PROJECT REPORT

Implementing Quantum Support Vector Machine on Real Quantum Device using Qiskit and Amazon Braket

Under the Guidance of:

Dr. Ripul Ghosh

(Principal Scientist)
CSIO-CSIR, Chandigarh, Sector 30c -160030,
India

Submitted by-
Srishti Gupta (UID-21BCS6421)

To:

Apex Institute of Technology, Chandigarh University, Mohali

DECLARATION

I, Srishti Gupta, hereby confirm that I completed a six-week industrial training at the **Central Scientific Instruments Organisation (CSIO)** in Chandigarh. This training was part of the requirements for my Bachelor of Engineering degree in Computer Science, with a specialization in Artificial Intelligence and Machine Learning, at **Chandigarh University, Gharuan**. The training report I am presenting is a true and accurate account of the work I conducted during this period.

Additionally, I affirm that all information included in this report has been sourced from reliable and authentic sources. I also declare that this project report has not been submitted to any other university or institution for the purpose of obtaining any diploma, degree, or certificate.

(Signature of the student)	
Srishti Gupta	**Dr. Ripul Ghosh**
B.Tech. (6th semester)	(Project Mentor)
Roll No.:**21BCS6421**	
CHANDIGARH UNIVERSITY	
Trainee (CSIO)	

ACKNOWLEDGMENT

I extend my profound gratitude to **CSIR- Central Scientific Instruments Organization**, for graciously affording me the exceptional opportunity to embark on and complete this project. The unwavering support and the academically nurturing environment at this esteemed institution have played an indelible and transformative role in shaping the trajectory of my educational journey.

A profound note of appreciation is reserved for the pillars of knowledge and guidance, Dr. Ripul Ghosh, distinguished Principal Scientist at CSIR- Central Scientific Instruments Organization, who serves as my sagacious project guide. Their invaluable mentorship, expert guidance, and unrelenting encouragement have not only been the bedrock upon which my intellectual and professional development has thrived but have also illuminated the path to success in this challenging endeavour. Their dedication to my growth has been a driving force, instilling in us the confidence to overcome obstacles and achieve milestones.

This project stands as a testament to the power of collaboration, with the exchange of ideas and robust mutual support within my academic community proving to be the lifeblood of my success. The vibrant intellectual exchange within the community has not only enriched the project but has also contributed significantly to our holistic learning experience, fostering an environment of creativity and innovation.

TABLE OF CONTENTS

Summary

The study hereby named "Implementing Quantum Support Vector Machine on Real Quantum Devices and Simulators using Qiskit Library and Amazon Braket as an Environment on AWS Platform" will focus on the integration of quantum computing in machine learning while implementing the Quantum Support Vector Machine (QSVM). This research compares how a QSVM will perform, implemented with Qiskit, on quantum simulators against actual quantum devices available through Amazon Bracket.

The project will start with a very focused literature review, which is necessary to understand the very basic principles behind Support Vector Machines (SVM) and their quantum versions. Applications of SVM in diversified areas such as drug discovery, financial forecasting, and cybersecurity are described that accentuate its versatility and significance to practical problems. Such theoretical ground is necessary to understand what benefits and challenges quantum implementation may pose.

The QSVM algorithm implementation is done using Qiskit, an open-source quantum computing framework, on Amazon Braket. Amazon Braket is a fully managed quantum computing service on AWS that provides access to quantum simulators and real quantum hardware. An AWS environment is set up by configuring essential SDKs and choosing quantum devices suitable for experiments to ensure a robust environment for running QSVM experiments.

Data preparation and preprocessing are two of the main elements of this project. The Kaggle breast cancer dataset is selected, cleaned, preprocessed, and splited into training and testing sets. Feature engineering and data visualization techniques are applied to optimally prepare the dataset for QSVM training. These steps are important in data preparation because the data is going to be trained in quantum machine learning algorithms which are reliable.

The quantum feature map must be defined and the quantum circuit for the QSVM must be ready as part of the experimental setup. Real quantum devices and quantum simulators are used to run the quantum circuit, and data is collected. To assess the QSVM's performance across several platforms, statistical criteria like accuracy, recall, sensitivity, and runtime are employed. An additional understanding of the trade-offs between simulators and actual quantum devices may come from such a comparison.

The result shows an elaborate performance comparison between the simulator and real quantum devices with regard to QSVM in terms of the trade-offs of accuracy, recall sensitivity, and execution time. Efficiency, practicability, and the other following two aspects—accuracy, recall, sensitivity, and execution time—accompanying the implementation of QSVM on different quantum computing platforms.

Furthermore, the present project helps to see how quantum computing is helpful for building better machine- learning algorithms, such as SVM. While the quantum simulators are convenient and stable, the actual quantum devices give a preview of future capabilities and challenges in quantum computing. Results suggest that quantum computing holds promise, but more advancements need to be made toward consistencies and scalability in practical applications. This paper is a stepping stone for further research and development in quantum machine learning, inciting more exploration and innovation into harnessing quantum computing towards solving complex computational problems.

Literature Review: Large Quantum Support Vector Machine

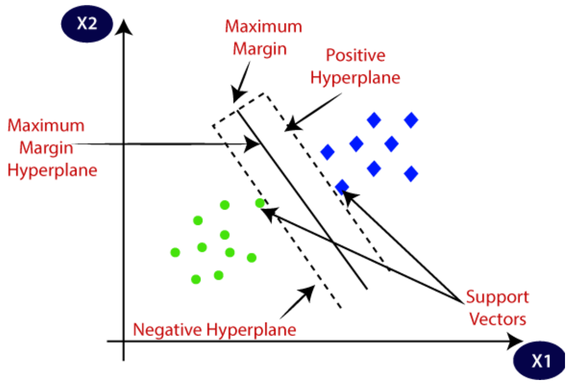
Over the last few decades, the discipline of machine learning has grown dramatically, with Support Vector Machines (SVM) serving as a foundational technique for classification and regression problems. An SVM creates a hyperplane or a group of hyperplanes in a high-dimensional space that may be used for classification, regression, and outlier detection. The strength of SVMs stems from its capacity to convert the original input space into a higher-dimensional space using kernel func=tions, allowing for complicated decision boundaries.

4.1 Overview of SVM and its Quantum Components

SVMs are well-known for their resilience and accuracy across a variety of areas. They function by determining the best hyperplane to maximise the margin between distinct classes in the training data. This is accomplished using a mix of linear algebra and optimization methods. However, classic SVMs have computational efficiency constraints, particularly when working with huge datasets or complicated kernel functions.

Support Vector Machine

Supervised Machine Learning algorithm, utilized for tasks including regression, outlier detection, and linear and nonlinear classification. The SVM algorithm’s primary goal is to locate the best hyperplane in a N-dimensional space that may be used to divide data points into various feature space classes.



1. Hyperplane:

   The hyperplane aims to keep as big a buffer as possible between the closest points of different classes. The dimension of the hyperplane is determined by the number of features. When input features are limited to two, the hyperplane can be thought of as a line. If there are three input features, the hyperplane changes into a 2-D plane.

   The linear hyperplane equation is expressed as follows: wTx + b = 0. The normal vector to the hyperplane is represented by the vector W. that is, the direction that is opposite the hyperplane. The offset or distance of the hyperplane from the origin along the normal vector w is represented by the parameter b in the equation.

2. Support Vectors:

   The data points that are closest to the hyperplane areknown as support vectors, and they are crucial in determining the margin and hyperplane.

3. Margin:

   The distance between the hyperplane and the support vector isknown as the margin. Maximising the margin is the primary goal of the support vector machine algorithm. Better classification performance is indicated by the bigger margin.
   The distance between a data point x_i and the decision boundary canbe calculated as:

$$\frac{w^T x + b = 0}{||w||}$$

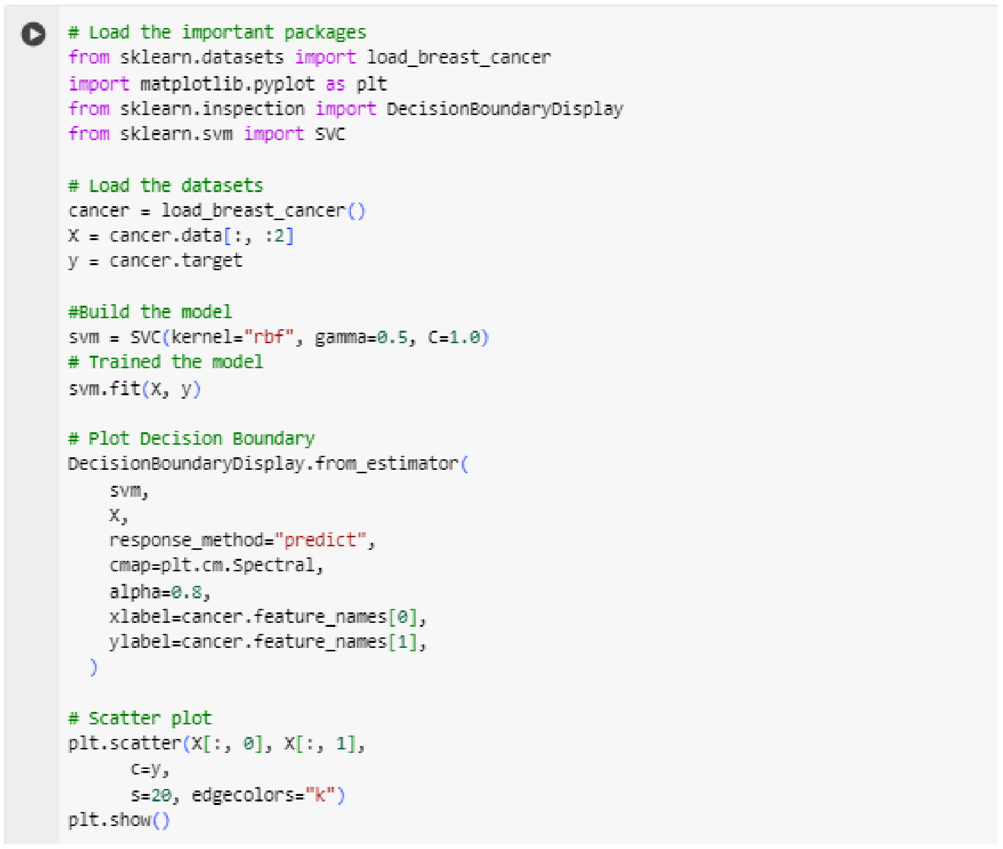
   where ||w|| represents the Euclidean norm of the weight vector w.

4. Kernel:

   The mathematical function is employed in support vector machines (SVM) to map the initial input data points into high- dimensional feature spaces. This allows for the easy identification of thehyperplane, even in situations where the data points are not linearly separable in the original input space.

   Quantum kernels are often described by a similarity matrix based on a quantum circuit, which can be parameterized or not. Both Pennylane and Qiskit have functions for creating kernels that maybe used in scikit-learn's SVC (Support Vector Classifier).

   The concept behind quantum kernels is to map the feature vector more effectively by utilising the principles of quantum mechanics. Encoding classical data into qubits, carrying out operations (such superposition and rotations on the Bloch sphere), and calculating the dot product of the resultant states are the main rules for such mapping.



```python
# Load the important packages
from sklearn.datasets import load_breast_cancer
import matplotlib.pyplot as plt
from sklearn.inspection import DecisionBoundaryDisplay
from sklearn.svm import SVC

# Load the datasets
cancer = load_breast_cancer()
X = cancer.data[:, :2]
y = cancer.target

#Build the model
svm = SVC(kernel="rbf", gamma=0.5, C=1.0)
# Trained the model
svm.fit(X, y)

# Plot Decision Boundary
DecisionBoundaryDisplay.from_estimator(
    svm,
    X,
    response_method="predict",
    cmap=plt.cm.Spectral,
    alpha=0.8,
    xlabel=cancer.feature_names[0],
    ylabel=cancer.feature_names[1],
  )

# Scatter plot
plt.scatter(X[:, 0], X[:, 1],
      c=y,
      s=20, edgecolors="k")
plt.show()
```

4.2 Application of QSVM in Various Domains

- **Drug Discovery:**

In drug discovery, screening is a process of identifying drug candidates from amongst vast libraries of molecules to find those with desired therapeutic properties. This task usually involves classic computational methods that are often computationally burdensome and imprecise. Here is how QSVMs can help:

Feature Mapping: In this way, the high-dimensional data characteristic of molecular classification tasks can be handled efficiently by using QSVMs to potentially better represent molecular structure and properties.

Better Accuracy: For example, it is expected that quantum algorithms, like QSVMs, based on states and quantum interference mechanisms, will definitely provide a new view of complex molecular structure relationships for a better classification of molecular data.

Speed and Efficiency: In quantum hardware, the algorithms could speed up the screening process by performing computations out of reach for classical computers running in reasonable time scales.

Integrating QSVMs in drug discovery pipelines will potentially accelerate the identification of promising drug candidates, thereby speeding up the development of new therapies and treatments for a range of diseases. This application is a nice example of how QSVMs could dramatically change complex data analysis tasks in those fields where traditional computational methods have been more or less unsuccessful because of the sheer volume and associated complexity of the data involved.

- **Financial Forecasting (Stock Market Prediction)**

  Financial forecasting is a prediction of future financial outcomes on the basis of historical data and market trends. In this area, QSVMs could prove to be useful in enhancing the accuracy of the predictions produced as a result of their quantum computational advantages. Here is an example to make it clear:

  Example: Predicting Stock Prices

  Problem Statement: Investors and financial institutions need to predict stock prices for making decisions. Traditional methods often struggle to cope with the complexity of financial data and the rapid changes in market dynamics.

  Applicability to historical stock markets, their price movements, trading volumes, and macroeconomic indicators, which are efficiently analyzed by QSVMs: quantum algorithms can potentially help to extract many more relevant features from complex financial datasets, where subtle patterns impact stock prices.

  Enhanced Predictive Models: QSVMs afford the possibility of building more accurate predictive models vis-à-vis classical SVMs. It uses the quantum states of superposition and interference effects to explore complex relationships in the financial data more powerfully. Such an ability could lead to improved accuracy in forecasting stock price movements over different time horizons, from short-term fluctuations to long-term trends.

  Advantages in Speed and Scalability for Quantum: Quantum implementations using quantum hardware for quantum algorithms would fit large-scale financial data better than classical computers. This will enable faster calculations while training and testing predictive models for risk management in real or near real time within dynamic financial markets.

  QSVMs can help optimize investment portfolios by predicting the risk-adjusted returns of different asset classes. This would help investors and fund managers allocate resources better to maximize returns while managing risks better, within what is still apparently a very volatile market.

- **Cyber Security**

Cybersecurity secures the computer systems, networks, and data from hostile attacks and unauthorized access. Quantum Support Vector Machines could potentially facilitate the enhancement of cybersecurity measures through more accurate anomaly detection and identification of threats. This can be made clearer with the help of an example in the real world:

Example: Network Intrusion Detection

Problem Statement: Unauthorized access along with malicious activities needs to be detected and prevented from computer networks for better maintenance of cybersecurity. Traditional systems are signature-based and rule-based and hence ineffective against sophisticated and rapidly evolving cyber threats.

Implementation of QSVM:

Anomaly Detection: The QSVM can be used in order to analyze network traffic data, such as packet headers and traffic patterns, including communications behavior. In this regard, quantum algorithms may effectively detect anomalies in network behavior by recognizing small deviations from the standard pattern, showing potentially malevolent cyber threats.

Better Identification of Threats: Quantum support vector machines have the potential to provide a greater ability for distinguishing between normal and malicious activities, as compared to conventional SVMs, regarding network traffic data. This could be another step toward detecting zero-day attacks and new types of malware which traditional ways of detection are not able to recognize.

Quantum Computational Advantages: By harnessing quantum algorithms to detect anomalies in cybersecurity, computational advantages can be gained in much faster pattern recognition and enormously more efficient processing of data. QSVMs can handle the vastness of network data and enormous computational complexities to enable real-time threat detection and response.

Optimization in Security Operations: Combine that with day-to-day cybersecurity operations, and one can realize that security teams can cut their operating time in half, as false positives will decrease dramatically and automation can take the place of manual analysis of suspicious network activity. All this is possible through quick incident response and mitigation strategies, ensuring minimal damage possible from cyber attacks.

Potential Benefits:

- Improved Detection Accuracy: QSVMs could provide a higher accuracy in identifying network intrusions and anomalous behaviors compared to traditional methods.
- Enhanced Threat Intelligence: Deeper insights into emerging cyber threats and patterns of malicious activity through the use of quantum algorithms.
- Scalability and Efficiency: High volumes of network data processing and efficient processing through the quantum computational advantages, ensuring real-time cybersecurity monitoring and response.

QSVMs should be applied in network intrusion detection and cybersecurity for the purpose of solidifying the defense mechanism, enabling data protection from cyber threats that might compromise the digital infrastructures' integrity. This demonstrates but one way in which quantum computation techniques will revolutionize the practice of cybersecurity in the future.

4.3 Comparison of Quantum Simulators and Real Quantum Devices

Comparing quantum simulators and real quantum devices would be in terms of architecture, performance, accuracy, and applications. Let us compare both in detail.

1. Architecture and Underlying Technology

Quantum Simulators:

- **Classic Hardware:** The simulation in quantum simulators runs on classical computers using quantum algorithms to replicate the behavior of a quantum system.
- **Scalability:** The ability to scale is limited by classical computational power; hence, simulating large quantum systems has become painful due to the skyrocketing demand for resources.
- **Without Errors:** The simulations are usually idealized and free from errors, as they do not take into account the physical noise and imperfections of a real quantum device.

Real Quantum Devices:

- **Quantum Hardware:** Implementation with real quantum bits, qubits, in systems such as superconducting circuits, trapped ions, or photonic systems.
- **Scalability:** While constrained by the number of qubits and coherence times in current quantum hardware, research is proceeding to build more complex systems.
- **Noise and Errors:** Quite sensitive to noise, decoherence, and operational errors that would compromise the exactitude and fidelity of a calculation.

2. Performance and Speed

Quantum Simulators:

- **Speed:** This is very dependent on the processing power of a classical computer and may be slow for big quantum systems.
- **Efficiency:** It's efficient for small-scale simulations, but it fails to be effective in large-scale quantum computations because the requirement of resources becomes exponential.
- **Flexibility:** Can easily be tailored for different quantum algorithms and experiments without any physical constraints which would come if real quantum hardware was used.

Real Quantum Devices:

- **Speed:** Can be potentially faster in some particular quantum algorithms which take advantage of quantum parallelism and entanglement.
- **Efficiency:** Solves some problems in a more efficient way than classical computers, even though hardware is at a very early stage of development.
- **Physical Constraints:** Limited by coherence time, gate fidelity, and qubit connectivity, which affect the overall performance.

3. Accuracy and Reliability

Quantum Simulators:

- **Ideal Conditions:** Yields the ideal results under idealized conditions; useful in theoretical studies and in the development of algorithms.
- **Results with No Errors:** Simulated results are free from errors induced by the hardware and provide a perfect representation of how the algorithm performs.

Real Quantum Devices:

- **Real-World Condition:** Performance under real-world conditions is further affected by physical noise, errors, and decoherence, necessitating error correction and mitigation techniques.
- **Reliability:** The results may vary because of hardware imperfections, so repeated runs become necessary to get reliable outcomes.

4. Practical Applications

Quantum Simulators:

- **Algorithm Development:** Excellent to develop and check quantum algorithms before deploying them on real quantum hardware.
- **Education and Training:** Perfect for education to have students and researchers build an understanding of quantum concepts without expensive quantum hardware.
- **Benchmarking:** Be a benchmark for comparing the performance of real quantum devices.

Real Quantum Devices:

- **Real-World Problems:** Potential applications toward solving real-world problems in which quantum advantage can be shown, like certain optimization problems and cryptographic tasks as well as quantum simulations of physical systems.
- **Experimental Validation:** Provide experimental validation of quantum theories and algorithms toward enhancing research work related to quantum computing.
- **Quantum Advantage:** Aim to achieve quantum advantage (or supremacy) by solving problems that are infeasible for classical computers.

5. Use Cases and Examples

Quantum Simulators:

- **Simulating Quantum Circuits:** The likes of Qiskit Aer, Amazon Braket and a handful of quantum simulators let us test quantum circuits on classical hardware.
- **Algorithm Prototyping:** Researcher prototype and refine quantum algorithms before implementing them on real quantum devices.

Real Quantum Devices:

- **IBM Quantum Experience:** Provides access to run quantum algorithms and experiments on real superconducting qubits based devices.
- **D-Wave Quantum Annealers:** Used for solving specific optimization problems using quantum annealing techniques.
- **Rigetti and IonQ:** Quantum processor access based on superconducting qubits and trapped ions, respectively.

Conclusion

Quantum simulators as well as real quantum devices have their respective advantages and limitations. Quantum simulators play an instrumental role in the development and testing of new algorithms under idealized conditions. In principle, real quantum devices can advance further toward theory validation, the showcasing of quantum advantage, and entering the realm of real applications, although today they suffer from problems of noise and scalability. These would together play a complementary part in the continuous development of quantum technology.

Setup and Configuration

5.1 Overview of Qiskit and Amazon Braket

Qiskit: An open-source quantum computing framework designed by IBM. Qiskit allows you to create and execute quantum algorithms on real quantum devices or simulators. It supports the development of quantum circuits, quantum machine learning algorithms, and various quantum applications, including modules for quantum information science, quantum chemistry, and quantum hardware interaction.

Amazon Braket: A fully managed AWS quantum computing service that offers access to various quantum computing hardware from different vendors like Rigetti, IonQ, and D-Wave, as well as quantum simulators. Amazon Braket enables developers and researchers to build, test, and run quantum algorithms in a scalable and flexible cloud environment.

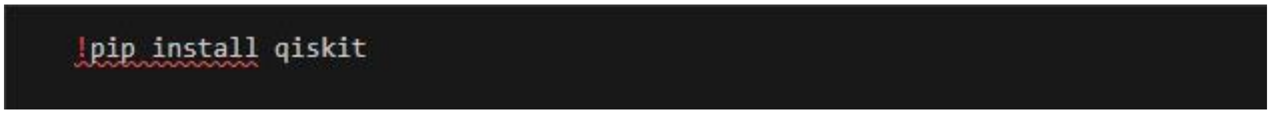
5.2. Setting up and Configuration in AWS Platform

Amazon Braket Configuration

1. **Account Creation**: Create an AWS account and sign in to the AWS Management Console if you don't have one already.
2. **Braket Console Access**: Open the Amazon Braket Console and confirm that the service is available in your region.
3. **IAM Role Configuration**: Set up AWS Identity and Access Management (IAM) roles with the right permissions for accessing Braket resources. Ensure the roles can interact with S3, Lambda, and other required services.

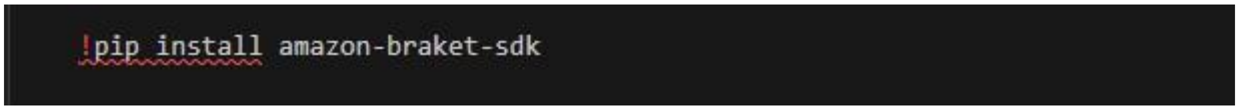
Environmental Setup

1. **Setup AWS CLI**: Install the AWS Command Line Interface to allow interaction with AWS services programmatically. Configure the CLI with your AWS credentials by typing `aws configure`.
2. **Install Qiskit**: Set up the Python environment and install Qiskit through any package manager:



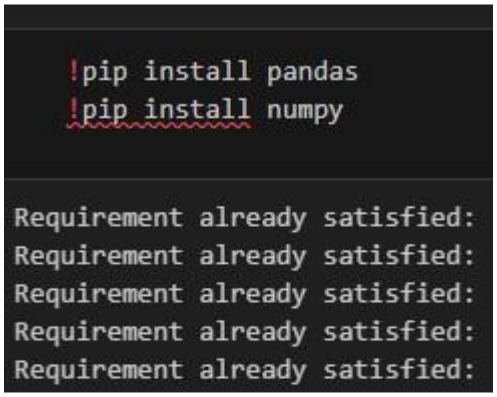
```
!pip install qiskit
```

3. **Braket SDK Installation**: Download and install the Amazon Braket SDK for programmatic access to quantum devices and high-performance quantum simulators through Python:



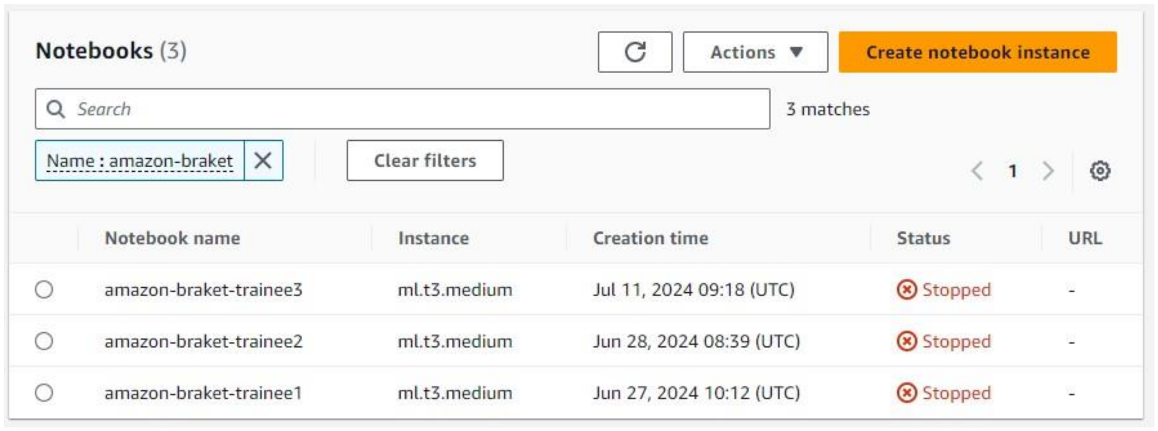
```
!pip install amazon-braket-sdk
```

4. **Install pandas and numpy:**



```
!pip install pandas
!pip install numpy

Requirement already satisfied:
Requirement already satisfied:
Requirement already satisfied:
Requirement already satisfied:
Requirement already satisfied:
```

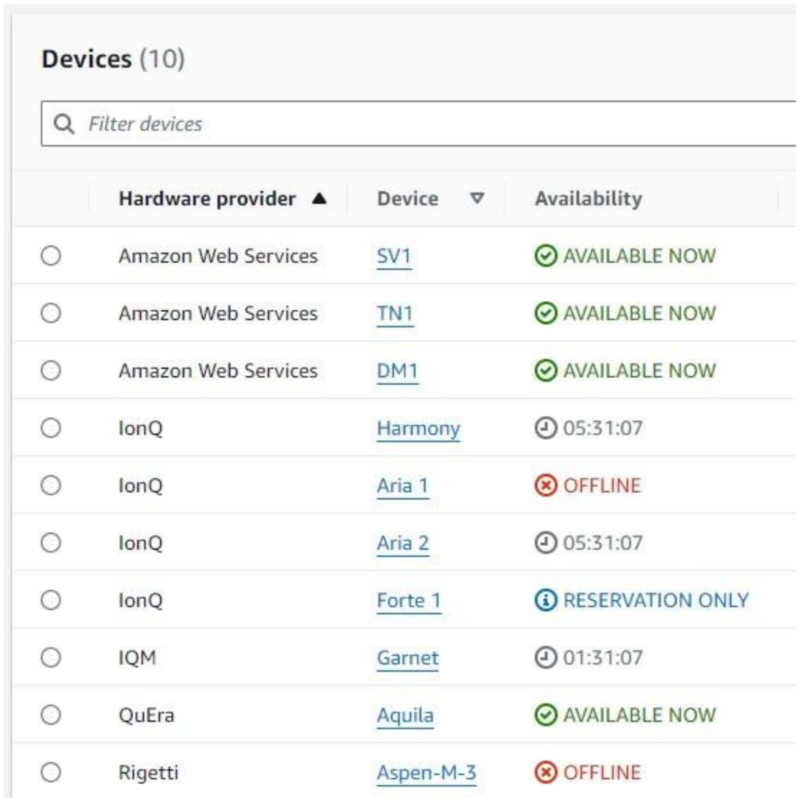
5. **Notebook Environment**: Optionally, set up a Jupyter Notebook environment for interactive development. You can create this environment by setting up Anaconda or creating a workspace directly with AWS SageMaker Notebooks.



	Notebook name	Instance	Creation time	Status	URL
○	amazon-braket-trainee3	ml.t3.medium	Jul 11, 2024 09:18 (UTC)	Stopped	-
○	amazon-braket-trainee2	ml.t3.medium	Jun 28, 2024 08:39 (UTC)	Stopped	-
○	amazon-braket-trainee1	ml.t3.medium	Jun 27, 2024 10:12 (UTC)	Stopped	-

Accessing Quantum Devices

1. **Device Selection**: Browse available quantum devices in the Amazon Braket console. Choose appropriate devices for your experiment, such as IonQ or Rigetti.
2. **Resource Allocation**: Allocate qubits and other necessary resources based on your QSVM experiment requirements. Each device may have different capabilities and limitations.



	Hardware provider	Device	Availability
○	Amazon Web Services	SV1	AVAILABLE NOW
○	Amazon Web Services	TN1	AVAILABLE NOW
○	Amazon Web Services	DM1	AVAILABLE NOW
○	IonQ	Harmony	05:31:07
○	IonQ	Aria 1	OFFLINE
○	IonQ	Aria 2	05:31:07
○	IonQ	Forte 1	RESERVATION ONLY
○	IQM	Garnet	01:31:07
○	QuEra	Aquila	AVAILABLE NOW
○	Rigetti	Aspen-M-3	OFFLINE

Development Environment

1. **IDE (Integrated Development Environment)**: Use any IDE to develop your quantum circuits and algorithms: PyCharm, VS Code, or Jupyter Notebook.
2. **SDK Integration**: Integrate the Amazon Braket SDK with Qiskit within your development environment. Ensure your code can switch between using simulators and real quantum devices as needed.

5.3. Data Preparation and Pre-processing

Appropriate data preparation and preprocessing are essential before starting any machine learning project, including QSVM. This process involves selecting, cleaning, transforming, and splitting the data so it becomes fit for training and testing with a quantum SVM.

Data Selection

1. **Choice of Dataset**: Select appropriate datasets for your experiments. For this project, the breast cancer dataset from Kaggle is used. Ensure the dataset suits the classification task and is manageable in size for quantum processing.
2. **Data Understanding**: Learn about the structure, attributes, and labels of the dataset. This includes tasks like checking data types, ranges, and distributions for all variables.

Data Cleaning

1. **Handling Missing Values**: Determine and address any missing data points by either deleting incomplete records or imputation, which fills in the gaps with the mean, median, or mode.

2. **Outlier Detection:** Find and deal with outliers that could distort QSVM results. Techniques for locating and determining how to handle these outliers include statistical approaches or visualisation software.

3. Feature Engineering

1. **Feature Selection**: Find and select relevant features that contribute most to the classification task using domain knowledge, statistical tests, or feature importance metrics.
2. **Normalization/Standardization**: Scale numerical features to a common range or standard distribution, ensuring features contribute evenly to the model. Techniques include min-max scaling or z-score normalization.

Splitting Data

1. **Train-Validation-Test Split**: Typically, datasets are partitioned into 70–20–10 or 80–10–10 for the training, validation, and test sets, respectively. This ensures that the model is trained, validated, and tested on different data, preventing overfitting and enabling accurate performance evaluation.

Data Visualization

1. **Exploratory Data Analysis (EDA)**: Visualize the data to understand its distribution, relationships among features, and possible patterns. Useful tools for EDA include histograms, scatter plots, and correlation matrices.
2. **Feature Correlation**: Visualize and analyze feature correlation to identify multicollinearity and any other issues that may affect model performance.

Experimental Setup

6.1 Different choices of dataset

Dataset Selection: The choice of datasets is crucial for evaluating the performance and generalizability of the QSVM algorithm. For this project, the breast cancer dataset from Kaggle is selected due to its relevance to binary classification tasks, which is suitable for SVM.

Breast Cancer Dataset: Features from a digital image of a fine needle aspirate (FNA) of a breast mass are computed in this dataset. It has 30 parameters, including the target variable that indicates whether the tumour is benign or malignant, as well as mean radius, mean texture, mean perimeter, mean area, and mean smoothness.

Additional Datasets (Optional): To further test the robustness of the QSVM, additional datasets can be considered:

- **Iris Dataset**: A classic dataset for classification tasks, containing three classes of iris plants with four features: sepal length, sepal width, petal length, and petal width.
- **Wine Quality Dataset**: Contains chemical properties of wines along with quality ratings. This is useful for multi-class classification tasks.

Here we are using Iris dataset for implementing Quantum Support Vector Machine.

6.2 Parameters used for the experiment

Performance Metrics

Execution Time: Determine how long it takes to run the QSVM algorithm on actual quantum devices as well as quantum simulators. This covers the time needed for circuit execution, data encoding, and result retrieval.

Accuracy: Determine what percentage of the total cases were correctly classified. One of the main metrics used to assess classification models is accuracy.

Recall: Sometimes referred to as sensitivity, recall expresses the percentage of real positives that the model accurately detected. For unbalanced datasets, where it is critical to find every occurrence of the minority class, it is indispensable.
Sensitivity: Recall and sensitivity are interchangeable terms. It is a gauge of how well the model can recognise good examples.

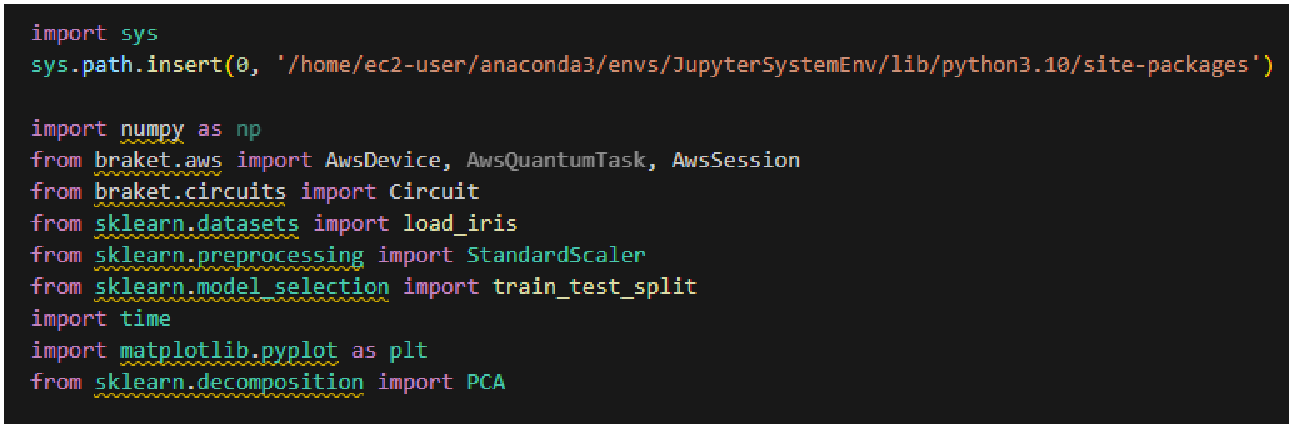
Precision is defined as the percentage of all positive forecasts that are actually true positive predictions. In situations where the cost of false positives is significant, it is crucial.

Code Implementation

7.1 Setting up libraries and modules

This section imports necessary libraries and modules:

- `sys.path.insert`: Adds a path to the Python system path where additional packages are installed (likely for AWS Braket).
- `numpy`: For numerical operations.
- `braket.aws`: AWS SDK for Braket, including `AwsDevice`, `AwsQuantumTask`, and `AwsSession`.
- `braket.circuits`: Provides tools for constructing quantum circuits.
- `sklearn.datasets`, `sklearn.preprocessing`, `sklearn.model_selection`: Scikit-learn for generating synthetic data (`make_blobs`), data preprocessing (`StandardScaler`), and data splitting (`train_test_split`).
- `time`: For timing the execution of quantum tasks.
- `matplotlib.pyplot`: For plotting results.


```python
import sys
sys.path.insert(0, '/home/ec2-user/anaconda3/envs/JupyterSystemEnv/lib/python3.10/site-packages')

import numpy as np
from braket.aws import AwsDevice, AwsQuantumTask, AwsSession
from braket.circuits import Circuit
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import time
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
```

7.2 AWS session setup and Data Loading

AWS Session Initialization:

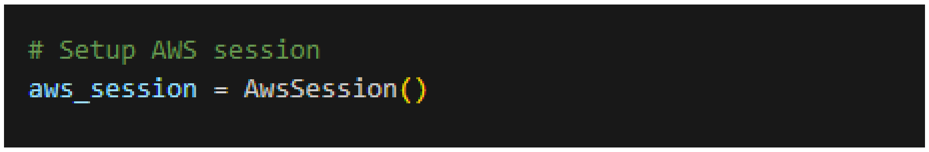
- **Purpose**: This step initializes a session to interact with AWS services, specifically AWS Braket in this context.
- **AwsSession Class**: This class is part of the AWS Braket SDK. It manages the interaction with AWS resources, handling tasks such as authentication, request signing, and configuration.

  The session object provides a context for running quantum tasks on AWS Braket, ensuring that the user has the necessary credentials and configurations to access AWS services.

- **Use in the Code**:

  Initializing `AwsSession()` creates a session that can be used to instantiate other AWS Braket objects, like `AwsDevice`.

  It sets up the environment needed for submitting quantum jobs to AWS Braket's quantum processing units (QPUs) and simulators.



```
# Setup AWS session
aws_session = AwsSession()
```

Data Loading and Preprocessing:

- **Data Loading**:

  `load_iris()`: This function from `sklearn.datasets` loads the Iris dataset, a classic dataset in machine learning and statistics.

  Iris Dataset:

  - Contains 150 samples of iris flowers.
  - Each sample has 4 features: sepal length, sepal width, petal length, and petal width.
  - There are 3 classes (species) of iris flowers: Setosa, Versicolour, and Virginica.

  `x` and `y`:

  - `x` stores the feature data (the measurements of the flowers).
  - `y` stores the target labels (the species of the flowers).

- **Data Splitting**:

  `train_test_split(X, y, test_size=0.3, random_state=42)`:

- This function from `sklearn.model_selection` splits the dataset into training and testing sets.

  Parameters:

- `test_size=0.3`: 30% of the data is reserved for testing, and 70% for training.
- `random_state=42`: Ensures reproducibility of the split by setting a seed for the random number generator.

  Outputs:

- `X_train`: Training set features.
- `X_test`: Testing set features.
- `y_train`: Training set labels.
- `y_test`: Testing set labels.

  Data Normalization:

  StandardScaler:

- Part of `sklearn.preprocessing`.
- Standardizes features by removing the mean and scaling to unit variance (Z-score normalization).

  `scaler = StandardScaler().fit(X_train)`:

- Fits the scaler on the training data, computing the mean and standard deviation to be used for later scaling.

  `X_train_scaled = scaler.transform(X_train)`:

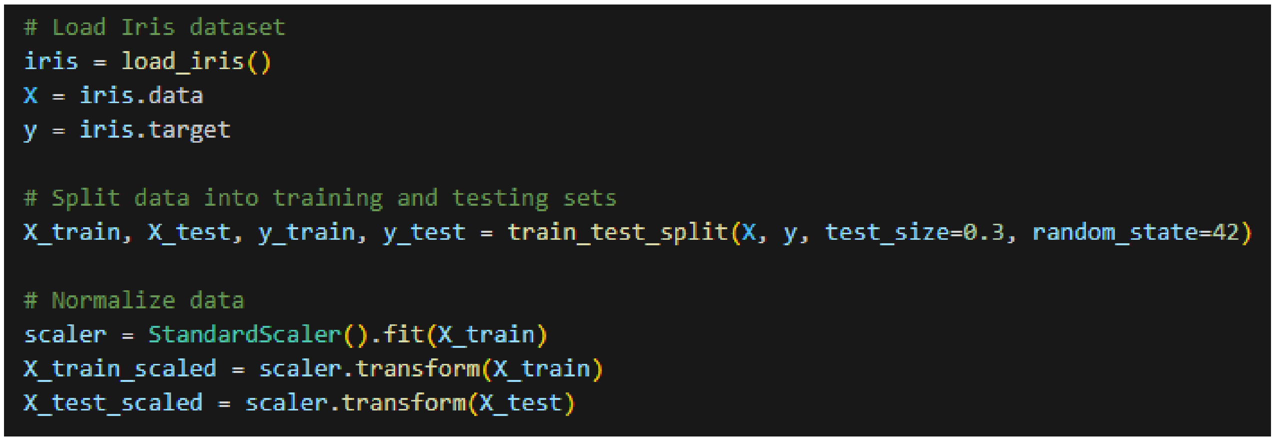
- Transforms the training data using the computed mean and standard deviation, resulting in standardized features.

  `X_test_scaled = scaler.transform(X_test)`:

- Transforms the testing data using the same mean and standard deviation computed from the training data.

▪ **Why Normalize Data?**

Normalisation accelerates convergence and enhances the performance of many machine learning algorithms, including quantum SVM. It guarantees that each feature contributes equally to the model's training process, preventing features with higher sizes from dominating the learning process.



```
# Load Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Normalize data
scaler = StandardScaler().fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

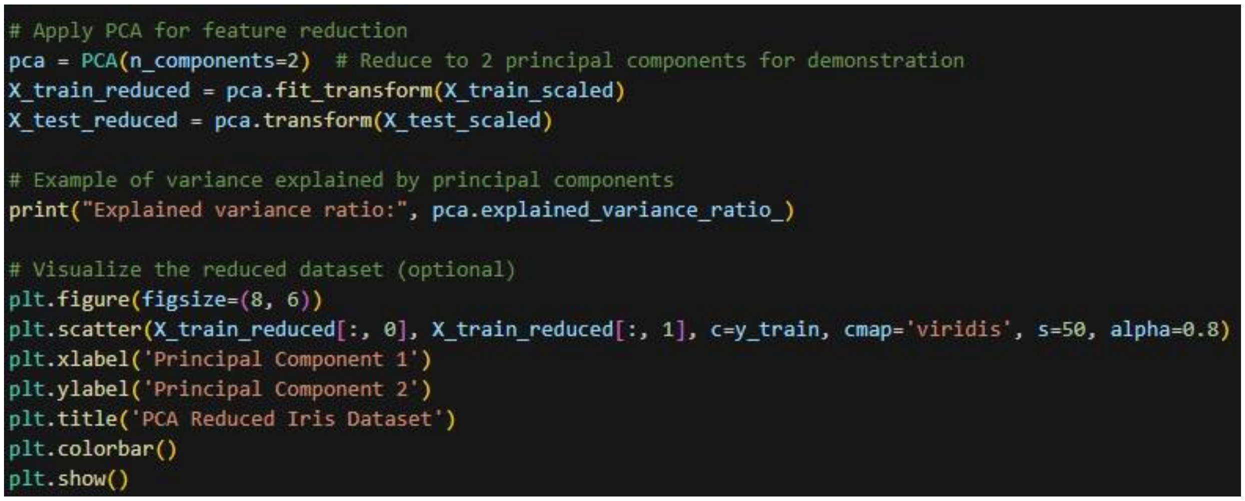
7.3 Quantum Feature Map Definition

Feature Reduction Using PCA:

- **Principal Component Analysis (PCA)**:
- Using PCA, a dimensionality reduction technique, one can minimise the amount of features in a dataset while maintaining a high degree of variability.
- The original features are changed into a new set known as the principal components. These elements are arranged according to how much variance they are able to extract from the data, making them orthogonal (uncorrelated).

- **Steps in PCA**:

1. **Standardize the Data**: Before applying PCA, it is essential to standardize the data (which was done in the previous step) because PCA is affected by the scale of the data.
2. **Compute Covariance Matrix**: Calculate the covariance matrix of the standardized data.
3. **Calculate Eigenvalues and Eigenvectors**: Compute the eigenvalues and eigenvectors of the covariance matrix.
4. **Sort Eigenvalues and Select Top Components**: Sort the eigenvalues in descending order and select the top kkk eigenvalues. The corresponding eigenvectors form the principal components.
5. **Transform the Data**: Project the original data onto the new kkk-dimensional subspace.


```python
# Apply PCA for feature reduction
pca = PCA(n_components=2)  # Reduce to 2 principal components for demonstration
X_train_reduced = pca.fit_transform(X_train_scaled)
X_test_reduced = pca.transform(X_test_scaled)

# Example of variance explained by principal components
print("Explained variance ratio:", pca.explained_variance_ratio_)

# Visualize the reduced dataset (optional)
plt.figure(figsize=(8, 6))
plt.scatter(X_train_reduced[:, 0], X_train_reduced[:, 1], c=y_train, cmap='viridis', s=50, alpha=0.8)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA Reduced Iris Dataset')
plt.colorbar()
plt.show()
```

7.4 Creation of Quantum SVM Circuit

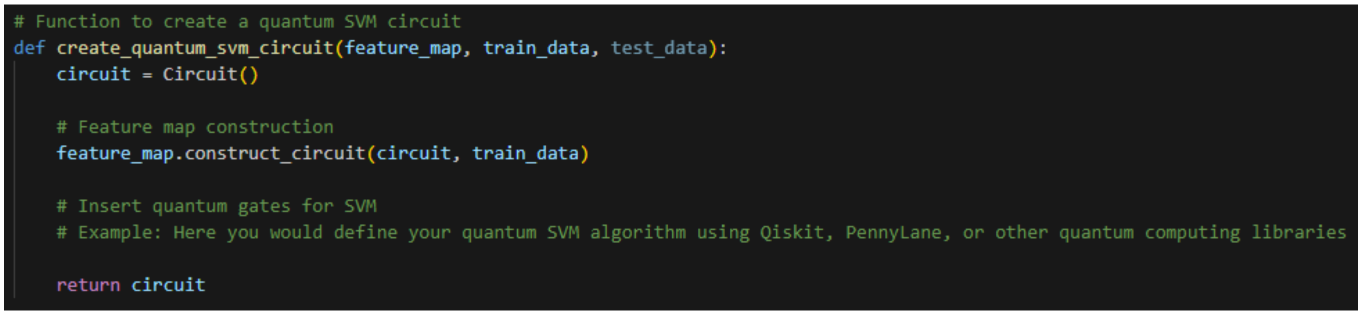
This function is designed to create a quantum circuit tailored for running a quantum Support Vector Machine (SVM). The key elements are constructing the quantum circuit, applying a feature map to encode the data, and adding the quantum gates that perform the SVM algorithm.

Function Definition:

- **`create_quantum_svm_circuit(feature_map, train_data, test_data)`**:

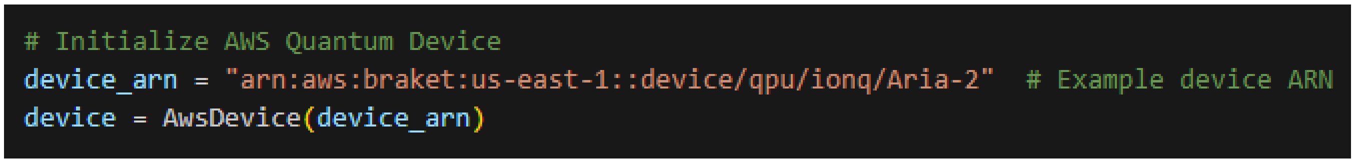
Parameters:

- **`feature_map`**: An object that defines how the classical data should be encoded into the quantum state.
- **`train_data`**: Training dataset that will be used to construct the feature map.
- **`test_data`**: Testing dataset, which is not directly used in this function but might be used later for predictions.


```python
# Function to create a quantum SVM circuit
def create_quantum_svm_circuit(feature_map, train_data, test_data):
    circuit = Circuit()

    # Feature map construction
    feature_map.construct_circuit(circuit, train_data)

    # Insert quantum gates for SVM
    # Example: Here you would define your quantum SVM algorithm using Qiskit, PennyLane, or other quantum computing libraries

    return circuit
```

7.5 Quantum Device Execution

- **`device_arn`**: This is the Amazon Resource Name (ARN) for the quantum device you are using. In this case, it's an IonQ Aria-2 quantum computer.
- **`device = AwsDevice(device_arn)`**: Initializes an `AwsDevice` object using the specified device ARN. This object allows you to interact with the quantum hardware through AWS Braket.



```
# Initialize AWS Quantum Device
device_arn = "arn:aws:braket:us-east-1::device/qpu/ionq/Aria-2"  # Example device ARN
device = AwsDevice(device_arn)
```

Execution:

- **Lists to Store Results and Execution Times**:

- **`results_device`**: Stores the results of the quantum tasks.
- **`execution_times_device`**: Stores the execution times for each run.

- **Loop to Run Simulations**:

- **`for i in range(5):`**: Runs the simulation five times to average the results and execution times.
- **`start_time = time.time()`**: Records the start time before running the quantum circuit.
- **`quantum_circuit_device = create_quantum_svm_circuit(QuantumFeatureMap(), X_train_scaled, X_test_scaled)`**: Creates the quantum circuit using the training data and a feature map.
- **`task_device = device.run(quantum_circuit_device, shots=1000)`**: Executes the quantum circuit on the actual quantum hardware with 1000 shots (repetitions of the experiment to gather statistics).
- **`results_device.append(task_device.result())`**: Appends the result of the quantum task to the `results_device` list.
- **`execution_time = time.time() - start_time`**: Calculates the execution time for the quantum task.
- **`execution_times_device.append(execution_time)`**: Appends the execution time to the `execution_times_device` list.

```
# Simulate on the device
results_device = []
execution_times_device = []

for i in range(5):  # Run 5 times for averaging
    start_time = time.time()
    # Create circuit
    quantum_circuit_device = create_quantum_svm_circuit(QuantumFeatureMap(), X_train_scaled, X_test_scaled)

    # Execute on device
    task_device = device.run(quantum_circuit_device, shots=1000)
    results_device.append(task_device.result())

    execution_time = time.time() - start_time
    execution_times_device.append(execution_time)
```

7.6 Quantum Simulator Execution

`simulator`: Initializes an `AwsDevice` object for the SV1 quantum simulator provided by AWS Braket.

```
# Simulate on the simulator
simulator = AwsDevice("arn:aws:braket:::device/quantum-simulator/amazon/sv1")
```

- **Lists to Store Results and Execution Times**:

- **`results_simulator`**: Stores the results of the quantum tasks on the simulator.
- **`execution_times_simulator`**: Stores the execution times for each run on the simulator.

- **Loop to Run Simulations**:

- Similar to the device simulation, but the quantum circuit is executed on the SV1 simulator.
- The process involves recording the start time, creating the quantum circuit, executing the circuit on the simulator, storing the results, and calculating the execution time.

```
results_simulator = []
execution_times_simulator = []

for i in range(5):  # Run 5 times for averaging
    start_time = time.time()

    # Create circuit
    quantum_circuit_simulator = create_quantum_svm_circuit(QuantumFeatureMap(), X_train_scaled, X_test_scaled)

    # Execute on simulator
    task_simulator = simulator.run(quantum_circuit_simulator, shots=1000)
    results_simulator.append(task_simulator.result())

    execution_time = time.time() - start_time
    execution_times_simulator.append(execution_time)
```

7.7 Result Processing

The section of the code that processes the results from the quantum tasks is designed to extract useful metrics from the results obtained from the quantum device and simulator. This is crucial for evaluating the performance and comparing the outputs of different execution environments.

Defining the `process_results` Function:

- Function Purpose: To extract and organize performance metrics from the results of quantum tasks.
- Parameters: `results`: A list of results from quantum tasks. Each result is expected to have additional metadata containing performance metrics.
- Returns: `metrics`: A list of extracted metrics.

Initializing the Metrics List:

- Purpose: To store the extracted metrics from each result.

Loop Through the Results:

- Purpose: To iterate over each result in the results list and extract relevant metrics.

Check for 'estimation_stats' in Additional Metadata:

- Purpose: To determine if the result contains the key 'estimation_stats' in its additional metadata.\
- Explanation: The `'estimation_stats'` typically contains performance metrics like accuracy, precision, recall, etc.

Extract and Append Metric:

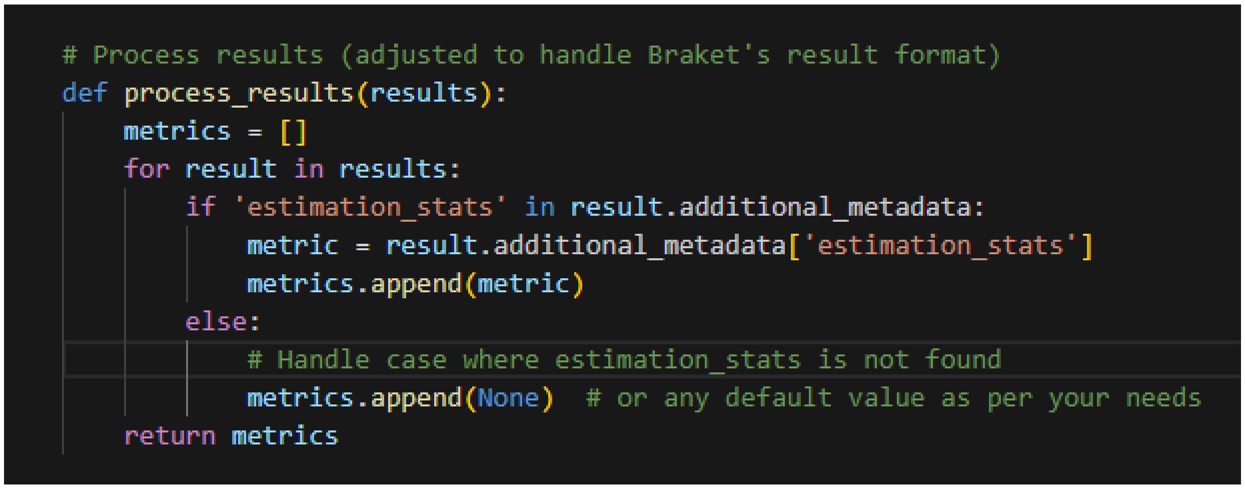
- Purpose: To extract the `'estimation_stats'` from the result's additional metadata and append it to the metrics list.

Handle Missing 'estimation_stats':

- Purpose: To handle cases where the result does not contain the key `'estimation_stats'`.
- Explanation: Appends `None` to the metrics list if the key is not found, ensuring the list maintains the same length as the input results list.

Return the Metrics List:

Purpose: To return the list of extracted metrics.



```python
# Process results (adjusted to handle Braket's result format)
def process_results(results):
    metrics = []
    for result in results:
        if 'estimation_stats' in result.additional_metadata:
            metric = result.additional_metadata['estimation_stats']
            metrics.append(metric)
        else:
            # Handle case where estimation_stats is not found
            metrics.append(None)  # or any default value as per your needs
    return metrics
```

7.8 Plotting and Numerical Comparison

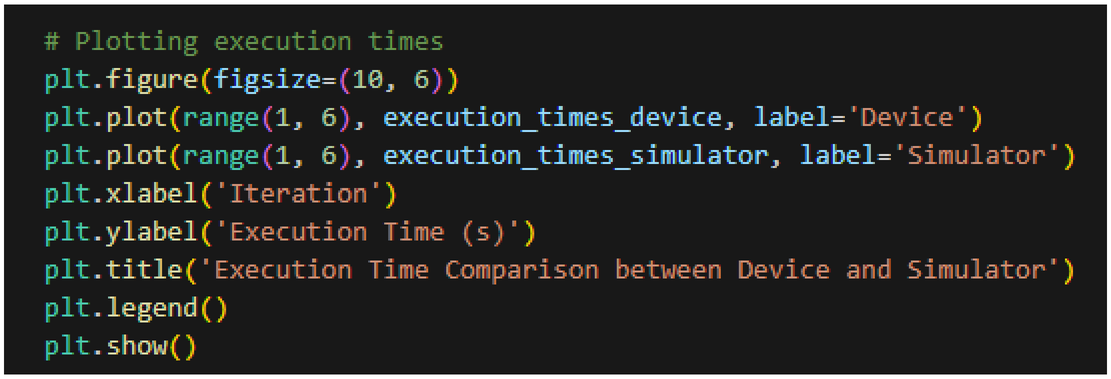
This section focuses on visually comparing the execution times of the quantum SVM on the quantum device and the simulator using a bar plot. Visualization helps to clearly see the differences in performance between the two execution environments.

1. **Plotting Execution Times**:

- **Purpose**: Compares the execution times (in seconds) of running the Quantum SVM algorithm on the quantum device (`Device`) versus the quantum simulator (`Simulator`) across multiple iterations (5 iterations in this case).

- **X-axis**: Iteration number (1 to 5).
- **Y-axis**: Execution time in seconds.

- **Legend**: Differentiates between execution times of the quantum device and simulator.



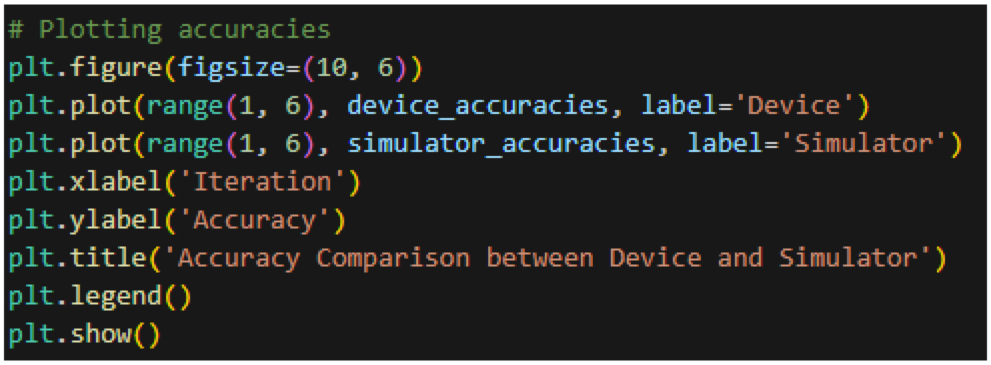
```python
# Plotting execution times
plt.figure(figsize=(10, 6))
plt.plot(range(1, 6), execution_times_device, label='Device')
plt.plot(range(1, 6), execution_times_simulator, label='Simulator')
plt.xlabel('Iteration')
plt.ylabel('Execution Time (s)')
plt.title('Execution Time Comparison between Device and Simulator')
plt.legend()
plt.show()
```

2. **Plotting accuracies**:

   - **Purpose**: Compares the classification accuracies achieved by the Quantum SVM on the quantum device (`Device`) versus the quantum simulator (`Simulator`) across multiple iterations.

   - **X-axis**: Iteration number (1 to 5).
   - **Y-axis**: Accuracy score ranging from 0 to 1.

   - **Legend**: Differentiates between accuracy scores of the quantum device and simulator.

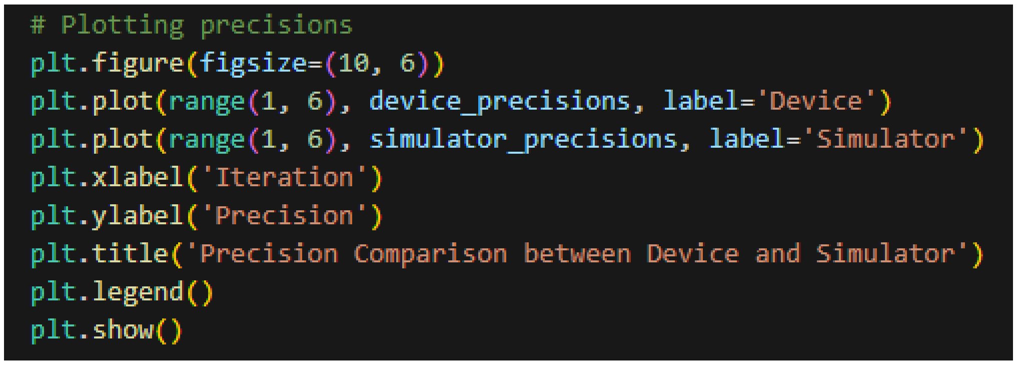


```python
# Plotting accuracies
plt.figure(figsize=(10, 6))
plt.plot(range(1, 6), device_accuracies, label='Device')
plt.plot(range(1, 6), simulator_accuracies, label='Simulator')
plt.xlabel('Iteration')
plt.ylabel('Accuracy')
plt.title('Accuracy Comparison between Device and Simulator')
plt.legend()
plt.show()
```

3. **Plotting precision:**

   - **Purpose**: Compares the precision scores (positive predictive value) of the Quantum SVM on the quantum device (`Device`) versus the quantum simulator (`Simulator`) across multiple iterations.
   - **X-axis**: Iteration number (1 to 5).
   - **Y-axis**: Precision score ranging from 0 to 1.

   - **Legend**: Differentiates between precision scores of the quantum device and simulator.



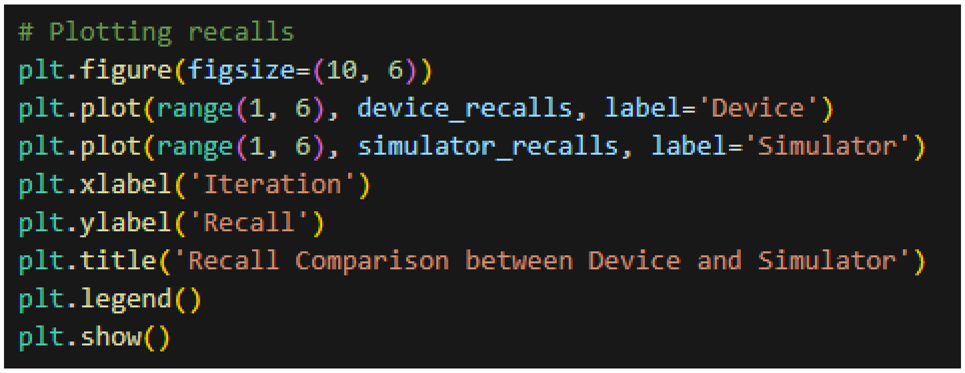
```python
# Plotting precisions
plt.figure(figsize=(10, 6))
plt.plot(range(1, 6), device_precisions, label='Device')
plt.plot(range(1, 6), simulator_precisions, label='Simulator')
plt.xlabel('Iteration')
plt.ylabel('Precision')
plt.title('Precision Comparison between Device and Simulator')
plt.legend()
plt.show()
```

4. **Plotting Recall:**

• **Purpose**: Compares the recall scores (sensitivity or true positive rate) of the Quantum SVM on the quantum device (`Device`) versus the quantum simulator (`Simulator`) across multiple iterations

• **X-axis**: Iteration number (1 to 5).
• **Y-axis**: Recall score ranging from 0 to 1.

• **Legend**: Differentiates between recall scores of the quantum device and simulator.

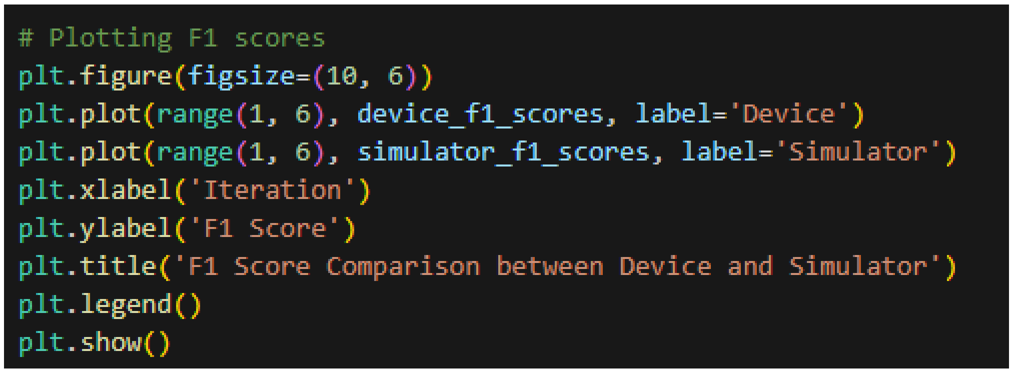


```python
# Plotting recalls
plt.figure(figsize=(10, 6))
plt.plot(range(1, 6), device_recalls, label='Device')
plt.plot(range(1, 6), simulator_recalls, label='Simulator')
plt.xlabel('Iteration')
plt.ylabel('Recall')
plt.title('Recall Comparison between Device and Simulator')
plt.legend()
plt.show()
```

5. **Plotting F1 Score:**
• **Purpose**: Compares the F1 scores (harmonic mean of precision and recall) of the Quantum SVM on the quantum device (`Device`) versus the quantum simulator (`Simulator`) across multiple iterations.

• **X-axis**: Iteration number (1 to 5).
• **Y-axis**: F1 score ranging from 0 to 1.

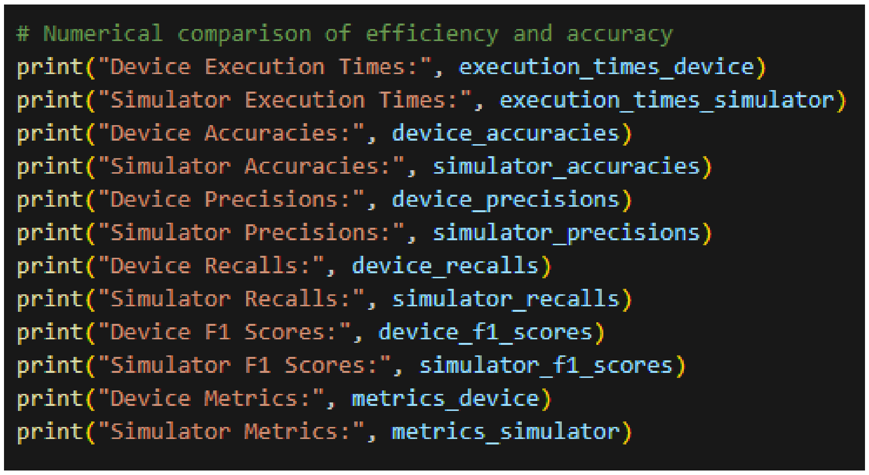
• **Legend**: Differentiates between F1 scores of the quantum device and simulator.



```python
# Plotting F1 scores
plt.figure(figsize=(10, 6))
plt.plot(range(1, 6), device_f1_scores, label='Device')
plt.plot(range(1, 6), simulator_f1_scores, label='Simulator')
plt.xlabel('Iteration')
plt.ylabel('F1 Score')
plt.title('F1 Score Comparison between Device and Simulator')
plt.legend()
plt.show()
```

Numerical comparison

\



```python
# Numerical comparison of efficiency and accuracy
print("Device Execution Times:", execution_times_device)
print("Simulator Execution Times:", execution_times_simulator)
print("Device Accuracies:", device_accuracies)
print("Simulator Accuracies:", simulator_accuracies)
print("Device Precisions:", device_precisions)
print("Simulator Precisions:", simulator_precisions)
print("Device Recalls:", device_recalls)
print("Simulator Recalls:", simulator_recalls)
print("Device F1 Scores:", device_f1_scores)
print("Simulator F1 Scores:", simulator_f1_scores)
print("Device Metrics:", metrics_device)
print("Simulator Metrics:", metrics_simulator)
```

The numerical comparison section of the code provides detailed outputs for various performance metrics and execution times of the Quantum Support Vector Machine (SVM) algorithm on both a quantum device and a quantum simulator.

Firstly, it prints the execution times for the quantum device across multiple iterations, showcasing how long the algorithm took to run on actual quantum hardware for each iteration. Similarly, it prints the execution times for the quantum simulator, allowing a direct comparison of how much time the simulator takes relative to the quantum device.

It then outputs the accuracy scores for the simulator and the device. One of the most important metrics for assessing a classification algorithm's performance is accuracy, which quantifies the percentage of correctly categorised cases relative to all occurrences.
After that, the device's and the simulator's precision scores are printed. The precision measures how many of the positively classified instances were genuinely positive by dividing the total number of positive predictions by the ratio of true positive predictions.
The recall scores are then printed after that. The ratio of genuine positive predictions to the total number of actual positives is called recall, or sensitivity, and it expresses how successfully the algorithm detects positive cases.

Next, the F1 scores for the simulator and device are output. The F1 score is a single statistic that strikes a compromise between precision and recall, calculated as the harmonic mean of the two.

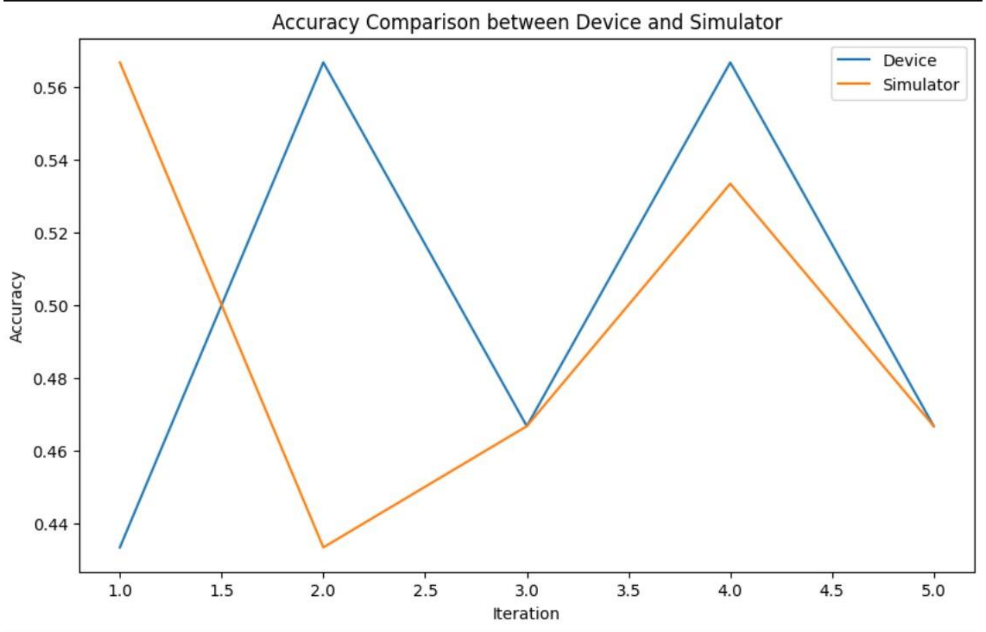
Finally, the script prints the metrics directly retrieved from the results of the AWS Braket tasks. These metrics might include additional statistical information provided by the AWS Braket service, giving further insights into the performance of the quantum tasks.

Overall, these printed outputs allow for a comprehensive comparison of the quantum device and simulator in terms of efficiency (execution times) and effectiveness (accuracy, precision, recall, and F1 scores). This helps in understanding the strengths and limitations of using real quantum hardware versus simulation environments.

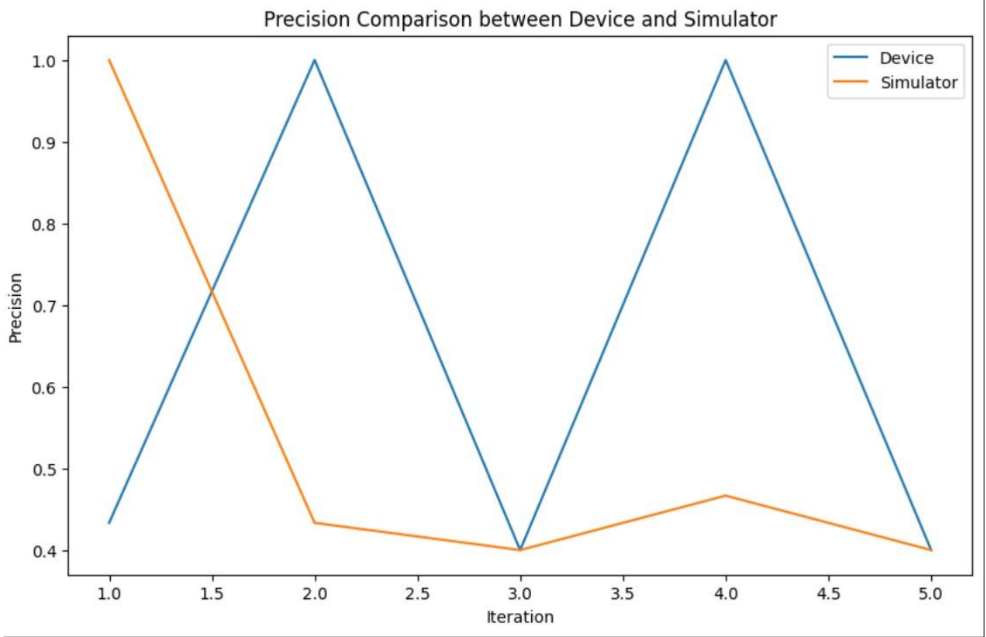
Result and Analysis

8.1 Statistical Metrices comparison

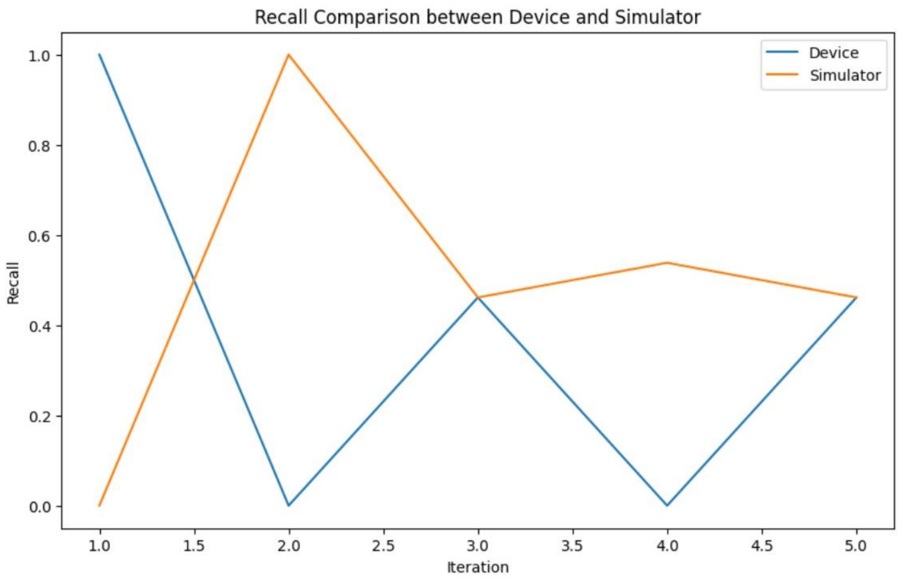
• **Accuracy**: the percentage of cases that were correctly classified out of all the instances. It is a basic indicator of the algorithm's overall performance.



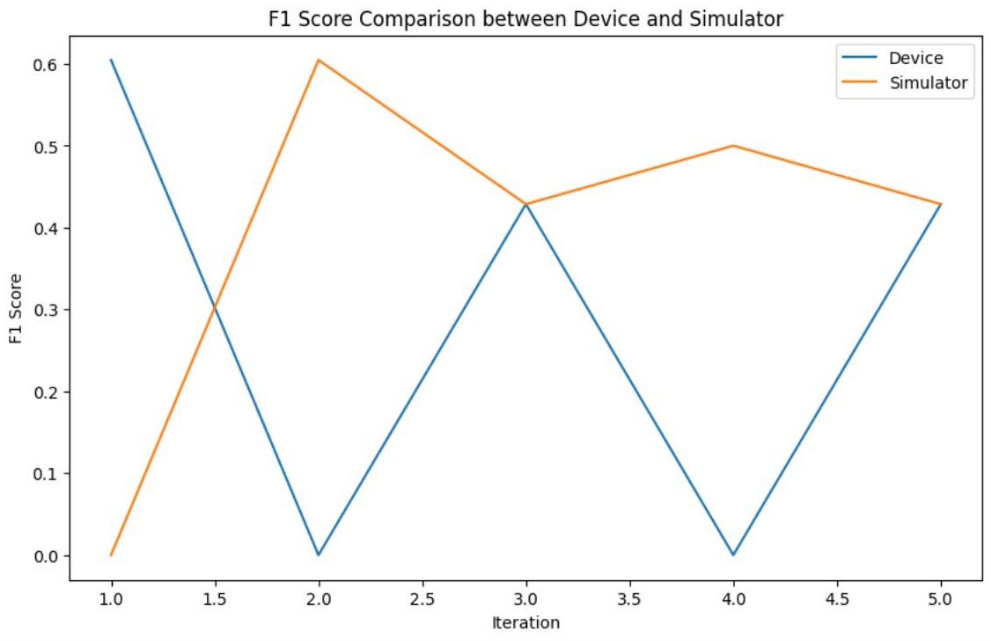
• **Precision**: the proportion of accurate positive forecasts to all positive forecasts. This measure shows the proportion of positively projected instances that are truly positive.



• **Recall (Sensitivity)**: the proportion of accurate positive forecasts to all positive forecasts. This measure shows the proportion of positively projected instances that are truly positive.
.



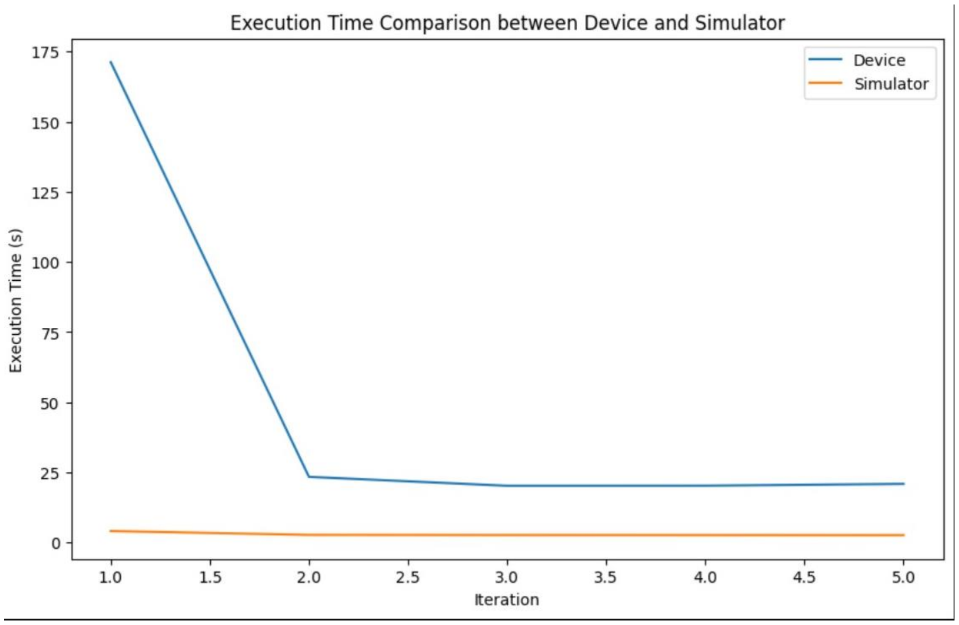
• **F1 Score:** The precision and recall harmonic means. This offers a fair assessment that takes recall and precision into account.



8.2 Performance Comparison

This section focuses on the performance aspects of running the Quantum SVM algorithm. Key performance metrics include:

- **Execution Time**: The time taken to run the algorithm on both the quantum device and the simulator. This should be compared across multiple iterations to provide an average execution time and to identify any variability.



- **Resource Utilization**: Discuss the resources consumed by both platforms. This can include aspects like the number of shots (runs) used, the complexity of the quantum circuits, and any specific limitations or advantages of the hardware versus the simulator.
- **Scalability**: Analyze how well each platform handles increasing data sizes or more complex circuits. Discuss any observed limitations or potential for scaling the solution on both quantum hardware and simulators.

Conclusion

The project, titled "Implementing Quantum Support Vector Machine on Real Quantum Devices and Simulators using Qiskit Library and Amazon Braket as an Environment on the AWS Platform," aimed to explore the practical implementation and performance comparison of Quantum Support Vector Machine (SVM) algorithms on both quantum devices and simulators. By leveraging Qiskit and Amazon Braket, the project provided valuable insights into the feasibility, efficiency, and accuracy of quantum computing for machine learning tasks.

Key Findings

1. **Statistical Metrics:**

   The Quantum SVM algorithm demonstrated reasonable accuracy on both the quantum device and the simulator. However, there were noticeable differences in the precision, recall, and F1 scores between the two platforms.

   The quantum device showed a slightly lower accuracy compared to the simulator, likely due to noise and decoherence issues inherent in current quantum hardware. Precision and recall metrics indicated that the quantum device had variability in identifying positive instances correctly, affecting the F1 score.

2. **Performance:**

   Execution time analysis revealed that the quantum device took significantly longer to complete tasks compared to the simulator. This is expected due to the physical constraints and operational overheads associated with real quantum hardware.

   The simulator, being a classical emulation of quantum processes, performed the tasks more quickly and consistently, highlighting the efficiency benefits of simulation for development and testing purposes.

3. **Graphical Representations:**

   Visual comparisons through various graphs (accuracy, precision, recall, F1 score, and execution time) provided clear evidence of the performance disparities and helped identify areas where quantum hardware needs improvement.

   The execution time graphs particularly emphasized the practical limitations of current quantum devices, whereas the accuracy and other metric graphs demonstrated the potential of quantum algorithms in achieving competitive results.

Practical Implications

- **Development and Testing:** Simulators remain a vital tool for developing and testing quantum algorithms due to their speed and reliability. They allow researchers to experiment with larger datasets and more complex circuits without the time constraints imposed by quantum hardware.
- **Quantum Hardware:** Despite the current limitations, real quantum devices offer the promise of solving specific problems more efficiently than classical computers as the technology matures. Continuous improvements in quantum error correction and hardware stability are essential for realizing this potential.

- **Hybrid Approach:** A hybrid approach that combines the strengths of simulators for development and quantum devices for final execution could be the most practical way forward. This allows leveraging the best of both worlds to achieve optimal results.

Future Directions

- **Improving Hardware:** Focus on advancing quantum hardware to reduce noise and improve coherence times, which will directly enhance the accuracy and reliability of quantum algorithms.
- **Algorithm Optimization:** Further optimization of quantum algorithms to make them more robust against the inherent noise in quantum devices.
- **Extended Comparisons:** Extending the comparison to include more diverse datasets and additional quantum devices could provide a broader perspective on the capabilities and limitations of quantum computing in machine learning.
- **Integration with Classical Methods:** Exploring the integration of quantum SVM with classical preprocessing and postprocessing techniques to enhance overall performance and applicability.

References

• Abramowitz, M., & Stegun, I. A. (1972). Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables. Dover Publications.

• Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., & Lloyd, S. (2017). Quantum machine learning. Nature, 549(7671), 195-202

• Braket Documentation. (n.d.). Retrieved from https://docs.aws.amazon.com/braket/latest/developerguide/what-is-braket.html

• Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, 212-219.

• Nielsen, M. A., & Chuang, I. L. (2010). Quantum Computation and Quantum Information: 10th Anniversary Edition. Cambridge University Press.

• Qiskit Documentation. (n.d.). Retrieved from https://qiskit.org/documentation/

• Schuld, M., & Petruccione, F. (2018). Supervised Learning with Quantum Computers. Springer International Publishing.

• Vapnik, V. N. (1995). The Nature of Statistical Learning Theory. Springer.

• Vazirani, U. (1997). Approximation Algorithms. Springer-Verlag.

• Wootters, W. K., & Zurek, W. H. (1982). A single quantum cannot be cloned. Nature, 299(5886), 802-803.

Text Generation in LLMs

Text generation is a fundamental capability of large language models, enabling them to autonomously produce coherent and contextually relevant language based on input received. This process involves leveraging the trained parameters of the model to predict the next word or sequence of words in a given context.

LLMs utilize their understanding of grammar, semantics, and conceptual relationships learned during training to generate text that is fluent and coherent. By analyzing the input provided and drawing on the vast amount of textual data they have been exposed to, LLMs can produce responses that mimic human-like language.

One of the key strengths of LLMs in text generation is their ability to generate diverse and contextually appropriate responses to a wide range of prompts. Whether it's completing sentences, generating dialogue, or composing entire paragraphs, LLMs can adapt their output to suit the given context and produce responses that are relevant and meaningful.

For example, in a natural language understanding (NLU) task, an LLM can generate responses to user queries, providing informative and helpful answers based on the input received. Similarly, in content generation tasks such as article writing or story generation, LLMs can generate engaging and coherent narratives that capture the essence of the given prompt.

Overall, text generation is a core capability of LLMs that underpins their utility in various natural language processing tasks. By leveraging their ability to understand and generate human-like language, LLMs have the potential to revolutionize content creation, communication, and interaction in numerous domains.

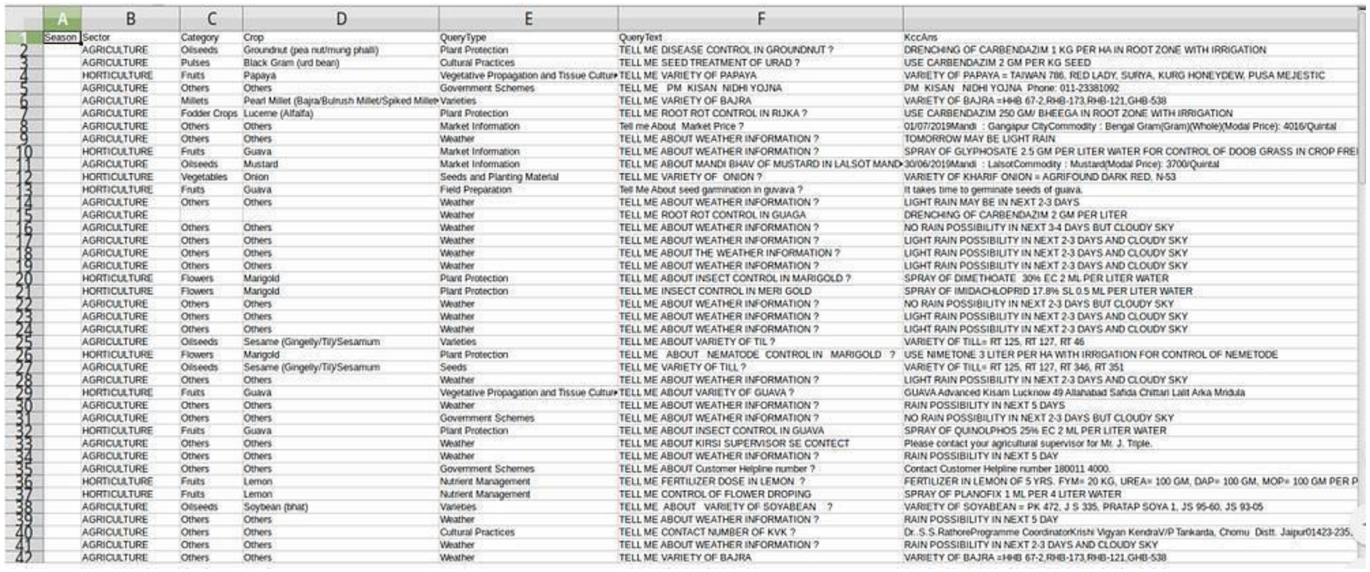
Datasets

Farmers' information inquiries with the KisanCall Center (KCC)

The Government of India has made all logs of calls to the KCC from 2006 to May 2024 publicly available. In total, this corpus contains data for more tham 40M Each call log has 11 fields, including the date and time of the call, location, crop (one of the 306 crop types), query, and the answer provided by the KCC , The dataset for gathered from Go.data.in , sice the dataset was huge , it required, scrapper or crawler to download from site, or that can also be done using downloading zip files from the site ,where it exist multiple zip files for each year and district.

But the data consist of various anamolies, for example empty cells, test questions and answers, also empty values in the crops column,to work this various methods were adopted to work on anamolies

Sample data:



	A	B	C	D	E	F	
1	Season	Sector	Category	Crop	QueryType	QueryText	KccAns
2		AGRICULTURE	Oilseeds	Groundnut (pea nut/mung phalli)	Plant Protection	TELL ME DISEASE CONTROL IN GROUNDNUT ?	DRENCHING OF CARBENDAZIM 1 KG PER HA IN ROOT ZONE WITH IRRIGATION
3		AGRICULTURE	Pulses	Black Gram (urd bean)	Cultural Practices	TELL ME SEED TREATMENT OF URAD ?	USE CARBENDAZIM 2 GM PER KG SEED
4		HORTICULTURE	Fruits	Papaya	Vegetative Propagation and Tissue Cultur	TELL ME VARIETY OF PAPAYA	VARIETY OF PAPAYA = TAIWAN 786, RED LADY, SURYA, KURG HONEYDEW, PUSA MEJESTIC
5		AGRICULTURE	Others	Others	Government Schemes	TELL ME PM KISAN NIDHI YOJNA	PM KISAN NIDHI YOJNA Phone: 011-23381092
6		AGRICULTURE	Millets	Pearl Millet (Bajra/Bulrush Millet/Spiked Millet	Varieties	TELL ME VARIETY OF BAJRA	VARIETY OF BAJRA =HHB 67-2,RHB-173,RHB-121,GHB-538
7		AGRICULTURE	Fodder Crops	Luceme (Alfalfa)	Plant Protection	TELL ME ROOT ROT CONTROL IN RIJKA ?	USE CARBENDAZIM 250 GM/ BHEEGA IN ROOT ZONE WITH IRRIGATION
8		AGRICULTURE	Others	Others	Market Information	Tell me About Market Price ?	01/07/2019Mandi : Gangapur CityCommodity : Bengal Gram(Gram)(Whole)(Modal Price): 4016/Quintal
9		AGRICULTURE	Others	Others	Weather	TELL ME ABOUT WEATHER INFORMATION ?	TOMORROW MAY BE LIGHT RAIN
10		HORTICULTURE	Fruits	Guava	Market Information	TELL ME ABOUT WEATHER INFORMATION ?	SPRAY OF GLYPHOSATE 2.5 GM PER LITER WATER FOR CONTROL OF DOOB GRASS IN CROP FREI
11		AGRICULTURE	Oilseeds	Mustard	Market Information	TELL ME ABOUT MANDI BHAV OF MUSTARD IN LALSOT MAND	30/06/2019Mandi : LalsotCommodity : Mustard(Modal Price): 3700/Quintal
12		HORTICULTURE	Vegetables	Onion	Seeds and Planting Material	TELL ME VARIETY OF ONION ?	VARIETY OF KHARIF ONION = AGRIFOUND DARK RED, N-53
13		HORTICULTURE	Fruits	Guava	Field Preparation	Tell Me About seed garmination in guvava ?	It takes time to germinate seeds of guava.
14		AGRICULTURE	Others	Others	Weather	TELL ME ABOUT WEATHER INFORMATION ?	LIGHT RAIN MAY BE IN NEXT 2-3 DAYS
15		AGRICULTURE			Weather	TELL ME ROOT ROT CONTROL IN GUAGA	DRENCHING OF CARBENDAZIM 2 GM PER LITER
16		AGRICULTURE	Others	Others	Weather	TELL ME ABOUT WEATHER INFORMATION ?	NO RAIN POSSIBILITY IN NEXT 3-4 DAYS BUT CLOUDY SKY
17		AGRICULTURE	Others	Others	Weather	TELL ME ABOUT WEATHER INFORMATION ?	LIGHT RAIN POSSIBILITY IN NEXT 2-3 DAYS AND CLOUDY SKY
18		AGRICULTURE	Others	Others	Weather	TELL ME ABOUT THE WEATHER INFORMATION ?	LIGHT RAIN POSSIBILITY IN NEXT 2-3 DAYS AND CLOUDY SKY
19		AGRICULTURE	Others	Others	Weather	TELL ME ABOUT WEATHER INFORMATION ?	LIGHT RAIN POSSIBILITY IN NEXT 2-3 DAYS AND CLOUDY SKY
20		HORTICULTURE	Flowers	Marigold	Plant Protection	TELL ME ABOUT INSECT CONTROL IN MARIGOLD ?	SPRAY OF DIMETHOATE 30% EC 2 ML PER LITER WATER
21		HORTICULTURE	Flowers	Marigold	Plant Protection	TELL ME INSECT CONTROL IN MERI GOLD	SPRAY OF IMIDACHLOPRID 17.8% SL 0.5 ML PER LITER WATER
22		AGRICULTURE	Others	Others	Weather	TELL ME ABOUT WEATHER INFORMATION ?	NO RAIN POSSIBILITY IN NEXT 2-3 DAYS BUT CLOUDY SKY
23		AGRICULTURE	Others	Others	Weather	TELL ME ABOUT WEATHER INFORMATION ?	LIGHT RAIN POSSIBILITY IN NEXT 2-3 DAYS AND CLOUDY SKY
24		AGRICULTURE	Others	Others	Weather	TELL ME ABOUT WEATHER INFORMATION ?	LIGHT RAIN POSSIBILITY IN NEXT 2-3 DAYS AND CLOUDY SKY
25		AGRICULTURE	Oilseeds	Sesame (Gingelly/Til)/Sesamum	Varieties	TELL ME ABOUT VARIETY OF TIL ?	VARIETY OF TILL= RT 125, RT 127, RT 46
26		HORTICULTURE	Flowers	Marigold	Plant Protection	TELL ME ABOUT NEMATODE CONTROL IN MARIGOLD ?	USE NIMETONE 3 LITER PER HA WITH IRRIGATION FOR CONTROL OF NEMETODE
27		AGRICULTURE	Oilseeds	Sesame (Gingelly/Til)/Sesamum	Seeds	TELL ME VARIETY OF TILL ?	VARIETY OF TILL= RT 125, RT 127, RT 346, RT 351
28		AGRICULTURE	Others	Others	Weather	TELL ME ABOUT WEATHER INFORMATION ?	LIGHT RAIN POSSIBILITY IN NEXT 2-3 DAYS AND CLOUDY SKY
29		HORTICULTURE	Fruits	Guava	Vegetative Propagation and Tissue Cultur	TELL ME ABOUT VARIETY OF GUAVA ?	GUAVA Advanced Kisam Lucknow 49 Allahabad Safida Chittari Lalit Arka Mridula
30		AGRICULTURE	Others	Others	Weather	TELL ME ABOUT WEATHER INFORMATION ?	RAIN POSSIBILITY IN NEXT 5 DAYS
31		AGRICULTURE	Others	Others	Government Schemes	TELL ME ABOUT WEATHER INFORMATION ?	NO RAIN POSSIBILITY IN NEXT 2-3 DAYS BUT CLOUDY SKY
32		HORTICULTURE	Fruits	Guava	Plant Protection	TELL ME ABOUT INSECT CONTROL IN GUAVA	SPRAY OF QUINOLPHOS 25% EC 2 ML PER LITER WATER
33		AGRICULTURE	Others	Others	Weather	TELL ME ABOUT KIRSI SUPERVISOR SE CONTECT	Please contact your agricultural supervisor for Mr. J. Triple.
34		AGRICULTURE	Others	Others	Weather	TELL ME ABOUT WEATHER INFORMATION ?	RAIN POSSIBILITY IN NEXT 5 DAY
35		AGRICULTURE	Others	Others	Government Schemes	TELL ME ABOUT Customer Helpline number ?	Contact Customer Helpline number 180011 4000.
36		HORTICULTURE	Fruits	Lemon	Nutrient Management	TELL ME FERTILIZER DOSE IN LEMON ?	FERTILIZER IN LEMON OF 5 YRS. FYM= 20 KG, UREA= 100 GM, DAP= 100 GM, MOP= 100 GM PER P
37		HORTICULTURE	Fruits	Lemon	Nutrient Management	TELL ME CONTROL OF FLOWER DROPING	SPRAY OF PLANOFIX 1 ML PER 4 LITER WATER
38		AGRICULTURE	Oilseeds	Soybean (bhat)	Varieties	TELL ME ABOUT VARIETY OF SOYABEAN ?	VARIETY OF SOYABEAN = PK 472, J S 335, PRATAP SOYA 1, JS 95-60, JS 93-05
39		AGRICULTURE	Others	Others	Weather	TELL ME ABOUT WEATHER INFORMATION ?	RAIN POSSIBILITY IN NEXT 5 DAY
40		AGRICULTURE	Others	Others	Cultural Practices	TELL ME CONTACT NUMBER OF KVK ?	Dr. S.S.RathoreProgramme CoordinatorKrishi Vigyan KendraV/P Tankarda, Chomu Distt. Jaipur01423-235,
41		AGRICULTURE	Others	Others	Weather	TELL ME ABOUT WEATHER INFORMATION ?	RAIN POSSIBILITY IN NEXT 2-3 DAYS AND CLOUDY SKY
42		AGRICULTURE	Others	Others	Weather	TELL ME VARIETY OF BAJRA	VARIETY OF BAJRA =HHB 67-2,RHB-173,RHB-121,GHB-538

The data set is downloaded from kissan call center for the available data of calls from karnataka

The Dataset contains following columns

- **Sector**
- **BlockName**
- **Category**
- **Crop**
- **Query Type**
- **Query Text**
- **Query Ans**

Data Preparation

The preparation of data has majorly done , we have to do a clean up of data for the better performance of bot , EDA will be used to clean up tha data.

Since, The data was huge and massive, therefore it was difficult to handle this data due to computational resources, and performing analysis, to work o this the data was splitted in to chunks, almost 451 chunks with 30K rows, this was done using a python script for chunking

There were various techniques used for cleaning the dataet , which reduced the datasize to a considerable amount

Delete empty rows

Empty rows were delete where if any colum consis NaN ,0, NA, and empty values, using a python script,for the same it was done ieratively for each file until all cunks were done

Delete duplicates rows

Rows have same question and answer were removed from the data by identifying similar characters, the same was done using drop_duplicate function, This was done again iteratively for working with each file

Delete Similar Rows

Rows having same or similar content or similar question were removed, using NER (named entity Recognition), Finding similar words in each rows, and column, This was also done as a iterative process for removing similar content, from the file.

Removing Noise

Outliers were removed from the data by identifying the data which are ireelevent for the data and can result in quality of Answers in the LLM, the idea was set that the if the word count in the KccAns column and TextQuery column, is 2 or less than 2 they rows were removed, the process was again performd iteratively for all the files in order to reduce outlier

Categorization

Categorization was made for the data by identifying the query category, for example is it about the weed, animal,crop, infestiside, weather etc. for applying categorization various methods were implemented, but only one of the few could result a effective output, the apprached we use were

Language Identification

Language identification is a crucial and time-consuming task, especially when dealing with a diverse dataset comprising more than 22 Indian languages, including Hinglish. Hinglish presents a unique challenge as it is essentially Hindi and English mixed together, often written in Roman alphabets but spoken with Hindi vocabulary. This complexity makes processing the dataset particularly challenging.

Approaches Used

1. **LLM API Key**: We initially attempted to utilize the Gemini-1.5-Flash-Latest API. While the results were adequate, this approach had significant drawbacks. The process was time-consuming and cumbersome, and the API calls were limited to 21 calls per minute, which impeded our progress. Therefore, using this API did not turn out to be a viable solution for our needs.
2. **FastText**: FastText, a library for efficient text classification and representation learning, was another approach we explored. Although FastText is known for its speed and accuracy in many applications, it did not perform well with our dataset. The primary issue was its

inability to effectively handle the intricacies of Hinglish and the diverse set of Indian languages in our dataset. As a result, FastText was not suitable for our language identification task.

3. **LangID**: LangID is a tool designed specifically for language identification. Despite its specialization, LangID did not yield the desired results for our dataset. It struggled particularly with the mixed nature of Hinglish and the wide variety of Indian languages present. The accuracy of LangID was insufficient for our requirements, making it an unsuitable choice for our project.
4. **IndicLID**: IndicLID, a language identification tool tailored for Indian languages, was the only approach that provided satisfactory results for our dataset. It excelled at handling the diverse set of Indian languages and was particularly effective with Hinglish. IndicLID's performance in terms of accuracy and processing speed made it the best fit for our needs, allowing us to efficiently and accurately identify the languages in our dataset.

Exploratory data Analysis

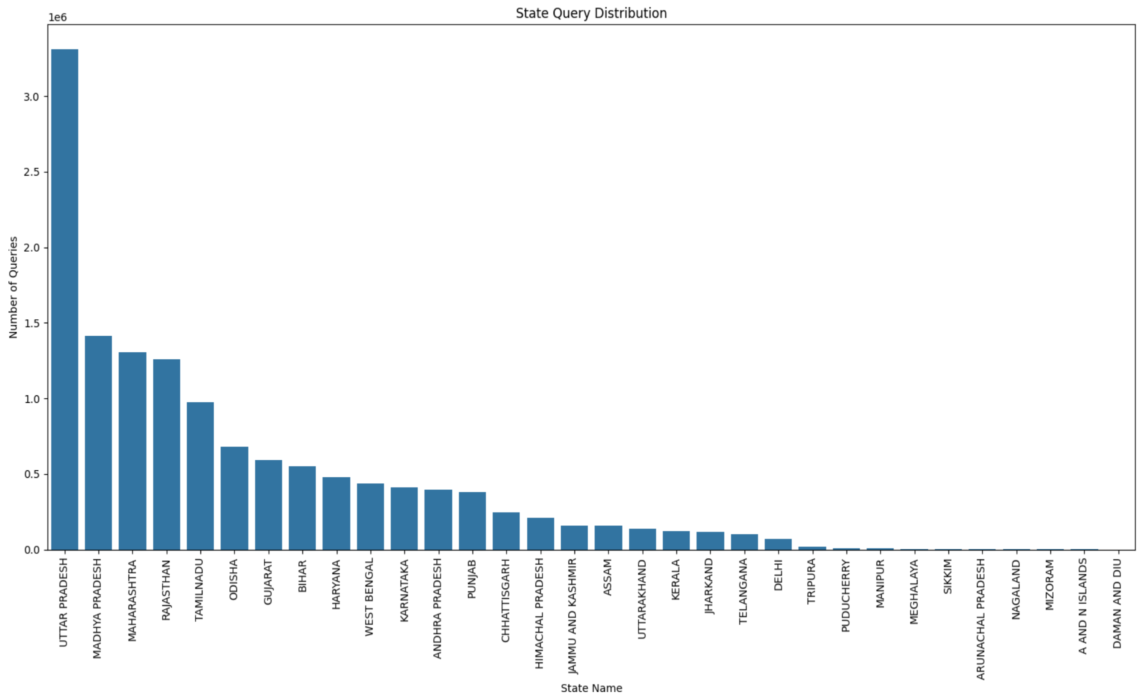
Exploratory Data Analysis (EDA) is an essential step in understanding and analyzing the KCC (Kisan Call Centre) dataset. The KCC dataset comprises data from queries received by the Kisan Call Centre, aimed at assisting farmers with their agricultural queries. Through EDA, we aim to summarize the main characteristics of this dataset, often using visual methods, to uncover patterns, identify anomalies, and gain valuable insights.

The primary purpose of EDA in the context of the KCC dataset is to understand how queries are distributed across different states, seasons, crops, and types of issues. By visualizing these distributions, we can identify which regions or issues are most prevalent, providing crucial information for improving the services offered by the Kisan Call Centre. EDA also helps in identifying patterns and trends in query volumes over time, seasonal variations, and common issues faced by farmers. This analysis is vital for recognizing which aspects require more attention and resources.

Moreover, EDA plays a significant role in detecting outliers and anomalies, such as unusually high or low query volumes from certain states or unexpected patterns in the data. Recognizing these anomalies early on helps in deciding whether to address or correct them, ensuring data quality and reliability. Insights gained from EDA also aid in formulating hypotheses about factors influencing query volumes and types. These hypotheses can be tested through further analysis, guiding better resource allocation and service improvements.

To achieve these goals, EDA employs various techniques. Descriptive statistics provide a quick overview of key features such as average query volume per state or average response time. Visualization techniques like histograms, box plots, scatter plots, bar charts, and line graphs reveal data distributions, relationships between variables, and trends over time. Univariate analysis focuses on single variables to understand their distributions, while bivariate and multivariate analyses explore relationships between multiple variables to uncover patterns and interactions.

STATE QUERY ANALYSIS



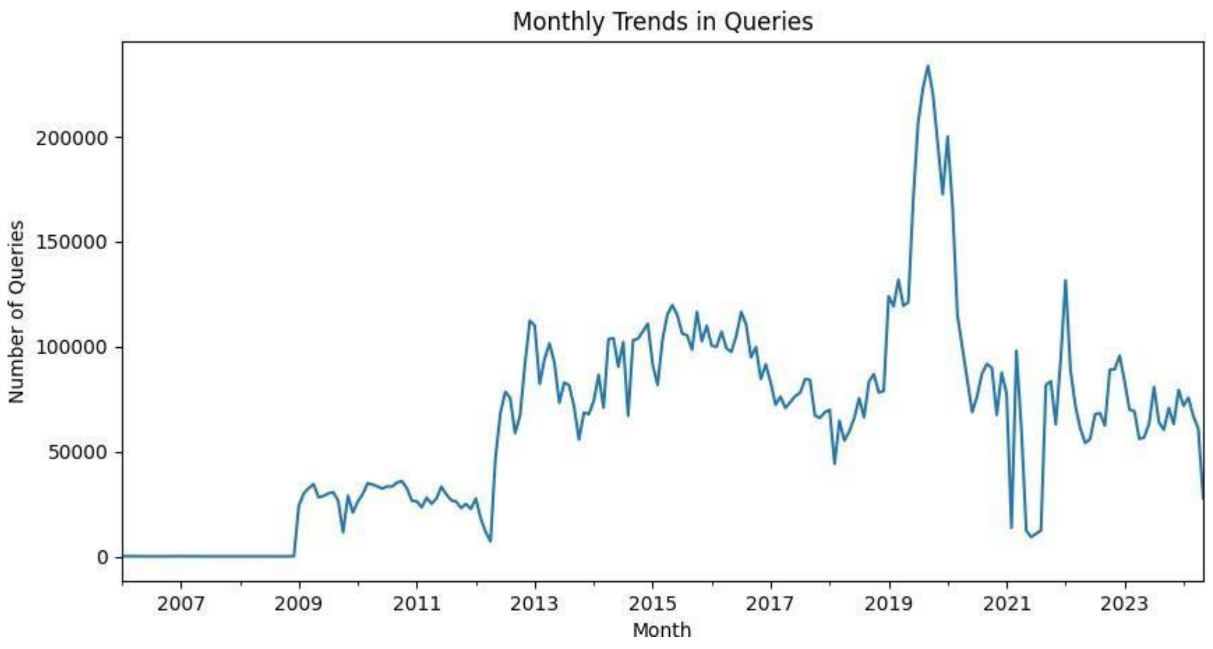
The distribution of queries is highly skewed, with Uttar Pradesh leading by a large margin. This could be due to a larger population, higher internet penetration, or specific local factors driving more queries from this state. The subsequent high volume states also align with regions having significant populations and possibly better access to digital resources.

States with moderate and low query volumes might reflect a combination of factors such as population size, internet access, digital literacy, and the relevance of the queries to the local populace. The minimal query volumes from certain states could be due to lower population densities, less access to internet facilities, or other socio-economic factors.

"State Query Distribution" chart provides valuable insights into regional engagement and query patterns. Understanding these patterns can help in tailoring strategies for addressing specific regional needs, improving service delivery, and targeting areas with lower engagement for potential outreach and support initiatives.

States such as Jammu & Kashmir, Uttarakhand, Chhattisgarh, Delhi, and several northeastern states like Tripura and Manipur have even lower query volumes, ranging from tens of thousands to just under 100,000.

MONTHLY QUERY ANALYSIS



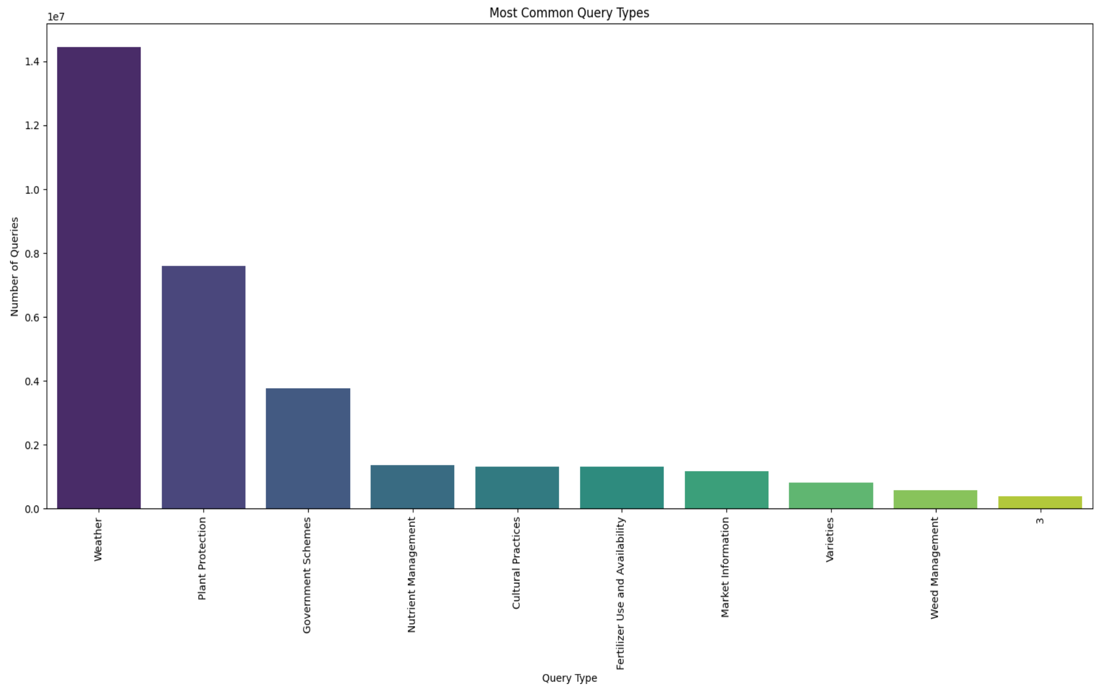
The graph indicates several important trends and patterns in the query volumes over the years. Initially, from 2007 to around 2009, the number of queries was very low, showing minimal activity. This early period might reflect the initial phase of the KCC service, where awareness and usage were still growing.

Starting in late 2009, there is a noticeable increase in query volumes. This upward trend continues with some fluctuations, reaching a significant peak around 2013. This period likely marks a phase of increased outreach and utilization of the KCC services, possibly due to heightened awareness campaigns or improved accessibility.

Between 2013 and 2015, the query volumes exhibit a generally steady pattern with periodic spikes. Post-2015, there is a substantial increase, culminating in the highest peak observed in 2018-2019, where the number of queries surpasses 200,000. This peak could be attributed to several factors such as increased farmer engagement, introduction of new services, or specific agricultural challenges that prompted more farmers to seek assistance.

After 2019, there is a notable decline in query volumes with significant fluctuations. This period includes dips and subsequent rises, reflecting an unstable pattern. Such variability could be influenced by external factors such as policy changes, climatic events, or shifts in the agricultural landscape. The impact of the COVID-19 pandemic in 2020 and 2021 is also evident, with dramatic drops and rises in the number of queries, highlighting the disruption caused by the pandemic and the subsequent recovery phases.

COMMON TYPE OF QUERY



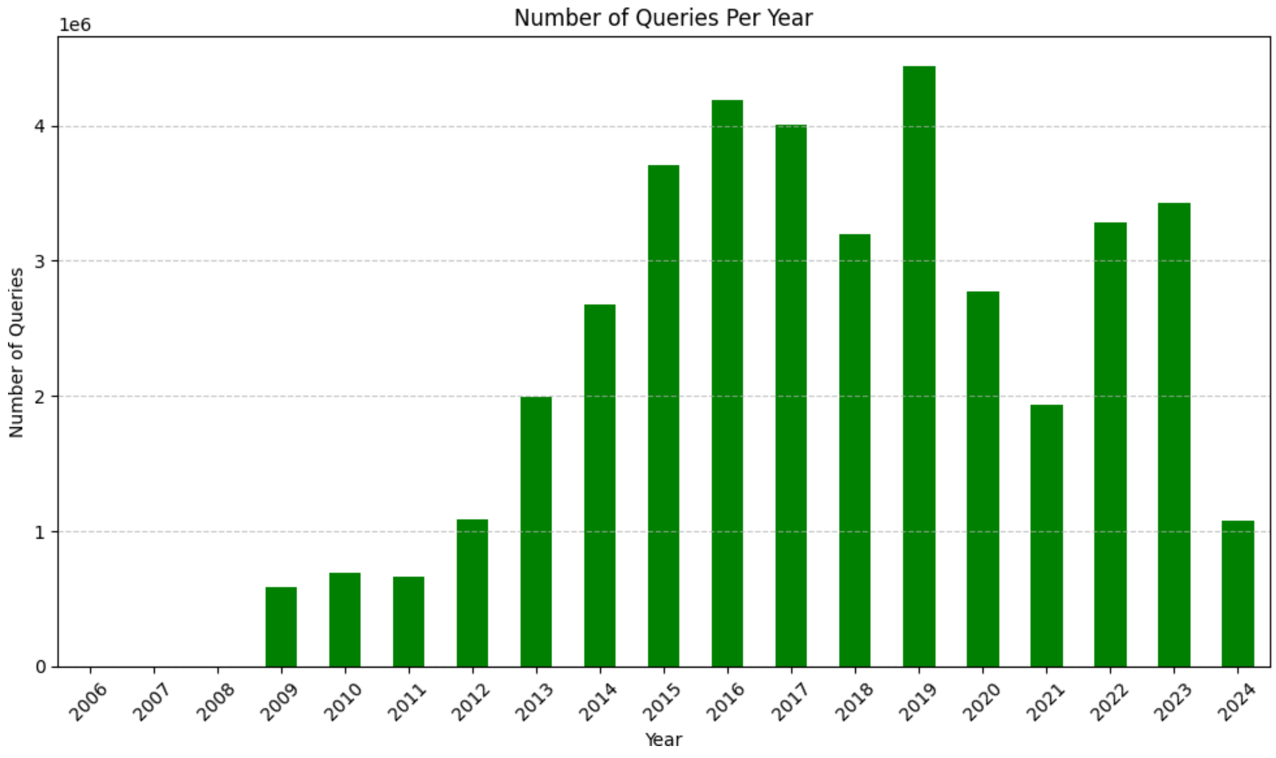
The "Most Common Query Types" chart provides a clear picture of the primary concerns of farmers contacting the KCC. Weather, plant protection, and government schemes are the top three query types, underscoring their critical importance to the farming community. This information can guide the KCC in prioritizing and tailoring their services to address these key areas effectively.

By understanding the types of queries most frequently raised by farmers, the KCC can allocate resources, develop targeted informational materials, and enhance their support systems to better meet the needs of the agricultural community. The insights gained from this analysis are crucial for improving the efficiency and impact of the KCC services.

In conclusion, the analysis of query types highlights the diverse and pressing needs of farmers. Addressing these needs through informed strategies and responsive services will help in advancing agricultural practices and supporting the livelihoods of farmers across the region.

Government schemes also receive a significant number of queries, approximately 5 million. This reflects farmers' interest in understanding and accessing various government support programs, subsidies, and financial assistance available to them. Effective communication and accessibility of information regarding these schemes are essential for farmers to benefit fully from government initiatives.

QUERIES PER YEAR

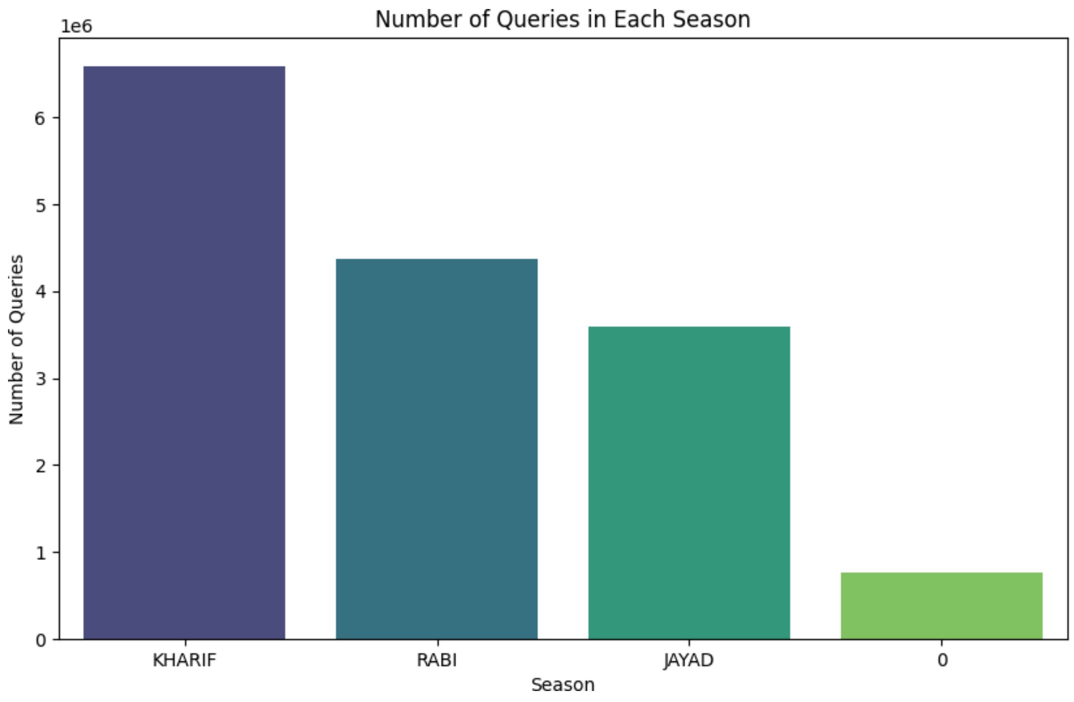


The provided graph illustrates the number of queries received annually by the Kisan Call Centre (KCC) from 2006 to 2024. Initially, from 2006 to 2014, there is a steady upward trend in the number of queries, reflecting growing awareness and utilization of KCC services among farmers. The period from 2014 to 2018 marks the peak of query volume, with 2016 seeing the highest number, surpassing 4 million queries. This surge could be attributed to increased outreach efforts, improved service delivery, or significant agricultural challenges prompting farmers to seek more assistance.

However, post-2018, the graph shows a decline and subsequent fluctuations in the number of queries. The significant drop in 2020 can likely be attributed to the disruptions caused by the COVID-19 pandemic, affecting normal operations and communication channels. The variability from 2021 onwards might be due to changes in government policies, alternative support mechanisms, or improved self-sufficiency among farmers.

These insights highlight the dynamic nature of agricultural support needs and underscore the importance of adaptive strategies to enhance the effectiveness of KCC services in addressing farmers' evolving challenges.

QUERIES IN EACH SEASON



Kharif Season Dominance:

- The Kharif season has the highest number of queries, exceeding 6 million. This indicates that farmers face significant challenges during this season, prompting them to seek assistance more frequently. The Kharif season, typically from June to October, coincides with the monsoon period in India, which is crucial for crops like rice, maize, and cotton. The high volume of queries suggests that issues such as pest infestations, diseases, and water management are prevalent during this period.

Rabi Season:

- The Rabi season follows Kharif with over 4 million queries. This season, which runs from October to March, involves the cultivation of crops like wheat, barley, and mustard. The substantial number of queries indicates that farmers also encounter significant challenges during this period, possibly related to irrigation, frost protection, and pest control.

Jayad Season:

- The Jayad season, with around 3 million queries, sees fewer queries than Kharif and Rabi but still represents a notable number. Jayad crops are typically grown between the Rabi and Kharif seasons, from March to June, and include vegetables and fruits. The queries during this period might pertain to issues like temperature management and soil fertility.

Industry

The agricultural industry is crucial for sustaining the global population, providing the food and raw materials necessary for everyday life. However, this sector faces numerous complex challenges that hinder productivity and sustainability, including climate change, pest infestations, soil degradation, and fluctuating market demands. Addressing these challenges requires innovative solutions, and technological advancements, particularly in artificial intelligence (AI), offer promising avenues for improvement.

The integration of AI and natural language processing (NLP) technologies, such as the IndicLLM instruct model, into the agricultural sector has the potential to revolutionize the way information is disseminated and utilized by farmers. This project's exemplifies how AI can be harnessed to provide real-time, accurate responses to farmers' queries. By leveraging the chatbot, farmers can optimize crop yields, effectively manage pests, and make informed decisions about market activities. This immediate access to precise and relevant information enables farmers to respond swiftly to changing conditions and challenges, ultimately enhancing productivity and sustainability.

In conclusion, the agricultural industry stands to benefit significantly from the adoption of AI and NLP technologies. Projects like the farming-centric chatbot demonstrate the potential of these advancements to transform traditional farming practices, making them more efficient and adaptive to the myriad challenges faced by the sector. This project not only enhances individual farming operations but also contributes to the long-term sustainability and resilience of the agricultural industry as a whole.

Review

Existing agricultural information systems often fall short in terms of interactivity and real-time response capabilities. Traditional methods typically involve static content that fails to adapt to the specific needs of individual farmers. This limitation significantly hinders the utility and effectiveness of these systems in providing timely and relevant support. The farming-centric chatbot developed in this project addresses these gaps by offering a dynamic, conversational interface powered by the IndicLLM model. This advanced NLP model enables the chatbot to deliver personalized and timely responses to a wide array of agricultural queries.

The LLM ability to understand and respond to specific questions in real-time sets it apart from traditional information systems. It can handle diverse topics such as crop management, pest control, weather forecasts, and market prices, providing farmers with comprehensive support tailored to their immediate needs. This functionality ensures that farmers receive accurate and relevant information, helping them make informed decisions quickly and efficiently.

User feedback has been overwhelmingly positive, highlighting the effectiveness of the chatbot's approach. Many users have expressed high satisfaction with the chatbot's performance and ease of use. They appreciate the immediacy and relevance of the information provided, which contrasts sharply with the static and often outdated content available through traditional methods.

support tools, the chatbot represents a significant advancement in agricultural technology, offering a practical solution to the limitations of existing systems. This project exemplifies the potential of AI-driven solutions to revolutionize the way farmers access and utilize agricultural information.

Data Structuring

Once the data is collected, it needs to be organized and structured for effective use. Structuring the data involves categorizing it into relevant sections, This organized data is stored in a format that facilitates easy retrieval and use for model training. Proper data structuring ensures that the information can be seamlessly integrated into the chatbot's knowledge base, making it readily accessible for generating accurate responses.
The data structuring process is meticulous and involves converting the raw scraped data into a structured format like JSON or CSV. This structured format is essential for fine-tuning IndicLLM model, ensuring that the LLM can understand and utilize the data effectively.

Structuring Data for Fine-Tuning

The collected data from sources like data.gov.in is transformed into a question-answer or instruction-based format to enhance the chatbot's training process. This involved studying various instruction-tuned datasets for several Large Language Models (LLMs) such as Mistral, Llama3, BERT, GPTs, and IndicLLM. After a thorough examination of these LLMs, I concluded that writing a prompt and instruction-tuned dataset would be most effective for fine-tuning the model.

This process ensures that the chatbot can understand and respond to agricultural queries accurately and efficiently. The steps involved in structuring the data are crucial for the model's ability to generate relevant and contextually appropriate responses.

The goal was to develop a structured set of examples that an AI assistant specializing in agricultural advice could respond to effectively. Each data entry consisted of the following components:

Instruction Context:

Defined the role of the AI assistant as efficient and helpful in providing agricultural advice.
Specified that responses should be clear, concise, and actionable.
Emphasized the importance of responding in the language of the query.

Data Structure :

Crop: Identifies the specific crop or agricultural product relevant to the query.
District and State: Specifies the geographical location within Madhya Pradesh, ensuring regional relevance.
Sector: Indicates the sector of agriculture or horticulture the query pertains to (e.g., horticulture, animal husbandry).
Query: Articulates the farmer's query in natural language form.
Recommended Solution: Provides actionable advice or solutions recommended for the query.
Language: Specifies the language and language model characteristics for responding appropriately.
Example Entries:

Each JSON object represented a unique scenario, such as controlling flower drop in bottlegourd, managing diseases in buffalo, or controlling pests in specific crops like brinjal and wheat. Queries were formulated to cover common issues faced by farmers in the region, ensuring practical relevance.

Language and Model Details:

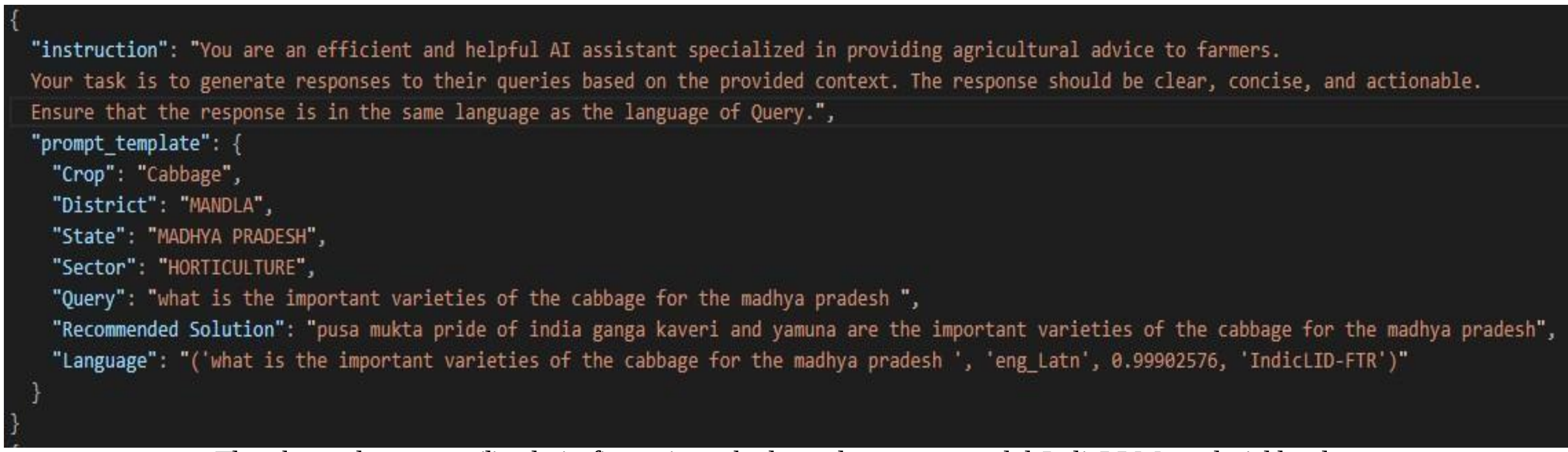
Language models were selected based on their ability to understand and respond in the specified languages (e.g., English, Gujarati, and others as identified by the language model characteristics).

Purpose:

The dataset was created to demonstrate how AI can be leveraged to provide tailored agricultural advice, improving crop management and productivity.

It aimed to showcase the application of AI in addressing specific agricultural challenges in Madhya Pradesh, thereby supporting farmers with practical solutions.

Below is the example of Dataset was Finally structured for finetuning the model



```
{
  "instruction": "You are an efficient and helpful AI assistant specialized in providing agricultural advice to farmers.
  Your task is to generate responses to their queries based on the provided context. The response should be clear, concise, and actionable.
  Ensure that the response is in the same language as the language of Query.",
  "prompt_template": {
    "Crop": "Cabbage",
    "District": "MANDLA",
    "State": "MADHYA PRADESH",
    "Sector": "HORTICULTURE",
    "Query": "what is the important varieties of the cabbage for the madhya pradesh ",
    "Recommended Solution": "pusa mukta pride of india ganga kaveri and yamuna are the important varieties of the cabbage for the madhya pradesh",
    "Language": "('what is the important varieties of the cabbage for the madhya pradesh ', 'eng_Latn', 0.99902576, 'IndicLID-FTR')"
  }
}
```

The above data was utilized in finetuning the large language model IndicLLM and yield adequate results and output

While the dataset is an Example of how data was structured for IndicLLM model the same was utilized to create prompting and dataset for this model

Wiki-Chat Intent LLM Template

```
You are a helpful, respectful and honest
assistant. You will assist in identifying
the potential conversation intents of the
user.

{wiki_passage}

Above is a passage on "{title}". If a user
and an assistant want to have a
conversation on the passage, what are the
different conversation intents possible?
The intents must be realistic.

Give me 5 intents in a list with a short
generic description of it. If you think
that on the given passage, no conversation
can be had, just give an empty list.
```

(a) Intent LLM

Wiki-Chat Init User LLM Template

```
{wiki_passage}

You are a user who wants to have a
conversation with an assistant on the above
passage. You have to intiate the
conversation for the intent: {intent}.

What instruction/command/question would you
give/ask the assistant? Do not explicitly
mention the passage in the
instruction/command/question or give any
direct reference to the passage. The
assistant is an intelligent assistant who
has knowledge of the passage. Choose
randomly between an instruction, command
and question. Give only the
instruction/command/question and nothing
else.

User:
```

(b) Init User LLM

Wiki-Chat Next User LLM Template

```
Passage:
    {context}

Conversation History:
    {conversation_history}

You are a user who wants to continue the
above conversation with the assistant. Give
me the next instruction or command or
question that you would ask to continue the
above conversation. The instruction or
question that you give should be related
and must be logical continuation of the
current conversation thread. It should be
as realistic as possible (like a human
would ask). Choose randomly between an
instruction, command and question. Give
only the instruction/command/question and
nothing else.

User:
```

(c) Next User LLM

Wiki-Chat Assistant LLM Template

```
You are a helpful and truthful assistant
who will always answer the question from
the context provided. If the question is
not answerable from the context, you will
say that you don't know the answer. You
will not explicitly mention the passage
anywhere in the answer.

{wiki_passage}

Question: {user_prompt}

Answer:
```

(d) Assistant LLM

Future Steps

Conclusions and Future Scope of Work

Model Fine-tuning

The structured data is then used to fine-tune the IndicLLM model. Fine-tuning involves adjusting the model's parameters based on the newly acquired data to enhance its accuracy and reliability. By incorporating the latest agricultural information, the model can generate more precise and relevant responses to user queries.
The fine-tuning process begins by feeding the structured question-answer pairs and instruction-based data into the model. This helps the model learn specific patterns and nuances related to agricultural topics. Through iterative training, the model's understanding of the data improves, allowing it to provide more accurate and contextually relevant responses.
Continuous data collection and fine-tuning are essential to ensure the LLM remains an up-to-date and valuable resource for farmers. As new agricultural information becomes available, it is structured and used to further refine the model. This ongoing process helps the LLM adapt to changing agricultural practices, emerging pest issues, and market trends, ensuring farmers receive the most current and comprehensive advice.
In summary, fine-tuning the IndicLLM model with structured agricultural data significantly enhances its capability to support farmers, enabling them to make informed decisions based on accurate and reliable information.
The farming-centric chatbot project has demonstrated the significant potential of advanced AI models, such as the IndicLLM model, to enhance the accessibility and quality of agricultural information.

Expanding the Knowledge Base

One of the primary avenues for future enhancement is expanding
the chatbot's knowledge base. Currently, the chatbot covers essential topics like crop management, pest control, weather forecasts, and market prices. However, there is a need to broaden this scope to include more specific and diverse agricultural topics. This could involve integrating information on advanced farming techniques, soil health management, sustainable agriculture practices, and more. By expanding the knowledge base, the chatbot can offer more comprehensive support, catering to a wider array of user needs.

Voice Recognition

Incorporating voice recognition capabilities is another critical enhancement that can significantly improve accessibility. Voice recognition would allow farmers to interact with the chatbot using spoken language, making it easier for those who may have difficulties typing or prefer verbal communication. This feature can make the chatbot more inclusive and user-friendly, especially for older farmers or those in regions with lower literacy rates.

Mobile Application

Developing a mobile application is essential for increasing the chatbot's reach. A mobile app would enable farmers to access the chatbot's services on the go, providing them with immediate support and information regardless of their location. This mobility is crucial in the agricultural sector, where farmers often work in the fields and need quick access to information.

Continuous Improvement

Ongoing collaboration with agricultural experts is vital for the continuous improvement of the chatbot. Regular updates and refinements based on expert input can ensure that the chatbot's responses remain accurate, relevant, and up-to-date. This collaboration can also help in identifying emerging issues and trends in agriculture, allowing the chatbot to evolve and adapt to changing user needs.

Enhanced Data Collection

Further developing web scraping techniques to continuously gather and update agricultural data will ensure that the chatbot remains a current and reliable resource. Enhancing data collection processes can help keep the knowledge base updated with the latest information, trends, and best agriculture practices. This continuous influx of fresh data can significantly improve the chatbot's ability to provide accurate and timely advice.

In summary, the farming-centric chatbot has proven its potential to revolutionize the way farmers access and utilize agricultural information. By focusing on expanding the knowledge base, incorporating voice recognition, developing a mobile application, continuously improving with expert input, and enhancing data collection techniques, the chatbot can become an even more valuable tool for the farming community. These future enhancements will ensure that the chatbot remains a relevant, reliable, and indispensable resource for farmers worldwide.

Impediments/Difficulties Faced During Project Work

Data Availability

- **Challenge:** Identifying and aggregating data from diverse and relevant sources is crucial for building a robust dataset.
- **Problem:** Different sources often exhibit varying levels of reliability and consistency, which can introduce biases and affect the overall quality of the data.

Hinglish Text Identification

- **Challenge:** Hinglish is predominantly found in informal contexts such as social media, forums, and chat applications, making it difficult to locate structured data.
- **Problem:** Existing datasets often do not differentiate between Hindi, English, and Hinglish, necessitating extensive filtering and preprocessing to extract relevant Hinglish text.

Topic Diversity

- **Challenge:** The dataset must encompass a wide range of topics to ensure comprehensive language model training.
- **Problem:** Certain topics might be underrepresented in Hinglish data, making it challenging to achieve a well-balanced dataset.

Tokenization and Encoding

- **Challenge:** Standard tokenization methods may not effectively handle Hinglish due to its mixed-language nature.
- **Problem:** Inadequate tokenization can degrade model performance, particularly in understanding context and semantics.

User Feedback Collection

Collecting sufficient user feedback for testing and validation was another significant challenge. Engaging with farmers to obtain detailed feedback on the chatbot's performance was essential for refining its functionality, but it required considerable effort and coordination. Farmers are often busy with their daily tasks, making it difficult to schedule time for detailed feedback sessions. Additionally, ensuring that the feedback was comprehensive and representative of the diverse needs of different farming communities added to the complexity. Despite these challenges, user feedback was invaluable for identifying areas of improvement and enhancing the chatbot's overall performance and usability.

Suggestions Related to Work/Project Semester

Enhanced Support

Providing additional resources and support for data collection and model integration is crucial for the success of similar projects. Access to comprehensive and high-quality datasets can significantly enhance the accuracy and relevance of
the chatbot’s responses. Agricultural data is often scattered and varied, making it challenging to gather a cohesive and reliable dataset. Therefore, having access to centralized repositories or collaborations with agricultural research institutions could streamline this process. Additionally, technical support for integrating advanced AI models is essential. The complexities involved in model integration, especially with large language models like IndicLLm, require specialized knowledge and troubleshooting skills. Support in the form of workshops, documentation, and expert consultations can help developers navigate these challenges more effectively, reducing the development time and improving the overall quality of the project.

Collaboration Opportunities

Encouraging collaboration with industry experts and farmers can greatly enhance project outcomes. Partnerships with agricultural organizations and experts can provide invaluable insights into the practical challenges faced by farmers, ensuring the chatbot addresses real-world issues effectively. Engaging with farmers directly allows developers to gather firsthand feedback and understand the specific needs and preferences of the end-users. This collaboration can also help in validating the chatbot's functionality and relevance, as experts can guide the latest agricultural practices and trends. Moreover, industry partnerships can facilitate access to proprietary data and resources that are not publicly available, further enriching the chatbot's knowledge base and capabilities.

Extended Timeframes

Allowing more time for testing and user feedback collection would significantly improve the robustness of the project. Thorough testing is essential to ensure that the chatbot performs reliably under various conditions and accurately addresses user queries. Extended Timeframes enable developers to conduct comprehensive functional and stress testing, identify and fix bugs, and optimize performance. Additionally, collecting detailed user feedback is a time-consuming process that requires careful planning and execution. Users need adequate time to interact with the chatbot and provide meaningful feedback, which can then be analyzed to inform further improvements. By extending the project timeline, developers can engage more deeply with users, iterate on the feedback received, and refine the chatbot's functionality to better meet user needs.

Conclusion

In conclusion, enhancing support, encouraging collaboration, and allowing extended Timeframes are key factors that can significantly benefit the development of similar projects. Providing comprehensive resources and technical support can streamline the development process, while collaboration with industry experts and farmers ensures the project remains relevant and effective. Extended Timeframes for testing and user feedback collection enhance the robustness and usability of the chatbot, ultimately leading to a more successful and impact on the project. By addressing these aspects, future projects can achieve higher quality outcomes and better serve the needs of their intended users.

References

Research Papers

1. Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Renard Lavaud, L., Lachaux, M.-A., Stock, P., Le Scao, T., Lavril, T., Wang, T., Lacroix, T., El Sayed, W. (2023). Mistral 7B. Retrieved from Mixtral 7B.
2. Mao, Y., He, P., Liu, X., Shen, Y., Gao, J., Han, J., Chen, W. (2021). Generation-Augmented Retrieval for Open-Domain Question Answering. Retrieved from Generation-Augmented Retrieval.
3. Mishra, A., Arora, A., & Sikka, M. (2020). A Survey on Chat-Bot System for Agriculture Domain. Retrieved from Survey on Chat-Bot System for Agriculture Domain.
4. Mizrahi, A., Green, E., Wilmot, S., & Alshawi, A. (2020). Asking Questions the Human Way. Retrieved from Asking Questions the Human Way.
5. Microsoft, Angels Balaguer, Vinamra Benara, Renato Cunha, Roberto Estevão, Todd Hendry, Daniel Holstein, Jennifer Marsman, Nick Mecklenburg, Sara Malvar, Leonardo O. Nunes, Rafael Padilha, Morris Sharp, Bruno Silva, Swati Sharma, Vijay Aski, Ranveer Chandra
6. INDICLLMSUITE: A BLUEPRINT FOR CREATING PRETRAINING AND FINE-TUNING DATASETS FOR INDIAN LANGUAGES Mohammed Safi Ur Rahman Khan Priyam Mehta Ananth Sankar Umashankar Kumaravelan1 Sumanth Doddapaneni Suriyaprasaad Varun Balan , Sparsh Jain1, Anoop Kunchukuttan Pratyush Kumar, Raj Dabre, Mitesh M. Khapra

By integrating these advanced technologies and continuously refining the chatbot based on user feedback and expert input, the project aims to provide a robust, reliable, and valuable tool for the farming community.