Q1: Differences between TensorFlow and PyTorch

- **Computation Graph**
  - *TensorFlow* (TF1.x) uses static graphs; TF2.x added eager execution.
  - *PyTorch* always uses dynamic ("define-by-run") graphs.
- **API & Syntax**
  - *TensorFlow*: More verbose; integrates Keras as a high-level API.
  - *PyTorch*: Pythonic, concise—preferred for research and rapid prototyping.
- **Deployment & Ecosystem**
  - *TensorFlow*: Built-in tooling for production (TF Serving, TFLite, TF.js).
  - *PyTorch*: Deploy via TorchServe or export to ONNX; ecosystem growing.
- **Visualization**
  - *TensorFlow*: Native TensorBoard support.
  - *PyTorch*: Needs external packages (e.g. `tensorboardX`, Weights & Biases).

When to choose

- **TensorFlow**: Production pipelines needing cross-platform deployment.
- **PyTorch**: Research/experimentation or when you require dynamic graph flexibility.

Q2: Two use cases for Jupyter Notebooks

1. **Exploratory Data Analysis (EDA)**
   - Load data, compute summary statistics, and plot distributions inline (e.g. with matplotlib).
   - Iterate on data-cleaning steps interactively.
2. **Model Prototyping & Documentation**
   - Build model architectures cell by cell and see intermediate outputs.
   - Combine code, results, and narrative in one sharable document.

Q3: How spaCy enhances NLP vs. basic string ops

- **Tokenization** that respects language rules (e.g., "U.S.A." vs. "USA").
- **POS tagging** and **dependency parsing** for syntactic structure.
- **Named Entity Recognition (NER)** out of the box (people, products, orgs).
- **Lemmatization** to reduce words to their base form.

Basic Python (`.split()`, `.find()`) only handles raw text operations without linguistic awareness.

Comparative Analysis: Scikit-learn vs. TensorFlow

Aspect	Scikit-learn	TensorFlow
Target applications	Classical ML (regression, trees)	Deep Learning (CNNs, RNNs, etc.)
Ease of use	Very beginner-friendly	Steeper learning curve (TF2+ easier)
Community support	Large for ML	Massive ecosystem backed by Google
Deployment	Research/prototyping	Production-ready (Serving, TFLite)

Part 3: Ethics & Optimization

1. **Bias Identification**
    - *MNIST*: Skewed toward certain handwriting styles; may misclassify non-digit characters.
    - *Reviews*: Rule-based sentiment misses sarcasm or domain-specific language.
2. **Mitigation Strategies**
    - **TensorFlow Fairness Indicators**: Evaluate model performance across subgroups (e.g., handwriting styles).
    - **spaCy rule enhancements**: Incorporate domain lexicons and fallback flags for ambiguous sentiment.