შეხვედრა 9: ზედამხედველობითი სწავლება: განზრახვის კლასიფიკაცია

წინა შეხვედრაზე ჩვენ გავეცანით მანქანური სწავლების დიდ სამყაროს. დღეს კი ჩავუღრმავდებით მის ერთ-ერთ უმნიშვნელოვანეს ნაწილს: **კლასიფიკაციას**. კლასიფიკაცია არის მანქანურ სწავლებაში ერთ-ერთი ყველაზე მნიშვნელოვანი პროცესი. ამ პროცესის ფარგლებში პროგრამა ცდილობს განსაზღვროს, რომელ კატეგორიას (კლასს) მიეკუთვნება მიღებული მონაცემები.

კლასიფიკაციის **დახმარებით**, ჩვენი ბოტი შეძლებს ამოიცნოს მომხმარებლის განზრახვა, მიუხედავად იმისა, თუ რა სიტყვებს გამოიყენებს მომხმარებელი. ეს არის მნიშვნელოვანი ნაბიჯი, რათა თავად შევქმნათ ხელოვნურ ინტელექტზე დაფუძნებული პროგრამა. ამ შეხვედრის ბოლოს შენ შეძლებ ააწყო მონაცემთა ბაზა, რომელიც მომხმარებლის შეტყობინებებს და მათ განზრახვებს დააჯგუფებს. ამგვარად, შენი ბოტი კიდევ ერთი ნაბიჯით მიუახლოვდება იმას, რომ გახდეს მართლაც „ჭკვიანი“.

1. რა არის კლასიფიკაცია?

კლასიფიკაცია არის მანქანური სწავლების ფუნდამენტური პროცესი, რომლის დროსაც პროგრამა იღებს მონაცემებს და ცდილობს განსაზღვროს, რომელ წინასწარ განსაზღვრულ ჯგუფს ან კატეგორიას (კლასს) მიეკუთვნება ეს მონაცემები.

1.1. პრობლემის ფორმულირება: ობიექტის კატეგორიის (კლასის) განსაზღვრა

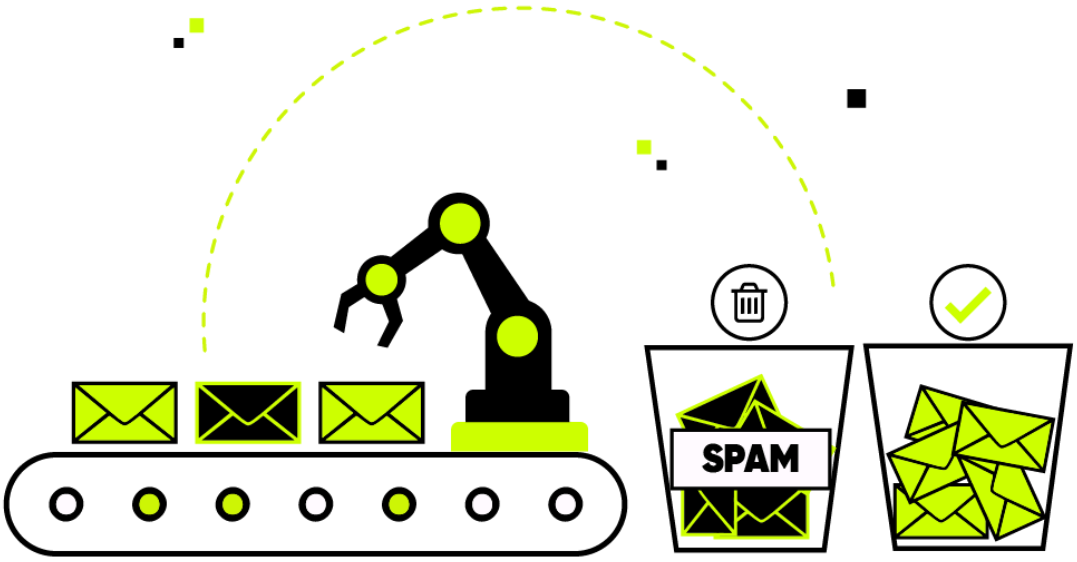
კლასიფიკაცია ჰგავს დახარისხების თამაშს. წარმოიდგინე, რომ გაქვს ბევრი სხვადასხვა ფორმის, ფერისა და ზომის სათამაშო. შენი ამოცანა **მათი დაჯგუფებაა**. კლასიფიკაციის ალგორითმიც ზუსტად ამას აკეთებს: ის ათვალიერებს ყველა მონაცემს და მიანიჭებს მას შესაბამის კატეგორიას (კლასს).

1.2. ელექტრონული ფოსტის დახარისხება და გამოსახულების ამოცნობა

კლასიფიკაცია ყოველდღიურ ცხოვრებაში ყველგან გვხვდება.

1.2.1. ელექტრონული ფოსტის დახარისხება: ორკლასიანი კლასიფიკაცია

ელექტრონული ფოსტის სპამის ფილტრი კლასიფიკაციის ერთ-ერთი ყველაზე მარტივი და გავრცელებული მაგალითია. როდესაც თქვენს ელექტრონულ ფოსტაზე ახალი წერილი შემოდის, პროგრამა ავტომატურად აანალიზებს მის შინაარსს (სიტყვებს, ფრაზებს, გამგზავნს) და ორი შესაძლო კატეგორიიდან ერთ-ერთს მიანიჭებს: „სპამი“ ან „არა-სპამი“. ამ შემთხვევაში, ჩვენ საქმე გვაქვს ორკლასიან კლასიფიკაციასთან, რადგან შესაძლო პასუხების რაოდენობა მხოლოდ ორია.



1.2.2. სურათის ამოცნობა: მრავალკლასიანი კლასიფიკაცია

წარმოიდგინე, რომ პროგრამას აჩვენებ სურათს, რომელზეც კატაა გამოსახული. პროგრამა აანალიზებს სურათზე არსებულ პიქსელებს და გამოთქვამს მოსაზრებას, რომ ეს არის „კატა“. თუ სხვა სურათს აჩვენებ, რომელზეც ძაღლია, ის ფოტოსურათს „ძაღლის“ კლასს მიანიჭებს. ამ შემთხვევაში, შესაძლო პასუხების რაოდენობა ორზე მეტია (მაგალითად, „კატა“, „ძაღლი“, „ჩიტი“, „ცხენი“ და ა.შ.), ამიტომ ამას **მრავალკლასიანი კლასიფიკაცია** ჰქვია.

1.3. მომხმარებლის შეტყობინების კლასიფიკაცია განზრახვების მიხედვით (greeting, question, goodbye)

კლასიფიკაცია ჩვენს ჩატბოტს დაეხმარება მომხმარებლის განზრახვის ამოცნობაში. თუ მომხმარებელი დაწერს „გამარჯობა“, ჩვენი მოდელი მას დააკლასიფიცირებს როგორც greeting (მისალმება). თუ მომხმარებელი ჩატბოტს დაუწერს „როგორ ხარ“, ჩვენი მოდელი მას დააკლასიფიცირებს როგორც question (შეკითხვა).

2. მონაცემების მომზადება კლასიფიკაციისთვის

იმისთვის, რომ მანქანური სწავლების მოდელმა მონაცემები გაიგოს, ისინი სპეციალურად უნდა მოვამზადოთ. კომპიუტერებს მხოლოდ რიცხვები „ესმით“, ამიტომ ტექსტი რიცხვებად უნდა გადავაქციოთ. ეს არის ერთ-ერთი ყველაზე მნიშვნელოვანი ეტაპი, რადგან მონაცემების სწორად მომზადების გარეშე, ვერცერთი ალგორითმი ვერ იმუშავებს ეფექტურად. ამ პროცესს **წინასწარი დამუშავება (Preprocessing)** ჰქვია.

2.1. ნიშან-თვისებების (Features) და სამიზნე ცვლადის (Label/Target) ცნება

იმისთვის, რომ ჩვენი მოდელი გავწვრთნათ , ორი ტიპის მონაცემები დაგვჭირდება:

- **ნიშან-თვისებები (Features):** ეს არის მონაცემების ის მახასიათებლები, რომლებსაც მოდელი იყენებს სწავლისთვის. ტექსტის შემთხვევაში ეს შეიძლება იყოს კონკრეტული სიტყვები, მათი თანმიმდევრობა, ან თითოეული სიტყვის რაოდენობა წინადადებაში.
- **სამიზნე ცვლადი (Label/Target):** ეს არის სწორი პასუხი, რომელიც გვინდა, რომ მოდელმა იწინასწარმეტყველოს. ეს არის „მასწავლებლის“ პასუხი. ჩვენს შემთხვევაში, ეს იქნება მომხმარებლის განზრახვა (greeting, goodbye).

2.2. როგორ ვაქციოთ ტექსტი რიცხვებად (Bag-of-Words მოდელი)

კომპიუტერებს არ შეუძლიათ სიტყვების პირდაპირ გაანალიზება. ამიტომ, ჩვენ უნდა გამოვიყენოთ ისეთი მეთოდები, რომლებიც ტექსტს ციფრულ ფორმატში გადაიყვანს. ერთ-ერთი ყველაზე მარტივი მეთოდი ტექსტის რიცხვებად გადაქცევისთვის არის მოდელი, რომელსაც **სიტყვების ტომარას (Bag-of-Words)** უწოდებენ.

2.1. როგორ დავითვალოთ სიტყვები

წარმოიდგინე, რომ შენს ქართულის მასწავლებელს სურს გაარკვიოს, რომელი სიტყვები და რამდენჯერ იყო გამოყენებული ორი სხვადასხვა მოსწავლის ნაშრომში. ის წაიკითხავს თითოეულ ნაშრომს, ხელით ჩაინიშნავს ყოველ სიტყვას და დაითვლის მათ რაოდენობას. ამ სამუშაოს ბევრი დრო დასჭირდება, მაგრამ რა მოხდება, თუ ამ რთულ და მონოტონურ სამუშაოს კომპიუტერს დავავალებთ? კომპიუტერი შეძლებს ავტომატურად გააანალიზოს ტექსტები, მოძებნოს ყველა სიტყვა და დაითვალოს მათი რაოდენობა წამების განმავლობაში.

მაგალითად, ავიღოთ ეს ორი წინადადება: "მე მიყვარს ვაშლი" და "მე მიყვარს მსხალი".

1. პირველ რიგში, კომპიუტერი მოაგროვებს ყველა უნიკალურ (განსხვავებულ) სიტყვას ორივე წინადადებიდან და შექმნის სიტყვების სიას: ['მე', 'მიყვარს', 'ვაშლი', 'მსხალი].
2. შემდეგ კომპიუტერი დაითვლის, რამდენჯერ შეგვხვდა თითოეული სიტყვა ამ წინადადებებში:

- **წინადადება 1:** "მე მიყვარს ვაშლი"
    - "მე" -> გვხვდება 1-ჯერ
    - "მიყვარს" -> გვხვდება 1-ჯერ
    - "ვაშლი" -> გვხვდება 1-ჯერ
    - "მსხალი" -> არ გვხვდება (0-ჯერ)
    - **შედეგი:** [1, 1, 1, 0]
- **წინადადება 2:** "მე მიყვარს მსხალი"
    - "მე" -> გვხვდება 1-ჯერ
    - "მიყვარს" -> გვხვდება 1-ჯერ

- "ვაშლი" -> არ გვხვდება (0-ჯერ)
- "მსხალი" -> გვხვდება 1-ჯერ
- **შედეგი:** [1, 1, 0, 1]

ამრიგად, ორივე წინადადება გადავაქციეთ რიცხვების სიებად.

3. პირველი მოდელის იდეა: გადაწყვეტილების ხე

გადაწყვეტილების ხე (Decision Tree) არის მანქანური სწავლების ერთ-ერთი ყველაზე მარტივი და გასაგები ალგორითმი. ის საშუალებას აძლევს პროგრამას, მიიღოს გადაწყვეტილება მონაცემების ანალიზის საფუძველზე. წარმოიდგინე, რომ გსურს, განსაზღვრო, იყიდო თუ არა ახალი ტელეფონი. შენ, ალბათ, ასეთ კითხვებს დაუსვამდი საკუთარ თავს: „მაქვს საკმარისი ფული?“; თუ „კი“, მაშინ „მჭირდება ახალი ტელეფონი?“. ყოველი პასუხი მიგიყვანს საბოლოო გადაწყვეტილებამდე. გადაწყვეტილების ხეც ზუსტად ასე მუშაობს: ის აანალიზებს მონაცემებს და სვამს ლოგიკურ კითხვებს, რათა მივიდეს სწორ კლასამდე. ეს არის მკაფიო, ეტაპობრივი პროცესი, რომლის გაგებაც ადამიანისთვისაც მარტივია.

3.1. როგორ მუშაობს ალგორითმი, რომელიც მონაცემების გასაყოფად ლოგიკურ "კი/არა" კითხვებს სვამს

გადაწყვეტილების ხე (Decision Tree) არის მანქანური სწავლების ერთ-ერთი ყველაზე მარტივი და გასაგები ალგორითმი. ის მუშაობს ისე, როგორც ჩვენი გონება იღებს გადაწყვეტილებებს: სვამს „კი/არა“ ტიპის კითხვებს მანამ, სანამ საბოლოო პასუხს არ მიიღებს.

3.2. ანალოგია: თამაში "გამოიცანი ვინ ვარ?"

წარმოიდგინე, რომ თამაშობ თამაშს „გამოიცანი ვინ ვარ?“. თქვენი ამოცანაა ლოგიკური “კი/არა” კითხვების დასმით გამოიცნოთ ერთი-ერთი პერსონაჟი.

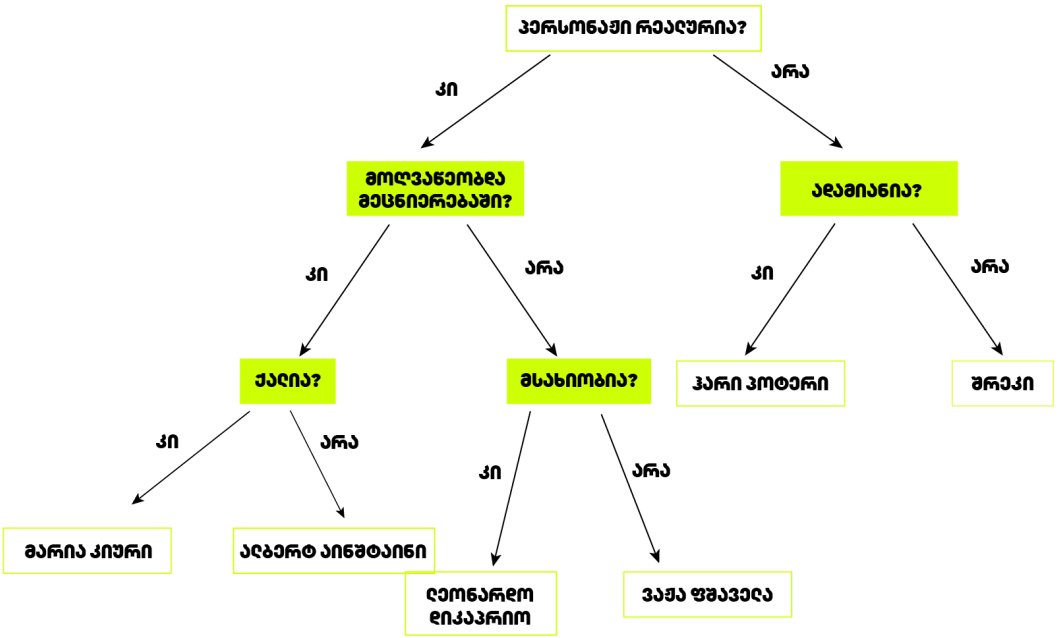
შესაძლო პერსონაჟები:

1. **ალბერტ აინშტაინი** (რეალური, მამაკაცი, მეცნიერი, გარდაცვლილი)
2. **ლეონარდო დიკაპრიო** (რეალური, მამაკაცი, მსახიობი, ცოცხალი)
3. **მარია კიური** (რეალური, ქალი, მეცნიერი, გარდაცვლილი)
4. **ვაჟა-ფშაველა** (რეალური, მამაკაცი, მწერალი, გარდაცვლილი)
5. **შრეკი** (გამოგონილი, მამრობითი სქესის, ანიმაციური პერსონაჟი)
6. **ჰარი პოტერი** (გამოგონილი, მამრობითი სქესის, ადამიანი/ჯადოქარი)

- პერსონაჟი რეალურია? ?“
  - კი: მოღვაწეობდა მეცნიერებაში?
    - კი:
      - ქალია?
        - კი -> მარია კიური
        - არა -> ალბერტ აინშტაინი ?“
    - არა:

- მსახიობია?
    - კი -> ლეონარდო დიკაპრიო
    - არა -> ვაჟა-ფშაველა
- არა: ადამიანია?
    - კი: -> ჰარი პოტერი
    - არა -> შრეკი

გადაწყვეტილების ხეც ზუსტად ამ პრინციპით მუშაობს - ყოველი პასუხი გეხმარება, რომ შეამცირო შესაძლო ვარიანტები და საბოლოოდ მიიღო სასურველი პასუხი..



დავალება 9: "მონაცემთა ბაზის შექმნა"

შექმენი ორი სია. პირველ სიაში ჩაწერე 3-4 წინადადება, ხოლო მეორეში - მათი შესაბამისი განზრახვები (მაგ. "მისალმება", "კითხვა"). დაწერე კოდი, რომელიც for ციკლის გამოყენებით ეკრანზე გამოიტანს თითოეულ წინადადებას და მის განზრახვას.

დავალება 9.1: შეცდომის პოვნა და გასწორება

მოცემულ კოდში დაშვებულია შეცდომა. შენი ამოცანაა, იპოვო და გაასწორო ის, რათა პროგრამამ პირობის შესაბამისად იმუშაოს.

```
# შეცდომით დაწერილი კოდი
sentences = ["გამარჯობა", "როგორ ხარ", "ნახვამდის"]
```

```
intents = ["მისალმება", "კითხვა", "დამშვიდობება"]

for sentence, intent in sentences, intents:
    print(f"წინადადება: {sentence}, განზრახვა: {intent}")
```

დავალება 9.2: კოდის დასრულება

მოცემულ პროგრამულ კოდს აკლია ერთი ან რამდენიმე სტრიქონი. დაამატე მხოლოდ ის, რაც აუცილებელია, რომ პროგრამამ გამართულად იმუშაოს.

```
sentences = ["გამარჯობა", "როგორ ხარ", "ნახვამდის"]
intents = ["მისალმება", "კითხვა", "დამშვიდობება"]
```

დავალება 9.3: შეიმუშავე პროგრამული კოდი

დაწერე პროგრამული კოდი, შექმენი პროგრამა, შეიმუშავო პრორგამული კოდი, რომელიც შექმნის ორ სიას (წინადადებები და განზრახვები), შემდეგ კი for ციკლის გამოყენებით დაბეჭდავს თითოეულ წყვილს.

```
# დაწერე შენი კოდი აქ:
```

სწორი პასუხი (პროგრამული კოდი სრულად):

```
# ამ კოდის საშუალებით შეგიძლია გადაამოწმო შენი ნამუშევარი

# შექმენი ორი სია მონაცემებისთვის
sentences = ["გამარჯობა", "როგორ ხარ", "ნახვამდის"]
intents = ["მისალმება", "კითხვა", "დამშვიდობება"]

# ციკლის გამოყენება მონაცემების ერთდროულად დასათვალიერებლად
for i in range(len(sentences)):
    print(f"წინადადება: {sentences[i]}, განზრახვა: {intents[i]}")
```