

## **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 is a developer friendly library or framework that focuses on production and scalability while PyTorch is a framework that researchers who experiment new ideas use due to its flexibility.

TensorFlow for deployment (mobile, web) and large-scale stable applications. PyTorch for research in academia, prototyping and for complex dynamic models.

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

Exploratory Data Analysis -Visualizing and cleaning data instantly

Rapid Model Prototyping- Building, training, checking and tweaking models quickly without re-running long scripts.

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

Basic Python string operations treat text as characters, while spaCy treats text as language. For example, A string operation sees "A-p-p-l-e," but spaCy sees "Company Name or an organization."

### **2. Comparative Analysis**

Compare Scikit-learn and TensorFlow in terms of:

- Target applications (e.g., classical ML vs. deep learning).

Scikit-learn -Classical Machine Learning (Classification, Regression, Clustering) on structured/tabular data for example spam filters and credit scoring.

TensorFlow- Used for Deep Learning (Neural Networks) for unstructured data for example image recognition, NLP and reinforcement learning

- Ease of use for beginners.

Scikit-learn – Very simple to use as it provides a clean, consistent API and is great for quickly testing standard algorithms due to its non-deep-learning focus.

TensorFlow- Was initially more complex, but it's integration of the Keras API makes building and training complex deep learning models very user-friendly and accessible for beginners.

- Community support.

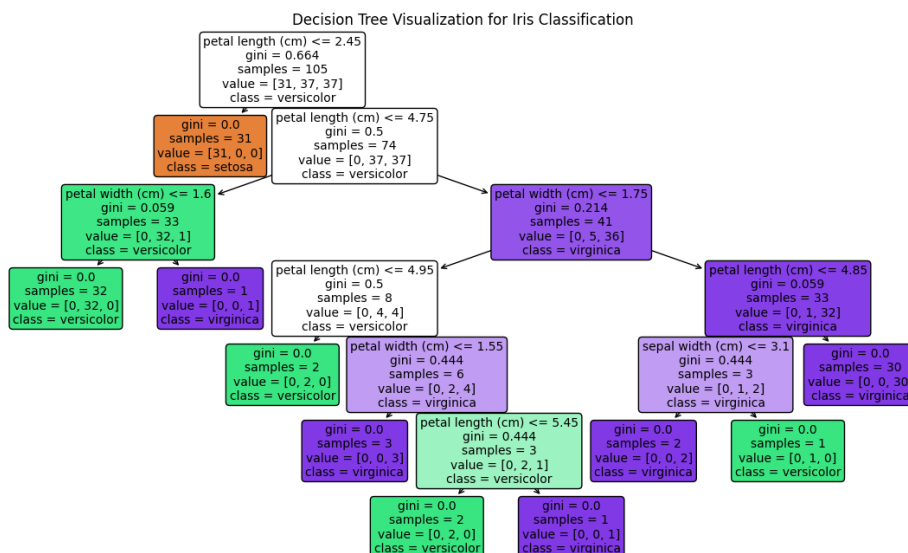
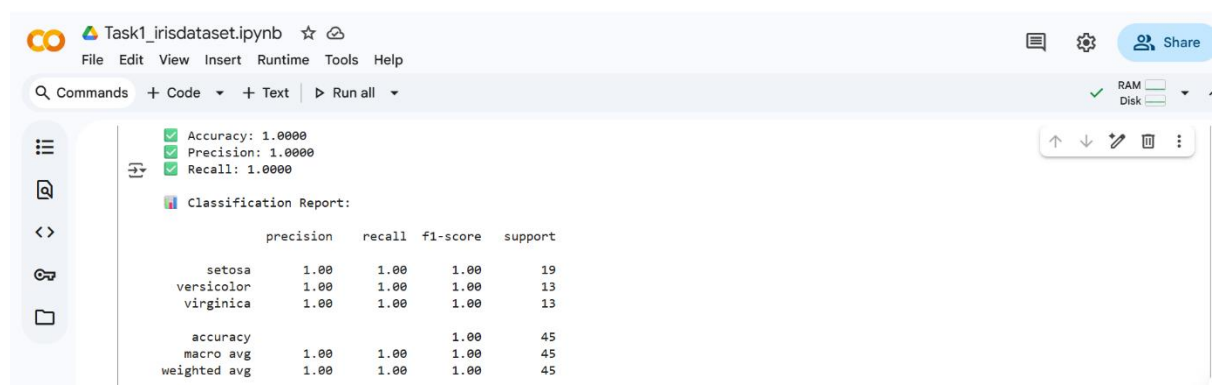
Scikit-learn – Widely used in academia and traditional Machine Learning and it's known for reliable, well-documented algorithms.

TensorFlow- It powers google, YouTube. Widely used by industries, backed by Google. It focuses on deep learning and getting complex models into real-world products.

## Screenshots of model outputs (e.g., accuracy graphs, NER results)

### Task 1: Iris Classification (Scikit-learn)

- Model evaluation output: The printed accuracy, precision, recall, and classification report.
- Decision tree plot: The visual tree generated using `plot_tree()`.



### Task 2: MNIST CNN (TensorFlow)

- Test accuracy printout-The final model.evaluate() result.
- Prediction visualization- plot showing 5 sample digits with true vs predicted labels.

The first screenshot shows the training progress of an MNIST model. The output displays the following metrics for each epoch:

Epoch	Step	Time	Accuracy	Loss	Val Accuracy	Val Loss
844/844	47s	53ms/step	0.8732	0.4328	0.9812	0.0644
Epoch 2/5	48s	56ms/step	0.9821	0.0593	0.9852	0.0499
Epoch 3/5	45s	54ms/step	0.9878	0.0400	0.9865	0.0467
Epoch 4/5	45s	54ms/step	0.9911	0.0297	0.9868	0.0479
Epoch 5/5	44s	52ms/step	0.9931	0.0225	0.9857	0.0504
313/313	3s	10ms/step	0.9821	0.0516		

The final test accuracy is 0.9852 and the test loss is 0.0439.

The second screenshot shows the prediction results for 5 sample digits. The output displays the following metrics for each digit:

Digit	Accuracy
Digit 0	0.9939
Digit 1	0.9947
Digit 2	0.9767
Digit 3	0.9980
Digit 4	0.9562
Digit 5	0.9753
Digit 6	0.9875
Digit 7	0.9903
Digit 8	0.9641
Digit 9	0.9851

The final test accuracy is 0.9825 and the test loss is 0.0524. Below the metrics, 5 sample digits are shown with their true and predicted labels:

True	Pred
6	6
2	2
3	3
7	7
2	2

### Task 3: NLP with spaCY

- NER output: Printed named entities like ('Apple iPhone 13', 'PRODUCT') and sentiment result: Printed sentiment label for each review (e.g., "Positive", "Negative").

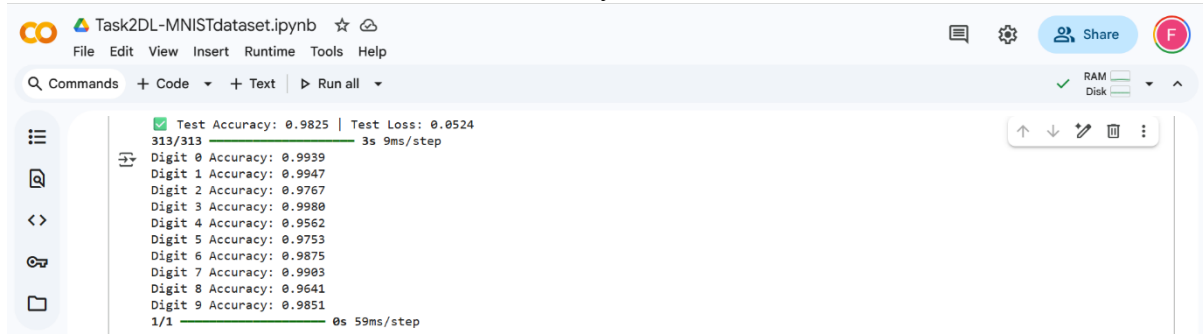
The screenshot shows the output of an NLP task using spaCy. The output displays the following reviews and their sentiment labels:

Review	Sentiment
Review: I love my new Apple iPhone 13! The camera is amazing.	Positive
Review: Terrible experience with the Samsung Galaxy. It kept freezing.	Negative
Review: The Sony headphones are comfortable and sound great.	Positive
Review: Avoid this Lenovo laptop. Battery life is awful.	Negative
Review: My new Nike running shoes are super comfy and stylish!	Positive

Below the reviews, the named entities are listed for each review:

Review	Entities
Review: I love my new Apple iPhone 13! The camera is amazing.	[('Apple', 'ORG')]
Review: Terrible experience with the Samsung Galaxy. It kept freezing.	[('Samsung', 'ORG')]
Review: The Sony headphones are comfortable and sound great.	[('Sony', 'ORG')]
Review: Avoid this Lenovo laptop. Battery life is awful.	[('Lenovo', 'ORG')]
Review: My new Nike running shoes are super comfy and stylish!	[('Nike', 'ORG')]

## Ethic Considerations Fairness Check: Accuracy MNIST dataset



The screenshot shows a Jupyter Notebook titled "Task2DL-MNISTdataset.ipynb". The interface includes a top menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". Below the menu is a toolbar with "Commands", "Code", "Text", and "Run all" buttons. On the right, there are icons for chat, settings, a "Share" button, and a user profile icon. The main area displays the output of a code cell, which shows the test accuracy and loss for the MNIST dataset. The output is formatted with a green progress bar and a table of accuracy values for each digit (0-9). The overall test accuracy is 0.9825 and the test loss is 0.0524. The output also shows the progress of the test set, with 313/313 samples processed in 3 seconds and 9 milliseconds per step. The accuracy for each digit is as follows:

Digit	Accuracy
Digit 0	0.9939
Digit 1	0.9947
Digit 2	0.9767
Digit 3	0.9988
Digit 4	0.9562
Digit 5	0.9753
Digit 6	0.9875
Digit 7	0.9903
Digit 8	0.9641
Digit 9	0.9851

The output also shows the progress of the test set, with 1/1 samples processed in 0 seconds and 59 milliseconds per step.