QubitReadout(SVM + KD Trees).ipynb

Description of Dataset Simulation

Allows simulation of a continuum of conditions — from well-separated, high-fidelity readouts to noisy, overlapping distributions

The function simulate_iq_dataset generates a synthetic dataset that mimics qubit readout results in the In-phase/Quadrature (I–Q) measurement space. Each sample in the dataset represents one measurement of the full *n*-qubit register, making it directly suitable for multi-qubit state classification or quantum readout calibration.
The mean vector for each multi-qubit state is constructed by concatenating the appropriate qubit-specific center vectors.
The covariance matrix cov introduces both diagonal noise and off-diagonal correlations:
noise_std defines variance along each I and Q axis, modeling readout noise.
Off-diagonal terms (set to 0.2) simulate realistic cross-qubit crosstalk correlations.
Each sample is drawn from a multivariate normal distribution using this covariance, giving clusters an elliptical shape similar to observed readout distributions in experiments.

Active Learning Protocol

The current approach is semi-supervised within an active learning framework. Leverage both labeled and unlabeled data (initial pools of data), using a mix of ground-truth and validated pseudo labels.

Initial training, test, and unlabeled pools are established via stratified sampling.
At each iteration:
Identify ambiguous samples - those with SVM decision probabilities below a tunable threshold → ensure that we are only querying those points that the SVM is truly uncertain about.
KD-tree pseudo-labeling: For ambiguous samples, apply k-nearest neighbor KD-tree classification using the current training set.
Advantage criterion (min_advantage): Pseudo-labels are added only if KD-tree accuracy on the ambiguous batch exceeds SVM's (by a user-defined margin).
SVM retraining: The training set is expanded, SVM is retrained, and test accuracy is logged.
Elements in the queried batch are removed from the unlabeled pool.
Termination: Iterations continue until the unlabeled pool is exhausted or advantage drops below threshold.

Instead of blindly trusting local geometric methods, pseudo-labels are only added if the KD-tree classifier empirically outperforms SVM on the batch of most ambiguous samples. This safeguards against model drift and ensures only genuinely informative points are incorporated. Each round refines the SVM's boundary, using feedback from both global and local structure. This iterative hybridization is rare in quantum readout ML, where most workflows commit to either model-based or local algorithms, not both.

Experimental Setup

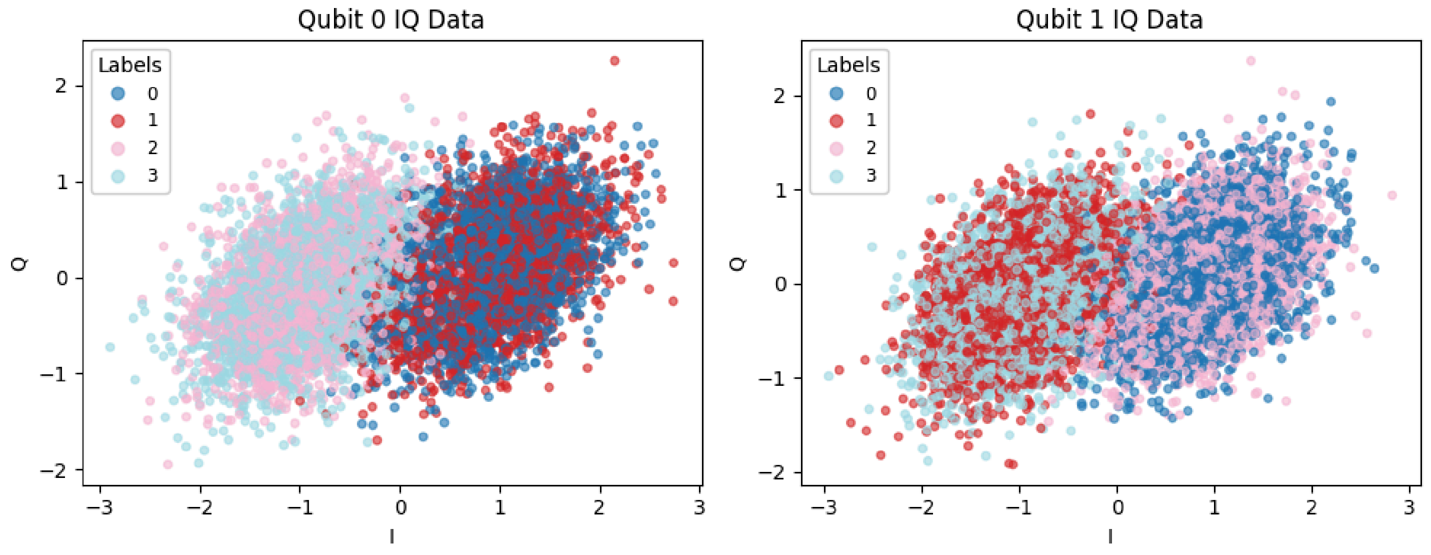
Initial accuracies of the KD-tree and SVMs are noted as a baseline.
Hyperparameters varied: initial training size, best k value, KD-tree configuration, min_advantages → Pseudo-labels from KD-tree are only added when batch-wise cross-validation demonstrates a clear advantage over SVM predictions, defending against drift and overfitting.
Two cluster separation tested (class_sep=0.5, class_sep=0.4)
Metrics: Accuracy on never-queried (unlabeled) pool, original test set, and per-iteration KD-tree advantage.
How do we know that the KD-tree targets ambiguous points?
KD-tree advantage is measured specifically on the batch of ambiguous samples identified by SVM uncertainty—not across all data.
KD-tree labels only help when, in these hardest cases, neighboring geometric structure provides more reliable assignment than the global model.

Adaptive threshold (min_advantage): By gating pseudo-labeling on measurable batch-wise improvement, you show the KD-tree is not "guessing," but focusing power exactly where the model's own uncertainty is highest.

2 Qubits

Each qubit's state (0 or 1) is represented by a 2D Gaussian cluster centered around a mean position, given by centers = {0: [center_distance, 0.1], 1: [-center_distance, -0.1]}. A low distance mimics measurement noise or hardware limitations, whereas a large distance mimics ideal, well-separated readout signals.
The SVM is classifying entire multi-qubit readout vectors—not individual qubits separately.

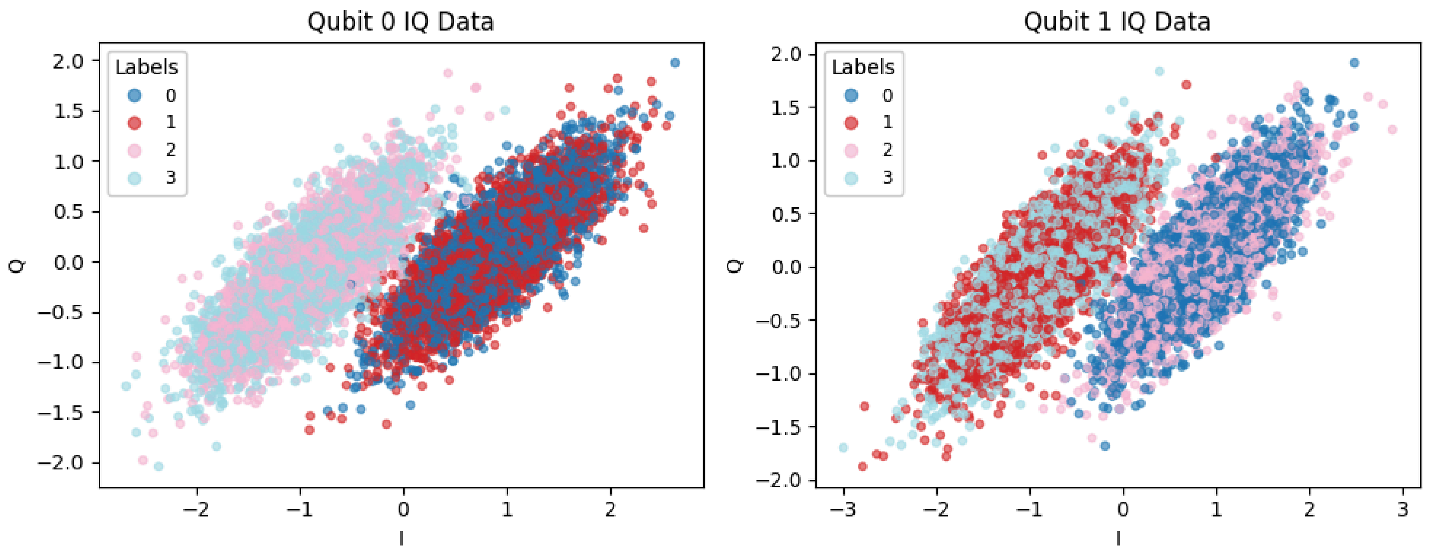
Class_separation = 0.9
noise_std=0.2
n_samples per class=3000



Initial Training Size	Best k values	Kd tree on initial training set	Kd tree on full training set	SVM init	SVM full training set	K value chosen ultimately (smallest to result in advantage)	Results of Active Learning
2000	11, 13	0.94625	**0.9455263157894737**	0.9458	**0.9454**	5	--- Active Learning Iteration 25 --- Ambiguous set size: 10 SVM acc on ambiguous: 0.500 KD-tree acc on ambiguous: 0.300 KD-tree advantage: -0.200 Advantage < 0.10 → Skipping label addition. Test accuracy: 0.9458 Training set size: 2050 | Unlabeled pool size: 7350

							KD-tree proved advantageous in: 5 iterations out of 25 Accuracy on remaining (never queried) pool: 0.9616 Accuracy on original Test set: **0.9458**

Class_separation = 0.9
noise_std=0.50
n_samples per class=3000

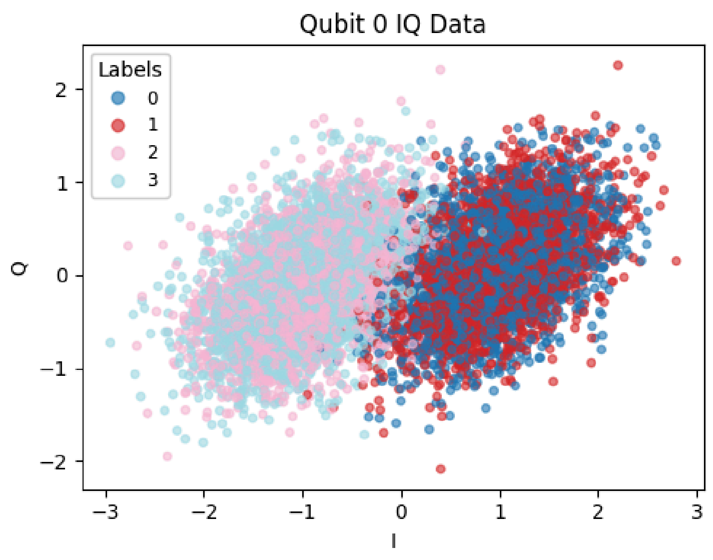


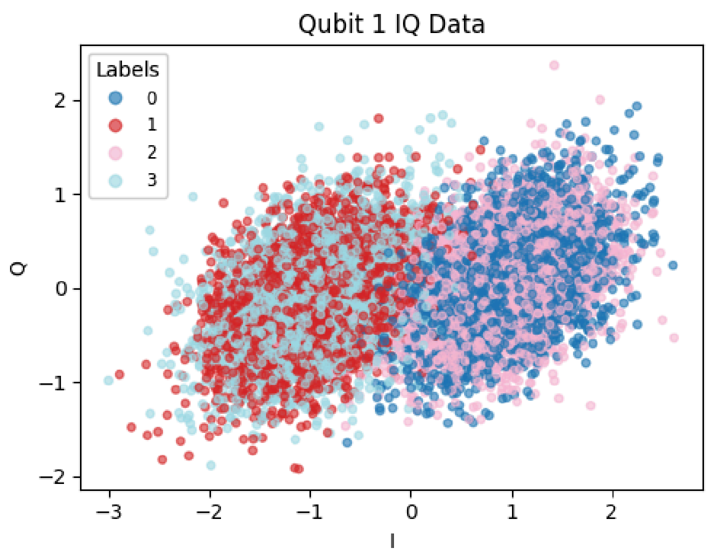
Changing the noise_std changes the cluster covariance shape and combines geometric overlap in a complex, multidimensional way.

Initial Training Size	Best k values	Kd tree on initial training set	Kd tree on full training set	SVM init	SVM full training set	K value chosen ultimately	Results of Active Learning
2000	5, 9	0.999583333333334	**0.9989473684210526**	0.9996	**0.9988**	5	--- Active Learning Iteration 3 --- No ambiguous samples below threshold. KD-tree proved advantageous in: 1 iterations out of 2 Accuracy on remaining (never queried) pool: 0.9995 Accuracy on original Test set: **0.9992**

Class_separation = 0.95
noise_std=0.20

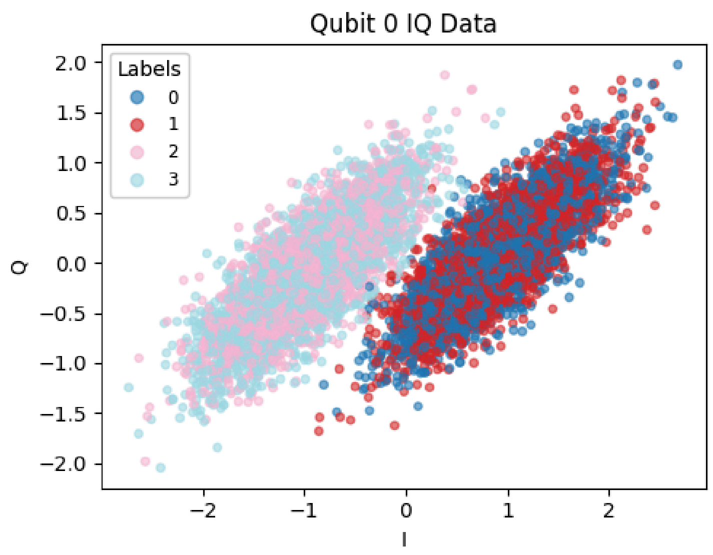
n_samples per class=3000

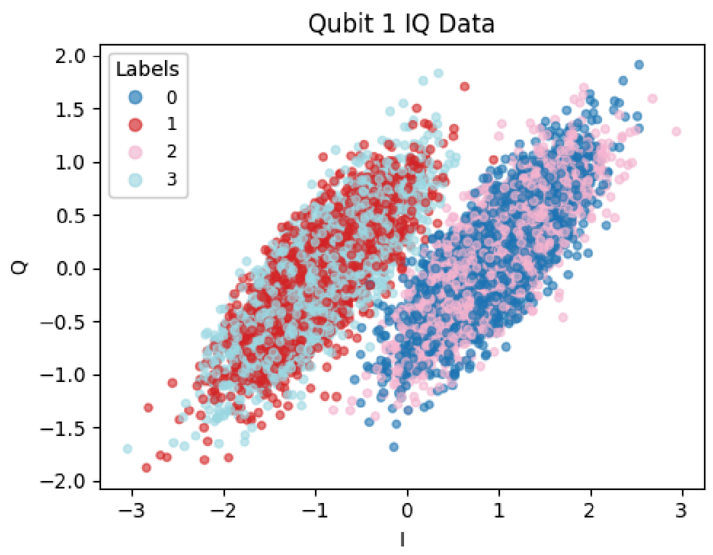




Initial Training Size	Best k values	Kd tree on initial training set	Kd tree on full training set	SVM init	SVM full training set	K value chosen ultimately	Results of Active Learning
2000	15, 15	0.95625	**0.959736842 1052631**	0.9558	**0.9592**	15	--- Active Learning Iteration 30 --- Ambiguous set size: 10 SVM acc on ambiguous: 0.400 KD-tree acc on ambiguous: 0.300 KD-tree advantage: -0.100 Advantage < 0.10 → Skipping label addition. Test accuracy: 0.9592 Training set size: 2040 | Unlabeled pool size: 7300 KD-tree proved advantageous in: 4 iterations out of 30 Accuracy on remaining (never queried) pool: 0.9744 Accuracy on original Test set: **0.9594**

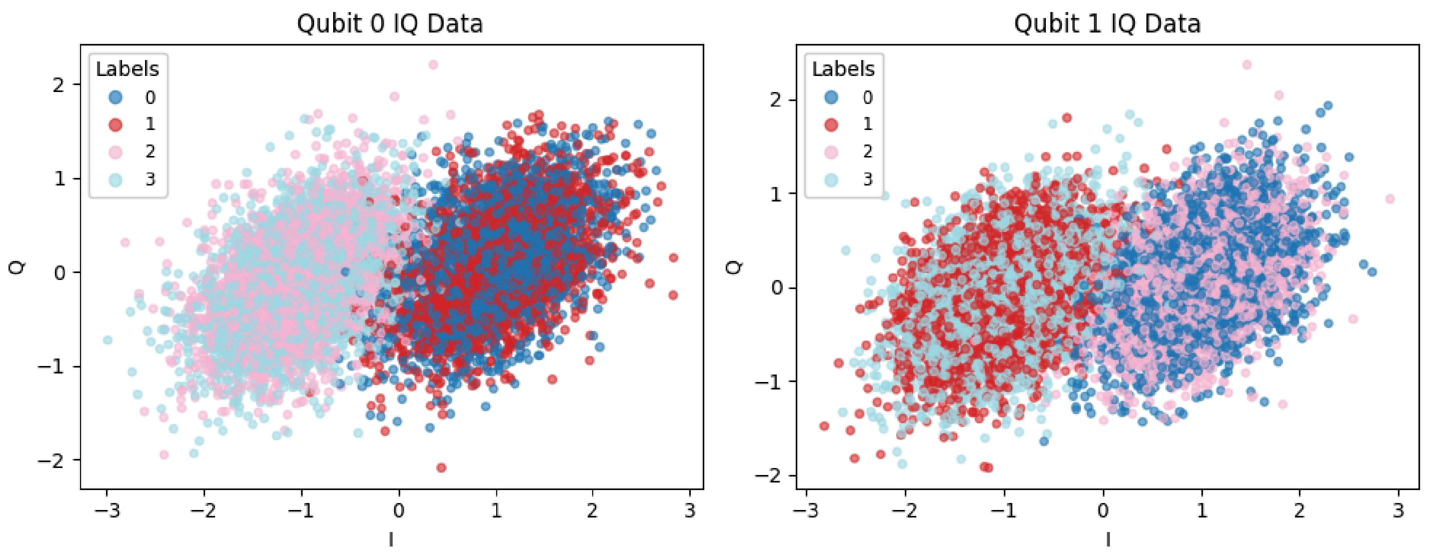
Class_separation = 0.95
noise_std=0.5
n_samples per class=3000





Initial Training Size	Best k values	Kd tree on initial training set	Kd tree on full training set	SVM init	SVM full training set	K value chosen ultimately	Results of Active Learning
2000	5, 5	1.0	**0.9994736842105263**	1.0000	**1.0000**	5	--- Active Learning Iteration 3 --- No ambiguous samples below threshold. KD-tree proved advantageous in: 1 iterations out of 2 Accuracy on remaining (never queried) pool: 0.9997 Accuracy on original Test set: **1.0000**

Class_separation = 0.99
noise_std=0.2
n_samples per class=3000



Initial Training Size	Best k values	Kd tree on initial training set	Kd tree on full training set	SVM init	SVM full training set	K value chosen ultimately	Results of Active Learning
2000	12, 16	0.969166666666666	**0.964078947368421**	0.9721	**0.9738**	14	--- Active Learning Iteration 20 --- No ambiguous samples below threshold. KD-tree proved advantageous in: 2 iterations out of 19 Accuracy on remaining (never queried) pool: 0.9776 Accuracy on original Test set: **0.9743**

Approach 1: SVM is a multi-class, one-vs-one classifier operating on multi-qubit readout vectors that represent the full quantum register state rather than any single qubit.

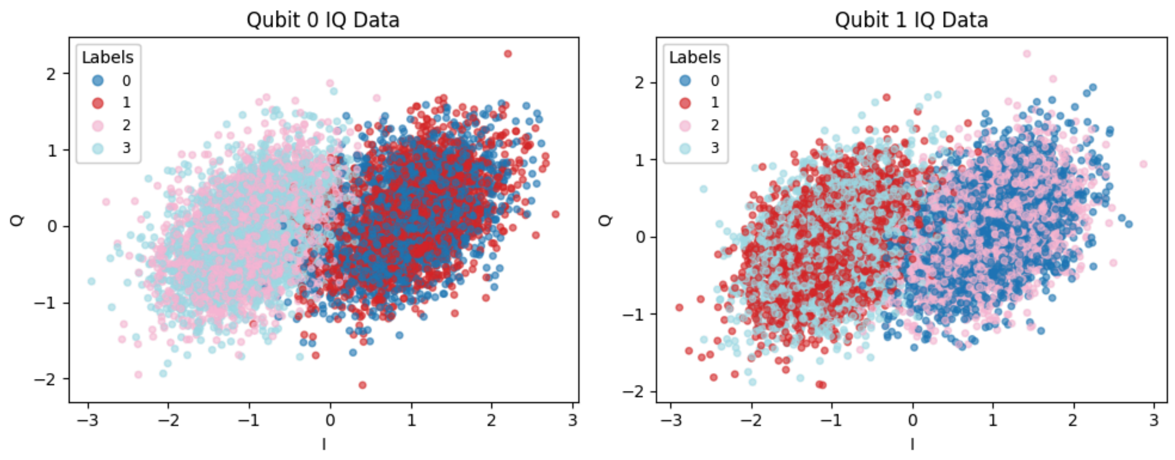
The current active_learning_with_kdtree_logging_and_advantage implements an SVM that classifies the entire multi-qubit readout vectors. Each feature vector in the the simulated dataset corresponds to the concatenated I-Q pairs from every qubit in the system, and the label (supervised learning) encodes a multi-qubit state such as: $0 \rightarrow |00\rangle$, $1 \rightarrow |01\rangle$, $2 \rightarrow |10\rangle$, $3 \rightarrow |11\rangle$. Thus, each training example represents one multi-qubit state readout (not individual qubit readouts). The SVM is implemented using sklearn.svm.svc which handles this via its one-vs-one strategy by default (for example, for 4 classes it requires 6 binary SVMs and during prediction, it performs a voting among all classifiers to determine the final label).

Scheme	Approach	Number of Classifiers	Pros	Cons

One-vs-Rest (OvR)	Build one classifier per class: distinguish “class k” vs “not class k”	N (number of classes)	Fewer classifiers, interpretable decision boundaries **Global perspective; can handle multi-class overlaps better**	In each binary classifier, the positive class (target class) is much smaller than the combined negative samples (the “rest”), causing class imbalance and poor calibration of decision boundaries
One-vs-One (OvO)	Build one classifier for each pair of classes	N(N-1)/2	Well-suited for small/medium N, usually higher accuracy for ambiguous multi-class sets	More classifiers, each sees fewer samples, more computation for high N Pairwise perspective; may be inconsistent if combined poorly

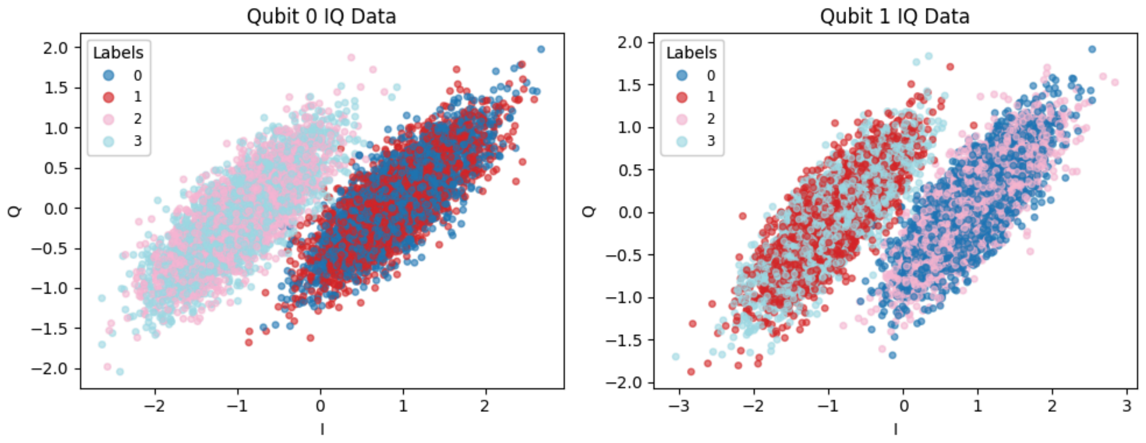
Approach 2: One-vs-Rest (OvR) SVM for Multi-Qubit Readout Classification
In the OvR scheme, the SVM trains one binary classifier per class, each learning to recognize one multi-qubit state against all others combined. During prediction, each classifier outputs a probability-like score (via predict_proba) for belonging to its chosen class. The sample is assigned the label corresponding to the classifier with the highest confidence. Each individual OvR binary classifier isolates a single quantum register state relative to all others, **clarifying which states are most easily confused.** When combined with KD-tree-based corrective augmentation and threshold-based abstention, this framework **selectively expands the classifier’s confident regions**—ideal for noisy, partially separable I–Q measurement data.

Class_separation = 0.95
noise_std=0.2
n_samples per class=3000



Initial Training Size	Best k values	Kd tree on initial training set	Kd tree on full training set	SVM init	SVM full training set	K value chosen ultimately	Results of Active Learning
2000	16, 16	0.962083333333333	**0.9588157894736842**	0.9650	**0.9650**	20	KD-tree proved advantageous in: 4 iterations out of 20 Accuracy on remaining (never queried) pool: 0.9712 Accuracy on original Test set: **0.9663**

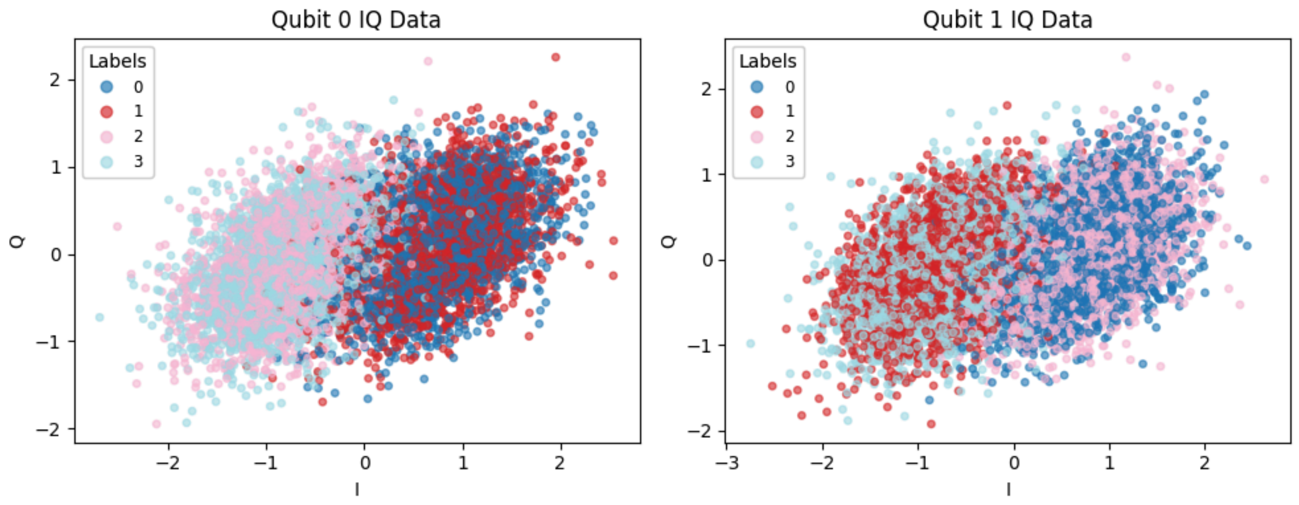
Class_separation = 0.95
noise_std=0.5
n_samples per class=3000



Initial Training Size	Best k values	Kd tree on initial training set	Kd tree on full training set	SVM init	SVM full training set	K value chosen ultimately	Results of Active Learning
2000	5, 6	1.0	**0.9992105263157894**	1.0000	**1.0000**	6	--- Active Learning Iteration 4 --- No ambiguous samples below threshold. KD-tree proved advantageous in: 1 iterations out of 3 Accuracy on remaining (never queried) pool: 0.9999 Accuracy on original Test set: **1.0000**

High Overlap/ Ambiguity When Determining the Optimal Hyperplane
Class_separation = 0.70

noise_std=0.2
n_samples per class=3000



Initial Training Size	Best k values	Kd tree on initial training set	Kd tree on full training set	SVM init	SVM full training set	K value chosen ultimately	Results of Active Learning
2000	15, 19	0.866666666666666667	**0.8688157894736842**	0.8717	**0.8738**	14	KD-tree proved advantageous in: 5 iterations out of 25 Accuracy on remaining (never queried) pool: 0.8849 Accuracy on original Test set: **0.8750**

Key Insights:

1. Introduction of a physically meaningful I–Q simulation benchmark with configurable inter-qubit noise structure.
2. Validation of a dual-model active learner that adaptively balances global margins and local geometry.
3. Evidence of bias reduction and trust-calibrated expansion of the classifier's confident region under realistic noisy conditions.

- The method defines ambiguity via probabilistic thresholds on SVM confidence.
- It performs local geometric correction (KD-tree k-neighbors) but only integrates pseudo-labels when KD accuracy exceeds the global model accuracy by a tunable min_advantage.
- This ensures pseudo-labels are adopted only when the local geometry adds genuine discriminative value, preventing drift and gradual error accumulation.

This creates an adaptive information acquisition loop, where active learning selectively expands labeled data at the SVM decision boundary, while maintaining trust calibration through the KD-tree's local density validation.
Hybrid KD-corrected OvR classification offers better explainability and adaptability for realistic N-qubit readouts with correlated measurement drift.

- High-class-separation (≥0.95): Both SVM and KD-tree achieved near-perfect consistency (≈99.9%), confirming convergence under minimal overlap.

- Low-separation (≈0.7): Active loops with KD-correction improved final accuracy by 1–2%, but, more importantly, reduced selection bias, by prioritizing ambiguous—but recoverable—regions rather than random samples.
- KD-tree advantage emerged in ≈15–25% of iterations, showing that local correction is selectively beneficial—not universal—matching the intuition that geometric similarity outperforms global hyperplanes near decision boundaries.

General Implementation Notes (Confidence-Gated Active Learning for Semi-Supervised Qubit Readout Calibration via Hybrid SVM–KD Tree Modeling)

Implementing the outline active learning framework on FPGA hardware involved the following key steps:

1. Hardware Selection and Architecture
   - FPGA Choice: Use high-performance FPGAs like Xilinx RFSoC or Intel Stratix series, capable of parallel processing and floating-point operations.
   - Core Components: Implement multiply-accumulate (MAC) units, decision logic, and data buffers to process I–Q signals, perform SVM inference, and KD-tree searches.
2. Signal Processing & Data Conversion
   - Demodulation & Quantization: Convert raw RF signals into IQ components, perform coarse digital demodulation if needed, and quantize inputs (e.g., 8-bit fixed point) for resource efficiency.
   - Preprocessing Hardware: Implement filters (matched filter) and normalization modules to prepare signals for classification.
3. SVM Inference Engine
   - Model Storage: Store trained SVM weights and biases in FPGA BRAM/ROM.
   - Fixed-Point Arithmetic: Quantize model parameters (weights, biases) to fixed-point formats matching FPGA bit widths.
   - Parallel MAC Units: Design dedicated MAC pipelines for each class's hyperplane calculation, enabling high-throughput inference.
   - Decision Logic: Implement probability or margin calculations (e.g., using logistic activation or softmax approximations) to compare confidence levels against thresholds.
4. KD-tree Search Module
   - Data Structure: Store training samples in FPGA BRAM or external memory-efficient structures.
   - Search Algorithm: Use tree traversal logic for k-nearest neighbor search—optimized via pipelined comparisons and early stopping.
   - Local Label Estimation: Use nearest neighbors to compute local class majority or confidence.
5. Active Learning Control Logic
   - Uncertainty Thresholding: Implement hardware to compare inference confidence against set thresholds, deciding whether a sample is ambiguous.
   - Pseudo-labeling Decision: If KD-tree's confidence exceeds a threshold and shows advantage, accept the local label.
6. Online/Iterative Updating
   - Model Update: After each batch, reload new hyperplane weights or adjust parameters via FPGA-embedded microcontroller cores or soft processors.
   - Data Management: Keep training data in FPGA memory for real-time KD-tree updates, or interface with external storage for larger datasets.

Need a method for selecting hyperparameters
Question Posed In Our Last Meeting: Are there certain discriminators that are sensitive to outliers?

1. query_batch_size: Number of unlabeled ambiguous samples queried per active learning iteration. Larger batches query more data but increase labeling cost; smaller batches allow finer, more frequent retraining.

2. threshold: Confidence threshold on SVM predicted probability to select ambiguous points. Lower thresholds yield fewer, more uncertain samples; higher thresholds capture broader ambiguity at risk of noisier labels.
3. k_neighbors: Number of neighbors in KD-tree for pseudo-label correction. Affects local smoothness; small k is sensitive to noise, large k may oversmooth.
4. min_advantage: Minimum KD-tree accuracy advantage over SVM on ambiguous batches to accept pseudo-labels. Prevents adding noisy corrections; tuning balances exploration vs. stability.
5. max_iterations: Limits active learning rounds; tradeoff between model refinement and labeling effort.

Use domain knowledge—e.g., known noise levels suggest initial threshold or k values.

- Noise-aware Threshold Initialization: HERQULES's matched filter outputs yield cleaner, dimension-reduced signal statistics reflecting underlying noise levels. These SNR-informed values can guide the initial setting of your SVM confidence threshold—for instance, setting the threshold around the confidence derived from matched filter performance for ambiguous states.
    - Matched Filter (MF): A matched filter per qubit projects the high-dimensional readout signal (each time bin is a separate dimension) into a lower-dimensional space. This increases signal-to-noise ratio, especially for signals affected by Gaussian noise and lacking state transitions.
    - Lightweight Neural Network: The output of the MF serves as the input to a compact neural network, which infers the qubit state with reduced computational cost and increased hardware compatibility.
    - Relaxation-Matched Filter (RMF): An additional RMF detects qubit state transitions during measurement (relaxation events), further improving overall discrimination accuracy.
- KD-tree Neighbor Count Tuning: The relaxation-matched filter enhances sensitivity to transient qubit relaxations, mapping better-defined clusters, which suggests spatial locality assumptions. This guides tuning k_neighbors to capture correlated measurement neighborhoods reflective of real hardware crosstalk and transition

Early Stopping / Convergence Criteria: Stop increasing max iterations if accuracy plateaus or advantage drops.

Meeting Notes - 10/17
Overview
The discussion centered on developing a method for selecting hyperparameters in our hybrid SVM–KD-tree active learning framework and relating these choices to the underlying data characteristics, especially noise and class discriminability.

Summary
For each cluster, the noise distribution can be modeled as Gaussian with comparable variance across clusters. The distance between cluster centroids provides a useful infidelity metric, reflecting how separable the classes are under current noise conditions. The variance-to-distance ratio between neighboring clusters indicates the intrinsic discriminability of a class; a higher ratio implies greater overlap and therefore greater classification ambiguity.

Assuming that noise scales with changes in the signal amplitude, clusters exhibiting substantially higher variance than others may not represent a single state but rather a mixture of several nearby states. Monitoring signal or trace fluctuation errors offers a natural convergence criterion: i*f observed variance fluctuations fall within the expected overlap range between two Gaussian distributions, the model has effectively converged.*

Furthermore, observing cases where the accuracy of the active learning loop decreases gradually then increases suggests that we may be trying to minimize a non-convex objective function.

From a hardware perspective, FPGA-based acceleration (as in the HERQULES readout architecture) shows strengths in performing arithmetic operations (e.g., filtered linear accumulation, or FLA). However, FPGAs are less efficient for operations involving recursive space partitioning, such as KD-tree searches or nonlinear hyperplane evaluations in SVMs.

HERQULES leverages Gaussian filtering under the assumption of static data distributions, and this filtering can be shown to be optimal in that regime.