PROJE

Haftalık Rapor – 10.04.2021

10 NİSAN 2021

KIRIKKALE ÜNİVERSİTESİ – BİLGİSAYAR MÜHENDİSLİĞİ İÖ

AHMET MUNGAN – 160255081

İÇİNDEKİLER

ÖZET

Veri bilimine yönelik Python kütüphanelerinden numpy ve pandas araştırılıp öğrenilmiştir. Bu kapsamda kullanımı ve özellikleri irdelenmiştir.

PYTHON'DA NUMPY UYGULAMALARI

Python'da bilinen listelerin bazı dönüşümler ile dizilere çevrilmesini numpy ile sağlayabiliriz. Burada avantaj aslında daha büyük verilerde ortaya çıkar.

JPY 1

```
>>> import numpy as np
>>> matrixList = [[10,20,30],[20,30,40],[30,40,50]]
>>> matrixList
[[10, 20, 30], [20, 30, 40], [30, 40, 50]]
>>> np.array(matrixList)
array([[10, 20, 30],
       [20, 30, 40],
       [30, 40, 50]])
```

JPY 1'de görüldüğü üzere numpy çok boyutlu dizilerde görsel bir çıktı sağlıyor. Bu da büyük verilerde aslında bir avantaj oluşturuyor. Bunun yanında dizi yapısı kullanıldığı için tüm veri yapıları kapsamında kullanılan teknik ve görece teknolojiler bu çerçevede uygulanabilir bir form kazanmıştır.

JPY 2

```
>>> np.ones((3,3))
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])
>>> np.zeros((3,3))
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

JPY 2'de tamamını 0 veya 1 olarak matrix oluşturma methodları mevcuttur. Bu matrix oluşturma method'ları, numpy dizilerini mantıksal süreçlere tabi tutmak için iyi bir araçtır. Örneğin bir sinyalin işlenmesi için tamamı 0'dan oluşan bir numpy dizisi ile OR'lanması veya tamamı 1'den oluşan bir numpy dizisi ile AND'lenmesi işe yarar sonuçlar doğurabilir. Bu sayede sinyal işlemede numpy hazır methodları yardımcı olacaktır.

JPY 3

```
>>> import numpy as np
>>> myArray = np.arange(0,10)
>>> myArray
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> slicingArray = myArray[3:6]
>>> slicingArray
array([3, 4, 5])
>>> slicingArray[:] = -1
>>> slicingArray
array([-1, -1, -1])
>>> myArray
array([ 0,  1,  2, -1, -1, -1,  6,  7,  8,  9])
```

Numpy ile alakalı tüm operasyonlara göz gezdirilmiştir ve gerekli olabilecek dökümanlar elde edilmiştir. Bu çerçevede numpy dizilerinin dinamik olduğu öğrenilmiştir. JPY 3'te görüldüğü üzere numpy dizileri class ile tanımlandığı için numpy dizilerinden alınan slicing'lerde yapılan değişikler, slicing yapılan diziyi etkileyecektir. Veya dizi kopyalarındaki değişiklikler ana dizinin değişimine neden olacaktır. Ayrıca numpy'nin dizi içindeki tüm elemanları -1 yapmak için JPY 3'te görüldüğü gibi basit kodlamaları mevcuttur.

PYTHON'DA PANDAS UYGULAMALARI

Python'ın pandas uygulamaları ile yapılan işlemler ile veriler sanki Microsoft Office Excel'de tutuluyormuş gibi bir düzene dönüşür. Bu hem programcının hem de kullanıcıların isteği üzerine veri dönüşümlerinde büyük avantajdır.

JPY 4

```
>>> import numpy as np
>>> import pandas as pd
>>> numpyArray = np.array([50,40,30])
>>> numpyArray
array([50, 40, 30])
>>> pd.Series(numpyArray)
0    50
1    40
2    30
dtype: int32
>>> numbers = [1,2,3]
>>> pd.Series(data=numbers, index=numpyArray)
50    1
40    2
30    3
dtype: int64
```

JPY 4'te görüldüğü üzere numpy dizilerini pandas serilerine dönüştürmek mümkündür. Index'leme işlemlerini numpy dizilerinden dönüştürürken identity bir sıralamalada otomatik bir şekilde vermektedir. Fakat serilerin data'sını ayrı index'ini ayrı bir şekilde JPY 4'te görüldüğü gibi farklı listelerden de çekebilmek ve bunları birleştirmek mümkündür. Python'ın normal sözlük yapısına benzemektedir fakat index ve data kısımlarını dinamik hale getirmek veri biliminde bir avantajdır.

JPY 5

```
>>> import pandas as pd
>>> import numpy as np
>>> data = np.random.randn(4,3)
>>> data
array([[-0.75189695,  1.09533415, -1.76472081],
       [ 0.75809641,  0.64601308,  0.60696653],
       [-0.11602351,  1.83853106, -0.02466869],
       [ 0.92969513,  0.60096824,  0.05256735]])
```

JPY 5'te numpy ile öğrenilen normal dağılıma uygun rastgele dizi oluşturulmuştur.

JPY 6

```
>>> dataFrame = pd.DataFrame(data)
>>> dataFrame
```

	0	1	2
0	-0.751897	1.095334	-1.764721
1	0.758096	0.646013	0.606967
2	-0.116024	1.838531	-0.024669
3	0.929695	0.600968	0.052567

```
>>> newDataFrame = pd.DataFrame(data, index = ["A", "B", "C", "D"]
, columns = ["X", "Y", "Z"])
>>> newDataFrame
```

	X	Y	Z
A	-0.751897	1.095334	-1.764721
B	0.758096	0.646013	0.606967
C	-0.116024	1.838531	-0.024669
D	0.929695	0.600968	0.052567

JPY 5'te oluşturulan bu dizi, daha sonra ise JPY 6'da pandas'ın hazır methodlarından DataFrame için kullanılabilir. Bu sayede var olan dizinin gerçek bir Excel görünümüne kavuşması sağlanabilir. JPY 6'da ayrıca "newDataFrame" değişkeni ile görüldüğü üzere index ve column isimlerini de manuel bir şekilde vermek mümkündür. Bu da dinamizmi arttırdığı ve veri biliminde işe yarayabileceğini göstermektedir.

ANAHTAR KELİMELER

- ***JPY X*** : X. Jupyter Notebook çıktısı.

REFERANS VE KAYNAKÇA

- Python – Link için tıklayınız.
- Python 3 Documents – Link için tıklayınız.
- Anaconda – Link için tıklayınız.
- Anaconda Individual Edition – Link için tıklayınız.
- Anaconda Resources – Link için tıklayınız.
- Numpy Documentation – Link için tıklayınız.
- Pandas Documentation – Link için tıklayınız.
- Matplotlib Documentation – Link için tıklayınız.