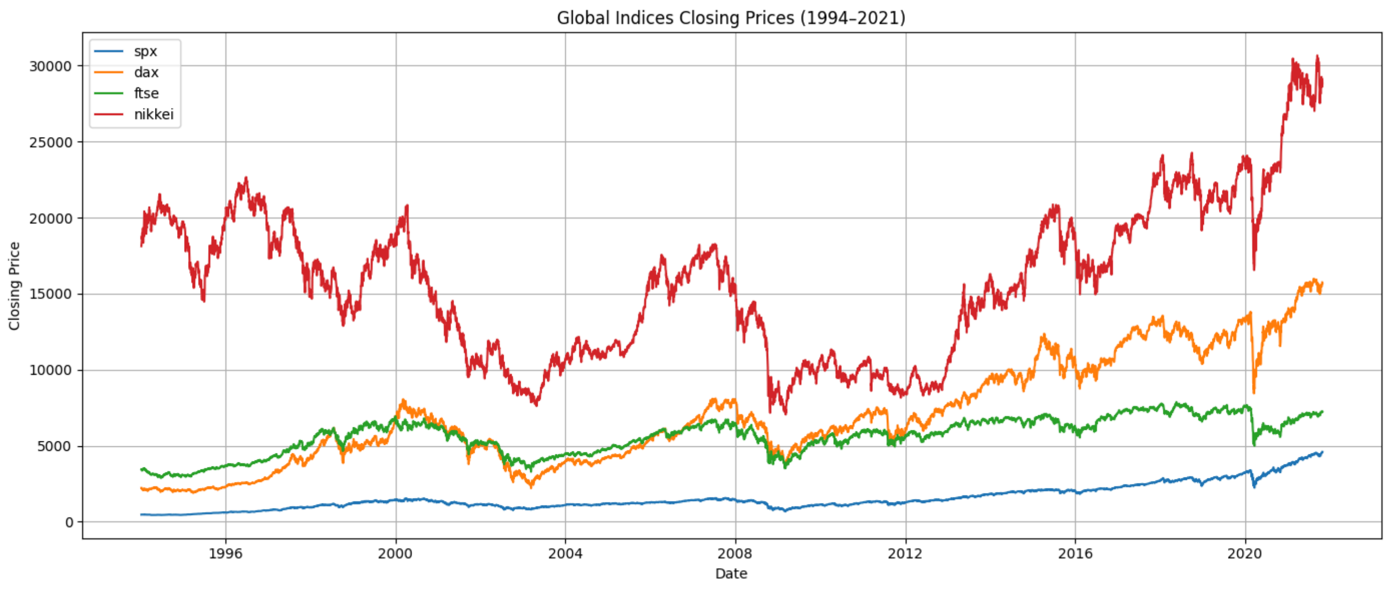
```
from google.colab import files

uploaded = files.upload()   # 打开文件上传对话框
```

```
选择文件 Index closing ...4 to 2021.csv
• Index closing price from 1994 to 2021.csv(text/csv) - 530225 bytes, last modified: 2025/3/31 - 100% done
Saving Index closing price from 1994 to 2021.csv to Index closing price from 1994 to 2021.csv
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error


# === Load and preprocess the data ===
df = pd.read_csv("Index closing price from 1994 to 2021.csv")
df['Date'] = pd.to_datetime(df['Date'])
df.sort_values('Date', inplace=True)
df.set_index('Date', inplace=True)


# === Visualization ===
plt.figure(figsize=(14, 6))
for col in ['spx', 'dax', 'ftse', 'nikkei']:
        plt.plot(df.index, df[col], label=col)
plt.title("Global Indices Closing Prices (1994-2021)")
plt.xlabel("Date")
plt.ylabel("Closing Price")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

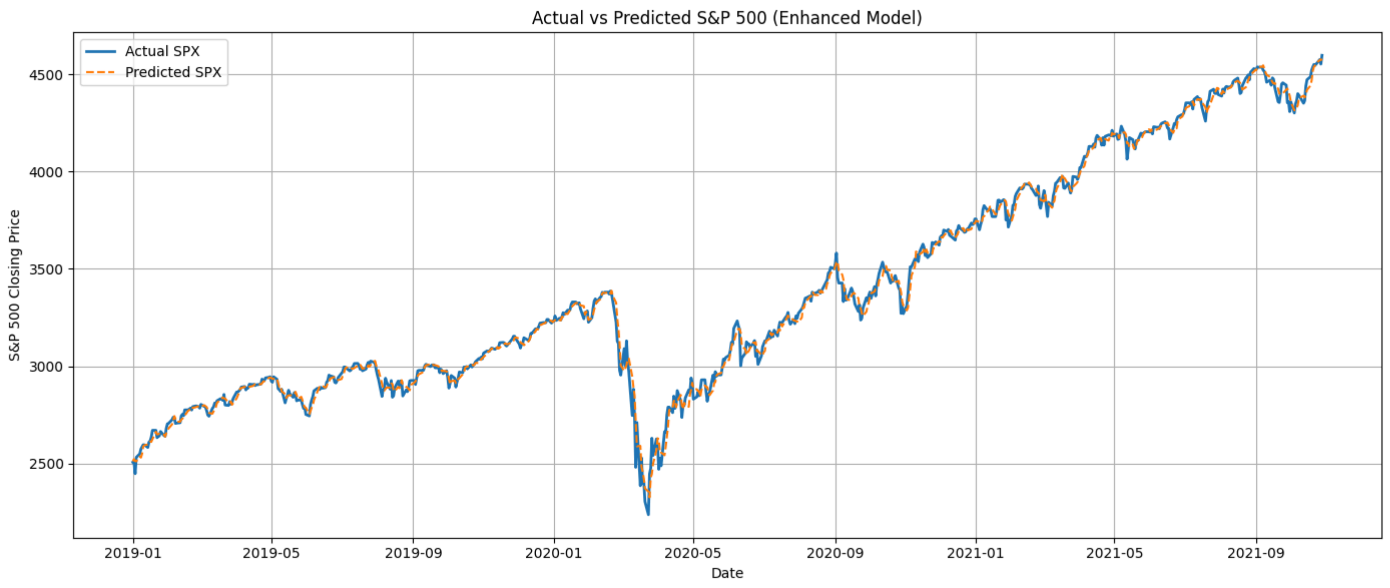


```
# === Add lag features (1-day and 2-day lags) for global indices ===
for col in ['dax', 'ftse', 'nikkei']:
        df[f'{col}_lag1'] = df[col].shift(1)
        df[f'{col}_lag2'] = df[col].shift(2)


# === Add technical indicators for SPX ===
df['spx_ma5'] = df['spx'].rolling(window=5).mean()
df['spx_ma10'] = df['spx'].rolling(window=10).mean()
```

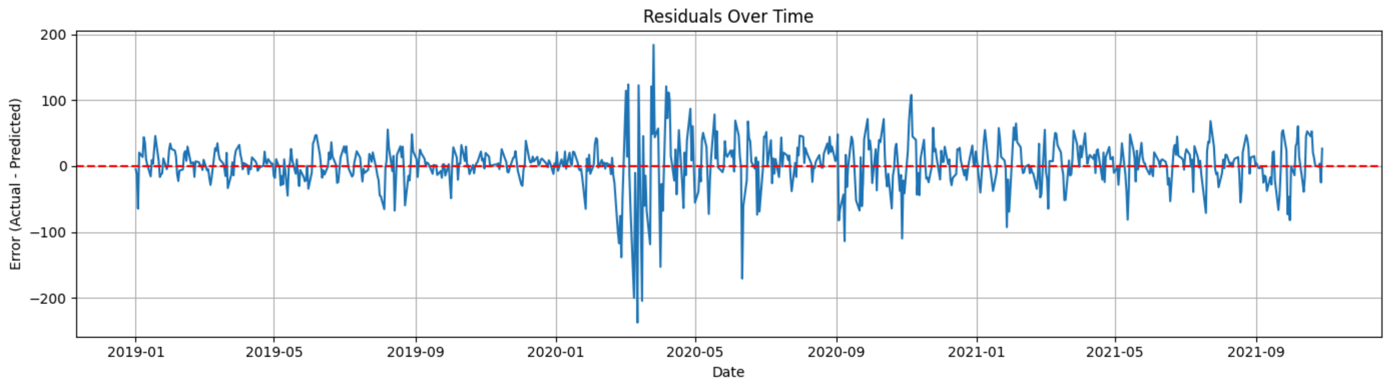
```
df['spx_ma20'] = df['spx'].rolling(window=20).mean()
df['spx_std20'] = df['spx'].rolling(window=20).std()
df['spx_upper'] = df['spx_ma20'] + 2 * df['spx_std20']
df['spx_lower'] = df['spx_ma20'] - 2 * df['spx_std20']


# === Drop rows with NaN values caused by lag or rolling operations ===
df.dropna(inplace=True)


# === Select features and target ===
features = [
        'dax_lag1', 'dax_lag2',
        'ftse_lag1', 'ftse_lag2',
        'nikkei_lag1', 'nikkei_lag2',
        'spx_ma5', 'spx_ma10',
        'spx_upper', 'spx_lower'
]
target = 'spx'

X = df[features]
y = df[target]


# === Train-test split (before and after 2019) ===
X_train = X[X.index < '2019-01-01']
X_test = X[X.index >= '2019-01-01']
y_train = y[y.index < '2019-01-01']
y_test = y[y.index >= '2019-01-01']


# === Train linear regression model ===
model = LinearRegression()
model.fit(X_train, y_train)
predictions = model.predict(X_test)


# === Evaluate model performance ===
mae = mean_absolute_error(y_test, predictions)
rmse = np.sqrt(mean_squared_error(y_test, predictions))
print(f"MAE: {mae:.2f}")
print(f"RMSE: {rmse:.2f}")
```

```
MAE: 24.29
RMSE: 36.57
```

```
# === Plot actual vs predicted SPX ===
plt.figure(figsize=(14,6))
plt.plot(y_test.index, y_test, label='Actual SPX', linewidth=2)
plt.plot(y_test.index, predictions, label='Predicted SPX', linestyle='--')
plt.title('Actual vs Predicted S&P 500 (Enhanced Model)')
plt.xlabel('Date')
plt.ylabel('S&P 500 Closing Price')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



```
# === Plot residuals over time ===
residuals = y_test - predictions
plt.figure(figsize=(14,4))
plt.plot(y_test.index, residuals)
plt.axhline(0, color='red', linestyle='--')
plt.title('Residuals Over Time')
plt.xlabel('Date')
plt.ylabel('Error (Actual - Predicted)')
plt.grid(True)
plt.tight_layout()
plt.show()
```



```
# === Save first 4 predictions as sample output ===
sample_output = y_test.to_frame(name='actual_spx').copy()
sample_output['predicted_spx'] = predictions
csv_sample = sample_output.head(4).reset_index().rename(columns={'Date': 'date'})
print(csv_sample)
```

```
         date   actual_spx  predicted_spx
0 2019-01-01  2506.850098    2511.675453
1 2019-01-02  2510.030029    2522.812535
2 2019-01-03  2447.889893    2512.448853
```

```
3 2019-01-04  2531.939941    2511.433545
```