```
# IMPORTANT: RUN THIS CELL IN ORDER TO IMPORT YOUR KAGGLE DATA SOURCES
# TO THE CORRECT LOCATION (/kaggle/input) IN YOUR NOTEBOOK,
# THEN FEEL FREE TO DELETE THIS CELL.
# NOTE: THIS NOTEBOOK ENVIRONMENT DIFFERS FROM KAGGLE'S PYTHON
# ENVIRONMENT SO THERE MAY BE MISSING LIBRARIES USED BY YOUR
# NOTEBOOK.

import os
import sys
from tempfile import NamedTemporaryFile
from urllib.request import urlopen
from urllib.parse import unquote, urlparse
from urllib.error import HTTPError
from zipfile import ZipFile
import tarfile
import shutil

CHUNK_SIZE = 40960
DATA_SOURCE_MAPPING = 'stock-market-prediction-and-sentimental-analysis:https%3A%2F%2Fstorage.googleapis.co

KAGGLE_INPUT_PATH='/kaggle/input'
KAGGLE_WORKING_PATH='/kaggle/working'
KAGGLE_SYMLINK='kaggle'

!umount /kaggle/input/ 2> /dev/null
shutil.rmtree('/kaggle/input', ignore_errors=True)
os.makedirs(KAGGLE_INPUT_PATH, 0o777, exist_ok=True)
os.makedirs(KAGGLE_WORKING_PATH, 0o777, exist_ok=True)

try:
  os.symlink(KAGGLE_INPUT_PATH, os.path.join("..", 'input'), target_is_directory=True)
except FileExistsError:
  pass
try:
  os.symlink(KAGGLE_WORKING_PATH, os.path.join("..", 'working'), target_is_directory=True)
except FileExistsError:
  pass

for data_source_mapping in DATA_SOURCE_MAPPING.split(','):
    directory, download_url_encoded = data_source_mapping.split(':')
    download_url = unquote(download_url_encoded)
    filename = urlparse(download_url).path
    destination_path = os.path.join(KAGGLE_INPUT_PATH, directory)
    try:
        with urlopen(download_url) as fileres, NamedTemporaryFile() as tfile:
            total_length = fileres.headers['content-length']
            print(f'Downloading {directory}, {total_length} bytes compressed')
            dl = 0
            data = fileres.read(CHUNK_SIZE)
            while len(data) > 0:
                dl += len(data)
                tfile.write(data)
                done = int(50 * dl / int(total_length))
                sys.stdout.write(f"\r[{'=' * done}{' ' * (50-done)}] {dl} bytes downloaded")
                sys.stdout.flush()
                data = fileres.read(CHUNK_SIZE)
            if filename.endswith('.zip'):
              with ZipFile(tfile) as zfile:
                zfile.extractall(destination_path)
            else:
              with tarfile.open(tfile.name) as tarfile:
                tarfile.extractall(destination_path)
            print(f'\nDownloaded and uncompressed: {directory}')
    except HTTPError as e:
        print(f'Failed to load (likely expired) {download_url} to path {destination_path}')
```

```
        continue
    except OSError as e:
        print(f'Failed to load {download_url} to path {destination_path}')
        continue

print('Data source import complete.')
```

```
Downloading stock-market-prediction-and-sentimental-analysis, 6474545 bytes compressed
[==================================================] 6474545 bytes downloaded
Downloaded and uncompressed: stock-market-prediction-and-sentimental-analysis
Data source import complete.
```

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout, Input
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.preprocessing.sequence import TimeseriesGenerator
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import nltk
nltk.download('vader_lexicon')

try:
    df = pd.read_csv('/kaggle/input/stock-market-prediction-and-sentimental-analysis/DJIA_table(train).csv')
    reddit_news = pd.read_csv('/kaggle/input/stock-market-prediction-and-sentimental-analysis/RedditNews(tra
except FileNotFoundError as e:
    print("File not found:", e)

try:
    df['Date'] = pd.to_datetime(df['Date'], dayfirst=True)
except KeyError as e:
    print("KeyError:", e)

merged_data = df.drop(columns=['Date']).dropna()

seq_length = 10
batch_size = 32

train_data, val_data = train_test_split(merged_data.to_numpy(), test_size=0.2, shuffle=False)

train_generator = TimeseriesGenerator(train_data, train_data[:, -1], length=seq_length, batch_size=batch_siz
val_generator = TimeseriesGenerator(val_data, val_data[:, -1], length=seq_length, batch_size=batch_size)

lstm_model = Sequential([
    Input(shape=(seq_length, merged_data.shape[1])),
    LSTM(128, return_sequences=True),
    Dropout(0.2),
    LSTM(64, return_sequences=True),
    Dropout(0.2),
    LSTM(32),
    Dense(1)
])
lstm_model.compile(optimizer=Adam(learning_rate=0.001), loss='mse')

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

history = lstm_model.fit(train_generator, epochs=50, validation_data=val_generator, callbacks=[early_stoppin

lstm_predictions = lstm_model.predict(val_generator)
```

```
scaler = MinMaxScaler()
scaler.fit(merged_data)

predicted_close_prices = lstm_predictions[:, -1]
inverse_transformed_predictions = np.hstack((np.zeros((lstm_predictions.shape[0], merged_data.shape[1] - 1))
lstm_predictions = scaler.inverse_transform(inverse_transformed_predictions)

mse = mean_squared_error(df[-len(predicted_close_prices):]['Close'], predicted_close_prices)
print("Model MSE:", mse)

submission_df = pd.DataFrame({
    'Id': range(1, len(predicted_close_prices) + 1),
    'Close': predicted_close_prices
})

submission_df.to_csv("submission.csv", index=False)
```

```
47/47 [==============================] - 1s 31ms/step - loss: 206599712.0000 - val_loss: 84605672.0
Epoch 24/50
47/47 [==============================] - 1s 31ms/step - loss: 206556608.0000 - val_loss: 84577512.0
Epoch 25/50
47/47 [==============================] - 1s 31ms/step - loss: 206513360.0000 - val_loss: 84549168.0
Epoch 26/50
47/47 [==============================] - 1s 31ms/step - loss: 206469504.0000 - val_loss: 84521400.0
Epoch 27/50
47/47 [==============================] - 2s 33ms/step - loss: 206426224.0000 - val_loss: 84493296.0
Epoch 28/50
47/47 [==============================] - 2s 47ms/step - loss: 206383008.0000 - val_loss: 84465864.0
Epoch 29/50
```