Neural Networks and deep learning – ICP6

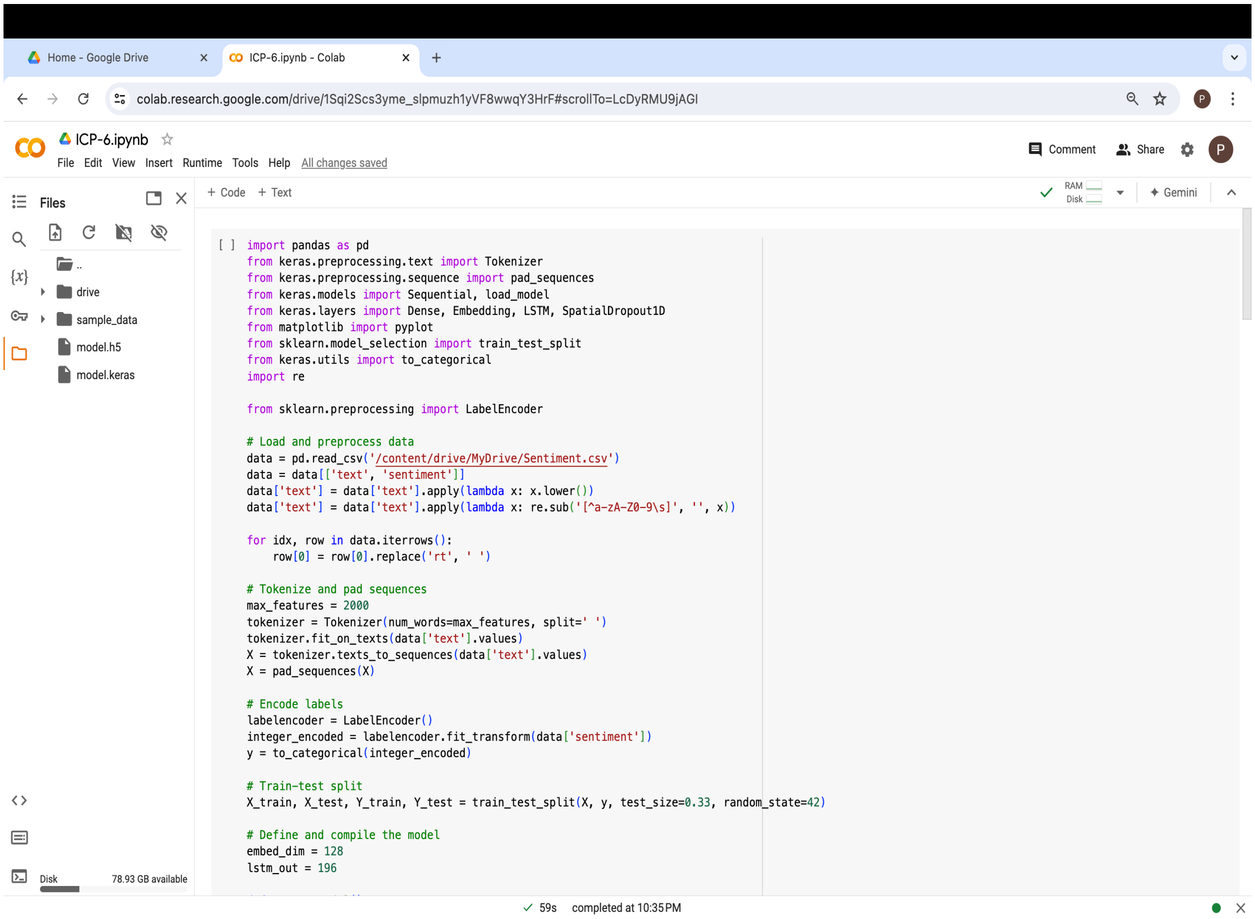
Name: Nalluri Prajwala

Student ID – 700766230

Github link: https://github.com/Prajwalanalluri/Neural-Assignment-6.git

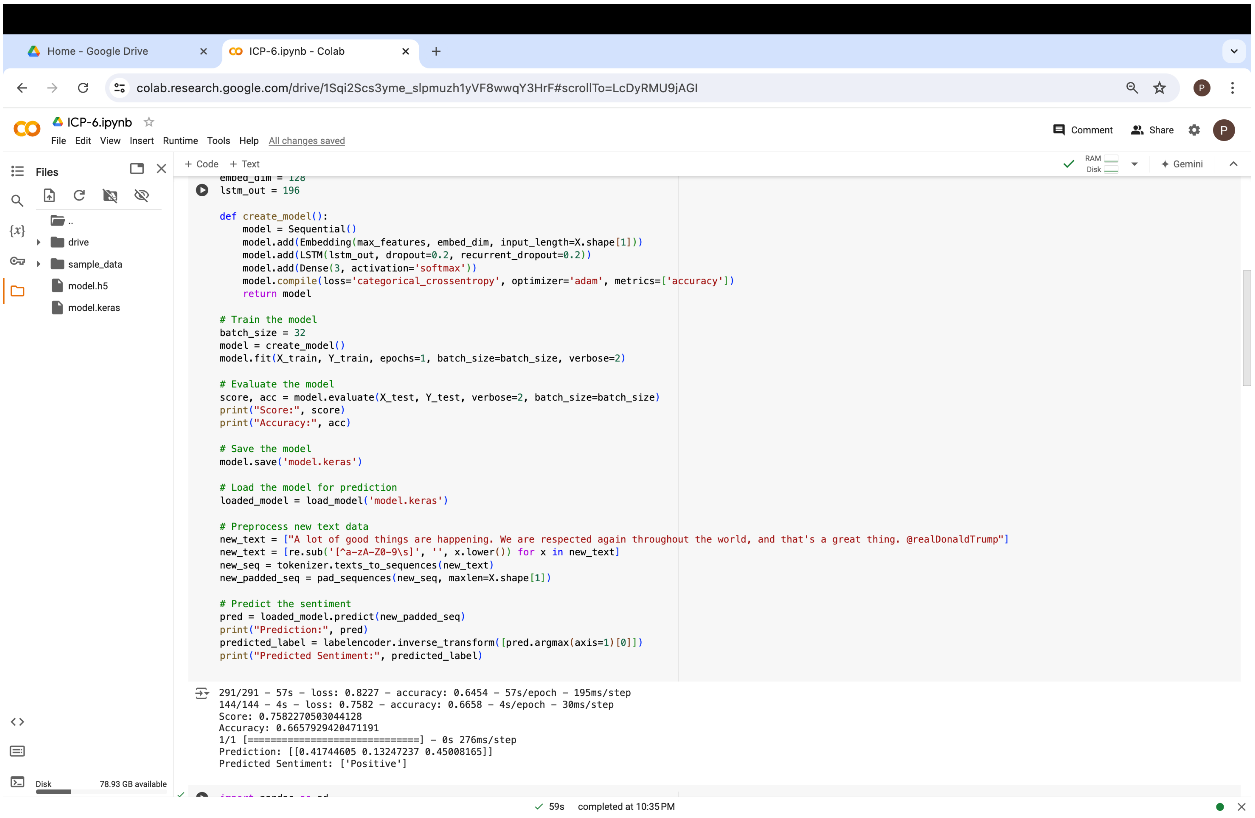
Video link: https://drive.google.com/file/d/1KxoX3mh3MuMg5vlrBuKptGD0_SlLyVTS/view?usp=drive_link

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")



Home - Google Drive
ICP-6.ipynb - Colab
colab.research.google.com/drive/1Sqi2Scs3yme_slpmuzh1yVF8wwqY3HrF#scrollTo=LcDyRMU9jAGI
ICP-6.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Gemini
Files
..
drive
sample_data
model.h5
model.keras
Disk 78.93 GB available
✓ 59s completed at 10:35 PM

```
[ ] import pandas as pd
    from keras.preprocessing.text import Tokenizer
    from keras.preprocessing.sequence import pad_sequences
    from keras.models import Sequential, load_model
    from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D
    from matplotlib import pyplot
    from sklearn.model_selection import train_test_split
    from keras.utils import to_categorical
    import re

    from sklearn.preprocessing import LabelEncoder

    # Load and preprocess data
    data = pd.read_csv('/content/drive/MyDrive/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', ' ')

    # Tokenize and pad sequences
    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)

    # Encode labels
    labelencoder = LabelEncoder()
    integer_encoded = labelencoder.fit_transform(data['sentiment'])
    y = to_categorical(integer_encoded)

    # Train-test split
    X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.33, random_state=42)

    # Define and compile the model
    embed_dim = 128
    lstm_out = 196
```



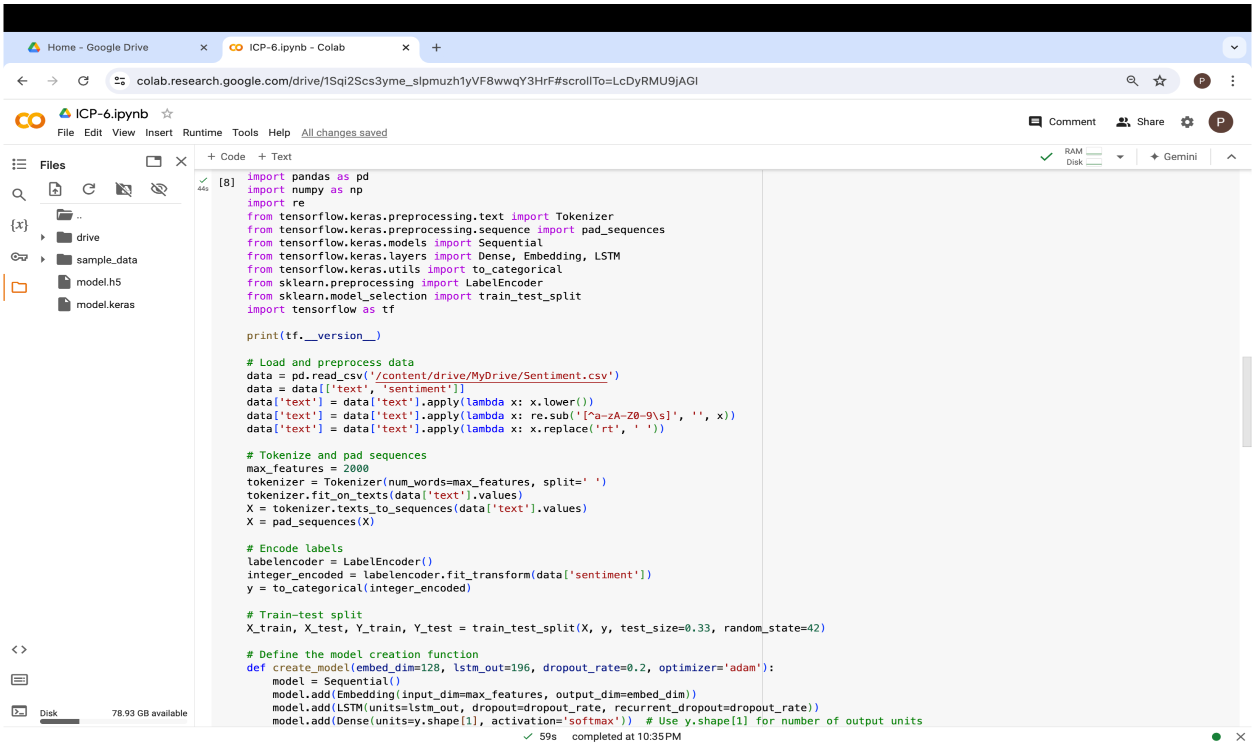
```
embed_dim = 128
lstm_out = 196

def create_model():
    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

# Train the model
batch_size = 32
model = create_model()
model.fit(X_train, Y_train, epochs=1, batch_size=batch_size, verbose=2)

# Evaluate the model
score, acc = model.evaluate(X_test, Y_test, verbose=2, batch_size=batch_size)
print("Score:", score)
print("Accuracy:", acc)

# Save the model
model.save('model.keras')

# Load the model for prediction
loaded_model = load_model('model.keras')

# Preprocess new text data
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 = [re.sub('[^a-zA-Z0-9\s]', '', x.lower()) for x in new_text]
new_seq = tokenizer.texts_to_sequences(new_text)
new_padded_seq = pad_sequences(new_seq, maxlen=X.shape[1])

# Predict the sentiment
pred = loaded_model.predict(new_padded_seq)
print("Prediction:", pred)
predicted_label = labelencoder.inverse_transform([pred.argmax(axis=1)[0]])
print("Predicted Sentiment:", predicted_label)
```

```
291/291 - 57s - loss: 0.8227 - accuracy: 0.6454 - 57s/epoch - 195ms/step
144/144 - 4s - loss: 0.7582 - accuracy: 0.6658 - 4s/epoch - 30ms/step
Score: 0.7582270503044128
Accuracy: 0.6657929420471191
1/1 [==============================] - 0s 276ms/step
Prediction: [[0.41744605 0.13247237 0.45008165]]
Predicted Sentiment: ['Positive']
```

```
import pandas as pd
```

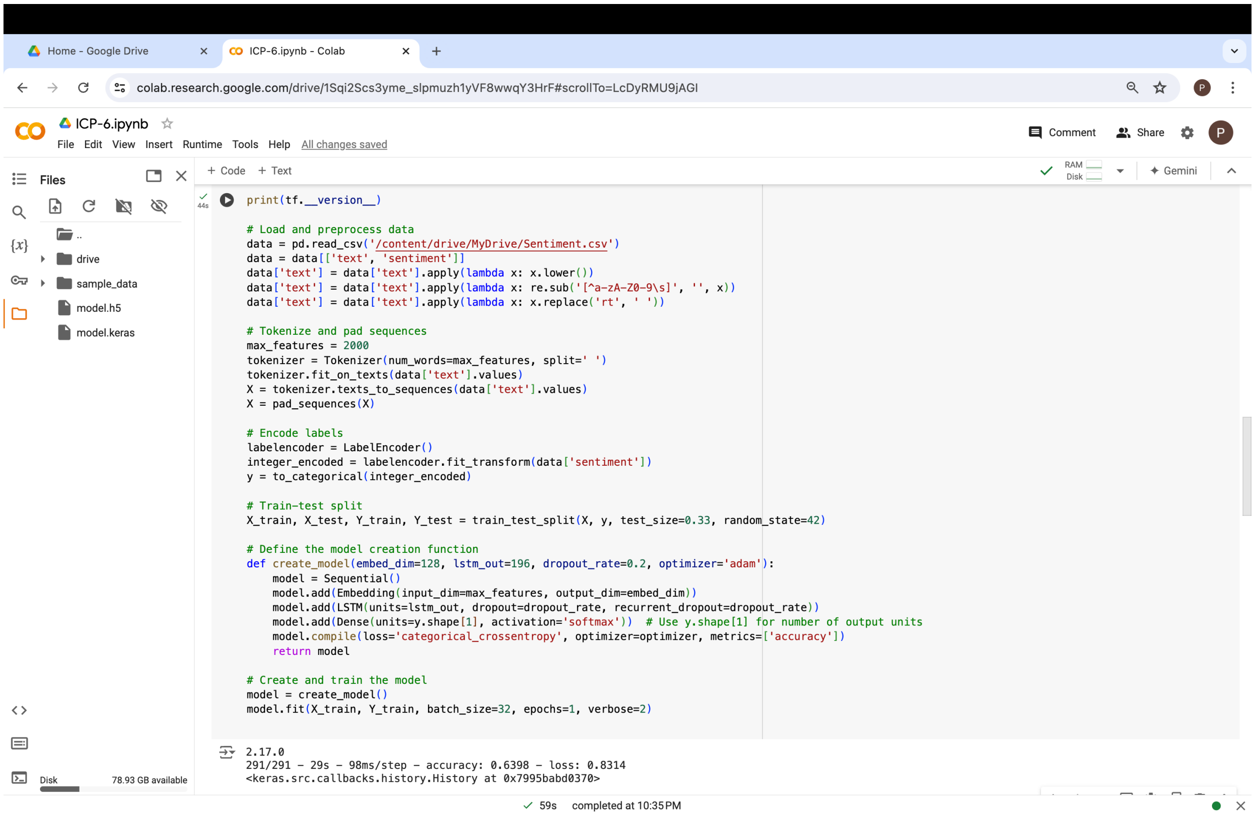
✓ 59s completed at 10:35 PM

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



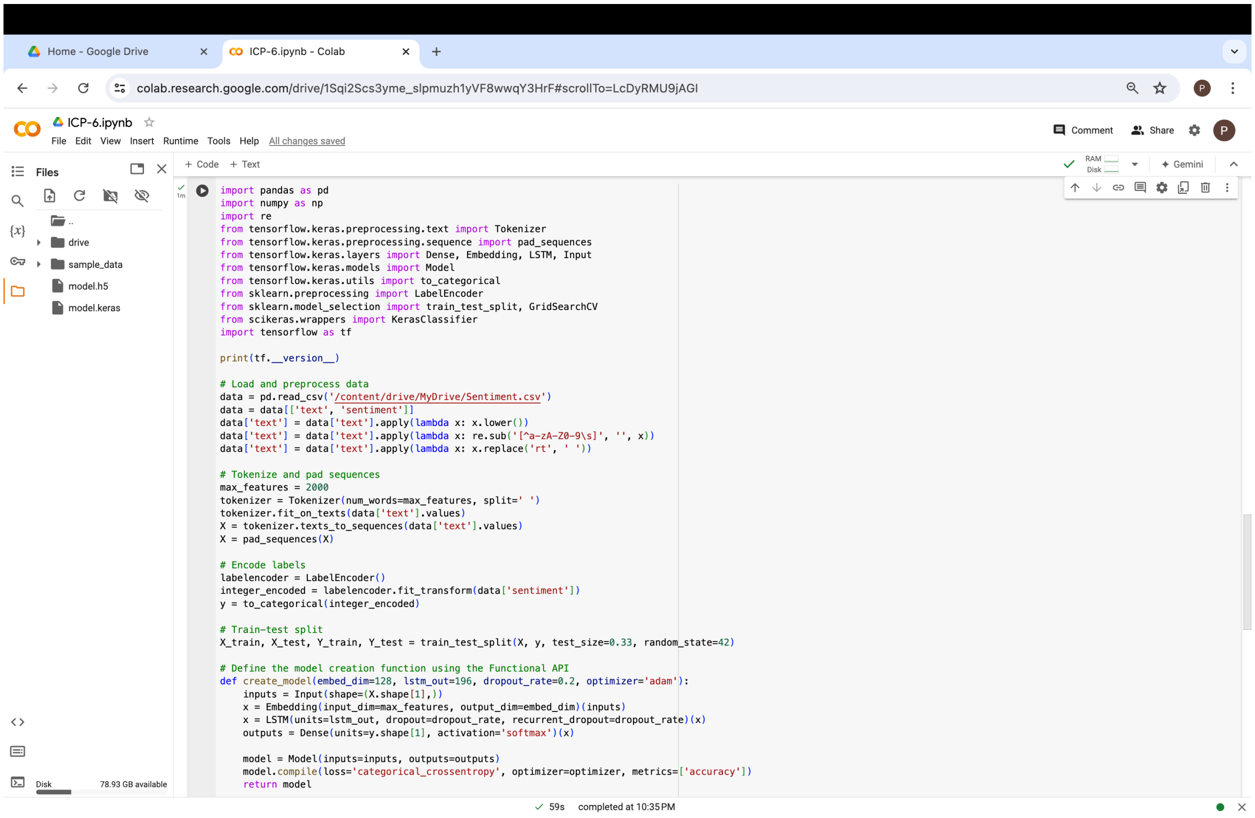
```
import pandas as pd
import numpy as np
import re
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Embedding, LSTM
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
import tensorflow as tf

print(tf.__version__)

# Load and preprocess data
data = pd.read_csv('/content/drive/MyDrive/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))
data['text'] = data['text'].apply(lambda x: x.replace('rt', ' '))

# Tokenize and pad sequences
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)

# Encode labels
labelencoder = LabelEncoder()
integer_encoded = labelencoder.fit_transform(data['sentiment'])
y = to_categorical(integer_encoded)

# Train-test split
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.33, random_state=42)

# Define the model creation function
def create_model(embed_dim=128, lstm_out=196, dropout_rate=0.2, optimizer='adam'):
    model = Sequential()
    model.add(Embedding(input_dim=max_features, output_dim=embed_dim))
    model.add(LSTM(units=lstm_out, dropout=dropout_rate, recurrent_dropout=dropout_rate))
    model.add(Dense(units=y.shape[1], activation='softmax'))  # Use y.shape[1] for number of output units
```

✓ 59s completed at 10:35 PM

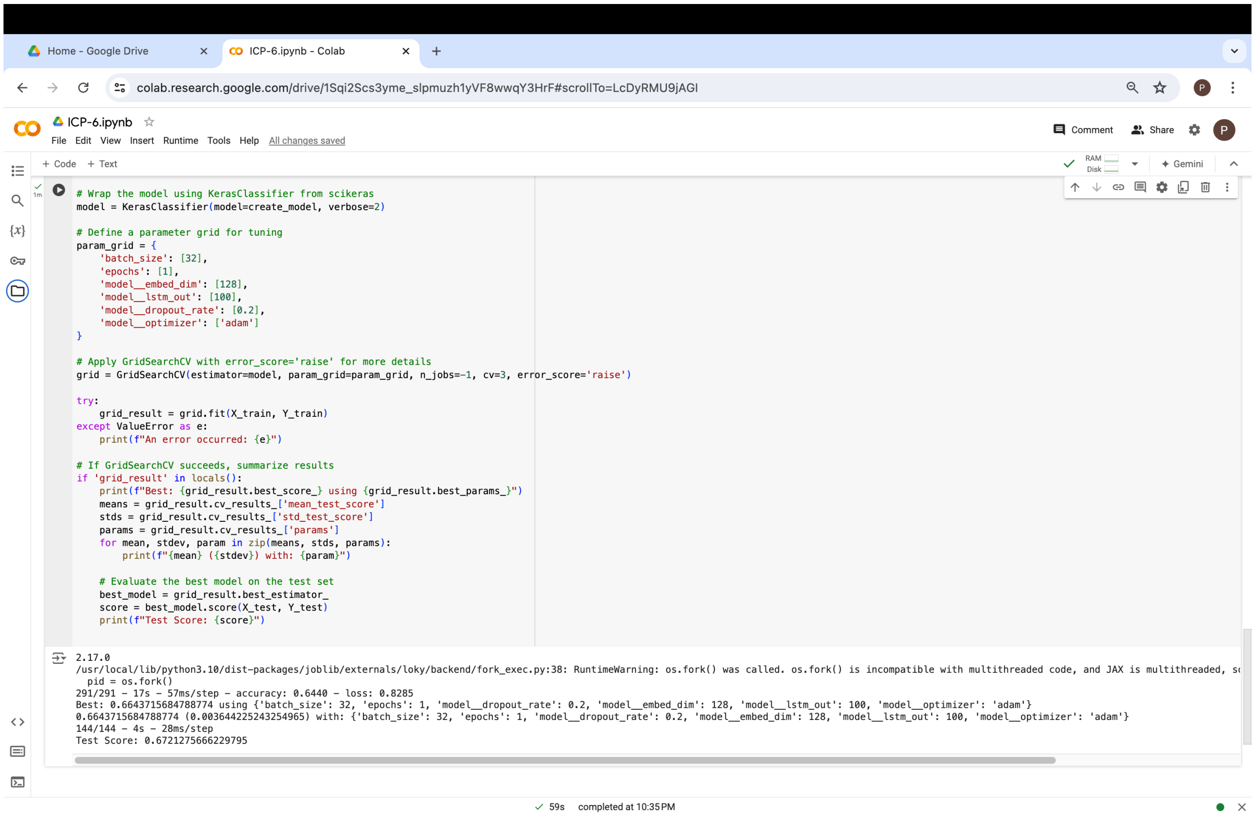

Home - Google Drive
ICP-6.ipynb - Colab
colab.research.google.com/drive/1Sqi2Scs3yme_slpmuzh1yVF8wwqY3HrF#scrollTo=LcDyRMU9jAGI
ICP-6.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
RAM
Disk
Gemini
..
drive
sample_data
model.h5
model.keras
Disk 78.93 GB available
44s
59s completed at 10:35 PM

```python
print(tf.__version__)

# Load and preprocess data
data = pd.read_csv('/content/drive/MyDrive/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))
data['text'] = data['text'].apply(lambda x: x.replace('rt', ' '))

# Tokenize and pad sequences
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)

# Encode labels
labelencoder = LabelEncoder()
integer_encoded = labelencoder.fit_transform(data['sentiment'])
y = to_categorical(integer_encoded)

# Train-test split
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.33, random_state=42)

# Define the model creation function
def create_model(embed_dim=128, lstm_out=196, dropout_rate=0.2, optimizer='adam'):
    model = Sequential()
    model.add(Embedding(input_dim=max_features, output_dim=embed_dim))
    model.add(LSTM(units=lstm_out, dropout=dropout_rate, recurrent_dropout=dropout_rate))
    model.add(Dense(units=y.shape[1], activation='softmax'))  # Use y.shape[1] for number of output units
    model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
    return model

# Create and train the model
model = create_model()
model.fit(X_train, Y_train, batch_size=32, epochs=1, verbose=2)
```

```
2.17.0
291/291 - 29s - 98ms/step - accuracy: 0.6398 - loss: 0.8314
<keras.src.callbacks.history.History at 0x7995babd0370>
```


Home - Google Drive
ICP-6.ipynb - Colab
colab.research.google.com/drive/1Sqi2Scs3yme_slpmuzh1yVF8wwqY3HrF#scrollTo=LcDyRMU9jAGI
ICP-6.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
RAM
Disk
Gemini
..
drive
sample_data
model.h5
model.keras
Disk 78.93 GB available
1m
59s completed at 10:35 PM

```python
import pandas as pd
import numpy as np
import re
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Dense, Embedding, LSTM, Input
from tensorflow.keras.models import Model
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split, GridSearchCV
from scikeras.wrappers import KerasClassifier
import tensorflow as tf

print(tf.__version__)

# Load and preprocess data
data = pd.read_csv('/content/drive/MyDrive/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))
data['text'] = data['text'].apply(lambda x: x.replace('rt', ' '))

# Tokenize and pad sequences
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)

# Encode labels
labelencoder = LabelEncoder()
integer_encoded = labelencoder.fit_transform(data['sentiment'])
y = to_categorical(integer_encoded)

# Train-test split
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.33, random_state=42)

# Define the model creation function using the Functional API
def create_model(embed_dim=128, lstm_out=196, dropout_rate=0.2, optimizer='adam'):
    inputs = Input(shape=(X.shape[1],))
    x = Embedding(input_dim=max_features, output_dim=embed_dim)(inputs)
    x = LSTM(units=lstm_out, dropout=dropout_rate, recurrent_dropout=dropout_rate)(x)
    outputs = Dense(units=y.shape[1], activation='softmax')(x)

    model = Model(inputs=inputs, outputs=outputs)
    model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
    return model
```


Home - Google Drive
ICP-6.ipynb - Colab
colab.research.google.com/drive/1Sqi2Scs3yme_slpmuzh1yVF8wwqY3HrF#scrollTo=LcDyRMU9jAGI
ICP-6.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Gemini

```python
# Wrap the model using KerasClassifier from scikeras
model = KerasClassifier(model=create_model, verbose=2)

# Define a parameter grid for tuning
param_grid = {
    'batch_size': [32],
    'epochs': [1],
    'model__embed_dim': [128],
    'model__lstm_out': [100],
    'model__dropout_rate': [0.2],
    'model__optimizer': ['adam']
}

# Apply GridSearchCV with error_score='raise' for more details
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1, cv=3, error_score='raise')

try:
    grid_result = grid.fit(X_train, Y_train)
except ValueError as e:
    print(f"An error occurred: {e}")

# If GridSearchCV succeeds, summarize results
if 'grid_result' in locals():
    print(f"Best: {grid_result.best_score_} using {grid_result.best_params_}")
    means = grid_result.cv_results_['mean_test_score']
    stds = grid_result.cv_results_['std_test_score']
    params = grid_result.cv_results_['params']
    for mean, stdev, param in zip(means, stds, params):
        print(f"{mean} ({stdev}) with: {param}")

    # Evaluate the best model on the test set
    best_model = grid_result.best_estimator_
    score = best_model.score(X_test, Y_test)
    print(f"Test Score: {score}")
```

```
2.17.0
/usr/local/lib/python3.10/dist-packages/joblib/externals/loky/backend/fork_exec.py:38: RuntimeWarning: os.fork() was called. os.fork() is incompatible with multithreaded code, and JAX is multithreaded, s
  pid = os.fork()
291/291 - 17s - 57ms/step - accuracy: 0.6440 - loss: 0.8285
Best: 0.6643715684788774 using {'batch_size': 32, 'epochs': 1, 'model__dropout_rate': 0.2, 'model__embed_dim': 128, 'model__lstm_out': 100, 'model__optimizer': 'adam'}
0.6643715684788774 (0.003644225243254965) with: {'batch_size': 32, 'epochs': 1, 'model__dropout_rate': 0.2, 'model__embed_dim': 128, 'model__lstm_out': 100, 'model__optimizer': 'adam'}
144/144 - 4s - 28ms/step
Test Score: 0.6721275666229795
```