K-Nearest Neighbor Algorithm

Code for loading dataset into 2D python list: [here](https://colab.research.google.com/drive/143As_UCnCtvXuDlFRrfQEUthxc4cYpMx?usp=sharing)

# 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>=0 and R<=0.7:

4. append S in Train\_set

5. elif R>0.7 and R<=0.85:

6. append S in Val\_set

7. else:

8. append S in Test\_set

# KNN Classification:

**Use Iris data** [**iris**](https://drive.google.com/file/d/18RHDmJWdlrZp7XUqoHpZWxJ4Zd74FmuJ/view?usp=sharing),

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 Vx (features->N-1) and Tx (features->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 validation\_accuracy = (correct VALIDATION samples)/(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](https://docs.google.com/document/d/1tuEe0p9IouBkJ4N6Ux_JIRT9DVvP50L7RQiuM6SfgZo/edit?usp=sharing))
* 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**](https://drive.google.com/file/d/18WbWHM552nit6_tEnvqwuJmx53Kc1ArA/view?usp=sharing)

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 Vx and Tx

4. Store Tx 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 true output - V 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](https://docs.google.com/document/d/1tuEe0p9IouBkJ4N6Ux_JIRT9DVvP50L7RQiuM6SfgZo/edit?usp=sharing))
* Now, take the h **m**K wit**inimum** 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](https://docs.google.com/document/d/1tuEe0p9IouBkJ4N6Ux_JIRT9DVvP50L7RQiuM6SfgZo/edit?usp=sharing)) .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](https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html), [sklearn.datasets.load\_diabetes — scikit-learn 1.1.1 documentation](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_diabetes.html)]

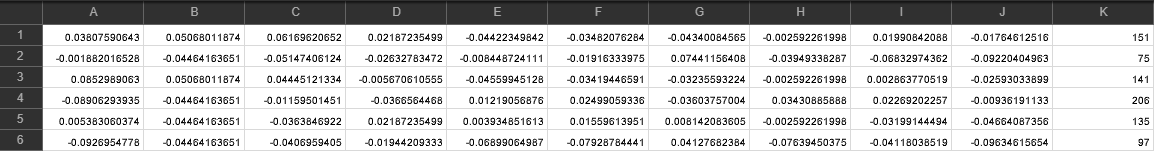
**Number of Instances:** 442

**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



## Iris:

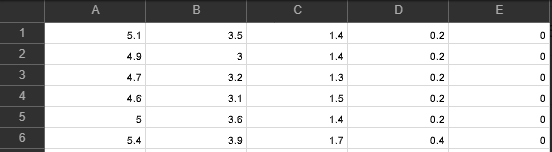
Source [[7.1. Toy datasets — scikit-learn 1.1.1 documentation](https://scikit-learn.org/stable/datasets/toy_dataset.html#iris-dataset) ]

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

**Number of Attributes** 4 numeric, predictive attributes and the class

**Attribute Information**

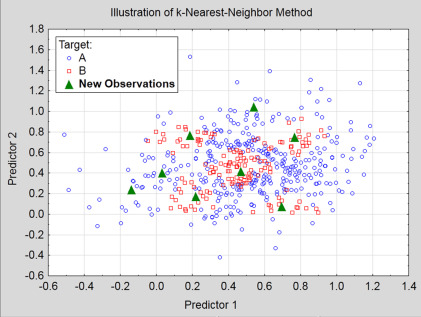
* sepal length in cm
* sepal width in cm
* petal length in cm
* petal width in cm
* class:
  + Iris-Setosa
  + Iris-Versicolour
  + Iris-Virginica

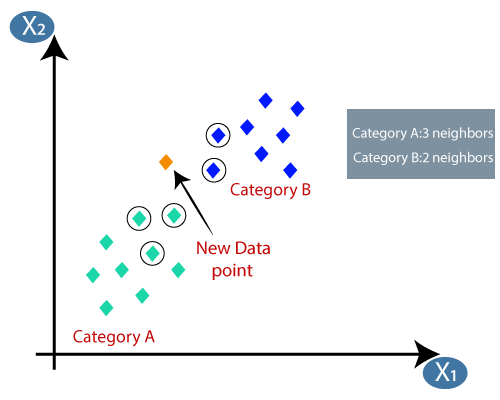


# 

# Resources

[7.1. Toy datasets — scikit-learn 1.0.2 documentation](https://scikit-learn.org/stable/datasets/toy_dataset.html)





* Dataset (samples, features/attributes, label/classes)
  + [iris](https://drive.google.com/file/d/18RHDmJWdlrZp7XUqoHpZWxJ4Zd74FmuJ/view?usp=sharing), [diabetes](https://drive.google.com/file/d/18WbWHM552nit6_tEnvqwuJmx53Kc1ArA/view?usp=sharing)
* 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>

