Part 1: Theoretical Understanding

1. Short Answer Questions

Q1: What are the main differences between TensorFlow and PyTorch? When would you choose one over the other?

The main difference is in how they build and run models.

- **PyTorch** uses a *dynamic* computation graph, meaning it builds the graph as the code runs. This makes it flexible, easier to use, and simple to debug. Because of that, it's very popular among researchers who need to test and change their models often.

- **TensorFlow** (especially in version 1.x) used a *static* graph, where you build the whole model first and then run it. Newer versions (TF 2.x) allow "eager execution," which works more like PyTorch, but TensorFlow is still best for big, production-level projects.

When to choose:

- **Use PyTorch:** When you're doing research, experimenting, or building complex models (like in NLP). It's more flexible and beginner-friendly.

- **Use TensorFlow:** When you're deploying large models to production, mobile, or web apps (with TensorFlow Lite or TensorFlow.js). It's powerful, optimized, and has great tools like TensorBoard and TensorFlow Serving.

Q2: What are two ways Jupyter Notebooks are used in AI development?

1. **Interactive Prototyping:**
   In Jupyter Notebooks, you can run code step by step in separate cells. This makes it easy to test small pieces of your AI workflow loading data, training models, checking results, and adjusting code without restarting everything.

2. **Data Visualization and Storytelling:**
   Jupyter can show charts, tables, and images right next to your code. This helps in exploring data, explaining model results, and sharing insights with teammates or in reports.

Q3: How does spaCy make NLP tasks better than using basic Python string functions?

Basic string methods like `.split()` or `.find()` only handle text at the character level they don't understand language or meaning.

spaCy goes much further by using pre-trained models that understand grammar and context. It can:

- Break sentences into words properly (e.g., turns “don’t” into “do” and “n’t”).
- Identify parts of speech (like verbs, nouns, or adjectives).
- Find how words relate to each other (dependency parsing).
- Recognize names of people, places, or organizations (Named Entity Recognition).

In short, spaCy doesn’t just read text it understands it, which makes it great for real NLP tasks.

2. Comparative Analysis

Scikit-learn vs TensorFlow

Criteria	Scikit-learn	TensorFlow
Target Applications	Best for *classical* machine learning tasks like regression, classification, or clustering. Great for models like Decision Trees, SVMs, and K-Means.	Best for *deep learning* — large, complex models like CNNs for images and Transformers for NLP.
Ease of Use (Beginners)	Very easy to learn and use. The steps are simple: `import`, `model.fit()`, `model.predict()`. Perfect for beginners learning ML basics.	Harder for beginners. Keras (built into TensorFlow) makes it easier, but understanding tensors and graphs still takes time.
Community Support	Has been around for years. You’ll find lots of tutorials, guides, and examples for classical ML problems.	Huge and growing fast. Backed by Google with tons of resources, papers, and pre-trained deep learning models.

Part 3: Ethics & Optimization

Ethical Review (Task 3 – Amazon Reviews Model)

Bias Problems Found:

- **Cultural Bias:** The spaCy model is trained mostly on Western data. It may not recognize brands or slang from other parts of the world, leading to unfair results.

- **Reporting Bias:** People usually write reviews only when very happy or angry. The model then learns mostly extreme emotions and struggles with neutral ones.

How to Fix It:

1. **Use TensorFlow Fairness Indicators (TFFI):** If we had user details (like country), we could check if the model performs worse for some groups and retrain it fairly.

Use spaCy's Rule-Based Additions:
Add specific rules to help spaCy recognize missing brands or names. For example:

```
 ruler = nlp.add_pipe("entity_ruler")
patterns = [{"label": "ORG", "pattern": "Samsung"}]
ruler.add_patterns(patterns)
```

This ensures the model treats all brands fairly and understands data from different regions.