AI Tools and Framework Assignment

Course: AI for Software Engineering

Platform: GitHub + Google Colab

Part 1: Theoretical Understanding

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

TensorFlow is a powerful deep learning framework developed by Google. It uses static computation graphs, which are optimized for performance and deployment. It's ideal for production environments and scalable applications.

PyTorch is a dynamic graph-based deep learning framework developed by Facebook. It is more intuitive and Pythonic, making it easier for research, experimentation, and debugging.

Choice:
- Use TensorFlow for large-scale, production-ready models.
- Use PyTorch for rapid prototyping, academic research, and experimentation.

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

1. Interactive Development: Jupyter allows step-by-step execution, making it easier to test and debug machine learning models.
2. Documentation and Reporting: Code, results, and explanations can be combined in one document, supporting reproducible research and collaboration.

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

spaCy provides pre-trained pipelines and linguistic features like tokenization, lemmatization, Named Entity Recognition (NER), and part-of-speech tagging. Unlike simple Python string operations, spaCy understands grammatical structure and can extract meaningful entities and relationships from text, making it ideal for real-world NLP applications.

Comparative Analysis: Scikit-learn vs TensorFlow

Scikit-learn is best suited for classical machine learning problems like classification, regression, and clustering, while TensorFlow excels in deep learning tasks including neural networks for vision and NLP.

Feature	Scikit-learn	TensorFlow
Primary Focus	Classical ML	Deep Learning
Common Use Cases	Regression, classification	Vision, NLP, neural nets
Ease of Use	Beginner-friendly	Steeper learning curve
Deployment	Limited	TensorFlow Serving, Lite
Community	Academic	Strong production support

Part 2: Practical Implementation

Task 1: Classical ML with Scikit-learn (Iris Dataset)

- Loaded and cleaned the dataset.
- Encoded labels.
- Trained a Decision Tree Classifier.
- Evaluated using accuracy, precision, and recall.

Sample Output:
Accuracy: 0.96
Precision: 0.94
Recall: 0.95

Task 2: Deep Learning with TensorFlow (MNIST Dataset)

- Built a CNN using Keras.
- Trained the model to classify handwritten digits.
- Achieved 98.3% test accuracy.
- Visualized predictions on 5 sample images.

Sample Output:
Final Accuracy: 98.3%

Task 3: NLP with spaCy (Amazon Reviews)

- Used spaCy's en_core_web_sm model.
- Performed Named Entity Recognition to extract product names and brands.
- Used rule-based matching for sentiment analysis.

Sample Output:
Entities: ["Samsung Galaxy" → PRODUCT, "Amazon" → ORG]
Sentiment: Positive

Part 3: Ethics & Optimization

Ethical Considerations

- Bias in MNIST: Poor performance on non-digit inputs or varying handwriting styles.
- Bias in Amazon Reviews: Brand dominance can skew sentiment.

Mitigation:
- Use TensorFlow Fairness Indicators.
- Use rule-based filtering and diverse datasets.

Debugging Challenge

- Replaced incorrect loss function.
- Fixed output layer mismatch.
- Ensured label encoding matches shape.

Conclusion

This project provided hands-on experience with AI tools and frameworks. It showcased theoretical knowledge, implementation skills, and ethical development practices, giving a rounded view of AI development pipelines.