AI Tools Assignment Report: Mastering the AI Toolkit

Part 1: Theoretical Understanding

Q1: TensorFlow vs. PyTorch

TensorFlow, developed by Google, uses a static computational graph optimized for production deployment, offering scalability and tools like TensorFlow Serving. PyTorch, developed by Meta AI, uses a dynamic graph, making it ideal for research and rapid prototyping due to its flexibility and ease of debugging.

Choose TensorFlow for large-scale production systems (e.g., healthcare, finance) or mobile deployment. Choose PyTorch for research, novel architectures, or quick experimentation.

Q2: Jupyter Notebooks Use Cases

1. Exploratory Data Analysis (EDA): Jupyter Notebooks enable interactive visualization of datasets (e.g., Iris, MNIST) using Matplotlib or Seaborn to identify patterns or outliers.
2. Model Prototyping and Debugging: Notebooks allow step-by-step execution of AI model code, facilitating debugging and hyperparameter tuning during development.

Q3: spaCy vs. String Operations

spaCy is a specialized NLP library with pre-trained models for tasks like NER and tokenization, offering efficient, context-aware processing. For example, it accurately extracts entities like 'Apple' as an organization. Basic Python string operations (e.g., regex, splitting) lack linguistic context, leading to error-prone tokenization. spaCy's pipelines are language-specific and scalable, unlike manual string operations.

Comparative Analysis: Scikit-learn vs. TensorFlow

- Target Applications: Scikit-learn is ideal for classical ML (e.g., decision trees, SVMs) on small to medium datasets. TensorFlow is suited for deep learning (e.g., CNNs, RNNs) on large-scale datasets like images or text.
- Ease of Use: Scikit-learn has simple APIs (e.g., fit(), predict()), making it beginner-friendly. TensorFlow has a steeper learning curve, though Keras simplifies it.
- Community Support: Scikit-learn has strong data science support; TensorFlow has enterprise backing (Google) and tools like TensorFlow Hub.

Part 2: Practical Implementation

Task 1: Classical ML with Scikit-learn

We trained a decision tree classifier on the Iris dataset after preprocessing (scaling features, no missing values). Evaluation metrics are shown below.

[Placeholder: Insert screenshot of accuracy, precision, recall from iris_decision_tree.ipynb]

Task 2: Deep Learning with TensorFlow

We built a CNN for MNIST digit classification, achieving >95% test accuracy. Predictions on 5 sample images are shown below.

[Placeholder: Insert screenshot of mnist_predictions.png and accuracy plot from mnist_cnn.ipynb]

Task 3: NLP with spaCy

We performed NER and rule-based sentiment analysis on Amazon reviews, extracting product names/brands and labeling sentiment.

[Placeholder: Insert screenshot of NER and sentiment output from amazon_nlp.ipynb]

Part 3: Ethics & Optimization

Ethical Considerations

MNIST Model: Biases may arise from limited diversity in handwriting styles, affecting performance across demographics. TensorFlow Fairness Indicators can evaluate model performance across groups.
Amazon Reviews Model: Sentiment analysis may misclassify nuanced reviews (e.g., sarcasm) or reflect socioeconomic biases. Expanding spaCy's rules to handle negation and using diverse datasets can mitigate biases.

Troubleshooting Challenge

We fixed a TensorFlow script with a dimension mismatch by adding input_shape=(28, 28, 1) to the CNN's first layer. See debug_example.py for details.

Bonus Task: Streamlit Deployment

We deployed the MNIST classifier using Streamlit. Below is a screenshot of the web interface and a live demo link.