ANALIZA VELIKIH PODATAKA

školska 2025/2026 godina

Vežba 2: Rad sa numeričkim podacima pomoću NumPy biblioteke

NumPy je biblioteka u Pythonu koja omogućava rad sa nizovima i matricama velikih dimenzija, kao i obavljanje matematičkih operacija sa njima. Nizovi (eng.arrays) u NumPy-u predstavljaju efikasniji način za čuvanje i manipulaciju podacima u odnosu na obične Python liste.

🔷 Kreiranje jednostavnog NumPy niza

NumPy nizovi se kreiraju pomoću funkcije numpy.array(), koja omogućava da se lista ili tuple konvertuju u niz.

- NumPy nizovi (arrays) mogu biti više dimenzionalni.
- Za razliku od običnih Python lista, NumPy nizovi imaju fiksnu veličinu i tip podataka, što omogućava efikasnije performanse za brojne matematičke operacije.

Primer koda:

```
import numpy as np

# Kreiramo jednostavan NumPy niz iz liste
python_lista = [1, 2, 3, 4, 5]
numpy_niz = np.array(python_lista)

print(numpy_niz)
```

Šta se dešava?

- U ovom primeru, kreiramo običnu Python listu python_lista i konvertujemo je u NumPy niz pomoću funkcije np.array().
- Rezultat je NumPy niz koji sadrži iste vrednosti kao lista, ali sa prednostima bržih operacija.

🌟 2. Matematičke operacije sa NumPy nizovima

Jedna od glavnih prednosti NumPy-a je mogućnost efikasnog izvođenja matematičkih operacija na celim nizovima, bez potrebe za petljama.

🔷 Sabiranje i oduzimanje

NumPy omogućava vektorizovane operacije, što znači da možete obavljati operacije između nizova element-po-element koristeći jednostavne operatora kao što su + i -.

- Sabiranje dva NumPy niza ili skalarne vrednosti se vrši na elementima sa odgovarajućim indeksima.
- Takođe, može se vršiti operacija sa skalarnim vrednostima (jedan broj koji se dodaje ili oduzima od svakog elementa niza).

Primer koda:

```
# Sabiranje dva NumPy niza
niz1 = np.array([1, 2, 3])
niz2 = np.array([4, 5, 6])

# Sabiranje nizova element po element
rezultat_sabiranja = niz1 + niz2
print("Rezultat sabiranja:", rezultat_sabiranja)

# Sabiranje sa skalarom
rezultat_sabiranja_skalar = niz1 + 10
print("Rezultat sabiranja sa skalarom:", rezultat_sabiranja_skalar)
```

Šta se dešava?

- Prvi primer sabira dva NumPy niza, element po element.
- Drugi primer dodaje broj 10 svakom elementu u nizu niz1, bez potrebe za petljama.

🌟 3. Indeksiranje i Slicing (Sečenje) NumPy nizova

Indeksiranje i sečenje su ključne operacije za ekstrakciju specifičnih delova niza. NumPy omogućava veoma efikasno indeksiranje i selektovanje podnizova.

🔷 Indeksiranje

Indeksiranje se koristi za pristup pojedinačnim elementima niza.

- NumPy nizovi podržavaju pozitivno i negativno indeksiranje, što znači da možemo pristupiti elementima sa početka ili kraja niza.
- Indeksiranje počinje od 0.

Primer koda:

```
# Pristup prvom elementu
prvi_element = numpy_niz[0]
print("Prvi element niza:", prvi_element)

# Pristup poslednjem elementu koristeći negativan indeks
poslednji_element = numpy_niz[-1]
print("Poslednji element niza:", poslednji_element)
```

Šta se dešava?

- Prvi element niza se dobija sa indeksom 0.
- Poslednji element niza se može dobiti sa negativnim indeksom -1.

🔷 Slicing (Sečenje)

Sečenje omogućava da izdvojimo podnizove na osnovu određenih granica. Sintaksa za sečenje je niz[start:end], gde start predstavlja početni indeks, a end završni indeks (koji nije uključen).

- Sečenje je vrlo efikasno i koristi se za izvlačenje delova podataka iz niza.
- Možemo koristiti korak (step), što omogućava selektovanje elemenata sa određenim intervalom (npr. svaki drugi element).

Primer koda:

```
# Sečenje niza od indeksa 1 do 3 (ne uključujući 3)
podniz = numpy_niz[1:3]
print("Podniz:", podniz)

# Sečenje sa korakom 2
podniz_korak = numpy_niz[::2]
print("Podniz sa korakom 2:", podniz_korak)
```

Šta se dešava?

- U prvom primeru, uzimamo elemente od indeksa 1 do 2.
- U drugom primeru, uzimamo svaki drugi element iz niza, počinjući od početka.

🌟 4. Reshaping (Promena oblika) NumPy nizova

Reshaping se koristi za promenu dimenzija niza, tj. da bi niz sa jednim oblikom (npr. 1D) postao niz sa drugim oblikom (npr. 2D).

🔷 Kako se koristi reshape?

Funkcija reshape() omogućava promenu oblika niza. Prilikom korišćenja ove funkcije, broj elemenata mora ostati isti, ali se njihov raspored može promeniti. Reshaping se generalno koristi kada želite da preoblikujete niz, na primer, da biste ga koristili kao matricu za operacije sa više dimenzija.

- Reshaping se koristi kada želite da preoblikujete niz, na primer, da biste ga koristili kao matricu za operacije sa više dimenzija.

Primer koda:

```
# Kreiramo NumPy niz od 12 elemenata
niz = np.arange(12)
print("Originalni niz:", niz)

# Preoblikujemo niz u matricu 3x4
matrica = niz.reshape(3, 4)
print("Preoblikovana matrica 3x4:\n", matrica)

# Preoblikujemo u 2x6 matricu
matrica2 = niz.reshape(2, 6)
print("Preoblikovana matrica 2x6:\n", matrica2)
```

Šta se dešava?

- Kreiramo niz sa 12 elemenata.
- Zatim preoblikujemo taj niz u matricu sa 3 reda i 4 kolone.
- Takođe, pokazujemo da možemo preoblikovati niz u različite oblike, sve dok broj elemenata ostaje isti.

Ova fleksibilnost je korisna kada radimo sa podacima koji dolaze u različitim formatima, jer nam omogućava da ih pripremimo za analizu ili unos u modele mašinskog učenja.

🌟 5. Razlika između Python lista i NumPy nizova

Python liste i NumPy nizovi se često koriste za rad sa podacima, ali se razlikuju po nekoliko ključnih karakteristika.

🔷 **Glavne razlike:**

1. **Brzina i performanse:**
    - NumPy nizovi su mnogo brži od Python lista za matematičke operacije jer koriste optimizovane C biblioteke.
2. **Podržani tipovi podataka:**
    - Python liste mogu sadržavati elemente različitih tipova, dok NumPy nizovi obavezno moraju imati isti tip podataka.
3. **Dimenzionalnost:**
    - Python liste su 1D, dok NumPy nizovi mogu biti višedimenzionalni.

- NumPy nizovi su efikasniji kada radimo sa velikim datasetovima i obavljamo matematičke operacije.
- NumPy takođe nudi mnogo naprednih funkcija za analizu podataka (npr. agregacije, statističke funkcije).

Primer koda:

```
# Python lista
lista = [1, 2, 3, 4, 5]
# NumPy niz
numpy_lista = np.array(lista)

# Proveravamo da li su oba objekta iste vrste
print("Lista je tipa:", type(lista))
print("NumPy niz je tipa:", type(numpy_lista))
```

Šta se dešava?

- Ispisuje se tip objekta za Python listu i NumPy niz, pokazujući razliku u načinu na koji Python i NumPy tretiraju ove strukture podataka.

📝 **Zadatak za samostalni rad**

1. **Kreirajte NumPy nizove** sa različitim vrednostima i obavite sledeće operacije:
    - Sabiranje dva niza.
    - Proširite jedan niz tako da dodate skalarni broj svakom elementu.
2. **Rad sa dimenzijama:**
    - Kreirajte 1D niz i preoblikujte ga u 2D matricu.
3. **Indeksiranje i Slicing:**
    - Kreirajte 1D niz i izaberite njegove delove koristeći indeksiranje i sečenje.
    - Probajte korak u sečenju i objasnite kako funkcioniše.