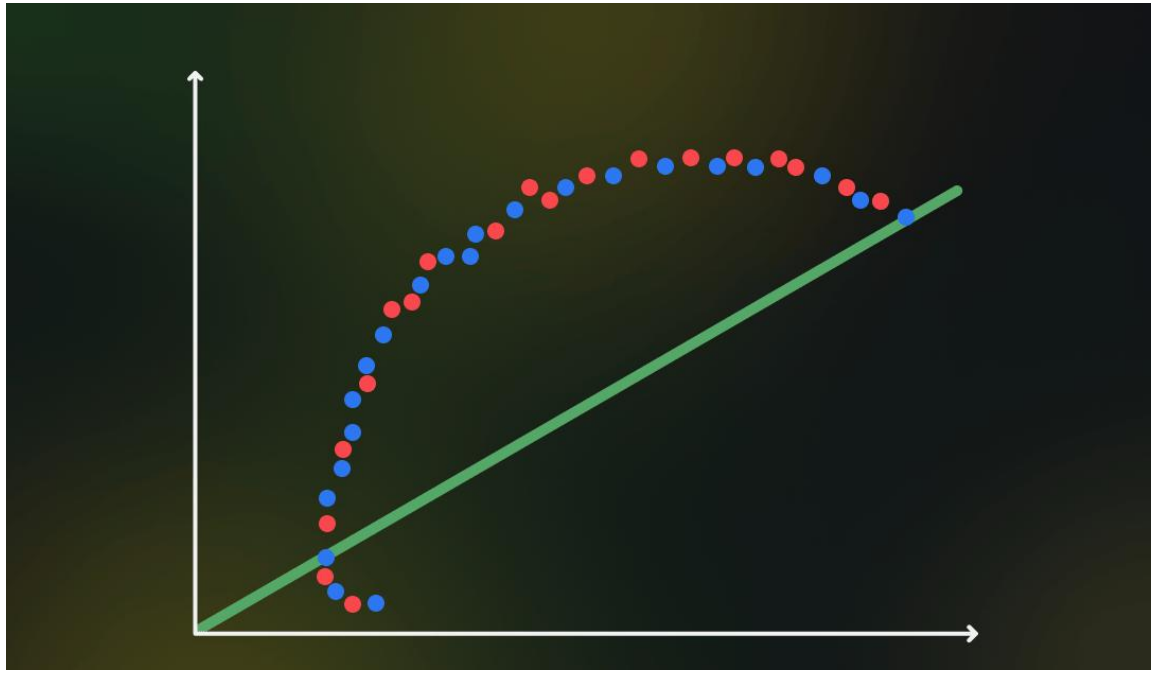


Only a small fraction of real-world ML systems is composed of the ML code

Contents

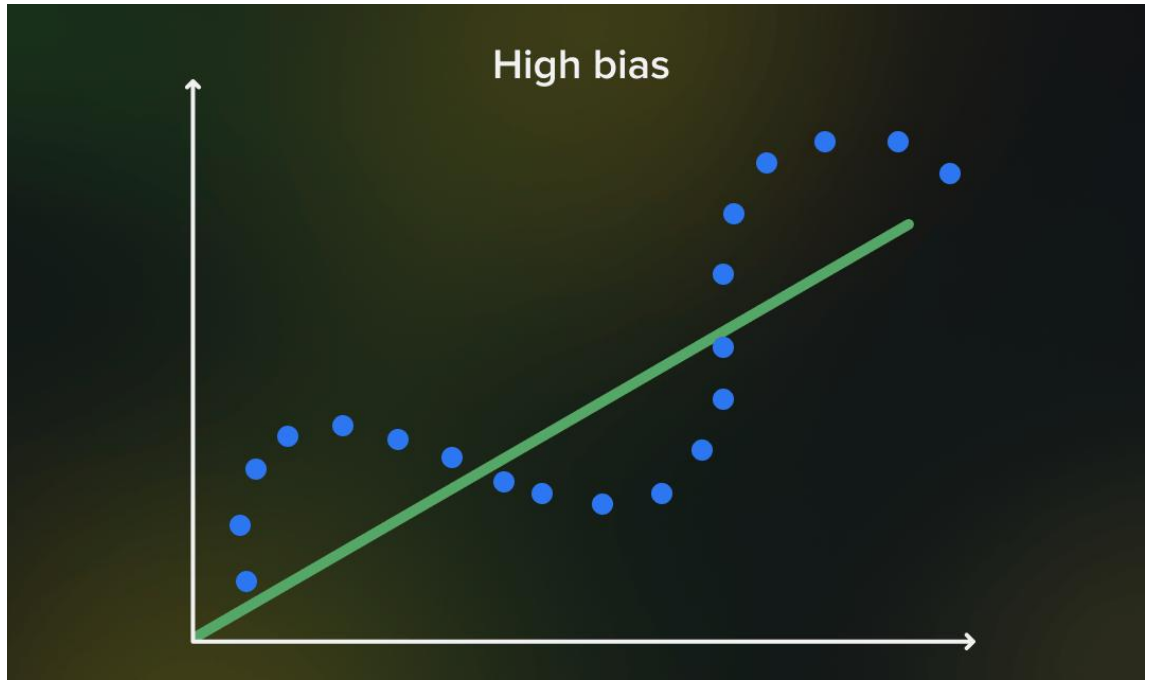
Bias in ML

- Bias in machine learning refers to the difference between a model's predictions and the actual distribution of the value it tries to predict.
- Models with high bias oversimplify the data distribution rule/function, resulting in high errors in both the training outcomes and test data analysis results.



High-bias model features

- Underfitting. the proposed distribution does not correspond to the actual distribution.
- Low training accuracy. The lack of proper processing of training data results in high training loss and low training accuracy.
- Oversimplification. The oversimplified nature of high-bias models limits their ability to identify complex features in the training data, making them inefficient for solving complicated problems.

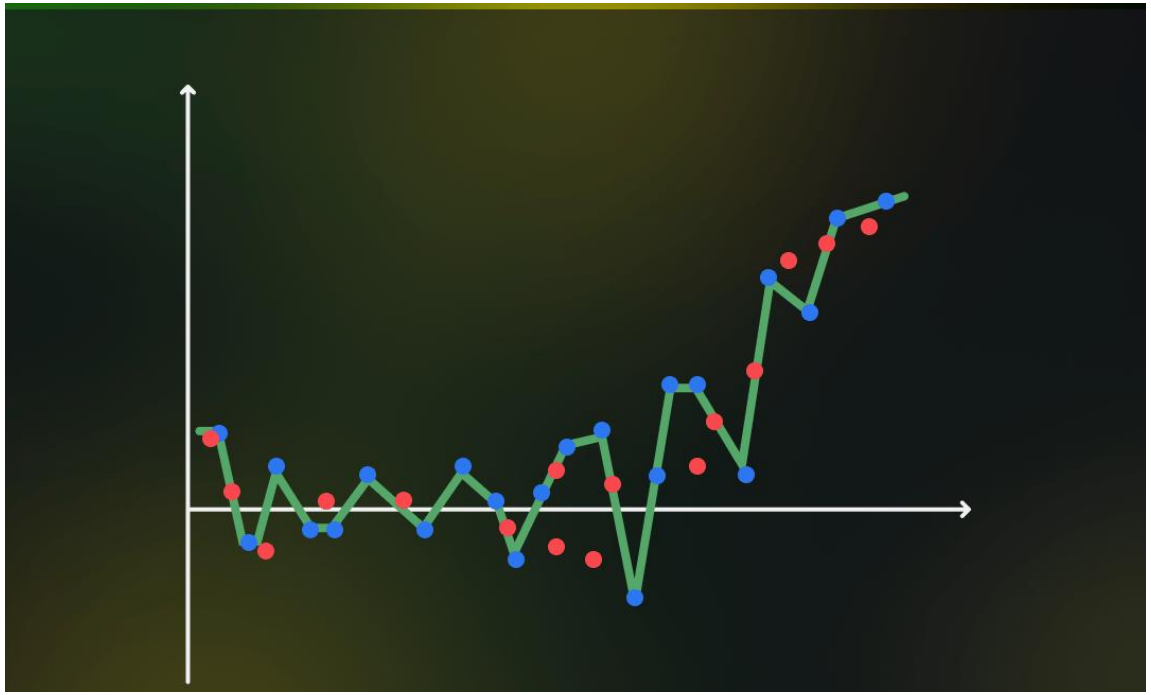


How to reduce high Bias

- Incorporating additional features from data to improve the model's accuracy.
- Increasing the number of training iterations to allow the model to learn more complex data.
- Avoiding high-bias algorithms such as linear regression, logistic regression, discriminant analysis, etc. and instead using nonlinear algorithms such as k-nearest neighbors, SVM, decision trees, etc.
- Decreasing regularization at various levels to help the model learn the training set more effectively and prevent underfitting.

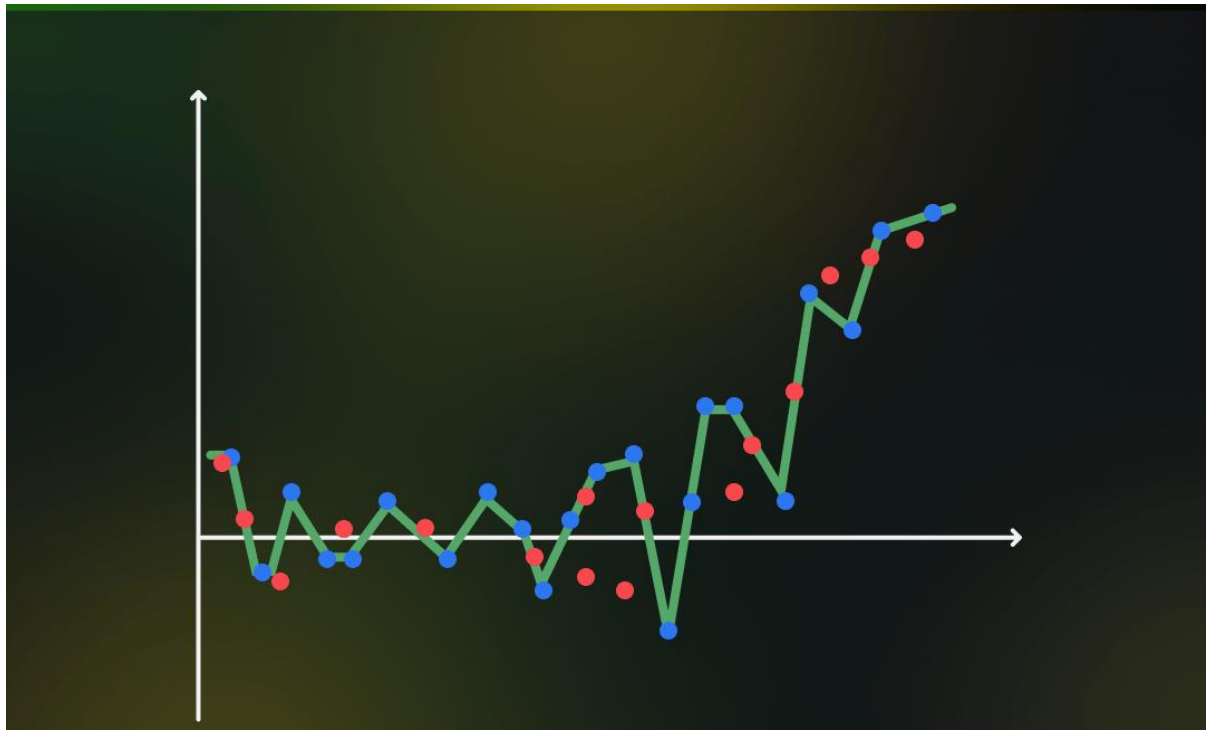
High Variance in ML

- A model with high variance is highly sensitive to the training data, meaning that small changes in the data can lead to large changes in the model's predictions.
- This typically happens when the model is too complex (e.g., a very deep decision tree or a high-degree polynomial regression), and it "overfits" to the training data.
- While the model may perform very well on the training set, it will likely perform poorly on unseen data (i.e., it doesn't generalize well).

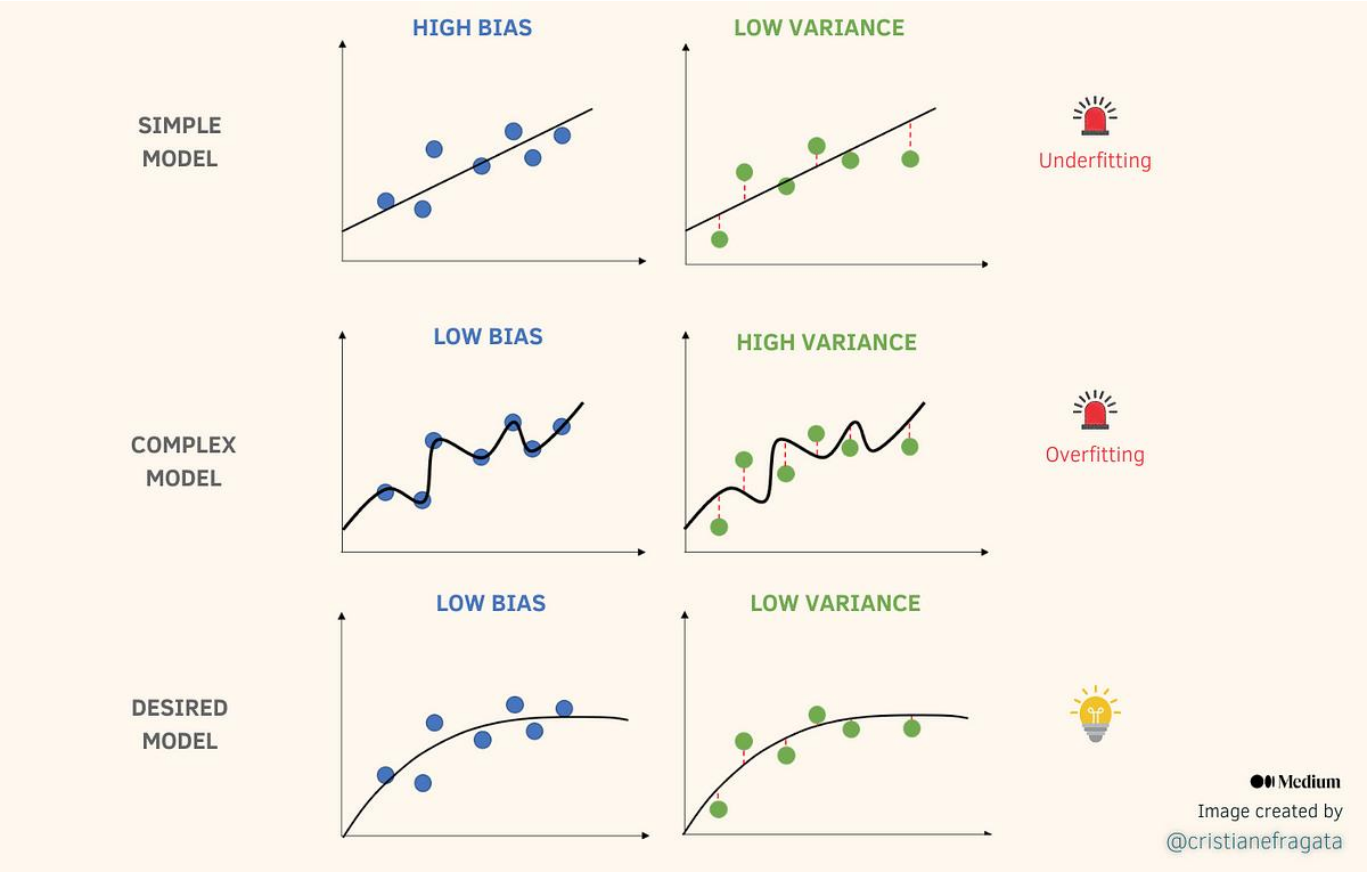


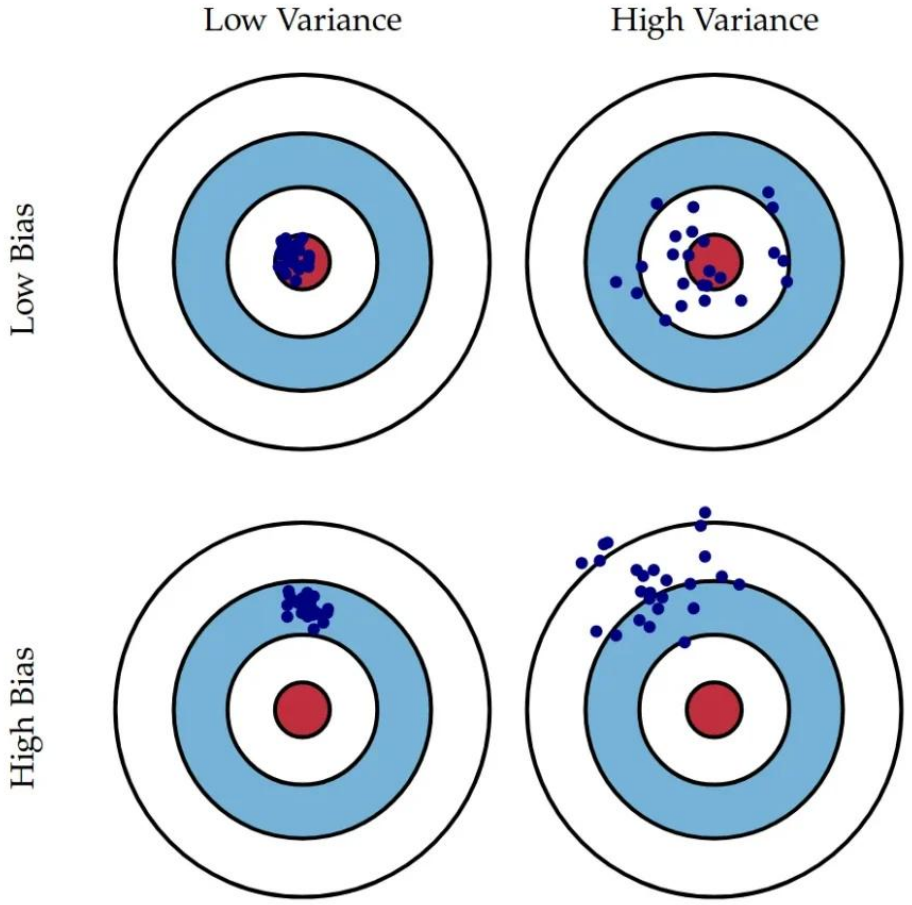
High Variance in ML

- Low testing accuracy. Despite high accuracy on training data, high variance models tend to perform poorly on test data.
- Overfitting. A high-variance model often leads to overfitting as it becomes too complex.
- Overcomplexity. As researchers, we expect that increasing the complexity of a model will result in improved performance on both training and testing data sets.



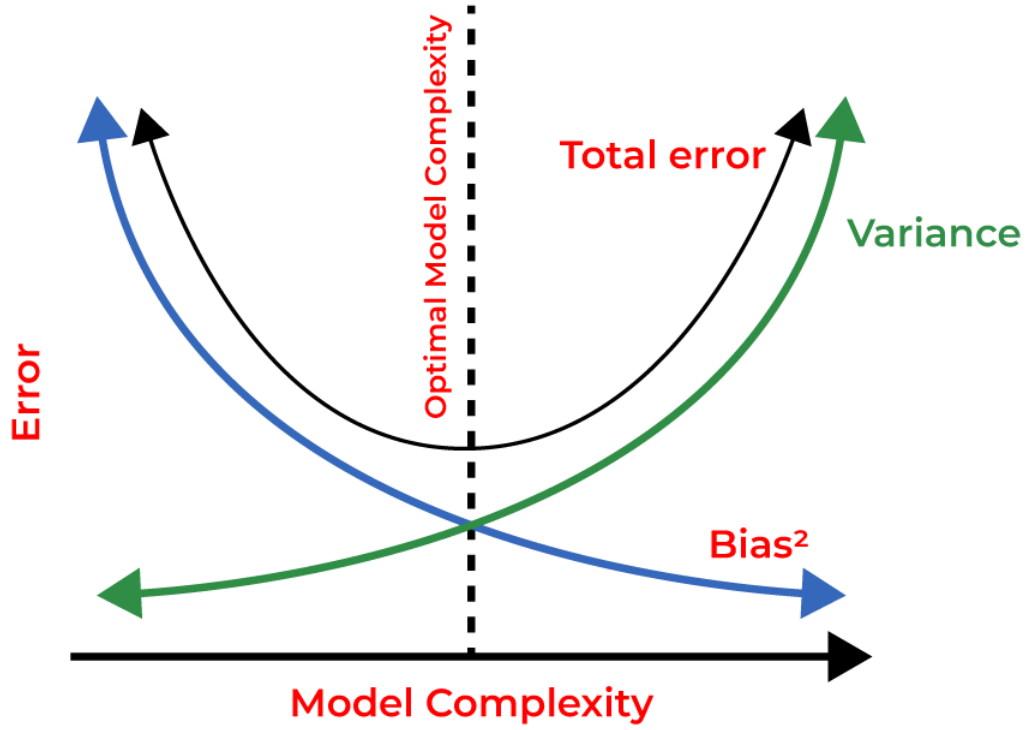
Bias Variance Tradeoff





Bias Variance Tradeoff

- Overcomplexity typically leads to **low bias** (since the model fits the training data very well) and **high variance** (since the model is too sensitive to fluctuations in the training data). The goal is to find the right balance where the model is complex enough to capture the underlying patterns but not so complex that it overfits to the training data.

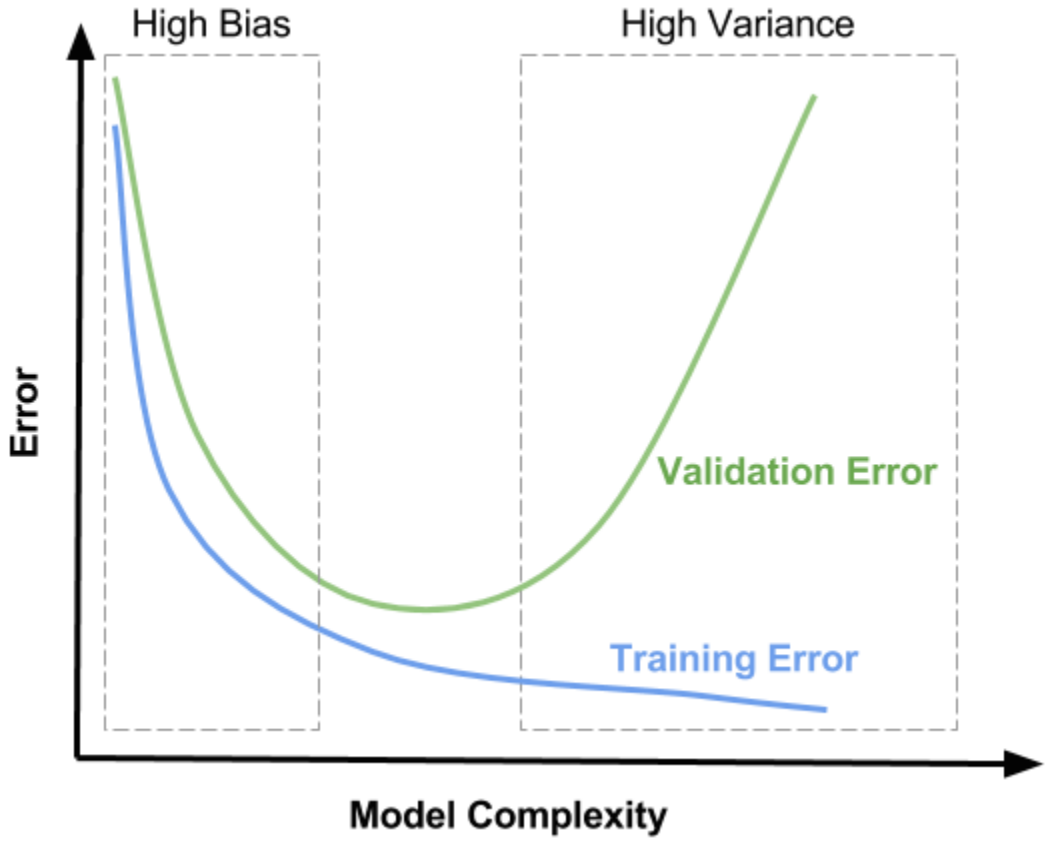


$$\text{Total Error} = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

Bias Variance Tradeoff

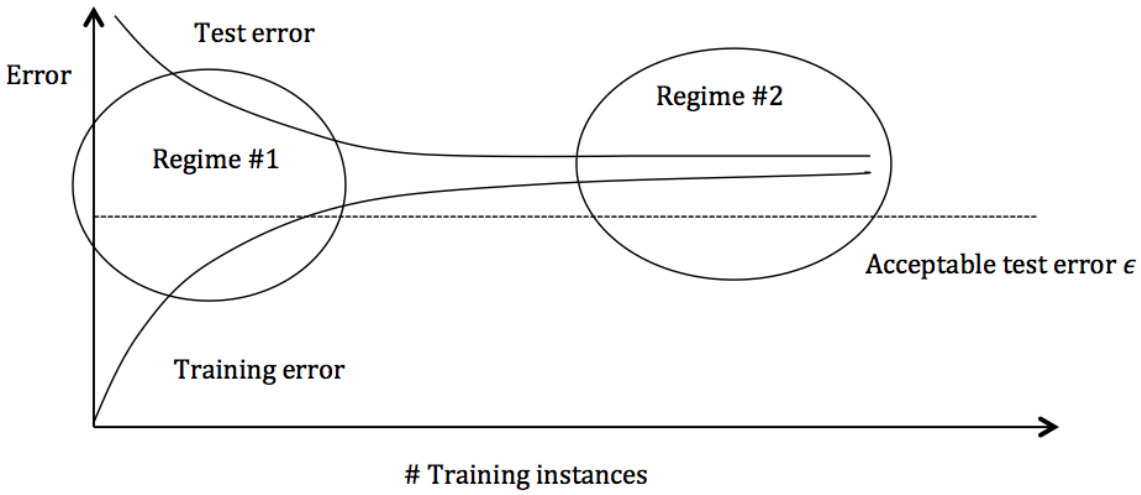
McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

- Low Bias (High Complexity): The model fits the training data well, but if it becomes too complex, it starts to overfit and have high variance. This leads to high validation error while the training error continues to decrease.
- High Bias (Low Complexity): The model underfits the data, resulting in high training error and high validation error.



Bias Variance Tradeoff

- The graph above plots the training error and the test error and can be divided into two overarching regimes. In the first regime (on the left side of the graph), training error is below the desired error threshold (denoted by ϵ), but test error is significantly higher. In the second regime (on the right side of the graph), test error is remarkably close to training error, but both are above the desired tolerance of ϵ



Bias Variance Tradeoff

Regime 1 (High Variance)

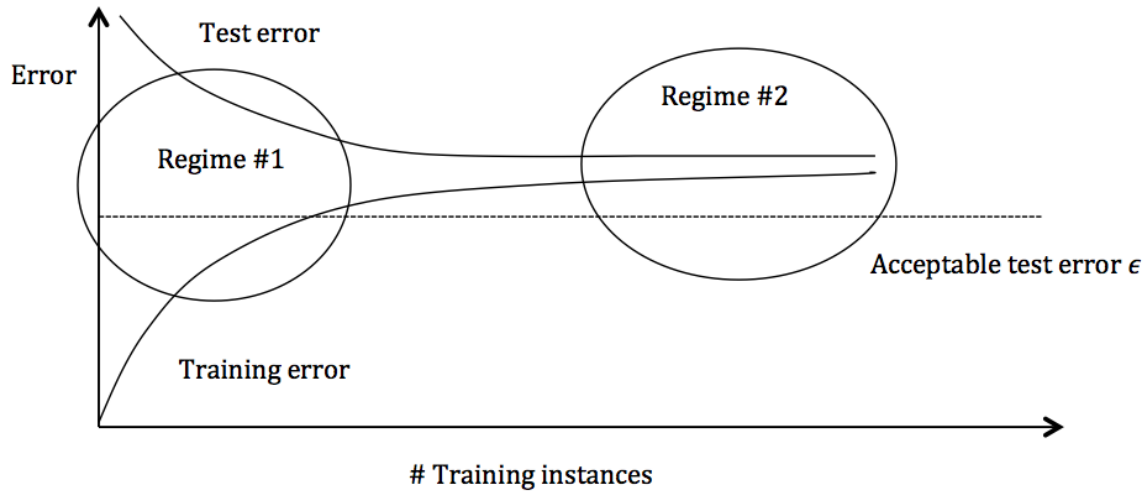
- In the first regime, the cause of the poor performance is high variance.

Symptoms:

1. Training error is much lower than test error
2. Training error is lower than ϵ
3. Test error is above ϵ

Remedies:

1. Add more training data
2. Reduce model complexity -- complex models are prone to high variance
3. Bagging (will be covered later in the course)



Bias Variance Tradeoff

Regime 2 (High Bias)

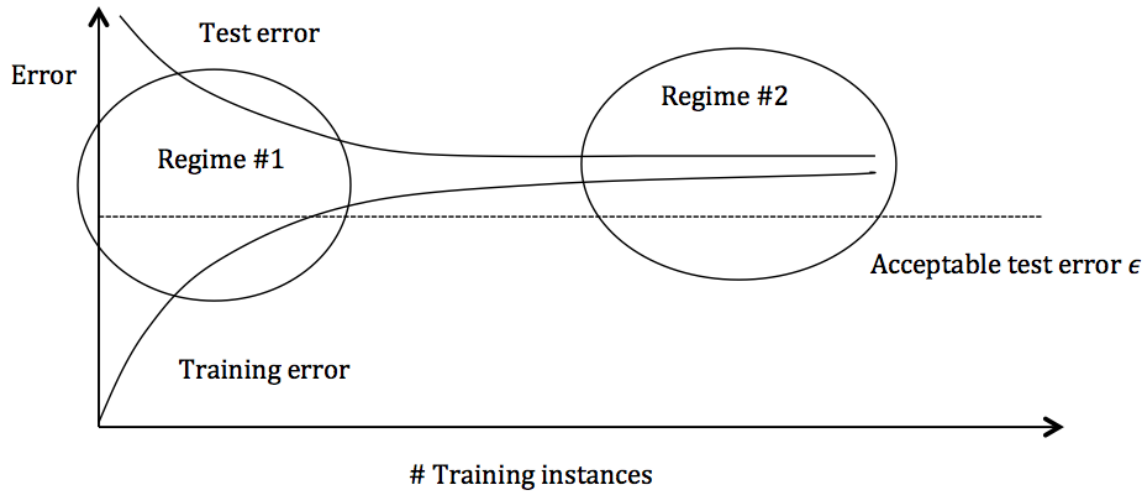
Unlike the first regime, the second regime indicates high bias: the model being used is not robust enough to produce an accurate prediction.

Symptoms:

1. Training error is higher than ϵ

Remedies:

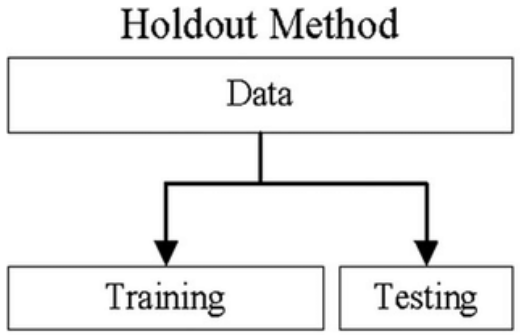
1. Use more complex model (e.g. kernelize, use non-linear models)
2. Add features
3. Boosting (will be covered later in the course)



Contents

Hold Out Vs Cross Validation

- Hold-out is when you split up your dataset into a ‘train’ and ‘test’ set. The training set is what the model is trained on, and the test set is used to see how well that model performs on unseen data.
- **Cross-validation** is a technique used in machine learning to assess how well a model generalizes to unseen data.

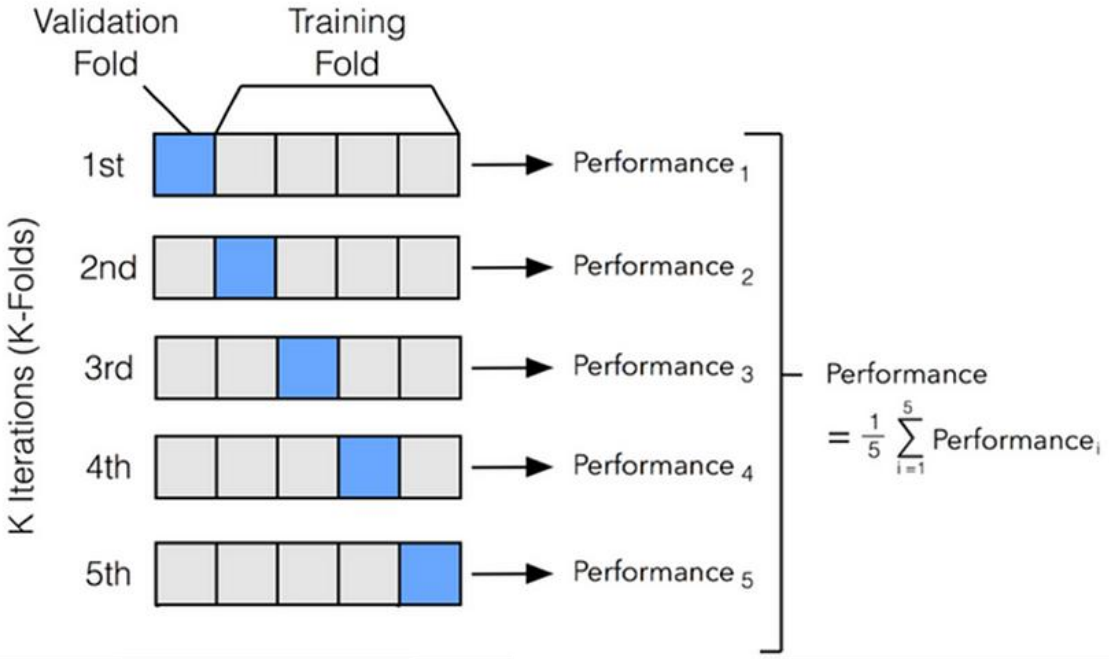


5 K-fold Validation Method

Data					
Test	Train	Train	Train	Train	Fold 1
Train	Test	Train	Train	Train	Fold 2
Train	Train	Test	Train	Train	Fold 3
Train	Train	Train	Test	Train	Fold 4
Train	Train	Train	Train	Test	Fold 5

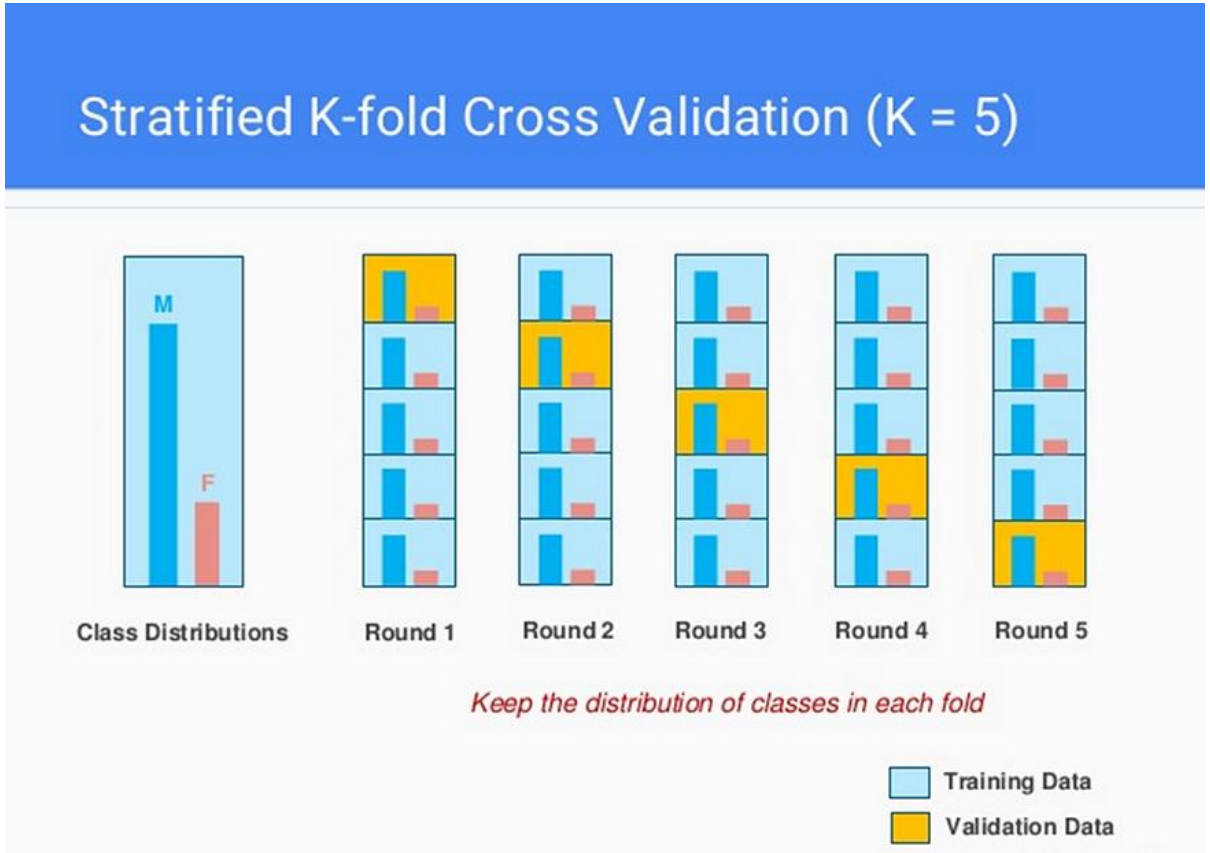
k-Fold Cross-Validation:

- The dataset is divided into k equally sized folds (subsets).
- The model is trained on k-1 of those folds and tested on the remaining one fold.
- This process is repeated k times, with each fold serving as the test set once.
- The final model performance is the average performance across all k iterations.
- **Test set** is used **after** cross-validation to assess final model performance, but it is not used during the cross-validation process itself.



Stratified k-Fold Cross-Validation:

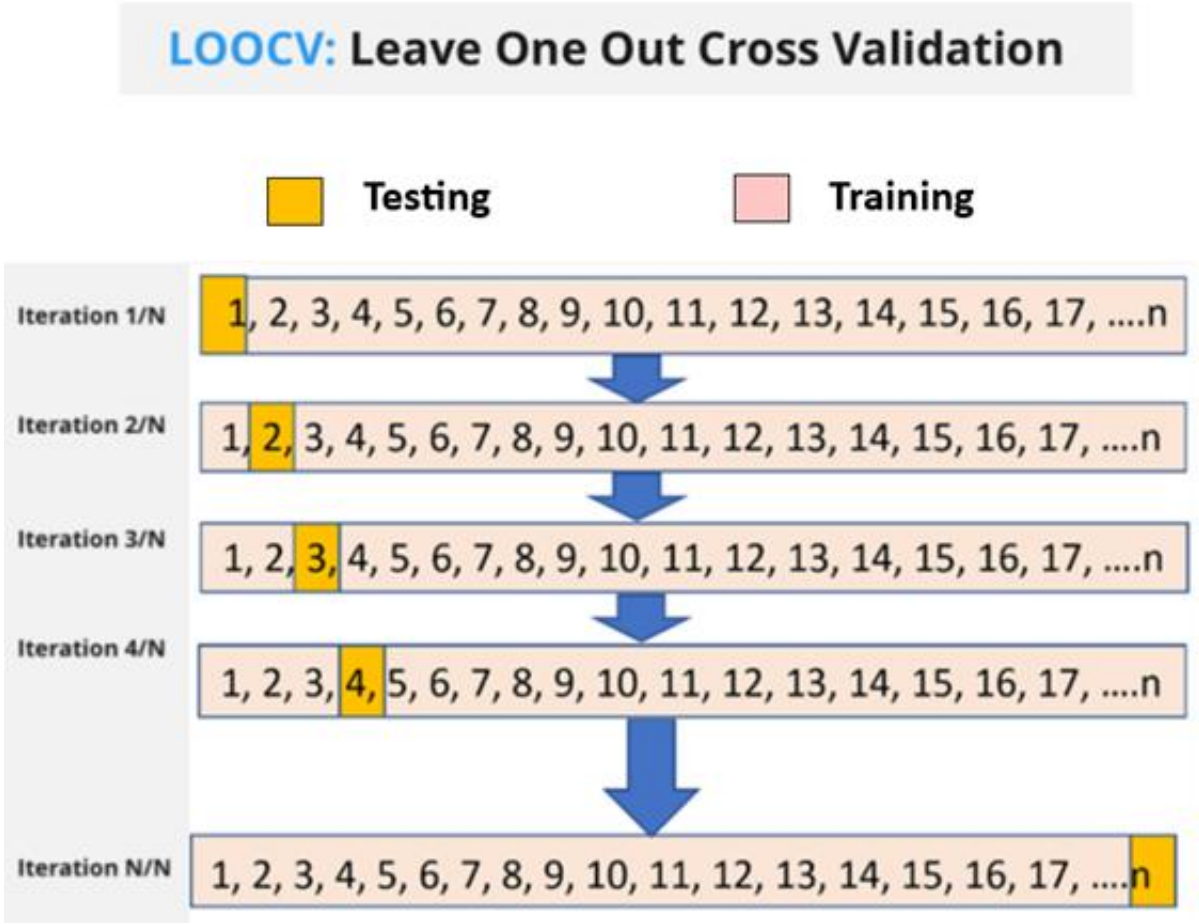
- A variation of k-fold cross-validation where the data is split in such a way that each fold has the same proportion of each class as the entire dataset (useful for imbalanced datasets).
- This ensures that the model is trained and tested on data that represents the overall class distribution, making the performance estimates more reliable.



Leave-One-Out Cross-Validation (LOO CV):

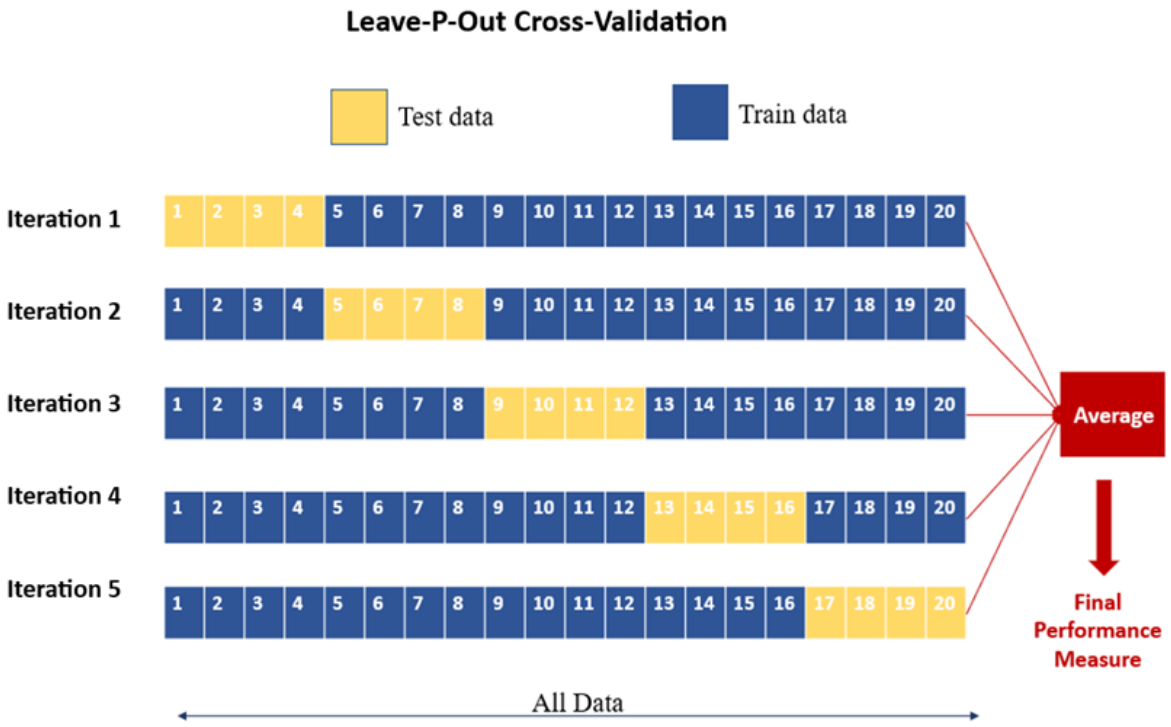
McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

- A special case of k-fold cross-validation where k is equal to the number of data points.
- In each iteration, a single data point is used as the test set, and the model is trained on the remaining data points.
- This is particularly useful when the dataset is small, but it can be computationally expensive for large datasets.



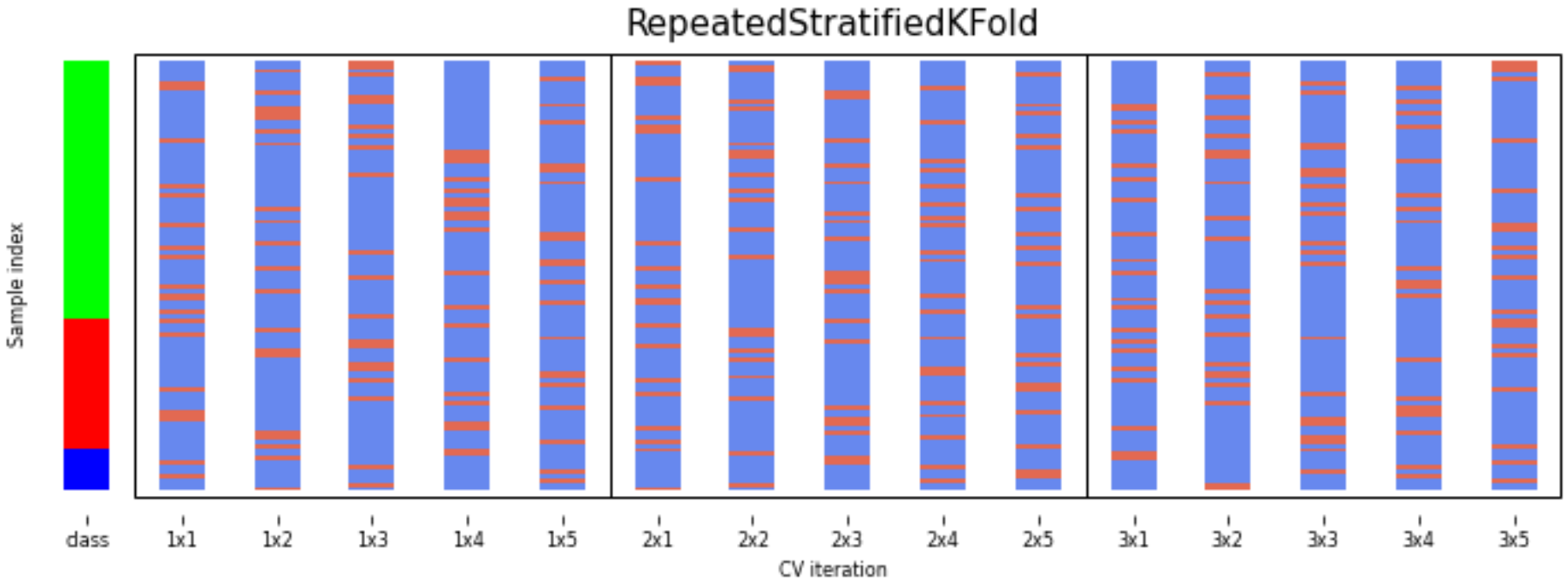
Leave-P-Out (LPO):

- Leave-P-Out (LPO) cross-validation is a method of cross-validation similar to Leave-One-Out (LOO) but instead of leaving out one sample at a time, it leaves out p samples at a time. In other words, it uses n-p samples of the dataset as the training set and p samples as the test set, where n is the total number of samples in the dataset. This process is repeated in all possible ways of choosing p samples from the dataset.



Repeated k-Fold cross-validation:

- Involves repeating k-fold cross-validation multiple times with different random splits of the data. This helps to get a more reliable estimate of model performance, especially when the data set is small or noisy.

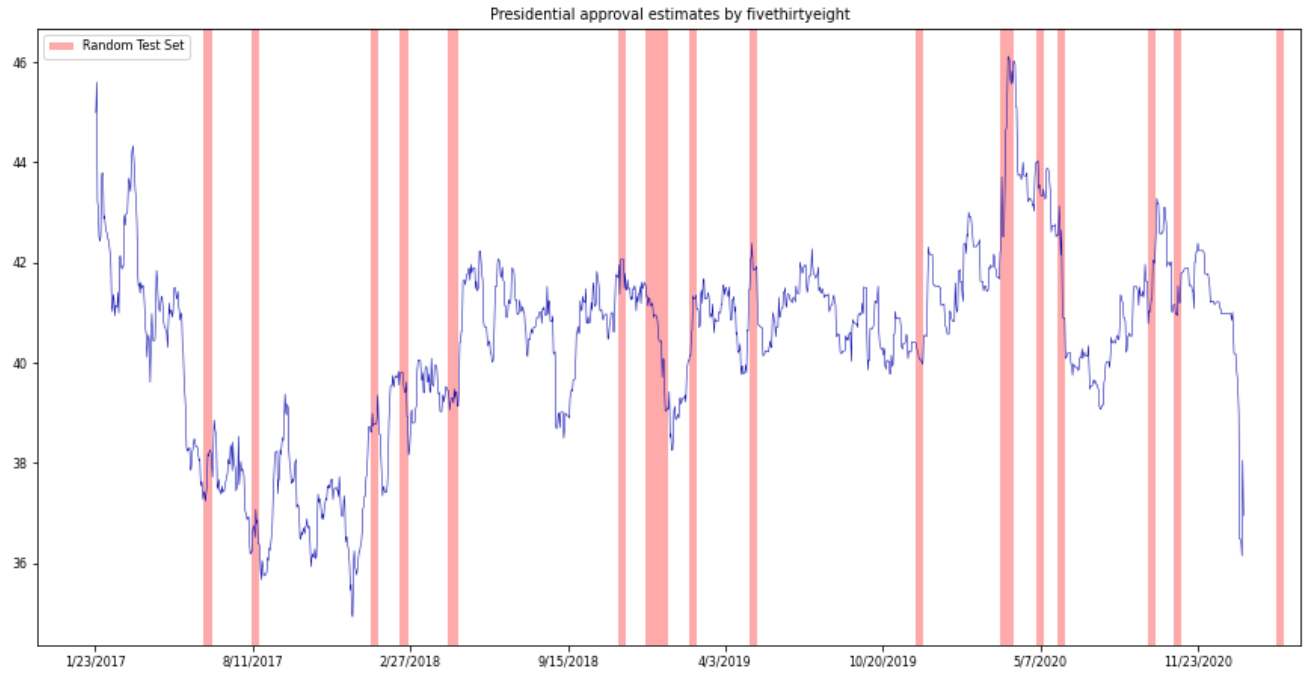


Time series

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

Time series data refers to data points indexed in time order, and it is used in various fields like finance, healthcare, and weather forecasting.

Cross-validation in time series requires special handling to respect the temporal structure of the data, with methods like chronological splitting, rolling window, and expanding window cross-validation.



Choosing between different CV methods

No strict rules, only guidelines:

- Always use stratification for classification (sklearn does this by default)
- Use holdout for very large datasets (e.g. >1.000.000 examples) Or when learners don't always converge (e.g. deep learning)
- Choose k depending on dataset size and resources
- Use leave-one-out for very small datasets (e.g. <100 examples)
- Use cross-validation otherwise
- Most popular (and theoretically sound): 10-fold CV
- Literature suggests 5x2-fold CV is better
- Use train-then-test for time series

Choosing between different CV methods

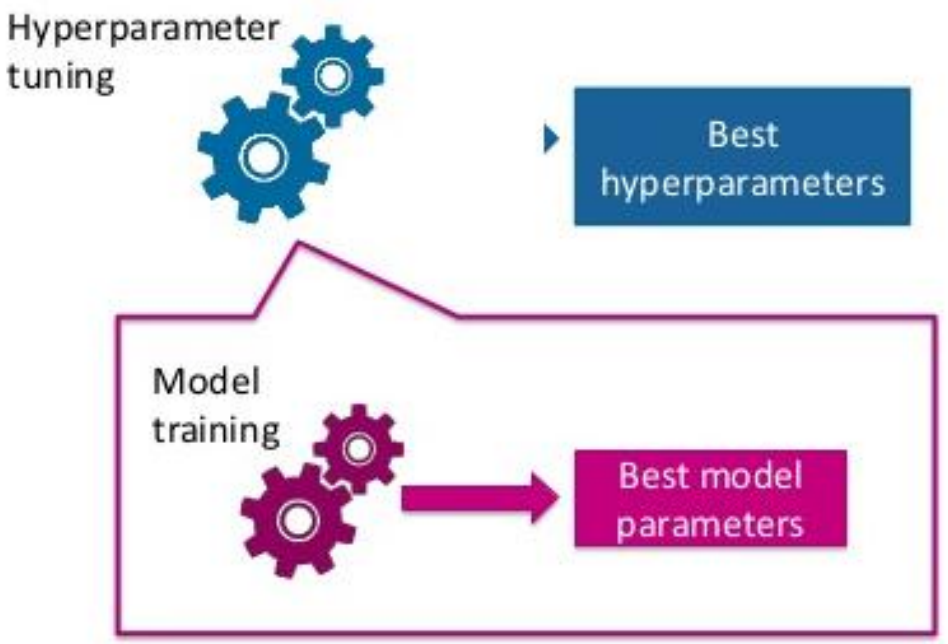
McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

Technique	Description	Use Case	Advantages	Disadvantages
K-Fold Cross-Validation	Split data into **k** folds, train on **k-1**.	General-purpose, moderate datasets.	Balanced training & validation.	Computationally expensive for large datasets.
Stratified K-Fold	K-Fold with class distribution preserved.	Imbalanced classification tasks.	Ensures representative classes.	More complex, costly.
Leave-One-Out (LOO)	Train on all but one sample, test on the left out.	Small datasets (<100 samples).	Maximizes data usage.	Expensive for large datasets.
Time Series Cross-Validation	Train on past, test on future (chronological).	Time-series data.	Avoids future data leakage.	Not for non-temporal data.
Repeated k-Fold (2x5)	Train/validate on 2 splits, repeat 5 times.	Reduces variance in performance.	Low variance in results.	Computationally expensive.

Contents

Parameters Vs Hyperparameters

- **Parameters** are the values that a machine learning model learns during training. These parameters are adjusted automatically during the training process to minimize the error in the model's predictions.
- **Hyperparameters** are values set before the training process begins and control how the model is trained. Hyperparameters are not learned from the data but are chosen manually or through optimization methods like cross-validation.



Parameters Vs Hyperparameters

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

Method	Parameters	Hyperparameters
Linear Regression	Weights (coefficients), Intercept (bias term).	Regularization strength (L1, L2), Learning rate (if using gradient descent).
Logistic Regression	Weights (coefficients), Intercept (bias term).	Regularization strength (L1, L2), Learning rate (if using gradient descent).
Decision Tree	Tree structure, Split points, Leaf values, Gini impurity or Entropy (for classification).	Maximum depth, Minimum samples per leaf, Maximum features to split on.
Support Vector Machine (SVM)	Support vectors, Weights (coefficients), Bias term.	C (regularization parameter), Kernel type (linear, RBF, etc.), Gamma.
K-Nearest Neighbors (KNN)	None (lazy learner, directly uses training data).	Number of neighbors (k), Distance metric (e.g., Euclidean, Manhattan).
Naive Bayes	Class probabilities, Likelihoods of features given class.	Smoothing parameter (Laplace, Lidstone), Priors for each class.

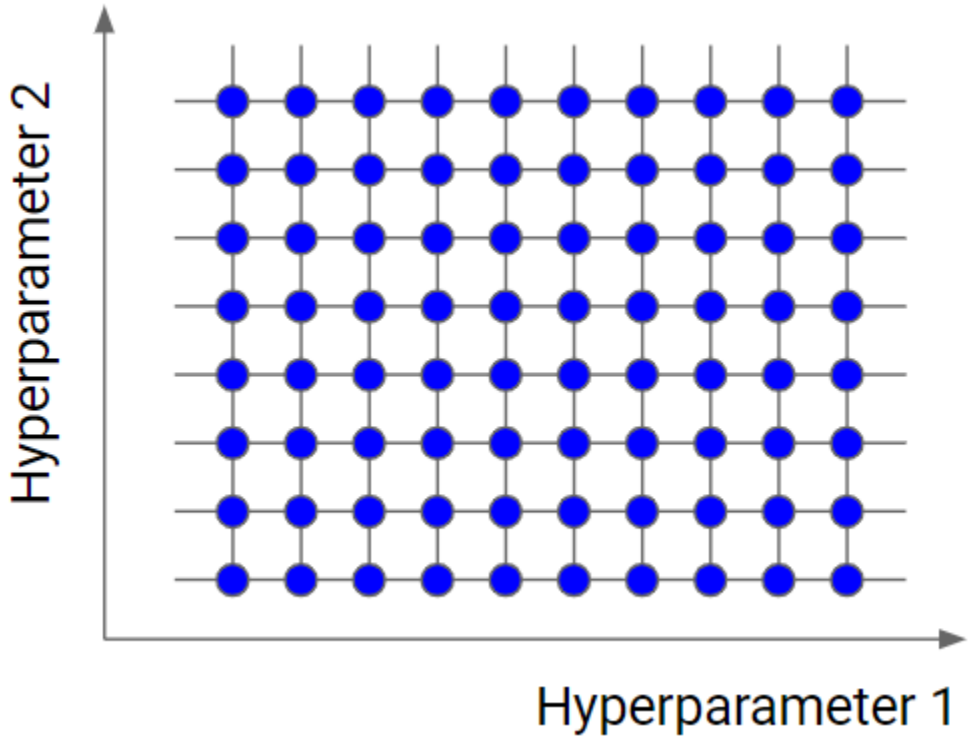
Parameters Vs Hyperparameters

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

Method	Parameters	Hyperparameters
Random Forest	Split points for each decision tree, Leaf values.	Number of trees (n_estimators), Maximum depth, Minimum samples per split.
Naive Bayes	Class probabilities, Likelihoods of features given class.	Smoothing parameter (Laplace, Lidstone), Priors for each class.
Neural Networks	Weights and biases of each layer, Activation outputs.	Number of layers, Number of neurons in each layer, Learning rate, Batch size, Dropout rate, Epochs.
K-Means Clustering	Cluster centers (centroids), Assignment of points to clusters.	Number of clusters (k), Initialization method (e.g., K-means++, Random), Number of iterations.
Gradient Boosting	Model weights, Leaf values, Gradient updates.	Number of trees, Learning rate, Maximum depth, Minimum samples per leaf.
XGBoost	Weights, Leaf values, Splitting criteria.	Number of boosting rounds (estimators), Learning rate, Maximum depth, Subsample ratio.

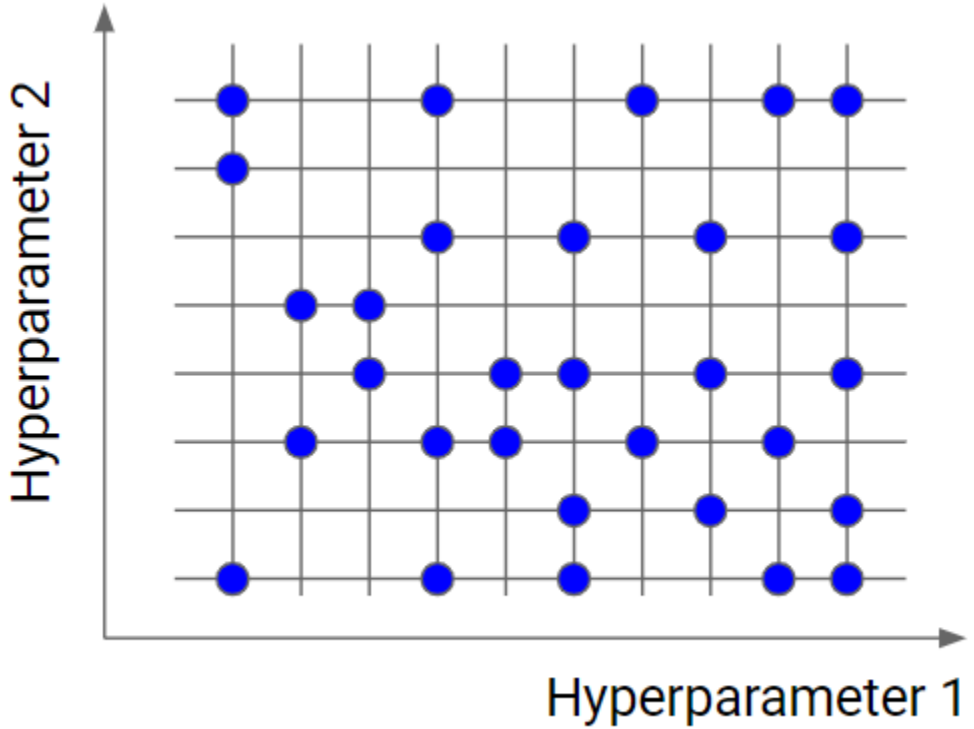
Grid Search

- **Grid search** involves exhaustively searching through a predefined set of hyperparameters.
- You define a grid of hyperparameters (e.g., a range of learning rates, number of layers, etc.) and train the model with each combination, then select the set of hyperparameters that yields the best performance (e.g., highest accuracy or lowest loss).



Random Search:

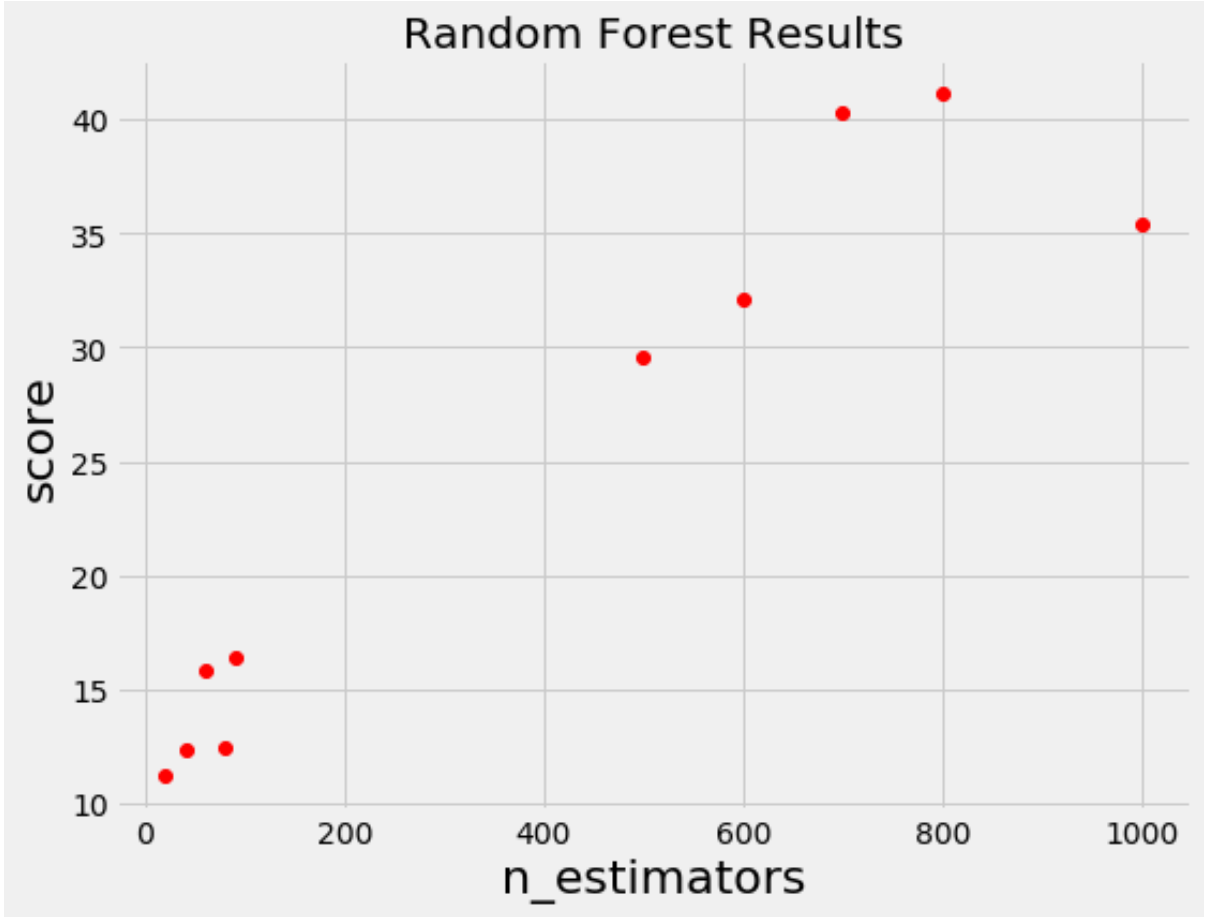
- **Random search** randomly samples from the hyperparameter space, rather than exhaustively searching through it.
- It can be more efficient than grid search when the hyperparameter space is large, as it doesn't try every possible combination but instead picks random sets of values to try.



Bayesian Optimization:

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

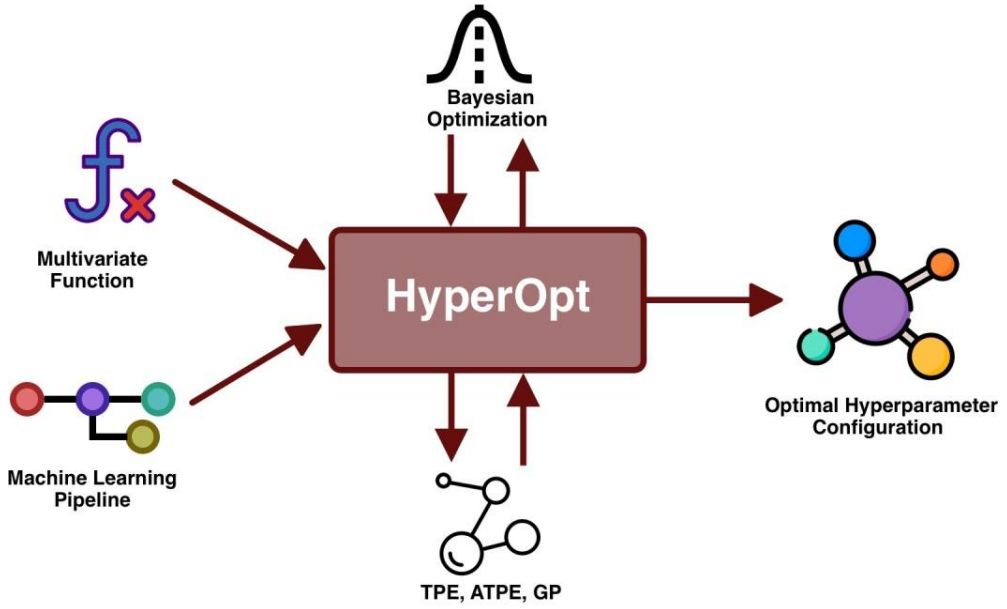
- **Bayesian optimization** is a probabilistic model-based approach to hyperparameter tuning. It builds a model of the objective function (e.g., model performance) and uses it to decide where to search next.
- It aims to explore the hyperparameter space more efficiently than grid or random search by focusing on regions of the space that are more likely to lead to better results.



Automated Hyperparameter Tuning (AutoML):

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

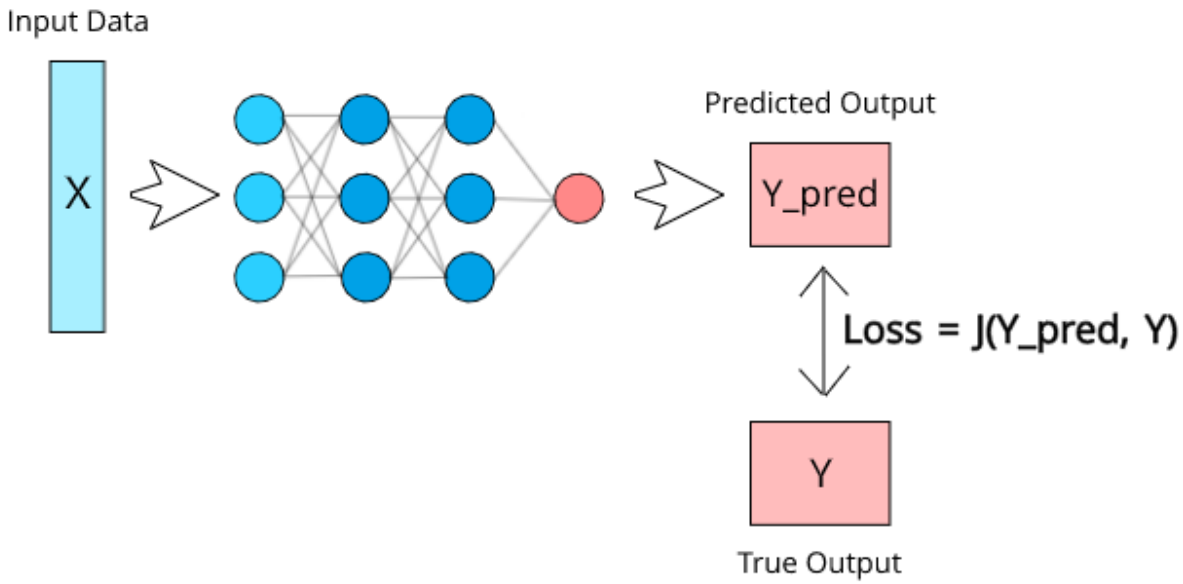
- AutoML systems, like TPOT or Auto-sklearn, automate the process of hyperparameter tuning by searching for the best hyperparameters, models, and feature transformations.
- These tools can save time by automating the trial-and-error process of model selection and hyperparameter optimization.



Contents

Loss Functions

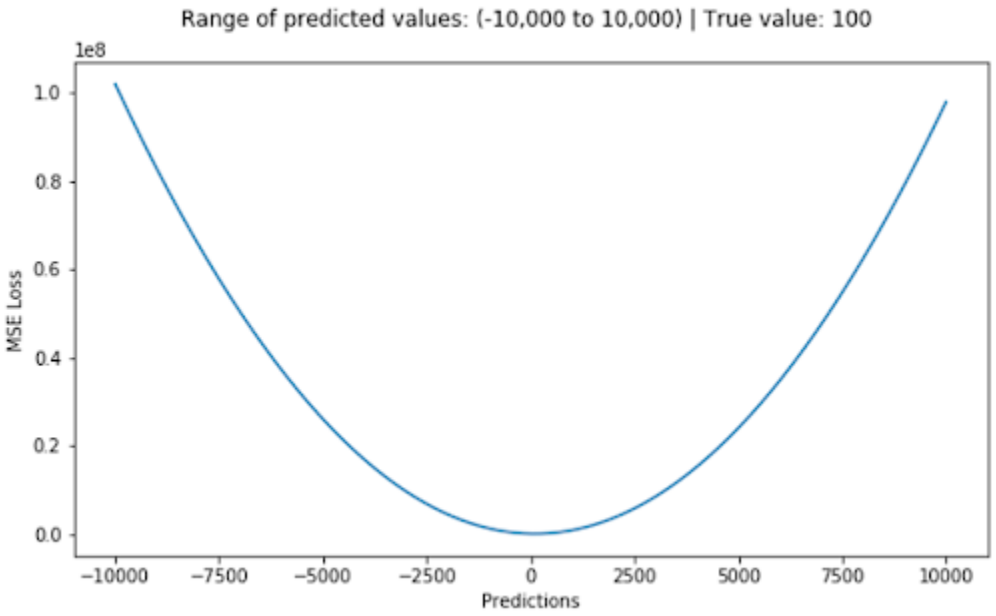
- In machine learning, loss functions (also known as cost functions) are used to measure how well a model's predictions match the actual data.
- The goal is to minimize the loss function during training, which allows the model to improve its predictions



Mean Squared Error

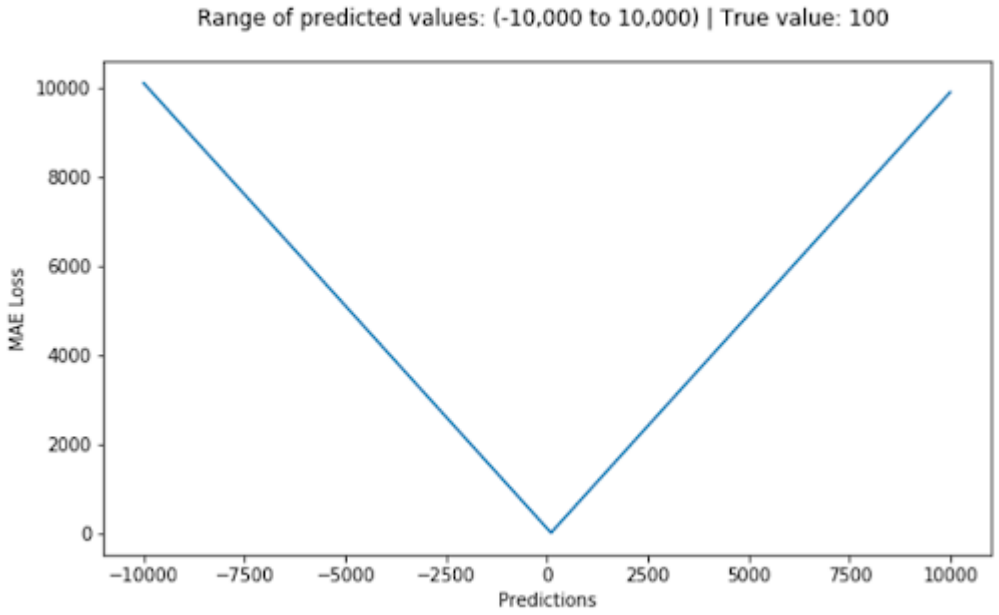
- The mean squared error (MSE) loss function, or L2 loss, as the average of squared differences between the actual value (Y) and the predicted value (Ŷ).
- It's the most commonly used regression loss function.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$$



Mean Absolute Error / L1 Loss

- the mean absolute error (MAE) loss function, or L1 loss, as the average of absolute differences between the actual and the predicted value.
- It's the second most commonly used regression loss function.
- It measures the average magnitude of errors in a set of predictions, without considering their directions.

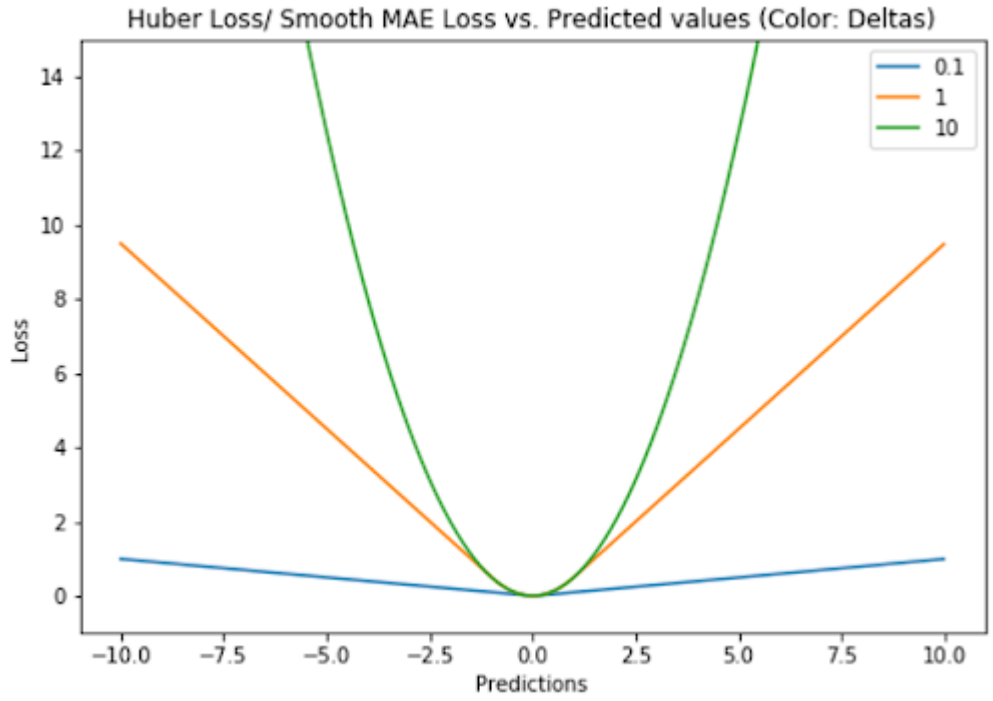


$$\text{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

Huber Loss / Smooth Mean Absolute Error

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

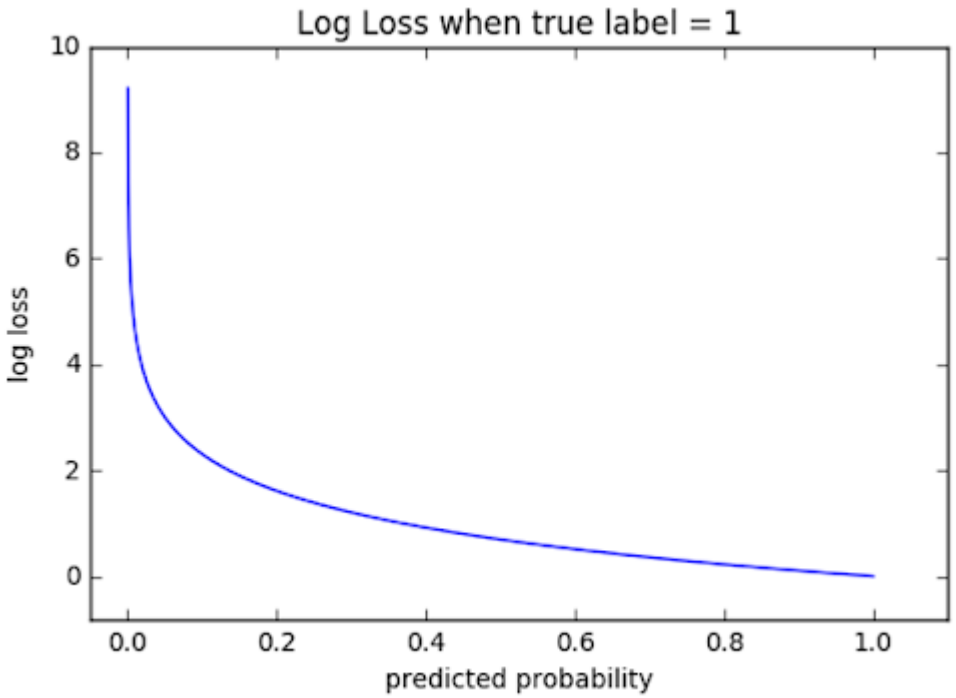
- The Huber loss function is defined as the combination of MSE and MAE loss functions because it approaches MSE when errors are small and MAE when errors are larger
- **Better than MSE**: Reduces the effect of outliers, which can dominate training if squared errors are used.
- **Better than MAE**: Provides smooth gradients for small errors, making optimization more stable.
- **Good balance**: Keeps sensitivity to small errors while ignoring extreme ones.



$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for} |y - f(x)| \le \delta, \\ \delta\,|y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases}$$

Binary Cross Entropy Loss

- This is the most common loss function use in classification problems.
- The cross-entropy loss decreases as the predicted probability converges to the actual label.
- It measures the performance of a classification model whose predicted output is a probability value between 0 and 1.

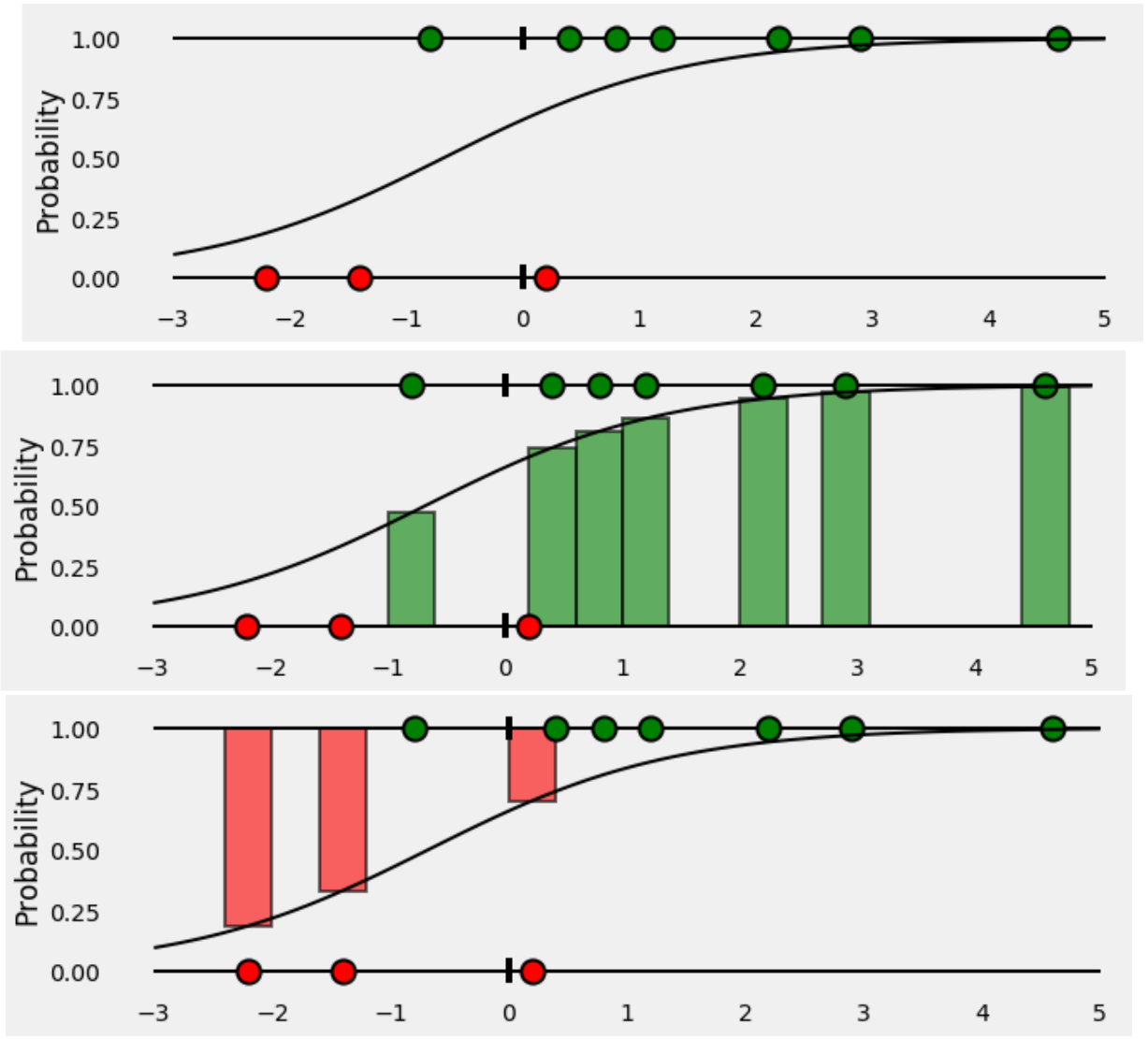


$$L = -\frac{1}{m}\sum_{i=1}^{m}(y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i))$$

Binary Cross Entropy Loss

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

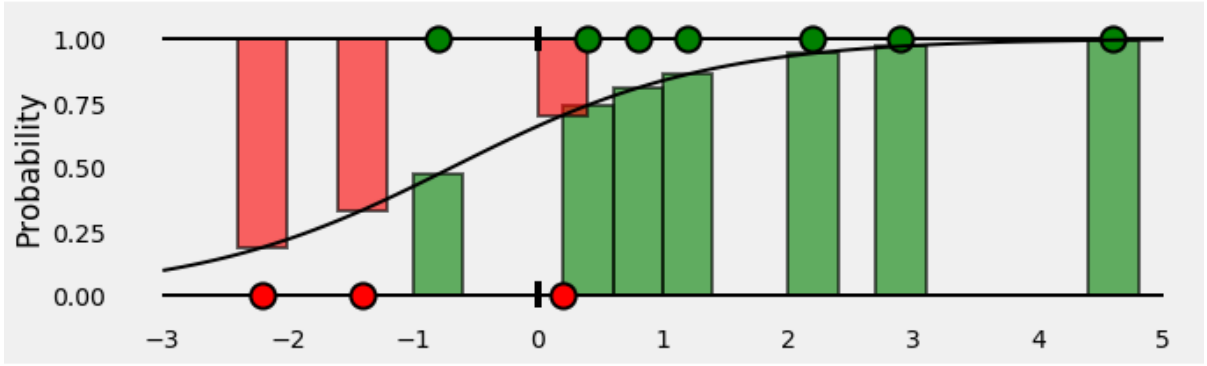
- The fitted regression is a sigmoid curve representing the probability of a point being green for any given x.
- for all points belonging to the positive class (green), what are the predicted probabilities given by our classifier
- what is the probability of a given point being red? The red bars ABOVE *the sigmoid curve*.

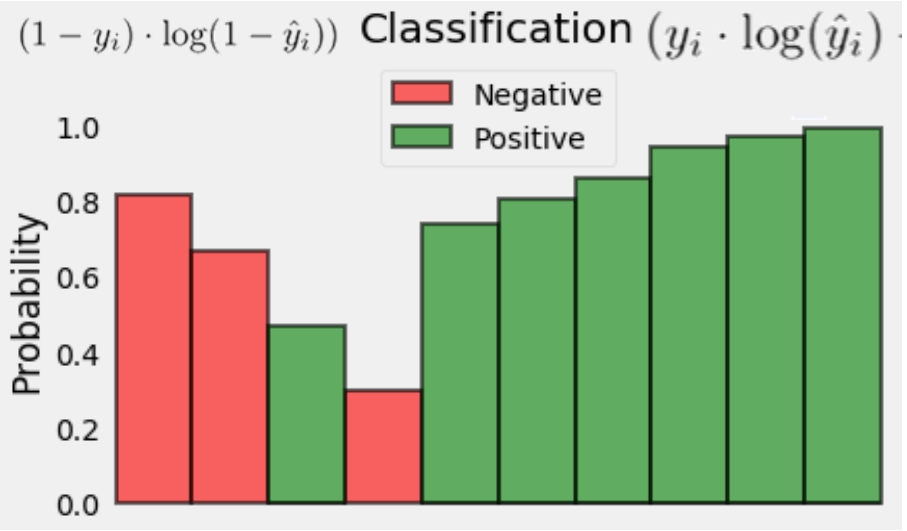


Binary Cross Entropy Loss

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

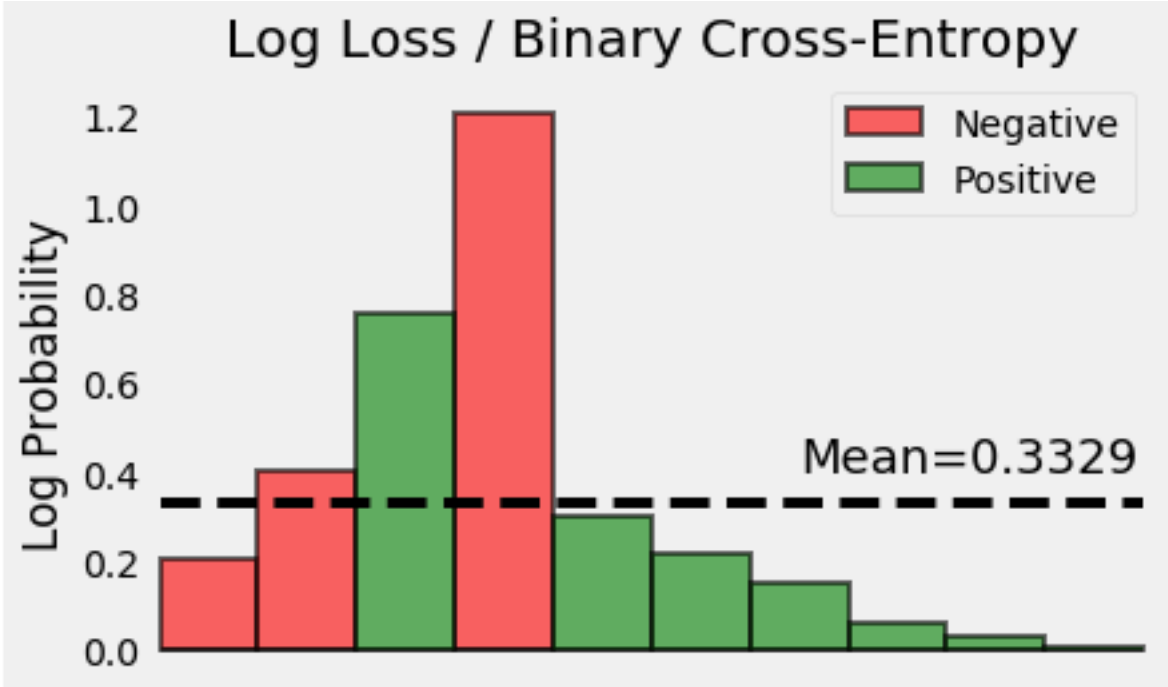
- The bars represent the predicted probabilities associated with the corresponding true class of each point
- Since we're trying to compute a loss, we need to penalize bad predictions.
- If the probability associated with the true class is 1.0, we need its loss to be zero.
- Conversely, if that probability is low, say, 0.01, we need its loss to be HUGE!





Binary Cross Entropy Loss

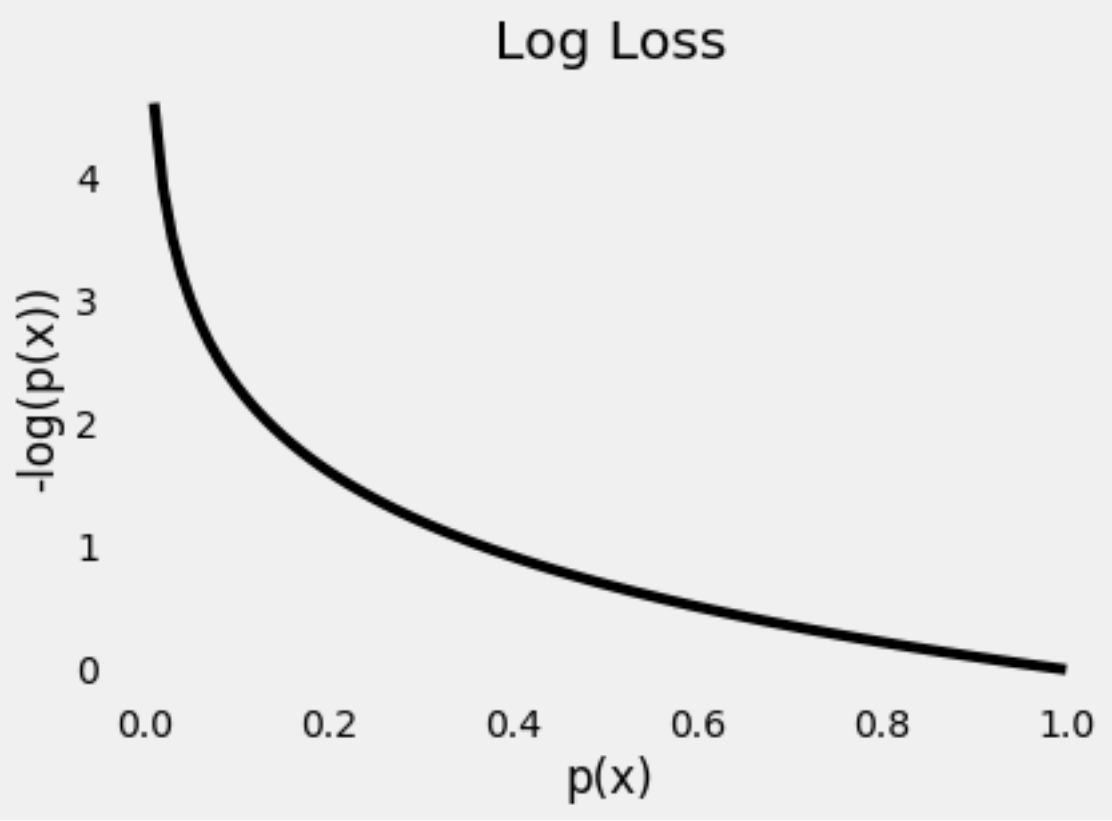
- Let's take the (negative) log of the probabilities — these are the corresponding losses of each and every point.
- Finally, we compute the mean of all these losses.



$$L = -\frac{1}{m}\sum_{i=1}^{m}(y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i))$$

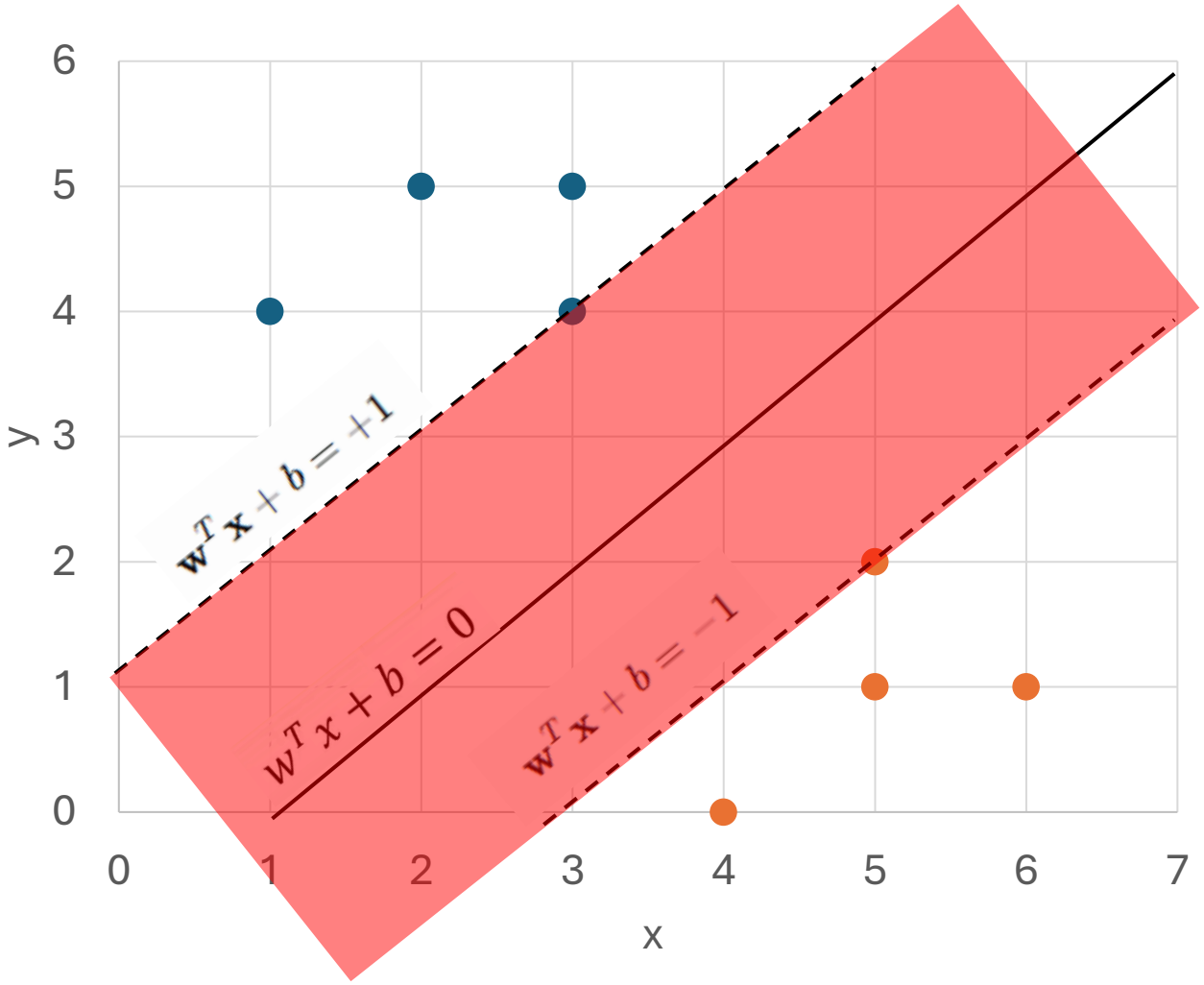
Binary Cross Entropy Loss

- taking the (negative) log of the probability suits us well enough for this purpose (*since the log of values between 0.0 and 1.0 is negative, we take the negative log to obtain a positive value for the loss*).
- The plot below gives us a clear picture —as the predicted probability of the true class gets closer to zero, the loss increases exponentially.



Hinge loss

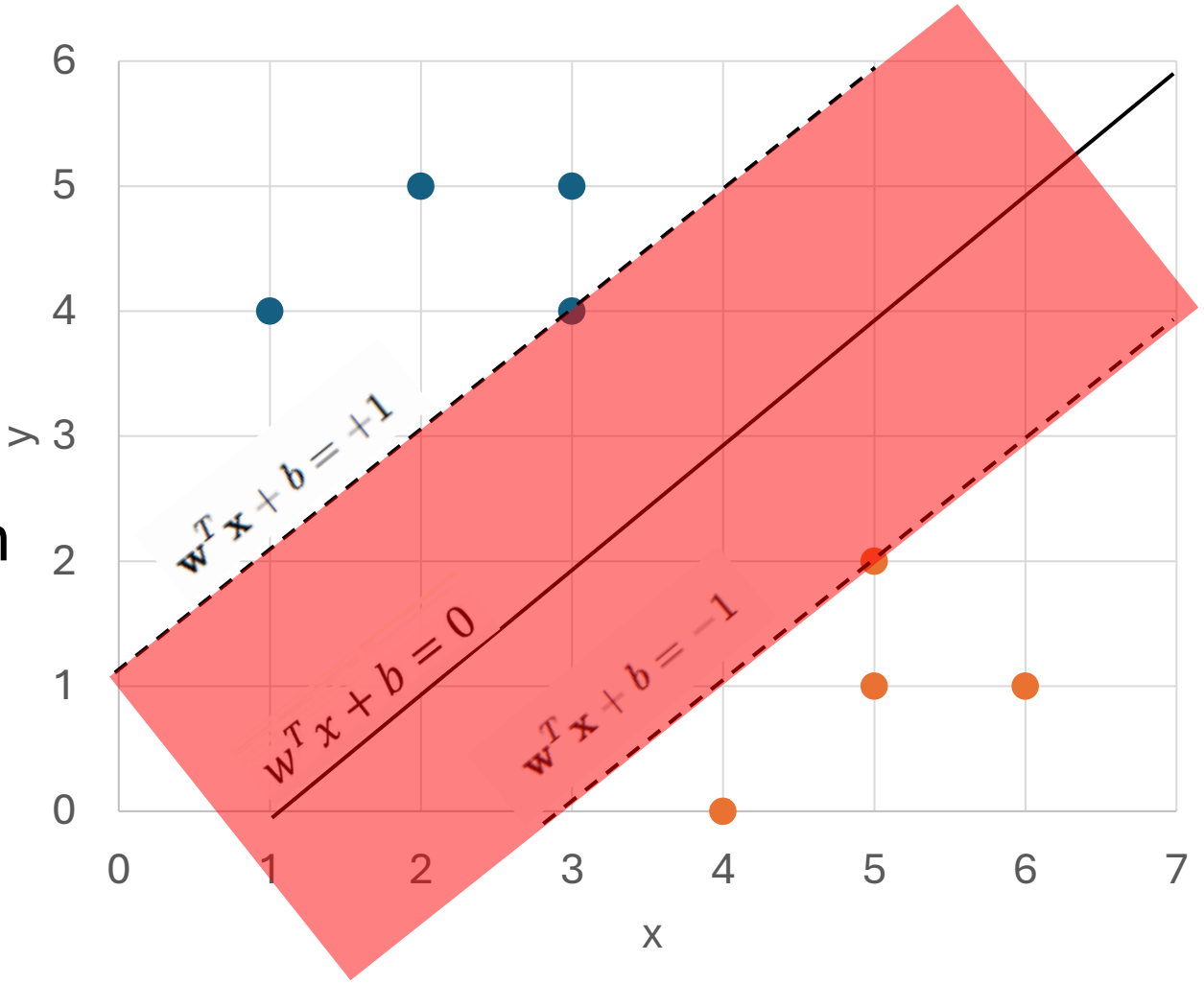
- Hinge loss is a loss function commonly used for training **Support Vector Machines (SVMs)** for classification tasks.
- It helps enforce a **margin** between different classes, encouraging the model to make confident predictions.



Hinge loss

- For a given training example (x,y) with:
- **Y ∈ {−1,+1}** (class labels in SVM are typically -1 and +1, not 0 and 1)
- **f(x)** being the model's prediction (e.g., the raw output of a linear classifier before applying a decision function),
- The hinge loss is defined as:

$$L(y, f(x)) = \max(0, 1 - y \cdot f(x))$$

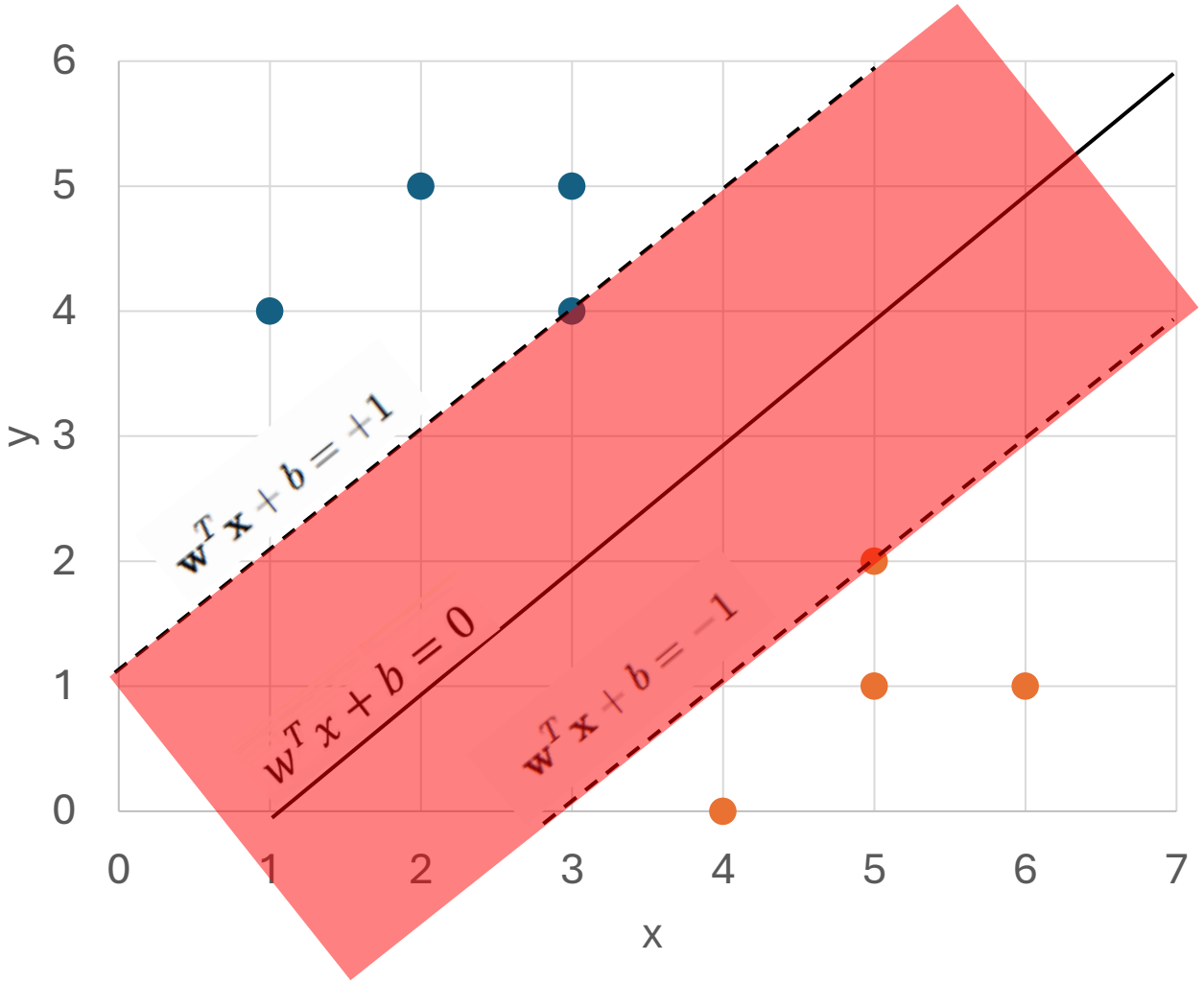


Hinge loss

- The hinge loss is defined as:

$$L(y, f(x)) = \max(0, 1 - y \cdot f(x))$$

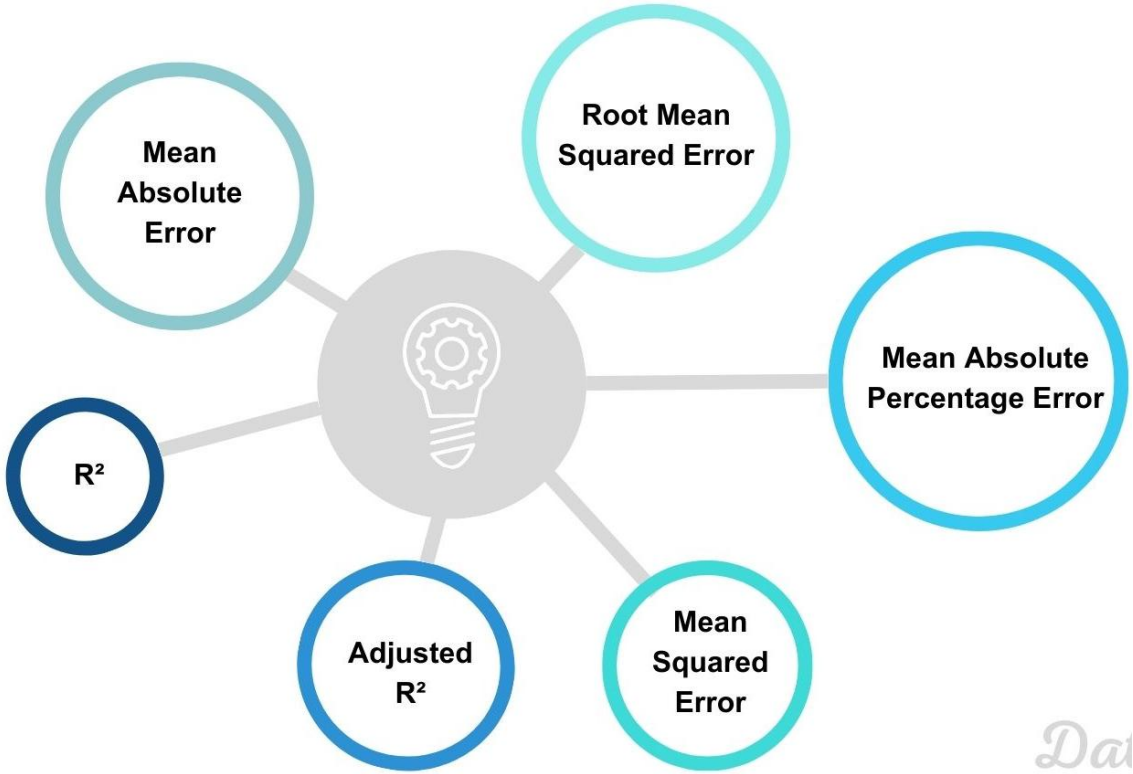
True Label	Predicted Output	Hinge Loss
+1	1.5	max(0,1−(1×1.5))= 0
+1	0.8	max(0,1-1 x 0.8)= 0.2
+1	-0.5	max(0,1−(1×-0.5)) = 1.5



Contents

Regression metrics

- Regression metrics are quantitative measures used to evaluate the performance of a regression model in predicting continuous values.
- They help determine how well the model's predictions match the actual values and whether the model is overfitting or underfitting the data.



Error-Based Metrics

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

Mean Absolute Error (MAE):

- Measures average absolute difference between predicted and actual values.
- Pros: Easy to interpret.
- Cons: Does not penalize large errors as much as squared errors.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Mean Squared Error (MSE)

Penalizes larger errors more due to squaring.

Pros: Useful for optimization (smooth gradient).

Cons: Sensitive to outliers.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Error-Based Metrics

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

Root Mean Squared Error (RMSE)

- Similar to MSE but in the same unit as the target variable.
- Pros: Easier to interpret than MSE.
- Cons: Still sensitive to outliers.

$$RMSE = \sqrt{MSE}$$

Mean Absolute Percentage Error (MAPE)

- Measures error as a percentage of actual values.
- Pros: Unit-free, easy to compare across datasets.
- Cons: Fails when yi=0, biased toward large values.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100$$

Goodness-of-Fit Metrics

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

These measure how well the model explains variability in the data

R^2, Coefficient of Determination)

- Measures proportion of variance explained by the model.
- Ranges from 0 to 1 (higher is better).
- Cons: Can be misleading if additional irrelevant features are added.

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

Adjusted R-Squared

- Adjusts for number of predictors p
- Number if observations (sample size) is n
- Better for multiple regression models.

$$R^2_{adj} = 1 - \left(\frac{(1 - R^2)(n - 1)}{n - p - 1}\right)$$

Akaike Information Criterion (AIC)

- **Akaike Information Criterion (AIC)** and **Bayesian Information Criterion (BIC)** are both **model selection criteria** used to compare different statistical models.
- Both metrics balance **goodness of fit** (how well the model fits the data) with **model complexity** (how many parameters the model has), preventing overfitting and underfitting

Akaike Information Criterion (AIC)

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

- The **AIC** evaluates a model's fit while penalizing for complexity (more parameters). It's useful when comparing different models on the same data.

Interpretation:

- **Lower AIC values** indicate a better model.
- The AIC penalizes models with more parameters, encouraging simpler models.
- AIC is useful when you're **comparing different models** on the same dataset.

$$AIC = 2k - 2\ln(L)$$

k is the number of model parameters (including the intercept).
L is the likelihood of the model (how well the model explains the data).

Bayesian Information Criterion (BIC)

- The **BIC** (also called the **Schwarz Information Criterion**) is similar to AIC but imposes a **stronger penalty** for having more parameters. It is derived from a Bayesian perspective and considers the sample size in the penalty term.

$$BIC = k \ln(n) - 2 \ln(L)$$

Interpretation:

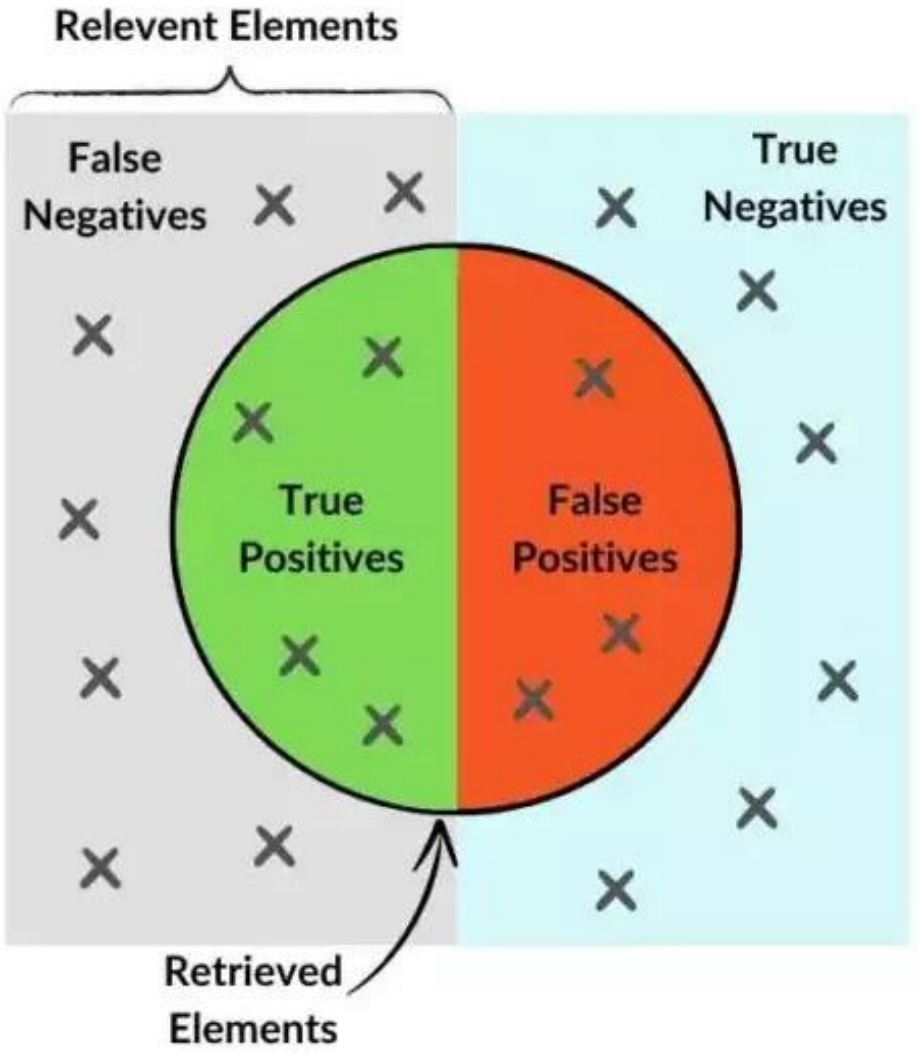
- **Lower BIC values** indicate a better model.
- The BIC generally imposes a **stronger penalty for complexity** compared to AIC, particularly as the sample size n grows.
- BIC is often preferred when you have **larger datasets**, as the penalty for adding more parameters grows with the sample size.

k is the number of model parameters (including the intercept).
n is the number of data points.
L is the likelihood of the model (how well the model explains the data).

Contents

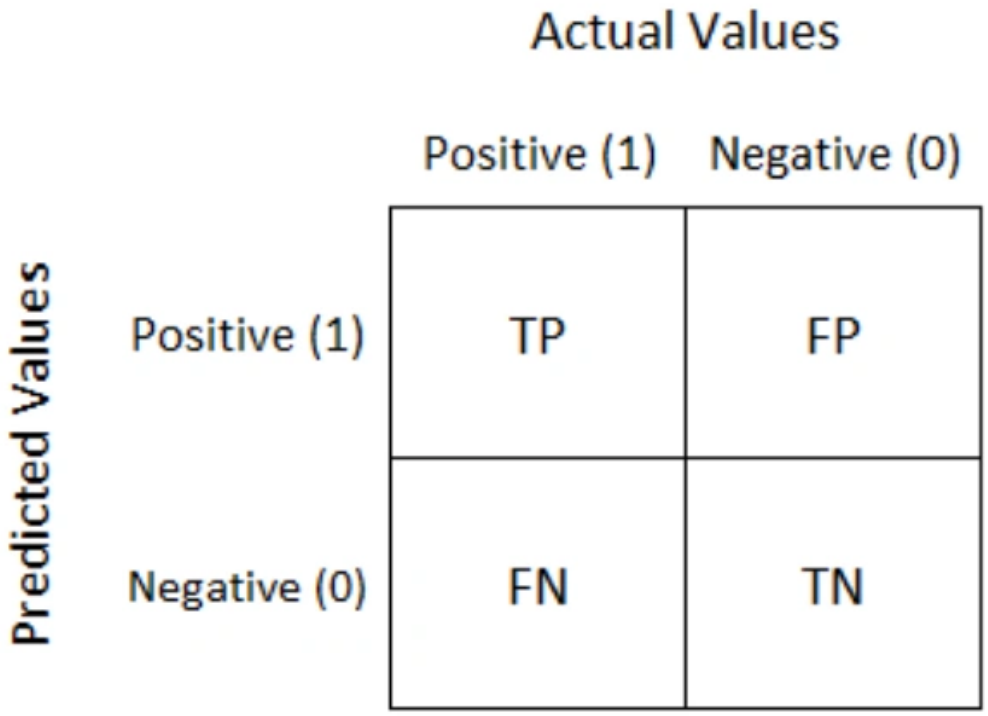
Classification Metrics

- When building a classification system, we need a way to evaluate the performance of the classifier. And we want to have evaluation metrics that are reflective of the classifier's true performance.



Confusion Matrix

- The **confusion matrix** is a table that allows you to visualize the performance of a classification model by displaying the number of **true positives (TP)**, **true negatives (TN)**, **false positives (FP)**, and **false negatives (FN)**.



Accuracy

- Measures the overall correctness of the model by calculating the ratio of correct predictions to total predictions.
- **Limitations**: Can be misleading if the dataset is imbalanced (e.g., in a dataset with 95% negatives and 5% positives, predicting only negatives could still yield high accuracy).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

TP = True Positives

TN = True Negatives

FP = False Positives

FN = False Negatives

Precision

- Measures how many of the predicted positives are actually positive.
- High precision means few false positives.
- Useful when the cost of false positives is high (e.g., in spam email detection, where wrongly classifying a legitimate email as spam is costly).

$$Precision = \frac{TP}{TP + FP}$$

TP = True Positives

TN = True Negatives

FP = False Positives

FN = False Negatives

Recall

- Measures how many of the actual positives are correctly identified by the model.
- High recall means fewer false negatives.
- Important when the cost of **false negatives** is high (e.g., in medical diagnostics, where missing a positive diagnosis could be very harmful).

$$Recall = \frac{TP}{TP + FN}$$

TP = True Positives

TN = True Negatives

FP = False Positives

FN = False Negatives

F1-Score

- **Harmonic mean** of precision and recall, providing a balance between the two.
- Useful when you need to balance precision and recall.
- It is especially important when you have an imbalanced dataset (i.e., one class is much larger than the other).

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Specificity (True Negative Rate)

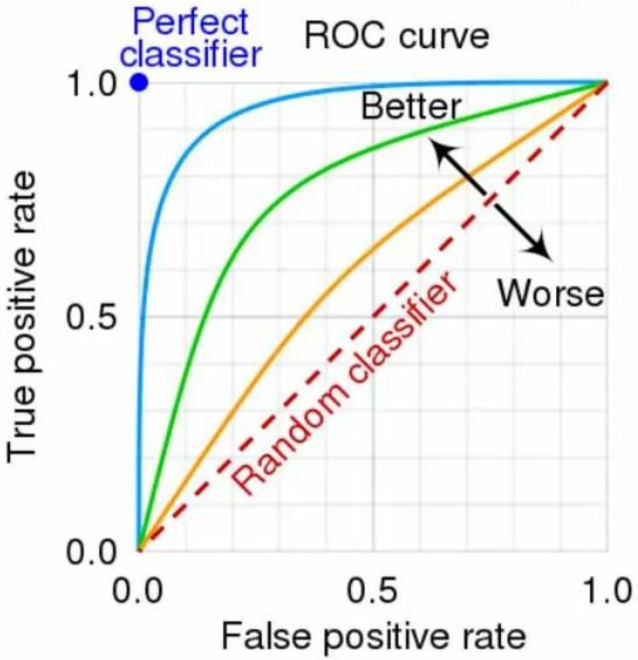
- Measures how many of the actual negatives are correctly identified by the model.
- Useful when you want to minimize false positives in cases where negative class predictions are important.

$$Specificity = \frac{TN}{TN + FP}$$

ROC Curve and AUC (Area Under the Curve)

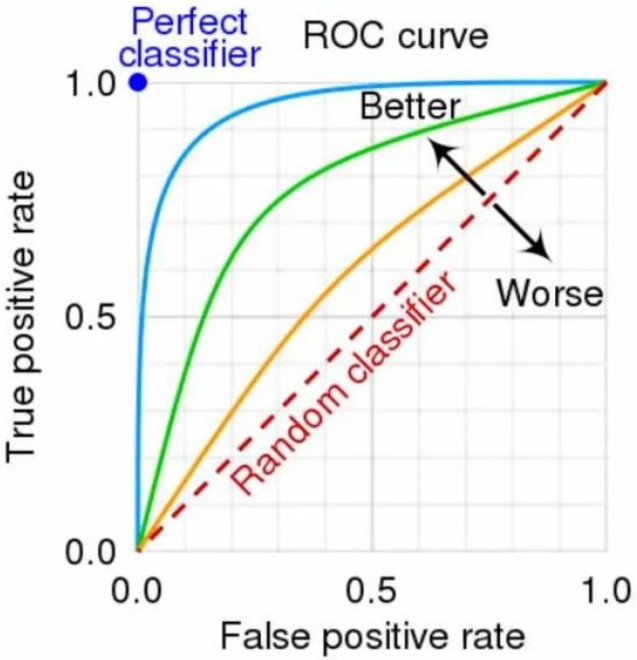
McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

- ROC and AUC are especially useful for evaluating **binary classification** problems and comparing the performance of different models.
- ROC Curve (Receiver Operating Characteristic Curve) plots the True Positive Rate (Recall) against the False Positive Rate (1 - Specificity) at various thresholds.



ROC Curve and AUC (Area Under the Curve)

- AUC is the area under the ROC curve, and it represents the model's ability to discriminate between the positive and negative classes.
  - AUC ranges from 0 to 1:
  - 1 means perfect prediction.
  - 0.5 means random prediction (no better than random guessing).
  - Closer to 1 = better model performance.

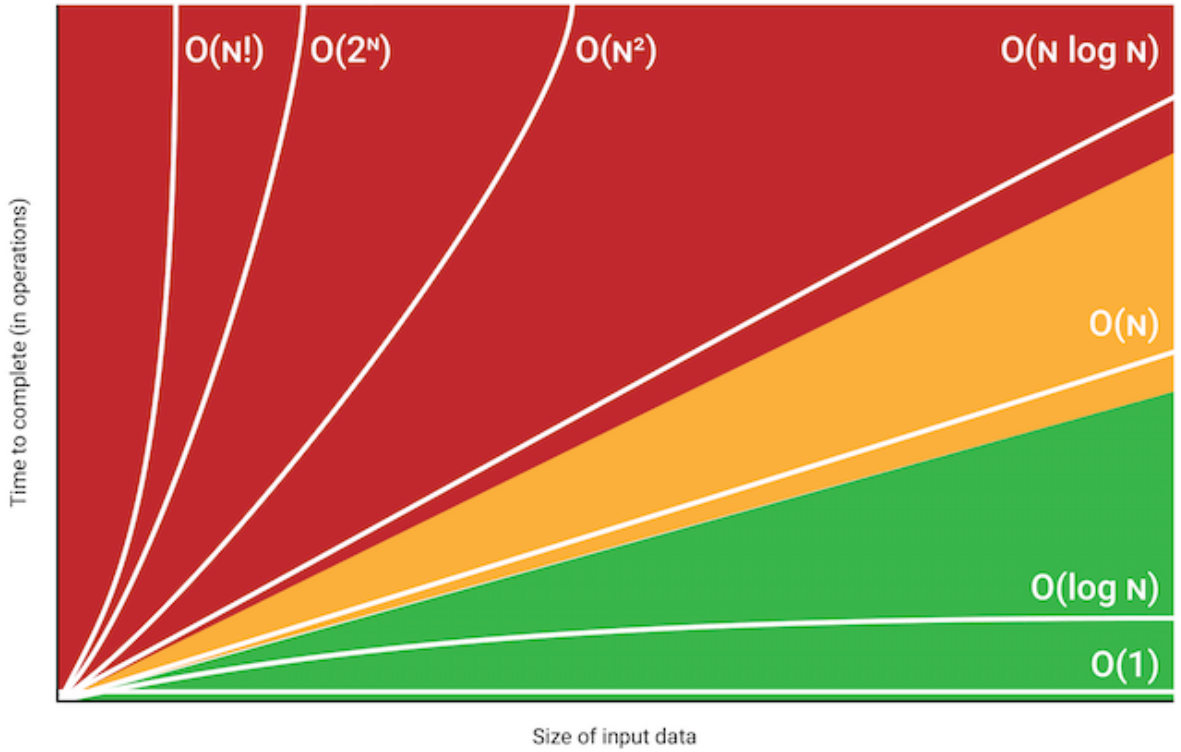


Contents

Big O notation

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

- **Big O notation** is a way of analyzing how the runtime (or space complexity) of an algorithm grows with the size of the input.
- In computation, big O notation is used to classify algorithms based on how their execution time or space requirements grow as the size of the input increases.
- In machine learning, Big O notation helps us understand the efficiency of training algorithms, inference processes, and optimization methods.



Linear Regression (Normal Equation)

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

- The Normal Equation :

Where: $$\theta = (X^T X)^{-1} X^T y$$

- X is the **m×n** design matrix (m samples, n features).
- y is the **×1** target vector.
- θ is the **×1** weight vector.
- X^TX is an **n×n** matrix.

Computing X^TX(Matrix Multiplication):

- X^TX has dimensions n×m, and X has dimensions m×n
- Multiplication results in an n×n matrix.
- Complexity: **O(mn2)**.

Computing (XTX)−1 (Matrix Inversion):

- Inverting an n×n matrix takes **O(n3)**
- using standard algorithms like Gaussian elimination or LU decomposition.

Computing (XTX)−1

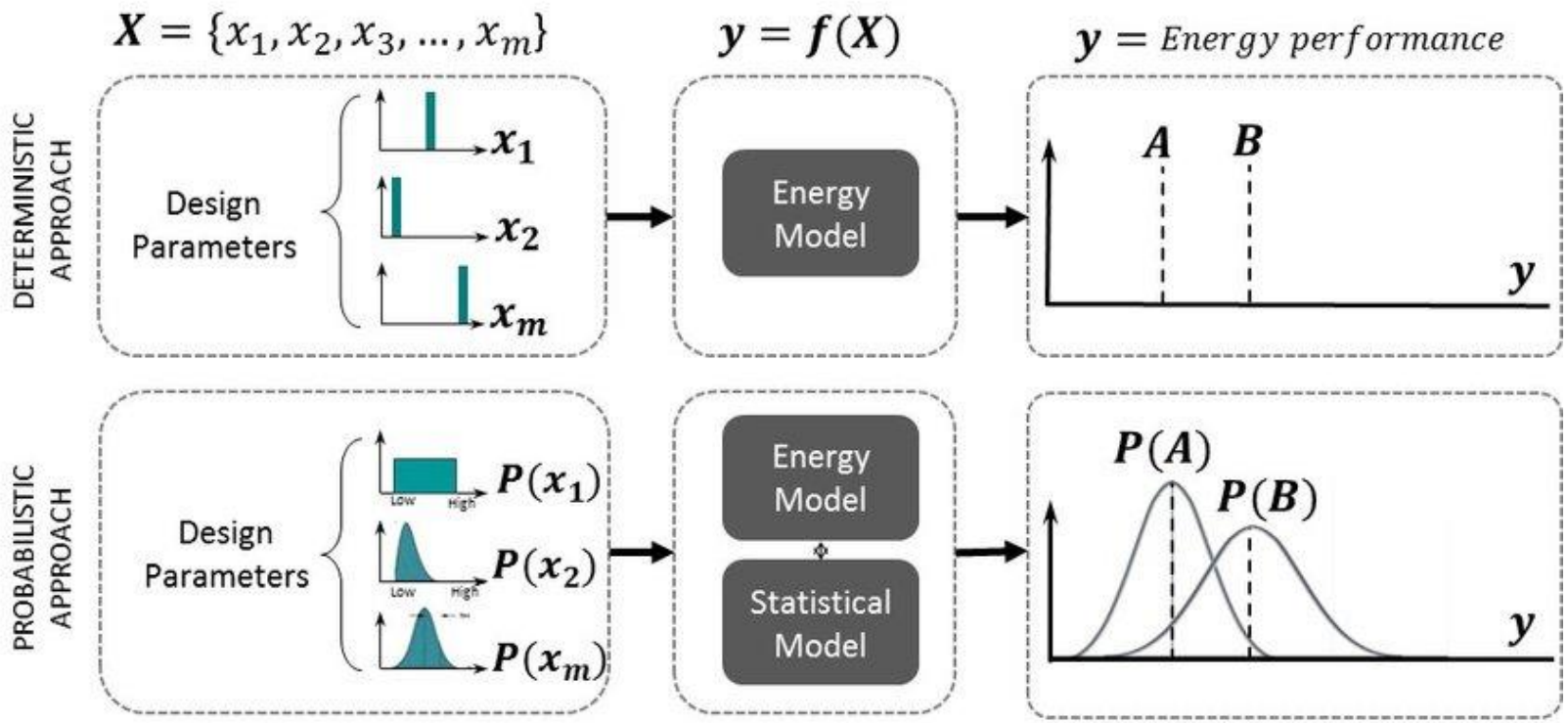
- (XTX)−1 is n×n, and Xty is n×1
- The multiplication takes **O(n2)**.

The dominant term is **matrix inversion O(n3)**.

Big O notation

Approach	Time Complexity	Explanation
Linear Regression (Normal Equation)	$O(n^3)$	Matrix inversion is expensive for large datasets.
Linear Regression (Gradient Descent)	O(n) per iteration	Linear in dataset size but requires multiple iterations.
K-Means Clustering	O(nkdt)	k clusters, d dimensions, t iterations.
K-Nearest Neighbors (Brute-Force)	$O(n^2)$	Computes distance between all pairs.
Decision Trees	O(nlogn)	Sorting is needed for split finding.
Random Forest	O(mnlogn)	m trees withO(nlogn) complexity each.
Support Vector Machines (SVM)	$O(n^2)$ or $O(n^3)$	Depends on kernel choice; quadratic for linear, cubic for RBF kernels.
Neural Networks (Forward + Backpropagation)	O(nd) per epoch	n data points, d weights in the model.

Contents



10.1017/S0890060415000426

Probabilistic Machine Learning

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

- A probabilistic ML model incorporates uncertainty and randomness, often expressing outputs as probability distributions rather than fixed values.

Characteristics:

- Outputs are associated with probabilities rather than single values.
- Can handle uncertainty and missing information better.
- Often uses Bayesian methods or probabilistic graphical models.

Deterministic Machine Learning

- A deterministic ML model produces the same output for a given input every time it is run. It follows a fixed set of rules or mathematical functions without incorporating randomness in its predictions.

Characteristics:

- Predictable and repeatable results.
- No randomness in model inference.
- Typically uses fixed equations or mappings from input to output

Deterministic VS Probabilistic

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

Feature	Deterministic ML	Probabilistic ML
Output Type	Fixed, deterministic values	Probability distributions
Handling Uncertainty	Poor	Strong
Example Algorithms	SVM, Deterministic Trees, Linear Regression	Bayesian Methods, Logisitc regression Random Forest
Interpretability	Usually higher	Can be more complex