Ian Pope 700717419 Big Data Analytics ICP 7

Ian Pope 700717419 Big Data Analytics ICP 7

Be able to get KerasClassifier

```
[2]  !pip install scikeras
```

```
Collecting scikeras
  Downloading scikeras-0.13.0-py3-none-any.whl.metadata (3.1 kB)
Requirement already satisfied: keras>=3.2.0 in /usr/local/lib/python3.10/dist-packages (from scikera
Requirement already satisfied: scikit-learn>=1.4.2 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: absl-py in /usr/local/lib/python3.10/dist-packages (from keras>=3.2.0
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from keras>=3.2.0->
Requirement already satisfied: rich in /usr/local/lib/python3.10/dist-packages (from keras>=3.2.0->s
Requirement already satisfied: namex in /usr/local/lib/python3.10/dist-packages (from keras>=3.2.0->
Requirement already satisfied: h5py in /usr/local/lib/python3.10/dist-packages (from keras>=3.2.0->s
Requirement already satisfied: optree in /usr/local/lib/python3.10/dist-packages (from keras>=3.2.0-
Requirement already satisfied: ml-dtypes in /usr/local/lib/python3.10/dist-packages (from keras>=3.2
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from keras>=3.2
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from scikit-
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from scikit
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: typing-extensions>=4.5.0 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.10/dist-packages (fro
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (f
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-packages (from markdown-
Downloading scikeras-0.13.0-py3-none-any.whl (26 kB)
Installing collected packages: scikeras
Successfully installed scikeras-0.13.0
```

Do all of the preprocessing of the data

```
# import the libraries
import tensorflow as tf
import pandas as pd
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from scikeras.wrappers import KerasClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
import numpy as np
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
import re
from sklearn.model_selection import train_test_split

data = pd.read_csv('/content/drive/MyDrive/Colab_Notebooks/Sentiment.csv')
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_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)

from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.utils import to_categorical
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)
print(X_train.shape,Y_train.shape)
print(X_test.shape,Y_test.shape)
```

```
<ipython-input-10-09df1dc9bd37>:23: FutureWarning: Series.__getitem__ treating keys as position
  row[0] = row[0].replace('rt','')
<ipython-input-10-09df1dc9bd37>:23: FutureWarning: Series.__setitem__ treating keys as position
  row[0] = row[0].replace('rt','')
(9293, 28) (9293, 3)
(4578, 28) (4578, 3)
```

Set up model to run GridSearchCV, get parameter list, and save model

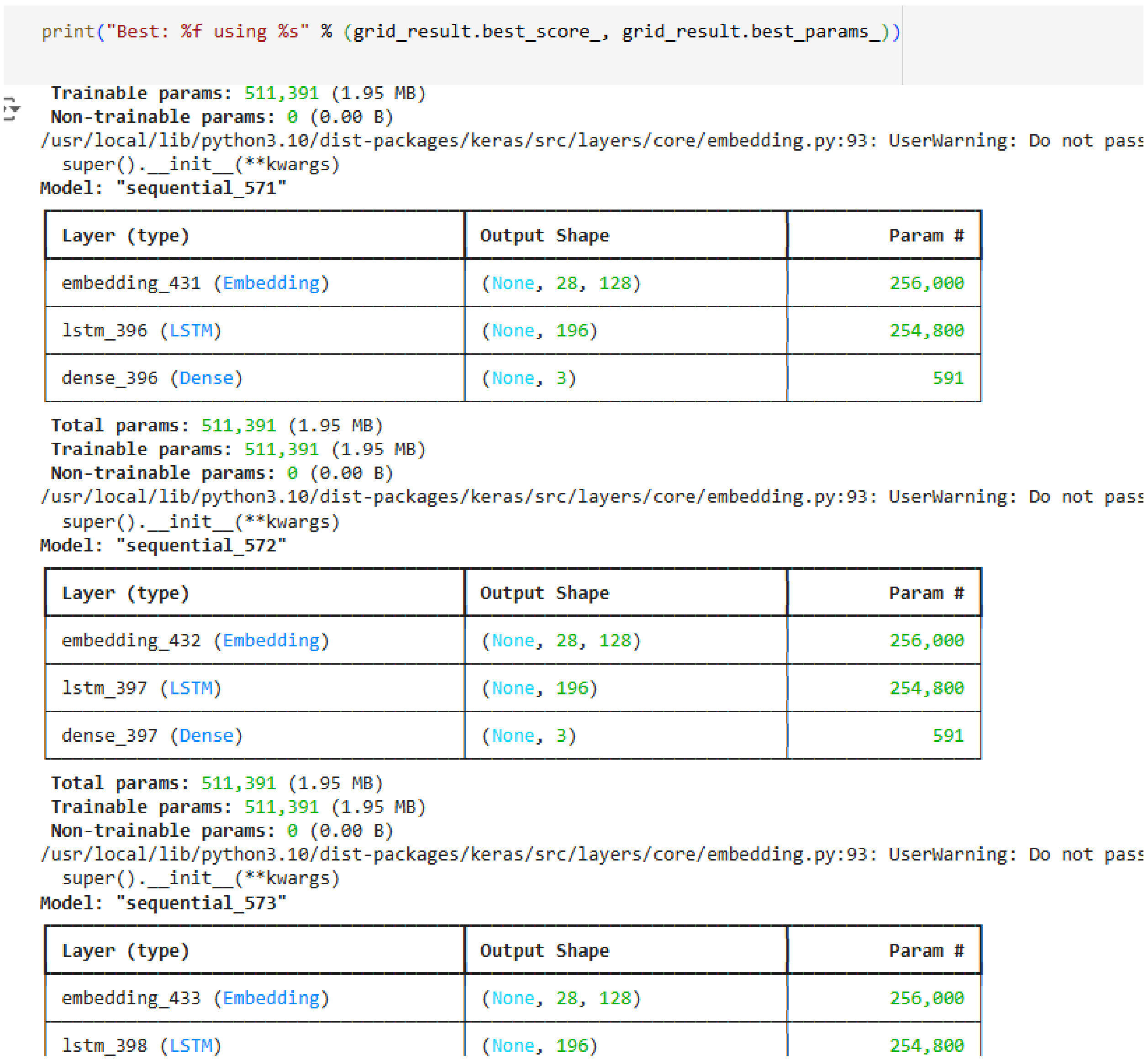
It should be noted that I had to reduce the number of hyperparameters to allow code to run in acceptable timeframe

```python
import tensorflow as tf
def create_model(optimizer='adam',activation='softmax',dropout_rate=0.2):
  max_features = 2000
  embed_dim = 128
  lstm_out = 196
  model = tf.keras.models.Sequential()
  model.add(tf.keras.layers.Embedding(max_features, embed_dim,input_shape=(X.shape[1],)))
  model.add(tf.keras.layers.LSTM(lstm_out, dropout=dropout_rate, recurrent_dropout=0.2))
  model.add(tf.keras.layers.Dense(3,activation=activation))
  model.compile(loss = 'categorical_crossentropy', optimizer=optimizer,metrics = ['accuracy'])
  (model.summary())
  return model

model = KerasClassifier(model=create_model,optimizer='adam',activation='softmax',dropout_rate=0.2,verbose=0)

batch_size = [40,]
epochs = [3,]
model__optimizer = ['SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax', 'Nadam']
model__init_mode = ['uniform', 'lecun_uniform', 'normal', 'zero', 'glorot_normal', 'glorot_uniform', 'he_normal', 'he_uniform']
model__activation = ['softmax', 'softplus', 'softsign', 'relu', 'tanh', 'sigmoid', 'hard_sigmoid', 'linear']
model__dropout_rate = [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
'''
batch_size = [10,20,40,]
epochs = [1,2,3,]
model__optimizer = ['SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax', 'Nadam']
model__init_mode = ['uniform', 'lecun_uniform', 'normal', 'zero', 'glorot_normal', 'glorot_uniform', 'he_normal', 'he_uniform']
model__activation = ['softmax', 'softplus', 'softsign', 'relu', 'tanh', 'sigmoid', 'hard_sigmoid', 'linear']
model__dropout_rate = [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
param_grid = dict(batch_size=batch_size, epochs=epochs, optimizer=optimizer,
                  init_mode=init_mode, activation=activation, dropout_rate=dropout_rate)
'''

param_grid = dict(batch_size=batch_size, epochs=epochs, model__optimizer=model__optimizer)
grid = GridSearchCV(estimator=model, param_grid=param_grid, cv=2)
grid_result = grid.fit(X_train, Y_train)
```



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

```
 Trainable params: 511,391 (1.95 MB)
 Non-trainable params: 0 (0.00 B)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/embedding.py:93: UserWarning: Do not pass
  super().__init__(**kwargs)
Model: "sequential_571"
```

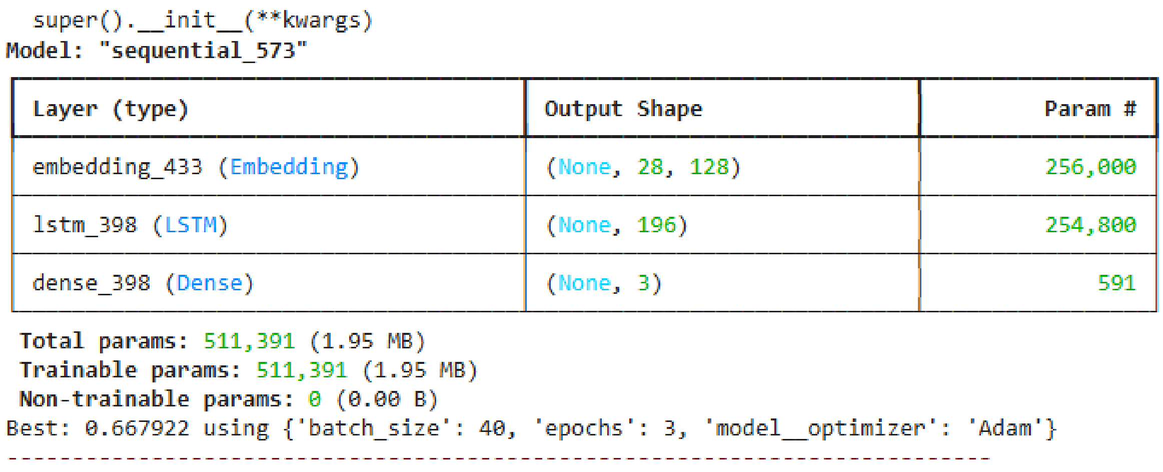
Layer (type)	Output Shape	Param #
embedding_431 (Embedding)	(None, 28, 128)	256,000
lstm_396 (LSTM)	(None, 196)	254,800
dense_396 (Dense)	(None, 3)	591

```
 Total params: 511,391 (1.95 MB)
 Trainable params: 511,391 (1.95 MB)
 Non-trainable params: 0 (0.00 B)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/embedding.py:93: UserWarning: Do not pass
  super().__init__(**kwargs)
Model: "sequential_572"
```

Layer (type)	Output Shape	Param #
embedding_432 (Embedding)	(None, 28, 128)	256,000
lstm_397 (LSTM)	(None, 196)	254,800
dense_397 (Dense)	(None, 3)	591

```
 Total params: 511,391 (1.95 MB)
 Trainable params: 511,391 (1.95 MB)
 Non-trainable params: 0 (0.00 B)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/embedding.py:93: UserWarning: Do not pass
  super().__init__(**kwargs)
Model: "sequential_573"
```

Layer (type)	Output Shape	Param #
embedding_433 (Embedding)	(None, 28, 128)	256,000
lstm_398 (LSTM)	(None, 196)	254,800



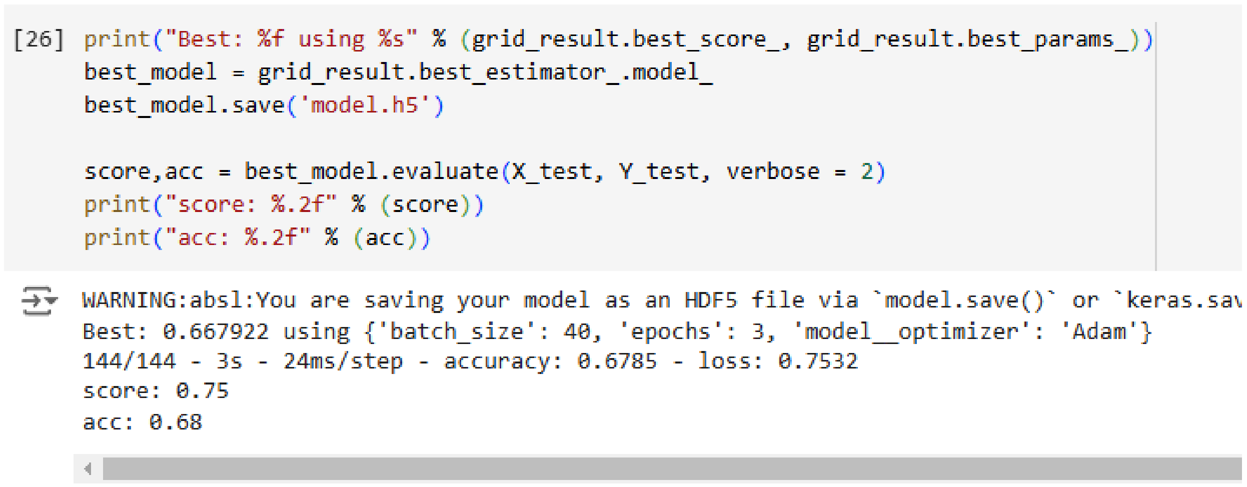
```
  super().__init__(**kwargs)
Model: "sequential_573"
```

Layer (type)	Output Shape	Param #
embedding_433 (Embedding)	(None, 28, 128)	256,000
lstm_398 (LSTM)	(None, 196)	254,800
dense_398 (Dense)	(None, 3)	591

```
 Total params: 511,391 (1.95 MB)
 Trainable params: 511,391 (1.95 MB)
 Non-trainable params: 0 (0.00 B)
Best: 0.667922 using {'batch_size': 40, 'epochs': 3, 'model__optimizer': 'Adam'}
```

Results: Batch: 40, Epochs: 3, Optimizer: 'Adam'

Get the score and accuracy of the model on the test data

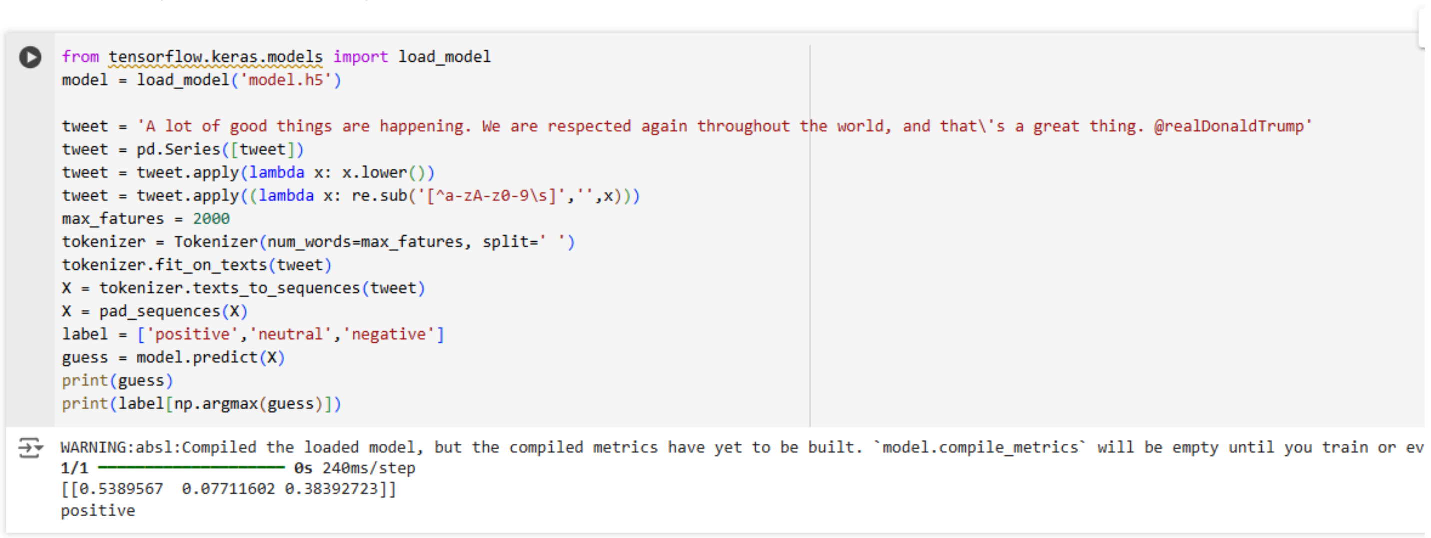


```
[26] print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
     best_model = grid_result.best_estimator_.model_
     best_model.save('model.h5')

     score,acc = best_model.evaluate(X_test, Y_test, verbose = 2)
     print("score: %.2f" % (score))
     print("acc: %.2f" % (acc))
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.sav
Best: 0.667922 using {'batch_size': 40, 'epochs': 3, 'model__optimizer': 'Adam'}
144/144 - 3s - 24ms/step - accuracy: 0.6785 - loss: 0.7532
score: 0.75
acc: 0.68
```

Load model and predict based on new input



```
from tensorflow.keras.models import load_model
model = load_model('model.h5')

tweet = 'A lot of good things are happening. We are respected again throughout the world, and that\'s a great thing. @realDonaldTrump'
tweet = pd.Series([tweet])
tweet = tweet.apply(lambda x: x.lower())
tweet = tweet.apply((lambda x: re.sub('[^a-zA-z0-9\s]','',x)))
max_fatures = 2000
tokenizer = Tokenizer(num_words=max_fatures, split=' ')
tokenizer.fit_on_texts(tweet)
X = tokenizer.texts_to_sequences(tweet)
X = pad_sequences(X)
label = ['positive','neutral','negative']
guess = model.predict(X)
print(guess)
print(label[np.argmax(guess)])
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or ev
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 240ms/step
[[0.5389567  0.07711602 0.38392723]]
positive
```