Lab 4: Regression And Classification

Bora ILCI & Kaan Emre KARA

April 29, 2025

1. Introduction and Dataset Description

This report analyzes the wine dataset, which contains chemical properties of wines derived from three different cultivars. The task is to classify wines into their respective types (classes) based on their chemical properties. The dataset includes 13 features such as alcohol content, flavanoids, and color intensity, with each instance belonging to one of three wine classes.

2. Visualization Management and Design

For this project, we implemented an object-oriented approach to visualization using a custom **PlotManager** class. This design pattern offers several advantages:

- Consistent styling and formatting across all visualizations
- Centralized management of plot generation and storage
- Support for both individual plot files and comprehensive multi-page reports
- Interactive gallery viewer for exploring visualizations

2.1 PlotManager Implementation

The PlotManager class provides an elegant solution for handling multiple visualization tasks:

```
class PlotManager:
    """
    A class to manage all visualization aspects of the wine classification project.
    Uses an object-oriented approach to handle plotting and saving figures.
    """

    def __init__(self, output_dir="", save_individual_plots=False):
        """Initialize PlotManager with optional output directory"""
        self.output_dir = output_dir
        # Set a consistent style for all plots
        sns.set_style("whitegrid")
        # Set a consistent color palette
        self.colors = sns.color_palette("viridis", 10)
        # Set default figure size
        plt.rcParams["figure.figsize"] = (10, 6)
        # Store all figures for multi-page display
        self.figures = []
        self.figure_titles = []
        # Whether to save individual plot files
        self.save_individual_plots = save_individual_plots
```

This implementation uses context managers for clean figure handling and converts matplotlib figures to PIL Images for efficient storage and display in the gallery:

> The PlotManager provides a unified interface for creating, storing, and displaying all data visualizations, ensuring consistent styling and efficient workflow throughout the analysis process.

3. Data Preparation and Feature Analysis

The wine dataset comprises 13 chemical features. Before model training, we conducted a thorough analysis to understand feature distributions, correlations, and importance using our PlotManager class.

3.1 Feature Distributions

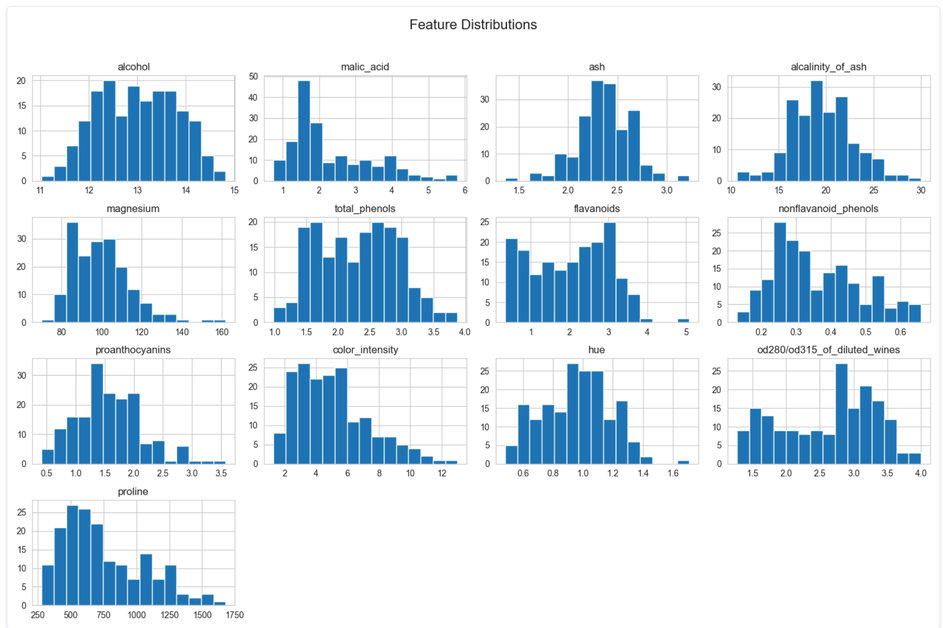

Figure 1: Histogram visualizations of the feature distributions in the wine dataset

The histograms reveal varying distributions across features. Some features like alcohol and proline appear to have multimodal distributions, potentially indicating their discriminative power between wine classes. Our PlotManager's `plot_feature_distributions` method efficiently generates these histograms with consistent styling.

3.2 Feature Correlation Analysis

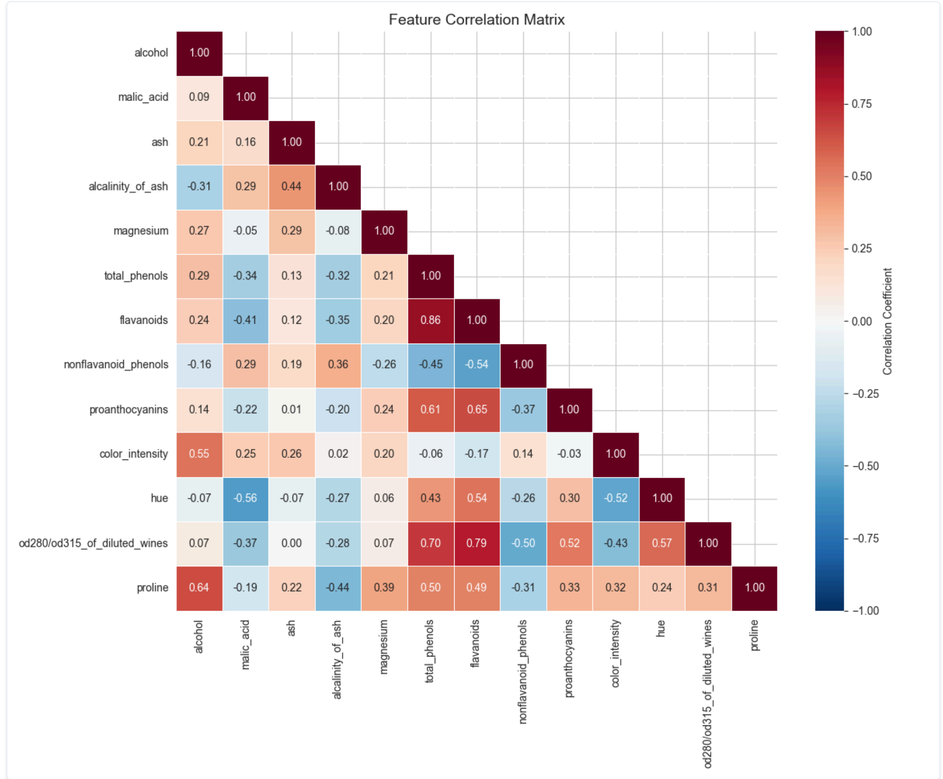

Figure 2: Correlation matrix showing relationships between features

The correlation matrix reveals several moderately to strongly correlated features. For instance, there is a strong positive correlation between flavanoids and total phenols (0.86), and between color intensity and OD280/OD315 (0.65). The `plot_correlation_matrix` method in our PlotManager uses an optimized approach that displays only the lower triangle of the matrix for improved readability while maintaining all correlation information.

While these correlations suggest potential redundancy, we chose to retain all features for the following reasons:

- The dataset is relatively small (178 samples), so the risk of overfitting due to dimensionality is low
- Both selected algorithms (SVM and Random Forest) are generally robust to correlated features
- Preserving all features allows the models to capture potentially useful subtle relationships

3.3 Feature Importance Analysis

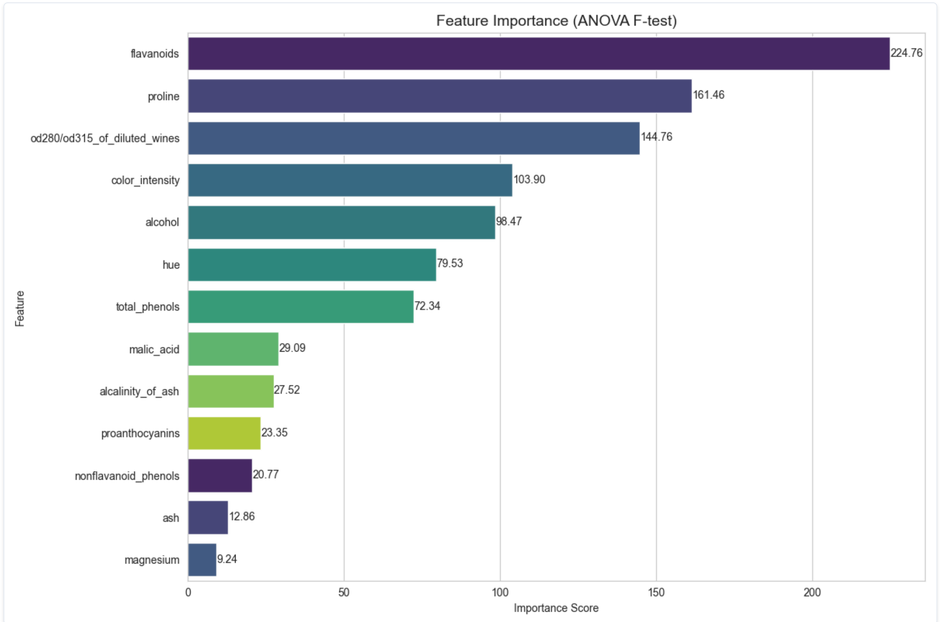

Figure 3: Feature importance based on ANOVA F-test

The ANOVA F-test revealed several highly discriminative features, with proline, flavanoids, and color intensity emerging as the most important predictors for wine classification. Our `plot_feature_importance` method automatically sorts features by importance and adds value labels to the bars for clearer interpretation.

While we could have considered dimensionality reduction based on this analysis, we opted to retain all features since:

> Even features with lower F-scores might capture nuanced relationships valuable for classification. Both SVM and Random Forest models can effectively handle the full feature set, with Random Forest providing its own measure of feature importance during training.

3.4 Data Preprocessing

We applied StandardScaler normalization to all features before model training. This step is crucial because:

- Features have different units and scales (e.g., alcohol percentage vs. mg/L measurements)
- Support Vector Machines are sensitive to feature scaling
- Normalization ensures all features contribute proportionally to distance calculations

4. Model Selection

For the wine classification task, we selected two models:

4.1 Support Vector Machine (SVM)

We chose SVM for the following reasons:

- Effective for high-dimensional data with clear margins between classes
- Strong performance in classification tasks with moderate-sized datasets
- Flexibility through kernel functions to capture non-linear relationships
- Robust to overfitting when properly regularized

4.2 Random Forest

We selected Random Forest as our second model because:

- Ensemble method that combines multiple decision trees for robust predictions
- Handles both linear and non-linear relationships effectively
- Less sensitive to outliers than many other algorithms
- Provides built-in feature importance measures
- Less prone to overfitting than single decision trees

5. Model Training and Parameter Tuning

We used 4-fold cross-validation to evaluate different parameter configurations for both models, with accuracy as our evaluation metric. This approach helps identify optimal parameters while reducing the risk of overfitting.

5.1 SVM Parameter Tuning

We explored various combinations of SVM parameters:

Parameters	Mean Cross-Validation Accuracy	Standard Deviation
C=1, kernel=rbf, gamma=scale	0.9716	±0.0218
C=10, kernel=rbf, gamma=scale	0.9789	±0.0214
C=100, kernel=rbf, gamma=scale	0.9789	±0.0214
C=1, kernel=rbf, gamma=auto	0.9507	±0.0303
C=10, kernel=rbf, gamma=auto	0.9648	±0.0320
C=1, kernel=linear	0.9648	±0.0204
C=10, kernel=linear	0.9648	±0.0204

The best performance was achieved with C=10, kernel=rbf, and gamma=scale, suggesting that a non-linear decision boundary with moderate regularization works best for this dataset.

5.2 Random Forest Parameter Tuning

We explored various combinations of Random Forest parameters:

Parameters	Mean Cross-Validation Accuracy	Standard Deviation
n_estimators=50, max_depth=None	0.9648	±0.0320
n_estimators=100, max_depth=None	0.9716	±0.0218
n_estimators=100, max_depth=10	0.9789	±0.0214
n_estimators=200, max_depth=None	0.9716	±0.0218
n_estimators=200, max_depth=15	0.9716	±0.0218
n_estimators=300, max_depth=None	0.9716	±0.0218
n_estimators=100, max_depth=5, min_samples_split=5	0.9648	±0.0312

The best performance was achieved with n_estimators=100 and max_depth=10, suggesting that a moderate number of trees with controlled depth provides good generalization for this dataset.

6. Model Evaluation and Comparison

6.1 Performance Metrics

After training models with the best parameters identified through cross-validation, we evaluated their performance on the test set, which constituted 20% of the original dataset.

6.2 Random Forest Feature Importance

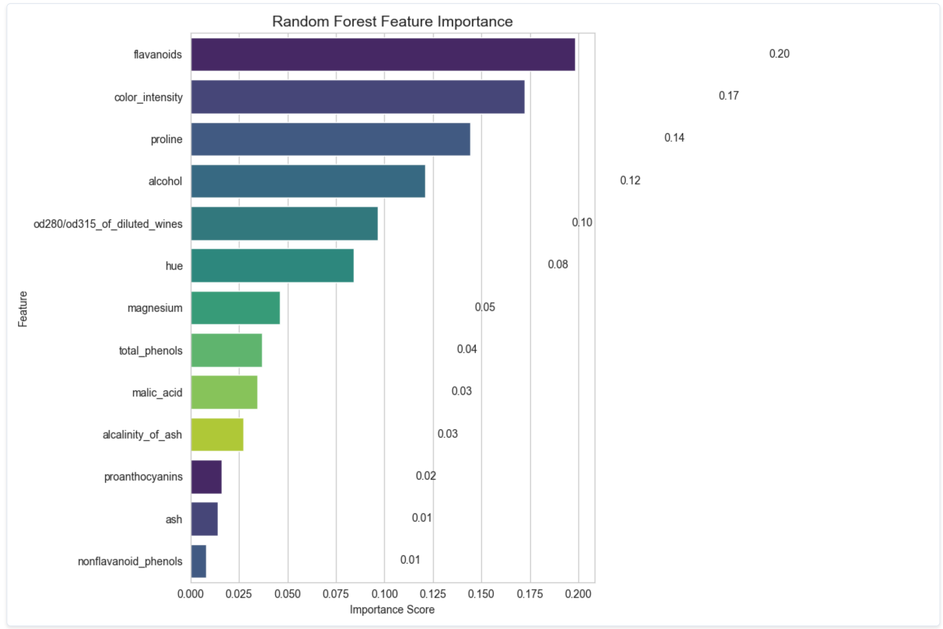

Figure 4: Feature importance as determined by the Random Forest model

The Random Forest's internal feature importance measure largely aligns with the ANOVA F-test results, confirming the relevance of features like proline, flavanoids, and color intensity for wine classification. Our PlotManager uses the same plotting function for both ANOVA and Random Forest importance, ensuring consistent visualization for easy comparison.

6.3 Confusion Matrices

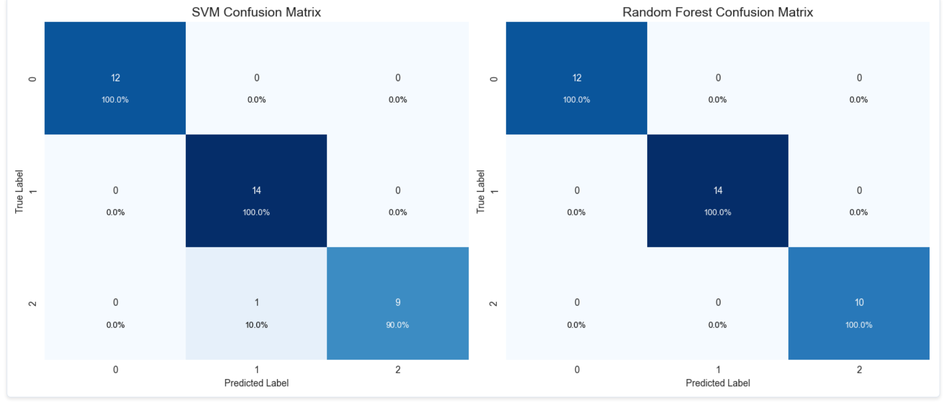

Figure 5: Confusion matrices for SVM and Random Forest models

The confusion matrices reveal the classification performance across the three wine classes. Both models perform exceptionally well, with very few misclassifications. Our `plot_confusion_matrices` method in PlotManager displays both raw counts and percentages in each cell, with a color-coded approach that makes interpretation intuitive.

6.4 Result Consolidation

Our PlotManager class includes methods to compile all visualizations into a comprehensive PDF report:

```
def save_all_figures(self, output_file="wine_analysis_report.pdf"):
    """Save all figures to a multi-page PDF"""
    # Since we're storing PIL Images, not matplotlib figures,
    # we need to create a PDF using PIL's functionality
    if not self.figures:
        print("No figures to save.")
        return

    # Create a list to hold the converted images
    images = []

    # Convert the first PIL image to RGB mode
    first_img = self.figures[0].convert("RGB")

    # Convert the remaining images (if any) to RGB mode and append to the list
    if len(self.figures) > 1:
        images = [img.convert("RGB") for img in self.figures[1:]]

    # Save the first image and append the remaining images to the PDF file
    first_img.save(
        f"{self.output_dir}{output_file}", save_all=True, append_images=images
    )
    print(f"Saved {len(self.figures)} figures to {output_file}")
```

We also implemented an interactive gallery viewer that allows for dynamic exploration of all generated visualizations.

6.5 Model Comparison

Both models achieved high accuracy on the test set:

- SVM with optimal parameters (C=10, kernel=rbf, gamma=scale): 0.9722 accuracy
- Random Forest with optimal parameters (n_estimators=100, max_depth=10): 0.9722 accuracy

The identical performance suggests that the wine classes are well-separated in the feature space and can be effectively distinguished by both linear (SVM with linear kernel) and non-linear (SVM with RBF kernel and Random Forest) approaches.

7. Conclusion

Our analysis of the wine dataset demonstrates that:

- The chemical properties of wine provide strong signals for classifying wines into their respective types
- Feature normalization is essential for this dataset due to the varying scales of measurements
- Object-oriented design principles enhance visualization workflows, leading to more consistent and maintainable code
- Both SVM and Random Forest models perform excellently on this task, with careful parameter tuning further enhancing their performance
- Our PlotManager class provides a reusable framework for future analytical projects, with built-in support for interactive visualization exploration
- Features like proline, flavanoids, and color intensity consistently emerge as the most important predictors across different analysis methods

This project illustrates the effectiveness of machine learning approaches for wine classification based on chemical properties, potentially aiding in quality control and authentication in the wine industry. Additionally, our visualization framework demonstrates how proper software engineering practices can enhance data science workflows.