Part 1: Theoretical Understanding (40%)

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

Tensorflow is used for building and training deep learning model while PyTouch is used for computation graph and has ease of use. I would choose Tensorflow for robust deployment across platforms while PyTouch for fast experimentation in research.

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

Exploratory Data Analysis (EDA) & Visualization

Use Case: Before building AI models, data scientists need to understand the dataset—its structure, missing values, distributions, correlations, etc. Jupyter allows combining code, visualizations (using libraries like Matplotlib, Seaborn), and explanatory text (Markdown) in one document.

Example in AI:

Analyzing a dataset of medical images to detect cancer

Prototyping Machine Learning Models

Use Case: AI developers use Jupyter Notebooks to experiment with model architectures, test algorithms, tune hyperparameters, and evaluate results quickly.

Example in AI:

Training a neural network for image classification on CIFAR-10 dataset

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

SpaCy provides advanced, accurate, and linguistically intelligent processing for real NLP tasks while Python useful for simple tasks like text cleaning.

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.

Scikit-learn focuses on classical machine learning algorithms and is designed for traditional ML tasks like classification, regression, clustering, and dimensionality reduction. It excels at problems involving structured/tabular data, feature engineering, and scenarios where you need interpretable models. Common use cases include customer segmentation, fraud detection, recommendation systems with engineered features, and any application where you're working with relatively small to medium-sized datasets.

TensorFlow is primarily built for deep learning and neural networks, though it can handle traditional ML tasks as well. It's designed for complex pattern recognition in unstructured data like images, text, audio, and video. TensorFlow shines in computer vision, natural language processing, speech recognition, and any application requiring neural networks with multiple layers. It's also built for production deployment at scale and distributed computing.

Ease of Use for Beginners Scikit-learn is significantly more beginner-friendly. Its API is intuitive and consistent across all algorithms, following a simple fit/predict pattern. You can build and evaluate models with just a few lines of code, and the documentation includes many practical examples. The learning curve is gentle, and you can achieve meaningful results quickly without deep theoretical knowledge.

TensorFlow has a steeper learning curve, especially for beginners. While TensorFlow 2.x with Keras integration has made it more accessible than earlier versions, it still requires understanding of neural network concepts, tensor operations, and often involves more complex code structure. However, high-level APIs like tf.keras have significantly reduced the barrier to entry compared to TensorFlow 1.x.

Community Support Scikit-learn has excellent community support with comprehensive documentation, extensive tutorials, and active forums. It's been around since 2007 and has a mature ecosystem. The community tends to focus on practical applications and educational content, making it easier to find help for common problems.

TensorFlow has massive community support, being backed by Google and widely adopted in both academia and industry. It has extensive documentation, official tutorials, and countless third-party resources. The community is very active on GitHub, Stack Overflow, and specialized forums