**Q1: TensorFlow vs PyTorch**

* TensorFlow uses **static graphs** (older versions) or eager + graph (2.x), while PyTorch uses **dynamic computation graphs** by default.
* PyTorch is more **intuitive and easier to debug**; TensorFlow is optimized for **production deployment** and mobile applications.
* **Use cases:**
  + **PyTorch:** Research, experimentation, prototyping new models.
  + **TensorFlow:** Large-scale deployment, production, or mobile integration.

**Q2: Two Use Cases for Jupyter Notebooks**

* **Data exploration & visualization:** Quickly inspect datasets, plot graphs, summarize statistics.
* **Model prototyping:** Step-by-step model building, training, and evaluation with immediate feedback for debugging and experimentation.

**Q3: How spaCy Enhances NLP Tasks**

* Provides **pre-trained models** for tokenization, POS tagging, named entity recognition, and dependency parsing.
* Identifies entities (e.g., brands, product names, dates) **more reliably than basic Python string operations**.
* Handles **linguistic structures** and context that string methods cannot, making NLP tasks faster and more accurate.

**Comparative Analysis: Scikit-learn vs TensorFlow**

* **Target applications:**
  + Scikit-learn → classical ML (decision trees, SVM, linear regression)
  + TensorFlow → deep learning (CNNs, RNNs, transformers)
* **Ease of use:**
  + Scikit-learn → beginner-friendly, high-level APIs, minimal setup
  + TensorFlow → moderate complexity, more boilerplate, better for deep learning
* **Community support:**
  + Scikit-learn → mature and stable
  + TensorFlow → very large and rapidly growing

**Task 1 – Classical ML (Iris Dataset)**

1. **Dataset Preview**
   * Screenshot the first 5 rows of df.head()

A screenshot of a computer

AI-generated content may be incorrect.

* + “Figure: First 5 rows of the Iris dataset showing features and target column.”

1. **Feature & Target Shapes**
   * Screenshot output of:
   * print("Features shape:", X.shape)
   * print("Target shape:", y.shape)

A screenshot of a computer

AI-generated content may be incorrect.

“Figure: Shape of feature matrix and target vector.”

1. **Decision Tree Training**

A screenshot of a video game

AI-generated content may be incorrect.

1. **Model Evaluation**
   * Screenshot the output of accuracy, precision, and recall.

A screenshot of a computer program

AI-generated content may be incorrect.

* + “Figure: Decision tree performance metrics on Iris dataset.”

**Task 2 – Deep Learning (MNIST CNN)**

1. **Data Preview / Shape Check**
   * Screenshot shapes of x\_train, y\_train\_cat, x\_test, y\_test\_cat

A screenshot of a computer program

AI-generated content may be incorrect.

“Figure: Shapes of training and test datasets after preprocessing.”

1. **CNN Model Summary**
   * Screenshot output of model.summary()

A screenshot of a computer

AI-generated content may be incorrect.

* + “Figure: CNN architecture with layer output shapes and parameters.”

1. **Training Progress**
   * Screenshot the **accuracy/loss plot** or training history output

A screen shot of a graph

AI-generated content may be incorrect.

* + “Figure: Training and validation accuracy over epochs.”

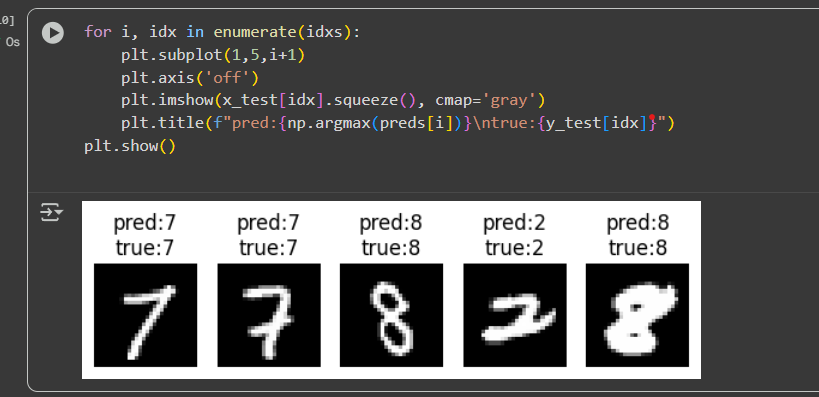
1. **Test Accuracy**
   * Screenshot printed test accuracy

A screenshot of a computer

AI-generated content may be incorrect.

* + “Figure: CNN model test accuracy on unseen MNIST data.”

1. **Sample Predictions**
   * Screenshot visual of **5 sample images with predicted vs actual labels**



* + “Figure: Example predictions of handwritten digits.”

**Task 3 – NLP with spaCy (Amazon Reviews)**

1. **Dataset Preview**
   * Screenshot first 5 reviews from df['reviewText']

A screenshot of a computer

AI-generated content may be incorrect.

* + “Figure: Sample Amazon reviews loaded from train.ft.txt.”

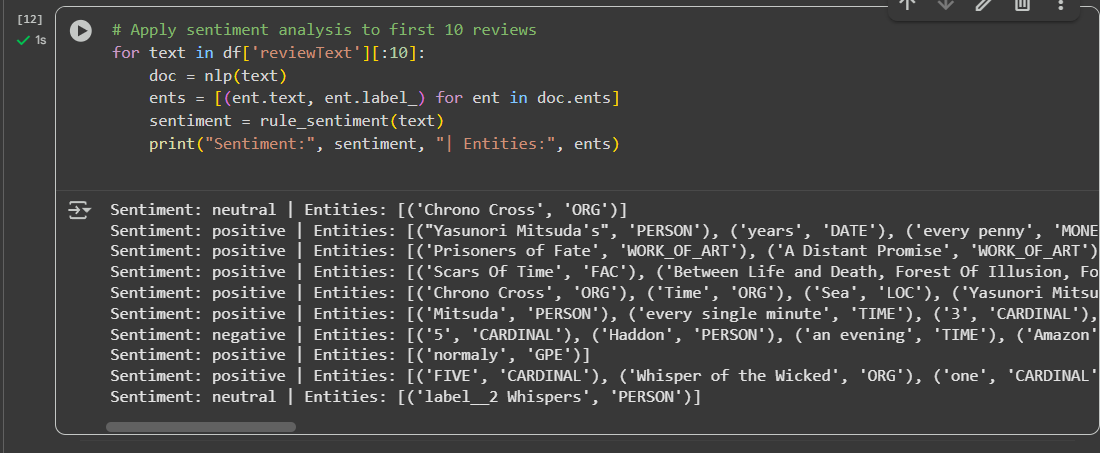
1. **NER Output**
   * Screenshot output for **first 20 reviews** showing detected entities

A screen shot of a computer

AI-generated content may be incorrect.

* + “Figure: Named entities extracted from Amazon reviews.”

1. **Sentiment + NER**
   * Screenshot first 10 reviews showing both **sentiment classification** and **entities**



* + “Figure: Rule-based sentiment analysis with extracted entities.”

**Ethical Considerations (using all 3 tasks)**

**Potential Biases**

* **Task 1 (Iris Dataset):**
  + Small dataset → model could overfit or misclassify species if new data slightly differs.
* **Task 2 (MNIST CNN):**
  + Bias toward digits that are overrepresented in the dataset.
  + Model may struggle with handwriting styles not seen in training.
* **Task 3 (Amazon Reviews):**
  + NER may miss obscure brands or informal product names.
  + Rule-based sentiment can misclassify sarcasm, slang, or ambiguous text.

**Mitigation Strategies**

* **TensorFlow Fairness Indicators (Task 2):**
  + Evaluate accuracy across digits to detect class imbalance issues.
* **spaCy Rule-Based Systems (Task 3):**
  + Expand entity patterns and sentiment lexicons to reduce bias.
* **General Strategies:**
  + Use diverse datasets (all tasks).
  + Cross-validation to detect overfitting.
  + Manual checks on NER outputs to ensure coverage of edge cases.

**Troubleshooting Challenge (using Task 2 as example)**

* Common TensorFlow CNN errors you might encounter:
  + **Dimension mismatches:** Ensure inputs have correct shape (28x28x1 for grayscale MNIST).
  + **Loss function mismatch:**
    - categorical\_crossentropy for one-hot labels.
    - sparse\_categorical\_crossentropy for integer labels.
  + **Activation function mistakes:** relu in hidden layers, softmax in output layer.
  + **Optimizer errors:** Use Adam or SGD with proper learning rate.
* **Debugging steps:**
  + Print shapes of inputs and outputs.
  + Verify label encoding matches loss function.
  + Run a small batch to test forward pass before full training.