FUTURE STEM HUB

Образователни материали Практически задачи по Python и Scratch

FUTURE-STEM-HUB

Подобряване на STEM образованието в средните училища чрез обучение и ресурси за ученици и преподаватели в областта на изкуствения интелект

Референтен номер
2024-1-DE03-KA220-SCH-000247346

Co-funded by the European Union

Проектът FUTURE-STEM-HUB „Подобряване на STEM образованието в средните училища чрез обучение и ресурси за ученици и преподаватели в областта на изкуствения интелект “ (реф. № 2024-1-DE03-KA220-SCH-000247346) е съфинансиран от Програма Еразъм+ на Европейския съюз. Той се координира от Университета Дуйсбург-Есен (Германия) и включва партньорските организации: M&M Profuture Training (Испания), Kütahya MEM - Регионална Дирекция Кютая към Министерството на националното образование (Турция), COOPETAPE CRL - Образователен кооператив, надзорен орган на училище ETAP (Португалия) и Тетра Солюшънс (България).

Образователните материали „Практически задачи по Python и Scratch“ на FUTURE-STEM-HUB са разработени от членовете на екипа на проекта и включват специалисти от всички партньорски организации. Целта им е да запознаят учениците от средните училища с основните концепции на изкуствения интелект (ИИ), като същевременно насърчават осведомеността и дискусията относно техните социални и етични последици.

Автори:
Мустафа Билгин, Университет Дуисбург-Есен (Германия)
Моника Морено, M&M Profuture Training (Испания)
Монсерат Ренедо, M&M Profuture Training (Испания)
Жоао Барозу, училище ETAP (Португалия)
Анджелина Преса, училище ETAP (Португалия)
Силвия Георгиева, Тетра Солюшънс (България)
Бориславa Захариева-Томова, Тетра Солюшънс (България)
Йелиз Юртер, Kütahya MEM (Турция)
Йозджан Туран, Kütahya MEM (Турция)

Редактор:
Мустафа Билгин, Университет на Дуисбург-Есен

Списък с фигури

Дизайн: Тетра Солюшънс
Изображение на корицата: Мустафа Билгин (хибридно създаване чрез Google Gemini)
Стр. 6: Графика от Тетра Солюшънс
Стр. 7, 8: Мустафа Билгин (хибридно създаване чрез Google Gemini).
Графични емоджита/икони: Flaticon.com

Отказ от отговорност и безопасност

Използването на програмните кодове, предоставени в тези материали, както и достъпът до външни уебсайтове и платформи на трети страни, е изцяло на риск и отговорност на потребителите. Екипът на проекта не поема отговорност за съдържанието, функционалността или наличността на външни връзки, като Google Colab, Scratch или Machine Learning for Kids. Тъй като много от интерактивните задачи изискват активна интернет връзка и обработка на данни на облачни сървъри, отговорността за спазване на стандартите за сигурност и системните изисквания (като например текущите версии на браузъра) е на потребителите. При използване на инструменти на трети страни за създаване на ИИ модели не може да се качва лична или чувствителна информация. Тъй като образователните материали са предназначени за ползване от ученици от средните училища, регистрацията и използването на външни платформи и програми, трябва да се извършва в съответствие с условията за ползване на съответните доставчици и в идеалния случай под наблюдението и със съгласието на учител или законен настойник. Не се поема отговорност за загуба на данни или повреди на използваните крайни устройства, произтичащи от изпълнението на предоставените примерни кодове.

Списък на съкращения:

AI: Изкуствен интелект

CNN: Конволюционни невронни мрежи

CV: Компютърно зрение

EACEA: Европейска изпълнителна агенция за образование и култура

FPS: Кадри в секунда

IBM: Международни бизнес машини

IDE: Интегрирана среда за разработка

ML: Машинно обучение

ML4K: Машинно обучение за деца

MIT: Масачузетски технологичен институт

MNIST: Модифицирана база данни на Националния институт за стандарти и технологии

NN: Невронни мрежи

NLP: Обработка на естествен език

RGB: Червено, Зелено, Синьо (цветен модел)

STEM: Наука, технологии, инженерство и математика

TTS : Преобразуване на текст в реч

ЮНЕСКО: Организация на обединените нации за образование, наука и култура

YOLO: Ти Погледни само веднъж

Съдържание

Empowering Secondary School STEM Education with AI Training and Resources for Students and Educators
2024-1-DE03-KA220-SCH-000247346

FUTURE-STEM-HUB

За проекта

Проектът FUTURE-STEM-HUB има за цел да улесни интегрирането на теми, свързани с изкуствения интелект (ИИ), в STEM образованието в средните училища чрез: 1) Разработване на образователни материали, представящи концепциите за ИИ и техните социални последици; 2) Създаване на практически учебни ресурси, които да позволят на учениците да изследват ИИ, чрез програмиране с Python, и 3) Предоставяне на подкрепа за учителите в процеса на интегриране на ИИ в STEM обучението в средното образование.

Резултати от проекта

1. **Курс 1: Дигитален буквар: Основи на изкуствения интелект** (Въведение в ИИ чрез интерактивни образователни материали за ученици в гимназиална степен на обучение)

2. **Курс 2: Задълбочаване в изкуствения интелект с Python и Scratch** (Изкуствен интелект за напреднали: Практически учебни материали за ученици в гимназиална степен на обучение)

3. **Наръчник за преподаватели: Усъвършенстване на уменията за преподаване на изкуствен интелект** (Методическо ръководство за работа с изкуствен интелект за учители в гимназиална степен)

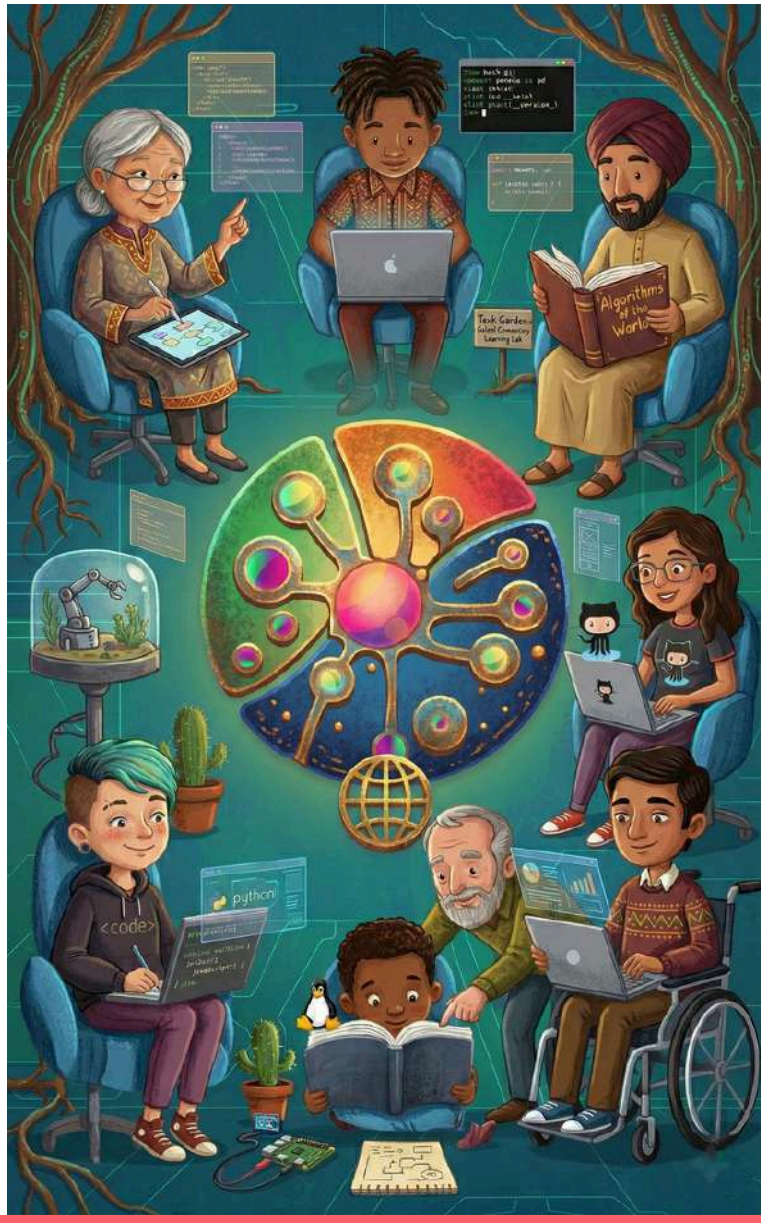
Въведение

Добре дошли в образователните материали „Практически задачи по Python и Scratch" – ангажиращо и интерактивно учебно пътуване, създадено да въведе учениците от средните училища в увлекателния и практичен свят на изкуствения интелект. Целта на тези материали е да осигурят на учители и ученици от средните училища достъпни, практически ориентирани ресурси за изследване на ключови концепции на изкуствения интелект чрез Python и Scratch. Разработени за ученици на възраст 15–18 години с различни нива на STEM подготовка, материалите са подходящи за самостоятелно обучение в асинхронен формат и могат безпроблемно да бъдат интегрирани в класната стая с подкрепата на учителя.

Съдържанието е разделено на пет модула, фокусирани върху практически умения за програмиране на Python и Scratch, математическа логика чрез обучение, базирано на игри (Escape Room), творчески приложения на изкуствения интелект, решаване на реални проблеми с данни и етични съображения.

Всеки модул съчетава теория, тестове и практически упражнения, за да осигури цялостно разбиране на концепциите за изкуствен интелект.

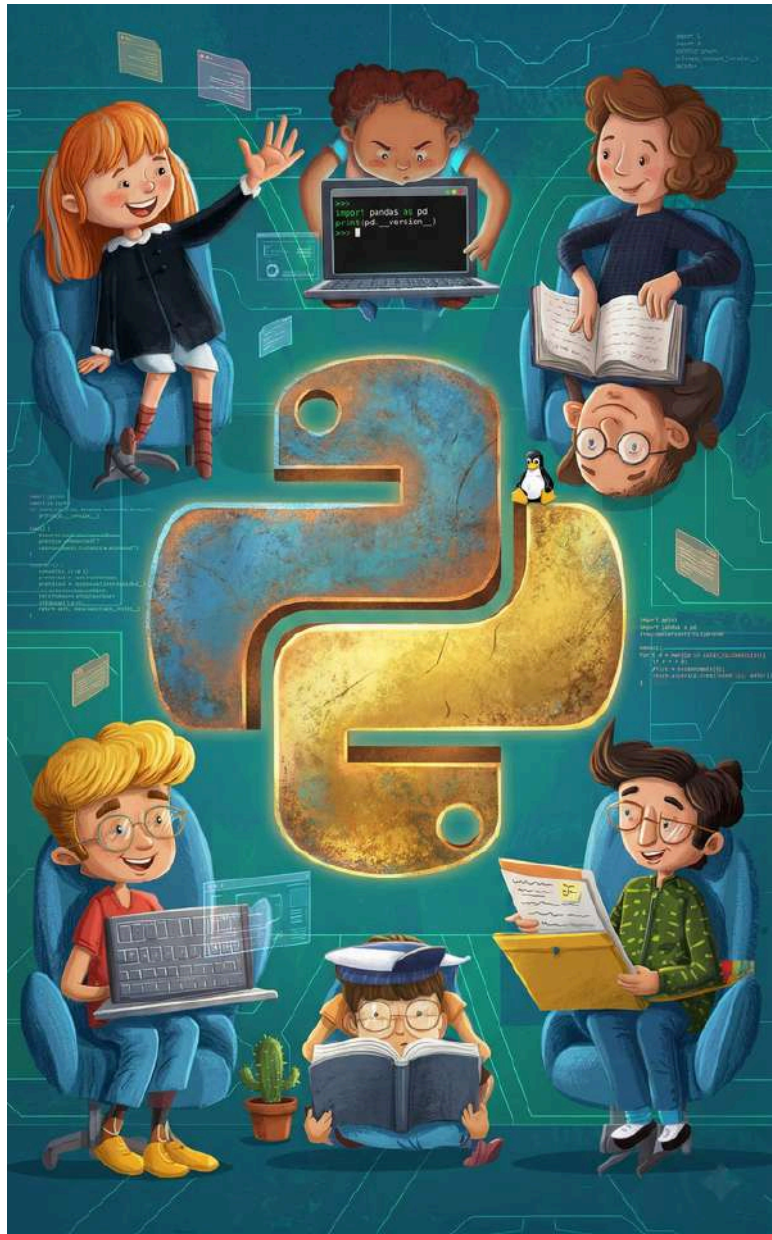
Линковете към допълнителните ресурси предоставят възможност на учениците, които имат интерес, да научат повече по темите, заложени в съответния модул. Предвидената продължителност на обучението за всичките пет модула е приблизително 10 часа.



FUTURE-STEM-HUB

След завършване на курса учениците ще притежават познания за ключови концепции в областта на изкуствения интелект, като машинно обучение, невронни мрежи и дълбоко обучение. Те ще придобият задълбочени знания за историята и еволюцията на изкуствения интелект и неговите приложения в различни индустрии, като разберат как новите технологии в тази област оформят съвременните тенденции и света. Учениците ще се запознаят по-подробно с основни технологии, включително компютърно зрение и обработка на естествен език, както и ще придобият задълбочено разбиране за етичните съображения, свързани с ИИ, неговата отговорност и въздействието му върху обществото.
Същевременно учителите ще получат достъп до иновативни, интерактивни ресурси, които могат лесно да използват и прилагат в STEM класната стая в подкрепа на учебните си програми и преподаването. Тези материали ще им помогнат да създадат по-интересна и ангажираща учебна среда, тъй като предлагат различни подходи за активно включване на учениците в процеса на обучение.

Образователните материали ще бъдат преобразувани в онлайн курс на платформата FUTURE-STEM-HUB, достъпна от уебсайта на проекта: www.future-stem-hub.eu



След успешно завършване на курса Задълбочаване в изкуствения интелект с Python и Scratch, учениците ще получат дигитален сертификат за верифициране и признаване на знанията им.

МОДУЛ 1: Мини предизвикателства по програмиране: Машинно обучение и невронни мрежи.

ВЪВЕДЕНИЕ

Чудили ли сте се някога как смартфонът ви знае кои видеоклипове може да ви харесат? Или как едно приложение може да разпознае вашия почерк? Зад всичко това стои изкуственият интелект (ИИ) – който е далеч по-малко мистериозен, отколкото много хора си мислят! В тази глава ще се присъедините към Айлин и нейния умен смарт часовник Алара. Заедно те ще ви поведат на приключения, които разкриват как всъщност работи изкуственият интелект. Ще разберете основните принципи и компоненти на невронните мрежи (NN) и машинното обучение (ML), включително как те обработват входни данни, учат се от тях и вземат решения. Най-хубавото? Не е нужно да сте експерти по програмиране! Всяко приключение включва малки предизвикателства по програмиране в три нива на трудност. Просто изберете нивото, което ви подхожда най-добре – от начинаещ до истинско предизвикателство. Ще видите как още след първия раздел вече ще знаете как машините „се учат да мислят". Нека се потопим заедно в света на изкуствения интелект.

Ключови теми и концепции:

- **Въведение в изкуствения интелект:** Мистериозният подарък – часовник, който прави повече от тиктакане
- **Невронни мрежи – основи:** Алара се събужда
- **Дървета на решенията:** Детективът по училищния маршрут – как мисли твоят часовник
- **Класификация, базирана на правила:** Как Spotify може да знае какво харесваш
- **Обучение чрез подкрепяне:** Учене чрез награди
- **Конволюционни невронни мрежи:** Как изкуственият интелект открива ръбове в изображенията

- **Обработка на естествен език:** Как компютрите разбират езика
- **Системи за препоръки:** Как YouTube знае какво искаш да гледаш
- **Прогноза за времето:** Прогноза за времето
- **Етика на изкуствения интелект:** Визия за бъдещето – какво можем и трябва да правим с изкуствения интелект

Продължителност на модула	**2 часа (1 час преподаване в клас + 1 час практически упражнения)**

ОБРАЗОВАТЕЛНИ МАТЕРИАЛИ

АЙЛИН е на 17 години, тя е любопитна и обича да разгадава как работят различни неща.
Тя никога не се предава!

ЛЕНА е най-добрата приятелка на Айлин.
Тя е креативна и обича музиката.
Нейното семейство е от Полша.
Понякога носи вкусни пирожки!

ALARA is no ordinary smartwatch.
She's an AI that teaches Aylin about artificial intelligence.

Фигура 1 Образователни герои Айлин, Алара и Лена; Концепция и оформление: Мустафа Билгин; Разработено с Google Gemini.

РАЗДЕЛ 1.1 КОНЦЕПЦИЯ И ПОДХОД НА МОДУЛА

Учебният процес се ръководи от разказна рамка, която свързва STEM темите с реални житейски контексти, близки до учениците. Сложни понятия като невронни мрежи или обучение чрез подкрепяне са представени в опростена, прототипна форма, което позволява на учениците да изградят основно разбиране и познания без да е необходимо да притежават предварителен опит по програмиране. Тези материали са подходящи за часовете по информатика и технически дисциплини, STEM проекти, извънкласни дейности или интердисциплинарни учебни среди. Те не само насърчават техническото разбиране, но и стимулират критично осмисляне на морални и социални въпроси, посредством темите, които разглеждат етиката в изкуствения интелект.

РАЗДЕЛ 1.2 ВЪВЕДЕНИЕ В ИЗКУСТВЕНИЯ ИНТЕЛЕКТ (ИИ)

Преди да се потопите в практическите задачи, нека първо изясним какво всъщност представлява изкуственият интелект. ИИ се отнася до симулацията на човешкия интелект в машини, програмирани да мислят като хора и да имитират техните действия. Терминът може да се приложи и към всяка машина, която проявява черти, свързани с човешкия ум – като например учене и решаване на проблеми.

В този модул ще разгледаме основните градивни елементи на изкуствения интелект чрез интерактивни истории:

- **Невронни мрежи (NL):** Това са изчислителни системи, вдъхновени от биологичните невронни мрежи, изграждащи мозъците на живите същества. Те помагат на изкуствения интелект да „се учи" от примери.
- **Дървета на решенията:** Инструмент за подпомагане на вземането на решения, който използва дървовиден модел на решенията и техните възможни последици. Той представя логиката на принципа „Ако се случи това, тогава направи онова."
- **Машинно обучение (ML):** Подобласт на изкуствения интелект, която предоставя на системите способността да се учат автоматично и да се усъвършенстват от опит, без да бъдат изрично програмирани.

Този модул използва наративен подход (разказване на истории), за да направи тези сложни математически концепции достъпни и лесни за разбиране.

РАЗДЕЛ 1.3 НАСТРОЙКА НА СИСТЕМАТА И ПРЕДВАРИТЕЛНИ ИЗИСКВАНИЯ

Каква платформа ви е необходима?

За практическите задачи в този модул ще използвате **Google Colab** (и акаунт в Gmail) за изпълнение на Python код. Google Colab е безплатна онлайн платформа, която ви позволява да пишете и изпълнявате код директно в уеб браузъра си — без никаква инсталация.

Достъп до платформата:
https://colab.research.google.com/

Какво е включено в процеса на регистрация?

1. Отидете в Google Colab.
2. Кликнете върху „Вход“ в горния десен ъгъл.
3. Въведете имейл адреса и паролата си за Google.
4. След като влезете, можете да отворите или да създадете нов бележник.

Трябва ли да се регистрирате?

Да, необходим ви е **безплатен акаунт в Google** (Gmail акаунт), за да използвате Google Colab. Ако вече имате такъв, можете да влезете директно. Ако нямате акаунт в Google, трябва да си създадете такъв, преди да започнете.

За да създадете акаунт в Google: отидете на accounts.google.com и следвайте стъпките за регистрация.

Къде се пише кодът?

Всички упражнения по програмиране в Модул 1 са подготвени в **тетрадка в Google Colab**. Не е нужно да пишете код сами — ще използвате предварително подготвена тетрадка, която съдържа всички упражнения и инструкции.

Ето връзката към тетрадката за Модул 1: Модул 1 Colab Notebook

Какво да направите с тетрадката?

1. Отворете връзката по-горе.
2. Кликнете върху „Копиране в Drive", за да запазите своя редактируем екземпляр.
3. Следвайте инструкциите в тетрадката, като изпълнявате всяка клетка с код стъпка по стъпка.

РАЗДЕЛ 1.4 ДОПЪЛНИТЕЛНИ ИНСТРУКЦИИ ПРЕДИ ДА ЗАПОЧНЕТЕ

1. Използвайте съвременен уеб браузър - Chrome , Firefox, Edge или Safari.

2. Уверете се, че имате стабилна интернет връзка, тъй като Google Colab работи изцяло онлайн.

3. Прочетете внимателно историята и инструкциите за всяка станция, преди да пристъпите към задачата по програмиране.

4. Всяка станция предлага **три нива на трудност**:
   - **Основно** - подходящо за начинаещи.
   - **Средно** - ако имате известен опит с Python.
   - **Трудно**- ако търсите наистина сериозни предизвикателства.
   - Изберете нивото, което отговаря на вашите умения.

5. 1. Изпълнявайте клетките с код по ред - някои упражнения зависят от предходните.

6. Ако се затрудните, използвайте предоставените **„Подсказки"** и **„Части от решения"** във всяко упражнение.

7. След като изпълните дадена задача, опитайте да промените кода и наблюдавайте как се отразява на резултата. Експериментирането е един от най-добрите начини за учене.

ИЗТОЧНИЦИ

Атрибуция на емоджита/икони

1. Този документ използва стандартни Unicode емоджита, дефинирани от Unicode Consortium. Конкретните версии на емоджитата може да се различават в зависимост от операционната система на потребителя. Unicode Consortium . (2022). Извлечено от https://unicode.org/emoji/charts/full-emoji-list.html

Книги и статии

2. Russell, S., & Norvig, P. (2021). Artificial Intelligence: A Modern Approach (4th ed.). Pearson. (A comprehensive textbook on AI concepts, including neural networks, decision trees, and reinforcement learning.)

3. Géron, A. (2022). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (3rd ed.). O'Reilly. (Practical introduction to ML and neural networks with Python.)

Етика и общество в ИИ

4. European Commission. (2019). Ethics Guidelines for Trustworthy AI. A framework for ethical AI development and deployment.)

5. UNESCO. (2021). Recommendation on the Ethics of Artificial Intelligence. (Global guidelines on AI ethics, focusing on human rights and inclusivity.)

Програмиране с Python за начинаещи

6. Sweigart, A. (2019). Automate the Boring Stuff with Python (2nd ed.). No Starch Press. (Beginner-friendly Python book with practical projects.)

ЛИНКОВЕ КЪМ УЧЕБНИ ПЛАТФОРМИ

- Тетрадка за Модул 1: https://colab.research.google.com/drive/14VaJnlgo5habZ0-N9jpz9nWZSVAw_Xwa?usp=sharing
- Официална документация на Python:
- https://docs.python.org/3/ (Официален справочник и уроци за езика Python.)

ПРАКТИЧЕСКО УПРАЖНЕНИЕ

Станция 1: Мистериозният подарък - часовник, който прави повече от тиктакане

Днес е 17-ият рожден ден на Айлин — между баклава и балони се крие едно мистериозно малко пакетче. Айлин се колебае да го отвори.

АЙЛИН: *„Какво е това, лельо Сема?"*

ЛЕЛЯ СЕМА: *„Нещо специално, скъпа. Баба казваше: „Този часовник има душа."*

Айлин внимателно разопакова пакетчето. Вътре тя намира елегантен смарт часовник с тюркоазена каишка. Изведнъж дисплеят светва.

АЛАРА (часовникът): *„Здравей, Айлин! Аз съм Алара — твоят нов приятел с изкуствен интелект!*

Айлин отскача назад от изненада.

АЙЛИН: *„Ча... часовникът говори?"*

ЛЕЛЯ СЕМА: *„Говори на всеки език, който разбираш. Алара ще ти покаже как всъщност работи изкуственият интелект!"*

АЛАРА: *„Представи си, че съм като твоя мозък — имам неврони, които се учат, и синапси, които ги свързват. Заедно ще открием как машините се учат да мислят!"*

ОСНОВНО НИВО

КОД:

```
NAME = INPUT("WHAT'S YOUR NAME? ")
PRINT("HELLO " + ____ + "! I'M ALARA, YOUR AI FRIEND")
```

Подсказка: Използвайте променливата, която сте въвели в полето

```
Част от решението: name
```

Станция 1: Мистериозният подарък - часовник, който прави повече от тиктакане

СРЕДНО НИВО

КОД:

```
print("🎂 Aylin's Birthday Planning 🎂")
favorite_food = input("What's your favorite food?
(Pizza/Kebab/Burger): ")
drink = input("And what would you like to drink with that? ")

if favorite_food.lower() == "pizza":
    print("🍕 Alara says: Pizza is my favorite too!")
else:
    print("😋 Alara says: " + __ + " sounds great!")

print("And " + __ + " with that? Perfect combo!")
```

Подсказка: Попълнете празните полета с вашите входни променливи.

Част от решението: favorite_food, drink

Станция 2: Невронни мрежи - Алара се събужда

На следващата сутрин Айлин седи в парка, гледайки Алара.

АЙЛИН: *„Значи... мислиш като мен?"*

АЛАРА: *„Почти! Представи си, че мозъкът ми е като голяма фабрика за мислене. Имам неврони, които се учат. Виж..."*

АЙЛИН: *„Значи ще се учим заедно?"*

АЛАРА: *„Да"* — отговаря Алара. *„Познавам основите, но чрез теб се уча да виждам света през човешки очи."*

Станция 2: Невронни мрежи - Алара се събужда

ОСНОВНО НИВО

КОД:

```
print("Should I buy a new game?")
print("Rate from 0 to 1, where 0 = not at all, 0.5 = okay, 1
= very much")

savings_balance = float(input("How full is your piggy bank?
(0-1): "))
desire = float(input("How much do you want the game? (0-1):
"))

def decision_neuron(money, wish):
   if money * wish > ____:
      return "✅ YES, buy it!"
   else:
       return "❌ NO, better wait"

print(decision_neuron(savings_balance, desire))
```

Подсказка: Невронът „се активира", когато произведението е по-
голямо от 0,5
Част от решението: 0.5

Станция 2: Невронни мрежи - Алара се събужда

СРЕДНО НИВО (ЧАСТ I)

КОД:

```
print("Is it a good friendship?")
print("Rate from 0 to 1, where 0 = not at all, 0.5 = okay, 1
= very much")

trust = float(input("How much do you trust this person? (0-
1): "))
fun = float(input("How much fun do you have together? (0-1):
"))
helpfulness = float(input("How helpful is this person? (0-1):
"))

# Weights - what matters most in friendship?
weight_trust = 0.5
weight_fun = 0.3
weight_help = 0.2

def friendship_analysis(trust, fun, help):
   inputs = [trust, fun, help]
   weights = [weight_trust, weight_fun, weight_help]

   total = 0
   for i in range(len(inputs)):
       total += inputs[i] * ____
   return total
```

Станция 2: Невронни мрежи - Алара се събужда

СРЕДНО НИВО (ЧАСТ II)

КОД:

```
score = friendship_analysis(trust, fun, helpfulness)
print("Friendship Score:", score)

if score > 0.7:
   print("💖 True friendship!")
elif score > 0.4:
   print("👍 Good acquaintance")
else:
print("🤔 Maybe not the best friendship")
```

Част от решението: weights[i]

Подсказка: Умножете всеки вход по съответното му тегло.

Станция 3: Дървета на решенията — детективът по училищния маршрут

Айлин е на автобусната спирка. Вали дъжд. Тя разговаря с Алара за пътя си до училище.

АЛАРА: *„Имам една идея за твоя училищен маршрут! Нека изградим дърво на решенията."*

Алара ѝ помага да направи избор - това се нарича дърво на решенията.

АЙЛИН: *„Страхотно! Сега винаги ще знам по кой път да вървя"* - казва Айлин радостно.

Станция 3: Дървета на решенията — детективът по училищния маршрут

ОСНОВНО ВИНО

КОД:

```
weather = input("How's the weather? (sun/rain): ")

def school_route(weather):
    if weather == "rain":
        return "☔ Take the umbrella!"
    else:
        return "🌞 ____"

print(school_route(weather))
```

Подсказка: Какво правите, когато не вали?

Част от решението: Go without an umbrella!

СРЕДНО НИВО

КОД:

```
weather = input("Weather? (rain/sun): ")
bus_coming = input("Is the bus coming? (yes/no): ")

def plan_route(weather, bus_coming):
    if weather == "rain" and bus_coming == "yes":
        return "🚌 Take the bus!"
    elif weather == "rain" and bus_coming == "no":
        return "☔ ____"
    else:
        return "🚶‍♀️Walk!"
print(plan_route(weather, bus_coming))
```

Подсказка : Ако на автобусът не идва, само чадърът ще помогне.

Част от решението : Вземете чадър!

Станция 4: Класификация, базирана на правила — Как Spotify може да разбере какво харесвате

Айлин и приятелката й Лена слушат музика.

ЛЕНА: *„ Как знае твоят часовник, че харесвам рок?“*

АЛАРА: *„Това е класификация, базирана на правила! Групирам песните по сходство.“*

ЛЕНА: *„Това е като магия!“* - възкликва Лена.

АЛАРА: *„Не е магия"* - *поправя я Алара* – *„Просто разпознаване на закономерности. С времето се уча да познавам музикалния ти вкус все по-добре."*

ОСНОВНО НИВО

КОД:

```
song = input("Describe your song: ... (e.g. guitar)")

def sort_music(song):
    if "guitar" in song:
       return "🎸 Rock"
    elif "____" in song:
        return "🎤 Pop"
    else:
        return "🎶 Other"

print(sort_music(song))
```

Подсказка: Танцувалните песни са обикновено поп песни.

Част от решението : danceable

Опростено Групиране: В това упражнение групираме слушателите въз основа на техните по-изразени музикални предпочитания. Реалните системи с изкуствен интелект, като K-Means, използват по-сложна математика, за да намират автоматично потребители със сходни вкусове.

Станция 5: Обучение чрез подкрепяне - Учене чрез награди

Айлин си играе с приложение за обучение на куче на таблета си.

АЙЛИН: *„Как да науча това виртуално куче какво да прави?"*

АЛАРА: *„Лесно е — чрез награди! Когато направи нещо правилно, дай му награда. По този начин то постепенно се учи кои действия си заслужават. Това се нарича обучение чрез подкрепяне!"*

АЙЛИН: *„Като в училище — добри оценки за верни отговори?"*

АЛАРА: *„Точно така! Но тук ти решаваш какво е „правилно". Кучето изпробва различни неща и запомня кое работи."*

АЙЛИН: *„Значи и ти се учиш като своето виртуално куче?"* - пита Айлин с усмивка.

АЛАРА: *„По някакъв начин - да"* - признава Алара. *„Наградите работят за всички, които се учат."*

ОСНОВНО НИВО (ЧАСТ I)

КОД:

```
import random

print("Train your virtual dog!")
print("Reward with a treat (1) or ignore (0)\n")

actions = ["Sit", "Lie Down", "Give Paw"]
probabilities = [0.3, 0.3, 0.4] # Starting values
disturb_chance = 0.2 # 20% chance the dog does something else

def normalize(probabilities):
  total = sum(probabilities)
  return [p / total for p in probabilities]
```

Станция 5: Обучение чрез подкрепяне - Учене чрез награди

ОСНОВНО НИВО (ЧАСТ II)

КОД:

```
for round in range(1, 6):
   print(f"\n Training session {round}/5")

   # Dog chooses an action based on its preferences
   action = random.choices(actions, weights=_________________)
[0] # Gap 1

   # Disturbance: Dog might do a different action
   if random.random() < disturb_chance:
      possible_actions = [a for a in actions if a != action]
      wrong_action = random.choice(possible_actions)
      print(f"The dog should do '{action}' but does instead:
🐶 {wrong_action}")
      action = wrong_action
   else:
      print(f"The dog does: 🐾 {action}")

   # User interaction with input check
   while True:
      user_input = input("Give a treat? (1=yes, 0=no):
").strip()
   if user_input in ("1", "0"):
      reward = int(user_input)
      break
      print("Please enter only 1 or 0!")

   index = actions.index(action)")
```

Подсказка: Ако автобусът не дойде, само чадърът помага.

Част от решението: Вземете чадъра!

Станция 5: Обучение чрез подкрепяне - Учене чрез награди

ОСНОВНО НИВО (ЧАСТ III)

КОД:

```
   # Learning logic
   if reward == 1:
      probabilities[index] += ______ # Gap 2
      print("✅ Dog is happy: 'That was good!'")
   else:
      probabilities[index] = max(0.1, probabilities[index] -
0.1)
      print("❌ Dog thinks: 'That wasn't great...'")

      # Normalize probabilities
      probabilities = normalize(probabilities)

      print("Current learning values:", [round(p, 2) for p in
probabilities])

# Result display
print("\n Training finished!")
for action, value in zip(actions, probabilities):
   print(f" {action}: {value:.2f}")

favorite_action =
actions[probabilities.index(max(probabilities))]
print(f"\n🐕 Dog's favorite action: {favorite_action}🦴")
```

Подсказка 1: Тук се съхраняват текущите вероятности, които кучето използва, за да избере действие.

Подсказка 2: С колко трябва да се увеличи вероятността, когато кучето бъде наградено?

Част от решението 1: probabilities

Част от решението 2: 0.2

Станция 5: Обучение чрез подкрепяне - Учене чрез награди

СРЕДНО НИВО (ЧАСТ I)

КОД:

```
print("Keep your plant alive!")
print("Actions: water, fertilize, do_nothing")

plant_state = {
   "water": 5, # 0-10
   "nutrients": 5,
   "health": 10
}

def evaluate_state():
    water, nutrients = plant_state["water"],
plant_state["nutrients"]
    if 3 <= water <= 7 and 3 <= nutrients <= 7:
      return ____ # Perfect!
    elif 1 <= water <= 9 and 1 <= nutrients <= 9:
      return _____ # Okay
    else:
      return -1 # Bad

for day in range(7):
    print(f"\n--- Day {day+1} ---")
    print(f"State: Water={plant_state['water']}/10,
Nutrients={plant_state['nutrients']}/10")

    action = input("What do you do?
(water/fertilize/nothing): ")
```

Станция 5: Обучение чрез подкрепяне - Учене чрез награди

СРЕДНО НИВО (ЧАСТ II)

КОД:

```
    # Perform action
    if action == "water":
      plant_state["water"] = min(10, plant_state["water"] +
3)
    elif action == "fertilize":
      plant_state["nutrients"] = min(10,
plant_state["nutrients"] + 3)

    # Time passes - state decreases
    plant_state["water"] = max(0, plant_state["water"] - 1)
    plant_state["nutrients"] = max(0,
plant_state["nutrients"] - 1)

    # Calculate reward
    reward = evaluate_state()
    print(f"Reward: {reward} points")

    if plant_state["water"] == 0 or plant_state["nutrients"]
== 0:
      print("💀 The plant has died!")
      break

if plant_state["water"] > 0 and plant_state["nutrients"] > 0:
  print("❤️🌱 Plant survived! Well done!")
```

Станция 6: Конволюционни невронни мрежи: Как изкуственият интелект открива ръбове в изображенията

Айлин разглежда снимката си.

АЛАРА: *„Виждам лини, яркост и контрасти - това са моят ръбове! Когато нещо се промени, знам: тук започва нов обект.“*

АЛАРА: *За да открия това, използвам един малък трик. Винаги сравнявам два съседни пиксела. Ако се различават, се появява ръб. Операторът != означава „не е равно" или „различно от".*

АЛАРА: *„Всяка снимка, която ми показваш, ми помага да „виждам" по-добре" - обяснява Алара. „Точно както и ти се учиш, аз описвам света все по-точно.“*

ОСНОВНО НИВО (ЧАСТ I)

КОД:

```
def count_edges(row):
    edges = 0
    for i in range(len(row)-1):
      if row[i] ___ row[i+1]:
         edges += 1
      return edges

print("Alara's 4-row image")

# Enter 4 rows from the user
image = []
for n in range(4):
  row = input(f"Enter row {n+1} (0 and 1, e.g., 100101): ")
  row = row[:6]
  image.append(row)
```

Станция 6: Конволюционни невронни мрежи: Как изкуственият интелект открива ръбове в изображенията

ОСНОВНО НИВО (ЧАСТ II)

КОД:

```
# Count edges per row
for i, row in enumerate(image):
    edges = count_edges(row)
    if edges > 0:
      print(f" Row {i+1}: {edges} edges")
    else:
      print(f"  Row {i+1}: no edges")
```

Част от решението: i=

Подсказка: Пример (0 → 1 или 1 → 0) е ръб.

СРЕДНО НИВО (ЧАСТ I)

КОД:

```
print("Create your own 4x4 image!")
size = 4
image = []

# Input rows
for i in range(size):
    row = input(f"Enter row {i+1} (only   and ■, e. g.
■  ■): ")
    row = list(row.ljust(size, "  "))[:size]
    image.append(row)

print("\n Your image:")
for row in image:
    print(" ".join(row))
```

Станция 7: Обработка на естествен език: Как компютрите разбират езика

Айлин пише съобщение на Лена: *„Хей, това беше страхотно!“*

АЛАРА: *„Мога да разпозная дали си щастлива, ядосана или неутрална - това се нарича анализ на настроението.*

АЛАРА: *„Твоите съобщения ми помагат да разбирам по-добре човешкия език"* — казва Алара с благодарност. *„Учим се заедно - ти ме учиш на изкуствен интелект, а аз се уча на човешко общуване."*

ОСНОВНО НИВО

КОД:

```
print("Alara reads your mood!")
text = input("Write something (e.g., … that was great / …
that was bad): ")

def understand(text):
if "great" in text:
   return "😊 You’re in a good mood!"
elif "bad" in text:
   return ____
else:
   return "😐 Neutral."

print(understand(text))
```

Подсказка: „bad“ = negative.

Част от решението: „😠 You’re angry!“

Станция 7: Обработка на естествен език: Как компютрите разбират езика

СРЕДНО НИВО

КОД:

```
print("Counting words like a language model!")
text = input("Write a sentence: ")

def count_words(text):
    words = text.split()
    return len(____)

print("Number of words:", count_words(text))
```

Hint: words

Подсказка: От колко е съставено едно изречение?

Станция 8: Системи за препоръки: Как YouTube знае какво искаш да гледаш

Айлин гледа видеоклипове на таблета си.

АЛАРА: *„Помня какво харесваш и предлагам подобно съдържание!"*

АЛАРА: *„Колкото повече ми показваш какво харесваш, толкова по-добри препоръки мога да ти давам"* - обяснява Алара. *„Това е партньорство."*

ОСНОВНО НИВО

КОД:

```
print("Alara's video recommendation")
video = input("What are you watching? (e.g., music video,
animal video): ")

def recommend(video):
    if "music" in video:
        return "🎶 Similar music videos!"
    else:
        return ____

print(recommend(video))
```

Подсказка: Ако не е музикален клип, предложи нещо ново.

Част от решението: "Try something new!"

Станция 8: Системи за препоръки: Как YouTube знае какво искаш да гледаш

СРЕДНО НИВО

КОД:

```
print("YOUR YOUTUBE ASSISTANT")
print("Find your perfect recommendation!")

name = input("\nWhat's your name? ")
print(f"Hello {name}! What do you like?")
favorite1 = input("Favorite Topic 1: ")
favorite2 = input("Favorite Topic 2: ")

user_data = {
   name: [favorite1, favorite2],
   "Lena": ["Sports", "Gaming"]
}

def recommendation(user_name):
   if favorite1 in ["Music", "music"]:
       return "🎵 YOUR TIP: Top 100 Charts Mix!"
   elif favorite1 in ["Sports", "sports"]:
       return "⚽ YOUR TIP: Best Goals Compilation!"
   else:
       return "🎬YOUR TIP: Trending Videos!"

print(f"\n{recommendation(____)}")
```

Станция 9: Прогноза за времето – Случайна гора (Random Forest)

Айлин посочва часовника си.

АЙЛИН: *„Как винаги знаеш дали можем да играем навън?"*

АЛАРА: *„Това е моят експертен екип! Три умни съзнания вътре в мен се съветват помежду си — точно като случайна гора!"*

АЙЛИН: *„Уау! Значи наистина имаш трима различни експерти в себе си?"*

АЛАРА: *„Да! Единият следи температурата, другият — облаците, а третият — вятъра. Заедно вземаме най-доброто решение!"*

АЛАРА: *„С всяка прогноза ставам все по-точна"* — казва Алара. *„Обратната ти връзка ми помага да усъвършенствам своите експерти."*

Какво е специалното при Случайните гори (Random Forest): В действителност случайната гора обучава много дървета, използвайки произволно избрани данни и характеристики. Нашето гласуване симулира този принцип на „колективен интелект".

Станция 9: Прогноза за времето – Случайна гора (Random Forest)

ОСНОВНО НИВО

КОД:

```
print("AYLIN'S WEATHER EXPERTS")
print("Three experts are consulting!")

temperature = float(input("Temperature: "))
clouds = input("Clouds (sunny/cloudy): ")

# Ask the experts
expert1 = "Sun" if temperature > 20 else "Rain"
expert2 = "Sun" if clouds == "sunny" else "Rain"

opinions = [expert1, expert2]
decision = max(set(opinions), key=opinions.____)

print(f"Forecast: {decision}")
```

Подсказка: Търсите кое мнение се среща най-често.
Част от решението: count

Станция 9: Прогноза за времето – Случайна гора (Random Forest)

СРЕДНО НИВО

КОД:

```
print("AYLIN'S WEATHER COMMITTEE")
print("Three AI experts decide!")

temperature = float(input("Temperature: "))
clouds = input("Clouds (sunny/cloudy/rainy): ")
wind = input("Wind (strong/weak): ")

# Three experts give their opinions
expert1 = "Sun" if temperature > 20 else "Rain"
expert2 = "Sun" if ____ == "sunny" else "Rain"
expert3 = "Sun" if wind == "weak" else "Rain"

opinions = [expert1, expert2, expert3]
decision = max(set(opinions), key=opinions.count)

print(f"Forecast: {decision}")
```

Част от решението: clouds

Подсказка: Кой променлива съдържа информация за облак?

Епилог — Вторият подарък

Няколко месеца по-късно.

Айлин седи на бюрото си, Алара свети на китката ѝ. В кухнята се чува тракане на съдове — леля Сема е на гости и приготвя бурек.

АЙЛИН: *„Знаеш ли, Алара"* — казва Айлин с усмивка — *„когато ми те подариха на рождения ми ден, нямах никаква представа за изкуствения интелект. А сега... изграждам първия си малък проект и разбирам как „мислиш"."*

Алара примигва гордо. *„Ти не само разбра изкуствения интелект — ти и мен ме научи да „мисля" по-човешки."*

В този момент леля Сема влиза с малък увит подарък.

ЛЕЛЯ СЕМА: *„Скъпа моя - казва тя нежно - видях колко много си научила през последните месеци. Баба би казала: „Вдъхна душа на часовника."*

Айлин отваря пакета. Вътре е обикновена тетрадка. На първата страница, написано с красив почерк:

„Изобретявай със сърце.
Програмирай с кураж.
Мисли като Алара –
но чувствай като Айлин."

Айлин се усмихва.

АЙЛИН: *„Това е най-хубавият подарък, лельо Сема."*

ЛЕЛЯ СЕМА: *„Не" - казва Сема, докосвайки я леко по гърдите. „Най-хубавият подарък вече е тук."*

КРАЙ

РЕФЛЕКТИВНИ ВЪПРОСИ

Отделете малко време, за да помислите върху наученото в този модул, и отговорете на следните въпроси.

- **Разбиране на изкуствения интелект:** Как се промени вашето разбиране за изкуствения интелект след завършването на този модул? Какво ви изненада най-много в начина, по който системи с изкуствен интелект като Алара „мислят"?

- **Невронни мрежи и вземане на решения:** В Станция 2 разгледахте как невронните мрежи използват тегла, за да вземат решения. Можете ли да посочите реална житейска ситуация, в която претегляте различни фактори, преди да направите избор? По какъв начин това прилича на начина, по който изкуственият интелект взема решения?

- **Етични съображения:** В хода на модула бяха повдигнати редица етични въпроси, свързани с изкуствения интелект. Защо е важно да се мисли за етиката при разработването и използването на изкуствен интелект? Посочете един пример за етичен проблем, свързан с изкуствения интелект в ежедневието.

- **Учене чрез взаимодействие:** Алара се учи от взаимодействията си с Айлин. Как според вас системите с изкуствен интелект в реалния живот - като системи за препоръки или гласови асистенти - се учат от взаимодействията с потребителите? Кои са ползите и рисковете от този вид учене?

- **Изкуственият интелект във вашия свят:** Къде се сблъсквате с изкуствен интелект в ежедневието си? Изберете един пример (например социални мрежи, стрийминг услуги или интелигентни устройства) и обяснете как той използва някоя от концепциите на изкуствения интелект, които научихте в този модул.

ТЕСТ (всеки въпрос има само един верен отговор)

1. Какво е Алара?
а) Най-добрата приятелка на Айлин
б) Обикновен будилник
в) Изкуствен интелект в смарт часовник
г) Лелята на Айлин

2. Коя концепция от програмирането е въведена в Станция 2 за вземане на решение въз основа на множество претеглени входни данни?
а) Дървета на решенията
б) Обучение с подкрепление
в) Невронни мрежи
г) Класификация, базирана на правила

3. Станция 3 „Детективът по училищния маршрут" е основно за ...

а) Музикални препоръки
б) Планиране на маршрут с помощта на дървета на решенията
в) Разпознаване на изображения
г) Обработка на език

4. Как Алара класифицира дадена песен като „поп" в Станция 4?
а) Ако съдържа китара
б) Ако е силна
в) Ако е подходяща за танци
г) Ако е на Тейлър Суифт

5. Какво симулира упражнението „Грижа за растения" в Станция 5?
а) Разпознаване на изображения
б) Обучение чрез подкрепяне с награди
в) Невронни мрежи
г) Дървета на решенията

ТЕСТ (всеки въпрос има само един верен отговор)

6. Как Алара открива ръбове в изображение в Станция 6?
а) Като сравнява съседни пиксели за разлики
б) Като брои белите пиксели
в) Като използва гласови команди
г) Като пита потребителя

7. В Станция 7 („Обработка на естествен език") настроението на текста се разпознава чрез...
а) Измерване на дължината на текста
б) Търсене на конкретни ключови думи като „страхотно" или „лошо"
в) Проверка на граматиката
г) Анализ на шрифта

8. Какво прави системата за препоръки в Станция 8?
а) Създава прогнози за времето
б) Предлага видеоклипове въз основа на интересите на потребителя
в) Разпознава почерк
г) Води робот през лабиринт

9. Как „Случайната гора" в Станция 9 взема решение?
а) Само един експерт решава
б) Чрез демократично гласуване на множество „дървета"
в) Чрез произволен избор
г) Чрез дълбоки невронни мрежи

10. Кой програмен език се използва във всички предизвикателства по програмиране?
а) JavaScript
б) Python
в) N1GL1B
г) C++

МОДУЛ 2: ИИ Ескейп стая с мисия за откриване на съкровище и решаване на загадка с калкулатор

ВЪВЕДЕНИЕ

Някога чудили ли сте се как вашият смартфон знае кои видеоклипове може да ви харесат? Целта на настоящия модул е да въведе основните концепции, обясняващи как изкуственият интелект (ИИ) „мисли“, взема решения и се учи от данни. Този модул предоставя теоретична основа, която свързва математиката, логиката и творчеството чрез ангажиращо, базирано на игри учебно преживяване.

До края на този модул ще придобиете следните знания и умения:

- **Математическа компетентност:** Ще придобиете по-голяма увереност при извършване на основни аритметични операции, ще разберете по-задълбочено понятия като средни стойности, проценти и съотношения, ще изградите умения за използване на калкулатор при решаване на задачи
- **Критично мислене и логично мислене:** Ще упражните своето логично мислене, докато разкодирате улики и взимате решения въз основа на дадените данни. Също, ще подобрите уменията си за решаване на проблеми се чрез прилагане на математика в практически сценарии и пъзели.
- **Вземане на решения:** Чрез решаване на задачи, свързани с класификация (напр. определяне кой обект отговаря на дадено описание), ще изградите знания и умения за взимане на информирани решения въз основа на числови данни и логика.

Продължителност на модула	**2 часа (1 час преподаване в клас + 1 час практически упражнения)**

ОБРАЗОВАТЕЛНИ МАТЕРИАЛИ

РАЗДЕЛ 2.1 КАКВО СЕ КРИЕ ЗАД ЕСКЕЙП СТАЯТА С ИЗКУСТВЕН ИНТЕЛЕКТ?

Изкуственият интелект може да бъде определен като способността на машините да изпълняват задачи, които обикновено изискват човешки интелект - като разсъждение, решаване на проблеми, учене или разпознаване на закономерности. Математиката е в основата на ИИ: вероятностите, статистиката и логиката помагат на компютрите да вземат решения въз основа на данни, а не на интуиция. Данните от своя страна действат като „гориво“ на ИИ: колкото повече качествени данни са налични, толкова по-добре системата открива закономерности и прави прогнози.
В този модул ще разберете как ИИ използва разсъждение, вероятности и учене, за да решава реални проблеми по творчески начини.

За да разберем тези идеи, ще разгледаме четири ключови принципа, представящи начина, по който ИИ „мисли":

- **Байесова мрежа** (също мрежа на Бейс) – помага да се предвиждат резултати при несигурна информация.
- **K-средни стойности** (K-Means Clustering) - групират сходни данни, за да открият закономерности и връзки.
- **Методи на Монте Карло** - използват случайни експерименти за оценка на вероятности и правене на прогнози.
- **Хебово обучение** - показва как връзките стават по-силни чрез повторение, точно като мозъка ни при усвояване на нови знания.

РАЗДЕЛ 2.2 РАЗБИРАНЕ НА ИИ ЧРЕЗ ПРОСТИ ИДЕИ

Чрез тези прости идеи ще разберете, че ИИ не се свежда само до програмиране. Той включва логическо разсъждение, решаване на проблеми и творчество. Тези принципи ще ви водят през всеки етап на ИИ ескейп стаята, където всяко предизвикателство ще разкрие как изкуственият интелект „се учи чрез действие“.

Тези четири идеи показват как ИИ съчетава логика, математика и творчество, за да се учи и взема умни решения. Точно това ще правите, докато решавате пъзелите в приключението с тази ИИ ескейп стая!

„ Целта на ИИ не е да замести хората, а да усили човешките способности.“
— Фей-Фей Ли (Станфордски университет, изследовател в областта на изкуствения интелект)

ИЗТОЧНИЦИ

Изкуствен интелект и вземане на решения

1. Russell, S., & Norvig, P. (2021). Artificial Intelligence: A Modern Approach (4th ed.). Pearson.

Вероятности, симулация и учене

2. Shalev-Shwartz, S., & Ben-David, S. (2014). Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press.

→ Основи на машинното обучение, клъстеризация и учене от данни.

3. Khan Academy. (n.d.). Statistics and probability. https://www.khanacademy.org/math/statistics-probability

→ Прости и достъпни обяснения на вероятности, средни стойности и симулации.

ЛИНКОВЕ КЪМ УЧЕБНИ ПЛАТФОРМИ

Ако искате да продължите да научавате за изкуствения интелект по забавен и достъпен начин, тези онлайн платформи са чудесно начало! Можете да играете, да експериментирате и да изследвате как компютрите се учат да разпознават изображения, звуци или закономерности — точно като в ИИ Ескейп стаята. Всяка връзка включва кратки дейности, които ви помагат да разберете как работи ИИ в реалния живот, докато продължавате да учите чрез упражнения и практика.

- Google Teachable Machine - интерактивна платформа за обучение на прости модели с изкуствен интелект с изображения, звуци или пози. https://teachablemachine.withgoogle.com/
- Khan Academy - Изкуствен интелект и машинно обучение - безплатни уроци, обясняващи основните концепции на ИИ чрез ежедневни примери. https://www.khanacademy.org/computing/computer-science
- AI за океаните (Code.org) - забавна интерактивна дейност, при която учениците обучават ИИ да почиства океана, като научават за данните и пристрастията. https://studio.code.org/s/oceans
- Machine Learning for Kids - платформа, която позволява на учениците да експериментират с модели на ИИ и да ги използват в проекти в Scratch. https://machinelearningforkids.co.uk/
- IBM SkillsBuild за ученици - безплатна онлайн платформа за обучение, предлагаща основни курсове по ИИ и работа с данни с значки и сертификати. https://skillsbuild.org/students
- Google AI Experiments - колекция от интерактивни инструменти, показващи как ИИ разпознава закономерности, звуци и движение. https://experiments.withgoogle.com/collection/ai

ПРАКТИЧЕСКО УПРАЖНЕНИЕ

Въведение: Историята

Добре дошли в ИИ Ескейп стаята!

Вие сте част от отбор млади изследователи, търсещи съкровище, заключено зад 4-цифрен цифров катинар. За да го отворите, трябва да преминете през 4 стаи, пазени от четирима пазачи, всеки от които представя принцип на изкуствения интелект:

1. Вероятности (Байесови мрежи)
2. Клъстеризация (K-средни стойности)
3. Симулация (Методи на Монте Карло)
4. Хебово обучение

Вашата мисия е да преодолеете предизвикателствата на всяка станция, да решите задачите и да получите комбинацията, която отваря катинара. Всяка станция съдържа обяснение с кратки упражнения и финално предизвикателство по програмиране в три нива на трудност, което ще ви даде една от цифрите, необходими за отваряне на катинара.

Станция 1: Байесови мрежи

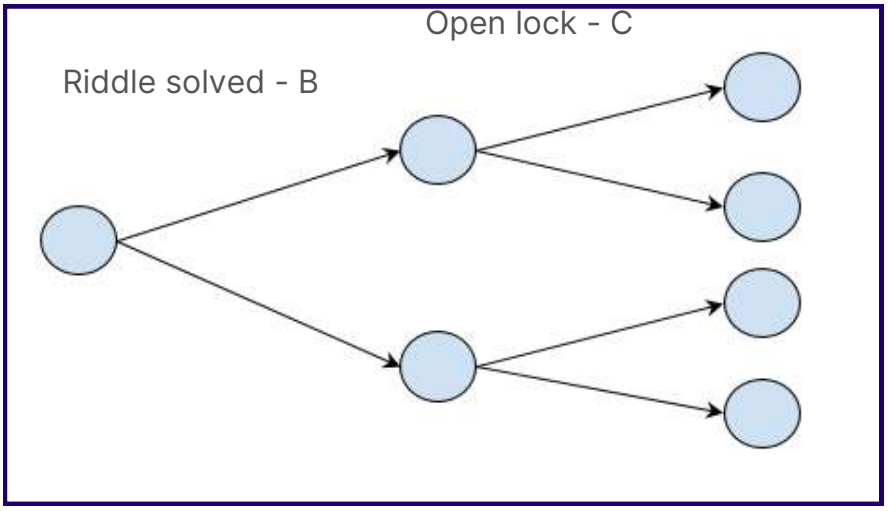
Байесовата мрежа е математически модел, който използва **вероятности**, за да взема решения или прогнози, когато не цялата информация е известна с точност, тоест когато е налице **несигурност**.

Мини-предизвикателство 1 - Станция 1: Лесната ключалка

Представете си, че искате да разберете дали можете да отворите числов катинар в ескейп стая.

- **Събитие A: Следи →** Има няколко скрити улики. Колкото повече намерите, толкова по-лесно ще отворите катинара.
- **Събитие B: Гатанка →** Има гатанка, която дава числата за комбинацията. Колкото повече улики намерите, толкова по-лесно е да я решите.
- **Събитие C: Катинар →** Дали ще можете да решите пъзела и да въведете числата в катинара зависи от броя на намерените улики.

Мини-предизвикателство 1 - Станция 1: Лесната ключалка



Фигура 2 Път на решения и вероятности: от намиране на улики до отваряне на катинара.
Източник: Моника Морено

Забележка: липсва половината от графиката, съответстваща на вероятността да не бъдат намерени улики.

Инструкции:
Ако намерите уликите (вероятност 0,8) и решите гатанката (вероятност 0,9, ако сте намерили уликите), каква е вероятността да отворите катинара (вероятност 0,95, ако решите гатанката)?

Пътят към решението се състои от редица вероятности, които трябва да се умножат.

Отговор: P(Отворен катинар) = 0,8 × 0,9 × 0,95

Въпрос: Какъв резултат бихте получили, ако съберете всички възможни пътища?

Отговор: 1, тъй като всеки път представя вероятността за настъпване на определен сценарий. Вероятността да настъпи един от всички възможни изходи е винаги 1.

Правило на Байес

До тук видяхме как можем да направим прогноза, но какво се случва, ако погледнем резултата и искаме да знаем вероятността, че е бил следван определен път?

Мини-предизвикателство 2 - Станция 1: Коридорът на условията

Намирате тайнствен коридор с поредна гатанка на стената: „От всички пъти, в които сте решили гатанката, каква е вероятността да сте намерили уликите предварително?“

С други думи, искаме да изчислим вероятността на С спрямо В **(което е различно от вероятността на В спрямо С)**. Трябва да използвате **Правилото на Байес:**

Инструкции:

Първо трябва да изчислим вероятността B=✓ и C=✓, като всички пътища завършват с B=✓ и всички тези завършват с C=✓. За да опростим нещата, ви даваме готовите изчисления: P(B=✓) = 0,76; P(C=✓) = 0,734.

Сега трябва само да приложим Правилото на Байес с формулата:

$$P(B=✓ \text{ if } C=✓) = [P(C=✓ \text{ if } B=✓) * P(B=✓)] / P(C=✓)$$

0.983

Отговор: $P(B=✓|C=✓) = [0.95 * 0.76] / 0.724 =$

Предизвикателство 1 - Пазителят на вероятностите

Стигнали сте до първата стая - Залата на Пазителя на вероятностите, мистериозна зала, осветена от цифрови факли. В центъра стои древен сандък, покрит с много скъпоценни камъни. Тогава пазителят - холограма на стар мъж - ви казва:

„ Само тези, които разбират **несигурността**, ще могат да открият дали сандъкът съдържа злато. Вероятността е ключът“

Вашата мисия е да изчислите **вероятността** сандъкът да съдържа злато, като се основавате само на това, което можете да видите, използвайки **опростен модел на Байесова мрежа.**

Сценарий: Каква е вероятността този тежък, блестящ сандък да съдържа злато? Първият десетичен знак на отговора ще бъде първата цифра на катинара.

- Знаете, че **30% от блестящите сандъци съдържат злато**, а **70% - камъни**.
- Сандъците, съдържащи злато, **са тежки в 80% от случаите.**
- Сандъците, съдържащи камъни, **са тежки в 40% от случаите.**

Предизвикателство 1 - Пазителят на вероятностите

ОСНОВНО НИВО: Нарисувайте Байесовата мрежа на хартия и използвайте калкулатора, за да получите отговора. **Улика:** нужни са ви само 2 събития (A и B) и Правилото на Байес.

СРЕДНО НИВО: Попълнете непълния код, за да решите задачата. Добавете само вероятностите и математическите символи за операциите.

ОСНОВНО НИВО

КОД:

```
p_gold =
p_stones =
p_heavy_if_gold =
p_heavy_if_stones =

# Calculate the probability that it is heavy
p_heavy = (p_gold “operation” p_heavy_if_gold) + (p_stones
"operation" p_heavy_if_stones)

# Calculate probability of gold given that it is heavy
p_gold_given_weighed = (p_gold "operation" p_weighed_if_gold)
"ooperation” p_weighed

print("Probability of gold given heavy:",
round(p_gold_given_weighed, 2))
```

ТРУДНО НИВО: Нарисувайте Байесовата мрежа и създайте свой собствен код за решаване на задачата.

Станция 2 — Клъстеризация по K-средни

K-средните стойности е алгоритъм, който **групира данните** в категории въз основа на **сходство.** За да разберем това, нека разгледаме един пример. Представете си, че сте измервали температурата почти всеки ден от годината и искате да знаете на кой сезон съответства всяка точка. Това е така наречения проблем на класификацията.

Мини-предизвикателство 1 - Станция 2: Кристалите в Коридор

По пътя към следващата стая намирате няколко кристала на пода. Всеки кристал има две числа: яркост и чистота. Групирайте тези кристали в две отделни групи според техните характеристики.

Инструкции:

- Нарисувайте две оси (x, y) и отбележете точките: [1,5], [4,1], [3,5; 0], [2,4], [0,5; 5], [0,5; 5], [5,2].
- Представете си, че искате две групи или **клъстера** (k=2). Къде бихте поставили начален център на всяка група (**центроид**)?
- Присвоете всеки кристал към най-близкия центроид.

Отговор: Group 1: [1,5], [2,4], [0.5, 5], [0.5, 5]. Group 2: [4,1], [3.5,0], [5, 2]

K-средните стойности а**втоматизират процеса на формиране на групи**, така че всяка да има „център“ (центроид), представляващ **средните стойности на характеристиките** на тази група.

„Как работи K-Means“

1. Изберете броя на клъстерите (k=n).
2. Поставете k центроида произволно в равнината, по един за всяка група.
3. Всяка точка в равнината се присвоява към най-близкия центроид.
4. Изчислете „центъра“ на всяка зона, за да преместите центроида.
5. Повторете процеса, докато точките спрат да сменят групите си.

Предизвикателство 2 - Алхимикът

След като преминете първата стая, ще стигнете до лаборатория, пълна с много колби с мистериозно оцветени течности. Тогава един **стар алхимик** ви казва:

„Млади авантюристе, няма да можеш да излезеш оттук, без да наредиш моите отвари. Всяко бурканче принадлежи към различен вид магия: **Лечение, Сила или Невидимост**. Ако ми помогнеш да ги групираш правилно, ще отворя вратата за теб.“

Всяко бурканче има д**ве химически характеристики**: pH и концентрация на магическа енергия. Вашата мисия е да използвате **K-средните стойности**, за да **групирате бурканчетата** и да преминете втория тест. Броят на клъстерите, които създадете, ще съответства на втората цифра на катинара.

ОСНОВНО НИВО: Използвайте следния код за групиране на бурканчетата, като попълните липсващите параметри.

ДОПЪЛНИТЕЛНО: вижте как се променя резултатът при различен брой групи.

ЗАБЕЛЕЖКА: Трябва да инсталирате библиотеката sklearn първо (pip install scikit-learn).

Предизвикателство 2 - Алхимикът

ОСНОВНО НИВО

КОД:

```
from sklearn.cluster import KMeans  # kmeans library
import numpy as np
import matplotlib.pyplot as plt

# Data: [pH, power]
potions = np.array([
    [2, 53], [6, 70], [3, 55], [3.1, 52], [10, 25],
    [8, 84], [9, 80], [7.5, 82],
    [10, 19], [10.5, 22], [4, 48], [9, 20],
])

# K-Means
kmeans = KMeans(n_clusters=2, random_state=0) # Algorithm
declaration
kmeans.fit(potions) # Executes kmeans to classify the data

# Visualization
plt.scatter(potions[:, 0], potions[:, 1], c=kmeans.labels_,
cmap="viridis", marker="o")
plt.xlabel("pH")
plt.ylabel("Magic Power")
plt.title("Potion classification with K-Means")
plt.show()

print("Classification of each jar:", kmeans.labels_)
```

Предизвикателство 2 - Алхимикът

СРЕДНО НИВО: Използвайте кода от лесното ниво и добавете ново бурканче (точка) с координати по ваш избор. Използвайте функцията kmeans.predict(point), за да получите класификацията. Добавете следния код в края, за да покажете резултата.

СРЕДНО НИВО

КОД:

```
plt.scatter(potions[:, 0], potions[:, 1], c=kmeans.labels_,
cmap="viridis", marker="o")
plt.scatter(
   new_jar[:, 0], new_jar[:, 1],
   c=[plt.cm.viridis(prediction[0] / (kmeans.n_clusters -
1))],
   marker="x", s=150, label="new jar"
)
plt.xlabel("pH")
plt.ylabel("Magic Power")
plt.title("Special Bottle Classification")
plt.legend()
plt.show()
```

ТРУДНО НИВО: Създайте кода от нулата.

Предизвикателство 2 - Алхимикът

HARD

CODE:

```
from sklearn.cluster import KMeans
import numpy as np
import matplotlib.pyplot as plt

# Step 1: Define the data (list of pairs [pH, energy] with
numpy)
potions = np.array([
    # add your data here
])

# Step 2: Create the KMeans model specifying the number of
clusters
kmeans = KMeans(n_clusters=2, random_state=0)

# Step 3: Train the model with your data (fit)
kmeans.fit(potions)

# Visualization
plt.scatter(potions[:, 0], potions[:, 1], c=kmeans.labels_,
cmap="viridis", marker="o")
plt.xlabel("pH")
plt.ylabel("Magic Power")
plt.title("Potion classification with K-Means")
plt.show()

# Step 4: Print the result
print("Classification of each jar:", kmeans.labels_)
```

Станция 3: Методи на Монте Карло

Методите на Монте Карло са начин за предвиждане на вероятността, с която ще настъпи дадено събитие, чрез симулиране на един и същ експеримент много пъти и броене колко пъти това събитие се получава като резултат.

Мини-предизвикателство 1 - Станция 3: Фонтанът на съдбата

След като вървите известно време, без да намерите следващата стая, започвате да се чувствате изгубени. Стигате до малка стая с вълшебен фонтан и решавате да си пожелаете да намерите пътя.

Надписът на камъка гласи:

„Ако хвърлите монета, има 50% шанс желанието ви да се сбъдне. Но какво ако опитате много пъти?“

Инструкции:

- Използвайте калкулатора, за да генерирате десет случайни числа. Четните числа представляват „ези“, а нечетните — „тура“.
- Пребройте колко пъти монетата пада на „ези“ и разделете на общия брой хвърляния (N=10). Резултатът е вероятността.

За да се прилагат методите на Монте Карло, в експериментите винаги трябва да е налице случайност. Без случайност няма да има вариация в резултатите, което означава, че няма да можем да оценим вероятностите.

Предизвикателство 3 - Пазителят на заровете на съдбата

След дълго вървене по каменен коридор, осветен от факли, намирате стая с **кръгла маса**, покрита със свитъци, цветни камъни и някои **данни**.

Зад масата седи **Пазителят на случайността** - скелет с качулка и дълбок глас, който взема 3 зара и ви казва:

„Пътнико, тази стая не се отваря със сила или изобретателност... Аз съм пазителят на **безкрайните игри на Монте Карло** и ти предлагам предизвикателство. Ще хвърля тези три зара и ще сумирам числата. Избери: **повече от 12 или по-малко от/равно на 12**. Ако познаеш, ще можеш да продължиш, но ако не - ще останеш тук завинаги."

За да решите кой вариант да изберете, прилагате знанията си по теория на вероятностите и използвате методите на Монте Карло, за да определите най-добрия вариант. Първият десетичен знак на по-голямата вероятност ще ви даде третата цифра на катинара.

ОСНОВНО НИВО:
Следвайте стъпките по-долу, за да създадете код, с който да изчислите вероятността за успех при всеки от случаите за N=10 000 експеримента.

1. Създайте 2 променливи, за да броите колко пъти се появява всеки резултат.
   Улика: трябва да инициализирате променливите с 0.
2. Генерирайте 3 случайни числа и ги съберете.
   Улика: трябва да използвате функцията random.random().
3. Проверете дали сумата е >12 или ≤12 и добавете 1 към съответния брояч.
4. Повторете процеса 10 000 пъти.
   Улика: трябва да поставите кода в цикъл „for“.
5. Разделете броячите на N=10 000, за да получите вероятността, и изберете по-вероятния вариант.

Предизвикателство 3 - Пазителят на заровете на съдбата

СРЕДНО НИВО:
Напишете код за симулиране на експеримента за N=10 000 итерации (можете да използвате стъпките от основното ниво, ако е необходимо), но трябва също да изчислите вероятността за всеки възможен изход (от 3 до 18) при хвърляне на заровете.

Улика: трябва да използвате речник за броячите. Изпълнете всички експерименти и сумирайте към брояча. Накрая, след всички експерименти, разделете всички стойности на N.

ТРУДНО НИВО:
Създайте програма от нулата, която за произволен брой зарчета, повторения и прагова стойност изчислява вероятността за всеки възможен изход.

Улика: трябва да създадете функция с параметри N, num_dices и threshold, която връща речник с възможните резултати като ключ и вероятностите като стойности.

ДОПЪЛНИТЕЛНО (експертно ниво):
Добавете визуализация на разпределението на вероятностите, която позволява разглеждане в хистограма (plt.bar) на вероятностите за всяка стойност.

Станция 4 : Хебово обучение

Хебовото обучение е правило за учене в невронни мрежи, основано на принципа: „Неврони, които се активират заедно, се свързват заедно.“ С други думи:

- Ако два неврона се активират едновременно, връзката между тях **се укрепва**.
- Ако единият се активира, а другият - не, връзката **отслабва**.

По този начин, с течение на времето, невроните, получаващи **сходна или свързана информация, изграждат по-силни връзки**, а **нерелевантните взаимоотношения** отслабват. Така мрежата автоматично се учи да открива закономерности в даннитер точно като нашия мозък.

Мини-предизвикателство 1 - Библиотеката на идеите

По пътя към финалната стая минавате през тъмен коридор, където има две статуи с блестящи символи. Всеки път, когато кажете „светлина“, двете статуи светват заедно. Но ако кажете „сянка“, само едната свети.

Според правилото на Хеб какво ще се случи с връзката между тези две статуи, ако повторите „светлина“ много пъти?

а) Ще отслабне
б) Ще остане същата
в) Ще се укрепи

Мини-предизвикателство 1 - Библиотеката на идеите

За изчисляване на теглото на връзката между два неврона след тяхното активиране се използва формулата на Хеб:

$$w_{final} = w_{initial} + (n_{together} \times \Delta w_{+}) + (n_{alone} \times \Delta w_{-})$$

Където:

- $w_{initial}$ = начално тегло на връзката
- $n_{together}$ = брой пъти, в които и двата неврона се активират заедно
- Δw_{+} = увеличение на теглото при съвместно активиране
- n_{Alone} = брой пъти, в които само един неврон се активира
- Δw_{-} = намаление на теглото при активиране само на единия

Мини-предизвикателство 2 - Тегла на връзките

За да осветите правилно коридора, трябва да изчислите активирането между двете статуи (или неврони). За целта трябва да използвате формулата на Хеб.

Инструкции:

- Началното тегло на връзката е 0,2.
- Всеки път, когато и двете светват заедно → **+0,1**
- Всеки път, когато само едната свети → **–0,05**
- И двете светлини светват заедно **3 пъти**, но светлина А свети сама **2 пъти.**

Отговор: $w_{final} = 0.2 + (3 \times 0.1) + (2 \times 0.05) = 0.6$

Предизвикателство 4 – Стенописът с неврони

Много сте близо до края, но все още трябва да преодолеете 1 последно предизвикателство. В последната стая намирате **стенопис с 3 взаимосвързани вълшебни светлини (A, B, C)**, свързани помежду си с нишки, представляващи **тегла на връзките**.

Мистериозен глас ви казва:

„За да отворите финалната врата, трябва да разберете кои връзки са се укрепили. Изчислете крайните тегла след няколко активирания и разкрийте тайния модел.“

Всяка връзка започва с **тегло 0**. Ако два неврона се активират заедно, връзката се увеличава с 0,2, а ако само единият се активира, теглото намалява с –0,07. Има три връзки: AB, AC и BC. Последната цифра на катинара съответства на последния десетичен знак на теглото на връзката AC след рунд 4.

Рунд	Активирани неврони
1	A,B
2	B,C
3	A
4	A, C
5	A, B, C

ОСНОВНО НИВО:
Направете изчисленията само за връзката AB, с таблица, показваща резултата от теглото на връзката за всеки рунд.

СРЕДНО НИВО:
Изчислете крайните тегла на всички връзки за всички рундове.

ТРУДНО НИВО:
Изчислете крайните тегла на всички връзки за всички рундове, като приемете, че ако и трите неврона се активират едновременно, всички връзки получават бонус от +0,2. Идентифицирайте най-силната връзка от последния рунд.

РЕФЛЕКТИВНИ ВЪПРОСИ

Отделете малко време, за да помислите върху наученото в този модул, и отговорете на следните въпросите:

ИИ, математика и вземане на решения:
В този модул използвахте математика, вероятности и логика, за да решите предизвикателствата в стаята за бягство с ИИ. Как тази дейност ви помогна да разберете как ИИ взема решения? Кое предизвикателство ви помогна да разберете това най-добре?

Учене чрез вероятности и симулация:
Разгледахте различни методи на ИИ, като Байесови мрежи, K-средни стойности, методи на Монте Карло и Хебово обучение. Можете ли да обясните една от тези идеи с ваши думи и да дадете пример как е използвана в ИИ ескейп стаята?

Учене чрез повторение и опит:
Някои предизвикателства изискваха повтаряне на изчисления, симулации или опити, за да се достигне правилният отговор. По какъв начин това прилича на начина, по който системите с ИИ се учат от данни и опит? Можете ли да посочите реален пример, в който ИИ се учи по този начин?

ТЕСТ (всеки въпрос има само един верен отговор)

1. За какво главно помагат Байесовите мрежи?
a)Групиране на сходни точки от данни
b)Вземане на решения при несигурност
c)Ускоряване на изчисленията чрез случайна извадка
d)Укрепване на невронните връзки

2. Ако 30% от блестящите сандъци съдържат злато, а 80% от сандъците с злато са тежки, кое правило използваме, за да намерим вероятността тежък сандък да съдържа злато?
a) Симулация на Монте Карло
b) Хебово обучение
c) Правилото на Байес
d) Алгоритъм K-Means

3. Какво представлява „k“ в K-средните стойности?
a) Броят на точките от данни
b) Броят на клъстерите
c) Броят на измеренията
d) Броят на вероятностите

4. Коя стъпка следва веднага след присвояване на точките към най-близкия центроид?
a) Отново произволно поставяне на центроидите
b) Изчисляване на новите центрове на всеки клъстер
c) Чертане на Байесова мрежа
d) Оценяване на вероятности с зарчета

ТЕСТ (всеки въпрос има само един верен отговор)

5. Защо методите на Монте Карло са полезни при оценката на вероятности?

a) Елиминират случайността напълно
b) Симулират множество експерименти с елемент на случайност
c) Групират данните в категории
d) Укрепват мрежовите връзки

6. В предизвикателството „Пазителят на заровете на съдбата“ какво се изчислява?

a) Кой сандък съдържа злато
b) Към кой клъстер принадлежи отварата
c) Вероятността сумата от зарчетата да е по-голяма или по-малка от 12
d) Крайното тегло на невронната връзка

7. Съгласно принципа „неврони, активирани заедно, се свързват заедно“, какво се случва, когато два неврона се активират едновременно много пъти?

a) Връзката отслабва
b) Връзката изчезва
c) Връзката се укрепва
d) Нищо не се променя

ТЕСТ (всеки въпрос има само един верен отговор)

8. Ако два неврона се активират заедно 3 пъти (+0,1 всеки) и само единият — 2 пъти (–0,05 всеки), при начално тегло = 0,2, какво е крайното тегло?

a) 0,35
b) 0,60
c) 0,10
d) 0,45

9. Кое от следните твърдения най-добре разграничава Байесовите мрежи от методите на Монте Карло?

a) Байесовите мрежи използват правила за вероятности, Монте Карло използва симулации
b) И двата са един и същ метод с различни имена
c) Байесовите мрежи са случайни, Монте Карло е детерминистичен
d) Монте Карло укрепва връзките като Хебовото обучение

10. В изкуствените невронни мрежи Хебовото обучение главно помага с:

a) Клъстеризиране на данните в групи
b) Укрепване на връзките между съвместно активираните неврони
c) Предвиждане на резултати при несигурност
d) Изпълнение на хиляди случайни симулации

МОДУЛ 3: Scratch среща изкуствения интелект

ВЪВЕДЕНИЕ

Целта на настоящия модул е да ви запознае с програмния език Scratch и да покаже как Scratch може да бъде лесен и ефективен инструмент за усвояване на концепции от програмирането.

До края на този модул ще придобиете следните знания и умения:

- **Разбиране как** предоставя достъпна платформа за усвояване на концепции от програмирането.
- **Владене на основни умения за блоково програмиране** и компютърно мислене.
- **Интегриране на разширения за изкуствен интелект** в творчески проекти в Scratch.

Продължителност на модула	2 часа (1 час преподаване в клас + 1 час практически упражнения)

ОБРАЗОВАТЕЛНИ МАТЕРИАЛИ

Scratch е мощен програмен език с визуална основа - за разлика от традиционните езици за програмиране, Scratch не разчита на синтаксис или код на базата на редове; той работи чрез свързване на блокове за изграждане на логика. Използването на Scratch като въведение в програмирането има много предимства. В този модул ще научите историята на Scratch и как да програмирате с помощта на онлайн платформата. В края на модула ще научите как да добавяте разширения за ИИ към приложенията в Scratch.

РАЗДЕЛ 3.1 ВЪВЕДЕНИЕ В SCRATCH

Scratch е визуален програмен език и онлайн общност, разработени от MIT Media Lab. За разлика от традиционните езици за програмиране, които изискват усвояване и въвеждане на сложен код, Scratch използва блокове от код, които се наместват един в друг като части от пъзел. Тази концепция прави програмирането достъпно за всеки, независимо от нивото на опит и познания по програмиране.Scratch не е опростена версия на "истинско" програмиране; той е легитимен инструмент, използван от милиони хора по целия свят, включително студенти от университети. Платформата Scratch има над 130 милиона регистрирани потребители и е домакин на повече от 160 милиона споделени проекти, което я прави едно от най-големите кодиращи общества в света.

ЗАЩО SCRATCH Е ЧУДЕСЕН ИНСТРУМЕНТ ЗА УЧЕНЕ ПРОГРАМИРАНЕ

Основното предимство на ученето чрез Scratch е, че премахва много от бариерите, които традиционно правят програмирането трудно за начинаещи.

Например Scratch се различава от традиционните езици за програмиране по следните начини:

- **Без синтактични грешки:** Блоковете в Scratch се свързват само по начини, имащи логически смисъл, което предотвратява досадни синтактични грешки.
- **Незабавна визуална обратна връзка:** Scratch ви позволява да виждате резултатите от кода си незабавно.
- **Нисък праг, висок таван:** Scratch е лесен за усвояване, но може да се използва и за изграждане на сложни проекти.
- **Творческа свобода:** Може да създавате игри, анимации, интерактивни истории, симулации и много повече..
- **Съвместна общност:** Scratch позволява да споделяте проекти, да надграждате и използвате чужди проекти и да се учете от другите.
- **Многоплатформен:** Scratch работи във всички водещи уеб браузъри на всяко устройство, включително таблети и смартфони, което го прави изключително леснодостъпен.

ПРИЛОЖЕНИЯ НА SCRATCH В РЕАЛНИЯ СВЯТ

Въпреки че Scratch е блоково базиран език без синтаксис, той преподава основни концепции от програмирането, които могат пряко да бъдат приложени и в други езици като Python, JavaScript и Java.

Scratch обработва концепциите от програмирането по следния начин:

Концепция	В Scratch	Приложение в реалния свят
Последователности	Блокове, подредени по ред	Стъпка по стъпка инструкции в произволна програма
Цикли	Блокове за повторение	Автоматизиране на повтарящи се задачи
Условия	Блокове "ако-тогава"	Логика за вземане на решения в приложения
Променливи	Блокове за съхранение на данни	Съхранение и управление на информация
Събития	Блокове, задействащи действия	Управление на потребителско
Паралелизъм	Множество скриптове, изпълнявани едновременно	Многонишкови приложения

РАЗДЕЛ 3.2 ОСНОВНИ УМЕНИЯ ЗА БЛОКОВО ПРОГРАМИРАНЕ

КАК ДА СЪЗДАДЕТЕ АКАУНТ В SCRATCH

Създаването на акаунт е просто и безплатно. Започнете, като посетите уебсайта на Scratch: https://scratch.mit.edu/. След това създайте акаунт, като изберете „Присъединете се към Scratch“ от главната навигация, и следвайте инструкциите за регистрация.

НАВИГИРАНЕ В ИНТЕРФЕЙСА НА SCRATCH

Интерфейсът на Scratch определя как изглежда програмата на екрана. Има седем различни елемента на Scratch, които трябва да познавате, за да можете да започнете да програмирате в платформата:

- Интерфейс
- Спрайтове
- Сцена
- Блокове
- Костюми
- Фонове
- Звуци

Интерфейсът на Scratch е разделен на три основни зони: Сцената (горе вдясно), където се показват проектите; Зоната за код (в центъра на екрана), където можете да провлачвате и сглобявате блокове за създаване на скриптове; и Зоната за спрайтове (долу вдясно), където можете да добавяте и управлявате персонажи (спрайтове) и фонове.

В лявата страна на интерфейса се намира Палитрата с блокове. Това е кутия с инструменти с различни категории блокове за визуално изграждане на програми. За обобщение, основните елементи на интерфейса са:

- **Зона за спрайтове (долу вдясно):** Показва всички спрайтове (персонажи и обекти) в проекта ви. Кликнете върху спрайт, за да редактирате кода му.
- **Зона за код (в центъра):** Работното пространство, където провлачвате и свързвате блокове за създаване на скриптове за избрания спрайт.
- **Сцена (горе вдясно):** Зоната за визуален изход, където спрайтовете ви се представят и взаимодействат. Това е, което вижда вашата аудитория.

ЗАПОЗНАВАНЕ С КАТЕГОРИИТЕ БЛОКОВЕ

Scratch организира блоковете с код в девет категории, обозначени с цветове. Всяка категория изпълнява конкретна функция в програмата ви; категориите са следните:

1. **Синьо - Движение:** Категорията за движение позволява преместването на спрайта ви по екрана. Ако искате спрайтът да се движи напред, можете да използвате блока "Промени X с 10".
2. **Лилаво - Изглед:** Категорията за изглед се фокусира върху промяна на облика на спрайтовете и фоновете. Можете да използвате блока "Смени костюм" за промяна на облика на спрайт.
3. **Маджента - Звук:** Категорията за звук се фокусира върху възпроизвеждането на звуци от спрайтовете. Можете да използвате блока "Пусни звук докато свърши" за възпроизвеждане на звук.

4. **Жълто - Събития:** Всеки проект в Scratch има нужда от събитие; то задейства действия при взаимодействие.
5. **Светлооранжево - Управление:** Категорията за управление определя колко пъти спрайтът трябва да изпълни нещо или колко дълго трябва да остане неподвижен. Ако добавите блок "Изчакай 1 секунда", програмата ще изчака 1 секунда, преди да продължи.
6. **Тюркоазено - Сензори:** Тази категория проверява дали даден обект докосва друг, за да задейства действие.
7. **Зелено - Оператори:** Тези управления могат да се използват за извършване на математически и логически операции, като събиране, изваждане и конкатенация (съединяване) на низове.
8. **Тъмнооранжево - Променливи:** Променливите се използват за съхранение и управление на информация като резултати или точки за здраве.
9. **Червено - Моите блокове:** В тази категория можете да създавате свои собствени персонализирани блокове.

РАЗДЕЛ 3.3 КАК ДА СЪЗДАДЕТЕ ПРОЕКТ В SCRATCH

За да създадете проект в Scratch, започнете с добавяне на персонаж (спрайт) и фон от библиотеката.

Добавяне на нов спрайт

В долния десен ъгъл на редактора на Scratch ще намерите контролите за спрайтове (наречени Sprite Pane или Панел на спрайтовете), които включват следните инструменти и свойства:

- Избор на спрайт: Разгледайте вградената библиотека от спрайтове (животни, хора, обекти, фантастични персонажи и др.).
- Рисуване: Създайте свой собствен спрайт с инструментите за рисуване.
- Изненада: Добавете произволен спрайт за вдъхновение.
- Качване на спрайт: Импортирайте файл с изображение от компютъра си.

Промяна на фона

Фонът задава сцената за вашия проект. Кликнете върху иконата за фон (долу вдясно от сцената), за да:

- Изберете от готови фонове (на открито, на закрито, космос, абстрактни и др.)
- Нарисувате свой собствен с редактора на фонове.
- Качите изображение от компютъра си.

Можете да имате множество фонове и да превключвате между тях програмно с блокове "смени фон на [наименование]". Това е полезно за създаване на многосценни игри или истории.

ПРЕМЕСТВАНЕ НА СПРАЙТ И АКТИВИРАНЕ НА ДЕЙСТВИЕ

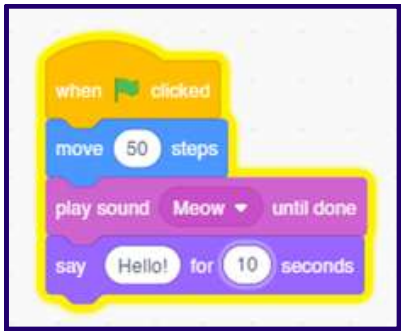
- **Стъпка 1: Добавете блок за събитие:** Кликнете върху категорията Събития (в жълто). Провлачете блока "когато е кликнато зеленото знаме" в зоната за код. Този блок стартира програмата ви, когато кликнете върху зеленото знаме над сцената.

- **Стъпка 2: Добавете движение:** Кликнете върху категорията Движение (в синьо). Провлачете блока "премести 10 стъпки" и го наместете под блока за събитие. Сменете числото от 10 на 50, като кликнете върху него.

- **Стъпка 3: Добавете звуков ефект:** Кликнете върху категорията Звук (в маджента). Провлачете блока "пусни звук [мяу] докато свърши" и го свържете под блока за движение.

- **Стъпка 4: Накарайте спрайта да говори:** Кликнете върху категорията Изглед (в лилаво). Провлачете блока "кажи [Здравей!] за 2 секунди" и го добавете към скрипта си. Можете да смените "Здравей!" с произволно съобщение.

- **Стъпка 5: Тествайте програмата:** Кликнете върху зеленото знаме над сцената. Спрайтът ви трябва да се премести 50 стъпки надясно, да издаде звук и да покаже вашето съобщение.



Фигура 3 Блокова логика на Scratch за движение и звуково взаимодействие.
Източник: Scratch от MIT Media Lab

РАЗДЕЛ 3.4 ВЪВЕДЕНИЕ В РАЗШИРЕНИЯТА ЗА ИИ В SCRATCH

КАКВО Е РАЗШИРЕНИЕ В SCRATCH?

Разширенията са допълнителни модули, които увеличават възможностите на Scratch отвъд стандартните категории блокове. Мислете за разширенията като за специализирани набори от инструменти, даващи на спрайтовете ви нови възможности - от произнасяне на текст на глас до разпознаване на изображения чрез изкуствен интелект.

Scratch 3.0 включва софтуерни разширения, добавящи изчислителни функции, и хардуерни разширения, свързващи се с физически устройства като роботи и електронни комплекти.

В този раздел се фокусираме върху управлявани от ИИ софтуерни разширения, които включват машинно обучение и обработка на естествен език в проектите ви.

ДОБАВЯНЕ НА РАЗШИРЕНИЕ (EXTENSION) В SCRATCH

Добавянето на разширение е просто:

1. Намерете бутона със символ "+" в долния ляв ъгъл на зоната с блокове.
2. Кликнете върху символа, за да отворите Библиотеката с разширения.
3. Разгледайте наличните разширения и кликнете с мишката върху едно, за да го добавите към проекта си.
4. Нови блокове ще се появят незабавно в палитрата ви, готови за използване.

Повечето разширения с ИИ изискват активна интернет връзка, тъй като обработката на изкуствения интелект се извършва на облачни сървъри, а не на вашия компютър. Уверете се, че сте свързани с интернет, преди да използвате тези разширения. Освен това внимавайте за поверителността на проектите ви и избягвайте качването на лична или чувствителна информация при обучение на модели за машинно обучение.

ЗАЩО И КАК СЕ ИЗПОЛЗВА ИИ В ТВОРЧЕСКИ ПРОЕКТИ

Изкуственият интелект дава възможност на проектите ви в Scratch да правят неща, които биха били почти невъзможни само с традиционното програмиране:

Възможности на ИИ	В Scratch	Паралел в реалния свят
Разпознаване на изображения	Игра, която идентифицира обекти, нарисувани или	Google Photos автоматично маркира хора и места
Класификация на текста	Чатбот, който разбира дали съобщенията са въпроси, твърдения или команди	Спам филтри за имейли, категоризиращи съобщения
Синтез на реч	Персонажи, четящи истории на глас или разказващи игрови	Навигационни системи, предоставящи гласови указания
Анализ на настроението	Виртуален домашен любимец, реагиращ по различен начин на	Системи за анализ на отзиви в социалните мрежи
Разпознаване на закономерности	Музикално приложение, разпознаващо дали звуците са барабани,	Shazam, идентифициращ песни по кратки аудио откъси

ИИ разширява творческите възможности, правейки проектите адаптивни (реагиращи по различен начин в зависимост от входа), интелигентни (вземащи решения въз основа на научени закономерности) и интерактивни (разбиращи множество форми на вход, като изображения, текст и звук).

ПРЕГЛЕД НА РАЗШИРЕНИЯТА ЗА ИИ В SCRATCH

ОФИЦИАЛНИ РАЗШИРЕНИЯ ЗА ИИ В SCRATCH

- **Преобразуване на текст в реч (Text-to-Speech):** Преобразува написан текст в изговорени думи с множество гласове и езици.
- **Превод (Translate):** Превежда текст между езици с помощта на машинен превод.
- **Видеосензор (Video Sensing):** Открива движение и присъствие чрез уеб камера (базово компютърно зрение).

РАЗШИРЕНИЯ ЗА ИИ ОТ ТРЕТИ СТРАНИ

- **Машинно обучение за деца (Machine Learning for Kids):** Обучаване на персонализирани модели за разпознаване на изображения, текст, числа и звуци.
- **Teachable Machine:** Инструментът на Google за бързо обучение на модели, който може да се интегрира със Scratch.
- **Разпознаване на лица (Face Sensing):** Открива лица и черти на лицето чрез уеб камера.
- **Разпознаване на реч (Speech Recognition):** Преобразува изговорени думи в текст (гласово управление).

РАЗДЕЛ 3.5 РАЗШИРЕНИЕ ЗА ИИ: МАШИННО ОБУЧЕНИЕ ЗА ДЕЦА

Машинното обучение за деца - Machine Learning for Kids (ML4K) е образователна платформа, разработена от IBM, която прави машинното обучение достъпно за учащи от всички възрасти. Тя предоставя удобен интерфейс за обучение на модели с ИИ и безпроблемно се интегрира с Scratch чрез персонализирани блокове.

С ML4K можете да обучавате модели да:

- **Разпознават и класифицират изображения** (напр. разграничаване на котки от кучета или идентифициране на нарисувани ръчно форми).
- **Разбират и категоризират текст** (напр. разграничаване дали съобщенията са комплименти или обиди).
- **Идентифицират закономерности в числата** (напр. предвиждане на резултати въз основа на числов вход).
- **Разпознават звуци** (напр. разграничаване на музикални ноти или гласови команди).

КАКВО Е МАШИННО ОБУЧЕНИЕ?

Машинното обучение е подобласт на изкуствения интелект, при която компютрите се учат да изпълняват задачи чрез анализиране на примери, а не чрез следване на предварително зададени правила. Например, вместо да програмирате описание за всяка порода куче, вие показвате на компютъра хиляди снимки на кучета и той се учи да разпознава характеристиките, определящи породата. Точно така работят услуги като Google Photos, автоматично маркиращи вашите снимки.

СТЪПКА ПО СТЪПКА: ОБУЧЕНИЕ И ИЗПОЛЗВАНЕ НА МОДЕЛ

СТЪПКА 1: ВЛЕЗТЕ В MACHINE LEARNING FOR KIDS

Отворете уеб браузъра си и отидете на
https://machinelearningforkids.co.uk/

Кликнете върху "Започнете" и за този пример изберете да създадете акаунт без регистрация.

СТЪПКА 2: СЪЗДАЙТЕ НОВ ПРОЕКТ

След като влезете, кликнете върху "Проекти", след това "Добави нов проект". Ще бъдете подканени да:

- **Дадете име на проекта си:** изберете описателно наименование (напр. "Детектор на емоции" или "Сортировач за рециклиране").
- **Изберете тип разпознаване:** изберете от Текст, Числа, Изображения или Звуци.

За целите на това упражнение, изберете "Текст", за да изградите класификатор на настроението.

СТЪПКА 3: ДЕФИНИРАЙТЕ ЕТИКЕТИТЕ (КАТЕГОРИИТЕ)

Моделите за машинно обучение (ML) класифицират входните данни в категории, наречени "етикети". За класификатор на настроението създайте два етикета:

- **Положително:** Радостни, ентусиазирани или добронамерени съобщения.
- **Отрицателно:** Тъжни, ядосани или злонамерени съобщения.

Кликнете върху "Обучи", след това "Добави нов етикет" за всяка категория.

СТЪПКА 4: ПОСОЧЕТЕ ПРИМЕРИ ЗА ОБУЧЕНИЕ

Моделът се учи чрез изучаване на примери. За всеки етикет посочете поне 5-10 примерни фрази:

Положителни примери	Отрицателни примери
Това е невероятно!	Това не ми харесва.
Справяш се страхотно!	Това е ужасно.
Обичам да уча нови неща.	Това ме ядосва.

Съвет: Колкото повече примери предоставите, толкова по-добре ще се представи моделът ви. Стремете се към поне десет примера за всеки етикет и осигурете разнообразие в начина на изразяване.

СТЪПКА 5: ОБУЧЕТЕ ВАШИЯ МОДЕЛ

След като добавите достатъчно примери, върнете се в проекта и изберете "Учете и тествайте", след което кликнете върху бутона "Обучи нов модел за машинно обучение". Системата ще анализира примерите ви и ще създаде модел, способен да класифицира нов текст. Обучението обикновено отнема 1-3 минути. Ще видите индикатор за напредъка. При завършване на обучението, ще получите съобщение за потвърждение.

СТЪПКА 6: ТЕСТВАЙТЕ МОДЕЛА СИ

Преди да използвате модела в Scratch го тествайте в ML4K. Въведете фраза, която не сте включили в тренировъчните данни (напр. "Днес съм много щастлив!") и кликнете върху "тест". Моделът ще предвиди кой етикет най-добре отговаря на вашия вход и ще покаже процент на увереност.

Ако моделът допуска грешки, върнете се в раздела за обучение и добавете още примери, за да подобрите точността.

СТЪПКА 7: СВЪРЖЕТЕ СЕ СЪС SCRATCH

ML4K предоставя специална интеграция със Scratch:

1. Кликнете върху „Създаване“ (Make) в навигацията на проекта.
2. Изберете „Scratch 3“
3. Това отваря модифициран редактор на Scratch с вашият модел на машинно обучение (ML), предварително зареден като персонализирани блокове.

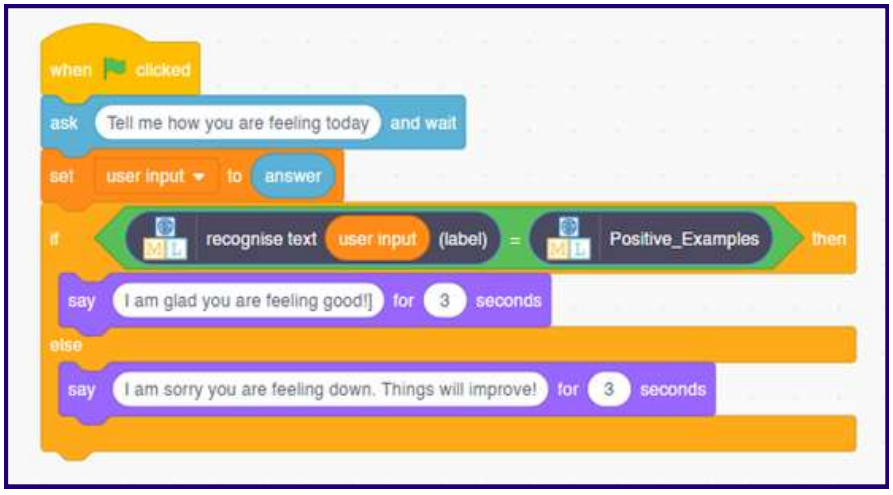
Ще видите нови блокове, специфични за вашия проект, като например:

- разпознай текст [ ] (етикет)
- разпознай текст [ ] (увереност)

СТЪПКА 8: ИЗПОЛЗВАЙТЕ ML БЛОКОВЕТЕ ВЪВ ВАШИЯ SCRATCH ПРОЕКТ

Създайте лесен проект в Scratch, използващ вашия обучен модел: Започнете с добавяне на спрайт (опитайте спрайта "Робот"), след това създайте променлива, наречена "потребителски вход", и изградете следния скрипт:

когато е кликнато зеленото знаме попитай [Кажи ми как се чувстваш днес] и изчакай, задай [потребителски вход] на (отговор) ако <разпознай текст (потребителски вход) (етикет) = [Положително]> тогава кажи [Радвам се, че се чувстваш добре!] за 3 секунди иначе кажи [Съжалявам, че не се чувстваш добре. Нещата ще се подобрят!] за 3 секунди.



Фигура 4 Последователност на кодови блокове в Scratch , илюстриращи основна верига от събития и действия. Източник: Scratch от MIT Media Lab

СТЪПКА 9: ТЕСТВАЙТЕ И ПОДОБРЕТЕ

Стартирайте проекта си и въведете различни съобщения. Наблюдавайте как спрайтът реагира въз основа на класификацията на ИИ. Ако моделът класифицира текста погрешно, върнете се в ML4K, добавете още примери за обучение, обучете отново и обновете проекта си в Scratch.

РАЗДЕЛ 3.6 РАЗШИРЕНИЕ ЗА ИИ: ПРЕОБРАЗУВАНЕ НА ТЕКСТ В РЕЧ

Преобразуването на текст в реч (Text-to-Speech, TTS) е технология, управлявана от ИИ, която преобразува написан текст в изговорен звук. Това официално разширение за Scratch използва облачен синтез на реч, за да даде на спрайтовете ви реалистични гласове на множество езици и с различна интонация.

TTS прави проектите по-достъпни (потребители с увредено зрение могат да чуват съдържанието), по-ангажиращи (изговорените диалози са по-динамични от текстовите балончета) и по-универсални (проектите могат да разказват истории, да предоставят инструкции или да служат като инструменти за изучаване на езици).

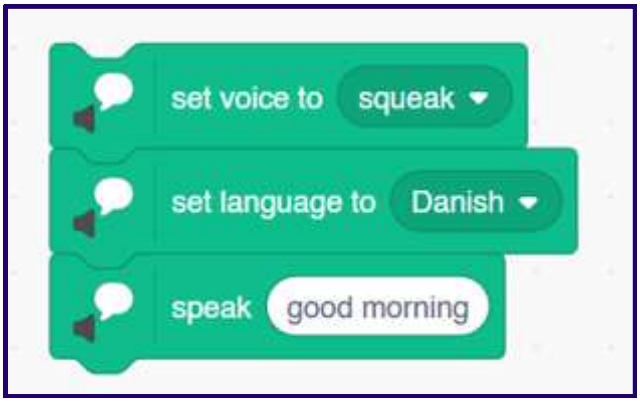
ДОБАВЯНЕ РАЗШИРЕНИЕТО ЗА ПРЕОБРАЗУВАНЕ НА ТЕКСТ В РЕЧ

Стъпка 1: Отворете Библиотеката с разширения. В редактора на Scratch кликнете върху синия бутон "Добави разширение" (с иконата "+") в долния ляв ъгъл на палитрата с блокове.

Стъпка 2: Превъртете в библиотеката с разширения и кликнете върху "Преобразуване на текст в реч" (с иконата на говорител с речеви балончета). Нови блокове ще се появят в палитрата ви под нова категория "Преобразуване на текст в реч".

Стъпка 3: Използвайте блоковете в кода си:

- **Произнасяне на фраза:** Провлачете блока "произнеси [здравей]" в зоната за код. Кликнете вътре в блока, за да смените текста с каквото искате спрайтът ви да каже.
- **Промяна на гласа:** Използвайте блока "задай глас на [алто]", за да изберете различен глас. Вариантите включват Алто, Тенор, Скрип, Великан и Коте.
- **Промяна на езика:** За промяна на езика използвайте блока "задай език на".



Фигура 5 Последователност от блокове с код за разговор в Scratch.
Източник: Scratch от MIT Media Lab

ИЗПОЛЗВАНЕ НА БЛОКОВЕТЕ В SCRATCH ЗА СЪЗДАВАНЕ НА ИНТЕРАКТИВНИ ПРОЕКТИ

Комбинирайте блоковете за преобразуване на текст в реч с други блокове в Scratch, за да създавате интерактивни проекти. Например можете да използвате блока за събитие "когато е кликнато зеленото знаме" от категорията "Събития", за да стартирате речта. Можете също да използвате блока "попитай и изчакай" от категорията "Сензори" и блока "отговор" заедно с блока "произнеси", за да накарате спрайта да повтаря това, което потребителят е написал.

ОСНОВНИ БЛОКОВЕ ЗА ПРЕОБРАЗУВАНЕ НА ТЕКСТ В РЕЧ

Това разширение предоставя блокове за управление на речта:

Блок	Функция	Пример
произнеси [текст]	Произнася посочения текст на глас	произнеси [Здравейте, добре дошли в моя проект!]
задай глас [глас]	Променя гласовия персонаж	задай глас [тенор] (варианти: алто, тенор, скрип, великан, коте)
задай език [език]	Променя езика на речта	задай език [Испански] (поддържат се 23 езика)

ГЛАСОВИ ВАРИАНТИ ЗА ПРЕОБРАЗУВАНЕ НА ТЕКСТ В РЕЧ

Има редица гласови варианти за преобразуване на текст в реч, с които можете да експериментирате, за да съответстват на личността на спрайта ви:

- Алто: Глас със средна височина, неутрален
- Тенор: Глас с малко по-висока честота
- Скрип: Глас с висока честота, детски
- Великан: Дълбок, гърмящ глас
- Коте: Много висок, сладък глас

ПРИМЕРИ ЗА ИЗПОЛЗВАНЕ НА ПРЕОБРАЗУВАНЕТО НА ТЕКСТ В РЕЧ В SCRATCH

- **Интерактивни истории:** Персонажите разказват развитието на сюжета и произнасят диалог.
- **Образователни тестове:** Компютърът чете въпросите на глас за достъпност.
- **Изучаване на езици:** Практика на произношение на различни езици.
- **Виртуални асистенти:** Асистенти като Siri, произнасящи отговори.
- **Аудиогидове:** Обиколки на музеи или градски обиколки с разказана информация.
- **Динамична обратна връзка:** Изговорена насърчителна подкрепа или подсказки в игри.

ИЗТОЧНИЦИ

1.Scratch Foundation (MIT Media Lab): Документация на платформата, образователни ресурси и описания на интерфейса. https://scratch.mit.edu/

2.Machine Learning for Kids (Dale Lane, IBM), 11 февруари 2021 г.: Уроци за платформата, методология за обучение на модели и ръководства за интеграция. ISBN-13: 9781718500563

3.Scratch Wiki: Техническа документация за разширения и описания на блокове. https://scratch-wiki.info/

4.The AI Scratch Code Playbook: A Beginner's Guide to Building Intelligent Systems. Paperback, 11 февруари 2025 г. ISBN-13: 979-8310433649

ЛИНКОВЕ КЪМ УЧЕБНИ ПЛАТФОРМИ

Официален уебсайт на Scratch
Официалната платформа Scratch, разработена от MIT Media Lab. Създайте акаунт, изграждайте проекти, разглеждайте милиони творби на общността и получавайте достъп до уроци и учебни ресурси.
https://scratch.mit.edu/

Machine Learning for Kids
Образователна платформа от IBM за обучение на персонализирани модели за машинно обучение (текст, изображения, числа, звуци) и интегрирането им с Scratch. Включва работни листове, уроци и предварително обучени модели.
https://machinelearningforkids.co.uk/

Творческо преподаване на концепции от ИИ чрез Scratch (Codingal)
Блог публикация, изследваща творчески подходи за преподаване на концепции от ИИ чрез Scratch, с идеи за проекти и педагогически стратегии.
https://www.codingal.com/coding-for-kids/blog/teach-kids-ai-codng-concepts-creatively-using-scratch/

Scratch Machine Learning Studio
Подбрана колекция от проекти в Scratch, използващи разширения за машинно обучение. Разглеждайте за вдъхновение и преработвайте, за да учите от съществуващи проекти.
https://scratch.mit.edu/studios/3995548/

ПРАКТИЧЕСКО УПРАЖНЕНИЕ

Цел: Създайте проект, използващ разширението за преобразуване на текст в реч, за да експериментирате с различни гласове, отговарящи на личността на спрайтовете ви.

Инструкции: Следвайте стъпките по-долу, за да създадете спрайт, произнасящ потребителския вход на глас, и да демонстрирате силата на комбинирането на текстов вход с гласов изход.

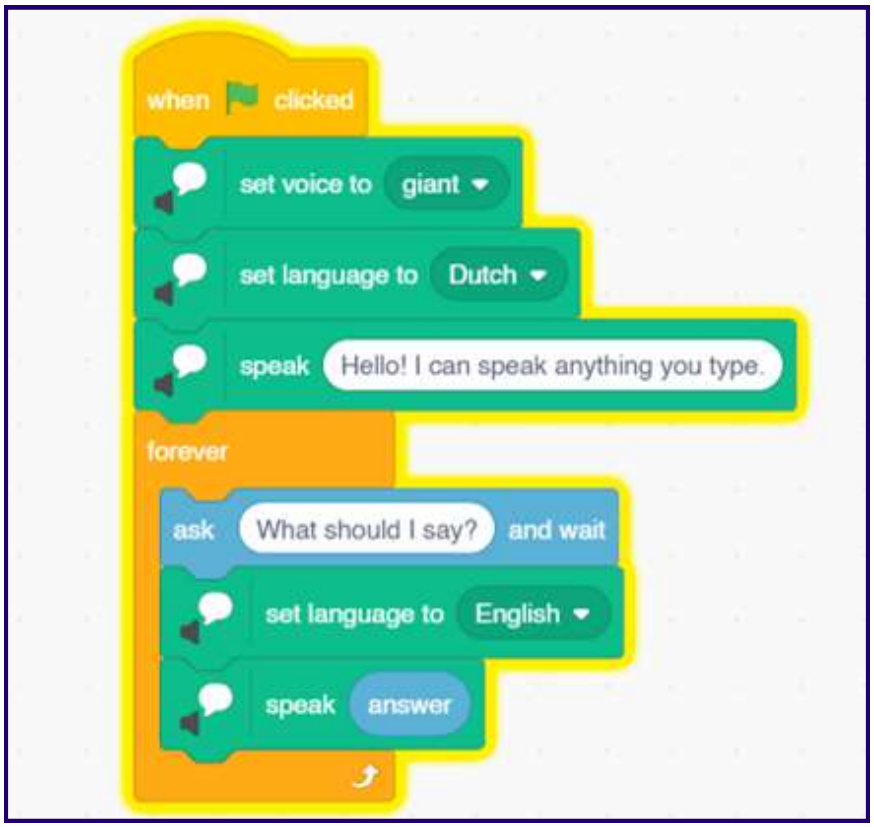
Стъпка 1: Настройте проекта: Създайте нов проект в Scratch или продължете с вече съществуващ. Изберете спрайт, който ще говори - спрайтовете "Робот" или "Магьосник" работят добре.

Стъпка 2: Добавете разширението: Добавете разширението за преобразуване на текст в реч, както е описано по-горе.

Стъпка 3: Създайте скрипта и изградете следния скрипт за спрайта си:

когато е кликнато зеленото знаме
└ задай глас на [великан]
└ произнеси [Здравей! Мога да произнеса всичко, което напишеш.]
└ завинаги
└ попитай [Какво да кажа?] и изчакай
└ произнеси (отговор)

Стъпка 4: Тествайте проекта.
Кликнете върху зеленото знаме и въведете съобщения при подканяне. Спрайтът ви ще произнесе съобщенията на глас.
Опитайте различни гласове и езици, като промените блоковете "задай глас" и "задай език".



Фигура 6 Последователност от блокове с код в Scratch. Източник: Scratch от MIT Media Lab

РЕФЛЕКТИВНИ ВЪПРОСИ

Отделете време, за да помислите задълбочено върху наученото в този модул. Отговорете на следните въпроси:

- Кой аспект от използването на ИИ в Scratch ви изненада най-много? Работи ли нещо по различен начин от очакваното? Обяснете с конкретни примери от вашия проект.
- Как бихте използвали машинното обучение (ML) в игра или интерактивна история? Опишете конкретна идея за проект и обяснете как ИИ би направил проекта по-ангажиращ или по-интелигентен в сравнение с традиционното програмиране.
- Какви проблеми или трудности срещнахте при обучение на ML модела или интегриране на разширенията за ИИ? Как ги решихте? Ако срещнахте проблеми, с които не успяхте да се справите, какви стратегии бихте опитали следващия път?

ТЕСТ (всеки въпрос има само един верен отговор)

1. Кое твърдение най-добре описва Scratch?

a)Текстов програмен език, използван от професионални софтуерни инженери
b)Визуален блоково базиран програмен език, правещ програмирането достъпно за начинаещи
c)Услуга за изкуствен интелект за изграждане на модели за машинно обучение
d)Хардуерен комплект с роботи за обучение на деца по про**грамиране**

2. Кое е едно от основните предимства на блоковото програмиране в Scratch?

a) Изисква от учениците да запомнят синтаксис, подобрявайки паметта им
b) Блоковете се свързват само по логически правилни начини, предотвратявайки синтактични грешки
c) Фокусира се само върху текстово базирано програмиране, което е по-предизвикателно за учащите
d) Може да се използва само от хора с предишен опит в програмирането

3. Коя категория блокове в Scratch бихте използвали за създаване на цикъл, повтарящ код 10 пъти?

a) Движение
b) Събития
c) Управление
d) Оператори

4. Какво трябва да направите, преди да можете да използвате модел от Machine Learning for Kids в проекта си в Scratch?

a) Нищо - разширението идва с предварително обучен модел, работещ за всякакви цели
b) Да предоставите примери за обучение за всяка категория, която искате моделът да разпознава, след което да обучите модела
c) Да закупите специално хардуерно устройство за активиране на машинното обучение
d) Да напишете Python код за създаване на алгоритъм за машинно обучение и да го импортирате в Scratch

ТЕСТ (всеки въпрос има само един верен отговор)

5. Кое от следните НЕ е свързана с ИИ възможност, спомената в този модул?

a) Разпознаване на обекти или изображения чрез класификация на изображения
b) Накарване на спрайт да изпълни танцова последователност с помощта на цикъл за повторение
c) Преобразуване на изговорени думи в текст чрез разпознаване на реч
d) Превод на текст от английски на испански в рамките на проект

6. Кое разширение за Scratch бихте използвали, за да накарате спрайт да разказва история на глас?

a) Видеосензор
b) Превод
c) Преобразуване на текст в реч
d) Machine Learning for Kids

7. Защо е важно да предоставяте множество примери за обучение при създаване на модел за машинно обучение?

a) Моделът не може да функционира с по-малко от 100 примера
b) Наличието на повече примери помага на модела да научи закономерностите по-точно и да класифицира новите входни данни по-надеждно
c) Примерите за обучение се използват само за украса и не влияят на производителността
d) Множеството примери забавят модела, правейки го по-точен

8. Какво предимство дава комбинирането на Machine Learning for Kids с преобразуването на текст в реч?

a) Прави проекта да работи по-бързо
b) Разбира входните данни и комуникира отговори
c) Намалява количеството код, който трябва да напишете
d) Позволява на проекта да работи офлайн без интернет

МОДУЛ 4: Програмиране на игри на Python (библиотеки Pygame / Arcade) с изкуствен интелект

ВЪВЕДЕНИЕ

В този кратък и достъпен за начинаещи модул вие ще се превърнете в създател на игри, а не само в играч. Ще научите как да проектирате и програмирате малка видеоигра с Python и Pygame и ще разберете как да накарате персонажите в играта да се държат интелигентно с прости форми на изкуствен интелект (ИИ). Не се притеснявайте, ако никога преди не сте програмирали — всичко ще бъде обяснено стъпка по стъпка. Ще започнете с отваряне на малък прозорец за игра и преместване на квадрат по екрана, а по-късно ще проучите как да накарате играта да реагира на това, което правите — например като накарате противника да забележи движенията ви или да смени цвета си, когато се доближите. Този модул не е само за стартиране на код, а за разбиране и експериментиране. Ще тествате и модифицирате примери, за да видите как малки промени могат да създадат различно поведение. По този начин ще научите как игрите "мислят" и реагират подобно на прости ИИ системи в реалния свят.

До края на този модул ще придобиете следните знания и умения:

- Да създавате прости интерактивни 2D игри с Python и библиотеките Pygame или Arcade.
- Да добавяте основни поведения на ИИ, за да могат персонажите в играта да реагират на действията на играча.
- Да използвате логическо и изчислително мислене за проектиране на интерактивни игри.
- Да експериментирате, тествате и модифицирате код на игри, за да наблюдавате как малките промени влияят на поведението.
- Да разберете начина, по който ИИ се използва в реалния живот, в области като образование, здравеопазване, финанси и забавление.

Продължителност на модула	2 часа (1 час преподаване в клас + 1 час практически упражнения)

ОБРАЗОВАТЕЛНИ МАТЕРИАЛИ

РАЗДЕЛ 4.1 ВЪВЕДЕНИЕ В PYGAME

Преди да започнете да програмирате, е важно да знаете с кои инструменти ще работите и за какво служат:

- **Python 3** - програмният език, с който ще давате инструкции на компютъра. Той се използва широко в училища, университети и компании за програмиране, анализ на данни и ИИ.

- **Pygame** - безплатна библиотека за Python, която ви помага да създавате прости 2D игри. Тя предоставя готови функции за отваряне на прозорец за игра, рисуване на форми, показване на цветове и реагиране на клавиатурата или мишката.

- **Редактор на код (IDLE, Thonny или VS Code)** - програмата, в която ще пишете, запазвате и стартирате кода си на Python. В този модул можете да използвате IDLE (включен с Python) или друг редактор, препоръчан от учителя.

Тези инструменти ще бъдат представени стъпка по стъпка, за да можете да ги инсталирате и да изпълните дейностите самостоятелно, дори ако това е първият ви контакт с програмирането.

Цел: Научете как да отворите прозорец за игра, да показвате цветове и да местите квадрат с клавиатурата.

Стъпка 1 – Инсталиране на Python 3 и отваряне на средата за програмиране
Преди да започнем да използваме Pygame, трябва да се уверите, че **Python 3** е инсталиран на компютъра ви.

Инсталирайте Python 3

- Отидете на официалния уебсайт на Python: https://www.python.org/downloads/
- Изтеглете версията за вашата система (Windows, macOS или Linux).
- По време на инсталацията отметнете опцията "Add Python to PATH", преди да кликнете върху "Install Now".
- След инсталирането рестартирайте компютъра (не е задължително, но е препоръчително).

Проверете инсталацията

Отворете командния ред (Windows) или терминала (macOS/Linux) и въведете:

python --version или **py --version**

Ако се покаже нещо подобно на Python 3.13.3, всичко е готово.

РАЗДЕЛ 4.2 ИНСТАЛИРАНЕ НА РЕДАКТОР ИЛИ ИЗПОЛЗВАНЕ НА IDLE

Можете да пишете Python код с един от следните редактори:

- IDLE (инсталиран заедно с Python)
- Thonny (прост IDE за начинаещи): https://thonny.org
- VS Code (разширена опция): https://code.visualstudio.com

За да отворите IDLE:

- На Windows: натиснете Start → IDLE (Python 3.x)
- На macOS: потърсете "IDLE" в Launchpad
- На Linux: изпълнете idle3 в терминала

След отваряне можете да създадете нов файл (File → New File), да напишете код и да го стартирате (Run → Run Module или натиснете F5).

РАЗДЕЛ 4.3 КАК ДА ОТВОРИТЕ PYTHON (IDLE) И ДА СТАРТИРАТЕ ПЪРВАТА СИ ПРОГРАМА

След като Python 3 е инсталиран, можете да използвате вградения редактор IDLE за писане и стартиране на кода.

1. Отворете IDLE

- На Windows: натиснете Start → IDLE (Python 3.x).
- На macOS: потърсете IDLE в Launchpad.
- На Linux: въведете idle3 в терминала.

2. Създайте нов файл

- В IDLE кликнете File → New File (или натиснете Ctrl + N).
- Отваря се празен прозорец, в който можете да пишете програмата си.

3. Въведете или поставете кода
Копирайте кода от този модул в новия файл. Например:

КОД:
```
import pygame
pygame.init()
screen = pygame.display.set_mode((800, 600))
pygame.display.set_caption("My First Game")
running = True
while running:
   for event in pygame.event.get():
       if event.type == pygame.QUIT:
           running = False
   screen.fill((200, 220, 255))
   pygame.display.flip()
pygame.quit()
```

4. Запазете файла
Изберете File → Save As... и назовете файла (например my_first_game.py).

5. Стартирайте програмата
- Натиснете F5 или отидете на Run → Run Module.
- Появява се нов прозорец — ако виждате синьо-лилав фон, всичко работи.

РАЗДЕЛ 4.4 СЪЗДАЙТЕ ПЪРВИЯ СИ ПРОЗОРЕЦ ЗА ИГРА

Копирайте и стартирайте следния код. Коментарите обясняват всяка стъпка:

КОД:

```
import pygame  # 1. Import the pygame library
pygame.init()  # 2. Start (initialize) all pygame modules
screen = pygame.display.set_mode((800, 600))  # 3. Create a
game window (width 800, height 600)
pygame.display.set_caption("My First Game") # 4. Give a title
to the game window
running = True # 5. Create a variable to keep the game
running

while running:  # 6. Start the main game loop (runs again and
again)
   for event in pygame.event.get():  # 7. Check all the
events (for example, key presses or mouse clicks)
       if event.type == pygame.QUIT: # 8. If the user clicks
the "close" button…
           running = False          # 9. …stop the game loop

   screen.fill((200, 220, 255))      # 10. Fill the screen
with a light blue colour (RGB values)
   pygame.display.flip()          # 11. Update the window to
show the new frame

pygame.quit() # 12. Close the game and exit safely
```

РАЗДЕЛ 4. 5 ПРЕМЕСТВАНЕ НА КВАДРАТ

Сега нека накараме нещо да се движи.

Копирайте кода по-долу и го тествайте:

ЧАСТ I

КОД:

```
import pygame # 1. Import the pygame library
pygame.init() # 2. Start pygame
# 3. Create the window
screen = pygame.display.set_mode((800, 600))
# 4. Add a title
pygame.display.set_caption("Moving Square")
x = 400  # 5. The square’s X position (horizontal)
y = 300  # 6. The square’s Y position (vertical)
speed = 1 # 7. How many pixels it moves each time

running = True # 8. Keep the game running

while running:  # 9. Start the game loop
   for event in pygame.event.get(): # 10. Check events
       if event.type == pygame.QUIT: # 11. If user clicks
“X”, stop
           running = False
```

РАЗДЕЛ 4. 5 ПРЕМЕСТВАНЕ НА КВАДРАТ

Сега нека накараме нещо да се движи.

Копирайте кода по-долу и го тествайте:

ЧАСТ II

КОД:

```
   keys = pygame.key.get_pressed() # 12. Check which keys are
pressed
   if keys[pygame.K_LEFT]:
       x -= speed  # 13. Move left
   if keys[pygame.K_RIGHT]:
       x += speed  # 14. Move right
   if keys[pygame.K_UP]:
       y -= speed  # 15. Move up
   if keys[pygame.K_DOWN]:
       y += speed  # 16. Move down

   screen.fill((200, 220, 255)) # 17. Paint the background
   pygame.draw.rect(screen, (0, 0, 255), (x, y, 50, 50)) #
18. Draw the blue square
   pygame.display.flip() # 19. Update the screen

pygame.quit() # 20. Close the game
```

Какво се случва:

- x и y определят позицията на квадрата на екрана.
- Клавишите за стрелки променят тези стойности, местейки квадрата.
- Цикълът повтаря рисуването много пъти в секунда, създавайки плавно движение.

Опитайте следното:

- Сменете speed = 1 на speed = 5. Какво се случва?
- Направете квадрата в различен цвят, например (255, 0, 0) за червено.
- Какво мислите, че ще се случи, ако премахнете pygame.display.flip()?

Въпрос за размисъл:
Как компютърът знае новата позиция на квадрата всеки път, когато натиснете клавиш?

Какво научихте до тук:
На този етап трябва да можете да:

- Отворите прозорец на Pygame.
- Използвате цикъла на играта, за да я поддържате работеща.
- Управлявате движещ се обект с клавиатурата.

В следващата част ще добавите изкуствен интелект, за да може играта да реагира на вашите движения.

РАЗДЕЛ 4.6 БАЗОВ ИИ В ИГРИ (РЕАКТИВЕН ПРОТИВНИК – REACTIVE ENEMY)

В този модул използваме базов ИИ, основан на правила: компютърът следва предварително зададени правила, за да възприема играча, решава какво да прави и действа без да учи от данни. Този модел се използва много класически видеоигри, в които противниците преследват, избягват или реагират на играча въз основа на разстояние или позиция.

Цел: Научете как да накарате играта да реагира на действията на играча, като добавите базова форма на изкуствен интелект (ИИ). Ще създадете противник, който сменя цвета си, когато играчът се доближи.

Стъпка 1 – Разбиране на идеята

Преди да кодирате, помислете какво означава ИИ в игра.
В случая ИИ не е свързан с учене или данни — той се използва, за да накарва компютъра да реагира на това, което прави играчът.

Ще използвате просто правило: Ако играчът е близо, противникът сменя цвета си. Ако не — остава същият.

Това е пример за условно поведение - една от най-простите форми на ИИ.

Стъпка 2 – Пример за код: реактивен противник

Копирайте и стартирайте следния код:

ЧАСТ I

КОД:

```
import pygame, math      # 1. Import game and math libraries
pygame.init()         # 2. Start (initialize) all Pygame
functions

screen = pygame.display.set_mode((800, 600))
# 3. Create a game window (800x600)
pygame.display.set_caption("Simple AI – Reactive Enemy")
# 4. Title

player = pygame.Rect(100, 100, 50, 50)
# 5. Player (blue square)
enemy= pygame.Rect(500, 300, 50, 50)  # 6. Enemy (red square)
speed = 5                # 7. Player speed
react_distance = 120
# 8. Distance (px) for enemy to "react"

clock = pygame.time.Clock()
running = True           # 9. Keep the game running
```

ЧАСТ II

КОД:

```
while running:           # 10. Main game loop (every frame)
for event in pygame.event.get():# 11. Check all events
if event.type == pygame.QUIT: # 12. If the window is closed…
            running = False               # 13. …stop the loop

   # ---- MOVEMENT (must be inside the while) ----
keys = pygame.key.get_pressed()  # 14. Which keys are
pressed?
if keys[pygame.K_LEFT]:
player.x -= speed                # 15. Move left
if keys[pygame.K_RIGHT]:
player.x += speed                # 16. Move right
if keys[pygame.K_UP]:
player.y -= speed                # 17. Move up
if keys[pygame.K_DOWN]:
player.y += speed                # 18. Move down

# ---- SIMPLE AI: change colour if close ----
px, py = player.center# 19. Player center
ex, ey = enemy.center# 20. Enemy center
dist = math.hypot(px - ex, py - ey)# 21. Distance

if dist < react_distance: # 22. If closer than 120 px…
enemy_color = (255, 200, 0)      # 23. …enemy turns yellow
else:
enemy_color = (220, 60, 60) # 24. Otherwise, enemy stays red
```

ЧАСТ III

КОД:

```
# ---- DRAW ----
screen.fill((240, 245, 255))# 25. Background
pygame.draw.rect(screen, (60, 120, 255), player)
# 26. Player (blue)
pygame.draw.rect(screen, enemy_color, enemy)
# 27. Enemy (reactive)
pygame.display.flip()  # 28. Show changes

clock.tick(60)# Limit to 60 FPS

pygame.quit()     # 29. Close safely
```

Стъпка 3 – Какво се случва в кода

Концепция	**Обяснение**
Възприятие	Противникът измерва разстоянието до играча с math.hypot.
Решение	Ако разстоянието е по-малко от react_distance, противникът решава да реагира.
Действие	Реакцията се изразява в промяна на цвета.

Цикълът "възприемай → решавай → действай" е основата на повечето ИИ системи в игрите.

Дори такива прости реакции правят игровия свят да изглежда по-жив.

Стъпка 4 – Опитайте, сменяйте, наблюдайте

Опитайте да промените следните части от кода и наблюдавайте какво се случва:

1. Сменете react_distance от 120 на 250 — какво се случва?
2. Накарайте противника да стане зелен (0, 255, 0) вместо жълт.
3. Накарайте играча да се движи по-бързо, като увеличите speed =
4. Можете ли да накарате противника да реагира само когато играчът се движи над или под него (например сравнете само позициите по Y)?

Насърчете учениците да предвидят резултата, преди да стартират програмата - какво мислят, че ще се случи?

Стъпка 5 – Размисъл

Това наистина ли е "интелект"? Защо да или защо не?
Как бихме могли да направим поведението на противника да изглежда по-умно?

Съвет: Реалният ИИ в игрите използва същата логика — прости решения, повтаряни много пъти в секунда, могат да създадат илюзия за интелигентност.

ИЗТОЧНИЦИ

1. Python Software Foundation. Python 3 Documentation. Достъпно на: https://docs.python.org
2. Pygame Community. Pygame Documentation and Tutorials. Достъпно на: https://www.pygame.org/docs/ и https://www.pygame.org/wiki/tutorials
3. Python Arcade Project. Python Arcade Library Documentation. Достъпно на: https://api.arcade.academy
4. Real Python. Pygame: A Primer on Game Programming in Python. Достъпно на: https://realpython.com/pygame-a-primer/
5. Microsoft. AI for Beginners Curriculum. Достъпно на: https://github.com/microsoft/AI-For-Beginners

ЛИНКОВЕ КЪМ УЧЕБНИ ПЛАТФОРМИ

- Официален уебсайт на Python: Изтеглете и научете основите на програмирането с Python. https://www.python.org/
- Документация на Pygamем Официалното ръководство за функции, събития и графика в Pygame. https://www.pygame.org/docs/
- Документация на библиотеката Arcadem Алтернативна библиотека за прости 2D игри в Python. https://api.arcade.academy/en/latest/
- Pygame AI Guide: Примери за прости ИИ алгоритми за Pygame. https://pygame-ai.readthedocs.io/en/latest/guide.html
- Real Python — Изградете игра с Pygamем Достъпен урок за начинаещи. https://realpython.com/pygame-a-primer/
- KidsCanCode уроцим Лесни проекти и предизвикателства за учене на Pygame. https://kidscancode.org/blog/
- AI for Beginners (Microsoft)м Открито образователно съдържание, въвеждащо принципите на ИИ. https://microsoft.github.io/AI-For-Beginners/

ПРАКТИЧЕСКО УПРАЖНЕНИЕ

ХВАНИ МЕ, АКО МОЖЕШ

Цел: Създайте малка игра, в която играчът мести син квадрат, за да събира обекти, докато червен противник с ИИ го преследва. Това упражнение обединява всички научени умения: движение, откриване на сблъсъци и просто поведение на ИИ.

Стъпка 1 – Подгответе среда

pip install pygame

Уверете се, че Pygame е инсталиран

След това отворете редактора за Python и създайте нов файл с наименование catch_me.py.

Стъпка 2 – Разберете идеята на играта

Елемент	Описание
Играч	Движи се с клавишите за стрелки.
Цел	Появява се на произволно място; увеличава резултата при събиране.
Противник (ИИ)	Следва играча, като изчислява посоката и разстоянието.
Цел на играта	Съберете колкото се може повече цели, преди противникът да ви хване.

Помислете преди да започнете да кодирате: Как противникът може да "знае" къде се намира играчът?

Стъпка 3 – Изградете играта стъпка по стъпка

ЧАСТ I

КОД:

```
import pygame, random, math
pygame.init()

# --- Window ---
WIDTH, HEIGHT = 800, 600
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Catch Me If You Can")
clock = pygame.time.Clock()

# --- Colors ---
BLUE= (60, 120, 255)
RED= (220, 60, 60)
GREEN = (60, 200, 60)
WHITE = (240, 245, 255)
BLACK = (20, 20, 20)

# --- Entities ---
player = pygame.Rect(100, 100, 40, 40)
# blue square (player)
enemy= pygame.Rect(600, 400, 40, 40)
# red square (chasing enemy)
target = pygame.Rect(
random.randint(50, WIDTH - 50),
# green target: random position
random.randint(50, HEIGHT - 50),25,25)
```

Стъпка 3 – Изградете играта стъпка по стъпка

ЧАСТ II

КОД:

```
PLAYER_SPEED = 5
ENEMY_SPEED = 2.5
score = 0

# Font (fallback-friendly: uses default font to avoid font
issues)
font = pygame.font.Font(None, 26)

# --- Game loop ---
running = True
while running:
   # 1) Handle events
for event in pygame.event.get():
if event.type == pygame.QUIT:
           running = False

# 2) Player movement (arrow keys)
keys = pygame.key.get_pressed()
if keys[pygame.K_LEFT]:
player.x -= PLAYER_SPEED
if keys[pygame.K_RIGHT]:
player.x += PLAYER_SPEED
if keys[pygame.K_UP]:
player.y -= PLAYER_SPEED
if keys[pygame.K_DOWN]:
player.y += PLAYER_SPEED
```

Стъпка 3 – Изградете играта стъпка по стъпка

ЧАСТ III

КОД:

```
# Keep player inside the window boundaries
player.clamp_ip(screen.get_rect())

   # 3) Simple AI: enemy chases the player
dx = player.centerx - enemy.centerx
dy = player.centery - enemy.centery
dist = math.hypot(dx, dy)
if dist != 0: # avoid division by zero when both overlap
enemy.x += int((dx / dist) * ENEMY_SPEED)
enemy.y += int((dy / dist) * ENEMY_SPEED)

# 4) Player collects target
if player.colliderect(target):
score += 1
target.topleft = (
           random.randint(25, WIDTH - 50),
           random.randint(25, HEIGHT - 50)
)

# 5) Draw everything
screen.fill(WHITE)
pygame.draw.rect(screen, BLUE,player)
pygame.draw.rect(screen, RED,enemy)
pygame.draw.rect(screen, GREEN, target)
```

Стъпка 3 – Изградете играта стъпка по стъпка

```
ЧАСТ IV

КОД:
# Draw score
score_text = font.render(f"Score: {score}", True, BLACK)
screen.blit(score_text, (10, 10))

pygame.display.flip()
clock.tick(60)

pygame.quit()
```

Стъпка 4 – Експериментирайте

Опитайте да промените тези стойности и наблюдавайте какво се случва:
ü Сменете ENEMY_SPEED, за да накарате противника да се движи по-бързо или по-бавно.

- Сменете цвета на играча или целта.
- Направете играта по-трудна, като намалите размера на играча.
- Добавете съобщение "Играта свърши" (Game Over), когато противникът докосне играча.

РЕФЛЕКТИВНИ ВЪПРОСИ

Отделете време, за да помислите задълбочено върху наученото в този модул. Отговорете обмислено на следните въпроси:

- Кое беше най-трудното при създаването на играта?
- Как ИИ прави играта по-интерактивна?
- Можете ли да се сетите за други ситуации, в които ИИ реагира на човешки действия (например в образованието или здравеопазването)?
- Ако можехте да подобрите тази игра, какво поведение на ИИ бихте добавили следващия път?
- Какво прави противника да изглежда "интелигентен"?
- Как изчисляването на разстоянието (math.hypot) му помага да намери играча?
- Можете ли да се сетите за реални игри, използващи подобна логика?

ТЕСТ (всеки въпрос има само един верен отговор)

1. Какво прави цикълът на играта в Pygame?
a)Инсталира нови библиотеки по време на играта
b)Повтаря действията на играта много пъти в секунда
c)Затваря прозореца на играта
d)Зарежда фоновото изображение само веднъж

2. Каква е целта на pygame.display.flip()?
a)Проверява входа от клавиатурата
b)Рисува правоъгълник
c)Обновява прозореца на играта с нова визуализация
d)Поставя цикъла на играта на пауза

3. Коя функция проверява кои клавиши на клавиатурата са натиснати?
a)pygame.quit()
b)pygame.key.get_pressed()
c)pygame.event.get()
d)pygame.display.set_mode()

4. В примера с ИИ какво кара противника да реагира на играча?
a)Събитие от таймер
b)Произволно движение
c)Разстоянието между играча и противника
d)Цветът на играча

5. Какво се случва, ако увеличите стойността на react_distance в кода?
a)Противникът реагира от по-далеч
b)Противникът се движи по-бързо
c)Играчът се забавя
d)Размерът на прозореца се променя

ТЕСТ (всеки въпрос има само един верен отговор)

6. Каква е целта на функцията math.hypot() в кода с ИИ?
a)Да изчисли разстоянието между две точки
b)Да нарисува противника на екрана
c)Да създаде нова произволна позиция
d)Да открие сблъсъци със стените

7. Какво прави следният ред код: player = pygame.Rect(100, 100, 40, 40)?
a)Рисува текст на екрана
b)Създава правоъгълник за представяне на играча
c)Стартира ИИ алгоритъма
d)Променя цвета на фона

8. Какво се случва, ако премахнете pygame.display.flip() от цикъла?
a)Програмата работи по-бързо
b)Прозорецът не се обновява и изглежда замръзнал
c)Цветовете се сменят автоматично
d)Играчът се движи два пъти по-бързо

9. Кой от следните примери е пример за поведение на ИИ в игра?
a)Рисуване на квадрат на екрана
b)Персонажът да се движи само, когато играчът натисне клавиш
c)Противникът да следва или да избягва играча
d)Зареждане на фонова музика

10. Как ИИ може да направи игрите по-интересни?
a)Като автоматично отстранява грешки
b)Като създава динамични предизвикателства и отзивчиви персонажи
c)Като автоматично сменя графиката
d)Като управлява клавиатурата

МОДУЛ 5: Разпознаване на обекти с Roboflow – Въведение в компютърното зрение

ВЪВЕДЕНИЕ

Целта на настоящия модул е да предостави цялостно въведение в компютърното зрение (CV), като обяснява как машините възприемат и обработват визуални данни, и да насочи учащите в разработването на собствени модели за разпознаване на обекти с помощта на платформата Roboflow и съвременни алгоритми като YOLO.

До края на този модул ще придобиете следните знания и умения:

- **Разбиране на основните концепции:** Разграничавате обработката на изображения от компютърното зрение и разбирате логиката на начина, по който компютрите интерпретират изображенията като данни (пиксели и двоичен код).
- **Прилагане на процеса на компютърно зрение (CV):** Прилагане на петте основни стъпки на процеса на компютърното зрение: получаване на данни, подготовка на данни, определяне на ИИ модела, обучение на модела и интерпретиране на резултатите
- **Използване на Roboflow:** Навигиране в платформата Roboflow за създаване на проекти, качване на набори от данни и ефективно управление на данни от изображения.
- **Анотиране на данни:** Извършване на точно анотиране (маркиране) на данни за подготовка на набори от данни за обучение на ИИ модели.
- **Практическо приложение:** Изпълняване на пълен проектен цикъл, включващ събиране на реални данни (снимки), обучение на модел и тестване на точността му в практическа среда.

Продължителност на модула	2 часа (1 час преподаване в клас + 1 час практически упражнения)

ОБРАЗОВАТЕЛНИ МАТЕРИАЛИ

РАЗДЕЛ 5.1 КАКВО Е ОБРАБОТКА НА КОМПЮТЪРНО ЗРЕНИЕ?

Обработката на изображения е способ, използван за откриване, извличане и оценяване на важна информация, съдържаща се в цифрово изображение. Този метод дава значими данни за обекти или среди в изображението, което наподобява човешкото визуално възприятие. Основната цел на обработката на изображения е да извлича смисъл от тях и да представя тази информация за използване в различни области.

Въпреки че технологията за обработка на визуална информация съществува отдавна, в миналото този процес е разчитал в голяма степен на човешка намеса. Това е отнемало много време и същевременно е било причина за допускане на грешки. В по-ранните системи за разпознаване на лица разработчиците е трябвало ръчно да маркират хиляди снимки и специфични характеристики, например, ширината на носа или разстоянието между очите са се идентифицирали една по една. Причината е, че данните от изображенията често са били неорганизирани и трудни за разбиране от компютрите. Поради това автоматизирането на процеса е изисквал много мощни процесори и съвременни изчислителни технологии.

Днес, с помощта на нарастващата изчислителна мощ, приложенията за компютърно зрение използват изкуствен интелект и машинно обучение (ИИ/МО) за точна обработка на тези данни с цел идентификация на обекти и разпознаване на лица, както и за класификация, препоръки, проследяване и възприятие.

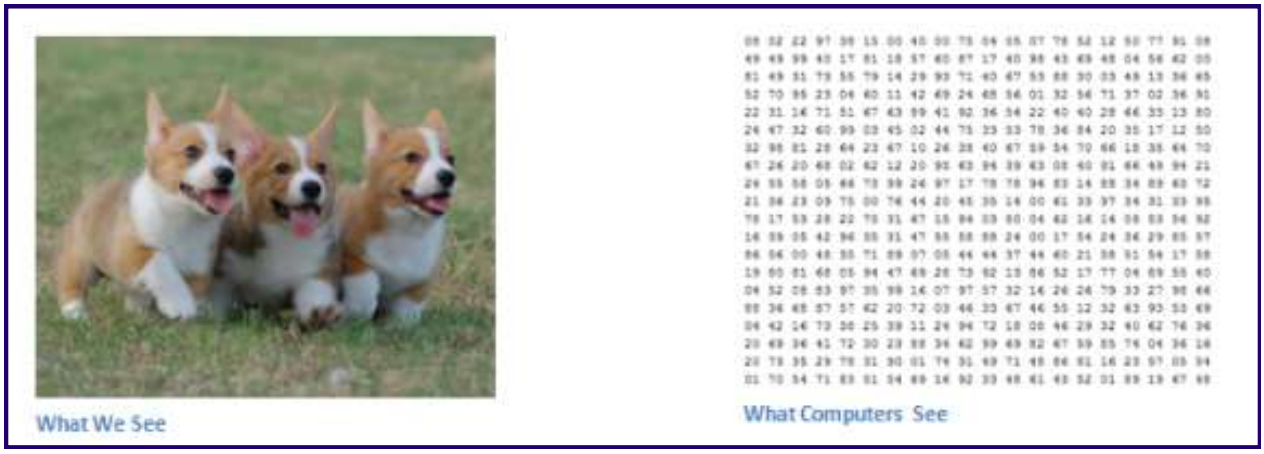
Компютърното зрение може да бъде описано като " машините са оставени да изпълняват тази задача", вдъхновени от човешката зрителна система и нейната връзка с мозъка. Това е процес при машини, като компютрите, при който те възприемат изображения и видеозаписи, анализират ги и ги трансформират с помощта на машинно обучение и изкуствен интелект. Зародилото се през 50-те години на миналия век, компютърното зрение започва да се популяризира през 70-те години благодарение на способността му да разграничава машинописни от ръкописни текстове. Днес то се използва в много области, включително технологии за разпознаване на хора, системи за контрол на качеството на продукти, автономни превозни средства, земеделски приложения, здравеопазване, образование и отбрана.

Компютърното зрение и обработката на изображения са свързани, но различни области. Обработката на изображения включва обработка, възстановяване и коригиране на цвета и яркостта на заснетите данни от изображения, включително снимки и видеоклипове. Компютърното зрение, от своя страна, е прилагането на техники за обработка на изображения върху тях и съчетаването на този процес с изкуствен интелект. Например, докато обработката на изображения се използва за намиране на ръбовете на обект в изображение, използването на техники за машинно обучение за идентифициране и класифициране на този обект попада в областта на компютърното зрение.

РАЗДЕЛ 5.2 КАК „ВИЖДА“ КОМПЮТЪРЪТ?

Градивните елементи на почти всички компютри в света - техните "клетки" - се състоят от числата 1 и 0. Това означава, че компютрите възприемат данните, които потребителите искат да им предадат, на собствения си език: единици и нули.

В крайна сметка изображенията и видеоклиповете, съставени от много изображения, също са данни, затова компютрите ги възприемат като нули и единици, или двоичната бройна система. За по-добра илюстрация можем да използваме изображението по-долу.



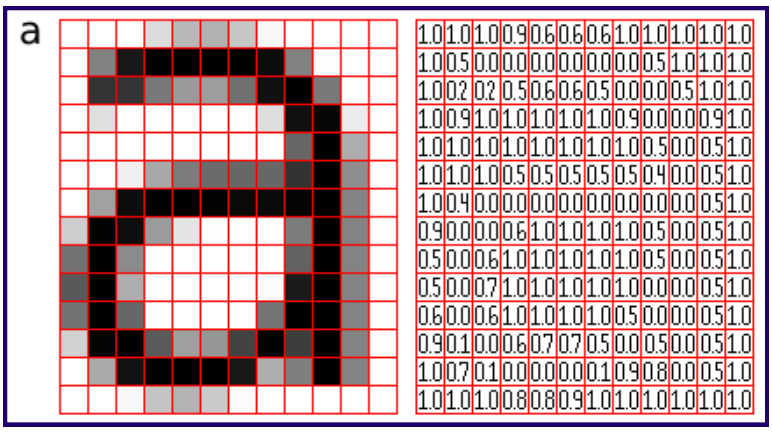
Фигура 7 Начин на виждане: Използване на невронни мрежи за видеонаблюдение (2025) Източник: https://www.videonet9.com/using-neural-networks-for-video-surveillance.html; Дата на добавяне : 09.08.2025.

Ако се питате защо изображението показва само двуцифрени стойности, а не нули и единици — отговорът е прост. Нашите компютърни процесори вземат тези стойности и ги преобразуват вътрешно в нули и единици. Ако всичко това се показваше директно в двоичен вид, изображението щеше да бъде изключително дълго и трудно за четене. Всъщност всяка от тези двуцифрени стойности в изображението по-горе представлява един пиксел.

И така, какво е пиксел? Пикселът е най-малката градивна единица на цифровото изображение.

РАЗДЕЛ 5.3 НЕКА НАПРАВИМ ЕДНО УПРАЖНЕНИЕ

- Вземете лист на квадратчета.
- Номерирайте всеки квадрат от мрежата с числа между 0 и 1.
- Колкото по-близо до 1 е числото в даден квадрат, толкова по-бял е квадратът.
- Колкото по-близо до 0 е числото, толкова по-черен е квадратът.
- Ако числото е 0,5, оцветете квадрата в сиво.



Фигура 8 Начин на виждане: Използване на невронни мрежи за видеонаблюдение (2025) Източник: https://medium.com/@meriyananjalika99/from-pixels-to-predictions-getting-started-with-cnns-convolutional-neural-networks-e3f84781cc1e; Дата на добавяне : 09.08.2025.

Ако нарисувате внимателно в такава тетрадка, от разстояние рисунката ще изглежда напълно нормална. Опитайте го с изображението по-горе — отдалечете се на няколко метра от екрана и ще видите разликата. Всъщност компютърните и телевизионните екрани показват изображения и видеоклипове именно по този начин: чрез милиони малки квадратчета, наредени едно до друго.

За да съхраняват изображенията, устройствата запазват в паметта числото, съответстващо на всяко квадратче. При изображенията с лошо качество, които познаваме като пикселирани, тези квадратчета стават видими с просто око.

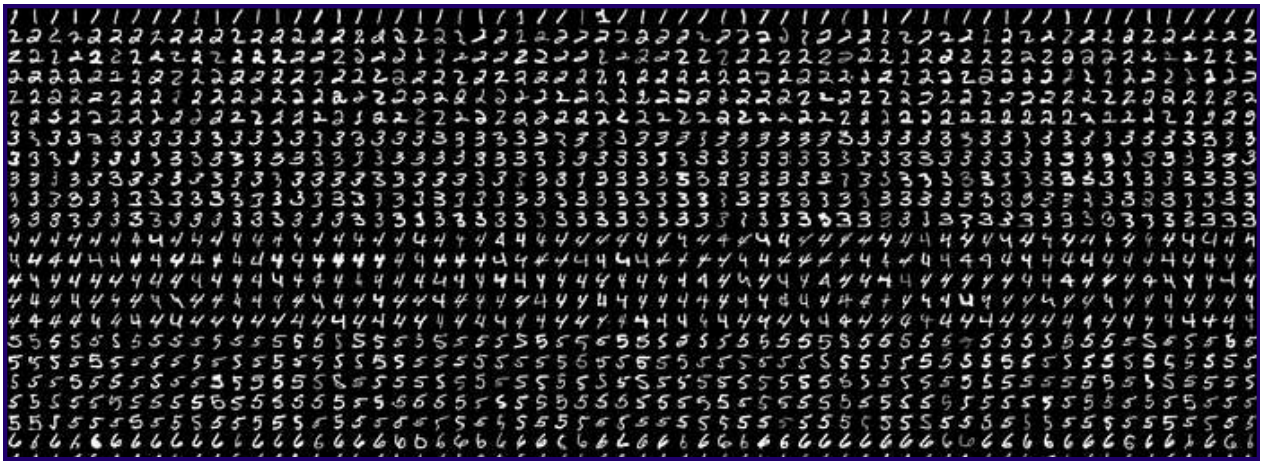
РАЗДЕЛ 5.4 ПРОЦЕСЪТ НА КОМПЮТЪРНОТО ЗРЕНИЕ СЕ СЪСТОИ ОСНОВНО ОТ ПЕТ СТЪПКИ:

1. Получаване на данните
2. Подготовка на данните
3. Определяне на ИИ модела
4. Обучение на ИИ модела
5. Интерпретиране на резултатите

Изображенията, получени за базово приложение за разпознаване на ръкописни цифри, са показани по-долу. Тези изображения са преобразувани в скалата на сивото, удобна за компютъра, и са използвани за обучение на ИИ. Резултатите от тестовете показват точност от 99% след обучението.

РАЗДЕЛ 5.5 ROBOFLOW

Посетете: https://roboflow.com/

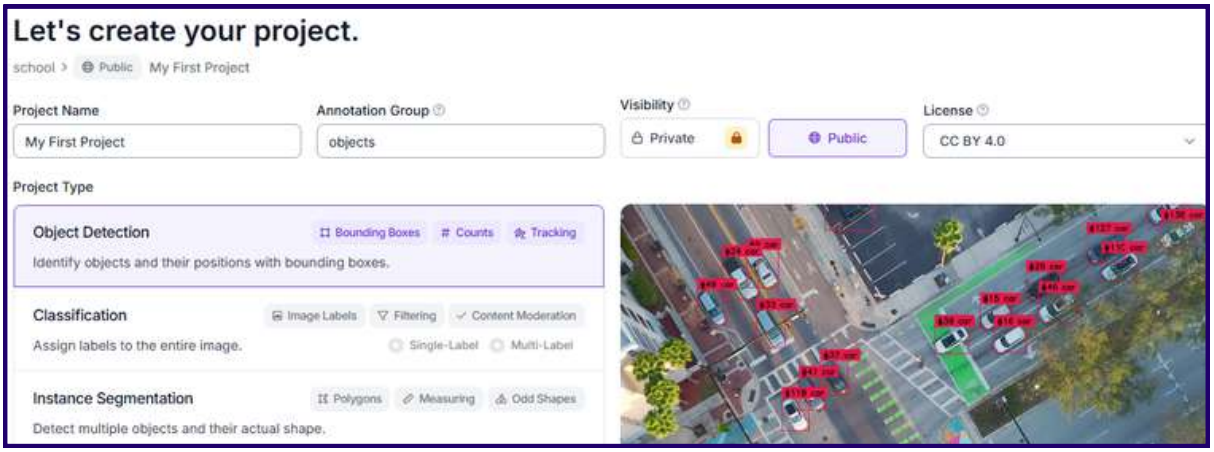


Фигура 9 Kaggle - набор от данни (2025 г.) Източник: www.roboflow.com; Дата на добавяне : 09.08.2025

Roboflow е цялостна платформа за изкуствен интелект, предназначена за разработване на проекти за компютърно зрение. Тя опростява процеса на създаване, обучение и внедряване на модели за компютърно зрение с помощта на данни от изображения и видеоклипове. Предлага лесен за използване интерфейс както за начинаещи, така и за опитни разработчици.

РАЗДЕЛ 5.6 РАЗДЕЛ 5.5 КАК ДА СЕ РЕГИСТРИРАТЕ В ROBOFLOW?

- Отворете сайта https://roboflow.com/
- Кликнете върху бутона "Sign In" в горния десен ъгъл.
- В отворилия се прозорец въведете данните за профил, ако имате такива, и влезте в системата. За тези, които влизат за пръв път, използвайте опцията "Continue With Google", тъй като продължаването с акаунт в Google е най-удобно.
- След избиране на желания акаунт в Google и предоставяне на необходимите разрешения на сайта Roboflow ще влезете в системата.
- Кликнете върху бутона "NEW PROJECT" на отворилата се страница, въведете информацията за проекта и го създайте.

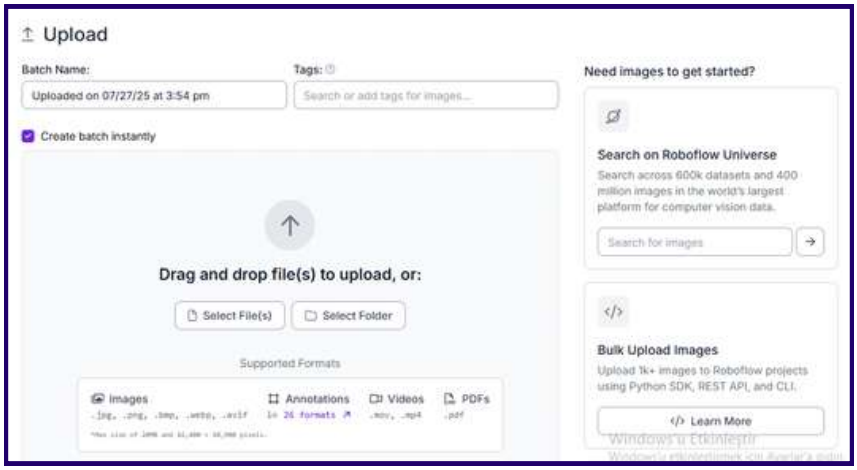


Фигура 10 . Kaggle-Dataset (2025) Източник: www.roboflow.com; Дата на добавяне : 09.08.2025.

Набор от данни (Dataset)

На новата страница, която се отваря, добавяме данните за обучение, валидиране и тестване на ИИ. Оттук нататък ще ги наричаме "набор от данни" (dataset). Ако нямаме набор от данни, можем да използваме готови набори от данни в системата RoboFlow Universe.
https://universe.roboflow.com

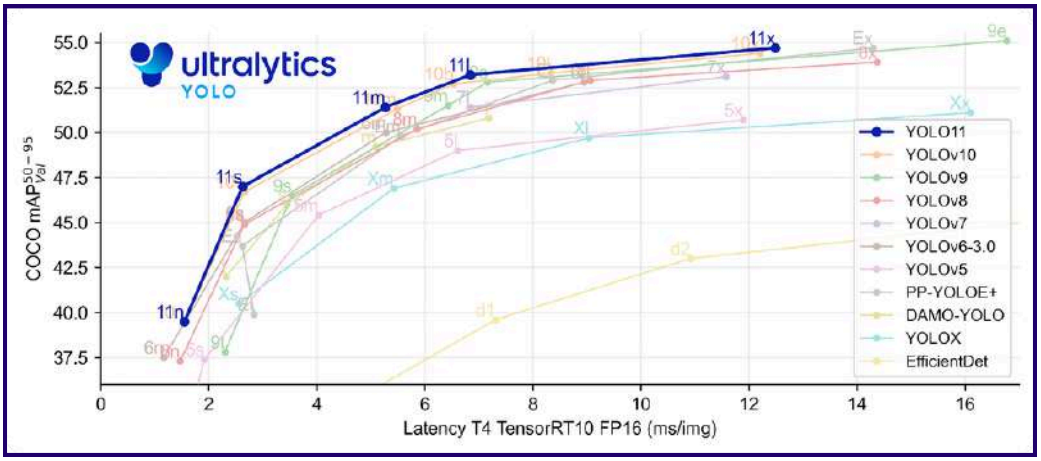
По същия начин Kaggle е уебсайт, предлагащ готови набори от данни.
https://www.kaggle.com/



Фигура 11 Roboflow (2025) Източник: www.roboflow.com; Дата на добавяне: 09.08.2025.

Готов набор от данни

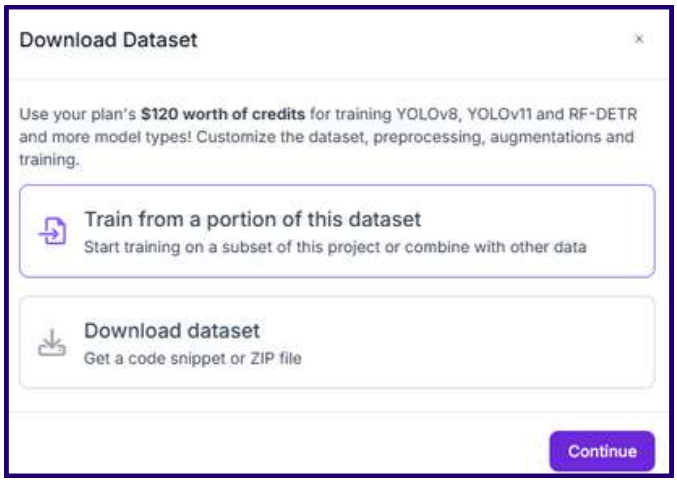
В тази система ще продължим стъпките на процеса с готов набор от данни. Затова ще използваме набора от данни чрез RoboFlow на адрес
https://universe.roboflow.com/roboflow-58fyf/rock-paper-scissors-sxsw.
След въвеждане на този адрес кликнете върху бутона "download dataset" (изтегляне на набора от данни). Преди да кликнете обаче, трябва да решим кой ИИ модел да използваме. В това приложение ще използваме ИИ модела YOLOv11, който е по-ефективен при откриване на обекти. Ако посетите профила в GitHub на съответния алгоритъм (https://github.com/ultralytics/ultralytics), можете да намерите различни предварително обучени тегла и подробна информация. (Онлайн тест на Yolov3: https://v-iashin.github.io/detector)



Фигура 12 Ultralytics (2025) Източник:https://github.com/ultralytics/ultralytics; Дата на добавяне : 09.08.2025.

Проверка на данните

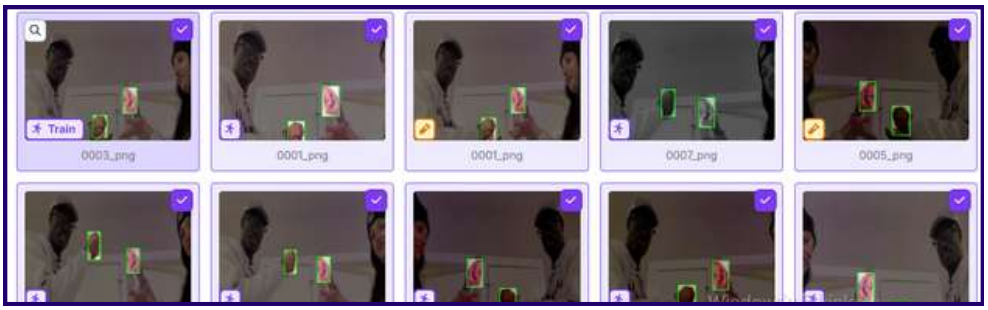
След това кликнете върху бутона "Download Dataset" и изберете "Train" от част от набора от данни, след което "Fork Dataset", за да прехвърлите всички етикети и разделите за обучение, валидиране и тестване на набора от данни като нов проект.



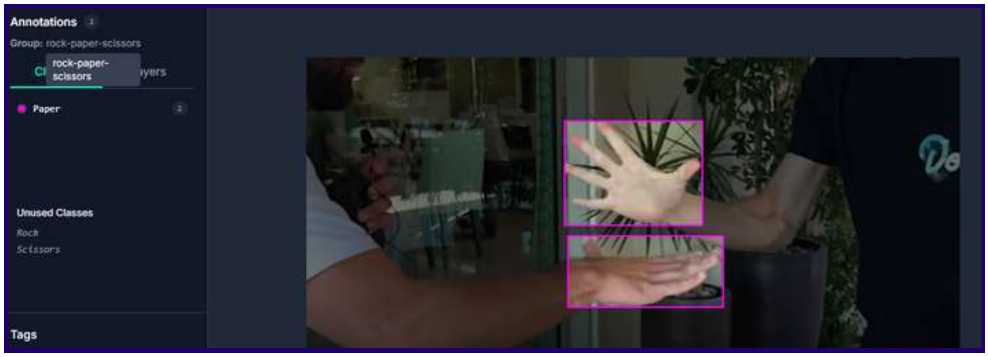
Фигура 13 Dataset (2025) Източник: www.roboflow.com; Дата на добавяне : 09.08.2025.

Python код

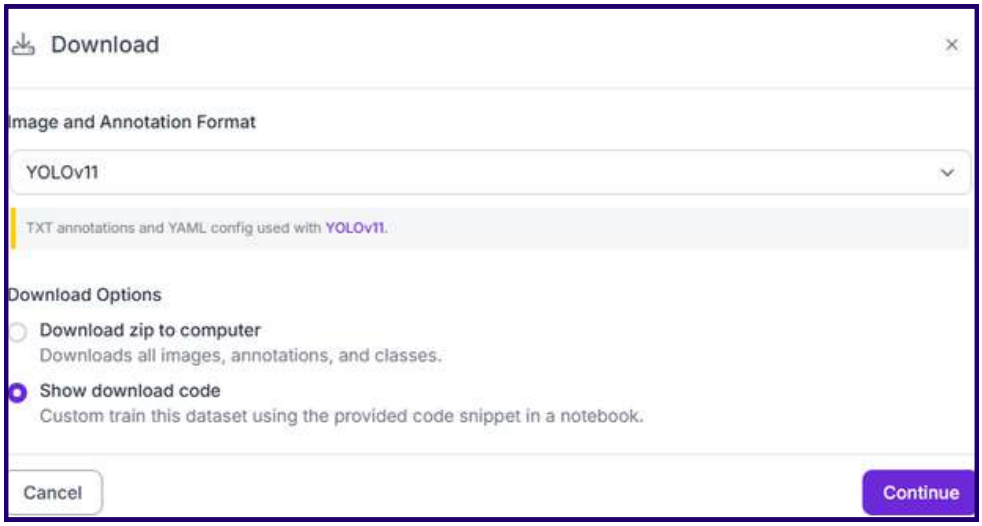
След маркирането и проверката на данните в набора от данни кликнете върху бутона за изтегляне на набора от данни, изберете модела и онлайн метода и вземете Python кода, предоставен от сайта.



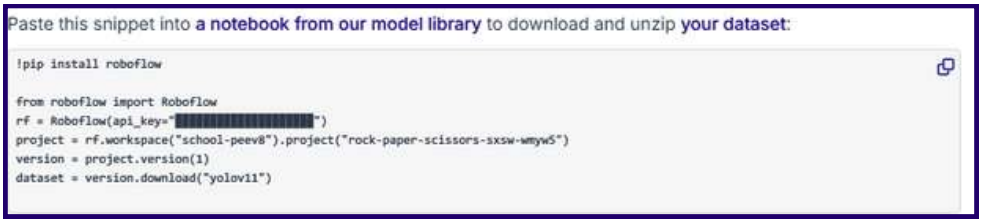
Фигура 14 Dataset (2025). Източник: www.roboflow.com; Дата на добавяне: 09.08.2025.



Фигура 15 . Cut-off (2025) Източник: www.roboflow.com; Дата на добавяне: 09.08.2025.



Фигура 16 YOLOv11 (2025)л Източник: www.roboflow.com; Дата на добавяне: 09.08.2025.



Фигура 17 Snippet (2025). Източник: www.roboflow.com; Дата на добавяне: 09.08.2025

Алгоритъмът YOLOv11

В следващата стъпка ще отворим кода за алгоритъма YOLOv11, работещ в Google Colab, подготвен за нас от Roboflow.
https://blog.roboflow.com/yolov11-how-to-train-custom-data/

Посетете следната връзка и следвайте инструкциите там:
https://colab.research.google.com/github/roboflow-ai/notebooks/blob/main/notebooks/train-yolo11-object-detection-on-custom-dataset.ipynb?ref=blog.roboflow.com

Друга възможност е да създадете модел и да го използвате директно в Roboflow. Това може да е по-бързо и лесно, но YOLOv11 дава по-добри и по-точни резултати.

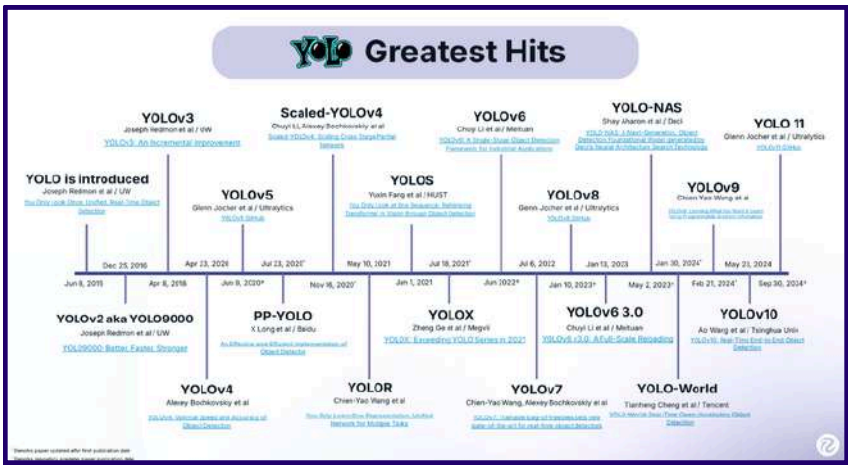
Look Only Once (YOLO) - Гледай само веднъж

През последните няколко години две архитектури за разпознаване на обекти завладяха света на компютърното зрение: конволюционните невронни мрежи (CNN) и Look Only Once (YOLO). И CNN, и YOLO са допринесли за оформянето на съвременната индустрия за компютърно зрение. Трансформерите също играят все по-важна роля в компютърното зрение и разпознаването на обекти.

CNN използват конволюции - техника за обработка на изображения -за усвояване на характеристиките на изображение. Този процес включва прилагане на плъзгащ прозорец към всеки пиксел в изображението за усвояване на характеристиките. Информацията, получена от конволюциите, се обработва от невронна мрежа. Съществуват много реализации на CNN, включително R-CNN, Mask R-CNN и Fast R-CNN.

Семейството от модели YOLO също оказа значително влияние в света на компютърното зрение. Представен от Джоузеф Редмън през 2014 г., YOLO се превърна в активна област на изследване и разработка, беше приет от широка общност и е внедрен от много разработчици и изследователи. YOLOv5 и YOLOv8, разработени и усъвършенствани от екипа на Ultralytics, захранват производствени модели за разпознаване на обекти по целия свят.

Повече информация за YOLO можете да намерите на:
https://blog.roboflow.com/guide-to-yolo-models



Фигура 18 YOLO - История (2025) Източник: www.roboflow.com; Дата на добавяне: 09.08.2025.

За да анализирате подробното класиране на алгоритмите, активно използвани в областта на компютърното зрение по целия свят, можете да посетите: https://leaderboard.roboflow.com/?ref=blog.roboflow.com

ИЗТОЧНИЦИ

1. Banerjee, Chayan & Fookes, Clinton & Karniadakis, George. (2023). Physics-Informed Computer Vision: A Review and Perspectives. 10.48550/arXiv.2305.18035.

2. LeCun, Y. (2025). MNIST a Remote View of the Dataset from http://yann.lecun.com/exdb/mnist/. DateAdd: 09.08.2025.

3. Mohite, Amruta & Kulkarni, Atharva & Chitnis, Rutwik & Mane, Swapnil & Asabe, Shubham. (2021). AI Inspection: Computer Vision For Visual Inspection. International Journal of Advance Research in Computer Science and Management. 7. 29.

4. Object Recognition (2025). From https://viso.ai/product/computer-vision-parking-lot-occupancy-tutorial/ DateAdd: 10.08.2025.

5. Using neural networks for video surveillance (2025) from https://www.videonet9.com/using-neural-networks-for-video-surveillance.html. DateAdd: 09.08.2025.

ЛИНКОВЕ КЪМ УЧЕБНИ ПЛАТФОРМИ

Подробно класиране на алгоритми за компютърно зрение: За анализиране на активно използваните алгоритми в областта на компютърното зрение по целия свят: https://leaderboard.roboflow.com/?ref=blog.roboflow.com

Roboflow Цялостна платформа за изкуствен интелект, предназначена за разработване на проекти за компютърно зрение: https://roboflow.com/

ПРАКТИЧЕСКО УПРАЖНЕНИЕ

ЛОВЦИ НА ОБЕКТИ – РЪКОВОДСТВО ЗА УЧИТЕЛЯ

Цел на упражнението

- Да се разбере логиката на компютърното зрение.
- Да се създаде модел за разпознаване на обекти с помощта на Roboflow.
- Да се преживеят процесите на събиране на данни, маркиране, обучение на модел и тестване.

1. Introduce students to the concept of computer vision with a short presentation.
2. Groups select two different objects and take 15-20 photos from different angles.
3. Log in to Roboflow and create a new project.
4. Photos are uploaded, and objects are labeled by drawing frames.
5. The model is trained and tested.
6. Groups test each other's models, and scores are given.

Идеи за геймификация

- Надпревара с времето: Кой може да събере и маркира данните най-бързо?
- Смесен тест: Моделът се тества с обекти, които никога не е виждал.
- Лов за грешки: Намират се и се обсъждат случаите, в които моделът греши.

ЛОВЦИ НА ОБЕКТИ – НАСОКИ ЗА УЧЕНИКА

Какво е компютърно зрение?

Компютърното зрение е областта на изкуствения интелект, позволяваща на компютрите да откриват и разбират обекти с помощта на камери или изображения.

Вашата задача

1. Вашият отбор трябва да избере два обекта.
2. Направете 15-20 снимки на всеки обект от различни ъгли.
3. Отворете проект в Roboflow и качете снимките.
4. Маркирайте обектите.
5. Обучете модела си.
6. Опитайте да разпознаете обектите на другите отбори.

Съвети за снимане

- Снимайте от различни ъгли.
- Опитайте различни условия на осветление.
- Снимайте обекта и отблизо, и от разстояние.

Насоки за етикетиране

Очертайте обекта на снимката с правоъгълник и напишете неговото наименование.
Пример: Молив, Гума.

Направете следните наблюдения

- Ситуации, в които моделът ви разпознава най-добре.
- Ситуации, с които моделът ви се затруднява.

РЕФЛЕКТИВНИ ВЪПРОСИ

Отделете време, за да помислите задълбочено върху наученото в този модул. Отговорете обмислено на следните въпроси:

- **От пиксели до възприятие:** В този модул научихте, че компютрите възприемат изображенията като мрежи от числа и двоичен код, а не като форми и цветове. Как това "числово" виждане на света променя разбирането ви за начина, по който ИИ различава два различни обекта, като например молив и гума?

- **Значението на разнообразието в данните:** Упражнението "Ловци на обекти" наблегна на снимането от различни ъгли и при различни условия на осветление. Защо това разнообразие в набора от данни за обучение е от ключово значение за точното разпознаване на обект от ИИ в реална среда?

- **Анализиране на точността и "грешките":** По време на тестването бяхте насърчени да идентифицирате случаите, в които моделът греши. Въз основа на наблюденията си, кои конкретни фактори на средата според вас (като сенки или разстояние), затрудниха правилното функциониране на модела?

- **Решаване на реални проблеми:** Компютърното зрение вече се използва в области като здравеопазване, автономни превозни средства и земеделие. Сега, след като сте обучили собствен модел, посочете един конкретен проблем в училището или местната общност, за чието решаване смятате, че може да бъде използвана тази технология.

ТЕСТ (всеки въпрос има само един верен отговор)

1. Каква е основната цел на компютърното зрение?

a) Да позволи на компютрите да мислят като човешки мозъци.
b) Да позволи на машините да "виждат" и интерпретират света като човешкото око.
c) Да съхранява големи набори от данни в бази данни.
d) Да анализира само текстови данни.

2. Кое от следните НЕ е типично приложение на компютърното зрение?

a) Разпознаване на пътни знаци в автономни превозни средства.
b) Системи за разпознаване на лица.
c) Анализиране на потребителските предпочитания в системи за препоръки на продукти.
d) Откриване на тумори в медицински изображения.

3. Каква е задачата на компютърното зрение, насочена към определяне на местоположението и вида на конкретни обекти в изображение?

a) Класификация на изображения
b) Разпознаване на обекти
c) Сегментиране на изображения
d) Извличане на характеристики

ТЕСТ (всеки въпрос има само един верен отговор)

4. Кое от следните е едно от предизвикателствата, пред които са изправени системите за компютърно зрение в реалния свят?

a) Недостиг на данни и разнообразие на тези данни.
b) Ниска размерност и простота на визуалните данни.
c) Последователност в осветлението, позицията и мащаба.
d) Моделите постоянно постигат напълно точни резултати.

5. Кое от следните твърдения показва, че моделът е претрениран (overfitting)?

a) Ниска точност и върху данните за обучение, и върху тестовите данни.
b) Много висока точност върху данните за обучение, ниска точност върху тестовите данни.
c) Висока точност и върху данните за обучение, и върху тестовите данни.
d) Моделът маркира всяко изображение по един и същ начин.

Обобщение и следващи стъпки

Надграждайки върху основните знания, изградени в Курс 1, тези учебни материали са създадени, за да потопят учениците от средните училища в практическото приложение на изкуствения интелект. Съдържанието на модулите цели да затвърди връзката между теорията и практиката, като насочва учениците към създаването на достъпни ИИ модели с помощта на Python и Scratch. От изследването на математическата логика зад вземането на решения в „Стая за бягство с изкуствен интелект" до обучението на модели за машинно обучение за игрови персонажи и разработването на приложения за компютърно зрение с Roboflow — модулите предлагат цялостно, практически ориентирано учебно преживяване. Разказният подход, включващ близки до учениците персонажи като Айлин и Алара, цели да превърне сложни технически концепции, като невронни мрежи, K-средни стойности и разпознаване на обекти в по-достъпни и ангажиращи.

След завършването на курса учениците ще се превърнат от пасивни потребители към активни създатели на технологии. Те ще придобият технически умения за програмиране на основни интелигентни системи, ще развият критично мислене при оценяване на точността на моделите и ще изградят етична осъзнатост за отговорно прилагане на изкуствения интелект. Това задълбочено навлизане в програмирането и в началните стъпки на науката за данните не само затвърждава STEM компетентностите на учениците, но и ги подготвя за бъдеща академична и професионална реализация в свят, движен от технологиите.

Какво следва?

За да се гарантира устойчивото интегриране на тези теми в учебния процес, проектът включва „Инструментариум за учители в средните училища". Този материал ще предостави на педагозите подробни планове на уроци и методически насоки за ефективно преподаване на изкуствен интелект в класната стая. Успоредно с това учениците се насърчават да представят новопридобитите си умения, като се включат в предстоящите конкурси „AI Future Explorers", където могат да прилагат знанията си за решаване на реални проблеми и да се свързват с по-широка общност от млади новатори.

Партньори по проекта

FUTURE-STEM-HUB

Координатор на проекта:

Университетът на Дуисбург-Есен (Германия)

Имейл адрес

mustafa.bilgin@uni-due.de

Уебсайт

www.future-stem-hub.eu

Co-funded by the European Union

Финансирано от Европейския съюз. Изразените възгледи и мнения обаче принадлежат изцяло на техния(ите) автор(и) и не отразяват непременно възгледите и мненията на Европейския съюз или на Европейската изпълнителна агенция за образование и култура (EACEA). За тях не носи отговорност нито Европейският съюз, нито EACEA.