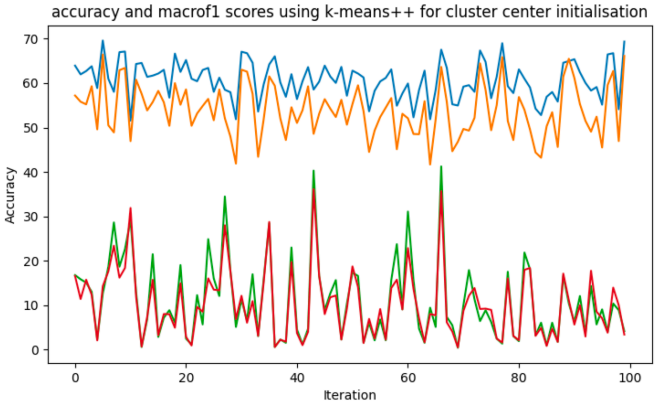
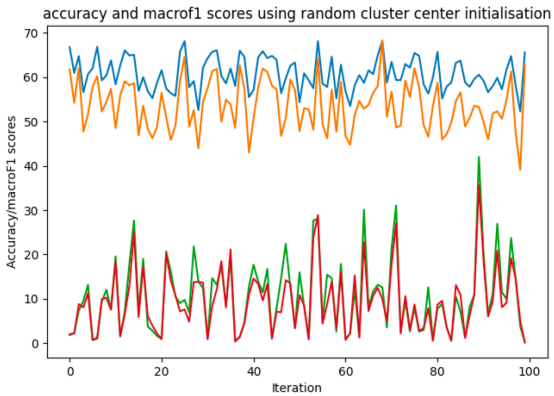
Machine Learning Project Milestone 1 Report
347304_314355_345583_project
Nikolay Mikhalev, Erik Georges, Gafsi Amene

Introduction:
In this project, our aim was to implement a few rather basic machine learning models such as: k-means clustering, logistic regression and SVM. We were then supposed to experiment with our models and tend to get the best results possible on our data classification task, using various techniques such as cross validation for finding the beat hyper parameters, that would maximise the accuracy of prediction and the macroF1 score.

Presentation of the work:
K-means:
First of the models to implement was k-means clustering. Out of the 3 different ones present in ms1, this is the only model that is technically unsupervised, however, using the fact that we're working with tabled data, we can use this to our advantage, and that's exactly what we did. Implementing and running a classic version of k-means (initialising clusters randomly) didn't give us any reasonable result, visibly overfitting on the training data compared to the poor results obtained with the test data. We, then, tried using a modified version of k-means called k-means++ where the only part that's modified is the algorithm for cluster initiation. However, this approach did not work either. The results were rather similar to those of classical k-means, you can observe them on the graphs below (macroF1 scores were multiplied by 100 to be visible next to the accuracy).On the graphs, green and red represent testing results, while blue and orange represent training results. Finally, we implemented a version of cluster initialisation that would initialise cluster centres in the mean of the labelled samples. This time, it gave us a satisfying result: 73.5% train accuracy and 73.1% test accuracy which is a great leap compared to the previous results. Needless to say, this method always yields the same result because of the predicted nature of cluster initialisation. It also only works with the right amount of clusters, however logically the other two ways we tried did not grant any visible results with k different from 10 (the amount of data table classes), and the algorithm would usually converge after max 10 cycles, hence using cross validation here would be unnecessary.





Logistic regression :
Secondly, we implemented a logistic regression model. This model is a classification algorithm used to predict categorical outputs. In our implementation, we defined it as a class with an initialization function and two methods, fit and predict. The fit method trains the model on the given training data and returns predicted labels, while the predict method uses the trained weights to run predictions on the test data. We used multi-class entropy as our loss function and stochastic gradient descent to update the weights during training. This implementation provides an efficient and accurate approach for classification tasks, especially when dealing with tabular data.

To train our logistic regression model, we initialized the weights randomly and updated them at each iteration based on the gradient of the multi-class cross-entropy loss function with respect to the weights. Then, we applied the softmax function to the linear combination of the input data and learned weights to predict the class labels of new data points.
To find the best hyper-parameters for our model, we used cross-validation. We split our dataset into 4 folds and used 3 folds for training and 1 fold for validation. We repeated this process for all possible combinations of hyper-parameters, including learning rate and maximum number of iterations. After cross-validation, we found that the best hyper-parameters were a learning rate of 0.001 and a maximum number of iterations of 500, giving us an accuracy of around 93% on the test set.
Overall, our implementation of logistic regression was a powerful and effective method for our classification task, with the added benefit of being relatively simple to implement and interpret.

SVM :

Finally, we used the scikit-learn SVM implementation as our SVM model. We looked into the linear and rbf kernels and tried to find the best hyper-parameters for each of them by using the cross-validation method previously described. The penalty term C which is used in both kernels was picked from the values [0.001, 0.01, 0.1, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10], the results were the following:

- linear kernel: we got a macro F1-score of 0.928 with C = 0.001 and a score of 0.935 for all other values of C, which corresponds to an accuracy of 94.5% on the test set. For this kernel the choice of the hyper-parameter seems to have a rather small impact on the accuracy

- rbf kernel: we tried 0.0001, 0.001, 0.01, 0.1, 1 and 10 as values for gamma (the scaling factor for the distance between points). Gamma = 0.001 was the value which gave the best results for every value of C. With that value for gamma, we got macro F1-scores of 0.453 with C <= 0.01, 0.678 with C = 0.1, 0.940 with C = 1, 0.946 with C = 2 and 0.945 with C > 2, choosing C = 2, we got an accuracy of 96.6% on the test set. Using gamma = 0.0001 gives slightly poorer results (score of 0.921 with C = 2), however the other values performed significantly worse (gamma = 0.01 had a macro F1-score of 0.122 at best and the others had a score below 0.025 regardless of the value of C).

The SVM model therefore seems to be a very effective method to solve our problem. The linear kernel outperforms the two other models regardless of the choice of the hyper-parameter. The rbf kernel is more dependent on the choice of the hyper-parameters, but with the right ones it is the best performing model out of all the ones we tested.

Cross Validation with fold:

A very useful and technically simple tool to find the best hyper parameters for our models. If it doesn't really play any role with k-means and only small one with logistic regression, helping check that we're using an appropriate learning rate, this tool really gets handy with the SVM, where there are multiple parameters to test.
One important part of cross validation was also using folds: instead of taking (for example) the first fourth of training data as validation data and the rest as training, we would iterate and in this case do 4 tests, then taking the average of the four and assigning the params in question this very score. If we didn't do this, then the information would be rather random and imprecise because of the data shuffling, we would get drastically different results each time.

Conclusion:

In conclusion, through our implementation of k-means clustering, logistic regression, and SVM, we were able to successfully classify our data and achieve high accuracy rates. By experimenting with different techniques, such as cross-validation to find the best hyperparameters, we were able to optimize our models and improve our results even further. However, it is important to note that depending on the model we use, we get different accuracies. Therefore, choosing the best model for our specific needs is crucial. Overall, our project demonstrated the effectiveness and versatility of these basic machine learning models for classification tasks, and highlighted the importance of careful experimentation and optimization in achieving the best possible results.