Part 1: Theoretical Understanding (40%)

1. Short Answer Questions

Q1.The primary differences between TensorFlow and PyTorch.

	PyTorch	TensorFlow
computational graph	Dynamic Graph - As it operates imperatively, meaning code is run line by line the computational graph is built on the fly as operations are executed.	Static graph - In its early versions (v1.x), you had to define a static computational graph first by describing how data flows then execute it later in session, however TF (v2.x) **introduced "Eager Execution"**, making it more PyTorch-like and dynamic.
Debugging	Easy - uses standard python debuggers.	Historically harder - Due to the nature of the static graph but Improved significantly with **Eager Execution** in TF 2.x
Ecosystem and Libraries	Relies on external tools and libraries becoming community-driven, modular, and fast-evolving .	The integrated MLOps ecosystem feels like a **one-stop shop** for ML lifecycle management.

When would you choose one over the other?

Use case	Best choice	Reason
Prototyping and research	PyTorch	The intuitive, Pythonic nature and excellent debugging experience allow you to iterate quickly on ideas.
Deploying models to production	TensorFlow	Offers a more mature and extensive ecosystem for deploying models across various platforms (web, mobile, embedded systems) using specialized tools like TensorFlow Serving and Lite
Using TPUs	TensorFlow	Native TPU support and optimization.
Working in Academia	PyTorch	The research community has largely standardized on PyTorch, making collaboration and code sharing easier.

Q2: Describe two use cases for Jupyter Notebooks in AI development.

1. **Exploratory Data Analysis -** loading dataset from various sources i.e CSV, databases and JSON. Then clean the data by inspecting and handling the missing values,outliners and perform transformations.
    - Visualization - Using libraries such as Matplotlib ,Seaborn and Plotly to create inline plots to visualize distributions and to understand data correlations, and relationships.
2. **Model Prototyping and Experimentation -** Prototyping models using frameworks like scikit-learn, TensorFlow, PyTorch, or Keras which allows for fast, iterative testing of ideas.
    - Trying multiple algorithms for comparison in side by side cells (e.g Logic Regression vs Random Forest) while also monitoring the training progress with live metrics and graphs.

Q3: How does spaCy enhance NLP tasks compared to basic Python string operations?

1. The major factor that distinguishes spaCy from basic Python string operations is its use of linguistic intelligence.
    - spaCy uses **natural language models** that analyze **syntax, semantics, and entities**, allowing it to perform the following;
        i. **Tokenization:** It breaks the text into **meaningful tokens** (words, punctuation, contractions, etc.)
        ii. **Entity Recognition**:Detect **named entities** (people, places, dates)
        iii. Understand **word relationships** and **sentence structure**
        iv. **Linguistic Tagging**: It assigns Part-of-Speech (POS) tags (Noun, Verb, Adjective, etc.) and performs Lemmatization (reducing a word to its base form, e.g., "running" to "run").
    - Basic Python string operations (like .split(), .replace(), or .find()) handle text literally, without understanding meaning, grammar, or context.

2. Comparative Analysis

- Compare Scikit-learn and TensorFlow in terms of:

	Scikit-learn	TensorFlow
Target applications	Primary focus is classical **Machine Learning**.	Primary focus is **Deep learning** and Large scale numerical computation.
Ease of use for beginners.	**Gentle** - widely recommended as the **starting point** for machine learning beginners. Its consistent and simple API allows users to quickly implement and compare different traditional algorithms like Logistic Regression, Random Forests, and K-Means clustering with minimal code.	**Steeper** - TensorFlow's ease of use has improved dramatically, primarily due to the full integration of Keras as its high-level API. However,its design to give you fine-grained control for building custom, complex deep learning models creates a steeper learning curve.
Community support.	Large and active community focused around data scientists, analysts, and researchers in fields that rely on traditional statistics and tabular data.	Massive global community - The community is split between cutting-edge AI research (new model architectures, papers) and industrial deployment (mobile, web, edge devices).

Part 3: Ethics & Optimization (10%)

01. Ethical Considerations

a. Identify potential biases in your MNIST or Amazon Reviews model. How could tools like **TensorFlow Fairness Indicators** or **spaCy's rule-based systems** mitigate these biases?

i. Understanding the bias:

1. In MINST, dataset may overrepresent certain handwriting styles, ages, or nationalities (e.g., mostly digits written by adults or people using a Western writing style) this causes the model to perform well on the handwriting with similar style and poorly on the ones with a different style.

    a. spaCy Rule-Based Systems - Adds controlled linguistic rules to correct biased model behavior

2. As for Amazon Reviews,uneven distribution of reviews leading to positive reviews for popular products, or biased language based on product category, region, or reviewer demographics causing unfair predictions on certain less popular brands.

    a. In the **Amazon Reviews** dataset, **TensorFlow Fairness Indicator** could show that the sentiment model is less accurate for reviews written in non-standard English or by non-U.S. customers — signaling a need to improve representation or preprocessing.

    b. In the Amazon Reviews,**spaCy's rule-based systems,**if the model unfairly associates the word "cheap" with negative sentiment, a rule-based override could help interpret "cheap but reliable" as positive in context.