Lab 2: Nearest Neighbor (`sklearn Version`)

Evgueni N. Smirnov
smirnov@maastrichtuniversity.nl

11 November 2020

1. Introduction

Neareast neighbor is one of the most popular and simple approaches for classification and prediction. In this lab you will study a particular implementation of this approach in `Python`. You will be asked to add several methods to the implementation and then to test these methods for classification and regression problems.

2. Lab Tasks

A. Study class `kNN` of the nearest neighbor approach provided in the `Jupiter` notebook `Lab-02` that accompanies this lab. We note that this class is designed for data sets with numeric input attributes.

B. Add to class `kNN` method `normalize` that normalizes the input attributes of the training data `X_train` and test data `X_test`. We note that attribute normalization is important since all the attributes receive equal weights when instance distances are being computed.

To implement method `normalize` you might use method `max` of `pandas.DataFrame` since both `X_train` and `X_test` are `pandas.DataFrame`.

Test the `kNN` classifier on the `diabetis` and `glass` classification data sets (see Appendix A) for the case when the data is not normalized and the case when the data is normalized. Indicate whether the training and hold-out accuracy rates improve with normalization.

For testing the `kNN` classifier you might use the script provided in the `Jupiter` notebook. The script provides a plot with training and hold-out accuracy rates in function of parameter `k` of the `kNN` classifier.

Test the `kNN` classifier on the `glass` classification data sets the data is normalized for different values of the `exp` parameter of the Minkowski distance. Indicate whether the training and hold-out accuracy rates changes due to `exp`. For this task you might use the second testing script provided in the Jupiter note.

C. Add to class `kNN` method `getClassProbs` that computes for all the test instances in `X_test` the posterior class probabilities. This means that the method computes for each row (instance) in `X_test` a row with probability of class 1, probability of class 2, and probability of class N. Combine the rows of the posterior class probabilities in `pandas.DataFrame` object that will be the output of the method `getClassProbs`.

D. Add to class `kNN` method `getPrediction` that computes for all the test instances in `X_test` regression values for the output attribute. This means that the method computes for each instance (row) in `X_test` a regression value equal to the average of `y` values in `Y_train` of the `k`-nearest neighbors of the instance in `X_train`. Combine the computed regression values for all the instances in `X_test` in `pandas.DataFrame` object that will be the output of the method `getPrediction`.

   Test the `method` `getPrediction` on the `autoprice` data set which is a regression data set (see Appendix A). For that purpose you can adapt the test script tha you have already used for Task B. Please use mean absolute error as the main metric for estimating regression performance[1] instead of the accuracy rate. To compute the mean absolute error you can use method `mean_absolute_error` from `sklearn.metrics`.

Report: Prepare a pdf file of the Jupiter notebook with your code for tasks B, C, and D. Provide answers in markdown fields for questions from task B.

[1] Mean absolute error (MAE) is the average absolute difference between what the predicted value $\hat{y}_i$ and true value y_i for all the test instances (x_i, y_i) in training data D; i.e., $MAE = \sum_{(x_i,y_i)\in D} |y_i - \hat{y}_i|$.

Appendix A: Data Sets

I. Diabetes Data (Classification)

1. Title: Pima Indians Diabetes Database

2. Sources:
   (a) Original owners: National Institute of Diabetes and Digestive and Kidney Diseases
   (b) Donor of database: Vincent Sigillito (vgs@aplcen.apl.jhu.edu)
   Research Center, RMI Group Leader
   Applied Physics Laboratory
   The Johns Hopkins University
   Johns Hopkins Road
   Laurel, MD 20707
   (301) 953-6231

3. Relevant Information:
   Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage. ADAP is an adaptive learning routine that generates and executes digital analogs of perceptron-like devices. It is a unique algorithm; see the paper for details.

4. Number of Instances: 768

5. Number of Attributes: 8 plus class

6. For Each Attribute: (all numeric-valued)
   1. Number of times pregnant
   2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
   3. Diastolic blood pressure (mm Hg)
   4. Triceps skin fold thickness (mm)
   5. 2-Hour serum insulin (mu U/ml)
   6. Body mass index (weight in kg/(height in m)^2)
   7. Diabetes pedigree function
   8. Age (years)
   9. Class variable (0 or 1)

7. Missing Attribute Values: None

8. Class Distribution: (class value 1 is interpreted as "tested positive for diabetes")

Class Value	Number of instances
0	500
1	268

9. Brief statistical analysis:

Attribute number:	Mean:	Standard Deviation:
1.	3.8	3.4
2.	120.9	32.0
3.	69.1	19.4
4.	20.5	16.0
5.	79.8	115.2
6.	32.0	7.9
7.	0.5	0.3
8.	33.2	11.8

Relabeled values in attribute 'class'
From: 0 To: tested_negative
From: 1 To: tested_positive

II. Glass Data (Classification)

1. Title: Glass Identification Database

2. Sources:
(a) Creator: B. German
-- Central Research Establishment
Home Office Forensic Science Service
Aldermaston, Reading, Berkshire RG7 4PN
(b) Donor: Vina Spiehler, Ph.D., DABFT
Diagnostic Products Corporation
(213) 776-0180 (ext 3014)

3. Relevant Information:n
Vina conducted a comparison test of her rule-based system, BEAGLE, the nearest-neighbor algorithm, and discriminant analysis. BEAGLE is a product available through VRS Consulting, Inc.; 4676 Admiralty Way, Suite 206; Marina Del Ray, CA 90292 (213) 827-7890 and FAX: -3189. In determining whether the glass was a type of "float" glass or not, the following results were obtained (# incorrect answers):

Type of Sample	Beagle	NN	DA	
Windows that were float processed	(87)	10	12	21
Windows that were not:	(76)	19	16	22

The study of classification of types of glass was motivated by criminological investigation. At the scene of the crime, the glass left can be used as evidence...if it is correctly identified!

4. Number of Instances: 214

5. Number of Attributes: 10 (including an Id#) plus the class attribute
   -- all attributes are continuously valued

6. Attribute Information:
   1. RI: refractive index
   2. Na: Sodium (unit measurement: weight percent in corresponding oxide, as are attributes 4-10)
   3. Mg: Magnesium
   4. Al: Aluminum
   5. Si: Silicon
   6. K: Potassium
   7. Ca: Calcium
   8. Ba: Barium
   9. Fe: Iron
   10. class: type of glass
      -- 1 building_windows_float_processed
      -- 2 building_windows_non_float_processed
      -- 3 vehicle_windows_float_processed
      -- 4 vehicle_windows_non_float_processed (none in this database)
      -- 5 containers
      -- 6 tableware
      -- 7 headlamps

7. Missing Attribute Values: None

Summary Statistics:

Attribute:	Min	Max	Mean	SD	Correlation with class
2. RI:	1.5112	1.5339	1.5184	0.0030	-0.1642
3. Na:	10.73	17.38	13.4079	0.8166	0.5030
4. Mg:	0	4.49	2.6845	1.4424	-0.7447
5. Al:	0.29	3.5	1.4449	0.4993	0.5988
6. Si:	69.81	75.41	72.6509	0.7745	0.1515
7. K:	0	6.21	0.4971	0.6522	-0.0100
8. Ca:	5.43	16.19	8.9570	1.4232	0.0007
9. Ba:	0	3.15	0.1750	0.4972	0.5751
10. Fe:	0	0.51	0.0570	0.0974	-0.1879

8. Class Distribution: (out of 214 total instances)
   -- 163 Window glass (building windows and vehicle windows)
      -- 87 float processed
         -- 70 building windows
         -- 17 vehicle windows
      -- 76 non-float processed
         -- 76 building windows
         -- 0 vehicle windows
   -- 51 Non-window glass
      -- 13 containers
      -- 9 tableware
      -- 29 headlamps

Relabeled values in attribute class

From: '1'	To: 'build wind float'
From: '2'	To: 'build wind non-float'
From: '3'	To: 'vehic wind float'
From: '4'	To: 'vehic wind non-float'
From: '5'	To: containers
From: '6'	To: tableware
From: '7'	To: headlamps

III. Auto Price Data (Regresion)

This data set consists of three types of entities:
(a) the specification of an auto in terms of various characteristics;
(b) its assigned insurance risk rating,;
(c) its normalized losses in use as compared to other cars.
The second rating corresponds to the degree to which the auto is more risky than its price indicates. Cars are initially
assigned a risk factor symbol associated with its price. Then, if it is more risky (or less), this symbol is adjusted by
moving it up (or down) the scale. Actuarians call this process "symboling". A value of +3 indicates that the auto is
risky, -3 that it is probably pretty safe.The third factor is the relative average loss payment per insured vehicle year.
This value is normalized for all autos within a particular size classification (two-door small, station wagons,
sports/speciality, etc...), and represents the average loss per car per year.
- Note: Several of the attributes in the database could be used as a "class" attribute.
The original data (from the UCI repository)
(http://www.ics.uci.edu/~mlearn/MLSummary.html) has 205 instances
described by 26 attributes :
- 15 continuous
- 1 integer
- 10 nominal

The following provides more information on these attributes:

1. symboling: -3, -2, -1, 0, 1, 2, 3.
2. normalized-losses: continuous from 65 to 256.
3. make: alfa-romero, audi, bmw, chevrolet, dodge, honda, isuzu, jaguar, mazda, mercedes-benz, mercury, mitsubishi, nissan, peugot, plymouth, porsche, renault, saab, subaru, toyota, volkswagen, volvo
4. fuel-type: diesel, gas.
5. aspiration: std, turbo.
6. num-of-doors: four, two.

7. body-style: hardtop, wagon, sedan, hatchback,convertible.
8. drive-wheels: 4wd, fwd, rwd.
9. engine-location: front, rear.
10. wheel-base: continuous from 86.6 120.9.
11. length: continuous from 141.1 to 208.1.
12. width: continuous from 60.3 to 72.3.
13. height: continuous from 47.8 to 59.8.
14. curb-weight: continuous from 1488 to 4066.
15. engine-type: dohc, dohcv, l, ohc, ohcf, ohcv, rotor.
16. num-of-cylinders: eight, five, four, six, three, twelve, two.
17. engine-size: continuous from 61 to 326.
18. fuel-system: 1bbl, 2bbl, 4bbl, idi, mfi, mpfi, spdi, spfi.
19. bore: continuous from 2.54 to 3.94.
20. stroke: continuous from 2.07 to 4.17.
21. compression-ratio: continuous from 7 to 23.
22. horsepower: continuous from 48 to 288.
23. peak-rpm: continuous from 4150 to 6600.
24. city-mpg: continuous from 13 to 49.
25. highway-mpg: continuous from 16 to 54.
26. price: continuous from 5118 to 45400.

The original data also has some missing attribute values denoted by "?" :

Attribute #:	Number of instances missing a value:
2.	41
6.	2
19.	4
20.	4
22.	2
23.	2
26.	4

I've changed the original data in the following way :
- All instances with unknowns were removed giving 159 instances.
- The goal variable is "price"
- All nominal attributes (10) were removed.

Original source: UCI machine learning repository.
(http://www.ics.uci.edu/~mlearn/MLSummary.html).
Source: collection of regression datasets by Luis Torgo (ltorgo@ncc.up.pt) at
http://www.ncc.up.pt/~ltorgo/Regression/DataSets.html
Characteristics: 159 cases; 14 continuous variables; 1 nominal vars..