Wine Quality Prediction Using SVM, Random Forest and XGBoost

Burcu Arslan
Ozyegin University
Istanbul, Turkey
burcu.arslan@ozu.edu.tr

Umut Oskay
Ozyegin University
Istanbul, Turkey
umut.oskay@ozu.edu.tr

***Abstract*—This study presents a comprehensive application of machine learning methods to predict wine quality. Wine quality, influenced by various factors like grape type, climate, and wine-making process, traditionally undergoes physicochemical and sensory evaluations. However, the subjective nature of sensory assessments complicates this process, prompting a need for more accurate, objective methods. Utilizing a dataset from Kaggle consisting of 6497 wine entries and 12 attributes, the study employs Support Vector Machines (SVMs), Random Forest, and XGBoost algorithms to forecast quality scores. Preprocessing techniques including Synthetic Minority Over-sampling Technique (SMOTE) and StandardScaler were employed to address data imbalance and scale data respectively. Results showed a promising ability of the mentioned models to predict wine quality with Random Forest yielding the highest accuracy. The paper aims to contribute to the wine industry by providing an effective, data-driven means to assess wine quality, potentially enhancing production decisions and market strategies.**

***Index Terms*—Classification, Wine Quality, Wine Quality Prediction, Support Vector Machine (SVM), Random Forest, XGBoost**

I. Introduction

Wine, a globally relished alcoholic drink, derives its quality from a multitude of variables. These include the grape variety used, the climatic conditions where it is grown, and the procedures implemented during its production. Wine quality evaluation is a complex procedure that traditionally combines physicochemical and sensory assessments. Physicochemical evaluations often involve measuring various characteristics such as density, pH, and alcohol content. Sensory testing, on the other hand, involves subjective evaluations from trained experts. The correlation between these two sets of tests is still not entirely understood, especially given the intricate nature of the sense of taste. This study proposes the application of machine learning methods to predict wine quality based on key attributes like pH, sulphates, citric acid, and alcohol content among others.

In this study, we will use support vector machines (SVMs), random forests, and XGBoost machine learning models to predict wine quality. Machine learning models called SVMs can be applied to classification or regression problems. Random forests are a type of ensemble learning model that combines multiple decision trees to make predictions. XGBoost is a type of gradient boosting machine that is known for its high performance.

II. Dataset Description and Preparation

A. Data Exploration

The dataset used in this project is the 'Wine_Quality_Data' dataset in Kaggle. The dataset itself consists of 6497 entries and 12 attributes, with a target variable which is the quality of the wine. These attributes include fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol, quality, and color. The 'quality' attribute ranges from 3 to 9, with 9 signifying the highest quality. Besides 'color', which is a categorical variable that either is "red" or "white", all other attributes are numerical. Given that there are no missing values, imputation techniques are not necessary in this case. For the categorical attribute, One-Hot-Encoding was applied.

The data is very imbalanced, with the most instances being on wine quality score 6 with 2836, and the least amount of instances being on wine quality score 9 with 5.

TABLE I
The Number of Instances for Each Wine Quality Score

Wine Quality Score	Number of Instances	Class Proportions (%)
3	30	0.46
4	216	3.32
5	2138	32.9
6	2836	43.7
7	1079	16.6
8	193	2.97
9	5	0.08

B. Data Preprocessing

As the dataset is very imbalanced, SMOTE (Synthetic Minority Over-sampling Technique) was used to combat the data imbalance problem. SMOTE creates synthetic data points that are similar to the minority class data points. It does this by selecting a minority sample and finding its k nearest neighbors in the feature space. In order to construct synthetic samples, one of the k closest neighbors is chosen at random, and a new sample is then created along the line segment linking the chosen neighbor and the original minority sample. This helps to balance the data and improve the performance of the machine learning models by removing the bias towards the

majority class by adding synthetic samples to minority class. [1]

After SMOTE, number of instances for each wine quality score was fixed to 2836.

StandardScaler is a data pre-processing method commonly employed in machine learning that normalizes the attributes of a dataset. This process adjusts each feature so that it has an average value of zero and a variance of one. This normalization technique can prove to be highly beneficial for various machine learning models, particularly those that are significantly influenced by the scale of input attributes. [2]

As the dataset variety of feature with different units and scales, such as chloride level and total sulphur dioxide, that have different units of measurements, using StandardScaler helps machine learning algorithms to learn the relationships between the features more effectively.

III. Methods

A. Software

Code is mainly written in Python. Python libraries that are used in this research are pandas [5], numpy [6], matplotlib [3], seaborn [4], scikit-learn [2], imblearn [7], xgboost [8] and warnings [9].

B. Models

1) Support Vector Machine (SVM): The objective of SVM is to find a hyperplane with the maximum margin of separation between the positive and negative data points [10]. The optimal boundary between various events is established by creating a hyperplane for each event or class based on these changes after the given data is transformed using the kernel technique.

SVM was implemented with the help of Scikit-learn library [2]. The split of training to testing data was with a ratio of 70/30. Two different kernels were used, 'rbf' was used for the optimal results and 'linear' was used to determine feature importance.

2) Random Forest: Random forest is an ensemble learning method that constructs a model by combining a large number of decision trees [14] [11]. A random subset of the features is used to construct each decision tree, and the predictions from all the trees are combined using majority voting [12]. Random Forest may be used to carry out a variety of tasks, including classification, regression, and ranking.

3) XGBoost: XGBoost (eXtreme Gradient Boosting) is an open-source software library for gradient boosting [8]. It is used for a various machine learning tasks, which includes regression, classification and ranking. XGBoost is known for its high performance and scalability [11]. It has been used to win many machine learning competitions, and it is widely used in industry [12].

XGBoost is based on the gradient boosting algorithm. Gradient boosting is a powerful technique for constructing a model that minimizes a loss function by combining weak predictive models such as decision trees into an ensemble. The model is constructed by iteratively adding new trees to the model. Each new tree is added in a way that reduces the loss function [8].

XGBoost is run with 'mlogloss' evaluation method, which stands for multiclass log loss metric, and is suitable for multiclass classification problem.

C. Algorithms

1) StandardScaler: StandardScaler is a data normalization method used to standardize the features of the data from python Scikit-learn library. It does so by removing the mean and scaling to unit variance [2].

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

2) SMOTE: SMOTE (Synthetic Minority Over-sampling Technique) is an algorithm used to address the class imbalance problem in machine learning. It aims to overcome the issue of imbalanced datasets where the number of samples in the minority class is significantly smaller than the majority class [1]. imblearn's SMOTE was used in the context of this project [7].

Algorithm 1 SMOTE: Synthetic Minority Over-sampling Technique

Require: Minority class samples, k (number of nearest neighbors), N (number of synthetic samples to generate)
Ensure: Synthetic samples
0: **foreach** minority sample x **do**
0: $neighbors \leftarrow$ Find k nearest neighbors of x from the minority class
0: $syntheticSamples \leftarrow$ empty list
0: **for** $i \leftarrow 1$ **to** N **do**
0: $neighbor \leftarrow$ Randomly select one of the k nearest neighbors
0: $newSample \leftarrow$ Interpolate between x and $neighbor$ by a random factor r where $0 \leq r \leq 1$
0: $syntheticSamples$.append($newSample$)
0: **end for**
0: **end foreach**
0: **return** Synthetic samples =0

Pseudocode referenced from 2002 paper, *SMOTE: synthetic minority over-sampling technique.* [1]

3) One-Hot Encoding: One-hot encoding is a technique used in machine learning to convert categorical data into a numerical format. This is done by creating a new binary feature for each unique category in the original feature. For the project, there is only one categorical feature, *'color'*, that represents the color of the wine which can be red or white. After One-Hot Encoding, there have been two new features created, which are color_red and color_white, making the total number of features 13. scikit-learn's OneHotEncoder was used in the context of this project [2].

IV. Evaluation

Each model was fitted with the training data and evaluated for its performance using both training and test data sets. Model performance was measured using the classification report, which provides an overview of the precision, recall, f1-score, and support for each class. This was followed by a confusion matrix that provides a visual depiction of the model performance.

The SVC model achieved an overall accuracy of 71.7% on the training set and 72.0% on the test set. The precision, recall, and F1-score for each class are summarized in the following tables:

TABLE II
Results of SVM for Training Set

Class	F1 Score	Precision	Recall
3	0.91	0.96	0.93
4	0.73	0.80	0.77
5	0.62	0.60	0.61
6	0.54	0.38	0.44
7	0.57	0.55	0.56
8	0.67	0.81	0.73
9	1.00	1.00	1.00

TABLE III
Results of SVM for Test Set

Class	F1 Score	Precision	Recall
3	0.92	0.96	0.94
4	0.72	0.82	0.76
5	0.63	0.58	0.60
6	0.49	0.36	0.42
7	0.56	0.53	0.55
8	0.65	0.78	0.71
9	0.99	1.00	1.00

The overall accuracy on the training set is 0.73, suggesting that the model correctly predicts the wine quality for approximately 73% of the samples.

The overall accuracy on the test set is 0.72, indicating that the model performs similarly on unseen data.

For SVM, the model performs the 'best' in class 9, with perfect 1 scores for precision, recall and F1 score both in training and test datasets. This could be potential overfitting, as class 9 was the minority class in the dataset before applying SMOTE.

The Random Forest Classifier achieved an overall accuracy of 77.5% on the training set and 77.0% on the test set. The precision, recall, and F1-score for each class are summarized in the following tables:

The precision, recall, and F1-scores for each class show relatively higher performance compared to the SVM model. The model achieves high scores for most classes, indicating its ability to capture the patterns in the training data. The overall accuracy on the training set is 0.84, suggesting that the model correctly predicts the wine quality for approximately 84% of the samples.

TABLE IV
Results of Random Forest Model for Training Set

Class	F1 Score	Precision	Recall
3	0.97	0.96	0.97
4	0.81	0.89	0.85
5	0.69	0.71	0.70
6	0.71	0.57	0.63
7	0.80	0.82	0.81
8	0.91	0.95	0.93
9	0.99	1.00	0.99

TABLE V
Results of Random Forest Model for Test Set

Class	F1 Score	Precision	Recall
3	0.97	0.95	0.96
4	0.74	0.84	0.79
5	0.61	0.63	0.62
6	0.55	0.39	0.46
7	0.70	0.72	0.71
8	0.81	0.90	0.86
9	0.98	1.00	0.99

For the testing set, the precision, recall and F-1 Scores are consistent with the training set results, as the values do not fluctuate that much, and the accuracy for the test set is 0.84.

The XGBoost Classifier achieved an overall accuracy of 83.4% on the training set and 83.0% on the test set. The precision, recall, and F1-score for each class are summarized in the following tables

TABLE VI
Results of XGBoost Model for Training Set

Class	F1 Score	Precision	Recall
3	0.98	1.00	0.99
4	0.92	0.98	0.95
5	0.85	0.82	0.84
6	0.86	0.76	0.81
7	0.89	0.92	0.90
8	0.94	0.98	0.96
9	1.00	1.00	1.00

TABLE VII
Results of XGBoost Model for Test Set

Class	F1 Score	Precision	Recall
3	0.97	0.99	0.98
4	0.87	0.94	0.90
5	0.71	0.67	0.69
6	0.65	0.56	0.60
7	0.74	0.78	0.76
8	0.86	0.92	0.89
9	1.00	1.00	1.00

For XGBoost, overall accuracy on the training set is 0.92,

indicating that the model correctly predicts the wine quality for approximately 92% of the samples, and for test dataset the accuracy is 83%, with a 10% drop.

The evaluation metric results for wine quality score 9 are similar to that of SVM, hinting at potential overfitting.

We can see that for all models, according to the number of features in the dataset before SMOTE, the classes that have more number of samples have a lower precision, recall and F1 score compared to the classes with less number of samples. This could be because of after SMOTE, the minority class could be represented, resulting in higher scores.

TABLE VIII
RESULTS FOR TEST SET OF CLASS 8

Model	Accuracy	F1 Score	Precision	Recall
SVM	0.71	0.71	0.67	0.76
Random Forest	0.76	0.87	0.86	0.89
XGBoost	0.83	0.91	0.89	0.93

For test set for class 8, the accuracy, F1 Score, Precision and Recall

TABLE IX
RESULTS FOR TEST SET OF CLASS 6

Model	Accuracy	F1 Score	Precision	Recall
SVM	0.71	0.42	0.49	0.36
Random Forest	0.77	0.46	0.55	0.39
XGBoost	0.83	0.46	0.55	0.39

TABLE X
RESULTS OF ACCURACY WITH CROSS-VALIDATION

Model	Cross-Validation Accuracy	Accuracy
SVM	0.57	0.71
Random Forest	0.61	0.77
XGBoost	0.65	0.83

1) Cross Validation: The SVM model achieved the smallest accuracy during cross-validation with 57%. The highest accuracy with cross-validation is of XGBoost with 65%, meaning that the model correctly predicted 65% of the samples across all of the cross-validation folds. A higher accuracy during cross-validation implies that the model can classify the instances more correctly.

A. Learning Curve

We plot learning curves to further determine if there is overfitting for each model.

For SVM, the training accuracy starts at approximately 0.68 and increases until 0.72, and validation accuracy also increases from 0.58 to 0.69. This shows that the model performs better with more training data added as the validation accuracy and training accuracy increases as training data increases. The model is performing reasonably according to the learning curve.

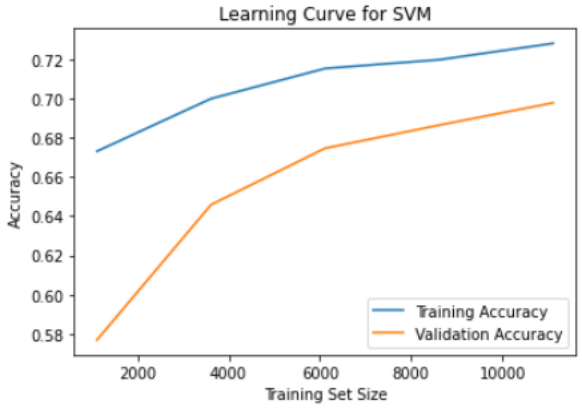

Fig. 1. Learning Curve for SVM

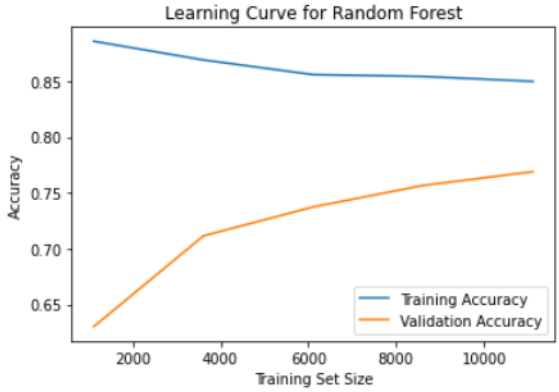

Fig. 2. Feature Importance for Random Forest

For the Random Forest model in the learning curve, training accuracy starts from 0.9 and decreases until 0.85, while the validation accuracy curve starts from 0.5 and increases until 0.75. This shows that the model's performance decreases as training data increases, implying that the model struggles to fit the data reasonably. We can say that the model fits better to unseen data, as validation accuracy increases with more training data. This contrast between the trainig accuracy and the validation accuracy could imply overfitting.

In the learning curve for XGBoost, training accuracy starts from 1.0 and decreases until 0.95, while validation accuracy curve starts from 0.65 and increases until 0.82. This shows that the model first performs very well on the training data, but then the performance decreases as training data increases, implying that the model struggles to fit the data reasonably. The validation accuracy increases nearly 20% as validation data increases, meaning that the model performs better with unseen data as the unseen data size increases. This could also imply potential overfitting, like the case with Random Forest model.

B. Feature Importance

In Feature Importance Graph from Random Forest Model chlorides and alcohol has higher score, color_red and

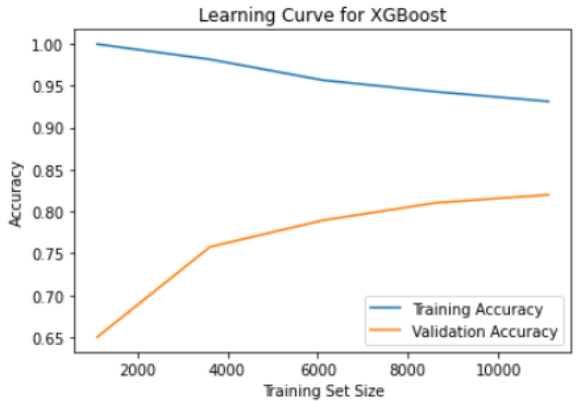

Fig. 3. Feature Importance for XGBoost

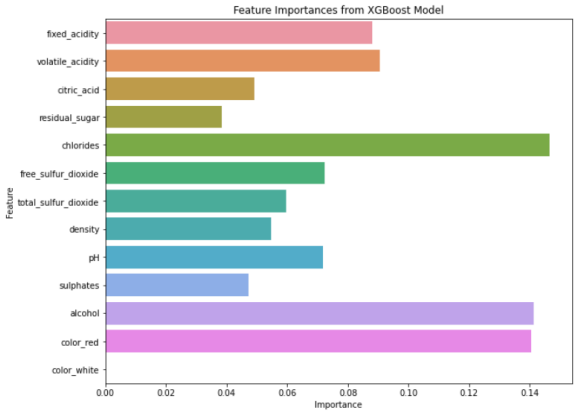

Fig. 5. Feature Importance for XGBoost

color_white has lower score; in Feature Importance Graph from XGBoost Model has color_red, alcohol and chlorides has higher score, color_white has lowest score.

For the Support Vector Machines Model we cannot draw Feature Importance Graph because when the kernel is not linear. When SVM employs non-linear kernels such as the polynomial or RBF, it shifts the attributes into a distinct, often higher-dimensional space. The outcome is a decision boundary that could be a sophisticated combination of all input attributes. Within this transformed dimension, it's not easy to pinpoint how the initial features influence the predictions.

Therefore, feature importance becomes difficult or even impossible to understand in many scenarios, because the modified features can't be directly traced back to the original dataset.

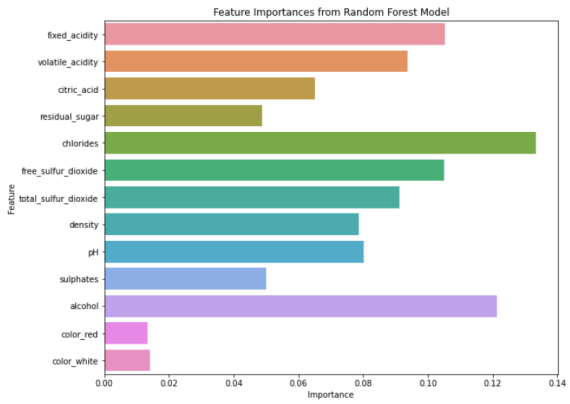

Fig. 4. Feature Importance for Random Forest

C. Conclusion

Overall, we observe that both the Random Forest and XGBoost models outperform the SVM model in terms of accuracy and F1-scores, they show better performance in predicting wine quality across different classes, but there could be potential overfitting in these models. Between Random Forest and XGBoost, the XGBoost model achieves a slightly higher accuracy on both the training and test sets. We also see signs of overfitting for class 9, and that could be due to class 9 being the minority class with just 5 samples before SMOTE.

D. Future Work

Methods such as Feature Selection, with the important features selected above and regularization of the hyperparameters using algorithms such as RandomSearchCV and GridSearchCV could be applied to reduce the risk of overfitting for minority classes. Regularization techniques also could be implied in the future to combat overfitting.

APPENDIX A
CONFUSION MATRICES

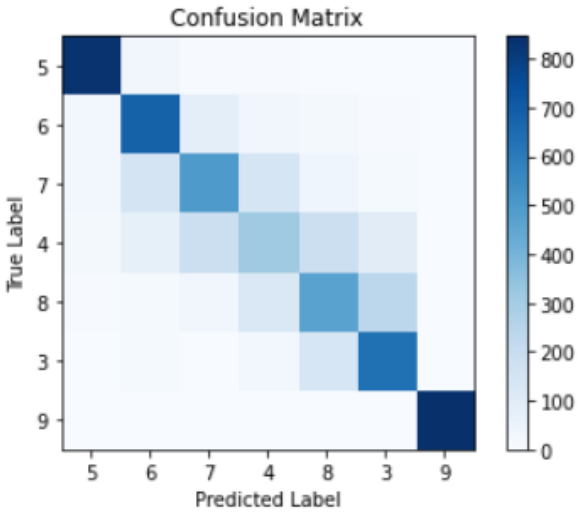

Fig. 6. Confusion Matrix for SVM

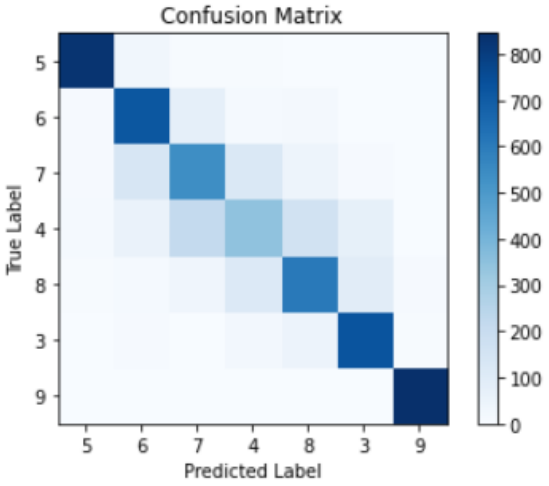

Fig. 7. Confusion Matrix for Random Forest

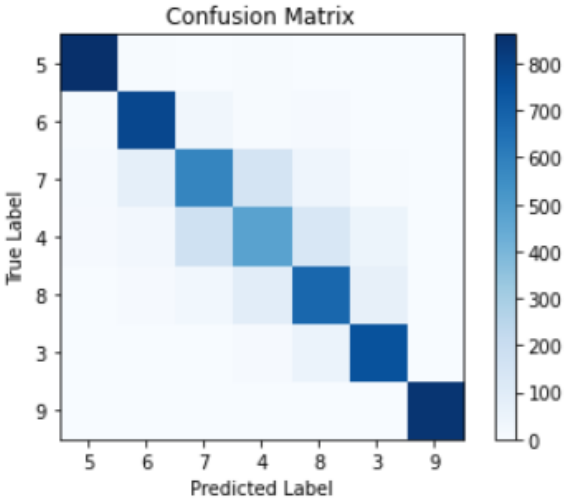

Fig. 8. Test Results for XGBoost

REFERENCES

[1] Chawla, N. V., Bowyer, K. W., Hall, L. O., Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 16, 321–357.

[2] Scikit-learn contributors. (2023). Scikit-learn: Machine Learning in Python. Retrieved from

[3] Hunter, J. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90-95.

[4] Waskom, M., & Seaborn, W. (2018). Seaborn: Statistical data visualization. Journal of Open Source Software, 3(33), 1050.

[5] McKinney, W. (2011). Data structures for statistical computing in Python. Proceedings of the 9th Python in Science Conference, 51–56.

[6] Oliphant, T. E. (2006). Guide to NumPy: Python for scientific computing. Trelgol Publishing.

[7] Gómez-Hernández, J. M., & García-Osorio, A. (2016). Imbalanced-learn: A Python toolbox to tackle class imbalance in machine learning. The Journal of Machine Learning Research, 17(1), 6559-6563.

[8] Chen, T., Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794). ACM.

[9] Patil, A. (2021). warnings: A Python module for managing warnings. Retrieved from https://docs.python.org/3/library/warnings.html

[10] Li, T., Ogihara, M., & Li, Q. (2003). A comparative study on content-based music genre classification. In Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval (pp. 282-289). ACM.

[11] Géron, A. (2017). Hands-on machine learning with Scikit-learn and TensorFlow. O'Reilly Media.

[12] Hastie, T., Tibshirani, R., Friedman, J. (2009). The elements of statistical learning. Springer.

[13] Chen, T., Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794). ACM.

[14] Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5-32.