Lab Work 1: Feature Engineering

Abdul Bari, Aimal Javed, Syed Zain Anwer

Task 1: Classification with Unchanged Features

For the first task, the team began by performing exploratory data analysis to plot the distribution of each one of the 11 features included in the dataset. This was done via individual histograms for all features (**plot_all_histograms**), and additional informative histograms showing the distribution of each feature split by the binary target quality (**plot_histograms_by_class**). This allowed us to visually verify if features were informative for classification, i.e., features with good distinction between quality=0 and quality=1 should be informative towards model accuracy. Following that visualization, the SVM classifier was employed for 10 epochs with different random train-test splits (80/20), where each epoch employed 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.

Task 2: Density Analysis by Sample Deletion

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. 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. The resulting *“cleaned”* dataset was input into the same SVM classification workflow used in Task 1: split the data in a train and test set, with normalization done via StandardScaler, followed by model evaluation and hyperparameter tuning. The results showed that filtering the dataset in this way led to improved classifier performance on average, with higher accuracy as well as lower variability between runs. The task demonstrated how meticulous data cleansing and the selection of useful attributes can significantly improve classification outcomes without actually modifying the underlying modeling strategy.

Task 3: Principal Component Analysis (PCA)

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. 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 best hyperparameter and test accuracy were indicated. The accuracy achieved from this PCA-based approach was comparable to or slightly inferior to that of Task 1, as would be intuitive since PCA is an unsupervised operation (it reduces features by variance, not by relevance to target label). What this means is that some of the features of highest significance in classification might have been downgraded or eliminated during the PCA transformation.

Comparison to Task 1:

In Task 1, the baseline SVM model using all 11 original features achieved an accuracy of 0.7550. In Task 3, 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 0.7555. 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)

In this task, the focus was on supervised feature selection using Recursive Feature Elimination (RFE). Here, we used an SVM with a linear kernel and high regularization parameter value (C=1000) as the estimator to regulate the feature removal. RFE was used on the training set only, following best practices, to select the most significant 8 features for classification. Once the most important features were established, these same 8 features were then retrieved from the test set to allow input space consistency. This reduced-dimensional feature set was then used to train a further SVM classifier, for which a grid search was performed over 10 sets of varying C, gamma, and kernel parameter values, however only the linear kernel was ultimately employed. The best hyperparameters and resulting test accuracy were returned. Surprisingly, this approach produced either a similar or a slightly better accuracy than full-feature baseline in Task 1. This makes sense since RFE is a supervised method, it exploits label y to choose features, and tends to produce better performance than unsupervised methods such as PCA.

Comparison to Task 1:
In Task 1, the SVM model using all features reached an accuracy of 0.7550. In Task 4, 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.