

Ваш корисний гайд

"Практика написання коду Python за допомогою AI"

Вступ до штучного інтелекту

Штучний інтелект (ШІ) є галуззю комп'ютерних наук. Ця галузь займається створенням систем, які можуть виконувати завдання, що зазвичай вимагають людського інтелекту. Це включає машинне навчання (МН), обробку природної мови (NLP), розпізнавання зображень і багато іншого.

Машинне навчання — це підгалузь ШІ, що зосереджується на розробці алгоритмів, які можуть навчатися з даних і покращуватися з часом. Ось деякі основні поняття:

- Модель: статистичний або математичний представник даних.
- Навчання: процес, у якому модель аналізує дані та вдосконалюється.
- Перевірка: оцінка якості моделі на нових даних.

Але не потрібно забувати, що штучний інтелект — це технологія, яка надає машинам здатність навчатися та приймати рішення, імітуючи людський інтелект. ШІ використовує алгоритми для аналізу великих обсягів даних і побудови моделей, що можуть прогнозувати результати або виконувати автоматизовані завдання.

Основними підходами до ШІ є машинне навчання, глибинне навчання, нейронні мережі та обробка природної мови, про які ми говорили раніше. Ці підходи дозволяють створювати моделі для розпізнавання зображень, голосу, передбачення трендів та іншого.

Чому Python є ключовим елементом для штучного інтелекту?

Python став однією з найпопулярніших мов програмування для роботи зі штучним інтелектом завдяки своїй простоті та гнучкості. Python має широкий спектр бібліотек та фреймворків, спеціально розроблених для задач машинного навчання та ШІ, як-от TensorFlow, Keras, PyTorch і Scikit-learn.

Його зрозумілий синтаксис дозволяє швидко прототипувати моделі та працювати з великими даними. Крім того, Python підтримує інтеграцію з іншими мовами та середовищами, що робить його універсальним інструментом для розробників ШІ.

ШІ вже зараз присутній у нашому повсякденному житті, хоча багато хто цього не усвідомлює.

Від автоматизованих рекомендацій на платформах на кшталт Netflix до розпізнавання обличчя на смартфонах — ці технології використовують алгоритми ШІ для поліпшення користувацького досвіду.

Наприклад, розумні голосові асистенти (Siri, Alexa) використовують обробку природної мови, яка є одним із підрозділів ШІ. Вивчення цих технологій стає важливим для сучасних Python-розробників.

Одним із ключових аспектів ШІ є машинне навчання, яке передбачає навчання моделей на основі даних для автоматичного прийняття рішень. **Глибинне навчання** — це підрозділ машинного навчання, який використовує нейронні мережі для моделювання складних патернів у даних. Ці моделі особливо ефективні для розпізнавання образів, тексту та мови.

Python пропонує бібліотеки, як-от TensorFlow та PyTorch, для реалізації цих підходів, надаючи розробникам доступ до сучасних методів створення моделей.

Python має величезну екосистему інструментів для ШІ та машинного навчання, які роблять роботу над проектами швидшою та ефективнішою. Бібліотеки, як-от Pandas і NumPy, допомагають у роботі з великими обсягами даних, а TensorFlow, Keras і PyTorch полегшують побудову та тренування моделей. Завдяки своїй простоті, Python дозволяє навіть початківцям швидко опанувати основи ШІ та створювати робочі прототипи моделей.

Але де ж можна використовувати Python та у яких цілях? Читайте нижче 

Ось декілька прикладів:

- 1. Обробка природної мови (NLP).** Python використовується в таких додатках, як чатботи або голосові помічники (наприклад, Siri), для обробки тексту або голосу за допомогою бібліотек, як NLTK або SpaCy. Вони дозволяють аналізувати тексти, перекладати мови та визначати емоційне забарвлення повідомлень.
- 2. Рекомендаційні системи.** Python застосовується для побудови рекомендаційних систем, як на Netflix або Amazon, використовуючи бібліотеки машинного навчання, наприклад Scikit-learn. Вони допомагають прогнозувати, що користувач захоче подивитися або купити, на основі минулої поведінки.
- 3. Комп'ютерний зір.** Python використовується в задачах розпізнавання зображень і відео за допомогою бібліотек OpenCV та TensorFlow. Наприклад, технологія розпізнавання облич або автономні автомобілі використовують моделі Python для аналізу відеопотоків у реальному часі.
- 4. Глибинне навчання.** Python є основною мовою для розробки нейронних мереж у глибинному навчанні через бібліотеки, як-от Keras і PyTorch. Наприклад, його використовують для навчання моделей, що генерують текст, створюють зображення або розв'язують завдання класифікації.
- 5. Розпізнавання мови та голосу.** Python дозволяє створювати системи розпізнавання мови, які застосовуються в голосових помічниках або для перетворення голосових команд у текст. Бібліотеки, як-от SpeechRecognition та DeepSpeech, допомагають розробляти ці рішення.

Тепер, коли ми розглянули теоретичні основи використання Python для ШІ, час перейти до практичної частини. Практика допоможе вам краще зрозуміти, як використовувати інструменти та бібліотеки Python для розв'язання реальних задач у сфері штучного інтелекту.

Ми почнемо з простих прикладів, які демонструють, як створювати моделі машинного навчання, обробляти дані, працювати з нейронними мережами та розгорнати ШІ-моделі в проектах.

Готові дізнатися, як ці теорії працюють на практиці? Переходимо до завдань!

ПРАКТИЧНІ ЗАВДАННЯ

У цьому гайді ви зможете попрактикуватись у написанні коду та створити мініпроєкти.

Для виконання цих завдань ви можете використовувати Google Colab, що є онлайн-платформою для виконання Python-коду. Google Colab забезпечує безкоштовний доступ до потужних апаратних ресурсів, як-от GPU, що може бути корисним для обчислювальних завдань у машинному навчанні.

Відкриття Colab:

1. Перейдіть на <https://colab.research.google.com/>.
2. Натисніть "New Notebook" для створення нового ноутбука або "Upload" для завантаження наявного.

Класифікація ірисів Фішера

На наступній сторінці ви отримаєте детальний код, що допоможе запустити роботу програми.

Мета: навчитися класифікувати різні види ірисів на основі їхніх фізичних характеристик.

Завдання: використовуючи бібліотеку `scikit-learn`, натренуйте модель машинного навчання для визначення виду ірису на основі наданих даних.

```
from sklearn.datasets import load_iris  
from sklearn.model_selection import train_test_split  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.metrics import accuracy_score  
import matplotlib.pyplot as plt  
import seaborn as sns
```

Завантаження даних

```
data = load_iris()  
X = data.data  
y = data.target
```

Розділення даних на навчальний та тестовий набори

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,  
random_state=42)
```

Навчання моделі

```
model = RandomForestClassifier()  
model.fit(X_train, y_train)
```

Прогнозування та оцінка

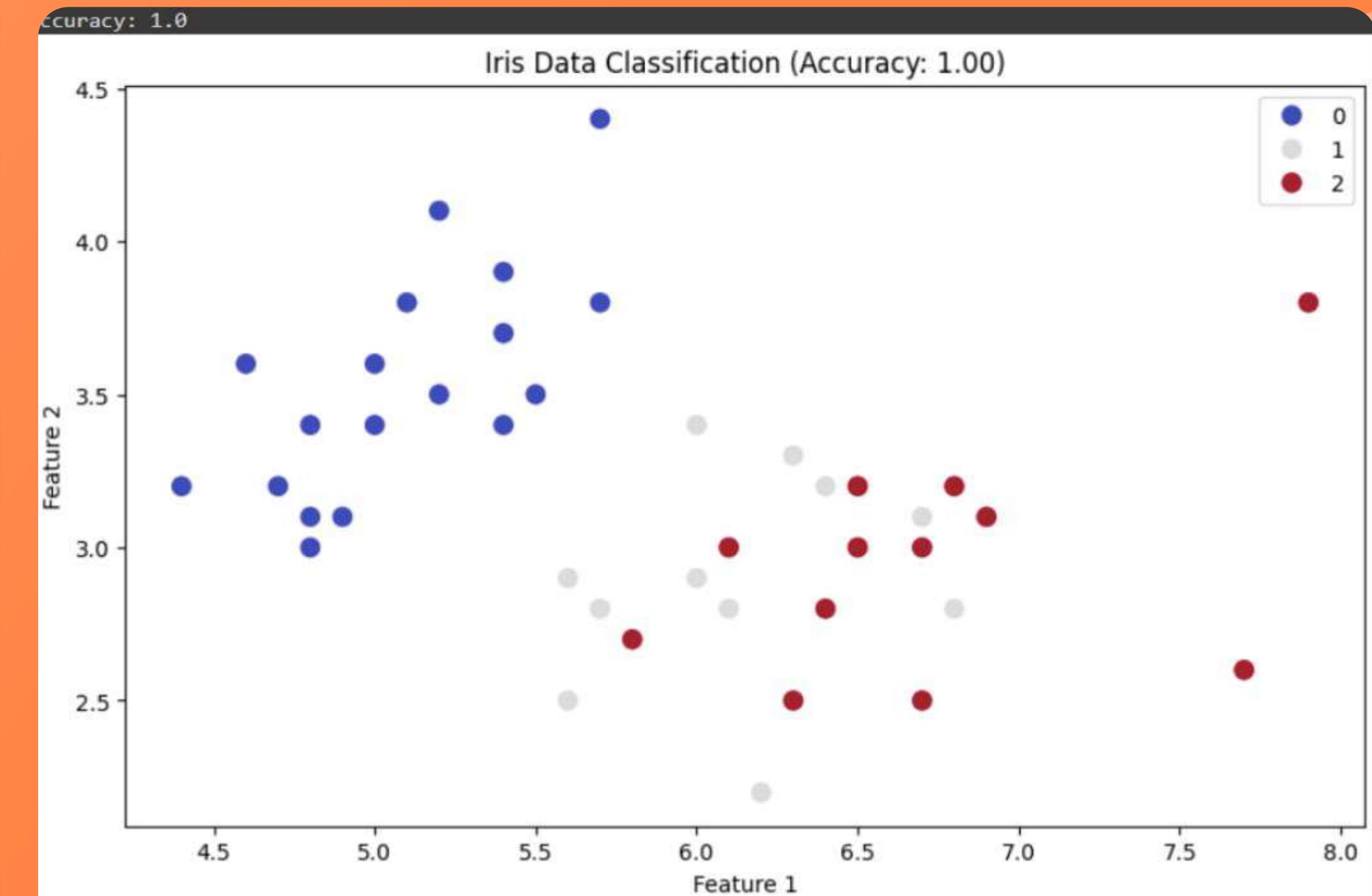
```
y_pred = model.predict(X_test)  
accuracy = accuracy_score(y_test, y_pred)  
print(f'Accuracy: {accuracy}')
```

Візуалізація результатів

```
plt.figure(figsize=(10, 6))  
sns.scatterplot(x=X_test[:, 0], y=X_test[:, 1], hue=y_pred,  
palette='coolwarm', s=100)  
plt.title(f'Iris Data Classification (Accuracy: {accuracy:.2f})')  
plt.xlabel('Feature 1')  
plt.ylabel('Feature 2')  
plt.show()
```

Опис візуалізації

- Розсіювальний графік відображає перші дві ознаки з тестових даних набору Iris.
- Кожна точка представляє один зразок з тестового набору, а колір точки відповідає прогнозованій моделі мітки (класу квітки).
- Заголовок графіка показує точність моделі (Accuracy) у відсотках.



Розпізнавання зображень

Мета: опанувати основи нейронних мереж для обробки зображень.

Завдання: створіть та натренуйте просту нейронну мережу для класифікації зображень із набору даних CIFAR-10, використовуючи `TensorFlow` або `Keras`.

```
import tensorflow as tf
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,
Flatten, Conv2D, MaxPooling2D, Dropout
from tensorflow.keras.utils import
to_categorical
import matplotlib.pyplot as plt
import numpy as np
import random
```

```
# Завантаження даних
(X_train, y_train), (X_test, y_test) =
cifar10.load_data()
```

```
# Нормалізація даних
X_train, X_test = X_train / 255.0, X_test / 255.0
```

```
# Перетворення міток в категоріальні дані
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

```
# Побудова моделі
model = Sequential([
    Conv2D(32, (3, 3), activation='relu',
input_shape=(32, 32, 3)),
    MaxPooling2D((2, 2)),
    Dropout(0.25),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Dropout(0.25),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Dropout(0.5),
    Flatten(),
    Dense(512, activation='relu'),
    Dense(10, activation='softmax')
])
```

Компіляція моделі

```
model.compile(optimizer='adam',
loss='categorical_crossentropy',
metrics=['accuracy'])
```

Навчання моделі

```
history = model.fit(X_train, y_train, epochs=10,
validation_data=(X_test, y_test))
```

Вибір випадкових зображень для перевірки

```
indices = random.sample(range(len(X_test)), 5)
images = X_test[indices]
true_labels = y_test[indices]
predictions = model.predict(images)
predicted_labels = np.argmax(predictions,
axis=1)
```

Виведення результатів

```
class_names = ['Airplane', 'Automobile', 'Bird', 'Cat', 'Deer', 'Dog', 'Frog', 'Horse', 'Ship', 'Truck']
```

Налаштування відступів між підграфіками

```
plt.figure(figsize=(8, 5))  
plt.subplots_adjust(wspace=0.5)
```

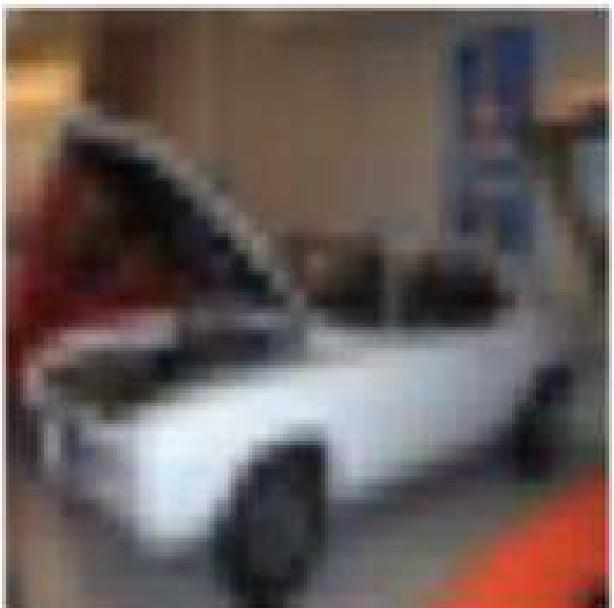
```
for i, idx in enumerate(indices):
```

```
    plt.subplot(1, 5, i+1)  
    plt.imshow(images[i])  
    plt.title(f'True: {class_names[np.argmax(true_labels[i])]}', fontsize=12)  
    plt.xlabel(f'Pred: {class_names[predicted_labels[i]]}', fontsize=12)  
    plt.axis('off')  
  
plt.show()
```

Опис візуалізації

Як результат виконання коду, ми бачимо, що всі зображення розпізнані. Біля кожного з них є чіткий підпис, який відповідає дійсності.

True: Automobile



True: Horse



True: Bird



True: Bird



True: Bird



Створення простого чатботу

Мета: оволодіти базовими навичками створення чатботів для спілкування з користувачами.

Завдання: реалізуйте простий чатбот за допомогою бібліотеки `NLTK`, який зможе відповісти на типові запитання користувачів.

```
import nltk
from nltk.chat.util import Chat, reflections

# Визначення шаблонів і відповідей
pairs = [
    (r'Привіт|Добрий день', ['Привіт! Як я
можу допомогти?']),
    (r'Як справи?', ['Добре, дякую! Як у вас?']),
    (r'Які у тебе хобі?', ['Моє хобі -
допомагати людям.']),
    (r'Вибач, але я йду.', ['До побачення!'])
]
```

```
# Додати універсальну відповідь на
незнайомі запити
def respond(user_input):
    response = chat_bot.respond(user_input)
    if response is None:
        response = "Вибачте, я не зрозумів
вашого питання. Можете задати щось
інше?"
    return response
```

```
# Створення чатботу
chat_bot = Chat(pairs, reflections)
while True:
    user_input = input("Ви: ")
    print("Чат-бот: ", respond(user_input))
```

Результат

```
Ви: Добрий день
Чат-бот: Привіт! Як я можу допомогти?
Ви: Які у тебе хобі?
Чат-бот: Моє хобі - допомагати людям.
Ви: Як тебе звати?
Чат-бот: Вибачте, я не зрозумів вашого питання. Можете задати щось інше?
Ви: 
```

Автозаповнення тексту

Мета: навчитися використовувати моделі автозаповнення тексту для генерації контенту.

Завдання: використовуючи модель GPT-2 через бібліотеку `transformers`, створіть програму, яка передбачає і доповнює текст на основі введених користувачем слів.

```
from transformers import pipeline,  
AutoTokenizer,  
AutoModelForCausalLM  
  
# Завантаження моделі й  
токенізатора  
model_name = "gpt2"  
tokenizer =  
AutoTokenizer.from_pretrained(mod  
el_name)  
model =  
AutoModelForCausalLM.from_pretrai  
ned(model_name)
```

```
# Створення генератора тексту  
generator = pipeline('text-generation', model=model,  
tokenizer=tokenizer)  
def generate_text(prompt, max_length=100):  
    result = generator(  
        prompt,  
        max_length=max_length,  
        num_return_sequences=1,  
        temperature=0.7,  
        top_p=0.9,  
        eos_token_id=tokenizer.encode('.')[0],  
        pad_token_id=tokenizer.pad_token_id,  
        truncation=True  
    )
```

```
generated_text = result[0]
['generated_text']
if '.' in generated_text:
    generated_text =
generated_text.split('.')[0] + '.'
return generated_text

while True:
    prompt = input("Enter the prompt for
text generation (or type 'exit' to quit): ")
    if prompt.lower() == 'exit':
        break
    print("Generated text:")
    print(generate_text(prompt))
    print("\n" + "="*40 + "\n")
```

Результат



```
Enter the prompt for text generation (or type 'exit' to quit): The morning starts with
Setting `pad_token_id` to `eos_token_id`:13 for open-end generation.
Generated text:
The morning starts with a bit of a lull in the night.
=====
```

```
Enter the prompt for text generation (or type 'exit' to quit): Sunshine is
Setting `pad_token_id` to `eos_token_id`:13 for open-end generation.
Generated text:
Sunshine is the only one that can say no to the current incarnation of The New York Times.
=====
```

```
Enter the prompt for text generation (or type 'exit' to quit): exit
```

Чи зможе ШІ повністю замінити Python-розробника?

ШІ може автоматизувати багато завдань, але повна заміна Python-розробників поки що неможлива. Алгоритми ШІ здатні генерувати код, знаходити помилки та оптимізувати процеси, однак творчі рішення, архітектурне планування і складні проекти потребують людської інтуїції та досвіду.

ШІ діє на основі попередніх даних і не має гнучкості до нових ситуацій так, як люди. Тому роль розробників еволюціонує, але не зникає.

Штучний інтелект успішно допомагає Python-розробникам, автоматизуючи рутинні завдання, як-от автодоповнення коду, виявлення багів або оптимізація продуктивності. Інструменти на базі ШІ, як-от GitHub Copilot чи Tabnine, дозволяють скоротити час, витрачений на монотонну роботу.

Проте ці інструменти є лише помічниками і не можуть замінити людську творчість, необхідну для розв'язання складних задач, створення алгоритмів і управління проєктами.

Чи зменшиться кількість робочих місць для розробників?

З розвитком ШІ може зменшитися кількість вакансій для виконання простих і повторюваних завдань, але, водночас, з'являються нові можливості для більш складних і цікавих ролей. Python-розробники, які здатні інтегрувати ШІ у свої робочі процеси, стануть ще більш цінними на ринку праці.

Автоматизація відкриває нові горизонти для створення інноваційних продуктів, що вимагають кваліфікованих фахівців.

Замість повної заміни Python-розробників ШІ стає інструментом, що покращує їхню роботу. Розробники можуть використовувати ШІ для підвищення продуктивності, швидшого пошуку рішень і автоматизації тестування.

Співпраця людини з ШІ допомагає зменшити кількість помилок, підвищує ефективність і дозволяє зосередитися на творчих і стратегічних завданнях. Це дає змогу створювати більш якісні продукти за коротший час.

Тож підсумуймо

Штучний інтелект (ШІ) для Python-розробників **стає важливим інструментом**, що значно полегшує їхню роботу, дозволяючи автоматизувати рутинні завдання, покращувати якість коду та пришвидшувати процеси розробки.

Завдяки потужним бібліотекам, як-от TensorFlow, Keras, PyTorch і Scikit-learn, розробники можуть легко створювати складні моделі машинного навчання, обробляти великі обсяги даних і використовувати інструменти глибинного навчання для розв'язання реальних проблем.

ШІ не тільки підвищує ефективність, але й дає можливість Python-розробникам розвиватися в нових напрямках. Наприклад, з розвитком глибокого навчання та обробки природної мови, Python розширює можливості у таких галузях, як комп'ютерний зір, розпізнавання мови, автоматизація рекомендаційних систем тощо.

Це робить Python універсальним інструментом для створення інтелектуальних рішень і відкриває нові можливості для кар'єрного росту.

З іншого боку, варто зазначити, що хоча ШІ суттєво покращує робочі процеси розробників, **він не може повністю замінити людину.** ШІ може автоматизувати певні аспекти, але складні завдання, що потребують інтуїції, творчості та глибокого розуміння бізнес-контексту, залишаються за розробниками. Крім того, ШІ залежить від якості даних та коректності моделей, тому роль людського втручання в створення та оптимізацію цих моделей залишається ключовою.

Так Python у поєднанні з ШІ стає потужним інструментом для сучасних розробників, надаючи їм нові можливості для автоматизації, покращення якості роботи та розвитку у перспективних сферах, зберігаючи при цьому важливу роль людського контролю та творчого підходу.

Додаткові ресурси

Книги:

- "Практичне машинне навчання з Scikit-Learn, Keras та TensorFlow" — Орелін Жерон
- "Глибинне навчання з Python" — Франсуа Шолле
- "Python та машинне навчання" — Василь Костюк

Онлайн-платформи:

- [[Kaggle](#)] — платформа для змагань з машинного навчання й аналізу даних.
- [[Google Colab](#)] — інструмент для написання та виконання Python-коду з підтримкою GPU.
- Архів коду з бонуса можна отримати тут - [клікайте](#)