chapter1

December 30, 2020

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

(BSM model)

[ ]:
```
#

import numpy as np

S0 = 100 #
K = 105 #
T = 1.0 #
r = 0.05 #
sigma = 0.2 #
I = 100000 #

#

z = np.random.standard_normal(I) #
ST = S0 * np.exp((r - 0.5 * sigma ** 2)*T + sigma*np.sqrt(T)*z) #
hT = np.maximum(ST-K, 0) #
C0 = np.exp(-r*T)*np.sum(hT)/T

print( "Value of the European Call option %5.3f" %  C0 )
```

```
Value of the European Call option 795633.402
```

Plot the index level data and the volatility results

[ ]:
```
import numpy as np
import pandas as pd
from pylab import plt, mpl

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

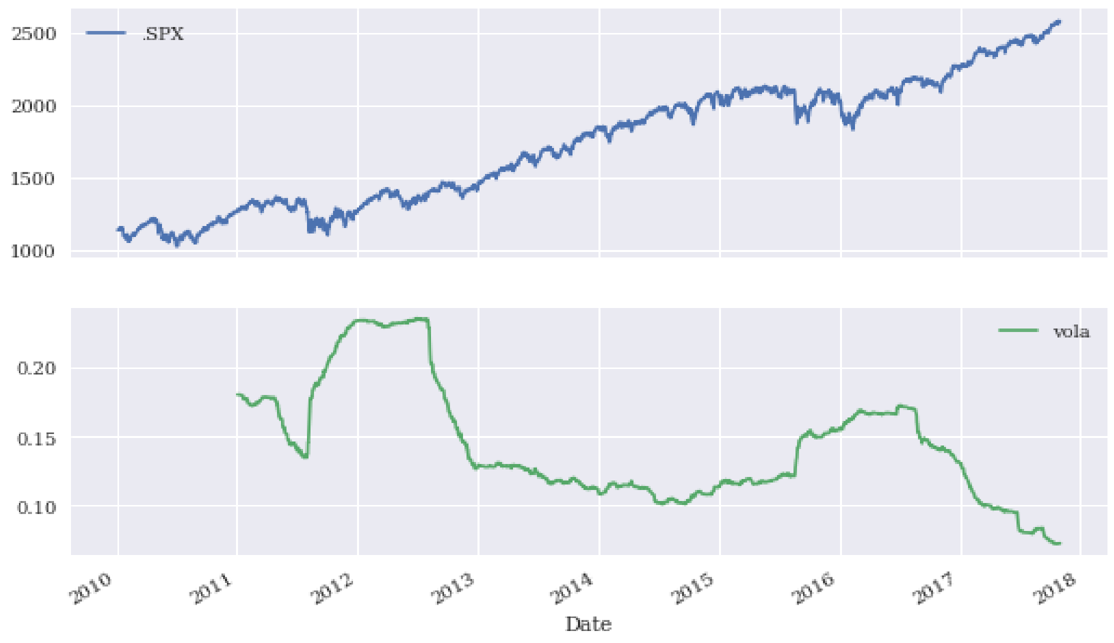
```
mpl.rcParams['font.family'] = 'serif'
%matplotlib inline

data = pd.read_csv('/content/drive/My Drive/Python_for_Finance/data/
 ↪tr_eikon_eod_data.csv',
                   index_col=0, parse_dates=True)
data = pd.DataFrame(data['.SPX'])
data.dropna(inplace=True)
data.info()

data['rets'] = np.log(data / data.shift(1))
data['vola'] = data['rets'].rolling(252).std()*np.sqrt(252)

data[['.SPX', 'vola']].plot(subplots=True, figsize=(10,6));
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1972 entries, 2010-01-04 to 2017-10-31
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   .SPX    1972 non-null   float64
dtypes: float64(1)
memory usage: 30.8 KB
```



PERFORMANCE COMPUTING WITH PYTHON
( )

[ ]:
```
import math
loops = 250000
a = range(1, loops)
def f(x):
    return 3 * math.log(x) + math.cos(x) ** 2
%timeit r = [f(x) for x in a]
```

```
10 loops, best of 3: 112 ms per loop
```

[ ]:
```
import numpy as np
a = np.arange(1, loops)
%timeit r = 3 *np.log(a) + np.cos(a)**2
```

```
100 loops, best of 3: 12.7 ms per loop
```

[ ]:
```
import numexpr as ne
ne.set_num_threads(1)
f = '3 * log(a) + cos(a) ** 2'
%timeit r = ne.evaluate(f)
```

```
100 loops, best of 3: 11.4 ms per loop
```

[ ]:
```
ne.set_num_threads(4)
%timeit r=ne.evaluate(f)
```

```
100 loops, best of 3: 9.36 ms per loop
```

AI-First Finance
scikit-learn

[ ]:
```
!pip install sklearn
```

```
Requirement already satisfied: sklearn in /usr/local/lib/python3.6/dist-packages
(0.0)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.6/dist-
packages (from sklearn) (0.22.2.post1)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-
packages (from scikit-learn->sklearn) (1.0.0)
Requirement already satisfied: scipy>=0.17.0 in /usr/local/lib/python3.6/dist-
packages (from scikit-learn->sklearn) (1.4.1)
Requirement already satisfied: numpy>=1.11.0 in /usr/local/lib/python3.6/dist-
packages (from scikit-learn->sklearn) (1.19.4)
```

[ ]:
```
import numpy as np
import pandas as pd
```

[ ]:
```
data = pd.read_csv('/content/drive/My Drive/Python_for_Finance/data/
 ↪tr_eikon_eod_data.csv',
                      index_col=0, parse_dates=True)
data = pd.DataFrame(data['AAPL.O'])
data['Returns'] = np.log(data/data.shift())
data.dropna(inplace=True)
```

[ ]:
```
lags = 6
```

[ ]:
```
cols = []
for lag in range(1, lags + 1):
    col = 'lag_{}'.format(lag)
    data[col] = np.sign(data['Returns'].shift(lag))
    cols.append(col)
data.dropna(inplace=True)
```

[ ]:
```
from sklearn.svm import SVC
```

[ ]:
```
model = SVC(gamma='auto')
```

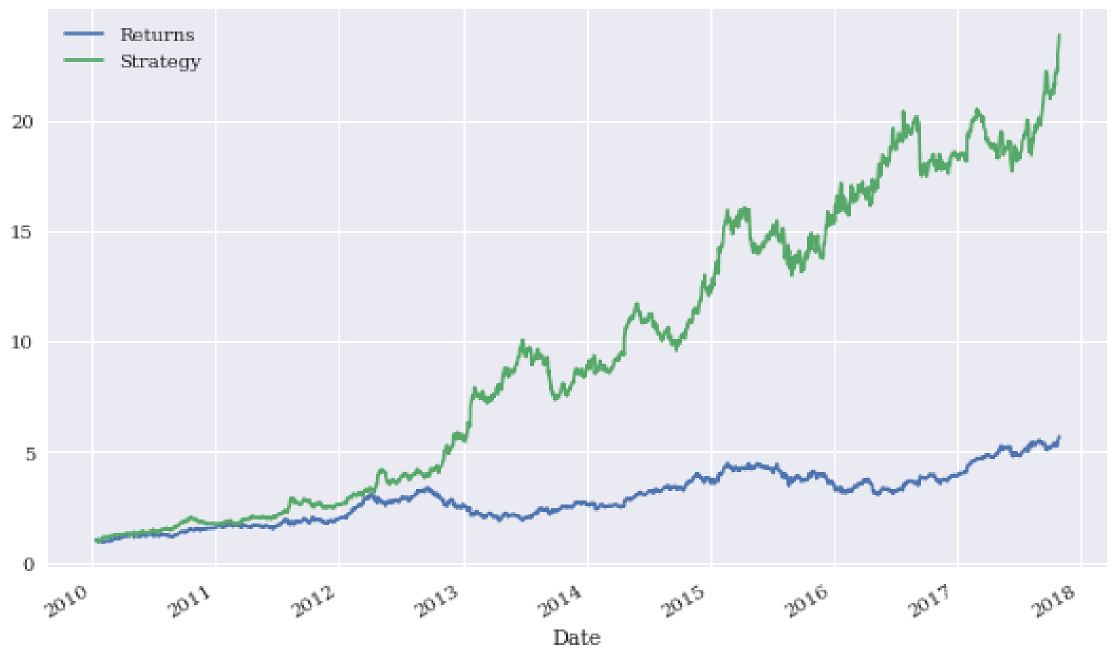
[ ]:
```
model.fit(data[cols], np.sign(data['Returns']))
```

[ ]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

[ ]:
```
data['Prediction'] = model.predict(data[cols])
```

[ ]:
```
data['Strategy'] = data['Prediction'] * data['Returns']
```

[ ]:
```
data[['Returns', 'Strategy']].cumsum().apply(np.exp).plot(figsize=(10, 6));
```


Returns
Strategy
Date