CSDS 313 Homework 4: Clustering, Dimensionality Reduction, and Predictive Modeling

Name: Camden Larson Case ID: csl117

Date: November 25 2025

1 Task 1: Clustering and Dimensionality Reduction

1.1 Dataset + Preprocessing (brief)

Key steps / details:

- Files used: "p1-congress-1984-votes.csv"
- Data shape: 435×16 votes matrix; party vector size
- Encoding: -1/0/1 semantics
- Preprocessing for PCA: confirmed that any missing values were replaced with 0.0
- Random seed(s): 42

1.2 Part A: PCA

1.2.1 A1. Cumulative variance explained vs. k

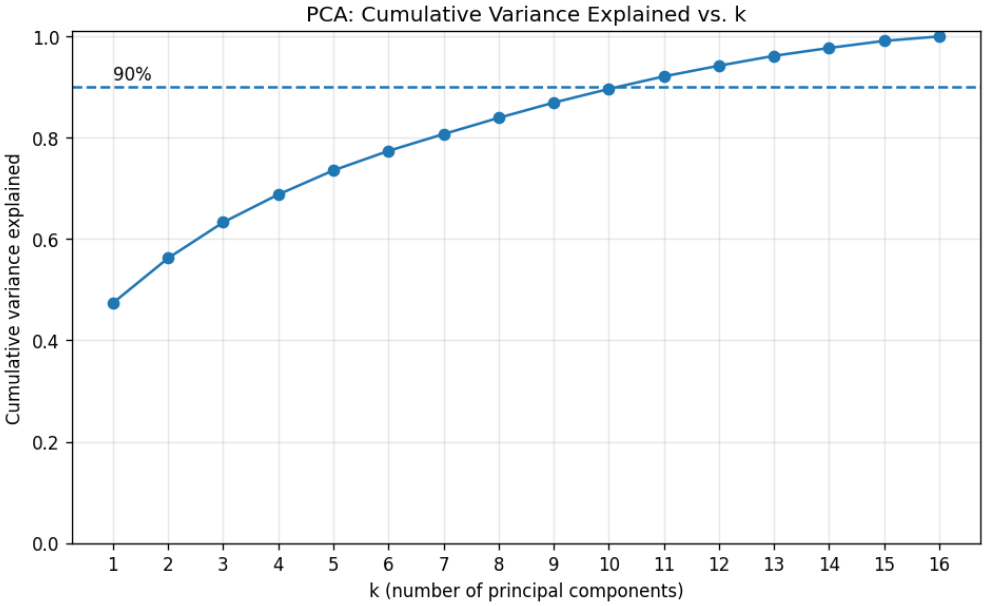


Figure 1: Cumulative variance explained by top k PCs.

Answer:

- Chosen number of PCs sufficient to summarize data: 11
- Criterion used: The 90% threshold is sufficient for this dataset, which means the least number of principal components k that is greater than 90% cumulative variance explained, in this case 11, are enough to summarize the data.

1.2.2 A2. Scatter plots on first 3 PCs (colored by party)

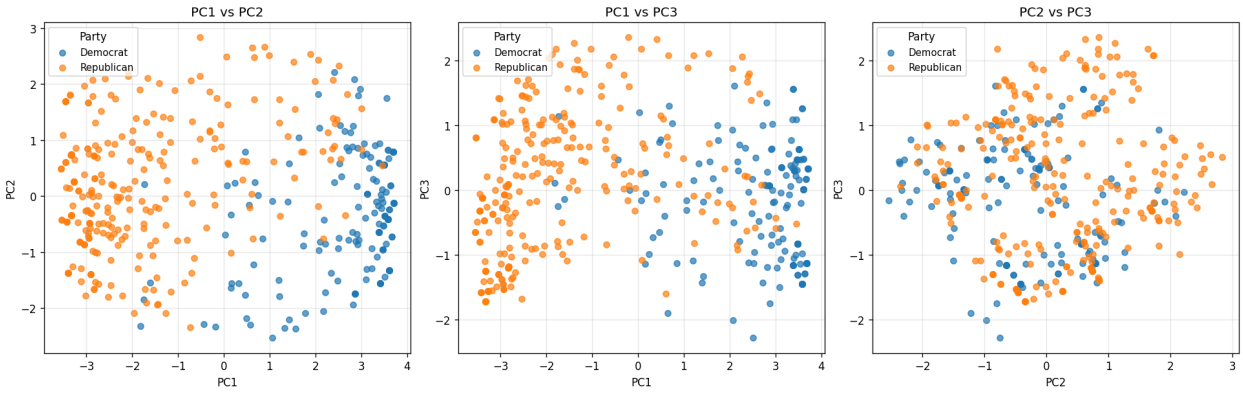


Figure 2: PCA projections colored by party affiliations.

Answer:

- Best-separating PC pair: PC1 vs PC2
- Do same-party members appear clustered by votes? Yes, in all three plots there is clear clustering by party affiliation. In the first two plots (PC1 vs PC2 and PC1 vs PC3) especially, you can see a divide between the democrat and republican votes with republican votes dominating the top left quadrant and democrat votes dominating the bottom right quadrant. In the third image it is a little less clear, but still noticeable that the democrat votes were closer to each other than they were to republican votes, which were closer to themselves.

1.3 Part B: Clustering on 16-dimensional votes

1.3.1 B1. Clustering algorithm + distance function

Answer:

- Algorithm choice: K-means clustering.
- Parameters: $k = 2$, best of 20 random initializations.
- Distance function: Euclidean distance. In the assignment step, each point is assigned to the closest centroid under squared Euclidean distance

$$d^2(x, \mu) = \sum_{j=1}^{16} (x_j - \mu_j)^2,$$

- Clustering quality score: K-means objective (within-cluster sum of squared errors, SSE),

$$\text{SSE} = \sum_{i=1}^{n} \left\| x_i - \mu_{c(i)} \right\|_2^2 .$$

- Implementation notes: Implemented squared Euclidean distance and the K-means update steps from scratch using NumPy.

1.3.2 B2. Visualization on top two PCs

Answer: Figure 3 shows the two K-means clusters (computed in the original 16-D vote space) projected onto PC1–PC2 for visualization. The two groups appear visually separated with a few misclassifications for each party.

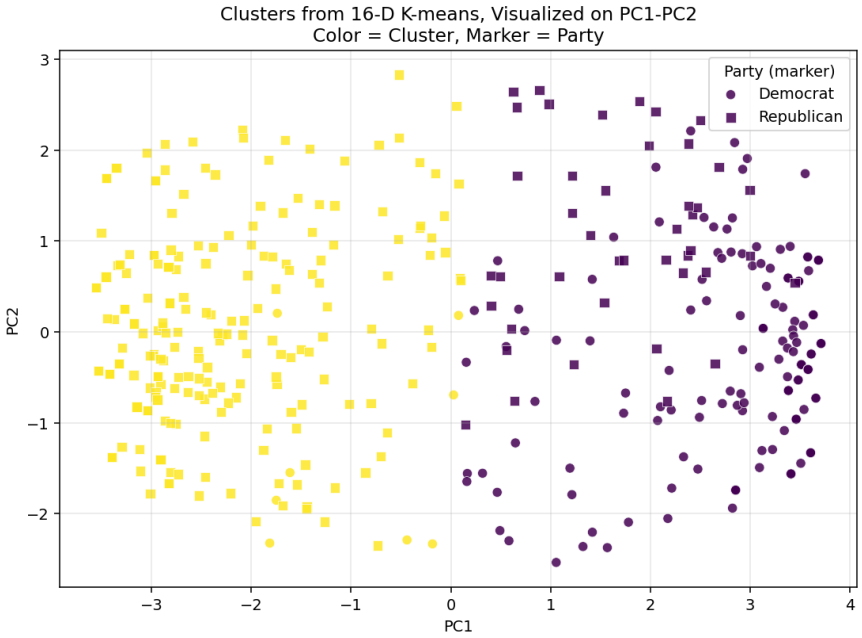


Figure 3: K-means clusters (fit on 16-D votes) visualized on the first two principal components. Colors indicate cluster assignment; marker shape indicates party.

Answer:

- Visual separation: The clusters are largely separated on PC1–PC2 (roughly left vs. right), with some overlap due to outliers.
- Agreement with party affiliations (contingency table):
  - Democrats: 160 in Cluster 1, 8 in Cluster 2.
  - Republicans: 43 in Cluster 1, 224 in Cluster 2.
- Overall agreement accuracy: 0.883
- Per-party accuracy:
  - Democrat accuracy: 0.952 (160/168).
  - Republican accuracy: 0.839 (224/267).

1.3.3 B3. Permutation test for clustering significance

Key steps / details:

1. Score: use the K-means objective (SSE), where smaller values indicate tighter clustering.
2. Compute the observed score on the original 16-D vote dataset: $S_{\text{obs}} = 3778.339$.
3. For $b = 1, \ldots, B$ permutations:
   - Permute each congress member's 16 votes across issues. This preserves each member's distribution of Reject/Neutral/Accept votes but destroys structure tied to specific issues.

- Re-run K-means with the same settings and record the permuted score S_b.

4. Compute the permutation-test p-value (lower tail since smaller SSE is 'more clustered'):

$$p = \frac{1 + \sum_{b=1}^{B} \mathbf{1}[S_b \leq S_{\text{obs}}]}{B + 1}.$$

Answer:

- Number of permutations: $B = 500$.
- Observed score: $S_{\text{obs}} = 3778.339$.
- Null distribution summary: mean SSE $= 6115.635$, std. $= 8.635$.
- p-value: $p = 0.0020$.
- Conclusion: Yes, the original dataset is significantly clustered. The observed SSE is far smaller than the SSE values produced by row-wise-permuted datasets, and the permutation test yields $p = 0.0020 < 0.05$, indicating such strong clustering is very unlikely under the null model.

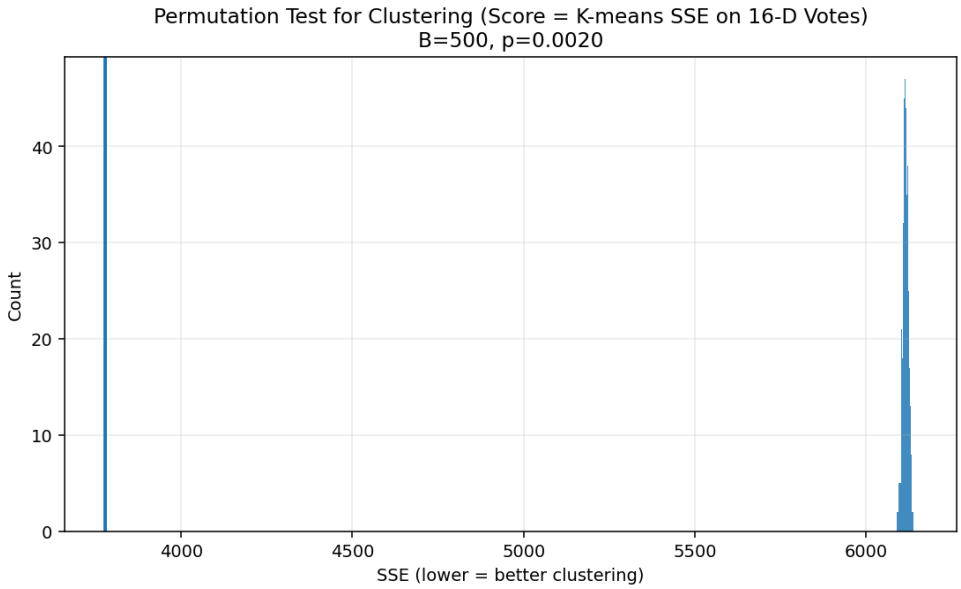


Figure 4: Permutation-test null distribution of K-means SSE on row-wise permuted votes, with the observed SSE marked. Lower SSE indicates stronger clustering.

1.4 Part C: Agreement with party affiliations + PCA-based clustering

1.4.1 C1. Quantify agreement on 16D-votes clustering

Answer:

- Agreement metric: Mutual Information (MI) and Normalized Mutual Information (NMI) between cluster membership and party affiliation.
- Computed value: MI(Cluster; Party) $= 0.346055$ nats, NMI $= 0.509754$.

- Brief interpretation: The NMI is moderately high (about 0.51 on a 0–1 scale), indicating that the unsupervised clusters capture a substantial amount of the party information, though not perfectly (there is still noticeable overlap/misclassification).

1.4.2 C2. Repeat clustering using only first two PCs

Answer:

- PCA dimensions used: PC1 and PC2 (these together explain 0.5624 of the total variance).
- Clustering settings: Same algorithm and settings as before (K-means, $k = 2$, Euclidean distance / squared Euclidean objective; best of 20 initializations), but run on the 2D PC representation instead of the full 16D votes.
- Agreement metric value: $\mathrm{MI}(\mathrm{Cluster}; \mathrm{Party}) = 0.329045$ nats, $\mathrm{NMI} = 0.485188$.

1.4.3 C3. Compare: 16 votes vs. first two PCs

Answer:

- Which agrees more with party affiliations?: Clustering on all 16 votes agrees more (NMI = 0.509754 vs. 0.485188 on PC1–PC2; accuracy 0.883 vs. 0.878).
- Why might this happen?: Using only PC1-PC2 can denoise the data by keeping the highest-variance directions, but it also discards information contained in lower-variance PCs. Some party-related signal can live in those lower-variance dimensions, so clustering in the full 16D vote space can retain more information relevant to party affiliation, achieving slightly higher agreement.

2 Task 2: Predictive Modeling (Wine Quality Classification)

2.1 Dataset + Setup (brief)

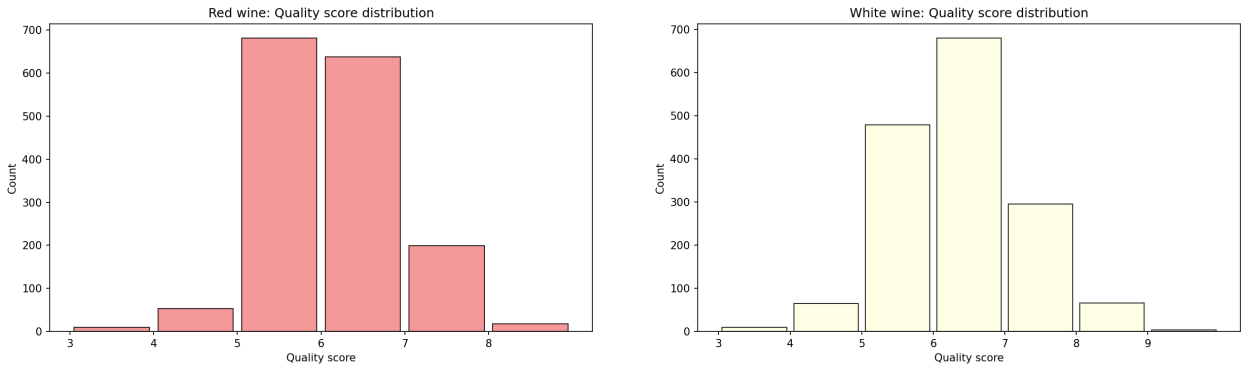
Key steps / details:

- Wine types: red (1599), white (4898)
- Features used: all 11
- Train/test split: 80/20
- Downsampling: Only used first 1599 white samples

2.2 Part A: Label design via discretization

2.2.1 A1. Quality score distributions (before discretization)

Both red and white appear to have 5 and 6 as the most prominent scores in the distribution, with red being fairly centered around 5/6 with a min of 3 and a max of 8 and white being skewed a bit towards the higher quality values with a min of 3 and a max of 9.



(a) Red wine quality distribution

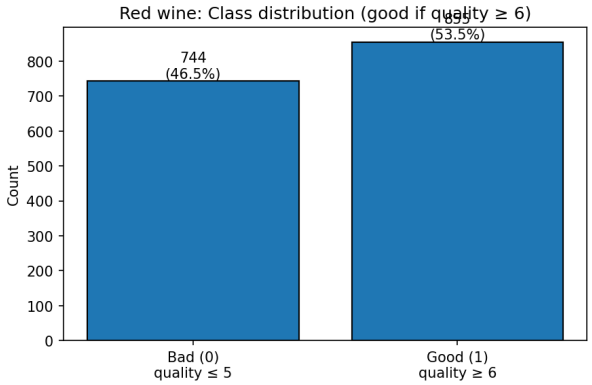
(b) White wine quality distribution

Figure 5: Original quality score histograms.

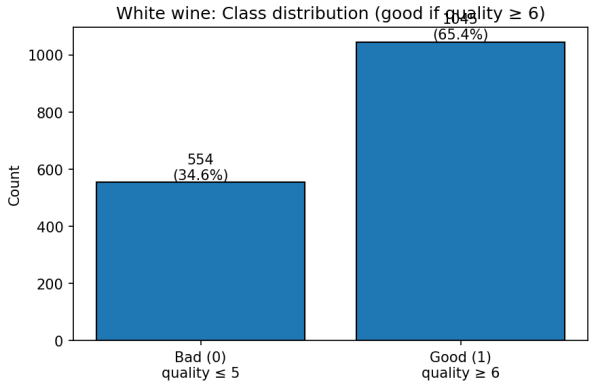
2.2.2 A2. Discretization rule + class balance

Answer:

- Label rule: good if quality is 6 or greater, bad if quality is 5 or less.
- Number of classes: 2
- Applied consistently across red and white: yes



(a) Red: class distribution



(b) White: class distribution

Figure 6: Class distributions after discretization.

2.2.3 A3. Rationale

A binary classification system seemed to be the best for this problem so I decided to keep the classes limited at 2. After looking at the quality distributions visually and considering that on a 1-10 scale 6 and higher is usually above average, I decided to separate the bad and good wine at the 5/6 line. In order to effectively train and test cross domain I decided to keep the separation the same on both wines.

2.3 Part B: Train classifiers + in-/cross-domain evaluation

2.3.1 B0. Models selected

Answer:

- Model 1: Logistic Regression (with standardization; class_weight = balanced)
- Model 2: Random Forest (class_weight = balanced)
- Model 3: MLP (1 hidden layer; standardization; early stopping)
- Key hyperparameters:
  - Logistic Regression: solver=lbfgs, max_iter=5000, class_weight=balanced
  - Random Forest: n_estimators=500, min_samples_leaf=2, class_weight=balanced
  - MLP: hidden_layer_sizes=(32,), max_iter=500, early_stopping=True, alpha=1e-4, batch_size=64

2.3.2 B1. Evaluation protocol

Key steps / details:

- Split: stratified train/test split (80/20) for each wine type (red, white).
- In-domain testing: train red → test red; train white → test white.
- Cross-domain testing: train red → test white; train white → test red.
- Metrics reported: Accuracy, Balanced Accuracy, Precision, Recall, and F1 (positive class = "good" = 1).

2.3.3 B2. Results tables

Answer:

Table 1: Performance metrics (in-domain). Positive class is "good" (quality ≥ 6).

Train → Test	Model	Acc.	Bal. Acc.	Prec.	Rec.	F1
Red → Red	Logistic Regression	0.7469	0.7502	0.8000	0.7018	0.7477
Red → Red	Random Forest	0.7938	0.7949	0.8261	0.7778	0.8012
Red → Red	MLP	0.7438	0.7460	0.7871	0.7135	0.7485
White → White	Logistic Regression	0.7125	0.7250	0.8462	0.6842	0.7566
White → White	Random Forest	0.8125	0.7762	0.8311	0.8947	0.8618
White → White	MLP	0.7250	0.6670	0.7553	0.8565	0.8027

Table 2: Performance metrics (cross-domain). Positive class is "good" (quality ≥ 6).

Train → Test	Model	Acc.	Bal. Acc.	Prec.	Rec.	F1
Red → White	Logistic Regression	0.6000	0.6642	0.8716	0.4545	0.5975
Red → White	Random Forest	0.6188	0.6236	0.7605	0.6077	0.6755
Red → White	MLP	0.6219	0.6493	0.8014	0.5598	0.6592
White → Red	Logistic Regression	0.5469	0.5752	0.9333	0.1637	0.2786
White → Red	Random Forest	0.5719	0.5960	0.8400	0.2456	0.3801
White → Red	MLP	0.6656	0.6820	0.8636	0.4444	0.5869

2.3.4 B3. Supporting visualizations

Answer: Of the three models, random forest seems like the best model for training and testing within the same domain (training on red and testing on red or training on white and testing on white) as it has the highest accuracy and balanced accuracy of the three. It also has the highest f1 value. For cross-domain, MLP has the highest accuracy score for both red → white and white → red, but has a lower balanced accuracy than logistic regression for red → white. Logistic regression also has the highest precision value for both test sets.

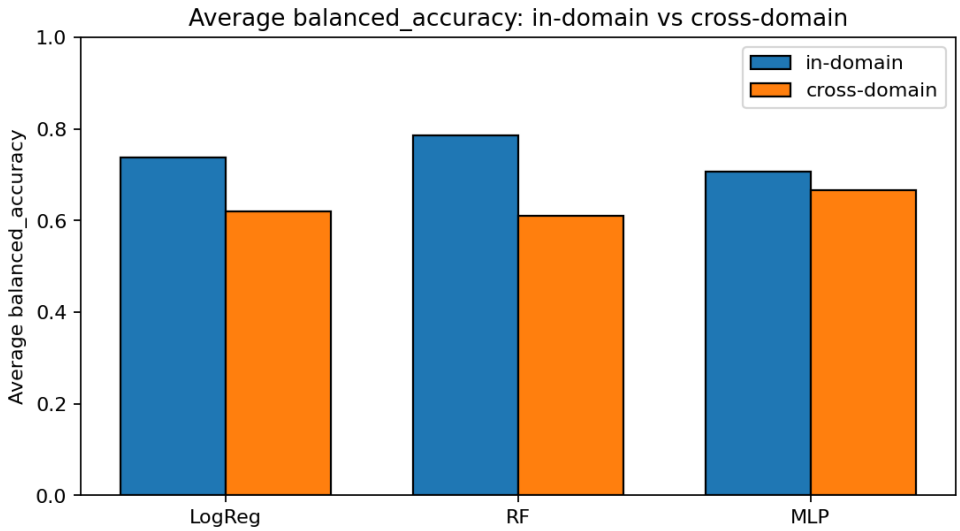


Figure 7: Average balanced accuracy for in-domain vs. cross-domain testing for each model.

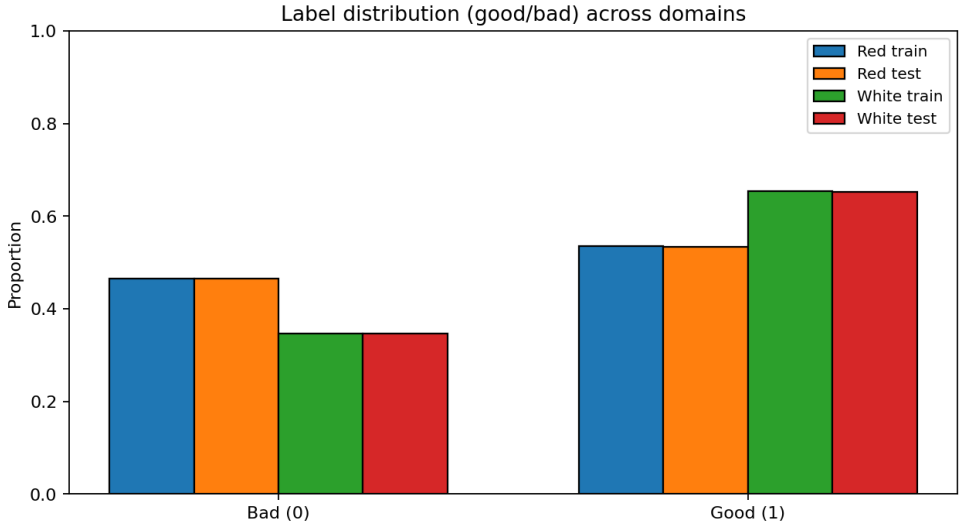


Figure 8: Label distribution (bad vs. good) across red/white train/test splits.

2.4 Part C: Interpretation + visualization

2.4.1 C1. In-domain consistency + overfitting

Answer: Across the in-domain settings (red→red and white→white), Random Forest was the strongest and relatively consistent: it achieved the best accuracy/F1 on both domains (0.794/0.801 on red; 0.813/0.862 on white). Logistic Regression and MLP were more similar to each other and generally lower. The large gap between RF's in-domain and cross-domain results suggests domain sensitivity (learning wine-type-specific patterns) rather than classic overfitting, since all scores reported are on held-out test sets.

2.4.2 C2. Cross-domain generalization + degradation

Answer: For cross-domain testing, MLP generalized best overall. It achieved the highest performance on white→red (balanced accuracy 0.682, F1 0.587) and remained competitive on red→white (balanced accuracy 0.649). Compared to its in-domain performance, MLP dropped by about **0.06** in average balanced accuracy (in-domain ≈0.706 vs cross-domain ≈0.666). In contrast, Logistic Regression and Random Forest showed larger degradations, especially on white→red where recall for the "good" class dropped sharply.

2.4.3 C3. Why cross-domain differs

Answer: Cross-domain differences are most likely caused by distribution shift between red and white wines: feature ranges and feature-label relationships can differ by wine type (e.g., alcohol, acidity, sulfur compounds). Label imbalance also affects transfer: Logistic Regression and Random Forest showed very high precision but low recall on white→red, indicating a conservative bias toward predicting bad. Random Forest can also specialize to domain-specific interactions, while simpler models may underfit but sometimes transfer more smoothly.

2.4.4 C4. Required visualization(s)

Answer: Figure 9 compares balanced accuracy across in-domain and cross-domain settings for all three models (and highlights the cross-domain performance drop). This visualization supports the conclusions in C1–C3 regarding consistency, generalization, and domain shift.

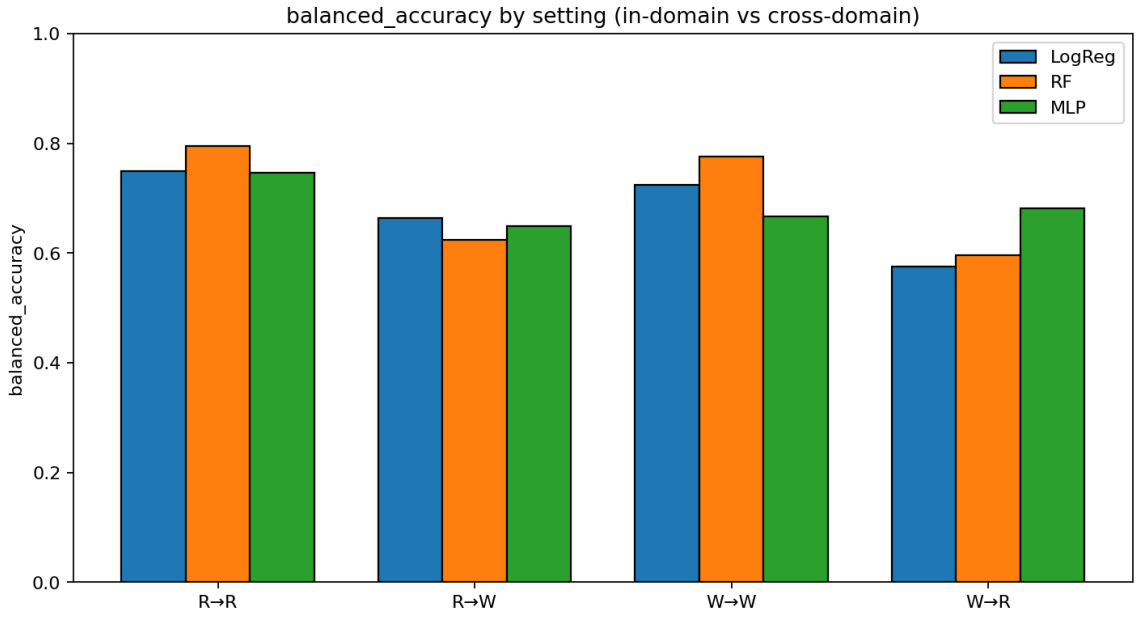


Figure 9: Balanced accuracy by evaluation setting (R→R, R→W, W→W, W→R) for Logistic Regression, Random Forest, and MLP.