

Course: AI for Software Engineering

Student: Luke Mbogo

Group 71.

Monday, June 9th, 2025.

Week 3: Mastering the AI Toolkit.

Group Members.

1. Ongezwa Ngotana- Ngotanao@gmail.com
2. Luke Mbogo -lukembogo5@gmail.com
3. Evans Odhiambo -evansathuol@gmail.com
4. OCHISHI CHRISTOPHER MBAM ochishichristopher@gmail.com
5. Derrick Njuguna njugunaderrick468@gmail.com

Contents

Introduction	3
Theoretical Understanding	3
Q1: Differences Between TensorFlow and PyTorch	3
Q2: Use Cases for Jupyter Notebooks in AI.....	4
Q3: How spaCy Enhances NLP Over Basic Python String Operations.....	4
Q4: Comparative Analysis – Scikit-learn vs TensorFlow	4
Part 2: Practical Implementation	5
Task 1: Classical ML with Scikit-learn – Iris Species Classifier	5
Goal	5
Output screenshot	5
Task 2: Deep Learning with TensorFlow – MNIST Digit Classifier	6
Goal	6
Output screenshot	6
Task 3: NLP with spaCy – NER & Sentiment on Amazon Reviews.....	7
Goal	7
Output screenshot	7
Part 3: Ethics & Optimization	7
1. Ethical Considerations.....	7
2. Troubleshooting Challenge (Sample Debug).....	8
Conclusion.....	8

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

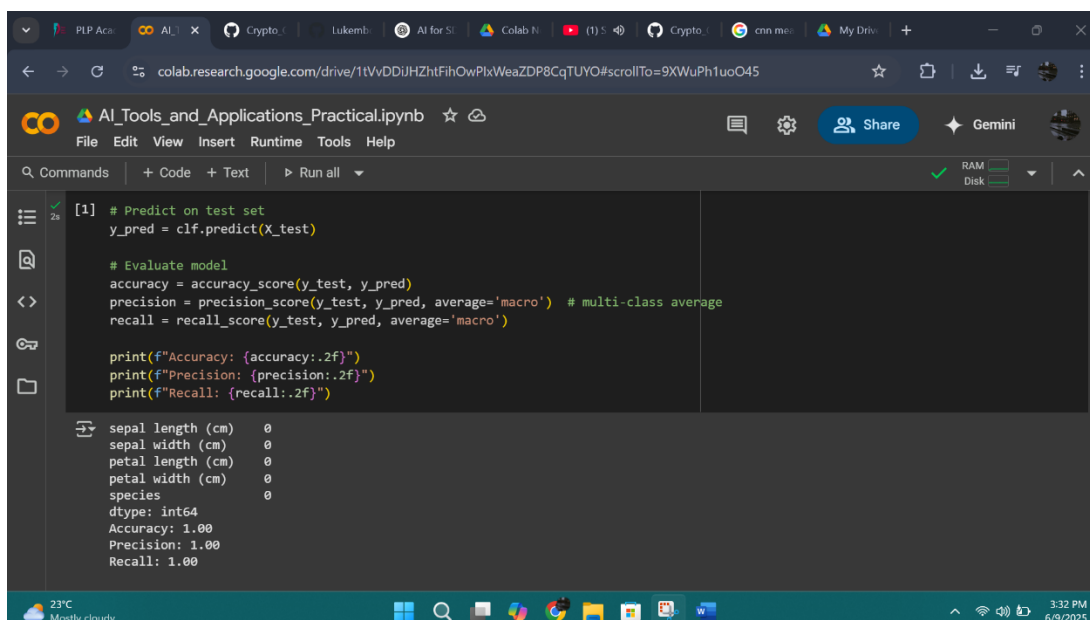
Goal

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



```
[1] # Predict on test set
y_pred = clf.predict(X_test)

# Evaluate model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='macro') # multi-class average
recall = recall_score(y_test, y_pred, average='macro')

print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")

sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
species              0
dtype: int64
Accuracy: 1.00
Precision: 1.00
Recall: 1.00
```

Task 2: Deep Learning with TensorFlow – MNIST Digit Classifier

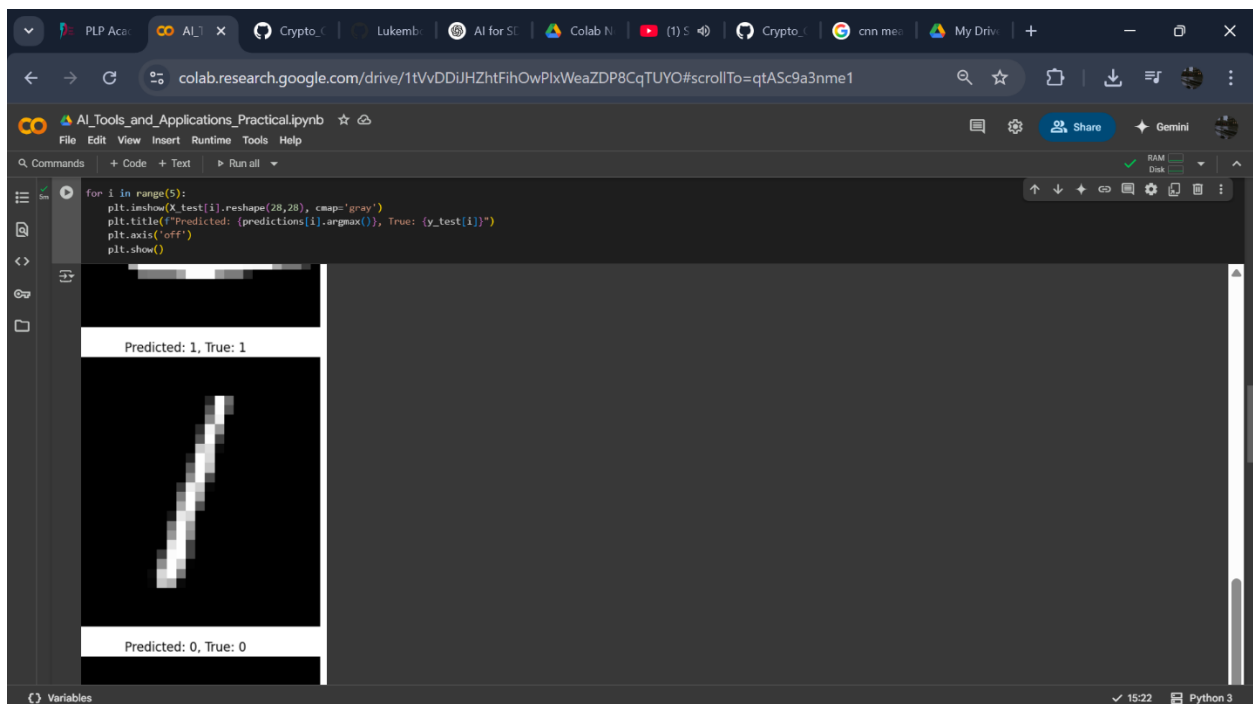
Goal

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

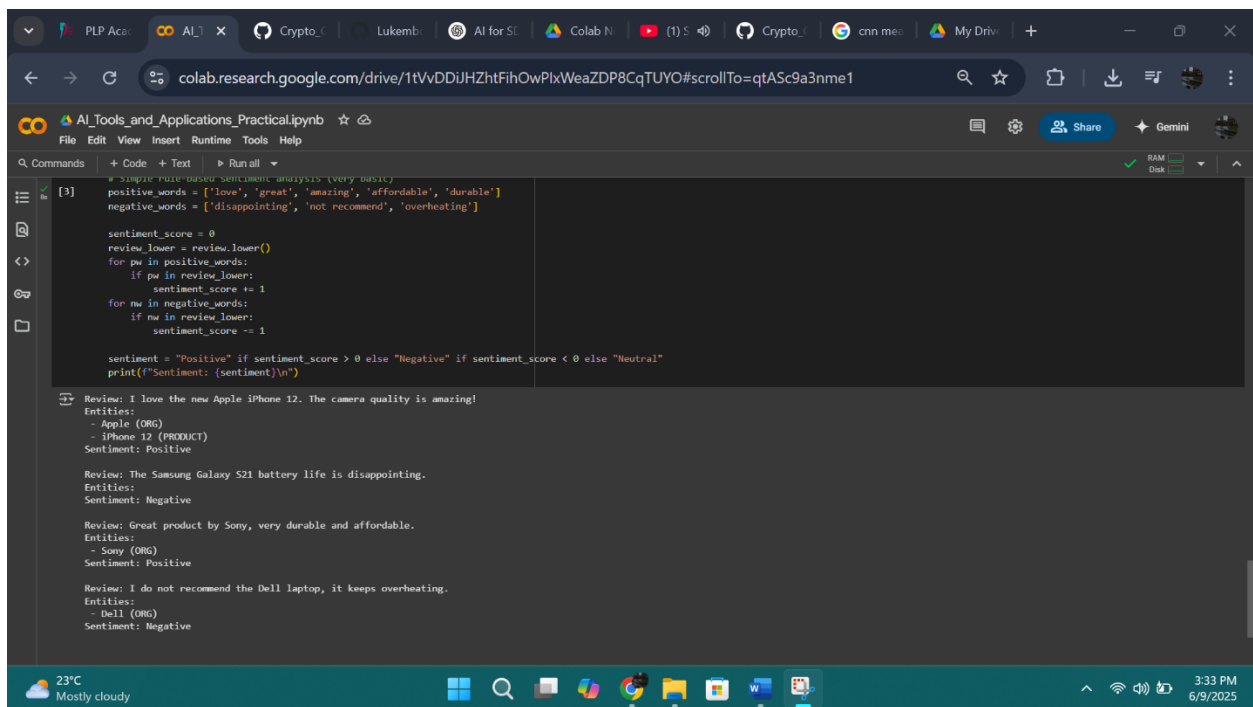
Use spaCy to:

- 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



The screenshot shows a Google Colab notebook titled "AI_Tools_and_Applications_Practical.ipynb". The code in the notebook uses spaCy to process three Amazon reviews. It defines a list of positive words and a list of negative words, then iterates through the words in each review to calculate a sentiment score. The output shows the extracted entities and the sentiment for each review.

```
# Import rule-based sentiment analysis (very basic)
positive_words = ['love', 'great', 'amazing', 'affordable', 'durable']
negative_words = ['disappointing', 'not recommend', 'overheating']

sentiment_score = 0
review_lower = review.lower()
for pw in positive_words:
    if pw in review_lower:
        sentiment_score += 1
for nw in negative_words:
    if nw in review_lower:
        sentiment_score -= 1

sentiment = "Positive" if sentiment_score > 0 else "Negative" if sentiment_score < 0 else "Neutral"
print(f'Sentiment: {sentiment}\n')
```

Review: I love the new Apple iPhone 12. The camera quality is amazing!
Entities:
- Apple (ORG)
- iPhone 12 (PRODUCT)
Sentiment: Positive

Review: The Samsung Galaxy S21 battery life is disappointing.
Entities:
Sentiment: Negative

Review: Great product by Sony, very durable and affordable.
Entities:
- Sony (ORG)
Sentiment: Positive

Review: I do not recommend the Dell laptop, it keeps overheating.
Entities:
- Dell (ORG)
Sentiment: Negative

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.