הטכניון
מכון טכנולוגי
לישראל

יישום KNN באמצעות קוד מלא

חיבור וחיסור מטריצות

```
m1
```

```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2]])
```

```
m2
```

```
array([[7. , 3.2, 4.7, 1.4],
       [6.4, 3.2, 4.5, 1.5],
       [6.9, 3.1, 4.9, 1.5]])
```

```
m1+m2
```

```
array([[12.1,  6.7,  6.1,  1.6],
       [11.3,  6.2,  5.9,  1.7],
       [11.6,  6.3,  6.2,  1.7]])
```

```
m1-m2
```

```
array([[-1.9,  0.3, -3.3, -1.2],
       [-1.5, -0.2, -3.1, -1.3],
       [-2.2,  0.1, -3.6, -1.3]])
```

Broadcasting 2D

Numpy יכול לבצע פעולות בין אלמנטים בממדים שונים, על ידי שכפול.

לדוגמא: חיבור של מטריצה + מטריצת שורה

או חיבור של מטריצה + מטריצת עמודה

```
a = np.array([[0,0,0],
              [10,10,10],
              [20,20,20],
              [30,30,30]])
b=np.array([[1,2,3]])
a+b
```

```
array([[ 1,  2,  3],
       [11, 12, 13],
       [21, 22, 23],
       [31, 32, 33]])
```

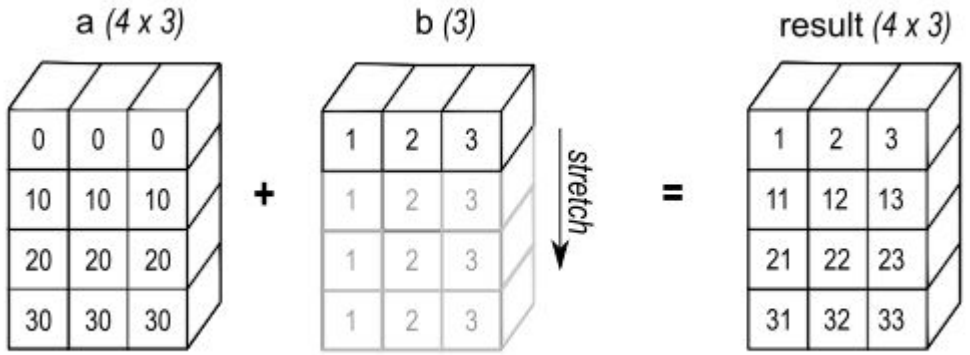

Figure 2

Creating numpy arrays

```
df
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.8	4.0	1.2	0.2	Iris-setosa
1	5.1	2.5	3.0	1.1	Iris-versicolor
2	6.6	3.0	4.4	1.4	Iris-versicolor
3	5.4	3.9	1.3	0.4	Iris-setosa
4	7.9	3.8	6.4	2.0	Iris-virginica
...	...	...	...	...	...
145	6.3	2.8	5.1	1.5	Iris-virginica
146	6.4	3.1	5.5	1.8	Iris-virginica
147	6.3	2.5	4.9	1.5	Iris-versicolor
148	6.7	3.1	5.6	2.4	Iris-virginica
149	4.9	3.1	1.5	0.1	Iris-setosa

150 rows × 5 columns

```
sample = np.array([[0,1,2,3]])
sample
```

```
array([[0, 1, 2, 3]])
```

```
dfnp = df.drop('Species',axis=1).to_numpy()
dfnp[:5]
```

```
array([[5.8, 4. , 1.2, 0.2],
       [5.1, 2.5, 3. , 1.1],
       [6.6, 3. , 4.4, 1.4],
       [5.4, 3.9, 1.3, 0.4],
       [7.9, 3.8, 6.4, 2. ]])
```

Calculate distance method 1

```
[157] dfnp2 = dfnp - sample
      dfnp2[:5]
```

```
array([[ 5.8,  3. , -0.8, -2.8],
       [ 5.1,  1.5,  1. , -1.9],
       [ 6.6,  2. ,  2.4, -1.6],
       [ 5.4,  2.9, -0.7, -2.6],
       [ 7.9,  2.8,  4.4, -1. ]])
```

```
[158] dfnp2 = dfnp2**2
      dfnp2[:5]
```

```
array([[33.64,  9.  ,  0.64,  7.84],
       [26.01,  2.25,  1.  ,  3.61],
       [43.56,  4.  ,  5.76,  2.56],
       [29.16,  8.41,  0.49,  6.76],
       [62.41,  7.84, 19.36,  1.  ]])
```

```
[159] dfnp2 = np.sum(dfnp2,axis=1)
      dfnp2[:5]
```

```
array([51.12, 32.87, 55.88, 44.82, 90.61])
```

```
[160] dis1 = np.sqrt(dfnp2)
      dis1[:5]
```

```
array([7.14982517, 5.73323643, 7.47529264, 6.69477408, 9.51892851])
```

```
sample = np.array([[0,1,2,3]])
sample
```

```
array([[0, 1, 2, 3]])
```

```
dfnp = df.drop('Species',axis=1).to_numpy()
dfnp[:5]
```

```
array([[5.8, 4. , 1.2, 0.2],
       [5.1, 2.5, 3. , 1.1],
       [6.6, 3. , 4.4, 1.4],
       [5.4, 3.9, 1.3, 0.4],
       [7.9, 3.8, 6.4, 2. ]])
```

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2}.$$

Calculate distance method 2

```
dis2 = np.linalg.norm(dfnp - sample,axis=1)
dis2[:5]
```

```
array([7.14982517, 5.73323643, 7.47529264, 6.69477408, 9.51892851])
```

```
sample = np.array([[0,1,2,3]])
sample
```

```
array([[0, 1, 2, 3]])
```

```
dfnp = df.drop('Species',axis=1).to_numpy()
dfnp[:5]
```

```
array([[5.8, 4. , 1.2, 0.2],
       [5.1, 2.5, 3. , 1.1],
       [6.6, 3. , 4.4, 1.4],
       [5.4, 3.9, 1.3, 0.4],
       [7.9, 3.8, 6.4, 2. ]])
```

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2}.$$

Create distance column and sort

```
df2 = df.copy()
df2['distance'] = pd.Series(dis2)
df2.sort_values(by='distance',ascending=True,inplace=True)
df2
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species	distance
102	4.5	2.3	1.3	0.3	Iris-setosa	5.451605
124	4.4	2.9	1.4	0.2	Iris-setosa	5.583010
129	4.9	2.4	3.3	1.0	Iris-versicolor	5.626722
84	4.4	3.0	1.3	0.2	Iris-setosa	5.629387
122	4.3	3.0	1.1	0.1	Iris-setosa	5.631163
...	...	...	...	...	...	...
94	7.6	3.0	6.6	2.1	Iris-virginica	9.150410
58	7.7	2.8	6.7	2.0	Iris-virginica	9.253108
45	7.7	2.6	6.9	2.3	Iris-virginica	9.292470
52	7.7	3.8	6.7	2.2	Iris-virginica	9.479451
4	7.9	3.8	6.4	2.0	Iris-virginica	9.518929

150 rows × 6 columns

Frequent label in K nearest neighbours

```
k=5
top_k = df2['Species'].iloc[:k]
top_k
```

```
102        Iris-setosa
124        Iris-setosa
129    Iris-versicolor
84         Iris-setosa
122        Iris-setosa
Name: Species, dtype: object
```

```
top_k.value_counts()
```

```
Iris-setosa        4
Iris-versicolor    1
Name: Species, dtype: int64
```

```
top_k.value_counts().max()
```

```
4
```

```
top_k.value_counts().idxmax()
```

```
'Iris-setosa'
```

Predicting one samples & many samples (example usage)

```
def predict_one(vec):
  return np.sum(vec)
predict_one(sample)
```

```
6
```

```
def predict_all(mat):
  preds = [predict_one(vec) for vec in mat] #list comprehension
  return np.array(preds,dtype=int)
predict_all(samples)
```

```
array([ 4, 16, 11])
```

Predicting one samples & many samples

```
def KNN_predict_sample(df, sample, k):
  pass
```

```
def KNN_predict(df,samples,k):
   pass
```

KNN exercise 2

KNN iris exercise 2

הטכניון
מכון טכנולוגי
לישראל

תודה על ההשתתפות