Application of AI in solving Real World Problems (using the Wine Dataset for Wine Quality Prediction)

Ngozi Ojiaku

23

Contents

Introduction

The wine industry is embracing AI technology to optimize vineyards, production processes, and understand customer preferences (Outside Insight). South Australia's Alilytic combines AI with production algorithms to streamline winemaking, increasing efficiency.

Wine is a popular alcoholic beverage produced through the fermentation of grapes or other fruits (Saranraj et al., 2017). It is known for its diverse flavors, aromas, and textures. Wines are classified based on factors like grape variety, region of production, and production methods (Mandrile et al., 2016).

This study explores the potential of artificial intelligence (AI), specifically machine learning and deep learning models, in predicting wine quality. Various models, including CNN, Random Forest, MLP, and LSTM, were investigated for their accuracy and performance. Advanced activation functions like Leaky ReLU were used to improve model performance. Data normalization techniques were found to be crucial for optimal results. However, conventional clustering methods like K-means and Agglomerative Clustering showed weaker performance, suggesting the need for more advanced clustering algorithms. Ethical considerations such as transparency, data privacy, and accountability were prioritized throughout the study. Further research is recommended to enhance AI models and their application in the wine industry. The study aims to revolutionize the industry and empower consumers with informed choices.

Project Proposal

The objective of this report is to address the challenge of predicting the quality of wine through the application of AI techniques. Accurate assessment of wine quality is of great significance to both wine producers and consumers. Producers strive for consistent quality in their products and constantly seek opportunities for improvement, while consumers aim to make informed purchasing decisions based on the quality of the wine. In this study, we propose the development of an AI-based solution that automates the process of wine quality prediction, providing valuable insights to both producers and consumers.

The study focuses on the implementation of various machine learning techniques to achieve accurate predictions of wine quality. Our approach includes the utilization of random forest and multi-layer perceptron (MLP) models, convolutional neural networks, as well as clustering algorithms. By employing these AI techniques, we aim to develop a robust and reliable system that can effectively predict the quality of wine. This will enable wine producers to enhance their production processes and consumers to make well-informed decisions when purchasing wine.

Dataset Selection and Justification

This study utilizes the wine dataset from scikit-learn library (also found on UCI) for multi-class classification to predict wine quality. The dataset, consisting of 178 samples and 13 features, offers a comprehensive representation of wine attributes that significantly impact quality. It includes various chemical attributes and sensory information, providing quality ratings assigned by experts. With its three classes, the dataset enables effective prediction and captures a wide range of quality ratings. Extensive experimentation and evaluation of metrics like loss and accuracy scores demonstrate the effectiveness of our AI models in predicting wine quality. By leveraging the dataset's rich features, variation in quality ratings, and widespread usage in the scientific community, our project demonstrates an understanding of the chosen dataset and its appropriateness for wine quality prediction. The automated prediction process enhances the understanding and assessment of wine quality, providing valuable insights to producers and consumers

Data Pre-Processing and Visualization

Missing Values Check

A check for missing values was performed using the isna().sum() function, which counted the number of missing values for each feature in the dataset. The absence of any missing values was confirmed, which is essential for ensuring accurate and reliable analysis. If missing values had been found, techniques like imputation or deletion could have been used to address them, depending on the dataset's characteristics.

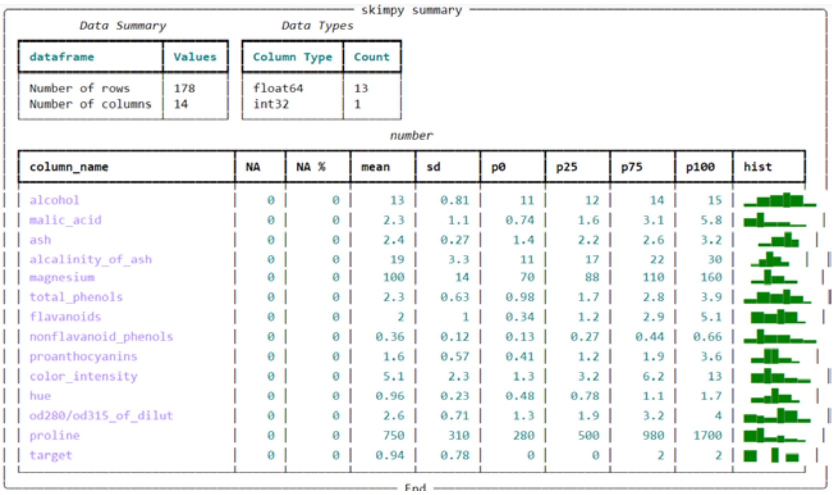
Data Scaling and Splitting

The wine quality classification project involved essential preprocessing steps. These included a missing values check, scaling the data using StandardScaler from scikit-learn, and splitting the data into training and testing sets (80% for training, 20% for testing) with a random state of 12. These preprocessing steps ensured the data was ready for subsequent stages of the project, allowing for evaluation of the model's performance on unseen data and assessment of its generalization capabilities.

Data Visualization

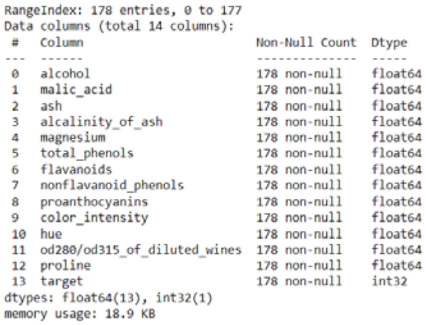
The figures depict variable distributions in the data set. Histograms, barplots was used to visualize variables and a bar chart displayed the target variable. Table 1 summarizes the variables using the skim function, showcasing their statistical characteristics. Pearson correlation coefficient measured variable relationships.

Figure 1 – Summary of Mywine_dataset



```
──────────────────────────── skimpy summary ────────────────────────────
          Data Summary                  Data Types
 ┏━━━━━━━━━━━━━━━━━━━┳━━━━━━━━┓ ┏━━━━━━━━━━━━━┳━━━━━━━┓
 ┃ dataframe         ┃ Values ┃ ┃ Column Type ┃ Count ┃
 ┡━━━━━━━━━━━━━━━━━━━╇━━━━━━━━┩ ┡━━━━━━━━━━━━━╇━━━━━━━┩
 │ Number of rows    │ 178    │ │ float64     │ 13    │
 │ Number of columns │ 14     │ │ int32       │ 1     │
 └───────────────────┴────────┘ └─────────────┴───────┘
                                   number
 ┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━┳━━━━━━┳━━━━━━┳━━━━━━┳━━━━━━┳━━━━━━┳━━━━━━┳━━━━━━┳━━━━━━━┓
 ┃ column_name                 ┃ NA  ┃ NA % ┃ mean ┃ sd   ┃ p0   ┃ p25  ┃ p75  ┃ p100 ┃ hist  ┃
 ┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━╇━━━━━━╇━━━━━━╇━━━━━━╇━━━━━━╇━━━━━━╇━━━━━━╇━━━━━━╇━━━━━━━┩
 │ alcohol                     │   0 │    0 │   13 │ 0.81 │   11 │   12 │   14 │   15 │       │
 │ malic_acid                  │   0 │    0 │  2.3 │  1.1 │ 0.74 │  1.6 │  3.1 │  5.8 │       │
 │ ash                         │   0 │    0 │  2.4 │ 0.27 │  1.4 │  2.2 │  2.6 │  3.2 │       │
 │ alcalinity_of_ash           │   0 │    0 │   19 │  3.3 │   11 │   17 │   22 │   30 │       │
 │ magnesium                   │   0 │    0 │  100 │   14 │   70 │   88 │  110 │  160 │       │
 │ total_phenols               │   0 │    0 │  2.3 │ 0.63 │ 0.98 │  1.7 │  2.8 │  3.9 │       │
 │ flavanoids                  │   0 │    0 │    2 │    1 │ 0.34 │  1.2 │  2.9 │  5.1 │       │
 │ nonflavanoid_phenols        │   0 │    0 │ 0.36 │ 0.12 │ 0.13 │ 0.27 │ 0.44 │ 0.66 │       │
 │ proanthocyanins             │   0 │    0 │  1.6 │ 0.57 │ 0.41 │  1.2 │  1.9 │  3.6 │       │
 │ color_intensity             │   0 │    0 │  5.1 │  2.3 │  1.3 │  3.2 │  6.2 │   13 │       │
 │ hue                         │   0 │    0 │ 0.96 │ 0.23 │ 0.48 │ 0.78 │  1.1 │  1.7 │       │
 │ od280/od315_of_dilut        │   0 │    0 │  2.6 │ 0.71 │  1.3 │  1.9 │  3.2 │    4 │       │
 │ proline                     │   0 │    0 │  750 │  310 │  280 │  500 │  980 │ 1700 │       │
 │ target                      │   0 │    0 │ 0.94 │ 0.78 │    0 │    0 │    2 │    2 │       │
 └─────────────────────────────┴─────┴──────┴──────┴──────┴──────┴──────┴──────┴──────┴───────┘
───────────────────────────────── End ──────────────────────────────────
```

Figure 2 – Mywine_dataset Information



```
RangeIndex: 178 entries, 0 to 177
Data columns (total 14 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   alcohol                       178 non-null    float64
 1   malic_acid                    178 non-null    float64
 2   ash                           178 non-null    float64
 3   alcalinity_of_ash             178 non-null    float64
 4   magnesium                     178 non-null    float64
 5   total_phenols                 178 non-null    float64
 6   flavanoids                    178 non-null    float64
 7   nonflavanoid_phenols          178 non-null    float64
 8   proanthocyanins               178 non-null    float64
 9   color_intensity               178 non-null    float64
 10  hue                           178 non-null    float64
 11  od280/od315_of_diluted_wines  178 non-null    float64
 12  proline                       178 non-null    float64
 13  target                        178 non-null    int32  
dtypes: float64(13), int32(1)
memory usage: 18.9 KB
```

Figure 3 – Bar Chart showing Distribution of Target Variable

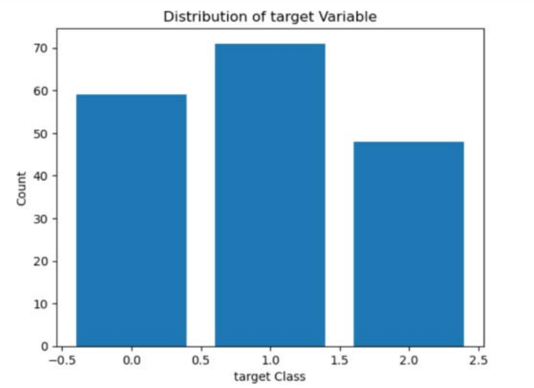


Figure 4 – Histogram Representation of The Mywine_dataset features

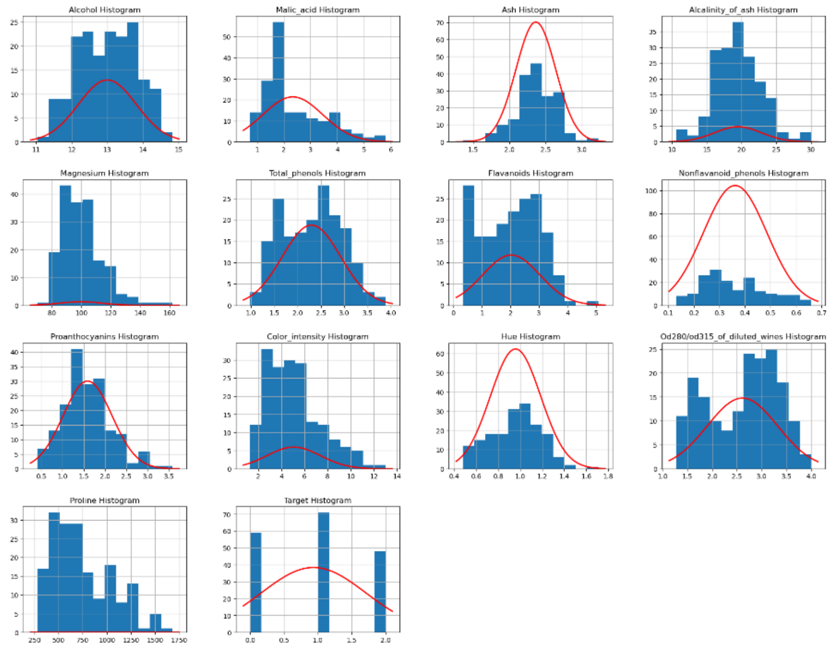


Figure 5- Boxplot showing features with target variable

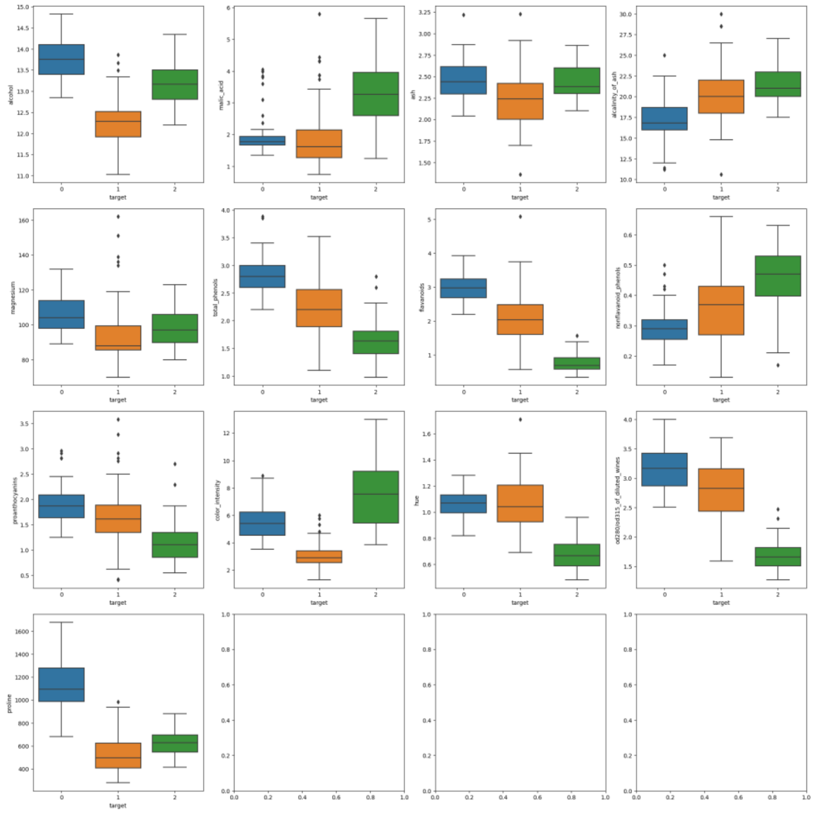
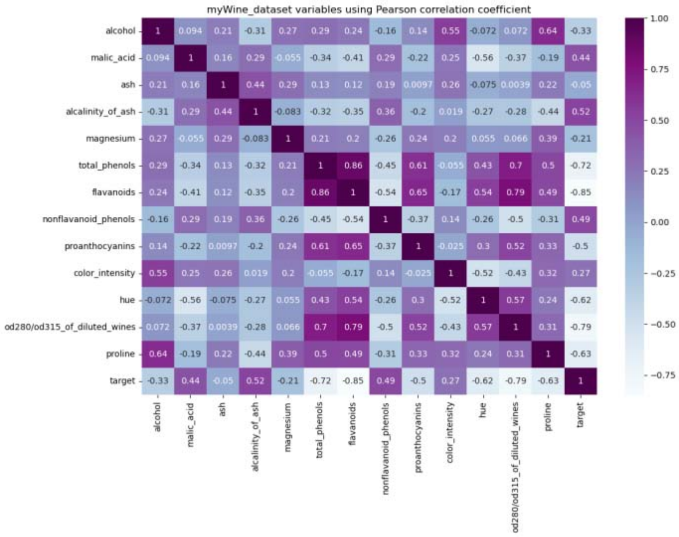


Figure 6 - MyWine_dataset variables using Pearson correlation coefficient



myWine_dataset variables using Pearson correlation coefficient

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315_of_diluted_wines	proline	target
alcohol	1	0.094	0.21	-0.31	0.27	0.29	0.24	-0.16	0.14	0.55	-0.072	0.072	0.64	-0.33
malic_acid	0.094	1	0.16	0.29	-0.055	-0.34	-0.41	0.29	-0.22	0.25	-0.56	-0.37	-0.19	0.44
ash	0.21	0.16	1	0.44	0.29	0.13	0.12	0.19	0.0097	0.26	-0.075	0.0039	0.22	-0.05
alcalinity_of_ash	-0.31	0.29	0.44	1	-0.083	-0.32	-0.35	0.36	-0.2	0.019	-0.27	-0.28	-0.44	0.52
magnesium	0.27	-0.055	0.29	-0.083	1	0.21	0.2	-0.26	0.24	0.2	0.055	0.066	0.39	-0.21
total_phenols	0.29	-0.34	0.13	-0.32	0.21	1	0.86	-0.45	0.61	-0.055	0.43	0.7	0.5	-0.72
flavanoids	0.24	-0.41	0.12	-0.35	0.2	0.86	1	-0.54	0.65	-0.17	0.54	0.79	0.49	-0.85
nonflavanoid_phenols	-0.16	0.29	0.19	0.36	-0.26	-0.45	-0.54	1	-0.37	0.14	-0.26	-0.5	-0.31	0.49
proanthocyanins	0.14	-0.22	0.0097	-0.2	0.24	0.61	0.65	-0.37	1	-0.025	0.3	0.52	0.33	-0.5
color_intensity	0.55	0.25	0.26	0.019	0.2	-0.055	-0.17	0.14	-0.025	1	-0.52	-0.43	0.32	0.27
hue	-0.072	-0.56	-0.075	-0.27	0.055	0.43	0.54	-0.26	0.3	-0.52	1	0.57	0.24	-0.62
od280/od315_of_diluted_wines	0.072	-0.37	0.0039	-0.28	0.066	0.7	0.79	-0.5	0.52	-0.43	0.57	1	0.31	-0.79
proline	0.64	-0.19	0.22	-0.44	0.39	0.5	0.49	-0.31	0.33	0.32	0.24	0.31	1	-0.63
target	-0.33	0.44	-0.05	0.52	-0.21	-0.72	-0.85	0.49	-0.5	0.27	-0.62	-0.79	-0.63	1

Methodology and Plan of Action

This report outlines the methodology and plan of action for predicting wine quality using Python for machine learning. It includes the following tasks:

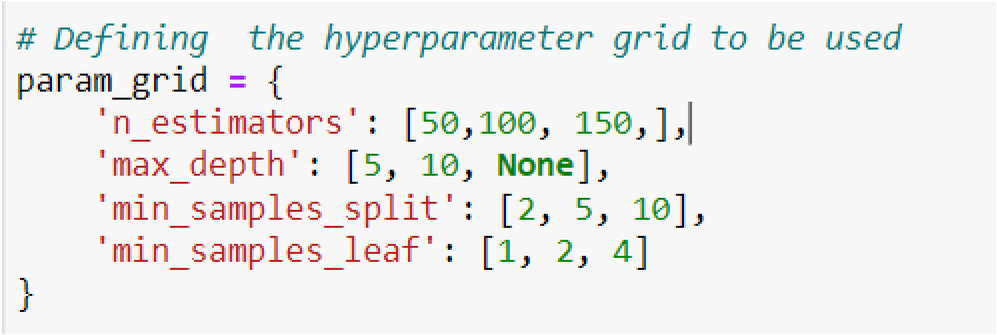
- Implementing a Random Forest Classifier: Creating a random forest classifier using sci-kit-learn. Discussing parameters, training method, and accuracy.
- Modifying with an MLP Classifier: Modifying the random forest model to use a multilayer perceptron (MLP) classifier. Justifying parameters and training method, and comparing accuracy with Task 1.
- Utilizing a Deep CNN: Replacing the classifier model with a deep convolutional neural network (CNN). Reporting and analyzing parameters, training method, and accuracy.
- Applying Clustering Methods: Using clustering on the wine dataset. Discussing cluster accuracy, optimal number of clusters, methodology, and evaluation metrics.

In addition, advanced techniques such as multiple clustering methods, hyperparameter optimization, and advanced activation functions will be explored to improve performance. The goal is to develop accurate AI models for wine quality prediction, showcasing a comprehensive understanding of the dataset and the application of AI techniques.

Task 1: Implementing Random Forest Classifier:

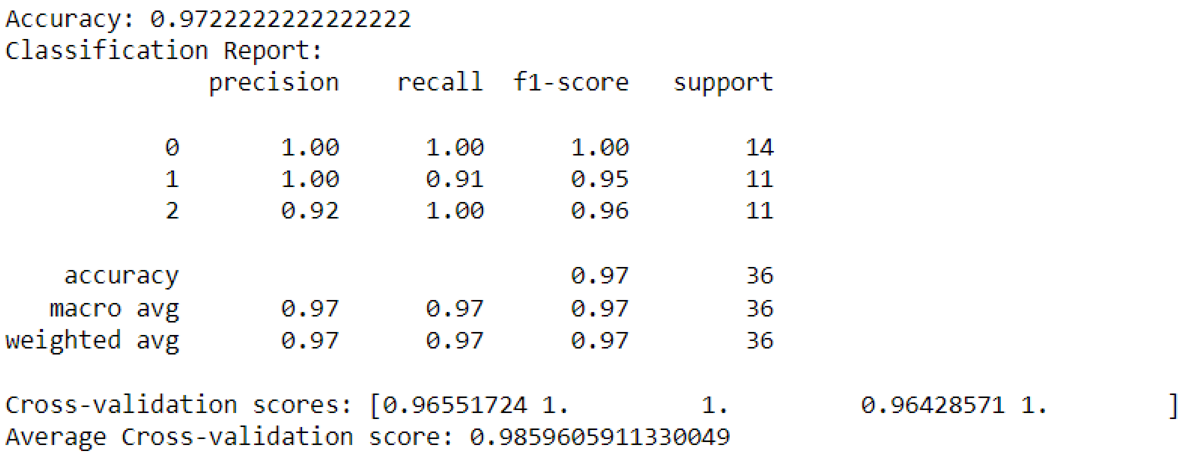
The Random Forest Classifier (RFC) from sci-kit-learn was implemented. The RFC model was instantiated with default parameters, including 100 trees, and trained using the “fit” method on the training data. The model's accuracy was evaluated using the accuracy_score metric, resulting in an accuracy of 97.22% on the test set.

To further optimize the RFC model's performance, hyperparameter tuning was conducted using GridSearchCV. A grid of hyperparameters, including the number of estimators, maximum depth, minimum samples split, and minimum samples leaf, was defined. The best hyperparameters, determined through the grid search, were 'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 2, and 'n_estimators': 150. These hyperparameters yielded a training accuracy of 99.29% and a test accuracy of 97.22%. The hyperparameter tuning process enhanced the model's performance by finding the optimal combination of hyperparameters. The RFC model, with the best hyperparameters obtained, proved to be highly effective for the wine quality classification task, achieving a high accuracy of 97.22% on the test set.



```
# Defining  the hyperparameter grid to be used
param_grid = {
    'n_estimators': [50,100, 150,],
    'max_depth': [5, 10, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

The RFC model achieved an accuracy of 0.9722 on the test set. The classification report provides additional evaluation metrics such as precision, recall, and F1-score for each class.The precision, recall, and F1-score for class 0 are all 1.00, indicating perfect performance. For class 1, the precision is 1.00, recall is 0.91, and F1-score is 0.95. For class 2, the precision is 0.92, recall is 1.00, and F1-score is 0.96.In terms of overall performance, the macro-average precision, recall, and F1-score are all 0.97. The weighted-average precision, recall, and F1-score, which take into account class imbalance, are also 0.97. The cross-validation scores for the RFC model across 5 folds are [0.96551724, 1.0, 1.0, 0.96428571, 1.0]. The average cross-validation score is 0.9859.These results indicate that the RFC model performs well in accurately classifying the wine quality, with high precision, recall, and F1-scores. The cross-validation scores further validate the model's performance and suggest good generalization capabilities.



```
Accuracy: 0.9722222222222222
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        14
           1       1.00      0.91      0.95        11
           2       0.92      1.00      0.96        11

    accuracy                           0.97        36
   macro avg       0.97      0.97      0.97        36
weighted avg       0.97      0.97      0.97        36

Cross-validation scores: [0.96551724 1.         1.         0.96428571 1.        ]
Average Cross-validation score: 0.9859605911330049
```

Figure 7 – Random Forest Accuracy and Classification Report

TASK 2 - Modifying with an multilayer perceptron (MLP) classifier

To further explore the wine quality classification task, the MLP classifier was employed as an alternative approach. The MLP model is a type of neural network that consists of multiple layers of interconnected nodes, allowing for complex nonlinear relationships to be captured.

The MLP classifier was trained using the following parameters: hidden_layer_sizes = (20, 15, 10), activation = 'logistic', random_state = 2, and max_iter = 1000. These parameter values were selected based on prior knowledge and experimentation to achieve a good balance between model complexity and performance. The hidden_layer_sizes parameter defines the number of nodes in each hidden layer, with three hidden layers in this case. The activation parameter specifies the activation function used in the nodes, with logistic activation chosen for its suitability in classification tasks. The random_state parameter ensures reproducibility of the results, while the max_iter parameter determines the maximum number of iterations for the model to converge.

The MLP classifier achieved an accuracy of 0.3056 when the data was normalized using the MinMaxScaler. This accuracy indicates the proportion of correctly predicted wine quality labels on the normalized test set. Using the bagging method, the Bagging MLP Classifier Accuracy accuracy of 0.9444 was achieved(same as MLP using the un-normalized dataset).

Model	Random Forest	MLP (with original feature data)	MLP (with MinMaxScaler)	MLP (with StandardScaler)	MLP(with bagging method)
Accuracy Score	0.9722	0.9444	0.3056	0.9722	0.9444

Table 1 – Showing Accuracy Scores for RF and MLP Models

In Table 1, we have the accuracy scores for different models: Random Forest (RF), Multi-Layer Perceptron (MLP) with original feature data, MLP with MinMaxScaler, and MLP with StandardScaler.

The Random Forest model achieved an accuracy score of 0.9722, indicating its high accuracy in classifying the data. It also demonstrated excellent performance across all classes, as reflected in the precision, recall, and F1-scores reported in the classification report above.

The MLP model trained on the original feature data as well as using the bagging method obtained an accuracy score of 0.9444, showcasing its effectiveness in classification. However, it had slightly lower accuracy compared to the Random Forest model. While the MLP model with MinMaxScaler normalization achieved an accuracy score of 0.3056, indicating a significant drop in performance.

Contrastingly, the MLP model with StandardScaler normalization achieved an accuracy score of 0.9722, equal to the Random Forest model. This implies that the MLP model, when utilizing StandardScaler normalization, achieved a comparable accuracy to the Random Forest model in classifying the data.

Therefore, both the Random Forest model and the MLP model with StandardScaler normalization showed the highest accuracy scores of 0.9722, highlighting their effectiveness in classification. The MLP model with original feature data also performed well with an accuracy score of 0.9444. However, the MLP model with MinMaxScaler normalization had a significantly lower accuracy score of 0.3056, indicating poorer performance. Generally, the Random Forest model demonstrated stronger performance, exhibiting high accuracy and balanced precision, recall, and F1-scores for each class, as depicted in the classification report.

Task 3: Utilizing a Deep Convolutional Neural Network (CNN)

In this subsection, we present the parameters and training method for a Convolutional Neural Network (CNN) model. The CNN model includes a single convolutional layer with 32 filters and a kernel size of 3, applying the ReLU activation function. A max pooling layer with a pool size of 2 was used, followed by a dense layer with 64 neurons and a ReLU activation function. The output layer consists of 3 neurons with a softmax activation function. The Adam optimizer with default learning rate and sparse categorical cross-entropy loss function are employed. The model's performance was evaluated using accuracy as the metric. The normalized data was reshaped to be compatible with the CNN architecture. It assumes that the original data is a 2D array, so it reshapes it to a 3D array with dimensions (samples, features, channels=1).

The initial CNN model achieved 88.89% accuracy on the test set. To improve performance, adjustments were made to the model's architecture and training parameters. The number of filters in the convolutional layer was increased from 32 to 64, kernel size from 3 to 5, and units in the first dense layer from 64 to 128. Training parameters were also modified, with the number of epochs increased from 10 to 20 and batch size from 32 to 64. These changes resulted in an improved accuracy of 97.22% on the test set and a test loss of 0.0905. These adjustments aimed to enhance the model's ability to learn complex patterns, while increased epochs and batch size allowed for longer training, more updates per iteration, and improved generalization to unseen data. Further experimentation and parameter tuning could yield even better results.

Comparison with LSTM Model:

Comparing the adjusted CNN model to an LSTM model, the CNN model exhibited higher accuracy (97.22% vs. 83.33%) and lower test loss (0.0905 vs. 0.5643). The adjusted CNN model outperformed both the initial CNN model and the LSTM model, highlighting the importance of parameter tuning.

Advanced Activation Function Report:

An advanced activation function, specifically the Leaky ReLU, was incorporated into the CNN model. The Leaky ReLU activation function helps address the "dying ReLU" problem by allowing a small positive gradient for negative inputs, which can prevent neurons from becoming completely inactive. The updated CNN model with the Leaky ReLU activation function was trained and evaluated on the test set. The normalized data was reshaped to fit the CNN architecture, and the model was compiled with the Adam optimizer and sparse categorical cross-entropy loss function. After training the model for 10 epochs with a batch size of 32, the accuracy on the normalized test set was computed and found to be approximately 91.67%. This indicates that the inclusion of the Leaky ReLU activation function contributed to the improved performance of the CNN model.

The combined improvements made to the CNN model, including adjustments to the architecture, training parameters, and the addition of the Leaky ReLU activation function, resulted in a significantly improved accuracy of 97.22% on the test set. The inclusion of the Leaky ReLU activation function enhanced the model's ability to capture and learn from complex patterns in the data. These findings highlight the importance of parameter tuning and the use of advanced activation functions in optimizing the performance of deep learning models in classification tasks. Further experimentation and parameter tuning could yield even better results."

Task 4- Clustering Analysis for Wine Quality Classification

In this section, various clustering techniques were applied to the wine quality dataset in order to explore patterns and groupings within the data. The objective was to investigate the effectiveness of clustering algorithms for classifying wine quality based on the provided features and determine optimum number of clusters.

K-means Clustering:

The first clustering technique utilized was K-means clustering. The dataset was preprocessed by performing data normalization using the StandardScaler. K-means clustering with three clusters was then applied to the normalized data. The cluster labels obtained were evaluated for accuracy by comparing them with the actual target labels using the accuracy_score metric. The accuracy of the K-means clustering was computed and found to be 0.2865.

Determining Optimum Number of Clusters:

To determine the optimal number of clusters for the dataset, the elbow method was employed. This involved running K-means clustering with different numbers of clusters and calculating the inertia (within-cluster sum of squares) for each case. The inertia values were then plotted against the number of clusters, and the "elbow point" on the plot indicated the optimal number of clusters. The code snippet used for this analysis is as follows: According to the elbow curve, the optimum number of clusters was found to be 7.

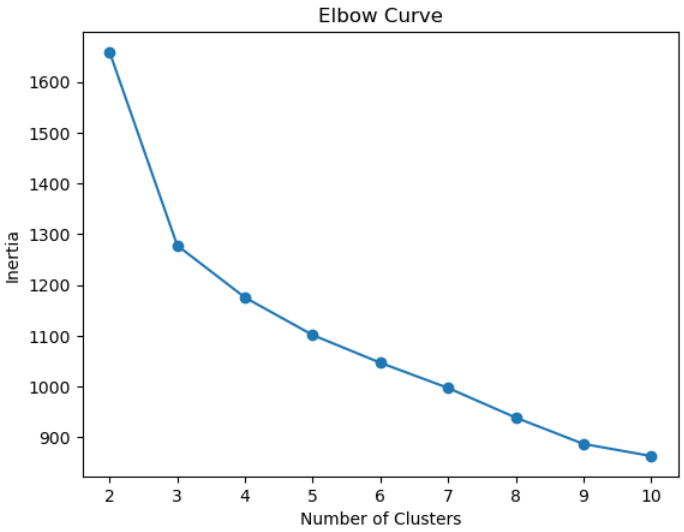


Figure 8- Elbow Curve

Silhouette Analysis:

Silhouette analysis was performed to assess the quality of the clusters obtained through K-means clustering. The silhouette score measures the compactness and separation of the clusters. The higher the silhouette score, the better the clustering results. Furthermore, the silhouette score was computed and found to be 0.2849. This indicates that the clustering results have low consistency and overlap among the clusters.

Investigating Methods to Improve Accuracy:

To enhance clustering accuracy, an alternative algorithm called Agglomerative Clustering(hierarchical clustering) was explored. The dataset underwent StandardScaler normalization before applying Agglomerative Clustering with three clusters. Cluster accuracy was evaluated by comparing the resulting labels with the actual targets. Unfortunately, the Agglomerative Clustering method only achieved an accuracy of 0.0449, indicating poor performance in accurately classifying the wine samples into clusters. The mean shift clustering was applied and produced a better accuracy of 0.3539.

K-means clustering and Agglomerative Clustering were applied to the wine quality dataset. K-means clustering achieved a higher cluster accuracy of 0.2865, while Agglomerative Clustering resulted in a lower accuracy of 0.0449. However, both methods exhibited low accuracy in accurately grouping the wine samples into clusters.

Ethical and Social Impact of AI Solutions in the Wine Industry

AI's application in wine dataset classification has significant societal and ethical implications. It offers valuable insights for optimizing wine quality and empowering consumers to make informed choices. However, concerns arise regarding transparency, accountability, and data privacy.

Transparency and explainability of AI algorithms pose ethical challenges. When AI models make decisions without providing clear justifications, trust issues may arise among producers and consumers. This is particularly important for contentious classifications. To address this, interpretable models must be carefully designed and implemented.

Accountability is another crucial concern. In cases of classification errors, determining responsibility becomes essential. Developers, dataset providers, and wine producers all need a robust framework that assigns accountability and safeguards their interests.

Data privacy is also at stake. AI models may require sensitive information like geographic indicators, grape types, and production methods, which are trade secrets in the wine industry. Strict adherence to data protection laws and consent protocols is crucial to handle this information appropriately.

Addressing these ethical considerations is vital to ensure responsible use of AI in wine dataset classification. This promotes trust, fairness, and harnesses the technology's potential benefits.

Conclusion/Summary

This report demonstrates the efficacy of AI techniques, including deep learning and machine learning algorithms, in accurately predicting wine quality. Models like CNN, Random Forest, MLP, and LSTM have proven their adaptability and efficiency, especially when optimized with advanced activation functions like Leaky ReLU.

However, the study reveals that clustering techniques such as Mean Shift, K-means, and Agglomerative Clustering performed poorly in wine quality classification. This suggests the need for more advanced clustering or ensemble methods to achieve better results.

The report emphasizes the significance of data normalization, which greatly influences the performance of models like MLP. It also raises ethical and social concerns associated with AI solutions in sensitive industries like wine production, stressing the importance of transparency, accountability, and stringent data privacy measures.

References

By, W. (2018) Old world, new tech: how the wine industry is taking on AI, Outside Insight. Available at: https://outsideinsight.com/insights/old-world-new-tech-how-the-wine-industry-is-taking-on-ai/ (Accessed: May 29, 2023).

Harrington, R.J., 2007. Food and wine pairing: A sensory experience. John Wiley & Sons.

Krishna, A. S. (2022) “Wine Dataset from Scikit Learn.”

Mandrile, L., Zeppa, G., Giovannozzi, A.M. and Rossi, A.M., 2016. Controlling protected designation of origin of wine by Raman spectroscopy. Food chemistry, 211, pp.260-267.

Python built-in datasets (2021) Python and R Tips. cmdlinetips. Available at: https://cmdlinetips.com/2021/11/access-datasets-from-scikit-learn/ (Accessed: May 26, 2023).

Ryan, M., 2022. The social and ethical impacts of artificial intelligence in agriculture: mapping the agricultural AI literature. AI & SOCIETY, pp.1-13.

Saranraj, P., Sivasakthivelan, P. and Naveen, M., 2017. Fermentation of fruit wine and its quality analysis: A review. Australian Journal of Science and Technology, 1(2), pp.85-97.

UCI machine learning repository: Wine data set (no date) Uci.edu. Available at: http://archive.ics.uci.edu/ml/datasets/wine (Accessed: May 10, 2023).