

K-Nearest Neighbor Algorithm

Code for loading dataset into 2D python list: [here](#)

Dataset preparation:

Randomly Split the dataset into Training (70%), Validation (15%) and Test (15%) set

```
Train_set=[], Val_set=[], Test_set=[]
```

```
//Following code shuffles your dataset list
```

1. for each sample S in the dataset:
2. generate a random number R in the range of [0,1]
3. if $R \geq 0$ and $R \leq 0.7$:
4. append S in Train_set
5. elif $R > 0.7$ and $R \leq 0.85$:
6. append S in Val_set
7. else:
8. append S in Test_set

KNN Classification:

Use Iris data [iris](#),

```
K = 5
```

1. **for** each sample V in the VALIDATION set:
2. **for** each sample T in the TRAINING set:
3. Find Euclidean distance between V_x (features $\rightarrow N-1$) and T_x (features $\rightarrow N-1$)
4. Store T and the distance in list L
5. Sort L in ascending order of distance
6. Take the first K samples
7. Take the majority class from the K samples (this is the detected class for sample V)
8. Now, check if this class is correct or not
9. Calculate $\text{validation_accuracy} = (\text{correct VALIDATION samples}) / (\text{total VALIDATION samples}) * 100$

- ☐ Calculate validation accuracy in a similar way for $K = 1, 3, 5, 10, 15$
- ☐ Make a table with 2 columns: K and Validation Accuracy ([report template](#))
- ☐ Now, take the K with **highest** Validation Accuracy
- ☐ Use this best K to determine **Test Accuracy** (Simply replace the VALIDATION set with TEST set)

KNN Regression:

Use diabetes data [diabetes](#)

$K = 5$, Error = 0

1. for each sample V in the VALIDATION set:
 2. for each sample T in the TRAINING set:
 3. Find Euclidean distance between V_x and T_x
 4. Store T_x and the distance in list L
 5. Sort L in ascending order
 6. Take the first K samples
 7. Take the average output of the K samples (this is the determined output for sample V)
 8. Error = Error + $(V \text{ true output} - V \text{ determined output})^2$
9. Calculate Mean_Squared_Error = Error / (total number of samples in VALIDATION set)

- ☐ Calculate Mean_Squared_Error in a similar way for $K = 1, 3, 5, 10, 15$
- ☐ Make a table with 2 columns: K and **Mean_Squared_Error** ([report template](#))
- ☐ Now, take the K with **minimum** Mean_Squared_Error
- ☐ Use this best K to determine **Mean_Squared_Error for the Test set** (Simply replace the VALIDATION set with TEST set)

Instruction

- Submit the .ipynb file and a report ([report template](#)) .pdf file.
- **DO NOT USE LIBRARIES SUCH AS: "Sklearn", "Scikit learning" or "pandas" for this assignment**
- **Copying will result in -100% penalty**

Marks Distribution

- (1) Dataset loading: 1.5
- (2) Train, Validation, Test split: 2.5
- (3) KNN classification algorithm + K tuning (table) + test accuracy : 5 + 1.5 + 1.5
- (4) KNN regression algorithm + K tuning (table) + test mean squared error : 5 + 1.5 + 1.5

Dataset description:

Diabetes

[source: [Diabetes dataset](#), [sklearn.datasets.load_diabetes — scikit-learn 1.1.1 documentation](#)]

Number of Instances: 442

Number of Attributes: First 10 columns are numeric predictive values

Target: Column 11 is a quantitative measure of **disease progression** one year after baseline

Attribute Information:

- age in years
- sex
- bmi body mass index
- bp average blood pressure
- s1 tc, total serum cholesterol
- s2 ldl, low-density lipoproteins
- s3 hdl, high-density lipoproteins
- s4 tch, total cholesterol / HDL
- s5 ltg, possibly log of serum triglycerides level
- s6 glu, blood sugar level

	A	B	C	D	E	F	G	H	I	J	K
1	0.03807590643	0.05068011874	0.06169620652	0.02187235499	-0.04422349842	-0.03482076284	-0.04340084565	-0.002592261998	0.01990842088	-0.01764612516	151
2	-0.001882016528	-0.04464163651	-0.05147406124	-0.02632783472	-0.008448724111	-0.01916333975	0.07441156408	-0.03949336287	-0.06832974362	-0.09220404963	75
3	0.0852989063	0.05068011874	0.04445121334	-0.005670610555	-0.04559945128	-0.03419446591	-0.03235593224	-0.002592261998	0.002863770519	-0.02593033699	141
4	-0.08906293935	-0.04464163651	-0.01159501451	-0.0366564468	0.01219056876	0.02499059336	-0.03603757004	0.03430885888	0.02269202257	-0.00936191133	206
5	0.005383060374	-0.04464163651	-0.0363646922	0.02187235499	0.003934851613	0.01559613951	0.008142083605	-0.002592261998	-0.03199144494	-0.04664087356	135
6	-0.0926954778	-0.04464163651	-0.0406959405	-0.01944209333	-0.06899064987	-0.07928784441	0.04127682384	-0.07639450375	-0.04118038519	-0.09634615654	97

Iris:

Source [[7.1. Toy datasets — scikit-learn 1.1.1 documentation](#)]

Number of Instances 150 (50 in each of three classes)

Number of Attributes 4 numeric, predictive attributes and the class

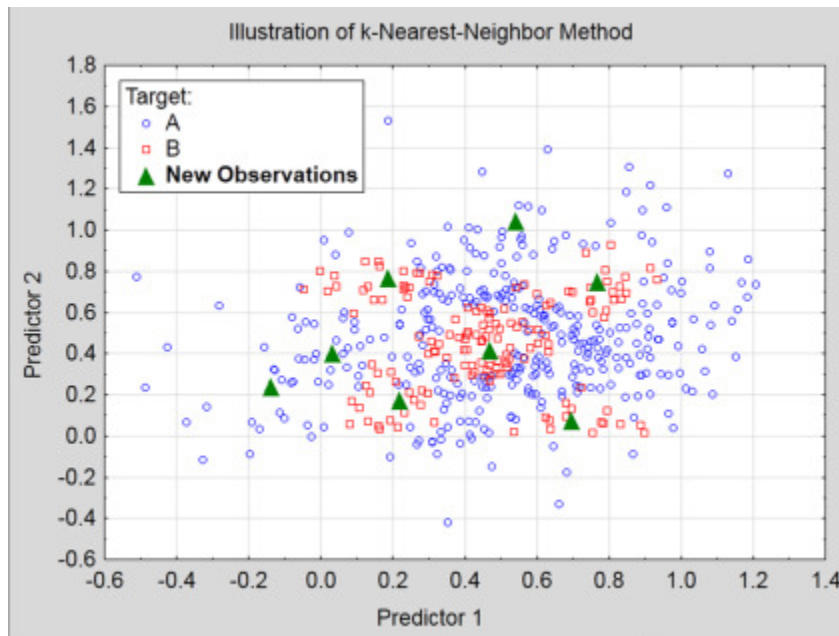
Attribute Information

- sepal length in cm
- sepal width in cm
- petal length in cm
- petal width in cm
- class:
 - Iris-Setosa
 - Iris-Versicolour
 - Iris-Virginica

	A	B	C	D	E
1	5.1	3.5	1.4	0.2	0
2	4.9	3	1.4	0.2	0
3	4.7	3.2	1.3	0.2	0
4	4.6	3.1	1.5	0.2	0
5	5	3.6	1.4	0.2	0
6	5.4	3.9	1.7	0.4	0

Resources

[7.1. Toy datasets — scikit-learn 1.0.2 documentation](#)



- Dataset (samples, features/attributes, label/classes)
 - [iris](#), [diabetes](#)
- Model high level concept from the perspective of supervised learning

- supervised learning, Classification, Regression
- dataset -> train, val, test

- KNN high level overview
- KNN pseudocode
- Instructions

- Classification: majority
- Regression: squared error

<https://www.quora.com/What-are-industry-applications-of-the-K-nearest-neighbor-algorithm>

<https://stackoverflow.com/questions/53704811/is-k-nearest-neighbors-algorithm-used-a-lot-in-real-life>

4. Using K-nearest neighbor algorithm, classify the following Test dataset, where $K=3$. Show details of calculations. [5]

Train dataset		
x	y	Label
3	4	No
4	5	Yes
5	-1	Yes
6	3	Yes
4	5	No

Test dataset		
x	y	Label
5	0	?
6	4	?

Yes

Yes.

For $(x, y) = (5, 0)$

$$\text{dis}[(5, 0), (3, 4)] = \sqrt{20}$$

$$\text{dis}[(5, 0), (4, 5)] = \sqrt{26}$$

$$\text{dis}[(5, 0), (5, -1)] = \sqrt{1}$$

$$\text{dis}[(5, 0), (6, 3)] = \sqrt{10}$$

$$\text{dis}[(5, 0), (4, 5)] = \sqrt{26}$$

So, 3 nearest neighbors of $(5, 0)$

$(5, -1)$, $(6, 3)$, $(3, 4)$

Yes Yes No

As 2 Yes and 1 No.

So label for $(5, 0)$ is Yes.

For $(x, y) = (6, 4)$

$$\text{dis}[(6, 4), (3, 4)] = \sqrt{9}$$

$$\text{dis}[(6, 4), (4, 5)] = \sqrt{5}$$

$$\text{dis}[(6, 4), (5, -1)] = \sqrt{26}$$

$$\text{dis}[(6, 4), (6, 3)] = \sqrt{1}$$

$$\text{dis}[(6, 4), (4, 5)] = \sqrt{5}$$

So 3 nearest neighbors of $(6, 4)$

$(6, 3)$, $(4, 5)$, $(4, 5)$

Yes

Yes

No

As 2 Yes and 1 No

So label for $(6, 4)$ is Yes