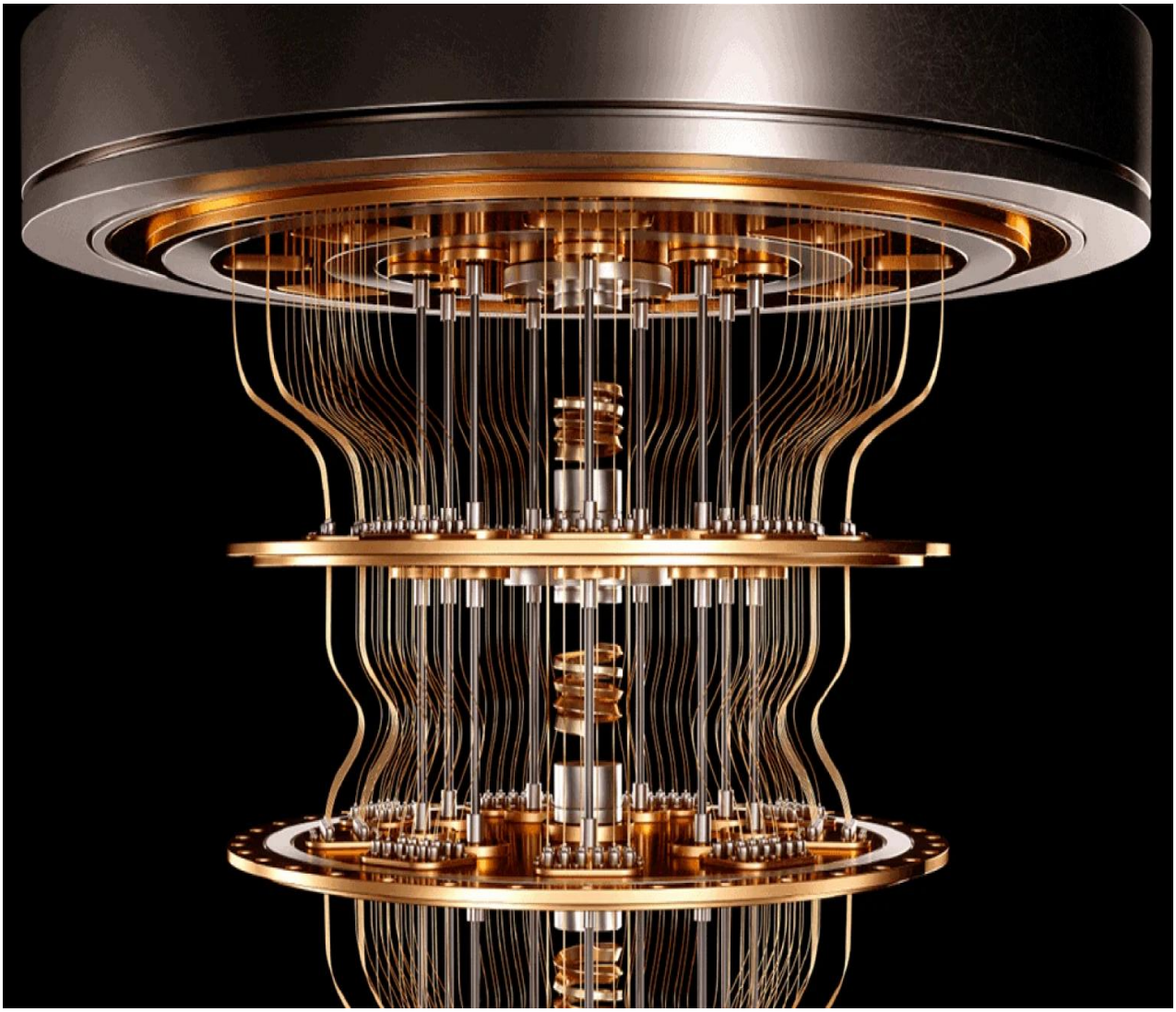


Space Debris Risk Challenge

Quantum Machine Learning

Braja Gopal Das
d.braja@iitg.ac.in

Roll Number: 230121013

Date of Submission: July 6, 2025

Overview

This project addresses the challenge of predicting space debris collision risk—categorized as Low (0), Moderate (1), or High (2)—using telemetry data. It presents a comparative study between a classical Support Vector Machine (SVM) and a Quantum Support Vector Machine (QSVM). Feature engineering, Principal Component Analysis (PCA), and class rebalancing techniques are applied.

Objectives

- Data Preprocessing.
- Feature Engineering.
- Class balancing.
- Apply PCA for dimensionality reduction.
- Train and evaluate classical and quantum models.
- Compare results using performance metrics.

Data Preprocessing

- Load data with pandas.
- Handling missing values :-
  - I am using Percentile-Based Clipping to make our model more reliable, stable, and accurate.
- Normalizing Inconsistent Units (Most of them in SI units).

Feature Engineering

- Environmental Factors :-
  - These are not very effective for risk level calculation, so PCA is used to combine five environmental factor features into one principal component, named 'space_weather_pca'. This reduces five features to one and helps remove noise from the data.

- Sensor Data :-
  - Sensor data features are also not very effective for risk score calculation. PCA is used to combine four sensor data features into one principal component, named 'sensor_quality_pca'. Two less important features ('azimuth_deg' and 'elevation_deg')

are dropped. Thus, six features are reduced to one, helping to remove noise from the data.

- Debris Properties :-

  - PCA is used to combine 'material_density_kgm3', 'radar_cross_section_m2', and 'debris_size_cm' into one principal component. The 'debris_age_years' feature is dropped due to its low importance. Thus, three features are reduced to one, which helps remove noise from the data.

- Orbital Parameters :-

  - Two new features, 'mean_altitude_km' and 'altitude_range_km', are defined from 'apogee_km' and 'perigee_km' as shown in the code. Features such as 'semi_major_axis_km', 'eccentricity', 'orbital_period_min', 'right_ascension_deg', 'argument_perigee_deg', and 'inclination_deg' are dropped, as they do not contribute significantly to risk level and their removal helps reduce noise.

- Physical Parameters :-

  - 'Relative_kinetic_energy' and 'relative_momentum' are defined, from which 'impact_parameter' and 'collision_energy_window' are derived. 'Energy_ratio' is calculated from 'debris_kinetic_energy_j' and 'sat_orbital_energy_j', and 'mass_ratio' from 'debris_mass_kg' and 'sat_mass_kg'. Ten features are dropped, and four new features are defined: 'impact_parameter', 'mass_ratio', 'energy_ratio', and 'collision_energy_window'.

- Closest Risk :-

  - A new feature, 'closest_risk', is defined using 'miss_distance_km' and 'closest_approach_time_hr', which are then dropped. This reduces two features to one.

- After feature engineering there are 16 selected features those are : 'relative_angular_momentum', 'collision_prob_raw', 'sat_altitude_km', 'orbital_energy_j', 'relative_inclination_deg', 'sat_maneuverability_score', 'space_weather_pca', 'sensor_quality_pca', 'debris_impact_pca', 'mean_altitude_km', 'altitude_range_km', 'energy_ratio', 'mass_ratio', 'impact_parameter', 'collision_energy_window', 'closest_risk_prob' .

Class Balancing

For class balancing, class weights are used in the Support Vector Classifier (SVC). The class weight for each class is the ratio of the number of datasets in the majority class to the number of datasets in that particular class. These class weights are then provided to the SVC.

Principal Component Analysis (PCA)

Although the number of features is reduced from 40 to 16, this is still computationally expensive for the Quantum Support Vector Machine. Therefore, the number of features is further reduced to 10 using PCA.

Classical And Quantum SVM Model

Classical SVM :-

A Gaussian kernel is used for better classification, and class weights from the class balancing step are applied.

Quantum SVM :-

For the Quantum SVM, data encoding is performed to map features into Hilbert space. After feature engineering and PCA, 10 features remain, requiring 10 qubits for the QSVM. A second-order ZZFeatureMap is used, involving angle (phase) encoding on the Z-axis and controlled-phase interactions (entangling nonlinear terms).

```
def encoding_data(x, wires):
    for i in range(len(x)):
        qml.Hadamard(wires=wires[i])
        qml.RZ(x[i], wires=wires[i])

    for i in range(len(x)):
        for j in range(i+1, len(x)):
            qml.CNOT(wires=[wires[i], wires[j]])
            qml.RZ(x[i] * x[j], wires=wires[j])
            qml.CNOT(wires=[wires[i], wires[j]])
```

The figure at the left shows the code representation of data encoding in Pennylane.

Circuit Depth Analysis of the Data Encoding Scheme of QSVM

The encoding_data function implements a second-order entangling feature map using Hadamard, RZ, and CNOT gates. Circuit depth depends on the number of input features and the degree of parallelization.

- Structure Breakdown

1. Initial Layer
   Each qubit receives:
   - Hadamard gate (creates superposition)
   - RZ(x[i]) gate (encodes input as phase)

These can be applied in parallel, contributing a depth of 1–2.

2. Entangling Layer
   For every unique qubit pair (i, j), the following sequence is applied:
   - CNOT(i, j)
   - RZ(x[i] * x[j]) on qubit j
   - CNOT(i, j) again

This structure creates pairwise entanglement and encodes second-order correlations. With $\frac{n(n-1)}{2}$ such pairs, this contributes up to $\frac{3n(n-1)}{2}$ gates in sequence (if not parallelized).

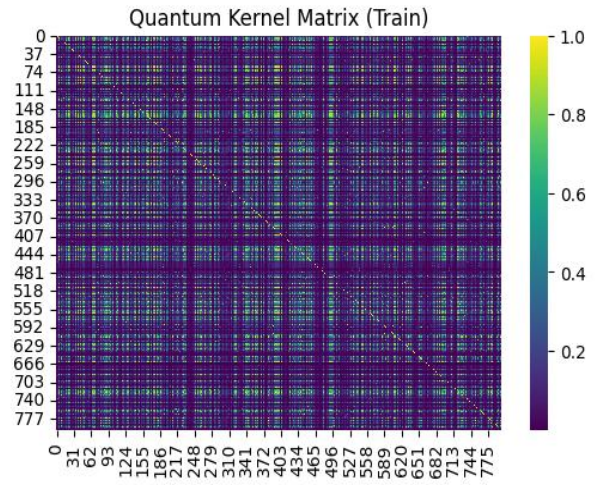
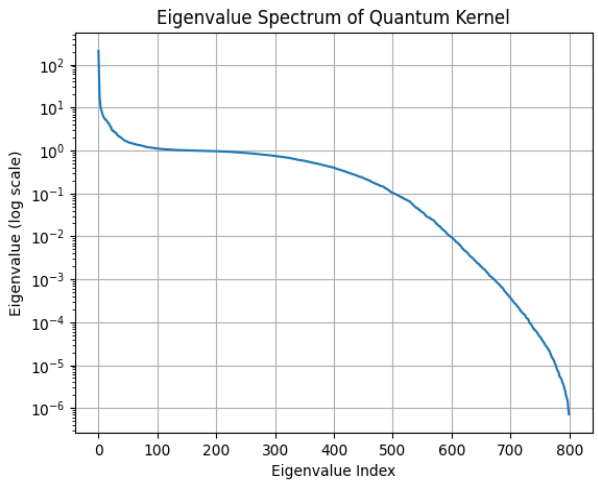
After that we build a function for Quantum Kernel after calculating Quantum Kernel we fed it into classical SVC with class weights .

Features	Entangled Pairs	Entangling Gates	Initial Depth	Worst-case Depth
10	45	135	2	137

Analysis of Kernel Matrix Structure

- The heatmap displays the symmetric quantum kernel matrix for the training data, where each value represents the similarity between quantum-encoded samples. Diagonal values are 1, while off-diagonal patterns indicate meaningful correlations captured by the quantum feature map, supporting the model's ability to distinguish between classes.

- The eigenvalue spectrum of the 800×800 quantum kernel matrix shows a gradual decay, with 673 eigenvalues greater than 0.001. This indicates that a large portion of the feature space contributes meaningfully to the model, reflecting a rich and expressive kernel structure. This distribution helps the QSVM capture complex patterns in the data without being overly sparse or dominated by a few components.

Heat Map of Quantum Kernel Matrix	Eigenvalue Spectrum of Quantum Kernel
	

Runtime Evaluation: QSVM vs CSVM

N = 1000 Dataset (80 % training | 20 % validation) | M = 300 Dataset (For Prediction)

Stage	QSVM Time (min)	CSVM Time
Training	31.05	Negligible (Very Fast)
Validation	15.81	Negligible (Very Fast)
Prediction	24.35	Negligible (Very Fast)

QSVM exhibited significantly longer runtimes due to quantum kernel evaluations, while CSVM completed all stages almost instantly. This highlights the current computational overhead of quantum models compared to classical ones.

Classification Metrics Report

N = 1000 Dataset (80 % training | 20 % validation)

Metric	Classical SVM	Quantum SVM
Accuracy	46.5%	61.5%
Balanced Accuracy	0.4197	**0.4483**
Macro F1 Score	0.3699	**0.4406**
Class 0 F1 Score	0.6207	**0.7544**
Class 1 F1 Score	0.1519	**0.2500**

Metric	Classical SVM	Quantum SVM
Class 2 F1 Score	0.3371	**0.3175**

The Quantum SVM outperformed the Classical SVM across all key metrics, including accuracy, balanced accuracy, and macro F1 score. It handled class imbalance better and improved performance on minority classes, confirming its advantage in more complex prediction scenarios.

Final Prediction Results: Classical vs Quantum SVM

A final predictions.csv file has been generated containing the model predictions on the test data. It includes the following columns:

- sample_id : Unique identifier for each test sample
- label_classical : Predicted label from the Classical SVM
- label_quantum : Predicted label from the Quantum SVM