Neural Networks & Deep Learning -

Saikumar Reddy Mandha(700765227)

GitHub Link: https://github.com/saikumarreddyMandha/Assignment-7/tree/main

Video Record : https://drive.google.com/file/d/1SWcUZeBOPzXle66drvvgTijO64AakAGC/view?usp=drive_link

Use Case Description:

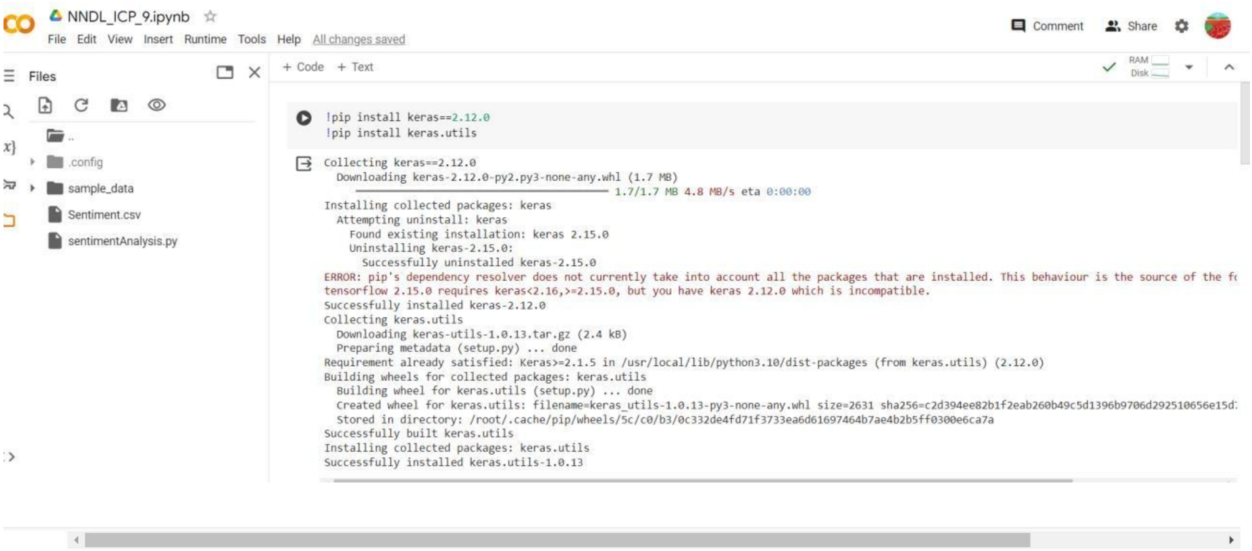
Sentiment Analysis on the Twitter dataset

Programming elements:

1. Basics of LSTM
2. Types of RNN
3. Use case: Sentiment Analysis on the Twitter data set

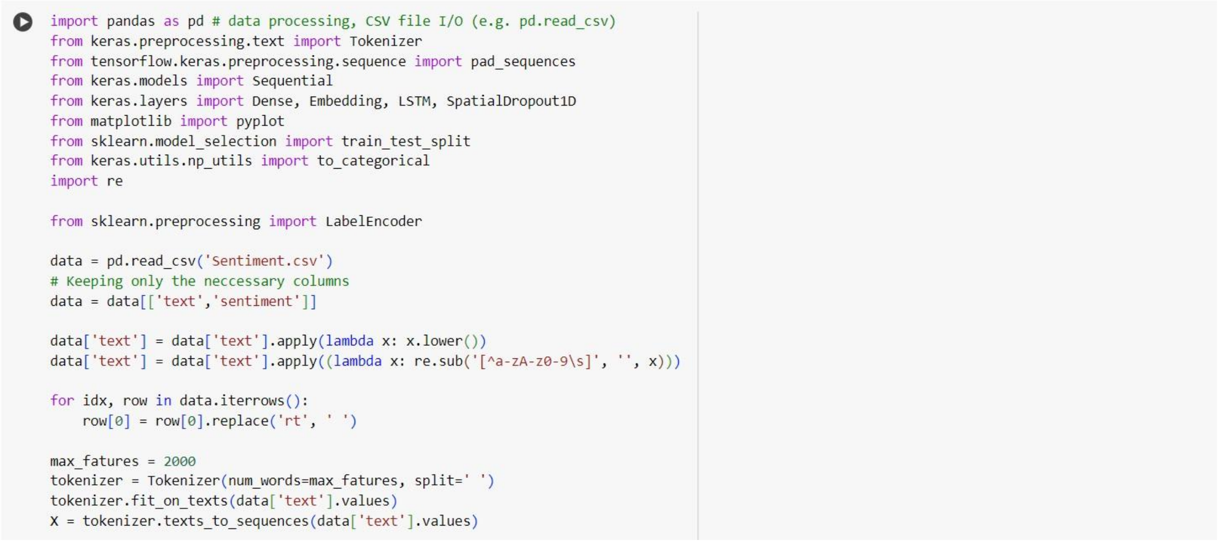
In class programming:

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

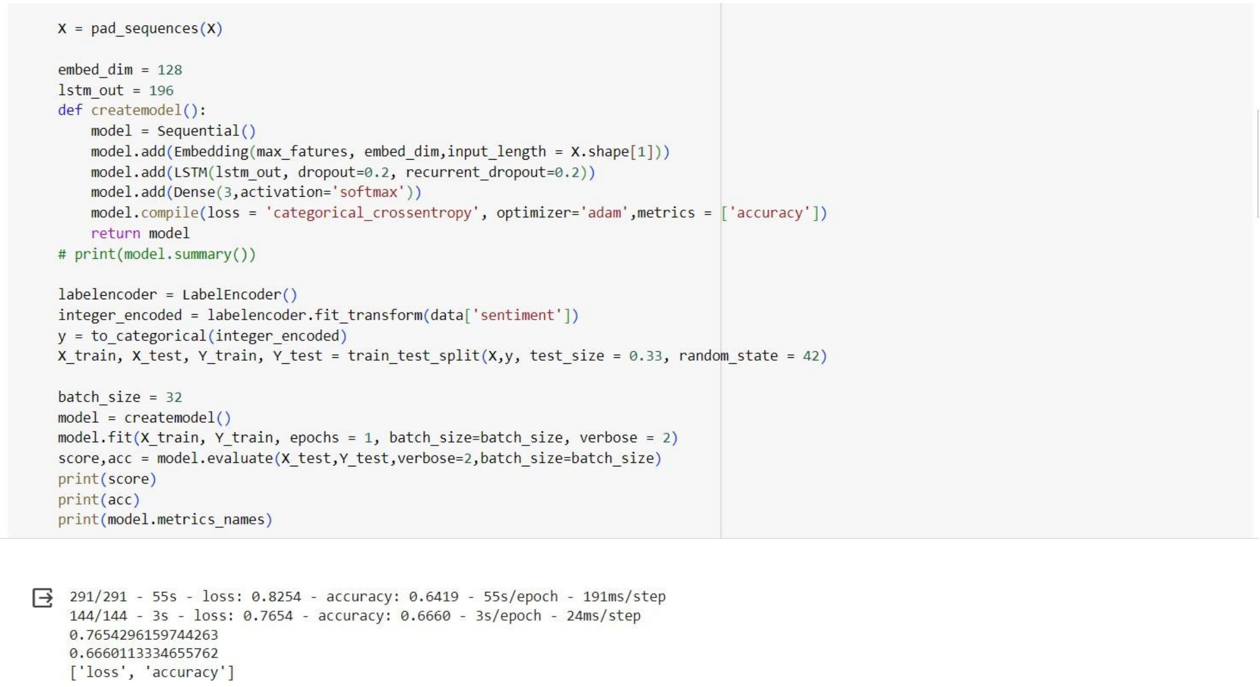


```
!pip install keras==2.12.0
!pip install keras.utils
```

```
Collecting keras==2.12.0
  Downloading keras-2.12.0-py2.py3-none-any.whl (1.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 4.8 MB/s eta 0:00:00
Installing collected packages: keras
  Attempting uninstall: keras
    Found existing installation: keras 2.15.0
    Uninstalling keras-2.15.0:
      Successfully uninstalled keras-2.15.0
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the fo
tensorflow 2.15.0 requires keras<2.16,>=2.15.0, but you have keras 2.12.0 which is incompatible.
Successfully installed keras-2.12.0
Collecting keras.utils
  Downloading keras-utils-1.0.13.tar.gz (2.4 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: Keras>=2.1.5 in /usr/local/lib/python3.10/dist-packages (from keras.utils) (2.12.0)
Building wheels for collected packages: keras.utils
  Building wheel for keras.utils (setup.py) ... done
  Created wheel for keras.utils: filename=keras_utils-1.0.13-py3-none-any.whl size=2631 sha256=c2d394ee82b1f2eab260b49c5d1396b9706d292510656e15d
  Stored in directory: /root/.cache/pip/wheels/5c/c0/b3/0c332de4fd71f3733ea6d61697464b7ae4b2b5ff0300e6ca7a
Successfully built keras.utils
Installing collected packages: keras.utils
Successfully installed keras.utils-1.0.13
```

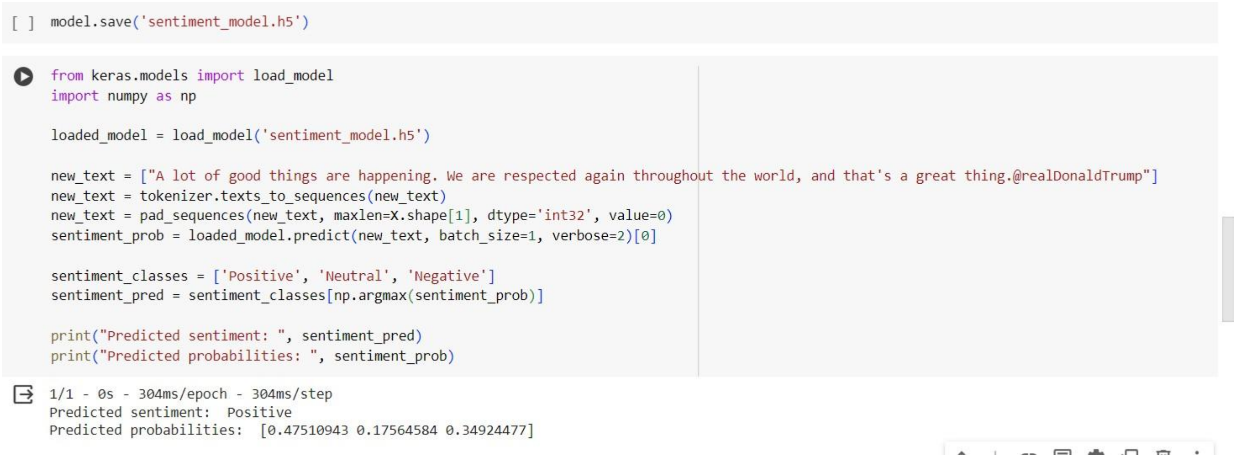


```python
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from keras.utils.np_utils import to_categorical
import re

from sklearn.preprocessing import LabelEncoder

data = pd.read_csv('Sentiment.csv')
# Keeping only the neccessary columns
data = data[['text','sentiment']]

data['text'] = data['text'].apply(lambda x: x.lower())
data['text'] = data['text'].apply((lambda x: re.sub('[^a-zA-z0-9\s]', '', x)))

for idx, row in data.iterrows():
    row[0] = row[0].replace('rt', ' ')

max_fatures = 2000
tokenizer = Tokenizer(num_words=max_fatures, split=' ')
tokenizer.fit_on_texts(data['text'].values)
X = tokenizer.texts_to_sequences(data['text'].values)
```



```
X = pad_sequences(X)

embed_dim = 128
lstm_out = 196
def createmodel():
    model = Sequential()
    model.add(Embedding(max_fatures, 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
# print(model.summary())

labelencoder = LabelEncoder()
integer_encoded = labelencoder.fit_transform(data['sentiment'])
y = to_categorical(integer_encoded)
X_train, X_test, Y_train, Y_test = train_test_split(X,y, test_size = 0.33, random_state = 42)

batch_size = 32
model = createmodel()
model.fit(X_train, Y_train, epochs = 1, batch_size=batch_size, verbose = 2)
score,acc = model.evaluate(X_test,Y_test,verbose=2,batch_size=batch_size)
print(score)
print(acc)
print(model.metrics_names)
```

```
291/291 - 55s - loss: 0.8254 - accuracy: 0.6419 - 55s/epoch - 191ms/step
144/144 - 3s - loss: 0.7654 - accuracy: 0.6660 - 3s/epoch - 24ms/step
0.7654296159744263
0.6660113334655762
['loss', 'accuracy']
```



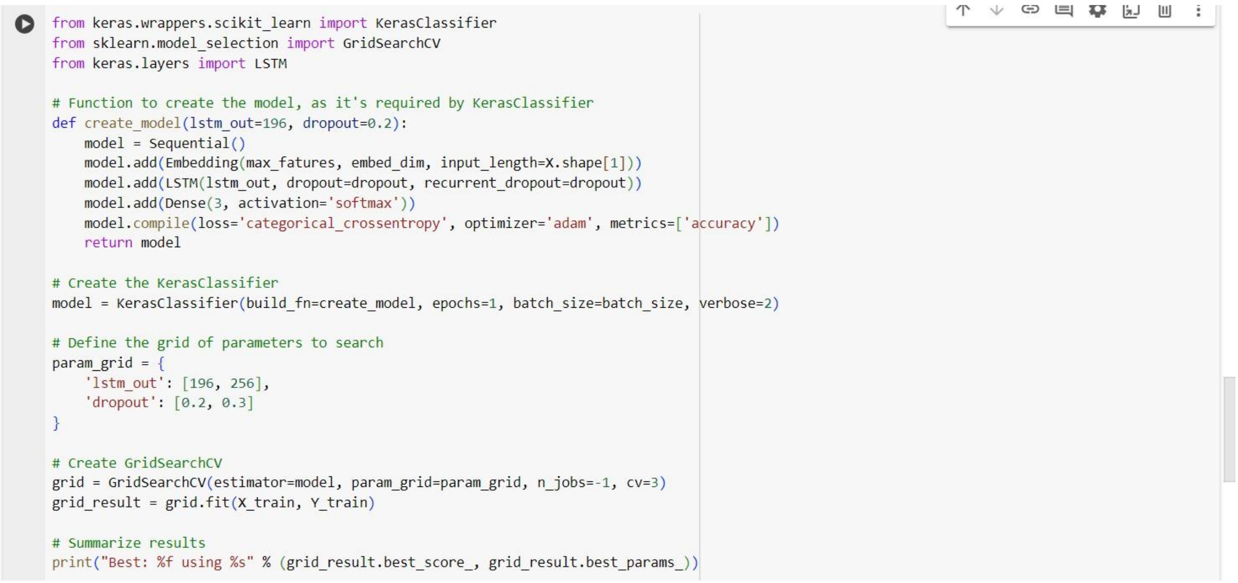
```
[ ] model.save('sentiment_model.h5')
```

```
from keras.models import load_model
import numpy as np

loaded_model = load_model('sentiment_model.h5')

new_text = ["A lot of good things are happening. We are respected again throughout the world, and that's a great thing.@realDonaldTrump"]
new_text = tokenizer.texts_to_sequences(new_text)
new_text = pad_sequences(new_text, maxlen=X.shape[1], dtype='int32', value=0)
sentiment_prob = loaded_model.predict(new_text, batch_size=1, verbose=2)[0]

sentiment_classes = ['Positive', 'Neutral', 'Negative']
sentiment_pred = sentiment_classes[np.argmax(sentiment_prob)]

print("Predicted sentiment: ", sentiment_pred)
print("Predicted probabilities: ", sentiment_prob)
```

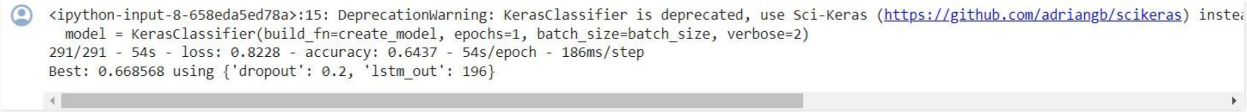
```
1/1 - 0s - 304ms/epoch - 304ms/step
Predicted sentiment:  Positive
Predicted probabilities:  [0.47510943 0.17564584 0.34924477]
```

This code loads the saved model using the load_model function, and then preprocesses the new text data in the same way as the training data. The predict method is called on the loaded model to get the predicted class probabilities for the new text data. The class with the highest probability is chosen as the predicted sentiment. The predicted sentiment and probabilities are then printed to the console.

To apply GridSearchCV on the provided source code, we can use the GridSearchCV class from sklearn to search for the best combination of hyperparameters for the LSTM model. The hyperparameters that can be tuned are the number of LSTM units, the dropout rate, and the learning rate of the optimizer.



```
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import GridSearchCV
from keras.layers import LSTM

# Function to create the model, as it's required by KerasClassifier
def create_model(lstm_out=196, dropout=0.2):
    model = Sequential()
    model.add(Embedding(max_fatures, embed_dim, input_length=X.shape[1]))
    model.add(LSTM(lstm_out, dropout=dropout, recurrent_dropout=dropout))
    model.add(Dense(3, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model

# Create the KerasClassifier
model = KerasClassifier(build_fn=create_model, epochs=1, batch_size=batch_size, verbose=2)

# Define the grid of parameters to search
param_grid = {
    'lstm_out': [196, 256],
    'dropout': [0.2, 0.3]
}

# Create GridSearchCV
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1, cv=3)
grid_result = grid.fit(X_train, Y_train)

# Summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
```



```
<ipython-input-8-658eda5ed78a>:15: DeprecationWarning: KerasClassifier is deprecated, use Sci-Keras (https://github.com/adriangb/scikeras) instea
  model = KerasClassifier(build_fn=create_model, epochs=1, batch_size=batch_size, verbose=2)
291/291 - 54s - loss: 0.8228 - accuracy: 0.6437 - 54s/epoch - 186ms/step
Best: 0.668568 using {'dropout': 0.2, 'lstm_out': 196}
```

This code defines the create_model function that returns a Keras model with the specified hyperparameters. The KerasClassifier class is used to create a wrapper for the create_model function,
which can be used as an estimator for GridSearchCV. The hyperparameters to be tuned are defined in the param_grid dictionary. GridSearchCV is then called with the KerasClassifier object, the param_grid dictionary.