Introduction to time series

ANOMALY DETECTION IN PYTHON

Bekhruz (Bex) Tuychiev
Kaggle Master, Data Science Content Creator

Google stocks dataset

```python
import pandas as pd

google = pd.read_csv("google.csv")
google.head()
```

```
              Open    High     Low   Close    Volume
Date
2006-01-03  211.47  218.05  209.32  217.83  13137450
2006-01-04  222.17  224.70  220.09  222.84  15292353
2006-01-05  223.22  226.00  220.97  225.85  10815661
2006-01-06  228.66  235.49  226.85  233.06  17759521
2006-01-09  233.44  236.94  230.70  233.68  12795837
```

DateTime datatype

```
print(google['Date'].dtype)
```

```
object
```

```
google['Date'] = pd.to_datetime(google['Date'])

print(google.dtypes)
```

```
Date      datetime64[ns]
Open             float64
High             float64
...
```

Extracting features

```
google['day_of_week'] = google['Date'].dt.day_of_week
google['day_of_month'] = google['Date'].dt.day
google['month'] = google['Date'].dt.month

google.sample(5)
```

Date	Open	Low	Close	Volume	day_of_week	month	day_of_month
2016-02-29	721.00	716.84	717.22	2237474	0	2	29
2007-01-24	242.46	241.89	249.78	6074077	2	1	24
2007-05-24	237.81	235.99	237.40	4200474	3	5	24
2008-09-16	213.19	212.96	221.69	6991767	1	9	16
2008-03-31	218.04	216.22	220.46	4446368	0	3	31

DatetimeIndex

```
google.set_index("Date", inplace=True)


google.head()
```

```
              Open    High     Low   Close    Volume  day_of_week  month  \
Date
2006-01-03  211.47  218.05  209.32  217.83  13137450            1      1
2006-01-04  222.17  224.70  220.09  222.84  15292353            2      1
2006-01-05  223.22  226.00  220.97  225.85  10815661            3      1
2006-01-06  228.66  235.49  226.85  233.06  17759521            4      1
2006-01-09  233.44  236.94  230.70  233.68  12795837            0      1
```

Choosing periods

```
google["2008": "2010"].head()
```

```
              Open    High     Low   Close   Volume  day_of_week  month  \
Date
2008-01-02  346.78  349.03  339.20  342.94  4306848            2      1
2008-01-03  342.97  343.77  338.60  343.01  3252846            3      1
2008-01-04  340.18  340.82  327.83  328.83  5359834            4      1
2008-01-07  327.30  331.47  318.99  324.95  6404945            0      1
2008-01-08  326.83  330.31  315.82  316.16  5341949            1      1
```

Choosing periods

```
google["2012-03": "2015-10-04"].head()
```

```
              Open    High     Low   Close   Volume  day_of_week  month  \
Date
2012-03-01  311.44  313.16  309.38  311.51  2238010            3      3
2012-03-02  311.31  312.31  310.47  310.94  1573214            4      3
2012-03-05  310.53  311.56  306.00  307.43  1593250            0      3
2012-03-06  304.33  304.71  297.22  302.78  3175216            1      3
2012-03-07  304.83  305.90  303.23  303.70  1264892            2      3
```

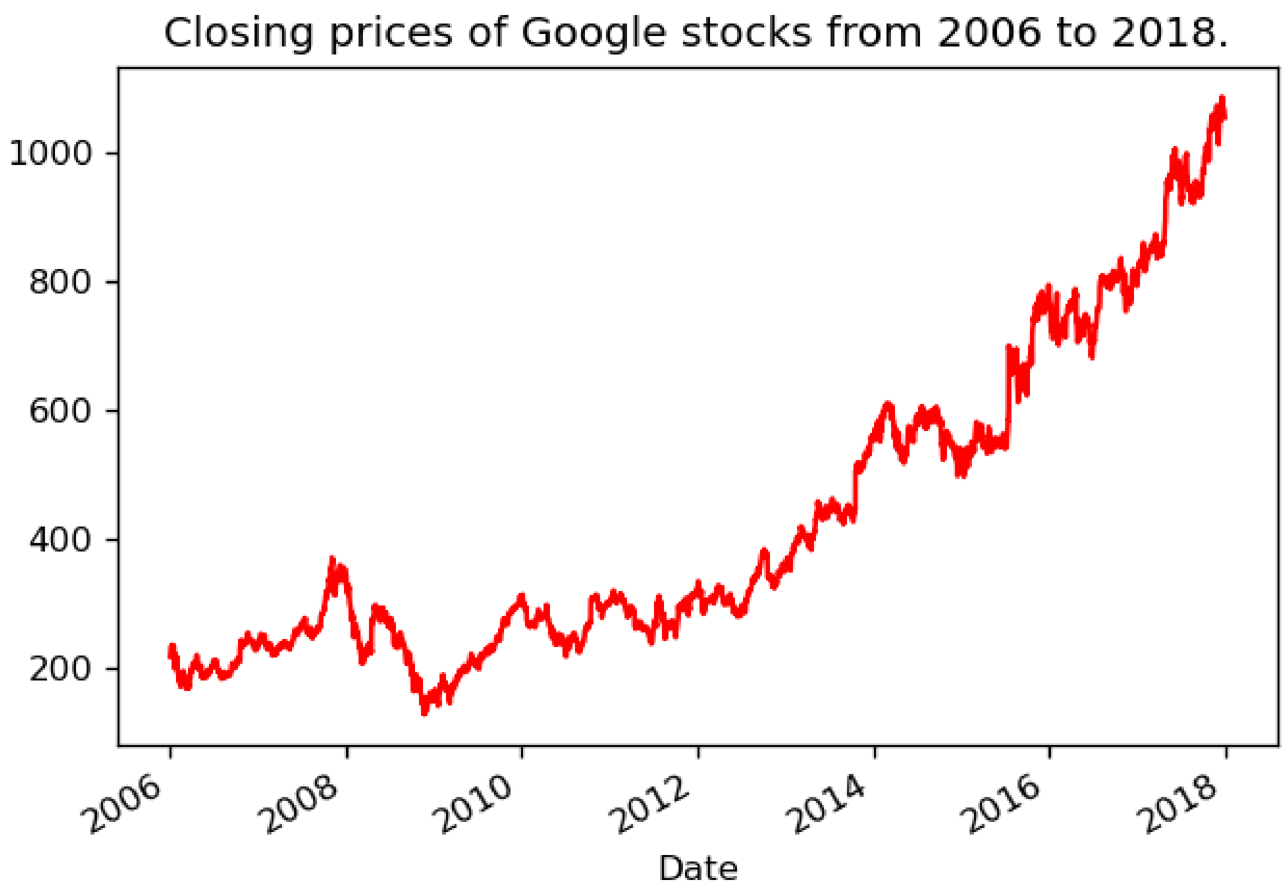
Loading datasets with a DatetimeIndex

```
google = pd.read_csv("google.csv", parse_dates=["Date"], index_col="Date")

google.head()
```

```
              Open    High     Low   Close    Volume
Date
2006-01-03  211.47  218.05  209.32  217.83  13137450
2006-01-04  222.17  224.70  220.09  222.84  15292353
2006-01-05  223.22  226.00  220.97  225.85  10815661
2006-01-06  228.66  235.49  226.85  233.06  17759521
2006-01-09  233.44  236.94  230.70  233.68  12795837
```

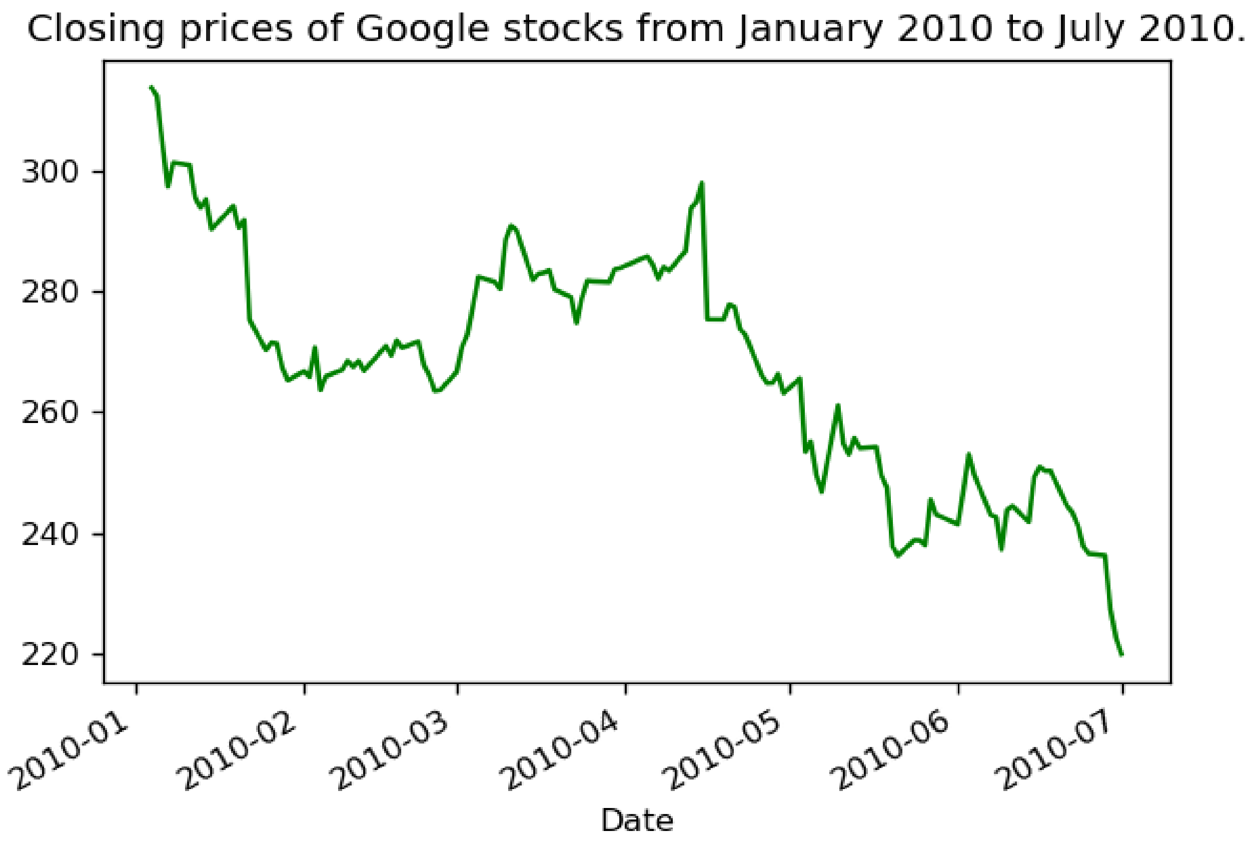
Plotting time series

```
import matplotlib.pyplot as plt

google["Close"].plot(color="red")

plt.title("""Closing prices of Google
stocks from 2006 to 2018.""")
plt.show()
```



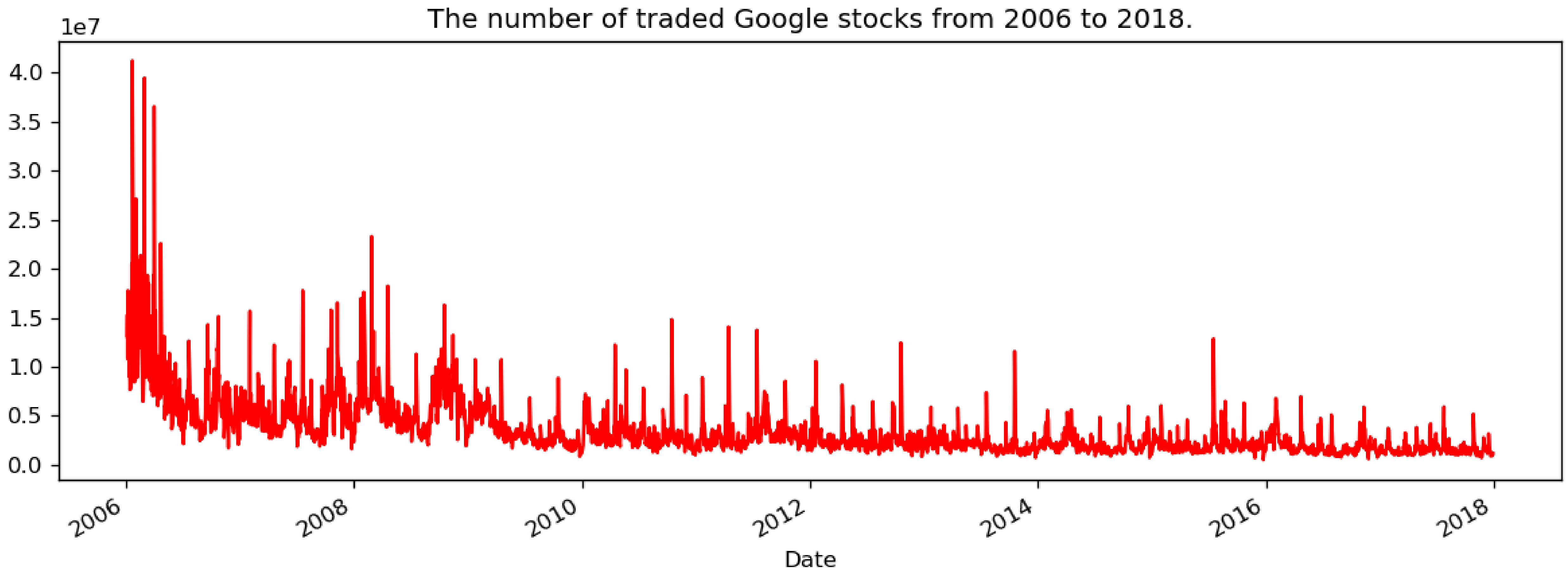
Plotting time series

```
google["2010": "2010-07-01"]['Close'].plot(color="green")

plt.title("Closing prices of Google stocks from January 2010 to July 2010.")
plt.show()
```



Plotting time series

```
google['Volume'].plot(color='red', figsize=(12, 4))

plt.title("The number of traded Google stocks from 2006 to 2018.")
```



MAD on time series

```
from pyod.models.mad import MAD

mad = MAD().fit(google[['Volume']])

is_outlier = mad.labels_ == 1

print(len(google[is_outlier]))
```

```
236
```

IForest on time series

```
google['day_of_week'] = google.index.day_of_week
google['month'] = google.index.month
google['day_of_month'] = google.index.day

google.head()
```

```
              Open    High     Low   Close    Volume  day_of_week  month  \
Date
2006-01-03  211.47  218.05  209.32  217.83  13137450            1      1
2006-01-04  222.17  224.70  220.09  222.84  15292353            2      1
2006-01-05  223.22  226.00  220.97  225.85  10815661            3      1
2006-01-06  228.66  235.49  226.85  233.06  17759521            4      1
2006-01-09  233.44  236.94  230.70  233.68  12795837            0      1
```

IForest on time series

```python
from pyod.models.iforest import IForest

iforest = IForest().fit(google)

# Generate probabilities
probs = iforest.predict_proba(google)

# Isolate the outliers
is_outlier = probs[:, 1] > 0.75
outliers = google[is_outlier]

print(len(outliers))
```

```
60
```

Let's practice!

ANOMALY DETECTION IN PYTHON

Time Series Decomposition for Outlier Detection

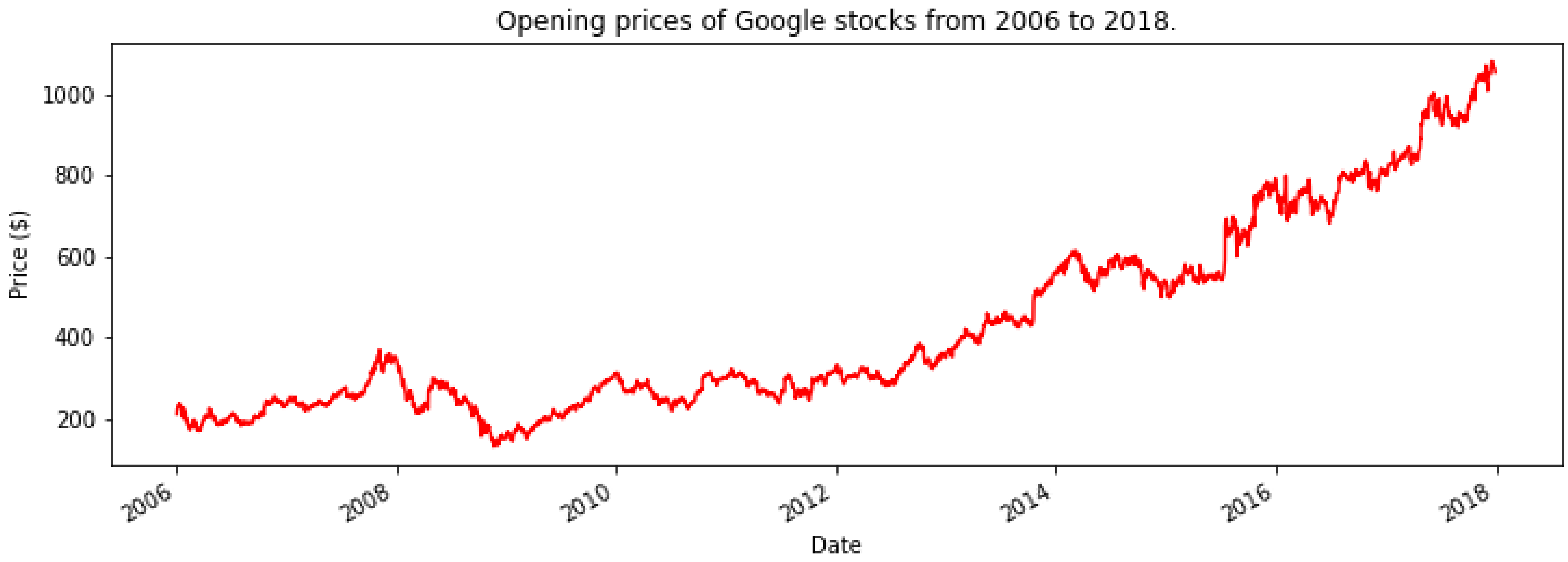
ANOMALY DETECTION IN PYTHON

Bekhruz (Bex) Tuychiev
Kaggle Master, Data Science Content Creator

Seasonality

- Repeating patterns in the time series
- Has a fixed frequency:
  - hourly
  - daily
  - weekly
  - monthly, etc.
- Examples:
  - Daily temperatures
  - Ice-cream sales

Seasonality



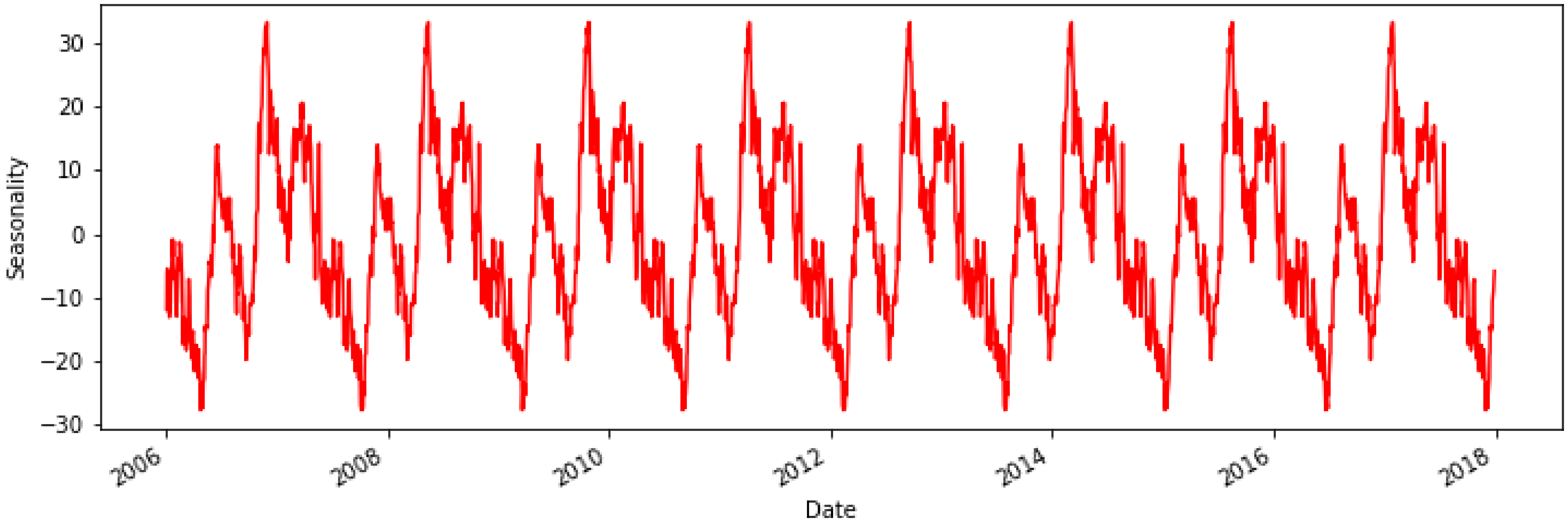
seasonal_decompose

```
from statsmodels.tsa.seasonal import seasonal_decompose

results = seasonal_decompose(google['Open'], period=365)

print(results)
```

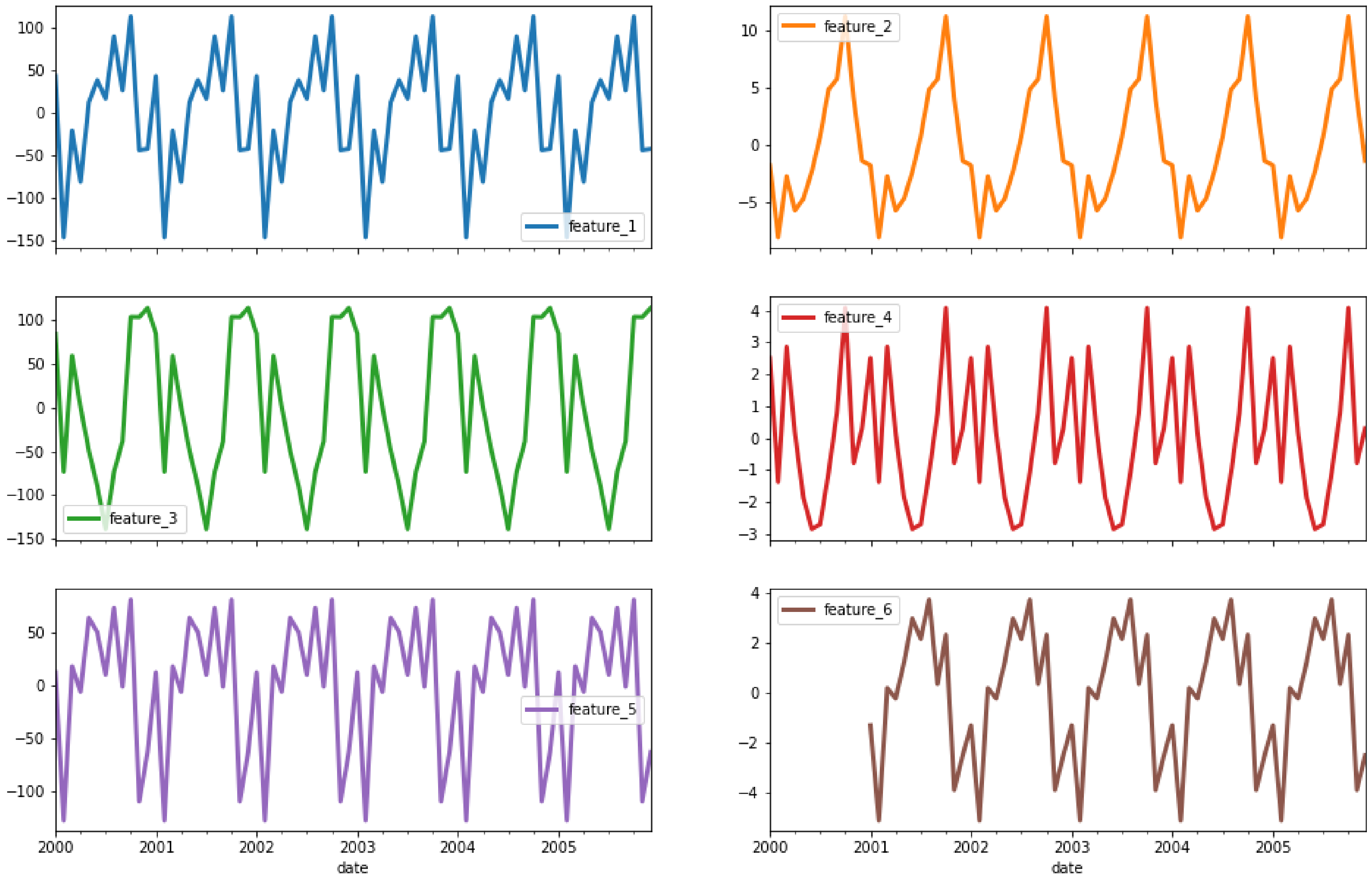
```
<statsmodels.tsa.seasonal.DecomposeResult object at 0x7f0a67fac820>
```

Plotting seasonality

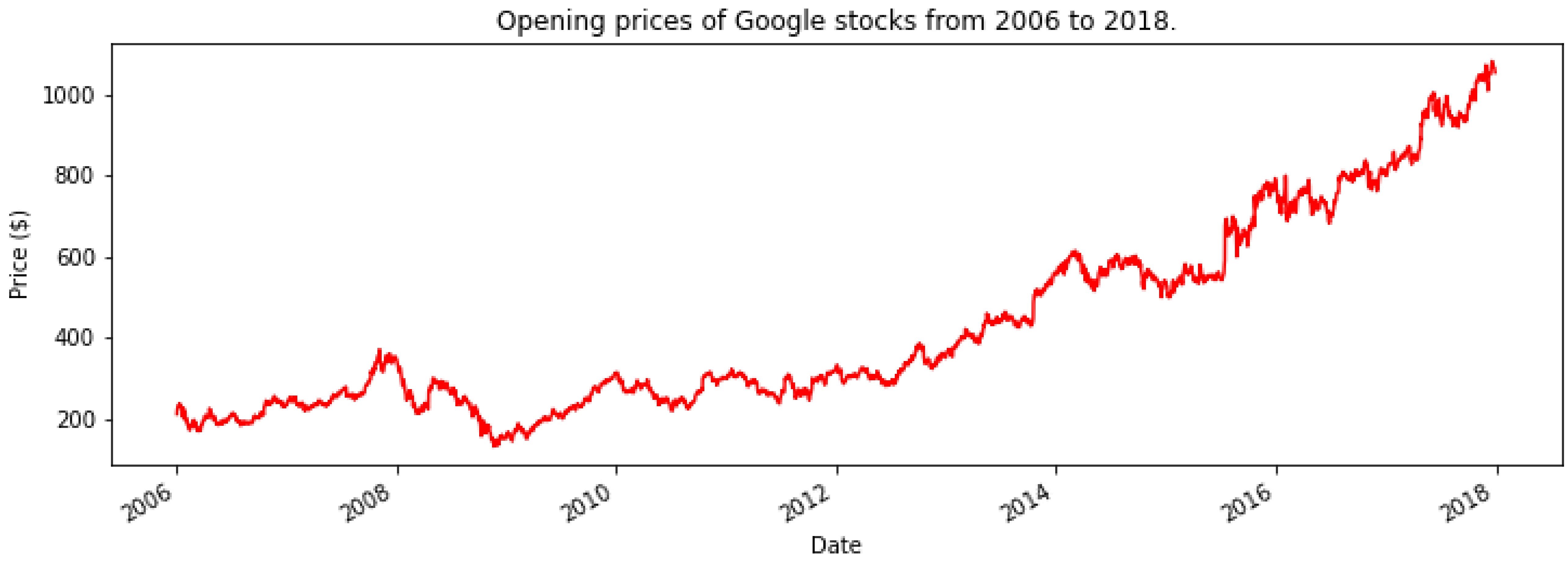
```
results.seasonal.plot(color="red", figsize=(12, 4))
```



Seasonality examples

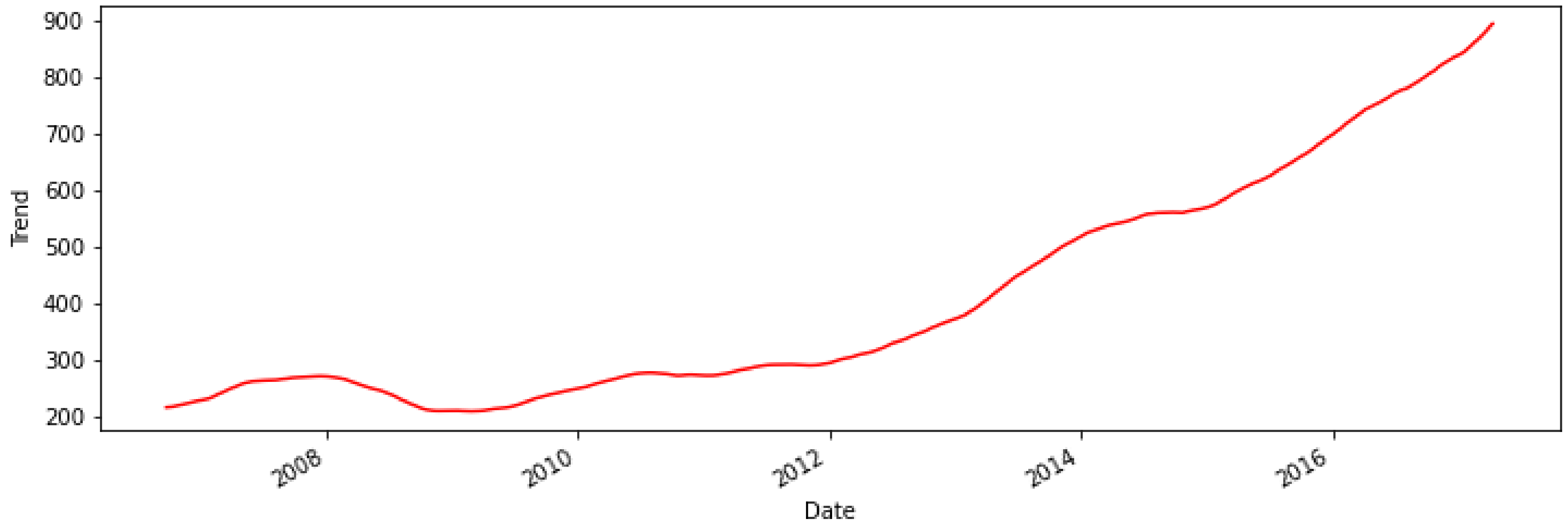


Initial plot of stocks

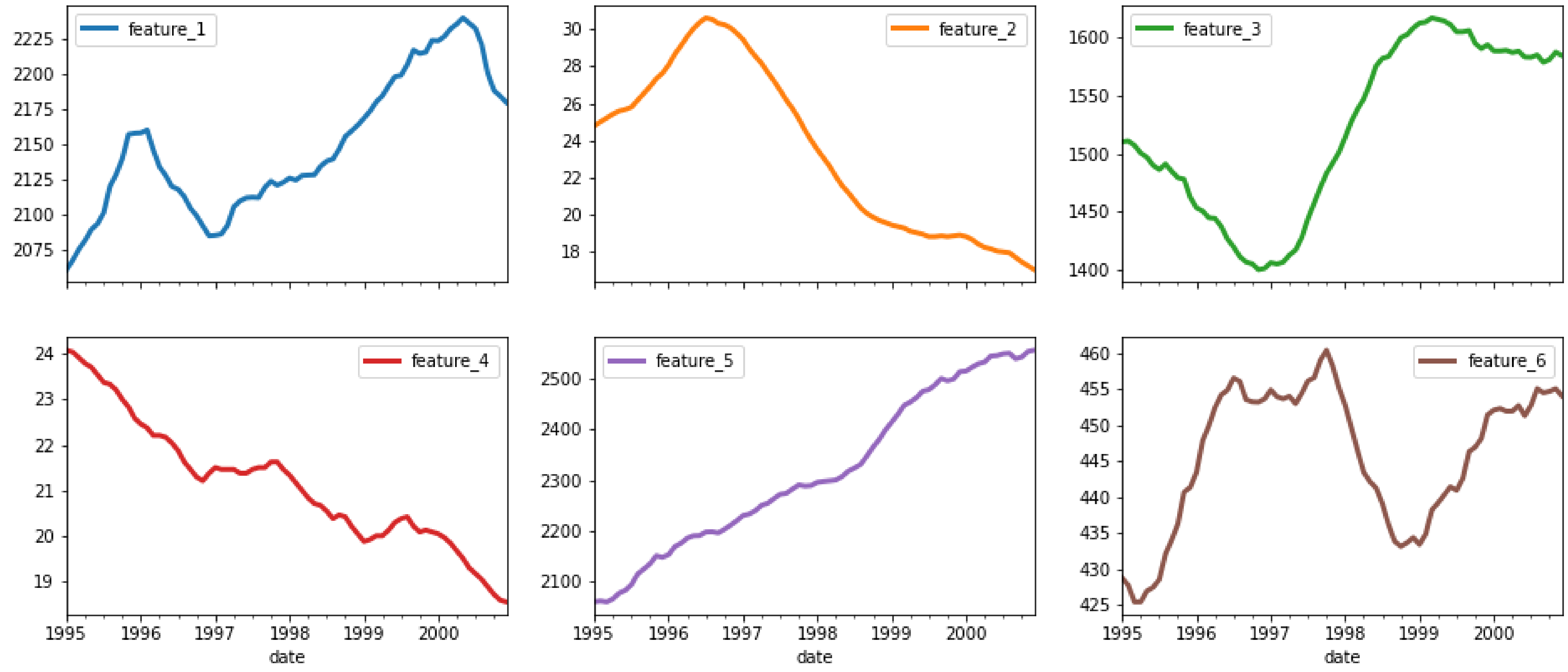


Trend

```
results.trend.plot(color="red", figsize=(12, 4))
```

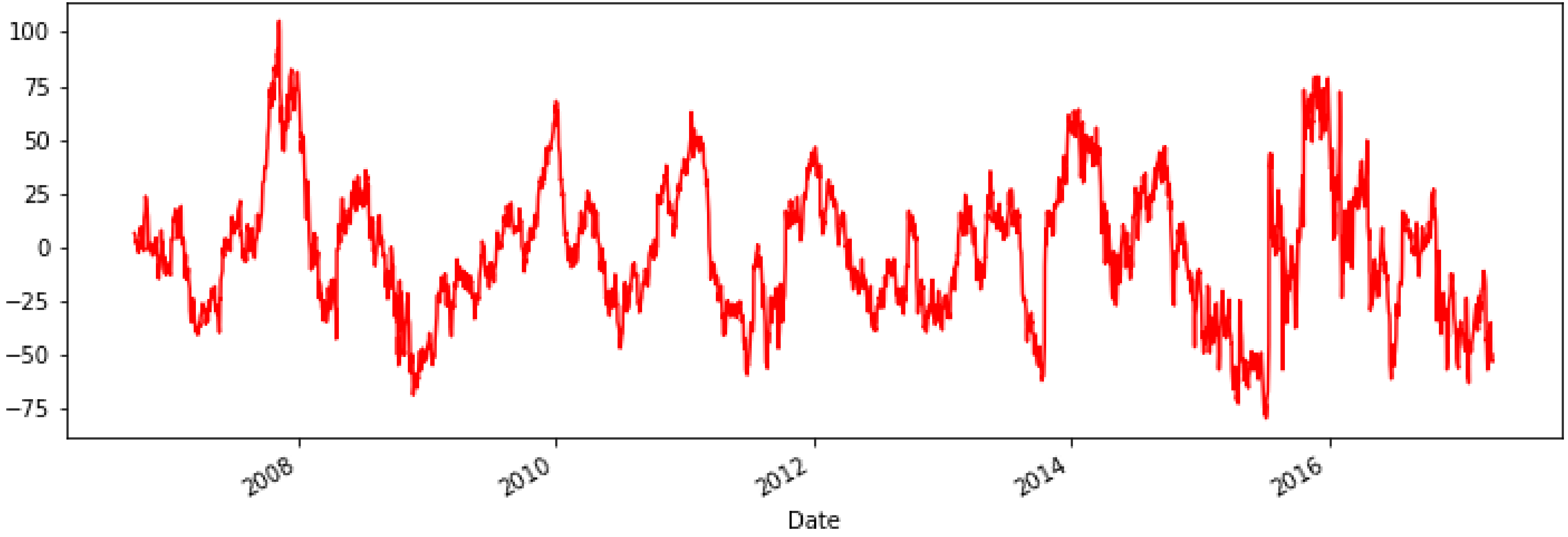


Trend examples



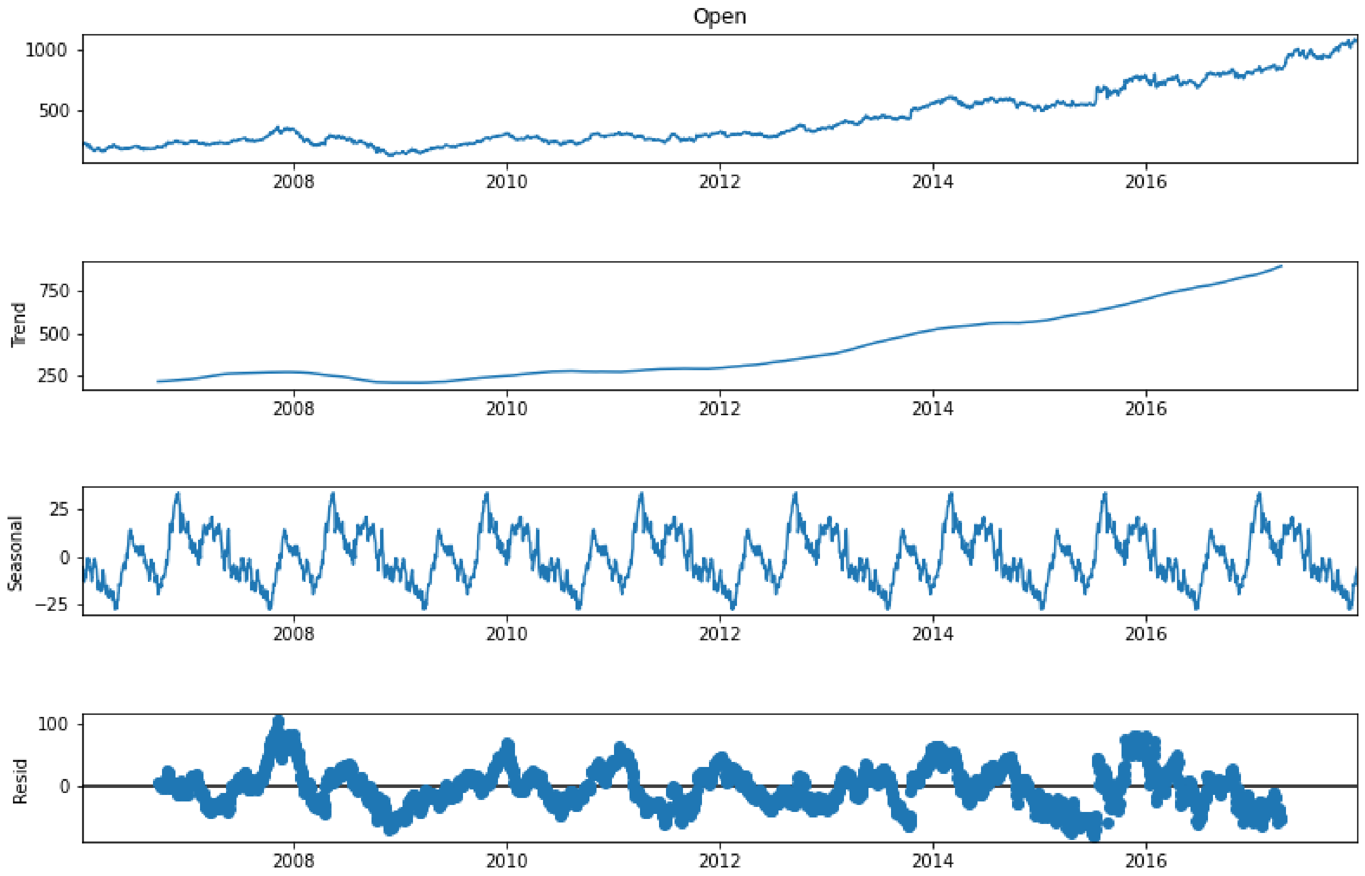
Residuals

```
results.resid.plot(color="red", figsize=(12, 4))
```



Decomposition

```
figure = results.plot()


figure.set_figwidth(12)
figure.set_figheight(8)
```



Fitting a classifier

```
# Extract and reshape residuals
results = seasonal_decompose(google['Volume'], period=365)
residuals = results.resid
residuals = residuals.values.reshape(-1, 1)

# Fit MAD
mad = MAD().fit(residuals)

# Find the outliers
is_outlier = mad.labels_ == 1
outliers = google[is_outlier]

print(len(outliers))
```

```
81
```

Let's practice!

ANOMALY DETECTION IN PYTHON

Outlier classifier ensembles

ANOMALY DETECTION IN PYTHON

Bekhruz (Bex) Tuychiev
Kaggle Master, Data Science Content Creator

Back to Airbnb

```python
outliers = airbnb_df[iforest.labels_ == 1]

outlier_probs = iforest.predict_proba(outliers)
outlier_probs[:10]
```

```
array([[0.51999538, 0.48000462],
       [0.61789522, 0.38210478],
       [0.61802032, 0.38197968],
       [0.35184434, 0.64815566],
       [0.57533286, 0.42466714],
       [0.59038933, 0.40961067],
       [0.57677613, 0.42322387],
       [0.54158826, 0.45841174],
       [0.49118093, 0.50881907],
       [0.21387357, 0.78612643]])
```

Probability threshold best practice

Threshold:

- 75% in low-risk cases
- >90% in high-cost cases like:
  - medicine
  - cyber security
  - limited data

What is an ensemble?

- Combination of two or more classifiers
- Predictions are more stable

Look at the data

```
google.head()
```

```
             Open    High     Low   Close    Volume  day_of_week  month  day
Date
2006-01-03  211.47  218.05  209.32  217.83  13137450            1      1    3
2006-01-04  222.17  224.70  220.09  222.84  15292353            2      1    4
2006-01-05  223.22  226.00  220.97  225.85  10815661            3      1    5
2006-01-06  228.66  235.49  226.85  233.06  17759521            4      1    6
2006-01-09  233.44  236.94  230.70  233.68  12795837            0      1    9
```

Scaling numeric features

```
from sklearn.preprocessing import QuantileTransformer

# Define the cols to be scaled
to_scale = ['Open', 'High', 'Low', 'Close', 'Volume']

# Initiate the transformer
qt = QuantileTransformer(output_distribution="normal")

# Scale and store the columns back
google.loc[:, to_scale] = qt.fit_transform(google[to_scale])
```

Creating arrays

```
# Create a list of estimators
estimators = [KNN(n_neighbors=20), LOF(n_neighbors=20), IForest()]

# Create an empty array
shape = (len(google), len(estimators))
probability_scores = np.empty(shape=shape)
```

Inside the loop

```
estimators = [KNN(n_neighbors=20), LOF(n_neighbors=20), IForest()]

shape = (len(google), len(estimators))
probability_scores = np.empty(shape=shape)

# Loop over and fit
for index, est in enumerate(estimators):
    est.fit(google)

    # Create probabilities
    probs = est.predict_proba(google)

    # Store the probs
    probability_scores[:, index] = probs[:, 1]
```

Aggregating - mean

```
mean_scores = np.mean(probability_scores, axis=1)


mean_scores
```

```
array([0.20699869, 0.21455413, 0.17166271, ..., 0.31255075, 0.33553513,
       0.32217186])
```

Aggregating - median

```python
median_scores = np.mean(probability_scores, axis=1)


median_scores
```

```
array([0.20699869, 0.21455413, 0.17166271, ..., 0.31255075, 0.33553513,
       0.32217186])
```

Probability filter

```
# Create a mask with 75% threshold
is_outlier = median_scores > 0.75

# Filter the outliers
outliers = google[is_outlier]

len(outliers)
```

```
3
```

Summary of the steps

```
# Create a list of estimators
estimators = [KNN(n_neighbors=20), LOF(n_neighbors=20), IForest()]
probability_scores = np.empty(shape=(len(google), len(estimators)))

for index, est in enumerate(estimators):
    # Fit and generate probabilities
    est.fit(google)
    probs = est.predict_proba(google)

    # Store the probabilities
    probability_scores[:, index] = probs[:, 1]
```

Summary of the steps

```python
# Average the scores
mean_scores = np.mean(probability_scores, axis=1)

# Filter with 75% threshold
outliers = google[mean_scores > 0.75]

print(len(outliers))
```

```
3
```

Let's practice!

ANOMALY DETECTION IN PYTHON

How to deal with found outliers

ANOMALY DETECTION IN PYTHON

Bekhruz (Bex) Tuychiev
Kaggle Master, Data Science Content Creator

Applications of anomaly detection

- Medicine
- Cyber security
- Fraud detection

Perform two analyses - with and without outliers.

The reasons for outlier presence

- Data entry errors:
  - Typos
  - Measurement errors
  - Human mistakes
  - Drop unless fixed
- Sampling errors:
  - Not from the target distribution
  - Drop
- Natural:
  - Naturally odd but comes from the population
  - Do not drop

Drop based on magnitude

- Too few: confirm and drop
- Too many: raises suspicion - use different models:
  - GLMs
  - Quantile Regression
  - GEEs
- Forms a cluster: perform deeper analysis

Trimming

```
# Calculate the percentiles
percentile_first = google['Volume'].quantile(0.01)
percentile_99th = google['Volume'].quantile(0.99)

# Trim
google['Volume'] = google['Volume'].clip(percentile_first, percentile_99th)
```

Replacing

```
google.replace(0, 100, inplace=True)
```

Let's practice!

ANOMALY DETECTION IN PYTHON

Congratulations!

ANOMALY DETECTION IN PYTHON

Bekhruz (Bex) Tuychiev
Kaggle Master, Data Science Content Creator

Chapters 1-2 recap

- Chapter 1 - univariate outlier detection:
  - Visual methods
  - Median Absolute Deviation of `pyod`
- Chapter 2 - Isolation Forest:
  - `IForest` of `pyod`
  - iTrees
  - Using outlier probabilities

Chapters 3-4 recap

- Chapter 3 - distance and density-based algrotihms:
  - `KNN` for outlier detection
  - `QuantileTransformer` for normalization
  - Local Outlier Factor algorithm
- Chapter 4 - time series anomalies and outlier ensembles:
  - Time series decomposition
  - Time series outlier detection from residuals
  - Building outlier ensembles manually
  - How to deal with found outliers

Thank you!

ANOMALY DETECTION IN PYTHON