Mikias Berhanu | 2021280115

Assignment 7
K-Nearest Neighbors & Artificial Neural Networks

Lab Experiments with K-Nearest Neighbors and ANNs

K-Nearest Neighbor is one of the most widely used Supervised Learning algorithms which is used for both classification and regression tasks. This algorithm is also used for imputing missing values and resampling datasets. The algorithm works by considering K Nearest Neighbors to predict the class of a data point or some sort of continuous numeric value. The algorithm has different learning methods: **Instance-Based Learning,** in this form of learning weights are not derived from training data rather the entire dataset is used to predict the output for the unseen data. **Lazy Learning**: is the form of learning where the model is not trained on a data and learning process happens only when prediction is needed on the new data instance. **Non parametric**: this form of learning has no predefined form of mapping function.

Artificial Neural Networks or ANN are designed to mimic the functioning of the human brain which allow computers to learn from data and make decisions without being intervened by humans. Their name is derived in reference to the human brain structure called neurons. ANNs are widely used in Machine Learning, Deep Learning and Artificial Intelligence. One thing to note here is that all the three mentioned fields are different in their implementation and what they stand for. ANNs are one of the algorithms used in Machine Learning and Deep learning. Artificial Neural networks are composed of several layers of nodes or neurons where each node has its own purpose and contains weights and threshold values called bias which is later used for the decision making process. Usually ANNs start with input layers or input nodes and then middle layers which are commonly used for data processing and data sampling then the obvious output layer. Depending on the given task the output layer may give different results, for regression tasks the output will be continuous numeric value, for classification tasks it will be a probability between 0 and 1 for each class available in the data.

This experiment is using a CTG dataset, this dataset has 2126 entries with 22 columns, compared to the previous dataset this one is a bit larger in size and has many features. Each entry is part of a class either 1, 2 or 3. The dataset is not balanced like the previous one. 1655 entries are part of class 1, 295 are part of class 2 and 176 are part of class 3. When we split the dataset we have to make sure that each class is represented well both in the training and testing set otherwise it will create baise in the results. Due to this reason rather than a plain 80:20 ratio split, I used stratified sampling where the data is splitted into 5 folds and each class will be represented well and helps the performance of the model as well.

	LB	AC	FM	UC	DL	DS	DP	ASTV	MSTV	ALTV	...	Min	Max	Nmax	Nzeros	Mode	Mean	Median	Variance	Tendency	NSP
0	120	0.000000	0.0	0.000000	0.000000	0.0	0.0	73	0.5	43	...	62	126	2	0	120	137	121	73	1	2
1	132	0.006380	0.0	0.006380	0.003190	0.0	0.0	17	2.1	0	...	68	198	6	1	141	136	140	12	0	1
2	133	0.003322	0.0	0.008306	0.003322	0.0	0.0	16	2.1	0	...	68	198	5	1	141	135	138	13	0	1
3	134	0.002561	0.0	0.007682	0.002561	0.0	0.0	16	2.4	0	...	53	170	11	0	137	134	137	13	1	1
4	132	0.006515	0.0	0.008143	0.000000	0.0	0.0	16	2.4	0	...	53	170	9	0	137	136	138	11	1	1

5 rows × 22 columns

The first 5 entries of the dataset

	LB	AC	FM	UC	DL	DS	DP	ASTV	MSTV	ALTV	...	Min	Max
count	2126.000000	2126.000000	2126.000000	2126.000000	2126.000000	2126.000000	2126.000000	2126.000000	2126.000000	2126.00000	...	2126.000000	2126.000000
mean	133.303857	0.003170	0.009474	0.004357	0.001885	0.000004	0.000157	46.990122	1.332785	9.84666	...	93.579492	164.025400
std	9.840844	0.003860	0.046670	0.002940	0.002962	0.000063	0.000580	17.192814	0.883241	18.39688	...	29.560212	17.944183
min	106.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	12.000000	0.200000	0.00000	...	50.000000	122.000000
25%	126.000000	0.000000	0.000000	0.001876	0.000000	0.000000	0.000000	32.000000	0.700000	0.00000	...	67.000000	152.000000
50%	133.000000	0.001630	0.000000	0.004482	0.000000	0.000000	0.000000	49.000000	1.200000	0.00000	...	93.000000	162.000000
75%	140.000000	0.005631	0.002512	0.006525	0.003264	0.000000	0.000000	61.000000	1.700000	11.00000	...	120.000000	174.000000
max	160.000000	0.019284	0.480634	0.014925	0.015385	0.001353	0.005348	87.000000	7.000000	91.00000	...	159.000000	238.000000

8 rows × 22 columns

Numerical description of the data

Once the data is loaded properly we can go ahead and extract the X and y values just like before. Scale the values back to 0 - 1 since the values have different ranges. The split the dataset using stratified sampling. Once the dataset is splitted properly we can start training our model. The performance of the model can be checked using previously mentioned metrics. This model has a f1-score of 91%, recall score 91% and precision score 91%.

The other experiment is based on ANNs, which in this case I used a Multilayer Perceptron Classifier from scikit learn library. The performance was much better which gave a f1-score of 93%, recall score 93% and precision score 93%. The settings

for this model are all the same with KNNs and nothing has been changed for the sake of comparing the performance of the two models.

Summary

K-Nearest Neighbor is one of the most widely used Supervised Learning algorithms which is used for both classification and regression tasks. This algorithm is also used for imputing missing values and resampling datasets.Artificial Neural Networks or ANN are designed to mimic the functioning of the human brain which allow computers to learn from data and make decisions without being intervened by humans. Their name is derived in reference to the human brain structure called neurons. ANNs are widely used in Machine Learning, Deep Learning and Artificial Intelligence. In this experiment we have performed classification tasks using two different models KNNs and MLPC. The performance of the classification was examined using confusion matrix, precision, recall and f1-score. When our data is not balanced properly we have to represent all the classes properly to avoid baise in our prediction. The source code of this project is linked here