Part 1: Classification

Task 1A: Data pre-processing:

1. Introduction:

In this task, the report outlines the steps taken in the data pre-processing of the dataset “winequality-RED.csv”. The goal was to handle missing values by replacing them with the average value of the respective attributes and then exporting the cleaned dataset.

1. Steps Undertaken:
2. **Data Loading:** The dataset “winequality- RED.csv” was load into Pandas DataFrame for analysis and manipulation.
3. **Initial Data Check:** The dimensions of the DataFrame were verified to ensure it contained the expected number of instances (880 rows and 7 columns, excluding headings)
4. **Missing Value Identification:** Missing values in the dataset were identified. The attributes ‘density’ and ‘chlorides’ were found to contain missing values.
5. **Replacement of Missing Values:** Missing values were replaced with the mean value of the respective attribute. The table below summarizes the number of missing values, and the mean values used for replacement.

|  |  |  |
| --- | --- | --- |
|  | Number of missing values | Mean value (round to 3 decimal points) |
| Density | 8 | 0.997 |
| Chlorides | 14 | 0.089 |

1. **Exporting Cleaned Data:** The processed DataFrame was then exported to a new CSV file named “newwinequality.csv”.

Task 1B: Wine Quality Prediction

1. **Introduction:**

In this section, the wine quality will be predicted using two different model which is Decision Tree (using default parameters) and Neural Netweek (NN). We also explore the effect of hyperparameters on model performance by experimenting with different configurations.

The performance metrics used for comparison include:

* Average Test Accuracy from Cross-Validation (using 10-fold)
* Average Test ROC AUC (Receiver Operating Characteristic for Area Under the Curve) score from Cross-Validation (using 10-fold)
* Confusion Matrix measures True Positives, False Positives, True Negatives and False Negatives. This matrix is computed for the best estimator based on the highest test accuracy.

Performance metrics from Cross-Validation was introduced to evaluate a model’s generalisation ability. It helps estimate how well the model will perform on unseen data. The average test accuracy gives more robust and reliable measure of the model’s performance compared to just a single train-test split.

1. **Predicting Wine Quality Using Decision Tree Model:**

Based on the significant influence of the performance complexity, the key hyper-parameters tested include:

* Maximum depth (by default = None): Limits the depth of the tree, controlling its complexity that can prevent overfitting.
* Minimum sample split (by default = 2): The minimum number of samples required to split an internal node. Higher values can prevent the model from learning overly specific patterns.
* Minimum sample leaf (by default = 1): The minimum number of samples required to be at a leaf node. This helps to smooth out the model by ensuring that leaf nodes contain enough data.
* Criterion (by default = gini): measuring the quality of a split. Gini and Entropy are the most

commonly used criteria, each providing a different method of information gain calculation.

These parameters are crucial in balancing the model's ability to generalize while preventing overfitting and underfitting.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Decision Tree hyper-parameters | | | | Cross Validation | | Confusion Matrix | | | |
| Max Depth | Min Samples Split | Min Samples Leaf | Criterion | Average Test Accuracy | Average Test ROC AUC | True Positive (TP) | False Positive (FP) | True Negative (TN) | False Negative (FN) |
| **None** | **2** | **1** | **gini** | **0.833** | **0.756** | **17** | **3** | **65** | **3** |

As we can see, the default parameters provided a strong baseline. By balancing all parameters, we can maintain good performance with potential minor gains in specific areas. It's important to tailor these parameters based on the specific requirements and characteristics of the dataset.

1. **Predicting Wine Quality Using Neural Network (NN) Models and Hyperparameter Tuning:**

Based on the significant influence of the performance complexity, the key hyper-parameters tested include:

* **Hidden Layer Sizes (default = 100,1)**: represents the number of neurons (100) in a hidden layer (1). This controls the model’s complexity, more neurons and or hidden layers can potentially capture intricate patterns, while fewer may lead to faster but simpler models.
* **Solver (default = 'adam')**: determines the optimisation algorithm used to minimise the loss function during training. We will experiment with Adam(adaptive) and SGD (manually turned) solver and see the difference.
* **Learning Rate Initialization (default = 0.001)**: sets the initial learning rate for weight updates during training. Changing learning rate affects how quickly the model learns. More stable learning can achieve with lower rates, while higher rates can accelerate the process but risk skipping over optimal solutions.
* **Max Iterations (default = 200)**: specifies the maximum number of iterations over the training data. Higher figures allow the model to run for more epochs and improve learning ability over complex dataset, preventing underfitting. Fewer iterations result in faster training but may lead to stopping early (underfitting).

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ANN hyper-parameters | | | | Cross Validation | | Confusion Matrix | | | |
| Hidden Layer Sizes | Solver | Learning Rate | Max Iter | Average Test Accuracy | Average Test ROC AUC | TP | FP | TN | FN |
| **(100,)** | **adam** | **0.001** | **200** | **0.889** | **0.927** | **19** | **3** | **65** | **1** |
| (500, 50) | adam | 0.001 | 200 | 0.880 | 0.890 | 17 | 2 | 66 | 3 |
| (50,) | adam | 0.001 | 200 | 0.897 | 0.926 | 16 | 0 | 68 | 4 |
| (100,) | sgd | 0.001 | 200 | 0.886 | 0.921 | 16 | 0 | 68 | 4 |
| (100,) | adam | 0.01 | 200 | 0.860 | 0.888 | 16 | 2 | 66 | 4 |
| (100,) | adam | 0.1 | 200 | 0.858 | 0.895 | 16 | 1 | 67 | 4 |
| (100,) | adam | 0.001 | 600 | 0.878 | 0.916 | 15 | 0 | 68 | 5 |
| **(100,)** | **adam** | **0.001** | **100** | **0.893** | **0.928** | **16** | **0** | **68** | **4** |
| (100,) | adam | 0.01 | 300 | 0.869 | 0.883 | 15 | 2 | 66 | 5 |

Experimenting these hyper-parameters will assist in finding the optimal algorithms for maximizing model accuracy while avoiding overfitting or underfitting. The key findings are represented in the table below:

* Default Parameters (in bold) (Hidden Layer Sizes=100,1, Solver = adam, Learning Rate Init = 0.001, Max Iterations = 200)provided a good balancewith an average accuracy of 0.889 and average ROC AUC of 0.927. It showed decent performance in minimising False Positive (FP=3) and False Negatives (FN=1).
* Adjusting the Hidden Layer Size had mixed effects on model performance. Increasing the size of neurons to 500 and hidden layers to 50 resulted in deteriorated model performance. Conversely, decreasing the size to 50 neurons while keeping other parameters unchanged lead to a slight increase in average Accuracy and 100% correct prediction in True Positive but increase in number of False Negative to 4. It could mean that increasing in complexity of model may not necessarily improve the model performance.
* Switching Solver to SGD (Stochastic Gadient Descent) while keeping other hyper-parameters at their default values led to a diminished model performance in term of Accuracy and ROC AUC metrics as well as a growth in number of False Negative. However, model achieved 0 number of False Positive.
* Increasing Learning Rate to 0.01 and 0.1 demonstrated a slight deduction in both Accuracy and ROC AUC score as well as confusion matrix elements.
* Adjusting the Maximum Iteration has mixed effects on model performance. While increasing Maximum Iteration to 600 negatively impact the model performance, reducing this parameter to 100 interestingly improve the model performance overall with higher average Accuracy, average ROC AUC score. It also reduced the number of False Positive to 0. However, it created undesirable impact on False Negative (FN=4).
* On the last experiment, with an increasement on Learning rate and Maximum Iteration, model performance reduced slightly.

Overall, from the above findings, the 8th experiment (in Bold & highlighted) with 100 Maximum Iteration while keeping all other hyper-parameters as default achieved the highest average Test Accuracy and average Test ROC AUC at 0.893 and 0.928 respectively. This model also resulted in 100% correct prediction for the True Positive.

1. **Conclusion on model performance using Decision Tree Classifier and Neural Network (NN) Learne**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Classifier | Cross Validation | | Confusion Matrix | | | |
| Average Test Accuracy | Average Test ROC AUC | True Positive (TP) | False Positive (FP) | True Negative (TN) | False Negative (FN) |
| **Decision Tree**  (Default Model) | 0.833 | 0.756 | 17 | 3 | 65 | 3 |
| **Neural Network**  **(Adjusted Max Iter to 100)** | **0.893** | **0.928** | **16** | **0** | **68** | **4** |

The table above summarized the model performance from the previous experiments. The Neural Network demonstrated advanced predictive capabilities, achieving a higher average Test Accuracy (0.893), a greater average Test ROC AUC score (0.928) and a more favourable confusion matrix, with only 4 incorrect predictions, compared to Decision Tree model.

Part 2: Clustering

Task 2B: Clustering label results

|  |  |  |
| --- | --- | --- |
|  | y= ‘g’ | y= ‘b’ |
| Cluster 0 | 93 | 68 |
| Cluster 1 | 33 | 157 |

The table above presented the clustering results for task 2B. Using the K-means clustering algorithm with K= 2, each data point was assigned to one of two clusters. The class label ‘y’ was used to count the number of signals within each cluster. This clustering analysis provides valuable insights into the structure of the data.

However, we can observe that the model did not perform remarkably well in separating the labels into distinct clusters. This could be due to several factors, such as the inherent overlap in the features of the 'g' and 'b' signals, insufficient differentiation in the data, noise in data or the chosen value of K not being optimal for the given dataset.