```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout
from keras.callbacks import EarlyStopping

df = pd.read_csv('final.csv')

df = df.dropna()

X = df.iloc[:, :-1]
y = df.iloc[:, -1]

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

model = Sequential()
model.add(LSTM(units=64, input_shape=(X_train.shape[1], 1),
return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(units=64, return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(units=64, return_sequences=False))
model.add(Dropout(0.2))
model.add(Dense(units=1, activation='sigmoid'))

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

early_stop = EarlyStopping(monitor='val_loss', patience=5, verbose=1)

history = model.fit(np.expand_dims(X_train, axis=2), y_train,
validation_data=(np.expand_dims(X_test, axis=2), y_test),
batch_size=32, epochs=20, callbacks=[early_stop])
```

```
Epoch 1/20
7121/7121 [==============================] - 177s 24ms/step - loss:
0.0069 - accuracy: 0.9989 - val_loss: 0.0040 - val_accuracy: 0.9993
Epoch 2/20
7121/7121 [==============================] - 173s 24ms/step - loss:
0.0044 - accuracy: 0.9992 - val_loss: 0.0037 - val_accuracy: 0.9994
Epoch 3/20
7121/7121 [==============================] - 173s 24ms/step - loss:
0.0085 - accuracy: 0.9988 - val_loss: 0.0038 - val_accuracy: 0.9993
Epoch 4/20
7121/7121 [==============================] - 174s 24ms/step - loss:
0.0041 - accuracy: 0.9993 - val_loss: 0.0037 - val_accuracy: 0.9994
Epoch 5/20
7121/7121 [==============================] - 177s 25ms/step - loss:
```

```
0.0043 - accuracy: 0.9993 - val_loss: 0.0039 - val_accuracy: 0.9993
Epoch 6/20
7121/7121 [==============================] - 176s 25ms/step - loss:
0.0040 - accuracy: 0.9993 - val_loss: 0.0036 - val_accuracy: 0.9994
Epoch 7/20
7121/7121 [==============================] - 179s 25ms/step - loss:
0.0039 - accuracy: 0.9993 - val_loss: 0.0037 - val_accuracy: 0.9994
Epoch 8/20
7121/7121 [==============================] - 204s 29ms/step - loss:
0.0040 - accuracy: 0.9993 - val_loss: 0.0033 - val_accuracy: 0.9994
Epoch 9/20
7121/7121 [==============================] - 177s 25ms/step - loss:
0.0038 - accuracy: 0.9994 - val_loss: 0.0037 - val_accuracy: 0.9993
Epoch 10/20
7121/7121 [==============================] - 174s 24ms/step - loss:
0.0038 - accuracy: 0.9994 - val_loss: 0.0035 - val_accuracy: 0.9993
Epoch 11/20
7121/7121 [==============================] - 176s 25ms/step - loss:
0.0039 - accuracy: 0.9994 - val_loss: 0.0045 - val_accuracy: 0.9993
Epoch 12/20
7121/7121 [==============================] - 180s 25ms/step - loss:
0.0038 - accuracy: 0.9994 - val_loss: 0.0032 - val_accuracy: 0.9994
Epoch 13/20
7121/7121 [==============================] - 177s 25ms/step - loss:
0.0037 - accuracy: 0.9994 - val_loss: 0.0032 - val_accuracy: 0.9994
Epoch 14/20
7121/7121 [==============================] - 177s 25ms/step - loss:
0.0036 - accuracy: 0.9994 - val_loss: 0.0035 - val_accuracy: 0.9994
Epoch 15/20
7121/7121 [==============================] - 176s 25ms/step - loss:
0.0036 - accuracy: 0.9994 - val_loss: 0.0033 - val_accuracy: 0.9993
Epoch 16/20
7121/7121 [==============================] - 181s 25ms/step - loss:
0.0036 - accuracy: 0.9994 - val_loss: 0.0034 - val_accuracy: 0.9994
Epoch 17/20
7121/7121 [==============================] - 180s 25ms/step - loss:
0.0034 - accuracy: 0.9994 - val_loss: 0.0033 - val_accuracy: 0.9994
Epoch 18/20
7121/7121 [==============================] - 186s 26ms/step - loss:
0.0034 - accuracy: 0.9994 - val_loss: 0.0032 - val_accuracy: 0.9994
Epoch 19/20
7121/7121 [==============================] - 239s 34ms/step - loss:
0.0033 - accuracy: 0.9994 - val_loss: 0.0031 - val_accuracy: 0.9994
Epoch 20/20
7121/7121 [==============================] - 260s 37ms/step - loss:
0.0034 - accuracy: 0.9994 - val_loss: 0.0032 - val_accuracy: 0.9995

loss, accuracy = model.evaluate(np.expand_dims(X_test, axis=2),
y_test)
```

```python
print('Test Loss:', loss)
print('Test Accuracy:', accuracy)
```

```
1781/1781 [==============================] - 20s 11ms/step - loss:
0.0032 - accuracy: 0.9995
Test Loss: 0.0031691512558609247
Test Accuracy: 0.9994733333587646
```

```python
import pickle
# Dump the trained Naive Bayes classifier with Pickle
DT_pkl_filename = 'model.pkl'
# Open the file to save as pkl file
DT_Model_pkl = open(DT_pkl_filename, 'wb')
pickle.dump(model, DT_Model_pkl)
# Close the pickle instances
DT_Model_pkl.close()
```

```
WARNING:absl:Found untraced functions such as lstm_cell_layer_call_fn,
lstm_cell_layer_call_and_return_conditional_losses,
lstm_cell_1_layer_call_fn,
lstm_cell_1_layer_call_and_return_conditional_losses,
lstm_cell_2_layer_call_fn while saving (showing 5 of 6). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: ram://5ff685fb-3ca9-40b2-9183-
9b3271c99cd8/assets

INFO:tensorflow:Assets written to: ram://5ff685fb-3ca9-40b2-9183-
9b3271c99cd8/assets
WARNING:absl:<keras.layers.recurrent.LSTMCell object at
0x0000027C4C5ACC88> has the same name 'LSTMCell' as a built-in Keras
object. Consider renaming <class 'keras.layers.recurrent.LSTMCell'> to
avoid naming conflicts when loading with `tf.keras.models.load_model`.
If renaming is not possible, pass the object in the `custom_objects`
parameter of the load function.
WARNING:absl:<keras.layers.recurrent.LSTMCell object at
0x0000027C4C769400> has the same name 'LSTMCell' as a built-in Keras
object. Consider renaming <class 'keras.layers.recurrent.LSTMCell'> to
avoid naming conflicts when loading with `tf.keras.models.load_model`.
If renaming is not possible, pass the object in the `custom_objects`
parameter of the load function.
WARNING:absl:<keras.layers.recurrent.LSTMCell object at
0x0000027C4C7FC6A0> has the same name 'LSTMCell' as a built-in Keras
object. Consider renaming <class 'keras.layers.recurrent.LSTMCell'> to
avoid naming conflicts when loading with `tf.keras.models.load_model`.
If renaming is not possible, pass the object in the `custom_objects`
parameter of the load function.
```