AI Tools & Applications - Week 3 Report

By: Nimo Hussein - AI Tools Project

Part 1: Theoretical Understanding

Q1: TensorFlow vs PyTorch

TensorFlow uses static graphs, ideal for deployment. PyTorch uses dynamic graphs, better for research. TensorFlow is preferred for production; PyTorch is great for experimentation.

Q2: Jupyter Notebook Use Cases

- Interactive data analysis and model prototyping
- Reproducible research and teaching notebooks

Q3: spaCy Advantages

spaCy is optimized for fast, accurate NLP tasks. Unlike basic Python string methods, it uses trained pipelines for tokenization, NER, POS tagging, etc.

Comparative Analysis: Scikit-learn vs TensorFlow

Scikit-learn: classical ML, beginner-friendly, less code.

TensorFlow: deep learning, high scalability, more complex but powerful.

Part 2: Practical Implementation

GitHub Notebooks Submitted:

- Iris Classifier (Scikit-learn): https://github.com/Nim-creator/week3-ai-ml-nlp/blob/main/iris_clasifier_week3.ipynb
- MNIST CNN (TensorFlow): https://github.com/Nim-creator/week3-ai-ml-nlp/blob/main/mnist_cnn_week3.ipynb
- spaCy NER + Sentiment:

https://github.com/Nim-creator/week3-ai-ml-nlp/blob/main/spacy_ner_sentiment_week3.ipynb

Part 3: Ethics & Optimization

Ethical Concerns:

- Bias in MNIST: Some digit styles may not generalize (e.g., handwritten 7s).

- Amazon Reviews: Sentiment rules may miss sarcasm or context.

Mitigation Tools:

- Use more diverse training data.

- Apply spaCy rule-based checks.

- Use fairness libraries like TensorFlow Fairness Indicators.

Debugging Notes:

- Verified CNN model shapes using model.summary()

- Ensured correct activation/loss (softmax + categorical_crossentropy)

- spaCy NER correctly extracted entities using pre-trained models.