Course: AI for Software Engineering

Student: Luke Mbogo

Monday, June 9th, 2025.

Week 3: Mastering the AI Toolkit.

Contents

Introduction

This report explores core AI tools—Scikit-learn, TensorFlow, PyTorch, and spaCy—through both theory and practical implementation. By working with real datasets like Iris, MNIST, and Amazon reviews, we apply classical machine learning, deep learning, and natural language processing techniques. The goal is to understand how these tools function, compare their strengths, and consider ethical implications in building fair and effective AI models.

Theoretical Understanding

Q1: Differences Between TensorFlow and PyTorch

Answer:
TensorFlow and PyTorch are the two leading deep learning frameworks. Here's a comparison:

Feature	TensorFlow	PyTorch
API Style	Static computation graph (eager by default in TF 2.x)	Dynamic computation graph (define-by-run)
Ease of Use	More verbose, especially in older versions	Pythonic, intuitive and easier for debugging
Deployment	Excellent support with TensorFlow Serving, TensorFlow Lite, and TensorFlow.js	Requires third-party tools for deployment (TorchServe, ONNX)
Visualization	TensorBoard integrated	Limited native support (can use TensorBoard with extra setup)
Community/Industry Use	Backed by Google; dominant in production	Widely used in research, fast development

When to choose which:

- Use **TensorFlow** for scalable production models and mobile/edge deployment.

- Use **PyTorch** for rapid prototyping, academic research, and easier model debugging.

Q2: Use Cases for Jupyter Notebooks in AI

Answer:

1. **Interactive Data Exploration and Visualization:**
   Jupyter allows inline plotting with libraries like Matplotlib or Seaborn for EDA (Exploratory Data Analysis), making it perfect for testing small data segments and visual insights.

2. **Model Development & Documentation:**
   You can write code, test models, explain steps in markdown, and show live outputs – ideal for teaching, reporting, and collaboration.

Q3: How spaCy Enhances NLP Over Basic Python String Operations

Answer:

- **Tokenization:** spaCy understands linguistic context, unlike basic string .split().
- **NER (Named Entity Recognition):** spaCy can extract proper nouns and classify them (e.g., *ORG*, *PRODUCT*).
- **POS Tagging and Dependency Parsing:** Unlike string operations, spaCy identifies parts of speech and syntactic relationships.
- **Efficiency:** spaCy is optimized for large-scale NLP and faster than many traditional tools.

Q4: Comparative Analysis – Scikit-learn vs TensorFlow

Criteria	**Scikit-learn**	**TensorFlow**
Target Applications	Classical ML (SVMs, trees, clustering)	Deep Learning (CNNs, RNNs, GANs)
Beginner-Friendly	Very – simple syntax, great for small projects	More complex – better for large-scale neural networks
Community Support	Strong community, especially for academics and ML courses	Huge backing from Google, very active industry support

Part 2: Practical Implementation

Task 1: Classical ML with Scikit-learn – Iris Species Classifier

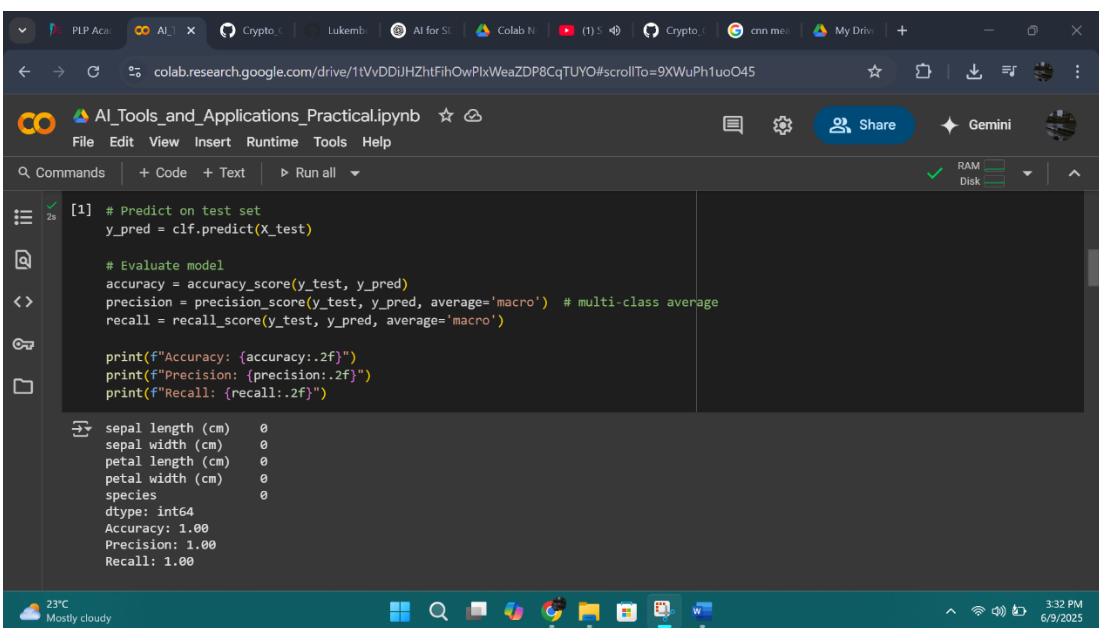
Goal

Train a **Decision Tree** classifier to predict iris species.

Steps Taken

1. Loaded Iris dataset from sklearn.datasets
2. Checked for missing values – none found
3. Label encoded species
4. Split data (80% train, 20% test)
5. Trained a Decision Tree model
6. Evaluated using:
    - Accuracy
    - Precision
    - Recall
    - Classification Report

Output screenshot



Task 2: Deep Learning with TensorFlow – MNIST Digit Classifier

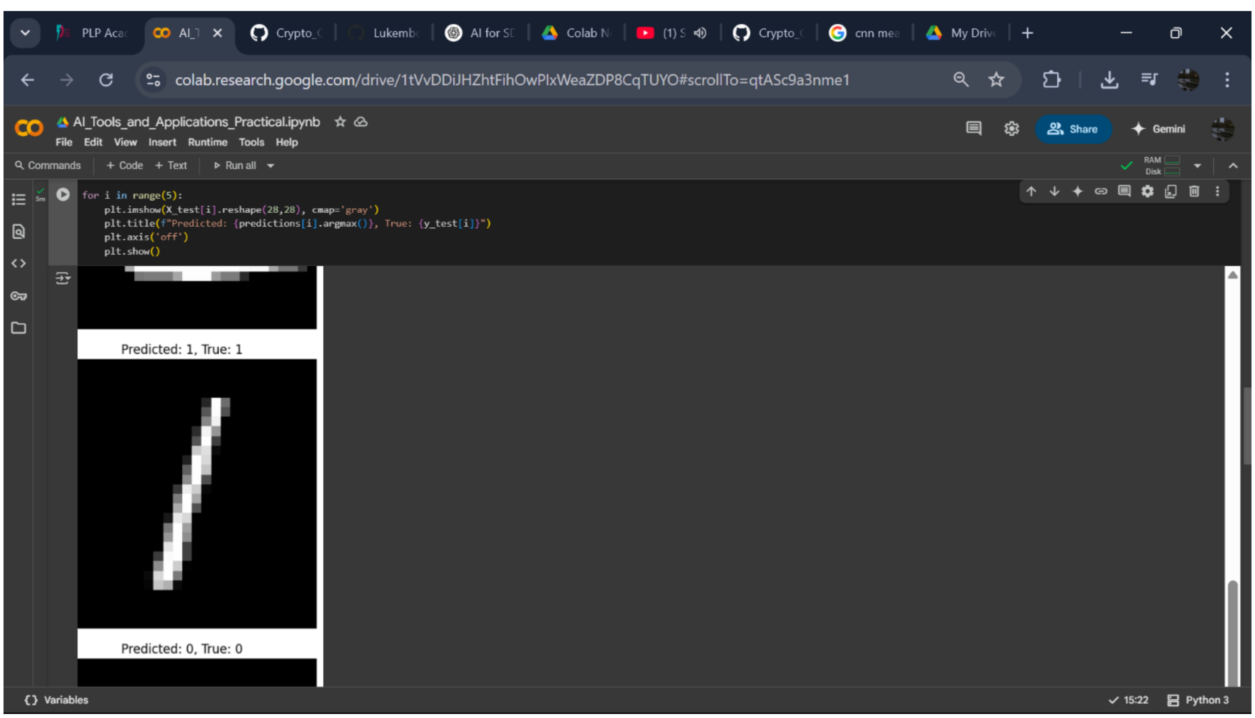
Goal

Use CNN to classify MNIST digits with >95% accuracy.

Steps Taken

1. Loaded MNIST from tf.keras.datasets
2. Normalized pixel values (0–255 → 0–1)
3. Built CNN with:
    - Conv2D + ReLU
    - MaxPooling
    - Dropout
    - Flatten → Dense(128) → Dense(10, softmax)
4. Trained model (5 epochs)
5. Evaluated accuracy on test set
6. Visualized 5 predictions

Output screenshot



Task 3: NLP with spaCy – NER & Sentiment on Amazon Reviews

Goal

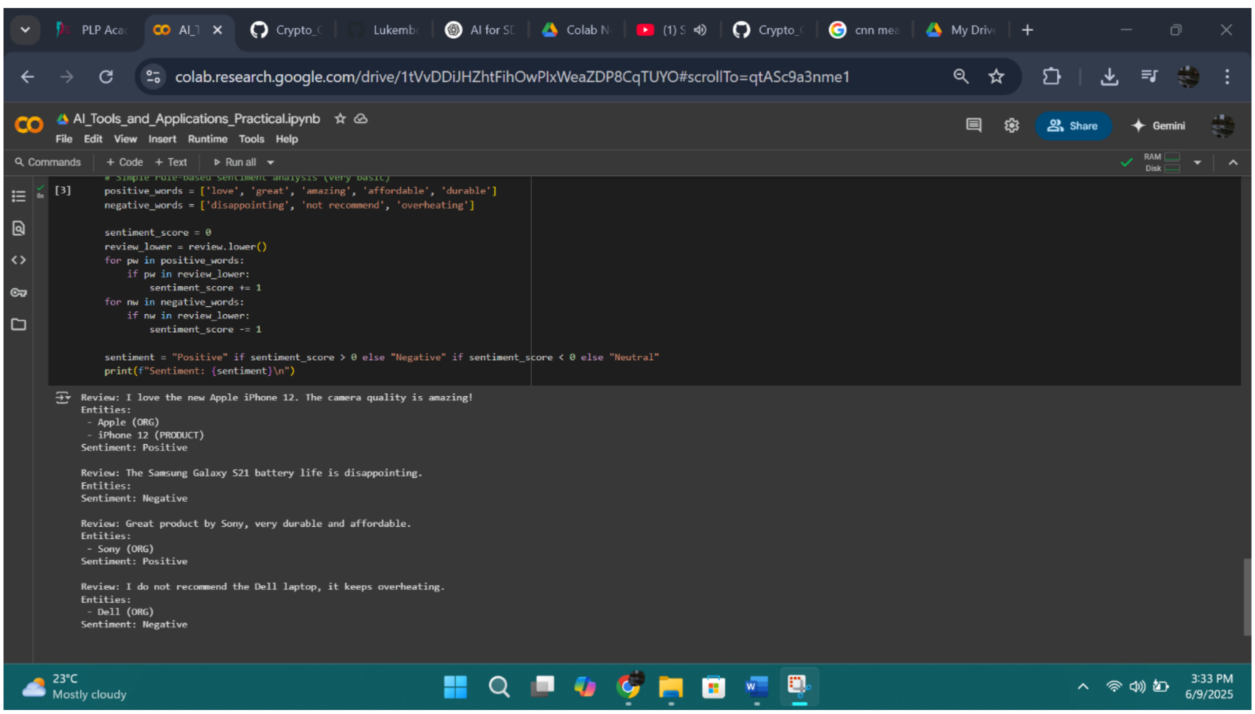
Use spaCy to:

- Extract named entities (products/brands)
- Analyze sentiment (rule-based)

Steps Taken

1. Loaded en_core_web_sm model
2. Parsed 3+ Amazon product reviews
3. Extracted entities using ent.text and ent.label_
4. Used a keyword-based sentiment detector

Output screenshot



Part 3: Ethics & Optimization

1. Ethical Considerations

a) Bias in MNIST

- Potential bias: Overfitting to a specific handwriting style or digit class.

- Mitigation:
    - Use **data augmentation** for diversity.
    - Evaluate model with fairness tools like **TensorFlow Fairness Indicators** to check performance gaps across digit types.

b) Bias in Amazon Reviews

- Bias source: Sentiment keywords are fixed; may not reflect real sentiment.
- Mitigation:
    - Use **trained sentiment models** instead of rule-based.
    - Check for demographic or linguistic bias with spaCy pipelines or add custom rules per context.

2. Troubleshooting Challenge (Sample Debug)

Problem: TensorFlow model throws a dimension mismatch error due to mismatched input shapes.
Solution:

- Verified shape of input with .shape and used .reshape() on input before model.predict().
- Ensured the final Dense layer matches number of classes (10 for MNIST).
- Used SparseCategoricalCrossentropy if labels were not one-hot encoded.

Conclusion.

This project provided hands-on experience with popular AI frameworks, helping us understand how to build and evaluate models for classification and NLP tasks. Beyond accuracy, it highlighted the importance of ethical AI—addressing bias and fairness. Mastering these tools is key to building impactful and responsible AI solutions.