AMATH 445/645 – Practice Problems for Lectures 5–8

1. A node in a decision tree contains:

   - Class A: 8 samples
   - Class B: 2 samples

   Compute the Gini impurity of this node.

   First calculate class probabilities:

   $$p_A = \frac{8}{10} = 0.8, \quad p_B = \frac{2}{10} = 0.2$$

   Gini impurity is:

   $$G = 1 - \sum_{i=1}^{C} p_i^2 = 1 - (p_A^2 + p_B^2) = 1 - (0.8^2 + 0.2^2)$$

   $$G = 1 - (0.64 + 0.04) = 1 - 0.68 = 0.32$$

2. What is the maximum possible Gini impurity for a binary classification problem? When does this occur?

   Gini impurity is maximized when classes are perfectly balanced (equal probability):

   $$p = 0.5, \quad 1 - p = 0.5$$

   $$G_{\max} = 1 - (0.5^2 + 0.5^2) = 1 - (0.25 + 0.25) = 1 - 0.5 = 0.5$$

   For binary classification, Gini impurity ranges from 0 (pure node) to 0.5 (completely impure).

3. Explain intuitively why Gini impurity measures the "impurity" or uncertainty in a node.

   Gini impurity can be interpreted as the probability of misclassification if we randomly assign labels according to the class distribution in that node.
   If we randomly pick a sample and then randomly assign it a label following the same distribution, the probability of misclassification equals $\sum_i p_i(1 - p_i) = 1 - \sum_i p_i^2$, which is exactly the Gini impurity.
   Thus:

   - $G = 0$: All samples same class → no uncertainty
   - $G = 0.5$: Classes equally mixed → maximum uncertainty

4. A dataset has initial Gini impurity $G_{\text{parent}} = 0.48$.

   After a candidate split:

   - Left child: 4 samples, $G = 0$

- Right child: 11 samples, $G = 0.40$

Compute the impurity decrease ΔG for this split.

Weighted impurity after split:

$$G_{\text{after}} = \frac{4}{15} \times 0 + \frac{11}{15} \times 0.40 = 0 + \frac{4.4}{15} = 0.2933$$

Impurity decrease:

$$\Delta G = G_{\text{parent}} - G_{\text{after}} = 0.48 - 0.2933 = 0.1867$$

5. Why do decision tree algorithms prefer splits that maximize the impurity decrease ΔG?

   Larger impurity decrease means the split creates purer child nodes, better separating the classes. This leads to:

   - More confident predictions at leaves
   - Shallower trees (fewer splits needed to achieve pure nodes)
   - Better generalization (if not overfitting)

   The algorithm searches over all features and thresholds to find the split with maximum ΔG.

6. A decision tree achieves 100% accuracy on training data but only 60% accuracy on test data. What phenomenon is occurring, and why?

   This is **overfitting**. The tree has grown too deep, memorizing noise and specific patterns in the training data instead of learning generalizable patterns. It captures idiosyncrasies that don't exist in the test set, resulting in poor performance on unseen data.

7. Name three stopping criteria that can prevent overfitting in decision trees. Explain each.

   (a) **Maximum depth**: Limit how deep the tree can grow. Shallower trees are simpler and less likely to overfit.

   (b) **Minimum samples per leaf**: Require at least N samples in each leaf node. Prevents leaves from capturing noise from very few samples.

   (c) **Minimum impurity decrease**: Only split if the impurity decrease exceeds a threshold ϵ. Stops splitting when gains are negligible.

   (d) **Maximum number of leaves**: Limit total complexity of the tree.

   Alternatively, grow a full tree then apply **pruning** (removing branches that don't improve validation performance).

8. What is bootstrap sampling (bagging), and why is it used in random forests?

   Bootstrap sampling: Creating multiple datasets by randomly sampling from the original data *with replacement*. Each bootstrap dataset has the same size as the original but contains duplicate samples and omits about 1/3 of the original samples (out-of-bag samples).
   Why used:

   - Creates diverse training sets for each tree

- Reduces correlation between trees (they see different data)
- Lower correlation → greater variance reduction when averaging
- Out-of-bag samples provide built-in validation

9. For classification, how does a random forest produce the final prediction from its individual trees?

   Random forests use **majority voting** (or mode) across all trees for classification:

   $$\hat{y} = \text{mode}\{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_T\}$$

   For regression, they use averaging:

   $$\hat{y} = \frac{1}{T}\sum_{t=1}^{T} \hat{y}_t$$

   This aggregation smooths out individual tree errors and reduces variance.

10. Explain why random forests are generally more robust than a single deep decision tree.

    A single deep decision tree has **low bias but high variance** — small changes in training data can produce completely different trees.
    Random forests reduce variance through:

    (a) **Bagging**: Each tree trained on different bootstrap samples
    (b) **Feature randomness**: Each split considers random feature subsets
    (c) **Averaging**: Combining many uncorrelated trees smooths out errors

    The bias increases slightly (each tree is weaker), but variance decreases significantly, leading to better generalization.

11. Consider a tiny dataset with 4 samples and 2 features:

    $$(0,0) : A, \quad (0,1) : A, \quad (1,0) : B, \quad (1,1) : B$$

    (a) Compute the Gini impurity of the root node.
    (b) Consider splits on $x_1 > 0.5$ and $x_2 > 0.5$. Which split yields the larger impurity decrease?

    (a) Root node: 2 A, 2 B → $p_A = 0.5$, $p_B = 0.5$

    $$G_{\text{root}} = 1 - (0.5^2 + 0.5^2) = 1 - 0.5 = 0.5$$

    (b) Split on $x_1 > 0.5$:
    - Left ($x_1 \leq 0.5$): $(0,0) : A$, $(0,1) : A$ → 2 A, 0 B → $G = 0$
    - Right ($x_1 > 0.5$): $(1,0) : B$, $(1,1) : B$ → 0 A, 2 B → $G = 0$

    $G_{\text{after}} = \frac{2}{4}(0) + \frac{2}{4}(0) = 0$, $\Delta G = 0.5 - 0 = 0.5$
    Split on $x_2 > 0.5$ gives identical result. Both splits achieve perfect separation.

12. Explain step-by-step how the K-Nearest Neighbors algorithm makes a prediction for a new data point x_{new}.

(a) Compute the distance between x_{new} and every training point x_i using a chosen distance metric (e.g., Euclidean distance).

(b) Sort the distances and identify the k training points with the smallest distances (the k nearest neighbors).

(c) For classification: Take a majority vote among the labels of these k neighbors to determine the predicted class.

(d) For regression: Calculate the average (or weighted average) of the target values of the k neighbors.

13. Why is it essential to scale/normalize features before applying KNN?

KNN relies on distance calculations. If features have different scales (e.g., age 0-100 vs. income 0-100,000), the feature with larger magnitude will dominate the distance, effectively ignoring other features.
Common scaling methods:

- Standardization: $x' = \frac{x-\mu}{\sigma}$ (zero mean, unit variance)
- Min-max scaling: $x' = \frac{x-x_{\min}}{x_{\max}-x_{\min}}$ (range [0,1])

Without scaling, KNN produces biased and unreliable results.

14. Consider a tiny dataset:

$$(1,2):A, \quad (2,3):A, \quad (3,1):B, \quad (6,5):B$$

A new point $x_{\text{new}} = (2,2)$ arrives. Using $k = 3$ and Euclidean distance:

(a) Compute distances to all training points.

(b) Identify the 3 nearest neighbors.

(c) Predict the class using majority vote.

(a) Distances:

- To $(1,2)$: $\sqrt{(2-1)^2+(2-2)^2} = \sqrt{1+0} = 1.00$
- To $(2,3)$: $\sqrt{(2-2)^2+(2-3)^2} = \sqrt{0+1} = 1.00$
- To $(3,1)$: $\sqrt{(2-3)^2+(2-1)^2} = \sqrt{1+1} = 1.41$
- To $(6,5)$: $\sqrt{(2-6)^2+(2-5)^2} = \sqrt{16+9} = 5.00$

(b) Sorted distances: 1.00 (A), 1.00 (A), 1.41 (B), 5.00 (B)
The 3 nearest neighbors are: $(1,2)$:A, $(2,3)$:A, $(3,1)$:B

(c) Majority vote: 2 votes for A, 1 vote for B $\rightarrow$ predict $\boxed{A}$

15. A classifier produces the following confusion matrix on a test set:

	Predicted Positive	Predicted Negative
Actual Positive	30	10
Actual Negative	20	40

Compute:

(a) Accuracy

(b) Precision

(c) Recall (Sensitivity)

(d) Specificity

First identify values: TP = 30, FN = 10, FP = 20, TN = 40 Total = 100

(a) Accuracy = $\frac{TP+TN}{TP+TN+FP+FN} = \frac{30+40}{100} = 0.70$

(b) Precision = $\frac{TP}{TP+FP} = \frac{30}{50} = 0.60$

(c) Recall = $\frac{TP}{TP+FN} = \frac{30}{40} = 0.75$

(d) Specificity = $\frac{TN}{TN+FP} = \frac{40}{60} \approx 0.667$

16. In the planet habitability example, a lazy classifier that always predicts "non-habitable" achieved 98.36% accuracy. Explain why this high accuracy is misleading.

The dataset is highly imbalanced (about 98.4% non-habitable), and the lazy classifier finds zero habitable planets! The model is completely useless for the actual task (finding habitable planets) despite high accuracy. This illustrates why accuracy alone is insufficient for imbalanced datasets.

17. When would you prioritize precision over recall? When would you prioritize recall over precision?

Prioritize precision (minimize false positives) when false positives are costly:

- Spam detection (marking legitimate email as spam)
- Medical screening for expensive follow-up tests
- Fraud detection (flagging legitimate transactions as fraud)

Prioritize recall (minimize false negatives) when missing positives is costly:

- Cancer detection (missing a real case)
- Airport security (missing a threat)
- Search and rescue operations

F1-score balances both when equal importance is desired.

18. Describe K-fold cross-validation in detail. How does it work?

(a) Randomly shuffle the dataset and split it into K equal-sized folds (subsets).

(b) For each fold $i = 1$ to K:
- Use fold i as the validation set
- Use the remaining $K - 1$ folds as the training set
- Train the model on the training set and evaluate on the validation set
- Record the performance metric

(c) Average the K performance metrics to get the final estimate.

Each data point is used for validation exactly once and for training $K - 1$ times.

19. What are the advantages of cross-validation over a single train/test split?

- **More reliable estimate**: Uses all data for both training and validation, reducing variance in performance estimates.

- **Better use of limited data**: Every sample contributes to both training and validation.
- **Less dependent on a particular split**: A single split might be "lucky" or "unlucky"; CV averages over multiple splits.
- **Helps detect overfitting**: Consistently low CV score but high training score indicates overfitting.
- **Model selection**: Can compare different models or hyperparameters more reliably.

20. Define bias and variance in the context of machine learning.

- **Bias**: Error due to overly simplistic assumptions in the learning algorithm. High bias models consistently miss relevant patterns (underfitting).
- **Variance**: Error due to excessive sensitivity to small fluctuations in the training set. High variance models change dramatically with different training data (overfitting).

21. Distinguish between feature engineering and feature selection.

- **Feature engineering**: Creating *new* features or transforming existing ones to better represent the underlying problem. It expands or modifies the feature space.
- **Feature selection**: Choosing a *subset* of existing features to use in the model. It reduces the feature space.

Both aim to improve model performance, but through different mechanisms:

- Engineering: Create more informative representations
- Selection: Remove irrelevant or redundant features

22. PCA finds a unit vector v_1 that maximizes:

$$v_1^T C v_1 \quad \text{subject to } \|v_1\| = 1$$

Explain in words what $v_1^T C v_1$ represents.

$v_1^T C v_1$ is the variance of the data when projected onto the direction v_1.

23. Using Lagrange multipliers, show that maximizing $v^T C v$ subject to $\|v\| = 1$ leads to the eigenvalue equation $Cv = \lambda v$.

We set up the Lagrangian:

$$\mathcal{L}(v, \lambda) = v^T C v - \lambda(v^T v - 1)$$

Take the gradient with respect to v and set to zero:

$$\nabla_v \mathcal{L} = 2Cv - 2\lambda v = 0$$

Simplify:

$$Cv = \lambda v$$

Thus the maximizing directions are eigenvectors of the covariance matrix C, and the Lagrange multiplier λ is the corresponding eigenvalue (variance along that direction).

24. What does the eigenvalue λ_k represent in PCA? How do we interpret a very small eigenvalue?

λ_k is the variance of the data along the k-th principal component v_k.

- λ_k large: The data varies significantly in this direction; important structure.
- λ_k small: The data has little variation in this direction; likely noise or irrelevant features.

The sum of all eigenvalues equals the total variance in the data: $\sum_{k=1}^{d} \lambda_k = \text{trace}(C)$.

25. You have a dataset with 50 features. After running PCA, you obtain the following eigenvalues:

$$\lambda_1 = 15.2,\ \lambda_2 = 8.7,\ \lambda_3 = 4.1,\ \lambda_4 = 1.8,\ \lambda_5 = 0.9,\ \lambda_{6-50} < 0.5$$

(a) What is the total variance?

(b) What percentage of variance is explained by the first 3 components?

(c) How many components would you keep? Justify.

(a) Total variance = sum of all eigenvalues. We don't have all values, but we can approximate from the given:

$$\text{Total} \approx 15.2 + 8.7 + 4.1 + 1.8 + 0.9 + 45 \times 0.3 \approx 30.7 + 13.5 = 44.2$$

(b) Variance explained by first 3 components:

$$\frac{15.2 + 8.7 + 4.1}{44.2} = \frac{28.0}{44.2} \approx 0.634 = 63.4\%$$

(c) **Possible answers depending on goal:**

- Keep 3-4 components: They capture most variance (63-67%) and subsequent components add little.
- If target is 90% variance, would need more components (maybe 8-10).
- For visualization, keep 2-3 components regardless of variance.

26. The centered data matrix $X_c \in \mathbb{R}^{N \times d}$ has SVD decomposition $X_c = U\Sigma V^T$. Show how the eigenvectors and eigenvalues of the covariance matrix $C = \frac{1}{N-1} X_c^T X_c$ relate to the SVD components.

Starting with the covariance matrix:

$$C = \frac{1}{N-1} X_c^T X_c = \frac{1}{N-1} (U\Sigma V^T)^T (U\Sigma V^T)$$

Since $U^T U = I$ (orthogonal):

$$C = \frac{1}{N-1} V\Sigma^T U^T U\Sigma V^T = \frac{1}{N-1} V\Sigma^T \Sigma V^T$$

$\Sigma^T\Sigma$ is a diagonal matrix with squared singular values σ_k^2. Thus:

$$C = V \left(\frac{1}{N-1} \Sigma^2 \right) V^T$$

Therefore:

- The right singular vectors V are the eigenvectors of C (principal components)
- The eigenvalues of C are $\lambda_k = \frac{\sigma_k^2}{N-1}$
- The left singular vectors U contain the projected data (scores)

27. What is the key difference between PCA and Fisher Discriminant Analysis (FDA)?

- **PCA (unsupervised)**: Finds directions that maximize *overall variance* regardless of class labels. Good for reconstruction and capturing data spread.
- **FDA (supervised)**: Finds directions that maximize *separation between classes* while minimizing variance within each class. Good for classification tasks.

FDA maximizes the ratio of between-class scatter to within-class scatter:

$$\max_v \frac{v^T S_B v}{v^T S_W v}$$

where S_B is between-class scatter matrix and S_W is within-class scatter matrix.

28. Define the between-class scatter matrix S_B for two classes. What does each term represent?

$$S_B = n_0(m_0 - m)(m_0 - m)^T + n_1(m_1 - m)(m_1 - m)^T$$

where:

- n_0, n_1 = number of samples in class 0 and class 1
- m_0, m_1 = mean vectors of class 0 and class 1
- m = overall mean of all data

Each term $(m_k - m)(m_k - m)^T$ is the outer product of the deviation of class k's mean from the overall mean, weighted by the class size. S_B captures how far apart the class means are from each other and from the global center.

29. Show that for two classes, S_B can be simplified to:

$$S_B = \frac{n_0 n_1}{n_0 + n_1}(m_1 - m_0)(m_1 - m_0)^T$$

Start with the overall mean:

$$m = \frac{n_0 m_0 + n_1 m_1}{n_0 + n_1}$$

For class 0 deviation:

$$m_0 - m = m_0 - \frac{n_0 m_0 + n_1 m_1}{n_0 + n_1} = \frac{n_1}{n_0 + n_1}(m_0 - m_1)$$

Similarly for class 1:

$$m_1 - m = \frac{n_0}{n_0 + n_1}(m_1 - m_0)$$

Substitute into S_B:

$$S_B = n_0 \left[\frac{n_1}{n_0 + n_1}(m_0 - m_1)\right]\left[\frac{n_1}{n_0 + n_1}(m_0 - m_1)\right]^T + n_1 \left[\frac{n_0}{n_0 + n_1}(m_1 - m_0)\right]\left[\frac{n_0}{n_0 + n_1}(m_1 - m_0)\right]^T$$

Factor out $(m_1 - m_0)(m_1 - m_0)^T$ (note sign squared becomes positive):

$$S_B = \left[\frac{n_0 n_1^2}{(n_0+n_1)^2} + \frac{n_1 n_0^2}{(n_0+n_1)^2}\right](m_1 - m_0)(m_1 - m_0)^T$$

Simplify the coefficient:

$$\frac{n_0 n_1 (n_1 + n_0)}{(n_0+n_1)^2} = \frac{n_0 n_1}{n_0 + n_1}$$

Thus:

$$S_B = \frac{n_0 n_1}{n_0 + n_1}(m_1 - m_0)(m_1 - m_0)^T$$

30. Define the within-class scatter matrix S_W for two classes. What does it measure?

$$S_W = \sum_{i:y_i=0}(x_i - m_0)(x_i - m_0)^T + \sum_{i:y_i=1}(x_i - m_1)(x_i - m_1)^T$$

S_W measures the spread (covariance) of points within each class. It is the sum of scatter matrices for each class:

- First term: sum of outer products of deviations from class 0 mean
- Second term: sum of outer products of deviations from class 1 mean

A smaller S_W means points in each class are tightly clustered around their class mean.

31. Show that maximizing $J(w)$ leads to a generalized eigenvalue problem.

Maximizing $J(w)$ is equivalent to maximizing $w^T S_B w$ subject to $w^T S_W w = 1$ (fixing denominator).
Set up Lagrangian:

$$\mathcal{L}(w, \lambda) = w^T S_B w - \lambda(w^T S_W w - 1)$$

Take gradient with respect to w and set to zero:

$$\nabla_w \mathcal{L} = 2S_B w - 2\lambda S_W w = 0$$

Simplify:

$$S_B w = \lambda S_W w$$

This is a generalized eigenvalue problem. If S_W is invertible:

$$S_W^{-1} S_B w = \lambda w$$

The optimal w is the eigenvector corresponding to the largest eigenvalue λ.

32. For two classes, show that the optimal FDA direction is:

$$w \propto S_W^{-1}(m_1 - m_0)$$

For two classes, we showed:

$$S_B = \frac{n_0 n_1}{n_0 + n_1}(m_1 - m_0)(m_1 - m_0)^T = dd^T$$

where $d = \sqrt{\frac{n_0 n_1}{n_0+n_1}}(m_1 - m_0)$
Substitute into the generalized eigenproblem $S_B w = \lambda S_W w$:

$$dd^T w = \lambda S_W w$$

Note $d^T w$ is a scalar. Let $\alpha = d^T w$. Then:

$$d\alpha = \lambda S_W w \quad \Rightarrow \quad w = \frac{\alpha}{\lambda} S_W^{-1} d$$

Since α/λ is just a scalar, w is proportional to $S_W^{-1} d$, and d is proportional to $(m_1 - m_0)$. Therefore:

$$w \propto S_W^{-1}(m_1 - m_0)$$

33. Explain how FDA can be used for feature selection. What do large $|w_i|$ values indicate?

The discriminant direction $w = [w_1, w_2, \ldots, w_d]^T$ gives weights for each original feature.

- Features with **large absolute weights** $|w_i|$ contribute strongly to the discriminant; they are important for class separation.
- Features with **small weights** contribute little; they may be irrelevant or redundant.
- Sign of w_i indicates direction of relationship: positive means larger feature values increase the discriminant score (push toward class 1).

To select features, we can:

(a) Rank features by $|w_i|$

(b) Keep top k features with largest absolute weights

(c) Retrain classifier using only selected features