AI Tools and Applications

By Asanda Mthembu

1. **Short Answer questions**

Q1.

Primary Differences?

a. **TensorFlow** and **PyTorch** are both powerful deep learning frameworks, but they differ significantly in design and use cases. **TensorFlow** primarily uses a **static computation graph**, which is more efficient for deployment, though it now supports eager execution. **PyTorch** uses a **dynamic computation graph**, making it more intuitive and easier to debug—ideal for research and experimentation.

b. When it comes to **debugging**, PyTorch is often preferred because of its Pythonic nature and real-time execution. **TensorFlow**, while more complex initially, excels in **industry deployment**, thanks to robust tools like **TensorFlow Serving**, **TensorFlow Lite**, and **TensorFlow.js**, making it better suited for mobile and web deployment. PyTorch is catching up in this area with **TorchServe** and **TorchScript**, but it's not as mature.

When to choose?

a. Choose TensorFlow when building a **production-grade** application with a focus on **deployment**, **scalability**, or **cross-platform support.** You need **robust tools** like **TensorFlow Lite** for mobile, **TensorFlow.js** for browser, or **TensorFlow Serving** for model deployment. You prefer using **Keras** as a high-level API.

b. Choose PyTorch when working on **research, rapid prototyping**, or need **dynamic behavior** in your models. If you value **simplicity, Pythonic syntax**, and ease of debugging. If you want to iterate quickly and explore new ideas in deep learning.

Q2.

Two key use cases for Jupyter Notebooks in AI development

a. **Prototyping and Experimentation**

- **Jupyter** Notebooks are ideal for quickly testing and refining AI models. Developers can write code in small, manageable cells, visualize outputs immediately (e.g., model accuracy, loss curves), and tweak parameters interactively. This makes it perfect for experimenting with algorithms, testing different model architectures, and performing hyperparameter tuning during early-stage development.

b. **Data Exploration and Visualization**

- Before building AI models, understanding the dataset is crucial. Jupyter Notebooks support rich visualizations using libraries like **Matplotlib**, **Seaborn**, and **Plotly**, allowing developers to explore data distributions, detect anomalies, and clean the data efficiently. This helps ensure better model training and performance down the line.

Q3.

a. spaCy is a robust open-source library for Python, ideal for natural language processing (NLP) tasks. It offers built-in capabilities for tokenization, dependency parsing, and named-entity recognition, making it a popular choice for processing and analysing text. With spaCy, you can efficiently represent unstructured text in a computer-readable format, enabling automation of text analysis and extraction of meaningful insights.

2. **Compatative Analysis**

A. **Targe Application**

Features	**Scikit-learn**	**TensorFlow**
Primary Focus	Classical Machine Learning	Deep Learning and Neural Networks
Algorithms	Decision Trees, SVMs, Linear/Logistic Regression, Clustering	Deep Neural Networks, CNNs, RNNs, Transformers
Best used for	Small to medium structured datasets and traditional ML tasks	Complex models, unstructured data

B. **Easy for Beginners**

Features	Scikit-learn	TensorFlow
Learning Curve	Easy, beginner-friendly	Moderate to steep
API simplicity	High – clean, consistent, simple API	More complex, especially with low-level APIs
Documentation	Well-structured, concise	Comprehensive but can be overwhelming for beginners

C. Community Support

Features	Scikit-learn	TensorFlow
Maturity	Long-established in academia and industry	Widely adopted for deep learning since 2015
Community Size	Large and active	Very large and global community
Ecosystem	Integrates well with NumPy, Pandas, Matplotlib	Extensive ecosystem including Keras, TFX, TensorBoard
Resources	Many beginner tutorials and research support	Rich tutorials, courses, and deployment tools

Ethics and Optimization

A. Potential biases in your MNIST or Amazon Reviews model

MNIST

May not generalize to non-handwritten digits, or digits from different cultures/writing styles. If the data lacks representation from certain groups, the model may be biased.

Amazon Reviews

Sentiment models may inherit biases from the data, e.g., underrepresenting certain products, brands, or user demographics. NER models may miss new brands or misclassify entities if training data is imbalanced.

B. Mitigating bias with tools

TensorFlow Fairness Indicators

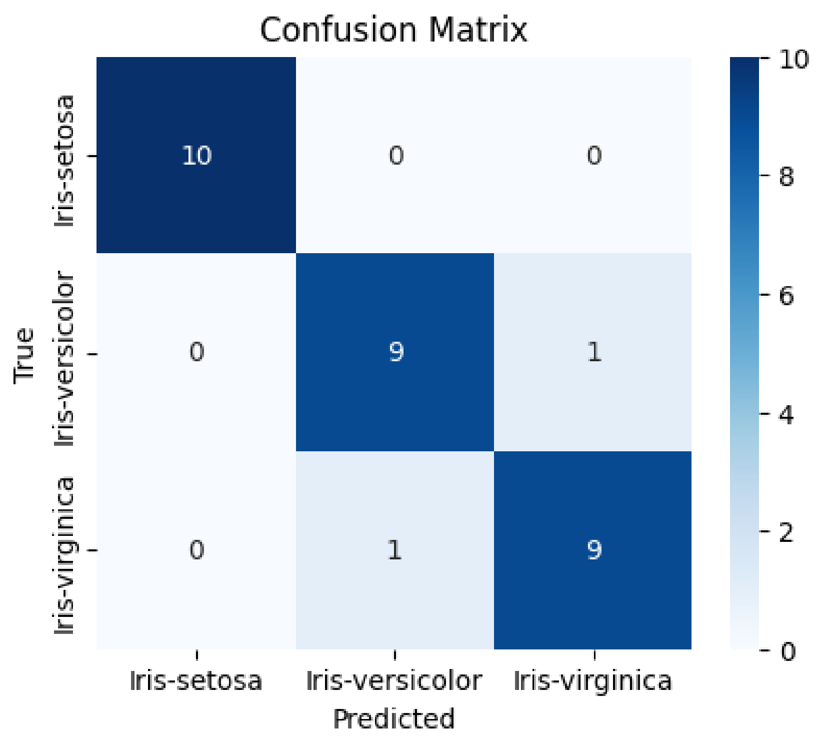
Can measure and visualize metrics like accuracy and false positive rates across different slices of data (e.g., by product category or reviewer demographics). This helps identify and address unfair model behaviour.

spaCy’s rule-based systems

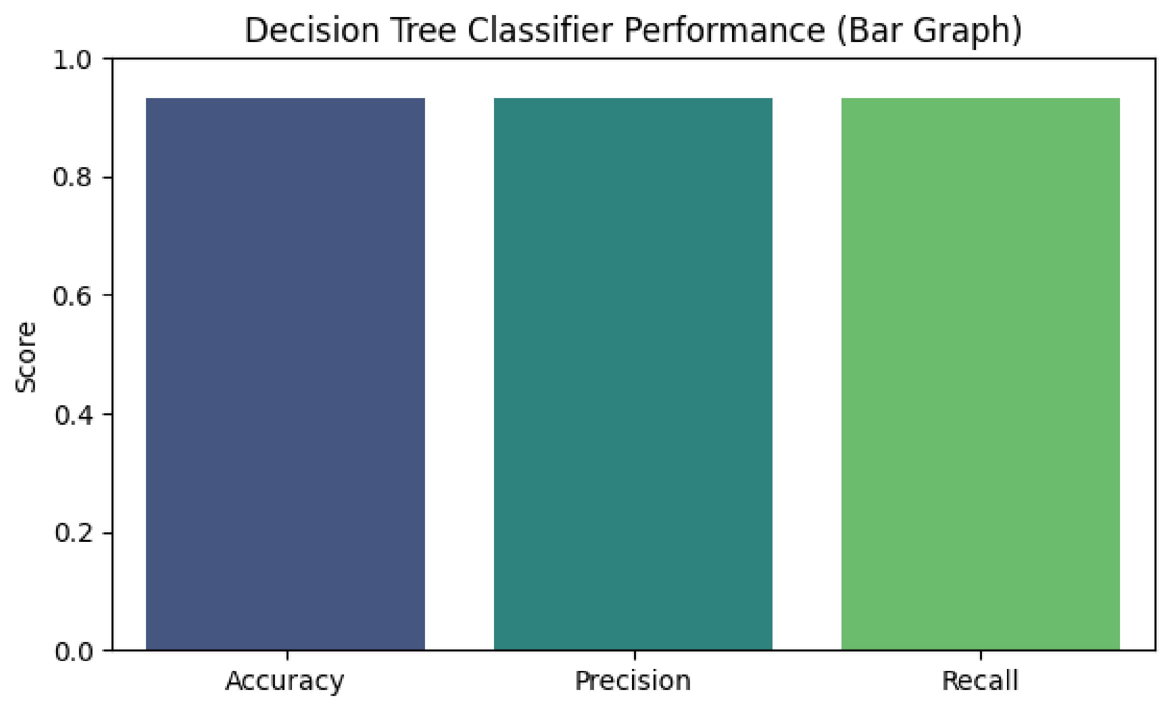
Allow you to add custom patterns for underrepresented brands/products, helping correct for bias in entity recognition that arises from insufficient training data.

Model Output

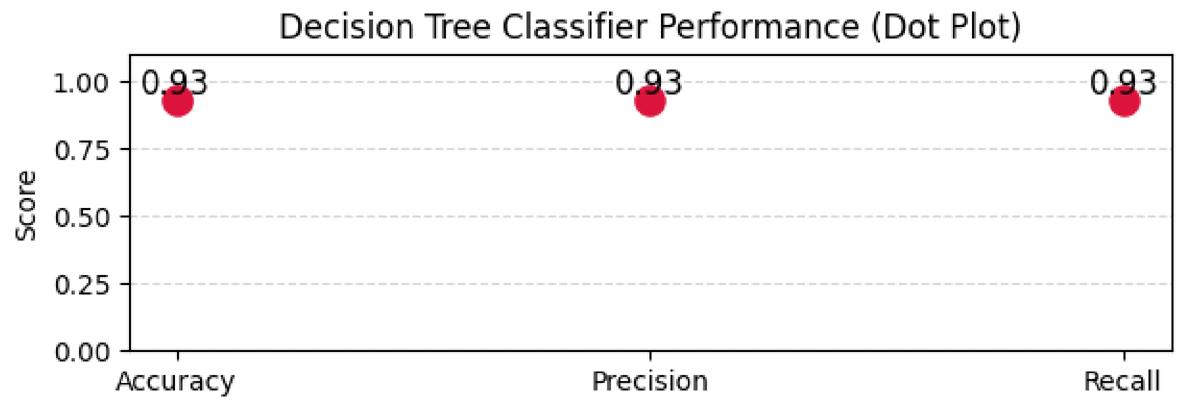
Confusion Matric visualization



Bar Graph visualization



Dot Plot Visualization


Decision Tree Classifier Performance (Dot Plot)
Score
Accuracy
Precision
Recall