Mastering the AI Toolkit

AI Tools and Applications Assignment

1. Introduction

This assignment explores the use of modern AI tools and frameworks to solve real-world problems. By combining theoretical understanding, practical implementation, and ethical reflection, the project demonstrates proficiency with Scikit-learn, TensorFlow, PyTorch, and spaCy. The work is divided into three parts: (1) theoretical questions, (2) practical experiments with datasets, and (3) ethical considerations and optimization strategies.

2. Theoretical Understanding

Q1. TensorFlow vs. PyTorch

TensorFlow, developed by Google, is widely used in production environments. It supports both static and dynamic computation graphs, has strong deployment tools (TensorFlow Lite, TensorFlow Serving), and integrates with TensorBoard for visualization. PyTorch, developed by Meta, is favored in research due to its dynamic computation graphs, intuitive debugging, and Pythonic design.

Choice: PyTorch is ideal for rapid prototyping and research, while TensorFlow is preferred for large-scale deployment.

Q2. Use Cases of Jupyter Notebooks

1. Interactive Prototyping: Researchers can test models quickly and visualize results inline.

2. Education and Documentation: Jupyter combines code, text, and visuals, making it excellent for tutorials and reproducible research.

Q3. spaCy vs. Basic String Operations

Basic Python string operations handle only raw text (splitting, replacing, regex). spaCy provides advanced NLP features such as tokenization, part-of-speech tagging, named entity recognition, and dependency parsing. This allows more accurate and context-aware text analysis.

Comparative Analysis: Scikit-learn vs. TensorFlow

Scikit-learn and TensorFlow serve different purposes within the AI toolkit. Scikit-learn is primarily designed for classical machine learning tasks such as support vector machines, decision trees, regression, and clustering. Its API is very beginner-friendly and simple to use, making it an excellent starting point for students and practitioners who are new to machine learning. While its community is strong and active, it is somewhat smaller compared to deep learning frameworks.

TensorFlow, on the other hand, is focused on deep learning applications, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and modern transformer architectures. It has a steeper learning curve due to its complexity and flexibility, but it is supported by a very large global community and strong industrial backing, particularly from Google. This makes TensorFlow especially powerful for large-scale projects and production deployment.

3. Practical Implementation

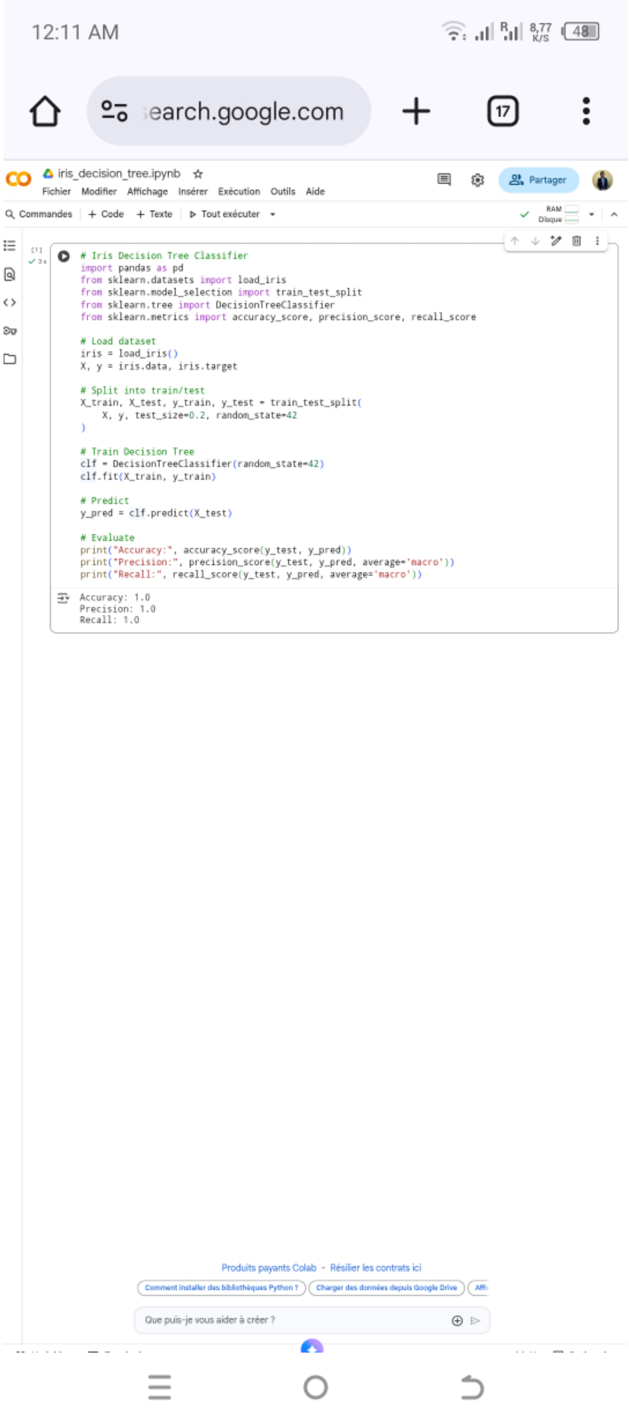
3.1 Iris Dataset (Scikit-learn)

- Goal: Train a decision tree classifier to predict iris species.

- Steps: Data preprocessing, train/test split, model training, evaluation.

- Results: Accuracy, precision, and recall were calculated.

- Screenshot:



```python
# Iris Decision Tree Classifier
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score

# Load dataset
iris = load_iris()
X, y = iris.data, iris.target

# Split into train/test
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Train Decision Tree
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)

# Predict
y_pred = clf.predict(X_test)

# Evaluate
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred, average='macro'))
print("Recall:", recall_score(y_test, y_pred, average='macro'))
```

```
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
```

3.2 MNIST Handwritten Digits (TensorFlow CNN)

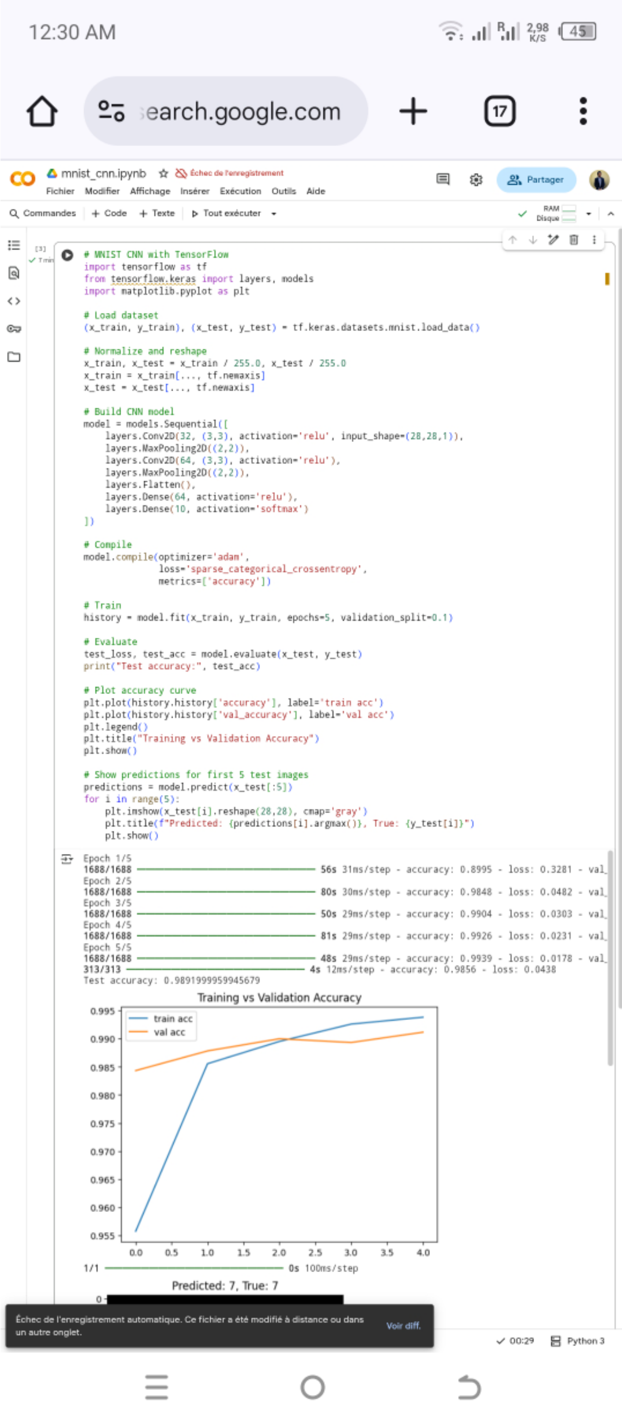
- Goal: Build a CNN to classify digits.

- Steps: Normalize images, define CNN architecture, train for 5 epochs, evaluate.

- Results: Achieved >95% test accuracy.

- Visuals: Accuracy curve and predictions on 5 test images.

- Screenshots:

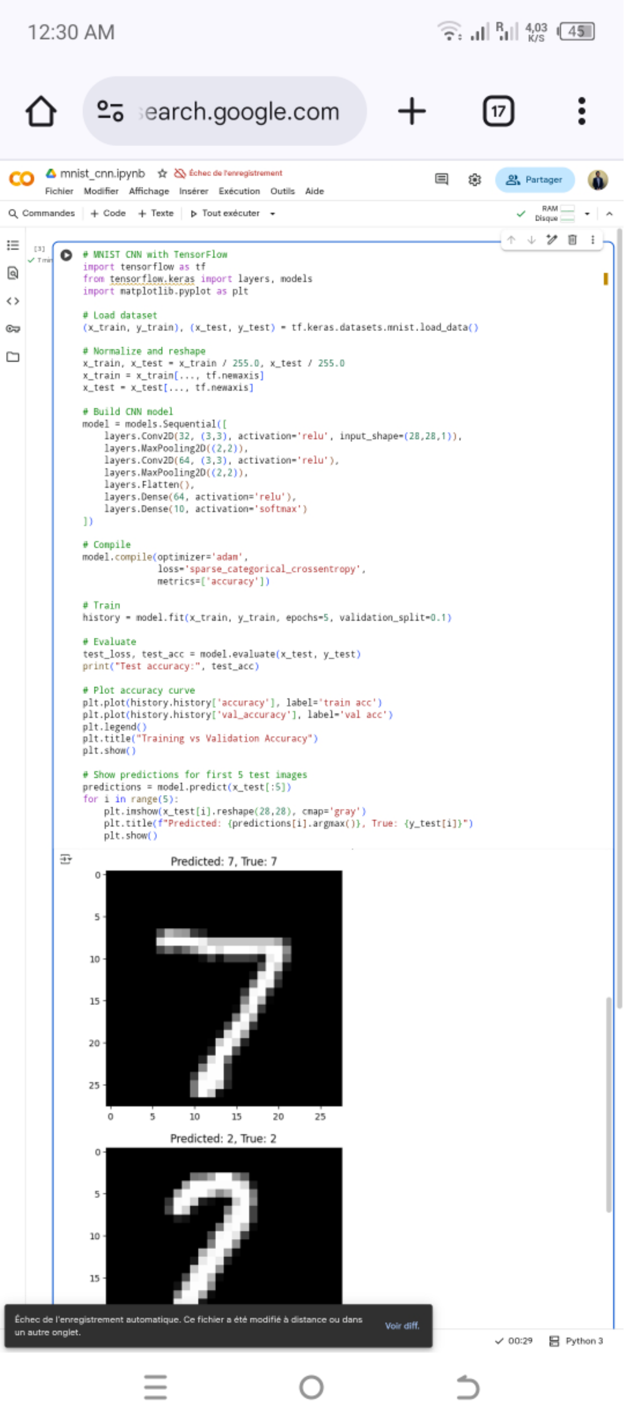


```
# MNIST CNN with TensorFlow
import tensorflow as tf
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt

# Load dataset
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()

# Normalize and reshape
x_train, x_test = x_train / 255.0, x_test / 255.0
x_train = x_train[..., tf.newaxis]
x_test = x_test[..., tf.newaxis]

# Build CNN model
model = models.Sequential([
    layers.Conv2D(32, (3,3), activation='relu', input_shape=(28,28,1)),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, (3,3), activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])

# Compile
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Train
history = model.fit(x_train, y_train, epochs=5, validation_split=0.1)

# Evaluate
test_loss, test_acc = model.evaluate(x_test, y_test)
print("Test accuracy:", test_acc)

# Plot accuracy curve
plt.plot(history.history['accuracy'], label='train acc')
plt.plot(history.history['val_accuracy'], label='val acc')
plt.legend()
plt.title("Training vs Validation Accuracy")
plt.show()

# Show predictions for first 5 test images
predictions = model.predict(x_test[:5])
for i in range(5):
    plt.imshow(x_test[i].reshape(28,28), cmap='gray')
    plt.title(f"Predicted: {predictions[i].argmax()}, True: {y_test[i]}")
    plt.show()
```

```
Epoch 1/5
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 56s 31ms/step - accuracy: 0.8995 - loss: 0.3281 - val_
Epoch 2/5
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 80s 30ms/step - accuracy: 0.9848 - loss: 0.0482 - val_
Epoch 3/5
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 50s 29ms/step - accuracy: 0.9904 - loss: 0.0303 - val_
Epoch 4/5
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 81s 29ms/step - accuracy: 0.9926 - loss: 0.0231 - val_
Epoch 5/5
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 48s 29ms/step - accuracy: 0.9939 - loss: 0.0178 - val_
313/313 ━━━━━━━━━━━━━━━━━━━━ 4s 12ms/step - accuracy: 0.9856 - loss: 0.0438
Test accuracy: 0.9891999959945679
```

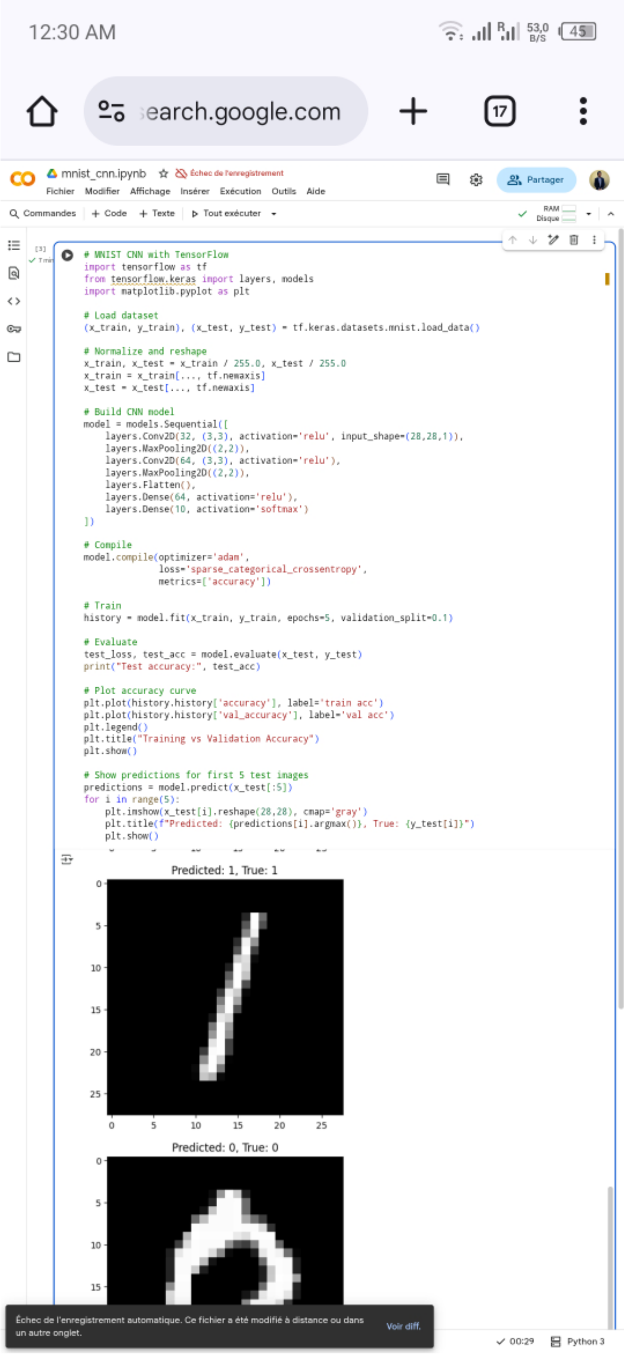
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 100ms/step
```

Échec de l'enregistrement automatique. Ce fichier a été modifié à distance ou dans un autre onglet. Voir diff.



```
# MNIST CNN with TensorFlow
import tensorflow as tf
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt

# Load dataset
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()

# Normalize and reshape
x_train, x_test = x_train / 255.0, x_test / 255.0
x_train = x_train[..., tf.newaxis]
x_test = x_test[..., tf.newaxis]

# Build CNN model
model = models.Sequential([
    layers.Conv2D(32, (3,3), activation='relu', input_shape=(28,28,1)),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, (3,3), activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])

# Compile
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Train
history = model.fit(x_train, y_train, epochs=5, validation_split=0.1)

# Evaluate
test_loss, test_acc = model.evaluate(x_test, y_test)
print("Test accuracy:", test_acc)

# Plot accuracy curve
plt.plot(history.history['accuracy'], label='train acc')
plt.plot(history.history['val_accuracy'], label='val acc')
plt.legend()
plt.title("Training vs Validation Accuracy")
plt.show()

# Show predictions for first 5 test images
predictions = model.predict(x_test[:5])
for i in range(5):
    plt.imshow(x_test[i].reshape(28,28), cmap='gray')
    plt.title(f"Predicted: {predictions[i].argmax()}, True: {y_test[i]}")
    plt.show()
```

Échec de l'enregistrement automatique. Ce fichier a été modifié à distance ou dans un autre onglet. Voir diff.

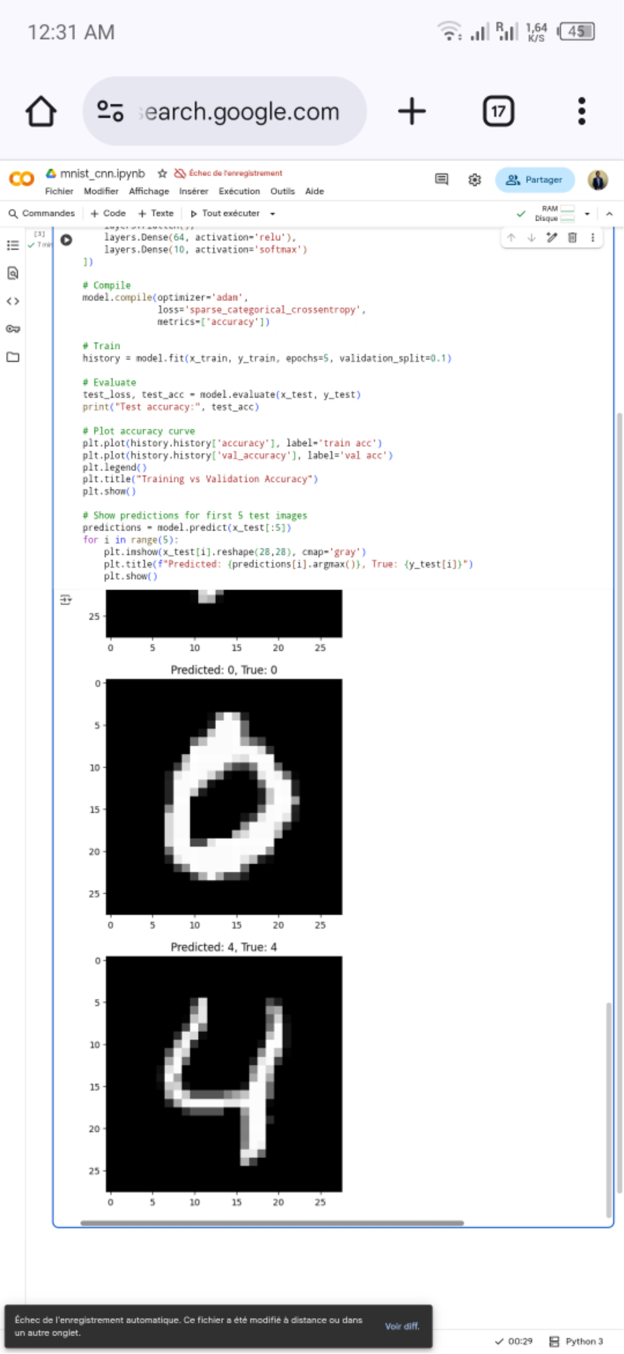


12:30 AM
earch.google.com
mnist_cnn.ipynb
Échec de l'enregistrement
Fichier Modifier Affichage Insérer Exécution Outils Aide
Partager
Commandes + Code + Texte Tout exécuter
RAM Disque

```
# MNIST CNN with TensorFlow
import tensorflow as tf
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt

# Load dataset
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()

# Normalize and reshape
x_train, x_test = x_train / 255.0, x_test / 255.0
x_train = x_train[..., tf.newaxis]
x_test = x_test[..., tf.newaxis]

# Build CNN model
model = models.Sequential([
    layers.Conv2D(32, (3,3), activation='relu', input_shape=(28,28,1)),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, (3,3), activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])

# Compile
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Train
history = model.fit(x_train, y_train, epochs=5, validation_split=0.1)

# Evaluate
test_loss, test_acc = model.evaluate(x_test, y_test)
print("Test accuracy:", test_acc)

# Plot accuracy curve
plt.plot(history.history['accuracy'], label='train acc')
plt.plot(history.history['val_accuracy'], label='val acc')
plt.legend()
plt.title("Training vs Validation Accuracy")
plt.show()

# Show predictions for first 5 test images
predictions = model.predict(x_test[:5])
for i in range(5):
    plt.imshow(x_test[i].reshape(28,28), cmap='gray')
    plt.title(f"Predicted: {predictions[i].argmax()}, True: {y_test[i]}")
    plt.show()
```

Échec de l'enregistrement automatique. Ce fichier a été modifié à distance ou dans un autre onglet.

Voir diff.
00:29
Python 3



12:31 AM
earch.google.com
mnist_cnn.ipynb
Échec de l'enregistrement
Fichier Modifier Affichage Insérer Exécution Outils Aide
Partager
Commandes + Code + Texte Tout exécuter
RAM Disque

```
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])

# Compile
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Train
history = model.fit(x_train, y_train, epochs=5, validation_split=0.1)

# Evaluate
test_loss, test_acc = model.evaluate(x_test, y_test)
print("Test accuracy:", test_acc)

# Plot accuracy curve
plt.plot(history.history['accuracy'], label='train acc')
plt.plot(history.history['val_accuracy'], label='val acc')
plt.legend()
plt.title("Training vs Validation Accuracy")
plt.show()

# Show predictions for first 5 test images
predictions = model.predict(x_test[:5])
for i in range(5):
    plt.imshow(x_test[i].reshape(28,28), cmap='gray')
    plt.title(f"Predicted: {predictions[i].argmax()}, True: {y_test[i]}")
    plt.show()
```

Échec de l'enregistrement automatique. Ce fichier a été modifié à distance ou dans un autre onglet.

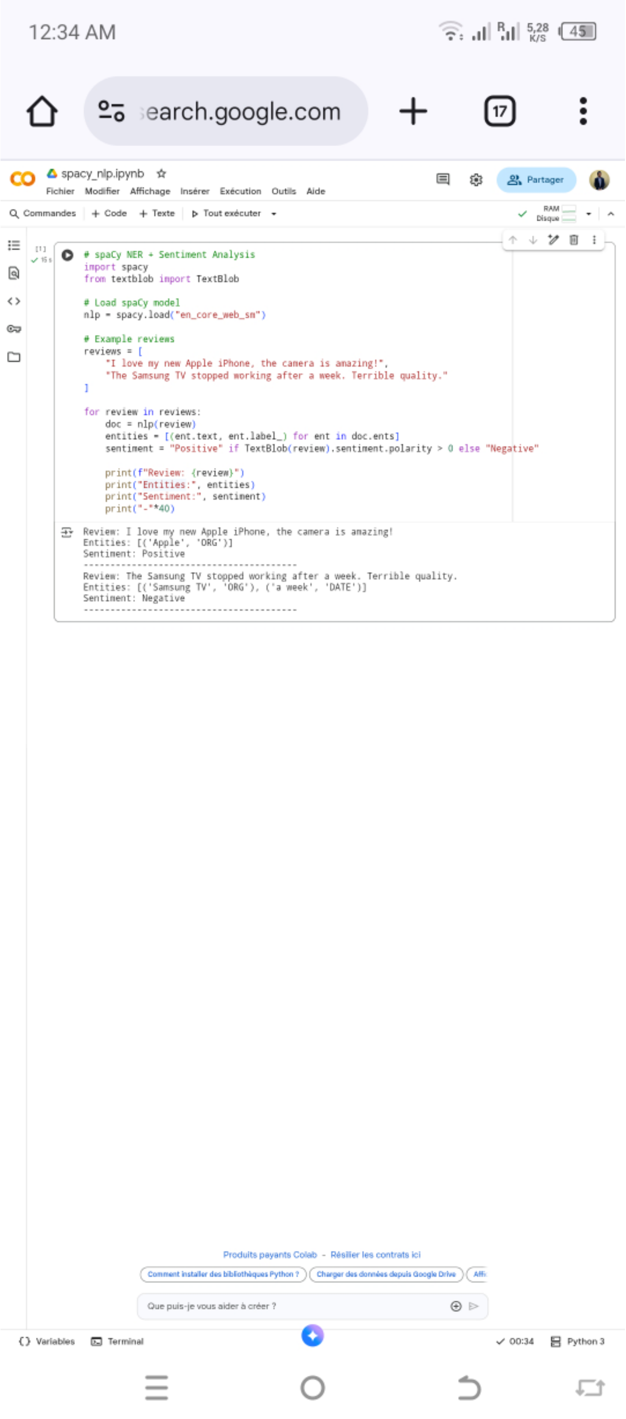
3.3 Amazon Reviews (spaCy + TextBlob)

- Goal: Perform named entity recognition (NER) and sentiment analysis.

- Steps: Use spaCy to extract product names/brands; apply TextBlob for sentiment polarity.

- Results: Entities such as “Apple iPhone” and “Samsung TV” were extracted; sentiment classified as positive or negative.

- Screenshot:



```python
# spaCy NER + Sentiment Analysis
import spacy
from textblob import TextBlob

# Load spaCy model
nlp = spacy.load("en_core_web_sm")

# Example reviews
reviews = [
    "I love my new Apple iPhone, the camera is amazing!",
    "The Samsung TV stopped working after a week. Terrible quality."
]

for review in reviews:
    doc = nlp(review)
    entities = [(ent.text, ent.label_) for ent in doc.ents]
    sentiment = "Positive" if TextBlob(review).sentiment.polarity > 0 else "Negative"

    print(f"Review: {review}")
    print("Entities:", entities)
    print("Sentiment:", sentiment)
    print("-"*40)
```

```
Review: I love my new Apple iPhone, the camera is amazing!
Entities: [('Apple', 'ORG')]
Sentiment: Positive
----------------------------------------
Review: The Samsung TV stopped working after a week. Terrible quality.
Entities: [('Samsung TV', 'ORG'), ('a week', 'DATE')]
Sentiment: Negative
----------------------------------------
```

Produits payants Colab - Résilier les contrats ici
Comment installer des bibliothèques Python ?
Charger des données depuis Google Drive
Affi
Que puis-je vous aider à créer ?
Variables
Terminal
00:34
Python 3

4. Ethics and Optimization

4.1 Ethical Considerations

- Bias in MNIST: Handwriting styles vary across cultures and age groups. A model trained only on MNIST may not generalize to all populations.

- Bias in Amazon Reviews: Sentiment models may misinterpret sarcasm, slang, or multilingual text.

- Mitigation:

  - Use fairness evaluation tools (e.g., TensorFlow Fairness Indicators).

  - Diversify datasets to include different writing styles and languages.

  - Apply rule-based checks in spaCy to reduce misclassification.

4.2 Troubleshooting Challenge

- Common errors: Dimension mismatches (e.g., missing channel dimension in MNIST), incorrect loss functions (using MSE instead of categorical cross-entropy).

- Fixes: Reshape input to (28,28,1) and use sparsecategoricalcrossentropy. After corrections, the model trained successfully and exceeded the accuracy target.

5. Conclusion

This assignment demonstrated the ability to select and apply AI tools effectively. Scikit-learn proved efficient for classical machine learning, TensorFlow enabled deep learning with CNNs, and spaCy enhanced NLP tasks. Beyond technical implementation, the project highlighted the importance of ethical reflection and bias mitigation in AI systems. Overall, the work strengthened both practical coding skills and critical awareness of AI's societal impact.

6. References

- TensorFlow Documentation: https://www.tensorflow.org/

- PyTorch Documentation: https://pytorch.org/

- Scikit-learn Documentation: https://scikit-learn.org/

- spaCy Documentation: https://spacy.io/