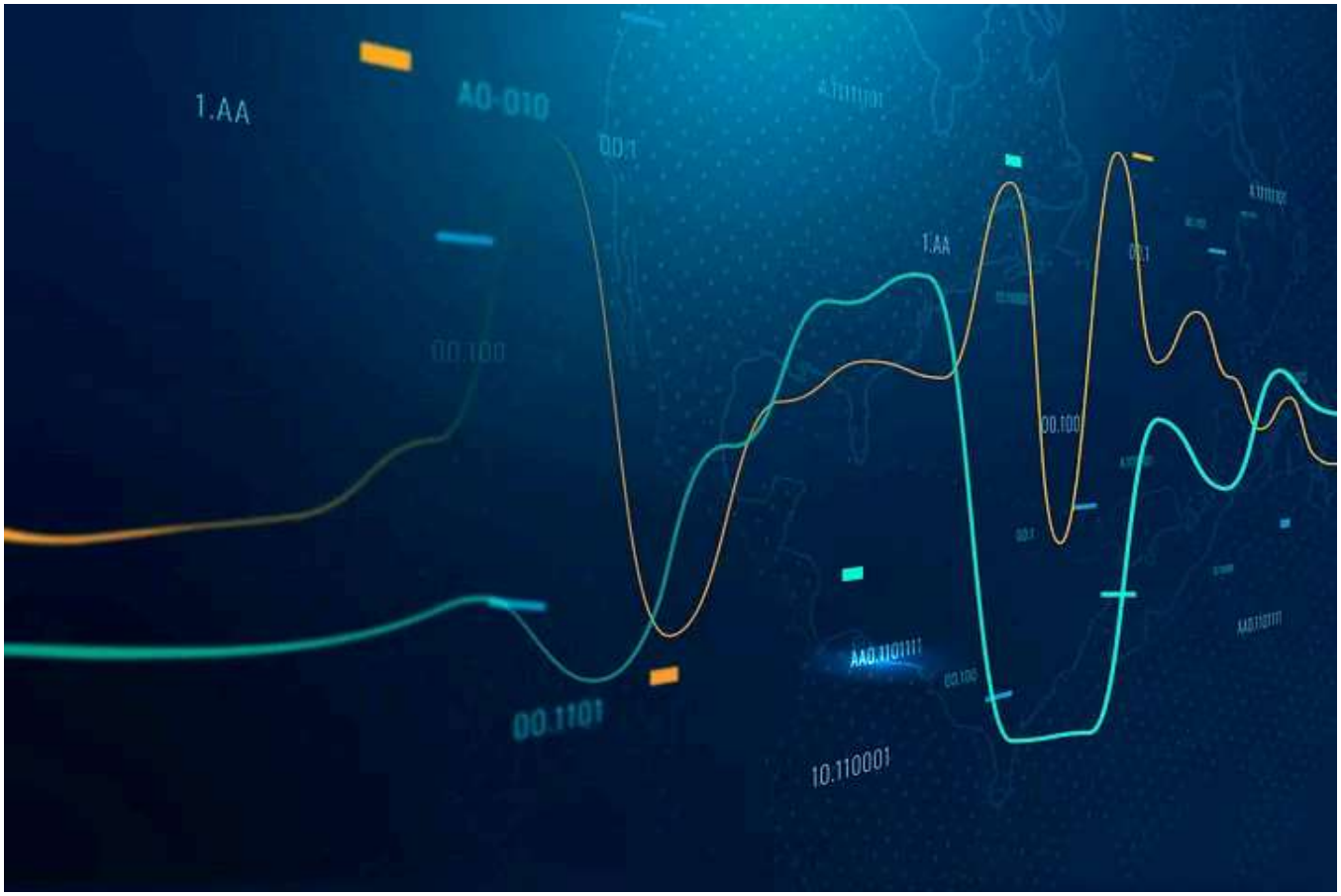
Algorithmic Trading with Python

12 min read · Oct 10, 2021

Lumos Student Data Consulting Follow

In this article three members of Lumos Adrian Ciarnau, Maroan El Sirfy and Kerstin Scharinger will share their journey in the world of algorithmic trading with the purpose of building a trading bot in Python over the summer.



General Introduction

Algorithmic trading is a trading system based on a defined set of instructions that generates signals for trades on the market. The combined fields of Finance and Programming turn the analysis of mathematical, statistical and technical models into a trading decision. This article aims to demonstrate the implementation of a trading strategy based on multiple technical indicators and sentiment analysis for trading on a sample depot. Therefore, some programming steps in Python with explanations of the used algorithms are included.

We start with a little journey into the history of AlgoTrading to make yourself familiar with the evolution from the very beginnings to today's high-frequency trading. Back in the 1930s, the first investors priced stocks and bonds by started calculating mathematical formulas. Later on, Harry Markowitz solved the portfolio selection problem in the 1950s and thus marked the beginning of computational finance. Due to the shortage of computer power at that time very complex analyses were difficult, so the traders simplified assumptions to establish simple patterns that generated trading signals. With the rise of personal computers and the increasing computing power the field of computational finance became even more important. [1]

The first developments of full electronic order executions started in the late 1980s and 1990s. One reason, why the rule-based trading start developing back then, was the decreasing spread, the difference between the bid and offer prices of assets, which make the market-makers business less profitable.

Later the computerized High-Frequency Trading was authorized and marked the beginning for the evolution and the rapid growth of faster order executions. Nowadays around 60 percent of the traders trust in algorithmic trading and its benefits of executing a large number of trades within seconds without human interventions. [2]

Importing the Data

First, we have to import the desired data on which we want to perform our algorithmic trading algorithms. In this article, we used the stock prices of Apple, Microsoft and Tesla for demonstration purposes, but any other stocks can be used. Furthermore, we obtained current news data for these stocks to enhance our trading strategy. In the next paragraphs, we will show how to obtain the data for further analysis.

Stock Data

The required stock data is downloaded from yahoo finance in regular intervals (which can be chosen, based on the needed actuality of the data). In our example, we decided to stick with 3 major blue-chip stocks for the sake of this article: Apple, Microsoft and Tesla. More can always be added as yahoo finance allows to dynamically download the required stocks.

News Data

To analyze the current market mood regarding a desired stock, we need to retrieve current news from the web belonging to our selected company. Therefore, we decided to use the freely available news API which provides a search engine for news of the last month from a wide range of well-known newspapers. The developer version can be subscribed to free with an e-mail address, allowing 100 requests per day. The News API Package can be easily installed with the NewsApiClient. Finally, the API key serves authentication purposes. [3]

To filter the news articles for the specified stocks, we used the endpoint that searches the article title and body for keywords and phrases. The endpoint provides various further attributes to determine the search results like sources, date range (from, to) and the language. [4]

```
import pandas as pd
from newsapi import NewsApiClient
api = NewsApiClient(api_key='your key')

def news_api(company, date):
  all_articles = api.get_everything(
    q = company,
    language = 'en',
    from_param = date,
    to = date,
    page_size = 100,
    sort_by = 'relevancy')
  return all_articles
```

In order to obtain the news articles of the last 30 days, the date range function calculates the dates for our defined time range and deliver the dates from yesterday until 30 days before. We iterate over our defined companies and the desired time range and obtain 100 news articles per company per day and store the results.

After retrieving the available historic data, we are able to only request the actual daily news and append it to our existing historical news. From there on, we can run these lines of code every day to always obtain the most recent headlines.

```
#get current news data from today and store in csv file
for company in my_companies:
    df_news = get_news(company,
datetime.date.today()+datetime.timedelta(-1))
    df_news.to_csv('data/news_articles.csv', mode='a', header=False,
index=False)
```

These news data will serve as a source for the following market mood check regarding the selected companies. We will elaborate on the next steps in the paragraph for the sentiment analysis.

Technical Indicators

Technical Indicators are calculations based on the price of a security that measures important trading concepts such as trends, volatility or momentum. These indicators aim to identify supply, demand and patterns in the market to determine in which direction the price will continue.

As technical indicators, we decided to test three very basic ones: EMA, RSI and MACD.

The Exponential Moving Average (EMA) is a trend indicator based on a simple moving average that gives more weight to recent price data. [5]

Exponential Moving Average:

$$EMA(today) = (Close(today) - EMA(today-1)) * Multiplier + EMA(today-1)$$

Exponential Moving Average formula

The Relative Strength Indicator (RSI) is an oscillator indicator that can have a value between 100 and 0. Usually the experts say that if the RSI is over 70 the market is overbought and if it is under 30 it is oversold. We use this approach in our strategy.

Relative Strength Indicator (RSI):

$$RS = \frac{Avg.Gain}{Avg.Loss}$$

$$RSI = 100 - \frac{100}{1 + RS}$$

Relative Strength Indicator formula

The Moving Average Convergence Divergence (MACD) is a trend-following momentum indicator that demonstrates the relationship between a longer and a shorter exponential moving average with a signal line that crosses that MACD line and indicates possible trading signals. [6]

Moving Average Convergence Divergence (MACD):

$$MACD(today) = EMA\_12(today) - EMA\_26(today)$$

$$Signal\ Line = EMA\_9_{MACD}(today)$$

Moving Average Convergence Divergence formula

Image by Sabrina Jiang © Investopedia 2020

The way a trading signal can be generated using the MACD is straightforward. The crossover between the signal line and the MACD line represents the signal points. A long signal can be seen as the crossover of the MACD (blue) above the signal line (yellow). A short signal is the opposite, where the MACD falls below the signal line.

Sentiment Analysis

> *"Sentiment analysis (also known as opinion mining or emotion AI) is the use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information." [7]*

Sentiment analysis classifies a text into positive, negative or neutral from the range -1 for negative and +1 for positive. A weighted sentiment score is assigned to sentences or phrases with natural language processing (NLP) techniques. To retrieve the sentiment of our news data, we use the VADER sentiment analysis a lexicon and a rule-based feeling analysis algorithm. A sentiment rating, depending on the word's rather positive or pessimistic meaning, is assigned to each word in the lexicon. To customize the sentiment for our news data analysis regarding stock mood detection, VADER provides the possibility to update the lexicon and specify the sentiment rating for our specific content. Therefore, we evaluate relevant words that may indicate a possible strong movement in the company's stock price with a higher/lower sentiment rating. Furthermore, we did not perform any data cleaning tasks like we would do with other NLP tasks, as we would possibly remove useful information for the sentiment score. VADER is case sensitive and assigns higher scores to uppercase words, considers punctuations, exclamation marks, emojis and other special

characters that help to assess the general mood. The output of the analysis comprises the percentage of negative, neutral and positive words and a compound score for the entire text. [8] [9]

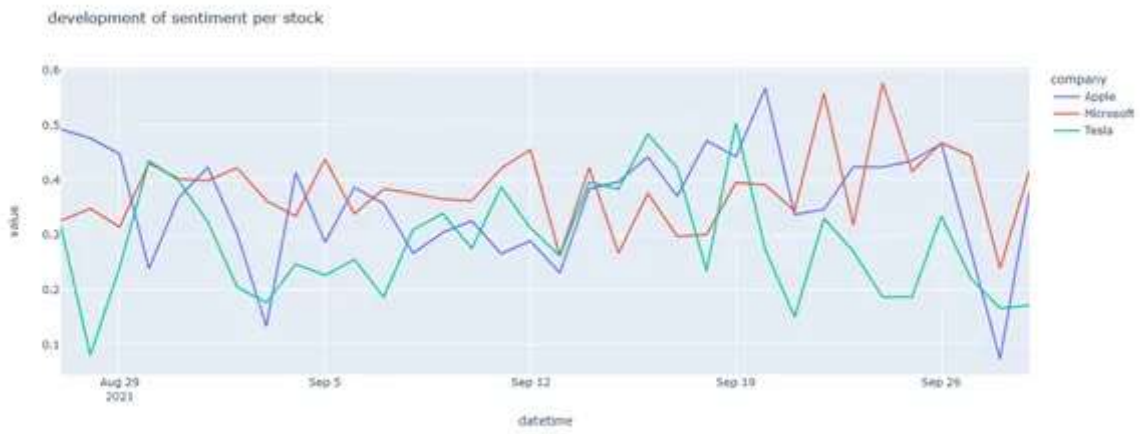
To evaluate the broad market opinion regarding our desired companies, we decided to aggregate the compound sentiment score for each company per day, which delivers us the average market mood.

```
#calculate sentiment
def calculate_sentiment(text):
    sentiment = SentimentIntensityAnalyzer()
```

```
    #update sentiment lexicon
        sentiment.lexicon.update({
            'bullish':0.9,
            'bearish':-0.9,
            'shorters':-0.6,
            'hedgies':-0.4,
            'hold':0.5,
            'buy':0.9,
            'sell':-0.9,
            'loss':-0.9,
            'profit':0.8,
            'rise':0.9,
            'fall':-0.9})
        return sentiment.polarity_scores()

    def calc_sentiment(df_newsapi, columns):
        df_newsapi['sentiment_vader_all'] =
    df_newsapi[columns].fillna('').mapply(calculate_sentiment)
        df_newsapi['sentiment_vader'] =
    df_newsapi['sentiment_vader_all'].apply(lambda _all:
    _all['compound'])
        df_newsapi['sentiment_vader_pos'] =
    df_newsapi['sentiment_vader_all'].apply(lambda _all: _all['pos'])
        df_newsapi['sentiment_vader_neu'] =
    df_newsapi['sentiment_vader_all'].apply(lambda _all: _all['neu'])
        df_newsapi['sentiment_vader_neg'] =
    df_newsapi['sentiment_vader_all'].apply(lambda _all: _all['neg'])
        return df_newsapi

    #Aggregate sentiment by date and ticker and calculate median and
    mean sentiment and number of news:
    df aggregate_sentiment(df_sentiment, groupcol, sentimentcol,
    countcol):
        df_sentiments = df_sentiment\
            .groupby(groupcol)[sentimentcol].mean().reset_index()\
            .rename(columns={sentimentcol:sentimentcol+'_mean'})\
            .merge(
                df_sentiment\
                    .groupby(groupcol)[countcol].count().reset_index()\
                    .rename(columns={countcol:'number_of_posts}),
                on=groupcol)
        return df_sentiments

    #get news sentiment
    def news_sentiment(df_articles, text):
        df_sentiment = calc_sentiment(df_articles, text)
        df_sentiment_agg = aggregate_sentiment(df_sentiment,
    ['company','datetime'], 'sentiment_vader', text)
        return df_sentiment_agg
```

In order to identify huge mood swings on the market that are reflected in rapid sentiment movements, we further calculate the daily percentage sentiment change for each stock and visualize the timeline.

```
#compare the sentiment of the stocks over time
df_sentiment_prepared = df_sent_change\
.pivot_table(index='datetime',columns='company',values='sentiment_va
der_mean')\
.fillna(0)

df_sentiment_prepared = df_sentiment_prepared.reset_index()

#plot sentiment change of stocks over time
fig = px.line(df_sentiment_prepared,
              x='datetime',
              y=df_sentiment_prepared.columns,
              hover_data={'datetime':'%Y-%m-%d'},
              title='development of sentiment per stock')
fig.show()
```



Finally, a wordcloud with the most frequent words used in the articles demonstrates the most relevant topics related to the companies.

```
show_wordcloud(df_news_articles['news'],title='Frequent Words in
News')
```

Frequent Words in News

Trading Strategy

We implemented a Trading Strategy as a combination of mathematical formulas which generates signals based on historical stock price movements and most current trading signals from the news analysis. Therefore, the mentioned technical indicators provide Trading Signals based on mid to long term trends and volatility of the stock price movements whereas the advantage of the sentiment analysis, which lies in indicating short term trends, are also taken into account.

Of course, the combination possibilities with technical and soft indicators are endless and often leads to discussions within the literature. For this article and its sake of simplicity, two distinct strategies are tested: EMA with sentiment analysis and MACD with sentiment analysis.

Moreover, a *stop loss* (SL) and *take profit* (TP) is added, as this is rather standard. It stops a trade from losing too much of the investment and finishes a trade once a certain percentage threshold of profit is realized within a single trade. These indicators depend on the risk management of the person trading. They are also often iterated in different configurations in back-testing, intending to maximize profit. This would often require long computing times, as different configurations of TP and SL are tested with the technical indicators. In this case, a rule of thumb has been taken for TL and SL. The SL is 3% and the TP is 5%, for both long and short trades.

We use historical stock price data to evaluate how good the trading strategy would have performed in the past. To assess the strategy the total profits and losses over the whole periods are summed up resulting in the equity curve. A positive equity curve means that we would have gained money with our Strategy in the past.

Code Implementation of Trading Strategies

```
#EMA
ema_short = df2.ewm(span=20,adjust=False).mean()
close_list = df2[symbol].tolist()
ema_list = ema_short[symbol].tolist()
```

This is a fairly standard implementation for an EMA strategy. The code generates a long or short signal based on the position of the close price, relative to the EMA line. If the close price is higher than the EMA, a long signal is generated. Vice-versa if the close price is below the EMA a short signal is sent.

In this case, the EMA is used as a base signal generator for short and long, and in case a sentiment change is detected, the signal is overwritten.

```
#Get the 12-day EMA of the closing price
k = df2[symbol].ewm(span=12,adjust=False,min_periods=12).mean()
#Get the 26-day EMA of the closing price
d = df2[symbol].ewm(span=12,adjust=False,min_periods=26).mean()

#Subtract the 26-day EMA from the 12-day EMA to get teh MACD
macd = k - d
#Get the 9-day EMA of the MACD for the Signal line
signal = macd.ewm(span=9,adjust=False,min_periods=9).mean()

#Calculate the difference between the MACD - Trigger for the
Convergence/Divergence values

listLongShort = ["No data"] #Since you need at least two days in the
for loop

for i in range(1, len(signal)):
    #If the MACD crosses the signal line upward
    if macd[i] > signal[i] and macd[i-1] <= signal[i-1]:
        listLongShort.append(1)
    #The other way around
    if macd[i] <signal[i] and macd[i-1] >= signal[i-1]:
        listLongShort.append(-1)
    #Do nothing if not crossed
    else:
         listLongShort.append(0)
```

Like the EMA, the MACD is a straightforward implementation, where the trade signal is generated based on the MACD line position relative to the signal line position. MACD above the signal line indicates a long trade is better, below the signal line means a short trade is better suited.

In all cases of EMA or MACD, long or short trade, the sentiment change, as mentioned before is calculated and implemented to overwrite a long or short signal. The sentiment, numerically calculated before, is now used as a threshold: a sentiment change above 2.5 indicates a long signal, below -3 the long trade is not accepted, while in a short trade a sentiment above 2.5 overwrites the short signal and below -1 indicates the short trade to be correct.

The trade is stopped if the signal changes or the TP or SL indicate the trade should be stopped.

Meta Trader

MetaTrader is a multi-asset platform for trading in the Forex market and stock exchange. The Platform provides a python API to exchange trading signals generated by the trading strategy with a MetaTrader Terminal where the Signals are executed as Trades on a sample depot. As a result, the sample depot demonstrates the executions of the strategy on actual stock prices and simulate the ongoing strategy performance.

```
indicator_long = 0
indicator_short = 0
if listLongShort[-1] == 1
    indicator_long = 1
    if sentiment_change[-1] >= 2.5:
         indicator_long = 1
    elif sentiment_change[-1] < -3:
        indicator_long = 0
    else:
        indicator_long = 1
listLongShort[-1] == -1
    indicator_short = 1
    if sentiment_change[-1] >= 2.5:
        indicator_short = 0
    elif sentiment_change[-1] < -1:
        indicator_short = 1
    else:
        indicator_short = 1
if indicator_long == 1 and indicator_short == 0:
    Pair=symbol
    signal='OP_BUY'
    SL=sl_long
    TP=tp_long
    place_order = pd.DataFrame({'':
[Pair+','+signal+','+str(SL)+','+str(TP),',,,']})
#automatically determines file locations string to match our own
system
place_order = place_order.to_csv(mt5_location, header=None,
```

```
index=None, mode='w', sep=' ', quoting=csv.QUOTE_NONE, quotechar="-
",  escapechar="-"))
print(Pair+'Order has opened long')
print("============================")
elif indicator_long == 0and indicator_short == 1:
    Pair=symbol
    signal='OP_SELL'
    SL=sl_short
    TP=tp_short
    place_order = pd.DataFrame({'':
[Pair+','+signal+','+str(SL)+','+str(TP),',,,']})
place_order = place_order.to_csv(mt5_location, header=None,
index=None, mode='w', sep=' ', quoting=csv.QUOTE_NONE, quotechar="-
",  escapechar="-")
print(Pair+'pair has opened short')
print("============================")
x=x+1
print('Iteration'+str(x)+'of'+str(iteration))
time.sleep(600)
```

For this article, the MetaTrader code implementation can be used, to potentially send an order based on the strategies implemented.

The MetaTrader script opens the CSV file generated by the python script and gets the important fields symbol, order type, stop loss and take profit. With this information the MetaTrader script creates an order and sends it to the market. Below you can see code snippets from the MetaTrader script.

```
string FileName = "LastSignal.csv";
string s[10][1000];
string lines = "  ";
string a1="";
string a2="";
string a3="";
string a4="";

int OnInit(){
   int row=0,col=0;
   int rowCnt,colCnt;
   int handle=FileOpen("LastSignal.csv",FILE_READ|FILE_CSV|FILE_ANSI,",");
   int str_size;
   str_size=FileReadInteger(handle,INT_VALUE);




   if(handle==INVALID_HANDLE){
      Alert("Error opening file");

   }

   while(!FileIsEnding(handle)){
      string temp=FileReadString(handle);

      s[col][row]=temp;
```

```
if( a2=="OP_SELL") {
Alert("Sell Order sent");
MqlTradeRequest request={};
request.action=TRADE_ACTION_DEAL;
request.type_filling=SYMBOL_FILLING_FOK;
request.symbol=a1;
request.volume=0.1;
request.sl=a3;
request.tp=a4;

request.type=ORDER_TYPE_SELL;


MqlTradeResult result={};
OrderSend(request,result);
FileDelete(FileName);
Sleep(30);
}
```

Conclusion

This blog investigated the potential implementation of simple technical indicators like EMA, RSI and MACD, with sentiment analysis. The strategy can then be used to send an automatic order to a sample depot.

EMA, RSI and MACD are used separately but in future implementations it would be interesting to backtest different configurations combinations and values of EMA, RSI, MACD, the numerical sentiment, TP and SL with maybe different stock data. Personal experience has taught us, however, that such a backtest requires an immense amount of computing power and time to gain adequate results.

References

[1] RIALTO.AI, “Beginnings of Algorithmic Trading”. 29 June 2018. [Online]. Available: https://medium.com/rialto-ai/beginnings-of-algorithmic-trading-19eccce902a1. [Accessed 25 September 2021].

[2] N. Shiroha, “A Brief History of Algorithmic Trading”. 12 May 2020. [Online]. Available: https://medium.com/the-capital/a-brief-history-of-algorithmic-trading-1ac7e90e153f. [Accessed 25 September 2021].

[3] News API [Online]. Available: https://newsapi.org/. [Accessed 03 September 2021].

[4] M. Lisivick, „newsapi-python 0.2.6 documentation,“ 27 March 2021. [Online]. Available: https://newsapi-python.readthedocs.io/en/latest/api.html. [Accessed 03 September 2021].

[5] J. Chen, „Investopedia,“ 28 April 2021. [Online]. Available: https://www.investopedia.com/terms/e/ema.asp . [Accessed 01 October 2021].

[6] J. Fernando, „Investopedia,“ 04 September 2021. [Online]. Available: https://www.investopedia.com/terms/m/macd.asp. [Accessed 01 October 2021].

[7] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Sentiment_analysis. [Accessed 25 September 2021].

[8] G. Pipis, „python-bloggers,“ MH Corporate basic by MH Themes, 11 October 2020. [Online]. Available: https://python-bloggers.com/2020/10/how-to-run-sentiment-analysis-in-python-using-vader/. [Accessed 25 September 2021].

[9] A. Bajaj, „Analytics Vidhaya,“ 17 June 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/06/vader-for-sentiment-analysis/. [Accessed 25 September 2021].

Algorithmic Trading | Trading | Python | Finance | Sentiment Analysis

Follow

Written by Lumos Student Data Consulting

81 followers · 0 following

Lumos is a student-led Enterprise with a sole focus on Data Science related services based in Vienna. | Feel free to contact us: https://lumos-consulting.at/

No responses yet

Maro E

What are your thoughts?

More from Lumos Student Data Consulting

Lumos Student Data Consulting

The Art of Quarterback Evaluation—NFL and Data Science 101

The broad public understanding of American Football is changing. Data science applications break up old patterns and bring in fresh ideas.

Nov 14, 2020 218

Lumos Student Data Consulting

Mean-Variance Portfolio Optimization using Python

In this article we share a beginner's guide to perform simple Portfolio Optimizations using Python.

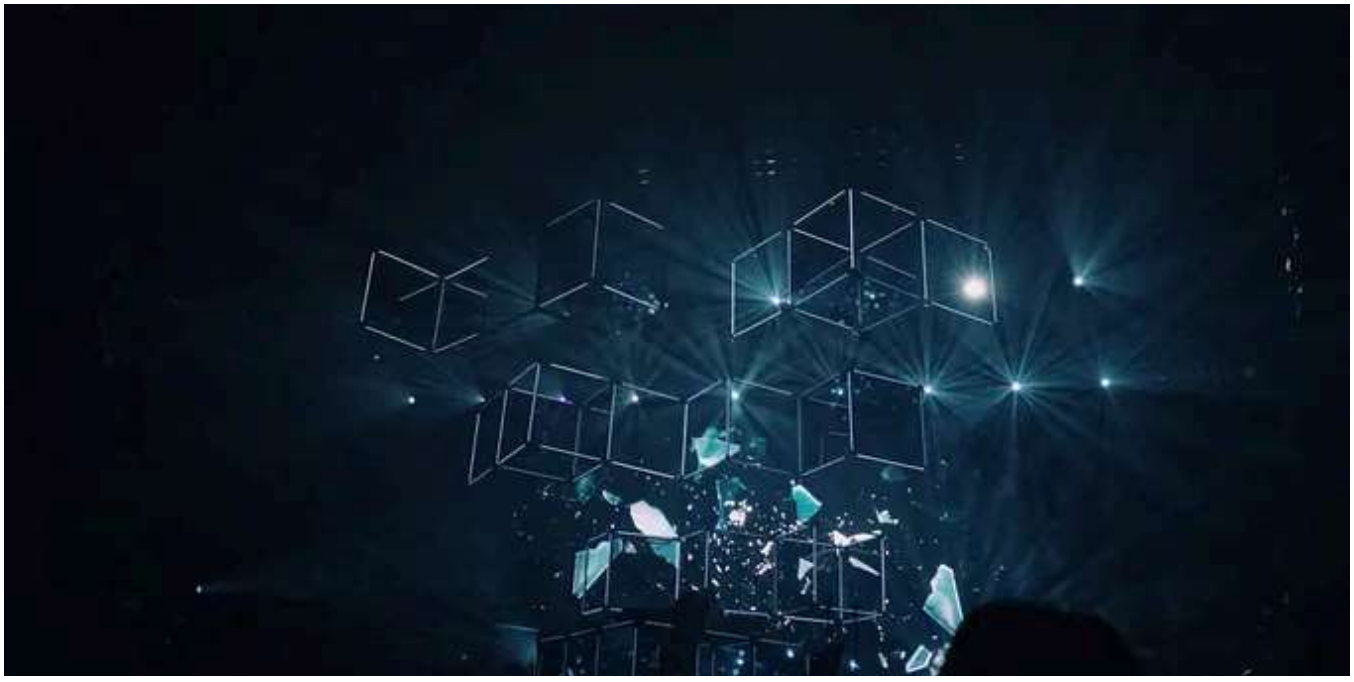
Nov 25, 2020 147

Lumos Student Data Consulting

Uses of AI in the Financial Industry: New Opportunities

The authors present three use cases, explaining how AI can be applied to make impactful contributions in the financial industry

Apr 9, 2022



Lumos Student Data Consulting