Q1: Explain the primary differences between TensorFlow and PyTorch.

When would you choose one over the other?

- TensorFlow uses static computation graphs (though it now supports eager execution), making it suitable for production and deployment environments like TensorFlow Serving or TensorFlow Lite.
- PyTorch uses dynamic computation graphs (eager execution by default), which makes debugging and experimentation easier.

✅ Choose PyTorch for research, quick prototyping, or dynamic models.

✅ Choose TensorFlow for scalable production systems and mobile deployment.

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

Interactive Model Development: Easily train and test models in a step-by-step manner, perfect for prototyping machine learning and deep learning workflows.

Data Exploration & Visualization: Integrate code with inline plots (e.g., using matplotlib, seaborn) for exploring datasets and cleaning data interactively.

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

spaCy provides out-of-the-box support for tokenization, part-of-speech tagging, named entity recognition, and dependency parsing, which basic string operations cannot handle.

It uses pretrained statistical models for fast, accurate NLP, whereas string operations are

rule-based and limited in context.

2. Comparative Analysis: Scikit-learn vs TensorFlow

Feature Scikit-learn TensorFlow

Target Applications Classical ML (SVMs, decision trees, linear models) Deep Learning (neural networks, CNNs, RNNs)

Ease of Use Simple, intuitive API, ideal for beginners More complex, but powerful and customizable

Community Support Strong, mature community for traditional ML Large and growing community, especially for production AI