PLP WEEK 3 AI TOOLS ASSIGNMENT

GROUP 59

1. Short Answer Questions

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

- **TensorFlow** is more production-oriented with tools like TensorFlow Serving and TensorFlow Lite for deployment.
- **PyTorch** is more Pythonic and dynamic, which makes it preferred for research and prototyping.

Choose TensorFlow if you're building a production-ready system and choose PyTorch if you want easier debugging and flexible model experimentation.

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

-**Exploratory Data Analysis (EDA):** Jupyter Notebooks allow interactive data visualization and code execution, helping data scientists explore datasets efficiently.

-**Model Prototyping and Experimentation:** Developers can build, test, and tweak machine learning models step-by-step, documenting each stage with markdown and visualizations.

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

2. Comparative Analysis

- Compare Scikit-learn and TensorFlow in terms of:
    - Target applications (e.g., classical ML vs. deep learning).
    - Ease of use for beginners.
    - Community support.

Target Application - Scikit-learn targetsClassical machine learning (e.g., regression, classification, clustering) while **TensorFlow** targets Deep learning (e.g., neural networks, CNNs, RNNs, transformers)

Learning Curve - Scikit-learn has Gentle; easy for beginners with consistent, simple API while **TensorFlow** has Steeper; requires understanding of computational graphs and model structures.

Community Support - Scikit-learn has Strong for classical ML users; mature and stable while **TensorFlow has** Very large and active, especially for cutting-edge AI research

Part 3: Ethics & Optimization

1. Ethical Considerations

Bias in MNIST Digit Classifier

MNIST seems objective, but still has potential biases:

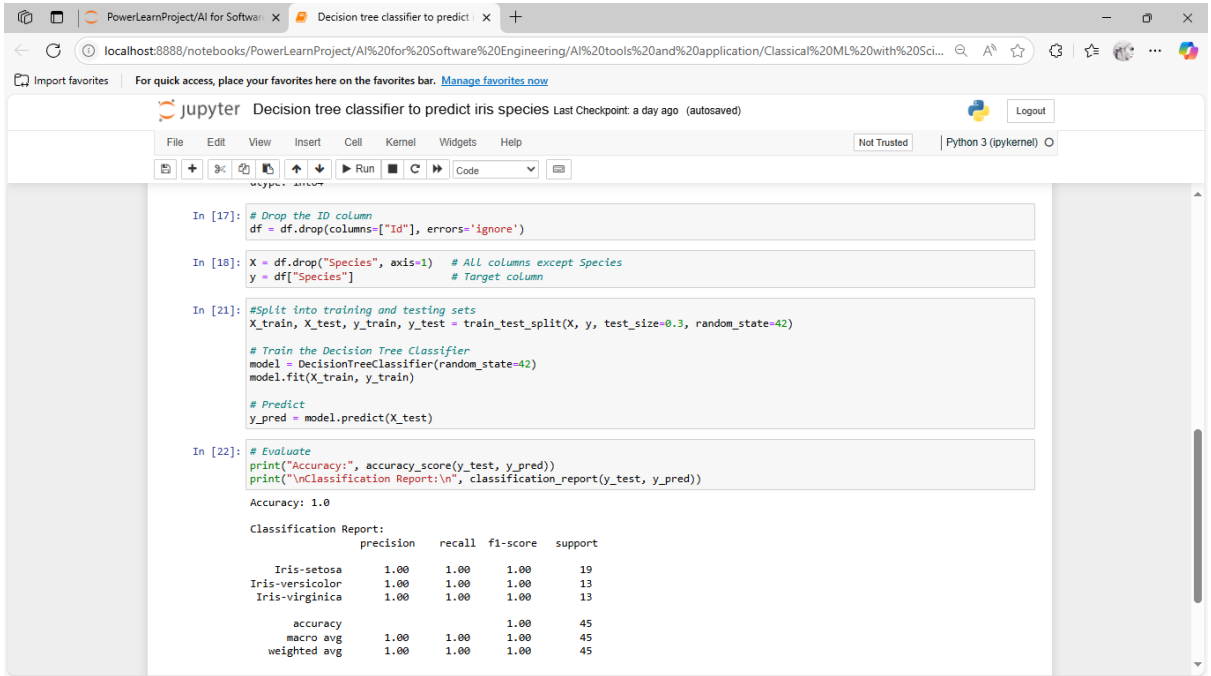
- **Input quality bias**: MNIST digits are centered and grayscale. Real-world input may vary (off-center, noisy, colored).
- **Model overfitting**: The model may overfit to "clean" data and misclassify stylized or handwritten digits from diverse populations (e.g., left-handed writing styles).
- **Accessibility**: Model may fail for users with motor impairments or alternative drawing styles.

Bias in Amazon Reviews Classifier

When using text data like Amazon reviews:

- **Sentiment bias**: Models might associate certain product categories, demographics, or dialects with negative or positive sentiment.
- **Representation bias**: If most training reviews come from one region or language style, the model may underperform elsewhere.
- **Toxicity or gender bias**: Words associated with certain groups may receive skewed sentiment scores.

Accuracy scores for Decision tree classifier for iris species



jupyter Decision tree classifier to predict iris species Last Checkpoint: a day ago (autosaved)

In [17]:
```
# Drop the ID column
df = df.drop(columns=["Id"], errors='ignore')
```

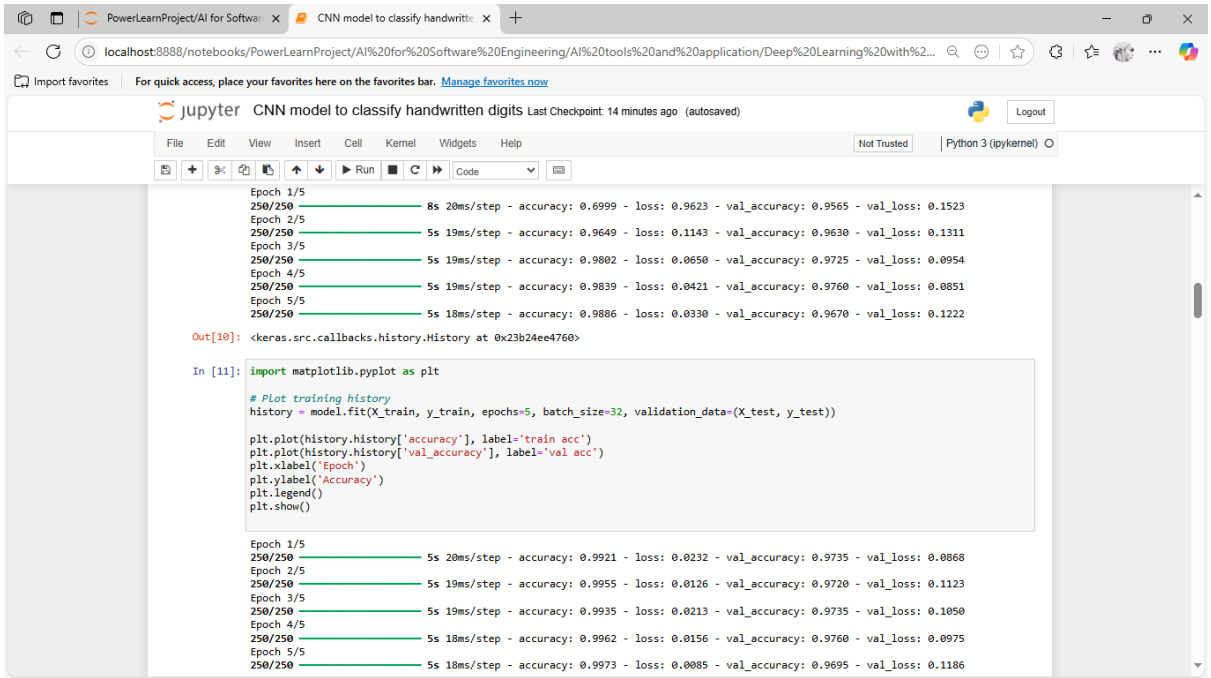
In [18]:
```
X = df.drop("Species", axis=1)   # All columns except Species
y = df["Species"]                # Target column
```

In [21]:
```
#Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Train the Decision Tree Classifier
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)
```

In [22]:
```
# Evaluate
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 1.0

Classification Report:
                   precision    recall  f1-score   support

       Iris-setosa       1.00      1.00      1.00        19
   Iris-versicolor       1.00      1.00      1.00        13
    Iris-virginica       1.00      1.00      1.00        13

          accuracy                           1.00        45
         macro avg       1.00      1.00      1.00        45
      weighted avg       1.00      1.00      1.00        45
```

Epoch values for CNN model to classify handwrittings



jupyter CNN model to classify handwritten digits Last Checkpoint: 14 minutes ago (autosaved)

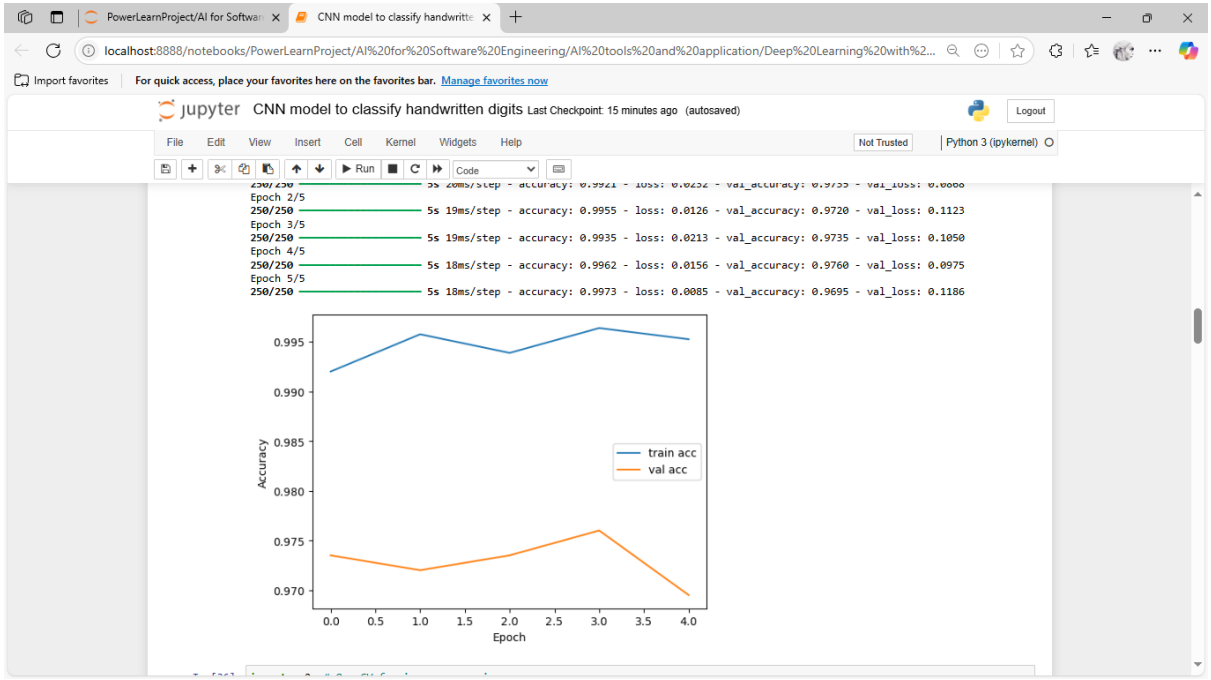
```
Epoch 1/5
250/250 ━━━━━━━━━━━━━━━━━━━━ 8s 20ms/step - accuracy: 0.6999 - loss: 0.9623 - val_accuracy: 0.9565 - val_loss: 0.1523
Epoch 2/5
250/250 ━━━━━━━━━━━━━━━━━━━━ 5s 19ms/step - accuracy: 0.9649 - loss: 0.1143 - val_accuracy: 0.9630 - val_loss: 0.1311
Epoch 3/5
250/250 ━━━━━━━━━━━━━━━━━━━━ 5s 19ms/step - accuracy: 0.9802 - loss: 0.0650 - val_accuracy: 0.9725 - val_loss: 0.0954
Epoch 4/5
250/250 ━━━━━━━━━━━━━━━━━━━━ 5s 19ms/step - accuracy: 0.9839 - loss: 0.0421 - val_accuracy: 0.9760 - val_loss: 0.0851
Epoch 5/5
250/250 ━━━━━━━━━━━━━━━━━━━━ 5s 18ms/step - accuracy: 0.9886 - loss: 0.0330 - val_accuracy: 0.9670 - val_loss: 0.1222
```

Out[10]:
```
<keras.src.callbacks.history.History at 0x23b24ee4760>
```

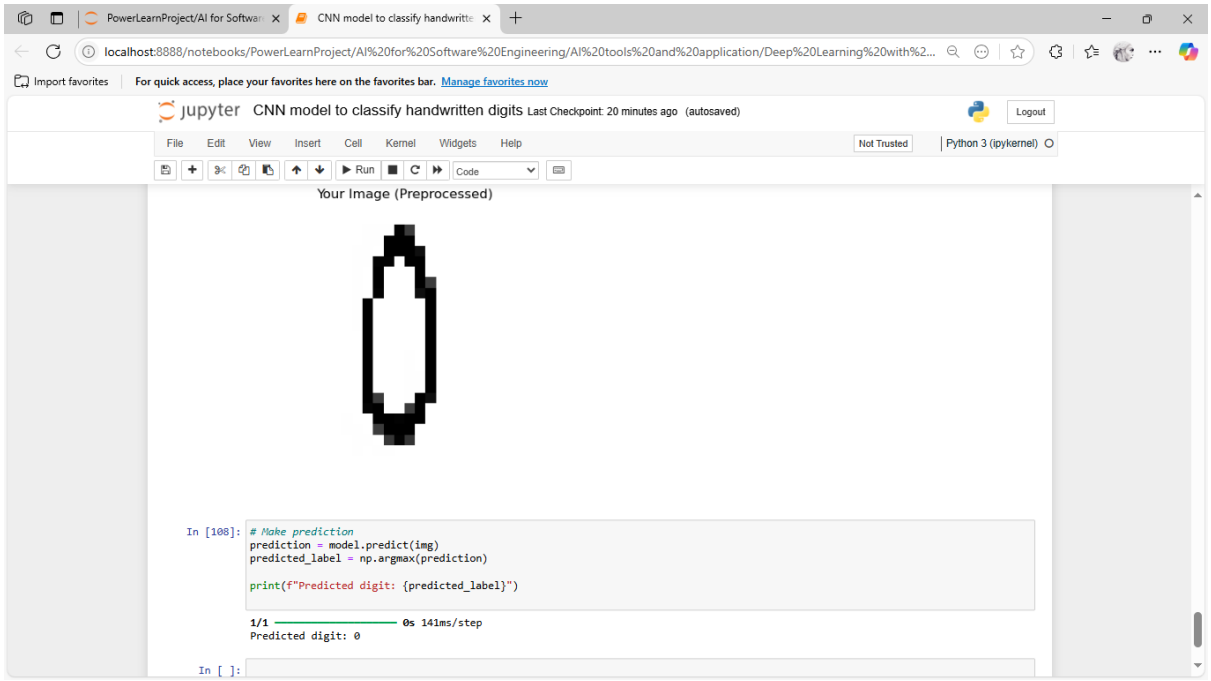
In [11]:
```
import matplotlib.pyplot as plt

# Plot training history
history = model.fit(X_train, y_train, epochs=5, batch_size=32, validation_data=(X_test, y_test))

plt.plot(history.history['accuracy'], label='train acc')
plt.plot(history.history['val_accuracy'], label='val acc')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

```
Epoch 1/5
250/250 ━━━━━━━━━━━━━━━━━━━━ 5s 20ms/step - accuracy: 0.9921 - loss: 0.0232 - val_accuracy: 0.9735 - val_loss: 0.0868
Epoch 2/5
250/250 ━━━━━━━━━━━━━━━━━━━━ 5s 19ms/step - accuracy: 0.9955 - loss: 0.0126 - val_accuracy: 0.9720 - val_loss: 0.1123
Epoch 3/5
250/250 ━━━━━━━━━━━━━━━━━━━━ 5s 19ms/step - accuracy: 0.9935 - loss: 0.0213 - val_accuracy: 0.9735 - val_loss: 0.1050
Epoch 4/5
250/250 ━━━━━━━━━━━━━━━━━━━━ 5s 18ms/step - accuracy: 0.9962 - loss: 0.0156 - val_accuracy: 0.9760 - val_loss: 0.0975
Epoch 5/5
250/250 ━━━━━━━━━━━━━━━━━━━━ 5s 18ms/step - accuracy: 0.9973 - loss: 0.0085 - val_accuracy: 0.9695 - val_loss: 0.1186
```

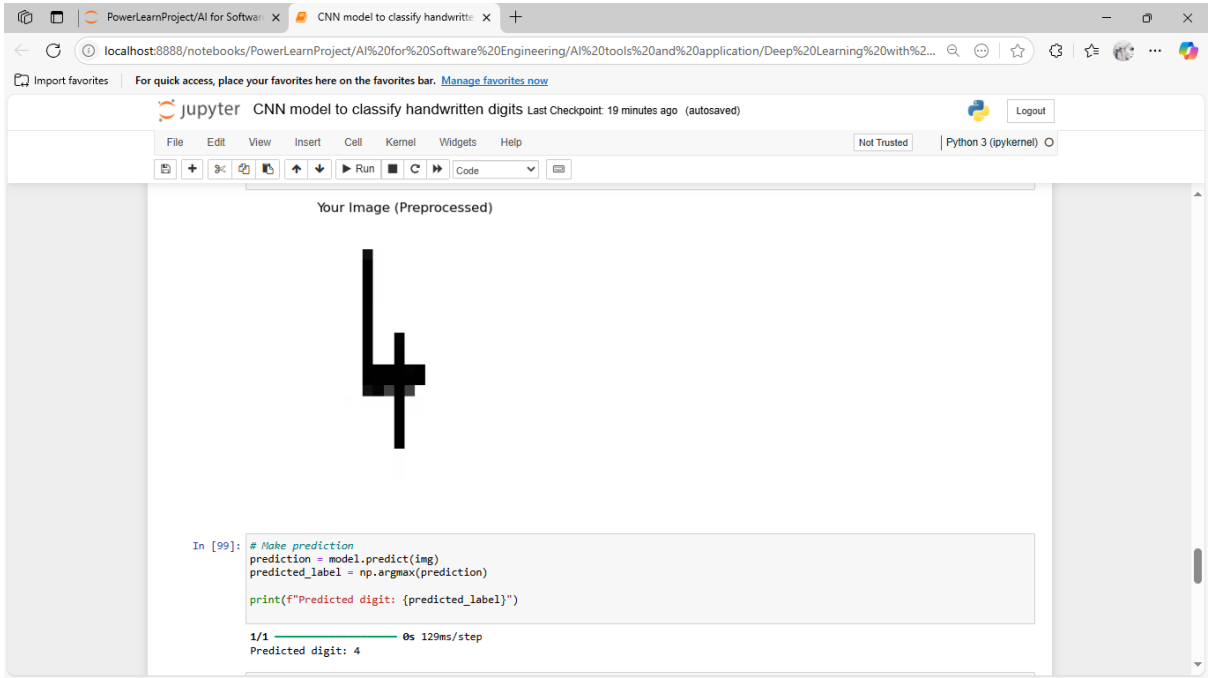
Graph for Epoch against Accuracy CNN model



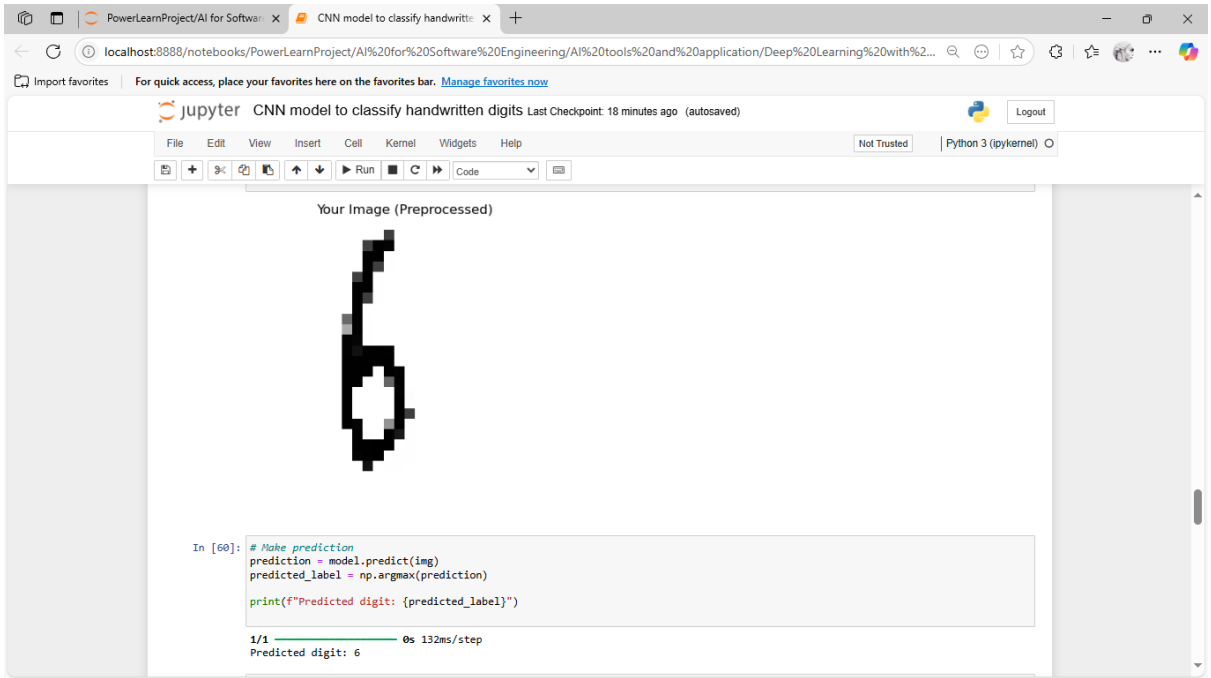
Handwritten digit and prediction(0)



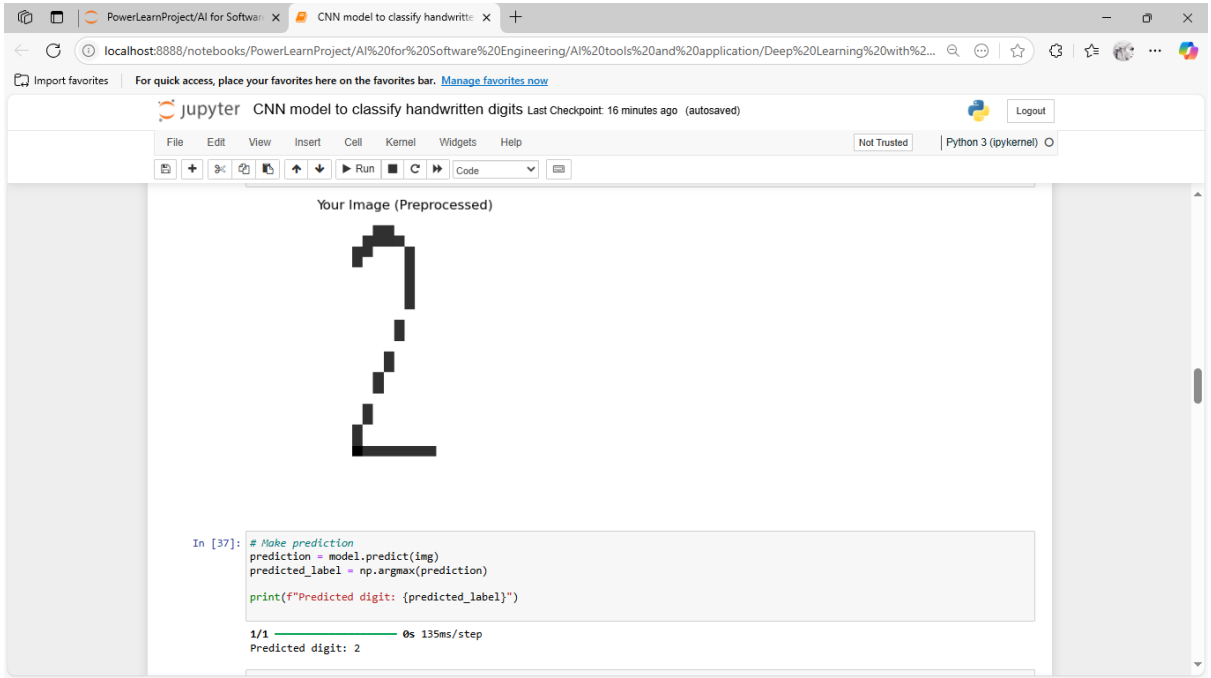
Handwritten digit and prediction(4)



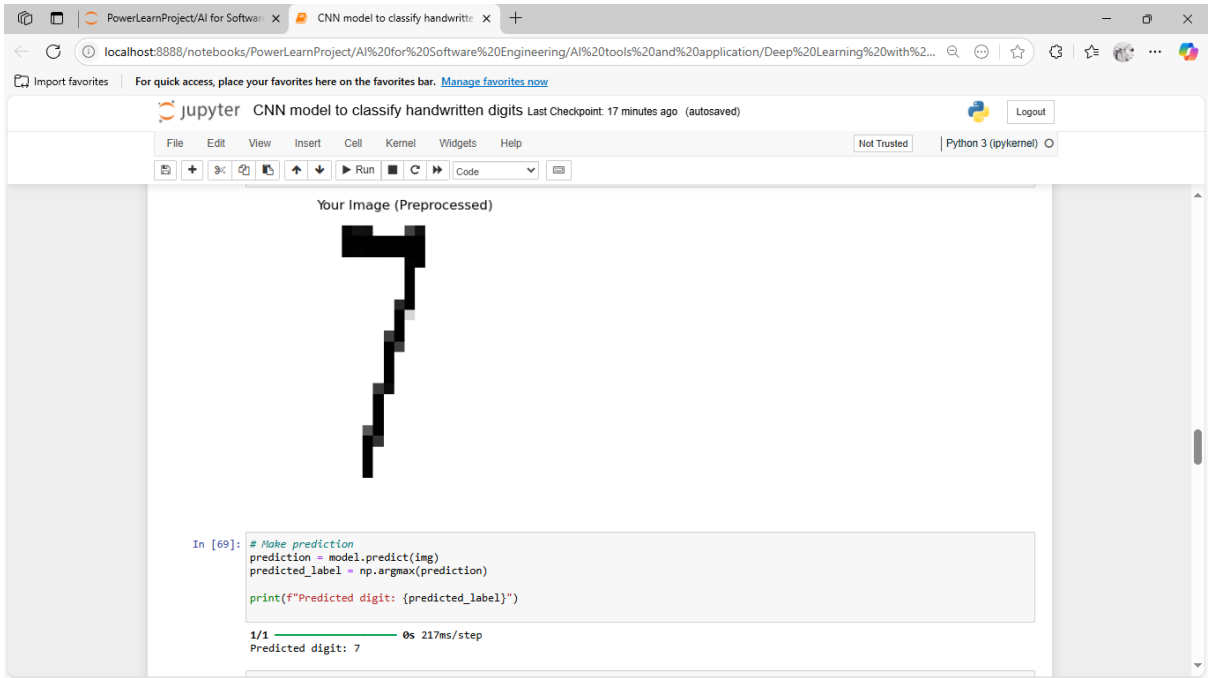
Handwritten digit and prediction(6)



Handwritten digit and the prediction(2)



Handwritten digit and the prediction(7)



Classical ML with spaCy

colab.research.google.com/drive/1_eqiOAXCio5Q6WNxXL00b0VahUxNF26x#scrollTo=_WqpFs6cV14b

ML with spaCY.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

```
    "Adidas sneakers are perfect for running. Super light and comfy."
]

def analyze_review(text):
    doc = nlp(text)
    entities = [(ent.text, ent.label_) for ent in doc.ents if ent.label_ in ['ORG', 'PRODUCT']]
    score = analyzer.polarity_scores(text)['compound']
    sentiment = "Positive" if score > 0.05 else "Negative" if score < -0.05 else "Neutral"
    return {
        "Review": text,
        "Entities": entities,
        "Sentiment": sentiment,
        "Compound Score": score
    }

results = [analyze_review(review) for review in reviews]
df = pd.DataFrame(results)
pd.set_option('display.max_colwidth', None)
display(df[['Review', 'Entities', 'Sentiment', 'Compound Score']])

plt.figure(figsize=(8, 5))
sns.countplot(data=df, x='Sentiment', palette='coolwarm')
plt.title("Sentiment Distribution")
plt.xlabel("Sentiment")
```

Variables Terminal 5:28 PM Python 3

Type here to search 24°C Sunny 5:37 PM 6/24/2025

colab.research.google.com/drive/1_eqiOAXCio5Q6WNxXL00b0VahUxNF26x#scrollTo=_WqpFs6cV14b

ML with spaCY.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

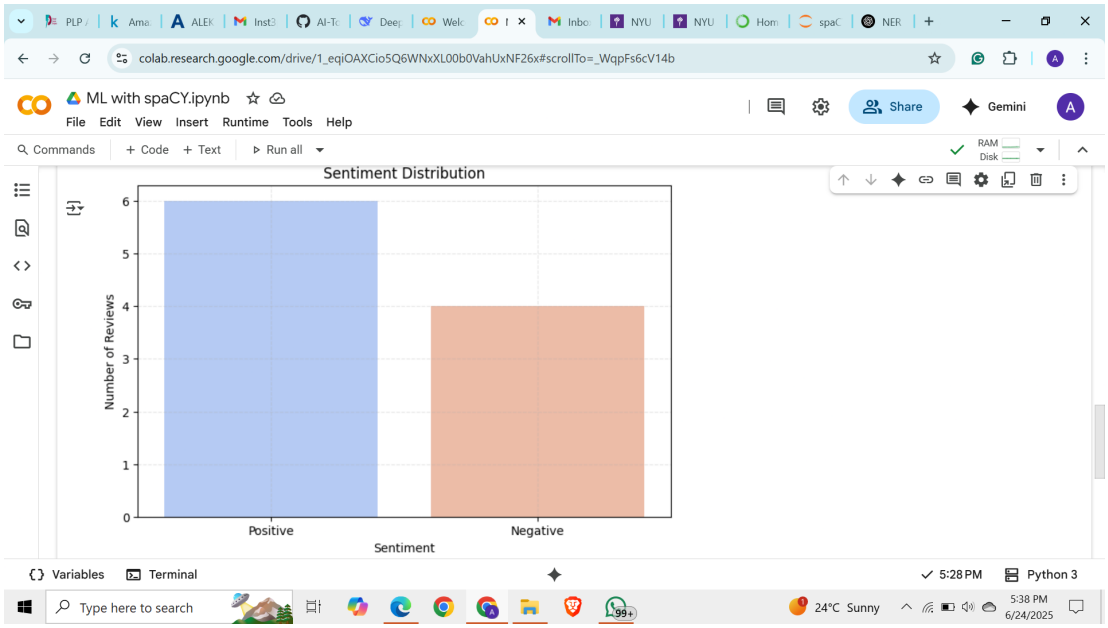
```
plt.show()

df.to_csv("review_analysis_output.csv", index=False)
```

	Review	Entities	Sentiment	Compound Score
0	I absolutely love this Apple iPhone! It's fast and the camera is amazing.	[(Apple, ORG)]	Positive	0.8620
1	This Samsung TV stopped working after two weeks. Very disappointed.	[(Samsung TV, ORG)]	Negative	-0.6478
2	The Nike shoes are comfortable and stylish. Will buy again!	[(Nike, ORG)]	Positive	0.5562
3	Terrible experience with Lenovo laptop. It crashes every hour.	[(Lenovo, ORG)]	Negative	-0.4767
4	Sony headphones are top-notch. Great sound quality!	[(Sony, ORG)]	Positive	0.6588
5	Avoid the Huawei tablet. It's slow and full of bugs.	[(Huawei, ORG)]	Negative	-0.2960
6	Bought a JBL speaker. Great bass and battery life.	[(JBL, ORG)]	Positive	0.6249
7	Dell monitor works fine but the colors are washed out.	[]	Positive	0.1027
8	My ASUS laptop overheats a lot. Not recommended.	[(ASUS, ORG)]	Negative	-0.1511
9	Adidas sneakers are perfect for running. Super light and comfy.	[]	Positive	0.8225

```
/tmp/ipython-input-4-3240904600.py:44: FutureWarning:
```

Variables Terminal 5:28 PM Python 3

Type here to search 24°C Sunny 5:37 PM 6/24/2025


colab.research.google.com/drive/1_eqiOAXCio5Q6WNxXL00b0VahUxNF26x#scrollTo=_WqpFs6cV14b
ML with spaCY.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share
Gemini
Sentiment Distribution
Number of Reviews
0
1
2
3
4
5
6
Positive
Negative
Sentiment
Variables
Terminal
5:28 PM
Python 3
Type here to search
24°C Sunny
5:38 PM
6/24/2025