Introduction

Machine learning is the use of computers to perform a task using patterns instead of specific instructions. Its applications are wide-ranging, including loan approval, flower classification, and generative artificial intelligence. In this project, we use machine learning to predict the quality of red wine based on 11 different variables (fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, and alcohol). All the data comes from a study by the University of Minho in Portugal. In this project, we used a k-nearest neighbors classifier, an artificial neural network model, and a random forest classifier to optimally predict and classify wine quality. We found that a random forest model had the strongest performance with an accuracy of 0.68 on the test data.

Exploratory Analysis

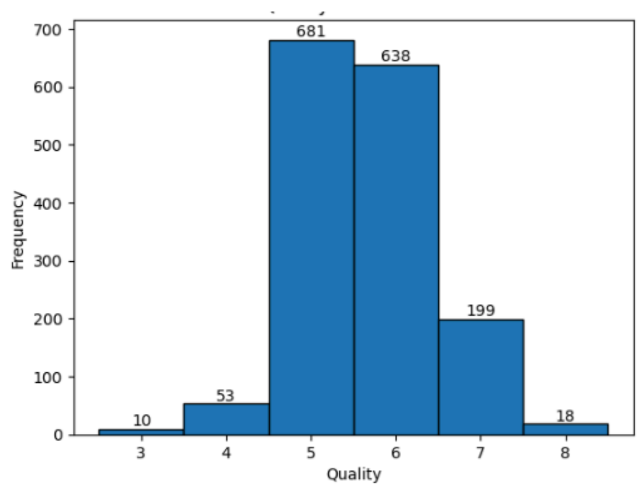
The data consists of 1599 different red wines and their accompanying 12 features. Importantly, none of these features are null, enabling a more robust and comprehensive analysis of each wine. Table 1 highlights the key characteristics of each of the 12 features. Importantly, quality, which is the feature we seek to predict, is measured on a scale from 0-10 (where 10 is very good) and it is the median of 3 wine experts' opinions. However, the wines in the dataset have quality values ranging from 3 to 8, so there are only 6 discrete qualities from which we predict rather than 11.

Table 1: Feature Characteristic Analysis

Feature	Minimum	Mean	Maximum
Fixed Acidity	4.600000	8.319637	15.900000
Volatile Acidity	0.120000	0.527821	1.580000
Citric Acid	0.000000	0.270976	1.000000
Residual Sugar	0.900000	2.538806	15.500000
Chlorides	0.012000	0.087467	0.611000
Free Sulfur Dioxide	1.000000	15.874922	72.000000
Total Sulfur Dioxide	6.000000	46.467792	289.000000
Density	0.990070	0.996747	1.003690
pH	2.740000	3.311113	4.010000
Sulphates	0.330000	0.658149	2.000000
Alcohol	8.400000	10.422983	14.900000
Quality	3.000000	5.636023	8.000000

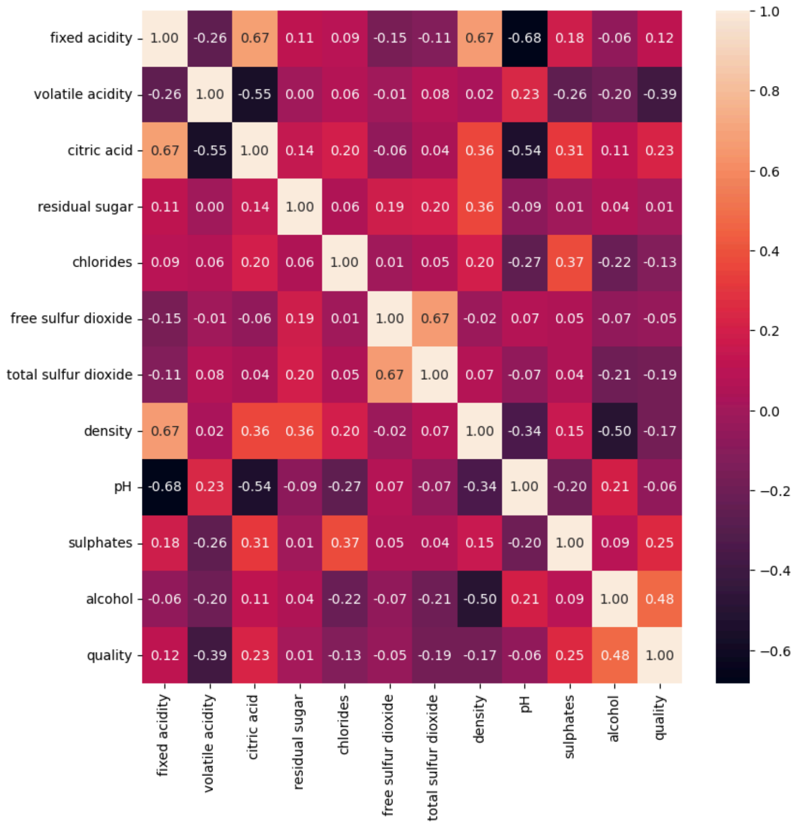
Looking at the distribution of quality in Figure 1, the quality of wine is centered around 5 and 6, with 82% of the wines in the data having quality 5 or 6. The distribution is relatively symmetric.

Figure 1: Quality Histogram



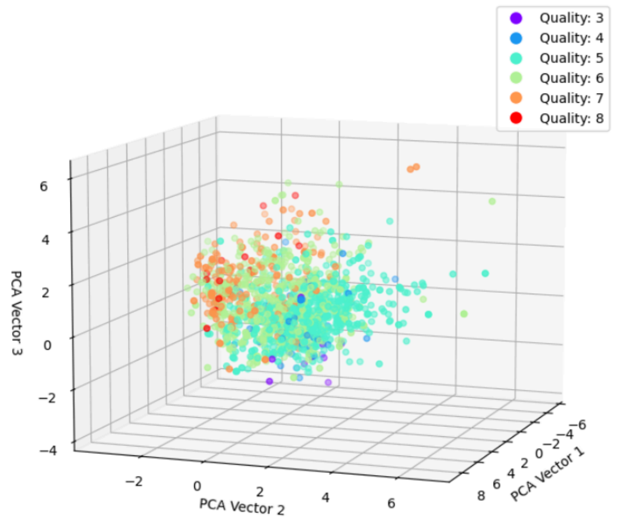
When looking at the correlation coefficients between features in Figure 2, no two features have a correlation coefficient with a magnitude greater than 0.7, meaning none of the features are strongly correlated. There are some features with correlation magnitudes greater than 0.6 such as relationships between the acidities and pH, as well as the two sulfur dioxides measurements. These make sense since acidity and pH are measuring the same thing, and free sulfur dioxide is a component of the total sulfur dioxide measurement. Overall, the relatively moderate correlation coefficients are not a cause for concern.

Figure 2: Feature Correlations



Since there are 11 features we use to predict quality, it is difficult to visualize the dataset. Thus, we use principal component analysis (PCA) to visualize the dataset. We chose to use 3 vectors for PCA since that is the most we can fit while making a scatterplot viewable. These 3 vectors comprise 59.8% of the variance in the data, which is quite strong given the number of dimensions was reduced from 11 to 3. The result is plotted in Figure 3 below, and it is clear that there is a bit of a stratification developing between the higher quality (warmer colors) and the lower quality (cooler colors), although it is certainly not linearly separable.

Figure 3: PCA Visualization



Methods

The first method used to predict quality was a k-nearest neighbor (KNN) classifier. The way a KNN classifier works is by taking the k-nearest observations to the one we try to predict and using the mode of the k-nearest neighbors as the predictor. For this model, the two hyperparameters are the number of neighbors (k) and the distance metric used. For distance metrics, we tested Euclidean, cosine, Manhattan, and Chebyshev distances. Euclidean distance is the classic distance metric that we think of in the real world where it is the square root of the squares of the differences in each dimension. Cosine distance uses a dot product to measure the similarity of two points. Manhattan distance adds the distance in each individual feature, similar to finding the distance between two points in Manhattan using blocks. Finally, Chebyshev distance calculates the maximum difference between the two observations for all features and returns the maximum. When running KNN with many dimensions, models sometimes run into what is known as the curse of dimensionality, where finding the k nearest neighbors requires searching almost the entire input space. Thus, it makes sense to run KNN with fewer dimensions to avoid this problem. The best way to do this is by using principal component analysis (PCA), which creates a new low-dimensional basis such that each vector captures the largest amount of variance in the data. Computers can find these vectors quite quickly by calculating eigenvectors. In this research, we conducted the k-nearest neighbors model regularly as well as with PCA.

Additionally, we used an artificial neural network (multi-layer perceptron) for classification. The artificial neural network we used had 3 layers: an input layer, a hidden layer, and an output layer. An artificial neural network works by inputting the data for a specific observation into the input layer. The network uses a series of dot products and non-linear activation functions to compute output nodes for each possible quality, where the largest is the predicted quality. To improve the weights, the model performs back-propagation, which uses the gradient for each weight to minimize the error term for each output node. This process repeats until either the model converges or the training reaches the maximum number of iterations, which is 1000 in this case. Artificial neural networks have universal expressive power, which means that any mapping can be approximated with a 3-layer artificial neural network (which we use here since there is only 1 hidden layer) with enough nodes in the hidden layer. However, too many nodes in the hidden layer can overfit the model to the training data which is why we tune the number of hidden nodes using a grid search with cross-fold validation to balance the tradeoff and find the optimal number. One way to potentially improve an artificial neural network is by scaling each feature. This works because differences in the scales of features can create large weights and an unstable learning process. After creating a regular ANN model, we used statistical normalization, converting each feature's mean to 0 and variance to 1 to remove issues that arise from scaling differences. This normalization works by subtracting the feature mean from each feature and then dividing it by the feature standard deviation. Using these new statistically normalized values, the new ANN model is hopefully more consistent and effective.

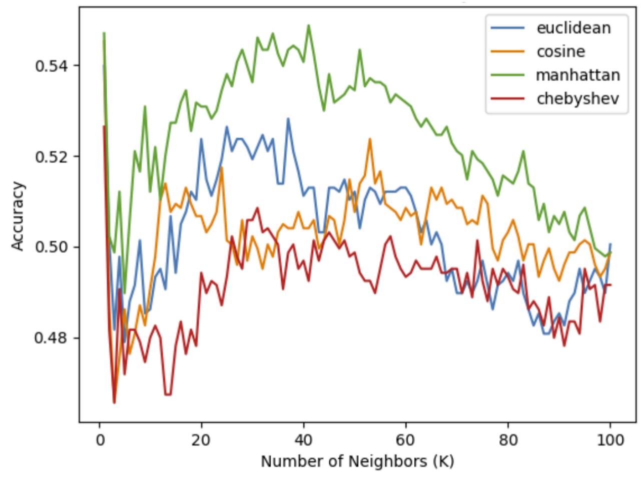
Finally, we used a random forest classifier, which uses a series of decision trees and takes the mode of their results. The two main hyperparameters in a random forest are the number of estimators/trees and the maximum depth permitted in each tree. Each decision tree is given a subset of features to choose from, and they choose the feature that maximizes the information gain, which essentially measures how well a feature distinguishes between different classes. This continues until either all training data is properly classified, there is no more data to split (the model will not split two observations), or the maximum depth of the tree is reached. To predict an observation's class, the random forest inputs the observation into each tree and calculates the mode prediction.

Results and Analysis

KNN:

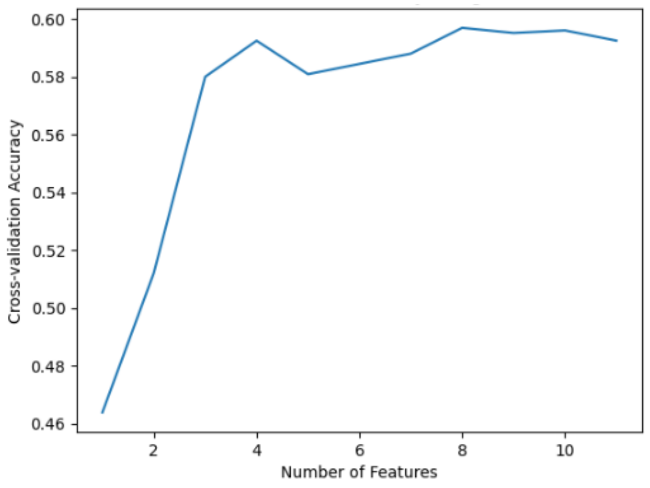
After running hyperparameter tuning using 5-fold cross-validation on the k-nearest neighbors model, the highest validation accuracy was with the Manhattan distance and 41 neighbors as shown in Figure 4 below. When testing this model on the test data, it had an accuracy of 0.55.

Figure 4: KNN Validation



We also used PCA to find a new KNN model. We created 11 data sets containing 1 to 11 PCA dimensions. We then use the same cross-validation approach to tune the hyperparameters to find the optimal number of neighbors and distance metric for each of the datasets. After finding the optimal hyperparameter for each number of PCA dimensions, we find the validation accuracy of each to determine the optimal model overall, which used 8 dimensions, cosine distance, and 49 neighbors. This model had a test accuracy of 0.59, a clear improvement over the initial KNN model.

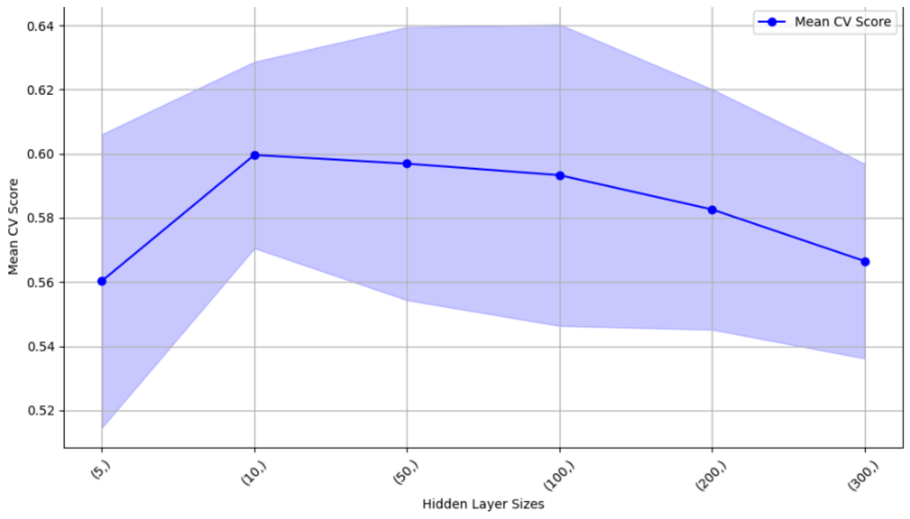
Figure 5: KNN PCA Validation



ANN:

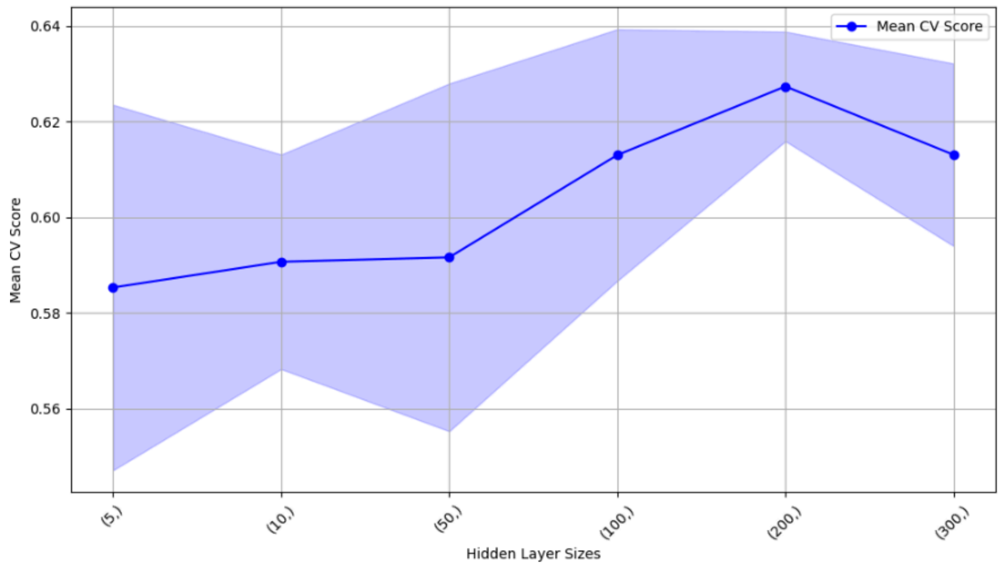
For the artificial neural network, not all possible hidden layer sizes were tested due to the exhaustive nature of training an ANN. Thus, we tested sizes of 5, 10, 50, 100, 200, and 300. Using 5-fold cross-validation on the training data, we found that the optimal hidden layer size was 10, with a mean cross-validation accuracy of 0.60. The model had a test accuracy of 0.57.

Figure 6: ANN Validation



We attempted to improve the ANN model by performing statistical normalization and running 5-fold cross-validation, which found a new optimal hidden layer size of 200 nodes. This model had both a cross-validation accuracy and test accuracy of <u>0.63</u>, which is a substantial improvement from the ANN model without normalization.

Figure 7: Normalized ANN Validation



Random Forest:

The final model used was the random forest model. For this model, we used 5-fold cross-validation to tune the number of estimators as well as the maximum depth for each estimator. However, not all possible depths and estimators were tested due to the exhaustive computation required to train and validate an entire random forest. Thus, we used a grid search to find an optimal value for the two hyperparameters. After cross-validation, the optimal model was found to use 200 estimators with a maximum depth of 20, which had a cross-validation score of 0.67. Additionally, when testing this model on the test data, it had a test set accuracy of <u>0.68</u>, the highest of any model. Its confusion matrix is visible in Table 2 below.

Table 2: Random Forest Confusion Matrix

	Predicted: 3	Predicted: 4	Predicted: 5	Predicted: 6	Predicted: 7	Predicted: 8
Actual: 3	0	1	1	1	0	0
Actual: 4	0	0	8	3	0	0
Actual: 5	0	0	166	41	0	0
Actual: 6	0	1	48	134	4	0
Actual: 7	0	0	1	38	26	0

Actual: 8	0	0	0	4	2	1

For our random forest model, we wanted to know the strongest feature in the prediction of wine quality. Feature importance measures how much the performance of the model decreases without its inclusion in the model. Alcohol is the most important feature in predicting wine quality in our dataset, as it has the highest feature strength of 15.1%. After alcohol, volatile acidity, and total sulfur dioxide are the most important features as they all have feature strengths of slightly over 10%. The remaining features all have strengths between 6 and 9%.

Table 3: Individual Feature Importance

Feature	**Importance**	**Rank**
Fixed Acidity	0.0760	8
Volatile Acidity	0.1068	2
Citric Acid	0.0769	7
Residual Sugar	0.0672	11
Chlorides	0.0770	6
Free Sulfur Dioxide	0.0682	10
Total Sulfur Dioxide	0.1036	4
Density	0.0937	5
pH	0.0741	9
Sulphates	0.1056	3
Alcohol	0.1510	1

Conclusion

The K-Nearest Neighbor model achieved a classification accuracy of 55% using the Manhattan distance metric and checking the 41 nearest neighbors. Using PCA to reduce the number of dimensions to 8, KNN improved its classification accuracy to 59% using cosine distance and 49 neighbors. The artificial neural network model with 10 hidden nodes had a classification accuracy of 57%, and using statistical normalization and 200 hidden nodes improved the model's accuracy to 63%. The best classification model for predicting wine quality is the **Random Forest Method** with a classification accuracy of **68%** on the test. It should be noted that a classification accuracy of 68% is still imperfect, as the model incorrectly classifies wine quality nearly one-third of the time. Other classification models should be explored such as the support vector method in an attempt to decrease the model error rate in future research.