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?

Differences:

- **Computation Model:**
  - *TensorFlow:* Uses a static computation graph (with support for eager execution).
  - *PyTorch:* Uses a dynamic computation graph, which is more intuitive and easier for debugging.
- **Syntax and Debugging:**
  - *TensorFlow:* Initially more complex, but TensorFlow 2.x made it more user-friendly.
  - *PyTorch:* Clean and Pythonic, making it easy to write and debug code.
- **Deployment:**
  - *TensorFlow:* Offers robust deployment options (e.g., TF Lite, TF Serving, TF.js).
  - *PyTorch:* Deployment improving via TorchScript and ONNX.

When to Choose:

- Use **PyTorch** for research, fast prototyping, and academic projects.
- Use **TensorFlow** for production-level systems and when deployment tools are a priority.

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

1. **Data Exploration and Visualization:** Jupyter Notebooks allow users to interactively explore datasets using visualization libraries like Matplotlib and Seaborn.
2. **Model Prototyping and Experimentation:** Ideal for building and testing machine learning models incrementally, enabling developers to document and visualize results inline.

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

- **Pre-trained Models:** spaCy provides models for tasks like NER and POS tagging, which basic string functions can't handle.

- **Context-aware Tokenization:** It tokenizes text more intelligently than `.split()` or regex.
- **Linguistic Features:** Provides lemmas, dependencies, and entity recognition efficiently.
- **Performance:** Written in Cython, spaCy is much faster and optimized for production.

---

2. Comparative Analysis: Scikit-learn vs. TensorFlow

Feature	Scikit-learn	TensorFlow
Target Applications	Classical ML (e.g., regression, k-NN, SVM)	Deep Learning (e.g., CNNs, RNNs, Transformers)
Ease of Use	Beginner-friendly with a simple API	Requires more setup; steeper learning curve
Community Support	Strong in academia and classical ML communities	Large community with strong production tools

Summary:

- Use **Scikit-learn** for traditional machine learning tasks on tabular data.
- Use **TensorFlow** when working with deep learning and when scalability during deployment is required.