```
!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 yfinance as yf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from fredapi import Fred


# Initialize FRED API
fred = Fred(api_key='42706529e06d9cfdbe1eb6fb60c24235')

# Download Crude Oil (WTI) prices
df_oil = yf.download('CL=F', start='2010-01-01', end='2025-04-17')['Close']
df_oil = df_oil.rename('WTI')

# Download exogenous variables from FRED
variables = {
    'Dollar_Index': 'DTWEXBGS',  # Broad Dollar Index
    'FedFunds': 'FEDFUNDS',      # Federal Funds Rate
    'CPI': 'CPIAUCSL',           # CPI All Items
    'Industrial_Production': 'INDPRO'  # Industrial Production Index
}

df_exog = pd.DataFrame()
for name, code in variables.items():
    df_exog[name] = fred.get_series(code, observation_start='2010-01-01', observation_end='2025-04-17')

# Merge data and forward-fill missing values (if any)
df = pd.concat([df_oil, df_exog], axis=1).ffill().dropna()
```

```
/tmp/ipython-input-6-256756570.py:5: FutureWarning: YF.download() has changed argument auto_adjust default to True
  df_oil = yf.download('CL=F', start='2010-01-01', end='2025-04-17')['Close']
[*********************100%***********************]  1 of 1 completed
```

```
# Separate target (WTI) and exogenous variables
target = df['CL=F'].values.reshape(-1, 1) # Access using 'CL=F'
target_series = df['CL=F'] # Create a temporary Series to rename
target_series.name = 'WTI' # Rename the Series
target = target_series.values.reshape(-1, 1) # Convert back to numpy array

exog = df.drop(columns=['CL=F']).values # Drop 'CL=F'
```

```
# Scale features
scaler_target = StandardScaler()
scaler_exog = StandardScaler()

target_scaled = scaler_target.fit_transform(target)
exog_scaled = scaler_exog.fit_transform(exog)


# Define sequence length (60 days as before)
sequence_length = 60

X, y = [], []
for i in range(sequence_length, len(target_scaled)):
    X.append(target_scaled[i-sequence_length:i])
    y.append(target_scaled[i])

X = np.array(X)
y = np.array(y)

# Add exogenous variables to sequences
X_exog = []
for i in range(sequence_length, len(exog_scaled)):
    X_exog.append(exog_scaled[i-sequence_length:i])

X_exog = np.array(X_exog)

# Combine target and exogenous variables
X_combined = np.concatenate([X, X_exog], axis=2)


split = int(0.8 * len(X_combined))
X_train, X_test = X_combined[:split], X_combined[split:]
y_train, y_test = y[:split], y[split:]

# Further split for validation
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2, shuffle=False)


model = Sequential([
    LSTM(64, return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2])),
    LSTM(64, return_sequences=False),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(1)
])

model.compile(optimizer='adam', loss='mae')
model.summary()
```

```
/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)
```

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 60, 64)	17,920
lstm_1 (LSTM)	(None, 64)	33,024
dense (Dense)	(None, 128)	8,320
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 1)	129

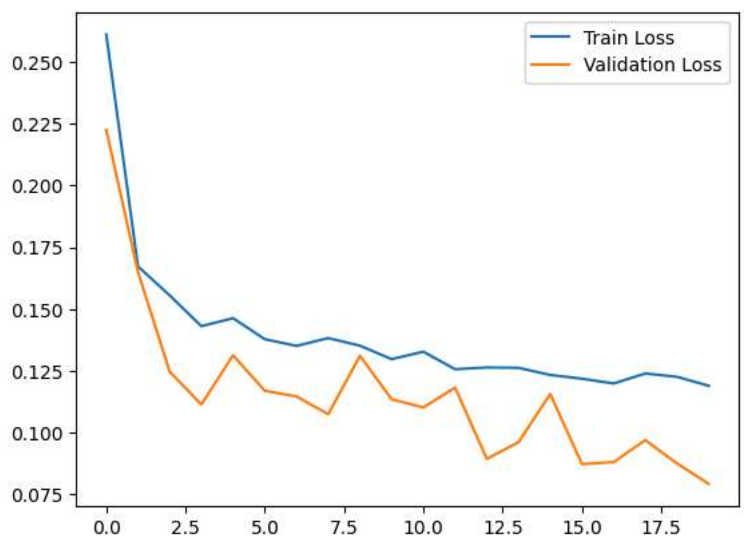
Total params: 59,393 (232.00 KB)
Trainable params: 59,393 (232.00 KB)
Non-trainable params: 0 (0.00 B)

```
history = model.fit(
    X_train, y_train,
    epochs=20,
    batch_size=32,
    validation_data=(X_val, y_val)
)
```

```
Epoch 1/20
79/79 ━━━━━━━━━━━━━━━━━━━━ 13s 108ms/step - loss: 0.3849 - val_loss: 0.2225
```

```
Epoch 2/20
79/79 ━━━━━━━━━━━━━━━━━━━━ 10s 112ms/step - loss: 0.1713 - val_loss: 0.1651
Epoch 3/20
79/79 ━━━━━━━━━━━━━━━━━━━━ 8s 84ms/step - loss: 0.1551 - val_loss: 0.1247
Epoch 4/20
79/79 ━━━━━━━━━━━━━━━━━━━━ 10s 78ms/step - loss: 0.1446 - val_loss: 0.1115
Epoch 5/20
79/79 ━━━━━━━━━━━━━━━━━━━━ 5s 68ms/step - loss: 0.1541 - val_loss: 0.1314
Epoch 6/20
79/79 ━━━━━━━━━━━━━━━━━━━━ 10s 71ms/step - loss: 0.1408 - val_loss: 0.1170
Epoch 7/20
79/79 ━━━━━━━━━━━━━━━━━━━━ 11s 85ms/step - loss: 0.1325 - val_loss: 0.1147
Epoch 8/20
79/79 ━━━━━━━━━━━━━━━━━━━━ 9s 74ms/step - loss: 0.1383 - val_loss: 0.1075
Epoch 9/20
79/79 ━━━━━━━━━━━━━━━━━━━━ 11s 83ms/step - loss: 0.1404 - val_loss: 0.1312
Epoch 10/20
79/79 ━━━━━━━━━━━━━━━━━━━━ 11s 87ms/step - loss: 0.1291 - val_loss: 0.1135
Epoch 11/20
79/79 ━━━━━━━━━━━━━━━━━━━━ 6s 73ms/step - loss: 0.1381 - val_loss: 0.1102
Epoch 12/20
79/79 ━━━━━━━━━━━━━━━━━━━━ 11s 80ms/step - loss: 0.1251 - val_loss: 0.1183
Epoch 13/20
79/79 ━━━━━━━━━━━━━━━━━━━━ 11s 95ms/step - loss: 0.1262 - val_loss: 0.0894
Epoch 14/20
79/79 ━━━━━━━━━━━━━━━━━━━━ 8s 72ms/step - loss: 0.1278 - val_loss: 0.0962
Epoch 15/20
79/79 ━━━━━━━━━━━━━━━━━━━━ 10s 70ms/step - loss: 0.1233 - val_loss: 0.1157
Epoch 16/20
79/79 ━━━━━━━━━━━━━━━━━━━━ 7s 85ms/step - loss: 0.1217 - val_loss: 0.0873
Epoch 17/20
79/79 ━━━━━━━━━━━━━━━━━━━━ 6s 71ms/step - loss: 0.1161 - val_loss: 0.0882
Epoch 18/20
79/79 ━━━━━━━━━━━━━━━━━━━━ 6s 82ms/step - loss: 0.1220 - val_loss: 0.0970
Epoch 19/20
79/79 ━━━━━━━━━━━━━━━━━━━━ 9s 68ms/step - loss: 0.1224 - val_loss: 0.0876
Epoch 20/20
79/79 ━━━━━━━━━━━━━━━━━━━━ 11s 73ms/step - loss: 0.1170 - val_loss: 0.0793
```

```
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.show()
```




```
predictions = model.predict(X_test)
predictions = scaler_target.inverse_transform(predictions)
actual = scaler_target.inverse_transform(y_test)
```

```
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 20ms/step
```

```
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error, mean_squared_error, r2_score
# Calculate metrics
mae = mean_absolute_error(actual, predictions)
mape = mean_absolute_percentage_error(actual, predictions)
rmse = np.sqrt(mean_squared_error(actual, predictions))
```

```
r2 = r2_score(actual, predictions)

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

```
MAE: 2.08996
MAPE: 2.63021%
RMSE: 2.67774
R²: 0.94322
```

```
# Inverse transform the test set actual values
y_test_actual = scaler_target.inverse_transform(y_test)

# Create a DataFrame for plotting
test_dates = df.index[split + sequence_length:]  # Align dates with the test set
results = pd.DataFrame({
    'Date': test_dates,
    'Actual': y_test_actual.flatten(),
    'Predicted': predictions.flatten()
}).set_index('Date')

# Plot
plt.figure(figsize=(12, 6))
plt.plot(results.index, results['Actual'], label='Test (Actual)', color='orange', linewidth=2)
plt.plot(results.index, results['Predicted'], label='Predictions', color='red', linewidth=2)
plt.title('Crude Oil (WTI) - Test Actuals vs Predictions (With Exogenous Variables)', fontsize=14)
plt.xlabel('Date', fontsize=12)
plt.ylabel('Close Price (USD)', fontsize=12)
plt.legend(fontsize=12)
plt.grid(True, alpha=0.7)
plt.show()
```