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=[]
//Shuffle 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,

```
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
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)
- [ ] Now, take the K with **highest** Validation Accuracy
- [ ] Use this best K to determine **Test Accuracy** (Simply replace the VALIDATION set of line 1. 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 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)
- [ ] 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 of line 1. 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	-0.03949338287	-0.06832974362	-0.09220404963	75
3	0.0852989063	0.05068011874	0.04445121334	-0.005670610555	-0.04559945128	-0.03419446591	-0.03235593224	-0.002592261998	0.002863770519	-0.02593033899	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	-0.0363846922	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