NN&DeepLearning_ICP10:

Bhavana Prathyusha Madhugani
700737914
https://github.com/BhavanaPrathyushaMadhugani/NNDL_ICP5

LSTM Lesson Overview:
In this lesson, we are going to discuss types of ANNs and Recurrent Neural Network.

Use Case Description:
Sentiment Analysis on the Twitter dataset

Programming elements:

1. Basics of LSTM

   LSTM is a type of recurrent neural network (RNN) designed to overcome the vanishing gradient problem in traditional RNNs. It introduces memory cells with gating mechanisms to selectively learn and retain information over long sequences. This allows LSTM to capture long-term dependencies in sequential data, making it ideal for tasks like natural language processing and time series analysis.

2. Types of RNN
   RNN can take different forms based on the input-output relationships:
   - One-to-One: Simple RNN processing one input to one output.
   - One-to-Many: Single input generating multiple outputs.
   - Many-to-One: Sequential input producing a single output (e.g., sentiment analysis).
   - Many-to-Many: Both input and output are sequences.

3. Use case: Sentiment Analysis on the Twitter data set
   Objective:

   Perform sentiment analysis on tweets to determine positive, negative, or neutral sentiments.

   Methodology:
   - Data Preprocessing: Clean and tokenize Twitter text data.
   - Word Embeddings: Represent words as continuous vectors using pre-trained embeddings.

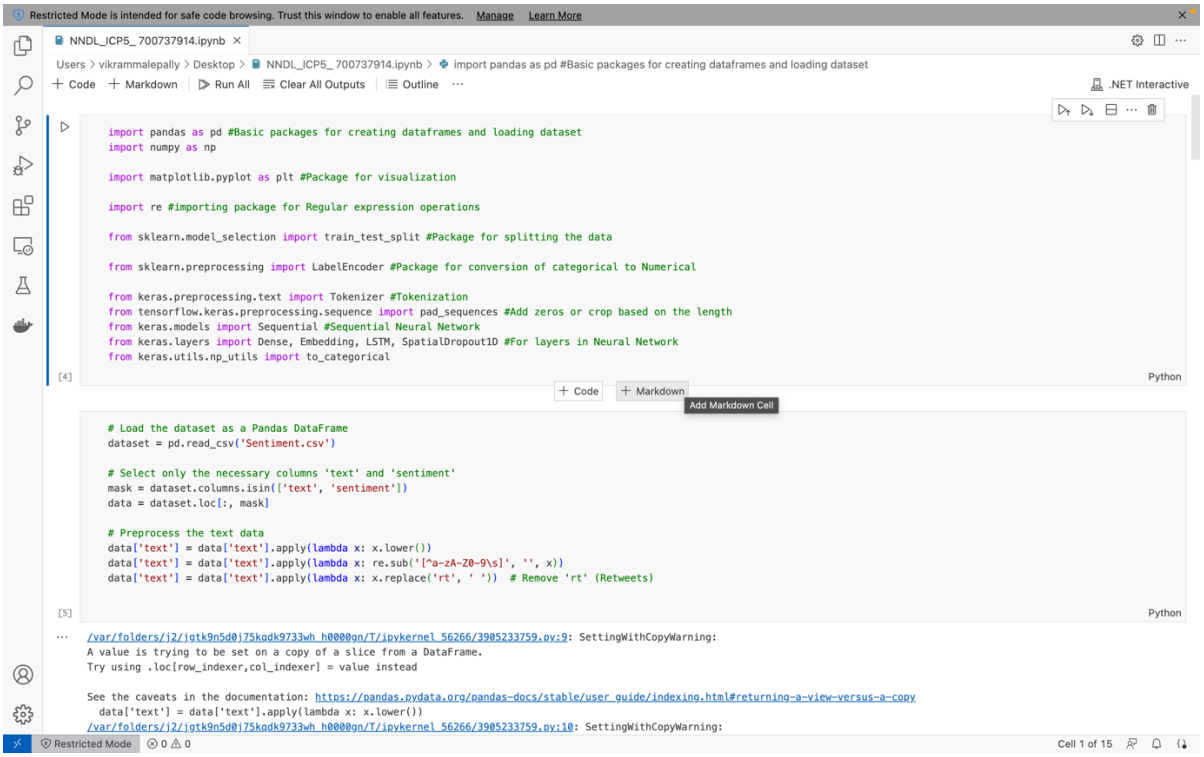
- LSTM Model: Build an LSTM-based deep learning model to capture sequential dependencies.
- Model Training: Train the LSTM model with sentiment labels as target.
- Evaluation: Evaluate model performance on a test dataset.
- Prediction: Use the model to predict sentiment for new tweets.

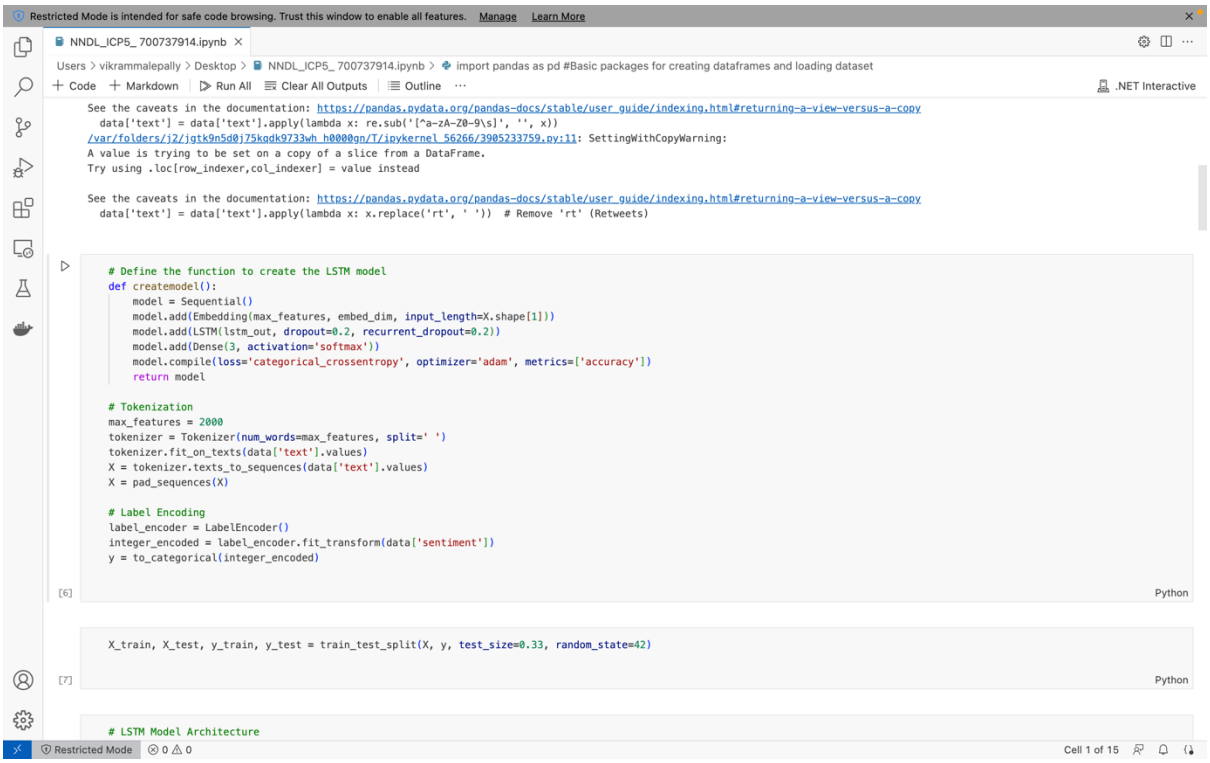
Benefits:

- LSTM's ability to handle sequential data makes it suitable for sentiment analysis on Twitter.
- Deep learning-based approaches, like LSTM, offer accurate sentiment predictions by learning complex patterns.
- Sentiment analysis on social media data provides valuable insights into public opinions and trends.

In class programming:

1. Save the model and use the saved model to predict on new text data (ex, “A lot of good things are happening. We are respected again throughout the world, and that's a great thing.@realDonaldTrump”)


Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
NNDL_ICP5_ 700737914.ipynb
Users > vikrammalepally > Desktop > NNDL_ICP5_ 700737914.ipynb > import pandas as pd #Basic packages for creating dataframes and loading dataset
Code Markdown Run All Clear All Outputs Outline
.NET Interactive
import pandas as pd #Basic packages for creating dataframes and loading dataset
import numpy as np
import matplotlib.pyplot as plt #Package for visualization
import re #importing package for Regular expression operations
from sklearn.model_selection import train_test_split #Package for splitting the data
from sklearn.preprocessing import LabelEncoder #Package for conversion of categorical to Numerical
from keras.preprocessing.text import Tokenizer #Tokenization
from tensorflow.keras.preprocessing.sequence import pad_sequences #Add zeros or crop based on the length
from keras.models import Sequential #Sequential Neural Network
from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D #For layers in Neural Network
from keras.utils.np_utils import to_categorical
[4]
Python
Code Markdown
Add Markdown Cell
# Load the dataset as a Pandas DataFrame
dataset = pd.read_csv('Sentiment.csv')
# Select only the necessary columns 'text' and 'sentiment'
mask = dataset.columns.isin(['text', 'sentiment'])
data = dataset.loc[:, mask]
# Preprocess the text data
data['text'] = data['text'].apply(lambda x: x.lower())
data['text'] = data['text'].apply(lambda x: re.sub('[^a-zA-Z0-9\s]', '', x))
data['text'] = data['text'].apply(lambda x: x.replace('rt', ' ')) # Remove 'rt' (Retweets)
[5]
Python
/var/folders/j2/jgtk9n5d0j75kqdk9733wh_h0000gn/T/ipykernel_56266/3905233759.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
data['text'] = data['text'].apply(lambda x: x.lower())
/var/folders/j2/jgtk9n5d0j75kqdk9733wh_h0000gn/T/ipykernel_56266/3905233759.py:10: SettingWithCopyWarning:
Restricted Mode
0 0
Cell 1 of 15


Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
NNDL_ICP5_ 700737914.ipynb
Users > vikrammalepally > Desktop > NNDL_ICP5_ 700737914.ipynb > import pandas as pd #Basic packages for creating dataframes and loading dataset
Code Markdown Run All Clear All Outputs Outline
.NET Interactive
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
data['text'] = data['text'].apply(lambda x: re.sub('[^a-zA-Z0-9\s]', '', x))
/var/folders/j2/jgtk9n5d0j75kqdk9733wh_h0000gn/T/ipykernel_56266/3905233759.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
data['text'] = data['text'].apply(lambda x: x.replace('rt', ' ')) # Remove 'rt' (Retweets)
# Define the function to create the LSTM model
def createmodel():
model = Sequential()
model.add(Embedding(max_features, embed_dim, input_length=X.shape[1]))
model.add(LSTM(lstm_out, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(3, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
return model
# Tokenization
max_features = 2000
tokenizer = Tokenizer(num_words=max_features, split=' ')
tokenizer.fit_on_texts(data['text'].values)
X = tokenizer.texts_to_sequences(data['text'].values)
X = pad_sequences(X)
# Label Encoding
label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(data['sentiment'])
y = to_categorical(integer_encoded)
[6]
Python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
[7]
Python
LSTM Model Architecture
Restricted Mode
0 0
Cell 1 of 15

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

NNDL_ICP5_ 700737914.ipynb

Users > vikrammalepally > Desktop > NNDL_ICP5_ 700737914.ipynb > import pandas as pd #Basic packages for creating dataframes and loading dataset

+ Code + Markdown | Run All | Clear All Outputs | Outline ··· .NET Interactive

```
# LSTM Model Architecture
embed_dim = 128
lstm_out = 196

model = Sequential()
model.add(Embedding(max_features, embed_dim, input_length=X.shape[1]))
model.add(LSTM(lstm_out, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(3, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Model Summary
print(model.summary())

# Train the model
history = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_test, y_test), verbose=2)
```

[8] Python

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 28, 128)           256000

 lstm (LSTM)                 (None, 196)               254800

 dense (Dense)               (None, 3)                 591

=================================================================
Total params: 511,391
Trainable params: 511,391
Non-trainable params: 0
_________________________________________________________________
None
Epoch 1/10
2023-07-31 14:58:59.547834: W tensorflow/tsl/platform/profile_utils/cpu_utils.cc:128] Failed to get CPU frequency: 0 Hz
291/291 - 22s - loss: 0.8312 - accuracy: 0.6380 - val_loss: 0.7549 - val_accuracy: 0.6660 - 22s/epoch - 75ms/step
Epoch 2/10
291/291 - 21s - loss: 0.6831 - accuracy: 0.7087 - val_loss: 0.7446 - val_accuracy: 0.6826 - 21s/epoch - 74ms/step
Epoch 3/10
291/291 - 22s - loss: 0.6141 - accuracy: 0.7429 - val_loss: 0.7552 - val_accuracy: 0.6817 - 22s/epoch - 75ms/step
Epoch 4/10
291/291 - 22s - loss: 0.5680 - accuracy: 0.7615 - val_loss: 0.7675 - val_accuracy: 0.6723 - 22s/epoch - 77ms/step
```

Restricted Mode ⊗ 0 ⚠ 0 Cell 1 of 15

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

NNDL_ICP5_ 700737914.ipynb

Users > vikrammalepally > Desktop > NNDL_ICP5_ 700737914.ipynb > import pandas as pd #Basic packages for creating dataframes and loading dataset

+ Code + Markdown | Run All | Clear All Outputs | Outline ··· .NET Interactive

```
None
Epoch 1/10
2023-07-31 14:58:59.547834: W tensorflow/tsl/platform/profile_utils/cpu_utils.cc:128] Failed to get CPU frequency: 0 Hz
291/291 - 22s - loss: 0.8312 - accuracy: 0.6380 - val_loss: 0.7549 - val_accuracy: 0.6660 - 22s/epoch - 75ms/step
Epoch 2/10
291/291 - 21s - loss: 0.6831 - accuracy: 0.7087 - val_loss: 0.7446 - val_accuracy: 0.6826 - 21s/epoch - 74ms/step
Epoch 3/10
291/291 - 22s - loss: 0.6141 - accuracy: 0.7429 - val_loss: 0.7552 - val_accuracy: 0.6817 - 22s/epoch - 75ms/step
Epoch 4/10
291/291 - 22s - loss: 0.5680 - accuracy: 0.7615 - val_loss: 0.7675 - val_accuracy: 0.6723 - 22s/epoch - 77ms/step
Epoch 5/10
291/291 - 23s - loss: 0.5267 - accuracy: 0.7850 - val_loss: 0.8480 - val_accuracy: 0.6662 - 23s/epoch - 80ms/step
Epoch 6/10
291/291 - 24s - loss: 0.4838 - accuracy: 0.8062 - val_loss: 0.8630 - val_accuracy: 0.6660 - 24s/epoch - 83ms/step
Epoch 7/10
291/291 - 23s - loss: 0.4405 - accuracy: 0.8217 - val_loss: 0.9373 - val_accuracy: 0.6601 - 23s/epoch - 78ms/step
Epoch 8/10
291/291 - 23s - loss: 0.4068 - accuracy: 0.8354 - val_loss: 0.9919 - val_accuracy: 0.6627 - 23s/epoch - 79ms/step
Epoch 9/10
291/291 - 23s - loss: 0.3770 - accuracy: 0.8469 - val_loss: 1.0417 - val_accuracy: 0.6560 - 23s/epoch - 80ms/step
Epoch 10/10
291/291 - 23s - loss: 0.3484 - accuracy: 0.8585 - val_loss: 1.2010 - val_accuracy: 0.6496 - 23s/epoch - 81ms/step
```

```
# Evaluate the model on test data
score, accuracy = model.evaluate(X_test, y_test, verbose=2, batch_size=32)
print("Test Loss:", score)
print("Test Accuracy:", accuracy)


# Plot training and validation accuracy over epochs
plt.plot(history.history['accuracy'], label='Training accuracy')
plt.plot(history.history['val_accuracy'], label='Validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.title('Training and Validation accuracy')
plt.show()

# Plot training and validation loss over epochs
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
```

Restricted Mode ⊗ 0 ⚠ 0 Cell 1 of 15

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

NNDL_ICP5_ 700737914.ipynb

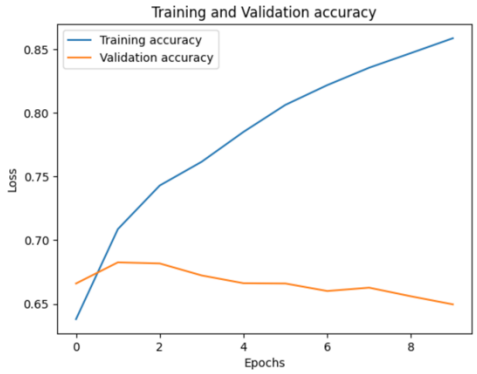
Users > vikrammalepally > Desktop > NNDL_ICP5_ 700737914.ipynb > import pandas as pd #Basic packages for creating dataframes and loading dataset

+ Code + Markdown | Run All | Clear All Outputs | Outline ··· .NET Interactive

```
# Plot training and validation loss over epochs
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.title('Training and Validation Loss')
plt.show()
```

[9] Python

```
144/144 - 2s - loss: 1.2010 - accuracy: 0.6496 - 2s/epoch - 12ms/step
Test Loss: 1.200998306274414
Test Accuracy: 0.6496286392211914
```



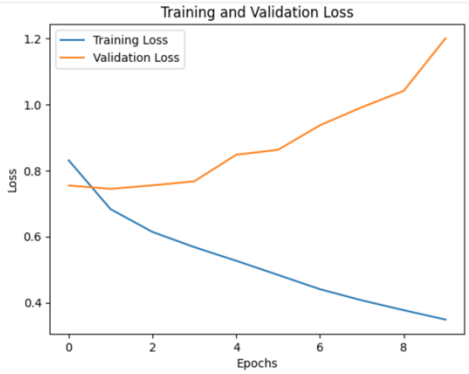
Restricted Mode 0 0 Cell 1 of 15

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

NNDL_ICP5_ 700737914.ipynb

Users > vikrammalepally > Desktop > NNDL_ICP5_ 700737914.ipynb > import pandas as pd #Basic packages for creating dataframes and loading dataset

+ Code + Markdown | Run All | Clear All Outputs | Outline ··· .NET Interactive




```
# Save the trained model
model.save('sentimentAnalysis.h5')
```

[10] Python

```
from keras.models import load_model

model = load_model('sentimentAnalysis.h5')
```

[11] Python

Restricted Mode 0 0 Cell 1 of 15

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

NNDL_ICP5_ 700737914.ipynb

Users > vikrammalepally > Desktop > NNDL_ICP5_ 700737914.ipynb > import pandas as pd #Basic packages for creating dataframes and loading dataset

+ Code + Markdown | Run All | Clear All Outputs | Outline ··· .NET Interactive


```
# Save the trained model
model.save('sentimentAnalysis.h5')
```

[10] Python

```
from keras.models import load_model

model = load_model('sentimentAnalysis.h5')
```

[11] Python

Restricted Mode ⊗ 0 ⚠ 0 Cell 1 of 15

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

NNDL_ICP5_ 700737914.ipynb

Users > vikrammalepally > Desktop > NNDL_ICP5_ 700737914.ipynb > import pandas as pd #Basic packages for creating dataframes and loading dataset

+ Code + Markdown | Run All | Clear All Outputs | Outline ··· .NET Interactive

```
from keras.models import load_model

model = load_model('sentimentAnalysis.h5')
```

[11] Python

```
# Define the text data to predict sentiment
sentence = ['A lot of good things are happening. We are respected again throughout the world, and that is a great thing. @realDonaldTrump']

# Tokenize and pad the sentence
sentence = tokenizer.texts_to_sequences(sentence)
sentence = pad_sequences(sentence, maxlen=28, dtype='int32', value=0)
```

[12] Python

+ Code + Markdown

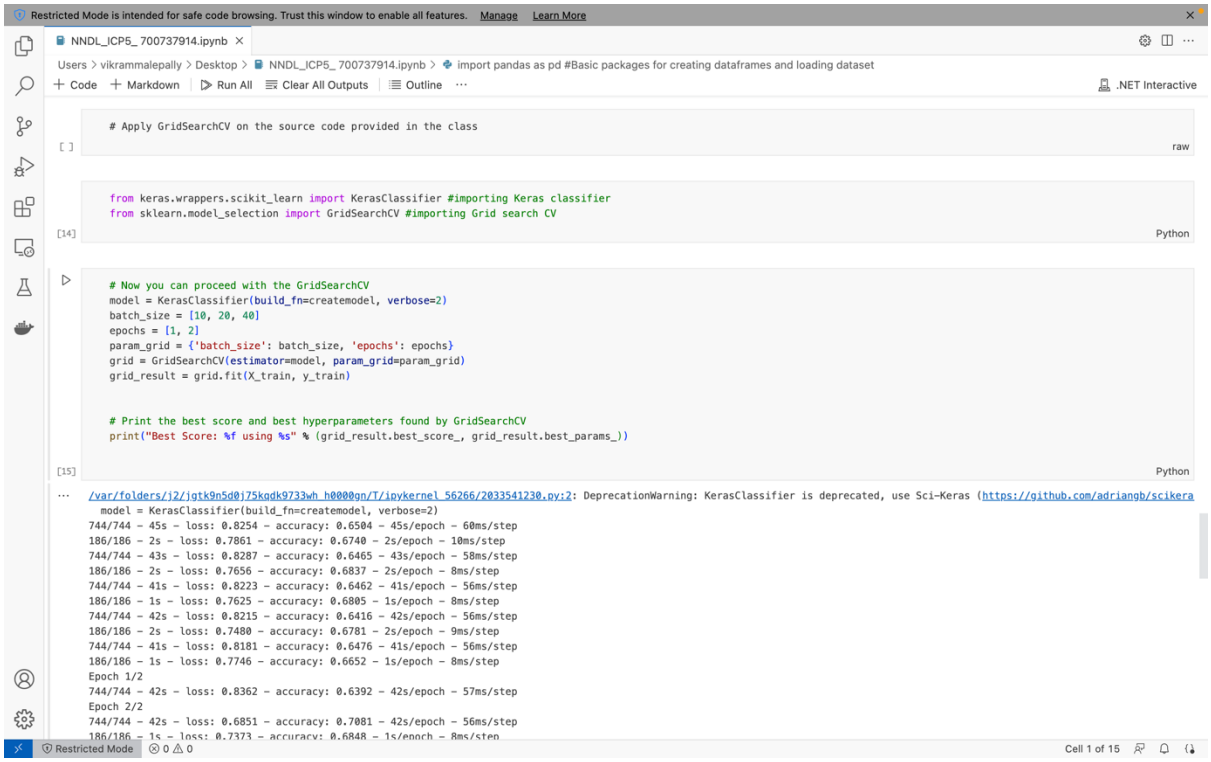
```
# Make predictions using the loaded model
sentiment_probs = model.predict(sentence, batch_size=1, verbose=2)[0]

# Convert sentiment probabilities to sentiment label
sentiment = np.argmax(sentiment_probs)

# Print the sentiment label
if sentiment == 0:
    print("Neutral")
elif sentiment < 0:
    print("Negative")
elif sentiment > 0:
    print("Positive")
else:
    print("Cannot be determined")
```

[13] Python

```
1/1 - 0s - 129ms/epoch - 129ms/step
Positive
```

Restricted Mode ⊗ 0 ⚠ 0 Cell 1 of 15

2. Apply GridSearchCV on the source code provided in the class



Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

NNDL_ICP5_ 700737914.ipynb

Users > vikrammalepally > Desktop > NNDL_ICP5_ 700737914.ipynb > import pandas as pd #Basic packages for creating dataframes and loading dataset

+ Code + Markdown | Run All | Clear All Outputs | Outline ... .NET Interactive

[15] Python

```
/var/folders/j2/jgtk9n5d0j75kqdk9733wh_h0000gn/T/ipykernel_56266/2033541230.py:2: DeprecationWarning: KerasClassifier is deprecated, use Sci-Keras (https://github.com/adriangb/scikera
  model = KerasClassifier(build_fn=createmodel, verbose=2)
744/744 - 45s - loss: 0.8254 - accuracy: 0.6504 - 45s/epoch - 60ms/step
186/186 - 2s - loss: 0.7861 - accuracy: 0.6740 - 2s/epoch - 10ms/step
744/744 - 43s - loss: 0.8287 - accuracy: 0.6465 - 43s/epoch - 58ms/step
186/186 - 2s - loss: 0.7656 - accuracy: 0.6837 - 2s/epoch - 8ms/step
744/744 - 41s - loss: 0.8223 - accuracy: 0.6462 - 41s/epoch - 56ms/step
186/186 - 1s - loss: 0.7625 - accuracy: 0.6805 - 1s/epoch - 8ms/step
744/744 - 42s - loss: 0.8215 - accuracy: 0.6416 - 42s/epoch - 56ms/step
186/186 - 2s - loss: 0.7480 - accuracy: 0.6781 - 2s/epoch - 9ms/step
744/744 - 41s - loss: 0.8181 - accuracy: 0.6476 - 41s/epoch - 56ms/step
186/186 - 1s - loss: 0.7746 - accuracy: 0.6652 - 1s/epoch - 8ms/step
Epoch 1/2
744/744 - 42s - loss: 0.8362 - accuracy: 0.6392 - 42s/epoch - 57ms/step
Epoch 2/2
744/744 - 42s - loss: 0.6851 - accuracy: 0.7081 - 42s/epoch - 56ms/step
186/186 - 1s - loss: 0.7373 - accuracy: 0.6848 - 1s/epoch - 8ms/step
Epoch 1/2
744/744 - 42s - loss: 0.8215 - accuracy: 0.6465 - 42s/epoch - 57ms/step
Epoch 2/2
744/744 - 42s - loss: 0.6880 - accuracy: 0.7086 - 42s/epoch - 56ms/step
186/186 - 1s - loss: 0.7494 - accuracy: 0.6826 - 1s/epoch - 7ms/step
Epoch 1/2
744/744 - 43s - loss: 0.8240 - accuracy: 0.6425 - 43s/epoch - 57ms/step
Epoch 2/2
744/744 - 73s - loss: 0.6729 - accuracy: 0.7140 - 73s/epoch - 98ms/step
186/186 - 2s - loss: 0.7761 - accuracy: 0.6794 - 2s/epoch - 9ms/step
...
465/465 - 30s - loss: 0.8117 - accuracy: 0.6530 - 30s/epoch - 64ms/step
Epoch 2/2
465/465 - 29s - loss: 0.6741 - accuracy: 0.7206 - 29s/epoch - 63ms/step
Best Score: 0.682126 using {'batch_size': 20, 'epochs': 2}
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```
# Plot the results of GridSearchCV
mean_scores = grid_result.cv_results_['mean_test_score']
param_batch_size = grid_result.cv_results_['param_batch_size']
param_epochs = grid_result.cv_results_['param_epochs']
```

Restricted Mode 0 0 Cell 1 of 15

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

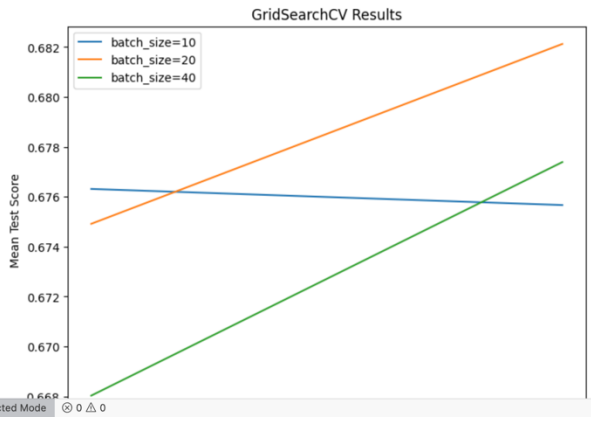
NNDL_ICP5_ 700737914.ipynb

Users > vikrammalepally > Desktop > NNDL_ICP5_ 700737914.ipynb > import pandas as pd #Basic packages for creating dataframes and loading dataset

+ Code + Markdown | Run All | Clear All Outputs | Outline ... .NET Interactive

```
# Plot the results of GridSearchCV
mean_scores = grid_result.cv_results_['mean_test_score']
param_batch_size = grid_result.cv_results_['param_batch_size']
param_epochs = grid_result.cv_results_['param_epochs']

plt.figure(figsize=(8, 6))
for i, batch_size in enumerate(batch_size):
    plt.plot(epochs, mean_scores[i * len(epochs): (i + 1) * len(epochs)], label=f'batch_size={batch_size}')

plt.xlabel('Number of Epochs')
plt.ylabel('Mean Test Score')
plt.title('GridSearchCV Results')
plt.legend()
plt.show()
```

[21] Python



Restricted Mode 0 0 Cell 1 of 15