GROUP 62: AI TOOLS ASSIGNMENT

Question 1: Explain the primary differences between TensorFlow and PyTorch. When would you choose one over the other?

1. **Execution mode:**
    - TensorFlow (v1.x): Uses a static computation graph (define-and-run), where you first define the model and then run it in a session.
    - PyTorch: Uses a dynamic computation graph (define-by-run), which is more intuitive and Pythonic, allowing immediate execution of operations.
    - **Advantage**: Use TensorFlow for production-scale workloads where static graphs offer performance optimization and choose PyTorch for flexibility and ease of debugging (especially in research).
2. **Debugging and Developer Experience:**
    - PyTorch integrates seamlessly with Python tools like pdb, and its code feels like regular Python.
    - TensorFlow, especially in v1, had a steeper learning curve. TF 2.x improved this with better debugging tools and Pythonic syntax.
    - **Advantage:** Choose TensorFlow if deployment to multiple platforms (especially mobile or web) is a priority. PyTorch is catching up in production readiness with better tools each year.
3. **Model Deployment and Production Support**
    - TensorFlow has TensorFlow Serving, TensorFlow Lite, and TensorFlow.js for serving models on different platforms (cloud, mobile, web).
    - PyTorch has TorchServe, and TorchScript for model serialization and deployment, but the ecosystem is newer and smaller.
    - **Advantage**: Choose TensorFlow if deployment to multiple platforms (especially mobile or web) is a priority. PyTorch is catching up in production readiness with better tools each year.
4. **Community and Ecosystem**
    - PyTorch is dominant in academic research. Many new papers and models are published with PyTorch.
    - TensorFlow is strong in industry and enterprise, especially due to Google's backing and tools like Keras, TFLite, and TensorBoard.
    - **Advantage**: For research: PyTorch. For enterprise-level solutions: TensorFlow.
5. **Integration with Other Tools**
    - TensorFlow integrates well with the TensorBoard for visualization and TPUs for training acceleration.
    - PyTorch integrates well with Weights & Biases, Hugging Face Transformers, and is easier to use with NumPy and native Python.

Question 2: When to choose which?

1. **Choose PyTorch if:**

- You're doing research or prototyping.
- You value code simplicity, readability, and debugging ease.
- You're using transformers, Hugging Face, or other research-friendly tools.

2. **Choose TensorFlow if:**
   - You're deploying to mobile, web, or enterprise-scale systems.
   - You need mature tools for production deployment.
   - You plan to use TPUs for high-performance training.

Question 3: Describe two use cases for Jupyter Notebooks in AI development.

1. **Exploratory Data Analysis (EDA) and Model Prototyping**
   - Use case: Data scientists and AI engineers use Jupyter Notebooks to explore datasets, clean data, visualize distributions, and quickly test machine learning models.
2. **Educational Demonstrations and Research Documentation**
   - Use case: Educators, researchers, and students use notebooks to explain AI concepts, algorithms, and model behavior in a readable, interactive format.

Question 4: How does spaCy enhance NLP tasks compared to basic Python string operations?

- spaCy significantly enhances NLP (Natural Language Processing) tasks compared to basic Python string operations by offering linguistically intelligent, high-performance tools.
- spaCy: Parses sentences using tokenization, part-of-speech tagging, dependency parsing, and named entity recognition (NER)—enabling deep understanding of language structure.
- spaCy is built in Cython and optimized for performance, making it faster than many other NLP libraries.

Comparative Analysis

1. Types of Applications

Criteria	**Scikit-learn**	**TensorFlow**
Type of ML	Classical Machine Learning (e.g., regression, SVMs)	Deep Learning (neural networks, CNNs, RNNs, transformers)
Best for	Small to medium tabular datasets	Large-scale image, text, audio, and sequential data
Typical Use Cases	Spam detection, credit scoring, churn prediction	Image classification, NLP tasks, time series forecasting
Model Complexity	Shallow models (trees,	Deep, layered models with

	ensembles, linear models)	millions of parameters

2. Ease of Use for Beginners

Criteria	**Scikit-learn**	**TensorFlow**
API Simplicity	Very beginner-friendly, clean, consistent	More complex, especially before TF 2.x
Code Structure	Single-line model training: model.fit(X, y)	Multi-step model building (esp. with Keras)
Learning Curve	Low	Moderate to high
Documentation	Excellent, clear examples	Comprehensive, but can be overwhelming

3. Community Support

Criteria	**Scikit-learn**	**TensorFlow**
Community Size	Large, especially in academic and business use	Massive, backed by Google and used widely in industry
Tutorials/Resources	Extensive, beginner-friendly	Huge variety, especially for deep learning
Ecosystem	Limited to classical ML pipelines	Vast: TensorBoard, TFLite, TF-Serving, etc.
Research & Industry	Common in data science and teaching	Dominant in deep learning research and deployment