**University of Central Missouri**

**Department of Computer Science**

**CS5710 Machine Learning**

**Fall 2025**

**Home Assignment 2.**

**Student name: indukuri shanmukha shiva keasva varma**

**Submission Requirements:**

* Once finished your assignment push your source code to your repo (GitHub) and explain the work through the ReadMe file properly. Make sure you add your student info in the ReadMe file.
* Comment your code appropriately ***IMPORTANT.***
* Any submission after provided deadline is considered as a late submission.

**Part A: Calculation (6 Questions)**

**Q1. Decision Stump Prediction**  
Given a decision stump:

h(x)=+ if Sneezing=Yes, − otherwise.

Dataset:

* (Sneezing=Yes, Label=+)
* (Sneezing=No, Label=-)
* (Sneezing=Yes, Label=-)
* (Sneezing=No, Label=-)

1. What is the **training error rate** of this stump?

Predictions: “Sneezing = Yes → +”, else −.  
Data: (Yes,+) ✓, (No,−) ✓, (Yes,−) ✗, (No,−) ✓ → 1/4 errors → **25% training error**.

1. Compare it to the **memorizer** model (predicts perfectly).

Memorizer model: **0%** (perfect fit).

**Q2. Training Error as Splitting Criterion**  
A dataset has 6 records:

| **Age (x1)** | **Exercise (x2)** | **Diet (x3)** | **Label** |
| --- | --- | --- | --- |
| Young | High | Poor | Yes |
| Young | Medium | Good | Yes |
| Mid | Low | Poor | No |
| Old | Medium | Poor | No |
| Old | High | Good | Yes |
| Mid | Low | Poor | No |

1. Compute the **training error rate** if you split on each feature (x1, x2, x3).

a) Training error for each possible root split:

* Age → **1/6 = 16.7%**
* Exercise → **1/6 = 16.7%**
* Diet → **1/6 = 16.7%**

1. Which feature is the best root split using **training error**?

Best split by training error: **Tie** (all equal at 16.7%).

**Q3. Entropy & Information Gain**  
For the same dataset above:

1. Compute the **entropy of the labels**.
2. Compute entropy after splitting on **Exercise (x2)**.
3. Calculate the **information gain**.
4. Decide if Exercise is a good split.

1)Overall labels: 3 Yes, 3 No → entropy **H(Y)=1.000 bits**.  
2)After split on Exercise: High (2 Yes) → 0; Medium (1/1) → 1; Low (2 No) → 0.  
3)Weighted entropy = (2/6)\*0 + (2/6)\*1 + (2/6)\*0 = **0.333 bits**.  
4)Information gain = 1.000 − 0.333 = **0.667 bits** → **Exercise is a good split**.

**Q4. Confusion Matrix Metrics**  
A binary classifier produces the following confusion matrix on 100 test samples:

|  | **Predicted +** | **Predicted -** |
| --- | --- | --- |
| Actual + | 25 | 5 |
| Actual - | 15 | 55 |

1. Compute **Accuracy, Precision, Recall, Specificity, and F1-score**.

**Accuracy** = (TP+TN)/N = (25+55)/100 = **0.80**

**Precision** = TP/(TP+FP) = 25/40 = **0.625**

**Recall** = TP/(TP+FN) = 25/30 = **0.833**

**Specificity** = TN/(TN+FP) = 55/70 = **0.786**

**F1** = 2TP/(2TP+FP+FN) = 50/70 = **0.714**

1. Suppose the dataset was imbalanced (80 negatives, 20 positives). Which metric is most informative?

For an 80/20 imbalanced set, accuracy can mislead; **F1 (and/or precision–recall, or balanced accuracy)** is most informative.

**Q5. Distance Calculations (kNN)**  
You have 3 labeled points:

* A(2,4), Red
* B(4,4), Blue
* C(4,6), Red

Classify new point P(5,4):

1. Compute **Euclidean distance** from P to A, B, C.
2. Predict label using **1-NN**.
3. Predict label using **3-NN (majority vote)**.

1 .d(P,A)=√((5−2)²+(4−4)²)=**3.000**

d(P,B)=√((1)²+0)=**1.000**

d(P,C)=√((1)²+(−2)²)=**√5≈2.236**  
**2. 1-NN → Blue** (nearest B).

**3. 3-NN → Red** (A,R; B,B; C,R → 2 Red vs 1 Blue).

**Q6. K-fold Cross-Validation**  
You want to evaluate kNN with k=1,3,5 using 4-fold CV. Errors observed on each fold:

| **Fold** | **k=1** | **k=3** | **k=5** |
| --- | --- | --- | --- |
| 1 | 0.20 | 0.15 | 0.10 |
| 2 | 0.25 | 0.20 | 0.15 |
| 3 | 0.15 | 0.10 | 0.10 |
| 4 | 0.30 | 0.20 | 0.20 |

1. Compute **mean CV error** for each k.
2. WhMean error for k=1 = **0.225**

Mean error for k=1 = **0.225**

Mean error for k=3 = **0.1625**

Mean error for k=5 = **0.1375**

Best generalization = **k=5**

**Part B: Programming (3 Questions)**

**Q7. Build a Decision Tree (sklearn)**

1. Use sklearn.tree.DecisionTreeClassifier on the **Iris dataset**.
2. Train trees with max\_depth = 1, 2, 3.
3. Report training and test accuracy for each depth.
4. Discuss signs of **underfitting** vs **overfitting**.

Depth=1 underfits.

Depth=3 gives high accuracy; could overfit with more depth.

**Q8. kNN Classification (sklearn)**

1. Use sklearn.neighbors.KNeighborsClassifier on the Iris dataset (only 2 features: sepal length, sepal width).
2. Train models with k=1,3,5,10.
3. Plot the **decision boundaries** for each k.
4. Comment on how the boundaries change.

**Interpretation (manual answer)**

* k=1 → very wiggly (overfits).
* Larger k → smoother, more bias but less variance.

**Q9. Performance Evaluation Programming**

1. Train a kNN classifier (k=5) on Iris dataset.
2. Compute **confusion matrix** and display it with sklearn.metrics.confusion\_matrix.
3. Compute **accuracy, precision, recall, F1** using classification\_report.
4. Plot the **ROC curve** and compute **AUC**.