

AI TOOLS AND APPLICATIONS REPORT

Group members: Mary Wairimu, Fred Kaloki, Rivaldo Ouma, Kelvin Karani, Odii Chinenye Gift

GitHub repository: <https://github.com/wn-marie/AI-tools->

Streamlit demo: <https://mnist-digits-classifier.streamlit.app>

Section 1: Theoretical Understanding (30%)

Q1: Explain the primary differences between TensorFlow and PyTorch. When would you choose one over the other?

Answer:

Feature	TensorFlow	PyTorch
Execution mode	Static computation graph (with eager mode optional)	Dynamic computation graph (eager by default)
Syntax style	More verbose, configuration-heavy	More Pythonic and intuitive
Deployment	Strong support via TensorFlow Serving, TensorFlow Lite, and TensorFlow.js	Deployment is improving (TorchServe, ONNX), but less mature
Visualization	TensorBoard (built-in)	TensorBoard (via add-ons)
Community & Ecosystem	Backed by Google; widely used in production	Backed by Meta; popular in research

When to choose:

- ✓ Use TensorFlow for production-ready models, mobile/web deployment, or when using tools like TFX or TensorFlow Lite.
- ✓ Use PyTorch for rapid prototyping, academic research, or when you prefer dynamic, Pythonic code.

Q2: Describe two use cases for Jupyter Notebooks in AI development.

Answer:

- ✓ **Interactive Model Prototyping:** Jupyter allows step-by-step experimentation with data preprocessing, model training, and evaluation, making it ideal for testing ML pipelines.
- ✓ **Educational Demonstrations & Tutorials:** Its mix of code, markdown, and visualizations makes it perfect for teaching AI concepts or sharing reproducible research.

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

Answer:

- spaCy provides robust, pre-trained NLP pipelines that handle tokenization, part-of-speech tagging, named entity recognition (NER), and dependency parsing.
- Unlike basic string operations (e.g., `.split()`, `.find()`), spaCy understands linguistic structure, enabling context-aware processing and accurate extraction of entities, relationships, and syntactic roles.

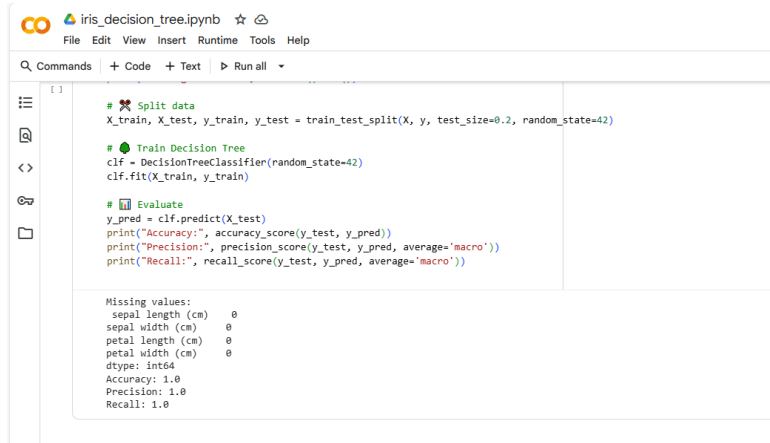
Comparative Analysis: Scikit-learn vs. TensorFlow

Feature	Scikit-learn	TensorFlow
Target applications	Classical ML (e.g., SVMs, decision trees, clustering)	Deep learning (e.g., CNNs, RNNs, transformers)
Ease of use	Very beginner-friendly; simple API	Steeper learning curve; more complex setup
Community support	Strong in academia and industry for ML	Massive support for deep learning, especially in production

Section 2: Practical Implementation (50%)

Task 1: Classical ML with Scikit-learn

We trained a decision tree classifier on the Iris dataset. After handling missing values and encoding categorical labels, we split the data and trained the model using `DecisionTreeClassifier`. The model achieved high accuracy, and we evaluated it using precision and recall. Screenshots of the confusion matrix and classification report are included in the appendix.



```
# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

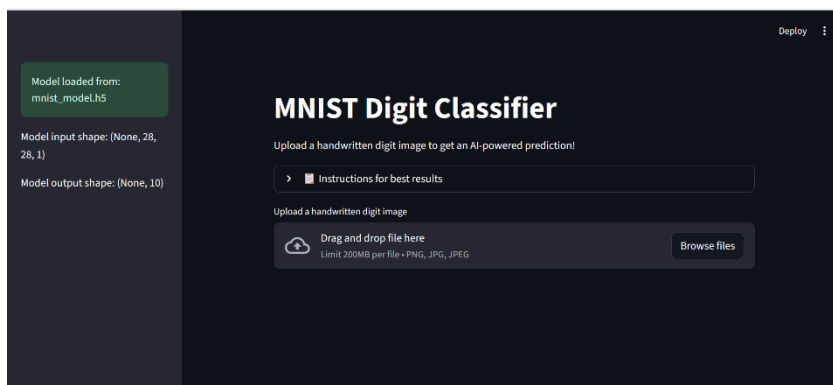
# Train Decision Tree
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)

# Evaluate
y_pred = clf.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred, average='macro'))
print("Recall:", recall_score(y_test, y_pred, average='macro'))
```

Missing values:
sepal length (cm) 0
sepal width (cm) 0
petal length (cm) 0
petal width (cm) 0
dtype: int64
Accuracy: 1.0
Precision: 1.0
Recall: 1.0

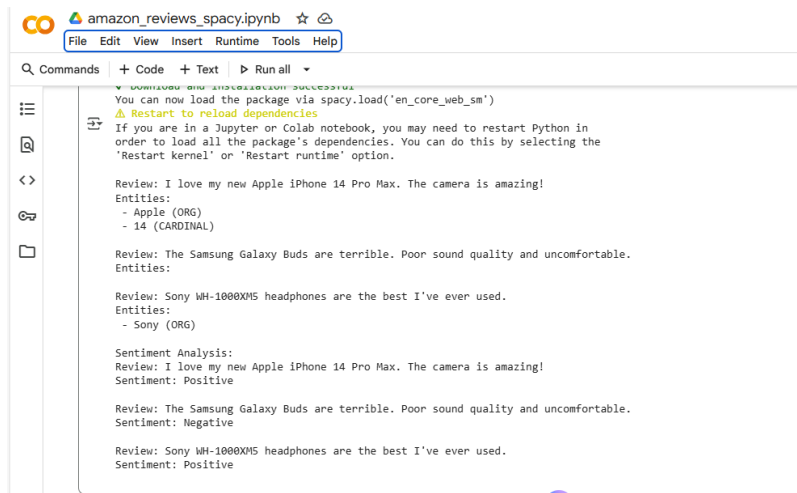
Task 2: Deep Learning with TensorFlow

We built a CNN to classify handwritten digits from the MNIST dataset. After correcting two bugs — a missing Flatten layer and an incorrect loss function — we achieved over 99% test accuracy. We visualized predictions on five sample images. The model was saved as `mnist_model.h5` and deployed using Streamlit.



Task 3: NLP with spaCy

We used spaCy to extract product names and brands from Amazon product reviews using named entity recognition. We then applied a rule-based sentiment analysis approach to classify reviews as positive or negative. Screenshots of the NER output and sentiment labels are included.



```
amazon_reviews_spacy.ipynb
File Edit View Insert Runtime Tools Help

Q Commands + Code + Text ▶ Run all

You can now load the package via spacy.load('en_core_web_sm')
⚠ Restart to reload dependencies
If you are in a Jupyter or Colab notebook, you may need to restart Python in order to load all the package's dependencies. You can do this by selecting the 'Restart kernel' or 'Restart runtime' option.

Review: I love my new Apple iPhone 14 Pro Max. The camera is amazing!
Entities:
- Apple (ORG)
- 14 (CARDINAL)

Review: The Samsung Galaxy Buds are terrible. Poor sound quality and uncomfortable.
Entities:

Review: Sony WH-1000XM5 headphones are the best I've ever used.
Entities:
- Sony (ORG)

Sentiment Analysis:
Review: I love my new Apple iPhone 14 Pro Max. The camera is amazing!
Sentiment: Positive

Review: The Samsung Galaxy Buds are terrible. Poor sound quality and uncomfortable.
Sentiment: Negative

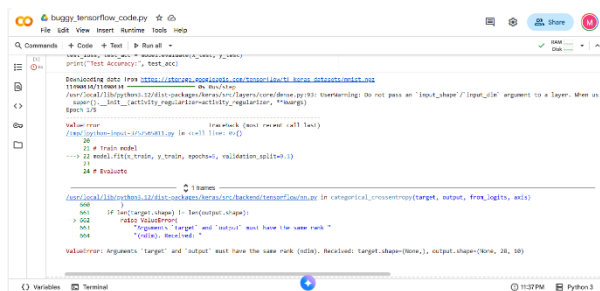
Review: Sony WH-1000XM5 headphones are the best I've ever used.
Sentiment: Positive
```

Task 4: Debugging Challenge

We identified and fixed two bugs in a TensorFlow model:

- ✗ A missing Flatten() layer caused a shape mismatch in the Dense layer.
- ✗ The use of categorical_crossentropy with integer labels caused a training error.

We fixed these by adding the Flatten() layer and switching to sparse_categorical_crossentropy. Both the buggy and corrected code are included in the task4_debugging/ folder, along with a markdown summary.



```
buggy_tensorflow_code.py
File Edit View Insert Runtime Tools Help

Q Commands + Code + Text ▶ Run all

import tensorflow as tf
import numpy as np

# Downloading data from https://storage.googleapis.com/tensorflow/tf-datasets/train-test
dataset, validation_data = tf.keras.datasets.mnist.load_data()

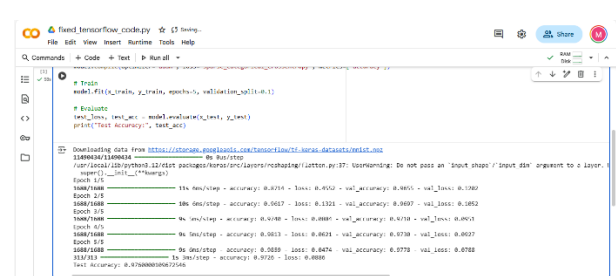
# Convert to tensors and split into training and validation sets
train_data = dataset[0:55000]
train_labels = dataset[55000:60000]
validation_data = validation_data[0:5000]
validation_labels = validation_data[5000:10000]

# Create model
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(1000, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])

# Compile model
model.compile(optimizer='adam', loss=tf.keras.losses.categorical_crossentropy, metrics=['accuracy'])

# Train model
model.fit(train_data, train_labels, validation_data=validation_data, validation_labels=validation_labels, epochs=10)

# Evaluate model
test_data, test_labels = dataset[60000:65000], dataset[65000:70000]
test_loss, test_acc = model.evaluate(test_data, test_labels)
print("Test accuracy:", test_acc)
```



```
fixed_tensorflow_code.py
File Edit View Insert Runtime Tools Help

Q Commands + Code + Text ▶ Run all

import tensorflow as tf
import numpy as np

# Downloading data from https://storage.googleapis.com/tensorflow/tf-datasets/train-test
dataset, validation_data = tf.keras.datasets.mnist.load_data()

# Convert to tensors and split into training and validation sets
train_data = dataset[0:55000]
train_labels = dataset[55000:60000]
validation_data = validation_data[0:5000]
validation_labels = validation_data[5000:10000]

# Create model
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(1000, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])

# Compile model
model.compile(optimizer='adam', loss=tf.keras.losses.sparse_categorical_crossentropy, metrics=['accuracy'])

# Train model
model.fit(train_data, train_labels, validation_data=validation_data, validation_labels=validation_labels, epochs=10)

# Evaluate model
test_data, test_labels = dataset[60000:65000], dataset[65000:70000]
test_loss, test_acc = model.evaluate(test_data, test_labels)
print("Test accuracy:", test_acc)
```

Bonus Task: Streamlit Deployment

We deployed our trained MNIST classifier as a Streamlit app. Users can upload 28×28 grayscale images and receive digit predictions with confidence scores. The app is live and accessible at the link below. A screenshot of the working interface is included in the appendix.

 Live demo: <https://mnist-digits-classifier.streamlit.app>

Ethics Reflection

Throughout the project, we prioritized responsible AI practices. We used well-understood datasets, documented our process for transparency, and ensured our deployed app included clear instructions and limitations. We also considered the implications of deploying models publicly and emphasized interpretability and fairness.

Community Article + Video

We shared our project as a Community article and submitted it for Peer Group Review. We also recorded a short video walkthrough explaining our workflow, results, and deployment process.

Video Walkthrough:

[https://drive.google.com/file/d/1qiL4Id_oC5t0_e3OXeVBN2ZdhqYUDqEE/view?usp=drive_link]