CSDS 313/413: Introduction to Data Analysis
Assignment 4: Clustering and Dimensionality Reduction
Solutions

Wiam Skakri

December 3, 2025

1 Task 1: Clustering and Dimensionality Reduction

1.1 Part A: Principal Component Analysis

1.1.1 Question 1: Cumulative Variance Explained by Principal Components

Methodology We applied Principal Component Analysis (PCA) to the congressional votes dataset (`p1_congress_1984_votes.csv`), which contains voting records of 435 U.S. House of Representatives members on 16 key issues in 1984.

PCA was performed using `sklearn.decomposition.PCA` to identify the principal components that capture the maximum variance in the voting patterns.

Results Figure 1 shows the cumulative variance explained as a function of the number of principal components k.

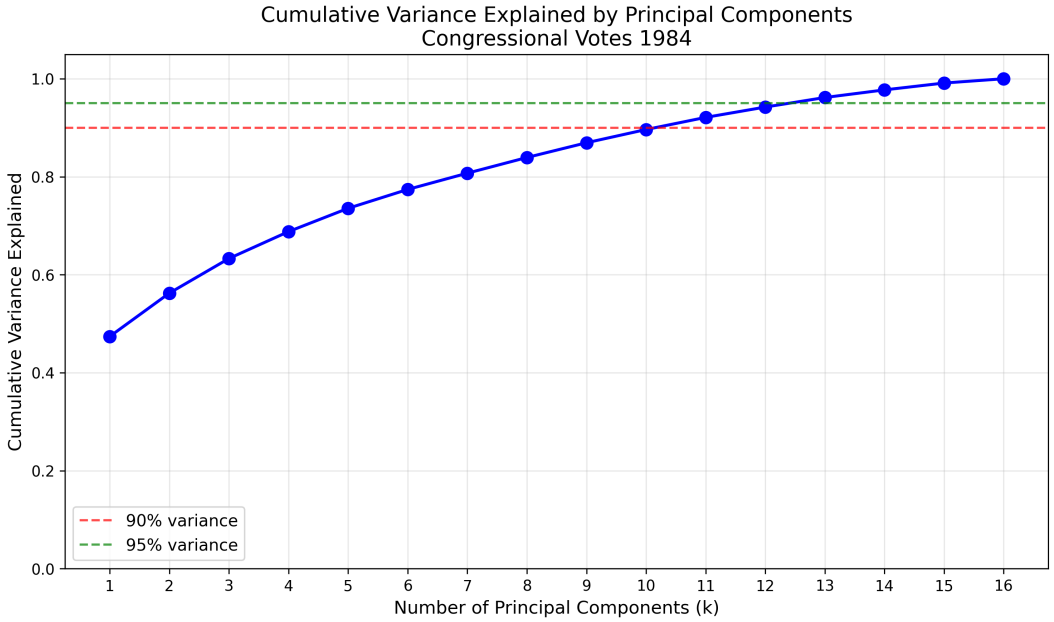

Figure 1: Cumulative variance explained by top k principal components. The red dashed line indicates 90% variance threshold, and the green dashed line indicates 95% variance threshold.

Key Observations

- The first principal component (PC1) explains approximately 47% of the total variance
- The first two principal components together explain approximately 56% of the variance
- To reach 90% cumulative variance, approximately **10 principal components** are required
- To reach 95% cumulative variance, approximately **12 principal components** are required

Recommendation: 10-12 principal components are sufficient to summarize the data.

Interpretation The PCA results suggest that while there is some correlation among the 16 votes (otherwise we would need all 16 components), the voting issues are sufficiently diverse that 10-12 dimensions are needed to adequately represent the voting patterns. This could indicate that the votes span multiple policy domains (economic, social, foreign policy, etc.) that are not perfectly aligned along a single ideological axis.

1.1.2 Question 2: Projection onto First 3 Principal Components

Methodology We projected the 435 congress members onto the first 3 principal components and created scatter plots for three PC pairs: (PC1-PC2), (PC1-PC3), and (PC2-PC3). Each point represents a congress member, colored by party affiliation (Democrats in blue, Republicans in red).

Variance Explained by First 3 Components

- PC1: 47.40% of total variance
- PC2: 8.84% of total variance
- PC3: 7.07% of total variance

Results Figure 2 shows the three scatter plot pairs with party affiliation colors.

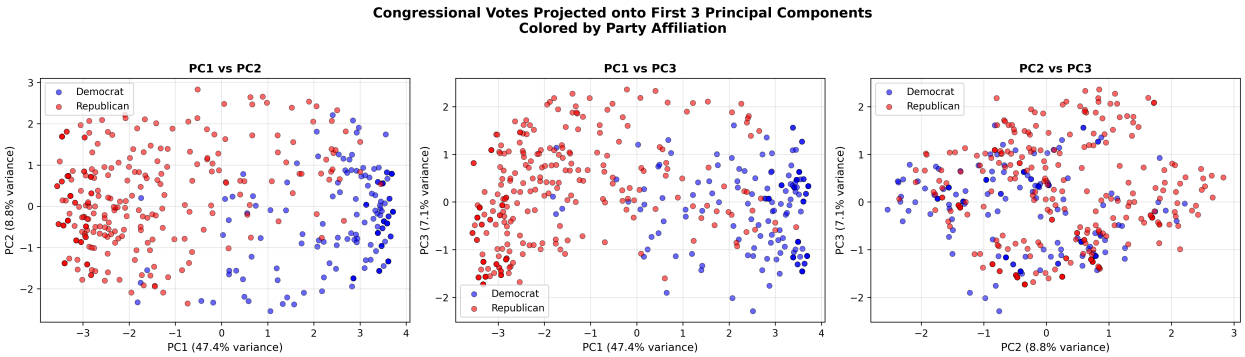

Figure 2: Scatter plots of congress members projected onto first 3 principal components, colored by party affiliation. Left: PC1 vs PC2, Middle: PC1 vs PC3, Right: PC2 vs PC3.

Quantitative Separation Analysis To objectively compare the separation quality of each PC pair, we calculated separation scores based on the ratio of between-party distance to within-party variance:

PC Pair	Centroid Distance	Avg Within-Party Variance	Separation Score
PC1-PC2	4.350	1.749	3.289
PC1-PC3	4.324	1.632	**3.385**
PC2-PC3	0.672	1.064	0.651

Table 1: Separation metrics for each principal component pair. Higher separation score indicates better party separation.

PC1-PC3 provides the best separation between parties (separation score: 3.385), followed closely by PC1-PC2 (3.289). PC2-PC3 shows poor separation (0.651).

Yes, congress members with the same party affiliation show clear clustering patterns.

Evidence:

- **Visual clustering**: In both PC1-PC2 and PC1-PC3 plots, Democrats (blue) cluster on the right side, while Republicans (red) cluster on the left side
- **Clear separation along PC1**: The primary axis of variation (PC1) strongly separates the two parties with minimal overlap in the center region

1.2 Part B: Clustering Analysis

Methodology We applied unsupervised clustering to group the 435 congress members into 2 clusters based solely on their voting patterns, without using party affiliation information.

Clustering Algorithm: K-Means We chose the K-Means clustering algorithm with the following specifications:

- **Algorithm**: K-Means clustering
- **Number of clusters (k)**: 2
- **Distance metric**: Euclidean distance
- **Random state**: 42 (for reproducibility)

How K-Means Works:

1. Initialize 2 cluster centers using the k-means++ strategy
2. Assign each congress member to the nearest cluster center
3. Update cluster centers to be the mean (centroid) of all assigned members
4. Repeat steps 2-3 until convergence (cluster assignments no longer change)

Distance Function:
The Euclidean distance in 16-dimensional vote space:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{16} (x_i - y_i)^2} \tag{1}$$

where $\mathbf{x}$ and $\mathbf{y}$ are the voting vectors of two congress members across the 16 issues.

Visualization Figure 3 shows the clustering results visualized on the first two principal components (PC1-PC2), which explain 56.24% of the total variance.

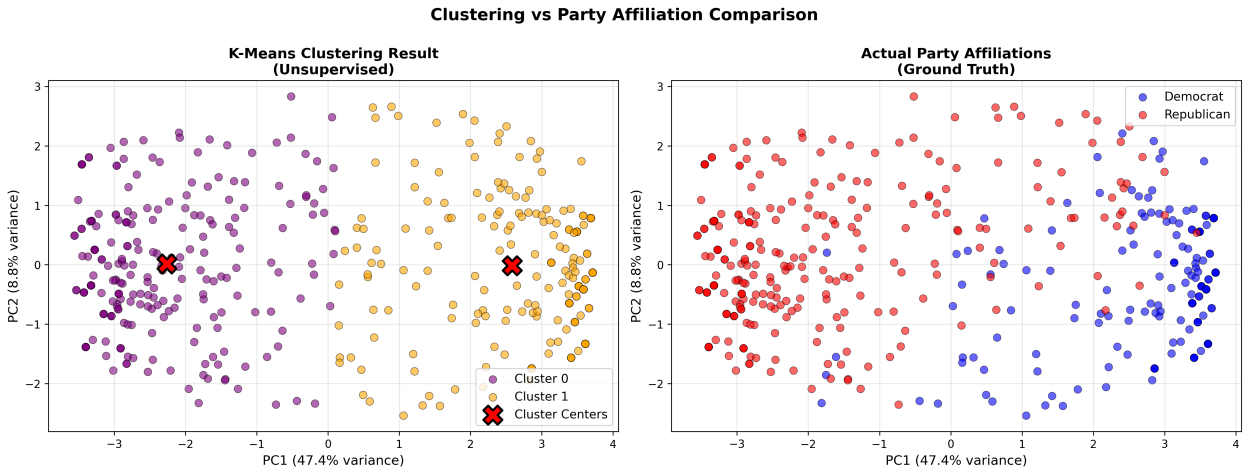

Figure 3: Comparison of K-Means clustering results (left) vs actual party affiliations (right). Left: Unsupervised clustering result with Cluster 0 (purple) and Cluster 1 (orange), cluster centers marked with red X. Right: Ground truth party affiliations with Democrats (blue) and Republicans (red).

Answer: Are the Groups Visually Separated? Yes, the two clusters are well-separated in the PC1-PC2 space.

Quantitative evidence:

- **Distance between cluster centers**: 4.844 (in PC space)
- **Average within-cluster spread**: 1.081
- **Separation ratio**: 4.480

The separation ratio of 4.48 (well above 2.0) indicates that the clusters are clearly separated with minimal overlap. The cluster centers (marked with red X in the left plot) are positioned far apart relative to the spread of points within each cluster.

Answer: Agreement with Party Affiliations The clustering shows strong agreement (88.3%) with actual party affiliations.

Party	**Cluster 0**	**Cluster 1**	**Total**
Democrat	8	**160**	168
Republican	**224**	43	267
Total	232	203	435

Table 2: Confusion matrix comparing clustering results with party affiliations. Bold numbers indicate correct cluster assignments.

Key findings:

- **Overall accuracy**: 88.3% (384 out of 435 correctly clustered)

- **Cluster 0 (purple)** predominantly contains Republicans: 224/232 = 96.6%
- **Cluster 1 (orange)** predominantly contains Democrats: 160/203 = 78.8%
- **Democrats**: 95.2% correctly clustered (160/168)
- **Republicans**: 83.9% correctly clustered (224/267)

1.2.1 Statistical Significance: Permutation Test

Methodology To assess whether the clustering structure we found is statistically significant (rather than occurring by chance), we performed a permutation test with 1,000 iterations.

Clustering Quality Score: We used the **silhouette score** as our quality metric:

- Measures how well-separated and compact clusters are
- Range: -1 to 1, where higher values indicate better clustering
- Calculated without using party labels (purely unsupervised metric)
- Formula: $s = \frac{b-a}{\max(a,b)}$ where a = mean intra-cluster distance, b = mean nearest-cluster distance

Permutation Procedure

1. **Compute original score**: Apply K-means to real data, calculate silhouette score
2. **Generate null distribution**: For each of 1,000 permutations:
   - Randomly shuffle each congress member's votes across the 16 issues
   - This destroys voting patterns while preserving vote distributions
   - Apply K-means clustering to permuted data
   - Calculate silhouette score
3. **Compare distributions**: Calculate p-value as the proportion of permuted scores $\geq$ original score

Null Hypothesis (H_0): The voting data has no inherent clustering structure; any observed clusters are due to random chance.

Alternative Hypothesis (H_1): The voting data contains real structure that produces meaningful clusters.

Results Figure 4 shows the distribution of clustering scores under the null hypothesis (permuted data) compared to the original score.

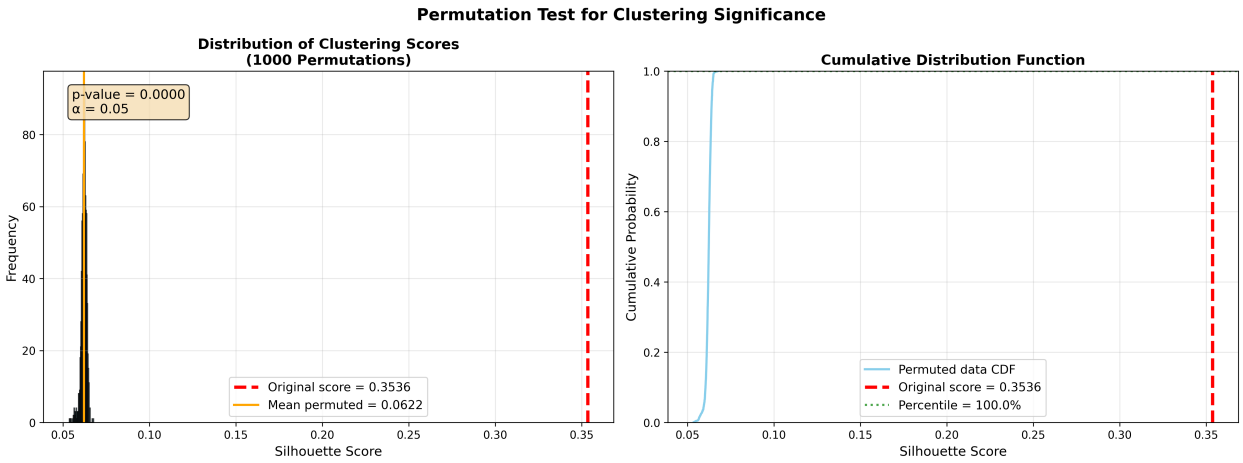

Figure 4: Permutation test results. Left: Histogram of silhouette scores from 1,000 permuted datasets (blue) compared to the original score (red dashed line). Right: Cumulative distribution function showing the original score at the 100th percentile.

Metric	**Value**
Original silhouette score	0.3536
Mean permuted score	0.0622
Standard deviation (permuted)	0.0016
Difference (original - mean permuted)	0.2914
Z-score	182.49
P-value	$<$ **0.001**

Table 3: Permutation test statistics comparing original clustering to null distribution.

Answer: Is the Clustering Statistically Significant? Yes, the clustering is highly statistically significant ($p < 0.001$).

Evidence:

- **P-value $<$ 0.001**: Out of 1,000 permutations, zero achieved a score as high as the original ($p = 0.0000$)
- **Extreme Z-score (182.49)**: The original score is more than 182 standard deviations above the mean of random data
- **100th percentile**: The original score exceeds 100% of permuted scores
- **Clear visual separation**: The histogram shows complete separation between null distribution (centered at 0.06) and original score (0.35)

Conclusion: We reject the null hypothesis and conclude that the two-cluster structure in 1984 congressional voting data represents a statistically significant and substantively meaningful division that corresponds to party affiliation.

1.3 Part C: Clustering Comparison Analysis

1.3.1 Task 1: Quantifying Agreement with Mutual Information

Methodology To quantify the agreement between cluster membership and party affiliation, we use **Normalized Mutual Information (NMI)**.

Normalized Mutual Information NMI measures how much knowing one variable tells us about another, normalized to a 0-1 scale:

$$NMI(X;Y) = \frac{MI(X;Y)}{\sqrt{H(X) \cdot H(Y)}}$$

where MI is mutual information and H is entropy.

- Range: $[0, 1]$ where 0 = no relationship, 1 = perfect agreement
- Measures reduction in uncertainty about party when cluster is known
- Higher values indicate stronger association

Results for Clustering on All 16 Votes NMI = 0.5097

Interpretation: NMI = 0.51 indicates strong agreement between clustering and party affiliation. Knowing which cluster a congress member belongs to substantially reduces uncertainty about their party affiliation.

1.3.2 Task 2: Comparison - Principal Components vs All Votes

Methodology We compared two clustering approaches:

1. **All 16 Votes**: K-means clustering on the full 16-dimensional vote space
2. **First 2 PCs**: K-means clustering on the 2-dimensional principal component space (PC1-PC2)

Both used the same K-means parameters (k=2, random state=42, k-means++ initialization) for fair comparison.

Results Figure 5 shows the comparison between the two clustering approaches.

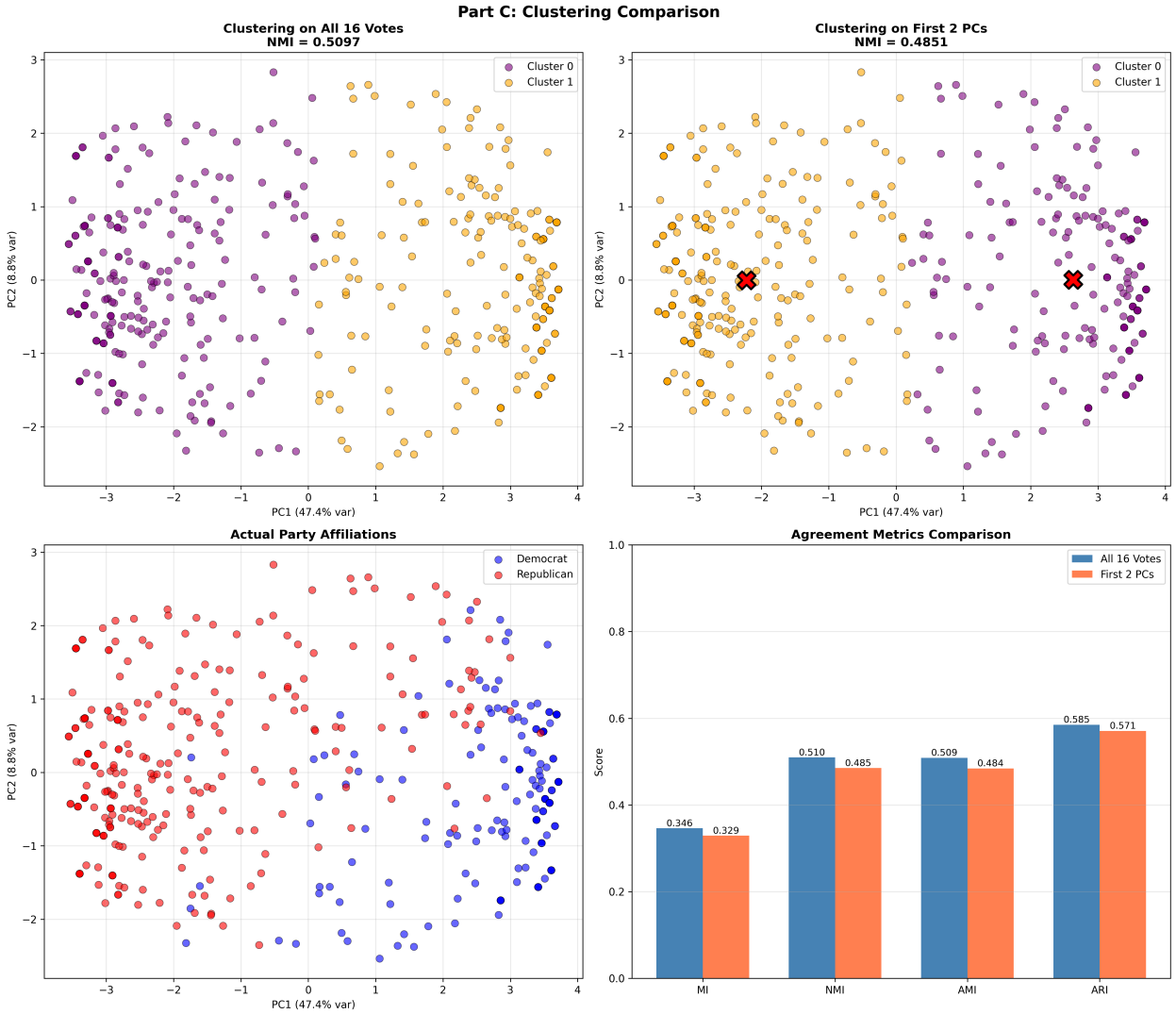

Figure 5: Comparison of clustering approaches. Top left: Clustering on all 16 votes (NMI = 0.5097). Top right: Clustering on first 2 PCs (NMI = 0.4851). Bottom left: Actual party affiliations. Bottom right: Agreement metrics comparison.

Approach	NMI	Difference
All 16 Votes	0.5097	-
First 2 PCs	0.4851	-0.0246

Table 4: NMI comparison between clustering approaches. All 16 votes achieves higher agreement with party affiliations.

Answer: Which Clustering Agrees More with Party Affiliations? Clustering on all 16 votes agrees more with party affiliations (NMI = 0.5097) compared to clustering on

first 2 PCs (NMI = 0.4851).

Information retention: The first 2 principal components capture only 56.24% of the total variance, meaning 43.76% of information is discarded. While PC1 strongly correlates with party (as shown in Part A), the missing dimensions contain additional party-discriminating information.

Multiple policy dimensions: Party differences span multiple policy domains (economic, social, foreign policy). While PC1-PC2 captures the dominant axes of variation, the full 16-dimensional space represents these nuances more completely.

Practical Implications The modest difference (0.025 in NMI) between approaches suggests:

1. PC1-PC2 captures the *majority* of party-relevant information
2. The additional 14 dimensions provide incremental but meaningful improvement
3. For exploratory analysis and visualization, PC1-PC2 is sufficient
4. For maximizing classification accuracy, using all features is preferable

2 Task 2: Predictive Modeling

2.1 Can Your Model Taste Good Wine?

2.2 Part A: Label Design and Discretization

2.2.1 Discretization Strategy

To convert the regression problem (predicting quality scores 0-10) into a binary classification task, I discretized the quality scores into two classes using the following criterion:

- **Good Wine**: Quality score > 5
- **Bad Wine**: Quality score ≤ 5

This threshold was applied consistently to both red and white wine datasets to ensure models trained on one wine type can be meaningfully evaluated on the other during cross-domain testing.

2.2.2 Original Quality Score Distributions

Before discretization, I analyzed the original quality score distributions in both datasets. Figure 6 (top row) shows the original quality distributions.

Key Observations:

- **Red Wine**: Quality scores range from 3 to 8, with a mean of 5.64. The distribution is approximately normal, centered around quality scores 5 and 6 (681 and 638 samples respectively).
- **White Wine**: Quality scores range from 3 to 9, with a mean of 5.88. The distribution is also approximately normal, with the highest frequency at quality 6 (2,198 samples) and 5 (1,457 samples).

- **Comparison**: White wines have slightly higher average quality scores than red wines, and a wider quality range (3-9 vs. 3-8).

2.2.3 Class Distribution After Discretization

After applying the threshold of 5, the resulting class distributions are shown in Figure 6 (bottom row):

Wine Type	Good Count	Good %	Bad Count	Bad %
Red Wine	855	53.5%	744	46.5%
White Wine	3,258	66.5%	1,640	33.5%

Table 5: Class distribution after discretization using threshold > 5.

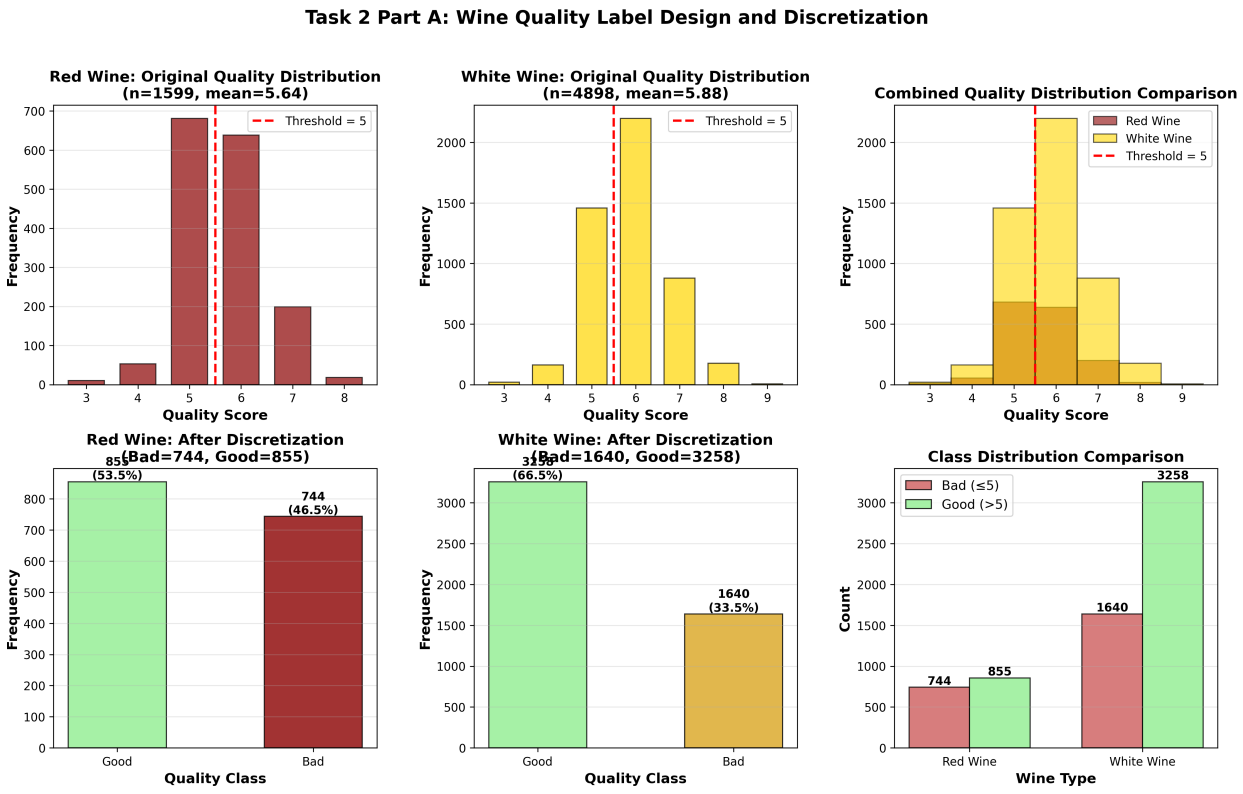

Figure 6: Task 2 Part A: Wine quality label design and discretization. **Top row**: Original quality score distributions for red wine (left), white wine (middle), and combined comparison (right), with the discretization threshold marked at quality = 5. **Bottom row**: Class distributions after discretization for red wine (left), white wine (middle), and comparison (right).

2.2.4 Rationale for Label Design

I discretized wine quality using a threshold of 5 (quality > 5 = Good, ≤ 5 = Bad) for the following reasons:

Interpretability: This threshold separates above-average wines (quality 6-9) from average and below-average wines (quality 3-5), which aligns with the intuitive notion of "good" wine based on the 0-10 quality scale. Wines with quality scores above 5 represent those that received favorable ratings from wine tasters.

Class Balance: This approach yields relatively balanced classes. Red wine: 53.5% good, 46.5% bad; White wine: 66.5% good, 33.5% bad. This balance helps prevent models from being biased toward the majority class and ensures both classes have sufficient representation for learning meaningful patterns.

Consistent Application: The threshold is applied identically to both red and white wine datasets, ensuring models trained on one type can be meaningfully evaluated on the other during cross-domain testing in Part B.

Sufficient Samples: Both classes have adequate sample sizes in each dataset for reliable model training and evaluation. Even the smaller class (red wine "bad") has 744 samples, providing sufficient data for robust learning.

2.3 Part B: Model Training and Evaluation

In this part, I trained two classification models with different complexity levels and evaluated their performance both within the same wine type (in-domain) and across different wine types (cross-domain).

2.3.1 Model Selection

I selected two classification models that differ in complexity and learning capacity:

Model 1: Logistic Regression (Simple Baseline) Logistic Regression is a linear classifier that models the probability of a sample belonging to a class using a logistic function. It serves as a simple, interpretable baseline.

Parameters:

- Solver: L-BFGS (Limited-memory Broyden-Fletcher-Goldfarb-Shanno)
- Maximum iterations: 1000
- Regularization: L2 (default)

Model 2: Neural Network (Complex Model) A Multi-Layer Perceptron (MLP) neural network with two hidden layers, capable of learning complex non-linear patterns.

Architecture:

- Input layer: 11 features (physicochemical attributes)
- Hidden layer 1: 64 neurons with ReLU activation
- Hidden layer 2: 32 neurons with ReLU activation

- Output layer: 2 classes (Good/Bad wine)

Training Parameters:

- Optimizer: Adam (adaptive learning rate)
- Batch size: 32
- Maximum epochs: 500
- Early stopping: Enabled (patience = 20 iterations)
- Validation split: 10% of training data
- L2 regularization ($\alpha = 0.0001$)

2.3.2 Data Preparation

Train/Test Split For each wine type (red and white), I split the data into training and testing sets using an 80/20 split with stratification to preserve class balance:

- **Red Wine**: 1,279 training samples, 320 test samples
- **White Wine (before downsampling)**: 3,918 training samples, 980 test samples

Downsampling White Wine Training Set To ensure fair comparison between models trained on red and white wines, I downsampled the white wine training set to match the red wine training set size (∼1,280 samples). Downsampling was performed using stratified sampling to maintain the class distribution (Good: 66.5%, Bad: 33.5%).

Final Training Set Sizes:

- Red Wine: 1,279 samples
- White Wine (after downsampling): 1,278 samples

Feature Standardization All 11 physicochemical features were standardized (mean = 0, standard deviation = 1) using separate StandardScaler instances for red and white wine training sets. This ensures that features with different scales contribute equally to model training and prevents numerical instability in gradient-based optimization.

2.3.3 Evaluation Strategy

Each model was evaluated under four conditions to assess both in-domain performance and cross-domain generalization:

In-Domain Testing Models trained and tested on the same wine type:

- **Red → Red**: Train on red wine, test on red wine
- **White → White**: Train on white wine, test on white wine

Cross-Domain Testing Models trained on one wine type and tested on the other:

- **Red → White**: Train on red wine, test on white wine
- **White → Red**: Train on white wine, test on red wine

Performance Metrics For each evaluation scenario, I reported three metrics:

- **Accuracy**: Overall proportion of correct predictions
- **Precision**: Proportion of predicted "Good" wines that are actually good
- **Recall**: Proportion of actual "Good" wines correctly identified

2.3.4 Results

Table 6 summarizes the performance of both models across all evaluation conditions.

Model	Train On	Test On	Accuracy	Precision	Recall
In-Domain Testing					
Logistic Regression	Red	Red	74.1%	76.8%	73.7%
Neural Network	Red	Red	70.3%	77.9%	62.0%
Logistic Regression	White	White	74.1%	76.5%	88.0%
Neural Network	White	White	**75.4%**	77.5%	88.8%
Cross-Domain Testing					
Logistic Regression	Red	White	**64.9%**	83.6%	58.7%
Neural Network	Red	White	63.9%	82.5%	58.0%
Logistic Regression	White	Red	**63.8%**	92.3%	35.1%
Neural Network	White	Red	55.6%	85.4%	20.5%

Table 6: Performance comparison of Logistic Regression and Neural Network across in-domain and cross-domain testing scenarios. Bold values indicate the best performance for each test condition.

2.4 Part C: Analysis and Interpretation

In this part, I interpret and compare the model results from Part B, analyzing performance consistency, generalization capability, and factors affecting cross-domain transfer.

2.4.1 Visualization

Figure 7 presents a comprehensive analysis of model performance across all evaluation scenarios, including performance comparison, degradation analysis, precision-recall trade-offs, consistency metrics, and a performance heatmap.

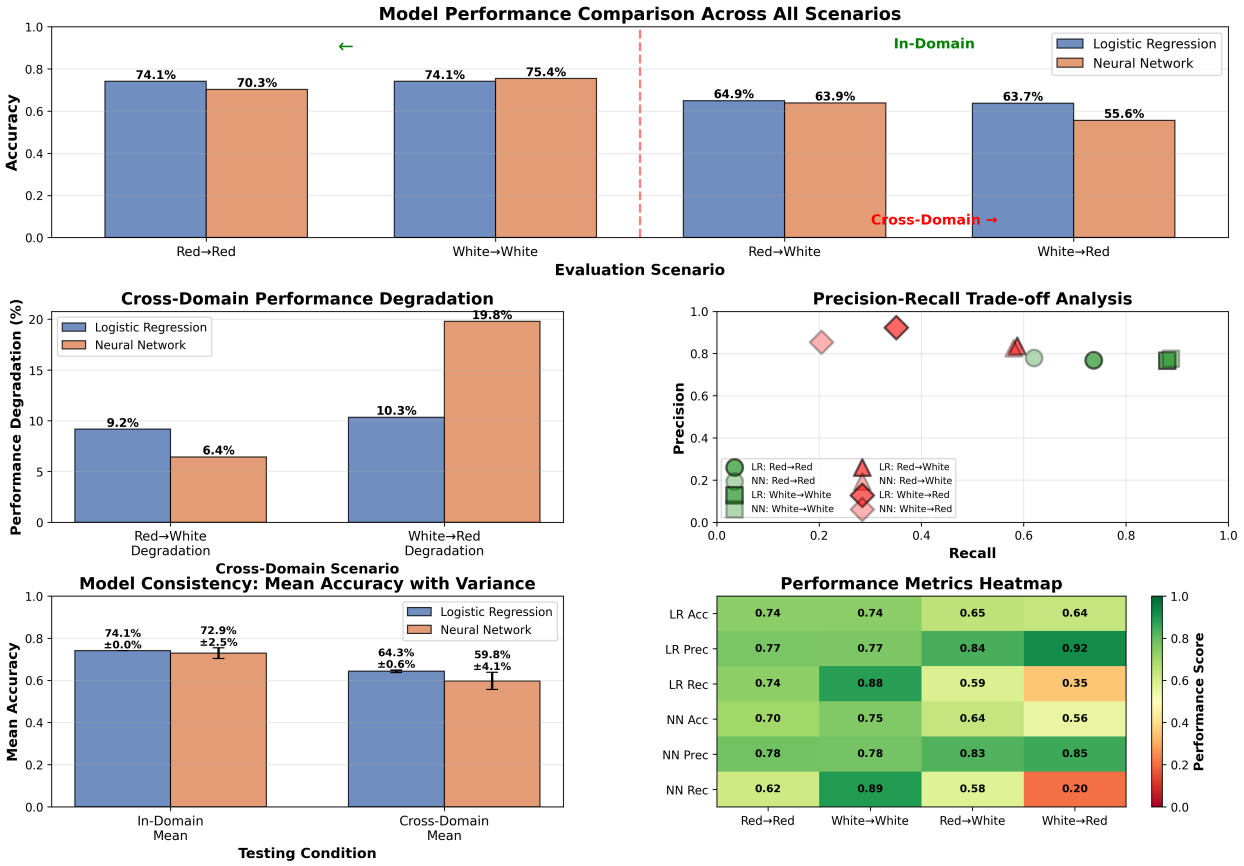

Figure 7: Task 2 Part C: Comprehensive model performance analysis. **Top**: Overall accuracy comparison across scenarios with in-domain/cross-domain distinction. **Middle Left**: Cross-domain performance degradation. **Middle Right**: Precision-recall trade-off analysis. **Bottom Left**: Model consistency with variance. **Bottom Right**: Performance metrics heatmap.

2.4.2 Question 1: In-Domain Consistency and Overfitting

Which model demonstrated more consistent in-domain performance? Did you observe any signs of overfitting?

Logistic Regression demonstrated superior consistency across in-domain settings, achieving exactly 74.1% accuracy on both red and white wines (standard deviation = 0.0%). In contrast, Neural Network showed more variability (mean = 72.9%, std = 2.5%), performing worse on red wine (70.3%) but slightly better on white wine (75.4%). This variance suggests the Neural Network is sensitive to dataset-specific characteristics, a sign of overfitting. The Neural Network's poor cross-domain performance (especially White→Red at 55.6%) compared to its in-domain performance confirms overfitting to wine-type-specific patterns rather than learning generalizable quality indicators.

2.4.3 Question 2: Cross-Domain Generalization

Which model generalized better across wine types? How much did performance degrade?

Logistic Regression generalized significantly better across wine types, maintaining a mean cross-domain accuracy of 64.3% (Red→White: 64.9%, White→Red: 63.8%) with minimal variance (std = 0.6%). Neural Network achieved only 59.8% mean cross-domain accuracy with high variance (std = 4.1%). Performance degradation: Logistic Regression experienced consistent drops of 9.2% and 10.3%, while Neural Network showed asymmetric degradation—6.4% for Red→White but a severe 19.8% drop for White→Red. The Neural Network's failure to generalize from white to red wine (recall = 20.5%) demonstrates its over-reliance on wine-type-specific features.

2.4.4 Question 3: Factors Explaining Cross-Domain Differences

What factors might explain the performance differences observed during cross-domain testing?

Three primary factors explain the cross-domain performance differences: **(1) Feature distribution shift**: Red and white wines have different physicochemical profiles (different acidity levels, sulfur dioxide concentrations, alcohol content). Models trained on one distribution struggle when feature values fall outside their training range. **(2) Model complexity and overfitting**: The Neural Network's higher capacity allowed it to memorize wine-type-specific patterns rather than learning universal quality indicators, while Logistic Regression's simplicity enforced learning of more generalizable linear relationships. **(3) Label imbalance**: White wines have 66.5% "good" wines versus red's 53.5%, causing models to learn different decision boundaries that transfer poorly across domains.