教程 | 如何用Python和机器学习炒股赚钱?

2017-07-04 机器之心

选自Hackernoon

作者：Gaëtan Rickter

机器之心编译

参与：熊猫

相信很多人都想过让人工智能来帮你赚钱，但到底该如何做呢？瑞士日内瓦的一位金融数据顾问 Gaëtan Rickter 近日发表文章介绍了他利用 Python 和机器学习来帮助炒股的经验，其最终成果的收益率跑赢了长期处于牛市的标准普尔 500 指数。虽然这篇文章并没有将他的方法完全彻底公开，但已公开的内容或许能给我们带来如何用人工智能炒股的启迪。机器之心对本文进行了编译介绍，代码详情请访问原文。

我终于跑赢了标准普尔 500 指数 10 个百分点！听起来可能不是很多，但是当我们处理的是大量流动性很高的资本时，对冲基金的利润就相当可观。更激进的做法还能得到更高的回报。

这一切都始于我阅读了 Gur Huberman 的一篇题为《Contagious Speculation and a Cure for Cancer: A Non-Event that Made Stock Prices Soar》的论文。该研究描述了一件发生在 1998 年的涉及到一家上市公司 EntreMed（当时股票代码是 ENMD）的事件：

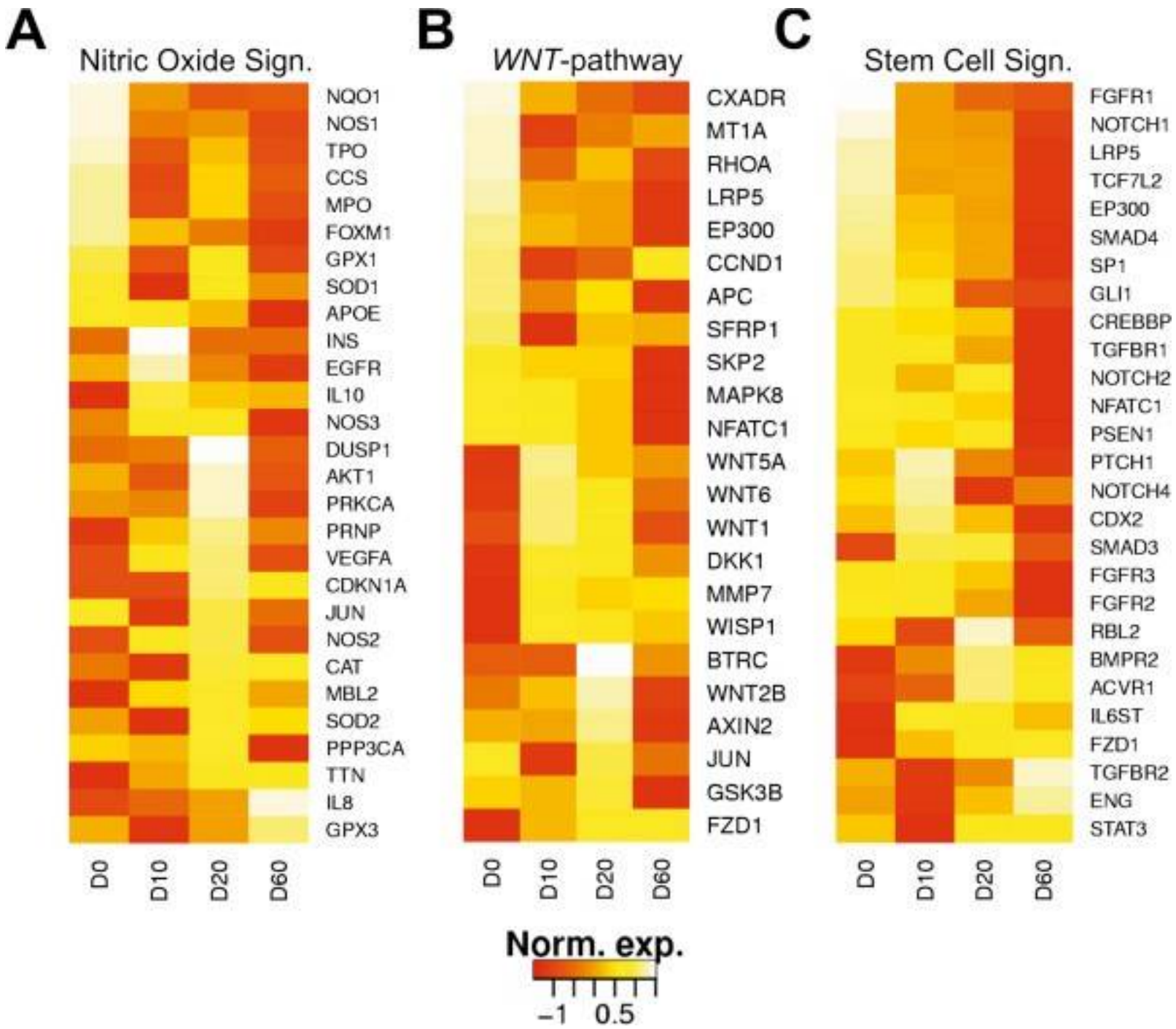
「星期天《纽约时报》上发表的一篇关于癌症治疗新药开发潜力的文章导致 EntreMed 的股价从周五收盘时的 12.063 飙升至 85，在周一收盘时接近 52。在接下来的三周，它的收盘价都在 30 以上。这股投资热情也让其它生物科技股得到了溢价。但是，这个癌症研究方面的可能突破在至少五个月前就已经被 Nature 期刊和各种流行的报纸报道过了，其中甚至包括《泰晤士报》！因此，仅仅是热情的公众关注就能引发股价的持续上涨，即便实际上并没有出现真正的新信息。」

在研究者给出的许多有见地的观察中，其中有一个总结很突出：

> *「（股价）运动可能会集中于有一些共同之处的股票上，但这些共同之处不一定要是经济基础。」*

我就想，能不能基于通常所用的指标之外的其它指标来划分股票。我开始在数据库里面挖掘，几周之后我发现了一个，其包含了一个分数，描述了股票和元素周期表中的元素之间的「已知和隐藏关系」的强度。

我有计算基因组学的背景，这让我想起了基因和它们的细胞信号网络之间的关系是如何地不为人所知。但是，当我们分析数据时，我们又会开始看到我们之前可能无法预测的新关系和相关性。



选择出的涉及细胞可塑性、生长和分化的信号通路的基因的表达模式

和基因一样，股票也会受到一个巨型网络的影响，其中各个因素之间都有或强或弱的隐藏关系。其中一些影响和关系是可以预测的。

我的一个目标是创建长的和短的股票聚类，我称之为「篮子聚类（basket clusters）」，我可以将其用于对冲或

单纯地从中获利。这需要使用一个无监督机器学习方法来创建股票的聚类，从而使这些聚类之间有或强或弱的关系。这些聚类将会翻倍作为我的公司可以交易的股票的「篮子（basket）」。

首先我下载了一个数据集：http://54.174.116.134/recommend/datasets/supercolumns-elements-08.html，这个数据集基于元素周期表中的元素和上市公司之间的关系。

然后我使用了 Python 和一些常用的机器学习工具——scikit-learn、numpy、pandas、matplotlib 和 seaborn，我开始了解我正在处理的数据集的分布形状。为此我参考了一个题为《Principal Component Analysis with KMeans visuals》的 Kaggle Kernel：https://www.kaggle.com/arthurtok/principal-component-analysis-with-kmeans-visuals

```
import numpy as np
import pandas as pd
from sklearn.decomposition
import PCA
from sklearn.cluster
import KMeans
import matplotlib.pyplot as plt
import seaborn as sb

np.seterr(divide='ignore', invalid='ignore')
# Quick way to test just a few column features

# stocks = pd.read_csv('supercolumns-elements-nasdaq-nyse-otcbb-general-UPDAT
E-2017-03-01.csv', usecols=range(1,16))

stocks = pd.read_csv('supercolumns-elements-nasdaq-nyse-otcbb-general-UPDATE-
2017-03-01.csv')

print(stocks.head())

str_list = []
for colname, colvalue in stocks.iteritems():
if type(colvalue[1]) == str:
        str_list.append(colname)
# Get to the numeric columns by inversion

num_list = stocks.columns.difference(str_list)
```

```
stocks_num = stocks[num_list]

print(stocks_num.head())
```

输出：简单看看前面 5 行：

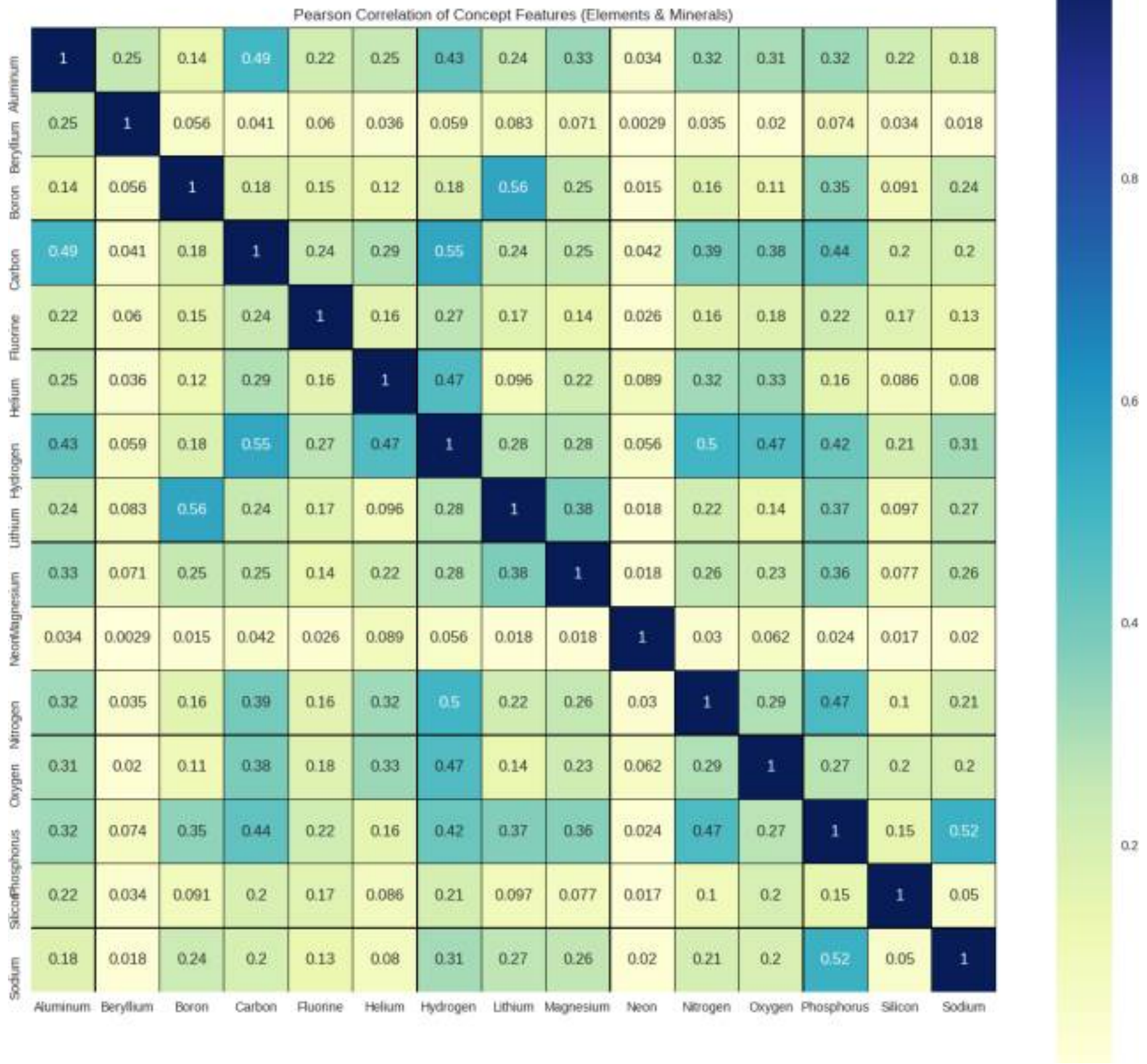
```
zack@twosigma-Dell-Precision-M3800:/home/zack/hedge_pool/baskets/hcluster$ ./hidden_relationships.py
  Symbol_update-2017-04-01  Hydrogen   Helium  Lithium  Beryllium  Boron  \
0                        A       0.0  0.00000      0.0        0.0    0.0
1                       AA       0.0  0.00000      0.0        0.0    0.0
2                     AAAP       0.0  0.00461      0.0        0.0    0.0
3                      AAC       0.0  0.00081      0.0        0.0    0.0
4                    AACAY       0.0  0.00000      0.0        0.0    0.0

     Carbon  Nitrogen    Oxygen  Fluorine      ...       Fermium  Mendelevium  \
0  0.006632       0.0  0.007576       0.0      ...      0.000000     0.079188
1  0.000000       0.0  0.000000       0.0      ...      0.000000     0.000000
2  0.000000       0.0  0.000000       0.0      ...      0.135962     0.098090
3  0.000000       0.0  0.018409       0.0      ...      0.000000     0.000000
4  0.000000       0.0  0.000000       0.0      ...      0.000000     0.000000

   Nobelium  Lawrencium  Rutherfordium  Dubnium  Seaborgium  Bohrium  Hassium  \
0  0.197030      0.1990         0.1990      0.0         0.0      0.0      0.0
1  0.000000      0.0000         0.0000      0.0         0.0      0.0      0.0
2  0.244059      0.2465         0.2465      0.0         0.0      0.0      0.0
3  0.000000      0.0000         0.0000      0.0         0.0      0.0      0.0
4  0.000000      0.0000         0.0000      0.0         0.0      0.0      0.0

   Meitnerium
0         0.0
1         0.0
2         0.0
3         0.0
4         0.0

[5 rows x 110 columns]
   Actinium  Aluminum  Americium  Antimony     Argon   Arsenic  Astatine  \
0  0.000000       0.0        0.0  0.002379  0.047402  0.018913       0.0
1  0.000000       0.0        0.0  0.000000  0.000000  0.000000       0.0
2  0.004242       0.0        0.0  0.001299  0.000000  0.000000       0.0
3  0.000986       0.0        0.0  0.003378  0.000000  0.000000       0.0
4  0.000000       0.0        0.0  0.000000  0.000000  0.000000       0.0

   Barium  Berkelium  Beryllium     ...       Tin  Titanium  Tungsten   Uranium  \
0     0.0   0.000000        0.0     ...       0.0  0.002676       0.0  0.000000
1     0.0   0.000000        0.0     ...       0.0  0.000000       0.0  0.000000
2     0.0   0.141018        0.0     ...       0.0  0.000000       0.0  0.004226
3     0.0   0.000000        0.0     ...       0.0  0.000000       0.0  0.004086
4     0.0   0.000000        0.0     ...       0.0  0.000000       0.0  0.000000

   Vanadium  Xenon  Ytterbium   Yttrium      Zinc  Zirconium
0  0.000000    0.0        0.0  0.000000  0.000000        0.0
1  0.000000    0.0        0.0  0.000000  0.000000        0.0
2  0.002448    0.0        0.0  0.018806  0.008758        0.0
3  0.001019    0.0        0.0  0.000000  0.007933        0.0
4  0.000000    0.0        0.0  0.000000  0.000000        0.0

[5 rows x 109 columns]
zack@twosigma-Dell-Precision-M3800:/home/zack/hedge_pool/baskets/hcluster$
```

概念特征的皮尔逊相关性（Pearson Correlation）。在这里案例中，是指来自元素周期表的矿物和元素：

```
stocks_num = stocks_num.fillna(value=0, axis=1)

X = stocks_num.values
```

```
from sklearn.preprocessing import StandardScaler
X_std = StandardScaler().fit_transform(X)

f, ax = plt.subplots(figsize=(12, 10))
plt.title('Pearson Correlation of Concept Features (Elements & Minerals)')
# Draw the heatmap using seaborn

sb.heatmap(stocks_num.astype(float).corr(),linewidths=0.25,vmax=1.0, square=T
rue, cmap="YlGnBu", linecolor='black', annot=True)
sb.plt.show()
```

输出：（这个可视化例子是在前 16 个样本上运行得到的）。看到元素周期表中的元素和上市公司关联起来真的很有意思。在某种程度时，我想使用这些数据基于公司与相关元素或材料的相关性来预测其可能做出的突破。



测量「已解释方差（Explained Variance）」和主成分分析（PCA）

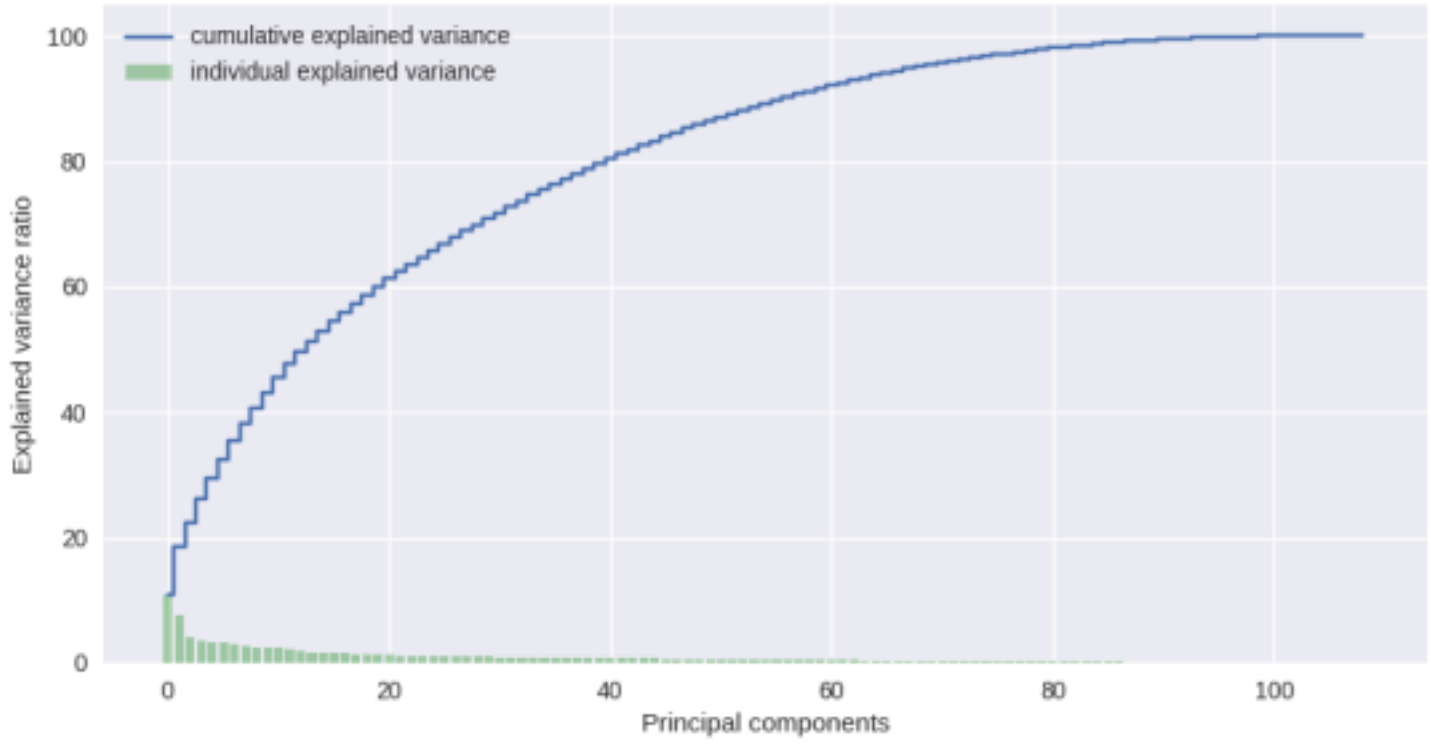
已解释方差=总方差-残差方差（explained variance = total variance - residual variance)。应该值得关注的 PCA 投射组件的数量可以通过已解释方差度量（Explained Variance Measure）来引导。Sebastian Raschka 的关于 PCA 的文章对此进行了很好的描述，参阅：

http://sebastianraschka.com/Articles/2015_pca_in_3_steps.html

```
# Calculating Eigenvectors and eigenvalues of Cov matirx
```

```
mean_vec = np.mean(X_std, axis=0)
cov_mat = np.cov(X_std.T)
eig_vals, eig_vecs = np.linalg.eig(cov_mat)
# Create a list of (eigenvalue, eigenvector) tuples

eig_pairs = [ (np.abs(eig_vals[i]),eig_vecs[:,i]) for i in range(len(eig_vals
))]
# Sort from high to low

eig_pairs.sort(key = lambda x: x[0], reverse= True)
# Calculation of Explained Variance from the eigenvaluestot = sum(eig_vals)
var_exp = [(i/tot)*100 for i in sorted(eig_vals, reverse=True)]
cum_var_exp = np.cumsum(var_exp)
# Cumulative explained variance# Variances plot

max_cols = len(stocks.columns) - 1plt.figure(figsize=(10, 5))
plt.bar(range(max_cols), var_exp, alpha=0.3333, align='center', label='indivi
dual explained variance', color = 'g')
plt.step(range(max_cols), cum_var_exp, where='mid',label='cumulative explaine
d variance')
plt.ylabel('Explained variance ratio')
plt.xlabel('Principal components')
plt.legend(loc='best')
plt.show()
```

输出：

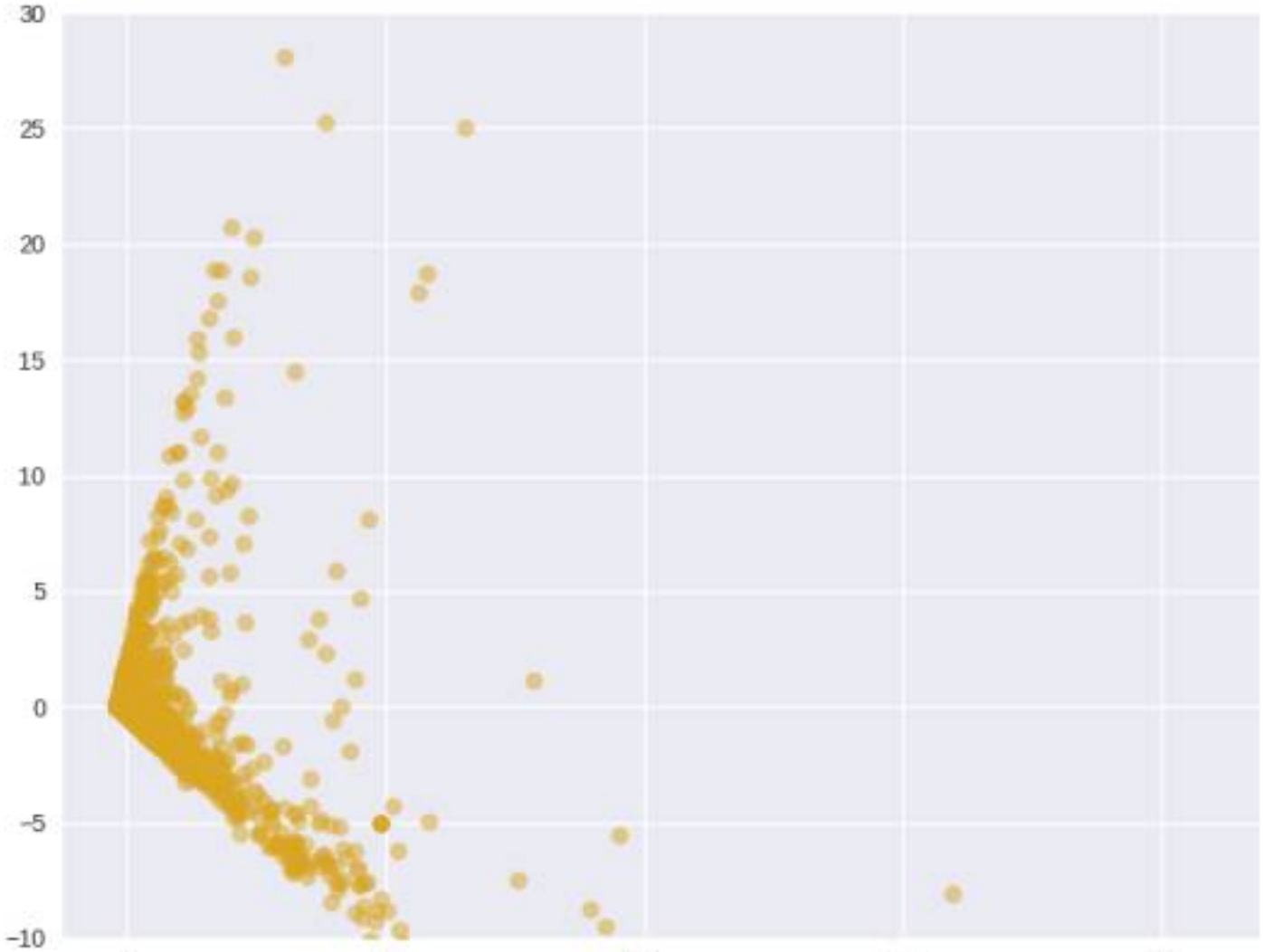


从这个图表中我们可以看到大量方差都来自于预测主成分的前 85%。这是个很高的数字，所以让我们从低端的开始，先只建模少数几个主成分。更多有关分析主成分合理数量的信息可参阅：http://setosa.io/ev/principal-component-analysis

使用 scikit-learn 的 PCA 模块，让我们设 n_components = 9。代码的第二行调用了 fit_transform 方法，其可以使用标准化的电影数据 X_std 来拟合 PCA 模型并在该数据集上应用降维（dimensionality reduction）。

```
pca = PCA(n_components=9)
x_9d = pca.fit_transform(X_std)

plt.figure(figsize = (9,7))
plt.scatter(x_9d[:,0],x_9d[:,1], c='goldenrod',alpha=0.5)
plt.ylim(-10,30)
plt.show()
```

输出：



这里我们甚至没有真正观察到聚类的些微轮廓，所以我们很可能应该继续调节 n_component 的值直到我们得到我们想要的结果。这就是数据科学与艺术（data science and art）中的「艺术」部分。

现在，我们来试试 K-均值，看看我们能不能在下一章节可视化任何明显的聚类。

K-均值聚类（K-Means Clustering）

我们将使用 PCA 投射数据来实现一个简单的 K-均值。

使用 scikit-learn 的 KMeans() 调用和 fit_predict 方法，我们可以计算聚类中心并为第一和第三个 PCA 投射预测聚类索引（以便了解我们是否可以观察到任何合适的聚类）。然后我们可以定义我们自己的配色方案并绘制散点图，代码如下所示：

```
# Set a 3 KMeans clustering
```

```
kmeans = KMeans(n_clusters=3)
# Compute cluster centers and predict cluster indices

X_clustered = kmeans.fit_predict(x_9d)# Define our own color map

LABEL_COLOR_MAP = {0 : 'r',1 : 'g',2 : 'b'}
label_color = [LABEL_COLOR_MAP[l] for l in X_clustered]
# Plot the scatter digram

plt.figure(figsize = (7,7))
plt.scatter(x_9d[:,0],x_9d[:,2], c= label_color, alpha=0.5)
plt.show()
```

输出：

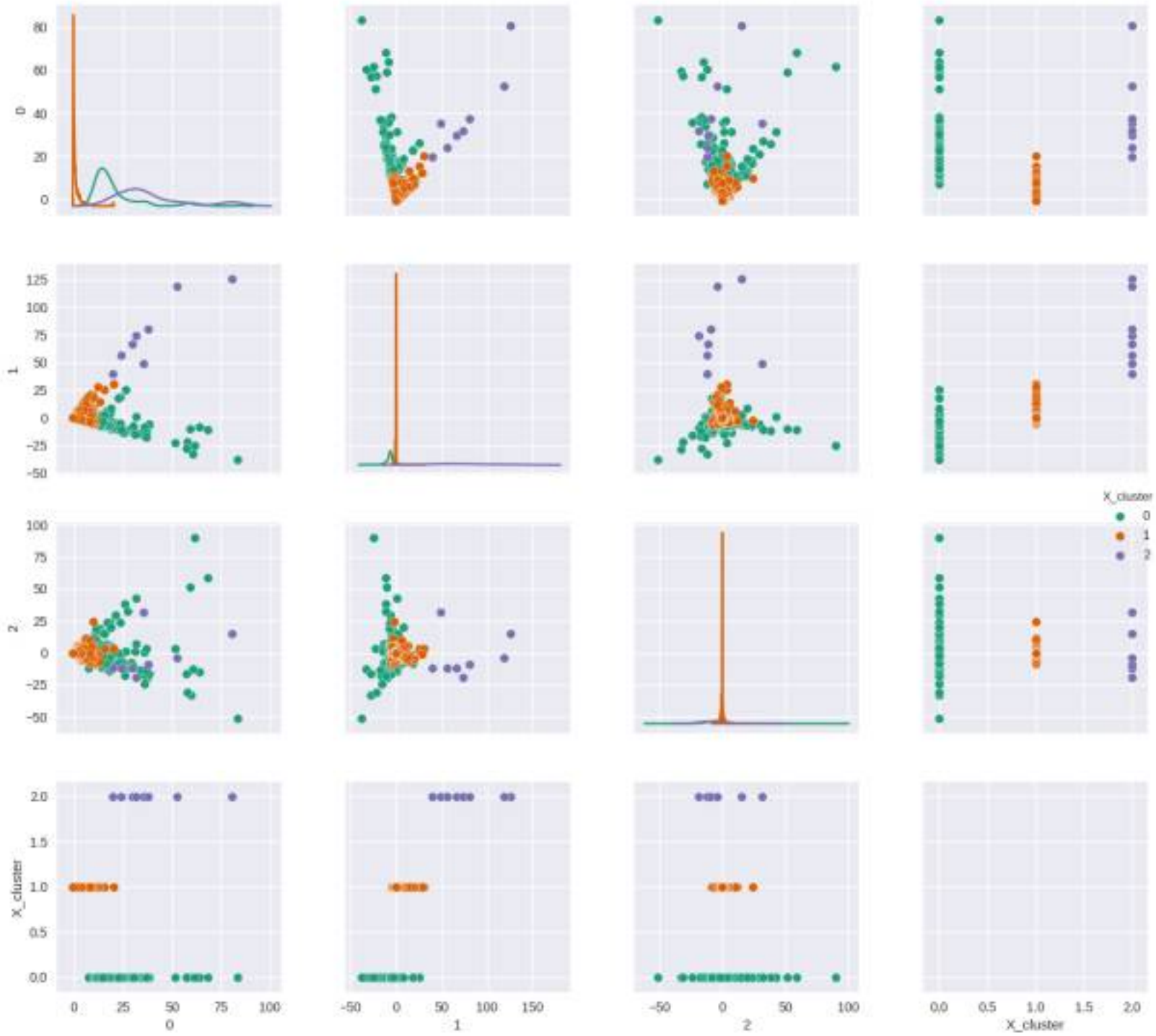
这个 K-均值散点图看起来更有希望，好像我们简单的聚类模型假设就是正确的一样。我们可以通过这种颜色可视化方案观察到 3 个可区分开的聚类。

当然，聚类和可视化数据集的方法还有很多，参考：https://goo.gl/kGy3ra

使用 seaborn 方便的 pairplot 函数，我可以以成对的方式在数据框中自动绘制所有的特征。我们可以一个对一个地 pairplot 前面 3 个投射并可视化：

```
# Create a temp dataframe from our PCA projection data "x_9d"

df = pd.DataFrame(x_9d)
df = df[[0,1,2]]
df['X_cluster'] = X_clustered
# Call Seaborn's pairplot to visualize our KMeans clustering on the PCA proje
cted data

sb.pairplot(df, hue='X_cluster', palette='Dark2', diag_kind='kde', size=1.85)
sb.plt.show()
```

输出：

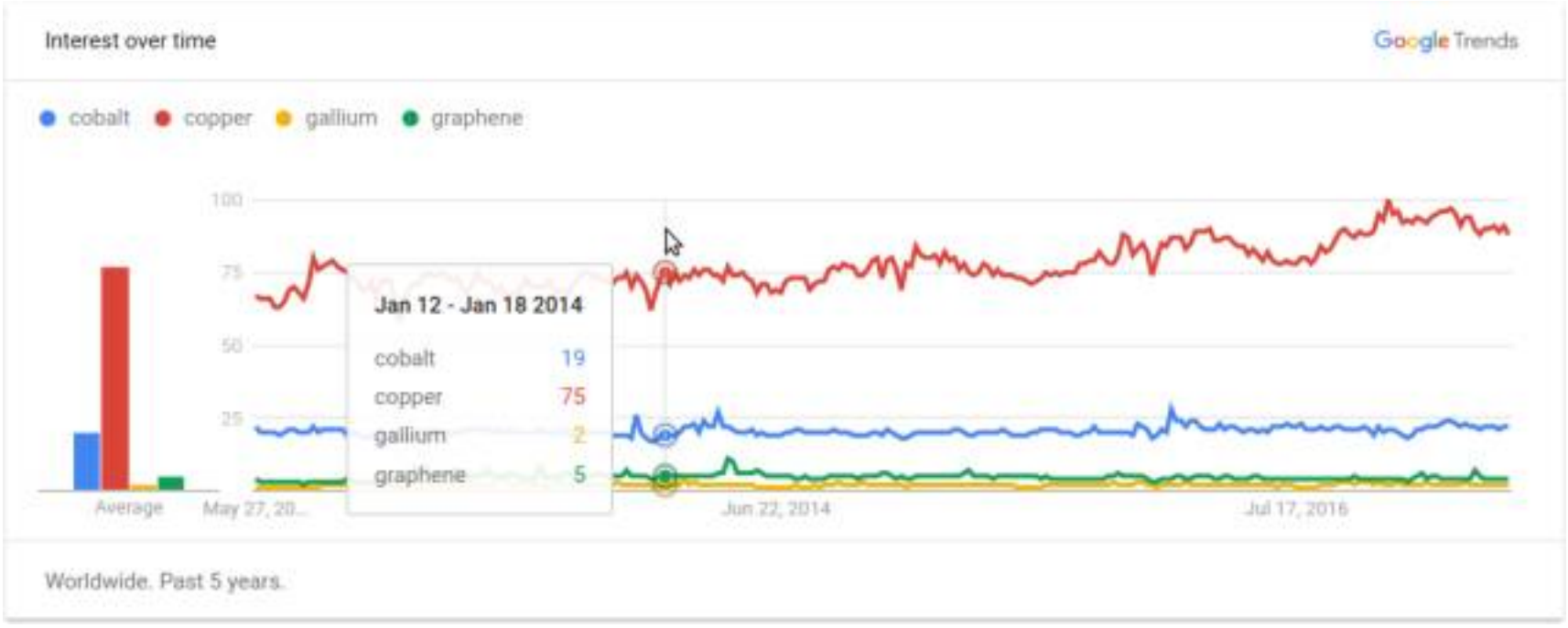


构建篮子聚类（Basket Clusters）

你应该自己决定如何微调你的聚类。这方面没有什么万灵药，具体的方法取决于你操作的环境。在这个案例中是由隐藏关系所定义的股票和金融市场。

一旦你的聚类使你满意了，你就可以设置分数阈值来控制特定的股票是否有资格进入一个聚类，然后你可以为一个给定的聚类提取股票，将它们作为篮子进行交易或使用这些篮子作为信号。你可以使用这种方法做的事情很大程度就看你自己的创造力以及你在使用深度学习变体来进行优化的水平，从而基于聚类或数据点的概念优化每个聚类的回报，比如 short interest 或 short float（公开市场中的可用股份）。

你可以注意到了这些聚类被用作篮子交易的方式一些有趣特征。有时候标准普尔和一般市场会存在差异。这可以提供本质上基于「信息套利（information arbitrage）」的套利机会。一些聚类则和谷歌搜索趋势相关。

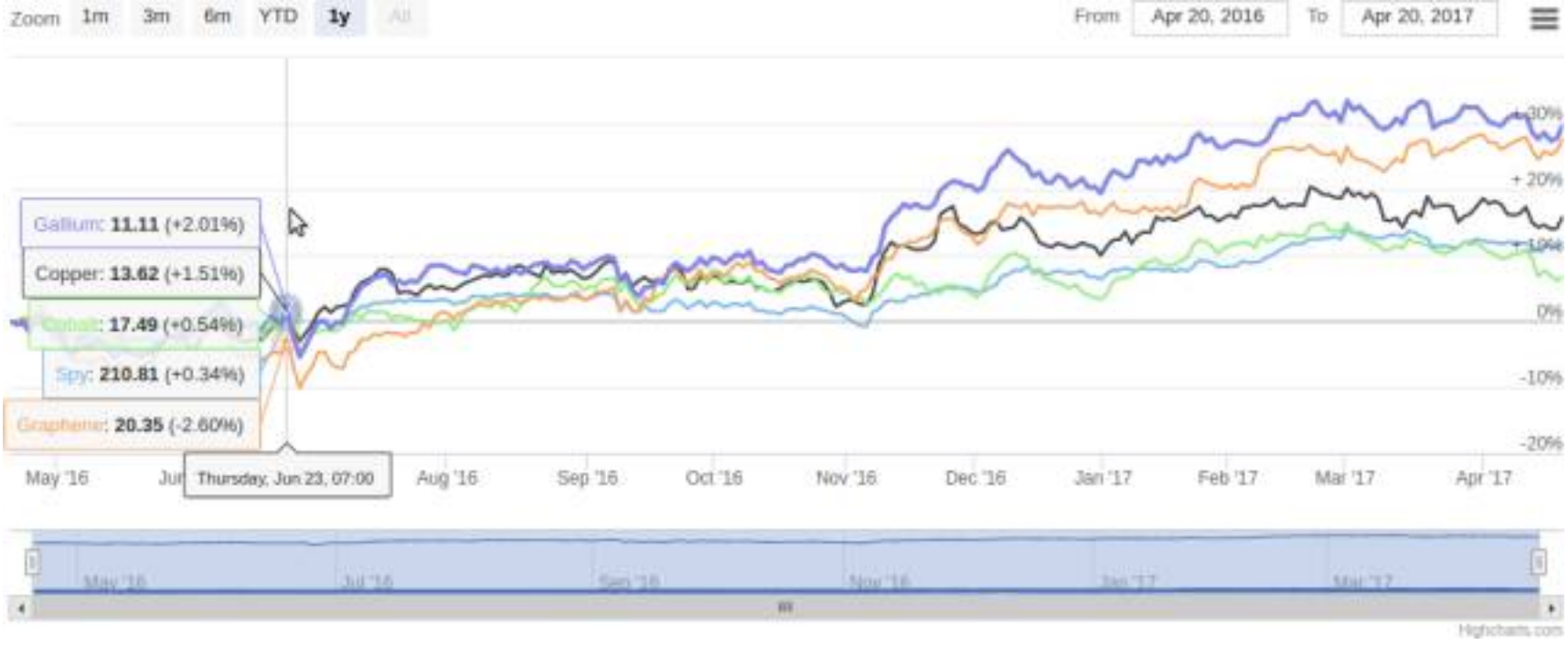


看到聚类和材料及它们的供应链相关确实很有意思，正如这篇文章说的一样：

https://www.fairphone.com/en/2017/05/04/zooming-in-10-materials-and-their-supply-chains/

我仅仅使用该数据集操作了 Cobalt（钴）、Copper（铜）、Gallium（镓）和 Graphene（石墨烯）这几个列标签，只是为了看我是否可能发现从事这一领域或受到这一领域的风险的上市公司之间是否有任何隐藏的联系。这些篮子和标准普尔的回报进行了比较。

通过使用历史价格数据（可直接在 Quantopian、Numerai、Quandl 或 Yahoo Finance 使用），然后你可以汇总价格数据来生成预计收益，其可使用 HighCharts 进行可视化：



我从该聚类中获得的回报超过了标准普尔相当一部分，这意味着你每年的收益可以比标准普尔还多 10%（标准普尔近一年来的涨幅为 16%）。我还见过更加激进的方法可以净挣超过 70%。现在我必须承认我还做了一些其它的事情，但因为我工作的本质，我必须将那些事情保持黑箱。但从我目前观察到的情况来看，至少围绕这种方法探索和包装新的量化模型可以证明是非常值得的，而其唯一的缺点是它是一种不同类型的信号，你可以将其输入其它系统的流程中。

生成卖空篮子聚类（short basket clusters）可能比生成买空篮子聚类（long basket clusters）更有利可图。这种方法值得再写一篇文章，最好是在下一个黑天鹅事件之前。

如果你使用机器学习，就可能在具有已知和隐藏关系的上市公司的寄生、共生和共情关系之上抢占先机，这是很有趣而且可以盈利的。最后，一个人的盈利能力似乎完全关乎他在生成这些类别的数据时想出特征标签（即概念

（concept））的强大组合的能力。

我在这类模型上的下一次迭代应该会包含一个用于自动生成特征组合或独特列表的单独算法。也许会基于近乎实时的事件，这可能会影响那些具有只有配备了无监督学习算法的人类才能预测的隐藏关系的股票组。

原文链接：https://hackernoon.com/unsupervised-machine-learning-for-fun-profit-with-basket-clusters-17a1161e7aa1

阅读原文