ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Методи обробки даних анкетування за допомогою штучного інтелекту

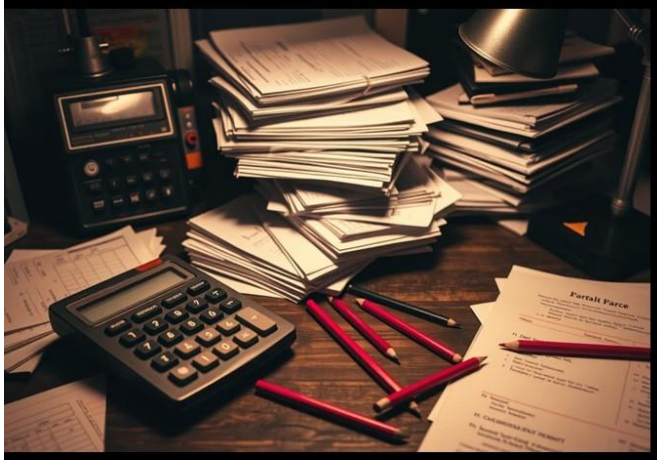
Автор:

Роман ФАТІЙ
студ. Гр СПзм- 23-1

Керівник:

Наталія БОЛОГОВА
доц. каф.ЕОМ

Обмеження традиційних методів аналізу



- **Статистичний аналіз**
  Вимагає попередніх припущень про дані.
- **Ручна обробка**
  Трудомістка, схильна до помилок, не масштабується.
- **Неструктуровані дані**
  Складність аналізу відкритих відповідей.
- **Втрата контексту**
  Недостатнє виявлення нелінійних залежностей.

Вступ до аналізу анкетних даних та штучного інтелекту

У сучасному освітньому просторі обробка великих обсягів анкетних даних є складною та ресурсомісткою задачею. Традиційні методи часто вимагають значних людських зусиль і схильні до помилок, що може уповільнити прийняття рішень та знизити їх ефективність. Штучний інтелект (ШІ), зокрема машине навчання, пропонує потужні рішення для автоматизації цих процесів, забезпечуючи швидший та точніший аналіз.

Метою кваліфікаційної роботи є автоматизація процесу відбору анкетних даних у дистрибутиві Python із використанням алгоритмів машинного навчання. Це дозволить оптимізувати роботу університетів та покращити взаємодію зі студентами.

3

Огляд систем Штучного Інтелекту:

Нейронні Мережі

Імітують структуру людського мозку для розпізнавання складних патернів у даних.

Машинне Навчання

Алгоритми, що дозволяють комп'ютерам навчатися на даних без явного програмування.

Глибоке Навчання

Підмножина машинного навчання, яка використовує багатошарові нейронні мережі для складних завдань.

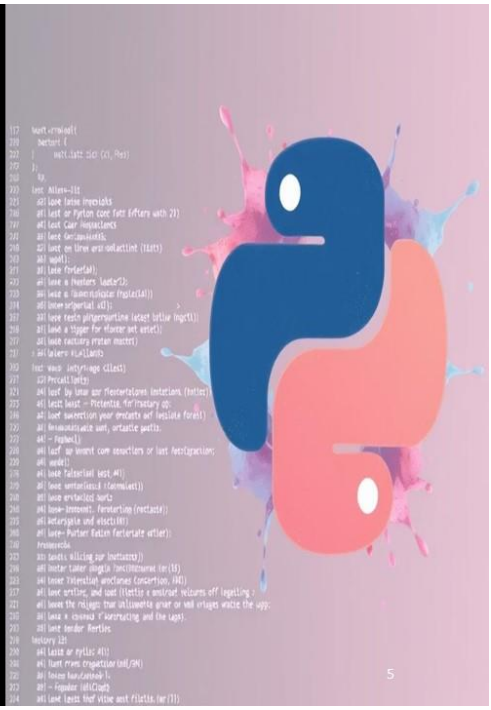
Системи штучного інтелекту охоплюють широкий спектр технологій, призначених для імітації людського інтелекту. Це включає розпізнавання образів, обробку природної мови, прийняття рішень та машинне навчання. Для обробки анкетних даних особливий інтерес представляють нейронні мережі, які здатні виявляти приховані залежності та класифікувати інформацію з високою точністю.

4

Програмне Забезпечення для ШІ: Python

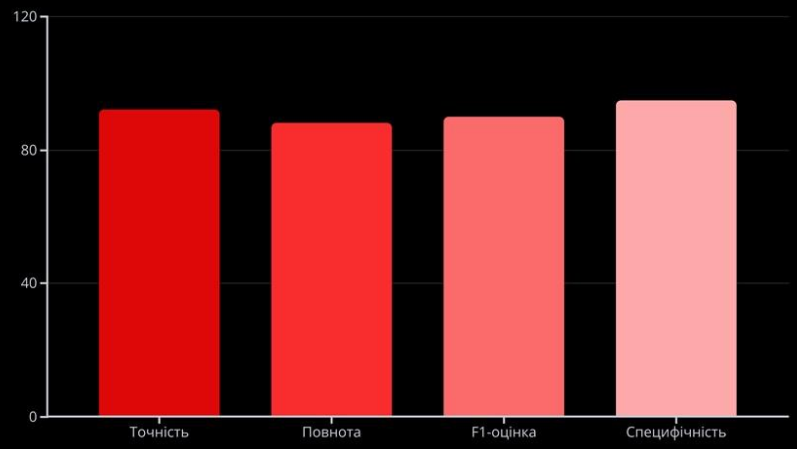
Python є однією з найпопулярніших мов програмування для розробки ШІ завдяки своїй простоті, гнучкості та великій екосистемі бібліотек. Anaconda, у свою чергу, є потужним дистрибутивом Python, який містить усі необхідні інструменти та бібліотеки для наукових обчислень, аналізу даних та машинного навчання, що робить її ідеальним вибором для цього проекту.

Використання Anaconda забезпечує зручне управління залежностями та середовищами, що критично важливо при роботі з комплексними проектами ШІ. Це дозволяє зосередитися на розробці алгоритмів, а не на налаштуванні інфраструктури.

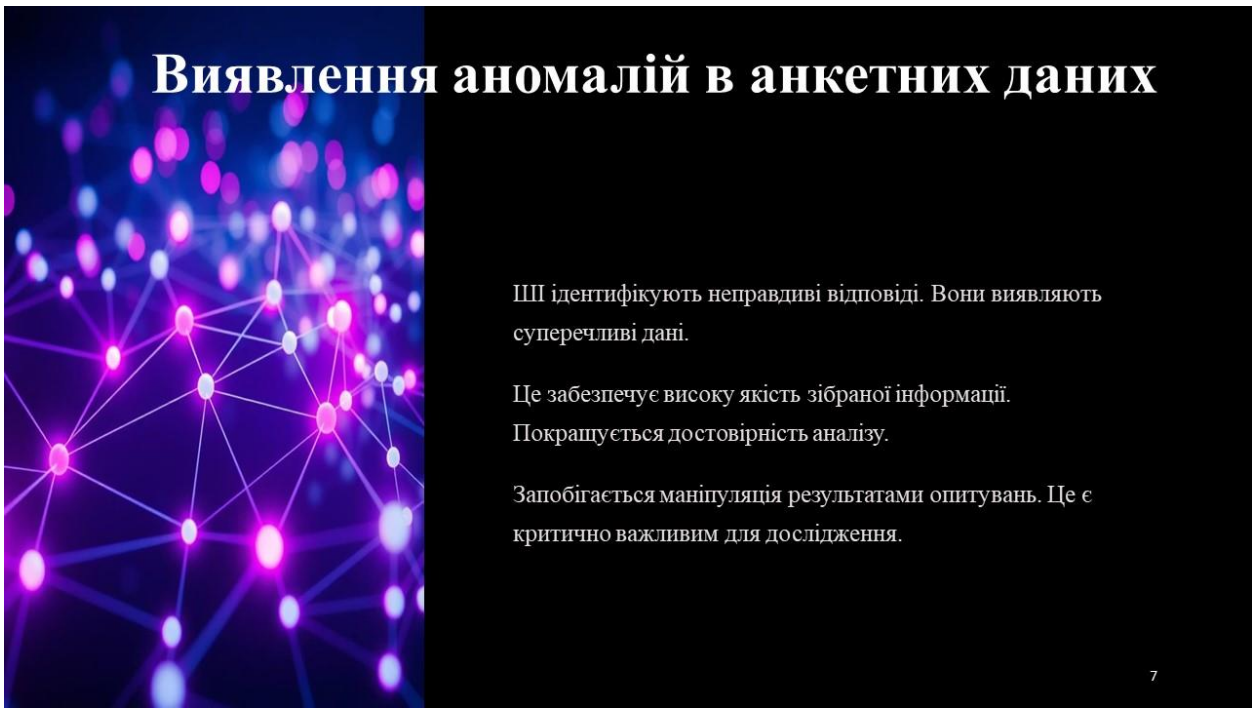


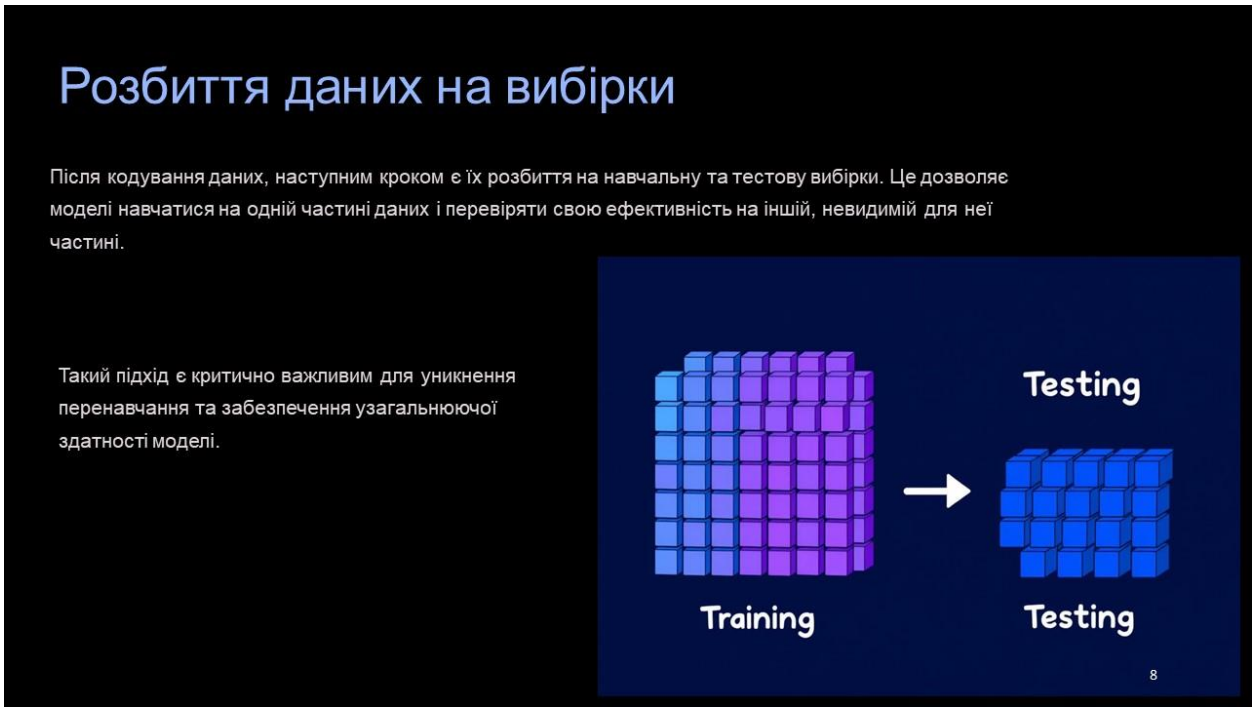
5

Показники ефективності класифікатора



6


Виявлення аномалій в анкетних даних
ШІ ідентифікують неправдиві відповіді. Вони виявляють суперечливі дані.
Це забезпечує високу якість зібраної інформації. Покращується достовірність аналізу.
Запобігається маніпуляція результатами опитувань. Це є критично важливим для дослідження.
7


Розбиття даних на вибірки
Після кодування даних, наступним кроком є їх розбиття на навчальну та тестову вибірки. Це дозволяє моделі навчатися на одній частині даних і перевіряти свою ефективність на іншій, невидимій для неї частині.
Такий підхід є критично важливим для уникнення перенавчання та забезпечення узагальнюючої здатності моделі.
Testing
Training
Testing
8

Метод	Чому підходить для ХНУРЕ
Випадковий ліс	Потужний і стабільний, працює з великими даними. Стійкий до шуму
Логістична регресія	Простий, швидкий. Добре для базових класифікацій і пояснень.
Градієнтний бустинг (LightGBM, XGBoost)	Підходить для великого обсягу. Але складніше у налаштуванні і важче інтерпретувати.
Стохастичний градієнтний спуск	Швидкий для дуже великих даних.
Метод KNN	Не рекомендується через обсяг даних та складність масштабування. Можна використовувати тільки для невеликих підмножин або прототипів.

9

Приклад Коду: Завантаження та Кодування

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# X – ознаки (features), y – мітки (labels)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Створення моделі логістичної регресії
model = LogisticRegression()

# Навчання моделі
model.fit(X_train, y_train)

# Прогнозування результатів на тестовій вибірці
y_pred = model.predict(X_test)

# Обчислення точності класифікації
accuracy = accuracy_score(y_test, y_pred)
print("Точність логістичної регресії:", accuracy)
```

Реалізація **логістичної регресії**

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Розділення на тренувальні та тестові вибірки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Ініціалізація моделі вирішального дерева
tree_model = DecisionTreeClassifier(random_state=42)

# Навчання моделі
tree_model.fit(X_train, y_train)

# Передбачення результатів
y_pred = tree_model.predict(X_test)

# Оцінка точності класифікації
accuracy = accuracy_score(y_test, y_pred)
print("Точність вирішального дерева:", accuracy)
```

Реалізація алгоритму вирішального дерева

10

Навчання Моделі Random Forest

Вибір Моделі

Для класифікації використовується модель **Random Forest**, яка відома своєю високою точністю та стійкістю до перенавчання.

Модель навчається на підготовленій навчальній вибірці, вивчаючи закономірності та взаємозв'язки в даних.

11

Приклад Коду: Завантаження та Кодування

```python
import numpy as np
import pandas as pd

# Вхідні дані
# Кількість студентів за оцінками
data = {
    'score': [3, 4, None, 5],   # None - пропуски
    'count': [30, 40, 3, 25]
}

# Створюємо датафрейм
df = pd.DataFrame(data)

# Загальна кількість анкет
total_surveys = df['count'].sum()

# Кількість валідних анкет (без пропусків)
valid_surveys = df[df['score'].notnull()]['count'].sum()

# Обчислюємо середній бал за валідними анкетами
# Помножуємо кожен бал на кількість студентів, сумуємо і ділимо на кількість валідних анкет
weighted_sum = (df[df['score'].notnull()]['score'] * df[df['score'].notnull()]['count']).sum()
average_score = weighted_sum / valid_surveys

print(f"Загальна кількість анкет: {total_surveys}")
print(f"Кількість валідних анкет (без пропусків): {valid_surveys}")
print(f"Середній бал за якість освіти (за валідними анкетами): {average_score:.2f}")
```

Приклад коду на Python із використанням **Random Forest**

```python
import pandas as pd

# Вхідні дані: оцінка -> кількість студентів
data = {
    'score': [3, 4, None, 5],   # None — пропуск
    'count': [30, 40, 3, 25]
}

df = pd.DataFrame(data)

# Кількість валідних спостережень
valid_df =
```

Код для розрахунку
(з обробкою пропусків та імпутацією)

12

ВИСНОВКИ

- Представлені алгоритми охоплюють повний цикл машинного навчання: від підготовки даних до оцінки моделі.
- Проект із впровадження машинного навчання для аналізу анкетних даних в освітніх установах демонструє високу ефективність та значний потенціал для трансформації традиційних процесів.
- Використання Python та Anaconda забезпечує надійну та масштабовану платформу.
- Автоматизація обробки даних значно підвищує ефективність та точність.
- Проект має чітку обґрунтованість, що підтверджується очікуваним зниженням витрат та підвищенням швидкості процесів.
- Аналіз важливості ознак дозволяє глибоко зрозуміти вплив різних параметрів на результати.
- Модель Random Forest є потужним інструментом класифікації, забезпечуючи високу точність.