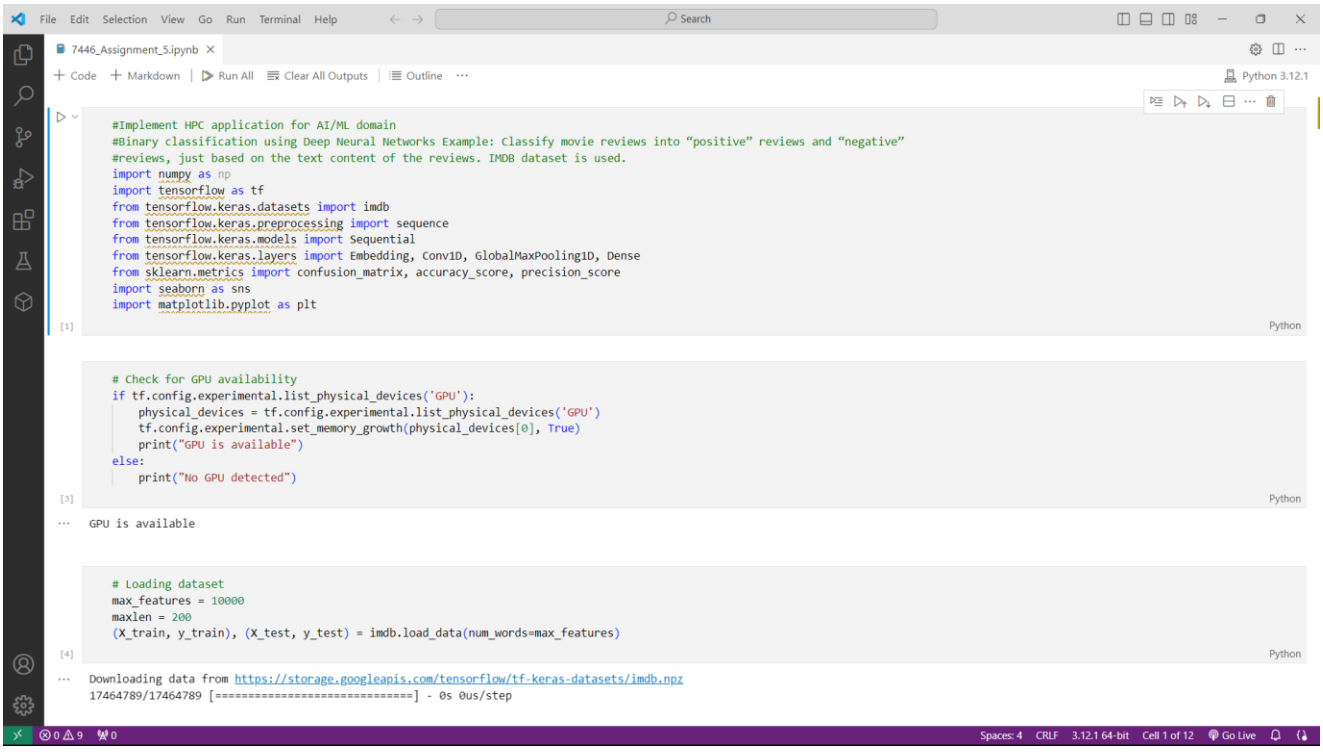
Assignment -5

Implement HPC application for AI/ML domain.

- Binary classification using Deep Neural Networks Example: Classify movie reviews into “positive” reviews and “negative” reviews, just based on the text content of the reviews. IMDB dataset is used.



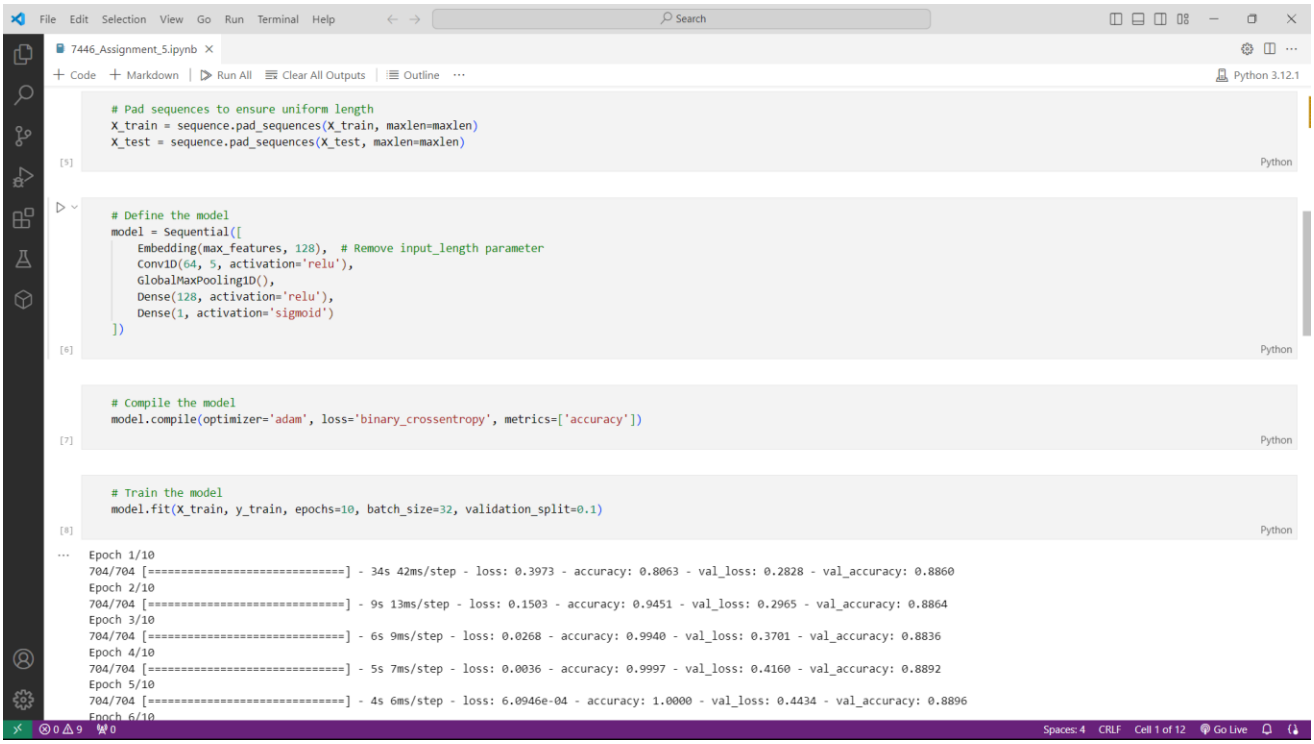
```
#Implement HPC application for AI/ML domain
#Binary classification using Deep Neural Networks Example: Classify movie reviews into “positive” reviews and “negative”
#reviews, just based on the text content of the reviews. IMDB dataset is used.
import numpy as np
import tensorflow as tf
from tensorflow.keras.datasets import imdb
from tensorflow.keras.preprocessing import sequence
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Conv1D, GlobalMaxPooling1D, Dense
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score
import seaborn as sns
import matplotlib.pyplot as plt
```

```
# Check for GPU availability
if tf.config.experimental.list_physical_devices('GPU'):
    physical_devices = tf.config.experimental.list_physical_devices('GPU')
    tf.config.experimental.set_memory_growth(physical_devices[0], True)
    print("GPU is available")
else:
    print("No GPU detected")
```

```
GPU is available
```

```
# Loading dataset
max_features = 10000
maxlen = 200
(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=max_features)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz
17464789/17464789 [==============================] - 0s 0us/step
```



```
# Pad sequences to ensure uniform length
X_train = sequence.pad_sequences(X_train, maxlen=maxlen)
X_test = sequence.pad_sequences(X_test, maxlen=maxlen)
```

```
# Define the model
model = Sequential([
    Embedding(max_features, 128),  # Remove input_length parameter
    Conv1D(64, 5, activation='relu'),
    GlobalMaxPooling1D(),
    Dense(128, activation='relu'),
    Dense(1, activation='sigmoid')
])
```

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

```
# Train the model
model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.1)
```

```
Epoch 1/10
704/704 [==============================] - 34s 42ms/step - loss: 0.3973 - accuracy: 0.8063 - val_loss: 0.2828 - val_accuracy: 0.8860
Epoch 2/10
704/704 [==============================] - 9s 13ms/step - loss: 0.1503 - accuracy: 0.9451 - val_loss: 0.2965 - val_accuracy: 0.8864
Epoch 3/10
704/704 [==============================] - 6s 9ms/step - loss: 0.0268 - accuracy: 0.9940 - val_loss: 0.3701 - val_accuracy: 0.8836
Epoch 4/10
704/704 [==============================] - 5s 7ms/step - loss: 0.0036 - accuracy: 0.9997 - val_loss: 0.4160 - val_accuracy: 0.8892
Epoch 5/10
704/704 [==============================] - 4s 6ms/step - loss: 6.0946e-04 - accuracy: 1.0000 - val_loss: 0.4434 - val_accuracy: 0.8896
Epoch 6/10
```

File Edit Selection View Go Run Terminal Help

Search

7446_Assignment_5.ipynb

Code Markdown Run All Clear All Outputs Outline

Python 3.12.1

```python
# Train the model
model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.1)
```

[8] Python

```
Epoch 1/10
704/704 [==============================] - 34s 42ms/step - loss: 0.3973 - accuracy: 0.8063 - val_loss: 0.2828 - val_accuracy: 0.8860
Epoch 2/10
704/704 [==============================] - 9s 13ms/step - loss: 0.1503 - accuracy: 0.9451 - val_loss: 0.2965 - val_accuracy: 0.8864
Epoch 3/10
704/704 [==============================] - 6s 9ms/step - loss: 0.0268 - accuracy: 0.9940 - val_loss: 0.3701 - val_accuracy: 0.8836
Epoch 4/10
704/704 [==============================] - 5s 7ms/step - loss: 0.0036 - accuracy: 0.9997 - val_loss: 0.4160 - val_accuracy: 0.8892
Epoch 5/10
704/704 [==============================] - 4s 6ms/step - loss: 6.0946e-04 - accuracy: 1.0000 - val_loss: 0.4434 - val_accuracy: 0.8896
Epoch 6/10
704/704 [==============================] - 5s 7ms/step - loss: 1.8936e-04 - accuracy: 1.0000 - val_loss: 0.4677 - val_accuracy: 0.8920
Epoch 7/10
704/704 [==============================] - 5s 7ms/step - loss: 1.0492e-04 - accuracy: 1.0000 - val_loss: 0.4885 - val_accuracy: 0.8928
Epoch 8/10
704/704 [==============================] - 4s 6ms/step - loss: 6.4349e-05 - accuracy: 1.0000 - val_loss: 0.5074 - val_accuracy: 0.8932
Epoch 9/10
704/704 [==============================] - 5s 7ms/step - loss: 4.0997e-05 - accuracy: 1.0000 - val_loss: 0.5256 - val_accuracy: 0.8940
Epoch 10/10
704/704 [==============================] - 4s 5ms/step - loss: 2.6574e-05 - accuracy: 1.0000 - val_loss: 0.5434 - val_accuracy: 0.8944

<keras.src.callbacks.History at 0x7a8acabe7d00>
```

```python
# Evaluate the model
y_pred_probs = model.predict(X_test)
y_pred_classes = (y_pred_probs > 0.5).astype(int)
```

[9] Python

```
782/782 [==============================] - 2s 2ms/step
```

```python
# Calculate confusion matrix
```

Spaces: 4 CRLF Cell 1 of 12 Go Live

File Edit Selection View Go Run Terminal Help

Search

7446_Assignment_5.ipynb

Code Markdown Run All Clear All Outputs Outline

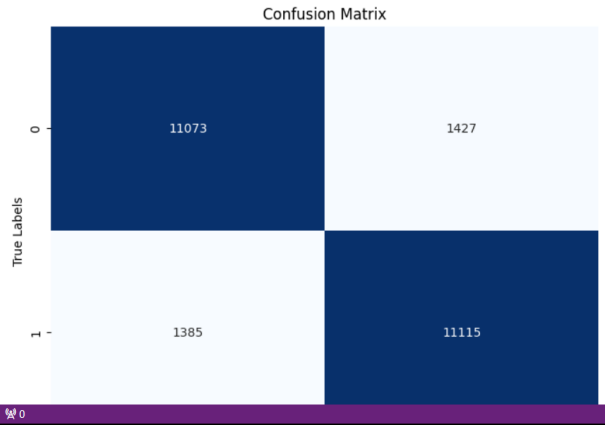
Python 3.12.1

```python
# Calculate confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred_classes)
```

[10] Python

```python
# Plot confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```

[11] Python



Spaces: 4 CRLF Cell 1 of 12 Go Live

File Edit Selection View Go Run Terminal Help

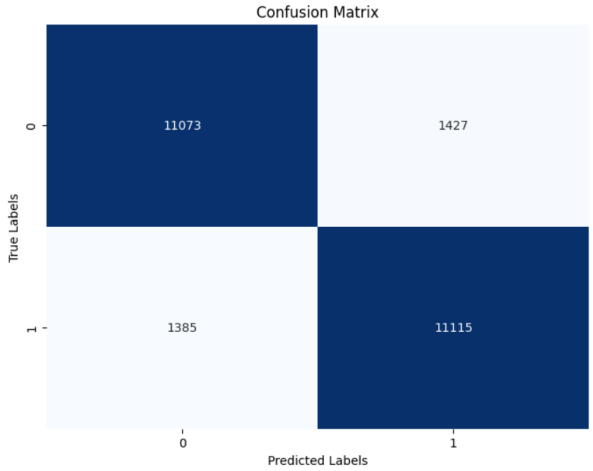
Search

7446_Assignment_5.ipynb

+ Code + Markdown | Run All Clear All Outputs | Outline ··· Python 3.12.1

```
# Plot confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```

[11] Python



⊗ 0 ⚠ 9 0 Spaces: 4 CRLF Cell 1 of 12 Go Live

File Edit Selection View Go Run Terminal Help

Search

7446_Assignment_5.ipynb

+ Code + Markdown | Run All Clear All Outputs | Outline ··· Python 3.12.1

```
# Calculate accuracy and precision
accuracy = accuracy_score(y_test, y_pred_classes)
precision = precision_score(y_test, y_pred_classes)

print("Accuracy : ", accuracy)
print("Precision : ", precision)
```

[12] Python

```
Accuracy :  0.88752
Precision :  0.8862222930952002
```

[ ] Python

⊗ 0 ⚠ 9 0 Spaces: 4 CRLF Cell 1 of 12 Go Live