AI/ML Comparative Analysis

1. Short Answer Questions

Q1: TensorFlow vs. PyTorch - Key Differences and Use Cases

Differences:

- TensorFlow:
  * Static computation graphs (define-then-run) by default
  * Strong production deployment tools (TF Serving, Lite, JS)
  * Better for distributed training and mobile/edge devices
- PyTorch:
  * Dynamic computation graphs (eager execution first)
  * More flexible for research (custom architectures)

When to Choose:

- TensorFlow: Production pipelines, large-scale deployment
- PyTorch: Research, rapid prototyping (NLP, GANs)

Q2: Jupyter Notebooks in AI - Two Key Use Cases

1. Exploratory Data Analysis (EDA):
   - Interactive visualization (Matplotlib, Seaborn)
   - Immediate feedback for feature engineering
2. Model Experimentation:
   - Step-by-step training/testing
   - Inline outputs (confusion matrices, loss curves)

Q3: spaCy vs. Basic String Operations

Feature	spaCy	Python Strings
Tokenization	Linguistically aware	Splits crudely
NER	Pre-trained models	Regex required
Speed	Optimized (Cython backend)	Slower for parsing

Best For:

- spaCy: NLP pipelines (chatbots, document analysis)

- Strings: Simple tasks (substring search)

2. Comparative Analysis: Scikit-learn vs. TensorFlow

Aspect	Scikit-learn	TensorFlow
Purpose	Classical ML (SVMs, RFs)	Deep Learning
Data Scale	Small-to-medium tabular	Large-scale
Learning Curve	Low (simple API)	Moderate
Deployment	Flask/Django	TF Serving, TFLite
Community	Strong	Massive (Google-backed)

Decision Guide:

- Scikit-learn: Small datasets, interpretability

- TensorFlow: Scalable deep learning