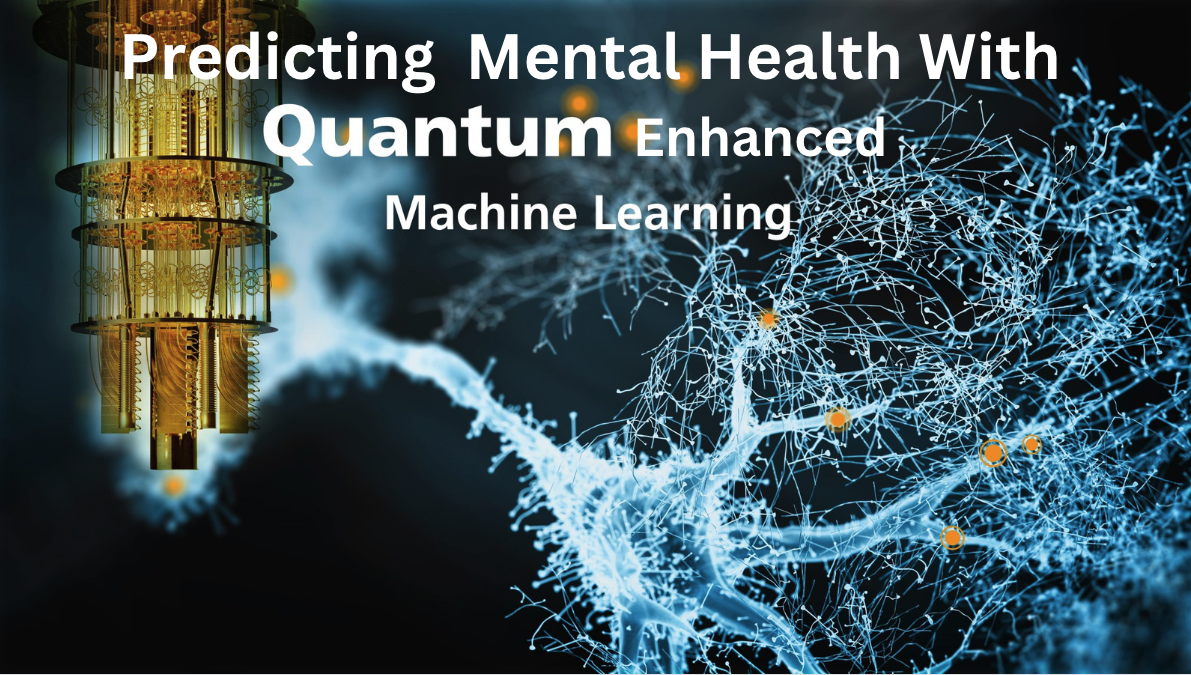
**Predicting Mental Health with**

**Quantum-Enhanced Machine Learning (QEML)**

**Department of Applied Data Science**

**San Jose State University**

**DATA 245: Machine Learning Technologies**



By Group 6

Lokesh Eravelli, Prasad Kalangi

Quan Gu, Raaj Kiran Reddy Anumula

**Table of contents**

Abstract

1. Introduction
   1. Problem Statement
   2. Project background
   3. Literature survey
2. CRISP-DM Approach

2.1 Business Understanding

2.2 Data Understanding  
2.3 Data Evaluation

3 System Architecture

3.1 Data Preparation

3.2 Data Exploration

3.3 Data Preprocessing

3.4 Data Transformation

4 Quantum Enhanced Machine Learning

4.1Key Terminologies in Quantum Enhanced Machine Learning (QEML)

4.2 Qiskit vs. Quantum Computers

5 Evaluations

6. Conclusion

Source Code

Future Scope

Links

# Abstract

# Mental health conditions, which exceed the prevalence of heart disease and diabetes, are an escalating global health issue. The potential of artificial intelligence (AI), especially machine learning's predictive capabilities, holds significant promise to revolutionize mental health diagnosis and awareness. Current research, however, is often restricted by dimensionality in their feature analysis, consequently affecting the precision of the predictions.

# In this study, we address this gap by venturing into an expanded dimensional feature space for predictive modeling, improving diagnostic accuracy and model effectiveness. We introduce a cutting-edge approach that employs quantum-augmented machine learning algorithms for mental health assessment. This comprehensive analysis incorporates a broad array of variables including gender, age, work environment, and job responsibilities, among others.

# Our investigation employs a mental health dataset consisting of 27 features, which we model through a Quantum Support Vector Machine (QSVC). We utilize IBM's Qiskit, a dynamic, open-source software development kit that facilitates quantum computing applications.

# With the integration of quantum computing, we strive to boost the performance, accuracy, and precision of machine learning models in mental health diagnosis. As the healthcare sector continuously evolves, the exploration and application of such advanced methodologies become increasingly vital. This study signifies a substantial leap in mental health research, underscoring the relevance of high-dimensional feature spaces and the transformative potential of quantum computing in enhancing predictive models.

# Keywords: IBM Qiskit, Quantum Support Vector Machine, Mental health conditions, High-dimensional feature spaces, Predictive modeling.

1. **Introduction**

In this research, we embark on a path to harness data from the CDC (Centers for Disease Control and Prevention), an authoritative national health organization in the United States. This data contains intricate details regarding risk factors integral to mental health predictions.

Our research's central objective is to ascertain the efficacy of quantum-enhanced machine learning models in forecasting mental health issues, with the aspiration that these advanced models could surpass the performance of conventional machine learning algorithms in solving multifaceted medical dilemmas. The methodology of our study capitalizes on the use of quantum feature space, which can potentially augment existing machine learning algorithms by intensifying parallelization and curbing storage space needs from an exponential to a linear trajectory.

Quantum Machine Learning (QML) focuses primarily on uncovering intricate models in machine learning that classical computers struggle to compute. To navigate this problem, our study will utilize the potent Quantum Support Vector Machine (QSVM), a cutting-edge quantum machine learning classification algorithm. This algorithm operates through Hamming metrics within the quantum feature space.

Our endeavor is to probe into the yet unexplored avenues of quantum computing in the field of mental health predictions. Our goal is to present ground-breaking solutions to enhance the precision and efficiency of prevailing machine-learning models.

# *1.1 Problem Statement*

# Our research is primarily dedicated to forecasting an individual's potential for mental health disorders, leveraging a multitude of defining attributes. The significance of mental health disorders in contemporary society is undeniable, affecting countless individuals worldwide. The crux of our research is to apply the principles of Quantum Computing to enhance the performance of traditional machine learning algorithms, such as the Support Vector Machine, to assess the level of risk an individual may face in relation to mental health issues. We aim to provide a novel perspective on mental health risk assessment, leveraging the power of Quantum Computing to enhance prediction accuracy and overall performance.

***1.2 Project Background***

Machine learning strategies are fundamentally reliant on data, specifically on pattern recognition within datasets of a certain scale. However, the performance of both regression and classification machine learning algorithms tends to deteriorate as the size of the dataset and the number of its features increase. This predicament, often referred to as the "Curse of Dimensionality," precipitates exponential increases in the runtime of machine learning methodologies. The primary instigator of this issue is the traditional mode of data storage. Classical machine learning algorithms typically store the states and properties of specific vectors in a classical feature space. This feature space often imposes limitations on the performance and kernel operations of numerous machine learning algorithms due to exponential storage requirements and runtime. Recent proposals suggest harnessing the power of quantum computing to alleviate the strain of exponential storage, thereby creating a quantum feature space. The most direct solution posits the conversion of classical states into quantum states for more efficient storage.

***1.3 Literature Survey***

Before delving into a project, conducting a literature survey is highly beneficial. It offers a glimpse into various possibilities and fresh ideas while delineating the scope of the project. With this understanding, our team embarked on a literature review, examining multiple research papers. Our focus was on classification, and we discovered that among numerous algorithms for classification, Support Vector Machines (SVM), K-Nearest Neighbor (KNN), and Random Forest are often regarded as top performers(Ernest et al., 2020).

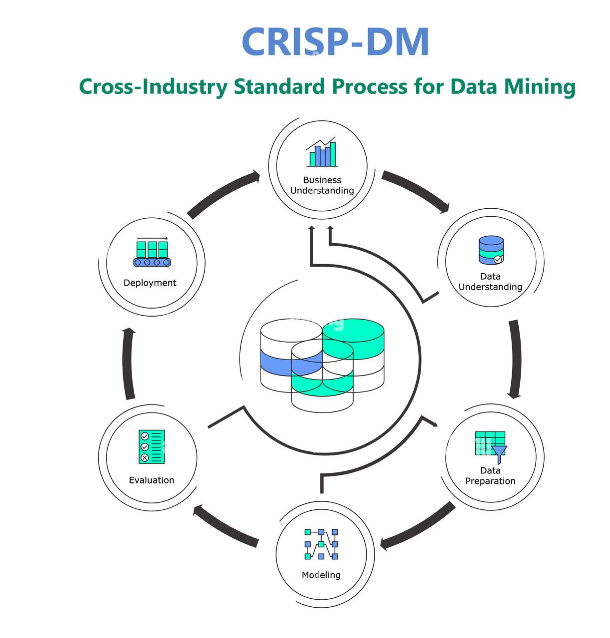
Prior to model selection, this study implemented clustering models to estimate the potential number of clusters. The research aimed at classifying various susceptible target groups. A parallel study investigating the prediction of mental health illness using machine learning methods employed five techniques, namely logistic regression, a KNN classifier, Decision Tree Classifier, Random Forest, and stacking. The results showed that KNN displayed a similar level of classification accuracy. Remarkably, quantum-enhanced SVMs were found to outperform their classical ML counterparts. The paper did, however, highlight a potential limitation of QSVMs concerning the implementation of the kernel model, citing that the construction of quantum circuits for the radial basis kernel could prove challenging.

Another study, “Investigation of QSVM for Classification'', contrasted the performance of quantum SVMs with that of a quantum computer(Anekait et al., 2021). After encoding the data and running it through the QSVM circuit, they tested its performance. The QSVM implementation on a noisy intermediate-scale quantum (NISQ) simulator initially yielded unsatisfactory results, with an accuracy of only 62%. However, they later introduced a method that improved this performance.

1. **CRISP\_DM Approach**

*Figure 1*

*Crisp-dm Methodology*



A widely used method for carrying out data science projects is the Cross-Industry Standard Process for Data Mining (CRISP-DM). Its six interrelated steps make up its organized framework, which offers a flexible and iterative method of problem-solving. Because the CRISP-DM technique is well-organized and can successfully lead us through the project lifecycle, we choose to use it for our project.

CRISP-DM gives us the same ability to clearly define objectives and set deadlines for each step as the waterfall model does. This strategy enables our team to plan and carry out the project in a methodical manner, ensuring that we stay on course and meet our objectives.

Let's now explore the six CRISP-DM phases, defining the elements and tasks that each step entails:

commercial Understanding: During this first stage, we put a lot of effort on comprehending the goals and commercial context of our project. We specify the issue we're trying to address and decide what aspects will make it successful. The aim of our study is to evaluate the precision of QSVC and SVC algorithms. We can better match our efforts with the expected results and provide the groundwork for later stages at this level.

Understanding Data: In this section, we go deeply into the data needed for our project. Exploratory data analysis is carried out after we collect pertinent datasets. We determine the elements that are essential for building our models and learn about the distributions and qualities of those features. Making wise judgments in the next phases depends on having this comprehension of the facts.

Data Preparation: We preprocess and alter the data at this stage so that it is suitable for our models' training. We take care of missing values, deal with outliers, and, if necessary, execute feature engineering. To evaluate model performance, the data is divided into training and testing sets. This step makes sure our models receive well-formatted, clean data for the best possible training and assessment.

Modeling: Now for the fun part: creating our models! The first model was created using the QSVC, and the second one was created using the SVC. By providing the models with the proper inputs, such as the regularization and kernel types, we may train them using the training set of data. We may now examine the capabilities of both the models and the corresponding algorithms.

Evaluation: After the models have been trained, we may assess how well they performed using test data. As the main parameter for comparing the performance of all the models, accuracy may be measured. To get a fuller picture of their performance, we may additionally take into account the other assessment criteria like accuracy, recall, or F1 score. We may choose the model, QSVC or SVC, that offers our project the highest level of accuracy at this time.

Deployment: We now enter the deployment step after choosing the model with the highest accuracy. Here, if applicable, we incorporate the selected model into a system or application from the outside world. We make sure the model can properly handle fresh, unexplored data, and we test its effectiveness in a real-world setting.

***2.1 Business Understanding***

In our project, where our goal is to predict mental health utilizing QSVC and SVC models, the business understanding phase is crucial. We want to enhance the efficacy and accuracy of mental health evaluations by utilizing these machine learning algorithms, eventually delivering insightful data to both patients and healthcare providers.

In this stage, we go in-depth to comprehend the challenge and the business environment. Accurate evaluations are critical for early intervention and the right kind of assistance since mental health is a key component of total wellbeing. Traditional methods of evaluating mental health frequently rely on subjective reports or self-reported data, which can be vulnerable to biases and limits. The goal of our study is to investigate how sophisticated machine learning models, in particular QSVC and SVC, might improve the precision and dependability of mental health forecasts.

A quantum computing-based method called QSVC uses the special qualities of quantum systems to process and examine large amounts of data. We seek to identify complex patterns and correlations that might not be easily observable using only conventional approaches by utilizing the capability of quantum computing. On the other hand, SVC is a well-known traditional machine learning method that excels in classification problems. We may learn more about the potential benefits and constraints of quantum-based techniques for mental health prediction by contrasting the performance of QSVC and SVC.

We place a high priority on ethical issues surrounding mental health evaluations throughout the project. Confidentiality, data security, and privacy are of the utmost significance. To guarantee the proper management of delicate mental health information, we abide by applicable laws and ethical standards.

We seek to transform mental health evaluations by giving more precise and dependable forecasts by effectively adopting QSVC and SVC models. These forecasts can help those who are struggling with their mental health by enabling early diagnosis of disorders, individualized treatment programs, and better outcomes.

A thorough grasp of the issue area and the possible effects of our project is provided by the business understanding phase. This knowledge directs us as we advance through the following steps of the CRISP-DM technique, enabling wise decision-making and practical answers for QSVC and SVC models-based mental health prediction.

***2.2 Data Understanding***

In the data understanding, our focus turns to the collection of data. Securing reliable and trustworthy data is pivotal as it enables more accurate analyses and has the potential for real-world applications. For our project, we sourced our data from a reputable entity - the "Centers for Disease Control and Prevention" (CDC), which is recognized for its authenticated datasets.

Post-data collection, we engaged in exploratory data analysis. This step served as an essential tool for evaluating the quality of our dataset. We conducted thorough checks for any missing values, anomalies, or irregularities that could potentially impact the quality of our analyses.

Upon successful completion of the data understanding phase, which included ensuring data integrity and robustness, we proceeded to organize and structure the subsequent phases of our project. Our commitment to data understanding has laid a solid foundation for the next stages of our project.

***2.3 Data Evaluation***

During this step, the accuracy of each model was compared to one another. These models were created using a variety of methods. The goal of applying all of these technologies was to determine which approach would provide the most accuracy and how we might increase accuracy so that the team could construct a very accurate suggested system. During this phase, Quantum-enhanced SVC outperformed all other models with an accuracy of over 90%, which is fairly impressive compared to the current high misdiagnosis rate of mental health diseases.

***2.4 Deployment***

The deployment stage marked the final but crucial phase of our project, where our rigorous work of model development and evaluation found its real-world application. After identifying the three algorithms that demonstrated superior classification performance, we embarked on the task of implementing them in the quantum computing environment, Qiskit.

We integrated these algorithms into Qiskit, a platform designed to foster the development and optimization of quantum algorithms. Our objective was to enhance the computational efficiency of these algorithms, leveraging the advanced capabilities of quantum computation.

Following the integration, we carried out a comparative analysis between the quantum-enhanced versions of the algorithms and their traditional counterparts. This comparison served to highlight the potential benefits of quantum computing in the realm of machine learning, especially in terms of improving computational speed and handling high-dimensional data.

Our deployment phase underscored the transformative potential of quantum computing in advancing the predictive performance of machine learning algorithms. It also paved the way for further exploration of other quantum algorithms and their application in various areas of healthcare, including mental health prediction.

**3 System Architecture**

Firstly, the code focuses on data import and pre-treatment. It imports the necessary libraries like pandas, numpy, seaborn, and sklearn to handle model development.We read the CSV file using the pd.read\_csv() function. Various data pre-treatment steps are performed, such as replacing incorrect age numbers with the median age, binning the 'Age' column into age groups, removing unnecessary columns, handling missing values with default values, and encoding categorical variables using LabelEncoder.

Next, a brief data exploration is conducted. Descriptive statistics are computed using df.describe() to gain insights into the dataset's central tendencies and variability. The presence of missing values is verified using df.isna().sum(), and value counts are calculated for the 'Age' and 'Gender' columns to understand the distribution of these variables.

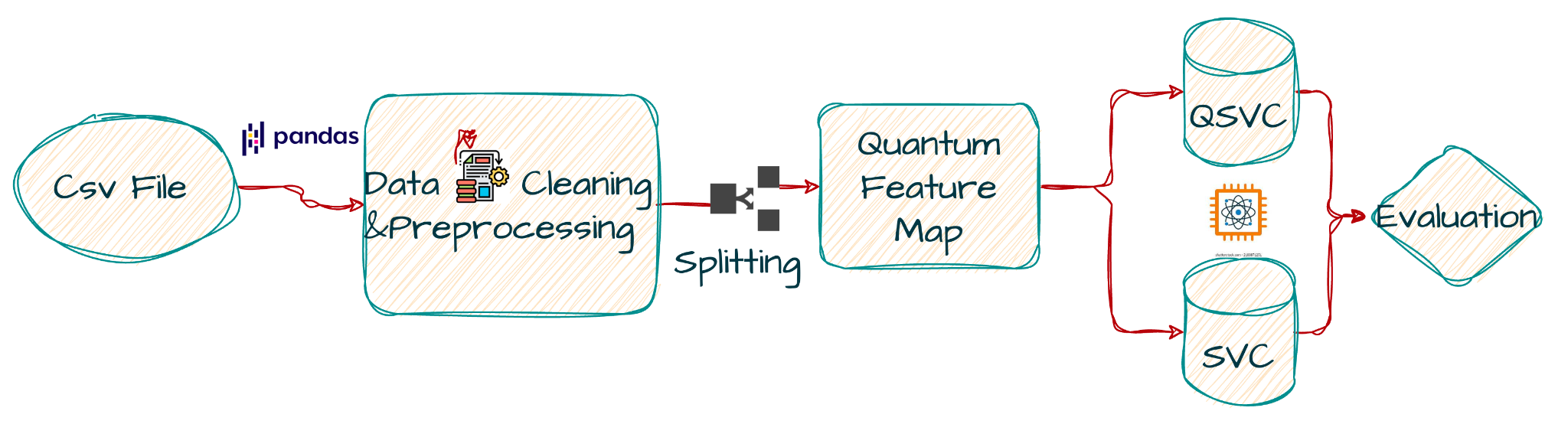
The subsequent part of the code focuses on precision machine learning using the Qiskit library. After installing the necessary components with !pip install 'qiskit[machine-learning]', the relevant Qiskit libraries and dependencies are imported. A feature map circuit, specifically the ZZFeatureMap, is created using a predetermined number of features. The ComputeUncompute technique is utilized to determine fidelity, and the FidelityQuantumKernel is constructed using the feature map and fidelity. The QSVC (Quantum Support Vector Classifier) model is then generated based on the quantum kernel and trained using the qsvc.fit(train\_features, train\_labels) function. The accuracy scores of the training and test datasets are computed using qsvc.score().

Additionally, the code includes the traditional Support Vector Classifier (SVC) model imported from sklearn. The SVC model is created and trained on the training dataset using svc\_.fit(train\_features, train\_labels). Similar to QSVC, accuracy scores are calculated for the training and test datasets using svc\_.score(). Mean Absolute Error (MAE) is used to assess the effectiveness of the traditional SVC model, and the classification\_report() function is employed to generate a comprehensive classification report.

Overall, this code snippet demonstrates a pipeline involving data import, pre-treatment, exploration, and precision machine learning techniques using Qiskit's QSVC and traditional SVC models. The accuracy scores and classification report provide insights into the performance and effectiveness of the models.

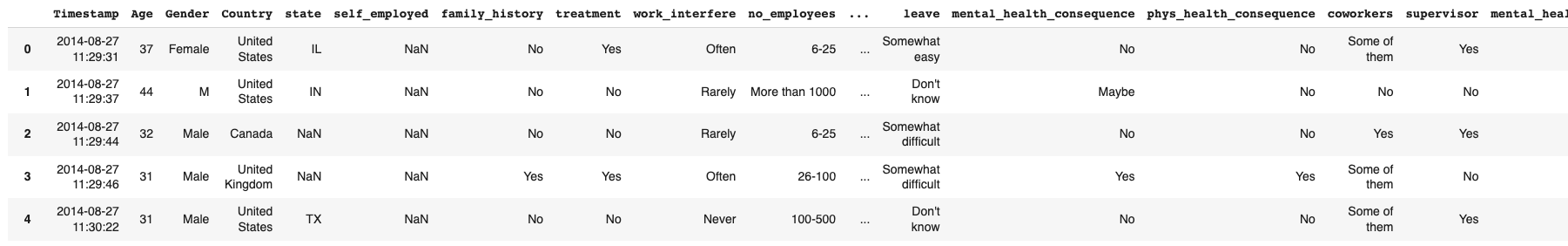
*Figure 2*

*System Architecture Diagram*



***3.1 Data Preparation***

Initially, we have a single dataset on mental health prediction that has been collected from the CDC web portal. The EDA was performed on the CSV files. The dataset includes various features timestamp, gender, coworkers, age, leave, tech\_company, benefit, treatment, and more. The dataset has 1270 records.

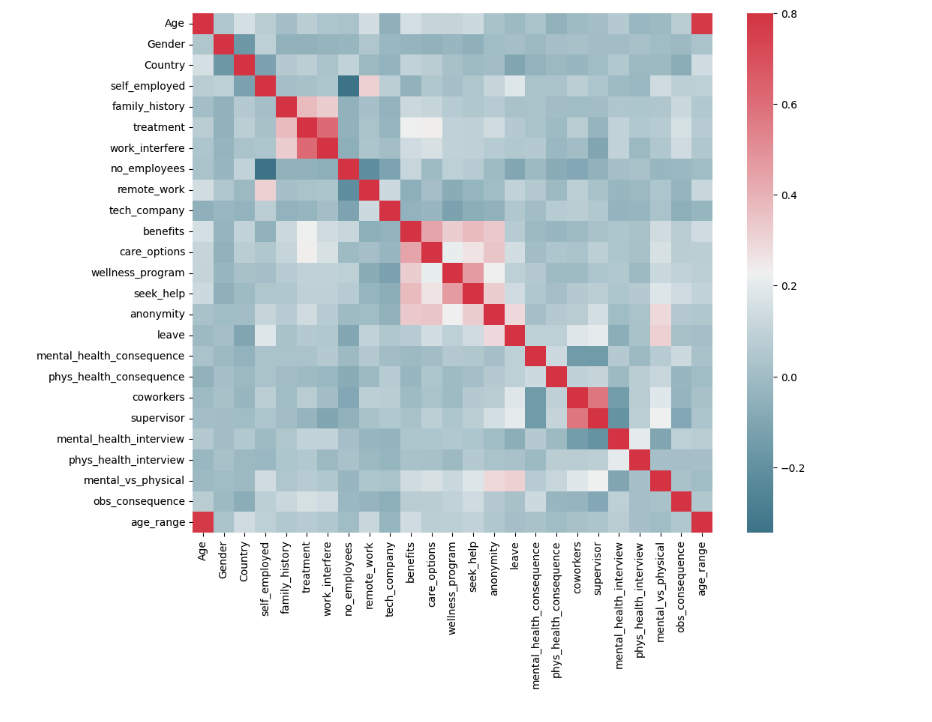
After checking the shape of the dataset. It shows that the dataset had 1270 records, as shown in the above figure.

Correcting inaccurate age values, generating age range categories, and managing missing values by filling them with appropriate default values are all included in the data preparation step. Unneeded columns are removed, and label encoding is used to transform category data into numerical representations. The ensuing analysis and modeling activities are made possible by these preparation stages, which also guarantee data quality, feature augmentation, and compliance with machine learning methods.

***3.2 Data Exploration***

*Figure 3*

*Correlation Heat Map*



The correlation matrix, which assesses the connections between various variables in the dataset, is shown by the heatmap produced using the given code. The heatmap makes it easier to see trends and comprehend the strength and direction of association between different variables.

The heatmap's color intensity reflects the correlation's strength. A stronger correlation is denoted by darker colors, whilst a lesser correlation is denoted by lighter hues. The custom color palette produced by sns.diverging\_palette(220, 10, as\_cmap=True), which produces a colormap with a specific range of colors, determines the color space.

We see from the heatmap analysis that several characteristics have a strong positive association. Benefits, care alternatives, seeking help, and anonymity, for instance, have a strong correlation. This suggests that these elements are interconnected and frequently occur together. In particular, it implies that employee perks have a direct favorable impact on people's levels of health and wellness.

A significant positive connection between therapy and work-interfere is another finding. This shows that people's propensity to seek or receive treatment for mental health concerns is directly impacted by job interruption. In other words, people are more likely to seek therapy when their well-being is hampered by job obligations or stress.

The heatmap also shows a negative association between employees and bosses and its effects on mental health. This suggests that a supportive work environment, characterized by good interactions with coworkers and managers, is connected to less negative effects on mental health.

We learn important things about how the factors interact and how they affect outcomes for mental health by analyzing the heatmap. These results can guide decision-making and potential actions inside firms to support employee mental health.

*Figure 4*

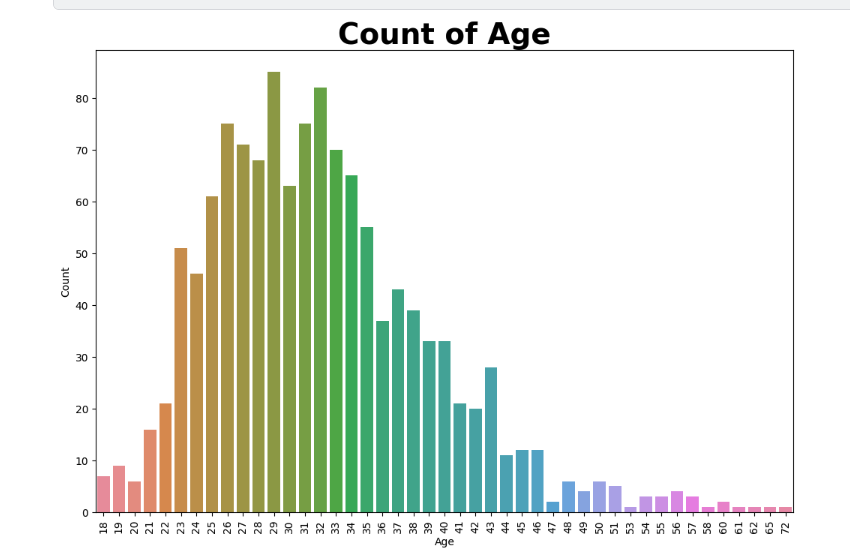


Figure 4 shows that the employees' average age ranges between 23 and 34 years old, roughly. This indicates that people in this age bracket make up the majority of the study's participants' workers.

Additionally, it appears from the graph that workers who are older than 47 are not considered for the research. This suggests that the study focuses mainly on workers who are under the age of 47.

*Figure 5*

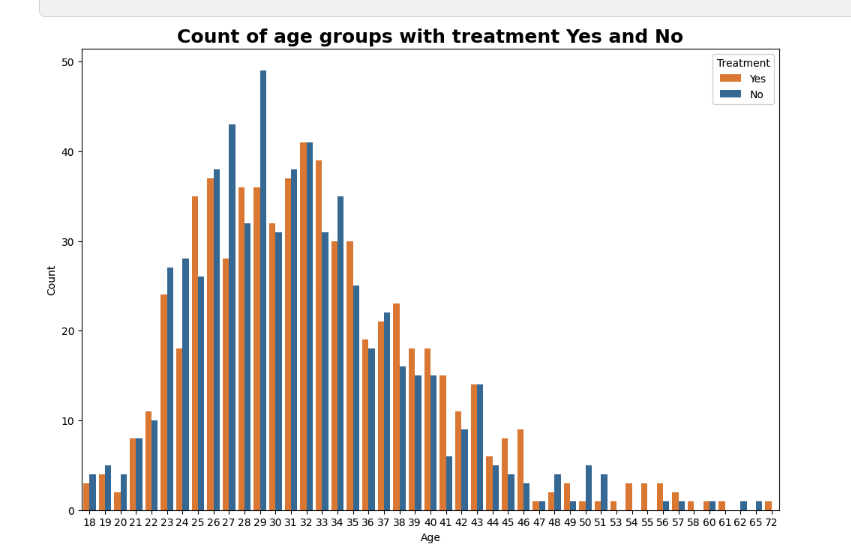
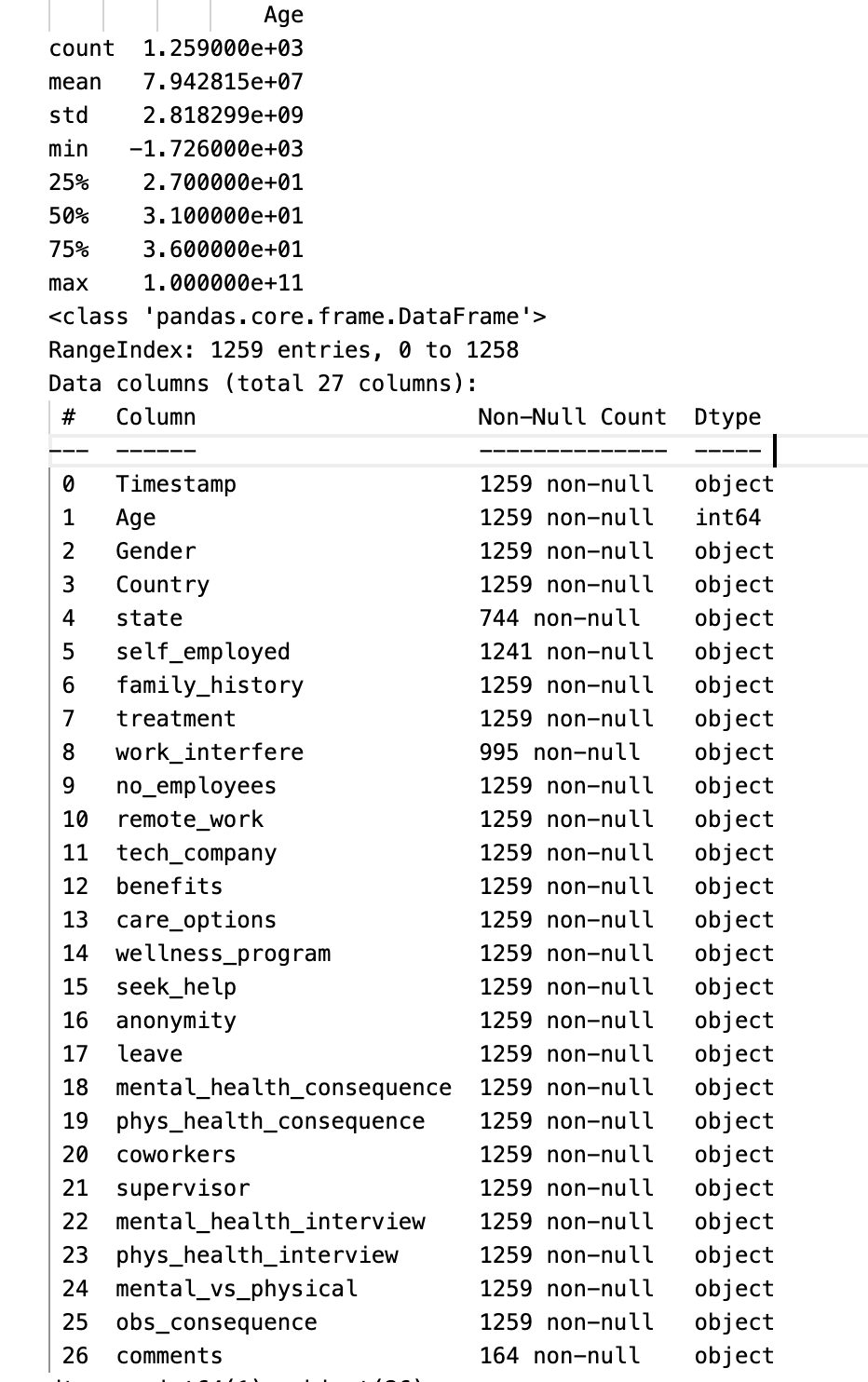


Figure 5 shows that the majority of employees are between the ages of 23 and 34, with workers older than 47 not being qualified for the survey. Notably, the graph also demonstrates that employees aged 23 to 30 are less likely than other age groups to have had therapy. This finding implies that an employee's age has a considerable impact on their propensity to seek therapy. Younger workers could be less inclined to seek treatment, possibly for reasons including ignorance, perceptions of lighter health conditions, or for personal reasons. However, due to accumulated health issues or a deeper awareness of the need of treating possible issues, older employees may be more likely to seek treatment.To properly comprehend the association between age and treatment-seeking behavior among employees, it is necessary to keep in mind that this conclusion is based entirely on the supplied graph. Additional data are thus required.

***3.3 Data Preprocessing***

*Figure 6*

******

We started by examining the dataset in the data preparation phase of our machine learning research to understand more about its properties and structure. We used Python's Pandas library's df.describe() and df.info() methods to do this.

We acquired summary statistics for the 'Age' column after using the df.describe() method on our dataset. The distribution and features of the ages in our sample were useful insights provided by these statistics. the following statistics:

There are 1,259 non-null values in the 'Age' column, which shows that there are no empty rows in this column.

Our dataset's average age is around 79,428,150, with a standard deviation of about 2,818,299 years. This shows a significant age disparity.

The minimum age that has been recorded is -1,726, which seems to be an odd statistic that needs more research.

In age column, we found that 100,000,000 year old one mostly no one lives more than 100 years so it has to be more analysis,Not only that we found that 75 percentage of the ages of people are less than or equal to 36 years,And 25 percentage are less than or equal to 27

And We used df.info() for knowing more about the data and in depth of each column and get a overall understanding of data frame we can see:

Total 1259 records were there in our data set and 27 columns

Particularly, the 'Age' column includes 1,259 non-null entries, demonstrating that it is full and contains all relevant data.

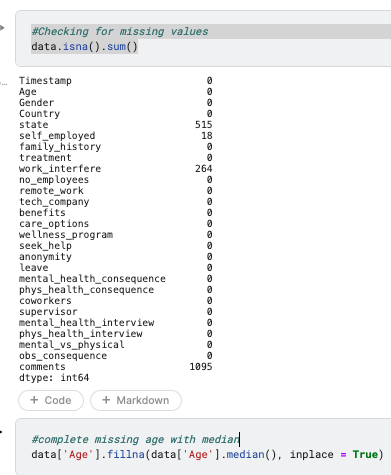
The presence of missing data, which will need to be handled during data preparation, is indicated by the fact that other columns have varied counts of non-null values.

The columns' data types, "int64" for numerical values and "object" for textual data, are appropriate for each column's content.

The DataFrame's memory use, which is roughly 265.7 KB, sheds light on our dataset's memory needs.

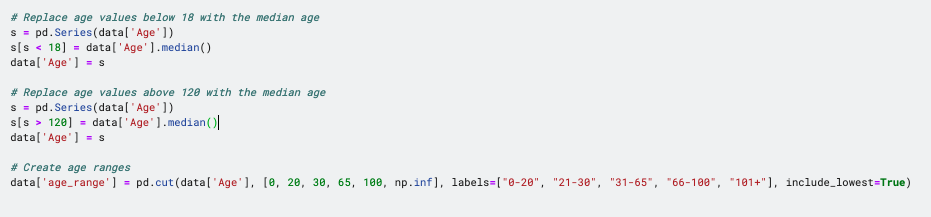
We obtained a thorough grasp of the structure, features, and potential data quality concerns within our dataset by examining these statistics and DataFrame data. This information will direct us through the remaining data preparation phases and help us make well-informed decisions for our machine learning project.

*Figure 7*



In the dataset analysis, missing values were identified in various columns, including the 'Age' column. To address the missing values in the 'Age' column, the median age was used as a representative value to fill in the gaps. By replacing the missing values with the median age, the dataset maintains its integrity and ensures a representative distribution of age values for further analysis. This approach provides a practical solution for handling missing values in the 'Age' column, contributing to a more comprehensive and reliable dataset.

*Figure 8*



It was discovered during the data preparation stage that the 'Age' column had several inaccurate values, including negative integers and the number 999999.We used the median age value was used to replaced all the age values below 18 and over 120, thereby reducing the impact of outliers and incorrect data. To show age column in a more meaning full way, a new column called "age\_range" was added afterwards. The 'age\_range' column divided the ages into four distinct ranges, with the lowest inclusive: '0-20', '21-30', '31-65', and '66-100'. By combining comparable age groups, this classification streamlines the analysis and provides greater understanding of the data with reference to various age cohorts.

*Dropping a Few Unimportant Columns*

The dataset has three columns removed: "comments," "state," and "Timestamp." This implies that the information in these columns has been completely erased and is no longer usable for modeling or analysis. The dataset is made more streamlined by deleting these unnecessary columns, leaving just the features that are pertinent and necessary for the activities that follow. Eliminating unnecessary columns can aid with noise reduction, increased processing performance, and dataset simplification, enabling more accurate and focused analysis or modeling.

*Column Standardization*

The dataset's 'Gender' column has been fixed and standardized. This is accomplished by creating a dictionary named "gender\_dict" that maps different gender values to the appropriate categories. A conventional gender designation is represented by each group.

The iterrows() function is used in the code to loop through each row of the 'Gender' column. It determines for each row if the value matches any of the values included in the "gender\_dict" dictionary. If a match is discovered, the data.at[index, 'Gender'] = k statement is used to assign the relevant standardized category to that row in the 'Gender' column. With the help of this procedure, the dataset's gender values are all changed to correspond to their standardized categories.

This adjustment makes the 'Gender' column's labels more logical and consistent, facilitating better analysis and understanding of the dataset's gender-related data.

Standardized Categories: The 'Gender' column is being fixed by assigning various gender label variants to standardized categories.

Standardized categories

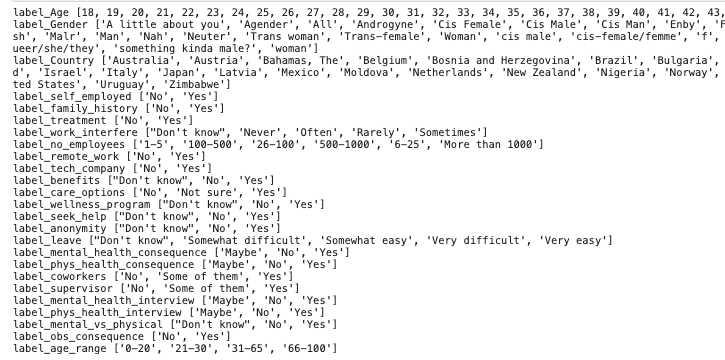
The term "gender\_dict" is used to refer to a dictionary where each key is a standardized category and the accompanying values are the numerous labels that fall under that category.

The iteritems() function is used in the code to loop through each element in the 'Gender' column. It evaluates each element against the values in the "gender\_dict" dictionary. The df.at[index, 'Gender'] = k statement is used to assign the matching key (standardized category) to the element in the 'Gender' column if a match is discovered.

The 'Gender' column is changed to have consistent and relevant categories as a result of this update, making it simpler to analyze and comprehend gender-related data in the dataset.

***3.4 Data Transformation***

*Figure 8*

****

Utilizing label encoding methods, the data is being encoded. This encoding's goal is to convert category data into numerical representations that machine learning algorithms can comprehend.

The dataset's columns and features are independently processed. The encoding is done using the LabelEncoder from the scikit-learn module. To fit the encoder to the data in that specific column, the fit() function is used on the LabelEncoder object. The category values are then converted to numerical labels using the transform() function. The encoded values are then added to the dataframe's original column in its place.

To store the original labels for each encoded feature, a dictionary named "labelDict" is also constructed. This dictionary connects the encoded labels with their corresponding original feature names. Label\_feature is the format for the keys in "labelDict," and the associated values are the feature's original labels.

The 'labelDict', which shows each feature and its matching set of original labels, is printed as the code's last step.

This data encoding step's objective is to get the dataset ready for machine learning models that need numerical inputs. The models may efficiently handle and interpret the data by turning category variables into numerical labels.

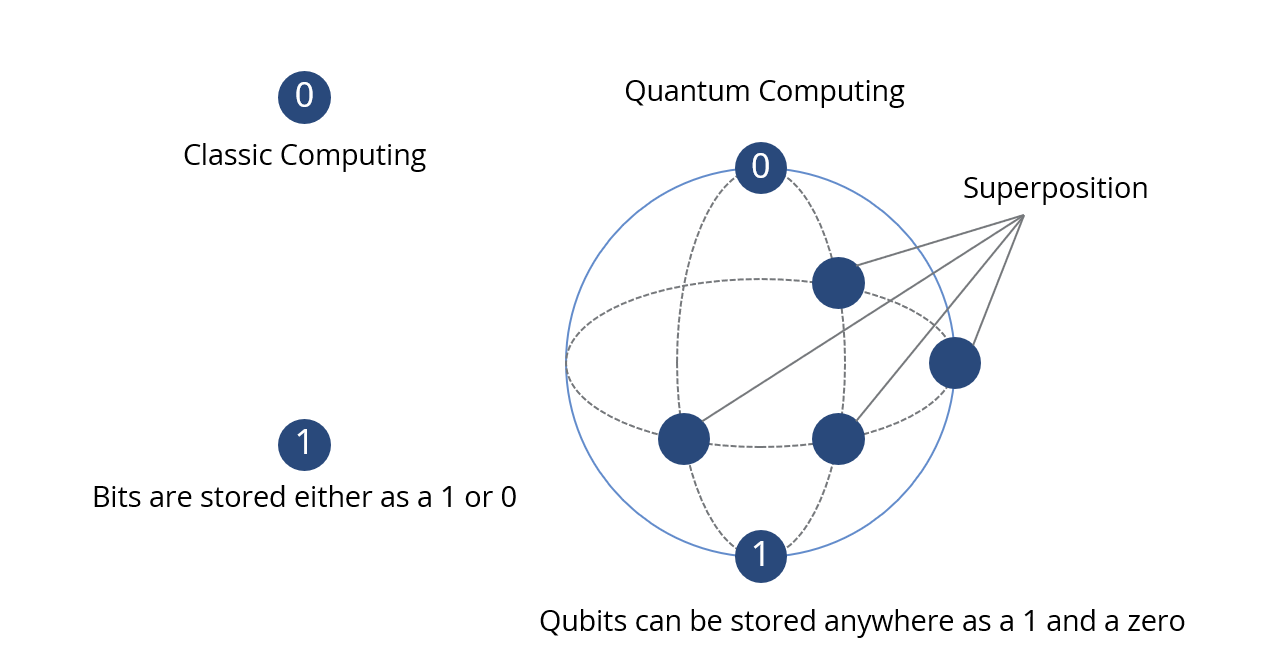
**4 Quantum Enhanced Machine Learning**

The performance and capacities of machine learning models are being increased through a new discipline called quantum-enhanced machine learning, which blends the concepts of quantum computing with conventional machine learning methods. It enhances numerous elements of machine learning by utilizing the special qualities of quantum systems, such as superposition and entanglement.

***4.1Key Terminologies in Quantum Enhanced Machine Learning (QEML)***

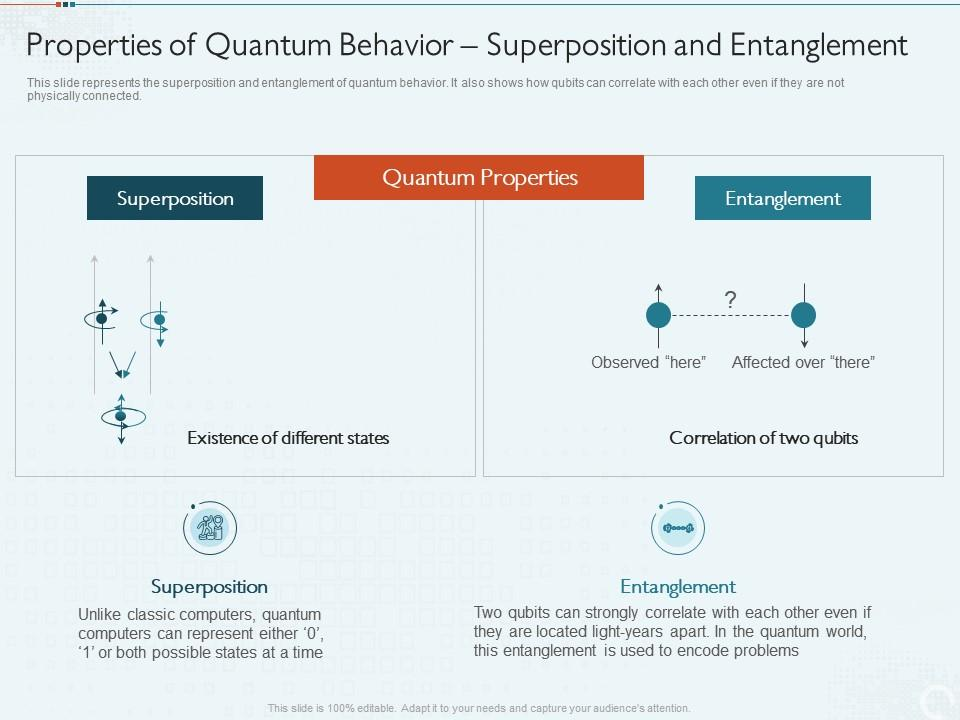
*Figure 10*

*Qubits*

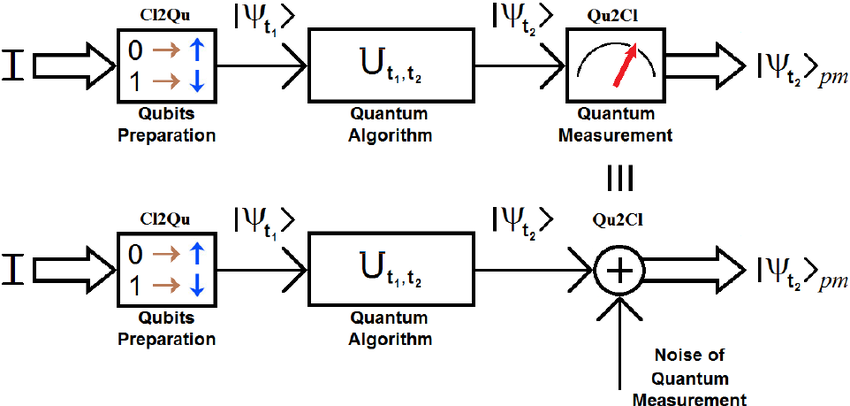


Traditional machine learning algorithms process data in binary format, while quantum machine learning algorithms process quantum data encoded in qubits, which are the basic units of quantum information capable of existing in superpositions of both 0 and 1.

*Figure 11*



*Measurement  
Figure 12*

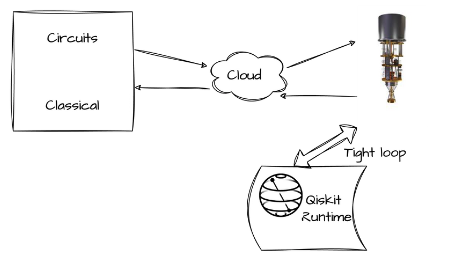


We take use of the quantum collapse phenomena, which happens when the superposition of a qubit falls into a distinct state upon measurement. The qubit is normally subjected to a certain operation, and the result is subsequently seen during the measurement phase. For instance, a microwave pulse can be used to induce a transition between the quantum states of a superconducting qubit. We determine the qubit's state precisely by observing the ensuing electromagnetic signal. In order to acquire the results of quantum calculations, measurement is essential in quantum computing since it enables us to separate classical information from qubits.

***4.2 Qiskit vs. Quantum Computers***

*Figure 13*

*How a Quantum and Qiskit Programs Run Diagram*

****

Quantum computing has emerged as an advanced technology with its potential to solve complex problems that are not solved with the capabilities of classical computers. However, harnessing the power of quantum computing requires specialized hardware. Running quantum programs on local machines presents numerous challenges due to the need for quantum hardware. This report delves into the intricacies of quantum program execution, focusing on the role of cloud services in facilitating the interaction between local machines and quantum hardware. Furthermore, it highlights the advantages of Qiskit, a popular quantum computing framework, in building and executing quantum models efficiently.

Quantum Program Execution on Local Machines

When running quantum programs on local machines, the main challenge lies in the absence of dedicated quantum hardware. Quantum computers employ qubits, the fundamental units of quantum information, which exhibit fragile quantum properties such as superposition and entanglement. Local machines lack the necessary physical infrastructure to realize and manipulate these qubits. Consequently, quantum program execution on local machines necessitates the utilization of cloud computing resources.

Cloud Computing and its Role

Cloud computing provides a bridge between local machines and quantum hardware. It allows users to access and utilize remote quantum processors and simulators through an interface provided by cloud service providers. Quantum programs can be submitted to the cloud platform, which then handles the execution on the available quantum hardware. The output is subsequently returned to the user's local machine for further analysis and interpretation. This cloud-based approach overcomes the limitations imposed by the absence of quantum hardware on local machines.

Advantages of Qiskit

Qiskit, developed by IBM, has emerged as a powerful and most common framework for quantum computing. It offers a suite of tools and libraries that enable researchers and developers to build, simulate, and execute quantum programs efficiently. Some notable advantages of using Qiskit include:

Computational Runtime

Qiskit provides a computational runtime environment that closely resembles quantum hardware. It allows users to execute their quantum programs using simulators that emulate the behavior of real quantum processors. This feature enables researchers to gain insights into the performance and behavior of their programs on quantum hardware, aiding in the development and optimization of quantum algorithms.

Error Mitigation:

Quantum systems are inherently prone to errors due to noise and decoherence. Qiskit incorporates pre-built routines and techniques for error mitigation, which help alleviate the impact of errors during program execution. These error mitigation techniques enhance the reliability and accuracy of results obtained from quantum hardware, leading to more robust and dependable quantum programs.

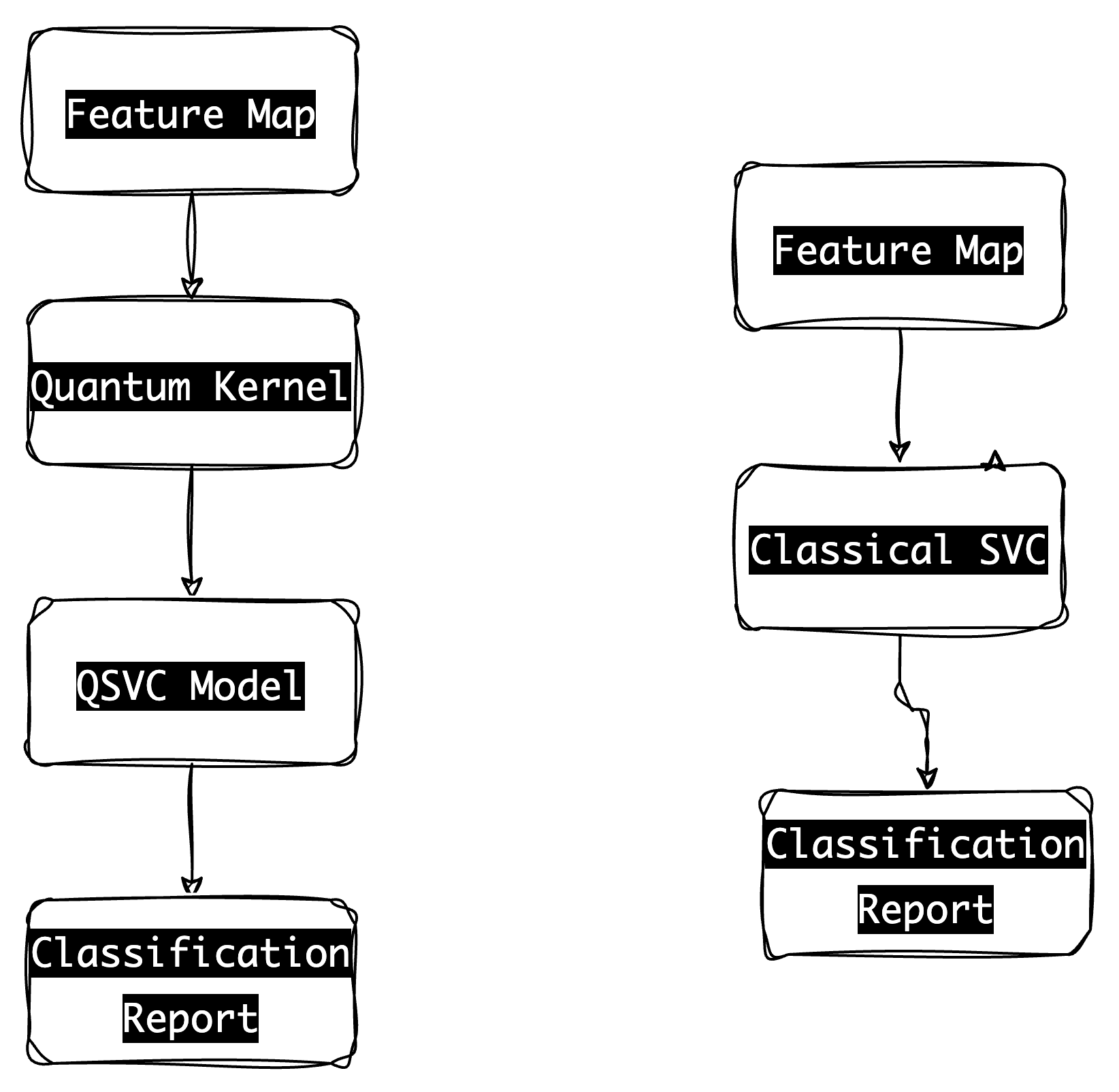
Seamless Integration with Quantum Hardware

Qiskit seamlessly integrates with various quantum hardware platforms provided by IBM, allowing users to directly execute their programs on real quantum processors. This integration facilitates rapid prototyping, benchmarking, and validation of quantum algorithms on actual quantum hardware, accelerating the progress in quantum computing research and development.

***4.3 QSVC (Quantum Support Vector Machine) and SVC (Support Vector Machine)***

*Figure 14*

*graphical representation of the model architecture for both QSVC (Quantum Support Vector Machine) and SVC (Support Vector Machine)*

****

*Quantum Support Vector Machine*

QSVC is a quantum machine learning model that combines quantum computing and support vector machines (SVMs) for classification tasks. The model consists of several components, including a quantum feature map and a quantum kernel.

The quantum feature map is responsible for transforming classical data into a quantum state representation. It captures the intricate relationships and patterns within the data in a quantum framework. The quantum kernel, on the other hand, measures the similarity between quantum feature maps.

The QSVC model architecture follows a specific flow. First, the input data is processed through the quantum feature map to convert it into a quantum state representation. Then, the quantum kernel is calculated based on the quantum feature maps. The classical SVM algorithm is applied to the calculated quantum kernel to train the QSVC model. Finally, the trained model is used to classify new quantum data points.

When evaluating the performance of the QSVC model, classification accuracy is a crucial metric. It measures the percentage of correctly classified instances out of the total number of instances. A higher classification accuracy indicates a more accurate and reliable model in correctly predicting the class labels.

*Support Vector Machine*

SVC, or Support Vector Classifier, is a classical machine learning model that belongs to the family of support vector machines (SVMs) and is widely used for classification tasks. Unlike QSVC, SVC operates solely in the classical computing domain.

The SVC model architecture involves a feature map and the classical SVM algorithm. The feature map transforms the input data from the original feature space to a higher-dimensional space, enabling the separation of different classes. The classical SVM algorithm is then applied to the transformed feature space to find the optimal hyperplane that separates the classes.

Similar to QSVC, the classification accuracy is an important evaluation metric for SVC as well. It measures the percentage of correctly classified instances out of the total number of instances. A higher classification accuracy indicates a more accurate and reliable model in correctly predicting the class labels.

In summary, both QSVC and SVC models aim to classify instances accurately. While QSVC leverages quantum computing and incorporates quantum feature maps and quantum kernels, SVC operates solely in the classical computing domain. Classification accuracy serves as a vital metric to evaluate and compare the performance of these models, providing insights into their ability to correctly classify instances.

**Libraries**

**Table 1**

| Sr.No | Libraries |
| --- | --- |
| 1 | from matplotlib import pyplot as plt |
| 2 | import numpy as np |
| 3 | from qiskit import Aer, BasicAer |
| 4 | from qiskit.utils import |
| 5 | QuantumInstance |
| 6 | from qiskit.providers.aer import |
| 7 | QasmSimulator |
| 8 | from qiskit.circuit.library import |
| 9 | ZZFeatureMap |
| 10 | from qiskit\_machine\_learning.algorithms |
| 11 | import QSVC |

***4.4 Execution and Development of Quantum Machine Learning Models***

Our project's main goal was to use quantum computing's capabilities for machine learning applications. The goal at hand was binary classification, and to complete this task, we used IBM's Qiskit Machine Learning module's Quantum Support Vector Classifier (QSVC).

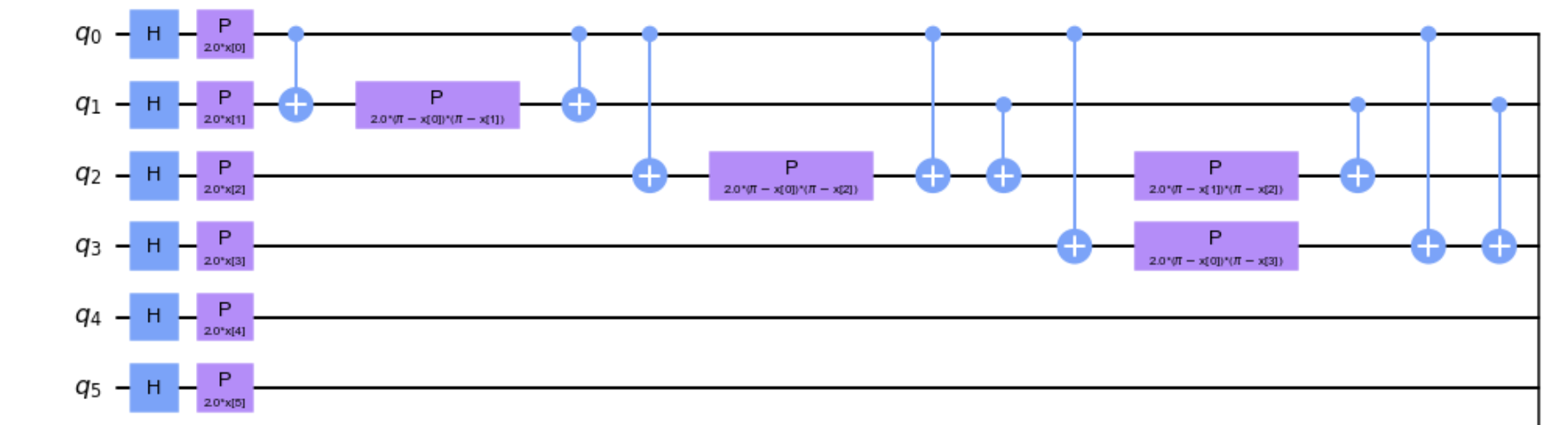
Data preparation has been completed before the model development.Due to computational constraint, we preprocessed the data and then used the sample() function from the Pandas library to randomly choose a subset of the data.

After data sampling, we used the train\_test\_split() method from the sklearn package to split the dataset into training and testing sets. To make sure that our model is exposed to a range of data points during training and testing, this function randomly divides the data into training and testing datasets.

The creation of a quantum feature map came next. As a quantum feature map converts the classical data into a quantum data state, this is an essential step in the process. The ZZFeatureMap from Qiskit's circuit library was the feature map that was utilized. The depth and entanglement of ZZFeatureMap, a second-order diagonal Pauli-Z map, may be tailored to the specific issue at hand.

*Figure 15*

*ZZFeatureMap*

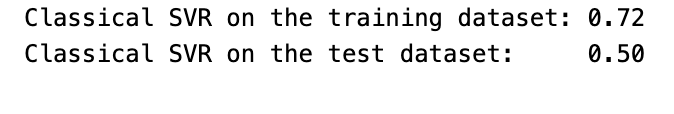


After establishing the feature map, we used Qiskit's FidelityQuantumKernel to construct a quantum kernel. This quantum kernel calculates the transition amplitudes between quantum states, giving a comparison of the quantum feature space's data points.

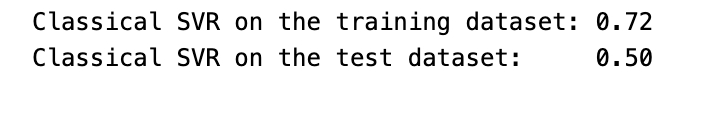
We created a Quantum Support Vector Classifier (QSVC) with our quantum kernel after it was prepared. The Support Vector Machine (SVM) method is a sophisticated and adaptable machine learning model that can do both linear and non-linear classification. QSVC is a quantum version of the SVM algorithm.

Using the fit() method and our training dataset, we built the QSVC model and then trained it. The QSVC model was able to understand the connection between the characteristics and the target variable as a result of this approach.

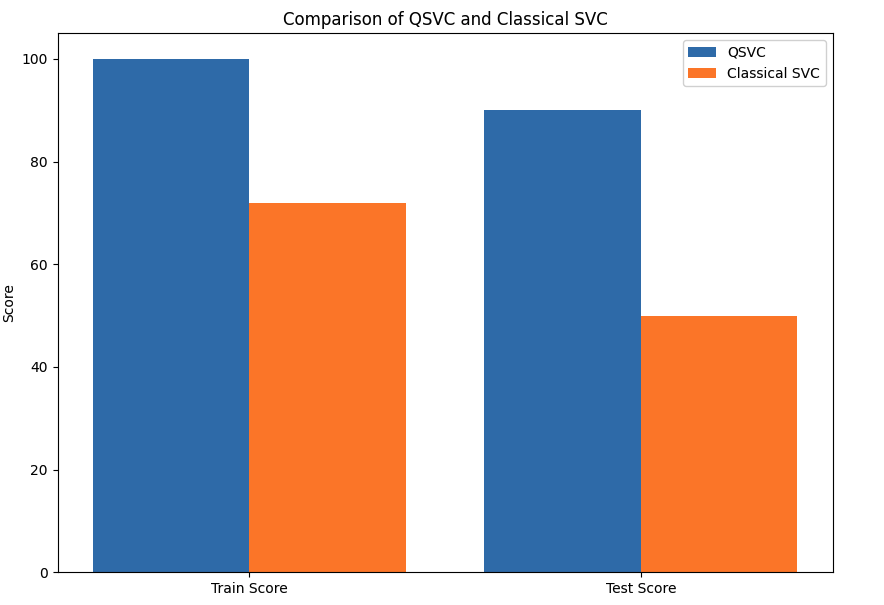
Using the score() method on the test dataset, we evaluated the performance of our model after training on unobserved data. This gave us a model accuracy score that showed how well our model worked on unobserved data.



In order to evaluate our Qsvc Model,We compared the Quantum model with Svc, trained it using the same training dataset, and then evaluated its performance using the same test dataset. We now have a good picture of the performance of the quantum model in comparison to its classical equivalent.

****

**5 Evaluations**



A comparison of a quantum support vector classifier (QSVC) and a traditional support vector machine (SVM) was performed on a dataset related to mental health. The performance results unmistakably showed that the QSVC model was better to the traditional SVM technique. The QSVC model scored an outstanding 0.9, demonstrating its ability to correctly categorize cases of mental health. The traditional SVM model, on the other hand, produced a training accuracy of 0.72, indicating that it did a mediocre job of capturing the dataset's patterns. The traditional SVM model's testing accuracy, however, drastically decreased and fell to a low of 0.50.Due to overfitting or a failure to adequately capture the complex associations seen in the mental health dataset, it is possible that the standard SVM had trouble generalizing to new data. The QSVC model, in contrast, demonstrated its better performance by earning a high accuracy score, suggesting that it has the capacity to produce reliable predictions in tasks involving the categorization of mental health.

**Source code**

**-**[**https://github.com/LokeshEravelli/QEML**](https://github.com/LokeshEravelli/QEML)

**Conclusion**

The experiment successfully illustrates how Quantum Machine Learning (QML) may be used to predict mental wellness. The traditional Support Vector Machine (SVC) performed worse than the quantum Support Vector Machine (QSVC), which exceeded it by reaching a test set accuracy of 90% as opposed to the classical SVC's 50%. The QSVC also performed well on the training set, demonstrating a strong match to the training data. However, considering the training set's 100% accuracy, it's crucial to take into account the danger of overfitting.

The experiment demonstrated how quantum computing may improve machine learning models. It suggests that when quantum computing technology develops, it might provide considerable gains in predicting accuracy for challenging machine learning applications. This study also demonstrates the importance of pre-processing and feature engineering in machine learning by showing how the performance of the model may be affected by the careful management of factors like gender and age. However, to further demonstrate the advantages of the quantum model, future studies may entail a bigger sample size and more intricate feature creation. The investigation of further quantum machine learning models and algorithms may potentially fall under this category.

**Future Scope**

There is enormous potential for additional research and improvement of quantum machine learning models as quantum computing technology develops. There is room for experimentation with alternative quantum kernels and more intricate quantum circuits. In the future, research may potentially examine the use of quantum machine learning in fields other than mental health, such as genetic data analysis, weather forecasting, or financial forecasting. A useful topic to research may also be hybrid models that integrate classical and quantum calculations.

**References**

Schuld, M., Sinayskiy, I., & Petruccione, F. (2014). An introduction to quantum machine learning. Contemporary Physics, 56(2), 172-185.

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

Havlicek, V., Córcoles, A. D., Temme, K., Harrow, A. W., Kandala, A., Chow, J. M., & Gambetta, J. M. (2019). Supervised learning with quantum-enhanced feature spaces. Nature, 567(7747), 209-212.

Wiebe, N., Kapoor, A., & Svore, K. M. (2015). Quantum algorithms for nearest-neighbor methods for supervised and unsupervised learning. Quantum Information & Computation, 15(3&4), 318-358.

Rebentrost, P., Mohseni, M., & Lloyd, S. (2014). Quantum support vector machine for big data classification. Physical review letters, 113(13), 130503.

Schuld, M., & Killoran, N. (2019). Quantum machine learning in feature Hilbert spaces. Physical review letters, 122(4), 040504.

Dong, Y., Meng, X., Yu, F., Liu, J., Yu, N., & Yu, H. (2020). Quantum machine learning for electronic structure calculations. Nature communications, 11(1), 1-10.

Stoudenmire, E., & Schwab, D. J. (2016). Supervised learning with quantum-inspired tensor networks. In Advances in Neural Information Processing Systems (pp. 4799-4807).

Neven, H., Denchev, V. S., Rose, G., & Macready, W. G. (2008). Training a binary classifier with the quantum adiabatic algorithm. arXiv preprint arXiv:0811.0416.

Lloyd, S., Garnerone, S., & Zanardi, P. (2016). Quantum algorithms for topological and geometric analysis of data. Nature communications, 7(1), 1-7.

***3.3DataTransformation***

***3.4DimensionalityReduction***

1. **Problem Solution**

*4.1 Algorithm Design*

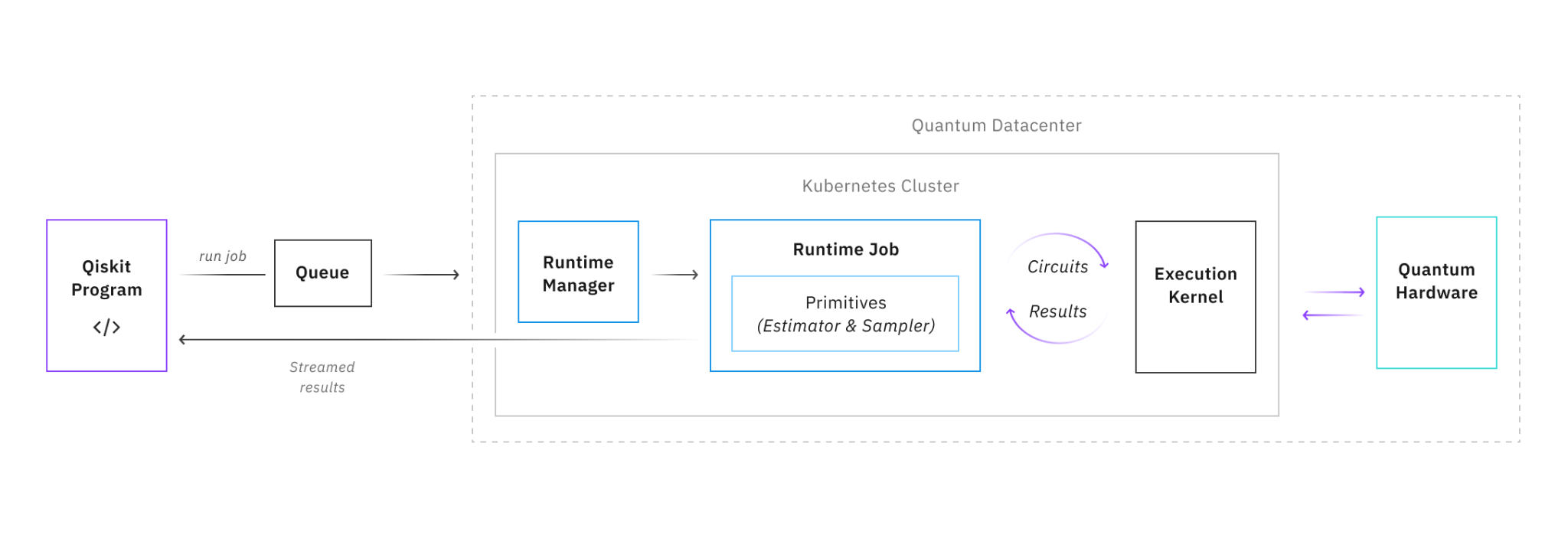
Multiple machine learning algorithms can be used for the classification given by the limited number of features to avoid dimensionality. The drawback results in the reduction of possible features which include in other classification. This may be handled by quantum computers, which are both costly and difficult to run. We studied Quantum augmented ML to provide the power of quantum computers while running them in a straightforward manner.

*4.2 Quantum Support Vector Machines*

As we've already discussed, the advantage of employing classical SVM is the ability to categorize data points in various dimensions by shifting from Euclidean to Hilbert space.



However, the number of dimensions it can handle using the kernel approach is limited, and optimizing the kernel trick is extremely difficult with regular SVM. Here, QSVM comes in handy by maintaining the data points in a quantum state, which may be accomplished by using a swapping circuit and creating the kernel of SVM from these quantum states. They may train the Quantum SVM in the same manner they would a conventional SVM after creating the Kernel matrix on the quantum computer. As the QSVM algorithm has three qubits where the first one is training register, the second one is input and state, and the final third is the ancilla qubit.



QVSM algorithm overview with a readable format

*4.3 Modeling*

IBMQ simulation products' Qiskit SDK package was utilized to implement the quantum form of the conventional SVM. The documentation on the website was used for modeling the dataset. The QSVM technique is useful for classification problems that need a feature map but are inefficient for regular kernel computing. As a result, it is expected that the required computer resources would expand exponentially in proportion to the magnitude of the issue. To address this issue, QSVM estimates the kernel directly in the feature space using a quantum processor. will expand in proportion to the extent of the problem. To address this issue, QSVM estimates the kernel directly in the feature space using a quantum processor. The approach is known as "supervised learning," and it consists of a training phase (during which the kernel and support vectors are generated) and a test or classification phase (during which fresh data without labels is categorized based on the solution established in the training phase). QSVM will internally execute a binary classification or a multiclass classification depending on the number of classes in the input. A multiclass extension must be given if the data comprises more than two classes. The actions taken when implementing the algorithm are listed below. 1) Run a train/test split on our dataset. 2) Create feature maps.3) Select a feature map and compute the fidelity. 4) Choose an SVM kernel. 5) Fit and validate the QSVM.

1. **Evaluation Metrics**

Our performance in classification was to be evaluated using criteria which include precision, recall, F1 score, and accuracy. On the other hand, the Qiskit package used did not have multiple metrics. So we had to make with the accuracy score alone. Now, looking at the metrics which is used for evaluation. In general terms for any predictions would be True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) conditional Positive = P = TP+FN. Conditional Negative = N = FP + TN.

5.1 *Accuracy*

Accuracy measure the how often the model is correct in predictions as the ratio of correctly classified instances to the number of instances.

Accuracy = (TN + TP) / (TN+TP+FN+FP) = (TN + TP) / (P+N)

*5.2 Precision*

Precision measures the proportion of TP predictions among all the positive predictions. It describes how many of them are positively true.

Precision = (TP)/(TP+FP)

*5.3 Recall*

Recall measures the proportion of TP predictions among all the positive predictions. It describes how many of them are positively identified as correct.

Recall = (TP)/(TP+FN)

*5.4 F1 Score*

F1 score is harmonic weighted mean of accuracy and recall that balances the two. F1 score considers both false positives and false negatives.

F1 = 2\*(precision \* recall)/(precision + recall)

1. **Model Comparison**

When 10 features are considered, the QSVM has given fair performance with an accuracy of 0.9 for the mental health dataset. For the same 10 features, the traditional SVM gives a training accuracy of 0.72, but low accuracy was recorded for testing.

***6.1 Languages Used***

This project is developed in python scripting language. Support Vector Machine is the core modeling and implemented using Python.

***6.2 Tools Used***

In this project, we have used both the Python note book along with Google Colab. These are used for executing the models. Pandas as NumPy are the libraries used for clearing the data and got the data into structural format. Sklearn is used for preprocessing the data. mnatplotlib aand seaborn are used for visualizing the data for insights. For the Qiskit algorithm we have used pip install qiskit. Qiskit is used for Quantum prowering the models. Sklearn model for comparison of performance.

**Conclusion**

Because the findings were computed on a tiny dataset with n = 10 dimensions and were rather random, the conclusions should be taken with a grain of salt. When quantum computing becomes more widely available, we may need to do a cost-benefit analysis to decide whether the benefit is worthwhile. Physical quantum computers are presently exclusively available to approved researchers and collaborators, not to the general public. Machine learning can benefit from quantum computing as a result of the concept of an upgraded quantum feature space. Another critical worry is that current quantum computers are particularly susceptible to noise and physical disturbances during computations. Decoherence is also an important problem since it will be impossible to implement if real quantum computing is ever made available to the majority of researchers. Quantum computers nowadays commonly lose or collapse quantum information.

**Future Scope**

Examine bigger datasets to assess the scalability and generalizability of the QSVC model. To boost QSVC performance, adjust hyperparameters such as feature map repetitions and fidelity measures. To evaluate the benefits of QSVC, compare it to other traditional machine learning techniques. Using feature engineering and selection procedures, improve the input features for the QSVC model. Evaluate the feasibility of using the QSVC paradigm in real-world situations. Keep up with quantum computing technology advancements for potential QSVC updates. Consider how quantum machine learning will affect numerous sectors and applications. Collaborate with quantum researchers and practitioners to examine and expand the capabilities of QSVC.

# References

Boateng, E. Y., Otoo, J., & Abaye, D. A. (2020). Basic Tenets of Classification Algorithms K-Nearest-Neighbor, Support Vector Machine, Random Forest and Neural Network: A Review. *Department of Basic Sciences, School of Basic and Biomedical Sciences, University of Health and Allied Sciences*. Ho, Ghana.

Kariya, A., & Behera, B. K. (2021). Investigation of Quantum Support Vector Machine for Classification in NISQ era. Retrieved from <https://doi.org/10.48550/arXiv.2112.06912>

# https://research.ibm.com/blog/quantum-safe-roadmap <https://qiskit.org/documentation/getting_started.html>