Wine Classification Workshop — Step■by■Step Guide

This guide walks you through the learner notebook, cell by cell. Follow the steps in order and type code only where you see TODO/___ in the notebook.

0) Open the Learner Notebook

Run the first imports cell. If a plot doesn't show, re-run the cell that created it and ensure you ran the imports cell first.

1) Load the Dataset

In the cell titled "Load the dataset and create df with a 'target' column":

a) Create the dataset object:

wine = load_wine()

b) Build a DataFrame from wine.data and wine.feature_names.

df = pd.DataFrame(wine.data, columns=wine.feature_names)

c) Add the target labels:

df['target'] = wine.target

■ Checkpoint: You should see a table with numeric columns and a 'target' column.

2) Explore the Data (EDA)

In the EDA cell:

a) Print shape:

print('Shape:', df.shape)

b) Show summary statistics:

df.describe()

c) Plot histograms for all features:

df.hist(figsize=(10, 8)); plt.tight_layout(); plt.show()

Tip: If the histograms look cramped, try a larger figsize, e.g., (12, 10).

3) Visualize with PCA (2D)

In the PCA cell:

a) Separate features from target:

X = df.drop('target', axis=1)

b) Create a PCA object for 2 components:

pca = PCA(n_components=2)

c) Fit and transform X:

components = pca.fit_transform(X)

d) Scatter plot the two components colored by class:

plt.scatter(components[:,0], components[:,1], c=df['target'])

plt.xlabel('PC1'); plt.ylabel('PC2'); plt.title('Wine Data PCA (2D)'); plt.show()

Question to consider: Do classes appear well separated? Which overlap?

4) Create Train/Test Split

In the split cell:

a) Define X and y if not already:

X = df.drop('target', axis=1); y = df['target']

b) Split with a 70/30 train/test and fixed seed (42):

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

Note: random_state=42 ensures you and your classmates see comparable results.

5) Train a Decision Tree

In the training cell:

a) Start with a shallow tree to avoid overfitting:

dt = DecisionTreeClassifier(max_depth=3, random_state=42)

b) Fit the model:

dt.fit(X_train, y_train)

6) Evaluate the Model

In the evaluation cell:

a) Predict on the test set:

y_pred = dt.predict(X_test)

b) Print a classification report:

print(classification_report(y_test, y_pred))

c) (Optional) Include target names:

from sklearn.datasets import load_wine; print(classification_report(y_test, y_pred, target_names=load_wine().target_names))

7) Plot a Confusion Matrix

In the confusion matrix cell:

a) Compute the matrix:

cm = confusion_matrix(y_test, y_pred)

b) Display the matrix with labels:

disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=dt.classes_)

disp.plot(cmap='Blues', values_format='d'); plt.title('Decision Tree - Confusion Matrix'); plt.show()

8) Extensions (Optional)

• Try deeper trees (increase max_depth) and compare F1/accuracy.

• Switch to criterion='entropy' and compare results.

• Add feature importance: pd.Series(dt.feature_importances_, index=X.columns).sort_values(ascending=False)

You’re done! Discuss which features seem most important and how PCA separation matches model performance.