**Part 1: Theoretical Understanding (40%)**

**1. Short Answer Questions**

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

* **TensorFlow**: Developed by Google, TensorFlow uses *static computation graphs* (defined before execution), making it ideal for production environments. It excels in scalability and deployment, such as with **TensorFlow Serving**.
* **PyTorch**: Developed by Facebook, PyTorch uses *dynamic computation graphs*, offering flexibility and ease of debugging—features that are highly beneficial in research and rapid experimentation.
* **When to Choose**:
  + **TensorFlow**: Best for industry projects requiring robust deployment or distributed training (e.g., healthcare systems).
  + **PyTorch**: Ideal for research, prototyping, or situations demanding intuitive and readable code (e.g., academic experiments).

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

* **Exploratory Data Analysis (EDA)**: Allows for interactive preprocessing and visualization of datasets (e.g., plotting Iris feature distributions).
* **Result Sharing**: Combines code, outputs, and written explanations into a single document, ideal for presentations or tutorials (e.g., showcasing MNIST predictions).

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

* **spaCy**: Provides pre-trained models for advanced NLP tasks such as Named Entity Recognition (NER), Part-of-Speech (POS) tagging, and dependency parsing. It includes efficient, production-ready tokenization pipelines.
* **Basic Python String Operations**: Limited to basic manipulations like splitting and regular expressions, offering no semantic or linguistic understanding.
* **Enhancement**: spaCy enables automated and accurate NLP processing (e.g., extracting brand names from reviews) that would otherwise require extensive manual coding with string operations.

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

| **Aspect** | **Scikit-learn** | **TensorFlow** |
| --- | --- | --- |
| **Target Applications** | Classical ML (e.g., regression, clustering) | Deep learning (e.g., CNNs, RNNs) |
| **Ease of Use** | Simple API, beginner-friendly | Steeper learning curve, requires tensor manipulation |
| **Community Support** | Active community with tutorials and documentation | Larger community, strong industry support |

* **Scikit-learn**: Ideal for quick ML prototypes and classical algorithms, but not designed for deep learning.
* **TensorFlow**: Suitable for large-scale, complex deep learning models, though it may be overkill for simple machine learning tasks.