შეხვედრა 14: შეჯამება და ფინალური პროექტი

მოხარული ვარ, რომ ამ ტექსტს კითხულობთ. ეს ნიშნავს, რომ გრძელი გზა გაიარე და დღეს ჩვენი კურსის ბოლო შეხვედრას ესწრები. დღეს ჩვენ შევაჯამებთ მიღებულ ცოდნას, ვისაუბრებთ ხელოვნური ინტელექტის ეთიკურ ასპექტებზე და, რაც მთავარია, დავიწყებთ მუშაობას ჩვენს ფინალურ პროექტზე. ამ შეხვედრის ბოლოს, შენ შეძლებ ააგო სრულფასოვანი ჩატბოტი, რომელიც დაფუძნებულია ყველა იმ ცოდნაზე, რომელიც კურსის განმავლობაში მიიღე - Python-ის საფუძვლებიდან მანქანური სწავლების ალგორითმებამდე.

1. რა ვისწავლეთ ამ კურსის ფარგლებში?

ამ კურსის განმავლობაში შენ გაიარე გზა `print("Hello World")`-დან მანქანური სწავლების მოდელის აგებამდე. თუ დავფიქრდებით, სავარაუდოდ, რამდენიმე თვის წინ, საერთოდ არ იცოდი Python-ის შესახებ. ახლა კი შეგიძლია შექმნა პროგრამები, მართო მონაცემები, გამოიყენო ციკლები და პირობები. შენ გესმის, რა არის ალგორითმი, როგორ ამუშავებს კომპიუტერი ენას (NLP), რა არის მანქანური სწავლება და როგორ მუშაობს ნეირონული ქსელი. ეს ნამდვილად დიდი მიღწევაა!

1.1. Python-ის საფუძვლები, NLP-ის კონცეფციები და ML-ის პრაქტიკული გამოყენება

ამ კურსის განმავლობაში, ჩვენ შევისწავლეთ:

- **Python-ის საფუძვლები:** ცვლადები, მონაცემთა ტიპები, `if-else` პირობები, `for` ციკლები, სიები, ლექსიკონები და ფუნქციები.
- **ბუნებრივი ენის დამუშავება (NLP):** ტექსტის მომზადება კომპიუტერისთვის.
- **მანქანური სწავლება (ML):** რა არის ზედამხედველობითი სწავლა `Supervised Learning` და როგორ მუშაობს `Logistic Regression` ალგორითმი.

1.2. როგორ ვაქციოთ იდეა მოქმედ პროგრამად

ჩვენ არა მხოლოდ თეორია შევისწავლეთ, არამედ ვისწავლეთ, თუ როგორ გადავიტანოთ ეს ცოდნა პრაქტიკაში და როგორ დავწეროთ პროგრამა, რომელიც რეალურ ამოცანას ხსნის.

2. ეთიკა და ხელოვნური ინტელექტი (შეჯამება)

ხელოვნურ ინტელექტთან მუშაობისას, ძალიან მნიშვნელოვანია გვახსოვდეს ეთიკური პასუხისმგებლობა. ტექნოლოგია, რომელიც ასე სწრაფად ვითარდება, ბადებს ახალ კითხვებს მონაცემთა კონფიდენციალურობასა და ალგორითმულ მიკერძოებასთან დაკავშირებით.

2.1. მონაცემთა კონფიდენციალურობა და ალგორითმული მიკერძოება

როგორც უკვე განვიხილეთ, ხელოვნური ინტელექტი იმდენად „ჭკვიანი“ და სანდოა, რამდენადაც ხარისხიანია ის ინფორმაცია, რომლის გამოყენებითაც არის ის გაწვრთნილი. თუ მონაცემები მიკერძოებულია (მაგალითად, შეიცავს გარკვეულ სოციალურ სტერეოტიპებს), მოდელიც ამ მიკერძოებას ისწავლის. ხელოვნურ ინტელექტზე დაფუძნებული სისტემა შეგვიძლია შევადაროთ სკოლის მოსწავლეს. თუ მოსწავლეს არასწორ ან არასრულ ინფორმაციაზე დაყრდნობით ვასწავლით, მას ცრუ წარმოდგენები შეექმნება.

როგორც მომავალი ექსპერტი ხელოვნური ინტელექტის მიმართულებით, ყოველთვის უნდა დაფიქრდე, ხომ არ შეიცავს მონაცემები რომელთანაც მუშაობ უზუსტობას. შეიცავს თუ არა პირად მონაცემებს და, ასეთ შემთხვევაში, რამდენად დაცულია პირადი მონაცემები.

2.2. ხელოვნური ინტელექტის პასუხისმგებლიანი გამოყენების მნიშვნელობა

შენ ახლა გაქვს უძლიერესი ინსტრუმენტი ხელში. ამიტომ, ყოველთვის დაფიქრდი, თუ როგორ გამოიყენებ მას. იფიქრე იმაზე, როგორ შეძლებ, ხელოვნური ინტელექტის დახმარებით უკეთესი მომავალი შექმნა.

3. ფინალურ პროექტზე მუშაობის დაწყება

დროა დავიწყოთ ფინალურ პროექტზე მუშაობა. გახსოვდეს, ეს პროექტი იმისთვისაა, რომ შენი ცოდნა პრაქტიკაში გამოსცადო. ასე, რომ არ შეგეშინდეს შეცდომების დაშვების.

3.1. კითხვა-პასუხის სესია და პროექტთან დაკავშირებული საკითხების გარკვევა

ამ ეტაპზე შენ უკვე გაქვს საკმარისი ცოდნა, რომ პროექტზე დამოუკიდებლად დაიწყო მუშაობა. მაგრამ ყოველთვის გახსოვდეს, რომ შეგიძლია მენტორს ან სხვა მოსწავლეებს მიმართო დახმარებისთვის.

3.2. დამოუკიდებელი მუშაობის დაწყება მენტორის დახმარებით

ჩვენი საბოლოო დავალება არის შევქმნათ anITა-ს ასისტენტი. ეს იქნება ჩატბოტი, რომელიც შეძლებს მომხმარებლის კითხვებზე პასუხის გაცემას ჩვენი კურსის შესახებ. შენ შექმნი ამ ბოტს Google Colab-ში. Colab-ი საშუალებას მოგცემს, მარტივად ატვირთო მონაცემები, დაწერო და გაუშვა Python-ის კოდი, და შეამოწმო როგორ მუშაობს შენი მოდელი. ამით შენ გეძლევა შესაძლებლობა შეაფასო რამდენად დაეუფლე პროგრამირებისა და მანქანური სწავლების პრინციპებს.

დავალება 14 (ფინალური პროექტი): "anITა-ს ასისტენტი"

აღწერა: Google Colab-ში, გამოიყენე კურსის განმავლობაში მიღებული ცოდნა და შექმენი მარტივი ჩატბოტი.

დავალების დეტალები:

1. **მონაცემები:** AnITa-ს კურსის შესახებ შექმენი მცირე ზომის კითხვა-პასუხის ბაზამონაცემთა ბაზა უნდა შეიცავდეს ორ სვეტს: `კითხვა` და `პასუხი`.
    - მაგალითად:
        - კითხვა: "რამდენი შეხვედრაა კურსში?"
        - პასუხი: "კურსი შედგება 14 შეხვედრისგან."
        - კითხვა: "რას ნიშნავს NLP?"
        - პასუხი: "NLP ნიშნავს ბუნებრივი ენის დამუშავებას."
2. **მოდელი:** გამოიყენე `scikit-learn`-ი, რათა გაავარჯიშო კლასიფიკატორი, რომელიც მომხმარებლის კითხვის მიხედვით ამოიცნობს სწორ პასუხს. ამისათვის, შეგიძლია გამოიყენო `TfidfVectorizer`-ი ტექსტის რიცხვებად გადასაქცევად და `LogisticRegression`-ი კლასიფიკაციისთვის.
    - **Hint:** ტექსტების სიტყვებად გადასაქცევად, გამოიყენე `TfidfVectorizer`-ი (ეს არის `CountVectorizer`-ის გაუმჯობესებული ვერსია, რომელიც სიტყვების მნიშვნელობას ითვალისწინებს).
3. **ინტერფეისი:** შექმენი უსასრულო `while` ციკლი, რომელიც მომხმარებელს საშუალებას მისცემს, მუდმივად დაუსვას კითხვები ბოტს და მიიღოს პასუხები.
    - **Hint:** `while True:` ბრძანება ქმნის უსასრულო ციკლს.

პლატფორმა: Google Colab (პროექტის ბმული უნდა აიტვირთოს AnITa-ს პლატფორმაზე).

დავალება 14.1: კოდის მომზადება ფინალური პროექტისთვის

შექმენი Google Colab-ის გარემოსთვის განკუთვნილი საწყისი კოდი, რომელიც მოიცავს საჭირო ბიბლიოთეკების იმპორტს და მონაცემთა ბაზის შექმნას.

```
# ამ ველში შექმენი კოდი, რომლითაც დაიწყებ ფინალურ პროექტს.
# 1. საჭირო ბიბლიოთეკების იმპორტი
# 2. მონაცემთა ბაზის შექმნა DataFrame-ის სახით
# და ბოლოს, დაბეჭდე DataFrame-ი, რომ დარწმუნდე, რომ სწორად შეიქმნა

import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
course_data_dict = {
"კითხვა 1": "პასუხი 1",
"კითხვა 2": "პასუხი 2",
}
```

სწორი პასუხი (პროგრამული კოდი სრულად):

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression

# ნაწილი 1: მონაცემთა ბაზის შექმნა
# ვარიანტი 1: მონაცემების გენერირება dictionary-თი, სადაც key არის კითხვა და value - პასუხი.
course_data_dict = {
    # შეხვედრა 9: ზედამხედველობითი სწავლა
    "რა არის კლასიფიკაცია?": "კლასიფიკაცია არის მანქანური სწავლების ფუნდამენტური პროცესი, რომლის დროსაც პროგრამა ცდილობს განსაზღვროს, რომელ წინასწარ განსაზღვრულ კატეგორიას (კლასს) მიეკუთვნება მიღებული მონაცემები. მაგალითად, ელექტრონული ფოსტის დახარისხება 'სპამად' და 'არა-სპამად'.",
    "რას ნიშნავს მრავალკლასიანი კლასიფიკაცია?": "მრავალკლასიანი კლასიფიკაციაა, როდესაც შესაძლო პასუხების რაოდენობა ორზე მეტია, მაგალითად, სურათის ამოცნობისას ('კატა', 'ძაღლი', 'ჩიტი').",
    "რა არის Bag-of-Words მოდელი?": "Bag-of-Words არის ტექსტის რიცხვებად გადაქცევის ერთ-ერთი მარტივი მეთოდი. მოდელი აგროვებს ყველა
```

უნიკალურ სიტყვას და ითვლის, რამდენჯერ გვხვდება თითოეული მათგანი წინადადებაში.",
    "რა არის გადაწყვეტილების ხე?": "გადაწყვეტილების ხე (Decision Tree) მანქანური სწავლების ერთ-ერთი ყველაზე მარტივი ალგორითმია. ის გადაწყვეტილების მისაღებად მონაცემებს ლოგიკური 'კი/არა' კითხვების დასმით ჰყოფს.",
    "რას ვისწავლით მე-9 შეხვედრაზე?": "მე-9 შეხვედრაზე ვისწავლით **ზედამხედველობით სწავლას** და განზრახვის კლასიფიკაციას, რომლის მიზანია მომხმარებლის შეტყობინებების და მათი განზრახვების დაჯგუფება.",

    # შეხვედრა 10: მონაცემთა ანალიზის ბიბლიოთეკები
    "რა არის Pandas-ი?": "Pandas-ი არის მონაცემთა ანალიზის ერთ-ერთი ყველაზე პოპულარული და ძლიერი ინსტრუმენტი Python-ში. ის შეიცავს ყველა საჭირო ინსტრუმენტს მონაცემების გასაანალიზებლად და დასამუშავებლად.",
    "რა არის DataFrame-ი?": "DataFrame-ი არის Pandas-ის მთავარი სტრუქტურული ელემენტი, რომელიც მონაცემებს ინახავს ჭკვიანი ცხრილის სახით, Excel-ის მსგავსად, მაგრამ მეტი შესაძლებლობებით.",
    "რას აკეთებს head() ფუნქცია Pandas-ში?": "head() ფუნქცია აჩვენებს DataFrame-ის ცხრილის პირველ ხუთ მწკრივს, რაც მონაცემების სტრუქტურის სწრაფად შესამოწმებლად გამოიყენება.",
    "როგორ ხდება მონაცემების ჩატვირთვა CSV ფაილიდან?": "Pandas-ს შეუძლია CSV ფაილების მარტივად ჩატვირთვა read_csv() ფუნქციის გამოყენებით.",

    # შეხვედრა 11: პირველი ML მოდელის აგება
    "რას ვისწავლით მე-11 შეხვედრაზე?": "მე-11 შეხვედრაზე ვისწავლით სრულფასოვანი მანქანური სწავლების მოდელის აგებას Scikit-learn ბიბლიოთეკის გამოყენებით, რომელსაც შეეძლება წინადადებებიდან განზრახვის ამოცნობა.",
    "რა არის Scikit-learn?": "Scikit-learn არის Python-ის ღია ბიბლიოთეკა, რომელიც განკუთვნილია მანქანური სწავლების მოდელების შექმნის, გაწვრთნისა და შეფასებისთვის. მას შეუძლია გადაჭრას კლასიფიკაციის, რეგრესიის და კლასტერიზაციის ამოცანები.",
    "რა არის ტექსტის ვექტორიზაცია?": "ვექტორიზაცია არის ტექსტის რიცხვების ვექტორად გადაქცევის პროცესი, რადგან მანქანური სწავლების მოდელებს მხოლოდ რიცხვებთან მუშაობა შეუძლიათ.",
    "რომელ ინსტრუმენტს ვიყენებთ ვექტორიზაციისთვის?": "ტექსტის ვექტორებად გარდაქმნისთვის ვიყენებთ Scikit-learn-ის ინსტრუმენტს CountVectorizer-ს, რომელიც Bag-of-Words მოდელის პრინციპით მუშაობს.",
    "რას აკეთებს fit() მეთოდი?": "fit() მეთოდი არის მოდელის სწავლის ყველაზე მნიშვნელოვანი ეტაპი. ის იღებს სასწავლო მონაცემებს (X_train)

```
და სწორ პასუხებს (y_train), რის საფუძველზეც მოდელი სწავლობს მათ შორის კავშირებს.",
    "რას აკეთებს predict() მეთოდი?": "predict() მეთოდი გამოიყენება გაწვრთნილი მოდელისთვის, რათა მან ახალ, უცნობ მონაცემებზე პროგნოზი გააკეთოს.",

    # შეხვედრა 12: შესავალი ნეირონულ ქსელებში
    "რა არის ნეირონული ქსელი?": "ნეირონული ქსელი არის კომპიუტერული მოდელი, რომელიც შთაგონებულია ადამიანის ტვინის სტრუქტურით და შედგება ერთმანეთთან დაკავშირებული ხელოვნური 'ნეირონებისგან'.",
    "რა არის ღრმა სწავლება?": "ღრმა სწავლება (Deep Learning) არის ხელოვნური ინტელექტის ქვესფერო, რომელიც იყენებს ნეირონულ ქსელებს მრავალი ფენით (ასობით ან ათასობით), რათა მონაცემებში რთული კანონზომიერებები აღმოაჩინოს.",
    "როგორია ნეირონული ქსელის სტრუქტურა?": "ნეირონული ქსელი შედგება ფენებისგან: შესასვლელი ფენა (Input Layer) იღებს ინფორმაციას, დამალული ფენები (Hidden Layers) ამუშავებენ მას, ხოლო გამომავალი ფენა (Output Layer) იღებს საბოლოო გადაწყვეტილებას.",
    "რა არის Fine-tuning-ი?": "Fine-tuning-ი არის პროცესი, როდესაც დიდ ენობრივ მოდელს ვასწავლით კონკრეტულ, მცირე მონაცემთა ბაზაზე, რათა ის კონკრეტულ სფეროში სპეციალისტად ვაქციოთ."
}

# რადგან dictionary-ს სტრუქტურა შევცვალეთ, მას pandas DataFrame-ად გარდავქმნით შემდეგნაირად:
# 1. ამოვიღოთ კითხვები (key-ები) ცალკე სიაში.
questions = list(course_data_dict.keys())
# 2. ამოვიღოთ პასუხები (value-ები) ცალკე სიაში.
answers = list(course_data_dict.values())
# 3. სიებისგან შევქმნათ DataFrame სასურველი სვეტებით.
df = pd.DataFrame({
    'კითხვა': questions,
    'პასუხი': answers
})


# --- ალტერნატივა: მონაცემების ჩატვირთვა ფაილიდან ---
# თუ კოდის ამ ნაწილს გამოიყენებთ, ზედა dictionary-ს შექმნის და DataFrame-ად გარდაქმნის ნაწილი უნდა დააკომენტაროთ.
# ფაილში უნდა იყოს ორი სვეტი: 'კითხვა' და 'პასუხი'.
```

```
# ვარიანტი 2.1: მონაცემების ჩატვირთვა CSV ფაილიდან
# df = pd.read_csv('anita_course_qa.csv')

# ვარიანტი 2.2: მონაცემების ჩატვირთვა Excel ფაილიდან
# # Excel-ის წასაკითხად შეიძლება დაგჭირდეთ openpyxl ბიბლიოთეკა: pip
install openpyxl
# df = pd.read_excel('anita_course_qa.xlsx')
# ----------------------------------------------------------


# მონაცემების მომზადება მოდელისთვის (ეს ნაწილი უცვლელია)
X = df['კითხვა']
y = df['პასუხი']

# ნაწილი 2: მოდელის შექმნა და გავარჯიშება
vectorizer = TfidfVectorizer()
X_vectorized = vectorizer.fit_transform(X)

model = LogisticRegression()
model.fit(X_vectorized, y)

print("AnITa Bot: გამარჯობა! მე მზად ვარ, ვუპასუხო თქვენს კითხვებს
AnITa-ს კურსის შესახებ.")
print("შეგიძლიათ ჩაწეროთ 'გასვლა' საუბრის დასასრულებლად.")
print("-" * 30)


# ნაწილი 3: ინტერაქტიული ინტერფეისი
while True:
    user_question = input("თქვენი კითხვა: ")

    if user_question.lower() == 'გასვლა':
        print("AnITa Bot: დროებით! წარმატებებს გისურვებთ!")
        break

    user_question_vectorized = vectorizer.transform([user_question])
    predicted_answer = model.predict(user_question_vectorized)[0]
    print(f"პასუხი: {predicted_answer}\n")
```