Classification of Red Wine Quality Data

Julie Baguio
McKelvey School of Engineering
Washington University in St. Louis
Saint Louis, MO
baguio@wustl.edu

Kanan Ahmadov
McKelvey School of Engineering
Washington University in St. Louis
Saint Louis, MO
a.kanan@wustl.edu

Nathan Zhou
McKelvey School of Engineering
Washington University in St. Louis
Saint Louis, MO
nathan.j.zhou@wustl.edu

***Abstract*—This project focuses on performing a classification task on the Red Wine Quality dataset from the UCI Machine Learning Repository. To achieve this, three machine learning models were implemented: Support Vector Machine (SVM), Artificial Neural Network (ANN), and K-Nearest Neighbors (KNN). The project began with exploratory data analysis and feature visualization to understand the dataset's characteristics, followed by standardizing the features to ensure consistent scaling. Hyperparameter tuning was performed for all models to optimize their performance. Finally, the models were evaluated using accuracy, precision, recall, and F1-score.**

The ANN model demonstrated the best overall performance, achieving an accuracy of 66% and higher weighted average scores across all metrics due to its ability to capture complex relationships within the data. The KNN model followed with an accuracy of 62%, benefiting from its proximity-based approach but suffering in high-dimensional space. The SVM model, while achieving an accuracy of 61%, struggled the most with class imbalance, particularly for minority classes.

I. Introduction

Machine learning is an application of artificial intelligence that allows systems to learn and improve from experience without being explicitly programmed. Through mathematical models based on a training dataset, machine learning transforms existing data into a model that can interpret new data. This can then be used for many purposes, such as pattern detection, regression, or classification. In this project, machine learning techniques were applied to a classification problem: predicting the quality of red wine based on its attributes.

The data set used is the Red Wine Quality dataset, sourced from the UCI Machine Learning Repository. It contains 1,599 samples, with each sample representing a wine characterized by 11 physicochemical features, including acidity, alcohol content, and chlorides. The target variable, quality, is a discrete value ranging from 3 to 8, corresponding to expert evaluations of wine quality.

The primary goal of this project was to create models that would accurately predict quality given a set of attributes. To accomplish this, we used three classification methods: Support Vector Machine (SVM), Artificial Neural Networks (ANN), and K-Nearest Neighbors (KNN). The project involved several critical steps. We first analyzed and visualized the data to understand the relationships between the attributes. This allowed us to implement feature selection and understand the distribution of the features and quality. The data set was then split into a training and testing dataset and then preprocessed through standardization to ensure consistent feature scales, before being trained in the models.

For each model, the features were treated as labeled points in a higher dimensional space. The SVM model is a supervised model which uses linear or nonlinear hyperplanes to separate different qualities. Our model does this in a one vs one manner, creating an optimal hyperplane for each pair of qualities. The ANN model employed a multi-layered structure of hidden layers with backpropogation to create a model. KNN relied on proximity-based predictions, classifying points by points nearest to it on some metric. The models were evaluated and compared using various metrics, including accuracy, precision, recall, F1-score, and confusion matrices. Additionally, hyperparameters were tuned to optimize performance, such as SVM's C parameter, ANN's activation functions and layers, and KNN's number of neighbors and distance metrics.

This project helped us to identify and analyze the unique strengths and limitations of each model: SVM demonstrated robustness in managing high-dimensional data, ANN effectively captured complex relationships among features, and KNN performed well in scenarios with clear local patterns.

II. Exploratory Analysis

The exploratory analysis step provides a detailed understanding of the dataset, focusing on the characteristics of the features, their relationships, and their impact on our target value wine quality. This step forms the foundation for building effective machine learning models.

To start, we interpret what each attribute means. This is expressed in Table I.

Attribute	Interpretation
Fixed Acidity (g/L)	Fixed acidity of wine
Volatile Acidity (g/L)	Volatile acidity of wine
Citric Acid (g/L)	Citric acid concentration
Residual sugar (g/L)	Residual sugar concentration
Chlorides (g/L)	Chloride concentration
Free Sulfur Dioxide (mg/L)	Free sulfur dioxide concentration
Total Sulfur Dioxide (mg/L)	Total sulfur dioxide concentration
Density (g/mL)	Wine density
pH (1-14)	Acidity level of wine
Sulfates (g/L)	Sulfate concentration
Alcohol (%)	Alcohol amount of wine
Quality (1-10)	Output variable

TABLE I: Table of features and their description

As a preprocessing step, feature selection was performed using the scikit-learn feature selection class. Data from features with an information gain less than 0.05 (see Fig. 1) were eliminated from the study as they have minimal ability to

predict the target: quality. This step helped prevent our models from learning irrelevant patterns and thus over-fitting as well as reduce dimensionality to lower computation time.

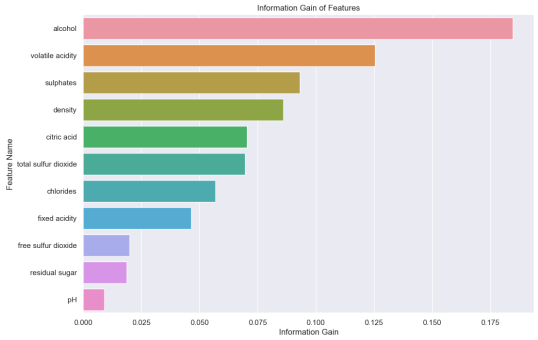

Fig. 1: Information Gain (IG) of Features. Features with IG $<$ 0.05 were eliminated: fixed acidity, free sulfur dioxide, residual sugar, and pH.

As mentioned in Introduction, the dataset comprises 1,599 samples, each characterized by 11 physicochemical attributes such as acidity, alcohol content, and residual sugar. The target variable, wine quality, ranges discretely from 3 to 8. As depicted in Fig. 2, the histogram of wine quality distribution highlights a class imbalance, with most wines rated as 5 or 6. This imbalance poses challenges for ensuring balanced predictions across all quality classes, as some metrics for scoring, such as accuracy would give values that do not reflect the actual performance of the model well.

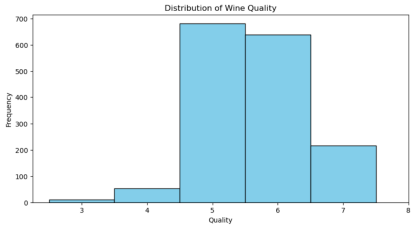

Fig. 2: Histogram of Wine Quality Distribution. The chart highlights the class imbalance in the dataset, with most wines receiving ratings of 5 or 6.

To explore feature inter-dependencies more, a correlation matrix was generated, as depicted in Fig. 3. The heatmap reveals strong correlations between some features, such as total sulfur dioxide, while most attributes have weak correlations with the target variable. This highlights the dataset's complexity and the need for machine learning algorithms to capture non-linear relationships. It also supports our feature selection decisions, as the features we removed all have very low correlation with our target variable "quality".

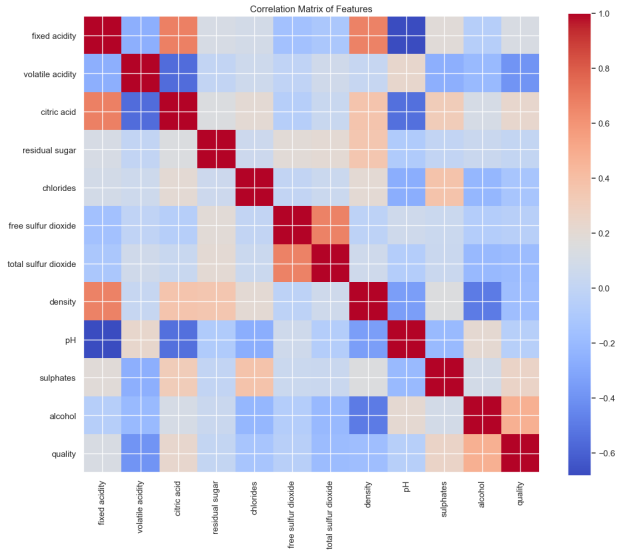

Fig. 3: Correlation Heatmap. The matrix showcases inter-dependencies between features and the target variable.

To further understand the dataset, histograms were created for each feature selected through information gain that we will use to train our models, as shown in Fig. 4. These histograms reveal variations in feature distributions, such as alcohol showing a nearly normal distribution, while others, like total sulfur dioxide and chlorides, are skewed. In response to this, we used techniques such as standardization after the Train-Test split. It is vital that we fit the standardization only to the training data, as otherwise we would have data leakage from the training dataset to the testing dataset.

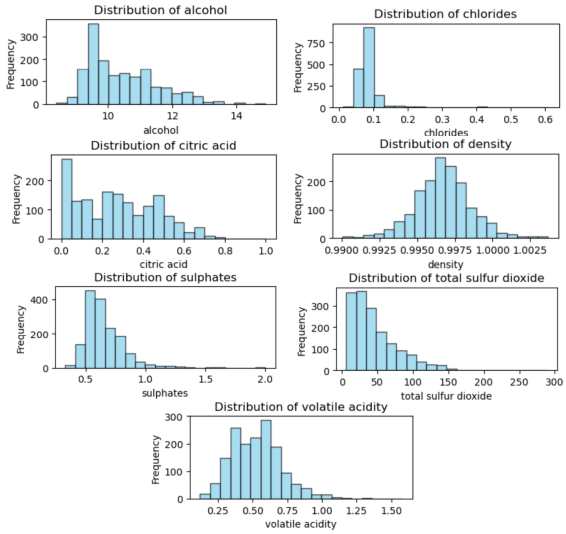

Fig. 4: Feature Distributions. The histograms display the variability in the distributions of the physicochemical features.

Furthermore, scatter-plots, shown in Fig. 5, were used to analyze the relationships between individual features and the target variable. These plots highlight trends such as a positive correlation between alcohol content and wine quality, while attributes like residual sugar show weaker relationships. This analysis provides initial insights into which features might

be more influential in predicting wine quality. The x-axis of the histograms and scatter-plots also show variance within the scales of the features. This can cause issues for some models like KNN, so for those we used scikit-learn's MinMaxScaler to scale all to the same range.

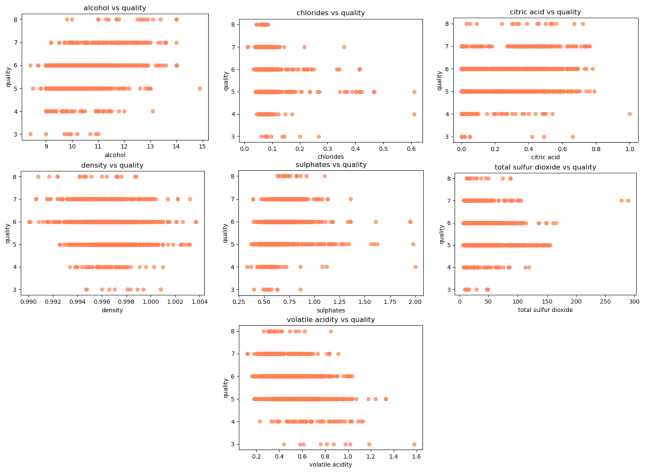

Fig. 5: Feature vs. Quality Scatterplots. These plots illustrate the relationships between physicochemical features and wine quality ratings.

Finally, Principal Component Analysis (PCA) was applied to reduce the dataset's dimensionality, offering a simplified view of feature interactions. The PCA scatter plot, shown in Fig. 6, illustrates the wine quality classes in reduced dimensions. Overlapping is evident for adjacent quality classes, which illustrates the that it is likely our models will not have very high scoring metrics, but still some patterns can be seen, indicating that classification is still feasible.

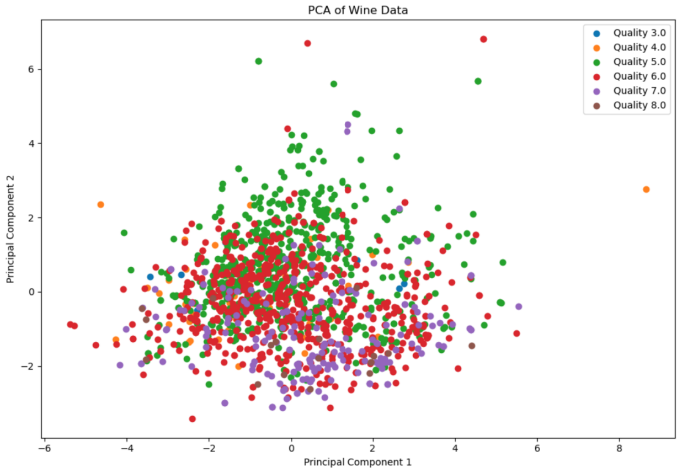

Fig. 6: PCA Scatter Plot. This visualization highlights the separability of wine quality classes in reduced dimensions.

In summary, the exploratory analysis revealed significant variability and complexity within the dataset. Insights from feature distributions, relationships, and dimensionality reduction emphasize the importance of preprocessing and inform the selection and optimization of classification models.

III. Methods

The machine learning models we used for classification in this project are Support Vector Machine (SVM), Artificial Neural Networks (ANN), and K-Nearest Neighbors (KNN). Each model was implemented with an emphasis on optimizing performance through preprocessing, hyperparameter tuning, and thorough evaluation. To tune the hyperparameters for SVM and ANN, scikit-learn's GridSearchCV was used. To combat the data imbalance, the F1-score was used as the scoring metric as opposed to the accuracy. The F1-score is the harmonic mean of precision and recall, and it considers both false negatives and false positives, making it better for this dataset.

A. *SVM*

The Support Vector Machine was implemented as one of the primary classification models in this project. The SVM algorithm aimed to find the optimal hyperplane that separates data points of different classes. Because there are more than two qualities, we used a one vs one scheme. A hyperplane is created for each pair of qualities, and the quality that is chosen the most is what a point is assigned to. To maximize its predictive accuracy, several steps were undertaken, including hyperparameter tuning, visualization of the model's behavior, and evaluation of its performance.

First, the SVM model was trained using the scikit-learn SVC class, with hyperparameter tuning conducted through grid search using GridSearchCV. To counteract the class imbalance in Quality, we chose to use the F1-score to find the optimal hyperparameters, as it works better than accuracy in this scenario. The hyperparameter grid included various values for the regularization parameter C ([0.1, 1, 10, 50, 100]), different kernel functions (linear, radial basis function [rbf], sigmoid, and polynomial), and gamma settings (scale and auto). C controls how lenient the model is to outliers. A higher C means less tolerance outliers, reducing margins and increasing model complexity. The different kernels allow for nonlinear hyperplanes to be used to separate qualities. Gamma affects the coefficient with rbf and sigmoid kernel affecting the curvature. A 5-fold cross-validation was used during the grid search to identify the best combination of parameters. Once the grid search was complete, the model with the optimal parameters was trained on the training data and used to make predictions on the test set.

To evaluate the SVM's performance, a confusion matrix was plotted (Fig. 7). This visualization provides insight into the model's strengths and weaknesses in classifying specific categories, showing how well the model performed in correctly classifying wine quality classes. Additionally, a classification report was generated, which included metrics such as accuracy, precision, recall, and F1-score, offering a comprehensive evaluation of the model's performance.

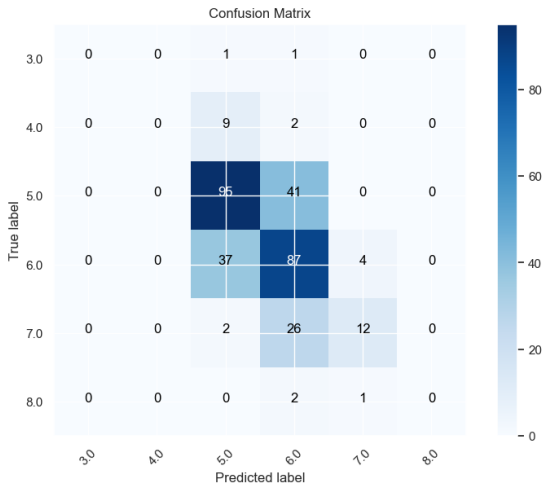

Fig. 7: Confusion Matrix for SVM. This matrix illustrates the classification performance of the SVM model across wine quality classes.

To further analyze the SVM model, the impact of the C parameter on accuracy was studied. The C parameter controls the trade-off between achieving a low error on the training set and maintaining a large margin that separates classes. A series of SVM models were trained with different C values, and their corresponding accuracies on the test set were plotted (Fig. 8). As C increases, so does the accuracy, but in the range from 1 to 100 it is constant, and past this range we run into issues with the F1-score which prevented higher C's from being optimal. This analysis highlighted the sensitivity of the model's performance to the choice of the regularization parameter, aiding in understanding the balance between underfitting and overfitting.

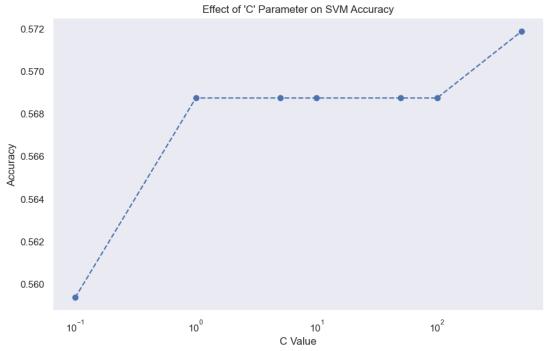

Fig. 8: Effect of C Parameter on SVM Accuracy. This plot shows how varying the regularization parameter C affects the model's accuracy on the test set.

B. ANN

Artificial Neural Networks were implemented as another key classification model in this project. The ANN model is inspired by biological neural networks. The model we used is a multi-layer perceptron. It has an input layer, one or two hidden layers, and an output layer. Each layer is fully connected to the next layer. The neurons each have an activation function, which introduces non-linearity by mapping the linear input through a potentially nonlinear smooth function. After a round of training, it updates the weights through backpropogation. The implementation focused on identifying the optimal hyperparameters for training the MLP and evaluating its performance.

To optimize the performance of the ANN, a hyperparameter tuning process was conducted using GridSearchCV. To counteract the class imbalance in Quality, we chose to use the F1-score to find the optimal hyperparameters, as it works better than accuracy in this scenario. One hyperparameter tuned was the activation function, which can be identity ($f(x) = x$,) logistic ($f(x) = \frac{1}{1+e^{-x}}$), tanh ($f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$), and ReLU ($f(x) = \max(0, x)$. The non-identity functions introduce non-linearity into the model allowing for a more complex model. Another hyperparameter tuned is the number of the hidden layers and the size of each. For one hidden layer we tested 50, 100, and 200 neurons. For two hidden layers we tried 100 neurons feeding to 50 neurons, 200 feeding into 100, and 200 feeding into 50. More layers and neurons allow for a more complex model but risk overfitting. Finally, the regularization term strength α is tuned. This term helps to prevent overfitting by penalizing large coefficients in the model but can lead to underfitting. 5-fold cross-validation was used to find the parameters with the highest validation score. Once the grid search was complete, the model with the optimal parameters was trained on the training data and used to make predictions on the test set.

To evaluate the ANN's performance, a confusion matrix was plotted (Fig. 9). Additionally, a classification report was generated, which included metrics such as accuracy, precision, recall, and F1-score, offering a comprehensive evaluation of the model's performance, which will be interpreted in the Results section.

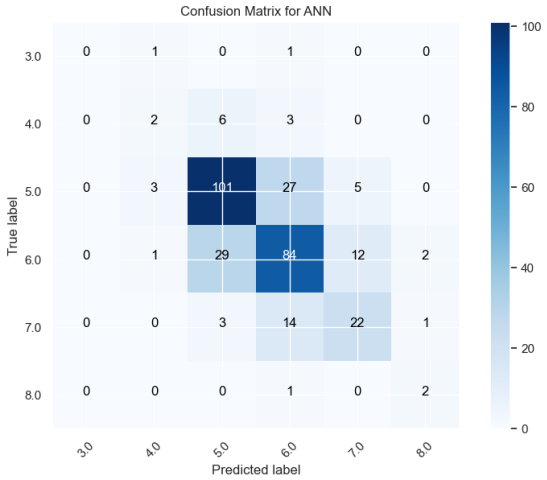

Fig. 9: Confusion Matrix for ANN. This matrix illustrates the classification performance of the ANN model across wine quality classes.

C. KNN

The final classification model used in this project was the K-Nearest Neighbors Algorithm. The KNN Algorithm works

under the assumption that data points are clustered together by class in the feature space. It predicts the class of a given data point to be the mode of the class of the K nearest neighbors based on a distance metric.

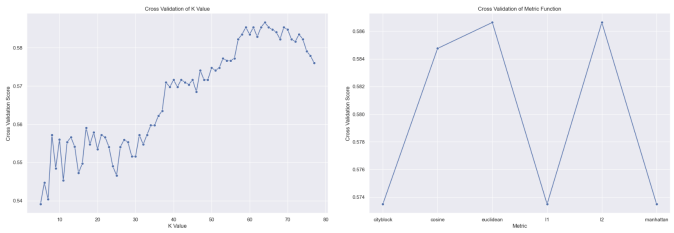

Fig. 10: A K value of 64 and the euclidean metric function had the highest cross validation scores.

As a preprocessing step, features were scaled to a range of (0,1) using the MinMaxScaler. This was done to make the impact of feature values uniform, preventing features with large ranges from skewing results in the distance function. The model used in this project was the KNeighborsClassifier sourced from the scikit-learn neighbors class. To select the most optimal settings for the model, a 5-fold cross-validation was performed with various K values (5, 6, ..., 78) and distance metrics (cityblock, euclidean, cosine, l1, l2, and manhattan). We then trained the model with the strongest performing values for parameters; the class of each point in the test set was determined to be the class mode of the 64 nearest data points in the training set based on the euclidean distance metric. Our KNN model was then evaluated based on accuracy and by a confusion matrix.

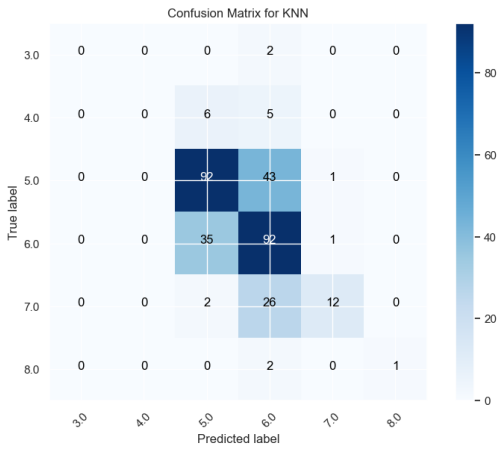

Fig. 11: KNN Confusion Matrix. Samples with quality 5 and 6 were reliably predicted by our model.

IV. Results and Analysis

The results of the three classification models— SVM, ANN, and KNN—are evaluated based on performance metrics such as accuracy, precision, recall, and F1-score. These metrics provide a comprehensive understanding of how effectively each model classified wine quality, considering the inherent class imbalance in the dataset.

The SVM model achieved an overall accuracy of 61%. The best hyperparameters for the SVM were C = 1, kernel = rbf, and gamma = auto. The classification report for SVM shows weighted averages for precision, recall, and F1-score of 0.59, 0.61, and 0.58, respectively. However, the SVM struggled to accurately predict minority classes such as quality ratings of 0, 1, and 5, as evident from their poor precision and recall scores. The model performed better on majority classes, particularly for quality ratings of 5 and 6, indicating that SVM is robust in handling high-dimensional data but is also affected by class imbalance in the dataset.

Classification Report for SVM:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	2
1	0.00	0.00	0.00	11
2	0.66	0.70	0.68	136
3	0.55	0.68	0.61	128
4	0.71	0.30	0.42	40
5	0.00	0.00	0.00	3
accuracy			0.61	320
macro avg	0.32	0.28	0.28	320
weighted avg	0.59	0.61	0.58	320

Fig. 12: SVM Classification Report. The trained model had an accuracy of 0.61.

The ANN model achieved an overall accuracy of 66%, with optimal hyperparameters including a tanh activation function, two hidden layers of size 200 and 100, and a regularization constant of $1 * 10^{-5}$. In the end,. the most complex model did not end up overfitting. The classification report for ANN indicates weighted averages for precision, recall, and F1-score of 0.65, 0.66, and 0.65. The ANN model performed comparably to the SVM but exhibited slightly higher recall for minority classes. The model took a longer time to train compared to the other models. This model still struggled to accurately predict minority classes such as quality ratings of 0, 1, and 5.

Classification Report for ANN:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	2
1	0.29	0.18	0.22	11
2	0.73	0.74	0.73	136
3	0.65	0.66	0.65	128
4	0.56	0.55	0.56	40
5	0.40	0.67	0.50	3
accuracy			0.66	320
macro avg	0.44	0.47	0.44	320
weighted avg	0.65	0.66	0.65	320

Fig. 13: ANN Classification Report. The trained model had an accuracy of 0.62.

The KNN model achieved an overall accuracy of 62%, making it the second best performing model after ANN. The best parameters for KNN were 64 neighbors and the euclidean

distance metric. The classification report for KNN shows weighted averages for precision, recall, and F1-score of 0.62, 0.62, and 0.60, respectively. The model struggled to predict the minority classes accurately, reflecting the challenges posed by the dataset's imbalance; its reliance on local density and proximity for predictions allowed it to perform better on classes with a larger sample size. The KNN algorithm may not have performed as well as the ANN due to the "curse of dimensionality". With the dimension of our feature space being high at seven, the variation in just one feature's value is not as impactful to the location of a data point. Thus, simply using euclidean distance may not capture complex models as well as an ANN algorithm which trains with multiple layers and varying weights to capture the impact of feature values.

Classification Report for KNN:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	2
1	0.00	0.00	0.00	11
2	0.68	0.68	0.68	136
3	0.54	0.72	0.62	128
4	0.86	0.30	0.44	40
5	1.00	0.33	0.50	3
accuracy			0.62	320
macro avg	0.51	0.34	0.37	320
weighted avg	0.62	0.62	0.60	320

Fig. 14: KNN Classification Report. The trained model had an accuracy of 0.68.

The results are summarized in Table II. As we can see, the ANN model performed best with regard to all metrics. The difference in the weighted and macro (unweighted) average metrics shows how all models struggled to classify the classes of smaller sizes. This is further reinforced by accuracy being the highest score for each model. The SVM model notably struggled the most in this regard, with a much lower F1-score, Recall, and Precision compared to the KNN model despite very similar accuracies.

		Weighted Average			**Macro Average**		
	Accuracy	F1-score	Recall	Precision	F1-score	Recall	Precision
SVM	0.61	0.58	0.61	0.59	0.28	0.28	0.32
ANN	0.66	0.65	0.66	0.65	0.44	0.47	0.44
KNN	0.62	0.60	0.62	0.62	0.37	0.34	0.51

TABLE II: Table of accuracy, weighted average scores, and macro average scores for all models

In terms of understanding the results presented in Table II, the differences in performance can be attributed to the inherent characteristics of each model and their interaction with the dataset. ANN's multilayered architecture and optimal hyperparameters allowed it to effectively capture complex relationships in the data, leading to better generalization. KNN had a proximity-based approach but still suffered due to the dimensionality in a high-feature space. SVM struggled the most, particularly with minority classes, as its reliance on maximizing margins made it more sensitive to class imbalance.

In conclusion, while ANN outperformed SVM and KNN in terms of overall accuracy and other performance metrics, all three models faced challenges in predicting minority classes. The results highlight the importance of preprocessing, hyperparameter tuning, and the need to address class imbalance to improve predictive performance for rare classes.

V. Conclusion

In this project, we successfully applied three machine learning classification algorithms—SVM, ANN, and KNN—to predict the quality of red wine based on physicochemical attributes. The project explored various stages, from data preprocessing and exploratory analysis to model implementation, hyperparameter tuning, and performance evaluation.

The SVM model demonstrated robustness and flexibility, achieving an accuracy of 61% with the optimal parameters. Its ability to handle high-dimensional data and its sensitivity to the regularization parameter C provided valuable insights into the trade-offs between overfitting and underfitting. The ANN model, while most accurate at 66%, captured complex relationships between features and highlighted the importance of tuning hyperparameters such as activation functions, layer numbers, and regularization coefficients. The KNN model achieved an accuracy of 62%, showcasing its effectiveness in leveraging local patterns and the significance of selecting an appropriate number of neighbors and distance metric.

Throughout the analysis, metrics such as accuracy, precision, recall, F1-score, and confusion matrices were used to evaluate the strengths and weaknesses of each model. The class imbalance in the dataset posed a challenge, as evident from the varying precision and recall values across wine quality classes. Despite these challenges, the models provided a reasonably effective prediction of wine quality, with each algorithm offering unique strengths based on the characteristics of the dataset.

Furthermore, the exploratory analysis and feature visualizations, including histograms, scatter plots, and correlation heatmaps, were instrumental in understanding the dataset's structure and guiding the preprocessing and model selection steps. Principal Component Analysis further highlighted the complexity of the classification task, with overlapping regions among adjacent quality classes.

Ultimately this project provided an opportunity to explore machine learning's practical applications in a real-world problem. The analysis of SVM, ANN, and KNN demonstrated the trade-offs between different algorithms and emphasized the importance of data preprocessing and hyperparameter tuning.

VI. Member Contributions

Julie Baguio: Implemented KNN model, feature selection, wrote own section descriptions, updating and rewrites.

Kanan Ahmadov: Implemented SVM model, data visualization, wrote Introduction, Exploratory Analysis, Results, Conclusion sections, updating and rewrites.

Nathan Zhou: Implemented ANN model, data visualization and standardization, wrote Abstract, updating and rewrites.

VII. Appendix