```
!pip install yfinance fredapi
```

```
Requirement already satisfied: yfinance in /usr/local/lib/python3.11/dist-packages (0.2.65)
Collecting fredapi
  Downloading fredapi-0.5.2-py3-none-any.whl.metadata (5.0 kB)
Requirement already satisfied: pandas>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from yfinance) (2.2.2)
Requirement already satisfied: numpy>=1.16.5 in /usr/local/lib/python3.11/dist-packages (from yfinance) (2.0.2)
Requirement already satisfied: requests>=2.31 in /usr/local/lib/python3.11/dist-packages (from yfinance) (2.32.3)
Requirement already satisfied: multitasking>=0.0.7 in /usr/local/lib/python3.11/dist-packages (from yfinance) (0.0.12)
Requirement already satisfied: platformdirs>=2.0.0 in /usr/local/lib/python3.11/dist-packages (from yfinance) (4.3.8)
Requirement already satisfied: pytz>=2022.5 in /usr/local/lib/python3.11/dist-packages (from yfinance) (2025.2)
Requirement already satisfied: frozendict>=2.3.4 in /usr/local/lib/python3.11/dist-packages (from yfinance) (2.4.6)
Requirement already satisfied: peewee>=3.16.2 in /usr/local/lib/python3.11/dist-packages (from yfinance) (3.18.2)
Requirement already satisfied: beautifulsoup4>=4.11.1 in /usr/local/lib/python3.11/dist-packages (from yfinance) (4.13.4)
Requirement already satisfied: curl_cffi>=0.7 in /usr/local/lib/python3.11/dist-packages (from yfinance) (0.12.0)
Requirement already satisfied: protobuf>=3.19.0 in /usr/local/lib/python3.11/dist-packages (from yfinance) (5.29.5)
Requirement already satisfied: websockets>=13.0 in /usr/local/lib/python3.11/dist-packages (from yfinance) (15.0.1)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.11/dist-packages (from beautifulsoup4>=4.11.1->yfinance) (2.7)
Requirement already satisfied: typing-extensions>=4.0.0 in /usr/local/lib/python3.11/dist-packages (from beautifulsoup4>=4.11.1->yfinanc
Requirement already satisfied: cffi>=1.12.0 in /usr/local/lib/python3.11/dist-packages (from curl_cffi>=0.7->yfinance) (1.17.1)
Requirement already satisfied: certifi>=2024.2.2 in /usr/local/lib/python3.11/dist-packages (from curl_cffi>=0.7->yfinance) (2025.7.14)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.3.0->yfinance) (2.9.0.p
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.3.0->yfinance) (2025.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests>=2.31->yfinance) (3.4.
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests>=2.31->yfinance) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests>=2.31->yfinance) (2.5.0)
Requirement already satisfied: pycparser in /usr/local/lib/python3.11/dist-packages (from cffi>=1.12.0->curl_cffi>=0.7->yfinance) (2.22)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas>=1.3.0->yfinance
Downloading fredapi-0.5.2-py3-none-any.whl (11 kB)
Installing collected packages: fredapi
Successfully installed fredapi-0.5.2
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import yfinance as yf
from keras.models import Sequential
from keras.layers import GRU, Dense
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error, mean_squared_error, r2_score
from fredapi import Fred
```

```
#my fred api key
fred = Fred(api_key='42706529e06d9cfdbe1eb6fb60c24235')
```

```
#Load WTI and Exogenous Variables

# Download WTI prices
df_wti = yf.download('CL=F', start='2010-01-01', end='2025-04-17')[['Close']]
df_wti.rename(columns={'Close': 'WTI'}, inplace=True)

# Download DXY (USD Index)
df_dxy = yf.download('DX-Y.NYB', start='2010-01-01', end='2025-04-17')[['Close']]
df_dxy.rename(columns={'Close': 'DXY'}, inplace=True)

# FRED economic indicators
date_range = pd.date_range(start='2010-01-01', end='2025-04-17')

df_fedfunds = fred.get_series('FEDFUNDS').reindex(date_range).to_frame(name='FEDFUNDS')
df_cpi      = fred.get_series('CPIAUCNS').reindex(date_range).to_frame(name='CPI')
df_indpro   = fred.get_series('INDPRO').reindex(date_range).to_frame(name='INDPRO')
df_all = df_wti.join([df_dxy, df_fedfunds, df_cpi, df_indpro])
df_all = df_all.fillna(method='ffill').dropna()
```

```
/tmp/ipython-input-5-1948028485.py:4: FutureWarning: YF.download() has changed argument auto_adjust default to True
  df_wti = yf.download('CL=F', start='2010-01-01', end='2025-04-17')[['Close']]
[*********************100%***********************]  1 of 1 completed
/tmp/ipython-input-5-1948028485.py:8: FutureWarning: YF.download() has changed argument auto_adjust default to True
  df_dxy = yf.download('DX-Y.NYB', start='2010-01-01', end='2025-04-17')[['Close']]
[*********************100%***********************]  1 of 1 completed
/tmp/ipython-input-5-1948028485.py:18: FutureWarning: DataFrame.fillna with 'method' is deprecated and will raise in a future version. U
  df_all = df_all.fillna(method='ffill').dropna()
```

```
# Combine all
df_all = df_wti.join([df_dxy, df_fedfunds, df_cpi, df_indpro])
df_all = df_all.ffill().dropna()  # updated to avoid warning

# Convert all columns to strings to avoid MinMaxScaler error
df_all.columns = df_all.columns.astype(str)

# Scale
scaler = MinMaxScaler()
scaled = scaler.fit_transform(df_all)


# Create sequences
def create_sequences(data, window):
    X, y = [], []
    for i in range(window, len(data)):
        X.append(data[i-window:i])
        y.append(data[i, 0])  # WTI is at column 0
    return np.array(X), np.array(y)

window_size = 60
X, y = create_sequences(scaled, window_size)


# Split into train/test
split = int(len(X) * 0.8)
X_train, X_test = X[:split], X[split:]
y_train, y_test = y[:split], y[split:]




model = Sequential()
model.add(GRU(units=50, return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(GRU(units=50))
model.add(Dense(1))

model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(X_train, y_train, epochs=20, batch_size=32)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass an `input_shape`/`input_dim` argument
  super().__init__(**kwargs)
Epoch 1/20
95/95 ━━━━━━━━━━━━━━━━━━━━ 13s 74ms/step - loss: 0.1023
Epoch 2/20
95/95 ━━━━━━━━━━━━━━━━━━━━ 10s 70ms/step - loss: 5.7485e-04
Epoch 3/20
95/95 ━━━━━━━━━━━━━━━━━━━━ 11s 78ms/step - loss: 4.3624e-04
Epoch 4/20
95/95 ━━━━━━━━━━━━━━━━━━━━ 10s 81ms/step - loss: 3.8722e-04
Epoch 5/20
95/95 ━━━━━━━━━━━━━━━━━━━━ 11s 83ms/step - loss: 4.7760e-04
Epoch 6/20
95/95 ━━━━━━━━━━━━━━━━━━━━ 10s 86ms/step - loss: 2.6281e-04
Epoch 7/20
95/95 ━━━━━━━━━━━━━━━━━━━━ 9s 70ms/step - loss: 2.8080e-04
Epoch 8/20
95/95 ━━━━━━━━━━━━━━━━━━━━ 11s 76ms/step - loss: 2.4837e-04
Epoch 9/20
95/95 ━━━━━━━━━━━━━━━━━━━━ 11s 82ms/step - loss: 1.9704e-04
Epoch 10/20
95/95 ━━━━━━━━━━━━━━━━━━━━ 10s 80ms/step - loss: 1.7817e-04
Epoch 11/20
95/95 ━━━━━━━━━━━━━━━━━━━━ 9s 70ms/step - loss: 1.3964e-04
Epoch 12/20
95/95 ━━━━━━━━━━━━━━━━━━━━ 10s 69ms/step - loss: 1.7419e-04
Epoch 13/20
95/95 ━━━━━━━━━━━━━━━━━━━━ 8s 84ms/step - loss: 1.5657e-04
Epoch 14/20
95/95 ━━━━━━━━━━━━━━━━━━━━ 9s 75ms/step - loss: 1.8723e-04
Epoch 15/20
95/95 ━━━━━━━━━━━━━━━━━━━━ 10s 69ms/step - loss: 1.3567e-04
Epoch 16/20
95/95 ━━━━━━━━━━━━━━━━━━━━ 8s 84ms/step - loss: 1.2618e-04
Epoch 17/20
95/95 ━━━━━━━━━━━━━━━━━━━━ 10s 81ms/step - loss: 1.6969e-04
Epoch 18/20
95/95 ━━━━━━━━━━━━━━━━━━━━ 9s 69ms/step - loss: 1.5000e-04
Epoch 19/20
95/95 ━━━━━━━━━━━━━━━━━━━━ 10s 70ms/step - loss: 1.7567e-04
Epoch 20/20
95/95 ━━━━━━━━━━━━━━━━━━━━ 10s 70ms/step - loss: 1.5430e-04
```

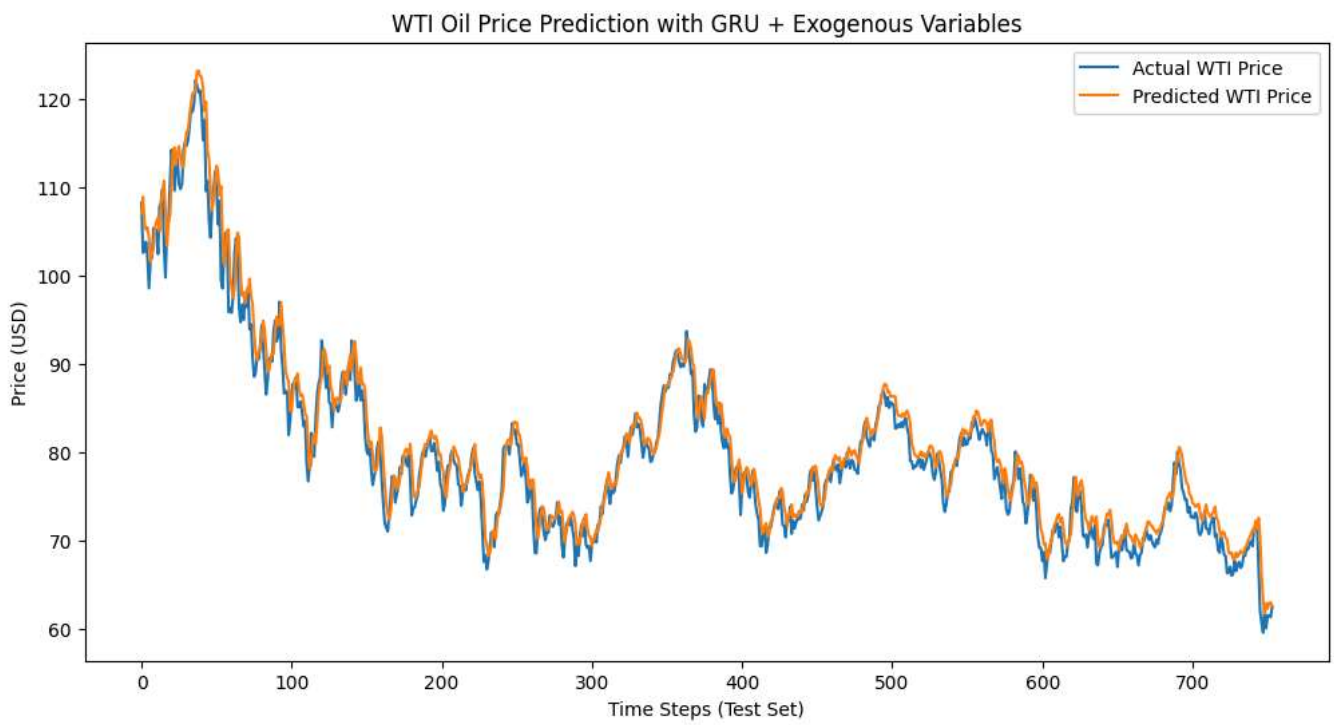
```
<keras.src.callbacks.history.History at 0x7964dd454550>
```

```
predictions = model.predict(X_test)
predicted_prices = scaler.inverse_transform(np.hstack([predictions, np.zeros((len(predictions), X.shape[2]-1))]))[:, 0]
actual_prices = scaler.inverse_transform(np.hstack([y_test.reshape(-1, 1), np.zeros((len(y_test), X.shape[2]-1))]))[:, 0]

# Combine into a DataFrame
results = pd.DataFrame({'Actual': actual_prices, 'Predicted': predicted_prices})
```

```
24/24 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step
```

```
plt.figure(figsize=(12,6))
plt.plot(results['Actual'], label='Actual WTI Price')
plt.plot(results['Predicted'], label='Predicted WTI Price')
plt.title('WTI Oil Price Prediction with GRU + Exogenous Variables')
plt.xlabel('Time Steps (Test Set)')
plt.ylabel('Price (USD)')
plt.legend()
plt.show()
```




```
mae = mean_absolute_error(results['Actual'], results['Predicted'])
mape = mean_absolute_percentage_error(results['Actual'], results['Predicted'])
rmse = np.sqrt(mean_squared_error(results['Actual'], results['Predicted']))
r2 = r2_score(results['Actual'], results['Predicted'])

print(f'MAE: {mae:.5f}')
print(f'MAPE: {mape*100:.2f}%')
print(f'RMSE: {rmse:.5f}')
print(f'R²: {r2:.5f}')
```

```
MAE: 1.75734
MAPE: 2.21%
RMSE: 2.32517
R²: 0.95647
```

```
df_dxy = yf.download('DX-Y.NYB', start='2010-01-01', end='2025-04-17')[['Close']]
df_dxy.rename(columns={'Close': 'DXY'}, inplace=True)
```

```
/tmp/ipython-input-13-2736900871.py:1: FutureWarning: YF.download() has changed argument auto_adjust default to True
  df_dxy = yf.download('DX-Y.NYB', start='2010-01-01', end='2025-04-17')[['Close']]
[*********************100%***********************]  1 of 1 completed
```

```
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
plt.plot(df_dxy.index, df_dxy['DXY'], color='green')
plt.title('U.S. Dollar Index (DXY) Over Time')
plt.xlabel('Date')
plt.ylabel('DXY Value')
plt.grid(True)
plt.show()
```


```
display(df_all.head())
```

Date	('WTI', 'CL=F')	('DXY', 'DX-Y.NYB')	FEDFUNDS	CPI	INDPRO
2010-02-01	74.430000	79.239998	0.13	216.741	89.5046
2010-02-02	77.230003	79.010002	0.13	216.741	89.5046
2010-02-03	76.980003	79.370003	0.13	216.741	89.5046
2010-02-04	73.139999	79.919998	0.13	216.741	89.5046
2010-02-05	71.190002	80.440002	0.13	216.741	89.5046