|  |  |
| --- | --- |
|  | # built-in package |
|  | import os |
|  | import sys |
|  |  |
|  | sys.path.append('/Users/pawan/google\_driver/github/ipython-notebook-spark/spark-1.6.0-bin-cdh4/spark-1.6.0-bin-cdh4/python') |
|  | sys.path.append('/Users/pawan/google\_driver/github/ipython-notebook-spark/spark-1.6.0-bin-cdh4/spark-1.6.0-bin-cdh4/python/lib/py4j-0.9-src.zip') |
|  |  |
|  | import json |
|  | import time |
|  | import socket |
|  | import operator |
|  | import itertools |
|  | import ConfigParser |
|  | import datetime as dt |
|  |  |
|  | # third-party package |
|  | import pandas as pd |
|  | from matplotlib import pylab |
|  | import matplotlib.pyplot as plt |
|  |  |
|  | import sklearn |
|  | import sklearn.preprocessing |
|  |  |
|  | import numpy as np |
|  | import scipy as sp |
|  | import seaborn |
|  |  |
|  | import pyspark |
|  | from pyspark import SparkContext, SparkConf |
|  | from pyspark.sql import SQLContext, HiveContext, Row |
|  | from pyspark.storagelevel import StorageLevel |
|  | from pyspark.streaming import StreamingContext |
|  |  |
|  | import market\_api |
|  |  |
|  |  |
|  | LOCAL\_STORE\_PATH = "./logs" |
|  | APP\_STORE\_PATH = "./static" |
|  |  |
|  |  |
|  | df\_today\_share = None |
|  | today\_length\_share = None |
|  | rdd\_history = None |
|  |  |
|  |  |
|  | def create\_sc(): |
|  | sc\_conf = SparkConf() |
|  | sc\_conf.setAppName("finance-similarity-app") |
|  | sc\_conf.setMaster('spark://10.21.208.21:7077') |
|  | sc\_conf.set('spark.executor.memory', '2g') |
|  | sc\_conf.set('spark.executor.cores', '4') |
|  | sc\_conf.set('spark.cores.max', '40') |
|  | sc\_conf.set('spark.logConf', True) |
|  | print sc\_conf.getAll() |
|  |  |
|  | sc = None |
|  | try: |
|  | sc.stop() |
|  | sc = SparkContext(conf=sc\_conf) |
|  | except: |
|  | sc = SparkContext(conf=sc\_conf) |
|  |  |
|  | return sc |
|  |  |
|  |  |
|  | def minute\_bar\_today(trade\_date, pre\_trade\_date, ticker="000001.XSHG"): |
|  | pre\_close = market\_api.MktIdxdGet(tradeDate=pre\_trade\_date.replace('-', ''), ticker=ticker[:6]) |
|  | df = market\_api.MktBarRTIntraDayGet(ticker=ticker) |
|  | df['ratio'] = df.closePrice / pre\_close - 1 |
|  |  |
|  | return df[['ticker', 'barTime', 'closePrice', 'ratio']] |
|  |  |
|  |  |
|  | def minute\_bar\_today\_demo(trade\_date, pre\_trade\_date, ticker="000001.XSHG"): |
|  | pre\_close = market\_api.MktIdxdGetDemo(tradeDate=pre\_trade\_date.replace('-', ''), ticker=ticker[:6]) |
|  | df = market\_api.MktBarRTIntraDayGetDemo(ticker=ticker) |
|  | df['ratio'] = df.closePrice / pre\_close - 1 |
|  |  |
|  | return df[['ticker', 'barTime', 'closePrice', 'ratio']] |
|  |  |
|  |  |
|  | def cal\_minute\_bar\_similarity(line\_data): |
|  |  |
|  | line\_data format: file\_path, json\_data |
|  | Return: |
|  | square diff and var diff of two lines. |
|  | [diff\_square, diff\_var, (line\_path)] |
|  | [diff\_square\_normalized, diff\_var\_normalized, (line\_path)] |
|  | """ |
|  | tmp = pd.DataFrame() |
|  |  |
|  | import sklearn.preprocessing |
|  | scaler = sklearn.preprocessing.MinMaxScaler() |
|  |  |
|  | today\_data = pd.DataFrame.from\_dict(json.loads(df\_today\_share.value)) |
|  | today\_data\_length = today\_length\_share.value |
|  | line\_path, line\_df = line\_data |
|  |  |
|  | line\_df = pd.DataFrame.from\_dict(json.loads(line\_df)) |
|  | line\_df.sort(columns=['barTime'], ascending=True, inplace=True) |
|  |  |
|  | tmp['first'] = list(today\_data[: today\_data\_length]['ratio']) |
|  | tmp['second'] = list(line\_df[: today\_data\_length]['ratio']) |
|  |  |
|  | \_first, \_second = list(tmp['first']), list(tmp['second']) |
|  | tmp['first\_normalized'] = list(scaler.fit\_transform(np.array(\_first))) |
|  | tmp['second\_normalized'] = list(scaler.fit\_transform(np.array(\_second))) |
|  |  |
|  | tmp['diff'] = tmp['first'] - tmp['second'] |
|  | tmp['diff\_normalized'] = tmp['first\_normalized'] - tmp['second\_normalized'] |
|  |  |
|  | diff\_square = sum(tmp['diff'] \*\* 2) |
|  | diff\_square\_normalized = sum(tmp['diff\_normalized'] \*\* 2) |
|  |  |
|  | diff\_var = float(tmp['diff'].var()) |
|  | diff\_var\_normalized = float(tmp['diff\_normalized'].var()) |
|  | res\_square = [round(diff\_square, 5), round(diff\_square\_normalized, 5), (line\_path)] |
|  | res\_var = [round(diff\_var, 5), round(diff\_var\_normalized, 5), (line\_path)] |
|  |  |
|  | return res\_square + res\_var |
|  |  |
|  |  |
|  |  |
|  |  |
|  | def draw\_similarity(df\_today, similarity\_data, minute\_bar\_length=90): |
|  | res = pd.DataFrame() |
|  | df\_today = pd.DataFrame.from\_dict(json.loads(df\_today)) |
|  |  |
|  | columns = [] |
|  | for line\_tuple in similarity\_data: |
|  | line\_id, line\_data = line\_tuple |
|  | line\_id = line\_id[-25 : -5] |
|  | line\_data = pd.DataFrame.from\_dict(json.loads(line\_data)) |
|  | res[line\_id] = line\_data['ratio'] |
|  |  |
|  | if 'minute' not in res : |
|  | res['minute'] = line\_data['barTime'] |
|  | columns.append(line\_id) |
|  |  |
|  | res['fitting'] = res[columns].sum(axis=1) / len(columns) |
|  | res.sort(columns=['minute'], ascending=True, inplace=True) |
|  | res['today\_line'] = list(df\_today['ratio']) + [None] \* (241 - len(df\_today)) |
|  |  |
|  | ### plot |
|  | ax = res.plot(x='minute', y=columns, figsize=(20, 13), legend=False, title=u'Minute Bar Prediction') |
|  | res.plot(y=['today\_line'], ax=ax, linewidth=5, style='b') |
|  | res.plot(y=['fitting'], ax=ax, linewidth=4, style='-y') |
|  | ax.vlines(x=minute\_bar\_length, ymin=-0.02, ymax=0.02, linestyles='dashed') |
|  | ax.set\_axis\_bgcolor('white') |
|  | ax.grid(color='gray', alpha=0.2, axis='y') |
|  |  |
|  | ### plot area |
|  | avg\_line = res['fitting'] |
|  | avg\_line = list(avg\_line)[minute\_bar\_length : ] |
|  | for line in columns: |
|  | predict\_line = res[line] |
|  | predict\_line = list(predict\_line)[minute\_bar\_length : ] |
|  | ax.fill\_between(range(minute\_bar\_length, 241), avg\_line, predict\_line, alpha=0.1, color='r') |
|  |  |
|  | ### store data on dist |
|  | current\_time = dt.datetime.now().strftime("%Y-%m-%dT%H-%M-%S") |
|  | res.to\_json(LOCAL\_STORE\_PATH + '/data-{}.json'.format(current\_time)) |
|  | res.to\_json(LOCAL\_STORE\_PATH + '/latest.json'.format(current\_time)) |
|  |  |
|  | fig = ax.get\_figure() |
|  | fig.savefig(LOCAL\_STORE\_PATH + '/plot-{}.png'.format(current\_time)) |
|  | fig.savefig(LOCAL\_STORE\_PATH + '/latest.png'.format(current\_time)) |
|  | fig.savefig(APP\_STORE\_PATH + '/latest.png'.format(current\_time)) |
|  |  |
|  |  |
|  | def pipeline(): |
|  | global df\_today\_share |
|  | global today\_length\_share |
|  | global rdd\_history |
|  |  |
|  | now = dt.datetime.now() |
|  | bar\_time = '{}:{}'.format(now.hour, now.minute) |
|  | print '###Loat history data {} ...'.format(time.ctime()) |
|  |  |
|  | rdd\_history = sc.wholeTextFiles('hdfs://10.21.208.21:8020/user/mercury/minute\_bar', minPartitions=80) \ |
|  | .setName('index\_minute\_bar') \ |
|  | .cache() |
|  |  |
|  | while bar\_time < '15:00': |
|  | print '###Start Prediction on {} ...'.format(time.ctime()) |
|  |  |
|  | df\_today = minute\_bar\_today('20160804', '20160803', ticker="000001.XSHG") |
|  | df\_today\_share = sc.broadcast(df\_today) |
|  | # df\_today\_share = sc.broadcast(120) |
|  | today\_length = len(df\_today) |
|  | today\_length\_share = sc.broadcast(today\_length) |
|  |  |
|  | ### do the calculation |
|  | rdd\_similarity = rdd\_history.map(cal\_minute\_bar\_similarity).setName("similariy") .cache() |
|  | res\_df = build\_similarity\_report(rdd\_similarity) |
|  | similarity\_data = get\_similarity\_data(res\_df, 40) |
|  | res = draw\_similarity(df\_today, similarity\_data, minute\_bar\_length=today\_length\_share.value) |
|  |  |
|  | print '###Done Prediction on {} ...'.format(time.ctime()) |
|  | time.sleep(65) |
|  | now = dt.datetime.now() |
|  | bar\_time = '{}:{}'.format(now.hour, now.minute) |
|  |  |
|  | sc.stop() |
|  |  |
|  |  |
|  | def demo(): |
|  | global df\_today\_share |
|  | global today\_length\_share |
|  | global rdd\_history |
|  |  |
|  | print '###Loat history data {} ...'.format(time.ctime()) |
|  | sc |
|  | sc = create\_sc() |
|  |  |
|  |  |
|  | rdd\_history = sc.wholeTextFiles('hdfs://10.21.208.21:8020/user/mercury/minute\_bar', minPartitions=80) \ |
|  | .setName('index\_minute\_bar') \ |
|  | .cache() |
|  |  |
|  | today\_length = 120 |
|  |  |
|  | while today\_length < 241: |
|  | print '###Start Prediction on ...'.format() |
|  |  |
|  | df\_today = minute\_bar\_today\_demo('20160702', '20160701', ticker="000001.XSHG") |
|  | df\_today = df\_today[: today\_length].to\_json() |
|  | df\_today\_share = sc.broadcast(df\_today) |
|  | today\_length\_share = sc.broadcast(today\_length) |
|  |  |
|  | ### do the calculation |
|  | rdd\_similarity = rdd\_history.map(cal\_minute\_bar\_similarity).setName("similariy").cache() |
|  |  |
|  | res\_df = build\_similarity\_report(rdd\_similarity) |
|  | similarity\_data = get\_similarity\_data(res\_df, 40) |
|  | res = draw\_similarity(df\_today, similarity\_data, minute\_bar\_length=today\_length\_share.value) |
|  |  |
|  | print '###Done Prediction on ...'.format() |
|  | time.sleep(65) |
|  | today\_length += 1 |
|  |  |
|  | sc.stop() |
|  |  |
|  |  |
|  | if \_\_name\_\_ == '\_\_main\_\_': |
|  | demo() |