Milestone 1 Report

This report presents a summary of our implementation and evaluation of three classification models: K-Nearest Neighbors (KNN), Logistic Regression, and K-Means. We detail the implementation structure, the hyperparameter tuning process using cross-validation, and the resulting performance on both training and test sets.

K-Nearest Neighbors (KNN)

Overview: KNN is a simple non-parametric algorithm that classifies a sample based on the labels of its K nearest training samples in feature space. It requires storing the entire training set and computes predictions by majority vote over the closest K samples using Euclidean distance.

Implementation Details:

- `__init__(k)`: Initializes the model with K neighbors.
- `fit(X, y)`: Stores training data and labels.
- `predict(X_test)`: Computes distances and returns majority label of nearest neighbors.

Hyperparameter Tuning: Cross-validation was conducted over K values from 1 to 49. The best value found was **K = 14**.

Performance:

- **Train Accuracy**: 55.70%
- **Train F1-score**: 0.1989
- **Test Accuracy**: 55.00%
- **Test F1-score**: 0.2513

Logistic Regression

Overview: Multiclass logistic regression is a parametric linear classifier that estimates class probabilities using the softmax function. Weights are updated using gradient descent.

Implementation Details:

- `__init__(lr, max_iters)`: Initializes learning rate and number of iterations.
- `f_softmax(X)`: Computes the softmax scores.
- `fit(X, y)`: Runs gradient descent to optimize weights.
- `predict(X)`: Assigns class with highest softmax probability.

Hyperparameter Tuning: Cross-validation was performed on learning rates `[1e-5, 1e-4, 1e-3, 1e-2]` and iteration counts `[100, 200, 300]`. The best configuration found was **lr = 1e-4**, **max_iters = 100**.

Performance:

- **Train Accuracy**: 54.01%
- **Train F1-score**: 0.1403
- **Test Accuracy**: 53.33%
- **Test F1-score**: 0.1391

K-Means Clustering

Overview: K-Means is an unsupervised clustering algorithm. For classification, we assign labels to clusters via majority vote from training labels. We support multiple initializations (n_init) to mitigate poor local minima.

Implementation Details:

- `__init__(n_clusters, max_iters, n_init, scoring)`: Configures clustering options.
- `fit(X, y)`: Finds best clustering across n_init random starts.
- `predict(X_test)`: Assigns points to nearest cluster and returns associated majority label.

Fixed K Evaluation: With fixed **K = 8**, running 10 random initializations yielded the following result:

- Best Accuracy (K=20): **58.65%** (Train), **50.00%** (Test)
- Best F1 (K=18): **0.3030** (Train), **0.2060** (Test)

K-Range Cross-Validation: When testing K in range 3 to 20, best results were found with **K = 3,** using **accuracy** scoring and **K = 18** using **F1** scoring, probably due to Oversampling of a specific class. We can conclude that the Fixed K Evaluation was the most efficient in our case.

- **Train Accuracy**: 56.54%
- **Train F1-score**: 0.2422
- **Test Accuracy**: 51.67%
- **Test F1-score**: 0.1853

Discussion

All three models showed comparable performance. Logistic regression consistently underperformed relative to KNN and KMeans, possibly due to limited feature separability or suboptimal gradient convergence. KNN proved slightly more robust on the test set after hyperparameter tuning. KMeans showed surprising competitiveness despite being unsupervised.

The class imbalance in the dataset had a notable effect on performance and F1 scores. Weighted scoring strategies or data resampling could improve performance further.