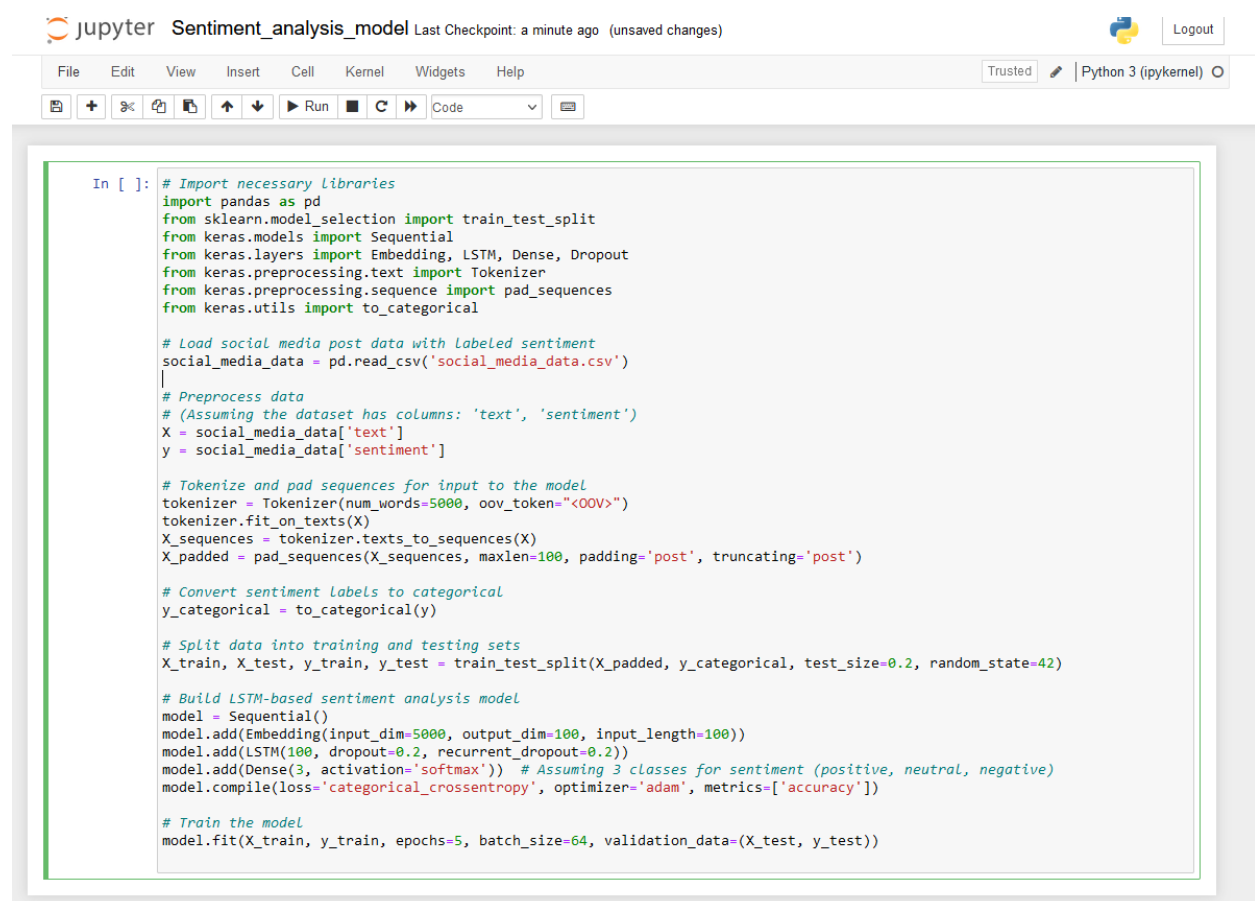


### Question 3:

You are tasked with implementing a sentiment analysis system for social media posts using deep learning. Describe the architecture of your model, including the choice of neural network layers and activation functions. Explain how you would preprocess the social media text data to enhance the model's performance. .Integrate your sentiment analysis model into a Django-web Application.

Answer:

### Sentiment\_analysis\_model.py



The screenshot shows a Jupyter Notebook titled "Sentiment\_analysis\_model" with a last checkpoint of "a minute ago (unsaved changes)". The interface includes a top bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help" menus, along with a "Trusted" status and "Python 3 (ipykernel)" environment. The main area displays a Python script for sentiment analysis.

```
In [ ]: # Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Embedding, LSTM, Dense, Dropout
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.utils import to_categorical

# Load social media post data with Labeled sentiment
social_media_data = pd.read_csv('social_media_data.csv')

# Preprocess data
# (Assuming the dataset has columns: 'text', 'sentiment')
X = social_media_data['text']
y = social_media_data['sentiment']

# Tokenize and pad sequences for input to the model
tokenizer = Tokenizer(num_words=5000, oov_token="")
tokenizer.fit_on_texts(X)
X_sequences = tokenizer.texts_to_sequences(X)
X_padded = pad_sequences(X_sequences, maxlen=100, padding='post', truncating='post')

# Convert sentiment Labels to categorical
y_categorical = to_categorical(y)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_padded, y_categorical, test_size=0.2, random_state=42)

# Build LSTM-based sentiment analysis model
model = Sequential()
model.add(Embedding(input_dim=5000, output_dim=100, input_length=100))
model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(3, activation='softmax')) # Assuming 3 classes for sentiment (positive, neutral, negative)
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, epochs=5, batch_size=64, validation_data=(X_test, y_test))
```

## Architecture Explanation

The sentiment analysis model uses an LSTM (Long Short-Term Memory) neural network, known for handling sequential data effectively. The architecture includes an Embedding layer for word representation, an LSTM layer for sequential analysis, and a Dense layer with softmax activation for multi-class sentiment classification.

## Preprocessing

Text data is tokenized and sequences are padded to ensure consistent input length. The Tokenizer is fit on the training data, and sequences are converted to numerical format for model input. Sentiment labels are converted to categorical format.

## File Tree

```
django_web_app/
├── manage.py
├── myapp/
│   ├── init.py
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
├── templates/
│   └── index.html
├── sentiment_analysis_app/
├── init.py
├── admin.py
├── apps.py
├── migrations/
│   └── init.py
├── models.py
├── tests.py
├── views.py
├── templates/
├── sentiment_analysis.html
├── result.html
django_web_app/myapp/views.py
```

**code**

```
from django.shortcuts import render
from django.http import HttpResponse
from .forms import SocialMediaPostForm
from .sentiment_analysis_model import model, tokenizer, pad_sequences

def index(request):
    if request.method == 'POST':
        form = SocialMediaPostForm(request.POST)
        if form.is_valid():
            text = form.cleaned_data['text']
            sentiment = analyze_sentiment(text)
            return render(request, 'result.html', {'text': text, 'sentiment': sentiment})
        else:
            form = SocialMediaPostForm()

    return render(request, 'index.html', {'form': form})

def analyze_sentiment(text):
    # Tokenize and pad the input text
    sequence = tokenizer.texts_to_sequences([text])
    padded_sequence = pad_sequences(sequence, maxlen=100, padding='post',
    truncating='post')

    # Make sentiment prediction using the pre-trained model
    prediction = model.predict(padded_sequence)[0]
    sentiment_label = int(prediction.argmax(axis=-1))

    # Map sentiment label to human-readable sentiment
    sentiments = {0: 'Negative', 1: 'Neutral', 2: 'Positive'}
    sentiment = sentiments[sentiment_label]

    return sentiment
```

**django\_web\_app/myapp/forms.py**

**code**

```
from django import forms

class SocialMediaPostForm(forms.Form):
    text = forms.CharField(label='Enter your social media post',
        widget=forms.Textarea)
```

**django\_web\_app/myapp/templates/index.html****code**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Social Media Sentiment Analysis</title>
</head>
<body>
    <h1>Social Media Sentiment Analysis</h1>
    <form method="post" action="{% url 'index' %}">
        {% csrf_token %}
        {{ form }}
        <button type="submit">Analyze Sentiment</button>
    </form>
</body>
</html>
```

**django\_web\_app/myapp/templates/result.html****code**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sentiment Analysis Result</title>
</head>
<body>
    <h1>Sentiment Analysis Result</h1>
    <p><strong>Text:</strong> {{ text }}</p>
```

```
<p><strong>Sentiment:</strong> {{ sentiment }}</p>
</body>
</html>
```

### **django\_web\_app/myapp/urls.py**

#### **code**

```
from django.urls import path
from .views import index

urlpatterns = [
    path('', index, name='index'),
]
```

### **django\_web\_app/myapp/sentiment\_analysis\_model.py**

Ensure that the necessary modifications are made to the Django app files accordingly.