Example : HMM of Financial Time Series using hmmlearn

In [1]:

```
import matplotlib.pyplot as plt
from scipy import stats
import numpy as np
from hmmlearn import hmm
import math
import os
import urllib
import urllib.request as ur
import datetime
import warnings

from numpy.lib.stride_tricks import as_strided
import scipy
from scipy.io import wavfile
from sklearn.model_selection import StratifiedShuffleSplit
import mpl_finance
import itertools

from matplotlib import cm, pyplot as plt
from matplotlib.dates import YearLocator, MonthLocator

from mpl_finance import candlestick_ohlc
import pandas as pd
from sklearn.model_selection import train_test_split
from tqdm import tqdm
from hmmlearn import hmm

plt.style.use('ggplot')
```

In [2]:

```
test_size=0.33
n_latency_days=10
n_hidden_states=4
n_steps_frac_change=50
n_steps_frac_high=10
n_steps_frac_low=10
```

In [3]:

```
data = pd.read_csv('yahoofinance-INTC-19950101-20040412.csv')
```

In [4]:

```
train_data, test_data = train_test_split(data, test_size=test_size, shuffle=Fals
e)
```

In [5]:

```
def extract_features(data):
        open_price = np.array(data['Open'])
        close_price = np.array(data['Close'])
        high_price = np.array(data['High'])
        low_price = np.array(data['Low'])
        # Compute the fraction change in close, high and low prices which would
be used as features
        frac_change = (close_price - open_price) / open_price
        frac_high = (high_price - open_price) / open_price
        frac_low = (open_price - low_price) / open_price
        return np.column_stack((frac_change, frac_high, frac_low))
```

In [6]:

```
train_features = extract_features(train_data)
```

In [7]:

```
def compute_all_possible_outcomes(n_steps_frac_change,n_steps_frac_high, n_steps
_frac_low):
        frac_change_range = np.linspace(-0.1, 0.1, n_steps_frac_change)
        frac_high_range = np.linspace(0, 0.1, n_steps_frac_high)
        frac_low_range = np.linspace(0, 0.1, n_steps_frac_low)
        possible_outcomes = np.array(list(itertools.product(frac_change_range, f
rac_high_range, frac_low_range)))
        return possible_outcomes
```

In [8]:

```
def get_most_probable_outcome(day_index, possible_outcomes):
        previous_data_start_index = max(0, day_index - n_latency_days)
        previous_data_end_index = max(0, day_index - 1)
        previous_data = test_data.iloc[previous_data_end_index: previous_data_st
art_index]
        previous_data_features = extract_features(previous_data)
        outcome_score = []
        most_probable_outcome =[]
        for possible_outcome in possible_outcomes:
            total_data = np.row_stack((previous_data_features, possible_outcome)
)
            outcome_score.append(intc_hmm.score(total_data))
        maxscore_index = np.argmax(outcome_score)
        most_probable_outcome = possible_outcomes[maxscore_index]
        #print(most_probable_outcome,maxscore_index )
        return most_probable_outcome
```

In [9]:

```
def predict_close_price(day_index):
        open_price = test_data.iloc[day_index]['Open']
        predicted_frac_change= get_most_probable_outcome(day_index, possible_out
comes)
        #print(predicted_frac_change)
        close_price_predict = (open_price * (1 + predicted_frac_change))
        return close_price_predict
```

In [10]:

```
def predict_close_prices_for_days(test_data, possible_outcomes, days, with_plot=
True):
        predicted_close_prices = []
        for day_index in tqdm(range(days)):
            cpp = predict_close_price(day_index)
            #print(cpp)
            predicted_close_prices.append(predict_close_price(day_index))
            #print(day_index, predicted_close_prices)

        #if with_plot:
        test_data = test_data[0: days]
        days = np.array(test_data['Date'], dtype="datetime64[ms]")
        actual_close_prices = test_data['Close']

        #fig = plt.figure()
        fig, ax = plt.subplots(figsize=(10,5))
        #axes = fig.add_subplot(111)
        ax.plot(days, actual_close_prices, 'bo-', label="actual")
        ax.plot(days, predicted_close_prices, 'r+-', label="predicted")
        ax.set_title('INTC Stock Movement')

        fig.autofmt_xdate()

        plt.legend()
        plt.show()




        return predicted_close_prices
```

In [11]:

```
intc_hmm =  hmm.GaussianHMM(n_components=n_hidden_states)
```

In [12]:

```
intc_hmm.fit(train_features)
```

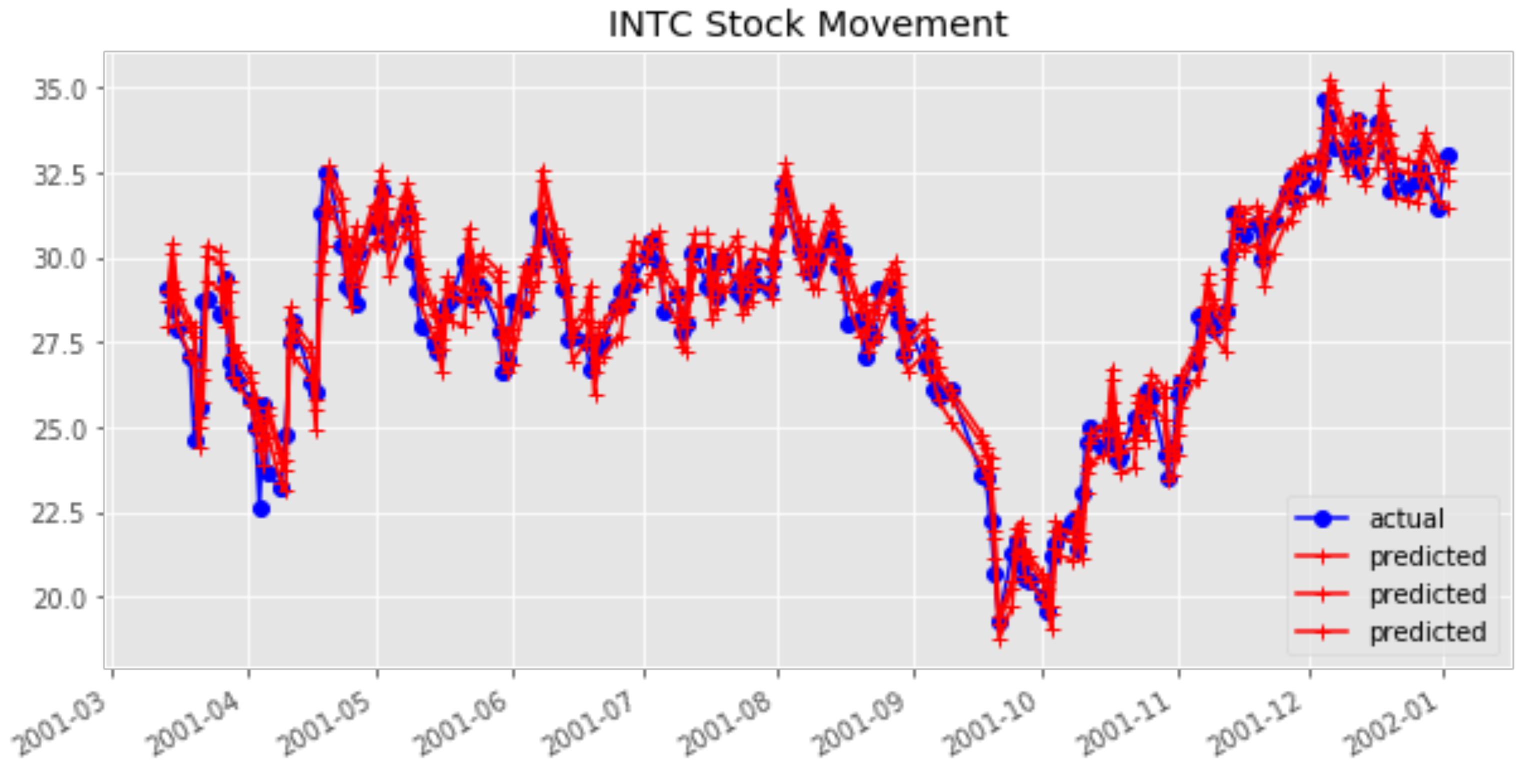
Out[12]:

```
GaussianHMM(algorithm='viterbi', covariance_type='diag', covars_prio
r=0.01,
      covars_weight=1, init_params='stmc', means_prior=0, means_weig
ht=0,
      min_covar=0.001, n_components=4, n_iter=10, params='stmc',
      random_state=None, startprob_prior=1.0, tol=0.01, transmat_pri
or=1.0,
      verbose=False)
```

In [13]:

```
possible_outcomes = compute_all_possible_outcomes(n_steps_frac_change,n_steps_frac_high, n_steps_frac_low)
predict_close_prices_for_days(test_data, possible_outcomes,200, with_plot=True)
```

```
100%|██████████| 200/200 [08:54<00:00,  2.06s/it]
```



Out[13]:

```
[array([27.96964286, 28.69027778, 29.00555556]),
 array([29.325     , 30.08055556, 30.41111111]),
 array([28.03125   , 28.75347222, 29.06944444]),
 array([27.10714286, 27.80555556, 28.11111111]),
 array([26.98392857, 27.67916667, 27.98333333]),
 array([24.39642857, 25.025     , 25.3       ]),
 array([25.75178571, 26.41527778, 26.70555556]),
 array([29.26339286, 30.01736111, 30.34722222]),
 array([29.07857143, 29.82777778, 30.15555556]),
 array([27.96964286, 28.69027778, 29.00555556]),
 array([28.27767857, 29.00625   , 29.325     ]),
 array([26.49107143, 27.17361111, 27.47222222]),
 array([26.24464286, 26.92083333, 27.21666667]),
 array([25.69017857, 26.35208333, 26.64166667]),
 array([24.95089286, 25.59375   , 25.875     ]),
 array([24.33482143, 24.96180556, 25.23611111]),
 array([23.90357143, 24.51944444, 24.78888889]),
 array([24.70446429, 25.34097222, 25.61944444]),
 array([23.35157241, 23.95322323, 24.21644547]),
 array([23.14457143, 23.74088889, 24.00177778]),
 array([27.50142857, 28.21      , 28.52      ]),
 array([27.0677133, 27.7651101, 28.0702212]),
 array([26.38757143, 27.06744444, 27.36488889]),
 array([24.88928571, 25.53055556, 25.81111111]),
 array([28.80257044, 29.54466566, 29.86933231]),
 array([30.35014384, 31.13211212, 31.47422324]),
```

```
array([31.51328473, 32.32522121, 32.68044342]),
array([30.62614286, 31.41522222, 31.76044444]),
array([29.5122867 , 30.27266768, 30.60533436]),
array([28.58571429, 29.32222222, 29.64444444]),
array([29.79814286, 30.56588889, 30.90177778]),
array([29.17714286, 29.92888889, 30.25777778]),
array([30.40928571, 31.19277778, 31.53555556]),
array([30.33042857, 31.11188889, 31.45377778]),
array([31.43442759, 32.24433232, 32.59866564]),
array([30.69514187, 31.48599899, 31.83199898]),
array([29.47285714, 30.23222222, 30.56444444]),
array([30.60642759, 31.39499899, 31.73999898]),
array([31.04014286, 31.83988889, 32.18977778]),
array([30.54728571, 31.33433333, 31.67866667]),
array([30.04457143, 30.81866667, 31.15733333]),
array([28.6152867 , 29.35255657, 29.67511213]),
array([27.64928473, 28.36166566, 28.67333231]),
array([27.22542956, 27.9268899 , 28.2337788 ]),
array([26.61428571, 27.3       , 27.6       ]),
array([28.36885813, 29.09977879, 29.41955658]),
array([28.12242956, 28.84700101, 29.16400102]),
array([27.96471527, 28.68522323, 29.00044547]),
array([29.75871527, 30.52544546, 30.86088991]),
array([28.95042956, 29.69633434, 30.02266769]),
array([28.428     , 29.16044444, 29.48088889]),
array([29.01942956, 29.76711212, 30.09422324]),
array([28.54628473, 29.28177677, 29.60355453]),
array([26.94942857, 27.64377778, 27.94755556]),
array([26.64385813, 27.33033434, 27.63066769]),
array([26.87057143, 27.56288889, 27.86577778]),
array([28.70400099, 29.44355657, 29.76711213]),
array([28.44771527, 29.18066768, 29.50133436]),
array([29.0292867 , 29.77722323, 30.10444547]),
array([29.28557044, 30.0401101 , 30.3702212 ]),
array([31.43442759, 32.24433232, 32.59866564]),
array([29.739     , 30.50522222, 30.84044444]),
array([29.27571527, 30.03000101, 30.36000102]),
array([29.45314187, 30.21199899, 30.54399898]),
array([28.15199901, 28.87733232, 29.19466564]),
array([26.94942857, 27.64377778, 27.94755556]),
array([27.47185813, 28.17966768, 28.48933436]),
array([28.12242956, 28.84700101, 29.16400102]),
array([25.93414187, 26.60233232, 26.89466564]),
array([26.95928571, 27.65388889, 27.95777778]),
array([27.07757044, 27.77522121, 28.08044342]),
array([27.6       , 28.31111111, 28.62222222]),
array([27.68871429, 28.40211111, 28.71422222]),
array([28.58571429, 29.32222222, 29.64444444]),
array([28.81242857, 29.55477778, 29.87955556]),
array([29.38414187, 30.14122121, 30.47244342]),
array([29.15742857, 29.90866667, 30.23733333]),
array([29.5122867 , 30.27266768, 30.60533436]),
array([29.65028571, 30.41422222, 30.74844444]),
```

```
array([28.7237133 , 29.46377677, 29.78755453]),
array([28.12242956, 28.84700101, 29.16400102]),
array([27.39300099, 28.09877879, 28.40755658]),
array([27.255     , 27.95722222, 28.26444444]),
array([28.93071429, 29.67611111, 30.00222222]),
array([29.58128571, 30.34344444, 30.67688889]),
array([29.60100099, 30.36366768, 30.69733436]),
array([28.16185714, 28.88744444, 29.20488889]),
array([28.45757241, 29.19077879, 29.51155658]),
array([29.16728571, 29.91877778, 30.24755556]),
array([28.98985714, 29.73677778, 30.06355556]),
array([29.54185616, 30.30299899, 30.63599898]),
array([28.29985616, 29.02899899, 29.34799898]),
array([28.55614187, 29.29188788, 29.61377676]),
array([28.68428571, 29.42333333, 29.74666667]),
array([29.18700099, 29.93900101, 30.26800102]),
array([29.08842857, 29.83788889, 30.16577778]),
array([28.74342857, 29.484     , 29.808     ]),
array([30.18257241, 30.96022323, 31.30044547]),
array([30.96128571, 31.759     , 32.108     ]),
array([31.59214187, 32.4061101 , 32.7622212 ]),
array([29.99528571, 30.76811111, 31.10622222]),
array([29.52214384, 30.28277879, 30.61555658]),
array([29.95585616, 30.72766566, 31.06533231]),
array([29.07857143, 29.82777778, 30.15555556]),
array([29.09828571, 29.848     , 30.176     ]),
array([30.23185714, 31.01077778, 31.35155556]),
array([30.27128473, 31.05122121, 31.39244342]),
array([29.84742956, 30.61644546, 30.95288991]),
array([28.98985714, 29.73677778, 30.06355556]),
array([28.75328571, 29.49411111, 29.81822222]),
array([27.669     , 28.38188889, 28.69377778]),
array([27.92528571, 28.64477778, 28.95955556]),
array([27.19585714, 27.89655556, 28.20311111]),
array([27.49157044, 28.19988788, 28.50977676]),
array([27.64928473, 28.36166566, 28.67333231]),
array([28.60542857, 29.34244444, 29.66488889]),
array([28.80257044, 29.54466566, 29.86933231]),
array([28.44771527, 29.18066768, 29.50133436]),
array([27.10714286, 27.80555556, 28.11111111]),
array([26.634     , 27.32022222, 27.62044444]),
array([27.16628473, 27.86622121, 28.17244342]),
array([26.55514384, 27.23933434, 27.53866769]),
array([26.37771429, 27.05733333, 27.35466667]),
array([25.78628571, 26.45066667, 26.74133333]),
array([25.1652867 , 25.81366768, 26.09733436]),
array([23.91342857, 24.52955556, 24.79911111]),
array([23.53885616, 24.14533232, 24.41066564]),
array([23.22342759, 23.82177677, 24.08355453]),
array([21.15342759, 21.69844343, 21.93688787]),
array([18.73842857, 19.22122222, 19.43244444]),
array([19.71428571, 20.22222222, 20.44444444]),
array([21.25199901, 21.79955454, 22.03911009]),
```

```
array([21.4097133 , 21.96133232, 22.20266564]),
array([20.61128571, 21.14233333, 21.37466667]),
array([20.44371429, 20.97044444, 21.20088889]),
array([19.92128473, 20.43455454, 20.65911009]),
array([19.71428571, 20.22222222, 20.44444444]),
array([19.02428473, 19.51444343, 19.72888787]),
array([21.41957143, 21.97144444, 22.21288889]),
array([21.20271429, 21.749     , 21.988     ]),
array([21.07457044, 21.61755454, 21.85511009]),
array([21.72514384, 22.2848899 , 22.5297788 ]),
array([21.12385714, 21.66811111, 21.90622222]),
array([23.05585616, 23.64988788, 23.90977676]),
array([23.9627133, 24.5801101, 24.8502212]),
array([24.14014286, 24.76211111, 25.03422222]),
array([24.17957241, 24.80255657, 25.07511213]),
array([25.76657044, 26.43044343, 26.72088787]),
array([24.24857143, 24.87333333, 25.14666667]),
array([23.64728571, 24.25655556, 24.52311111]),
array([23.77542956, 24.38800101, 24.65600102]),
array([25.05685714, 25.70244444, 25.98488889]),
array([24.83014384, 25.4698899 , 25.7497788 ]),
array([24.64285714, 25.27777778, 25.55555556]),
array([25.63842857, 26.299     , 26.588     ]),
array([25.23428571, 25.88444444, 26.16888889]),
array([23.43042857, 24.03411111, 24.29822222]),
array([23.58814286, 24.19588889, 24.46177778]),
array([24.17957241, 24.80255657, 25.07511213]),
array([25.5792867 , 26.23833434, 26.52666769]),
array([26.36785714, 27.04722222, 27.34444444]),
array([26.4072867 , 27.08766768, 27.38533436]),
array([27.51128571, 28.22011111, 28.53022222]),
array([28.45757241, 29.19077879, 29.51155658]),
array([27.77742857, 28.49311111, 28.80622222]),
array([27.21557241, 27.91677879, 28.22355658]),
array([28.64485616, 29.38288788, 29.70577676]),
array([30.0347133 , 30.80855454, 31.14711009]),
array([30.37971429, 31.16244444, 31.50488889]),
array([30.19242759, 30.97033232, 31.31066564]),
array([30.36985616, 31.15233232, 31.49466564]),
array([30.27128473, 31.05122121, 31.39244342]),
array([29.11800099, 29.86822323, 30.19644547]),
array([30.14314286, 30.91977778, 31.25955556]),
array([30.981     , 31.77922222, 32.12844444]),
array([31.05985714, 31.86011111, 32.21022222]),
array([31.464     , 32.27466667, 32.62933333]),
array([31.533     , 32.34544444, 32.70088889]),
array([31.76957143, 32.58811111, 32.94622222]),
array([31.82871527, 32.64877879, 33.00755658]),
array([31.74000099, 32.55777879, 32.91555658]),
array([32.58771527, 33.42733434, 33.79466769]),
array([34.00714286, 34.88333333, 35.26666667]),
array([33.672     , 34.53955556, 34.91911111]),
array([32.42014187, 33.25544343, 33.62088787]),
```

```
 array([32.87356946, 33.72055353, 34.09110907]),
 array([32.81442956, 33.6598899 , 34.0297788 ]),
 array([32.80457044, 33.64977677, 34.01955453]),
 array([32.12442857, 32.95211111, 33.31422222]),
 array([32.62714089, 33.46777576, 33.83555351]),
 array([33.66214483, 34.52944647, 34.90889093]),
 array([32.85385911, 33.70033536, 34.07066871]),
 array([32.38071231, 33.21499798, 33.57999796]),
 array([31.75971527, 32.57800101, 32.93600102]),
 array([31.70057143, 32.51733333, 32.87466667]),
 array([31.59214187, 32.4061101 , 32.7622212 ]),
 array([31.947     , 32.77011111, 33.13022222]),
 array([32.46942759, 33.30599899, 33.67199898]),
 array([31.69071626, 32.50722424, 32.86444649]),
 array([31.44428571, 32.25444444, 32.60888889])]
```

In [ ]:

In [ ]: