AI/ML Questions Report

Q1: TensorFlow vs PyTorch Differences

Primary Differences: - **PyTorch**: More Pythonic, uses dynamic computation graphs, easier debugging - **TensorFlow**: Uses static computation graphs (TF 2.x has eager execution), better for production deployment

When to Choose: - **Choose PyTorch when**: You're doing research, need fast prototyping, or prefer Python-like syntax - **Choose TensorFlow when**: You need production deployment, TensorFlow Serving, or mobile deployment

Q2: Jupyter Notebooks Use Cases in AI Development

Use Case 1: Rapid Prototyping - Great for experimenting with different models and parameters - Allows interactive visualization of results and intermediate outputs - Easy to modify and rerun code segments

Use Case 2: Data Exploration and Visualization - Perfect for data cleaning and preprocessing steps - Enables inline visualization of data distributions and model performance - Supports markdown documentation for explaining analysis steps

Q3: spaCy vs Basic Python String Operations

Enhancements Provided by spaCy: - **Pre-trained models**: Access to pre-trained language models for better accuracy - **Tokenization**: Advanced word and sentence tokenization that handles edge cases - **Part-of-speech tagging**: Automatic identification of word types (nouns, verbs, etc.) - **Named Entity Recognition**: Automatic identification of people, places, organizations - **Dependency parsing**: Understanding grammatical relationships between words - **Vector representations**: Context-aware word embeddings for semantic analysis

Benefits over Basic Python: - Much more accurate for complex NLP tasks - Handles linguistic nuances that basic string operations miss - Provides structured data output instead of raw text - Significantly faster for processing large amounts of text

Q4: Scikit-learn vs TensorFlow Comparison

Target Applications: **Scikit-learn**: - Classical ML algorithms (regression, classification, clustering) - Traditional ML models like SVM, Random Forest, K-means - Simple to moderate complexity tasks - Feature engineering and traditional statistical methods

TensorFlow: - Deep learning models (neural networks, CNNs, RNNs) - Large-scale complex pattern recognition - Computer vision and natural language processing - Production deployment of ML systems

Ease of Use for Beginners: **Scikit-learn**: - Simple, consistent API (fit/predict pattern) - Less complex learning curve - Built-in preprocessing and evaluation tools - Great for learning ML fundamentals

TensorFlow: - Steeper learning curve - More complex setup and debugging - Requires understanding of neural networks - More flexible but harder to master

Community Support: **Scikit-learn**: - Established, mature library - Strong academic and industry adoption - Extensive documentation and examples - Large user base for traditional ML problems

TensorFlow: - Massive corporate backing (Google) - Huge community for deep learning - Extensive tutorials and courses - Better support for production deployment