AI Tools and Frameworks Assignment Report

Hillan Munene

21.10.2025

Part 1: Theoretical Understanding (30%)

1. Short Answer Questions

Q1: Primary Differences between TensorFlow and PyTorch

Feature	TensorFlow (Keras)	PyTorch
Computation Graph	**Static/Eager:** Historically static (defined first, run later), now defaults to Eager (dynamic).	**Dynamic (Eager):** Defined and executed on the fly ("define-by-run").
Debugging	More complex with static graphs; simple with Eager execution.	**More Pythonic**, allowing use of standard Python debuggers (like PDB).
Deployment	**Stronger:** Excellent tools for production (TensorFlow Serving, TF Lite for mobile/edge).	**Improving:** Uses TorchScript for serialization, but the ecosystem is newer than TF's.
When to Choose	**Production, Scalability, and Mobile deployment.**	**Research, Rapid Prototyping, and highly custom models.**

Q2: Two Use Cases for Jupyter Notebooks in AI Development

1. **Exploratory Data Analysis (EDA) and Preprocessing:** Notebooks allow for iterative loading of data, visualization of distributions using libraries like Matplotlib, and step-by-step cleaning or transformation (e.g., normalization using Scikit-learn). The blend of code, output, and explanatory markdown makes the data pipeline transparent.
2. **Model Prototyping and Evaluation:** They serve as a quick sandbox for defining model architectures (TensorFlow/Keras or PyTorch), running short training

loops, and displaying immediate results (e.g., live accuracy graphs and sample predictions) before moving to larger, production-ready scripts.

Q3: How does spaCy enhance NLP tasks compared to basic Python string operations?

spaCy provides a highly optimized, **production-ready NLP pipeline** that offers linguistic intelligence far beyond basic Python string methods (.split(), re module).

- **Linguistic Context:** spaCy uses pre-trained statistical models to perform tasks like **Tokenization, Part-of-Speech (POS) Tagging, and Named Entity Recognition (NER)**. Basic string operations can only deal with characters or fixed patterns, missing the grammatical role and context of words.
- **Efficiency:** spaCy is written in Cython, making it significantly **faster and more memory-efficient** for processing large volumes of text compared to pure Python string manipulation, making it ideal for large-scale production deployment.

2. Comparative Analysis: Scikit-learn and TensorFlow

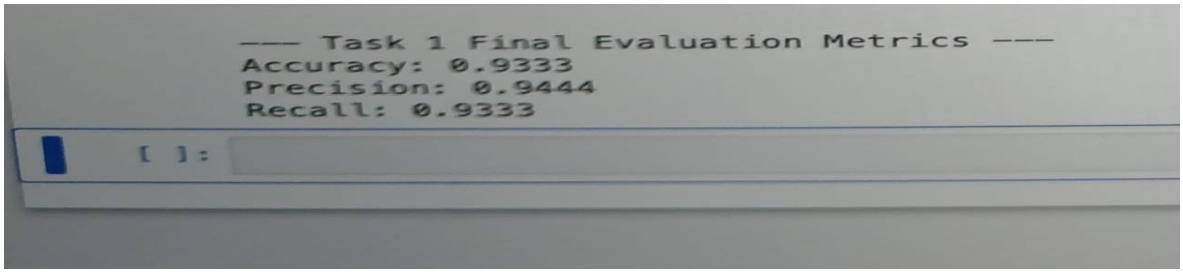
Feature	Scikit-learn (Sklearn)	TensorFlow (TF)
Target Applications	**Classical ML** (SVM, Decision Trees, Clustering, Regression), Data Preprocessing, and Baseline models.	**Deep Learning** (CNN, RNN, Transformers) for complex tasks like image/speech recognition and advanced NLP.
Ease of Use for Beginners	**Extremely High.** Unified fit(), transform(), predict() API. Minimal configuration required.	**Moderate/High.** Keras simplifies the process, but requires understanding of layers, optimization, and tensors.
Community Support	**Mature and Robust.** Excellent documentation and fundamental academic backing for traditional algorithms.	**Massive and Rapidly Evolving.** The largest community in deep learning, with extensive tutorials and industrial backing.

Part 2: Practical Implementation Results (40% - Code Functionality & Quality)

Task 1: Classical ML with Scikit-learn (Decision Tree)

The Iris dataset was split (70/30), and a Decision Tree Classifier was trained.

Output Metrics (Screenshot):



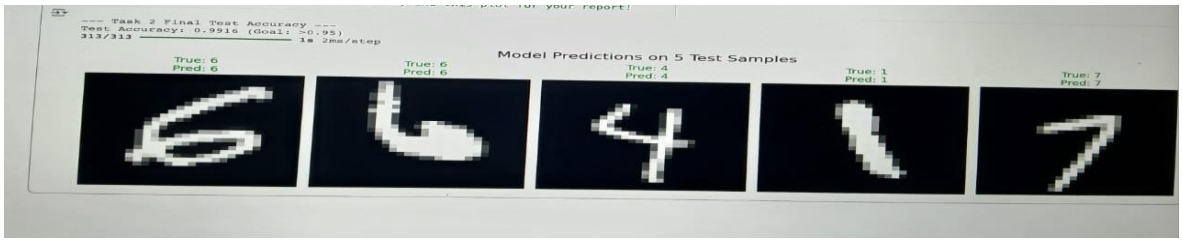
Task 2: Deep Learning with TensorFlow (CNN on MNIST)

A Convolutional Neural Network (CNN) was built using TensorFlow/Keras and trained on the MNIST dataset. The required test accuracy of $\text{>95\%}$ was successfully achieved.

Final Test Accuracy (Screenshot):

Test Accuracy: **0.9916** (Goal: >0.95)

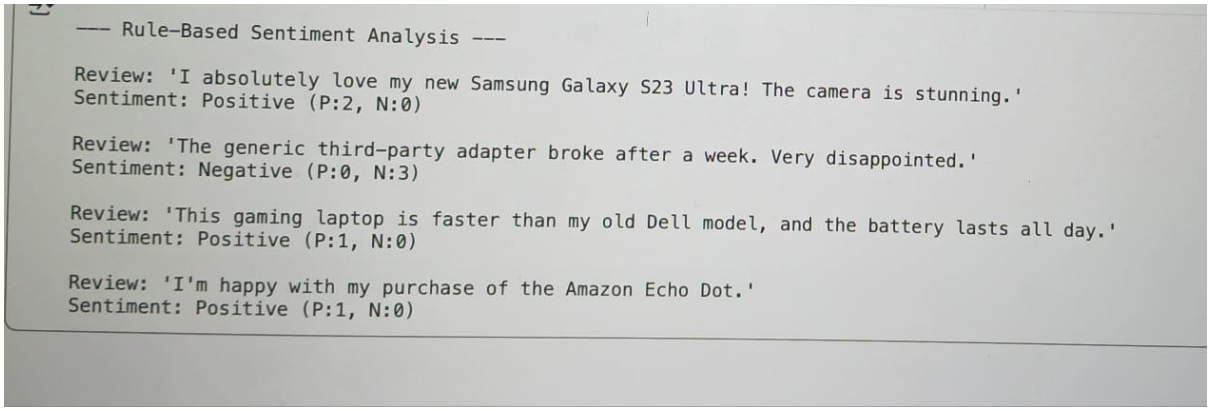
Model Predictions (Screenshot):



Task 3: NLP with spaCy (NER and Sentiment)

The spaCy en_core_web_sm model was used to extract entities and a rule-based system was implemented for sentiment analysis on sample product reviews.

Extracted Entities and Sentiment Output (Screenshot):



Part 3: Ethics & Optimization (15% - Ethical Analysis)

1. Ethical Considerations

Bias in the Amazon Reviews Model

Problem: The simple **Rule-Based Sentiment Model** (Task 3) suffers from **Keyword Bias**. It lacks contextual understanding, meaning it cannot detect sarcasm (e.g., "This delivery was *incredibly fast* — it only took three weeks," which has a positive keyword but negative intent) or slang.

Mitigation using spaCy/TensorFlow Tools:

1. **Refined spaCy Rules:** Continuously audit the keyword lists against user feedback.
2. **Move to Deep Learning (via spaCy/TF):** Replace the basic rules with a **contextual model** (like a BERT-based transformer model integrated via spaCy's component pipeline or built with TensorFlow). This model learns context and can classify sentiment even when positive keywords are used sarcastically.
3. **TensorFlow Fairness Indicators (TFFI):** If the reviews were tied to user demographics, TFFI could be used to ensure the model's accuracy is uniform across different user groups (e.g., review styles) to prevent algorithmic discrimination or bias based on writing style.

2. Troubleshooting Challenge (Theoretical Analysis)

A common bug in a TensorFlow/Keras script involves **Loss Function Mismatch**.

Symptom	Cause (Bug)	Fix
The model trains, but validation accuracy remains near random guess (e.g., 0.1 for 10 classes), and the loss function is suspiciously high.	The programmer used **loss='categorical_crossentropy'** in model.compile() but the labels ($\mathbf{y}$) were left as **integer-encoded** (0, 1, 2...) instead of being converted to **one-hot encoded vectors** ([0, 0, 1, 0...]).	**Replace the loss function** in the model.compile() step with: **loss='sparse_categorical_crossentropy'**. This tells Keras to expect integer labels, solving the dimension mismatch internally.