2025

THEORETICAL UNDERSTANDING

PLP ACADEMY

GROUP FIVE

MEMBERS

SIDNEY BARAKA MURIUKI

JECINTA NJERI KING'ATHIA

SHARON KIOKO

FESTUS PERTUS

ENOCK TAREI

EMMANUEL KIMARO

PLP ACADEMY

COURSE: SOFTWARE DEVELOPMENT

SPECIALIZATION: AI FOR SOFTWARE ENGINEERING

GROUP FIVE

THEORETICAL UNDERSTANDING

INSTRUCTOR: MR CHAKIN

JUNE 2025

Part 1: Theoretical Understanding (40%)

1. Short Answer Questions

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

TensorFlow and PyTorch are both widely used deep learning frameworks, but they differ in key ways. TensorFlow traditionally uses a static computation graph, while PyTorch uses a dynamic computation graph, which is more intuitive and easier to debug. PyTorch also has a more Pythonic and user-friendly syntax, making it ideal for rapid development.

In our group, we choose PyTorch when we focus on flexibility, easy debugging, and fast prototyping. On the other hand, we go for TensorFlow when we prioritize model deployment, especially using tools like TensorFlow Lite or TensorFlow Serving for production environments.

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

As a team, we use Jupyter Notebooks primarily for:

1. Interactive experimentation – We run code in smaller sections, test different models or parameters, and get instant feedback on performance.
2. Data exploration and visualization – We analyze datasets using tools like Pandas and visualize results with libraries such as Matplotlib and Seaborn, which helps us understand trends and patterns before modeling.

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

Compared to basic Python string methods, spaCy offers a powerful suite of natural language processing tools. It handles tasks like tokenization, part-of-speech tagging, lemmatization, and named entity recognition much more accurately.

In our projects, spaCy allows us to process language data more efficiently, extracting meaningful insights from text that would be difficult or time-consuming using only standard string operations.

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.
  - Community support.

Comparative Analysis: Scikit-learn vs. TensorFlow

Aspect	Scikit-learn	TensorFlow
Target Applications	We use it for **classical ML** like regression, SVMs, and clustering.	We use it mainly for **deep learning**, especially when building neural networks.
Ease of Use	It's very **beginner-friendly** with simple and clean APIs.	It's more complex, but TensorFlow 2.x makes it more accessible.
Community Support	It has a strong community, especially in research and education.	TensorFlow has broad **industry support** and strong documentation.

We prefer **Scikit-learn** for traditional machine learning tasks and when we need quick results without needing GPUs. We turn to **TensorFlow** when we are working on deep learning models or preparing for deployment in a production environment.