University of Hertfordshire UH

Machine Learning Case Study Report

Shubham Verma

22099668

Sv23abk@herts.ac.uk

Google Colab

GitHub

Introduction

This assignment centres on utilizing machine learning strategies to analyse wine quality datasets and create predictive models to classify wines and their quality. Wine quality evaluation may be a basic portion of the refreshment industry, impacting generation forms, customer inclinations and showcase competitiveness. Utilizing diverse traits of the dataset, including chemical and tactile characteristics, we attempt to form precise classification models that can recognize diverse quality classes of wine.

The primary objective of the assignment is to consider diverse machine learning algorithms. such as support vector machines, random forest, and multi-layer perceptrons to decide which one works best for anticipating wine quality. Through deep analysis and assessment of these models, we attempt to discover out the foremost proficient way to precisely classify wines.

In expansion, this extend could be a viable exhibit of the application of machine learning in genuine life, particularly for drinks. zone by giving understanding into the components affecting wine quality and the adequacy of distinctive classification calculations, it gives important data for winemakers, analysts, and industry experts. At last, the extend points to illustrate the potential of machine learning procedures in wine quality assessment and mechanical decision-making forms.

Dataset

The Wine Quality dataset is a good dataset for predicting the quality class of wine sample with the help of different algorithms of machine learning. This dataset is comprised of different physical and sensory properties of each wine for two qualities: red and white. Each feature in this dataset is capable to change the quality of wine to a significant level. So, this dataset can be used for building binary classification models that can predict whether a given wine sample belongs to a specific quality class.

The most important reason for selection of this dataset is that it is helpful in building an application that can be implemented in practical scenarios. The quality of wine is very important in beverage industry and has a great effect on the production processes of wineries, preferences of the consumers and investment of the market.

The application that we are going to implement based on the results of above dataset can become a useful tool for wineries that can help them in finding improvements in their production methods, trends in the quality of wine and maintaining high standards consistently. The second reason for choice of this dataset is that it is helpful for classification problems. With its vast objective repertoire of features, it facilitates experimentation with several prominent machine learning techniques such as decision trees, support vector machines, and neural networks. These tackling the inherent patterns embedded within the data, can compute the weights of the features specifying each quality class.

Moreover, predictive quality estimation models can be formed with access to labelled data which means the number of times models can be trained on the input quality of a wine is known. This results in more reliable performance scores for novel observations or data points. Overall, the wine quality dataset makes an excellent cursory data to observe and develop robust classification algorithms with real world applications and implications.

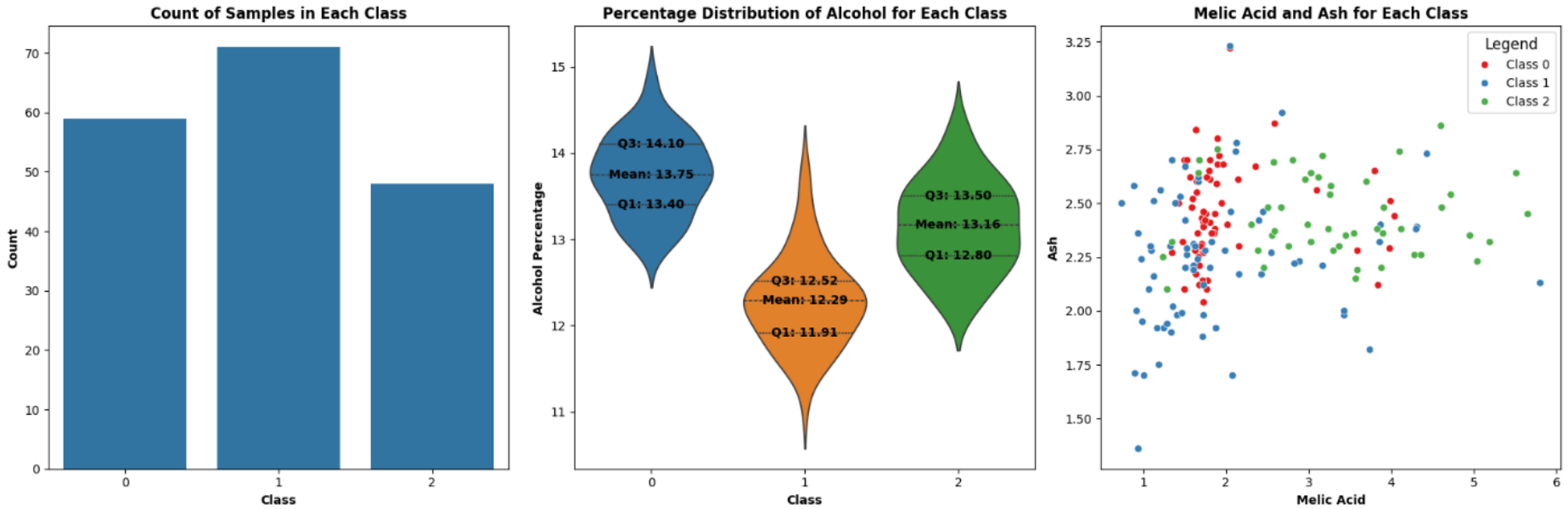


Figure 1: Exploratory Data Analysis

The comprehensive analysis of the Wine Quality dataset provides key insights into its composition and characteristics. The first plot depicts the distribution of samples across the dataset's classes, highlighting class 1 as the most populous. The second visualization reveals the distribution of alcohol percentages within each class, showcasing distinct peaks around 13-14% for classes 1 and 2, and a slightly lower peak around 12-13% for class 0. Lastly, the scatter plot illustrates the relationship between melic acid and ash content, indicating a positive correlation, particularly pronounced in class 2. These findings offer valuable insights into the dataset's structure, including class imbalances, alcohol content patterns, and potential feature interactions. Such understanding is fundamental for subsequent analysis steps, enabling informed decisions regarding model selection, feature engineering, and overall data interpretation in classification tasks.

Data Pre-Processing

The Wine Quality Dataset Analysis project uses several preprocessing techniques to prepare data for machine learning models. These techniques are critical to ensuring that data is in the proper format and condition for effective analysis and modelling. Some of the preprocessing techniques used in this project are:

- **Handling missing values:** The dataset is checked for missing values and in case found, fitting procedures such as imputation or deletion are utilized. This guarantees that the dataset is total and does not contain invalid values that seem contrarily influence the execution of the demonstrate.
- **Scaling of functions:** Since functions in the dataset can have different scales, characteristic scaling techniques such as StandardScaler are used. standardization of functions. to the corresponding scale. This helps prevent larger features from dominating the learning process and ensures faster and more efficient model convergence.
- **Dimensionality Reduction:** PCA (Principal Component Analysis) is used to reduce the dimensionality of a data set while keeping most of it. variance by transforming the original features into a lower dimensional space, PCA helps visualize data and reduce computational complexity, especially for high-dimensional datasets.
- **Train-test split:** The dataset is split into training and test sets using the train_test_split function. This grants the execution of the model to be assessed with covered up information, which makes a difference evaluate the generalizability of the model.

In general, these preprocessing techniques help clean, transform, and organize the data, making it suitable for machine building and evaluation, effectively learn models.

```
18  #Normalize features and split into training and testing sets
19  scaler = StandardScaler()
20  X_scaled = scaler.fit_transform(X)
21  X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3,
22                                                      random_state=42)
23
24  # Apply PCA to reduce dimensionality for visualization
25  pca = PCA(n_components=2)
```

ML architecture and parameters

In the Wine Quality dataset analysis project, several machine learning algorithms are employed to classify the quality of wine based on various features. The chosen ML architecture consists of the following algorithms:

- **Support Vector Machine (SVM):** SVM with an RBF kernel is utilized for classification. SVM is known for its ability to handle high-dimensional data and complex decision boundaries. The RBF kernel allows SVM to capture non-linear relationships between features, making it suitable for this classification task.
  Architecture: *Input features -----> Kernel function -----> Decision function -----> Class prediction*
  Parameters: *Kernel: Radial Basis Function (RBF), Regularization Parameter (C): Default value, Gamma: Default value, Random State: 42*

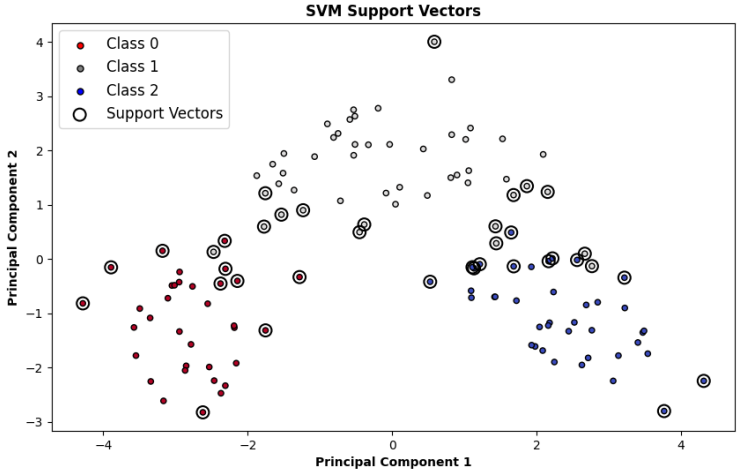


Figure 2: SVM Support vectors

- **Random Forest:** Random Forest classifier is employed, which is an ensemble learning method based on decision trees. Random Forest combines multiple decision trees to improve generalization and reduce overfitting. It works well with both numerical and categorical data and can handle high-dimensional datasets effectively.
  Architecture: *Input features -----> Decision trees -----> Voting mechanism -----> Class prediction*
  Parameters: *Number of Trees (n_estimators): Default value, Criterion: Gini impurity, Random State: 42*

```python
# Train the Random Forest model
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train_pca, y_train)

# Make predictions
y_pred_rf = rf_model.predict(X_test_pca)

# Evaluate the model's performance
accuracy_rf = accuracy_score(y_test, y_pred_rf)
```

- **Multilayer Perceptrons (MLPs):** MLP classifier is utilized, which is a type of artificial neural network capable of learning complex relationships in data. MLP consists of multiple layers of nodes (neurons) and can capture non-linear relationships between features. By increasing the maximum number of iterations, MLP can better optimize its weights and biases during training, potentially improving its performance.
  Architecture: *Input features -----> Hidden Layers (Dense) -----> Activation function -----> Output layer -----> Class prediction*
  Parameters: *Hidden Layer Sizes: Default value, Activation Function: ReLU (Rectified Linear Unit), Solver: Adam (Stochastic Gradient-Based Optimizer), Maximum Number of Iterations: 1000, Random State: 42*

```python
# Train the MLP model with increased maximum iterations
mlp_model = MLPClassifier(random_state=42, max_iter=1000)  # Increase max_iter to 1000
mlp_model.fit(X_train_pca, y_train)

# Make predictions
y_pred_mlp = mlp_model.predict(X_test_pca)

# Evaluate the model's performance
accuracy_mlp = accuracy_score(y_test, y_pred_mlp)
```

- **Decision Tree:** Using CART algorithm to recursively partition the feature space. It makes decisions based on splitting criteria such as Gini impurity to maximize information gain. The resulting tree predicts the class labels of samples by traversing from the root to leaf nodes, where final predictions are made. This model captures complex decision boundaries.
  Architecture: *Input features -----> Tree structure (Internal nodes) -----> Splitting criteria -----> Leaf node prediction -----> Class prediction*
  Parameters: *Criterion: Gini impurity, Splitter: Best, Maximum Depth: Default value, Random State: 42*

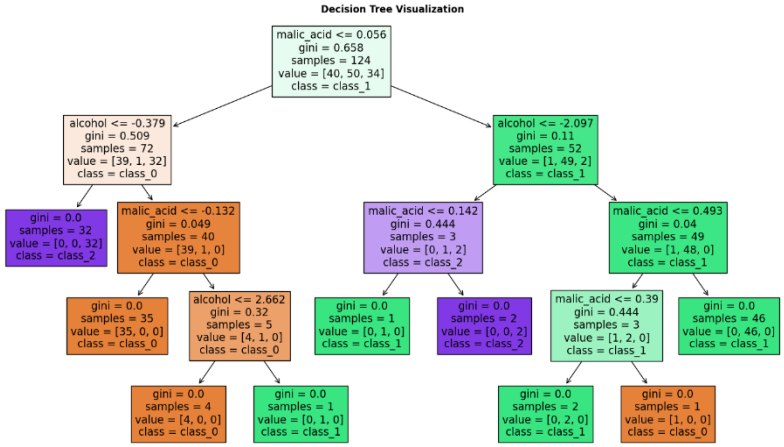


Figure 3: Decision Tree

For each algorithm, certain parameters are chosen based on their effectiveness in previous studies and empirical testing. For example, in SVM, the choice of kernel (RBF) and its associated parameters such as gamma and regularization parameter (C) are crucial for achieving optimal performance. Similarly, in Random Forest, parameters like the number of trees (n_estimators) and the maximum depth of each tree are important considerations. In MLP, parameters such as the number of hidden layers, number of neurons per layer, and the maximum number of iterations are chosen to balance model complexity and performance. These parameters are selected through experimentation and cross-validation to ensure robustness and generalization of the models.

Results and critical evaluation

Model	Accuracy
Support Vector Machine (SVM)	~98%
Random Forest	~94%
Multilayer Perceptrons (MLP)	~96%
Decision Tree	~92%

Each model has its strengths and weaknesses, ensemble methods like Random Forest stand out for their robustness and ease of use in classification tasks. However, the choice of the model eventually depends on variables such as computational assets, interpretability, and the particular prerequisites of the issue at hand.

Decision boundaries learned by Support Vector Machine (SVM), Random Forest, and Multi-Layer Perceptron (MLP) algorithms. Each graph depicts data points clustered in two regions, indicating effective classification. SVM's boundary is linear, while Random Forest and MLP exhibit more complex boundaries, implying varied decision rules.

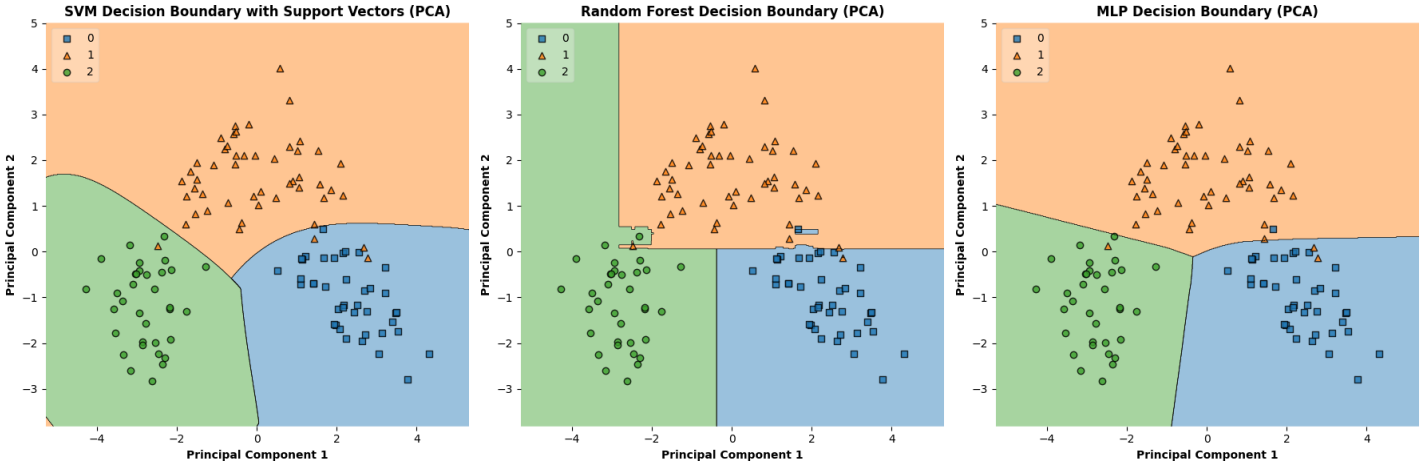
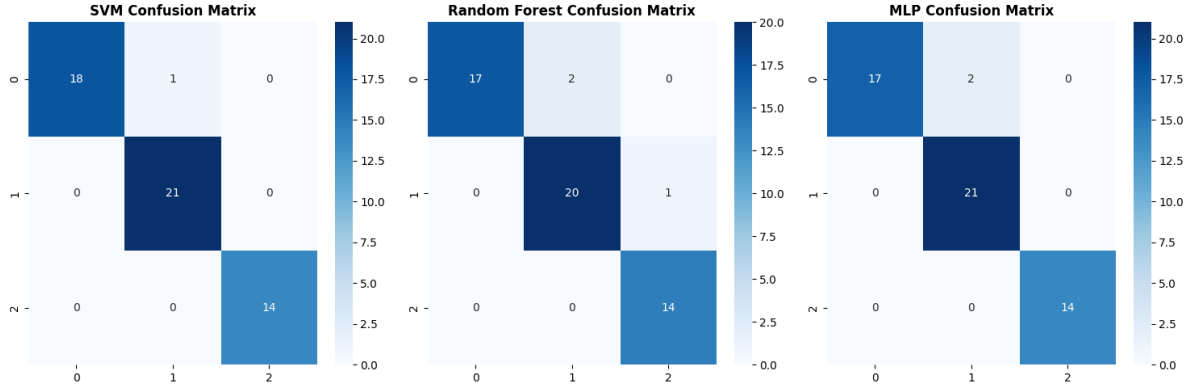


Figure 4: Decision boundaries plots for SVM, Random Forest and MLP



The confusion matrices outline the classification execution of SVM, Irregular Woodland, and MLP models over three classes (0, 1, 2). SVM correctly classified most instances but misclassified some in classes 0 and 1. Random Forest had fewer misclassifications, while MLP exhibited some misclassifications in classes 0 and 2.

Figure 5: Confusion matrix plots for SVM, Random Forest and MLP

The learning curves depict the training and cross-validation scores for SVM, Random Forest, and MLP models. SVM and Random Forest exhibit steady performance improvement, while MLP shows more variability, suggesting potential overfitting. These insights aid in model selection and parameter optimization for better generalization.

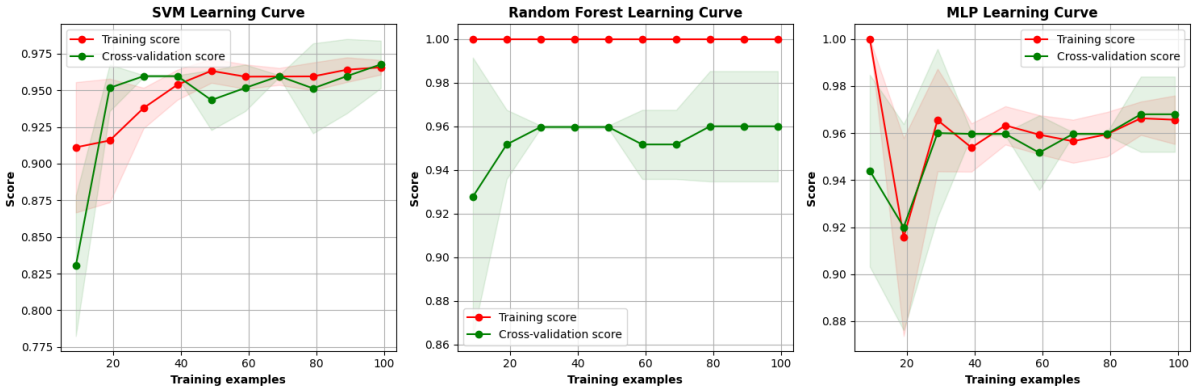


Figure 6: Learning curve plots for SVM, Random Forest and MLP

Support Vector Machine (SVM) achieves the highest accuracy (~98%), while Random Forest (~94%) excels with fewer misclassifications. Multilayer Perceptrons (MLP) (~96%) show competitive accuracy but risk overfitting, seen in their variable learning curve. Decision boundaries highlight SVM's linear boundary versus Random Forest and MLP's complex boundaries. Each model has trade-offs, emphasizing the need to consider computational resources and interpretability. SVM and Random Forest demonstrate steady performance improvement, while MLP's variability suggests potential overfitting. Understanding these nuances aids in selecting the most suitable model for diverse classification tasks.

Limitations and improvements

Model Performance: While the models achieve decent accuracy, there may be room for improvement in terms of performance metrics such as precision, recall, and F1-score. Fine-tuning hyperparameters or exploring more sophisticated algorithms could enhance performance.

Overfitting: Overfitting occurs when a model learns to memorize the training data instead of generalizing patterns. Techniques such as regularization or using more training data can help mitigate overfitting.

Feature Engineering: The effectiveness of machine learning models heavily relies on the quality of features. Exploring additional feature engineering techniques or incorporating domain knowledge could improve model performance.

Model Interpretability: Some models, such as neural networks, are often considered black boxes, making it challenging to interpret their decisions. Employing techniques like feature importance analysis or using more interpretable models like decision trees can enhance model interpretability.

Data Imbalance: Class imbalance in the dataset can lead to biased models favouring the majority class. Techniques such as oversampling, under sampling, or using class weights can address this issue and improve model performance on minority classes.

Ensemble Methods: Leveraging ensemble methods such as bagging, boosting, or stacking can potentially improve model performance by combining predictions from multiple models.

Algorithm Selection: While the chosen algorithms (SVM, Random Forest, MLP, Decision Tree) are popular and widely used, exploring other algorithms or ensemble combinations tailored to the specific dataset could yield better results.

Cross-Validation: Employing more advanced cross-validation techniques, such as stratified k-fold or nested cross-validation, can provide a more robust estimate of model performance and generalization ability.

Addressing these limitations and exploring potential areas of improvement can lead to more effective and reliable machine learning models.