1. Short Answer Questions

Q1: TensorFlow vs PyTorch Differences

Key Differences:

- **Execution Model**: TensorFlow uses static computation graphs (eager execution available), PyTorch uses dynamic computation graphs
- **Learning Curve**: PyTorch is more intuitive for beginners, TensorFlow has steeper learning curve
- **Production**: TensorFlow better for production deployment, PyTorch better for research
- **Debugging**: PyTorch easier to debug (Pythonic), TensorFlow improving with TF 2.x

When to Choose:

- **TensorFlow**: Production systems, mobile deployment, large-scale distributed training
- **PyTorch**: Research, prototyping, educational purposes, complex architectures

Q2: Jupyter Notebooks Use Cases in AI

1. **Data Exploration and Visualization**: Interactive data analysis, plotting, and EDA
2. **Model Prototyping**: Rapid experimentation, iterative development, and documentation

Q3: spaCy vs Basic Python String Operations

spaCy Advantages:

- Pre-trained models for tokenization, POS tagging, NER
- Efficient processing pipelines
- Language-specific optimizations
- Built-in linguistic annotations
- Better handling of edge cases in text processing

2. Comparative Analysis: Scikit-learn vs TensorFlow

Aspect	Scikit-learn	TensorFlow
Target Applications	Classical ML, traditional algorithms	Deep learning, neural networks
Ease of Use	Very beginner-friendly, simple API	Steeper learning curve, more complex
Community Support	Strong, mature ecosystem	Large, active community, extensive resources