WDBC Coursework

Davide Mecugni

December 8, 2023

Abstract

WDBC contains 569 instances of breast cancer data collected by the University of Wisconsin. Each instance is labeled as M (malignant) or B (benign). This Coursework aims to analyze the dataset using PCA and KNN in tasks 1 and 2. Task 3 uses a Deep Neural Network to classify the instances. I proposed an optional Enseble NN design. Task 4 discusses the advantages and disadvantages of each method.
All the code is written in Python. All the results are obtained using fixed seeds(in Numpy, Pytorch, and random) so that the exact results can be calculated every time.
The code is run on an HP Pavillion laptop using AMD Ryzen 7 5700U at 1.80 GHz.

Task 1

Task 1 was completed using the CSV python module for retrieving the dataset from the wdbc.data file. Once the dataset is retrieved it can be further split into training/test using the random library for pseudo-randomic splitting. The feature reduction was achieved using the PCA module from sklearn.

Task 2

Once I retrieved the dataset in task 1, I applied KNN to the different reduced normalized datasets and the full dataset. I used a normalized dataset because the results proved to be much better than the un-normalized one. I generated tables and graphs to help me understand the best number of features to use for further analysis. To get more stable results, I applied a 5-fold cross-validation, averaging the final results. Without using cross-validation, I obtained precisions and other values at 100%, after using cross-validation I obtained more reasonable results.
The tables and graphs show the various averaged metrics between the folds for each reduced dimension dataset at different k values:

Statistics for 5 dimentions

	1	3	5	7	9
accuracy	0.9101	0.9337	0.9337	0.9325	0.9396
precision	0.8621	0.904	0.914	0.9134	0.9174
recall	0.8992	0.9195	0.907	0.903	0.9185
f1	0.8769	0.9085	0.9072	0.9052	0.9154

Statistics for 10 dimentions

	1	3	5	7	9
accuracy	0.9195	0.9361	0.9385	0.9373	0.9349
precision	0.8891	0.9243	0.931	0.9257	0.9221
recall	0.8941	0.8993	0.8993	0.9033	0.8993
f1	0.8883	0.9099	0.913	0.912	0.9084

Statistics for 15 dimentions

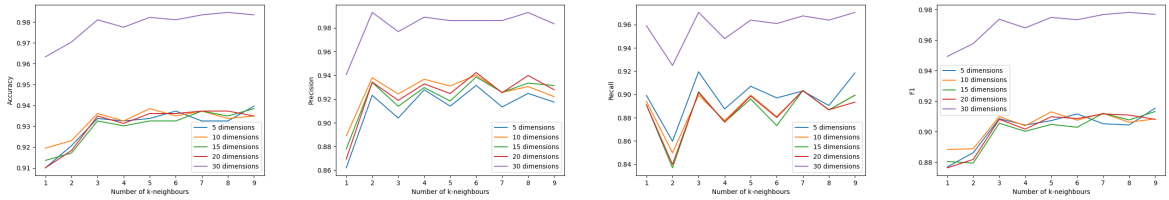
	1	3	5	7	9
accuracy	0.9136	0.9325	0.9325	0.9373	0.9385
precision	0.878	0.9139	0.9184	0.9256	0.9314
recall	0.8911	0.9024	0.8961	0.9034	0.8994
f1	0.8805	0.9055	0.9047	0.912	0.9132

Statistics for 20 dimentions

	1	3	5	7	9
accuracy	0.9101	0.9349	0.9361	0.9373	0.9349
precision	0.8694	0.9189	0.9246	0.9253	0.9278
recall	0.891	0.9016	0.8986	0.9034	0.8934
f1	0.8763	0.9081	0.9098	0.9116	0.9081

Statistics for 30 dimentions

	1	3	5	7	9
accuracy	0.9633	0.9811	0.9822	0.9834	0.9834
precision	0.9406	0.9768	0.9861	0.9861	0.9834
recall	0.9591	0.9705	0.9639	0.9675	0.9705
f1	0.9493	0.9736	0.9748	0.9767	0.9768



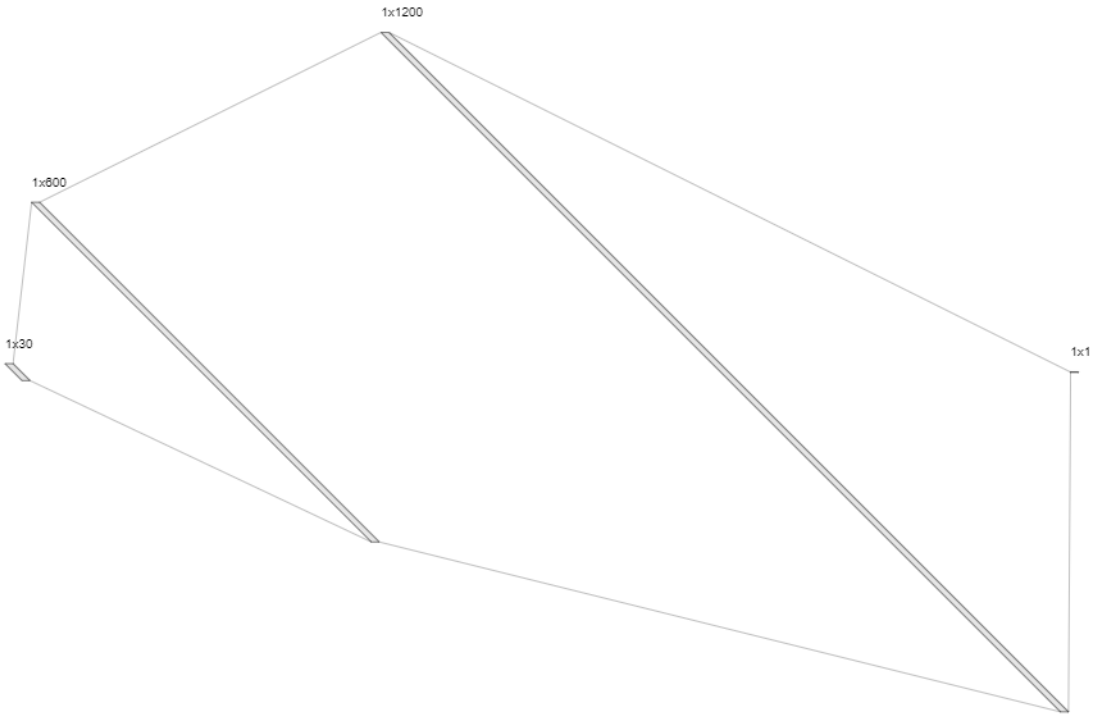
As can be easily seen in the graphs, the best metrics are achieved using 30 features.
In particular, the best performance on test data is achieved using KNeighbors with K=9 and 30 features, scoring:

Accuracy	98.34%
Precision	98.34%
Recall	97.05%
F1	97.68%

The computing time on my machine for compression and KNN on all 5 datasets is 0.59s.

Task 3

The MLP binary classifier is coded in Python using the Pytorch framework.
I tried different designs, and the one that worked the best is the following:



The design is a fully connected neural network of size 30x600x1200x1. Each layer uses batch normalization and dropout at 0.5.
The model uses Stochastic Gradient Descent, Binary Cross Entropy loss function and sigmoid activation function[1] .

[1]The choice is influenced by an article by Nghi Huynh at link

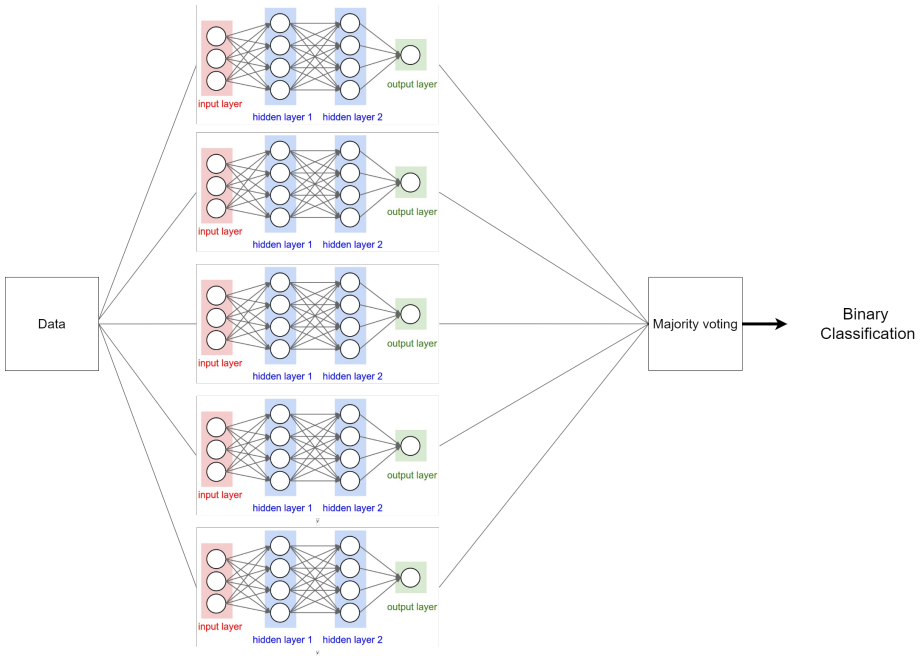
The model is trained using 5-fold cross-validation. It is further observed that the best performance is achieved with a batch size of 40(10% of the training set) and 200 epochs.
The initial learning rate is 0.06. The model adopts a scheduler that halves the learning rate when an improvement in the loss function is not seen for a patience constant(800 in my implementation). The final averaged results on test data between the 5 folds using this architecture are:

Accuracy	98.2393%
Precision	99.394%
Recall	95.804%
F1	97.45%

The average time for training a single MLP on my machine is 10.52s.

Optional Ensemble model

Furthermore, I tried using an Ensemble Model consisting of the aggregated vote of 5 MLP models obtained in task 3. The models make different choices given that they were trained from different initial weights. The underlying assumption is that the models make different errors, so using the most popular label between the models reduces the overall error.
Here is the design implemented:



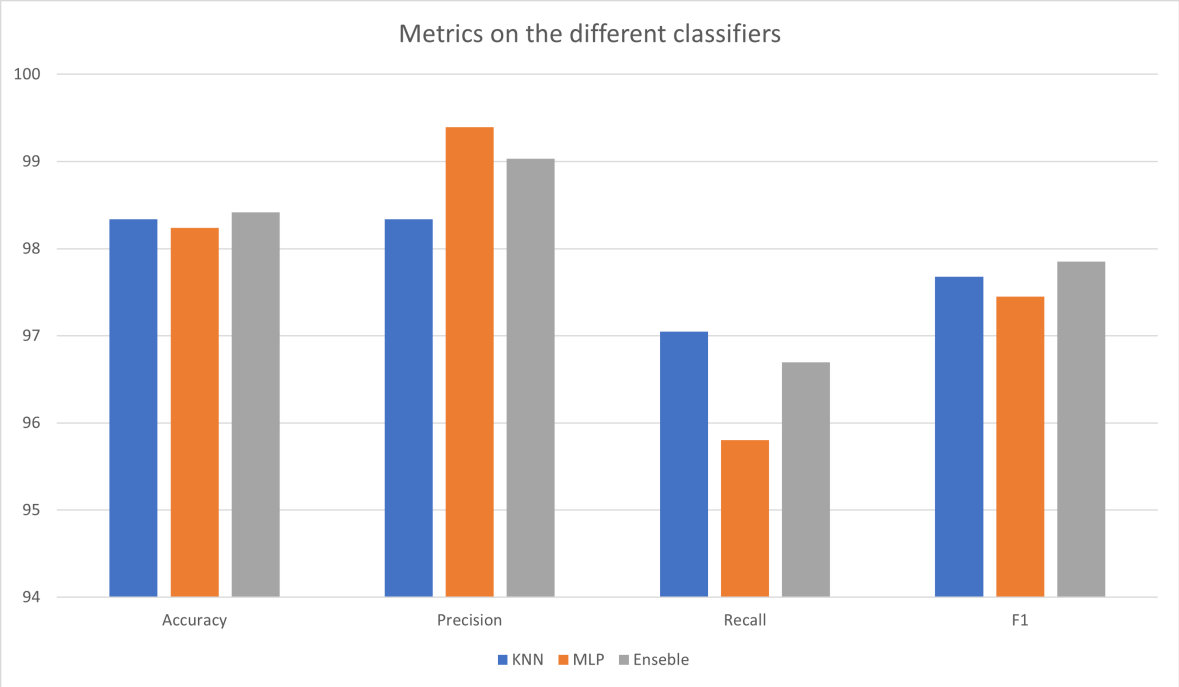
[2]

The results on the complete dataset are:

Accuracy	98.418%
Precision	99.034%
Recall	96.698%
F1	97.852%

[2]Image on binary NN classifier at link

Task 4



It must be noted that the computational time is far greater for MLP compared to KNN. This is justified by the fact that no training is required for KNN. Given that KNN is an instance-based ML algorithm, it finds the closest recordings in the training data to the data seen in testing. While this method is computationally efficient, it doesn't generate any underlying model of the data. MLP can identify patterns within data.

The most important metric in a binary classifier of this type is recall, given the cost of a false negative(the patient is told they don't have breast cancer when in reality they have it). The average recall of the best KNN(97.1%) is much better than the average recall of the MLP classifier(95.8%). The average precision of KNN(98.3%) is worse than MLP(99.4%), this means that if a patient is diagnosed with breast cancer, the probability of the diagnosis being true is better using MLP than KNN. The aggregated F1 score, 97.7% for KNN and 97.5% for MLP is fairly similar.

KNN proves to be a better algorithm for identifying all the people who have breast cancer, which would be the most important metric in this classifier.

MLP proves to be better at giving truthful positive labeling.

The best accuracy and F1 score are achieved by the optional ensemble model proposed in the last section.