Quantum-Classical Neural Networks

Samarth Chopra, Arvind Sundar

Abstract

In this paper, we explore quantum-classical neural networks for tasks such as pattern recognition, regression and classification. Quantum computers have shown incredible promise in solving problems that have been considered computationally infeasible by their classical counterpart, such as the factoring of large numbers by Shor's algorithm. This paper seeks to find whether there exists a quantum advantage in the domain of machine learning (ML) and artificial intelligence (AI). We investigate an array of distinct methodologies for prediction, such as QNN classifiers and quantum support vector machines (QSVM). Finally, we benchmark these methods, for accuracy, to see which ones are particularly well-suited for certain tasks.

Objectives

The objective of the paper is to compare the accuracy of predictions of quantum computers versus classical computers. We will use small popular datasets, such as the Social Networks ads dataset for binary classification and QSVM. The reason why we are using small and ad-hoc datasets is for a reasonable training time and inference of the models.

Background

In a classical system, a neural network is a machine learning program that most similarly models the human brain or a neural pathway. It aims to make decisions as a human would and recursively adjusts itself to optimize the decisions it makes. A neural network consists of a node (artificial neurons) and a weight (synapses) arranged into layers that push data from one layer to another. Each node is connected to the subsequent layer through a weight that passes data when meeting a certain threshold.

Similarly, a quantum model of neural networks is also algorithmic in nature and is trained to find patterns and solve complex problems, however the key difference lies in the processing of data. Unlike a classical model, classical data needs to be loaded into the quantum system and then encoded into a quantum state using gates or some other method. From here a parameterized mode/circuit is used for training the weights, also known as an ansatz, alongside a classical optimizer to process the data for which a

measurement can be made. Both classical and quantum systems work on the same underlying logic, but the way in which they interpret data is what makes them unique.

Methods

In our comparison between quantum vs classical neural networks, the quantum neural networks will have the same basic underlying topology. The data will be loaded into the QNN and input into a feature map, to transform data into the feature space. After this, the data processing stage occurs, where the data is input into the neural network or ansatz. This ansatz circuit contains the trainable weights of the network, which will be trained through the backpropagation algorithm. After a single forward pass, we can then make our measurement and compare against the ground truth.

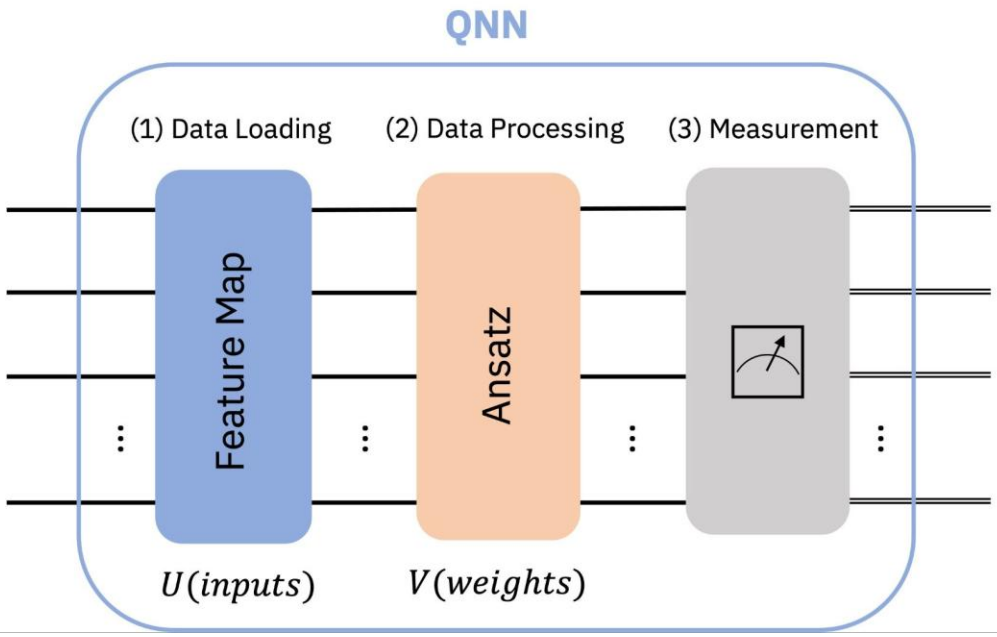

Figure 1: QNN Workflow

The way feature mapping is done is by encoding the features of the data in different ways. Hence, quantum gates are used to rotate qubit states along some axis to encode the feature, such as by rotating about the x-axis by the Rx gate. There is also entanglement between the qubit states, as the CNOT gate is applied. Some popular approaches to encoding include amplitude and angle encoding. Figure 2 shows an example of the ZZFeatureMap feature encoding two qubits in IBM Qiskit.

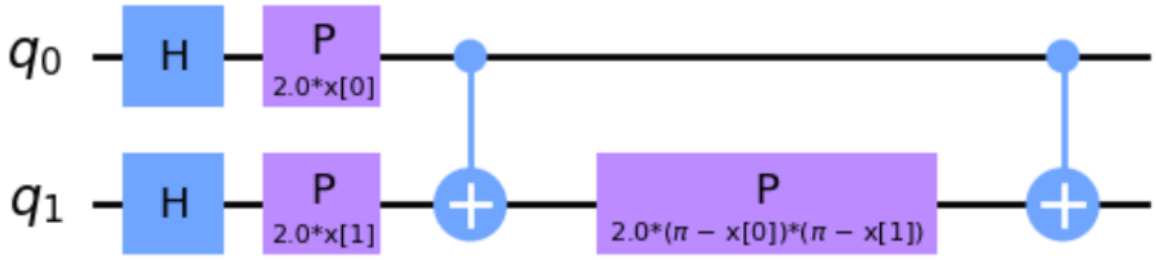

Figure 2: Feature Map for 2 Qubit System

The ansatz circuit is the actual neural network, and is the backbone of a QNN. The goal of a machine learning or artificial intelligence problem is to train the weights of this network such that it minimizes a cost function. The weights of the network can be encoded within the angle by which we rotate using a certain gate, such as the Ry gate. Figure 3 shows an example of a three-layer deep and two-qubit ansatz circuit, using the RealAmplitudes circuit.

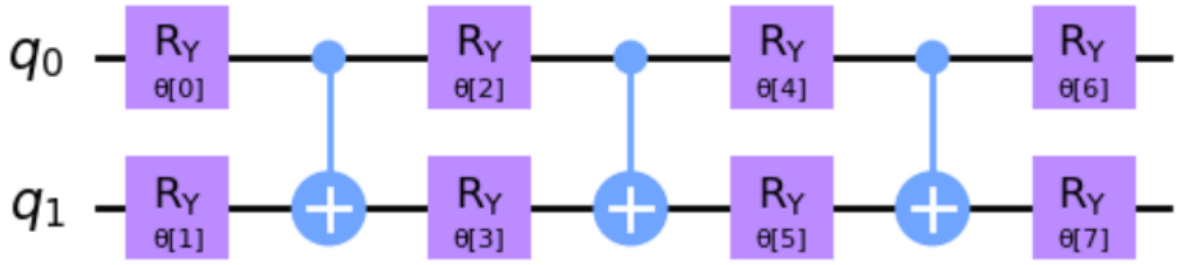

Figure 3: Ansatz for 2 Qubit System

For the classification task, we will compare the EstimatorQNN vs a classical logistic regression. Given a linearly separable dataset, we will compare the accuracies of the two methods. Similarly, we will compare the comparison between QSVM and a classical SVM using a dataset which is more difficult to separate linearly.

Results

We are using the Social Networks ads dataset, which can be visualized in a two-dimensional feature space in Figure 4.

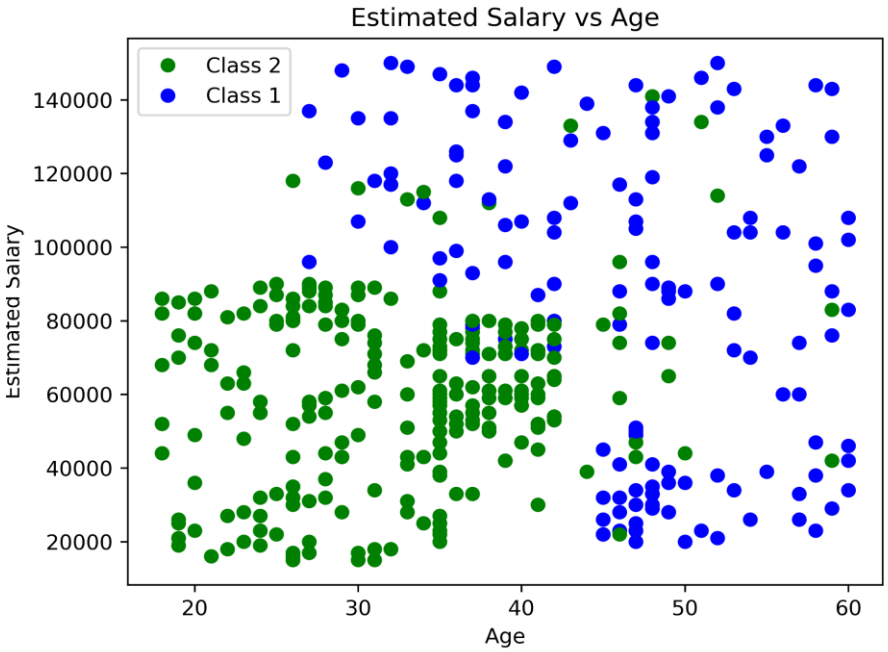

Figure 4: 2D Feature Space of Social Networks Ads Dataset

As we can see, while we can draw a linear decision boundary, it will not give a 100% accurate result in this binary classification task as the dataset is not completely linearly separable. Thus, it will be interesting to apply different techniques such as SVM and logistic regression.

I. EstimatorQNN vs Classical Logistic Regression

One particular use for a Neural Network is for the classification of data, specifically binary data. Binary data is data that is linearly separable into two distinct states and is used to show positive and negative results or opposing values. Within a quantum context, the EstimatorQNN only works with binary data so must return a one-dimensional output. For purposes of testing, the two features for the graph are age and estimated salary with output being 1 or 0.

From the results it is clear to see that the classical system drastically outperforms its quantum counterpart at classification. The EstimatorQNN showed an accuracy of 32% whereas the classical Logistic Regression algorithm had an accuracy of 89%. Specifically, it seems that the quantum system incorrectly identified every instance of class 0 from the test set.

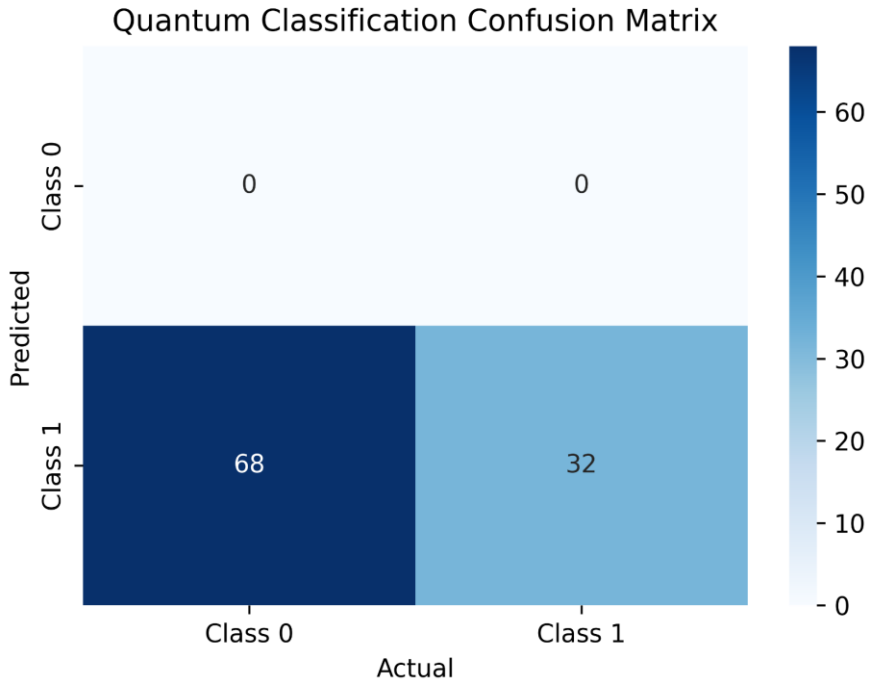

Figure 5: EstimatorQNN Confusion Matrix

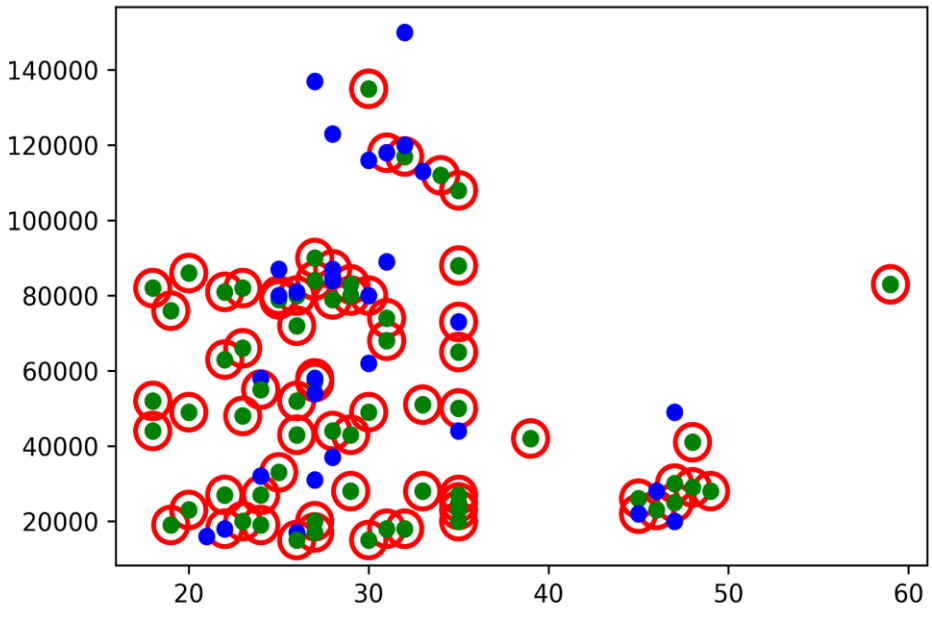

Figure 6: EstimatorQNN Test Set Predictions

Figure 6 represents the data points, and the misclassified data from the QNN is shown with red circles. Here we can see the data is not immediately linearly separable,

and as such it is more challenging for the QNN to create a linear decision boundary. As such, the QNN struggles in this classification task.

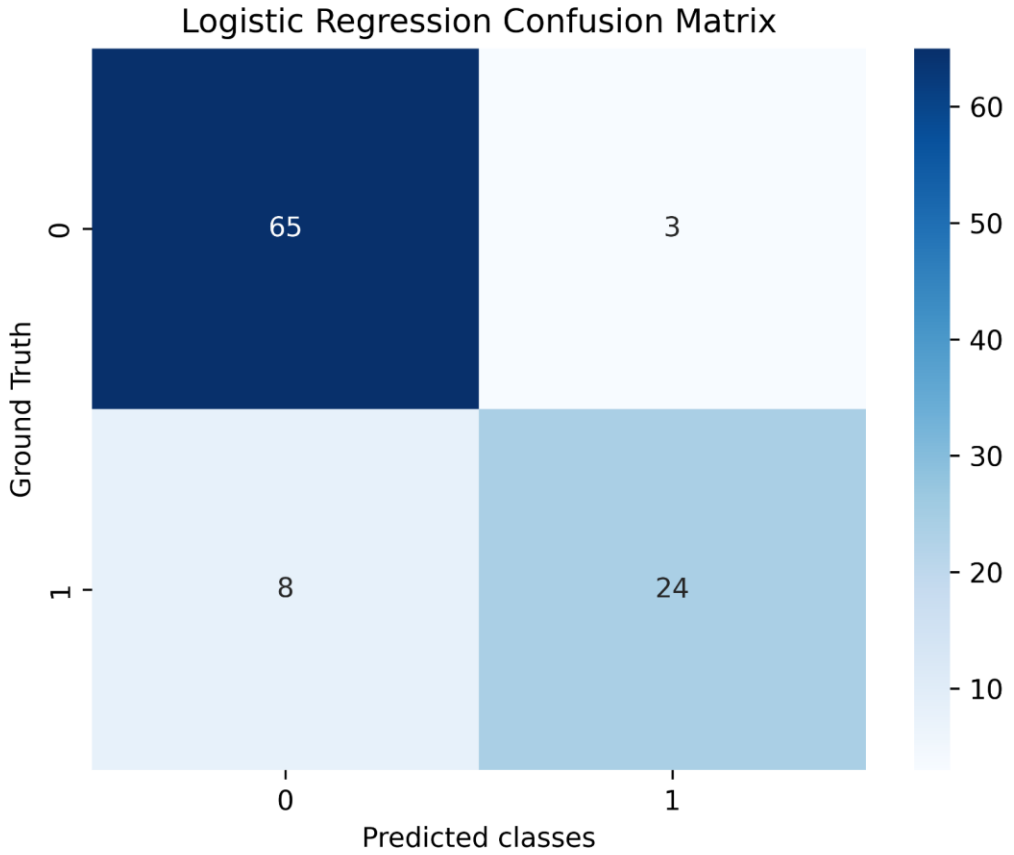

Figure 7: Logistic Regression Confusion Matrix

II. QSVM vs Classical SVM

The SVM is a popular machine learning technique used for tasks such as classification and regression. SVM classifiers are versatile because they can be used in tasks when the data is either linearly separable or not. In the latter case, the data is transformed into a higher dimensional feature space, which makes the data easier to separate. For the purposes of this paper, we will be comparing Linear SVMs in both the QNN and classical implementation.

The accuracy obtained using the QSVM in simulation was 61%. In comparison, the classical SVM yielded a 90% accuracy. As we can see, for this dataset, the classical SVM is the superior option. Also noticeably, the QSVM took a lot longer to train and classify the data points, in comparison to its classical analog. This makes sense, as the QSVM is actually a neural network with an ansatz that requires backpropagation to train

its weights. Figures 8 and 9 illustrate how the classical SVM separated the data in a linear fashion in the feature space as well as its confusion matrix.

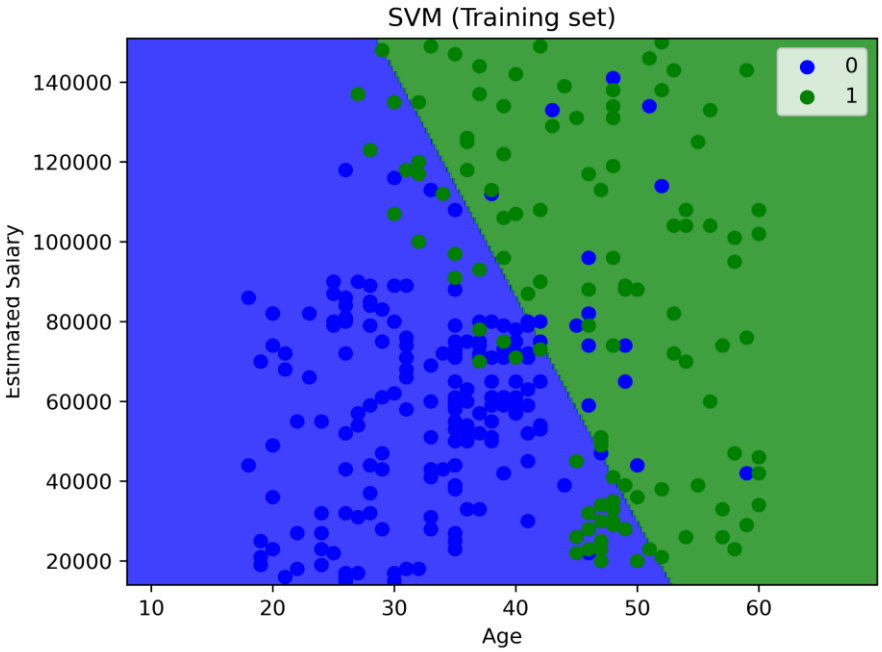

Figure 8: SVM Classification Results

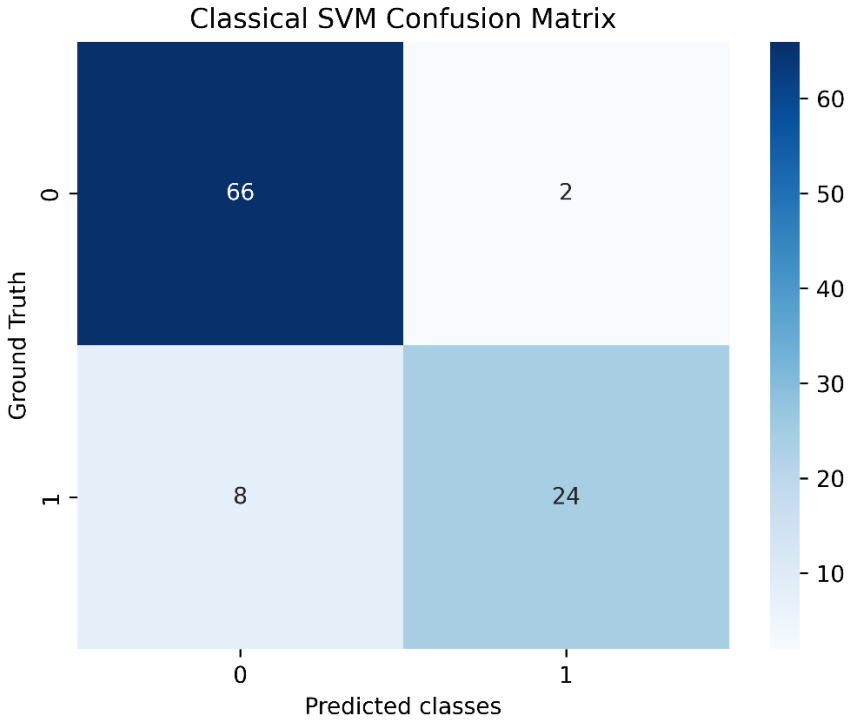

Figure 9: SVM Confusion Matrix

Discussion

Both the EstimatorQNN and QSVM were implemented using public tutorials where they were compared to their classical counterparts. The confusion matrices gave the performance metrics of the two systems, giving us information about the true positives, true negatives, false negatives and false positives. Both the quantum and classical systems used 100 data points for testing, and the rest 300 for training, resulting in a 75:25 train-test split. The output of the confusion matrix represents the overall accuracy in binary classification task, with the top-left and bottom-right quadrants representing accurate results and the inverse being inaccurately marked results.

Nonetheless, it is clear from both tests and the results depicted by the confusion matrices that the quantum systems have a long way to go before being able to compete with classical systems in the context of neural networks.

In order to improve the performance of the QNNs we can use the QSVM with a non-linear kernel function, as the dataset is not completely linearly separable. Using non-linear kernel functions can help project the data into a higher dimensional feature space, where we can find a decision surface separating the two classes with high accuracy.

Conclusion

In this project we have compared Quantum Neural Networks against their classical analogues. We have used the Social Networks dataset to investigate the EstimatorQNN and QSVM vs classical Logistic Regression and SVM. It has been enlightening to see how we can leverage QNNs and where their shortcomings are. Currently, QNNs struggle to compete with the classical neural networks, but there is hope that with a bigger ansatz and more qubits, QNNs can reach the same level of performance.

One area that excites us about QNNs is the built-in stochasticity of the state of the qubits, in that a qubit can be measured in both of the basis states with a certain probability. This is similar to an epsilon greedy algorithm, from Reinforcement Learning, where the policy picks the action with the highest probability majority of the time, but with a certain probability, it picks a randomly sampled action. The reason for this is to balance exploration vs exploitation, where the best policy is learnt by making some random choices and not getting stuck in suboptimal behavior (i.e. local maxima for an objective function). Maybe in the future, QNNs will be used for Reinforcement Learning and building autonomous systems.

References

Sim, Sukin, Peter D. Johnson, and Alán Aspuru-Guzik. "Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms." *Advanced Quantum Technologies* 2.12 (2019): 1900070.

Beer, Kerstin, et al. "Training deep quantum neural networks." *Nature communications* 11.1 (2020): 808.

Quskit. “Quantum Neural Networks - Qiskit Machine Learning 0.7.2.” Qiskit-Community.github.io, qiskit-community.github.io/qiskit-machine-learning/tutorials/01_neural_networks.html. Accessed 6 Apr. 2024.

Abbas, Ammira. “Lecture 5.1 - Building a Quantum Classifier.” Www.youtube.com, 5 Oct. 2021, www.youtube.com/watch?v=-sxlXNz7ZxU&list=PLOFEBzvs-VvqJwybFxkTiDzhf5E11p8BI&index=12. Accessed 6 Apr. 2024.

“How Is Data Encoded in a Quantum Neural Network?” *Quantum Computing Stack Exchange*, 1 Jan. 1966, quantumcomputing.stackexchange.com/questions/11370/how-is-data-encoded-in-a-quantum-neural-network.

Raushan, Rakesh. “Social Network Ads.” *Kaggle*, 6 Aug. 2017, www.kaggle.com/datasets/rakeshrau/social-network-ads/data.

“What Is Support Vector Machine?” *IBM*, www.ibm.com/topics/support-vector-machine. Accessed 12 Apr. 2024.

IBM. “What Are Neural Networks? | IBM.” www.ibm.com, IBM, 2023, www.ibm.com/topics/neural-networks.

GfG. “Epsilon-Greedy Algorithm in Reinforcement Learning.” *GeeksforGeeks*, GeeksforGeeks, 10 Jan. 2023, www.geeksforgeeks.org/epsilon-greedy-algorithm-in-reinforcement-learning/.