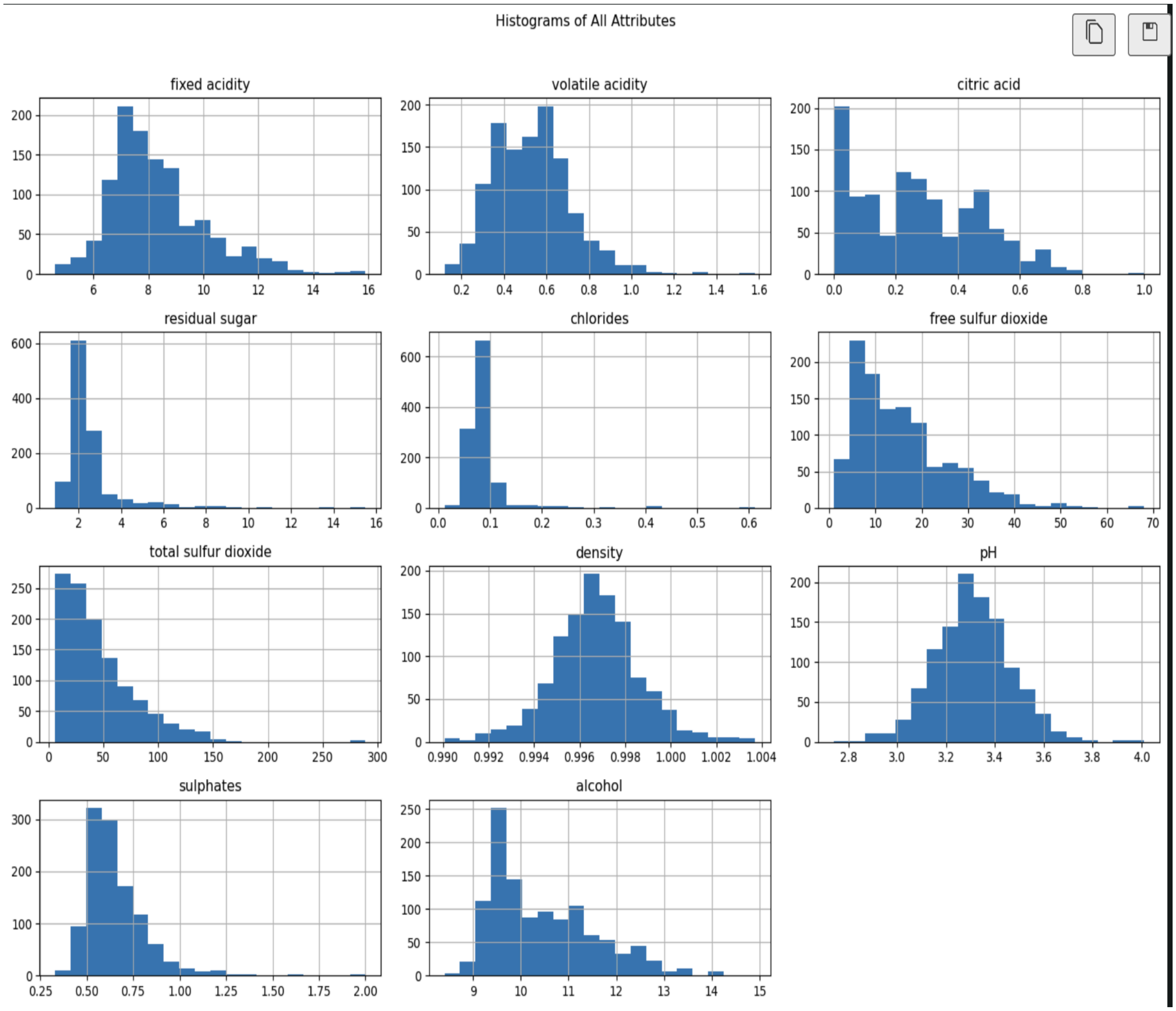
Lab Work 1: Feature Engineering

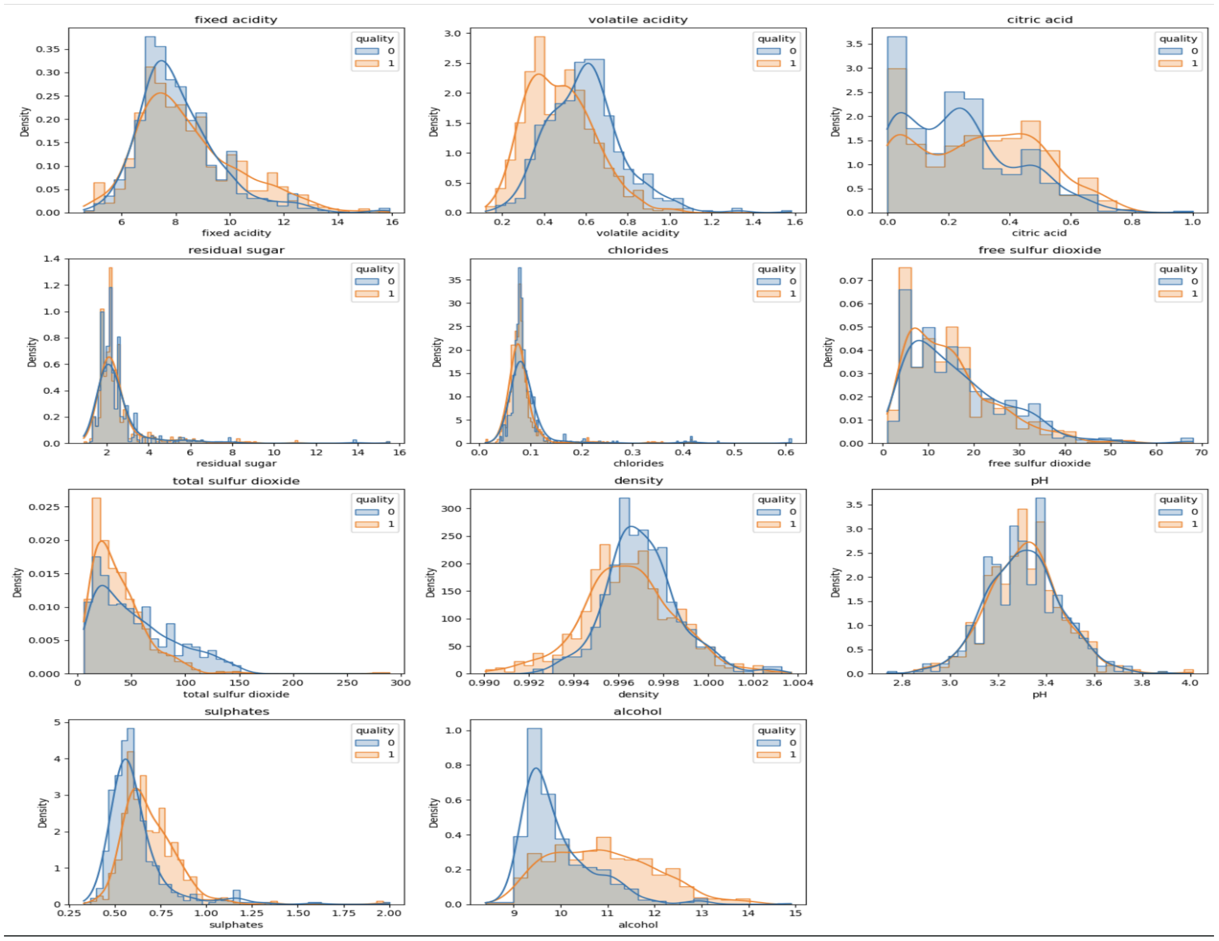
Abdul Bari, Aimal Javed, Syed Zain Anwer

Task 1: Classification with Unchanged Features

a) For the first task, we performed exploratory data analysis to plot the distribution of each one of the 11 features included in the dataset. This was done via the **plot_all_histograms** function, the result is shown below.



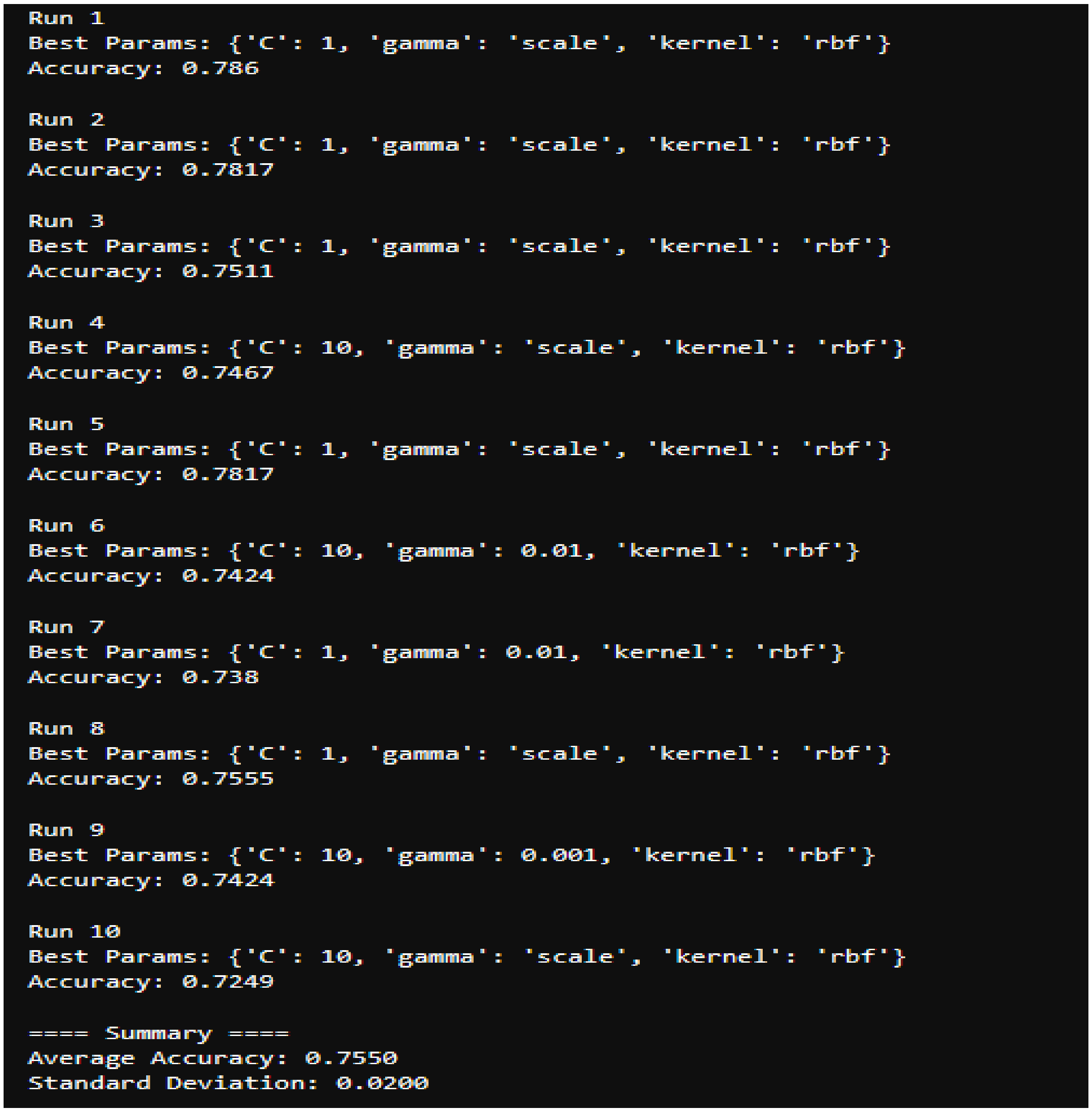
b) Histograms for each attribute with respect to Y are plotted using the **plot_histogram_by_class** function. This allowed us to visually verify if features were informative for classification.



Comment on Expected Classifier Performance (Task 1b)

- The histograms showed that many attributes have significant overlap between the two classes (Y = 0 and Y = 1), such as **volatile acidity, residual sugar**, and **chlorides**. This makes it difficult for a classifier to separate the classes based on these features alone.
- Since only a few features provide clear separation, and most others are noisy or overlapping, we can expect a moderate classification performance with an expected accuracy of around 70-80%.

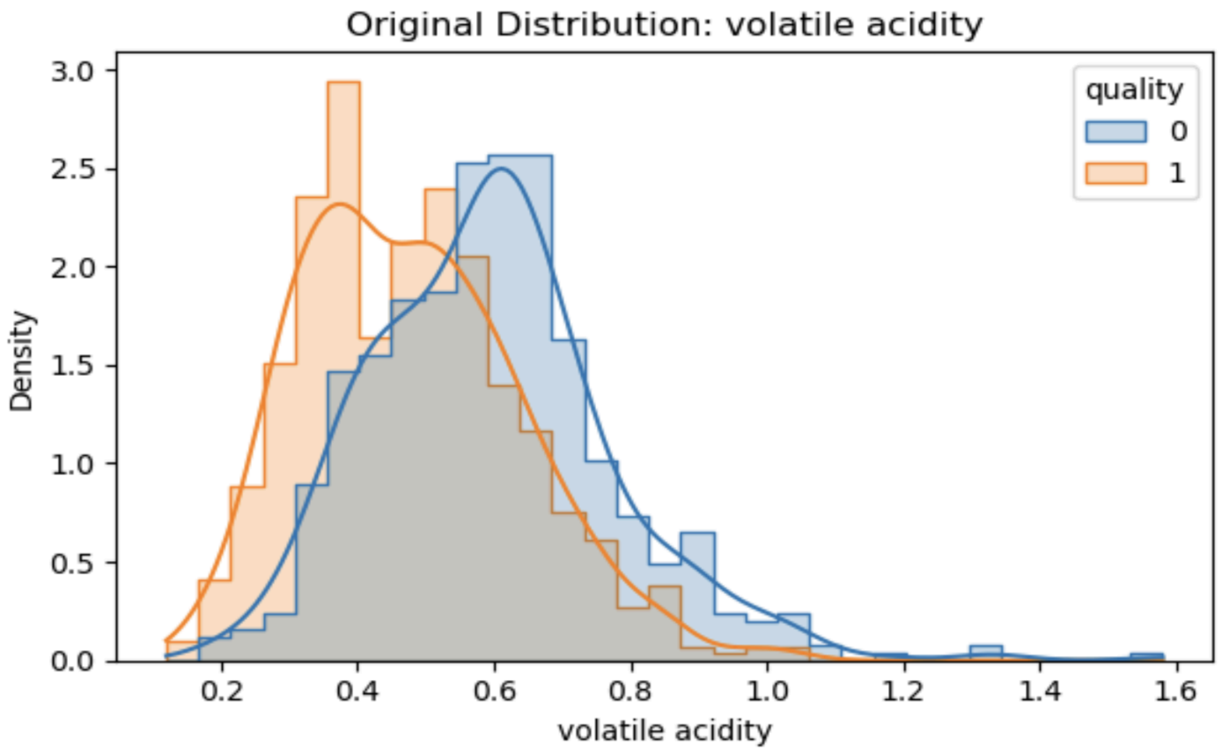
c) Following that visualization, the SVM classifier was employed for 10 epochs with a train-test split of (80/20), where each epoch used hyperparameter tuning using GridSearchCV. The grid parameter involved attempting combinations of C, gamma, and kernel parameters. Every execution recorded the best set of parameters and the best test accuracy. Finally, the overall mean accuracy and standard deviation across the ten epochs were calculated, giving a baseline classification ability with the entire, unabbreviated feature set. This exercise was done to determine how effectively the initial dataset runs without any feature reduction or alteration.



```
Run 1
Best Params: {'C': 1, 'gamma': 'scale', 'kernel': 'rbf'}
Accuracy: 0.786

Run 2
Best Params: {'C': 1, 'gamma': 'scale', 'kernel': 'rbf'}
Accuracy: 0.7817

Run 3
Best Params: {'C': 1, 'gamma': 'scale', 'kernel': 'rbf'}
Accuracy: 0.7511

Run 4
Best Params: {'C': 10, 'gamma': 'scale', 'kernel': 'rbf'}
Accuracy: 0.7467

Run 5
Best Params: {'C': 1, 'gamma': 'scale', 'kernel': 'rbf'}
Accuracy: 0.7817

Run 6
Best Params: {'C': 10, 'gamma': 0.01, 'kernel': 'rbf'}
Accuracy: 0.7424

Run 7
Best Params: {'C': 1, 'gamma': 0.01, 'kernel': 'rbf'}
Accuracy: 0.738

Run 8
Best Params: {'C': 1, 'gamma': 'scale', 'kernel': 'rbf'}
Accuracy: 0.7555

Run 9
Best Params: {'C': 10, 'gamma': 0.001, 'kernel': 'rbf'}
Accuracy: 0.7424

Run 10
Best Params: {'C': 10, 'gamma': 'scale', 'kernel': 'rbf'}
Accuracy: 0.7249

==== Summary ====
Average Accuracy: 0.7550
Standard Deviation: 0.0200
```

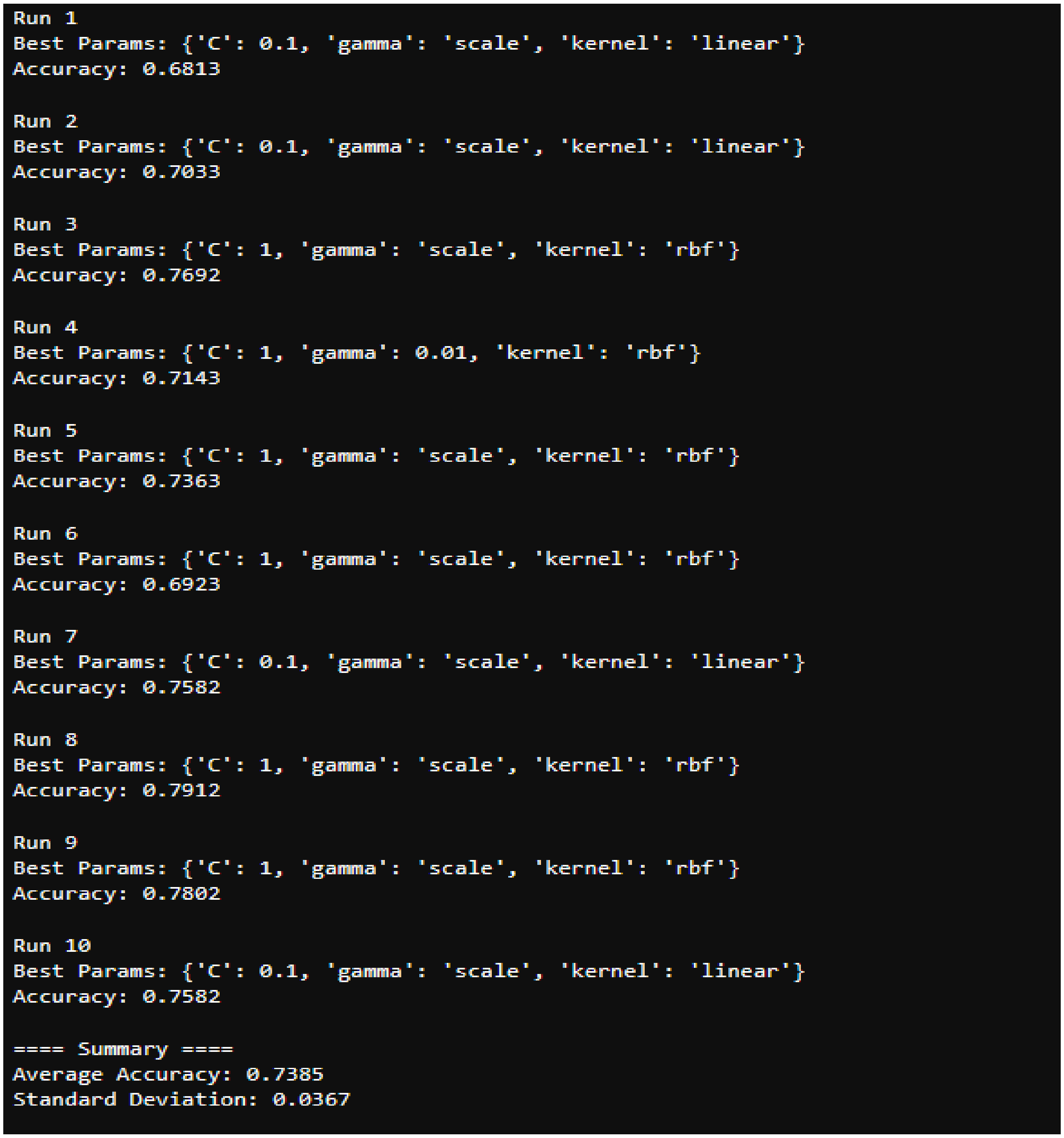
Task 2: Density Analysis by Sample Deletion

a) The second experiment tested the effect of modifying the dataset so that class separability for a specific attribute is increased. The focus was placed on volatile acidity, an attribute with significant overlap between the two classes in its distribution.



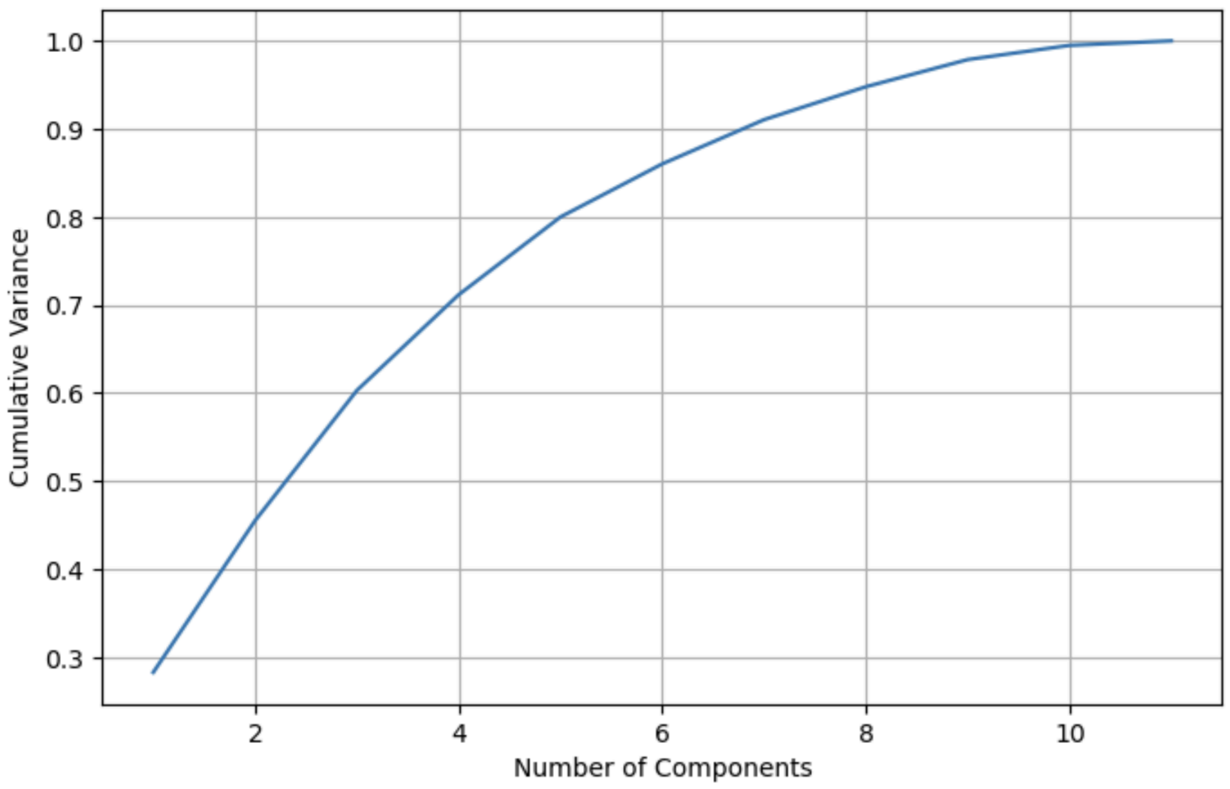
b) The **reduce_overlap** function removed samples in which volatile acidity was between 0.3 and 0.6 which was the range where the two classes overlapped the most. This selective instance removal reduced noise in the data and enhanced class separation for the feature.

c) The resulting *"cleaned"* dataset was input into the same SVM classification workflow used in Task 1. The results showed that filtering the dataset in this way led the **accuracy to decrease slightly** and the **variance to increase**. This is because removing samples reduces dataset size, which can hurt generalization.


```
Run 1
Best Params: {'C': 0.1, 'gamma': 'scale', 'kernel': 'linear'}
Accuracy: 0.6813

Run 2
Best Params: {'C': 0.1, 'gamma': 'scale', 'kernel': 'linear'}
Accuracy: 0.7033

Run 3
Best Params: {'C': 1, 'gamma': 'scale', 'kernel': 'rbf'}
Accuracy: 0.7692

Run 4
Best Params: {'C': 1, 'gamma': 0.01, 'kernel': 'rbf'}
Accuracy: 0.7143

Run 5
Best Params: {'C': 1, 'gamma': 'scale', 'kernel': 'rbf'}
Accuracy: 0.7363

Run 6
Best Params: {'C': 1, 'gamma': 'scale', 'kernel': 'rbf'}
Accuracy: 0.6923

Run 7
Best Params: {'C': 0.1, 'gamma': 'scale', 'kernel': 'linear'}
Accuracy: 0.7582

Run 8
Best Params: {'C': 1, 'gamma': 'scale', 'kernel': 'rbf'}
Accuracy: 0.7912

Run 9
Best Params: {'C': 1, 'gamma': 'scale', 'kernel': 'rbf'}
Accuracy: 0.7802

Run 10
Best Params: {'C': 0.1, 'gamma': 'scale', 'kernel': 'linear'}
Accuracy: 0.7582

==== Summary ====
Average Accuracy: 0.7385
Standard Deviation: 0.0367
```

Task 3: Principal Component Analysis (PCA)

a) Task 3 was focused on PCA dimensionality reduction. First, the feature space was normalized using StandardScaler because PCA is sensitive to input variable scale. PCA was then applied to the training data to compute the principal components and inspect their explained variance. A cumulative variance plot was generated to assist in deciding how many principal components to keep.



According to this trajectory, the top 8 components were selected that accounted for 95% of the total variance.

b) The next step was to use these components to map the train and test sets into the lower-dimensional PCA space. After this, classification was carried out using an SVM and grid search, this time with performance measured in the PCA-reduced space. The results for that are as follows:

```
Best Hyperparameters: {'C': 10, 'gamma': 'scale', 'kernel': 'rbf'}
Test Accuracy with PCA + SVM: 0.7773
```

Comparison to Task 1:
In Task 1, the baseline SVM model using all 11 original features achieved an accuracy of 0.7550. In this task, we applied PCA to reduce the dataset to 8 principal components based on cumulative variance. This PCA-transformed data gave a slightly better accuracy of 0.7773. While the improvement is small, it shows that PCA preserved most of the important information. Since PCA doesn’t use the target variable (it's unsupervised), it may have kept some less useful features and dropped some important ones. Still, it proved helpful in reducing the feature space without hurting performance.

Task 4.1: Recursive Feature Elimination (RFE)

a) For this task, we selected a Support Vector Machine (SVM) with a linear kernel and a high regularization parameter C=1000, along with gamma=0.1, as the estimator for Recursive Feature Elimination. This setup helps guide the feature selection process by emphasizing the importance of features that strongly influence the classification decision.

b) The RFE technique was applied to the training set only, with the goal of selecting the top 8 most important features. This was done using scikit-learn’s RFE class, which recursively removes the least important features based on the estimator’s learned weights. Before applying RFE, the data was standardized to ensure fair weighting across features.

c) After the RFE process, we extracted the mask of selected features using selector.support_ and retrieved the actual column names from the original DataFrame. These features represent the most relevant variables for predicting wine quality.

d) We then used the 8 selected features to reduce both the training and test datasets to a consistent feature space. Using these reduced datasets, we trained an SVM model with grid search over 10 hyperparameter combinations (C, gamma, and kernel). Note that while RFE was done with a linear kernel, the grid search also explored rbf to compare performance.
After fitting, the best-performing parameters were extracted and the model was evaluated on the test set. Accuracy was computed to quantify how well the model performed after feature selection:

Results:

```
Best Hyperparameters: {'C': 0.1, 'gamma': 'scale', 'kernel': 'linear'}
Test Accuracy with RFE + SVM: 0.7598
```

Comparison to Task 1:
In Task 1, the SVM model using all features reached an accuracy of 0.7550. In this task, we used RFE, a supervised method, to select the 8 most important features based on how well they

help predict the target. The SVM trained on this reduced set achieved a higher accuracy of 0.7598. This shows that removing less useful features can actually improve performance. Since RFE considers the target variable during selection, it was better at keeping only the most relevant data, making the model both simpler and more accurate.

Conclusion

Each of the tasks in this lab progressively added a new aspect of feature engineering and its impact on binary classification. Task 1 established a baseline performance with all the features. Task 2 illustrated that simple data cleaning (removing duplicate samples) can dramatically enhance model performance. Task 3 applied PCA to perform dimensionality reduction, which worked pretty well but performance declined slightly as the method is unsupervised. Task 4 applied RFE, a supervised technique, to shrink the feature space in a smart manner while preserving prediction capability, ultimately performing as well as or even better than the full model. Through these experiments, we observed how preprocessing techniques, especially those that include target-guided feature selection, can render models more interpretable, conserve computation, and even improve accuracy.

Feedback

This lab was intuitive and relatively easy to follow, because it offered a clear, hands-on approach to feature engineering techniques. The lab work helped us understand how different methods like PCA and RFE impact model performance. Overall, an engaging first experience with structured preprocessing.