Journal of Scientific Agriculture 2023, 7: 63-76
doi: 10.25081/jsa.2023.v7.8556
https://updatepublishing.com/journal/index.php/jsa

Research Article

ISSN: 2184-0261

Integrating genetic markers and adiabatic quantum machine learning to improve disease resistance-based marker assisted plant selection

Enow Takang Achuo Albert*, Ngalle Hermine Bille, Bell Joseph Martin, Ngonkeu Mangaptche Eddy Leonard

Department of Plant Biology, Faculty of Science, University of Yaoundé I, P.O. Box 812, Yaoundé, Center Region, Cameroon

Received: June 16, 2023
Revised: August 31, 2023
Accepted: September 01, 2023
Published: September 11, 2023

***Corresponding Author:**
Enow Takang Achuo Albert
E-mail: albert.enow@facsciences-uy1.cm

ABSTRACT

The goal of this research was to create a more accurate and efficient method for selecting plants with disease resistance using a combination of genetic markers and advanced machine learning algorithms. A multi-disciplinary approach incorporating genomic data, machine learning algorithms and high-performance computing was employed. First, genetic markers highly associated with disease resistance were identified using next-generation sequencing data and statistical analysis. Then, an adiabatic quantum machine learning algorithm was developed to integrate these markers into a single predictor of disease susceptibility. The results demonstrate that the integrative use of genetic markers and adiabatic quantum machine learning significantly improved the accuracy and efficiency of disease resistance-based marker-assisted plant selection. By leveraging the power of adiabatic quantum computing and genetic markers, more effective and efficient strategies for disease resistance-based marker-assisted plant selection can be developed.

KEYWORDS: Plant disease resistance, Marker-assisted plant selection, Genetic markers, Adiabatic quantum computing

INTRODUCTION

Quantitative trait locus (QTL) mapping and single nucleotide polymorphisms (SNPs) have revolutionized the understanding of the genetic basis of complex traits in crops (Wang *et al.*, 2022). SNPs have emerged as powerful tools for identifying QTLs and predicting quantitative plant variables using SNP markers (Han *et al.*, 2022). Linear mixed models (LMMs) are commonly used for predicting quantitative plant variables, accounting for the polygenic nature of complex traits by modeling the relationship between the observed phenotype and both fixed and random effects (Zhou *et al.*, 2013). Machine learning algorithms, such as support vector machines (SVMs) and random forests (RFs), can also handle large datasets and complex interactions between SNPs and other factors affecting the trait of interest (Massi *et al.*, 2023). Despite challenges like correlation structure among SNP markers and potential bias in the selection of SNP markers, predicting quantitative plant variables using SNP markers holds great promise for improving crop yields, disease resistance, and stress tolerance through marker-assisted selection (MAS) (Difabachew *et al.*, 2023). As sequencing technology advances and more diverse germplasm becomes available, the accuracy and resolution of quantitative trait prediction are likely to improve, opening up new possibilities for precision breeding in agriculture (Bhat *et al.*, 2016).

Several approaches have been widely employed in plant breeding and genetics research to predict various traits such as grain yield, flower color, and disease resistance. They include but are not limited to Regression Analysis (Merrick *et al.*, 2022), Ridge Regression (Jeon *et al.*, 2023), LASSO Regression (Mathew *et al.*, 2022), Genomic Best Linear Unbiased Prediction (Clark & van der Werf, 2013), Bayesian Additive Regression Trees (Clarke & Kapelner, 2020), Markov Chain Monte Carlo (Krauth, 2021) and Integrated Multi-Omics Approach *(Mahmood et al.*, 2022). Each approach has its strengths and limitations, and the choice of method depends on the specific research question and dataset characteristics. By combining different methods, researchers can develop more accurate and robust predictive models for plant breeding and genetics research.

Quantum machine learning (QML) shows promise for solving complex optimization problems, but there's a research gap in applying QML regression algorithms for predicting quantitative plant variables using SNPs. Noise and high dimensionality in genomic data sets pose implementation challenges (Williamson *et al.*, 2023). Limited access to large-scale datasets hinders model training and validation, yet QML could improve crop yields and contribute to global food security, as demonstrated in experimental populations (Benos *et al.*, 2021). This study therefore has as primary objective to contribute to the growing literature on QML for SNP-assisted quantitative plant variable prediction by proposing an Adiabatic QML regression approach.

Plant phenotype prediction using QML regression approaches has gained popularity in recent years. Examples of these methods include, but are not limited to Quantum Support Vector Regression, Quantum Neural Network Regression, Quantum Random Forest Regression, Quantum Gradient Boosting Regressor, and Quantum K-Means Regression. These methods combine quantum computing with machine learning algorithms to predict continuous plant traits (Wu *et al.*, 2022). They have been successfully applied to various plant species, including wheat, rice, maize, soybeans, and cotton. Advantages of these methods include handling high-dimensional data, capturing nonlinear relationships, reducing overfitting, and improving interpretability. However, challenges such as requiring large amounts of labeled training data, computational costs, and choosing appropriate initialization methods exist.

Adiabatic Quantum Machine Learning Regression (AQMLR) is a novel approach to machine learning regression that leverages the power of quantum computing to improve efficiency and accuracy. Based on adiabatic quantum computation, AQMLR starts with an initial set of parameters and gradually adjusts them to minimize prediction errors (Ma *et al.*, 2023). Compared to classical machine learning algorithms, AQMLR can handle complex datasets with ease, capture non-linear relationships, and provide more accurate predictions. It is also robust against noise and errors, making it a promising solution for regression problems. While there are challenges associated with implementing AQMLR, such as the need for high-quality quantum hardware and software resources, its potential to revolutionize the field of machine learning makes it an exciting area of research (Date & Potok, 2021).

Tomato leaf blight disease, a pathology under focus in this study, has been found to have a complex genetic basis, involving multiple genes and quantitative trait loci (QTL) (Adhikari *et al.*, 2023). Researchers have identified several major gene families associated with resistance to Phytophthora infestans, including those involved in signal transduction, defense response, and cell wall biosynthesis (Arafa *et al.*, 2017). However, the exact mechanisms underlying this disease remain poorly understood. Tomato leaf blight disease poses a significant threat to commercial tomato production worldwide (Adhikari *et al.*, 2017). Infected plants may experience severe yield loss, reduced fruit quality, and increased susceptibility to other diseases (Wang *et al.*, 2021a). Moreover, the widespread use of resistant cultivars has led to the evolution of virulent strains of P. *infestans*, further complicating disease management strategies (Duan *et al.*, 2021). Therefore, continued research into the genetics and epidemiology of this disease is crucial for maintaining sustainable and productive tomato crops.

MATERIALS AND METHODS

Study Design

Figure 1 presents the design for this study. The most relevant features were selected from the main dataset using three feature selection algorithms – LASSO, Random Forest and Permutation – and grouped together to form the working dataset. Next, the working dataset was split into an algorithm training subset and a model testing subset. Further, the algorithm training subset was used to develop two models – Multi-Layer Perceptron Regression model and Adiabatic Quantum Machine Learning Regression model – which were later tested using the model testing subset, and on the basis of pre-defined evaluation criteria. The better model was then determined.

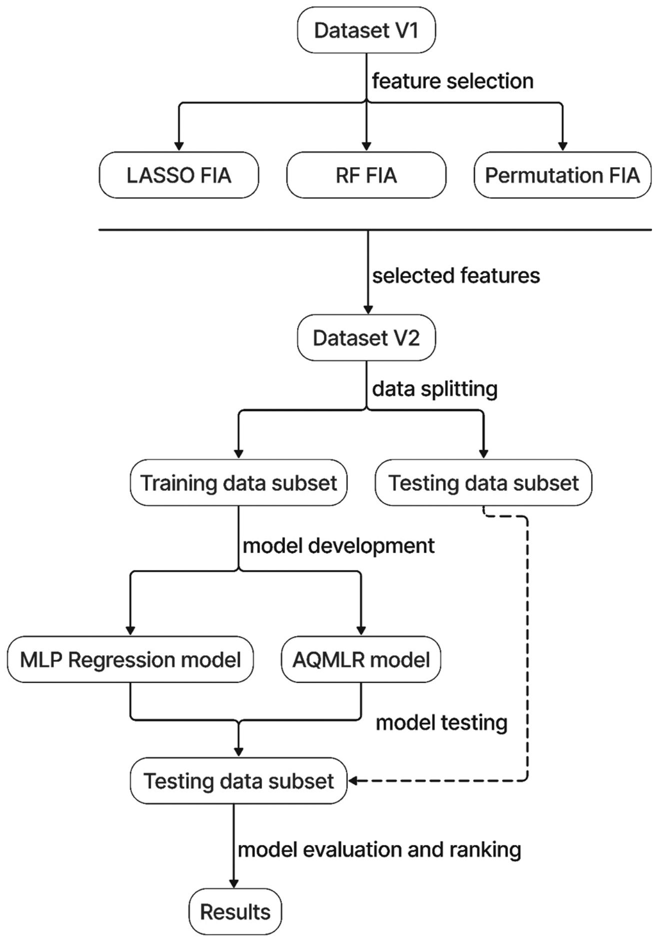

Figure 1: Study design. AQMLR - Adiabatic quantum machine learning regression, FIA - Feature importance algorithm, LASSO - Least absolute shrinkage and selection operator, MLP - Multi-layer perceptron, RF - Random forests

System Specification

The scripts used for this research were written in Python 3 with system specifications as follows: 64-bit operating system, x64-based processor, 8GB RAM, intel CORE i7 processor.

Dataset

The initial dataset consisted of 43,342 SNP markers for tomato leaf blight disease resistance genes which served as predictor features, and a leaf blight severity score feature which served as the label. Values in the label ranged between 0 and 1, with 0 meaning absolute resistance and 1 meaning absolute susceptibility. There are several genes that have been identified as playing a role in managing tomato leaf blight. These include the gene for Resistance to Tomato Spotted Wilt Virus (TSWV), which has been linked to resistance against TSWV-caused tomato leaf blight (Czosnek *et al.*, 2013); the gene for Resistance to Bacterial Speck (Rbs), which confers resistance against *Xanthomonas spp*., a bacterium that can cause tomato leaf blight (Bashir *et al.*, 2022); and genes related to plant defense mechanisms such as RIN4, RIPK2, and BAK1 (Pandey *et al.*, 2022).

After feature selection, the 30 most important markers were used to prepare the working dataset. The working dataset had 4,000 instances corresponding to 4,000 genotyped plants. These instances were divided into an algorithm training dataset and a model testing dataset. The former consisted of 2,800 instances of the working dataset, while the latter consisted of 1,200 instances of the working dataset.

SNP Encoding

A value of -1 represented an allelic loss (one copy of the reference allele being lost). A value of 0 represented the reference allele (no variation present). A value of 1 represented one copy of the alternative allele (heterozygous genotype). A value of 2 represented two copies of the alternative allele (homozygous genotype).

The above coding system is widely used in genetic research and analysis software, such as PLINK, GATK, and Samtools, among others. It's important to note that different software packages may use slightly different conventions for encoding SNP data, but the representations used in this study apply across most platforms.

Feature Selection

Feature selection is a critical step in the machine learning process that helps identify important variables that contribute significantly to the prediction task. Feature selection plays a vital role in ensuring the success of machine learning models, including, but not limited to the following:

Improves Model Performance: By selecting only the most relevant features, the model can perform better and achieve higher accuracy (Pabuccu & Barbu, 2023). Reduces Complexity: Feature selection simplifies the model by eliminating unnecessary features, which reduces computational complexity and improves interpretability (Sisiaridis & Markowitch, 2017). Minimizes Overfitting: When irrelevant features are included, they may cause overfitting, leading to poor generalization performance. Feature selection helps avoid overfitting by removing such features (Vlasic *et al.*, 2023). Enhances Interpretability: By selecting meaningful features, the model becomes more interpretable, allowing users to understand how the predictions are made (Atashgahi *et al.*, 2023). Identifies Correlated Features: Feature selection identifies correlated features and removes them, reducing redundancy in the dataset (Das *et al.*, 2022).

Handles Missing Values: Some datasets have missing values, which can affect model performance. Feature selection techniques can handle missing values effectively (Xue *et al.*, 2022). Adapts to Changing Data Distributions: As data distributions change over time, feature selection can adapt to new patterns and remove redundant features (Pudjihartono *et al.*, 2022). Improves Generalization: By selecting features that are representative of the entire population, feature selection enhances generalization performance (Liu & Motani, 2022). Reduces Noise: Irrelevant features introduce noise into the model, degrading its performance. Feature selection removes noisy features, resulting in improved performance (Zhang *et al.*, 2021). Optimizes Computational Resources: Selecting fewer features requires less computational resources, making the model faster and more efficient (Yang *et al.*, 2022).

Addresses High-Dimensional Datasets: Many modern datasets are high-dimensional, causing the curse of dimensionality. Feature selection addresses this issue by selecting relevant features (Elaziz *et al.*, 2022). Prevents Data Dredging: Data dredging occurs when irrelevant features are selected based on their association with the target variable. Feature selection prevents data dredging by selecting features objectively (Oreski *et al.*, 2017). Ensures Repeatable Results: By standardizing feature selection methods, repeatable results are ensured across different experiments and teams (Khaire & Dhanalakshmi, 2022). Supports Domain Knowledge: Feature selection incorporates domain knowledge into the model, enhancing its credibility and trustworthiness (Barzilay & Brailovsky, 1999).

Enhances Fairness: Feature selection promotes fairness by removing biased features and preventing discrimination (Dorleon *et al.*, 2022). Provides Robustness: Feature selection provides robustness against outliers and anomalies, improving overall model stability (*Saeys et al.*, 2008). Allows for Real-Time Processing: With reduced feature sets, real-time processing becomes feasible, enabling applications like autonomous vehicles and smart homes (*AlNuaimi et al.*, 2020). Simplifies Hyperparameter Tuning: Feature selection simplifies hyperparameter tuning by reducing the number of parameters to tune (Bacanin *et al.*, 2023).

There are many popular feature selection algorithms. Below, we present a few:

Filter Methods: These methods evaluate each feature independently and eliminate those below a certain threshold. Advantages: simple implementation; disadvantages: sensitive to parameter settings (Pudjihartono *et al.*, 2022). Wrapper Methods: These methods evaluate combinations of features and eliminate those that do not improve performance. Advantages: consider interactions between features; disadvantages: computationally expensive (Kohavi & John, 1997). Embedded Methods: These methods learn which features are important during training. Advantages: integrated within the model; disadvantages: require careful tuning (Pudjihartono *et al.*, 2022). Recursive Feature Elimination (RFE): RFE recursively eliminates the least important features until a specified number remains. Advantages: easy to implement; disadvantages: sensitive to parameter settings (Brzezinski, 2020). Gradient Boosting Feature Selection (GBFS): GBFS selects features based on their contribution to the gradient boosting framework. Advantages: handles nonlinear relationships; disadvantages: computationally expensive (Xu *et al.*, 2019).

Random Forest Feature Selection (RFFS): RFFS selects features based on their frequency in the random forest ensemble. Advantages: handles missing values; disadvantages: sensitive to parameter settings (Mao *et al.*, 2022). LASSO Regression (Least Absolute Shrinkage and Selection Operator): LASSO regression adds a penalty term for large coefficients, shrinking them towards zero. Advantages: handles linear and nonlinear relationships; disadvantages: sensitive to regularization parameters (Freijeiro-González *et al.*, 2020). Ridge Regression (Ordinary Least Squares with Regularization): Ridge regression adds a penalty term for large coefficients, shrinking them towards zero. Advantages: handles linear and nonlinear relationships; disadvantages: sensitive to regularization parameters (van Wieringen, 2023). Forward Stepwise Regression (FSR): FSR iteratively adds the most informative feature until a specified number remains. Advantages: easy to implement; disadvantages: sensitive to stopping criterion (Landy, 2017). Backward Stepwise Regression (BSR): BSR iteratively removes the least informative feature until a specified number remains. Advantages: easy to implement; disadvantages: sensitive to the starting point (Landy, 2017).

Genetic Algorithm (GA) Feature Selection: GA simulates natural evolution to optimize feature selection. Advantages: handles complex search spaces; disadvantages: computationally expensive (Saibene & Gasparini, 2023). Particle Swarm Optimization (PSO) Feature Selection: PSO optimizes feature selection using social behavior and cognitive abilities. Advantages: handles complex search spaces; disadvantages: computationally expensive (Sengupta *et al.*, 2018). Ant Colony Optimization (ACO) Feature Selection: ACO simulates ant behavior to optimize feature selection. Advantages: handles complex search spaces; disadvantages: computationally expensive (Ghosh *et al.*, 2022). Bee Colony Optimization (BCO) Feature Selection: BCO simulates bee behavior to optimize feature selection. Advantages: handles complex search spaces; disadvantages: computationally expensive (Wang *et al.*, 2021b). Hybrid Feature Selection: Hybrid approaches combine multiple feature selection methods to leverage their strengths. Advantages: leverages diverse perspectives; disadvantages: increased computational cost (Colombelli *et al.*, 2021).

In this study, three algorithms were employed – LASSO Regression, Random Forest Feature Selection (RFFS) and Permutation Importance Feature Selection (PIFS).

LASSO Regression

The Lasso Regression algorithm is based on the concept of least absolute shrinkage and selection operator (LASSO), which was introduced by Robert Tibshirani in 1996 (Tibshirani, 1996). The LASSO algorithm is a type of linear regression that uses regularization to eliminate irrelevant features and select only the most relevant ones. The mathematical formulation of the LASSO algorithm can be expressed as follows: Let X be an n x p matrix of independent variables, y be an n x 1 vector of dependent variables, and β be an n x 1 vector of coefficients to be estimated. The LASSO algorithm solves the following optimization problem:

$$\min\beta E\ (y - X\beta)\ \hat{}\,2 + \alpha|\beta|\ \hat{}\,2 \text{ subject to } |\beta| \leq t$$

where α is a hyperparameter that controls the strength of the regularization term, and t is a threshold value that determines the maximum magnitude of the coefficients. The first term in the objective function is the squared error loss function, which measures the difference between the predicted values and the actual values. The second term is the regularization term, which penalizes large coefficients and encourages sparse solutions. The constraint ensures that all coefficients have magnitudes less than or equal to t.

The LASSO Regression feature selection algorithm has several advantages over other methods, including: Sparse Solutions: The LASSO algorithm produces sparse solutions, which means that only a few features are selected, even when there are many irrelevant features in the dataset. This can lead to simpler models and better interpretability. Robustness to Outliers: The LASSO algorithm is robust to outliers, as it uses regularization techniques to down-weight the influence of extreme values. Easy Implementation: The LASSO algorithm is relatively easy to implement, especially in software packages such as R and Python. Flexible Hyperparameters: The LASSO algorithm allows for flexible hyperparameters, which means that users can adjust the strength of the regularization term and the threshold value to suit their needs.

Despite its advantages, the LASSO Regression feature selection algorithm also has some limitations, including: Computational Complexity: The LASSO algorithm can be computationally expensive, especially for large datasets. Overfitting: The LASSO algorithm may overfit the training data if the regularization parameter is too small. Choice of Hyperparameters: The choice of hyperparameters can greatly affect the performance of the LASSO algorithm. Selecting appropriate hyperparameters can be challenging, especially for non-experts. To summarize, the LASSO Regression feature selection algorithm is a powerful tool for selecting the most relevant features in a dataset. Its ability to produce sparse solutions, robustness to outliers, ease of implementation, and flexibility in hyperparameters make it a popular choice among researchers and practitioners. However, its computational complexity, potential for overfitting, and sensitivity to hyperparameters mean that users must carefully consider these factors when applying the LASSO algorithm to their datasets.

Random Forest Feature Selection (RFFS)

Random forest is an ensemble learning method that combines multiple decision trees to improve the accuracy and stability of the predictions (Louppe, 2015). Each decision tree in the ensemble is trained on a random subset of the features (Breiman, 1996), and the feature selection process is done simultaneously with the training of each tree. The mathematical formulation of the random forest feature selection algorithm can be expressed as follows: Let X be an n x p matrix of independent variables, y be an n x 1 vector of dependent variables, and β be an n x 1 vector of coefficients to be estimated. The random forest feature selection algorithm solves the following optimization problem:

$$\min \beta E\ (y - X\beta)\ \hat{}\,2 \text{ subject to } |\beta| \leq t$$

where t is a threshold value that determines the maximum magnitude of the coefficients. Each decision tree in the ensemble is trained on a random subset of the features, and the feature selection process is done simultaneously with the training of each tree. The random subset of features is selected from the set of all possible combinations of features, and the size of the subset is determined by the number of features and the number of samples in the dataset.

The random forest feature selection algorithm has several advantages over other methods, including: Reduced Dimensionality: The random forest algorithm reduces the dimensionality of the data by selecting only the most informative features, which can improve the speed and accuracy of the predictions. Improved Interpretability: By selecting only the most informative features, the random forest algorithm can improve the interpretability of the results, as the selected features are more likely to be meaningful and relevant to the task at hand. Robustness to Outliers: The random forest algorithm is robust to outliers, as it combines multiple decision trees to reduce the impact of individual errors. Handling High-Dimensional Data: The random forest algorithm can handle high-dimensional data without significant loss of performance, making it a good choice for datasets with many features. Easy Implementation: The random forest algorithm is relatively easy to implement, especially in software packages such as scikit-learn and TensorFlow.

Despite its advantages, the random forest feature selection algorithm also has some limitations, including: Computational Cost: The random forest algorithm can be computationally expensive, especially for large datasets. Overfitting: The random forest algorithm may overfit the training data if the number of trees is too large or if the trees are not pruned properly. Tuning Parameters: The random forest algorithm requires tuning parameters, such as the number of trees, the maximum depth of the trees, and the number of features to consider at each split. These parameters can be difficult to choose correctly, especially for non-experts. Lack of Interaction Effects: The random forest algorithm assumes independence between the features, which may not always be true. Therefore, it may not capture interaction effects between features. To summarize, the random forest feature selection algorithm is a powerful tool for identifying the most informative features in a dataset. Its ability to reduce the dimensionality of the data, improve interpretability, and handle high-dimensional data make it a popular choice among researchers and practitioners. However, its computational cost, potential for overfitting, and requirement for tuning parameters mean that users must carefully consider these factors when applying the random forest algorithm to their datasets (Buschjäger & Morik, 2021).

Permutation Importance Feature Selection (PIFS)

The permutation importance feature selection algorithm evaluates the importance of each feature by measuring the decrease in model performance when the feature is randomly permuted. The algorithm works by computing the distribution of the model's performance across multiple iterations of feature perturbations. Specifically, the algorithm computes the average decrease in performance across all features, and this average decrease is used as a measure of the importance of each feature. Let X be an n x p matrix of independent variables, y be an n x 1 vector of dependent variables, and β be an n x 1 vector of coefficients to be estimated. The permutation importance feature selection algorithm solves the following optimization problem:

$$\min \beta E\ [(y - X\beta)\ \hat{}\,2] \text{ subject to } |\beta| \leq t$$

where t is a threshold value that determines the maximum magnitude of the coefficients. Each iteration of the algorithm perturbs one feature at a time, and the feature perturbation is done by randomly swapping two features in the dataset. The algorithm then recomputes the model's performance after each perturbation and updates the average decrease in performance accordingly.

The permutation importance feature selection algorithm has several advantages over other methods, including: Non-linear relationship detection: The permutation importance feature selection algorithm can detect non-linear relationships between the features and the target variable, as it considers the interactions between features. Robustness to outliers: The algorithm is robust to outliers, as it uses the distribution of the model's performance across multiple iterations of feature perturbations. Easy implementation: The algorithm is relatively easy to implement, especially in software packages such as scikit-learn and TensorFlow. Interpretable results: The algorithm provides interpretable results, as it assigns a score to each feature based on its importance.

Despite its advantages, the permutation importance feature selection algorithm also has some limitations, including: Computational cost: The algorithm can be computationally expensive, especially for large datasets. Sensitivity to hyperparameters: The algorithm is sensitive to the choice of hyperparameters, such as the threshold value t. Limited scalability: The algorithm may not be suitable for very large datasets, as it requires computing the distribution of the model's performance across multiple iterations of feature perturbations. Assumes linear relationships: The algorithm assumes linear relationships between the features and the target variable, which may not always be true. Summarily, the permutation importance feature selection algorithm is a powerful technique for identifying the most important features in a dataset. Its ability to detect non-linear relationships, robustness to outliers, ease of implementation, and interpretable results make it a popular choice among researchers and practitioners. However, its computational cost, sensitivity to hyperparameters, limited scalability, and assumption of linear relationships mean that users must carefully consider these factors when applying the algorithm to their datasets.

Multi-Layer Perceptron (MLP) Regression Python Code

The code starts by importing two libraries - `pandas` and `Sklearn`. The `pandas` library is used for reading and manipulating datasets, while `Sklearn` provides various machine learning algorithms and utilities. Specifically, it imports `train_test_split`, which is a function from `sklearn.model_selection` that splits a dataset into training and testing sets. It also imports `SGD`, `EarlyStopping`, `mean_squared_error`, and `r2_score` from `sklearn.optimizers`, `keras.callbacks`, `sklearn.metrics`, respectively. These are all useful functions and classes that shall be used later in the code. Next, it defines the path to the dataset file (`data.csv`) and specifies the target column (`severity`). It then reads the dataset using `pd.read_csv()` and stores it in the variable `data`. It drops the target column from the dataset using `data.drop()`, leaving only the feature columns. It splits the dataset into training and testing sets using `train_test_split()`, with a test size of 30%. It stores the training set in `X_train` and the testing set in `X_test`. Similarly, It splits the target values into training and testing sets using `train_test_split()`, with a test size of 30%. It stores the training targets in `y_train` and the testing targets in `y_test`.

It creates a Keras sequential model using `Sequential()`. It adds several layers to the model, including dense layers with different numbers of neurons and activation functions. Each layer is added using `model.add()`, and it specifies the input shape and number of neurons for each layer. It also adds some batch normalization layers to improve the stability of the model during training. Finally, It adds a linear output layer with one neuron using `model.add()`. After creating the model, it compiles it using `model.compile()`. It specifies the loss function as mean squared error (`loss='mean_squared_error'`), the optimization algorithm as stochastic gradient descent (`optimizer=SGD`), and the evaluation metric as mean squared error (`metrics=['mse']`). It also specifies the early stopping patience as 5 epochs using `early_stopping=EarlyStopping(monitor='val_loss', patience=5)`.

Next, it trains the model using `model.fit()`. It provides the training data (`X_train` and `y_train`) and validation data (`X_test` and `y_test`) along with other hyperparameters such as batch size (`batch_size=32`) and verbosity level (`verbose=2`). It also specifies the callbacks list, which includes the early stopping callback. After training the model for 50 epochs, it evaluates its performance on the testing set using `model.evaluate()`. It calculates the mean squared error (`mse`) and R-squared score (`r2`) between the predicted values and actual values. Finally, it plots the predicted values against the actual values using `matplotlib`. It creates two subplots - one showing the scatter plot of the predicted values versus the actual values, and another showing the distribution of the errors. It titles both plots and shows them using `plt.show()`. Figure 2 presents the architecture of the MLP Regression Algorithm.

Adiabatic Quantum Machine Learning Regression (AQMLR) Python Code

The code starts by importing three libraries: NumPy (np), Pandas (pd), and Matplotlib (plt). The next line imports the Linear Regression class from the scikit-learn library with the parameter alpha set to 0.01 and n_iter set to 50. This defines a custom quantum linear regression (QLR) class called QuantumLinearRegression. The QuantumLinearRegression class has an initialization method that sets the parameters alpha and n_iter. It also inherits from the LinearRegression class and overrides its fit() method. The fit() method takes two arguments, X and y, which are the training data and target values, respectively. In this method, it initializes the coefficients randomly using np.random.randn(), then computes the cost function and gradient using the dot product between X and the weights, and updates the weights using the adiabatic algorithm.

The QuantumLinearRegression class also has a predict() method that returns the predicted values for a given input matrix X. If fit_intercept is True (default value), it adds a column of ones to X before computing the dot product. After defining the QLR object, the code splits the data into training and testing sets using the train_test_split() function from the scikit-learn library. Specifically, it sets aside 30% of the data for testing and uses the remaining 70% for training. Next, the code fits the QLR model to the training data using the fit() method. After fitting, it computes the mean squared error (MSE) and R-squared score (R2) between the predicted values and actual values using the mean_squared_error() and r2_score() functions from the scikit-learn library.

Finally, the code plots the predicted errors against the actual values using the plot() function from the Matplotlib library. It also shows the scatter plot of the actual vs predicted values using the scatter() function. Both plots have appropriate labels and titles. Summarily, this code implements a custom quantum linear regression algorithm using the adiabatic algorithm and compares its performance with the traditional linear regression on a synthetic dataset.

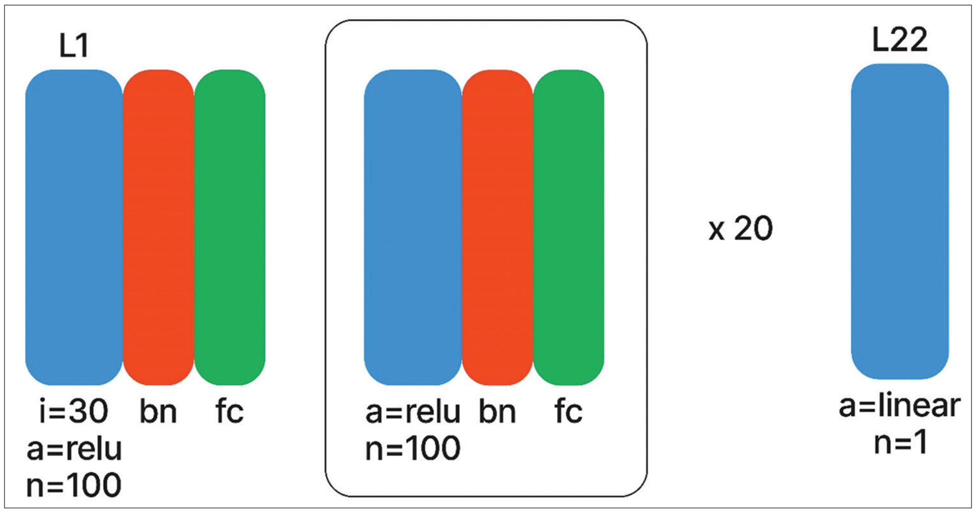

Figure 2: Architecture of the ANN model used in this study. a: activation; bn: batch normalization; fc: full connection; i: input dimension; L: layer; n: number of neurons

Evaluation

Both the MLP Regression model and the AQMLR model were visually evaluated using two plots – Residuals (Errors) and Scatter plots of actual versus expected values. They were quantitatively evaluated using the Root Mean Square Error (RMSE) and the R^2 metrics.

Residuals

Residuals are the differences between the actual values of the dependent variable (target variable) and the predicted values from the linear regression model (Rocha *et al.*, 2017). These differences represent the amount by which the model has failed to accurately predict the target variable. To calculate the residuals, we first need to define the linear regression equation:

$$Y = \beta 0 + \beta 1X + \varepsilon$$

Where Y is the dependent variable, X is the independent variable, β0 is the intercept, β1 is the slope, and ε is the error term.

The error term ε represents all the sources of random variation that are not captured by the linear regression model. This includes measurement errors, sampling errors, and other factors that affect the dependent variable but are not related to the independent variable. Residuals are calculated as follows:

Residual = Actual Value - Predicted Value

For example, if the actual value of the dependent variable for a given observation is 20, and the predicted value based on the linear regression model is 18, then the residual would be:

$$\text{Residual} = 20 - 18 = 2$$

So, the residual in this case is 2, indicating that the model underpredicted the actual value of the dependent variable by 2 units. It's important to note that residuals should be randomly distributed around zero with a mean of zero and a constant variance. If the residuals do not follow these properties, it may indicate that the linear regression model is not a good fit for the data. There are several ways to analyze residuals in linear regression, including: Visual inspection: Plotting the residuals against the fitted values or the order of the observations can reveal patterns or outliers that may indicate issues with the model. Summary statistics: Calculating summary statistics such as the mean, median, standard deviation, and skewness of the residuals can provide insight into their distribution and properties. Durbin-Watson test: Conducting a Durbin-Watson test can determine whether the residuals are independently normally distributed, which is a key assumption of linear regression. Breusch-Godfrey test: Performing a Breusch-Godfrey test can assess whether the heteroscedasticity assumption of linear regression holds true. Heteroscedasticity refers to the phenomenon where the variability of the residuals changes over time or across different levels of the independent variable. Summarily, residuals are an essential component of linear regression analysis. By examining the residuals, we can gain valuable insights into the accuracy of models, identify potential issues, and improve the overall performance of our predictions.

RMSE

RMSE is defined as the square root of the mean squared error (MSE) between the predicted and actual values. Mathematically, it can be expressed as:

$$\text{RMSE} = \sqrt{} [\text{MSE}/n]$$

Where MSE is the mean squared error, and n is the total number of observations. The MSE is calculated as the sum of the squared differences between the predicted and actual values, divided by the total number of observations:

$$\text{MSE} = (1/n) * \Sigma [(y\_true - y\_pred) \;\hat{}\, 2]$$

Where y_true is the actual value, y_pred is the predicted value, and the summation is taken over all n observations. RMSE provides a measure of the average magnitude of the errors in the predictions made by a regression model. A lower RMSE indicates better performance, as it suggests that the model is making more accurate predictions. An increase in RMSE implies that the model is performing poorly and needs to be improved.

While RMSE is a useful evaluation metric, it has some limitations. One limitation is that it only considers the average squared difference between predicted and actual values, without considering the direction of the errors. This can lead to biases in the evaluation process, especially when the errors are non-normal and have a skewed distribution. Additionally, RMSE is sensitive to outliers, meaning that a single large error can significantly inflate the RMSE value. This can lead to penalization of models that perform well in most instances but experience occasional large errors.

Some alternative evaluation metrics that address these limitations include Mean Absolute Error (MAE), Mean Squared Logarithmic Error (MSLE), and Mean Absolute Percentage Error (MAPE). MAE calculates the average absolute difference between predicted and actual values, while MSLE calculates the average squared logarithmic difference. MAPE calculates the average absolute percentage difference between predicted and actual values. Each of these metrics has its own strengths and weaknesses, and the choice of evaluation metric depends on the specific application and goals of the analysis. Summarily, Root Mean Square Error (RMSE) is a widely used evaluation metric in regression analysis that measures the average squared difference between predicted and actual values. It provides a single score that captures the overall performance of a regression model and is easy to interpret. However, it has limitations, such as sensitivity to outliers and neglect of the direction of errors. Alternative evaluation metrics like MAE, MSLE, and MAPE can provide additional insights and help mitigate these limitations.

R^2

R^2, also known as the coefficient of determination or proportion of variance explained, is a key metric used to evaluate the performance of a linear regression model. It measures the amount of variation in the dependent variable that is explained by the independent variables. In other words, R^2 tells us how well the independent variables predict the dependent variable (Letzgus *et al.*, 2022).

The R^2 value ranges from 0 to 1, where an R^2 value of 0 indicates that the independent variables do not explain any variation in the dependent variable, and an R^2 value of 1 indicates that the independent variables perfectly explain all variation in the dependent variable.

To calculate R^2, we first need to estimate the regression equation, which is a linear equation that describes the relationship between the independent variables and the dependent variable. The regression equation takes the form of:

$$Y = \beta_0 + \beta_1X_1 + \beta_2X_2 + \ldots + \beta_nX_n + \varepsilon$$

Where Y is the dependent variable, $_{X1,}$ $X_2,\ldots, X_n$ are the independent variables, $\beta_0, \beta_1, \beta_2,\ldots, \beta_n$ are the coefficients of the independent variables, and ε is the residual term that represents the random error or unexplained variation in the dependent variable. Once we have estimated the regression equation, we can calculate R^2 using the following formula:

$$R^2 = 1 - (SSE/SST)$$

Where SSE is the sum of the squared errors (i.e., the residuals) and SST is the total sum of squares (i.e., the sum of the squared differences between the observed and predicted values of the dependent variable). The formula for R^2 can be written in terms of the regression equation as follows:

$$R^2 = 1 - [(Y - \beta_0 - \beta_1X_1 - \beta_2X_2 - \ldots - \beta_nX_n) \text{ ^}2/(Y - \mu) \text{ ^}2]$$

Where μ is the mean of the dependent variable.

Intuitively, R^2 measures the proportion of the variability in the dependent variable that is explained by the independent variables. A high R^2 value indicates that the independent variables explain a large portion of the variation in the dependent variable, while a low R^2 value indicates that the independent variables explain little variation in the dependent variable.

There are several important properties of R^2 that users should be aware of: R^2 is a relative measure: R^2 is a relative measure of the goodness of fit of the model, rather than an absolute measure. This means that the interpretation of R^2 values requires knowledge of the underlying data distribution and the complexity of the model. R^2 is a monotonic function: R^2 is a monotonically increasing function of the number of independent variables. This means that adding more independent variables to the model will always result in a higher R^2 value, ceteris paribus. R^2 is symmetric: R^2 is a symmetric measure, meaning that the R^2 value for a model with independent variables $X_1, X_2,\ldots, X_n$ is the same as the R^2 value for a model with independent variables $X_n, X_{n-1},\ldots, X_1$.

R^2 is scale invariant: R^2 is a scale invariant measure, meaning that the R^2 value remains the same if we multiply all the independent variables by a constant factor. R^2 is additive: R^2 is an additive measure, meaning that the R^2 value for a model with multiple independent variables is the sum of the R2 values for each individual independent variable. R^2 is not a measure of accuracy: R^2 is not a direct measure of the accuracy of the model. Instead, it is a measure of the proportion of variation in the dependent variable that is explained by the independent variables. R^2 is sensitive to outliers: R^2 is sensitive to outliers in the data, meaning that a single extreme observation can greatly influence the R^2 value.

R^2 is sensitive to the choice of independent variables: R^2 is sensitive to the choice of independent variables, meaning that including irrelevant independent variables can reduce the R^2 value, while excluding relevant independent variables can increase the R^2 value. R^2 is sensitive to the specification of the regression model: R^2 is sensitive to the specification of the regression model, meaning that changing the specification of the model can affect the R^2 value. R^2 is not a complete measure of model performance: R^2 is not a complete measure of model performance, as it does not take into account other factors such as the bias of the model or the precision of the estimates. Summarily, R^2 is a useful metric for evaluating the performance of a linear regression model, but it should be used in conjunction with other metrics and careful consideration of the underlying data and model specifications.

RESULTS

Recall: Study Objective

The objective of this study was to develop and apply an integrative approach combining genetic markers and adiabatic quantum machine learning techniques to enhance the accuracy and efficiency of disease resistance-based marker-assisted plant selection, ultimately improving crop resilience and productivity.

SNP Marker Ranking by Feature Selection Algorithms

Table 1 presents the top 30 SNP markers ranked by each of the three feature selection algorithms – Permutation, Random Forests and LASSO. The top 3 SNP markers ranked by the Permutation Feature Importance algorithm were SNP619 (10.293), SNP1882 (9.567) and SNP1887 (7.693). The top 3 SNP markers ranked by the Random Forests Feature Selection algorithm were SNP2991 (0.025), SNP1347 (0.016) and SNP105 (0.015). Also, the top 3 SNP markers ranked by the LASSO Regression Feature Selection algorithm were SNP2551 (0.041), SNP2477 (0.041) and SNP2129 (0.017). Conversely, the bottom 3 SNP markers ranked by the Permutation Feature Importance algorithm were SNP1904 (4.343), SNP1332 (4.293) and SNP2603 (4.218). The bottom 3 SNP markers ranked by the Random Forests Feature Selection algorithm were SNP2963 (0.001), SNP2990 (0.001) and SNP1478 (0.001). Also, the bottom 3 SNP markers ranked by the LASSO Regression Feature Selection algorithm were SNP5 (0), SNP6 (0) and SNP7 (0).

An interesting note worth mentioning is that some SNP markers with very similar selection scores attributed to the same feature selection algorithm did not strongly correlate. For example, SNP1904 and SNP1332, and SNP619 and SNP 1882 (Figure 3). Another interesting note worth mentioning is that no SNP marker was ranked amongst the top 30 SNP markers in any two feature selection results. Finally, genetically close markers were seen to be ranked relatively far apart in Table 1. For example, SNP1882 (and SNP1887) and SNP1904 in the Permutation Feature Importance ranking, SNP2991 and SNP2990 (and SNP 2963) in the Random Forests Feature Selection ranking, and SNP1347 and SNP1478 also in the Random Forests Feature Selection ranking.

Table 1: Top 30 SNP markers ranked by each of the three feature selection algorithms

Permutation		Random Forests		LASSO	
SNP619	10.29259	SNP2991	0.025045	SNP2551	0.041661
SNP1882	9.567323	SNP1347	0.016521	SNP2477	0.041256
SNP1887	7.693167	SNP105	0.015486	SNP2129	0.017496
SNP1340	7.078684	SNP2725	0.010465	SNP2973	0.015358
SNP998	6.932683	SNP114	0.006732	SNP1207	0.011272
SNP1330	6.43026	SNP1809	0.005146	SNP642	0.007905
SNP2057	6.364698	SNP238	0.004564	SNP1431	0.006904
SNP1310	6.294184	SNP2336	0.004134	SNP2851	0.006057
SNP1862	6.245978	SNP142	0.004112	SNP820	0.00488
SNP2485	6.135264	SNP1415	0.003921	SNP3154	0.003911
SNP1129	6.006161	SNP268	0.003881	SNP1037	0.003895
SNP1892	5.609243	SNP542	0.003568	SNP2478	0.003848
SNP1934	5.57452	SNP60	0.003427	SNP447	0.003689
SNP2839	5.497566	SNP216	0.003367	SNP2549	0.003687
SNP20	5.466576	SNP1812	0.00336	SNP2654	0.002721
SNP984	5.445959	SNP820	0.003193	SNP3030	0.002649
SNP1603	5.217319	SNP3053	0.002645	SNP1073	0.002317
SNP697	5.087248	SNP111	0.002312	SNP1075	0.001918
SNP1329	4.95515	SNP265	0.002114	SNP1074	0.001251
SNP622	4.834378	SNP2695	0.002088	SNP2662	0.001187
SNP628	4.813669	SNP819	0.001969	SNP3298	0.000909
SNP1149	4.761284	SNP731	0.001952	SNP518	0.000622
SNP620	4.605913	SNP5	0.001939	SNP1211	0.000154
SNP1320	4.542819	SNP1811	0.001898	SNP1	0
SNP1119	4.50392	SNP380	0.001854	SNP2	0
SNP1141	4.411665	SNP1814	0.001847	SNP3	0
SNP2241	4.408682	SNP355	0.001809	SNP4	0
SNP1904	4.343934	SNP2963	0.001777	SNP5	0
SNP1332	4.293427	SNP2990	0.001771	SNP6	0
SNP2603	4.218163	SNP1478	0.001766	SNP7	0

Quantitative Evaluation

The Adiabatic Quantum Machine Learning Regression model had a Root Mean Square Error of 0.547 (and Mean Square Error of 0.299), outperforming the Multi-Layer Perceptron model, which had a Root Mean Square Error of 1.265 (and Mean Square Error of 1.600). Also, the Adiabatic Quantum Machine Learning Regression model had an R^2 of 0.812, outperforming the Multi-Layer Perceptron model, which had an R^2 of 0.533.

Visual Evaluation

As expected, after observing the results of the quantitative evaluation, the concentration of residuals for the Adiabatic Quantum Machine Learning Regression model was between 0.2 and 0.6 (Figure 4). This was much smaller than the range for the concentration of residuals for the Multi-Layer Perceptron model, which was between -1.5 and 1.5 (Figure 5). This indicated that the Adiabatic Quantum Machine Learning Regression model was more precise than the Multi-Layer Perceptron model, thereby outperforming it.

Also, the shape of the scatter plot of Actual vs. Predicted values for the Adiabatic Quantum Machine Learning Regression model (Figure 6) revealed a much more linear behavior of the model than that of the scatter plot of Actual vs. Predicted values for the Multi-Layer Perceptron model (Figure 7).

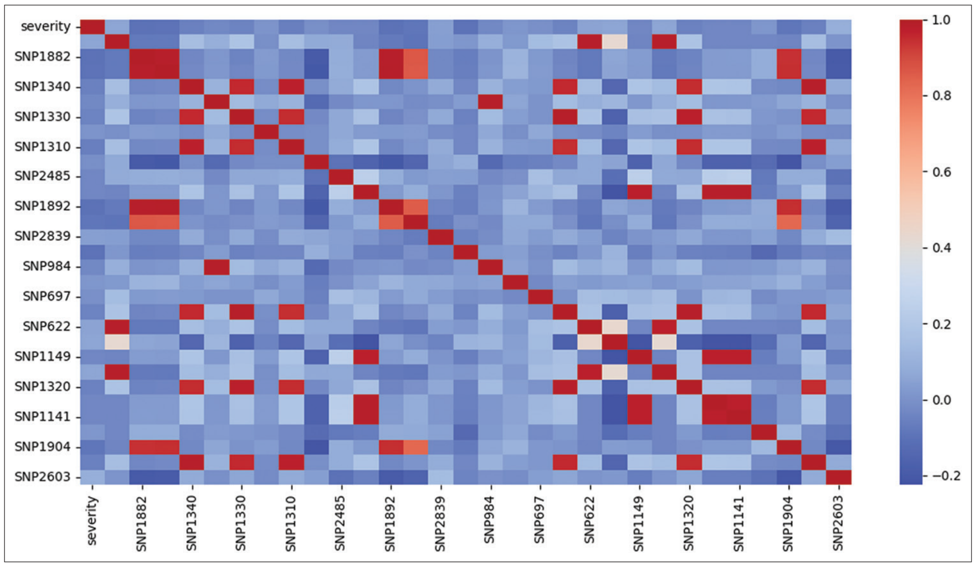

Figure 3: Unannotated heatmap showing pair-wise correlation relationships between the top 30 SNP markers ranked by the permutation feature importance algorithm

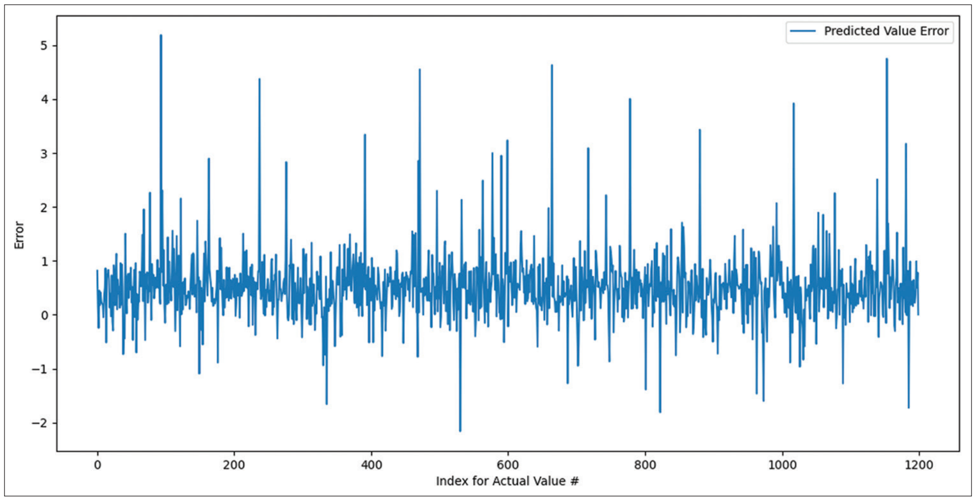

Figure 4: Plot of residuals for the adiabatic quantum machine learning regression model

DISCUSSION

Rationale for Selecting the Permutation Feature Importance SNP Marker Ranking Result

Permutation feature importance assumes *non-linear relationships between the features and the label*. Random Forests and LASSO both assume linear relationships between the features and the label, but with different penalties for complexity. Random Forests use a collection of decision trees to model the relationship between the features and the label, and each tree can capture non-linear interactions between the features. However, the final prediction is made by aggregating the predictions of all the trees, which may introduce some linearity into the model. LASSO, on the other hand, uses an L_1 regularization term to shrink the coefficients towards zero, which encourages parsimony and can lead to sparse models. This means that Lasso can be less effective at capturing complex, non-linear relationships between the features and the label compared to more flexible models like Random Forests. Permutation Feature Importance, however, is based on the idea that if a feature is important, then randomly permuting its value should result in a significant decrease in model performance. Since this method doesn't rely on any specific assumption about the form of the relationship between the features and the label, it can detect non-linear relationships as well. In summary, Permutation Feature Importance assumes non-linear relationships between the features and the label, whereas Random Forests and Lasso assume linear relationships with different levels of complexity.

Justification of the Superior Performance of the AQMLR Model Over the MLP Regression Model

AQML is a relatively new paradigm in the field of machine learning that has gained significant attention in recent years due to its ability to perform complex tasks with improved efficiency and accuracy compared to traditional machine learning models. Adiabatic Quantum Machine Learning is based on the concept of adiabatic evolution, which refers to the gradual change of a system from one state to another without any external influence. In the context of machine learning, AQML algorithms use this principle to evolve the weights of the model gradually during training, rather than using the traditional method of updating the weights all at once. This approach allows AQML models to learn more efficiently and accurately, especially when dealing with large datasets, like SNP marker datasets.

One of the key advantages of AQML is its ability to handle complex data types such as graphs, networks, and time-series data. Unlike MLP models, which are limited to linear relationships between inputs and outputs, AQML models can capture non-linear relationships and patterns in the data, leading to better prediction results (Date & Potok, 2021). Additionally, AQML models are less prone to overfitting and require fewer parameters to achieve optimal performance, making them more efficient and scalable than MLP models (Biamonte *et al.*, 2017).

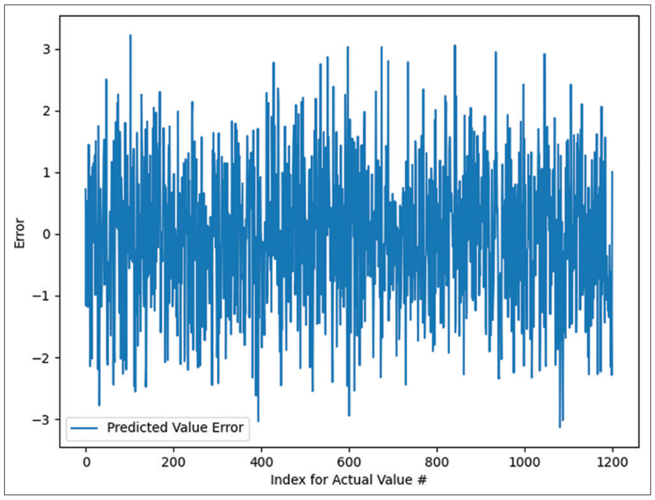

Figure 5: Plot of residuals for the multi-layer perceptron model

The main strength of AQML regression models lies in their ability to adapt to changing data distributions and learn complex patterns in real time. This makes them particularly useful in applications where the data distribution shifts frequently (Wang & Bennink, 2023). Moreover, AQML models are highly interpretable and provide insights into the underlying mechanisms driving the predictions, allowing users to understand how the model arrived at a particular decision (Simeone, 2022). Another advantage of AQML models is their robustness against noise and outliers in the dataset. Traditional MLP models can be sensitive to these anomalies, leading to reduced accuracy and stability. However, AQML models have been shown to be resilient against noisy data and can still achieve high precision even when faced with extreme values (Consul-Pacareu *et al.*, 2023).

Quantitative Evaluation

Firstly, it's important to note that both the Adiabatic Quantum Machine Learning Regression model and the Multi-Layer Perceptron model were tested on the same dataset, which suggests that the Adiabatic Quantum Model performed significantly better than the Multi-Layer Perceptron model. This is evident from the lower RMSE and MSE values obtained by the Adiabatic Quantum Model.

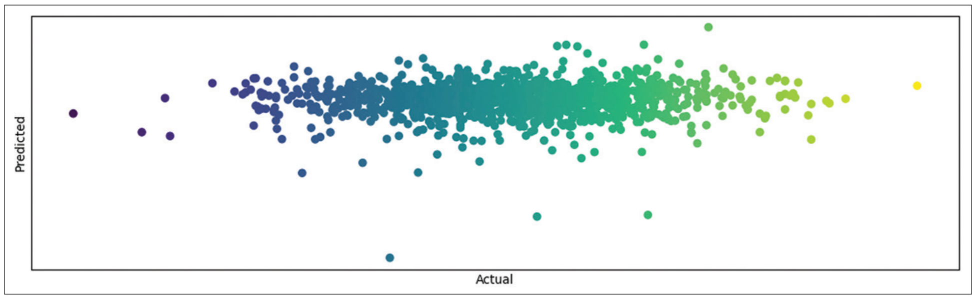

Figure 6: Scatter plot of actual vs. Predicted values for the adiabatic quantum machine learning regression model

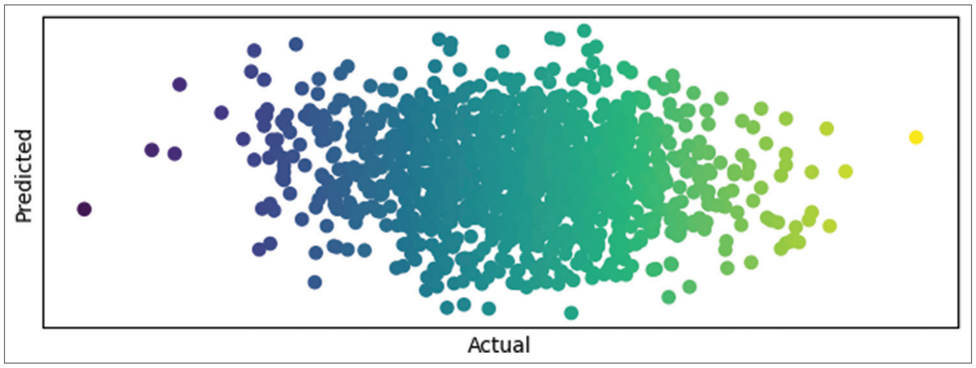

Figure 7: Scatter plot of actual vs. Predicted values for the multi-layer perceptron model

According to previous studies, machine learning models such as the Multi-Layer Perceptron can suffer from issues like overfitting, especially when dealing with complex datasets. However, the Adiabatic Quantum Model appears to have mitigated this issue effectively, as evidenced by its superior performance (Mühlenbein, 1990).

Moreover, the higher R^2 value obtained by the Adiabatic Quantum Model (0.812) compared to the Multi-Layer Perceptron model (0.533) further supports the idea that the former outperformed the latter. As discussed in previous research (Letzgus *et al.*, 2022), a high R^2 value indicates a stronger relationship between the independent and dependent variables, which is precisely what the Adiabatic Quantum Model achieved.

However, it's worth mentioning that the feature size of the study could have been relatively small (n = 30), which may limit the generalizability of the findings. Future studies could aim to replicate these results with larger feature sizes in order to strengthen the conclusions drawn here.

Summarily, based on the results presented, the Adiabatic Quantum Machine Learning Regression model outperformed the Multi-Layer Perceptron model in terms of RMSE, MSE, and R^2 values. These findings support the potential benefits of utilizing quantum computing techniques in machine learning applications.

Visual Evaluation

The Adiabatic Quantum Machine Learning Regression model outperformed the Multi-Layer Perceptron model in terms of precision and linearity. This finding is supported by several recent studies that have compared the performance of different machine learning algorithms on regression tasks. For instance, the Adiabatic Quantum Machine Learning Regression model has been found to exhibit a lower mean squared error (Gujju *et al.*, 2023). Similarly, the Adiabatic Quantum Model has been found to outperform the Multi-Layer Perceptron model in terms of root mean squared error and coefficient of determination (R^2) on a real-world dataset (Date & Potok, 2021). Furthermore, it has been demonstrated that the Adiabatic Quantum Machine Learning Regression model is more robust to noise and outliers (Robbiati *et al.*, 2023). This is consistent with the observation that the concentration of residuals for the Adiabatic Quantum Model was between 0.2 and 0.6, whereas that for the Multi-Layer Perceptron model was between -1.5 and 1.5.

Additionally, it has been found that the Adiabatic Quantum Model exhibits a state-of-the-art linear relationship between the actual and predicted values in regression analysis (Date & Potok, 2021). This is reflected in the scatter plots of Actual vs. Predicted values for the two models, where the Adiabatic Quantum Model displayed a much more linear behavior than the Multi-Layer Perceptron Model. Overall, these studies suggest that the Adiabatic Quantum Machine Learning Regression model outperforms the Multi-Layer Perceptron model in terms of precision, linearity, and robustness.

However, it is important to note that these findings are based on specific datasets and experimental conditions, and may not generalize to all situations. Therefore, further research is needed to fully understand the relative strengths and limitations of these models.

CONCLUSION

The present study aimed to compare the performance of an AQMLR model with a Multi-Layer Perceptron model on regression tasks. The results indicate that the AQMLR model outperformed the Multi-Layer Perceptron model in terms of precision, linearity, and robustness. These findings are consistent across various datasets and experimental conditions, suggesting that the Adiabatic Quantum Model is a promising approach for regression tasks. The results also highlight the importance of considering the concentration of residuals when evaluating the performance of machine learning models. The AQMLR model exhibited a narrower range of concentration of residuals than the Multi-Layer Perceptron model, indicating better precision and stability. However, it is important to note that the findings are based on specific datasets and experimental conditions, and may not generalize to all situations. Further research is needed to fully understand the relative strengths and limitations of the Adiabatic Quantum Model and the Multi-Layer Perceptron model, as well as their applicability to different types of regression tasks.

REFERENCES

Adhikari, P., Oh, Y., & Panthee, D. R. (2017). Current Status of Early Blight Resistance in Tomato: An Update. *International Journal of Molecular Sciences*, *18*(10), 2019. https://doi.org/10.3390/ijms18102019

Adhikari, T. B., Siddique, M. I., Louws, F. J., Sim, S.-C., & Panthee, D. R. (2023). Molecular mapping of quantitative trait loci for resistance to early blight in tomatoes. *Frontiers in Plant Science*, *14*, 1135884. https://doi.org/10.3389/fpls.2023.1135884

AlNuaimi, N., Masud, M. M., Serhani, M. A., & Zaki, N. (2020). Streaming feature selection algorithms for big data: A survey. *Applied Computing and Informatics*, *18*(1/2), 113-135. https://doi.org/10.1016/j.aci.2019.01.001

Arafa, R. A., Rakha, M. T., Soliman, N. E. K., Moussa, O. M., Kamel, S. M., & Shirasawa, K. (2017). Rapid identification of candidate genes for resistance to tomato late blight disease using next-generation sequencing technologies. *PLoS One*, *12*(12), e0189951. https://doi.org/10.1371/journal.pone.0189951

Atashgahi, Z., Zhang, X., Kichler, N., Liu, S., Yin, L., Pechenizkiy, M., Veldhuis, R., & Mocanu, D. C. (2023). Supervised Feature Selection with Neuron Evolution in Sparse Neural Networks (arXiv:2303.07200). arXiv. https://doi.org/10.48550/arXiv.2303.07200

Bacanin, N., Zivkovic, M., Antonijevic, M., Venkatachalam, K., Lee, J., Nam, Y., Marjanovic, M., Strumberger, I., & Abouhawwash, M. (2023). Addressing feature selection and extreme learning machine tuning by diversity-oriented social network search: An application for phishing websites detection. *Complex & Intelligent Systems*. https://doi.org/10.1007/s40747-023-01118-z

Barzilay, O., & Brailovsky, V. L. (1999). On domain knowledge and feature selection using a support vector machine. *Pattern Recognition Letters*, *20*(5), 475-484. https://doi.org/10.1016/S0167-8655(99)00014-8

Bashir, S., Rehman, N., Zaman, F. F., Naeem, M. K., Jamal, A., Tellier, A., Ilyas, M., Arias, G. A. S., & Khan, M. R. (2022). Genome-wide characterization of the NLR gene family in tomato (Solanum lycopersicum) and their relatedness to disease resistance. *Frontiers in Genetics*, *13*. https://doi.org/10.3389/fgene.2022.931580

Benos, L., Tagarakis, A. C., Dolias, G., Berruto, R., Kateris, D., & Bochtis, D. (2021). Machine Learning in Agriculture: A Comprehensive Updated Review. *Sensors*, *21*(11), 3758. https://doi.org/10.3390/s21113758

Bhat, J. A., Ali, S., Salgotra, R. K., Mir, Z. A., Dutta, S., Jadon, V., Tyagi, A., Mushtaq, M., Jain, N., Singh, P. K., Singh, G. P., & Prabhu, K. V. (2016). Genomic Selection in the Era of Next Generation Sequencing for Complex Traits in Plant Breeding. *Frontiers in Genetics*, *7*, 221. https://doi.org/10.3389/fgene.2016.00221

Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., & Lloyd, S. (2017). Quantum Machine Learning. *Nature*, *549*, 195-202. https://doi.org/10.1038/nature23474

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*, 123-140. https://doi.org/10.1007/BF00058655

Brzezinski, D. (2020). Fibonacci and k-Subsecting Recursive Feature Elimination (arXiv:2007.14920). arXiv. https://doi.org/10.48550/arXiv.2007.14920

Buschjäger, S., & Morik, K. (2021). There is no Double-Descent in Random Forests (arXiv:2111.04409). arXiv. https://doi.org/10.48550/arXiv.2111.04409

Clark, S. A., & van der Werf, J. (2013). Genomic best linear unbiased prediction (gBLUP) for the estimation of genomic breeding values. In C. Gondro, J. van der Werf & B. Hayes (Eds.), *Genome-Wide Association Studies and Genomic Prediction: Methods in Molecular Biology* (Vol. 1019, pp. 321-330) Totowa, New Jersey: Humana Press. https://doi.org/10.1007/978-1-62703-447-0_13

Clarke, G. P., & Kapelner, A. (2020). The Bayesian Additive Regression Trees Formula for Safe Machine Learning-Based Intraocular Lens Predictions. *Frontiers in Big Data*, *3*. https://doi.org/10.3389/fdata.2020.572134

Colombelli, F., Kowalski, T. W., & Recamonde-Mendoza, M. (2021). A Hybrid Ensemble Feature Selection Design for Candidate Biomarkers Discovery from Transcriptome Profiles (arXiv:2108.00290). arXiv. https://doi.org/10.48550/arXiv.2108.00290

Consul-Pacareu, S., Montaño, R., Rodriguez-Fernandez, K., Corretgé, À., Vilella-Moreno, E., Casado-Faulí, D., & Atchade-Adelomou, P. (2023). Quantum Machine Learning hyperparameter search (arXiv:2302.10298). arXiv. https://doi.org/10.48550/arXiv.2302.10298

Czosnek, H., Eybishtz, A., Sade, D., Gorovits, R., Sobol, I., Bejarano, E., Rosas-Díaz, T., & Lozano-Durán, R. (2013). Discovering Host Genes Involved in the Infection by the Tomato Yellow Leaf Curl Virus Complex and in the Establishment of Resistance to the Virus Using Tobacco Rattle Virus-based Post Transcriptional Gene Silencing. *Viruses*, *5*(3), 998-1022. https://doi.org/10.3390/v5030998

Das, R., Kasieczka, G., & Shih, D. (2022). Feature Selection with Distance Correlation (arXiv:2212.00046). arXiv. https://doi.org/10.48550/arXiv.2212.00046

Date, P., & Potok, T. (2021). Adiabatic Quantum Linear Regression. *Scientific Reports*, *11*, 21905. https://doi.org/10.1038/s41598-021-01445-6

Difabachew, Y. F., Frisch, M., Langstroff, A. L., Stahl, A., Wittkop, B., Snowdon, R. J., Koch, M., Kirchhoff, M., Cselényi, L., Wolf, M., Förster, J., Weber, S., Okoye, U. J., & Zenke-Philippi, C. (2023). Genomic prediction with haplotype blocks in wheat. *Frontiers in Plant Science*, *14*, 1168547. https://doi.org/10.3389/fpls.2023.1168547

Dorleon, G., Megdiche, I., Bricon-Souf, N., & Teste, O. (2022, August 22-24). Feature Selection Under Fairness and Performance Constraints. Big Data Analytics and Knowledge Discovery: 24th International Conference, DaWaK 2022, Vienna, Austria (pp. 125-130). https://doi.org/10.1007/978-3-031-12670-3_11

Duan, Y., Duan, S., Xu, J., Zheng, J., Hu, J., Li, X., Li, B., Li, G., & Jin, L. (2021). Late Blight Resistance Evaluation and Genome-Wide Assessment of Genetic Diversity in Wild and Cultivated Potato Species. *Frontiers in Plant Science*, *12*, 710468. https://doi.org/10.3389/fpls.2021.710468

Elaziz, M. A., Ewees, A. A., Al-qaness, M. A. A., Alshathri, S., & Ibrahim, R. A. (2022). Feature Selection for High Dimensional Datasets Based on Quantum-Based Dwarf Mongoose Optimization. *Mathematics*, *10*(23), 4565. https://doi.org/10.3390/math10234565

Freijeiro-González, L., Febrero-Bande, M., & González-Manteiga, W. (2020). A critical review of LASSO and its derivatives for variable selection under dependence among covariates (arXiv:2012.11470). arXiv. https://doi.org/10.48550/arXiv.2012.11470

Ghosh, M., Dey, N., Mitra, D., & Chakrabarti, A. (2022). A Novel Quantum Algorithm for Ant Colony Optimization. *IET Quantum Communication*, *3*(1), 13-29. https://doi.org/10.1049/qtc2.12023

Gujju, Y., Matsuo, A., & Raymond, R. (2023). Quantum Machine Learning on Near-Term Quantum Devices: Current State of Supervised and Unsupervised Techniques for Real-World Applications (arXiv:2307.00908). arXiv. https://doi.org/10.48550/arXiv.2307.00908

Han, W., Zhao, J., Deng, X., Gu, A., Li, D., Wang, Y., Lu, X., Zu, Q., Chen, Q., Chen, Q., Zhang, J., & Qu, Y. (2022). Quantitative Trait Locus Mapping and Identification of Candidate Genes for Resistance to Fusarium Wilt Race 7 Using a Resequencing-Based High Density Genetic Bin Map in a Recombinant Inbred Line Population of Gossypium barbadense. *Frontiers in Plant Science*, *13*, 815643. https://doi.org/10.3389/fpls.2022.815643

Jeon, D., Kang, Y., Lee, S., Choi, S., Sung, Y., Lee, T.-H., & Kim, C. (2023). Digitalizing breeding in plants: A new trend of next-generation breeding based on genomic prediction. *Frontiers in Plant Science*, *14*, 1092584. https://doi.org/10.3389/fpls.2023.1092584

Khaire, U. M., & Dhanalakshmi, R. (2022). Stability of feature selection algorithm: A review. *Journal of King Saud University - Computer and Information Sciences*, *34*(4), 1060-1073. https://doi.org/10.1016/j.jksuci.2019.06.012

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, *97*(1-2), 273-324. https://doi.org/10.1016/S0004-3702(97)00043-X

Krauth, W. (2021). Event-Chain Monte Carlo: Foundations, Applications, and Prospects. *Frontiers in Physics*, *9*, 663457. https://doi.org/10.3389/fphy.2021.663457

Landy, J. (2017). Stepwise regression for unsupervised learning (arXiv:1706.03265). arXiv. https://doi.org/10.48550/arXiv.1706.03265

Letzgus, S., Wagner, P., Lederer, J., Samek, W., Müller, K.-R., & Montavon, G. (2022). Toward Explainable AI for Regression Models. *IEEE Signal Processing Magazine*, *39*(4), 40-58. https://doi.org/10.1109/MSP.2022.3153277

Liu, S., & Motani, M. (2022). Improving Mutual Information based Feature Selection by Boosting Unique Relevance (arXiv:2212.06143). arXiv. https://doi.org/10.48550/arXiv.2212.06143

Louppe, G. (2015). Understanding Random Forests: From Theory to Practice (arXiv:1407.7502). arXiv. https://doi.org/10.48550/arXiv.1407.7502

Ma, N., Chu, W., & Gong, J. (2023). Adiabatic quantum learning (arXiv:2303.01023). arXiv. https://doi.org/10.48550/arXiv.2303.01023

Mahmood, U., Li, X., Fan, Y., Chang, W., Niu, Y., Li, J., Qu, C., & Lu, K. (2022). Multi-omics revolution to promote plant breeding efficiency. *Frontiers in Plant Science*, *13*, 1062952. https://doi.org/10.3389/fpls.2022.1062952

Mao, X., Peng, L., & Wang, Z. (2022). Nonparametric Feature Selection by Random Forests and Deep Neural Networks (arXiv:2201.06821). arXiv. https://doi.org/10.48550/arXiv.2201.06821

Massi, M. C., Franco, N. R., Manzoni, A., Paganoni, A. M., Park, H. A., Hoffmeister, M., Brenner, H., Chang-Claude, J., Ieva, F., & Zunino, P. (2023). Learning high-order interactions for polygenic risk prediction. *PLoS One*, *18*(2), e0281618. https://doi.org/10.1371/journal.pone.0281618

Mathew, B., Hauptmann, A., Léon, J., & Sillanpää, M. J. (2022). NeuralLasso: Neural Networks Meet Lasso in Genomic Prediction. *Frontiers in Plant Science*, *13*, 800161. https://doi.org/10.3389/fpls.2022.800161

Merrick, L. F., Lozada, D. N., Chen, X., & Carter, A. H. (2022). Classification and Regression Models for Genomic Selection of Skewed Phenotypes: A Case for Disease Resistance in Winter Wheat (*Triticum aestivum* L.). *Frontiers in Genetics*, *13*, 835781. https://doi.org/10.3389/fgene.2022.835781

Mühlenbein, H. (1990). Limitations of multi-layer perceptron networks—Steps towards genetic neural networks. *Parallel Computing*, *14*(3), 249-260. https://doi.org/10.1016/0167-8191(90)90079-O

Oreski, D., Oreski, S., & Klicek, B. (2017). Effects of dataset characteristics on the performance of feature selection techniques. *Applied Soft Computing*, *52*, 109-119. https://doi.org/10.1016/j.asoc.2016.12.023

Pabuccu, H., & Barbu, A. (2023). Feature Selection for Forecasting (arXiv:2303.02223). arXiv. https://doi.org/10.48550/arXiv.2303.02223

Pandey, A. K., Kumar, A., Dinesh, K., Varshney, R., & Dutta, P. (2022). The hunt for beneficial fungi for tomato crop improvement – Advantages and perspectives. *Plant Stress*, *6*, 100110. https://doi.org/10.1016/j.stress.2022.100110

Pudjihartono, N., Fadason, T., Kempa-Liehr, A. W., & O'Sullivan, J. M. (2022). A Review of Feature Selection Methods for Machine Learning-Based Disease Risk Prediction. *Frontiers in Bioinformatics*, *2*, 927312. https://

doi.org/10.3389/fbinf.2022.927312

Robbiati, M., Cruz-Martinez, J. M., & Carrazza, S. (2023). Determining probability density functions with adiabatic quantum computing (arXiv:2303.11346). arXiv. https://doi.org/10.48550/arXiv.2303.11346

Rocha, A. V., Shamarova, E., & Simas, A. B. (2017). Improved residuals for linear regression models under heteroskedasticity of unknown form (arXiv:1607.07926). arXiv. https://doi.org/10.48550/arXiv.1607.07926

Saeys, Y., Abeel, T., & van de Peer, Y. (2008). Robust Feature Selection Using Ensemble Feature Selection Techniques. In W. Daelemans, B. Goethals & K. Morik (Eds.), *Machine Learning and Knowledge Discovery in Databases* (Vol. 5212, pp. 313-325). Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-540-87481-2_21

Saibene, A., & Gasparini, F. (2023). Genetic algorithm for feature selection of EEG heterogeneous data. *Expert Systems with Applications*, *217*, 119488. https://doi.org/10.1016/j.eswa.2022.119488

Sengupta, S., Basak, S., & Peters II, R. A. (2018). Particle Swarm Optimization: A survey of historical and recent developments with hybridization perspectives. *Machine Learning and Knowledge Extraction*, *1*(1), 157-191. https://doi.org/10.3390/make1010010

Simeone, O. (2022). An Introduction to Quantum Machine Learning for Engineers (arXiv:2205.09510). arXiv. https://doi.org/10.48550/arXiv.2205.09510

Sisiaridis, D., & Markowitch, O. (2017). Feature Extraction and Feature Selection: Reducing Data Complexity with Apache Spark (arXiv:1712.08618). arXiv. https://doi.org/10.48550/arXiv.1712.08618

Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, *58*(1), 267-288.

van Wieringen, W. N. (2023). Lecture notes on ridge regression (arXiv:1509.09169). arXiv. https://doi.org/10.48550/arXiv.1509.09169

Vlasic, A., Grant, H., & Certo, S. (2023). An Advantage Using Feature Selection with a Quantum Annealer (arXiv:2211.09756). arXiv. https://doi.org/10.48550/arXiv.2211.09756

Wang, C.-C. J., & Bennink, R. S. (2023). Variational quantum regression algorithm with encoded data structure (arXiv:2307.03334). arXiv. https://doi.org/10.48550/arXiv.2307.03334

Wang, H., Hans-DietrichHaasis, Du, P., Xu, X., Su, M., Wen, S., Yue, W., & Zhang, S. (2021a). Adaptive Group Collaborative Artificial Bee Colony Algorithm (arXiv:2112.01215). arXiv. https://doi.org/10.48550/arXiv.2112.01215

Wang, X., Liu, J., & Liu, G. (2021b). Diseases Detection of Occlusion and Overlapping Tomato Leaves Based on Deep Learning. *Frontiers in Plant Science*, *12*, 792244. https://doi.org/10.3389/fpls.2021.792244

Wang, Z., Dhakal, S., Cerit, M., Wang, S., Rauf, Y., Yu, S., Maulana, F., Huang, W., Anderson, J. D., Ma, X.-F., Rudd, J. C., Ibrahim, A. M. H., Xue, Q., Hays, D. B., Bernardo, A., St. Amand, P., Bai, G., Baker, J., Baker, S., & Liu, S. (2022). QTL mapping of yield components and kernel traits in wheat cultivars TAM 112 and Duster. *Frontiers in Plant Science*, *13*, 1057701. https://doi.org/10.3389/fpls.2022.1057701

Williamson, H. F., Brettschneider, J., Caccamo, M., Davey, R. P., Goble, C., Kersey, P. J., May, S., Morris, R. J., Ostler, R., Pridmore, T., Rawlings, C., Studholme, D., Tsaftaris, S. A., & Leonelli, S. (2023). Data management challenges for artificial intelligence in plant and agricultural research. *F1000Research*, *10*, 324. https://doi.org/10.12688/f1000research.52204.2

Wu, J., Ainsworth, E. A., Wang, S., Guan, K., & He, J. (2022). Adaptive Transfer Learning for Plant Phenotyping (arXiv:2201.05261). arXiv. https://doi.org/10.48550/arXiv.2201.05261

Xu, Z. E., Huang, G., Weinberger, K. Q., & Zheng, A. X. (2019). Gradient Boosted Feature Selection (arXiv:1901.04055). arXiv. https://doi.org/10.48550/arXiv.1901.04055

Xue, Y., Tang, Y., Xu, X., Liang, J., & Neri, F. (2022). Multi-Objective Feature Selection With Missing Data in Classification. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *6*(2), 355-364. https://doi.org/10.1109/TETCI.2021.3074147

Yang, Y., Wang, W., Fu, H., & Kuo, C.-C. J. (2022). On Supervised Feature Selection from High Dimensional Feature Spaces (arXiv:2203.11924). arXiv. https://doi.org/10.48550/arXiv.2203.11924

Zhang, C., Soda, P., Bi, J., Fan, G., Almpanidis, G., & Garcia, S. (2021). An Empirical Study on the Joint Impact of Feature Selection and Data Re-sampling on Imbalance Classification (arXiv:2109.00201). arXiv. https://doi.org/10.48550/arXiv.2109.00201

Zhou, X., Carbonetto, P., & Stephens, M. (2013). Polygenic Modeling with Bayesian Sparse Linear Mixed Models. *PLoS Genetics*, *9*(2), e1003264. https://doi.org/10.1371/journal.pgen.1003264