Experiment 12: Real World Case Studies

Part 1: Case Studies – Theory

Case Study 1: A Course Advisor Expert System

Theory:

A Course Advisor Expert System is a rule-based expert system designed to help students choose the right course based on their interests, academic background, and career goals. It works by asking users a series of questions and matching their responses against a knowledge base of predefined rules. The system mimics the behavior of a human academic advisor.

Key Features:

* Uses if-else or rule-based logic to provide course suggestions.
* Can be extended using machine learning or recommendation algorithms.
* Helps students make informed decisions in course selection.

Example Rules:

* If the student is interested in design → suggest “UI/UX Design”
* If the student is from a computer science background and interested in AI → suggest “Machine Learning”

Case Study 2: Evaluation of Medical Expert Systems

Theory:

Medical Expert Systems are AI-based tools used to support clinical decision-making. Evaluating their performance is important to ensure safety, accuracy, and reliability in healthcare.

Performance Assessment Methods:

* Accuracy: Percentage of correct predictions made by the system.
* Sensitivity and Specificity: Measures how well the system detects diseases.
* User Feedback: Validation from real doctors or medical professionals.
* Comparison with clinical outcomes: Testing how well predictions align with actual diagnoses.

Tools used: Confusion matrix, ROC curves, precision, recall, and F1-score.

Purpose:

* To ensure medical expert systems make safe and correct recommendations.
* To analyze limitations and risks in automated healthcare.

Part 2: Python Code

1. Sentiment Analysis using NLP

Theory:

Sentiment analysis is used to determine whether a piece of text expresses a positive, negative, or neutral opinion. It's widely used in analyzing customer feedback, product reviews, and social media content.

Code:

import nltk from nltk.sentiment.vader import SentimentIntensityAnalyzer nltk.download('vader\_lexicon')

**Sample text**

text = "I really enjoyed the new course on Artificial Intelligence!"

**Initialize sentiment analyzer**

sid = SentimentIntensityAnalyzer() scores = sid.polarity\_scores(text)

**Display sentiment scores**

print("Sentiment Scores:", scores)

**Classify overall sentiment**

if scores['compound'] > 0: print("Overall Sentiment: Positive") elif scores['compound'] < 0: print("Overall Sentiment: Negative") else: print("Overall Sentiment: Neutral")

1. Fake News Detection using NLP

Theory:

Fake News Detection is the process of identifying misleading or false content in news articles using natural language processing (NLP) and machine learning. We use text preprocessing, TF-IDF for feature extraction, and a classifier like Logistic Regression to detect fake news.

Code:

import pandas as pd from sklearn.model\_selection import train\_test\_split from sklearn.feature\_extraction.text import TfidfVectorizer from sklearn.linear\_model import LogisticRegression from sklearn.metrics import classification\_report

**Sample dataset**

data = { 'text': [ 'COVID-19 vaccine causes magnetism, claim reports', 'Government launches AI training for students', 'Aliens spotted in New York, says anonymous source', 'New cancer treatment shows 95% success rate' ], 'label': [1, 0, 1, 0] # 1 = Fake, 0 = Real } df = pd.DataFrame(data)

**Preprocessing and vectorization**

tfidf = TfidfVectorizer() X = tfidf.fit\_transform(df['text']) y = df['label']

**Train-test split**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3)

**Train classifier**

model = LogisticRegression() model.fit(X\_train, y\_train)

**Predict and evaluate**

y\_pred = model.predict(X\_test) print(classification\_report(y\_test, y\_pred))