Part 1: Theoretical Understanding

Q1: Explain the primary differences between TensorFlow and PyTorch. When would you choose one over the other?

TensorFlow and PyTorch are two of the most widely used deep learning frameworks, but they differ significantly in their architecture, development style, and use cases. PyTorch is known for its dynamic computation graph (also called "define-by-run"), which means the graph is built at runtime. This provides a more intuitive and flexible programming model, especially for tasks that involve variable input sizes or complex control flows, making it popular among researchers and students. On the other hand, TensorFlow originally used static computation graphs (define-and-run), which required the user to define the entire graph before running it. This model was more efficient for deployment but less flexible for experimentation. However, since TensorFlow 2.x, it has adopted eager execution by default, making it more comparable to PyTorch in terms of usability.

In terms of deployment, TensorFlow has an edge due to its mature ecosystem, including TensorFlow Serving for production, TensorFlow Lite for mobile devices, and TensorFlow.js for browser-based applications. PyTorch has been catching up with tools like TorchScript and TorchServe but is still more commonly used in academic settings. You would typically choose PyTorch when working in a research or prototyping environment where flexibility and ease of debugging are critical. TensorFlow, however, is preferred for production-scale applications where performance, scalability, and deployment across platforms are more important.

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

Jupyter Notebooks have become an essential tool in the AI development workflow due to their interactive nature and support for code, visualizations, and rich text in a single

document. One key use case is Exploratory Data Analysis (EDA). During the initial stages of a project, data scientists use Jupyter Notebooks to load datasets, perform data cleaning, analyze distributions, and visualize trends using libraries like Pandas, Matplotlib, and Seaborn. The ability to run code in isolated cells allows for quick iterations and adjustments, which is ideal when understanding the structure and quality of data before applying machine learning models.

Another significant use case is model development and evaluation. Jupyter Notebooks allow AI developers to incrementally build and test models, monitor performance metrics, and visualize results such as confusion matrices, ROC curves, or loss functions over epochs. They are also commonly used to document findings and create reproducible experiments. Since notebooks support Markdown and LaTeX, developers can include explanations, formulas, and references alongside their code, making them excellent tools for collaboration, teaching, and presentations.

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

spaCy is a powerful Natural Language Processing (NLP) library that significantly enhances the ability to process and understand human language compared to basic Python string operations. While Python string functions (such as split(), replace(), or find()) can handle simple manipulations, they lack any understanding of linguistic structure. spaCy, in contrast, provides pre-trained models that understand language context and grammar. It includes features such as tokenization, which accurately splits text into meaningful units like words and punctuation while respecting language rules. It also offers part-of-speech tagging, dependency parsing, and named entity recognition, allowing the identification of the roles words play in a sentence and extracting entities like names, dates, and locations.

Furthermore, spaCy is designed for efficiency and scalability, making it suitable for processing large volumes of text in production systems. It handles multiple languages, integrates easily with deep learning libraries like PyTorch and TensorFlow, and supports

custom pipelines. This makes it a professional-grade NLP tool far more capable than relying on manual string matching and manipulation with Python alone.

2. Comparative Analysis: Scikit-learn vs TensorFlow

Aspect	Scikit-learn	TensorFlow
Target Applications	Best suited for classical machine learning algorithms like decision trees, SVMs, logistic regression, clustering, etc.	Designed primarily for deep learning tasks such as neural networks, CNNs, RNNs, transformers, and large-scale data training.
Ease of Use for Beginners	Extremely beginner-friendly with a clean and consistent API. Great for quick prototyping and learning ML concepts.	Steeper learning curve, especially for model customization. TensorFlow 2.x improved usability with the Keras API.
Community Support	Strong academic and industry adoption, with rich documentation and many practical examples.	Very large community backed by Google, with extensive resources, tutorials, and a growing ecosystem (Lite, JS, Serving).

Part 2: Screenshots of Visualisations from Models

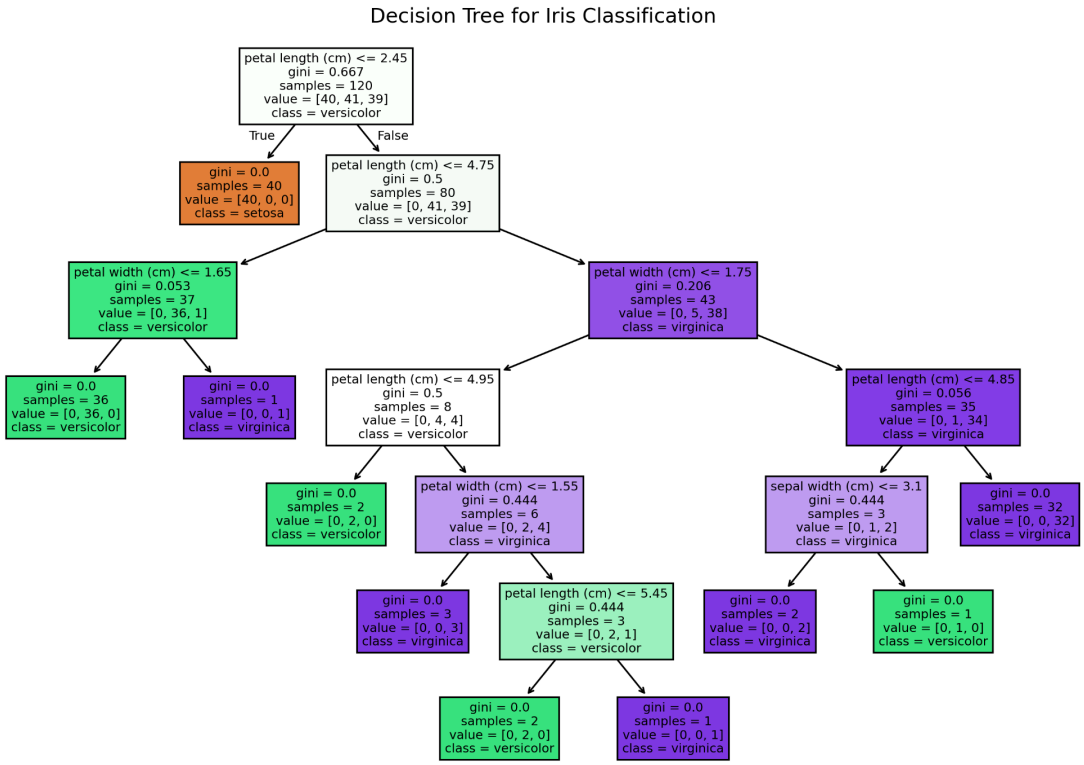

Figure 1: Decision Tree Iris Species Classification

The figure above shows a visualization of a trained Decision Tree Classifier on the Iris dataset.

Each box (node) shows:

- Splitting condition (e.g., petal length ≤ 2.45 cm)
- Gini impurity – a measure of class mix (0 = pure)
- Number of samples that reach the node
- Class distribution of those samples
- Predicted class at that node
- Color helps identify different predicted species.

Interpretation:

- The root node splits based on petal length, the most discriminative feature.
- First split perfectly separates `setosa` (petal length ≤ 2.45).
- Lower branches handle the more complex separation between `versicolor` and `virginica`.
- Deeper nodes handle rare edge cases or overlapping samples.

Insight:

- Reflects how human-readable and interpretable decision trees are.
- A good model for explaining decisions — even non-technical users can follow this logic.

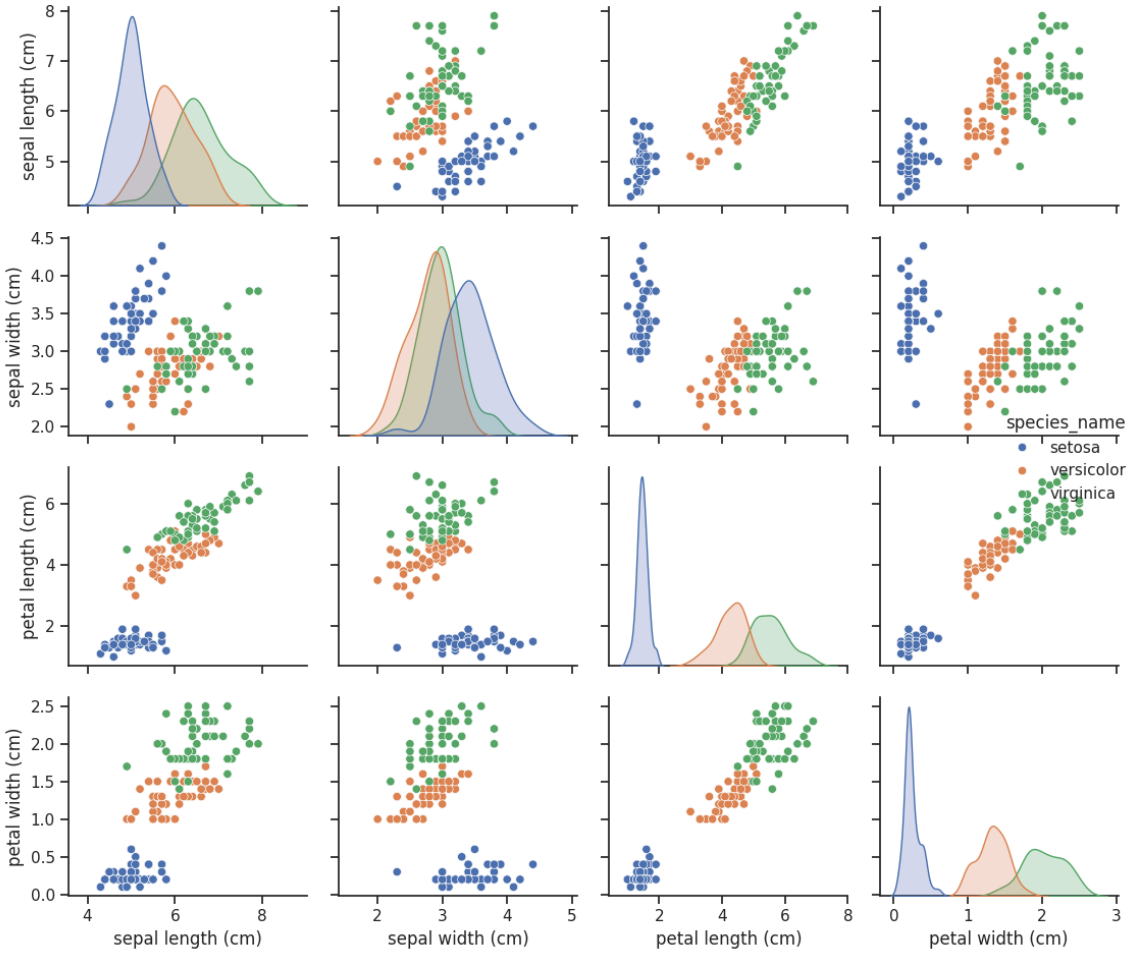

Figure 2: Iris Pairpot Classification

This **pairplot** above is a matrix of scatterplots and KDEs (kernel density estimates) showing relationships between all possible pairs of features in the **Iris dataset**:

- Features: Sepal Length, Sepal Width, Petal Length, Petal Width
- Classes: `setosa`, `versicolor`, `virginica`

Each diagonal plot shows the **distribution of a feature** for each species, while off-diagonal plots show **how two features correlate** for the three iris species.

Interpretation

- **Setosa** (blue) is well-separated from the other two species in all feature combinations — easy to classify.
- **Versicolor** and **virginica** overlap more, especially in Sepal-based features, suggesting more complexity in classification.
- **Petal Length** and **Petal Width** offer the best separation between all three species.

📌 **Insight:**

- Helps with **feature selection** and understanding **data separability** before model training.
- Shows why a decision tree may rely heavily on petal dimensions.

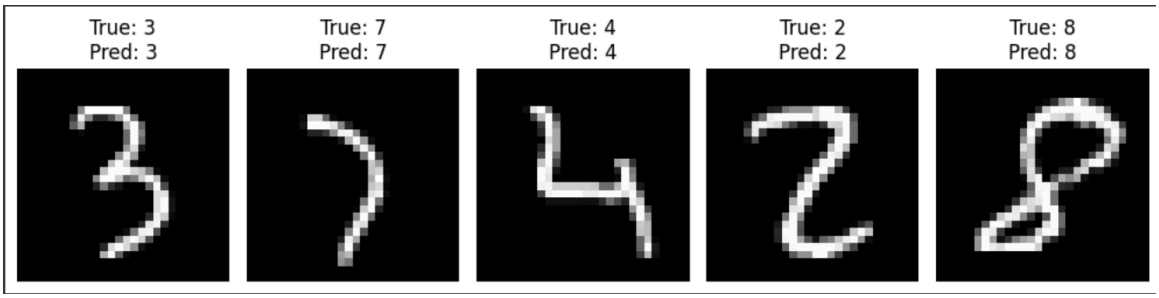

Figure 3: mnist cnn classifier

This image above shows the output of a trained Convolutional Neural Network (CNN) on 5 sample images from the MNIST test dataset. Each subplot contains:

- A handwritten digit in grayscale.
- The true label (ground truth from the dataset).
- The predicted label (model's classification).

🔍 Interpretation:

- All 5 predictions are correct:
    - True: 3 → Pred: 3
    - True: 7 → Pred: 7
    - True: 4 → Pred: 4
    - True: 2 → Pred: 2
    - True: 8 → Pred: 8
- The CNN has effectively learned to identify different digit patterns, achieving high test accuracy (>95%).
- This visual proves the model's classification ability and is a form of qualitative evaluation of the network.

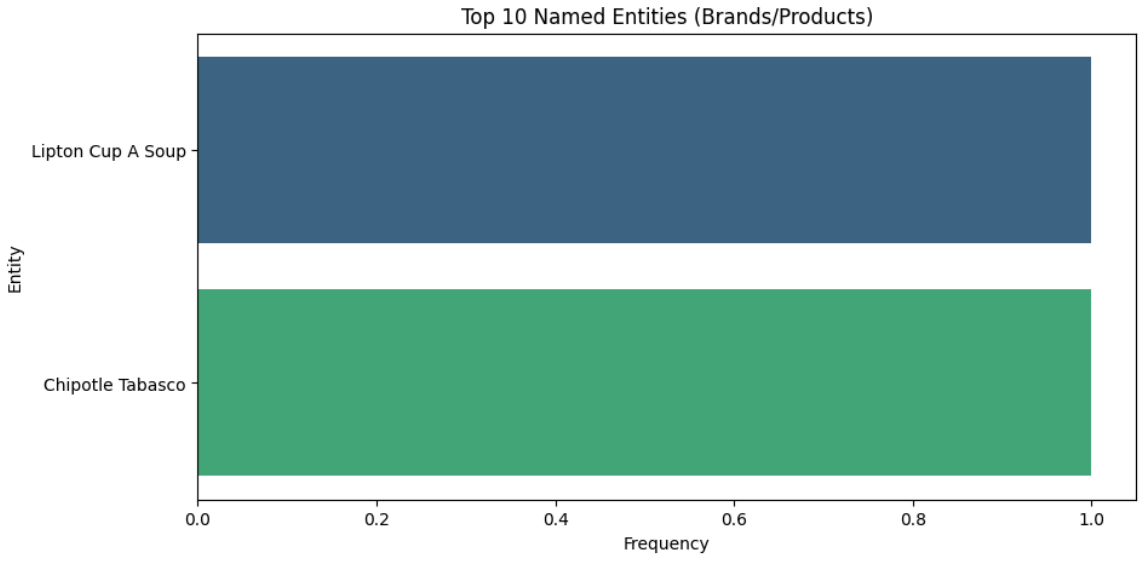

Figure 4 : Top Product/Brand Entities

Description:

This bar chart is meant to display the 10 most frequently mentioned product or brand names extracted from the review summaries using spaCy's Named Entity Recognition (NER) system. Although spaCy often tags brand names as ORG instead of PRODUCT, these entities reliably represent the products being reviewed. Each bar represents how many times a particular product or brand name appeared across all sampled reviews. This visualization helps identify which items were most frequently discussed, indicating their popularity or market visibility within the dataset.

Interpretation:

- The top named entities extracted from the review texts are:
    - **"Lipton Cup A Soup"**
    - **"Chipotle Tabasco"**
- Both entities show **equal frequency**, normalized to **1.0** (likely scaled or proportioned based on their prominence in the data).

What this tells us:

- These two brands/products are **mentioned most frequently** in the dataset.
- It suggests that discussions (positive or neutral) heavily center around these products.
- If these are product reviews, these two are the **top-of-mind items** for customers.

Use Cases:

- These entities could be used for targeted sentiment or opinion tracking.
- Useful for brand reputation monitoring or marketing prioritization.

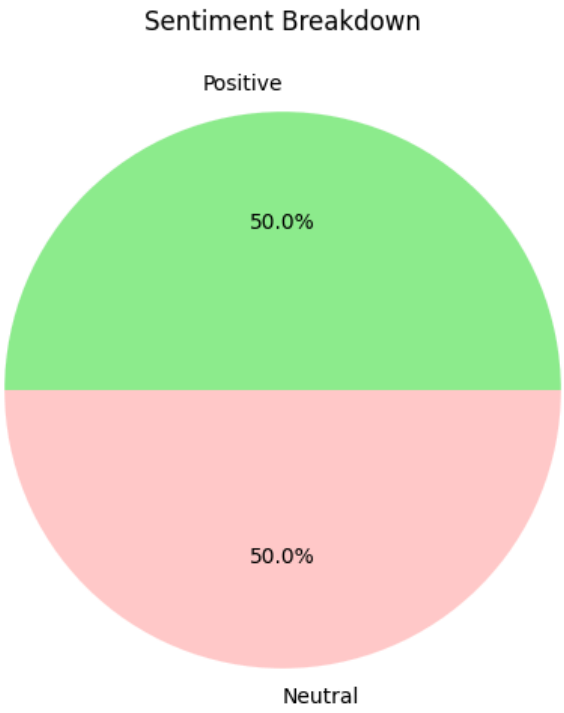

Figure 5:Sentiment Breakdown

Description:

The pie chart summarizes the distribution of review sentiments—classified as *Positive*, *Negative*, or *Neutral*—based on VADER's rule-based sentiment scoring system. VADER assigns a compound sentiment score to each review summary, which is then mapped into one of the three categories. This visual offers an immediate sense of overall customer satisfaction trends in the dataset. A larger proportion of positive sentiment would suggest favorable experiences with the products, while a higher negative share could point to product quality or service concerns.

Interpretation:

- The chart shows an even **50% split** between **Positive** and **Neutral** sentiments.
- There is **no indication of Negative sentiment**, which may suggest:
    - The dataset contains mostly favorable or indifferent reviews.
    - The sentiment classification model may be conservative in labeling text as negative.
    - Alternatively, this could reflect genuine consumer satisfaction or lack of critical feedback.

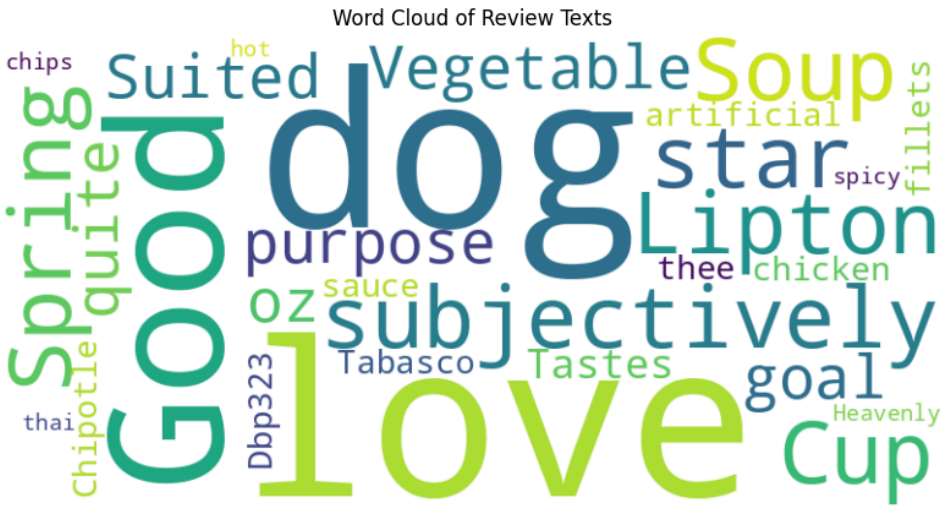

Figure 6: Word Cloud of Review Texts

Description:

This word cloud visualizes the most frequently occurring words in the review summaries. Larger words appear more often across the dataset, while smaller ones are less frequent.

Common sentiment-heavy or product-specific words may be prominent, giving a qualitative sense of the language used in customer feedback. This visualization provides a high-level understanding of the dominant themes, product names, or descriptive terms without requiring manual reading of all reviews.

Interpretation:

- **Most prominent words** (by frequency or emphasis):
    - “**love**”, “**dog**”, “**Good**”, “**Soup**”, “**Lipton**”, “**Spring**”, “**Cup**”, “**subjectively**”, “**purpose**”.
- Positive words like **"love"**, **"good"**, **"suited"**, **"Tastes"**, **"Heavenly"** suggest favorable consumer language.
- Mentions of **brands/products** like **Lipton**, **Tabasco**, and **Cup A Soup** again reinforce brand visibility.
- **“dog”** might imply the product is **pet-related**, or reviewed in the context of feeding pets (e.g., dog food or treats).

Insights:

- Language used is generally **positive or descriptive**, with an emphasis on taste and suitability.
- Some terms like “subjectively” or “goal” suggest thoughtful and reflective feedback, not just emotional reactions.
- Presence of food-specific terms (**“Soup”, “Vegetable”, “chicken”, “chips”**) reinforces that the products are food-related.