James Maina

Week 3 - AI Tools Assignment:

1. Short Answer Questions

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

Part 1

The key differences of TensorFlow and PyTorch lies in their **computational graph execution**, **ease of use/debugging**, and **ecosystem maturity/deployment focus**.

Feature	**TensorFlow (with Eager Execution)**	**PyTorch**
Computational Graph	Primarily static (though TF 2.x adopted Eager Execution, making it dynamic by default, but still excels with graph-based optimization).	**Dynamic** (Define-by-Run). Graphs are built on the fly.
Ease of Use/Debugging	Improved significantly with Keras integration and Eager Execution, but historically had a steeper learning curve.	Generally considered more **Pythonic** and intuitive, leading to easier debugging due to the dynamic nature.
Ecosystem & Deployment	Very **mature ecosystem**, strong support from Google, excellent tools for **production deployment** (TensorFlow Serving, TensorFlow Lite).	Rapidly growing, very popular in the **research community**. Deployment tools like TorchServe are available but TensorFlow often has an edge in established production pipelines.
Community Focus	Strong in **industry** and large-scale enterprise solutions.	Very strong in **academia** and cutting-edge **research** due to flexibility.

Part 2: When to use the tools

Choose PyTorch when:

- **Research and Experimentation are the priority:** Its **dynamic computation graph** offers superior flexibility for building novel, complex, or unconventional model architectures, making debugging during development much easier.
- **You prefer an intuitive, "Pythonic" experience:** Many find PyTorch's API feels closer to standard Python programming, which can result in cleaner and more readable code for rapid prototyping.
- **You are following the latest academic papers:** A significant portion of the cutting-edge research community implements their models in PyTorch.

Choose TensorFlow when:

- **Production Deployment and Scale are paramount:** TensorFlow has a more robust and mature suite of tools specifically designed for taking models to **production**, including serving, mobile deployment (TensorFlow Lite), and web deployment (TensorFlow.js).
- **You require strong enterprise support or specific hardware integration:** Backed by Google, it integrates seamlessly with Google Cloud Platform and has excellent, optimized support for **TPUs** (Tensor Processing Units).
- **You are working on mobile or embedded AI projects:** is a well-established solution for this space.

Ultimately, both frameworks are highly capable, and many advanced practitioners become proficient in both to leverage the strengths of each platform.

Use Case	**Best Choice**	**Why**
The E Research / Experimentation / Prototyping	**PyTorch**	Easier debugging, dynamic graphs, intuitive
Production / Deployment at Scale	**TensorFlow**	Mature deployment tools, mobile/web/edge support
Education / Learning Deep Learning Concepts	**PyTorch**	Simpler syntax, great for beginners
Enterprise or Cross-platform Apps (e.g., mobile)	**TensorFlow**	TensorFlow Lite + TensorFlow.js ecosystems
Academic papers and model replication	**PyTorch**	Most open-source research codebases use PyTorch

Question 2: Describe two use cases for Jupyter Notebooks in AI development.

1.Rapid Model Prototyping and Experiment Tracking

Jupyter Notebooks excel at quickly testing ideas and comparing different model configurations.

- **Quick Iteration:** Developers can swiftly define a model architecture (using **TensorFlow/Keras** or **PyTorch**), set up a training loop, run a few epochs, and immediately inspect the resulting **training curves** (loss and accuracy) right below the training cell. This rapid feedback loop is crucial for tuning hyperparameters like learning rate or batch size without the overhead of setting up traditional script files.
- **Experiment Documentation:** By combining code, the specific **hyperparameters** used in a text cell, and the **visual results** (e.g., a confusion matrix or attention maps), the notebook acts as a self-contained experiment log. This makes it easy to reproduce, share, and compare the results of various model attempts.

2. Exploratory Data Analysis (EDA) and Data Preprocessing

Jupyter Notebooks are the standard environment for the initial phases of any AI project.

- **Iterative Exploration:** You can load datasets, run descriptive statistics (like mean, standard deviation), and visualize feature distributions using libraries like **Pandas** and **Matplotlib/Seaborn**—all in sequential, executable cells. If an analysis reveals an issue (like an outlier or skewness), you can immediately modify the code in the preceding cell, re-run just that cell, and observe the updated result in-line.
- **Data Cleaning and Transformation:** Tasks like handling missing values, feature scaling, or one-hot encoding can be performed step-by-step. The **documentation cells** allow you to explain why you chose a specific transformation, creating a permanent, human-readable record of the data preparation process alongside the code itself.

Question 3: How does spaCy enhance NLP tasks compared to basic Python string operations?

While basic Python string functions handle low-level text manipulation, **spaCy** provides a **complete NLP pipeline** — turning raw text into structured, meaningful linguistic data ready for machine learning or analysis.

Aspect	Basic Python String Operations	spaCy
Text Processing Level	Works at the character or word level (e.g., .split(), .find(), .replace())	Works at the **linguistic level** (tokens, parts of speech, dependencies)
Tokenization	Requires manual splitting using delimiters (e.g., spaces or punctuation)	Uses built-in **linguistic tokenization**, handling punctuation, contractions, etc.
Part-of-Speech (POS) Tagging	Not available — must implement manually	Automatically assigns **POS tags** (noun, verb, adjective, etc.)
Named Entity Recognition (NER)	Not available	Identifies **named entities** like people, locations, organizations, dates, etc.
Lemmatization	Must write custom logic or use dictionary lookups	Built-in **lemmatizer** to get base word forms (e.g., “running” → “run”)
Dependency Parsing	Not supported	Provides **syntactic dependency trees** to show how words relate grammatically
Word Similarity	Can only compare strings literally	Uses **vector embeddings** to measure **semantic similarity** between words/sentences
Speed and Efficiency	Slow for complex NLP tasks; lacks optimization	**Highly optimized** for large-scale NLP tasks (Cython backend for speed)
Pre-trained Models	None	Offers **pre-trained statistical models** for multiple languages
Use Case	Simple text manipulation (cleaning, formatting)	**Advanced NLP tasks**: information extraction, text classification, summarization, etc.

2. Comparative Analysis

Compare Scikit-learn and TensorFlow in terms of: target applications (e.g., classical ML vs. deep learning), ease of use for beginners and community support.

A) Target Applications

Feature	**Scikit-learn (sklearn)**	**TensorFlow (TF)**
Primary Focus	**Classical Machine Learning** and data mining.	**Deep Learning** and building complex neural networks.
Ideal Tasks	Classification (e.g., Logistic Regression, SVM), Regression, Clustering (e.g., K-Means), Dimensionality Reduction (e.g., PCA), Model Selection, and Preprocessing for structured/tabular data.	Computer Vision (CNNs), Natural Language Processing (RNNs, Transformers), Reinforcement Learning, and large-scale, complex modeling with **unstructured data** (images, raw text).
Scalability	Generally designed for **small to medium-sized datasets** on a single machine.	Excellent **scalability** designed for **large datasets** and leverages hardware acceleration (GPUs/TPUs) for massive computations.

B) Ease of Use for Beginners

• **Scikit-learn:** It features a **simple, consistent, and high-level API**. A beginner can implement, train, and evaluate a standard ML model in just a few lines of code. It abstracts away complex mathematical details, making it ideal for learning core ML concepts like cross-validation and model evaluation. you can focus on the "what" (which algorithm) rather than the "how" (computational graph, tensors).

• **TensorFlow:** It has a **steeper learning curve**. While the high-level **Keras API** within TensorFlow has made it much more approachable, it fundamentally requires a deeper understanding of concepts like computational graphs, tensors, and neural network architecture. You'll often need more attention to model architecture, hyperparameters, GPU usage, memory management, deployment aspects.

C) Community Support

Both libraries boast very strong communities, but their focus differs:

scikit-learn

- Mature library, widely used in academia, data analysis and industry for classical ML. There are numerous tutorials, blog posts, StackOverflow questions, open-source examples.
- Since the API is stable and simpler, many educational resources target scikit-learn for teaching ML basics.
- Ecosystem integrates well with Python data stack (NumPy, pandas, matplotlib/seaborn). Good for data-analysis workflows.

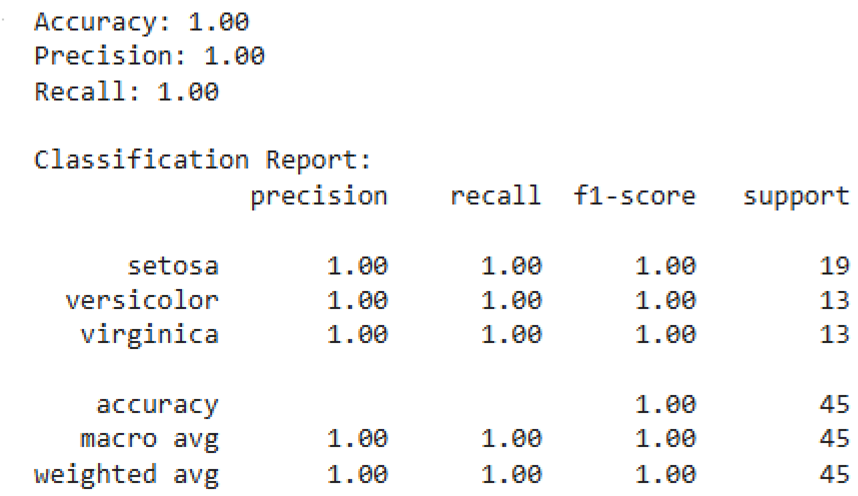
TensorFlow

- Very large community, backed by Google Brain initially, lots of resources, libraries built around it (TensorFlow Lite, TensorFlow.js, TensorBoard, etc).
- Because deep learning is a very active research/industrial area, there's a strong ecosystem of pre-trained models, tutorials, open-source code, production deployment tools.
- Some parts of the community are oriented toward advanced users (so resources may assume you know more) but beginner resources do exist as well.

SCREENSHOTS

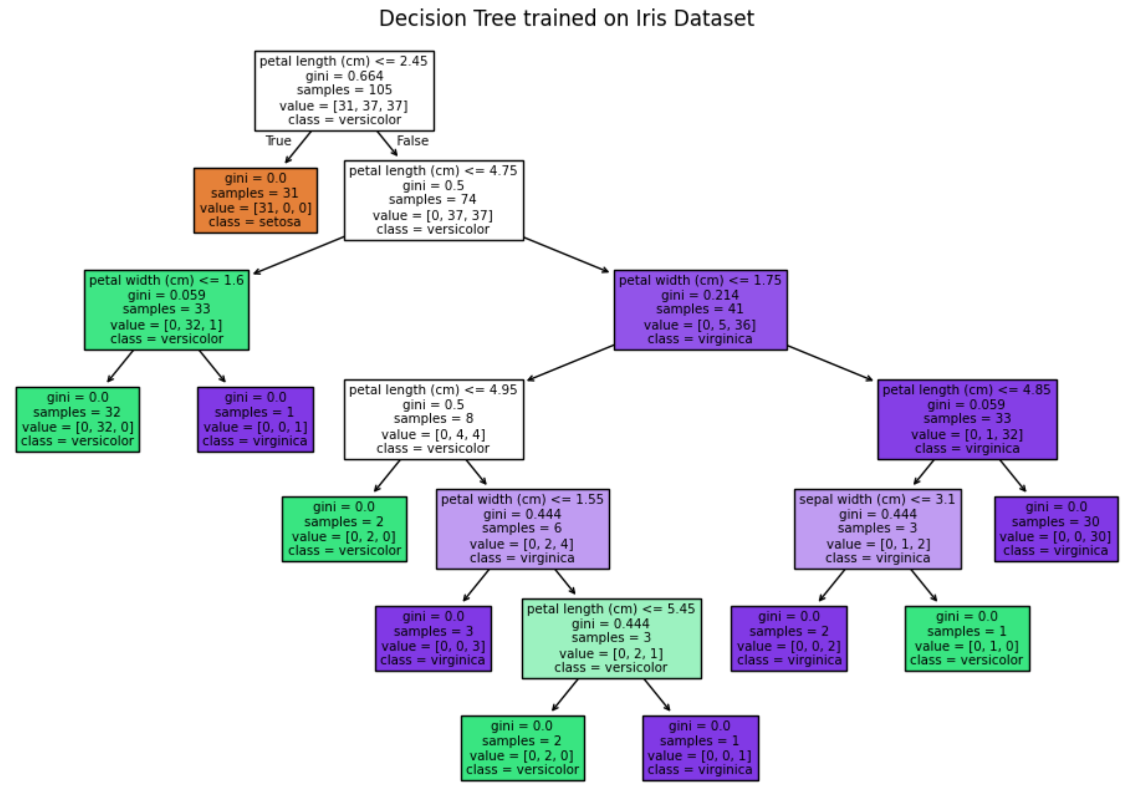
TASK 1

Figure 1: Accuracy & Loss Report



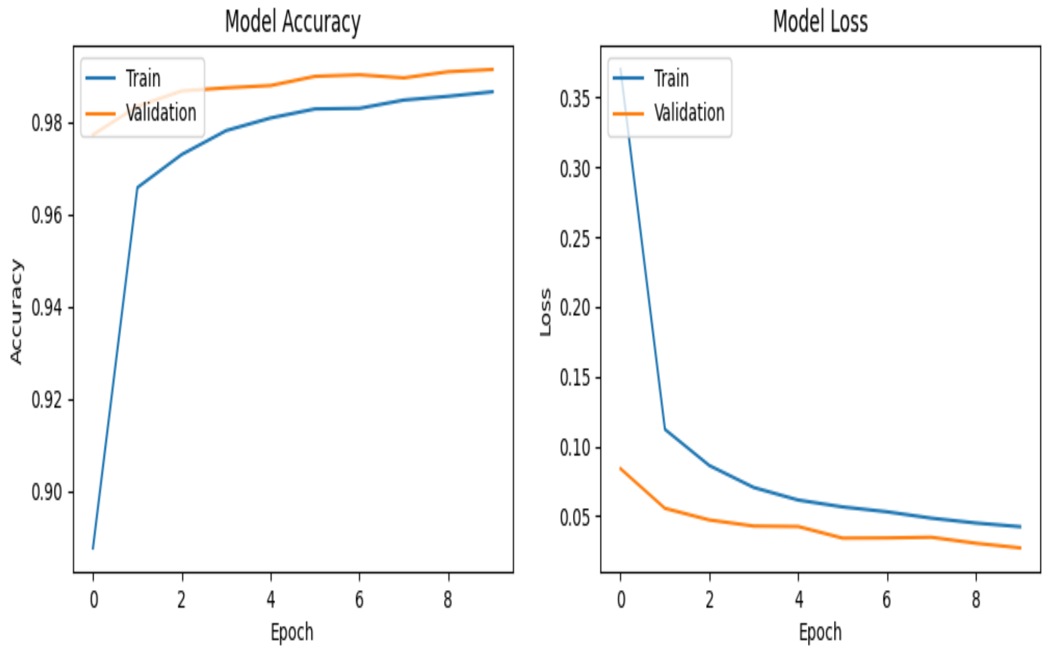
```
Accuracy: 1.00
Precision: 1.00
Recall: 1.00

Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        19
  versicolor       1.00      1.00      1.00        13
   virginica       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

Figure 2: Decision tree on trained Iris Dataset


Decision Tree trained on Iris Dataset
petal length (cm) <= 2.45 gini = 0.664 samples = 105 value = [31, 37, 37] class = versicolor
True
False
gini = 0.0 samples = 31 value = [31, 0, 0] class = setosa
petal length (cm) <= 4.75 gini = 0.5 samples = 74 value = [0, 37, 37] class = versicolor
petal width (cm) <= 1.6 gini = 0.059 samples = 33 value = [0, 32, 1] class = versicolor
petal width (cm) <= 1.75 gini = 0.214 samples = 41 value = [0, 5, 36] class = virginica
gini = 0.0 samples = 32 value = [0, 32, 0] class = versicolor
gini = 0.0 samples = 1 value = [0, 0, 1] class = virginica
petal length (cm) <= 4.95 gini = 0.5 samples = 8 value = [0, 4, 4] class = versicolor
petal length (cm) <= 4.85 gini = 0.059 samples = 33 value = [0, 1, 32] class = virginica
gini = 0.0 samples = 2 value = [0, 2, 0] class = versicolor
petal width (cm) <= 1.55 gini = 0.444 samples = 6 value = [0, 2, 4] class = virginica
sepal width (cm) <= 3.1 gini = 0.444 samples = 3 value = [0, 1, 2] class = virginica
gini = 0.0 samples = 30 value = [0, 0, 30] class = virginica
gini = 0.0 samples = 3 value = [0, 0, 3] class = virginica
petal length (cm) <= 5.45 gini = 0.444 samples = 3 value = [0, 2, 1] class = versicolor
gini = 0.0 samples = 2 value = [0, 0, 2] class = virginica
gini = 0.0 samples = 1 value = [0, 1, 0] class = versicolor
gini = 0.0 samples = 2 value = [0, 2, 0] class = versicolor
gini = 0.0 samples = 1 value = [0, 0, 1] class = virginica

TASK 2

Figure 3: Accuracy and Loss report



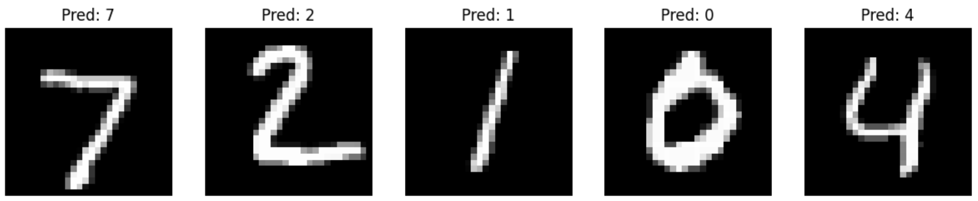
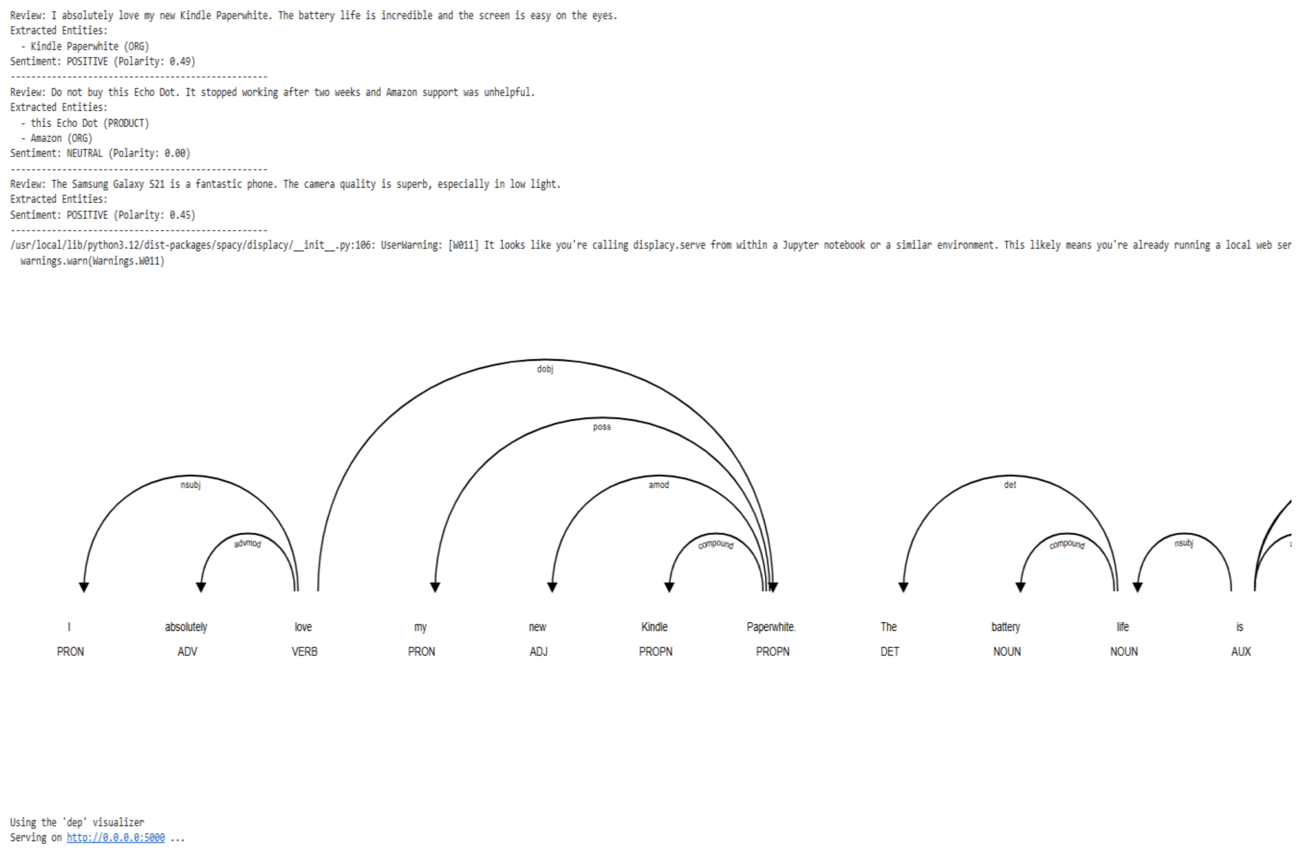


Figure 4: Number predictions from trained models

TASK 3



```
Review: I absolutely love my new Kindle Paperwhite. The battery life is incredible and the screen is easy on the eyes.
Extracted Entities:
  - Kindle Paperwhite (ORG)
Sentiment: POSITIVE (Polarity: 0.49)
--------------------------------------------------
Review: Do not buy this Echo Dot. It stopped working after two weeks and Amazon support was unhelpful.
Extracted Entities:
  - this Echo Dot (PRODUCT)
  - Amazon (ORG)
Sentiment: NEUTRAL (Polarity: 0.00)
--------------------------------------------------
Review: The Samsung Galaxy S21 is a fantastic phone. The camera quality is superb, especially in low light.
Extracted Entities:
Sentiment: POSITIVE (Polarity: 0.45)
--------------------------------------------------
/usr/local/lib/python3.12/dist-packages/spacy/displacy/__init__.py:106: UserWarning: [W011] It looks like you're calling displacy.serve from within a Jupyter notebook or a similar environment. This likely means you're already running a local web ser
  warnings.warn(Warnings.W011)
```

```
Using the 'dep' visualizer
Serving on http://0.0.0.0:5000 ...
```

Figure 5: Using spaCy to display reviews

Ethical Reflections on Use of AI Tools

Ethical Dimension	Description	Key Concerns	Ethical Responsibility / Action
Bias and Fairness	AI systems learn from historical data that may contain human bias.	Discrimination in hiring, lending, healthcare, and justice systems.	Audit datasets, apply fairness metrics, and ensure diverse, representative data sources.
Privacy and Data Protection	AI tools often require vast amounts of personal data for training and operation.	Unauthorized data collection, surveillance, and data breaches.	Ensure informed consent, data transparency, encryption, and compliance with privacy laws.
Automation and Employment	AI-driven automation can increase efficiency but also replace human workers.	Job loss, inequality, and lack of retraining opportunities.	Develop policies for retraining, equitable transitions, and human-centered automation.
Accountability and Transparency	AI decisions can be opaque, making it unclear who is responsible for errors.	Misdiagnosis, misinformation, or unfair algorithmic decisions.	Promote explainable AI (XAI), clear responsibility frameworks, and ethical governance.
Human Dignity and Autonomy	Overreliance on AI may reduce human judgment and critical thinking.	Loss of agency, manipulation through AI-driven systems.	Maintain human oversight, ensure AI complements rather than replaces human decision-making.

Ethical Dimension	**Description**	**Key Concerns**	**Ethical Responsibility / Action**
Overall Reflection	**AI can empower society but also deepen inequalities if misused.**	**Balance innovation with moral awareness and regulation.**	**Uphold fairness, privacy, transparency, and respect for human values in all AI applications.**