## Part 1: Theoretical Understanding (40%)

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

TensorFlow uses a static computation graph (define-and-run), where the graph is defined first and then executed. It's optimized for production deployment and has strong support for distributed computing.

PyTorch, on the other hand, uses a dynamic computation graph (define-by-run), which allows for more flexibility and easier debugging—ideal for research and experimentation.

When to choose one over the other:

Use TensorFlow if you're focused on production deployment, scalability, or using pre-built tools like TFLite or TF.js.

Use PyTorch if you're in a research environment, need flexibility, or are working with rapidly changing model architectures.

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

Exploratory Data Analysis (EDA): Jupyter Notebooks allow data scientists to load, visualize, and manipulate data interactively, making it easy to understand patterns, detect anomalies, and preprocess datasets before model training.

Model Prototyping and Visualization: Developers can incrementally build and test machine learning models, visualize training progress, and tweak hyperparameters in real time—all within a single, shareable document.

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

spaCy enhances NLP tasks by providing:

Pre-trained Language Models: spaCy includes trained pipelines for tasks like tokenization, named entity recognition (NER), part-of-speech tagging, and dependency parsing—far beyond what basic string methods can offer.

Efficiency and Accuracy: spaCy is optimized for performance and accuracy, leveraging modern NLP techniques under the hood, whereas string operations lack context understanding and require manual rule creation.

**2. Comparative Analysis**

- Compare Scikit-learn and TensorFlow in terms of:
    - Target applications (e.g., classical ML vs. deep learning).
    - Ease of use for beginners.
    - Community support.

Choose Scikit-learn when:

Working on traditional machine learning problems.

Need fast prototyping with minimal setup.

Dealing with structured or semi-structured data.

Looking for model interpretability.

Choose TensorFlow when:

Building deep learning models (CNNs, RNNs, Transformers).

Working with unstructured data (images, text, audio).

Planning to deploy models at scale (mobile, web, embedded systems).

Engaging in AI research or advanced neural network experimentation.