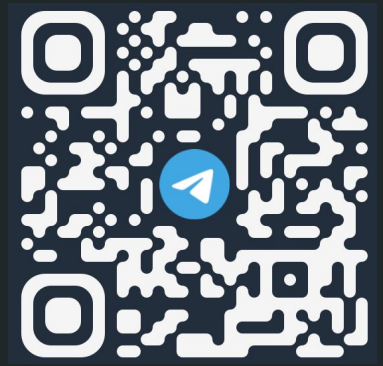
# Применение ML в задачах анализа и прогнозирования временных рядов.

Куликов Е. Ю.
Есалов К. Э.
НОЦ ИКНС СПбГУТ
https://vk.com/iktns
tg @asmodaay
https://t.me/+NGJKSSM3ohtkMzli

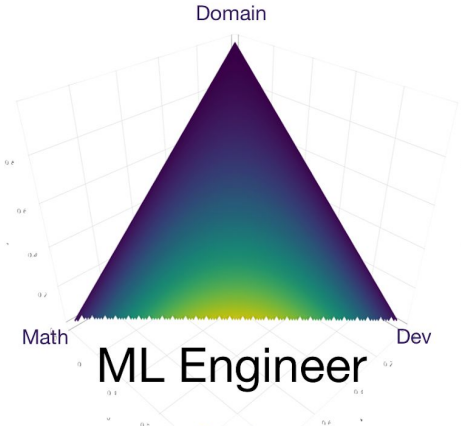
Чат в телеграмм



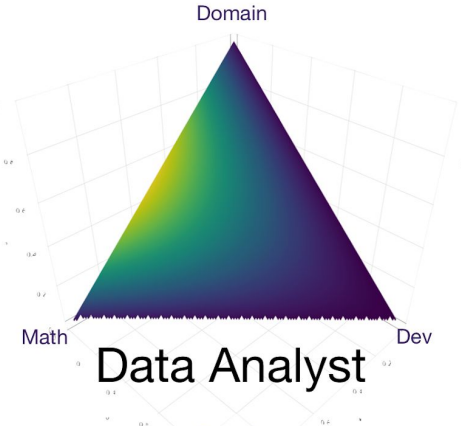
# План факультатива.

1) Введение в Data Science.
2) Jupyter notebook / google collab.
3) Введение в анализ временных рядов.
4) Парсинг и обработка данных.
5) Машинное обучение. Линейные модели.
6) Оценка моделей.
7) Машинное обучение. Регуляризация.
8) Машинное обучение. Дерево решений.
9) Машинное обучение. Случайный лес.
10) Машинное обучение. Градиентный бустинг.
11) Нейронные сети.
12) Kaggle.

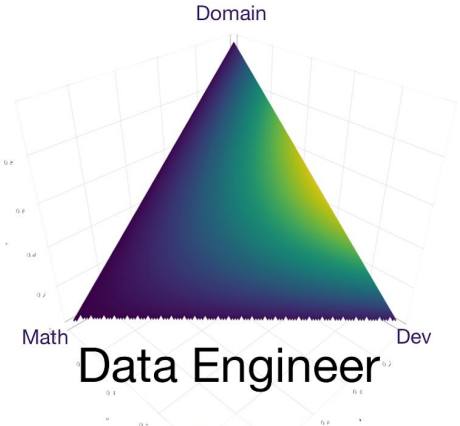
# ДАТАСАЕНТИСТ: ЗНАНИЯ И НАВЫКИ



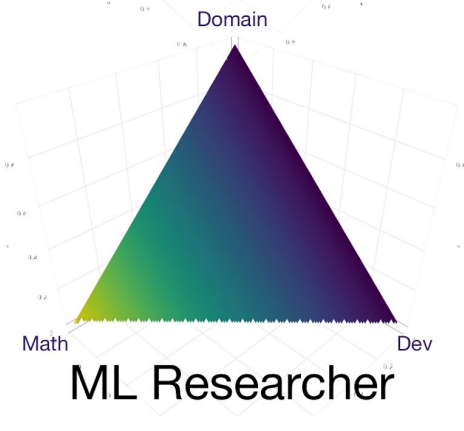

ML Engineer



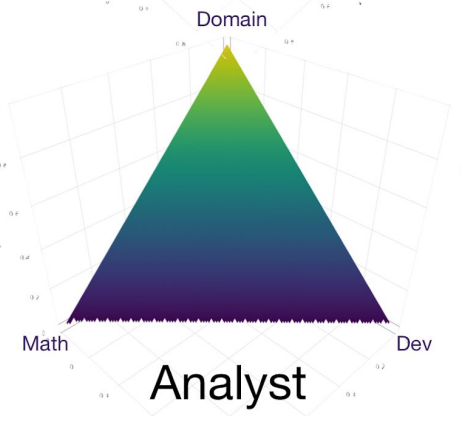

Data Analyst



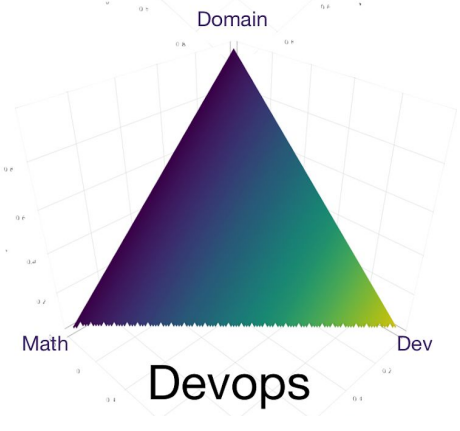

Data Engineer




ML Researcher




Analyst




Devops

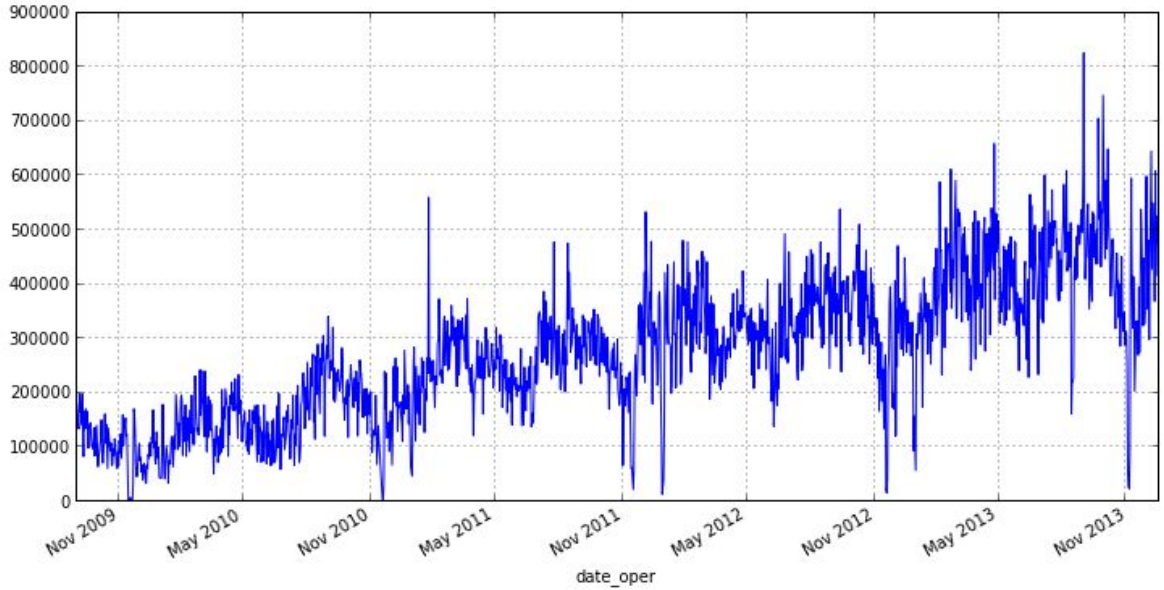
# Jupyter Notebook. Google Collab


https://jupyter.org/install

https://colab.research.google.com

# Введение в анализ временных рядов

Временно́й ряд — собранный в разные моменты времени статистический материал о значении каких-либо параметров исследуемого процесса.



# Понятие стационарности.

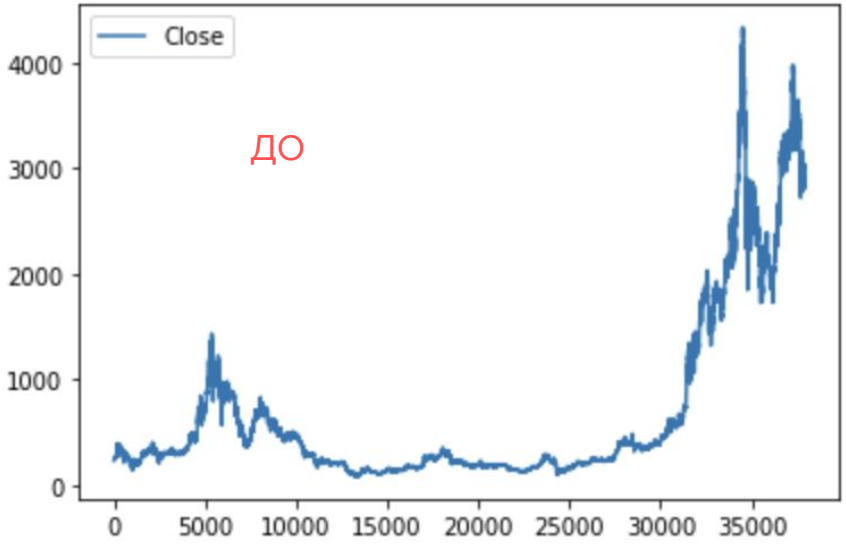
Временной ряд называется стационарным если выполняются следующие условия :

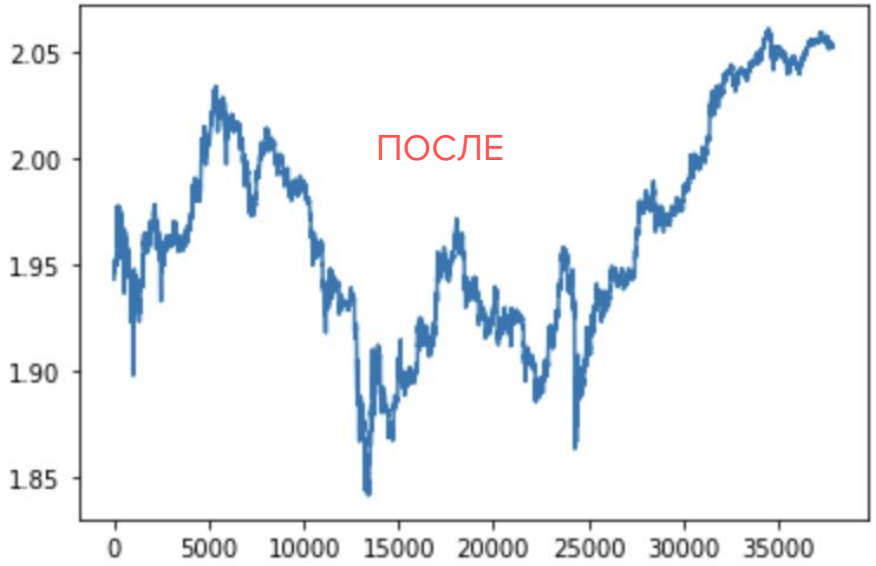
1) Математическое ожидание временного ряда не изменяется с течением времени
2) Дисперсия временного ряда не изменяется с течением времени

Иначе говоря временной ряд не имеет тренда

Преобразование бокса-кокса :

$$y_i^{\lambda} = \begin{cases} \dfrac{y_i^{\lambda}-1}{\lambda}, & \text{if } \lambda \neq 0, \\ \log(y_i), & \text{if } \lambda = 0. \end{cases}$$



# Преобразование бокса-кокса

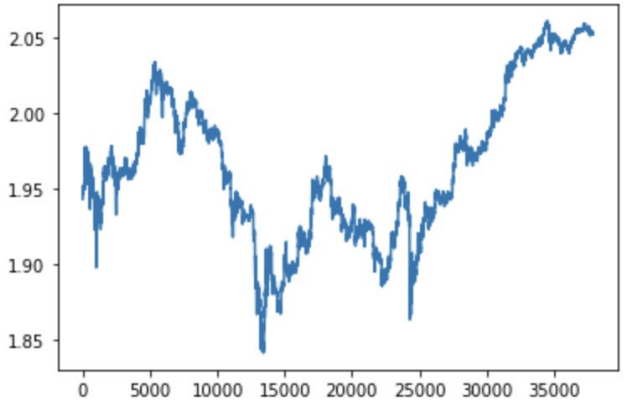
In [2]:
```
import pandas as pd
from scipy.stats import boxcox
```

In [6]:
```
data = pd.read_csv('asmoday/crypto_train_data/ETHUSD_60m_cutted.csv')[['Datetime','Close']]
```

In [12]:
```
transformed, lambda_ = boxcox(data['Close'].values)
data['box_cox'] = transformed
```

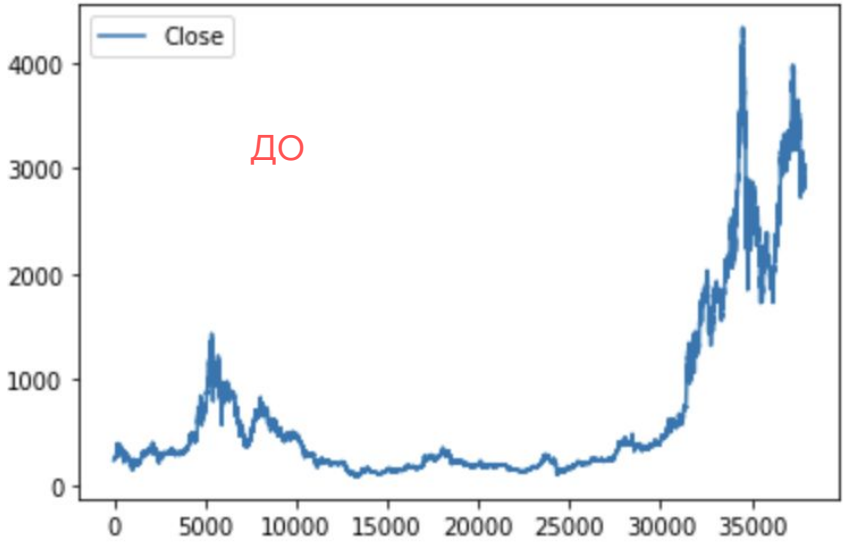
In [14]:
```
data.box_cox.plot()
```

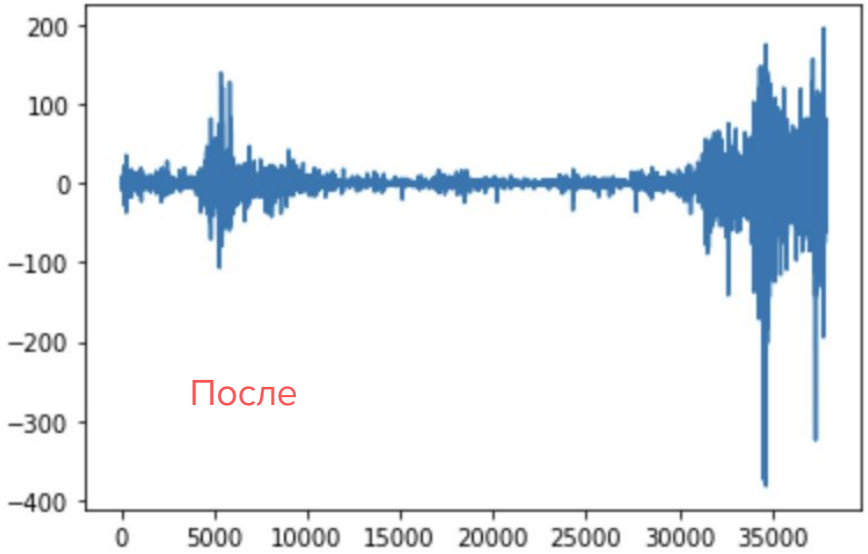
Out[14]: <AxesSubplot:>



# Дифференцирование временного ряда.

Дифференцированием временного ряда называют вычисление разницы следующего члена временного ряда от предыдущего. **y(t+1) = y(t) + delta**



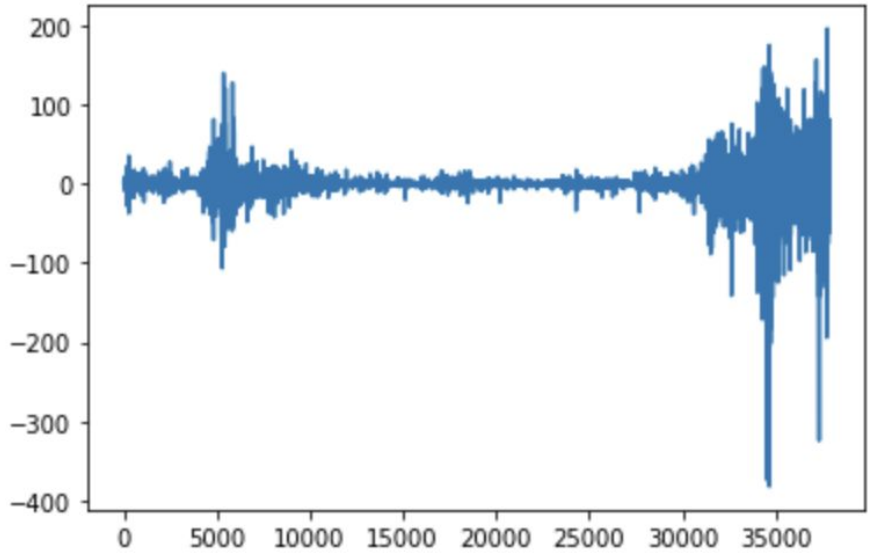



# Дифференцирование временного ряда.

```
data['diff'] = data.Close.diff()
```

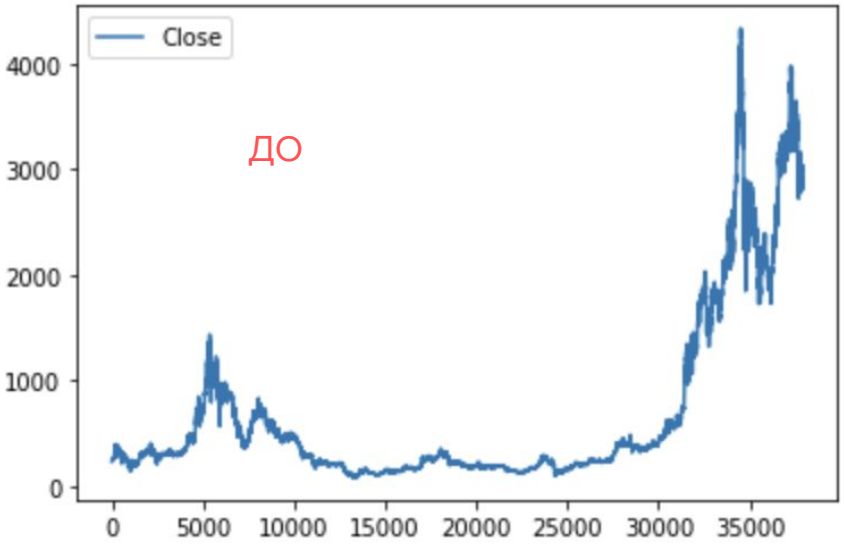
```
data['diff'].plot()
```

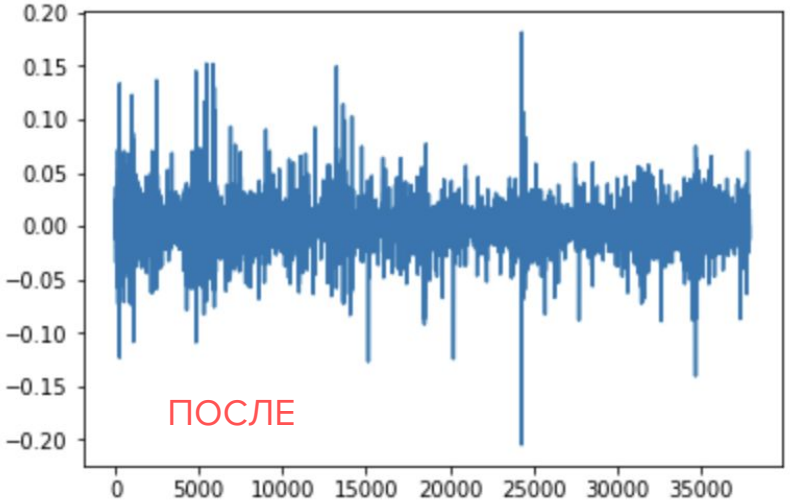
```
<AxesSubplot:>
```



# Долевое дифференцирование временного ряда *

**y(t+1) = y(t)* delta**

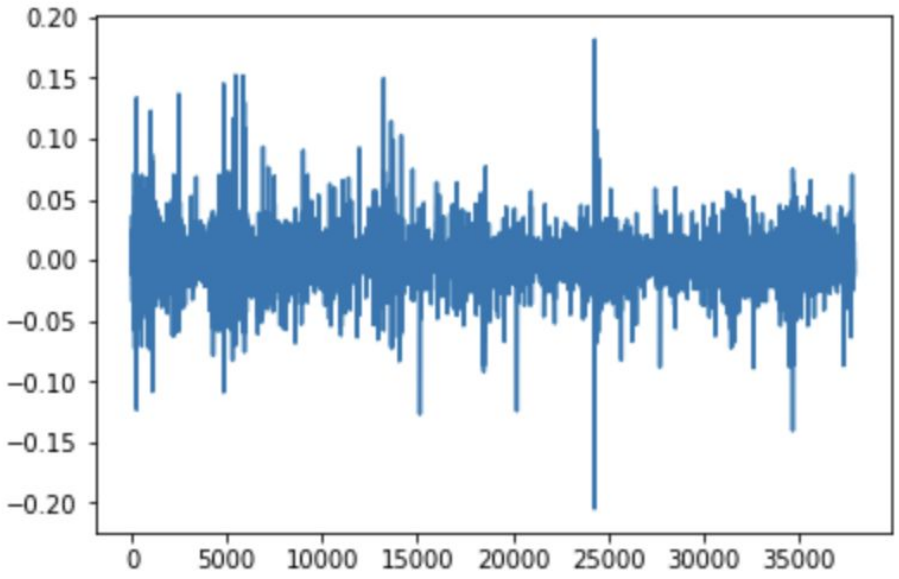


# Долевое дифференцирование временного ряда

```
data['pct_change'] = data.Close.pct_change()
data['pct_change'].plot()
```

```
<AxesSubplot:>
```



# Стандартизация/нормализация

Стандартизация ( z-преобразование)  z = (x- mu) / s

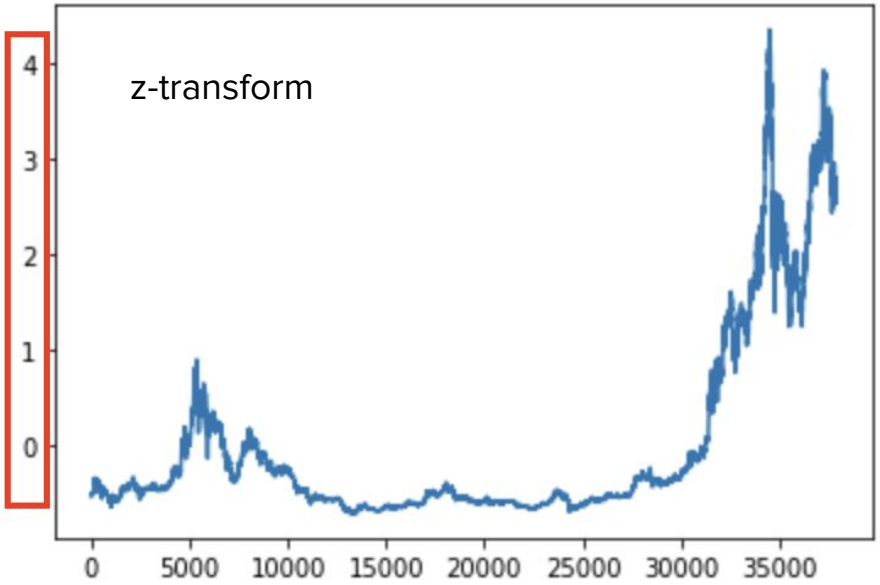
mu - математическое ожидание, s - стандартное отклонение

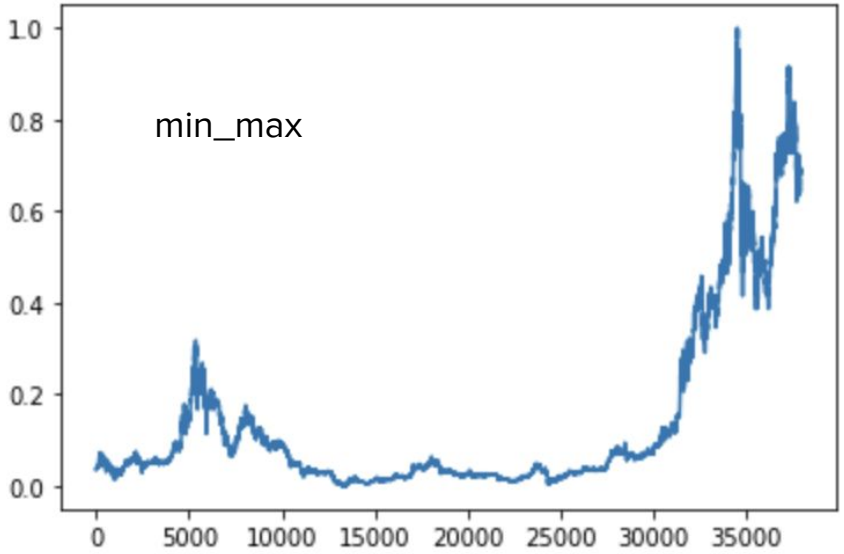
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
data['scaled'] = scaler.fit_transform(data.Close.values.reshape(-1,1))
```

Нормализация:  x_scaled = (x - x_min) / ( x_max - x_min)

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
data['minmax_scaled'] = scaler.fit_transform(data.Close.values.reshape(-1,1))
```

# Стандартизация/нормализация



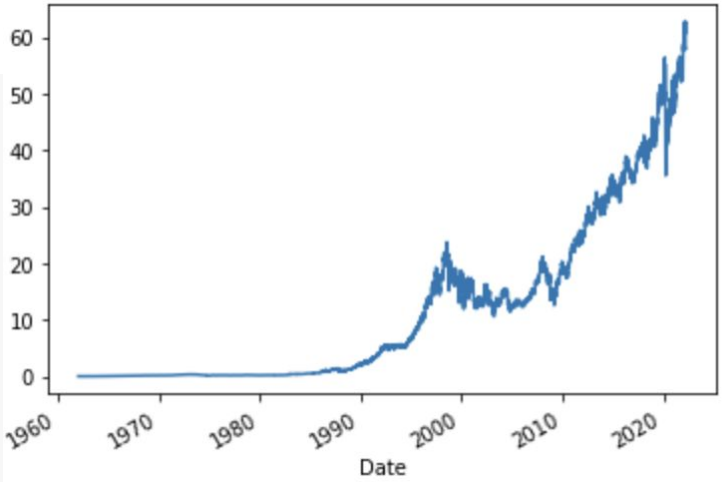



# Парсинг и обработка данных

```
!pip install yfinance
import yfinance as yf
tickerSymbol = 'KO'

#get data on this ticker
tickerData = yf.Ticker(tickerSymbol)

#get the historical prices for this ticker
tickerDf = tickerData.history(period='max')
tickerDf.Close.plot()
```



# Генерация лаговых фичей.

```
prediction_window = 10
prediction_columns = ['Close']
for i in range(1,prediction_window+1):
    col_name = f'shift_{i}'
    prediction_columns.append(col_name)
    tickerDf[col_name] = tickerDf.Close.shift(i)
data = tickerDf[prediction_columns]
data = data.dropna()
data.head()
```

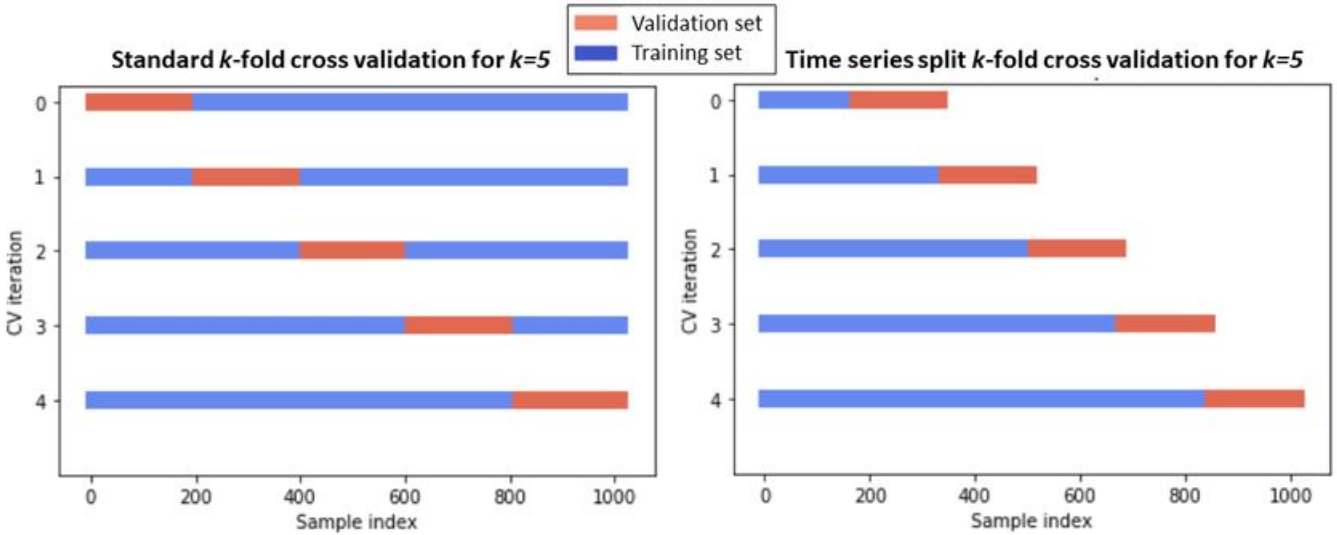
| Date | Close | shift_1 | shift_2 | shift_3 | shift_4 | shift_5 | shift_6 | shift_7 | shift_8 | shift_9 | shift_10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **1972-06-16** | 1.317395 | 1.310543 | 1.301977 | 1.291698 | 1.298551 | 1.298551 | 1.305404 | 1.310543 | 1.315682 | 1.319109 | 1.368612 |
| **1972-06-19** | 1.331100 | 1.317395 | 1.310543 | 1.301977 | 1.291698 | 1.298551 | 1.298551 | 1.305404 | 1.310543 | 1.315682 | 1.319109 |
| **1972-06-20** | 1.327674 | 1.331100 | 1.317395 | 1.310543 | 1.301977 | 1.291698 | 1.298551 | 1.298551 | 1.305404 | 1.310543 | 1.315682 |
| **1972-06-21** | 1.332814 | 1.327674 | 1.331100 | 1.317395 | 1.310543 | 1.301977 | 1.291698 | 1.298551 | 1.298551 | 1.305404 | 1.310543 |
| **1972-06-22** | 1.339666 | 1.332814 | 1.327674 | 1.331100 | 1.317395 | 1.310543 | 1.301977 | 1.291698 | 1.298551 | 1.298551 | 1.305404 |

# Стандартное разбиение данных

```
from sklearn.model_selection import train_test_split
```

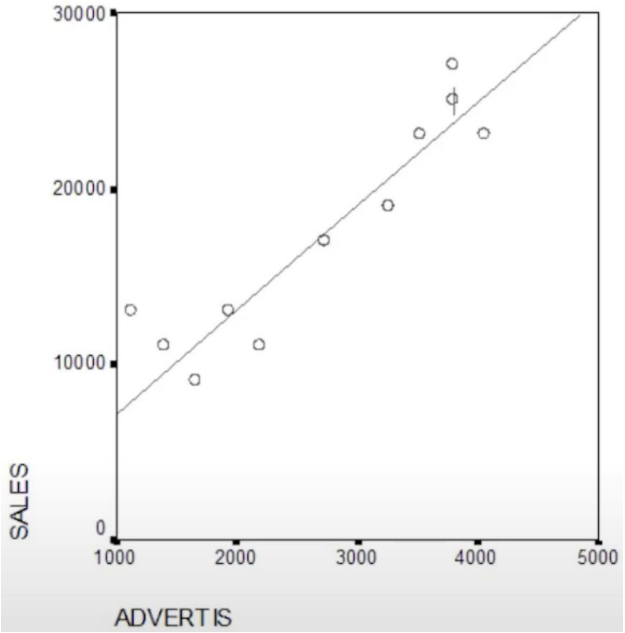
```
train_df, test_df= train_test_split(tickerDf, test_size=0.33, random_state=42,shuffle=False)
```

## Кросс валидация



# Линейная регрессия. Средняя квадратичная ошибка. (MSE)

y = k  * x + b




Изначально инициализируем случайные k,b ,  а дальше методом градиентного спуска корректируем коэффициенты в направлении минимизации MSE.

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

# Линейная регрессия. Python.

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

```
X = data[prediction_columns[:-1]].values
y = data[prediction_columns[-1]].values.reshape(-1, 1)

x_train,x_test,y_train,y_test = train_test_split(X,y, test_size=0.33, random_state=42,shuffle=False)
```

```
regressor = LinearRegression()
regressor.fit(x_train,y_train)
print(mean_squared_error(regressor.predict(x_test),y_test))
```

```
0.5119004324467789
```

# Задание на самостоятельную работу

1) Установить jupyter/ collab и необходимые библиотеки
2) Выбрать тикер акции (каждый свой)
3) Построить линейную регрессию на загруженных данных
4) Обработать данные
5) Оценить модель метриками MAE, MAPE
6) Попробовать различные методы преобразования данных
7) Попробовать различный размер предиктивного окна

# Оценка моделей

Качество работы моделей всегда хочется посчитать в понятных единицах измерения, mse, mae не очень годятся для этого.