Introduction to Machine Learning – Milestone 2 Report

Group: 392893_362668_364129

1. Introduction

The objective of this milestone was to implement and evaluate two machine learning methods on the Gaming and Mental Health dataset. We used a Multi-Layer Perceptron (MLP) for both classification and regression, and K-Means for classification by assigning labels to clusters through majority voting.

The classification task consists of predicting the addiction level, encoded as Low, Medium, or High, while the regression task consists of predicting the numerical addiction score. We evaluated classification using accuracy and macro-F1, and regression using mean squared error.

The project also required building a complete experimental pipeline, including data normalization, validation-based hyperparameter selection, L2 regularization, early stopping, confusion matrices, loss curves, and runtime measurements.

2. Methodology

Data preprocessing and validation strategy

All input features were normalized using the mean and standard deviation computed only from the training subset. This was done to avoid information leakage from validation or test data into preprocessing. When the --test flag was not used, the original training set was randomly split into 80% training data and 20% validation data. The validation set was used only for model selection and hyperparameter tuning. When --test was used, the full training set was used for training and the official test set was used only for final evaluation.

For classification, labels were converted to one-hot vectors before training the MLP. Predictions were converted back to class indices by taking the output neuron with the largest value. For regression, labels were reshaped into column vectors and the network directly predicted one scalar addiction score.

The final test results were obtained only after selecting the best hyperparameters on the validation set; we then reran the selected configuration with the `--test` flag, using the full training set for training and the official test set only for final evaluation.

Multi-Layer Perceptron

We implemented a fully connected MLP from scratch. The network stores one weight matrix and one bias vector for each layer. During the forward pass, each layer computes a pre-activation followed by an activation function. Hidden layers use ReLU activations. For classification, the output layer uses a Softmax activation and is trained with categorical Cross-Entropy on one-hot labels. This is the standard setup for mutually exclusive multi-class classification, since Softmax converts the output scores into class probabilities that sum to one, and Cross-Entropy directly penalizes low probability assigned to the correct class.

For regression, the output layer uses an Identity activation so that predictions are not restricted to the interval [0, 1]. This is necessary because the target addiction score ranges from 0 to 10.

Backpropagation was implemented manually. Gradients are propagated from the output layer to the first hidden layer using the chain rule. Activation gradients are computed using the pre-activation values. L2 regularization is added to all weight matrices but not to biases. Biases are updated using the summed delta over each mini-batch. The MLP is trained by mini-batch gradient descent, with shuffled batches at every epoch.

We also implemented early stopping. During training, both training and validation losses are stored. If the validation loss does not improve by at least a tolerance value for a fixed patience window, training stops early. The best weights observed on the validation loss are restored at the end, so the final model corresponds to the best validation epoch rather than simply the last epoch.

The MLP hyperparameter grid was:

- Hidden architecture: [16], [32], [64], [32, 16]
- Learning rate: 0.0001, 0.001, 0.01
- L2 regularization λ: 0.0, 0.001, 0.01
- Batch size: 16, 32, 64

For classification, the selected model maximized validation macro-F1. We used macro-F1 rather than only accuracy because the class distribution is imbalanced, with the High class being much rarer than the others. For regression, the selected model minimized validation MSE.

K-Means for classification

K-Means was implemented from scratch using vectorized Euclidean distances. Cluster centers are initialized by randomly selecting training samples. At each iteration, samples are assigned to the nearest center and each center is recomputed as the mean of the assigned samples. Empty clusters are reinitialized using a random training sample.

Since K-Means is unsupervised, we converted it into a classifier by assigning one label to each cluster after fitting. For each cluster, the label is chosen by majority vote among the training labels assigned to that cluster. A new point is classified by assigning it to the nearest center and returning the label of that center. We tuned K over 2, 3, 4, 5, 6, 8, 10 and selected the model with the highest validation macro-F1.

During grid search, we kept and reused the best trained model directly instead of recreating a new model with the same hyperparameters, since a new random initialization could lead to slightly different validation and test performances.

3. Experiments and Results

Classification

Best configuration for classification using MLP: hidden architecture [16], learning rate 0.01, L2 regularization 0.0, and batch size 16. The MLP obtained 86.25% validation accuracy and 0.782 validation macro-F1. On the official test split, the MLP obtained 87.00% test accuracy and 0.811 test macro-F1.

The best validation performance for K-Means was obtained with K = 4, giving 73.75% validation accuracy and 0.429 validation macro-F1. On the official test set, K-Means obtained 70.00% accuracy and 0.275 macro-F1.

Modèle	Hyperparamètres	Val. Acc	Val. F1	Test Acc	Test F1
MLP	[16], lr=0.01, λ=0.0, batch=16	86.3%	0.782	87.0%	0.811
K-Means	K=4	73.8%	0.429	70.0%	0.275

The MLP clearly outperformed K-Means. The reason is that the MLP is a supervised model trained directly on labels, while K-Means is unsupervised and only receives labels after clustering

through majority voting. Therefore, the clusters do not necessarily align with the addiction-level classes. This is confirmed by the test results: the MLP reached a much higher macro-F1 score than K-Means.

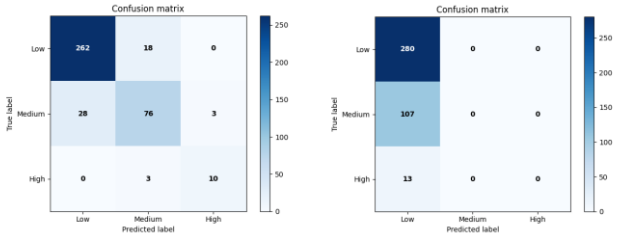

Figure 1 and 2: Test confusion matrices for MLP and K-Means.

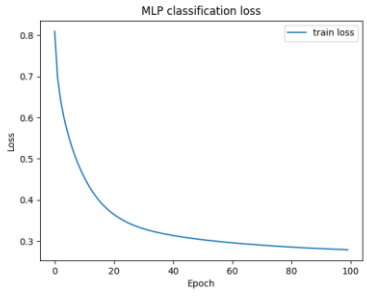

Figure 3: MLP classification Cross-Entropy *loss curve.*

The MLP separates the three classes reasonably well. The test confusion matrix shows that most Low samples are correctly classified, while the main confusion occurs between Low and Medium. The rare High class is also handled well: 10 out of 13 High samples are correctly classified, with only 3 assigned to Medium. This explains why the MLP obtains a much higher macro-F1 than K-Means.

Macro-F1 is more informative than accuracy for this classification task because the class distribution is imbalanced. This is especially visible for K-Means: although it reaches 70.0% test accuracy, its macro-F1 is only 0.275 because it predicts almost all samples as the majority class.

Regression

Best regression configuration: hidden architecture [16], learning rate 0.01, L2 regularization 0.001, and batch size 64. The MLP regression model obtained training MSE 0.933 and validation MSE 1.107 during model selection. After retraining on the full training set and evaluating on the official test split, it obtained training MSE 0.947 and test MSE 0.999.

The regression MLP generalized reasonably well because the validation and test MSE values are close. The gap between training MSE and validation/test MSE remains small, suggesting that the model did not strongly overfit. The selected value $\lambda = 0.001$ shows that L2 regularization was useful for this configuration.

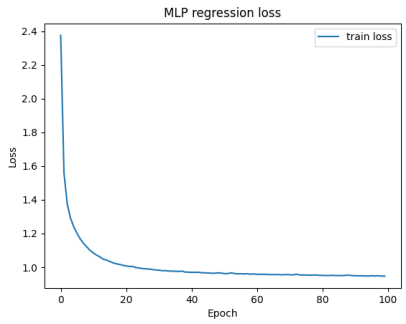

Figure 4: MLP regression MSE loss curve.

Runtime comparison

Model / task	Grid search time	Final training time
MLP classification	107.17 s	1.49 s
K-Means classification	0.20 s	0.022 s
MLP regression	77.32 s	0.38 s

MLP grid search was much slower than K-Means because it trains many networks over several hyperparameter combinations. However, prediction time was negligible for all methods.

4. Discussion and Conclusion

The best overall classification model in this milestone was the MLP. It reached 87.00% test accuracy and 0.811 test macro-F1, clearly outperforming K-Means, which reached 70.00% test accuracy and only 0.275 test macro-F1. The confusion matrix shows that the MLP correctly classifies most Low samples and also handles the rare High class well, with 10 out of 13 High samples correctly classified. Most remaining errors occur between Low and Medium. K-Means was less competitive, which is expected because it is unsupervised: it groups samples according to distances in feature space, but these groups do not necessarily correspond to addiction levels.

Compared with Milestone 1, the MLP performs slightly worse than Logistic Regression for classification. Logistic Regression reached 88.50% test accuracy and 0.847 test macro-F1, while our best MLP reached 87.00% and 0.811. This suggests that the classification problem is already well captured by a linear decision boundary, and that the additional flexibility of the MLP does not necessarily improve generalization on this dataset. However, the MLP still clearly improves over KNN from Milestone 1, which reached 79.75% test accuracy and 0.590 test macro-F1.

Using Softmax with Cross-Entropy substantially improved macro-F1 compared with the initial MSE-based classifier, especially because the model became able to predict the rare High class instead of ignoring it. This shows that the choice of loss function is important for imbalanced multi-class classification.

For regression, the MLP achieved a test MSE of 0.999. This is very close to the Linear Regression test MSE from Milestone 1, which was 0.994, and better than the KNN regression test MSE of 1.865. Since the validation MSE was 1.107 and the test MSE was close to it, the MLP generalized reasonably well. However, the fact that it does not clearly outperform Linear Regression again suggests that the regression target is mostly explained by relatively simple relationships in the features.

Overall, this milestone shows that a manually implemented MLP can solve both classification and regression tasks effectively when combined with careful preprocessing, validation-based hyperparameter selection, L2 regularization, Cross-Entropy for classification, and early stopping. Nevertheless, the comparison with Milestone 1 indicates that more complex models are not always better: Logistic Regression and Linear Regression remain very competitive on this dataset. Future improvements could include class-weighted Cross-Entropy, stratified validation splits, or resampling strategies to further improve minority-class performance.