AI Tools Assignment — Mastering the AI Toolkit

Group: [Group members names]
Course: [Course name]
Date: October 15, 2025

Part 1: Theoretical Understanding

Q1: TensorFlow vs PyTorch

- TensorFlow: originally used static computation graphs; now supports eager execution via Keras. Strong production tooling (TensorFlow Serving, TF Lite, TensorFlow Hub, TFX). Well-suited for production deployments and large-scale pipelines.
- PyTorch: uses eager execution by default, more pythonic, simpler debugging and fast iteration. Widely used in research and experimentation.

When to choose:
- Use PyTorch for research and rapid prototyping. Use TensorFlow when you need TF-specific production tooling or integration with certain deployment stacks, or when your team prefers TF tooling.

Q2: Two Jupyter Notebook use cases

1. Exploratory data analysis and visualization (plots, tables, interactive widgets).
2. Prototyping models and visualizing training progress (loss/accuracy plots, confusion matrices).

Q3: How spaCy enhances NLP vs basic string operations

spaCy provides robust tokenization, POS tagging, dependency parsing, pretrained NER, and fast rule-based matchers. These features produce structured linguistic annotations and are far more reliable than brittle, ad-hoc string matching.

Comparative Analysis: Scikit-learn vs TensorFlow

- Target applications: Scikit-learn — classical ML (SVM, RandomForest, gradient boosting, preprocessing). TensorFlow — deep learning and neural networks at scale.
- Ease of use: Scikit-learn has a simple fit/predict API, great for beginners. TensorFlow (Keras) has improved usability but introduces more concepts for advanced deep learning.
- Community: Both large; TensorFlow has extensive deep-learning resources and ecosystem; Scikit-learn is foundational for applied ML pipelines.

Part 2: Practical Implementation (summary)

Task 1: Iris (Scikit-learn Decision Tree)

- Preprocessing: checked for missing values, encoded labels (already numeric).
- Model: DecisionTreeClassifier (random_state=42).
- Evaluation: accuracy, precision (macro), recall (macro). See `iris_classification.py` and `AIToolsAssignment.ipynb` for code and outputs.

Task 2: MNIST (TensorFlow CNN)

- Model: simple CNN with Conv2D(32)->MaxPool->Conv2D(64)->MaxPool->Dense(128)->Dropout->Dense(10 softmax).
- Loss: sparse_categorical_crossentropy (labels are integer-encoded). Optimizer: Adam.
- Recommendation: run on GPU in Colab and train for at least 8-10 epochs to reach >95% test accuracy. Code in `mnist_cnn.py` and notebook.

Task 3: NLP (spaCy)

- NER: used spaCy's `en_core_web_sm` model and PhraseMatcher for products/brands.

- Sentiment: toy rule-based approach counting positive/negative words. For production, prefer a trained sentiment model or fine-tune transformer-based classifiers.
- Code: `spacy_ner_sentiment.py` and notebook cell.

Part 3: Ethics & Optimization

Bias identification

- MNIST: potential bias from handwriting styles not represented in training data. Use per-slice evaluation and tools like TensorFlow Fairness Indicators to detect differing performance across slices.
- Reviews: rule-based sentiment may misclassify sarcasm or cultural expressions.

Mitigations

- Augment datasets, audit per-slice performance, use fairness evaluation tools, and collect diverse labeled data.

Troubleshooting summary

- Fixed common TF bugs in `buggy_tensorflow_fixed.py`: ensure correct input shapes, use `softmax` for multi-class and `sparse_categorical_crossentropy` when using integer labels.

Attachments and Screenshots

Include screenshots of model outputs (training curves, sample NER outputs, prediction images) in the final PDF report. Placeholders below indicate where to add them after running the code:

- Screenshot: Iris classification report — path: `screenshots/iris_report.png`
- Screenshot: MNIST training accuracy/loss plot — path: `screenshots/mnist_training.png`
- Screenshot: spaCy NER output — path: `screenshots/spacy_ner.png`

(You can run the notebook in Colab and save these screenshots, then add them in the `screenshots/` folder before producing a final PDF if you want the images embedded.)

References

- TensorFlow docs: https://www.tensorflow.org
- PyTorch docs: https://pytorch.org
- spaCy docs: https://spacy.io
- Scikit-learn docs: https://scikit-learn.org

Group reflection (ethical)

Our team reflected on bias risks, privacy of user reviews, and the need to avoid using personally identifying information in datasets. We recommend data minimization, clear consent, and transparent model cards describing limitations.