Machine Learning zur Aktienprognose

Implementation und Anwendung in Python

Vincent Piegsa

Hamburg, den 14. September 2019

Inhaltsverzeichnis

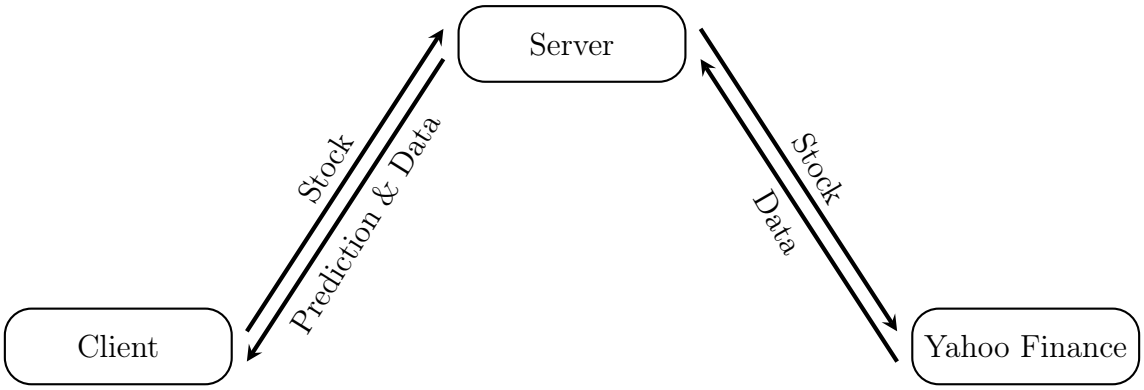
1 Einleitung

Algorithmischer Handel ist keinesfalls eine Neuheit – schon in den 1970ern gab es erste Versuche. Doch erst durch die steigende Computerleistung und automatische Datenanalyse in den 80ern waren die Algorithmen in der Lage, bis zu einem gewissen Grad tatsächlich relevante Vorhersagen zu treffen.

Heutzutage werden mehr als die Hälfte aller Transaktionen von Algorithmen durchgeführt.

2 Projektidee

Das Ziel dieses Projekts ist es, mit Hilfe von grundlegenden Algorithmen (z.B. Linear und Support Vector Regression) einfache Vorhersagen über den Verlauf von Aktienkursen zu machen. Diese Anwendung soll online über einen Webserver verfügbar sein, ebenso wie der historische Aktienkurs der jeweiligen Firma.



3 Datensatz

Welche Daten sind für die Prognose von Aktienkursen relevant? Zum einen spielen die historischen Daten eine große Rolle – Also der Öffnungs-, Höchst-, Tiefst- und Schlusspreis, sowie das Volumen. Zudem können Daten wie der Trend der letzten 10 Tage Aufschluss über den zukünftigen Verlauf geben.

Allerdings haben auch die Medien Auswirkungen auf den Aktienpreis. So können z.B. alle Artikel mit Bezug zur Firma oder Branche analysiert werden.

4 Algorithmen

4.1 Lineare Regression

Das Ziel von linearer Regression ist es, die Funktionsgleichung einer Geraden $\hat{y}$ zu errechnen, die die Datenpunkte bestmöglichst widerspiegelt.

$$\hat{y} = m \cdot \overline{x} + b \tag{1}$$

Der *y-Achsenabschnitt* b lässt sich durch Äquivalenzumformung der Funktionsgleichung errechnen:

$$b = \overline{y} - m \cdot \overline{x} \tag{2}$$

Die Steigung m wird durch folgende Formel ermittelt:

$$m = \frac{\overline{x} \cdot \overline{y} - \overline{xy}}{(\overline{x})^2 - \overline{x^2}} \tag{3}$$

Die Genauigkeit dieser Geraden kann durch *Bestimmtheitsmaß* R^2 beurteilt werden. Dazu wird der *Squared Error* E von $\hat{y}$ durch den *Squared Error* von $\overline{y}$ geteilt.

$$E = \sum_{n=0}^{k} (output_{\mathrm{n}} - target_{\mathrm{n}})^2 \tag{4}$$

$$R^2 = 1{,}0 - \frac{E(\hat{y})}{E(\overline{y})} \tag{5}$$

Je mehr sich R^2 an 1,0 annähert, desto besser repräsentiert $\hat{y}$ den Datensatz, und desto eher entsprechen die Voraussagen der Realität. Allerdings ist nicht jedes Problem linear lösbar.

5 Implementation

5.1 Module

Zur einfacheren Implementierung der Algorithmen und Datenanalyse sollten folgende Module über *pip* installiert werden:

- pandas
- pandas-datareader
- numpy
- scikit-learn
- matplotlib

5.2 Server

Das gesamte Projekt basiert auf einem *flask*-Webserver, der auf einem *Raspberry Pi 3* läuft. Alle Inhalte werden dynamisch mit der in *flask* integrierten Template-Engine *Jinja2* generiert. Die grafische Darstellung der Aktienkurse wird mit *plotly* umgesetzt.

5.3 Lineare Regression

```
1 from sklearn import preprocessing, cross_validation
  from sklearn.linear_model import LinearRegression

  import numpy as np

6 import pandas as pd
  import pandas_datareader.data as web
```

```
import datetime


def get_data(stock: str) -> pd.DataFrame:
    """
    Gets historical stock data from Yahoo Finance API

    Parameters
    ----------
    stock : str
        Stock Symbol

    Returns
    -------
    pd.DataFrame
        Historical stock data
    """
    # defining starting and ending time
    start = datetime.datetime(2010, 1, 1)
    end = datetime.datetime.today()

    # returning historical data
    return web.DataReader(stock, 'yahoo', start, end)


def process_data(df: pd.DataFrame) -> pd.DataFrame:
    """
    Processes stock data

    Parameters
    ----------
    df : pd.DataFrame
        Historical stock data

    Returns
    -------
    pd.DataFrame
        Processed stock data
    """
    # difference between high and low
    df['HL_PCT'] = (df['High'] - df['Low']) / df['Adj Close'] * 100.0
    # overall change over the day
    df['PCT_change'] = (df['Adj Close'] - df['Open']) / df['Open'] * 100.0
    # 10d average
    df['10D_AVG'] = df['Adj Close'].rolling(window=10).mean()

    df = df[['Adj Close', 'HL_PCT', 'PCT_change', '10D_AVG', 'Volume']]

    # replacing every nan value
    df.fillna(value=-99999, inplace=True)

    return df


def train(df: pd.DataFrame) -> LinearRegression:
    """
    Trains Linear Regression model on given dataset
```

```
66
    Parameters
    ----------
    df : pd.DataFrame
        Historical stock data

71
    Returns
    -------
    LinearRegression
        Linear Regression model
76  """
    df = process_data(df)

    # 5 day forecast
    df['Label'] = df['Adj Close'].shift(-5)
81  df.dropna(inplace=True)

    # defining X and y for algorithm
    X = np.array(df.drop(['Label'], 1))
    y = np.array(df['Label'])
86
    # splitting dataset into training and testing data
    X_train, X_test, y_train, y_test = cross_validation.train_test_split(
        X, y, test_size=0.2)

91  # training algorithm with as many threads as possible
    clf = LinearRegression(n_jobs=-1)
    clf.fit(X_train, y_train)

    # coefficient of determination (R2)
96  confidence = clf.score(X_test, y_test)
    print(confidence)

    return clf


101
def predict(df: pd.DataFrame, classfier: LinearRegression) -> pd.DataFrame:
    """
    Predicts the future stock prices

106 Parameters
    ----------
    df : pd.DataFrame
        Historical stock data
    classfier : LinearRegression
111     Linear Regression model

    Returns
    -------
    pd.DataFrame
116     Predicted stock price
    """
    # recent 5 days
    df = process_data(df)[-5:]

121 # predicting price for the next 5 days
    return clf.predict(df)
```

```
if __name__ == '__main__':

    # getting data for Apple
    data = get_data('AAPL')

    # training and predicting
    clf = train(data)
    predicted = predict(data, clf)

    # printing prediction
    print(predicted)
```

linearregression.py

Hinweis: Der ganze Code ist online in meinem GitHub Repository verfügbar.

6 Evaluierung

6.1 Vorhersagen

Datum	Adj. Close	Vorhersage	Differenz
2019-09-06	213,26 $	209,51 $	3,72 $
2019-09-09	214,17 $	209,20 $	4,97 $
2019-09-10	216,70 $	206,18 $	10,52 $
2019-09-11	223,59 $	209,7 $	13,89 $
2019-09-12	223,09 $	213,79 $	9,30 $

Tab. 1: Apple (AAPL): Vorhersage und tatsächlicher Verlauf

Datum	Adj. Close	Vorhersage	Differenz
2019-09-06	126,60 $	126,39 $	0,21 $
2019-09-09	126,84 $	125,00 $	1,84 $
2019-09-10	122,30 $	126,59 $	4,29 $
2019-09-11	123,50 $	127,71 $	4,21 $
2019-09-12	124,82 $	127,69 $	2,87 $

Tab. 2: Airbus (AIR.PA): Vorhersage und tatsächlicher Verlauf

6.2 Fazit

Wie zu sehen ist, stimmen die Vorhersagen kaum mit den tatsächlichen Preisen überein. Das kann mehrere Möglichkeiten haben:

- Das Problem ist nicht linear
- Unzureichende Auswahl der Trainingsdaten
- Zu kleiner Datensatz

Literatur

[1] Harrison Kinsley. Practical machine learning. 2016. Online verfügbar unter https://pythonprogramming.net/machine-learning-tutorial-python-introduction.

[2] Lukas Kohorst. Predicting stock prices with python. 2018. Online verfügbar unter https://towardsdatascience.com/predicting-stock-prices-with-python-ec1d0c9bece1.

[3] Wikipedia. Linear regression. 2019. Online verfügbar unter https://de.wikipedia.org/wiki/Lineare_Regression.