Machine Learning Main Techniques, Evaluation Metrics and Plots

By: Seyed Muhammad Hossein Mousavi
mosavi.a.i.buali@gmail.com
Lugano, Switzerland
August 2024

Outline:

- **Pre-Processing**
  - Parsing
  - Data Interpolation/Resizing
  - Denoising
  - NaNs
  - Normalization
  - Standardization

- **Mid-Level Processing**
  - Data Augmentation
  - Feature Extraction
  - Feature Selection
  - Dimensionality Reduction
  - Multi-Modal Fusion

- **Data Understanding**
  - Feature Distribution Over Classes
  - Feature Correlation
  - Feature Correlation by Regression
  - Feature Importance
  - SHAP
  - LIME
  - Class Distribution Ratio
  - PCA and t-SNE Feature Plots
  - The Baseline and The Ground Truth

- **Metrics**
  - TP, FN, FP, FN
  - Classification Report
  - Confusion Matrix
  - ROC-AUC
  - Precision-Recall AUC
  - Log Loss / Cross-Entropy Loss
  - Cohen's Kappa
  - MCC
  - Precision
  - Recall (sensitivity)
  - Unweighted Average Recall
  - Specificity
  - F1-Score
  - Accuracy
  - Balanced Accuracy
  - Hamming Loss
  - Jaccard Index

- **Evaluation Plots**
  - Violin Plot
  - Test Accuracy Plot
  - Cumulative Gain Chart
  - Prediction Probabilities Histogram
  - Box Plot

- **Classifiers**
  - Naïve Bayes (NB)
  - K-Nearest Neighborhood (KNN)
  - Support Vector Machine (SVM)
  - Logistic Regression (LR)
  - Decision Tree (DT)
  - Gradient Boosting
  - eXtreme Gradient Boosting (XGBoost)
  - Random Forest (RF)
  - AdaBoost

- **Advanced Techniques and Analysis**
  - Leave One Participant Out CV (LOPO)
  - Leave-One-Out Cross-Validation (LOO)
  - K-Fold Cross-Validation
  - Stratified K-Fold Cross-Validation
  - Repeated K-Fold Cross-Validation
  - Nested Cross-Validation
  - Bootstrapping

Pre-Processing

- **Parsing**
  - ➢ Parsing is the process of **analyzing and breaking down input data into its component parts** to understand its structure.
  - ➢ The parsing involves reading BVH files, which contain hierarchical structure and motion data.
  - ➢ This data is converted into a structured format like arrays or objects.
  - ➢ Frame interpolation is then used to generate additional frames between existing ones to create smoother transitions and standardize the frame rate across the dataset.

- **Data Interpolation**
  - ➢ Interpolation is the process of **estimating unknown values between known data points**.
  - ➢ Frame interpolation is then used to generate additional frames between existing ones to create smoother transitions and standardize the frame rate across the dataset.

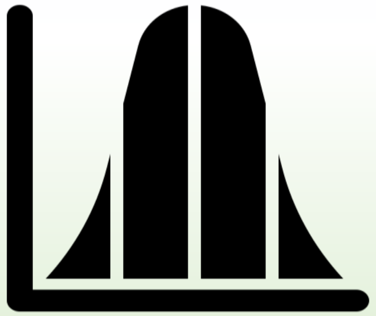
- **Denoising**
  - ➢ Denoising is the process of removing noise or unwanted distortions from data to enhance the clarity or quality of the underlying signal
  - ➢ Denoising refers to the process of reducing or removing irregularities or noise in the motion capture data, such as jittery or unnatural movements, to produce smoother and more realistic motion sequences like Gaussian denoising.
  - ➢ Wavelet denoising uses wavelet transforms to decompose the motion data into different frequency components.
    - ✓ By selectively removing or attenuating the high-frequency noise while preserving the important motion details (low-frequency components), it can smooth out the motion, reducing jitter and other unwanted artifacts while keeping the essential movement intact.
    - ✓ This technique is particularly effective for dealing with time-series data like motion capture in BVH files.

Pre-Processing

- **NaNs**
  - Handling NaNs (Not a Number) addresses missing data points that could disrupt motion capture continuity. Techniques like **removing NaNs, imputing values, or using interpolation** methods are employed to maintain the integrity of the motion data.

- **Normalization**
  - Data normalization is the process of scaling data to a standard range, typically between 0 and 1 or -1 and 1, to ensure that different **features contribute equally to a model**.
  - This helps improve the performance and **convergence speed of machine learning algorithms** by preventing features with larger ranges from dominating those with smaller ranges.

- **Standardization**
  - Data standardization is the process of rescaling data so that it has a **mean of 0 and a standard deviation of 1**.
  - Unlike normalization, which scales data to a fixed range, standardization transforms the data to have unit variance and is commonly used when the distribution of the data is Gaussian (normal) but with different scales for different features.
  - It is essential when working with BVH files from various sources that might have **different units of measurement or scales**.


Normalization

Mid-Level Processing

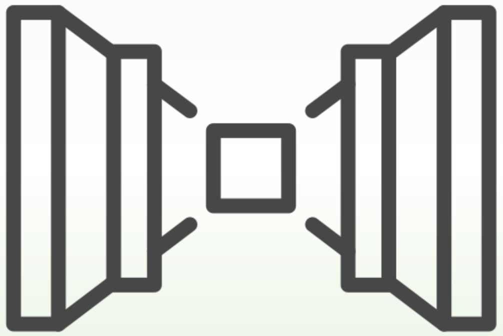
- **Data Augmentation**
  - ➢ Data augmentation is the process of **artificially increasing the size of a dataset** by creating modified versions of the original data.
  - ➢ This can be done through techniques such as **flipping, rotating, cropping, or adding noise** to data or through **generating synthetic data** in text or numerical datasets.
  - ➢ It helps improve model performance and generalization by introducing variability and **preventing overfitting**, especially in cases where the original dataset is limited.

- **Feature Extraction**
  - ➢ Feature extraction, in general, refers to the process of **transforming raw data into a set of representative attributes (features)** that are most relevant to the task at hand, such as classification or prediction.
  - ➢ The goal is to **reduce the dimensionality of the data while preserving its most important patterns or characteristics**. This process makes it easier for **machine learning models to understand and process the data**.
  - ➢ In body motion, it could be:
    - ✓ Rotation, average rotation, speed, acceleration, angular velocity, jerk, range of motion, spatial path, harmonics, and frequency analysis.
  - ➢ In image, it could be:
    - ✓ LBP, HOG, LPQ, Gabor Filters, SIFT, SURF, Wavelet feature, BRIEF and more.

- **Feature Selection**
  - ➢ Feature selection is the process of identifying and selecting the **most important and relevant features** from a dataset to improve model performance, **reduce complexity, and avoid overfitting**.
  - ➢ It aims to retain only those **features that contribute significantly** to the task while **discarding irrelevant or redundant data**. Such as:
    - ✓ Chi test, Mutual Information, ANOVA, Lasso, RFE, and more.

Mid-Level Processing

- **Dimensionality Reduction**
  - Dimensionality reduction is the process of **reducing the number of input variables (features)** in a dataset while preserving as much important information as possible.
  - This helps **simplify the data**, improve model performance, and **reduce computational cost,** especially in high-dimensional datasets.
  - Feature selection **chooses a subset of the original features** based on their relevance, **keeping the features as they are** (Chi test, Mutual Information, ANOVA, Lasso, RFE).
  - Dimensionality reduction **transforms the data into a lower-dimensional space** by **creating new features** that are combinations of the original ones, such as:
    - ✓ PCA, t-SNE, NCA, Autoencoders, LDA, NMF, and more.


Autoencoders

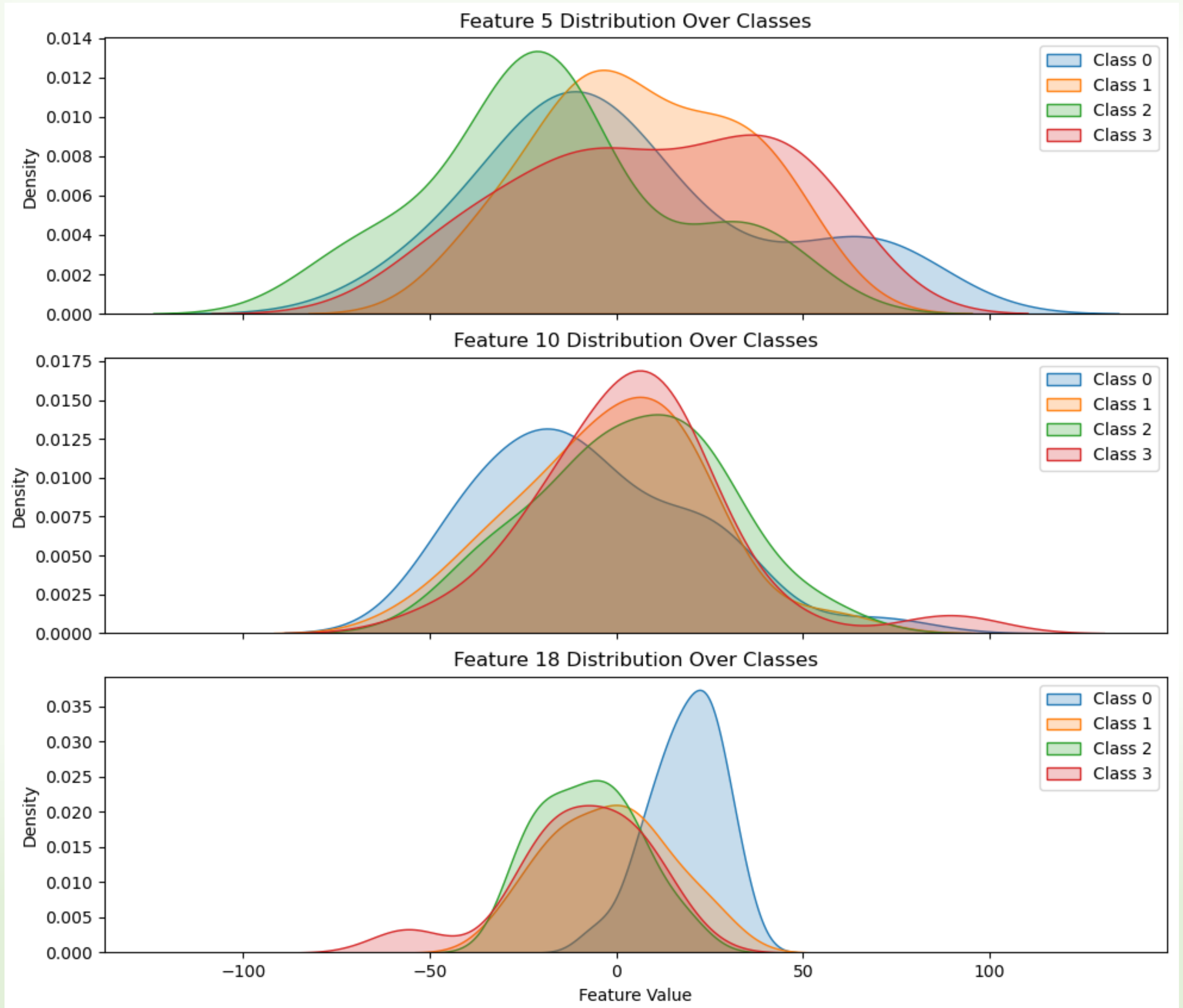
Mid-Level Processing

- **Multi-Modal Fusion**
  - ➢ Multimodal fusion refers to the process of combining data from multiple modalities (e.g., text, body tracking, images, audio, video, or sensor data) to **improve decision-making or prediction in machine learning models**.
  - ➢ By integrating information from different sources, multimodal fusion aims to **capture complementary information**, **providing a more comprehensive understanding and improving model accuracy**.
  - ➢ Some methods are:
    - ✓ **Early Fusion (Feature-Level Fusion) - Concatenation**: Combine feature vectors from different modalities **into a single feature vector** before feeding them into a machine learning model.
    - ✓ **Early Fusion (Feature-Level Fusion) - Principal Component Analysis (PCA):** Used **to reduce the dimensionality of concatenated features** from multiple modalities.
    - ✓ **Early Fusion (Feature-Level Fusion) - Canonical Correlation Analysis (CCA):** Identifies and fuses **correlated features across modalities.**
    - ✓ **Late Fusion (Decision-Level Fusion) - Weighted Averaging:** Combine predictions from different modality-specific models by **assigning weights to each modality's** output.
    - ✓ Weighting features **refers to assigning different levels of importance to features in a model.**
    - ✓ **Hybrid Fusion - Multimodal Deep Learning:** Use deep learning networks (e.g., CNNs, LSTMs) for each modality and **then fuse the learned representations** at different levels (either feature-level or decision-level).
    - ✓ **Hybrid Fusion - Attention Mechanism:** Use **attention layers to dynamically weigh contributions from different modalities**.
    - ✓ **Multimodal Autoencoders:** These **learn to compress multiple modalities into a shared latent space** and can reconstruct the input, making them effective in denoising or missing modality tasks.

Multi-Modal Fusion

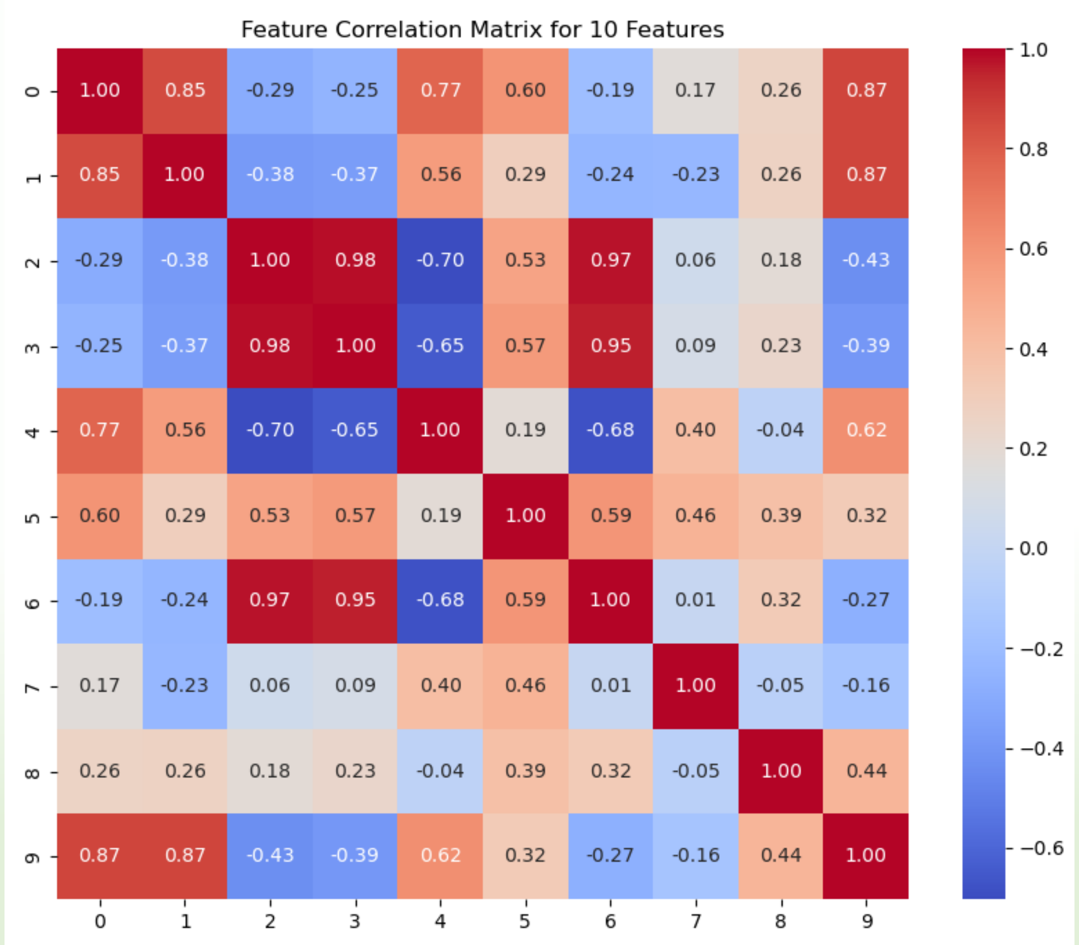
Data Understanding

- **Feature Distribution Over Classes**
  - ➢ Feature distribution over classes shows how a feature’s values are spread across different class labels in a classification problem.
  - ➢ It helps determine how well a feature can separate or distinguish between the classes, influencing its usefulness in the model.



Data Understanding

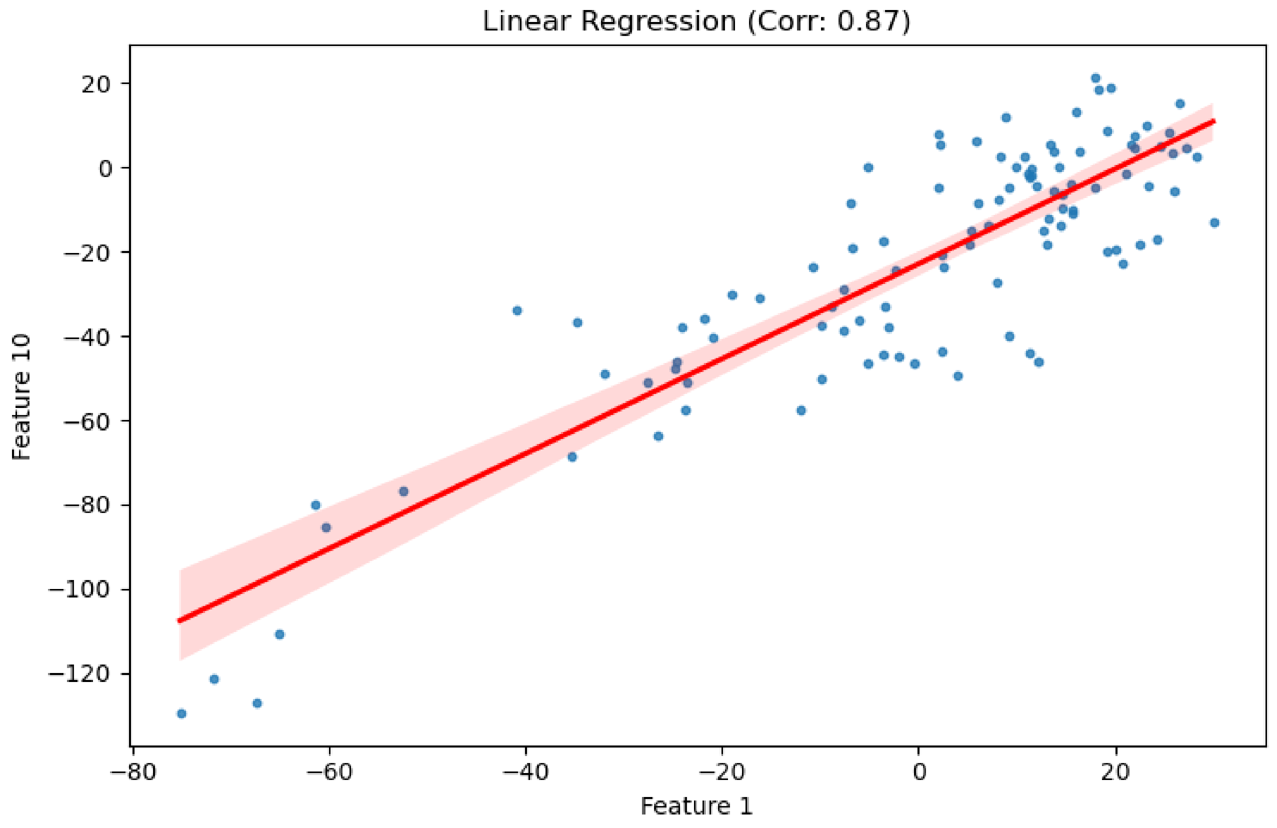
- **Feature Correlation**
  - Feature correlation measures the relationship between two features, **indicating how changes in one affect the other**.
  - A correlation coefficient (e.g., Pearson's) ranges from -1 to 1, with +1 meaning perfect positive correlation, 0 meaning no correlation, and -1 meaning perfect negative correlation. A high correlation between features can **lead to redundancy** and affect model performance. Reducing correlated features can improve model efficiency and accuracy.
  - **Feature redundancy occurs when two or more features are highly correlated, providing similar information. It is generally undesirable** as it can lead to overfitting, increase computational complexity, and reduce interpretability.

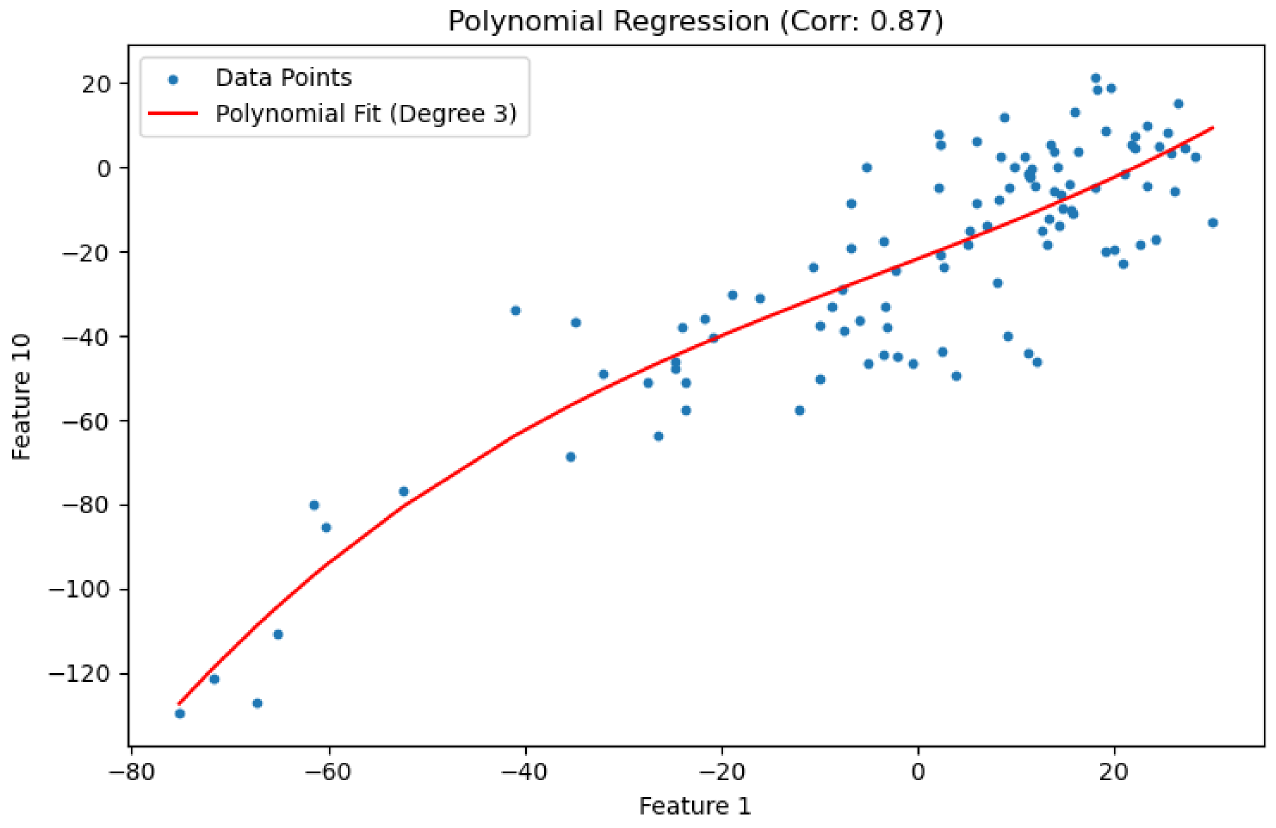


	0	1	2	3	4	5	6	7	8	9
0	1.00	0.85	-0.29	-0.25	0.77	0.60	-0.19	0.17	0.26	0.87
1	0.85	1.00	-0.38	-0.37	0.56	0.29	-0.24	-0.23	0.26	0.87
2	-0.29	-0.38	1.00	0.98	-0.70	0.53	0.97	0.06	0.18	-0.43
3	-0.25	-0.37	0.98	1.00	-0.65	0.57	0.95	0.09	0.23	-0.39
4	0.77	0.56	-0.70	-0.65	1.00	0.19	-0.68	0.40	-0.04	0.62
5	0.60	0.29	0.53	0.57	0.19	1.00	0.59	0.46	0.39	0.32
6	-0.19	-0.24	0.97	0.95	-0.68	0.59	1.00	0.01	0.32	-0.27
7	0.17	-0.23	0.06	0.09	0.40	0.46	0.01	1.00	-0.05	-0.16
8	0.26	0.26	0.18	0.23	-0.04	0.39	0.32	-0.05	1.00	0.44
9	0.87	0.87	-0.43	-0.39	0.62	0.32	-0.27	-0.16	0.44	1.00

Data Understanding

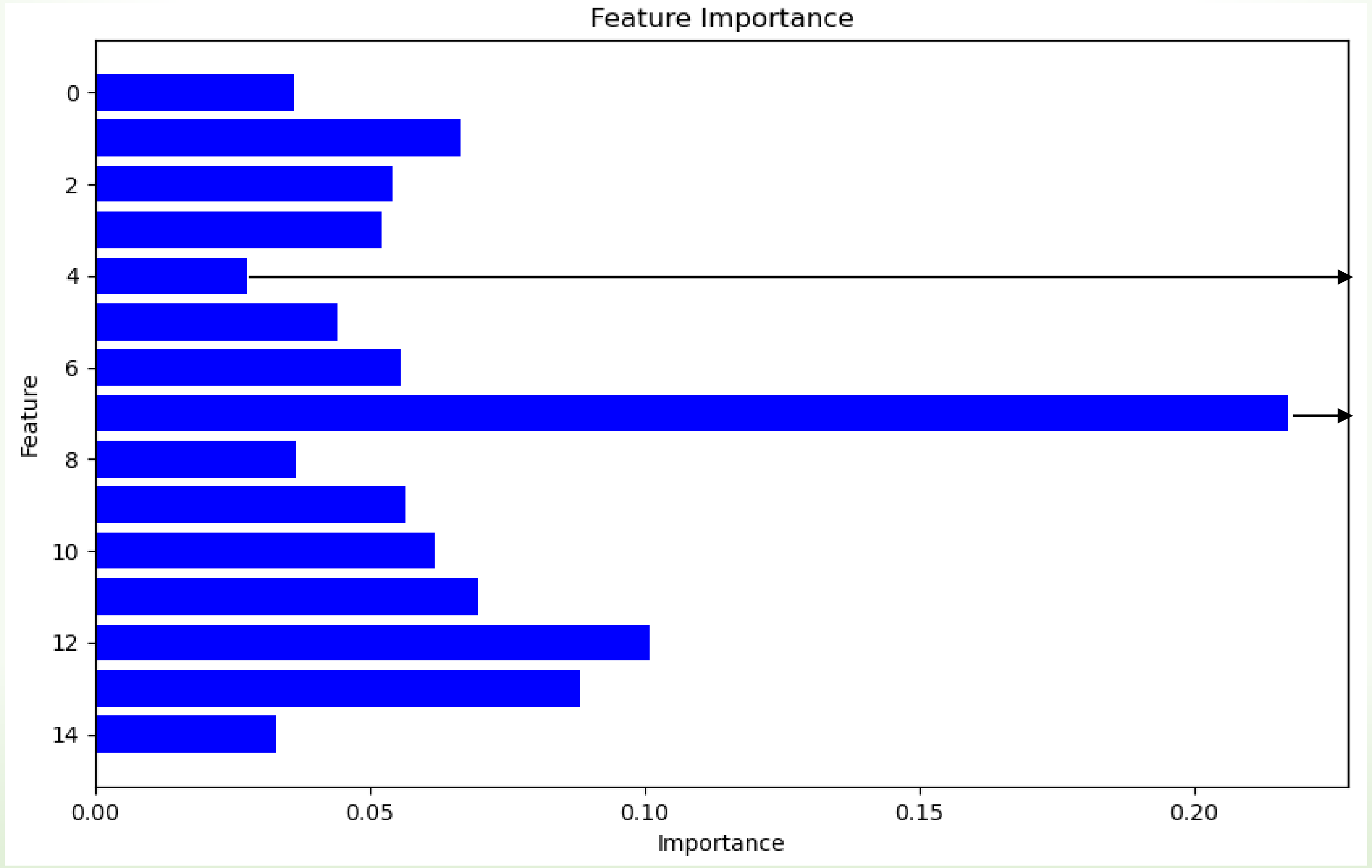
- **Feature Correlation by Regression**
  - ➢ It is the analysis of the relationship between two or more features using regression techniques, rather than simple correlation coefficients.
  - ➢ It plots **one feature against another and fits a regression line** to understand how one feature predicts or explains the other.
  - ➢ This method provides **more insight into the nature of the relationship, such as linearity or non-linearity**.





Data Understanding

- **Feature Importance**
  - Feature importance refers to techniques used to measure the **contribution of each feature in predicting the target variable** in a machine-learning model.
  - It quantifies **how much a feature influences the model's predictions**, helping to **identify the most relevant features**.
  - Feature importance is often derived from models like tree-based algorithms (e.g., Random Forest) or calculated through statistical methods.



Feature Importance
4 Low 0.027745

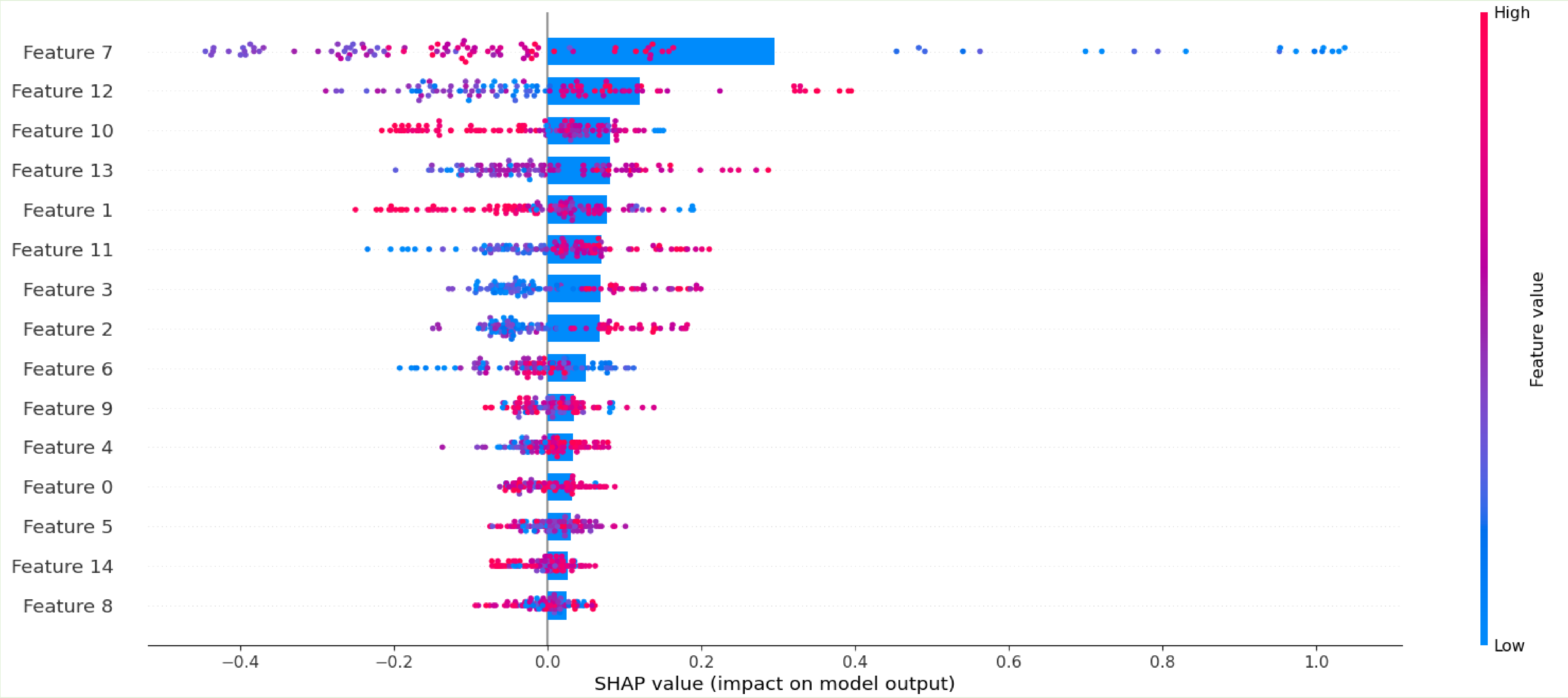
Feature Importance
7 High 0.217136

Data Understanding

- **SHAP**
  - SHAP (SHapley Additive exPlanations) is a method used to explain the output of machine learning models by **assigning each feature an importance value based on its contribution to a prediction**.
  - It is based on Shapley values from game theory, ensuring that the importance of each feature is fairly distributed.
  - SHAP helps make complex models like neural networks and tree-based models **interpretable by explaining how each feature influences a specific prediction**.
  - Importance in SHAP is **measured not by whether the feature increases or decreases the prediction but by how much it influences the prediction overall.**
  - **Even though most of the SHAP values (dots) for Feature 7 are on the negative side (decreasing the prediction), the range of SHAP values is wide**. This means that Feature 7 strongly influences predictions but in a negative direction for most samples.
  - The wide spread of SHAP values (from negative to positive) means that **Feature 7 is driving the predictions more than other features**, even if it mostly pushes predictions down.
  - Figure is in the next slide.

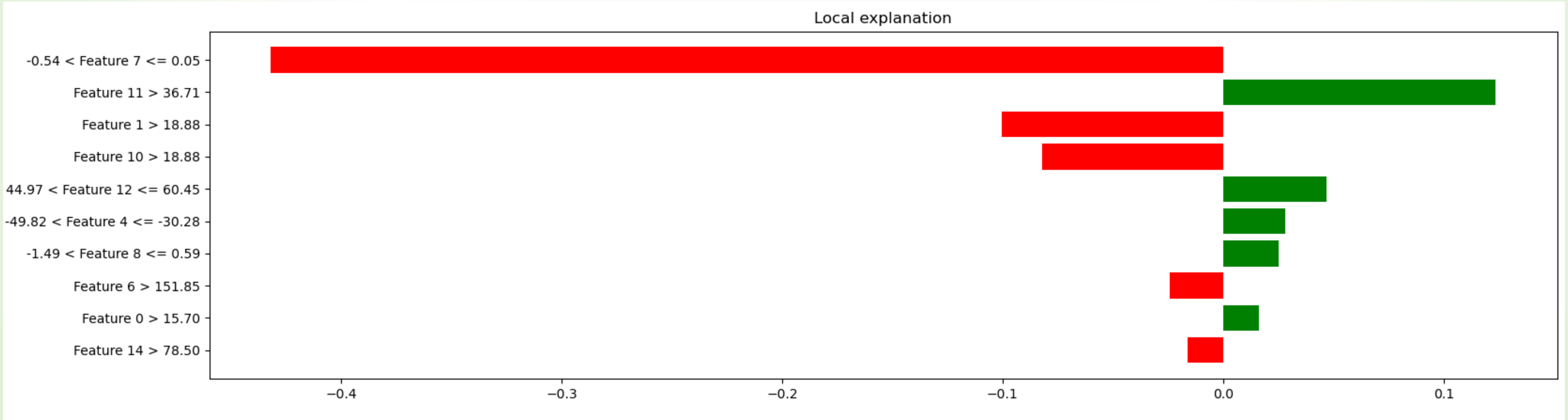
Data Understanding

- **SHAP**



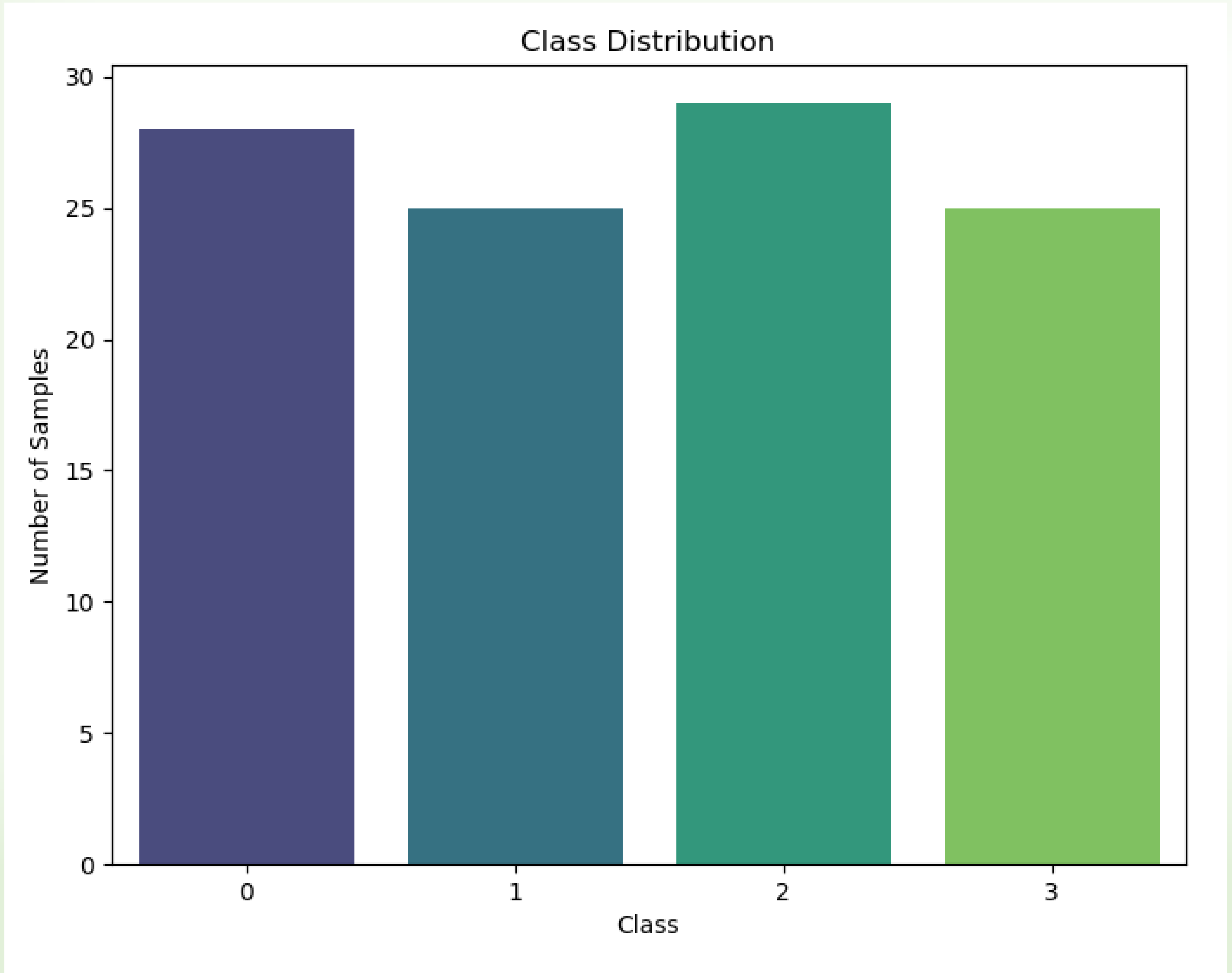
Data Understanding

- **LIME**
  - LIME (Local Interpretable Model-agnostic Explanations) is a technique used to **explain the predictions of any machine learning model** by approximating the model's behavior **locally around a specific instance**.
  - It generates **simple**, interpretable models (like linear models) for individual predictions to show how each **feature contributes to that specific prediction**.
  - **LIME is model-agnostic, meaning it can be applied to any machine learning model**, making it useful for explaining complex black-box models like neural networks and ensemble methods.
  - **It is instance - or sample-based**, **so we have to do it for different samples overall**. So, the figure is just for one sample.
  - Feature 7 has the largest negative impact, strongly reducing the prediction (red bar, far left).
  - Feature 11 has the largest positive impact, increasing the prediction (green bar, far right).
  - Feature 7 significantly decreases the prediction, while Feature 11 increases it.
  - LIME is showing how Feature 7 affects this specific prediction, but it **doesn't judge whether the effect is inherently "good" or "bad.“**
  - If we are trying to **minimize a prediction** (e.g., reducing costs, risks, or errors), **then a negative impact from Feature 7 is desirable**.
  - If we are trying to **maximize a prediction** (e.g., increasing profits or success probability), **then a negative impact from Feature 7 is undesirable.**



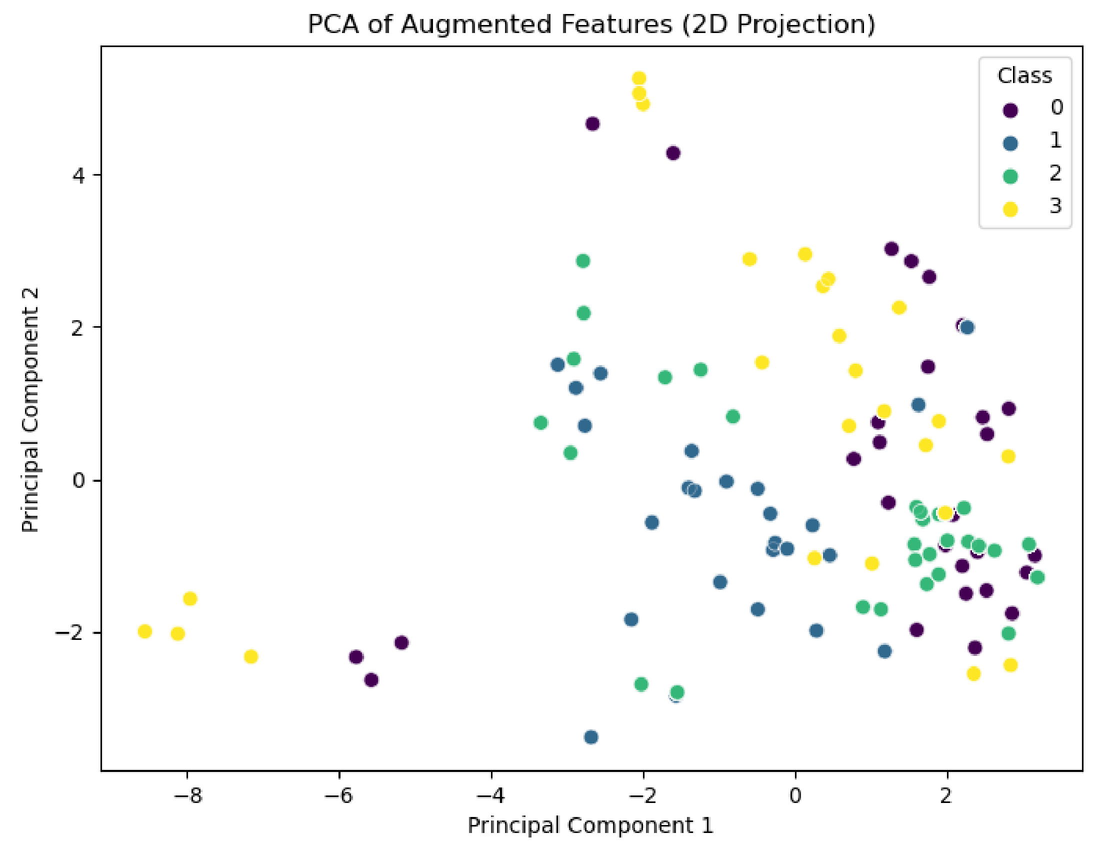
Data Understanding

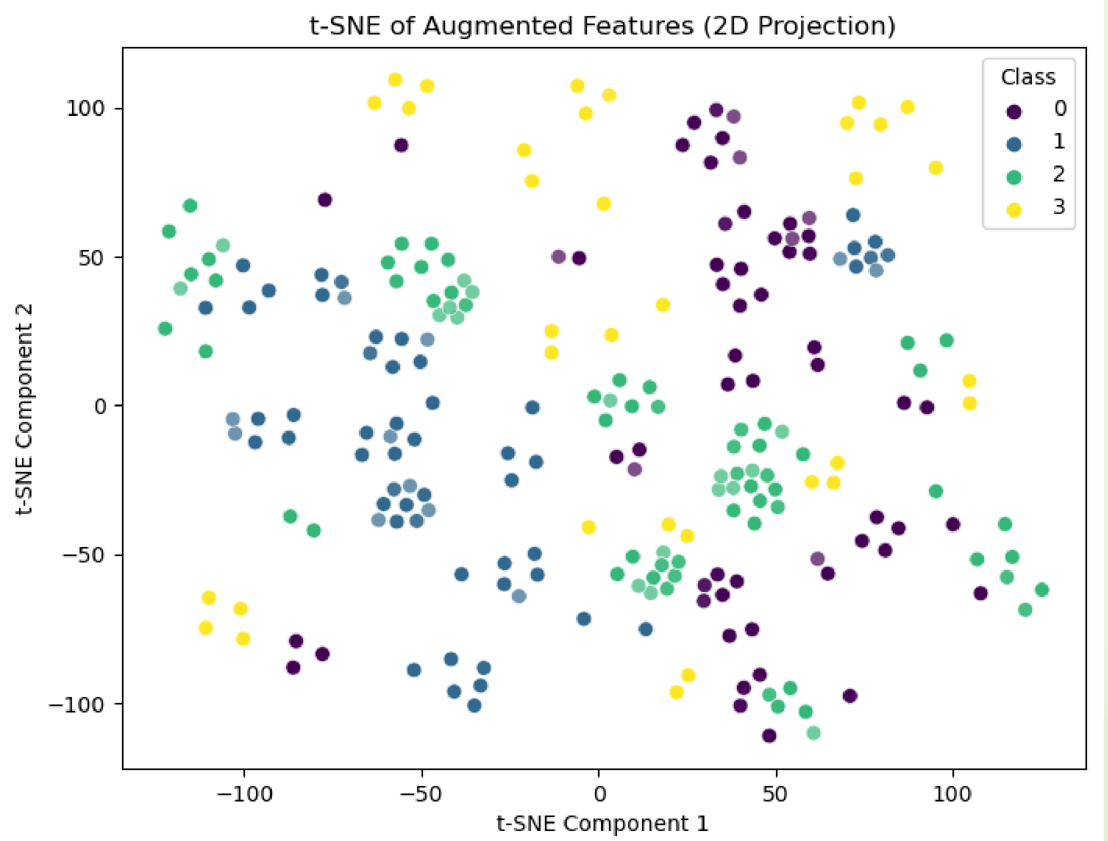
- **Class Distribution Ratio**
  - ➢ Class distribution ratio refers to the **proportion** of different classes in a dataset, typically in a classification problem.
  - ➢ It helps to understand whether the dataset is **balanced or imbalanced** in terms of the target classes.



Data Understanding

- **PCA and t-SNE Feature Plots**
  - A 2D PCA plot projects high-dimensional data onto two axes (PC1 and PC2) that **capture the most variance in the dataset**.
  - This allows us to visualize patterns, groupings, or clusters, with points that are **closer together being more similar in their features** and **points further apart being more distinct**.
  - t-SNE (t-distributed Stochastic Neighbor Embedding) is a non-linear **dimensionality reduction technique** that **visualizes high-dimensional data by preserving the local structure of data points**.
  - It is good at **displaying clusters and complex patterns in a lower-dimensional space** (typically 2D or 3D), **making similar data points appear close together and dissimilar ones far apart while maintaining local relationships better than PCA**.





Data Understanding

- **The Base Line and the Ground Truth**
  - A **baseline** is **a simple, minimal model used as a point of reference to compare the performance of more complex models**.
  - **It sets the foundation for understanding how well any more sophisticated algorithm performs.**
  - Using these baselines, **we can get a preliminary idea of what the minimum expected performance should be**, and any sophisticated model should ideally perform significantly better than these baselines to justify its complexity.

  - The **ground truth** refers to the **actual, real-world facts or correct labels** against which the predictions of a model are compared.
  - In classification, regression, or other supervised learning tasks, ground truth consists of the known labels or outputs provided in the training data.
    - ✓ For example, in image classification, the **ground truth would be the actual labels (like "cat," "dog," etc.)** that are correct for each image.
  - **The origin of the term comes from remote sensing and geographical information systems, where "ground truth" refers to data collected on location to verify the accuracy of aerial or satellite imaging.**
  - **it just means "the correct answers" or "actual facts.“**
  - **In classification, the "ground truth" refers to the correct labels assigned to the data points/samples.**

Metrics

- **TP, FN, FP, FN**
  - **True Positive (TP):** The model correctly predicts a positive class.
  - **False Negative (FN):** The model incorrectly predicts a negative class when the true class is positive.
  - **True Negative (TN):** The model correctly predicts a negative class.
  - **False Positive (FP):** The model incorrectly predicts a positive class when the true class is negative.

$$TP = \sum (\text{Actual} = 1 \wedge \text{Predicted} = 1)$$

It counts instances where both actual and predicted labels are positive (1).

$$FN = \sum (\text{Actual} = 1 \wedge \text{Predicted} = 0)$$

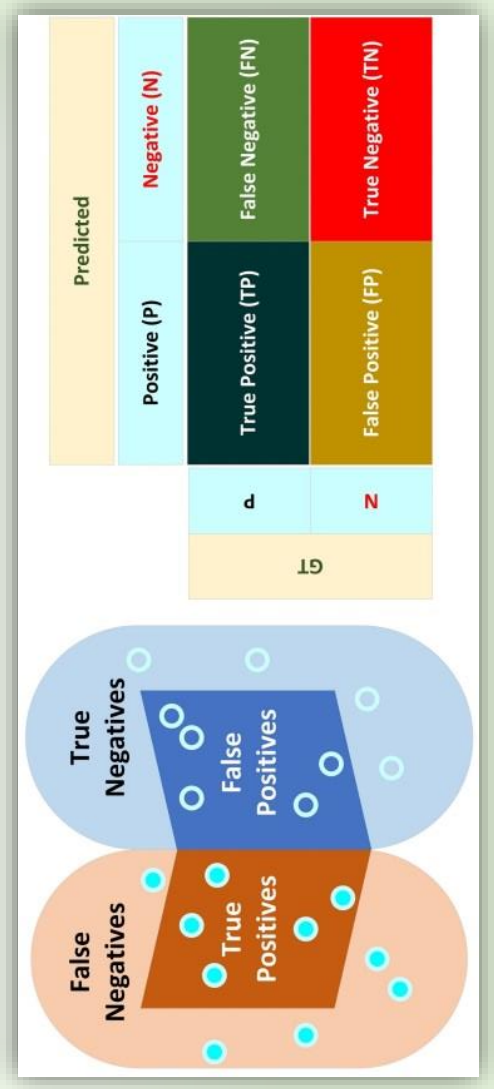
It counts instances where the actual label is positive (1), but the model predicted negative (0).

$$TN = \sum (\text{Actual} = 0 \wedge \text{Predicted} = 0)$$

It counts instances where both actual and predicted labels are negative (0).

$$FP = \sum (\text{Actual} = 0 \wedge \text{Predicted} = 1)$$

It counts instances where the actual label is negative (0), but the model predicted positive (1).



Metrics

- **Classification Report**
  - A classification report is a **performance summary** for a classification model that includes key metrics such as **precision (how many or percentage of selected items are relevant)**, **recall (how many or percentage relevant items are selected), F1-score (the harmonic mean of precision and recall),** and support (the number of true instances per class).
    - ✓ The **harmonic** mean is **a type of average** that is particularly useful when dealing with **ratios like precision and recall**. It gives **more weight to smaller values**, making it a **better measure** when both values are important.

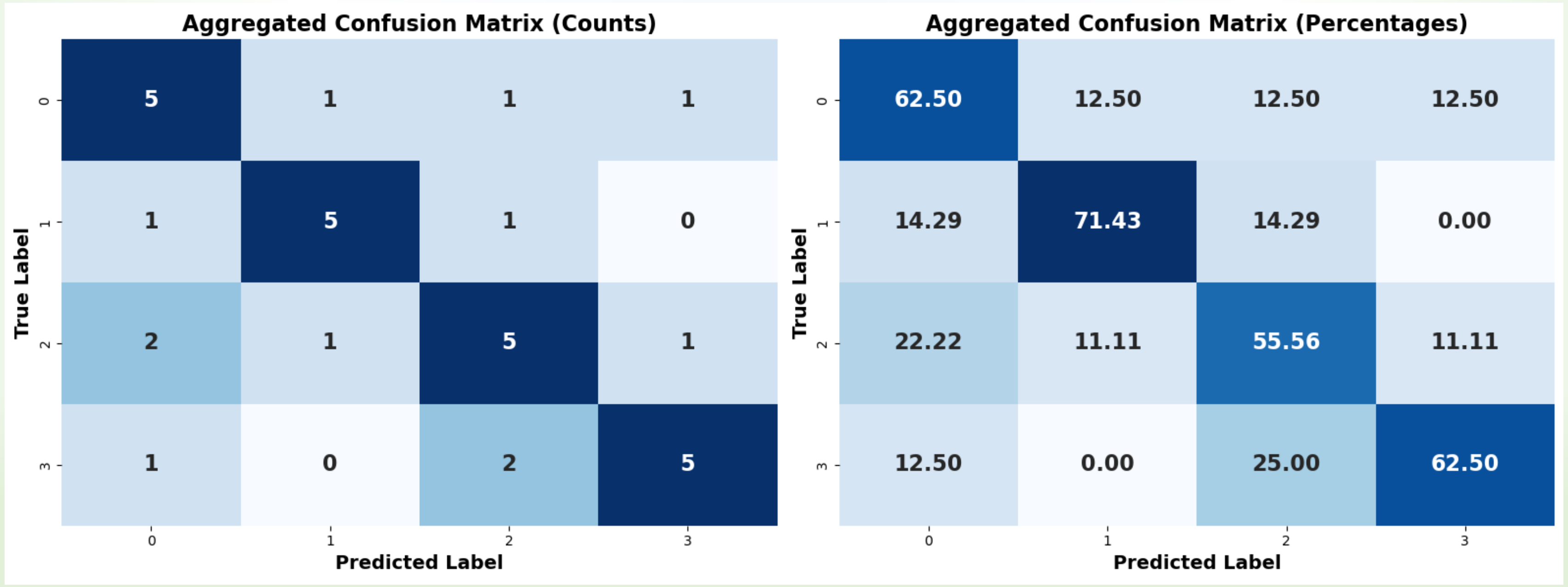
$$\text{Harmonic Mean} = \frac{2\times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

  - Each metric is calculated for each class, offering insight into how well the model performs across different categories. It helps evaluate model **accuracy, particularly when dealing with imbalanced data**.
  - TP (True Positive): These are cases where the **model correctly predicted the positive class.**
  - FN (False Negative): These are cases where the **model incorrectly predicted the negative class.**

Class	Precision	Recall	F1-Score	Support	ACC Per Class	TP	FN
0	0.60	0.64	0.62	56	0.64	36	20
1	0.62	0.77	0.69	56	0.77	43	13
2	0.58	0.51	0.54	63	0.51	32	31
3	0.79	0.66	0.72	56	0.66	37	19
Acc					**0.64**		
Macro Avg			0.65	0.64	0.64		
Weighted Avg			0.65	0.64	0.64		

Metrics

- **Confusion Matrix**
  - ➢ A confusion matrix is a table that **summarizes the performance of a classification model** by comparing **actual versus predicted classifications**.
  - ➢ It contains four key metrics: **True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).**
    - ✓ True Positives (TP): The top-left quadrant, where **the model correctly predicts positive outcomes**.
    - ✓ False Positives (FP): The top-right quadrant, where **the model incorrectly predicts positive when it should have predicted negative**.
    - ✓ False Negatives (FN): The bottom-left quadrant, where **the model incorrectly predicts negative when it should have predicted positive**.
    - ✓ True Negatives (TN): The bottom-right quadrant, where **the model correctly predicts negative outcomes.**
  - ➢ **Rows represent the actual classes, while columns represent the predicted classes**.
  - ➢ This matrix helps to assess the accuracy, precision, recall, and other evaluation metrics of the model across different classes.



Aggregated Confusion Matrix (Counts)

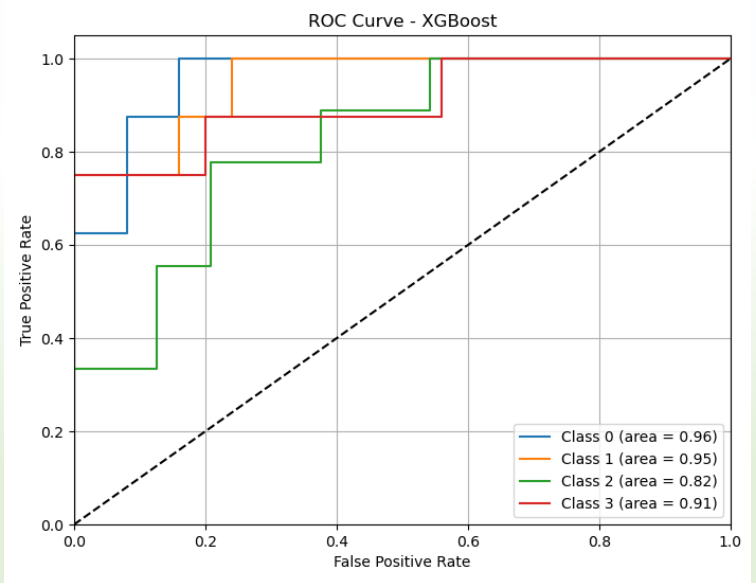
True \ Predicted	0	1	2	3
0	5	1	1	1
1	1	5	1	0
2	2	1	5	1
3	1	0	2	5

Aggregated Confusion Matrix (Percentages)

True \ Predicted	0	1	2	3
0	62.50	12.50	12.50	12.50
1	14.29	71.43	14.29	0.00
2	22.22	11.11	55.56	11.11
3	12.50	0.00	25.00	62.50

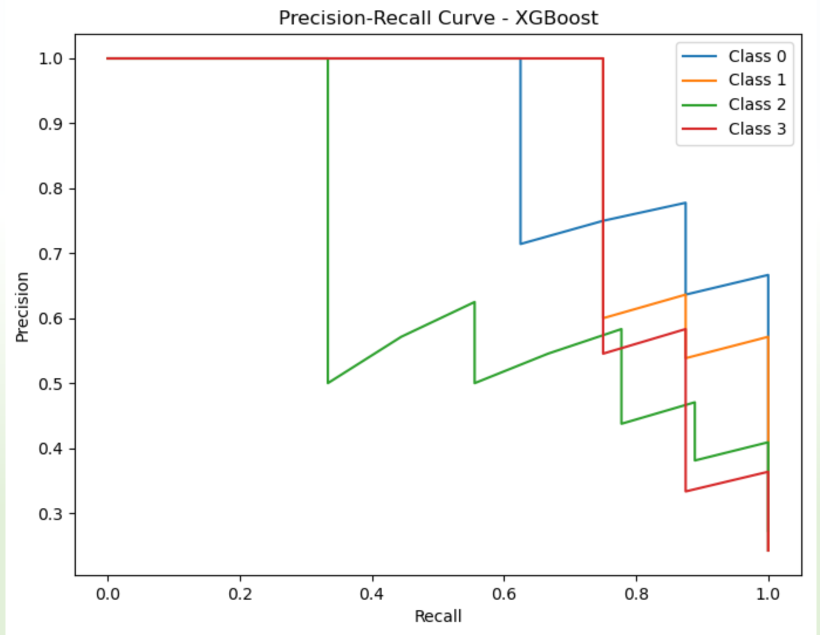
Metrics

- **ROC-AUC**
  - ROC-AUC (Receiver Operating Characteristic - Area Under the Curve) is a performance measurement for classification models, especially binary classifiers (here adapted for four classes).
  - **It's most useful when the class distribution is balanced.**
  - **ROC (Receiver Operating Characteristic):** It's a **curve that plots the True Positive Rate (Recall) against the False Positive Rate (FPR) at various threshold** settings. It shows the **trade-off between sensitivity and specificity**.
  - The threshold here is **0.5**.
  - **The ROC curve plots the True Positive Rate (Recall) on the Y-axis against the False Positive Rate on the X-axis.**
  - **AUC (Area Under the Curve):** This measures the **area under the ROC curve**. A perfect model will have an AUC of 1, while a random classifier will have an AUC of 0.5.
  - This is a single numerical value that represents the area under the ROC curve.
  - **The closer the AUC is to 1, the better the model's performance. (the legend in the figure).**
  - The **diagonal dashed line represents the baseline performance of a random classifier**. **It is always 0.5**.
  - The ROC curves for each class are significantly above the diagonal dashed line, which means your XGBoost model is performing well for each class.



Metrics

- **Precision-Recall AUC**
  - Precision-Recall AUC is the area under the Precision-Recall curve, which is an evaluation metric used especially when dealing with **imbalanced datasets**.
  - **It is more suitable for imbalanced datasets, where the positive class is much smaller than the negative class (unlike ROC-AUC)**
  - The curve plots Precision (the proportion of correctly predicted positives out of all predicted positives) against Recall (the proportion of actual positives that were correctly predicted) at different thresholds.
  - **Here, we have Recall in the X-axis and precision in the Y-axis.**
  - **Each step in the curve represents the performance at a specific threshold.**
  - **The threshold is a probability value that the model uses to decide whether a prediction belongs to a specific class.**
  - For example, if the threshold is 0.5, the model will predict a class if its probability is greater than or equal to 0.5 and predict otherwise if it's less than 0.5.
  - The threshold is changing continuously (increasing or decreasing) across the full probability range.
  - It does not matter if the curve is going down or up. The application determines if it is good or bad which is another topic.



Metrics

- **Log Loss / Cross-Entropy Loss**
  - Log Loss (also known as **logarithmic loss** or **cross-entropy loss**) is a performance metric for classification models that evaluates **how well the model's predicted probabilities match the actual class labels**.
  - **It penalizes incorrect predictions with a high probability more heavily**.
  - **A lower log loss indicates better model performance.**
  - Log Loss, cares about the **probability estimates**.
  - It **penalizes the model for being confident in wrong**.
  - **If the model predicts a class with a high probability as correct but gets it wrong, the log loss will be high (high is bad)**, even if the ROC-AUC is good.

$$\log \text{Loss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\log(p_{i,c})$$

Where:

- N is the number of samples.
- C is the number of classes.
- $y_{i,c}$ is 1 if sample i belongs to class c, and 0 otherwise.
- $p_{i,c}$ is the predicted probability that sample i belongs to class c.

Metrics

- **Cohen's Kappa**
  - Cohen's Kappa is a statistical measure used to evaluate the **level of agreement between two raters** (or a model and ground truth labels) while accounting for the possibility of agreement occurring by chance.
  - In classification problems, it provides a **more robust evaluation of model performance than simple accuracy**, **particularly in cases of class imbalance.**
  - **The range is from -1 to 1, and the higher, the better.**
  - It compares the observed accuracy of the model (how many predictions were correct) with the expected accuracy (the agreement that could occur by random chance).
  - It's especially useful **in imbalanced datasets**, where simple accuracy might be misleading.
  - It would indicate that the model is correctly classifying instances more often than what would be expected by chance.
  - The term **agreement** is to how often the model's predictions match the actual (true) class labels.
  - The term **chance** means the level of agreement that could occur randomly if the model were making random guesses rather than informed predictions.
  - In Cohen's Kappa, "chance" refers to the agreement between predictions and actual labels that could happen just based on the distribution of the classes, not actual random guessing by the classifier.

Cohen's Kappa (κ) is calculated as:

$$\kappa = \frac{p_o - p_e}{1 - p_e}$$

Where:

- p_o = Observed agreement (the accuracy of the model, or how often the predicted labels agree with the true labels).
- p_e = Expected agreement (the agreement you would expect by random chance).

Metrics

- **MCC**
  - The Matthews Correlation Coefficient (MCC) is a metric used to evaluate the quality of binary (and multi-class) classifications.
  - It takes into account true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), **providing a balanced measure even with imbalanced datasets**.
  - That means: **Unlike accuracy, which can be misleading in imbalanced datasets (where one class dominates), MCC gives a comprehensive view of how well the model is performing across both classes.**
  - **The range is from -1 to 1, and the higher, the better.**
  - In other words, MCC provides a single score that reflects the performance across both the majority and minority classes, ensuring that a good score requires the model to correctly classify both positive and negative classes, even if one class significantly outnumbers the other.
  - This makes **MCC more robust than simple accuracy** in cases where **class imbalance** exists.
  - Note:
    - ✓ **ROC-AUC** can be **influenced by the majority class** and might still show a good score even if the minority class is not well predicted.
    - ✓ **Precision-recall AUC** is **more focused on the minority class** and is often preferred in highly imbalanced datasets where positive class predictions matter more.
    - ✓ **MCC and Cohen's Kappa offer more balanced views and adjust for baseline performance**.

$$\text{MCC} = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Metrics

- **Precision**
  - Precision is a performance metric used in classification tasks, particularly when the focus is on **minimizing false positives**.
  - It measures the **proportion of true positive predictions out of all the instances predicted as positive** by the model.
  - Precision helps **evaluate the accuracy of positive predictions.**
  - It is especially useful in **situations where the cost of false positives is high**, such as in **medical diagnoses or spam detection**.
  - **In simple terms, it says that it is so precise to detect any wrong and recognize just true predictions as the applications are so sensitive.**
  - A high precision means that the model makes fewer false positive errors, but it does not consider how well the model identifies all positive instances (which is handled by recall).
  - Precision is part of the precision-recall trade-off in classification problems.

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

Where:

- TP is the number of true positives.
- FP is the number of false positives.

Metrics

- **Recall (Sensitivity)**
  - Recall (also known as sensitivity or true positive rate) is a metric used to measure **how well a classification model identifies all the actual positive instances** in a dataset.
  - It calculates the **proportion of true positive predictions out of the total number of actual positive cases**.
  - Recall is crucial in scenarios where it's important to minimize false negatives, such as in **medical screenings or fraud detection**, **where missing a positive case can have serious consequences.**
  - A high recall indicates that the model is effectively capturing most of the actual positive cases, but it doesn’t account for false positives, which is considered by precision.
  - Recall is part of the precision-recall trade-off, often used together with precision for model evaluation.

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

Where:

- TP is the number of true positives.
- FN is the number of false negatives.

Metrics

- **Unweighted Average Recall**
  - UAR is the **average of recall values for all classes** without considering the class distribution.
  - **It simply computes the recall (true positive rate) for each class and then averages them.**
  - Unweighted Average Recall (UAR) **is a bit better than accuracy for imbalanced data because it treats each class equally when calculating recall**.
  - UAR improves on accuracy by averaging recall across all classes, **giving minority classes a chance to influence the score**.
  - However, it is not specifically designed to handle class imbalance, as it doesn't fully address the disproportionate impact that a well-performing majority class can have on the overall metric.

$$\text{UAR} = \frac{1}{C}\sum_{i=1}^{C} \frac{\text{True Positives for Class i}}{\text{Total Actual Positives for Class i}}$$

where C is the number of classes.

Metrics

- **Specificity**
  - Specificity (also called the True Negative Rate) **measures the proportion of actual negative cases that are correctly identified** by the model.
  - It focuses on **how well the model is at avoiding false positives**.
  - High Specificity: The model is good at identifying negative cases and making a few false positive errors. **The higher, the better**.

$$\text{Specificity} = \frac{\text{True Negatives (TN)}}{\text{True Negatives (TN)} + \text{False Positives (FP)}}$$

Metrics

- **F1-Score**
  - F1-score is a metric that **balances precision and recall**, **providing a single score to evaluate a model's performance, especially when there's an uneven/imbalanced class distribution**.
  - It is particularly useful in scenarios where both false positives and false negatives carry significant consequences.
  - The F1-score is the **harmonic mean of precision and recall, giving a higher weight to low values**, making it a more conservative measure compared to the arithmetic mean.
  - Which means:
  - For example, if the model has:
    - ✓ Precision = 0.9 (meaning the model is good at avoiding false positives)
    - ✓ Recall = 0.4 (meaning the model is missing many actual positive instances)
    - ✓ The **F1-score (with harmonic mean) will be much closer to 0.4 than 0.9 because the harmonic mean pulls the score towards the lower value**.
    - ✓ But the **normal mean** goes in between, like 0.7.
    - ✓ This helps **prevent a high precision from misleadingly inflating the performance score** when recall is low.
  - **A high F1 score indicates both high precision and high recall**, while a low score shows that the model struggles in either precision, recall, or both.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Metrics

- **Accuracy**
  - Accuracy is a simple and **widely used metric** in classification tasks that measures the **proportion of correctly predicted instances (both positives and negatives) out of the total number of instances.**
  - **It is best suited when the classes in a dataset are relatively balanced.**
  - **However, accuracy can be misleading in imbalanced datasets, where one class dominates, and the model could achieve high accuracy simply by predicting the majority class.**
  - While it is an easy-to-understand metric, it doesn't give insight into the performance for individual classes like precision, recall, or F1-score.

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

Metrics

- **Balanced Accuracy**
  - Balanced Accuracy is a metric designed to handle imbalanced datasets by taking into account the performance of **both the majority and minority classes.**
  - It is **the average of the recall (or True Positive Rate) across all classes**, ensuring that **each class contributes equally to the final score**, **regardless of how imbalanced the dataset is**.
  - It **balances the recall across both the positive and negative classes**.
  - In cases of **imbalance**, **where one class dominates**, **Balanced Accuracy ensures that the minority class's performance is given equal importance**.
  - A Balanced Accuracy of 1 indicates perfect classification, while 0.5 means the model is performing no better than random chance.
  - Balanced Accuracy ensures that both majority and minority classes are treated equally in terms of recall.
  - It's simpler than MCC and gives an overall balanced view of classification performance.
  - **MCC is more robust** because it accounts for the relationships between all types of errors and correct predictions, not just recall.
  - **MCC is often more reliable** in very imbalanced datasets where both precision and recall are important.
  - **Balanced Accuracy** focuses more on recall across classes.
  - **MCC** provides a more comprehensive view by considering all aspects of the confusion matrix (precision, recall, true negatives, etc.), making it more robust for complex, imbalanced scenarios.
  - **So, MCC is better for imbalanced datasets as it covers more details.**

$$\text{Balanced Accuracy} = \frac{1}{2}\left(\frac{\text{TP}}{\text{TP+FN}} + \frac{\text{TN}}{\text{TN+FP}}\right)$$

Metrics

- **Hamming Loss**
  - Hamming loss is a metric used to measure the performance of classification models, particularly in multi-label settings.
  - It **calculates the fraction of labels that are incorrectly predicted**, thus **emphasizing the error rate per label**.
  - **The lower the Hamming loss, the better the classifier's performance**, as it indicates fewer misclassifications.
  - It's useful for comparing classifiers on an equal basis **when label imbalance is present**.
  - It focuses on **the number of label mismatches between the actual and predicted labels across all labels.**
  - Hamming loss does this by counting mismatches in binary label predictions, while MSE does it by calculating the square of numerical prediction errors.

$$L_H(y,\hat{y}) = \frac{1}{N}\sum_{i=1}^{N}\frac{\operatorname{xor}(y_i,\hat{y}_i)}{L}$$

Where:

- $L_H(y,\hat{y})$ is the Hamming loss between the true labels y and predicted labels $\hat{y}$.
- N is the number of samples.
- L is the number of labels.
- $\operatorname{xor}(y_i,\hat{y}_i)$ represents the XOR operation between the true and predicted labels for each label in a sample, effectively counting the number of label mismatches.

This calculation provides the average fraction of incorrect labels per sample, normalized by the total number of labels.

Metrics

- **Jaccard Index**
  - The Jaccard Index, also known as the **Jaccard similarity coefficient**, is a statistic used for measuring the similarity and diversity of sample sets.
  - It's calculated as **the size of the intersection divided by the size of the union of two label sets**.
  - In binary and multi-label classification tasks, it measures **how many labels are correctly predicted, ignoring the true negatives**.
  - **This metric ranges from 0 (no overlap) to 1 (perfect agreement). The higher, the better.**
  - **A higher Jaccard Index indicates a greater degree of similarity between the predicted labels and the true labels.**

$$J(y_i, \hat{y}_i) = \frac{TP}{TP + FP + FN}$$

Where for each class:

- TP (True Positives) is the count of correct predictions for a particular class.
- FP (False Positives) is the count of other class instances incorrectly predicted as this class.
- FN (False Negatives) is the count of instances of this class incorrectly predicted as some other class.

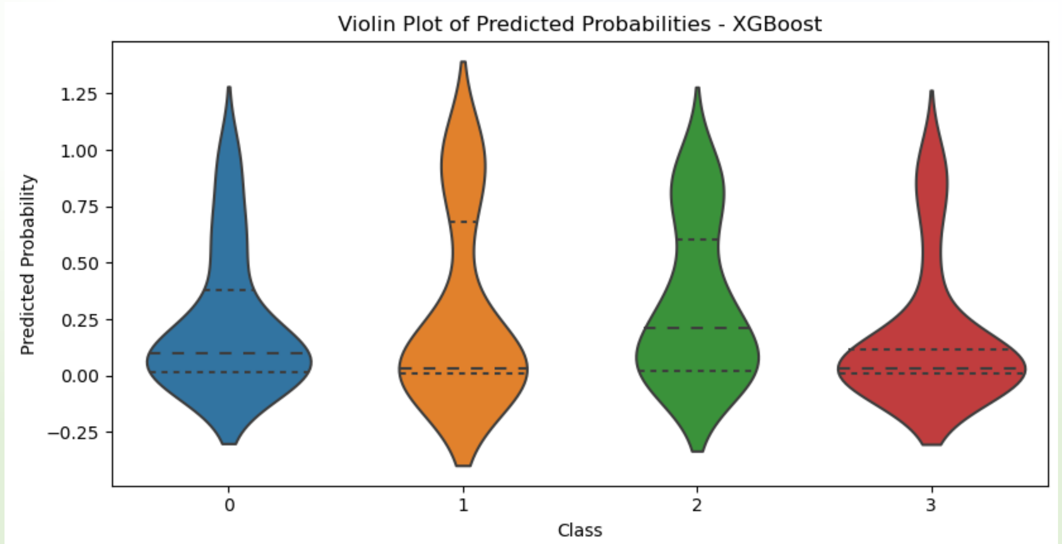
In practice, it **calculates this for each class and can then take the average Jaccard Index across all classes to get a single measure of performance.** This is often referred to as the mean Jaccard Index or Jaccard similarity score for multi-class classification scenarios.

Evaluation Plot

- **Violin Plot**
  - A violin plot is a method of visualizing the distribution of numerical data, combining aspects of a box plot and a kernel density plot.
  - In the classification, a violin plot can be used to visualize the distribution of feature values or prediction probabilities for different classes.
  - This helps in understanding how different features or predicted probabilities vary across known categories, which can aid in assessing the separability of classes based on these features or probabilities.
  - It provides insights into which features might be influential in predicting a class and if any overlap occurs between classes, indicating potential challenges in classification.
  - A **prediction** is the output of a model when you input some features. It's the **model's best guess at the label or class for the given input.**
  - **Probability** refers to the likelihood or **confidence the model has in its prediction**.
  - It's usually a number between 0 and 1. A **probability close to 1 means the model is very confident** that the input belongs to a particular class, and a probability close to 0 means the model is confident the input does not belong to that class.
  - Each violin plot displays the **distribution and density of the predicted probabilities** for a specific class from a model's output.
  - The "**density of the predicted probabilities**" refers to **how often certain probability values occur**.
    - ✓ For example, **if many predictions are close to 0.5, the plot will be wider at that value**, **indicating a higher density of predictions around 0.5.**
  - **if the wider sections are located towards the higher probability values (closer to 1), it indicates that a larger number of samples have been classified with high confidence for that particular class.**

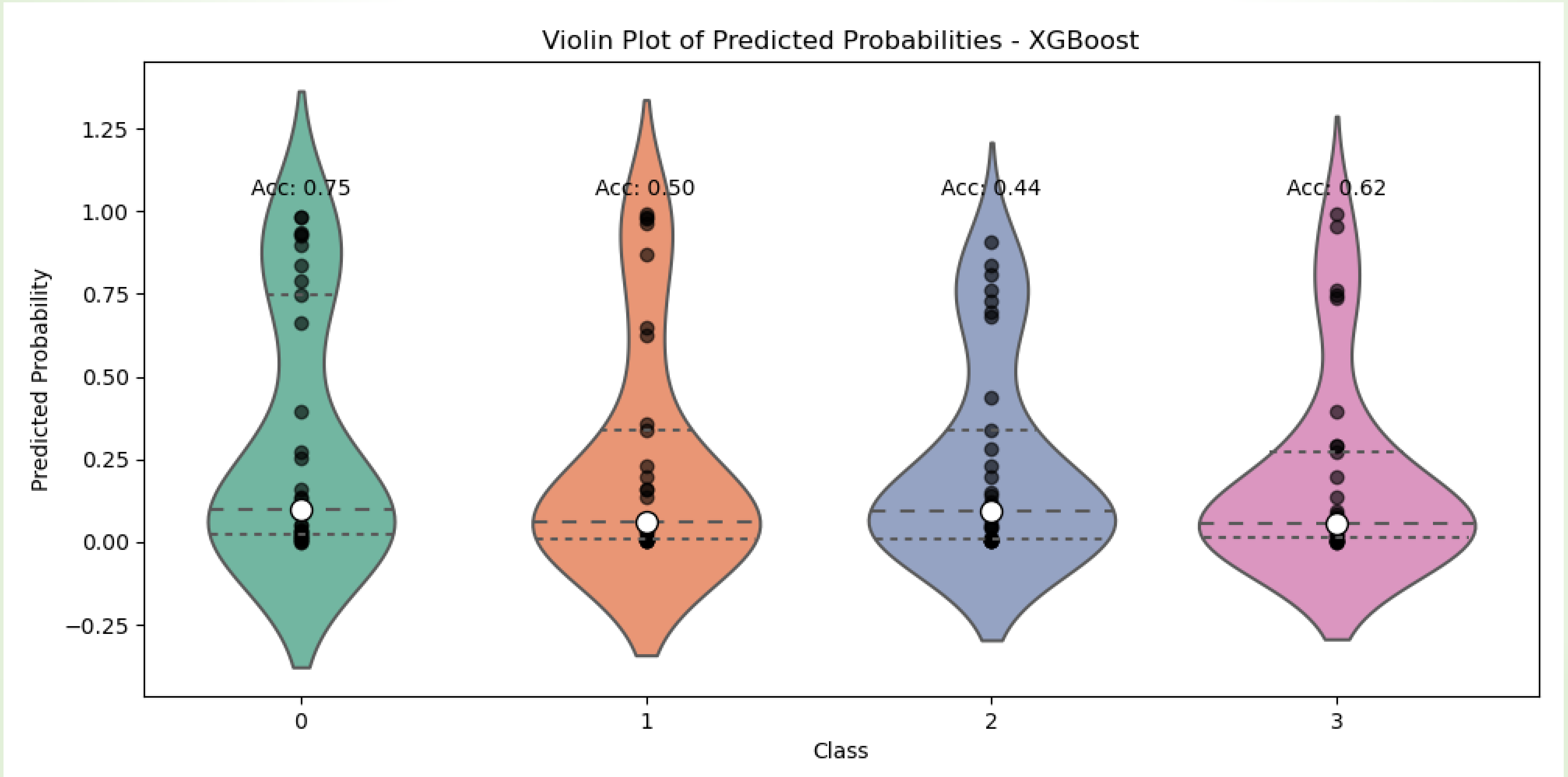
Evaluation Plot

- **Violin Plot**
  - ➢ **The wider, the more predictions signifying that classification percentage**. For example, if a line of around 0.25 is wide, there are more samples around that area for that class.
  - ➢ So, in the figure, except for the green class 2, things are not good in general.
  - ➢ The simplest interpretation is that the model is generally not very confident about assigning instances to Class 0, with most predictions falling below 0.5.
  - ➢ For class 0: The widest point of the violin is near the lower end of the probability scale, particularly around 0.25. This indicates that a significant number of predictions are clustered around this probability.
  - ➢ The dashed lines in a violin plot typically represent the **quartiles** of the data distribution.
    - ✓ Lower Quartile (bottom dashed line): This line marks the 25th percentile, **meaning 25% of the data falls below this value**.
    - ✓ Median (middle dashed line): This line marks the 50th percentile or the **median of the data**. It splits the data into two halves, with **50% of the data falling below this point and 50% above**.
    - ✓ Upper Quartile (top dashed line): This line marks the 75th percentile, meaning **75% of the data falls below this value**.
    - ✓ **So, the higher the dashed lines, the better.**
    - ✓ **Also, if the above parts are wider, it is better.**



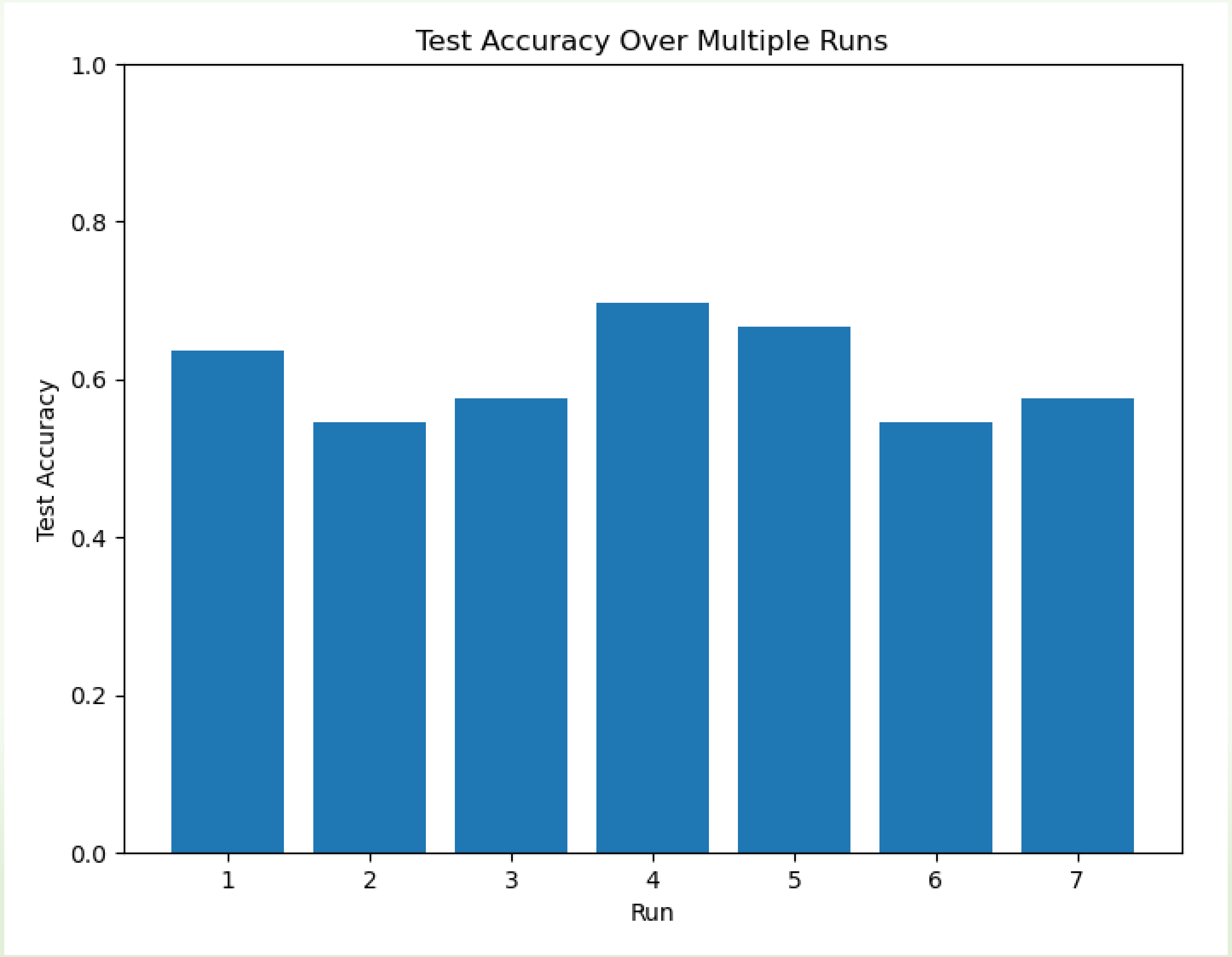
Evaluation Plot

- **Violin Plot**
  - Here, the **white dot is the median**, a statistical measure showing the middle value of predicted probabilities for each class.
  - Accuracy of each class is added.
  - Also, the density of samples per class is presented by black dots. The wider, the more samples in that area.
  - **First class (left, green), which has the upper part wider, has more accuracy. Also, higher upper dashed line.**



Evaluation Plot

- **Test Accuracy Plot**
  - ➢ The purpose of it is to observe the **variance** in test accuracy across different runs, indicating the **model's stability and performance consistency.**

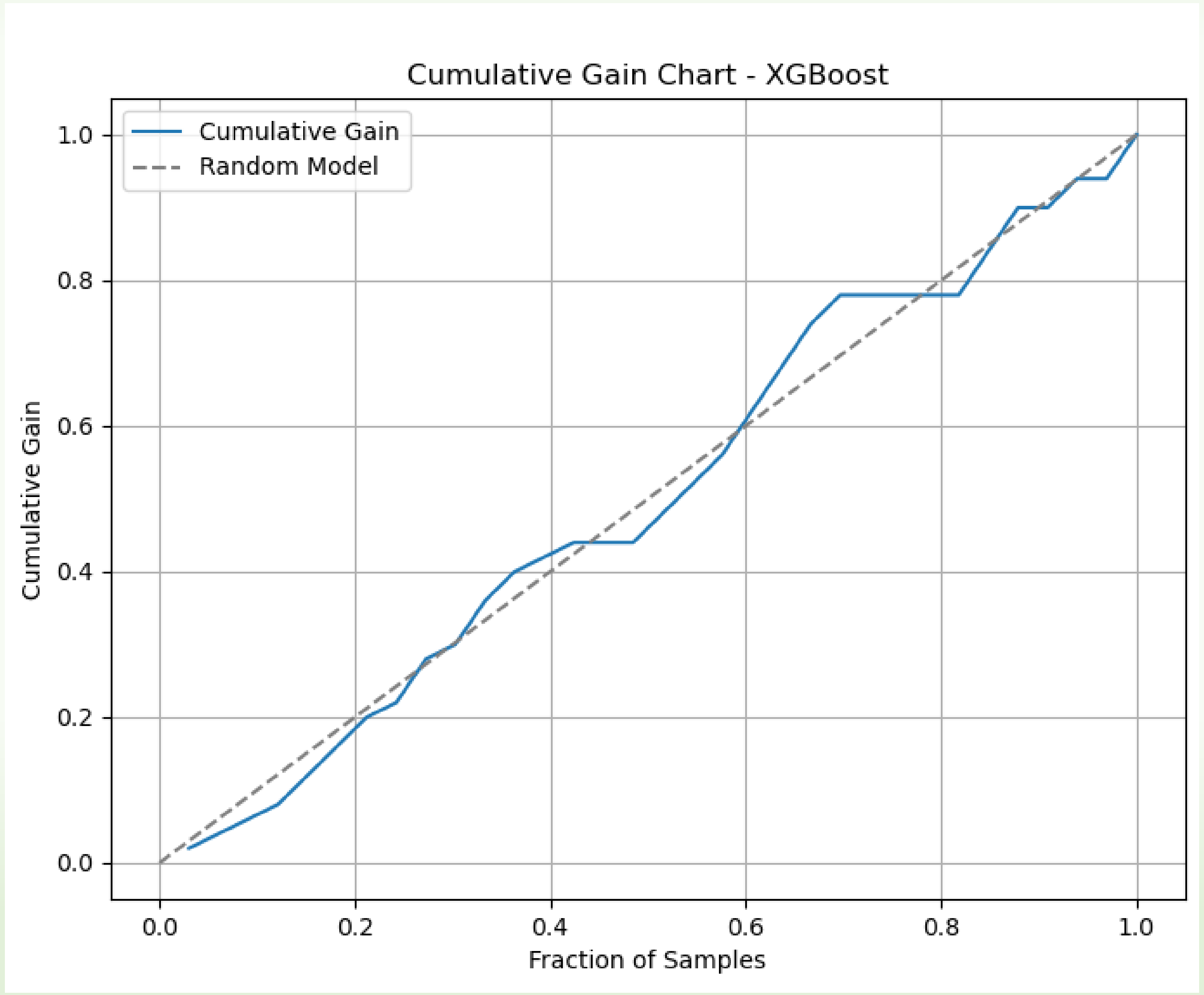


Evaluation Plot

- **Cumulative Gain Chart**
  - When the model classifies a sample; it doesn't just output a class (e.g., "Angry" or "Sad"). It **often provides a probability for each class** (e.g., **70% probability of "Angry," 20% probability of "Sad,"** etc.).
  - The model **ranks the samples based on these predicted probabilities** for the target class (e.g., "Angry") or for all classes.
  - The Cumulative Gain Chart **evaluates how well the model ranks positive cases (e.g., true "Angry" samples) at the top.**
  - Here, the Cumulative Gain Chart **evaluates how well the model is performing across all classes (e.g., Angry, Sad, Happy, Neutral) in ranking correct predictions.**
  - **In other words, for each sample, the model ranks how confident it is that the sample belongs to its actual class (whether that's Angry, Sad, Happy, or Neutral).**
  - **The chart shows how many correct predictions (across all emotions) are captured as we include more of these ranked samples, starting with the most confident predictions.**
  - The x-axis represents the cumulative fraction of samples (for all emotions) that the model has included, ordered by confidence in its predictions.
  - The y-axis represents the cumulative fraction of correctly classified samples (for all emotions).
  - The further the curve is above the diagonal (random model), the better your model is at prioritizing correct predictions.
  - If the curve is well above the diagonal, the model is doing a good job of ranking the correct predictions for all emotions higher.
  - If it's close to the diagonal, the model may be struggling to confidently identify and rank samples across the different emotions.

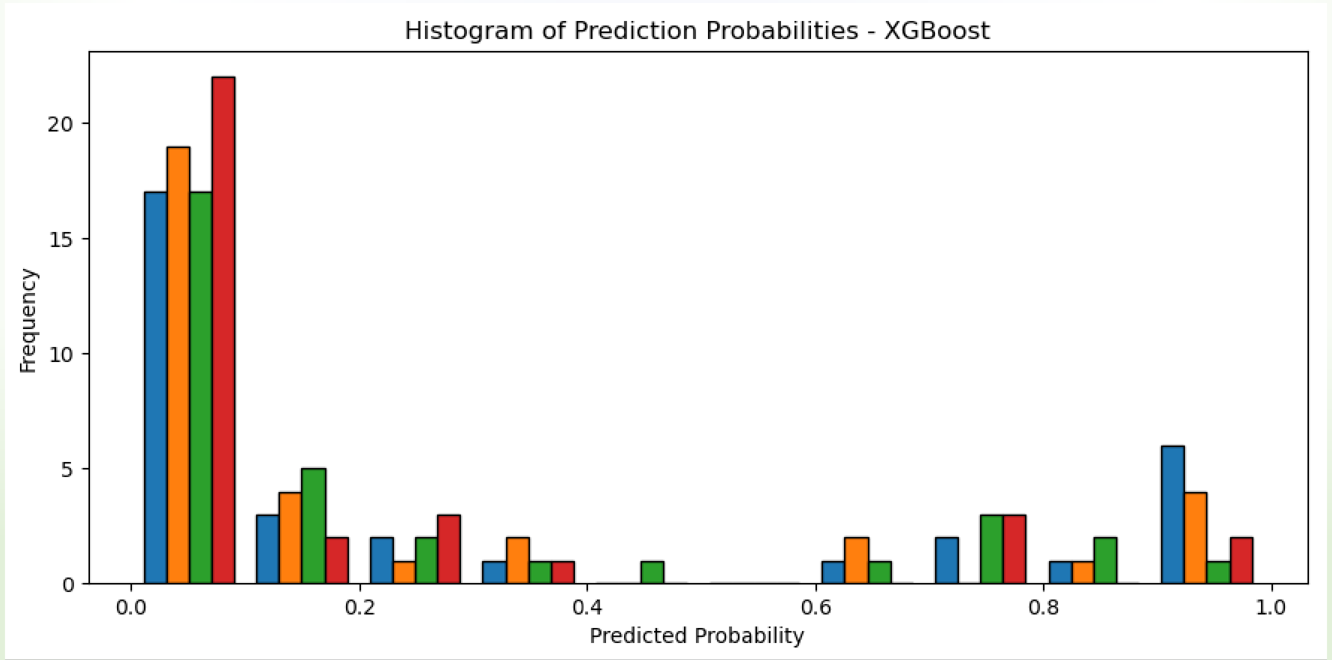
Evaluation Plot

- **Cumulative Gain Chart**
  - ➢ In this figure, the performance of the model **is slightly better than the random 50 percent**, **as accuracy is just 61 percent**.
  - ➢ **So, the higher blue line above the dashed line over time indicates better positive prediction of samples over time.**



Evaluation Plot

- **Prediction Probabilities Histogram**
  - A Prediction Probabilities Histogram visualizes the distribution of the model's predicted probabilities for different classes in classification.
  - It shows **how confident the model is about its predictions**.
  - The x-axis represents the predicted probability values (typically ranging from 0 to 1).
  - The y-axis represents the **frequency or count of predictions (number of samples)** at each probability level.
  - **A good model will have higher bars near 0 and 1, indicating confident predictions.**
  - It shows how many predictions the model made with **low, medium, or high confidence** (e.g., close to 0, around 0.5, or close to 1).
  - In the figure, the tall bars near 0 indicate that the model is assigning **a very low probability for some classes.**
  - There are a few bars near 1, which indicate that the model is highly confident in predicting certain classes correctly.
  - But in general, it is moderate to bad.
  - **Taller bar near 1 in the x axis, the better.**

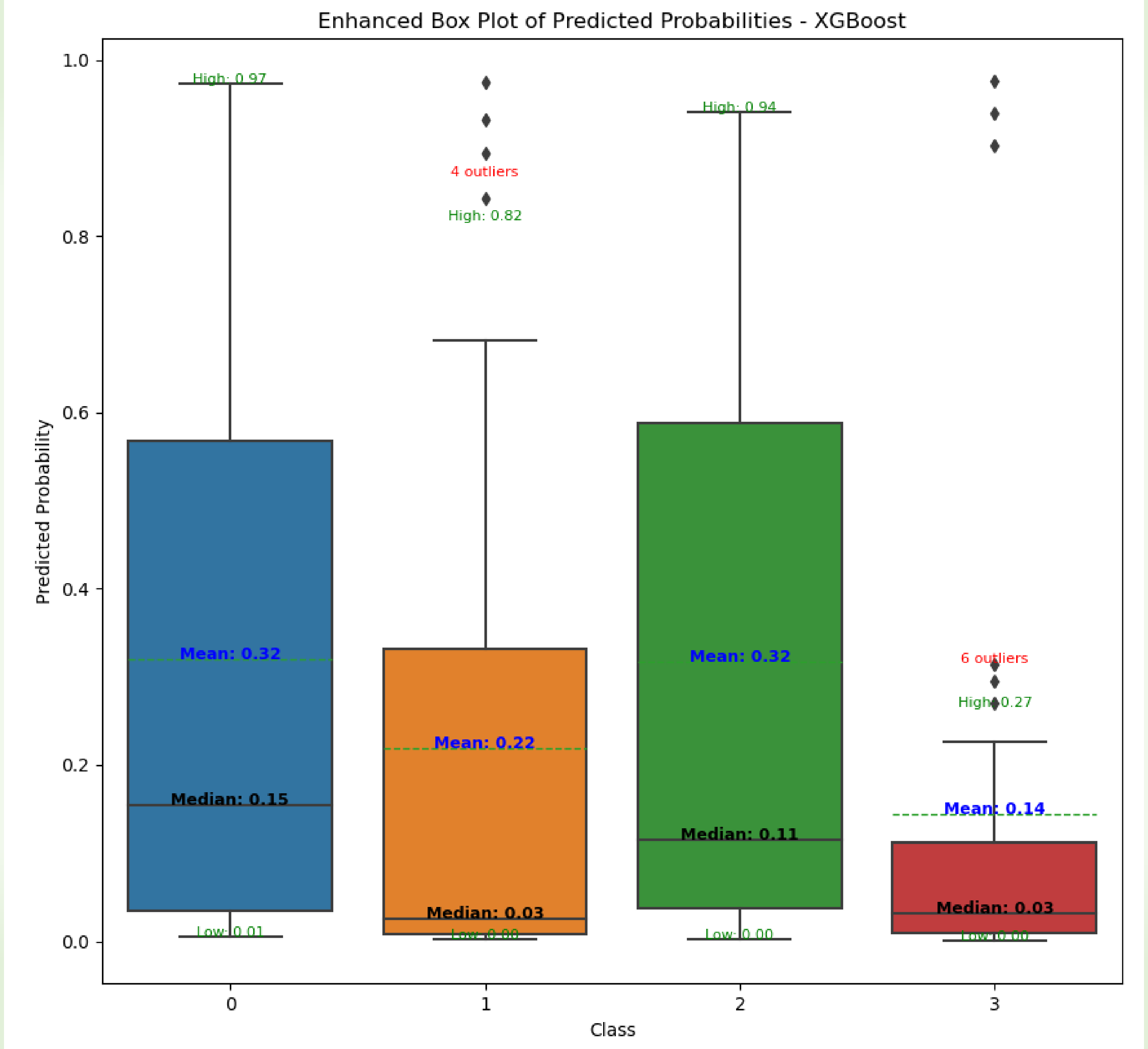


Evaluation Plot

- **Box Plot**
  - ➢ A box plot (or box-and-whisker plot) visually summarizes the **distribution of a dataset by showing its quartiles**.
  - ➢ The plot displays the **median (middle value), upper and lower quartiles (25th and 75th percentiles),** and **whiskers** representing the range within 1.5 times the interquartile range (IQR).
  - ➢ **Outliers are displayed as individual points beyond the whiskers.**
  - ➢ Box plots are useful for **identifying the spread, skewness, and outliers** in a dataset.
  - ➢ **Whiskers** in a box plot are the lines that **extend from the box to the minimum and maximum values**.
  - ➢ While outliers (data points beyond this range) are shown as individual points outside the whiskers.
  - ➢ In short, **whiskers represent the spread of most of the data but exclude extreme outliers**.
  - ➢ The **thick black line** is the **median (50th percentile),** which is annotated for each class.
  - ➢ The **dashed green line inside the box** is the **mean**, which is also annotated for each class.
  - ➢ The **mean represents the average predicted probability** for each class across all samples.
    - ✓ For example, if the mean for class 0 is 0.32, it means that, on average, the model predicts a 32% chance of class 0 being the correct class for the samples.
  - ➢ The **median is the middle predicted probability** for each class when all predicted probabilities are sorted.
    - ✓ For example, if the median for class 1 is 0.03, it means that 50% of the predicted probabilities for class 1 are below 0.03 and 50% are above it.
  - ➢ The box plot shows the summary statistics: **median, quartiles, IQR, and outliers**.
  - ➢ It focuses on specific data points and **how spread out the data is**, particularly focusing on **central tendencies (median and mean).**
  - ➢ **Violin plot** Combines elements of a **box plot and a density plot.**
  - ➢ In the box plot, "**6 outliers**" means there are **six predicted probabilities (samples) for class 3 that are far from the typical range,** indicating unusual predictions for that class.
  - ➢ **High (75th percentile), meaning 75% of the data falls below this value. Check the violin plot slide for more info.**

Evaluation Plot

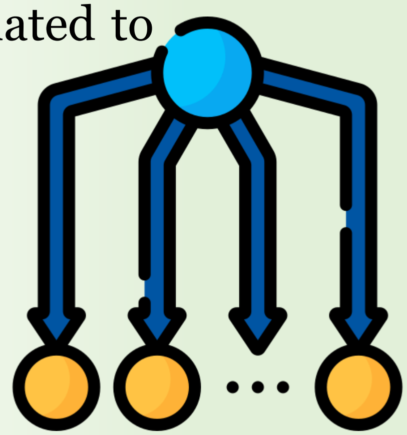
- **Box Plot**



Classifiers

- **Naïve Bayes (NB)**
  - Naive Bayes is a **simple probabilistic classifier** based on applying **Bayes' theorem** with the assumption that **features are independent of each other (hence "naive").**
  - It **calculates the probability of a class given the input features and chooses the class** with the highest probability.
  - Despite the **strong independence assumption, it works surprisingly well** in practice.
  - Naive Bayes is **fast, efficient**, and works well with **small datasets**, especially for **text classification and categorical data**.
  - It's commonly used in **spam filtering**, **document classification**, **sentiment analysis**, and **medical diagnosis due to its simplicity and efficiency.**
    - ✓ The **sentiment** typically refers to the **emotional tone or attitude expressed in text**, such as **positive, negative, or neutral** feelings.
    - ✓ **Sentiment analysis** often focuses on determining the **overall attitude (e.g., favorable or unfavorable) rather than specific emotions like anger, joy, or sadness.**
  - **Sentiment is a simple form of emotion with two or three classes and looks for positive or negative aspects.**

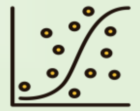
Bayes' Theorem describes the probability of an event based on prior knowledge of conditions related to that event. It states that:



$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

Where:

- $P(A \mid B)$ is the probability of event A occurring given that B is true (posterior probability).
- $P(B \mid A)$ is the probability of B given A (likelihood).
- $P(A)$ is the prior probability of A.
- $P(B)$ is the total probability of B.

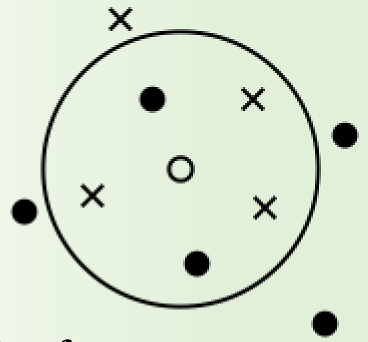


Classifiers

- **K-Nearest Neighborhood (KNN)**
  - ➢ k-Nearest Neighbors (k-NN) is a simple, instance-based learning algorithm that stores all available cases and classifies new cases based on a **similarity measure (e.g., distance functions).**
  - ➢ It is a type of **lazy learning where the function is only approximated locally**, and all computation is delayed until classification.
    - ✓ "Lazy" learning refers to algorithms that delay the generalization process until a prediction is requested, rather than building a model during training.
    - ✓ **Lazy learning can make the training process faster because it skips building a predictive model during training.**
  - ➢ As a classifier, k-NN works by **finding the most common class among the k-nearest neighbors** of the data point it is trying to predict.
  - ➢ The **choice of 'k' and the distance metric are crucial in determining the accuracy** of the classifier.
  - ➢ Some applications: **Recommendation Systems, Medical Diagnosis, Forecasting, and Image Recognition**.

Euclidean Distance: This is the most common distance metric, often used when the data points are continuous. It is calculated as:

$$d(\mathbf{p},\mathbf{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$



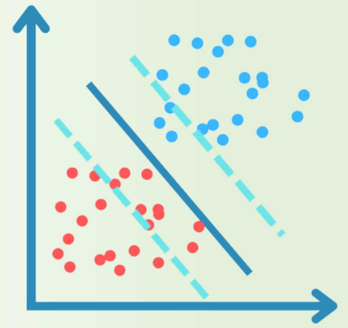
where **p** and **q** are two points in Euclidean n-space, and n is the number of dimensions (features) of the points.

Classifiers

- **Support Vector Machine (SVM)**
  - ➢ Support Vector Machine (SVM) is a powerful and versatile machine learning algorithm used for **classification and regression**.
  - ➢ It works by finding the **hyperplane that best separates different classes in the feature space with the maximum margin**, **which is the distance between the hyperplane and the nearest data points from each class.**
  - ➢ **These nearest points are known as support vectors.**
  - ➢ **Some applications:**
    - ✓ **Image Classification**: SVMs are used to categorize images into predefined categories, effectively handling high-dimensional spaces.
    - ✓ **Text and Hypertext Categorization**: SVMs help in classifying text for applications like spam detection and sentiment analysis.
    - ✓ **Bioinformatics**: SVMs are used for protein classification, cancer classification, and gene expression analysis where accuracy is critical.
    - ✓ **Handwriting Recognition**: SVMs can classify handwritten digits or letters by learning from a set of labeled examples.

SVM Equation: The fundamental idea of SVM is to solve the following optimization problem to find the hyperplane:

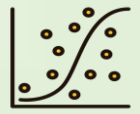
$$\min_{\mathbf{w},b} \frac{1}{2} \| \mathbf{w} \|^2$$



subject to the constraints:

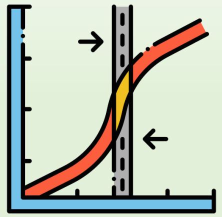
$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad \text{for all } i$$

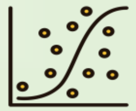
Here, $\mathbf{w}$ represents the weight vector of the hyperplane, b is the bias, $\mathbf{x}_i$ are the training samples, y_i are the class labels, and $\cdot$ denotes the dot product. This setup strives to maximize the margin between the classes, improving the classifier's generalization capabilities.



Classifiers

- **Logistic Regression (LR)**
  - Logistic Regression is a **statistical modeling technique** used primarily for **predicting binary outcomes**.
  - It employs a **logistic function to model a binary dependent variable**, where the outcome is a function of one or more independent variables.
  - The term "logistic" in logistic regression refers to the logistic function, also known as the **sigmoid function.**
  - This function outputs values between 0 and 1, effectively converting a linear combination of input features into a probability.
  - In logistic regression, if the **predicted probability p(x) is 0.5 or higher**, the model typically classifies the input x as belonging to the positive class (often coded as 1).
  - Logistic Regression is a statistical method for binary classification that **can be extended to multi-class classification via techniques like the One-vs-Rest (OvR) approach.**
    - ✓ One-vs-Rest (OvR), also known as One-vs-All, is a strategy for multi-class classification that involves training a single classifier per class, with the samples of that class as positive samples and all other samples as negatives.
  - It estimates the probabilities using a **logistic function**, which is structured to model binary dependent variables.
  - **Unlike linear regression, it models the probability of the default class** (e.g., "success" or "1") of a binary response variable based on one or more predictor variables (features). **It provides a probability in the range of 0 to 1.**
  - Some applications are:
    - ✓ **Medical Fields: Predicting the likelihood of a patient having a disease based on observed characteristics of the patient (like age, body mass index, symptoms, etc.).**
    - ✓ **Credit Scoring: Assessing the probability that a customer will default on a loan based on their credit history.**
    - ✓ **Marketing: Predicting a customer's propensity to purchase a product or halt a subscription.**
    - ✓ **Election Forecasting: Predicting the outcome of an election based on demographic data.**





Classifiers

- **Logistic Regression (LR)**

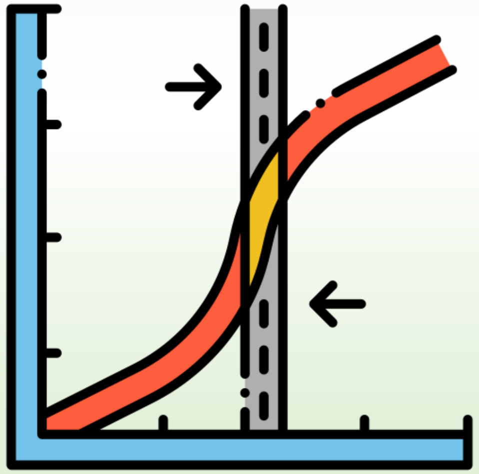
For logistic regression in a One-vs-Rest (OvR) configuration, each classifier $f_i(x)$ for class i uses a logistic function to estimate the probability that an instance x belongs to class i. The decision function for each class is given by:

$$p_i(x) = \frac{1}{1 + e^{-(\beta_{i0}+\beta_{i1}x_1+\beta_{i2}x_2+\cdots+\beta_{in}x_n)}}$$

Where $\beta_{i0}, \beta_{i1}, \ldots, \beta_{in}$ are the parameters of the logistic regression model for class i, and $x_1, x_2, \ldots, x_n$ are the feature values of instance x. For a given instance, you compute this probability for all classes and classify the instance as belonging to the class with the highest probability:

$$\text{Class}(x) = \arg\max_i p_i(x)$$

Here, $p_i(x)$ is the probability that x belongs to class i, and the arg max function selects the class with the highest predicted probability.



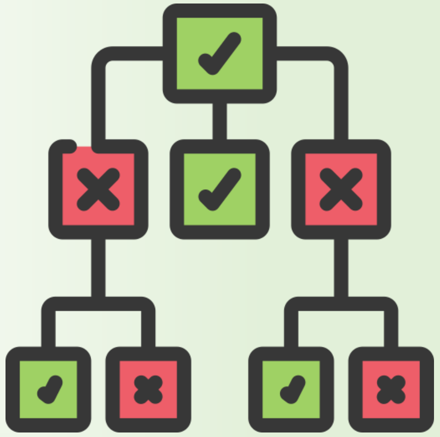
Classifiers

- **Decision Tree (DT)**
  - A Decision Tree is a **flowchart-like tree structure where an internal node represents a feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome**.
  - The **topmost node** in a decision tree is known as the **root node**.
  - It learns to partition on the basis of attribute value.
  - It partitions the tree in a manner that makes the groups as distinct as possible, with each group at any node having as pure a class as possible.
  - In classification, a Decision Tree makes decisions by splitting the data into subsets based on the value of input features.
  - These decisions are made at each node according to a criterion (like **Gini impurity or entropy**), aiming to progressively classify the data as accurately as possible until reaching the leaf nodes, which represent the classification outcomes.
  - Applications are the same and broad.

Decision Tree Equation: Decision trees do not use a single mathematical equation for classification. Instead, they use a series of decision rules that could be thought of as a sequence of if-then-else statements. However, the decision to split at each node is often based on metrics such as:

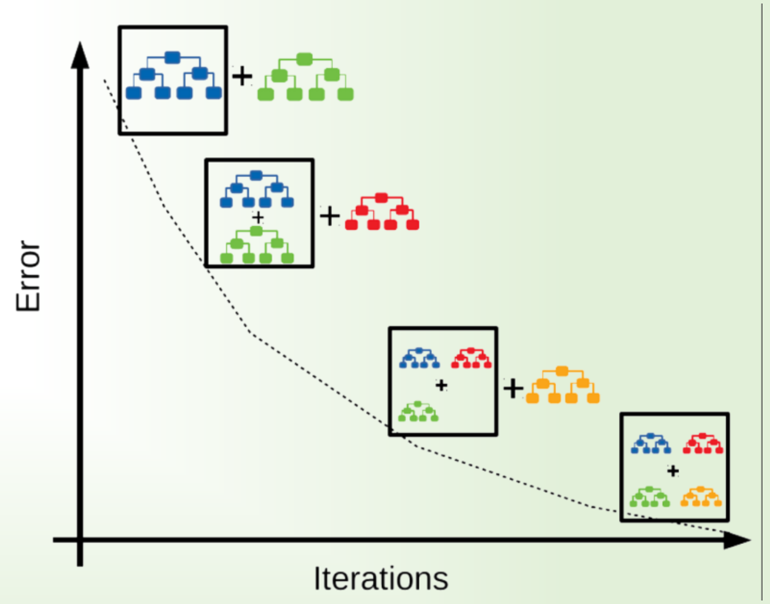
- Entropy: $H(S) = -\sum_{x\in X} p(x) \log_2 p(x)$
- GiniImpurity: $G(S) = 1 - \sum_{x\in X} p(x)^2$

where $p(x)$ is the proportion of the samples that belong to class x in set S. These metrics help determine the best feature and value to split the data at each node to achieve the most effective classification.



Classifiers

- **Gradient Boosting**
  - Gradient Boosting is a machine learning technique that **builds models incrementally in the form of an ensemble of weak prediction models, typically decision trees**.
  - It **optimizes a loss function over the space of possible models** in a stage-wise fashion: **it builds a model, then builds subsequent models that correct the errors of the previous ones**.
  - **The models are added together to make the final prediction, which is more accurate and robust than any individual model.**
  - **In classification, Gradient Boosting works by sequentially adding weak learners (usually decision trees) to the ensemble, each correcting its predecessor by focusing more on the instances that were incorrectly classified.**
  - The **output for the ensemble model is then the weighted sum of the predictions made by the individual models**, where weights are determined by how well each model corrects the residual errors made by the previous models.
  - Applications are the same and broad.
  - **That is why it takes a long time. A lot of DTs.**



Gradient Boosting Equation: Gradient Boosting constructs the final model $F(x)$ by adding up a sequence of K weak models $h_k(x)$, each scaled by a coefficient α_k :

$$F(x) = \sum_{k=1}^{K} \alpha_k h_k(x)$$

Here, each $h_k(x)$ typically represents a decision tree, and α_k is the contribution of the k-th tree to the final prediction. The values for α_k and the structure of each $h_k(x)$ are determined during the training process, where the algorithm minimizes the loss function by gradient descent, adjusting the weights and refining the models to better fit the data.

Classifiers

- **eXtreme Gradient Boosting (XGBoost)**
  - XGBoost is an **advanced implementation of gradient boosting** that is specifically designed to be **more efficient, flexible, and portable**.
  - It uses a **more regularized model formalization to control over-fitting**, which makes it highly effective.
  - XGBoost **optimizes** both the **computational speed and model performance** by employing **techniques such as tree pruning, hardware optimization, and parallel processing**.
  - XGBoost **enhances the gradient boosting method by introducing more regularization parameters to reduce overfitting**.
  - **Regularization** is a **technique used to prevent overfitting by penalizing models for their complexity**.
  - **This involves adding terms to the loss function that increase with the number of parameters or the magnitude of parameter values. Then, after a threshold, the penalty happens, which is to avoid or prevent that operation.**
  - In classification tasks, it **builds an ensemble of decision trees incrementally**.
  - **Each new tree corrects errors made by the previously built trees**, and the model makes predictions by summing the outputs of these trees, weighted by their contribution to reducing the overall prediction error.

XGBoost Equation: The final model in XGBoost, similar to gradient boosting, is given by:

$$F(x) = \sum_{k=1}^{K} \alpha_k h_k(x)$$

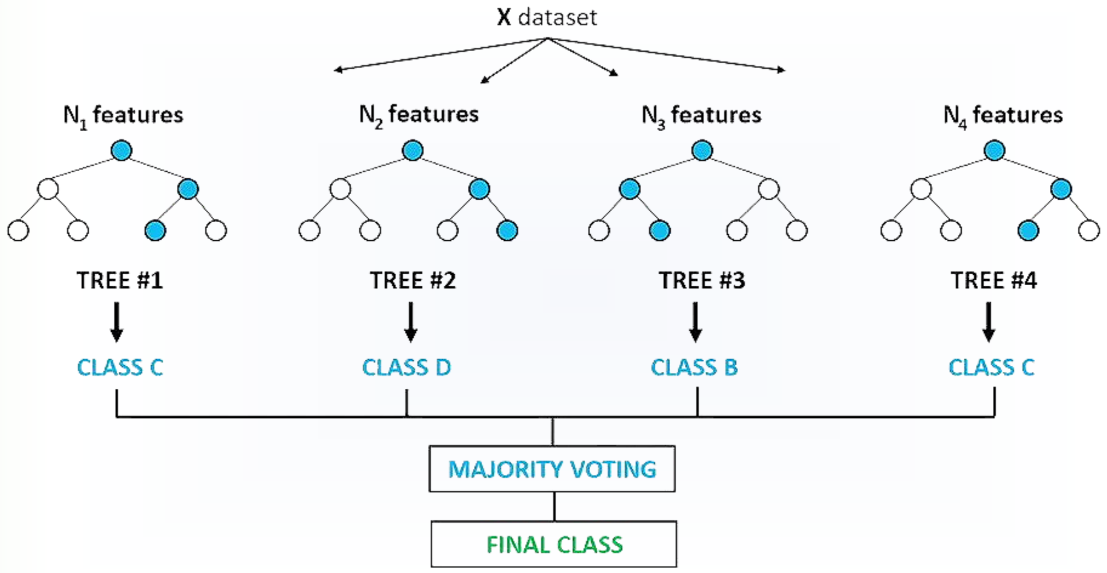
where $h_k(x)$ are the decision trees, and α_k are their corresponding weights in the ensemble. What sets XGBoost apart is its objective function, which includes a regularization term:

$$\text{Obj}(\Theta) = \sum_{i=1}^{n} l\big(y_i, F(x_i)\big) + \sum_{k=1}^{K} \Omega(h_k)$$

Here, l is a differentiable convex loss function that measures the difference between the predicted and actual values for the n training samples, and Ω is the regularization term that penalizes the complexity of the model h_k, effectively controlling overfitting.

Classifiers

- **Random Forest (RF)**
  - Random Forest is an **ensemble** learning method that **combines multiple decision trees** to improve prediction accuracy and control **overfitting**.
  - **Each tree in the forest is built from a random subset of data and features**, and the final prediction is made by averaging the predictions (for regression) or voting (for classification) across all trees.
  - An **ensemble** is a method that **combines multiple models to produce a more accurate or robust prediction** than any individual model.



Random Forest itself doesn't have specific equations like a regression model since it's a nonparametric method. However, its operation involves:

1. Tree Building: Each tree is grown using the formula for calculating split points in decision trees (e.g., Gini impurity or entropy in classification).
2. Aggregation: For classification, the final prediction is typically made by majority voting:

$$\text{Class Label } = \arg\max \sum_{i=1}^{N} \mathrm{I}(y_i = c)$$

where y_i is the prediction of the i-th tree, c represents a class label, and N is the number of trees.

Classifiers

- **AdaBoost**
  - AdaBoost (**Adaptive Boosting**) is an **ensemble** technique that improves the accuracy of classifiers by **combining multiple weak classifiers** into a strong classifier.
  - It **iteratively adjusts the weights of incorrectly classified instances** so that subsequent classifiers focus more on difficult cases.
  - In classification, AdaBoost **assigns a weight to each training example and updates these weights after each classifier is trained, emphasizing harder-to-classify instances**. The final output is a weighted vote of all the classifiers' predictions, enhancing the model's ability to generalize.

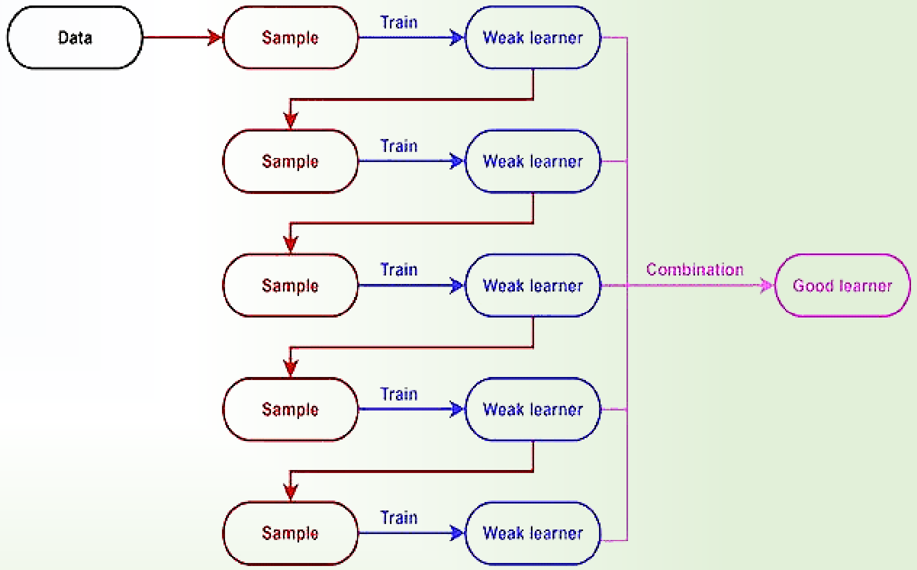
1. **Weight Update**: Weights are updated based on the accuracy of the current classifier, using the formula:

$$w_{i,t+1} = w_{i,t} \cdot e^{-\alpha_t y_i h_t(x_i)}$$

where $w_{i,t}$ is the weight of the i-th instance at iteration t, α_t is the weight of the classifier, y_i is the true class label, and $h_t(x_i)$ is the prediction of the classifier.

2. **Classifier Weight**: The weight α of each classifier is calculated based on its error rate ϵ :

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$



3. **Final Prediction**: The overall prediction is given by the sign of the weighted sum of the classifier outputs:

$$\text{Prediction } = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

where T is the total number of classifiers.

Advanced Techniques and Analysis

- **Leave One Participant Out (LOPO)**
  - Leave-One-Participant-Out (LOPO) cross-validation is a method where the data from one participant is left out for testing while the model is trained on data from all other participants.
  - It is typically used in scenarios where data is grouped by participants, such as in medical or human activity studies, **to ensure the model generalizes across different individuals**.
  - LOPO is crucial when the goal is to **evaluate how well a model performs on unseen participants, preventing overfitting to specific individuals.**
  - It is especially important when there are concerns that training on one participant's data might not generalize well to others.
  - **Cross-validation** is a technique used to assess a model's performance by **dividing the dataset into multiple parts, training on some parts, and testing on others, then averaging the results.**
  - In this context, it helps ensure that the model generalizes well to unseen data by repeatedly training and testing on different splits, such as in K-Fold or Leave-One-Participant-Out approaches, avoiding overfitting to a specific dataset portion.

Advanced Techniques and Analysis

- **Leave One Participant Out (LOPO)**

If we have P participants:

1. Split the dataset by participants into subsets $D_1, D_2, \ldots, D_P$, where D_i is the data for participant i.
2. For each participant i :

- Train the model on the data of all participants except i :

$$\text{Training set } = \bigcup_{j \neq i} D_j$$

- Test the model on participant i 's data:

$$\text{Test set } = D_i$$

3. Calculate the evaluation metric (e.g., accuracy) for each fold (participant left out) and then average the results:

$$\text{Final Accuracy } = \frac{1}{P} \sum_{i=1}^{P} \text{Accuracy }_i$$

Where:

- P is the number of participants.
- Accuracy $_i$ is the model's accuracy when participant i 's data was used for testing.

- **Leave-One-Out Cross-Validation (LOO)**
  - Leave-One-Out Cross-Validation (LOO) is a method where **each data point is used as the test set once, while the remaining points form the training set**.
  - This process is **repeated for each data point in the dataset, making it computationally expensive for large datasets**.
  - LOO is useful **when you have a small dataset** and need to evaluate the model's performance on every single sample.
  - It helps **avoid overfitting** while providing a very thorough evaluation of the model.
  - It is typically used in fields like medicine or psychology, where data is often limited.

If the dataset has N samples:

1. For each sample i in the dataset:

- Train the model on the remaining $N-1$ samples:

$$\text{Training set } = D - \{x_i\}$$

- Test the model on the left-out sample x_i :

$$\text{Test set } = \{x_i\}$$

2. Calculate the evaluation metric (e.g., accuracy) for each fold where one sample is left out, and then average the results:

$$\text{Final Accuracy } = \frac{1}{N}\sum_{i=1}^{N} \text{Accuracy }_i$$

Where:

- N is the total number of samples.
- Accuracy $_i$ is the model's accuracy when the i-th sample is used as the test set.

Advanced Techniques and Analysis

- **K-Fold Cross-Validation**
  - K-Fold Cross-Validation **splits the dataset into K equally sized folds**, where the model is **trained on K-1 folds and tested on the remaining fold**.
  - This process is **repeated K times**, with **each fold used exactly once as the test set**.
  - It provides a more balanced assessment of model performance by averaging the results across all folds.
  - K-Fold is commonly used when you want a more efficient way to evaluate model performance **without leaving out too much data for testing**.
  - It's widely used in machine learning for **generalization testing**, **especially with larger datasets (because we have a large portion out for the test)**. That means the more fold, the slower the computation.
  - K-Fold Cross-Validation is **one of the most commonly used and basic cross-validation techniques**.
    - ✓ So if we have 200 samples, 4 folds mean 50 samples for a test, so the test is four times, but 10 folds mean 20 samples for a test, so more training and testing and more computation.

Advanced Techniques and Analysis

- **K-Fold Cross-Validation**

For a dataset with N samples and K folds:

1. Split the dataset into K equal-sized subsets (folds):

$$D = \{D_1, D_2, \ldots, D_K\}$$

2. For each fold k :

- Train the model on $K - 1$ folds:

$$\text{Training set } = D - D_k$$

- Test the model on the left-out fold D_k :

$$\text{Test set } = D_k$$

3. Calculate the evaluation metric (e.g., accuracy) for each fold, then average the results:

$$\text{Final Accuracy } = \frac{1}{K} \sum_{k=1}^{K} \text{Accuracy }_k$$

Where:

- N is the total number of samples.
- K is the number of folds.
- Accuracy $_k$ is the accuracy of the model when fold k is used as the test set.

Advanced Techniques and Analysis

- **Stratified K-Fold Cross-Validation**
  - Stratified K-Fold Cross-Validation is a variation of K-Fold Cross-Validation that **ensures each fold has approximately the same proportion of class labels as the original dataset**.
  - This is particularly important when dealing with **imbalanced datasets**, as it **prevents some classes from being underrepresented or missing in some folds.**
  - It provides more reliable performance estimates for classification problems where **maintaining class balance in both the training and testing sets is essential.**
  - Stratified K-Fold is widely used in domains like medical data analysis or fraud detection, where class imbalance is common.
  - **So, basically, it is the same as k-fold cross-validation but for an imbalanced dataset.**
  - **So, if the dataset is balanced, the result is exactly as k-fold cross-validation.**

Advanced Techniques and Analysis

- **Stratified K-Fold Cross-Validation**

For a dataset with N samples, K folds, and C classes:

1. Stratify the dataset into K folds, ensuring that each fold maintains the same class distribution as the entire dataset: $D = \{D_1, D_2, \ldots, D_K\}$, where each D_k has the same proportion of each class as the original dataset.
2. For each fold k :

- Train the model on $K - 1$ folds:

$$\text{Training set } = D - D_k$$

- Test the model on the left-out fold D_k :

$$\text{Test set } = D_k$$

3. Calculate the evaluation metric (e.g., accuracy) for each fold, then average the results:

$$\text{Final Accuracy } = \frac{1}{K} \sum_{k=1}^{K} \text{Accuracy }_k$$

Where:

- N is the total number of samples.
- K is the number of folds.
- C is the number of classes.
- Accuracy $_k$ is the accuracy of the model when fold k is used as the test set.

Advanced Techniques and Analysis

- **Repeated K-Fold Cross-Validation**

  - Repeated K-Fold Cross-Validation is an **extension of K-Fold Cross-Validation, where the K-Fold process is repeated multiple times with different random splits of the data**.
  - After performing K-Fold Cross-Validation (where the dataset is split into K folds and evaluated), the **dataset is shuffled again, and K-Fold is repeated, often several times**.
  - This introduces randomness and ensures more varied train-test splits.
  - It is often used when you want **more confidence in your model's performance on unseen data**, especially **if the dataset is small** or if you're looking for more generalized performance metrics.
  - The advantage of Repeated K-Fold Cross-Validation is that **it reduces the variance in performance estimates by averaging results over multiple runs**, providing a more reliable and robust evaluation of model performance, especially for small or noisy datasets.

Advanced Techniques and Analysis

- **Repeated K-Fold Cross-Validation**

For a dataset with N samples, K folds, and R repetitions:

1. Repeat K-Fold Cross-Validation for R repetitions. In each repetition:

- Split the dataset into K folds:

$$D = \{D_1, D_2, \dots, D_K\}$$

2. For each repetition r and fold k :

- Train the model on $K-1$ folds:

$$\text{Training set } = D - D_k$$

- Test the model on the left-out fold D_k :

$$\text{Test set } = D_k$$

3. Calculate the evaluation metric (e.g., accuracy) for each fold and average over the K folds for each repetition:

$$\text{Accuracy}_r = \frac{1}{K}\sum_{k=1}^{K} \text{Accuracy}_{r,k}$$

4. Finally, average the results across all R repetitions:

$$\text{Final Accuracy } = \frac{1}{R}\sum_{r=1}^{R} \text{Accuracy}_r$$

Where:

- N is the total number of samples.
- K is the number of folds.
- R is the number of repetitions.
- Accuracy $_{r,k}$ is the accuracy of the model when fold k is used as the test set during repetition r.

Advanced Techniques and Analysis

- **Nested Cross-Validation**

  - Nested Cross-Validation is a technique used to **evaluate model performance while simultaneously tuning hyperparameters in a nested manner**.
  - **Nested** means **something that is contained within another**.
  - In general, it refers to a structure **where one element or process is placed inside another, creating layers**.
  - For example, in programming, a nested loop is a loop inside another loop, and in any system, a nested structure means one part is enclosed within another part.
  - So, the outer loop focuses on evaluation, while the inner loop focuses on finding the best hyperparameters.
  - The term "nested" refers to this two-level structure, where one cross-validation process is embedded (or nested) inside another.
  - **So, the inner loop (hyperparameter tuning) is nested inside the outer loop (model evaluation).**
  - It involves two cross-validation loops:
    - ✓ **The outer loop** is used to **evaluate model performance on unseen data**.
    - ✓ **The inner loop** is used for **hyperparameter tuning** or model selection.
  - ✓ The advantage of Nested Cross-Validation is that it **avoids overfitting during hyperparameter tuning**, providing an unbiased estimate of the model's performance.
  - ✓ **This method is commonly used when performing hyperparameter optimization, especially in complex models like SVMs or deep learning, where hyperparameter selection can significantly impact performance.**
  - These hyperparameters control the **complexity, learning speed, and regularization** of the XGBoost model.
  - For instance, in the code, the following happened:
    - ✓ The following hyperparameters are tuned in the **GridSearchCV** algorithm
    - ✓ **N-estimators:** The number of trees (e.g., [50, 100, 150]).
    - ✓ **learning_rate**: The step size at each boosting iteration (e.g., [0.01, 0.1, 0.2]).
    - ✓ **max_depth**: The maximum depth of each tree (e.g., [4, 6, 8]).
    - ✓ **subsample**: The fraction of samples used for each tree (e.g., [0.7, 0.8, 1.0]).
    - ✓ **colsample_bytree**: The fraction of features used for each tree (e.g., [0.7, 0.8, 1.0]).

Advanced Techniques and Analysis

- **Nested Cross-Validation**

1. Outer Loop (K outer folds):

- Split the dataset into K outer folds:

$$D = \{D_1, D_2, \dots, D_K\}$$

2. For each outer fold k :

- Train the model on $K-1$ outer folds:

$$\text{Outer Training set } = D - D_k$$

- Perform Inner Loop Cross-Validation to find the best hyperparameters.

3. Inner Loop (H inner folds):

- Split the outer training set into H inner folds for hyperparameter tuning:

$$D' = \{D'_1, D'_2, \dots, D'_H\}$$

- For each inner fold h :
- Train the model on $H-1$ inner folds and test on the h-th fold.
- Evaluate hyperparameters based on the performance in the inner loop.
- Select the best hyperparameters from the inner loop.

4. Test on Outer Fold:

- Once the best hyperparameters are found, test the model on the left-out outer fold D_k :

$$\text{Outer Test set } = D_k$$

5. Calculate the evaluation metric (e.g., accuracy) for each outer fold and average the results across all outer folds:

$$\text{Final Accuracy } = \frac{1}{K}\sum_{k=1}^{K} \text{Accuracy}_{\,k}$$

Where:

- K is the number of outer folds.
- H is the number of inner folds.
- Accuracy $_k$ is the accuracy of the model when fold k is used as the outer test set.

Advanced Techniques and Analysis

- **Bootstrapping**

  - Bootstrapping is a statistical technique that involves **randomly sampling a dataset with replacement to create multiple new datasets**, called "bootstrap samples."
  - Each bootstrap sample may **include duplicates of some data points and exclude others**.
  - **Bootstrapping is particularly effective when your dataset is small. In such cases, K-Fold Cross-Validation may not give enough variability because the data is divided into fewer training/test splits. At the same time, Bootstrapping allows for more resampled versions of the data to be created. In Bootstrapping, sampling with replacement means that the same data point can be selected more than once in a single bootstrap sample.**
  - So, each bootstrap sample is a random subset of the original dataset, **where some samples may appear multiple times, and others might be left out.**
  - Higher accuracy from Bootstrapping us not fully reflect the true generalization performance of your model on unseen data. **Since some data points are repeated across training and testing sets.**
  - **Small datasets**: Helps **estimate the uncertainty or variability** in small datasets when traditional methods might fail.
  - **Estimating confidence intervals**: Used to derive more reliable confidence intervals when analytical formulas are hard to apply.
  - **Any model**: Bootstrapping can be applied to any statistical or machine learning model to assess the robustness of model estimates.
  - In Bootstrapping, you're **creating new samples by randomly selecting data points with replacements from the original dataset**.
  - **These new samples are close to the original data but not exactly the same.**

Advanced Techniques and Analysis

- **Bootstrapping**

  - ➢ You're essentially **generating multiple versions of your dataset by rearranging and reusing the same data points**.
  - ➢ However, you're doing this **to estimate the uncertainty or variability in your results (e.g., mean, accuracy). For instance:**
    - ✓ Create new samples: You randomly pick numbers with replacements from the original dataset to create a new "sample."
    - ✓ For example, a new sample might look like: [1, 3, 3, 5, 4]. Here, the number 3 is repeated, and 2 is left out.
    - ✓ Repeat this: You do this many times (e.g., 1000 times) to create many new samples.
    - ✓ Each sample will look a bit different.
    - ✓ Calculate the average: For each new sample, you calculate the average.
    - ✓ So, you'll get 1000 averages (one for each new sample).
    - ✓ Look at the variation: By looking at all those averages, you can get an idea of how much the average might change if you had different data.
  - ➢ **So the main difference between bootstrapping and SDG is that bootstrapping resample and SDG generate new samples, which follow the underlying distribution of the samples.**
  - ➢ **For instance, if we have a dataset of [1, 2, 3, 4, 5], then we create 100 resamples of different variations of this like [1, 3, 3, 5, 4], but in SDG, we have [0.9, 2.2, 3.1, 3.9, 4.8].**
  - ➢ **So resample means it exists in the dataset, but SDG creates samples that are not in the dataset.**

Advanced Techniques and Analysis

- **Bootstrapping**

For a dataset with N samples and B bootstrap iterations:

1. For each bootstrap iteration b :

- Create a bootstrap sample by randomly sampling with replacement from the original dataset: $D_b = \{x_1, x_2, \ldots, x_N\},$ where some x_i may be repeated

2. Train the model on the bootstrap sample D_b.
3. Test the model on the out-of-bag (OOB) samples (samples not included in D_b) to estimate performance:

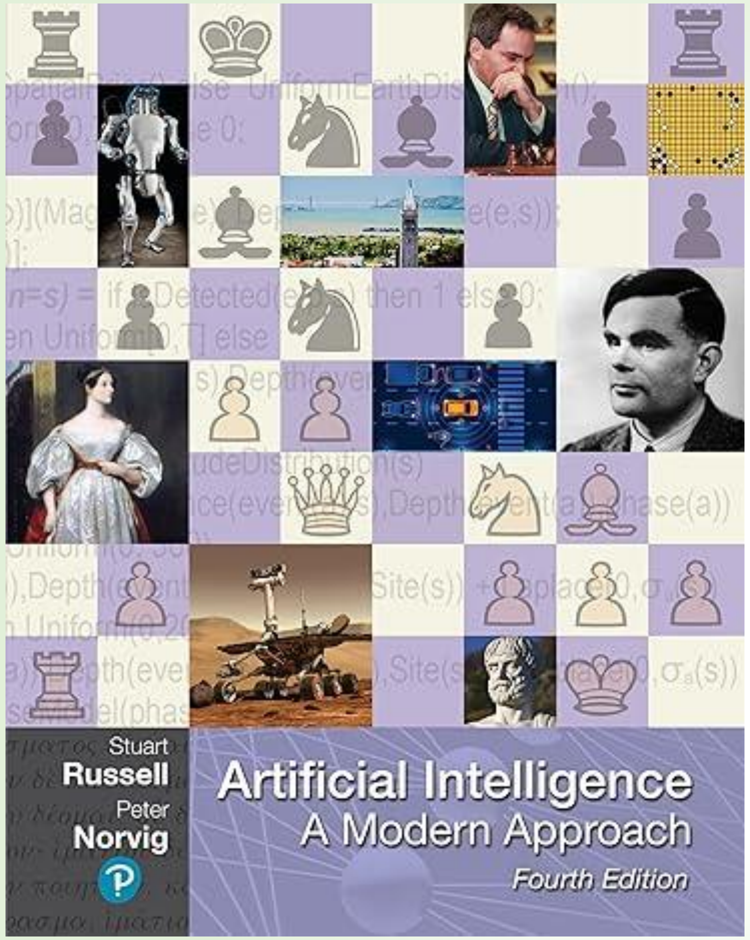
$$\text{Test set } = D - D_b$$

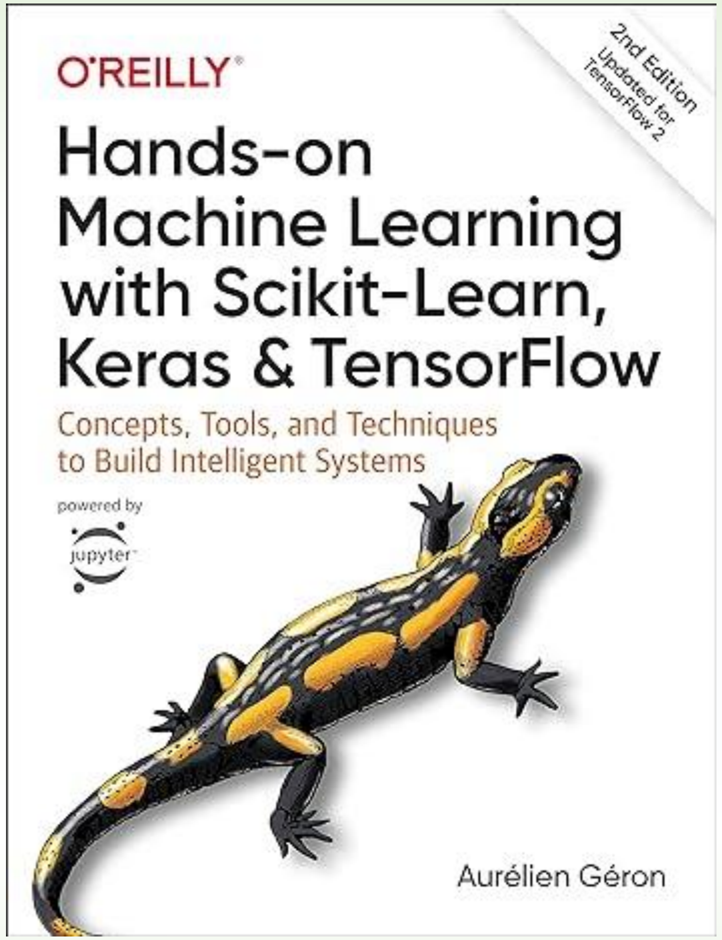
4. Calculate the evaluation metric (e.g., accuracy) for each bootstrap iteration b, and average the results:

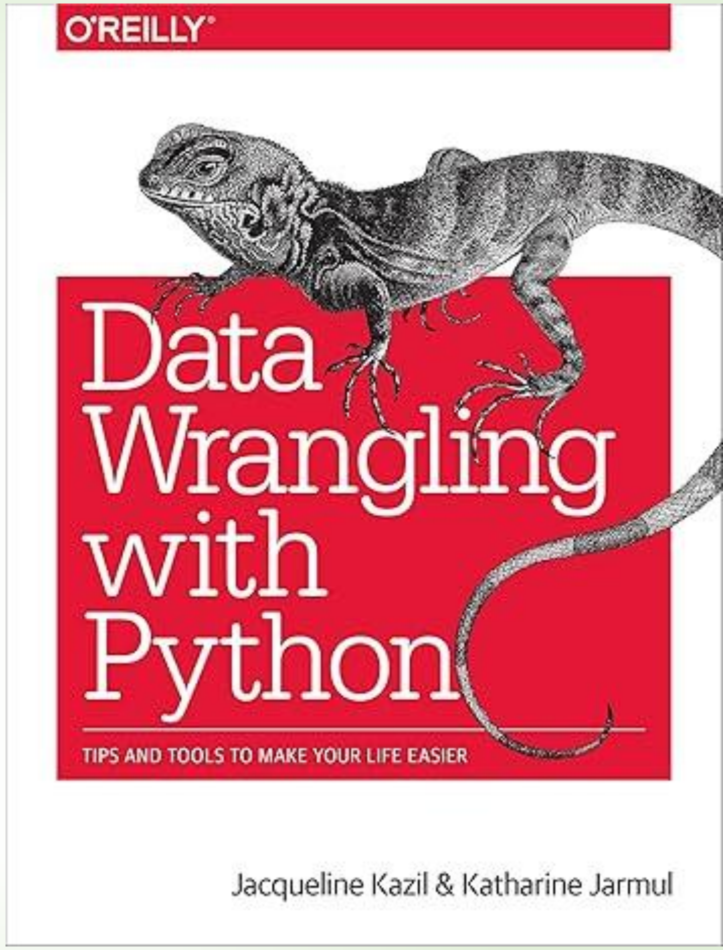
$$\text{Final Accuracy } = \frac{1}{B}\sum_{b=1}^{B} \text{Accuracy }_b$$

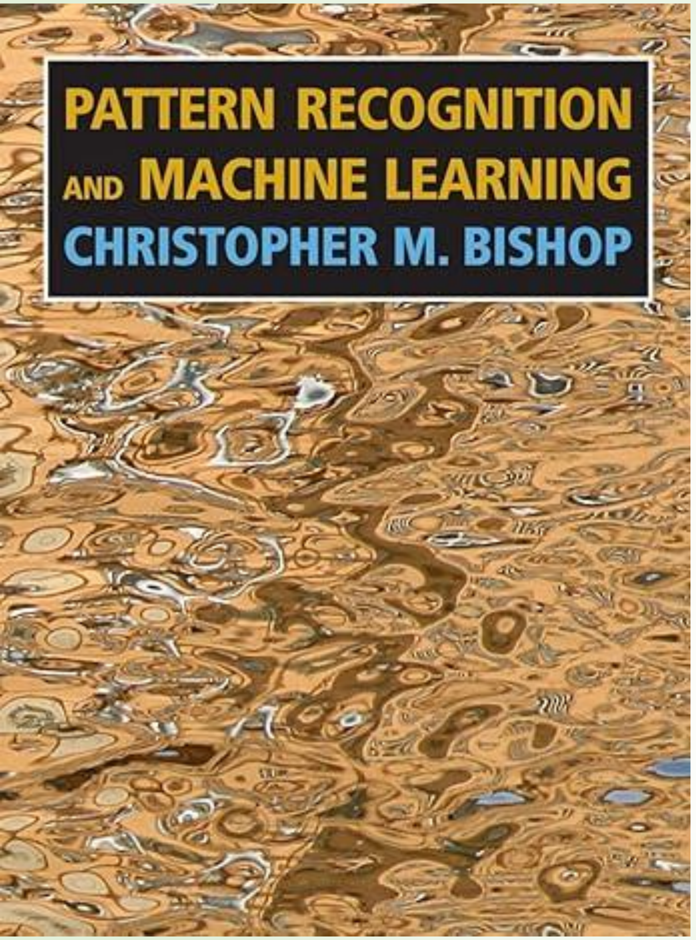
Where:

- N is the total number of samples.
- B is the number of bootstrap iterations.
- D_b is the bootstrap sample for iteration b.
- Accuracy $_b$ is the model's accuracy for bootstrap iteration b.

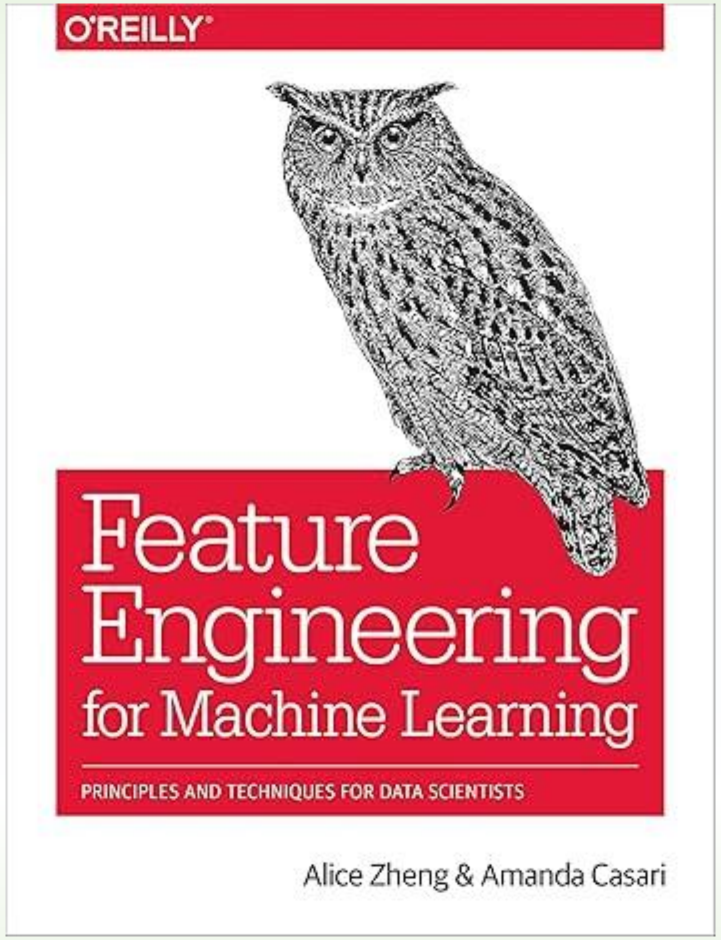

Stuart Russell Peter Norvig
Artificial Intelligence A Modern Approach
Fourth Edition


O'REILLY
2nd Edition Updated for TensorFlow 2
Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow
Concepts, Tools, and Techniques to Build Intelligent Systems
powered by jupyter
Aurélien Géron


O'REILLY
Data Wrangling with Python
TIPS AND TOOLS TO MAKE YOUR LIFE EASIER
Jacqueline Kazil & Katharine Jarmul


PATTERN RECOGNITION AND MACHINE LEARNING

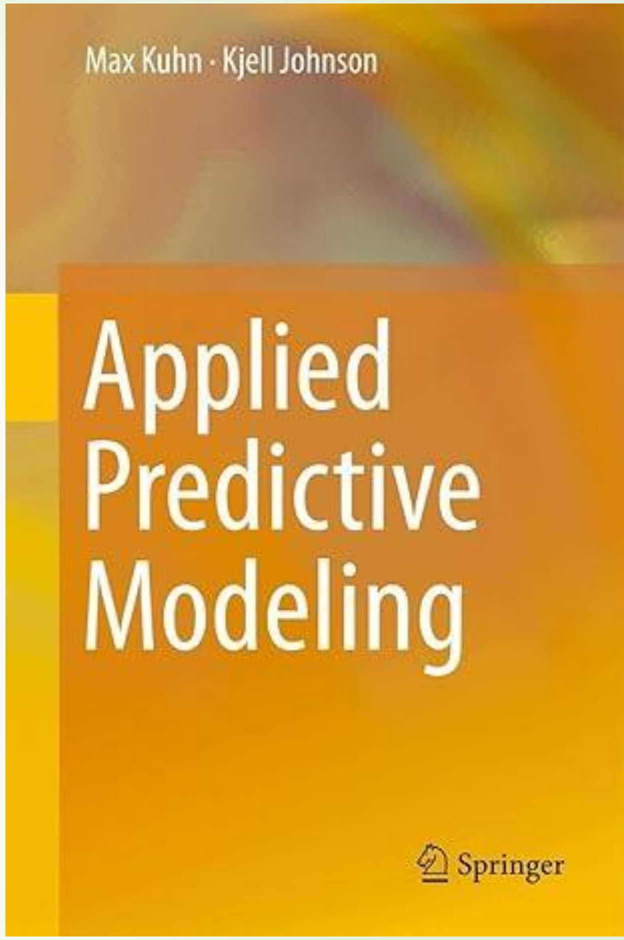
CHRISTOPHER M. BISHOP


O'REILLY

Feature Engineering for Machine Learning

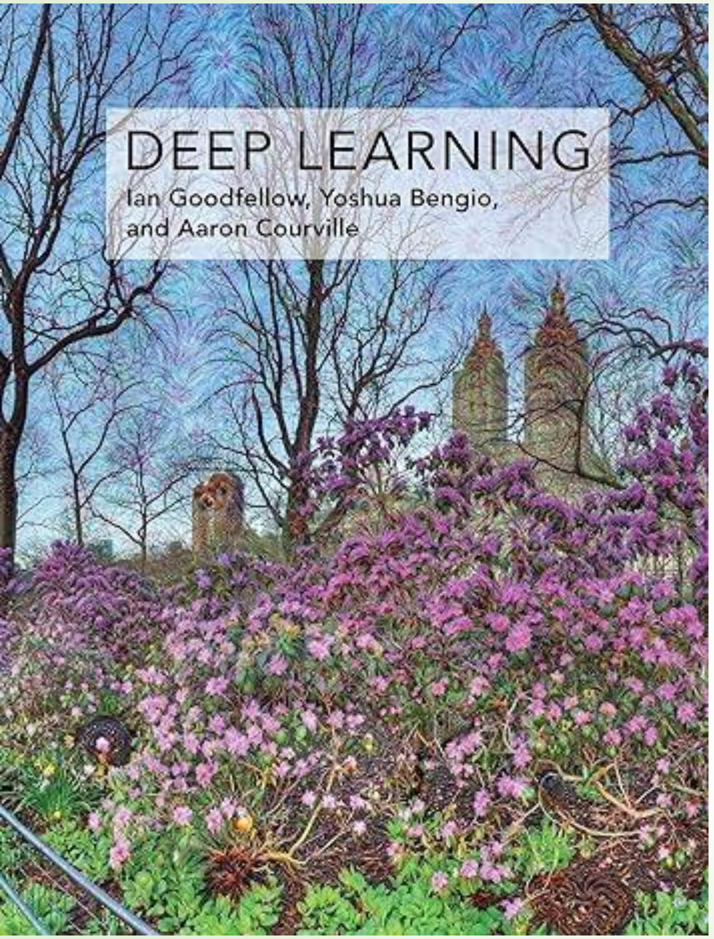
PRINCIPLES AND TECHNIQUES FOR DATA SCIENTISTS

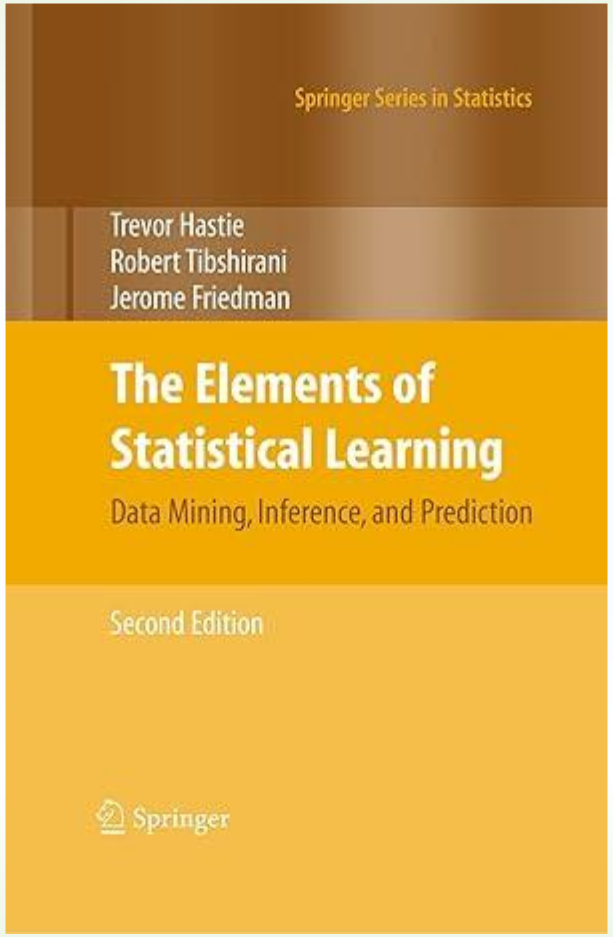
Alice Zheng & Amanda Casari

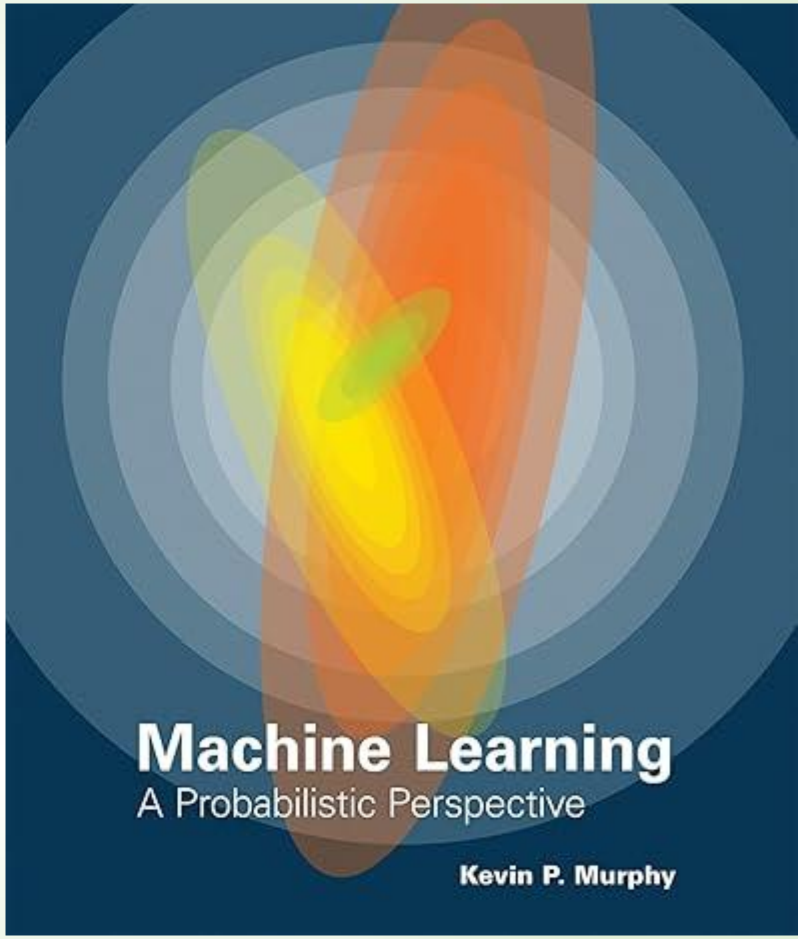

Max Kuhn · Kjell Johnson

Applied Predictive Modeling

Springer


DEEP LEARNING
Ian Goodfellow, Yoshua Bengio, and Aaron Courville


Springer Series in Statistics
Trevor Hastie
Robert Tibshirani
Jerome Friedman
The Elements of Statistical Learning
Data Mining, Inference, and Prediction
Second Edition
Springer


Machine Learning
A Probabilistic Perspective
Kevin P. Murphy