PART 1:

**1. Short Answer Questions**

Q1: Differences Between TensorFlow and PyTorch

**Primary Differences:**

1. **Graph Definition:**
   * **TensorFlow:** Uses a *static computational graph* (define graph first, then execute).
   * **PyTorch:** Uses a *dynamic computational graph* (define-on-the-fly), enabling more intuitive debugging and flexibility.
2. **API Design:**
   * **TensorFlow:** High-level APIs (e.g., Keras) simplify code but abstract low-level control.
   * **PyTorch:** More "Pythonic," with imperative coding and easier integration with Python logic.
3. **Deployment:**
   * **TensorFlow:** Better production tools (e.g., TensorFlow Serving, TF Lite for mobile).
   * **PyTorch:** Historically research-focused, though TorchServe / TorchScript now improve deployment.
4. **Community & Ecosystem:**
   * **TensorFlow:** Strong industry adoption, extensive documentation.
   * **PyTorch:** Preferred in academia/research for rapid prototyping.

**When to Choose:**

* **TensorFlow:** Production deployment, mobile/edge devices, or leveraging TPUs.
* **PyTorch:** Research, dynamic models (e.g., RNNs), or when Pythonic debugging is critical.

Q2: Use Cases for Jupyter Notebooks in AI

1. **Exploratory Data Analysis (EDA) & Visualization:**
   * Interactively explore datasets, visualize distributions, and generate plots (e.g., Matplotlib/Seaborn). Enables quick hypothesis testing and data cleaning.
2. **Interactive Model Prototyping & Debugging:**
   * Train models incrementally (e.g., tweak hyperparameters in real-time), log outputs, and visualize results (e.g., confusion matrices, loss curves). Facilitates iterative experimentation.

Q3: spaCy vs. Basic Python String Operations

**spaCy Enhances NLP via:**

1. **Linguistic Intelligence:**
   * Goes beyond regex/string splits by understanding context (e.g., lemmatization *"running" → "run"*, entity recognition *"Apple → ORG"*).
   * Handles tokenization, POS tagging, and dependency parsing with pre-trained models.
2. **Efficiency & Scalability:**
   * Optimized Cython backend processes large text corpora faster than native Python loops.
   * Built-in pipelines (e.g., en\_core\_web\_lg) encapsulate complex NLP tasks in 1–2 lines of code.

**2. Comparative Analysis**

**1. Target Applications**

| **Scikit-learn** | **TensorFlow** |
| --- | --- |
| **Classical ML algorithms:** | **Deep Learning (DL) & Neural Networks:** |
| **- Supervised (SVMs, Random Forests, Linear Regression)** | **- CNNs (image/video), RNNs (sequence/time-series)** |
| **- Unsupervised (K-Means, PCA)** | **- Transformers (NLP: BERT, GPT-style models)** |
| **- Tabular, structured data** | **- Unstructured data (images, text, audio)** |
| **Small-to-medium datasets** | **Large-scale data (distributed training)** |
| **CPU-centric workflows** | **GPU/TPU acceleration** |

**2. Ease of Use for Beginners**

| **Scikit-learn** | **TensorFlow** |
| --- | --- |
| **Simple, consistent API: fit ()/predict ()** | **Steeper learning curve: Graph ops, sessions (pre-2.x), tensor shapes** |
| **Minimal setup - pure Python** | **Keras - TF high-level API simplifies model building** |
| **Focus on algorithm selection / hyperparameter tuning** | **Debugging complexities e.g., graph errors, device placement** |
| **Extensive built-in datasets and tutorials** | **Requires understanding of DL concepts e.g., backprop, layers** |

**3. Community Support**

| **Scikit-learn** | **TensorFlow** |
| --- | --- |
| **Extensive documentation** with examples | **Vast resources:** official guides, tutorials, TF Hub |
| **Mature community**: Widely used in academia/industry | **Industry adoption**: Google-backed, used in production systems |
| Easy troubleshooting (common ML problems) | Strong DL research community (papers, GitHub repos) |
| Limited DL support | Rapid updates can cause version compatibility issues |