project_7_starter

April 24, 2023

1 Project 7: Combine Signals for Enhanced Alpha

1.1 Instructions

Each problem consists of a function to implement and instructions on how to implement the function. The parts of the function that need to be implemented are marked with a `# TODO` comment. After implementing the function, run the cell to test it against the unit tests we've provided. For each problem, we provide one or more unit tests from our `project_tests` package. These unit tests won't tell you if your answer is correct, but will warn you of any major errors. Your code will be checked for the correct solution when you submit it to Udacity.

1.2 Packages

When you implement the functions, you'll only need to you use the packages you've used in the classroom, like Pandas and Numpy. These packages will be imported for you. We recommend you don't add any import statements, otherwise the grader might not be able to run your code.

The other packages that we're importing are `project_helper` and `project_tests`. These are custom packages built to help you solve the problems. The `project_helper` module contains utility functions and graph functions. The `project_tests` contains the unit tests for all the problems.

1.2.1 Install Packages

```
In [2]: !pip install --upgrade setuptools

Collecting setuptools
  Downloading https://files.pythonhosted.org/packages/b0/3a/88b210db68e56854d0bcf4b38e165e03be37
    100% || 962kB 12.3MB/s ta 0:00:01
Installing collected packages: setuptools
  Found existing installation: setuptools 38.4.0
    Uninstalling setuptools-38.4.0:
      Successfully uninstalled setuptools-38.4.0
Successfully installed setuptools-59.6.0


In [3]: import sys
        !{sys.executable} -m pip install -r requirements.txt
```

```
Collecting sqlalchemy==1.3.0 (from -r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/35/9e/5eb467ed50cdd8e88b808a7e65045020fa12
    100% || 5.9MB 5.3MB/s eta 0:00:01
Collecting alphalens==0.3.2 (from -r requirements.txt (line 2))
  Downloading https://files.pythonhosted.org/packages/a5/dc/2f9cd107d0d4cf6223d37d81ddfbbdbf0d70
    100% || 18.9MB 1.8MB/s eta 0:00:01    63% |           | 12.0MB 46.9MB/s eta 0:00:01    75% |
Collecting graphviz==0.10.1 (from -r requirements.txt (line 3))
  Downloading https://files.pythonhosted.org/packages/1f/e2/ef2581b5b86625657afd32030f90cf271745
Collecting numpy==1.13.3 (from -r requirements.txt (line 4))
  Downloading https://files.pythonhosted.org/packages/57/a7/e3e6bd9d595125e1abbe162e323fd2d06f6f
    100% || 17.0MB 2.4MB/s eta 0:00:01
Collecting pandas==0.18.1 (from -r requirements.txt (line 5))
  Downloading https://files.pythonhosted.org/packages/11/09/e66eb844daba8680ddff26335d5b4fead77f
    100% || 7.3MB 9.3MB/s eta 0:00:01    38% |                   | 2.8MB 46.4MB/s eta 0:00:01
Requirement already satisfied: python-dateutil==2.6.1 in /opt/conda/lib/python3.6/site-packages
Requirement already satisfied: pytz==2017.3 in /opt/conda/lib/python3.6/site-packages (from -r r
Collecting scipy==1.0.0 (from -r requirements.txt (line 8))
  Downloading https://files.pythonhosted.org/packages/d8/5e/caa01ba7be11600b6a9d39265440d7b3be3d
    100% || 50.0MB 719kB/s eta 0:00:01  5% |                                | 2.9MB 43.1MB/s eta 0
Requirement already satisfied: scikit-learn==0.19.1 in /opt/conda/lib/python3.6/site-packages (f
Requirement already satisfied: six==1.11.0 in /opt/conda/lib/python3.6/site-packages (from -r re
Collecting tables==3.3.0 (from -r requirements.txt (line 11))
  Downloading https://files.pythonhosted.org/packages/09/e7/72ca83c7bd75db94c23fcaf58debe1be5e98
    100% || 4.6MB 12.6MB/s ta 0:00:01    66% |          | 3.0MB 47.1MB/s eta 0:00:01
Collecting tqdm==4.19.5 (from -r requirements.txt (line 12))
  Downloading https://files.pythonhosted.org/packages/71/3c/341b4fa23cb3abc335207dba057c790f3bb3
    100% || 61kB 16.1MB/s ta 0:00:01
Collecting zipline===1.2.0 (from -r requirements.txt (line 13))
  Downloading https://files.pythonhosted.org/packages/15/d3/689f2a940478b82ac57c751a4046059821f
    100% || 665kB 20.1MB/s ta 0:00:01
Requirement already satisfied: matplotlib>=1.4.0 in /opt/conda/lib/python3.6/site-packages (from
Requirement already satisfied: seaborn>=0.6.0 in /opt/conda/lib/python3.6/site-packages (from al
Requirement already satisfied: statsmodels>=0.6.1 in /opt/conda/lib/python3.6/site-packages (fro
Requirement already satisfied: IPython>=3.2.3 in /opt/conda/lib/python3.6/site-packages (from al
Requirement already satisfied: numexpr>=2.5.2 in /opt/conda/lib/python3.6/site-packages (from ta
Requirement already satisfied: pip>=7.1.0 in /opt/conda/lib/python3.6/site-packages (from ziplin
Requirement already satisfied: setuptools>18.0 in /opt/conda/lib/python3.6/site-packages (from z
Collecting Logbook>=0.12.5 (from zipline===1.2.0->-r requirements.txt (line 13))
  Downloading https://files.pythonhosted.org/packages/2f/d9/16ac346f7c0102835814cc9e5b684aaadea1
    100% || 92kB 16.9MB/s ta 0:00:01
Collecting requests-file>=1.4.1 (from zipline===1.2.0->-r requirements.txt (line 13))
  Downloading https://files.pythonhosted.org/packages/77/86/cdb5e8eaed90796aa83a6d9f75cfbd37af55
Collecting pandas-datareader<0.6,>=0.2.1 (from zipline===1.2.0->-r requirements.txt (line 13))
  Downloading https://files.pythonhosted.org/packages/40/c5/cc720f531bbde0efeab940de400d0fcc95e8
    100% || 81kB 17.9MB/s ta 0:00:01
Requirement already satisfied: patsy>=0.4.0 in /opt/conda/lib/python3.6/site-packages (from zipl
Requirement already satisfied: requests>=2.9.1 in /opt/conda/lib/python3.6/site-packages (from z
Requirement already satisfied: Cython>=0.25.2 in /opt/conda/lib/python3.6/site-packages (from zi
```

```
Collecting cyordereddict>=0.2.2 (from zipline===1.2.0->-r requirements.txt (line 13))
  Downloading https://files.pythonhosted.org/packages/d1/1a/364cbfd927be1b743c7f0a985a7f1f7e8a51
    100% || 143kB 16.7MB/s ta 0:00:01
Collecting bottleneck>=1.0.0 (from zipline===1.2.0->-r requirements.txt (line 13))
  Downloading https://files.pythonhosted.org/packages/4e/b1/abd8986be57a9ed6f93ad685c55ebe3fe93e
    100% || 358kB 19.9MB/s ta 0:00:01
Collecting contextlib2>=0.4.0 (from zipline===1.2.0->-r requirements.txt (line 13))
  Downloading https://files.pythonhosted.org/packages/76/56/6d6872f79d14c0cb02f1646cbb4592eef935
Requirement already satisfied: decorator>=4.0.0 in /opt/conda/lib/python3.6/site-packages (from
Requirement already satisfied: networkx<2.0,>=1.9.1 in /opt/conda/lib/python3.6/site-packages (f
Collecting bcolz<1,>=0.12.1 (from zipline===1.2.0->-r requirements.txt (line 13))
  Downloading https://files.pythonhosted.org/packages/6c/8b/1ffa01f872cac36173c5eb95b58c01040d8d
    100% || 624kB 19.3MB/s ta 0:00:01
Requirement already satisfied: click>=4.0.0 in /opt/conda/lib/python3.6/site-packages (from zipl
Requirement already satisfied: toolz>=0.8.2 in /opt/conda/lib/python3.6/site-packages (from zipl
Collecting multipledispatch>=0.4.8 (from zipline===1.2.0->-r requirements.txt (line 13))
  Downloading https://files.pythonhosted.org/packages/89/79/429ecef45fd5e4504f7474d4c3c3c4668c26
Requirement already satisfied: MarkupSafe>=0.23 in /opt/conda/lib/python3.6/site-packages (from
Requirement already satisfied: Mako>=1.0.1 in /opt/conda/lib/python3.6/site-packages/Mako-1.0.7-
Collecting alembic>=0.7.7 (from zipline===1.2.0->-r requirements.txt (line 13))
  Downloading https://files.pythonhosted.org/packages/b3/e2/8d48220731b7279911c43e95cd182961a703
    100% || 215kB 20.9MB/s ta 0:00:01
Collecting sortedcontainers>=1.4.4 (from zipline===1.2.0->-r requirements.txt (line 13))
  Downloading https://files.pythonhosted.org/packages/32/46/9cb0e58b2deb7f82b84065f37f3bffeb1241
Collecting intervaltree>=2.1.0 (from zipline===1.2.0->-r requirements.txt (line 13))
  Downloading https://files.pythonhosted.org/packages/50/fb/396d568039d21344639db96d940d40eb62be
Collecting lru-dict>=1.1.4 (from zipline===1.2.0->-r requirements.txt (line 13))
  Downloading https://files.pythonhosted.org/packages/3d/b7/7b1b55b515a8de71459ef97caedb7b1cbed4
Collecting empyrical>=0.4.2 (from zipline===1.2.0->-r requirements.txt (line 13))
  Downloading https://files.pythonhosted.org/packages/74/43/1b997c21411c6ab7c96dc034e160198272c7
    100% || 61kB 15.4MB/s ta 0:00:01
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.6/site-packages/cycler-0.1
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /opt/conda/lib/python
Requirement already satisfied: jedi>=0.10 in /opt/conda/lib/python3.6/site-packages (from IPytho
Requirement already satisfied: pexpect; sys_platform != "win32" in /opt/conda/lib/python3.6/site
Requirement already satisfied: traitlets>=4.2 in /opt/conda/lib/python3.6/site-packages (from IP
Requirement already satisfied: backcall in /opt/conda/lib/python3.6/site-packages (from IPython>
Requirement already satisfied: pickleshare in /opt/conda/lib/python3.6/site-packages (from IPyth
Requirement already satisfied: pygments in /opt/conda/lib/python3.6/site-packages (from IPython>
Requirement already satisfied: simplegeneric>0.8 in /opt/conda/lib/python3.6/site-packages (from
Requirement already satisfied: prompt-toolkit<2.0.0,>=1.0.15 in /opt/conda/lib/python3.6/site-pa
Collecting requests-ftp (from pandas-datareader<0.6,>=0.2.1->zipline===1.2.0->-r requirements.tx
  Downloading https://files.pythonhosted.org/packages/3d/ca/14b2ad1e93b5195eeaf56b86b7ecfd5ea2d5
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /opt/conda/lib/python3.6/site-packages (
Requirement already satisfied: idna<2.7,>=2.5 in /opt/conda/lib/python3.6/site-packages (from re
Requirement already satisfied: urllib3<1.23,>=1.21.1 in /opt/conda/lib/python3.6/site-packages (
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.6/site-packages (fro
Collecting importlib-resources; python_version < "3.9" (from alembic>=0.7.7->zipline===1.2.0->-r
```

```
  Downloading https://files.pythonhosted.org/packages/24/1b/33e489669a94da3ef4562938cd306e8fa915
Collecting importlib-metadata; python_version < "3.9" (from alembic>=0.7.7->zipline===1.2.0->-r
  Downloading https://files.pythonhosted.org/packages/a0/a1/b153a0a4caf7a7e3f15c2cd56c7702e2cf3d
Requirement already satisfied: ptyprocess>=0.5 in /opt/conda/lib/python3.6/site-packages (from p
Requirement already satisfied: ipython-genutils in /opt/conda/lib/python3.6/site-packages (from
Requirement already satisfied: wcwidth in /opt/conda/lib/python3.6/site-packages (from prompt-to
Collecting zipp>=3.1.0; python_version < "3.10" (from importlib-resources; python_version < "3.9
  Downloading https://files.pythonhosted.org/packages/bd/df/d4a4974a3e3957fd1c1fa3082366d7fff6e4
Collecting typing-extensions>=3.6.4; python_version < "3.8" (from importlib-metadata; python_ver
  Downloading https://files.pythonhosted.org/packages/45/6b/44f7f8f1e110027cf88956b59f2fad776cca
Building wheels for collected packages: sqlalchemy, alphalens, pandas, zipline, Logbook, cyorder
  Running setup.py bdist_wheel for sqlalchemy ... done
  Stored in directory: /root/.cache/pip/wheels/64/3c/23/a6e4cda86f6fbbe9aac330fd040190e6b473fb75
  Running setup.py bdist_wheel for alphalens ... done
  Stored in directory: /root/.cache/pip/wheels/77/1e/9a/223b4c94d7f564f25d94b48ca5b9c53e3034016e
  Running setup.py bdist_wheel for pandas ... done
  Stored in directory: /root/.cache/pip/wheels/a3/08/c3/8fdd52954d4b415624cff43c6dd32a22bac90306
  Running setup.py bdist_wheel for zipline ... done
  Stored in directory: /root/.cache/pip/wheels/5d/20/7d/b48368c8634b1cb6cc7232833b2780a265d4217c
  Running setup.py bdist_wheel for Logbook ... done
  Stored in directory: /root/.cache/pip/wheels/d2/70/07/68b99a8e05dcd1ab194a8e0ccb9e4d0ac5dd6d8d
  Running setup.py bdist_wheel for cyordereddict ... done
  Stored in directory: /root/.cache/pip/wheels/0b/9d/8b/5bf3e22c1edd59b50f11bb19dec9dfcfe5a479fc
  Running setup.py bdist_wheel for bcolz ... done
  Stored in directory: /root/.cache/pip/wheels/c5/cc/1b/2cf1f88959af5d7f4d449b7fc6c9452d0ecbd86f
  Running setup.py bdist_wheel for intervaltree ... done
  Stored in directory: /root/.cache/pip/wheels/f3/f2/66/e9c30d3e9499e65ea2fa0d07c002e64de63bd0ad
  Running setup.py bdist_wheel for empyrical ... done
  Stored in directory: /root/.cache/pip/wheels/ea/b2/c8/6769d8444d2f2e608fae2641833110668d0ffd1a
  Running setup.py bdist_wheel for requests-ftp ... done
  Stored in directory: /root/.cache/pip/wheels/2a/98/32/37195e45a3392a73d9f65c488cbea30fe5bad76a
Successfully built sqlalchemy alphalens pandas zipline Logbook cyordereddict bcolz intervaltree
tensorflow 1.3.0 requires tensorflow-tensorboard<0.2.0,>=0.1.0, which is not installed.
moviepy 0.2.3.2 has requirement tqdm==4.11.2, but you'll have tqdm 4.19.5 which is incompatible.
Installing collected packages: sqlalchemy, numpy, pandas, scipy, alphalens, graphviz, tables, tq
  Found existing installation: SQLAlchemy 1.1.13
    Uninstalling SQLAlchemy-1.1.13:
      Successfully uninstalled SQLAlchemy-1.1.13
  Found existing installation: numpy 1.12.1
    Uninstalling numpy-1.12.1:
      Successfully uninstalled numpy-1.12.1
  Found existing installation: pandas 0.23.3
    Uninstalling pandas-0.23.3:
      Successfully uninstalled pandas-0.23.3
  Found existing installation: scipy 1.2.1
    Uninstalling scipy-1.2.1:
      Successfully uninstalled scipy-1.2.1
  Found existing installation: tqdm 4.11.2
```

```
    Uninstalling tqdm-4.11.2:
      Successfully uninstalled tqdm-4.11.2
Successfully installed Logbook-1.5.3 alembic-1.7.7 alphalens-0.3.2 bcolz-0.12.1 bottleneck-1.3.7
```

1.2.2 Load Packages

```
In [10]: import project_helper
         import project_tests

         import numpy as np
         import pandas as pd
         from tqdm import tqdm
         import matplotlib.pyplot as plt

         %matplotlib inline
         plt.style.use('ggplot')
         plt.rcParams['figure.figsize'] = (14, 8)
```

1.3 Data Pipeline

1.3.1 Data Bundle

We'll be using Zipline to handle our data. We've created a end of day data bundle for this project. Run the cell below to register this data bundle in zipline.

```
In [11]: import os
         from zipline.data import bundles

         os.environ['ZIPLINE_ROOT'] = os.path.join(os.getcwd(), '..', '..', 'data', 'project_7_e

         ingest_func = bundles.csvdir.csvdir_equities(['daily'], project_helper.EOD_BUNDLE_NAME)
         bundles.register(project_helper.EOD_BUNDLE_NAME, ingest_func)

         print('Data Registered')
```

```
Data Registered
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:7: UserWarning: Overwriting bundle
  import sys
```

1.3.2 Build Pipeline Engine

We'll be using Zipline's pipeline package to access our data for this project. To use it, we must build a pipeline engine. Run the cell below to build the engine.

```
In [12]: from zipline.pipeline import Pipeline
         from zipline.pipeline.factors import AverageDollarVolume
```

```
from zipline.utils.calendars import get_calendar


universe = AverageDollarVolume(window_length=120).top(500)
trading_calendar = get_calendar('NYSE')
bundle_data = bundles.load(project_helper.EOD_BUNDLE_NAME)
engine = project_helper.build_pipeline_engine(bundle_data, trading_calendar)
```

View Data With the pipeline engine built, let's get the stocks at the end of the period in the universe we're using.

```
In [13]: universe_end_date = pd.Timestamp('2016-01-05', tz='UTC')

         universe_tickers = engine\
             .run_pipeline(
                 Pipeline(screen=universe),
                 universe_end_date,
                 universe_end_date)\
             .index.get_level_values(1)\
             .values.tolist()

         universe_tickers

Out[13]: [Equity(0 [A]),
          Equity(1 [AAL]),
          Equity(2 [AAP]),
          Equity(3 [AAPL]),
          Equity(4 [ABBV]),
          Equity(5 [ABC]),
          Equity(6 [ABT]),
          Equity(7 [ACN]),
          Equity(8 [ADBE]),
          Equity(9 [ADI]),
          Equity(10 [ADM]),
          Equity(11 [ADP]),
          Equity(12 [ADS]),
          Equity(13 [ADSK]),
          Equity(14 [AEE]),
          Equity(15 [AEP]),
          Equity(16 [AES]),
          Equity(17 [AET]),
          Equity(18 [AFL]),
          Equity(19 [AGN]),
          Equity(20 [AIG]),
          Equity(21 [AIV]),
          Equity(22 [AIZ]),
          Equity(23 [AJG]),
          Equity(24 [AKAM]),
```

```
Equity(25 [ALB]),
Equity(26 [ALGN]),
Equity(27 [ALK]),
Equity(28 [ALL]),
Equity(29 [ALLE]),
Equity(30 [ALXN]),
Equity(31 [AMAT]),
Equity(32 [AMD]),
Equity(33 [AME]),
Equity(34 [AMG]),
Equity(35 [AMGN]),
Equity(36 [AMP]),
Equity(37 [AMT]),
Equity(38 [AMZN]),
Equity(39 [ANDV]),
Equity(40 [ANSS]),
Equity(41 [ANTM]),
Equity(42 [AON]),
Equity(43 [AOS]),
Equity(44 [APA]),
Equity(45 [APC]),
Equity(46 [APD]),
Equity(47 [APH]),
Equity(48 [ARE]),
Equity(49 [ARNC]),
Equity(50 [ATVI]),
Equity(51 [AVB]),
Equity(52 [AVGO]),
Equity(53 [AVY]),
Equity(54 [AWK]),
Equity(55 [AXP]),
Equity(56 [AYI]),
Equity(57 [AZO]),
Equity(58 [BA]),
Equity(59 [BAC]),
Equity(60 [BAX]),
Equity(61 [BBT]),
Equity(62 [BBY]),
Equity(63 [BCR]),
Equity(64 [BDX]),
Equity(65 [BEN]),
Equity(66 [BIIB]),
Equity(67 [BK]),
Equity(68 [BLK]),
Equity(69 [BLL]),
Equity(70 [BMY]),
Equity(71 [BSX]),
Equity(72 [BWA]),
```

```
Equity(73 [BXP]),
Equity(74 [C]),
Equity(75 [CA]),
Equity(76 [CAG]),
Equity(77 [CAH]),
Equity(78 [CAT]),
Equity(79 [CB]),
Equity(80 [CBG]),
Equity(81 [CBOE]),
Equity(82 [CBS]),
Equity(83 [CCI]),
Equity(84 [CCL]),
Equity(85 [CELG]),
Equity(86 [CERN]),
Equity(87 [CF]),
Equity(88 [CFG]),
Equity(89 [CHD]),
Equity(90 [CHK]),
Equity(91 [CHRW]),
Equity(92 [CHTR]),
Equity(93 [CI]),
Equity(94 [CINF]),
Equity(95 [CL]),
Equity(96 [CLX]),
Equity(97 [CMA]),
Equity(98 [CMCSA]),
Equity(99 [CME]),
Equity(100 [CMG]),
Equity(101 [CMI]),
Equity(102 [CMS]),
Equity(103 [CNC]),
Equity(104 [CNP]),
Equity(105 [COF]),
Equity(106 [COG]),
Equity(107 [COL]),
Equity(108 [COO]),
Equity(109 [COP]),
Equity(110 [COST]),
Equity(111 [COTY]),
Equity(112 [CPB]),
Equity(113 [CRM]),
Equity(114 [CSCO]),
Equity(115 [CSRA]),
Equity(116 [CSX]),
Equity(117 [CTAS]),
Equity(118 [CTL]),
Equity(119 [CTSH]),
Equity(120 [CTXS]),
```

```
Equity(121 [CVS]),
Equity(122 [CVX]),
Equity(123 [CXO]),
Equity(124 [D]),
Equity(125 [DAL]),
Equity(126 [DE]),
Equity(127 [DFS]),
Equity(128 [DG]),
Equity(129 [DGX]),
Equity(130 [DHI]),
Equity(131 [DHR]),
Equity(132 [DIS]),
Equity(133 [DISCA]),
Equity(134 [DISCK]),
Equity(135 [DISH]),
Equity(136 [DLR]),
Equity(137 [DLTR]),
Equity(138 [DOV]),
Equity(139 [DPS]),
Equity(140 [DRE]),
Equity(141 [DRI]),
Equity(142 [DTE]),
Equity(143 [DUK]),
Equity(144 [DVA]),
Equity(145 [DVN]),
Equity(146 [EA]),
Equity(147 [EBAY]),
Equity(148 [ECL]),
Equity(149 [ED]),
Equity(150 [EFX]),
Equity(151 [EIX]),
Equity(152 [EL]),
Equity(153 [EMN]),
Equity(154 [EMR]),
Equity(155 [EOG]),
Equity(156 [EQIX]),
Equity(157 [EQR]),
Equity(158 [EQT]),
Equity(159 [ES]),
Equity(160 [ESRX]),
Equity(161 [ESS]),
Equity(162 [ETFC]),
Equity(163 [ETN]),
Equity(164 [ETR]),
Equity(165 [EVHC]),
Equity(166 [EW]),
Equity(167 [EXC]),
Equity(168 [EXPD]),
```

```
Equity(169 [EXPE]),
Equity(170 [EXR]),
Equity(171 [F]),
Equity(172 [FAST]),
Equity(173 [FB]),
Equity(174 [FBHS]),
Equity(175 [FCX]),
Equity(176 [FDX]),
Equity(177 [FE]),
Equity(178 [FFIV]),
Equity(179 [FIS]),
Equity(180 [FISV]),
Equity(181 [FITB]),
Equity(182 [FL]),
Equity(183 [FLIR]),
Equity(184 [FLR]),
Equity(185 [FLS]),
Equity(186 [FMC]),
Equity(187 [FOX]),
Equity(188 [FOXA]),
Equity(189 [FRT]),
Equity(190 [FTI]),
Equity(191 [GD]),
Equity(192 [GE]),
Equity(193 [GGP]),
Equity(194 [GILD]),
Equity(195 [GIS]),
Equity(196 [GLW]),
Equity(197 [GM]),
Equity(198 [GOOG]),
Equity(199 [GOOGL]),
Equity(200 [GPC]),
Equity(201 [GPN]),
Equity(202 [GPS]),
Equity(203 [GRMN]),
Equity(204 [GS]),
Equity(205 [GT]),
Equity(206 [GWW]),
Equity(207 [HAL]),
Equity(208 [HAS]),
Equity(209 [HBAN]),
Equity(210 [HBI]),
Equity(211 [HCA]),
Equity(212 [HCN]),
Equity(213 [HCP]),
Equity(214 [HD]),
Equity(215 [HES]),
Equity(216 [HIG]),
```

```
Equity(217 [HLT]),
Equity(218 [HOG]),
Equity(219 [HOLX]),
Equity(220 [HON]),
Equity(221 [HP]),
Equity(222 [HPE]),
Equity(223 [HPQ]),
Equity(224 [HRB]),
Equity(225 [HRL]),
Equity(226 [HRS]),
Equity(227 [HSIC]),
Equity(228 [HST]),
Equity(229 [HSY]),
Equity(230 [HUM]),
Equity(231 [IBM]),
Equity(232 [ICE]),
Equity(233 [IDXX]),
Equity(234 [IFF]),
Equity(235 [ILMN]),
Equity(236 [INCY]),
Equity(237 [INFO]),
Equity(238 [INTC]),
Equity(239 [INTU]),
Equity(240 [IP]),
Equity(241 [IPG]),
Equity(242 [IR]),
Equity(243 [IRM]),
Equity(244 [ISRG]),
Equity(245 [IT]),
Equity(246 [ITW]),
Equity(247 [IVZ]),
Equity(248 [JBHT]),
Equity(249 [JCI]),
Equity(250 [JEC]),
Equity(251 [JNJ]),
Equity(252 [JNPR]),
Equity(253 [JPM]),
Equity(254 [JWN]),
Equity(255 [K]),
Equity(256 [KEY]),
Equity(257 [KHC]),
Equity(258 [KIM]),
Equity(259 [KLAC]),
Equity(260 [KMB]),
Equity(261 [KMI]),
Equity(262 [KMX]),
Equity(263 [KO]),
Equity(264 [KORS]),
```

```
Equity(265 [KR]),
Equity(266 [KSS]),
Equity(267 [KSU]),
Equity(268 [L]),
Equity(269 [LB]),
Equity(270 [LEG]),
Equity(271 [LEN]),
Equity(272 [LH]),
Equity(273 [LKQ]),
Equity(274 [LLL]),
Equity(275 [LLY]),
Equity(276 [LMT]),
Equity(277 [LNC]),
Equity(278 [LNT]),
Equity(279 [LOW]),
Equity(280 [LRCX]),
Equity(281 [LUK]),
Equity(282 [LUV]),
Equity(283 [LVLT]),
Equity(284 [LYB]),
Equity(285 [M]),
Equity(286 [MA]),
Equity(287 [MAA]),
Equity(288 [MAC]),
Equity(289 [MAR]),
Equity(290 [MAS]),
Equity(291 [MAT]),
Equity(292 [MCD]),
Equity(293 [MCHP]),
Equity(294 [MCK]),
Equity(295 [MCO]),
Equity(296 [MDLZ]),
Equity(297 [MDT]),
Equity(298 [MET]),
Equity(299 [MGM]),
Equity(300 [MHK]),
Equity(301 [MKC]),
Equity(302 [MLM]),
Equity(303 [MMC]),
Equity(304 [MNST]),
Equity(305 [MO]),
Equity(306 [MON]),
Equity(307 [MOS]),
Equity(308 [MPC]),
Equity(309 [MRK]),
Equity(310 [MRO]),
Equity(311 [MS]),
Equity(312 [MSFT]),
```

```
Equity(313 [MSI]),
Equity(314 [MTB]),
Equity(315 [MTD]),
Equity(316 [MU]),
Equity(317 [MYL]),
Equity(318 [NAVI]),
Equity(319 [NBL]),
Equity(320 [NDAQ]),
Equity(321 [NEE]),
Equity(322 [NEM]),
Equity(323 [NFLX]),
Equity(324 [NFX]),
Equity(325 [NI]),
Equity(326 [NKE]),
Equity(327 [NLSN]),
Equity(328 [NOC]),
Equity(329 [NOV]),
Equity(330 [NRG]),
Equity(331 [NSC]),
Equity(332 [NTAP]),
Equity(333 [NTRS]),
Equity(334 [NUE]),
Equity(335 [NVDA]),
Equity(336 [NWL]),
Equity(337 [NWS]),
Equity(338 [NWSA]),
Equity(339 [O]),
Equity(340 [OKE]),
Equity(341 [OMC]),
Equity(342 [ORCL]),
Equity(343 [ORLY]),
Equity(344 [OXY]),
Equity(345 [PAYX]),
Equity(346 [PBCT]),
Equity(347 [PCAR]),
Equity(348 [PCG]),
Equity(349 [PDCO]),
Equity(350 [PEG]),
Equity(351 [PEP]),
Equity(352 [PFE]),
Equity(353 [PFG]),
Equity(354 [PG]),
Equity(355 [PGR]),
Equity(356 [PH]),
Equity(357 [PHM]),
Equity(358 [PKG]),
Equity(359 [PKI]),
Equity(360 [PLD]),
```

```
Equity(361 [PM]),
Equity(362 [PNC]),
Equity(363 [PNR]),
Equity(364 [PNW]),
Equity(365 [PPG]),
Equity(366 [PPL]),
Equity(367 [PRGO]),
Equity(368 [PRU]),
Equity(369 [PSA]),
Equity(370 [PSX]),
Equity(371 [PVH]),
Equity(372 [PWR]),
Equity(373 [PX]),
Equity(374 [PXD]),
Equity(375 [PYPL]),
Equity(376 [QCOM]),
Equity(377 [QRVO]),
Equity(378 [RCL]),
Equity(379 [RE]),
Equity(380 [REG]),
Equity(381 [REGN]),
Equity(382 [RF]),
Equity(383 [RHI]),
Equity(384 [RHT]),
Equity(385 [RJF]),
Equity(386 [RL]),
Equity(387 [RMD]),
Equity(388 [ROK]),
Equity(389 [ROP]),
Equity(390 [ROST]),
Equity(391 [RRC]),
Equity(392 [RSG]),
Equity(393 [RTN]),
Equity(394 [SBAC]),
Equity(395 [SBUX]),
Equity(396 [SCG]),
Equity(397 [SCHW]),
Equity(398 [SEE]),
Equity(399 [SHW]),
Equity(400 [SIG]),
Equity(401 [SJM]),
Equity(402 [SLB]),
Equity(403 [SLG]),
Equity(404 [SNA]),
Equity(405 [SNI]),
Equity(406 [SNPS]),
Equity(407 [SO]),
Equity(408 [SPG]),
```

```
Equity(409 [SPLS]),
Equity(410 [SRCL]),
Equity(411 [SRE]),
Equity(412 [STI]),
Equity(413 [STT]),
Equity(414 [STX]),
Equity(415 [STZ]),
Equity(416 [SWK]),
Equity(417 [SWKS]),
Equity(418 [SYF]),
Equity(419 [SYK]),
Equity(420 [SYMC]),
Equity(421 [SYY]),
Equity(422 [T]),
Equity(423 [TAP]),
Equity(424 [TDG]),
Equity(425 [TEL]),
Equity(426 [TGT]),
Equity(427 [TIF]),
Equity(428 [TJX]),
Equity(429 [TMK]),
Equity(430 [TMO]),
Equity(431 [TRIP]),
Equity(432 [TROW]),
Equity(433 [TRV]),
Equity(434 [TSCO]),
Equity(435 [TSN]),
Equity(436 [TSS]),
Equity(437 [TWX]),
Equity(438 [TXN]),
Equity(439 [TXT]),
Equity(440 [UAA]),
Equity(441 [UAL]),
Equity(442 [UDR]),
Equity(443 [UHS]),
Equity(444 [ULTA]),
Equity(445 [UNH]),
Equity(446 [UNM]),
Equity(447 [UNP]),
Equity(448 [UPS]),
Equity(449 [URI]),
Equity(450 [USB]),
Equity(451 [UTX]),
Equity(452 [V]),
Equity(453 [VAR]),
Equity(454 [VFC]),
Equity(455 [VIAB]),
Equity(456 [VLO]),
```

```
 Equity(457 [VMC]),
 Equity(458 [VNO]),
 Equity(459 [VRSK]),
 Equity(460 [VRSN]),
 Equity(461 [VRTX]),
 Equity(462 [VTR]),
 Equity(463 [VZ]),
 Equity(464 [WAT]),
 Equity(465 [WBA]),
 Equity(466 [WDC]),
 Equity(467 [WEC]),
 Equity(468 [WFC]),
 Equity(469 [WHR]),
 Equity(471 [WM]),
 Equity(472 [WMB]),
 Equity(473 [WMT]),
 Equity(474 [WRK]),
 Equity(475 [WU]),
 Equity(476 [WY]),
 Equity(477 [WYN]),
 Equity(478 [WYNN]),
 Equity(479 [XEC]),
 Equity(480 [XEL]),
 Equity(481 [XL]),
 Equity(482 [XLNX]),
 Equity(483 [XOM]),
 Equity(484 [XRAY]),
 Equity(485 [XRX]),
 Equity(486 [XYL]),
 Equity(487 [YUM]),
 Equity(488 [ZBH]),
 Equity(489 [ZION]),
 Equity(490 [ZTS])]
```

1.3.3 Get Returns

Not that we have our pipeline built, let's access the returns data. We'll start by building a data portal.

```
In [14]: from zipline.data.data_portal import DataPortal
```

```
data_portal = DataPortal(
    bundle_data.asset_finder,
    trading_calendar=trading_calendar,
    first_trading_day=bundle_data.equity_daily_bar_reader.first_trading_day,
    equity_minute_reader=None,
    equity_daily_reader=bundle_data.equity_daily_bar_reader,
```

```
            adjustment_reader=bundle_data.adjustment_reader)
```

To make the code easier to read, we've built the helper function `get_pricing` to get the pricing from the data portal.

```
In [15]: def get_pricing(data_portal, trading_calendar, assets, start_date, end_date, field='clo
             end_dt = pd.Timestamp(end_date.strftime('%Y-%m-%d'), tz='UTC', offset='C')
             start_dt = pd.Timestamp(start_date.strftime('%Y-%m-%d'), tz='UTC', offset='C')

             end_loc = trading_calendar.closes.index.get_loc(end_dt)
             start_loc = trading_calendar.closes.index.get_loc(start_dt)

             return data_portal.get_history_window(
                 assets=assets,
                 end_dt=end_dt,
                 bar_count=end_loc - start_loc,
                 frequency='1d',
                 field=field,
                 data_frequency='daily')
```

2 Alpha Factors

It's time to start working on the alpha factors. In this project, we'll use the following factors: - Momentum 1 Year Factor - Mean Reversion 5 Day Sector Neutral Smoothed Factor - Overnight Sentiment Smoothed Factor

```
In [16]: from zipline.pipeline.factors import CustomFactor, DailyReturns, Returns, SimpleMovingA
         from zipline.pipeline.data import USEquityPricing


         factor_start_date = universe_end_date - pd.DateOffset(years=3, days=2)
         sector = project_helper.Sector()

         def momentum_1yr(window_length, universe, sector):
             return Returns(window_length=window_length, mask=universe) \
                 .demean(groupby=sector) \
                 .rank() \
                 .zscore()

         def mean_reversion_5day_sector_neutral_smoothed(window_length, universe, sector):
             unsmoothed_factor = -Returns(window_length=window_length, mask=universe) \
                 .demean(groupby=sector) \
                 .rank() \
                 .zscore()
             return SimpleMovingAverage(inputs=[unsmoothed_factor], window_length=window_length)
                 .rank() \
                 .zscore()
```

```
class CTO(Returns):
    """
    Computes the overnight return, per hypothesis from
    https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2554010
    """
    inputs = [USEquityPricing.open, USEquityPricing.close]

    def compute(self, today, assets, out, opens, closes):
        """
        The opens and closes matrix is 2 rows x N assets, with the most recent at the b
        As such, opens[-1] is the most recent open, and closes[0] is the earlier close
        """
        out[:] = (opens[-1] - closes[0]) / closes[0]

class TrailingOvernightReturns(Returns):
    """
    Sum of trailing 1m O/N returns
    """
    window_safe = True

    def compute(self, today, asset_ids, out, cto):
        out[:] = np.nansum(cto, axis=0)

def overnight_sentiment_smoothed(cto_window_length, trail_overnight_returns_window_leng
    cto_out = CTO(mask=universe, window_length=cto_window_length)
    unsmoothed_factor = TrailingOvernightReturns(inputs=[cto_out], window_length=trail_
        .rank() \
        .zscore()
    return SimpleMovingAverage(inputs=[unsmoothed_factor], window_length=trail_overnigh
        .rank() \
        .zscore()
```

2.0.1 Combine the Factors to a single Pipeline

Let's add the factors to a pipeline.

```
In [17]: universe = AverageDollarVolume(window_length=120).top(500)
         sector = project_helper.Sector()

         pipeline = Pipeline(screen=universe)
         pipeline.add(
             momentum_1yr(252, universe, sector),
             'Momentum_1YR')
         pipeline.add(
             mean_reversion_5day_sector_neutral_smoothed(20, universe, sector),
             'Mean_Reversion_Sector_Neutral_Smoothed')
         pipeline.add(
             overnight_sentiment_smoothed(2, 10, universe),
```

```
        'Overnight_Sentiment_Smoothed')
```

2.1 Features and Labels

Let's create some features that we think will help the model make predictions. ### "Universal" Quant Features To capture the universe, we'll use the following as features: - Stock Volatility 20d, 120d - Stock Dollar Volume 20d, 120d - Sector

```
In [18]: pipeline.add(AnnualizedVolatility(window_length=20, mask=universe).rank().zscore(), 'vo
         pipeline.add(AnnualizedVolatility(window_length=120, mask=universe).rank().zscore(), 'v
         pipeline.add(AverageDollarVolume(window_length=20, mask=universe).rank().zscore(), 'adv
         pipeline.add(AverageDollarVolume(window_length=120, mask=universe).rank().zscore(), 'ad
         pipeline.add(sector, 'sector_code')
```

2.1.1 Regime Features

We are going to try to capture market-wide regimes. To do that, we'll use the following features: - High and low volatility 20d, 120d - High and low dispersion 20d, 120d

```
In [19]: class MarketDispersion(CustomFactor):
             inputs = [DailyReturns()]
             window_length = 1
             window_safe = True

             def compute(self, today, assets, out, returns):
                 # returns are days in rows, assets across columns
                 out[:] = np.sqrt(np.nanmean((returns - np.nanmean(returns))**2))


         pipeline.add(SimpleMovingAverage(inputs=[MarketDispersion(mask=universe)], window_lengt
         pipeline.add(SimpleMovingAverage(inputs=[MarketDispersion(mask=universe)], window_lengt
```

```
In [20]: class MarketVolatility(CustomFactor):
             inputs = [DailyReturns()]
             window_length = 1
             window_safe = True

             def compute(self, today, assets, out, returns):
                 mkt_returns = np.nanmean(returns, axis=1)
                 out[:] = np.sqrt(260.* np.nanmean((mkt_returns-np.nanmean(mkt_returns))**2))


         pipeline.add(MarketVolatility(window_length=20), 'market_vol_20d')
         pipeline.add(MarketVolatility(window_length=120), 'market_vol_120d')
```

2.1.2 Target

Let's try to predict the go forward 1-week return. When doing this, it's important to quantize the target. The factor we create is the trailing 5-day return.

```
In [21]: pipeline.add(Returns(window_length=5, mask=universe).quantiles(2), 'return_5d')
         pipeline.add(Returns(window_length=5, mask=universe).quantiles(25), 'return_5d_p')
```

2.1.3 Date Features

Let's make columns for the trees to split on that might capture trader/investor behavior due to calendar anomalies.

```
In [22]: all_factors = engine.run_pipeline(pipeline, factor_start_date, universe_end_date)

         all_factors['is_Janaury'] = all_factors.index.get_level_values(0).month == 1
         all_factors['is_December'] = all_factors.index.get_level_values(0).month == 12
         all_factors['weekday'] = all_factors.index.get_level_values(0).weekday
         all_factors['quarter'] = all_factors.index.get_level_values(0).quarter
         all_factors['qtr_yr'] = all_factors.quarter.astype('str') + '_' + all_factors.index.get
         all_factors['month_end'] = all_factors.index.get_level_values(0).isin(pd.date_range(sta
         all_factors['month_start'] = all_factors.index.get_level_values(0).isin(pd.date_range(s
         all_factors['qtr_end'] = all_factors.index.get_level_values(0).isin(pd.date_range(start
         all_factors['qtr_start'] = all_factors.index.get_level_values(0).isin(pd.date_range(sta

         all_factors.head()
```

Out[22]:

		Mean_Reversion_Sector_Neutral_Smoothed	\
2013-01-03 00:00:00+00:00	Equity(0 [A])	-0.26276899	
	Equity(1 [AAL])	0.09992624	
	Equity(2 [AAP])	1.66913824	
	Equity(3 [AAPL])	1.69874602	
	Equity(4 [ABBV])	nan	

		Momentum_1YR	\
2013-01-03 00:00:00+00:00	Equity(0 [A])	-1.20797813	
	Equity(1 [AAL])	1.71347052	
	Equity(2 [AAP])	-1.53506144	
	Equity(3 [AAPL])	1.19311071	
	Equity(4 [ABBV])	nan	

		Overnight_Sentiment_Smoothed	\
2013-01-03 00:00:00+00:00	Equity(0 [A])	-1.48566901	
	Equity(1 [AAL])	0.91934963	
	Equity(2 [AAP])	1.50773340	
	Equity(3 [AAPL])	-1.36799226	
	Equity(4 [ABBV])	-0.25006310	

		adv_120d	adv_20d	\
2013-01-03 00:00:00+00:00	Equity(0 [A])	1.33857307	1.39741144	
	Equity(1 [AAL])	1.13999355	1.08115517	
	Equity(2 [AAP])	-0.30154668	-0.91934963	
	Equity(3 [AAPL])	1.72837731	1.72837731	

```
                           Equity(4 [ABBV]) -1.72837731 -1.64747455

                                                 dispersion_120d  dispersion_20d  \
2013-01-03 00:00:00+00:00 Equity(0 [A])               0.01326964      0.01117804
                          Equity(1 [AAL])             0.01326964      0.01117804
                          Equity(2 [AAP])             0.01326964      0.01117804
                          Equity(3 [AAPL])            0.01326964      0.01117804
                          Equity(4 [ABBV])            0.01459524      0.01459524

                                                 market_vol_120d  market_vol_20d  \
2013-01-03 00:00:00+00:00 Equity(0 [A])               0.12966421      0.13758558
                          Equity(1 [AAL])             0.12966421      0.13758558
                          Equity(2 [AAP])             0.12966421      0.13758558
                          Equity(3 [AAPL])            0.12966421      0.13758558
                          Equity(4 [ABBV])            0.12966421      0.13758558

                                                 return_5d    ...     \
2013-01-03 00:00:00+00:00 Equity(0 [A])                 0    ...
                          Equity(1 [AAL])               1    ...
                          Equity(2 [AAP])               0    ...
                          Equity(3 [AAPL])              1    ...
                          Equity(4 [ABBV])             -1    ...

                                                 volatility_20d  is_Janaury  \
2013-01-03 00:00:00+00:00 Equity(0 [A])             -1.21980876        True
                          Equity(1 [AAL])            1.56621970        True
                          Equity(2 [AAP])           -1.47040391        True
                          Equity(3 [AAPL])           1.61781282        True
                          Equity(4 [ABBV])                  nan        True

                                                 is_December  weekday quarter  \
2013-01-03 00:00:00+00:00 Equity(0 [A])                False        3       1
                          Equity(1 [AAL])              False        3       1
                          Equity(2 [AAP])              False        3       1
                          Equity(3 [AAPL])             False        3       1
                          Equity(4 [ABBV])             False        3       1

                                                 qtr_yr  month_end  month_start  \
2013-01-03 00:00:00+00:00 Equity(0 [A])          1_2013      False        False
                          Equity(1 [AAL])        1_2013      False        False
                          Equity(2 [AAP])        1_2013      False        False
                          Equity(3 [AAPL])       1_2013      False        False
                          Equity(4 [ABBV])       1_2013      False        False

                                                qtr_end qtr_start
2013-01-03 00:00:00+00:00 Equity(0 [A])          False     False
                          Equity(1 [AAL])        False     False
                          Equity(2 [AAP])        False     False
```

```
                                Equity(3 [AAPL])   False     False
                                Equity(4 [ABBV])   False     False

          [5 rows x 23 columns]
```

2.1.4 One Hot Encode Sectors

For the model to better understand the sector data, we'll one hot encode this data.

```
In [23]: sector_lookup = pd.read_csv(
             os.path.join(os.getcwd(), '..', '..', 'data', 'project_7_sector', 'labels.csv'),
             index_col='Sector_i')['Sector'].to_dict()
         sector_lookup

         sector_columns = []
         for sector_i, sector_name in sector_lookup.items():
             secotr_column = 'sector_{}'.format(sector_name)
             sector_columns.append(secotr_column)
             all_factors[secotr_column] = (all_factors['sector_code'] == sector_i)

         all_factors[sector_columns].head()

Out[23]:                                              sector_Healthcare  \
         2013-01-03 00:00:00+00:00 Equity(0 [A])                 True
                                   Equity(1 [AAL])              False
                                   Equity(2 [AAP])              False
                                   Equity(3 [AAPL])             False
                                   Equity(4 [ABBV])              True

                                                       sector_Technology  \
         2013-01-03 00:00:00+00:00 Equity(0 [A])                 False
                                   Equity(1 [AAL])               False
                                   Equity(2 [AAP])               False
                                   Equity(3 [AAPL])               True
                                   Equity(4 [ABBV])              False

                                                       sector_Consumer Defensive  \
         2013-01-03 00:00:00+00:00 Equity(0 [A])                           False
                                   Equity(1 [AAL])                         False
                                   Equity(2 [AAP])                         False
                                   Equity(3 [AAPL])                        False
                                   Equity(4 [ABBV])                        False

                                                       sector_Industrials  \
         2013-01-03 00:00:00+00:00 Equity(0 [A])                  False
                                   Equity(1 [AAL])                 True
                                   Equity(2 [AAP])                 False
                                   Equity(3 [AAPL])                False
```

```
                           Equity(4 [ABBV])              False

                                                  sector_Utilities  \
2013-01-03 00:00:00+00:00 Equity(0 [A])                    False
                           Equity(1 [AAL])                 False
                           Equity(2 [AAP])                 False
                           Equity(3 [AAPL])                False
                           Equity(4 [ABBV])                False

                                                  sector_Financial Services  \
2013-01-03 00:00:00+00:00 Equity(0 [A])                              False
                           Equity(1 [AAL])                           False
                           Equity(2 [AAP])                           False
                           Equity(3 [AAPL])                          False
                           Equity(4 [ABBV])                          False

                                                  sector_Real Estate  \
2013-01-03 00:00:00+00:00 Equity(0 [A])                      False
                           Equity(1 [AAL])                   False
                           Equity(2 [AAP])                   False
                           Equity(3 [AAPL])                  False
                           Equity(4 [ABBV])                  False

                                                  sector_Communication Services  \
2013-01-03 00:00:00+00:00 Equity(0 [A])                                    False
                           Equity(1 [AAL])                                 False
                           Equity(2 [AAP])                                 False
                           Equity(3 [AAPL])                                False
                           Equity(4 [ABBV])                                False

                                                  sector_Consumer Cyclical  \
2013-01-03 00:00:00+00:00 Equity(0 [A])                            False
                           Equity(1 [AAL])                         False
                           Equity(2 [AAP])                          True
                           Equity(3 [AAPL])                        False
                           Equity(4 [ABBV])                        False

                                                  sector_Energy  \
2013-01-03 00:00:00+00:00 Equity(0 [A])                  False
                           Equity(1 [AAL])                 False
                           Equity(2 [AAP])                 False
                           Equity(3 [AAPL])                False
                           Equity(4 [ABBV])                False

                                                  sector_Basic Materials
2013-01-03 00:00:00+00:00 Equity(0 [A])                          False
                           Equity(1 [AAL])                       False
                           Equity(2 [AAP])                       False
```

```
                                    Equity(3 [AAPL])                 False
                                    Equity(4 [ABBV])                 False
```

2.1.5 Shift Target

We'll use shifted 5 day returns for training the model.

```
In [24]: all_factors['target'] = all_factors.groupby(level=1)['return_5d'].shift(-5)

         all_factors[['return_5d','target']].reset_index().sort_values(['level_1', 'level_0']).h
```

Out[24]:

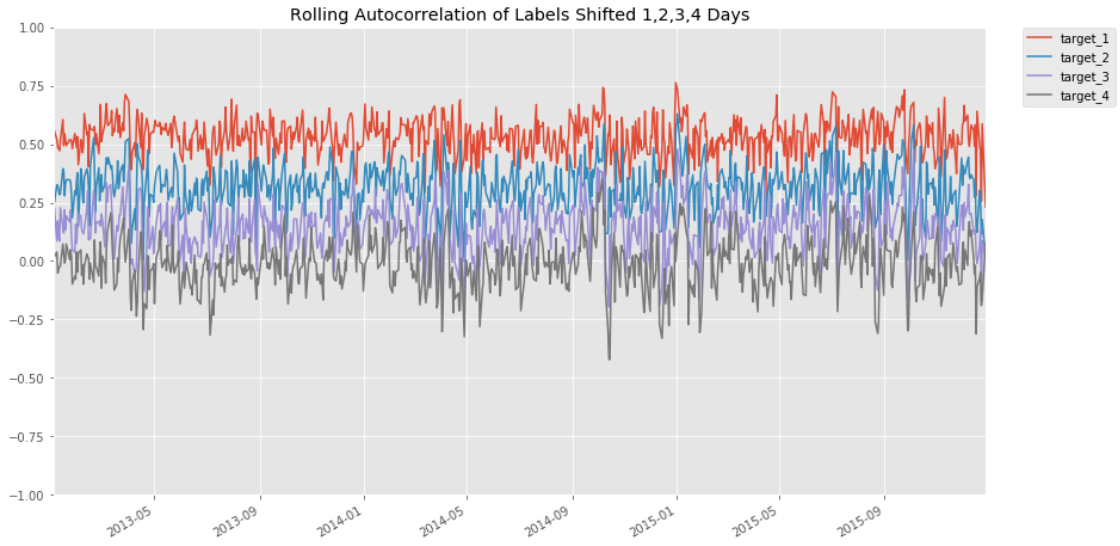
	level_0	level_1	return_5d	target
0	2013-01-03 00:00:00+00:00	Equity(0 [A])	0	0.00000000
471	2013-01-04 00:00:00+00:00	Equity(0 [A])	0	0.00000000
942	2013-01-07 00:00:00+00:00	Equity(0 [A])	0	0.00000000
1413	2013-01-08 00:00:00+00:00	Equity(0 [A])	0	1.00000000
1884	2013-01-09 00:00:00+00:00	Equity(0 [A])	0	0.00000000
2355	2013-01-10 00:00:00+00:00	Equity(0 [A])	0	0.00000000
2826	2013-01-11 00:00:00+00:00	Equity(0 [A])	0	0.00000000
3297	2013-01-14 00:00:00+00:00	Equity(0 [A])	0	0.00000000
3768	2013-01-15 00:00:00+00:00	Equity(0 [A])	1	0.00000000
4239	2013-01-16 00:00:00+00:00	Equity(0 [A])	0	0.00000000

2.1.6 IID Check of Target

Let's see if the returns are independent and identically distributed.

```
In [25]: from scipy.stats import spearmanr


         def sp(group, col1_name, col2_name):
             x = group[col1_name]
             y = group[col2_name]
             return spearmanr(x, y)[0]


         all_factors['target_p'] = all_factors.groupby(level=1)['return_5d_p'].shift(-5)
         all_factors['target_1'] = all_factors.groupby(level=1)['return_5d'].shift(-4)
         all_factors['target_2'] = all_factors.groupby(level=1)['return_5d'].shift(-3)
         all_factors['target_3'] = all_factors.groupby(level=1)['return_5d'].shift(-2)
         all_factors['target_4'] = all_factors.groupby(level=1)['return_5d'].shift(-1)

         g = all_factors.dropna().groupby(level=0)
         for i in range(4):
             label = 'target_'+str(i+1)
             ic = g.apply(sp, 'target', label)
             ic.plot(ylim=(-1, 1), label=label)
         plt.legend(bbox_to_anchor=(1.04, 1), borderaxespad=0)
```

```
plt.title('Rolling Autocorrelation of Labels Shifted 1,2,3,4 Days')
plt.show()
```



2.1.7 Question: What do you observe in the rolling autocorrelation of labels shifted?

#TODO: Put Answer In this Cell

The shifted labels correlate because they move in the same direction always. The autocorrelation decreases when the number of the lags increases

2.1.8 Train/Valid/Test Splits

Now let's split the data into a train, validation, and test dataset. Implement the function `train_valid_test_split` to split the input samples, `all_x`, and targets values, `all_y` into a train, validation, and test dataset. The proportion sizes are `train_size`, `valid_size`, `test_size` respectively.

When splitting, make sure the data is in order from train, validation, and test respectivly. Say `train_size` is 0.7, `valid_size` is 0.2, and `test_size` is 0.1. The first 70 percent of `all_x` and `all_y` would be the train set. The next 20 percent of `all_x` and `all_y` would be the validation set. The last 10 percent of `all_x` and `all_y` would be the test set. Make sure not split a day between multiple datasets. It should be contained within a single dataset.

```
In [27]: def train_valid_test_split(all_x, all_y, train_size, valid_size, test_size):
             """
             Generate the train, validation, and test dataset.

             Parameters
             ----------
             all_x : DataFrame
                 All the input samples
```

```
    all_y : Pandas Series
        All the target values
    train_size : float
        The proportion of the data used for the training dataset
    valid_size : float
        The proportion of the data used for the validation dataset
    test_size : float
        The proportion of the data used for the test dataset

    Returns
    -------
    x_train : DataFrame
        The train input samples
    x_valid : DataFrame
        The validation input samples
    x_test : DataFrame
        The test input samples
    y_train : Pandas Series
        The train target values
    y_valid : Pandas Series
        The validation target values
    y_test : Pandas Series
        The test target values
    """
    assert train_size >= 0 and train_size <= 1.0
    assert valid_size >= 0 and valid_size <= 1.0
    assert test_size >= 0 and test_size <= 1.0
    assert train_size + valid_size + test_size == 1.0

    # TODO: Implement
    dates = all_x.index.levels[0]

    train_split = round(train_size * len(dates))
    valid_split = round(valid_size * len(dates)) + train_split
    test_split = round(test_size * len(dates)) + valid_split

    x_train = all_x.loc[:dates[train_split-1]]
    x_valid = all_x.loc[dates[train_split]:dates[valid_split-1]]
    x_test = all_x.loc[dates[valid_split]:]

    y_train = all_y.loc[:dates[train_split-1]]
    y_valid = all_y.loc[dates[train_split]:dates[valid_split-1]]
    y_test = all_y.loc[dates[valid_split]:]

    return x_train, x_valid, x_test, y_train, y_valid, y_test


project_tests.test_train_valid_test_split(train_valid_test_split)
```

```
Tests Passed
```

With `train_valid_test_split` implemented, let's split the data into a train, validation, and test set. For this, we'll use some of the features and the 5 day returns for our target.

```
In [28]: features = [
             'Mean_Reversion_Sector_Neutral_Smoothed', 'Momentum_1YR',
             'Overnight_Sentiment_Smoothed', 'adv_120d', 'adv_20d',
             'dispersion_120d', 'dispersion_20d', 'market_vol_120d',
             'market_vol_20d', 'volatility_20d',
             'is_Janaury', 'is_December', 'weekday',
             'month_end', 'month_start', 'qtr_end', 'qtr_start'] + sector_columns
         target_label = 'target'

         temp = all_factors.dropna().copy()
         X = temp[features]
         y = temp[target_label]

         X_train, X_valid, X_test, y_train, y_valid, y_test = train_valid_test_split(X, y, 0.6,

         X_train.head()
```

Out[28]:

		Mean_Reversion_Sector_Neutral_Smoothed	Momentum_1YR	Overnight_Sentiment_Smoothed	adv_120d	adv_20d
2013-01-03 00:00:00+00:00	Equity(0 [A])	-0.26276899	-1.20797813	-1.48566901	1.33857307	1.39741144
	Equity(1 [AAL])	0.09992624	1.71347052	0.91934963	1.13999355	1.08115517
	Equity(2 [AAP])	1.66913824	-1.53506144	1.50773340	-0.30154668	-0.91934963
	Equity(3 [AAPL])	1.69874602	1.19311071	-1.36799226		
	Equity(5 [ABC])	-1.11399249	-0.50920924	-0.02941919		

```
                           Equity(3 [AAPL])  1.72837731  1.72837731
                           Equity(5 [ABC])   0.17651513  0.22799871

                                             dispersion_120d  dispersion_20d  \
2013-01-03 00:00:00+00:00 Equity(0 [A])           0.01326964      0.01117804
                           Equity(1 [AAL])        0.01326964      0.01117804
                           Equity(2 [AAP])        0.01326964      0.01117804
                           Equity(3 [AAPL])       0.01326964      0.01117804
                           Equity(5 [ABC])        0.01326964      0.01117804

                                             market_vol_120d  market_vol_20d  \
2013-01-03 00:00:00+00:00 Equity(0 [A])           0.12966421      0.13758558
                           Equity(1 [AAL])        0.12966421      0.13758558
                           Equity(2 [AAP])        0.12966421      0.13758558
                           Equity(3 [AAPL])       0.12966421      0.13758558
                           Equity(5 [ABC])        0.12966421      0.13758558

                                             volatility_20d  \
2013-01-03 00:00:00+00:00 Equity(0 [A])         -1.21980876
                           Equity(1 [AAL])       1.56621970
                           Equity(2 [AAP])      -1.47040391
                           Equity(3 [AAPL])      1.61781282
                           Equity(5 [ABC])      -1.36721767

                                                       ...            \
2013-01-03 00:00:00+00:00 Equity(0 [A])                ...
                           Equity(1 [AAL])             ...
                           Equity(2 [AAP])             ...
                           Equity(3 [AAPL])            ...
                           Equity(5 [ABC])             ...

                                             sector_Technology  \
2013-01-03 00:00:00+00:00 Equity(0 [A])                 False
                           Equity(1 [AAL])              False
                           Equity(2 [AAP])              False
                           Equity(3 [AAPL])              True
                           Equity(5 [ABC])              False

                                             sector_Consumer Defensive  \
2013-01-03 00:00:00+00:00 Equity(0 [A])                          False
                           Equity(1 [AAL])                       False
                           Equity(2 [AAP])                       False
                           Equity(3 [AAPL])                      False
                           Equity(5 [ABC])                       False

                                             sector_Industrials  \
2013-01-03 00:00:00+00:00 Equity(0 [A])                  False
                           Equity(1 [AAL])                True
```

```
                                    Equity(2 [AAP])                False
                                    Equity(3 [AAPL])               False
                                    Equity(5 [ABC])                False

                                                                   sector_Utilities  \
2013-01-03 00:00:00+00:00 Equity(0 [A])                                       False
                          Equity(1 [AAL])                                     False
                          Equity(2 [AAP])                                     False
                          Equity(3 [AAPL])                                    False
                          Equity(5 [ABC])                                     False

                                                                   sector_Financial Services  \
2013-01-03 00:00:00+00:00 Equity(0 [A])                                                 False
                          Equity(1 [AAL])                                               False
                          Equity(2 [AAP])                                               False
                          Equity(3 [AAPL])                                              False
                          Equity(5 [ABC])                                               False

                                                                   sector_Real Estate  \
2013-01-03 00:00:00+00:00 Equity(0 [A])                                         False
                          Equity(1 [AAL])                                       False
                          Equity(2 [AAP])                                       False
                          Equity(3 [AAPL])                                      False
                          Equity(5 [ABC])                                       False

                                                                   sector_Communication Services  \
2013-01-03 00:00:00+00:00 Equity(0 [A])                                                   False
                          Equity(1 [AAL])                                                 False
                          Equity(2 [AAP])                                                 False
                          Equity(3 [AAPL])                                                False
                          Equity(5 [ABC])                                                 False

                                                                   sector_Consumer Cyclical  \
2013-01-03 00:00:00+00:00 Equity(0 [A])                                               False
                          Equity(1 [AAL])                                             False
                          Equity(2 [AAP])                                              True
                          Equity(3 [AAPL])                                            False
                          Equity(5 [ABC])                                             False

                                                                   sector_Energy  \
2013-01-03 00:00:00+00:00 Equity(0 [A])                                    False
                          Equity(1 [AAL])                                  False
                          Equity(2 [AAP])                                  False
                          Equity(3 [AAPL])                                 False
                          Equity(5 [ABC])                                  False

                                                                   sector_Basic Materials
2013-01-03 00:00:00+00:00 Equity(0 [A])                                           False
```

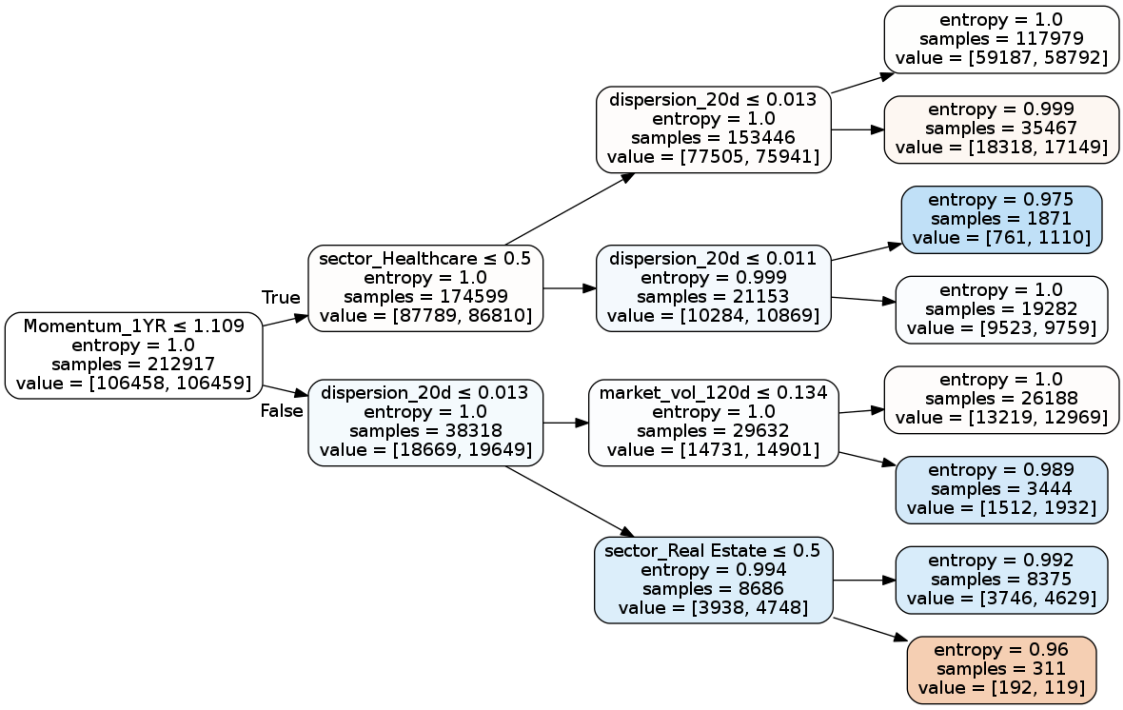
```
                                Equity(1 [AAL])                  False
                                Equity(2 [AAP])                  False
                                Equity(3 [AAPL])                 False
                                Equity(5 [ABC])                  False

[5 rows x 28 columns]
```

2.2 Random Forests

2.2.1 Visualize a Simple Tree

Let's see how a single tree would look using our data.

```
In [29]: from IPython.display import display
         from sklearn.tree import DecisionTreeClassifier


         # This is to get consistent results between each run.
         clf_random_state = 0

         simple_clf = DecisionTreeClassifier(
             max_depth=3,
             criterion='entropy',
             random_state=clf_random_state)
         simple_clf.fit(X_train, y_train)

         display(project_helper.plot_tree_classifier(simple_clf, feature_names=features))
         project_helper.rank_features_by_importance(simple_clf.feature_importances_, features)
```



```
     Feature                                  Importance
 1. dispersion_20d                        (0.46843340608552875)
 2. market_vol_120d                       (0.19358730055081794)
 3. sector_Real Estate                    (0.12813661543951108)
 4. Momentum_1YR                          (0.11211962120010666)
 5. sector_Healthcare                     (0.09772305672403549)
 6. sector_Basic Materials                (0.0)
 7. weekday                               (0.0)
 8. Overnight_Sentiment_Smoothed          (0.0)
 9. adv_120d                              (0.0)
10. adv_20d                               (0.0)
11. dispersion_120d                       (0.0)
12. market_vol_20d                        (0.0)
13. volatility_20d                        (0.0)
14. is_Janaury                            (0.0)
15. is_December                           (0.0)
16. month_end                             (0.0)
17. sector_Energy                         (0.0)
18. month_start                           (0.0)
19. qtr_end                               (0.0)
20. qtr_start                             (0.0)
21. sector_Technology                     (0.0)
22. sector_Consumer Defensive             (0.0)
23. sector_Industrials                    (0.0)
24. sector_Utilities                      (0.0)
25. sector_Financial Services             (0.0)
26. sector_Communication Services         (0.0)
27. sector_Consumer Cyclical              (0.0)
28. Mean_Reversion_Sector_Neutral_Smoothed (0.0)
```

2.2.2 Question: Why does dispersion_20d have the highest feature importance, when the first split is on the Momentum_1YR feature?

#TODO: Put Answer In this Cell

The first node to split does not not determine feature importance. Variable "dispersion_20d" has the higher information gain and is being used to split on other nodes because we can see that the entropy in the child nodes are lower than 1, so we can infer that is splitting cleaner.

2.2.3 Train Random Forests with Different Tree Sizes

Let's build models using different tree sizes to find the model that best generalizes. #### Parameters When building the models, we'll use the following parameters.

```
In [30]: n_days = 10
         n_stocks = 500
```

```
clf_parameters = {
    'criterion': 'entropy',
    'min_samples_leaf': n_stocks * n_days,
    'oob_score': True,
    'n_jobs': -1,
    'random_state': clf_random_state}
n_trees_l = [50, 100, 250, 500, 1000]
```

Recall from the lesson, that we'll choose a min_samples_leaf parameter to be small enough to allow the tree to fit the data with as much detail as possible, but not so much that it overfits. We can first propose 500, which is the number of assets in the estimation universe. Since we have about 500 stocks in the stock universe, we'll want at least 500 stocks in a leaf for the leaf to make a prediction that is representative. It's common to multiply this by 2,3,5 or 10, so we'd have min samples leaf of 500, 1000, 1500, 2500, and 5000. If we were to try these values, we'd notice that the model is "too good to be true" on the training data. A good rule of thumb for what is considered "too good to be true", and therefore a sign of overfitting, is if the sharpe ratio is greater than 4. Based on this, we recommend using min_sampes_leaf of 10 * 500, or 5,000.

Feel free to try other values for these parameters, but also keep in mind that making too many small adjustments to hyper-parameters can lead to overfitting even the validation data, and therefore lead to less generalizable performance on the out-of-sample test set. So when trying different parameter values, choose values that are different enough in scale (i.e. 10, 20, 100 instead of 10,11,12).

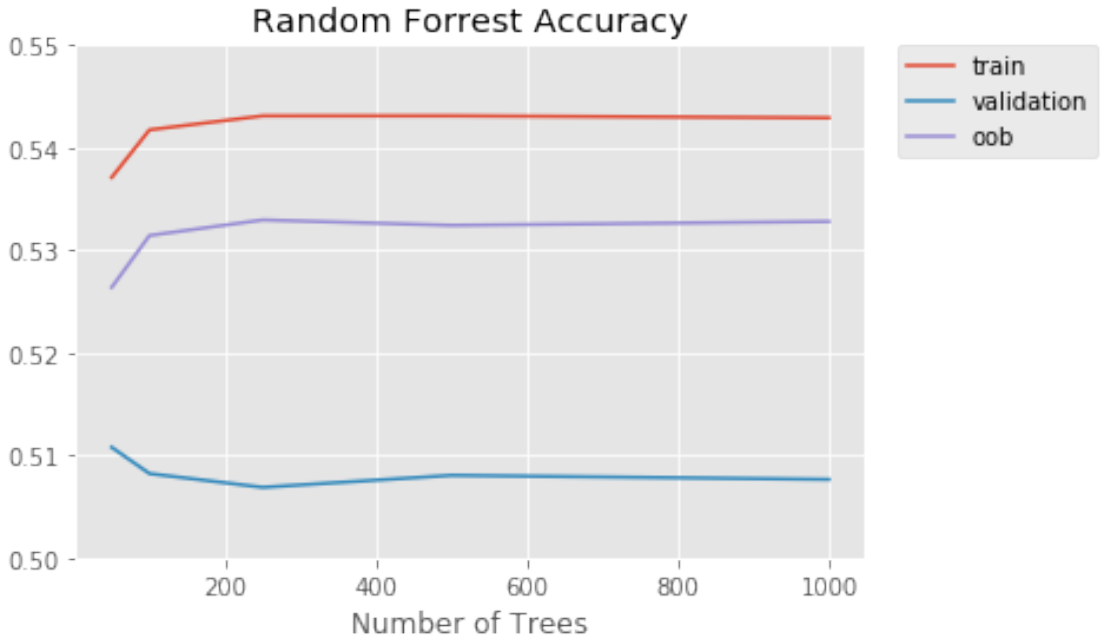
```
In [24]: from sklearn.ensemble import RandomForestClassifier


         train_score = []
         valid_score = []
         oob_score = []
         feature_importances = []

         for n_trees in tqdm(n_trees_l, desc='Training Models', unit='Model'):
             clf = RandomForestClassifier(n_trees, **clf_parameters)
             clf.fit(X_train, y_train)

             train_score.append(clf.score(X_train, y_train.values))
             valid_score.append(clf.score(X_valid, y_valid.values))
             oob_score.append(clf.oob_score_)
             feature_importances.append(clf.feature_importances_)
```

```
Training Models: 100%|| 5/5 [06:10<00:00, 74.12s/Model]
```

Let's look at the accuracy of the classifiers over the number of trees.

```
In [25]: project_helper.plot(
             [n_trees_l]*3,
             [train_score, valid_score, oob_score],
```

```
['train', 'validation', 'oob'],
'Random Forrest Accuracy',
'Number of Trees')
```



Question: 1) What do you observe with the accuracy vs tree size graph? 2) Does the graph indicate the model is overfitting or underfitting? Describe how it indicates this. *#TODO: Put Answer In this Cell*

- After about 250 trees the accuracy get flat and there is no benefit from using more trees.
- The validation accuracy is dropping so the tree must be overfitting.

Now let's looks at the average feature importance of the classifiers.

```
In [26]: print('Features Ranked by Average Importance:\n')
         project_helper.rank_features_by_importance(np.average(feature_importances, axis=0), fea
```

```
Features Ranked by Average Importance:

      Feature                                Importance
 1. dispersion_20d                     (0.12728185487944446)
 2. volatility_20d                     (0.1224273557087611)
 3. market_vol_120d                    (0.10513743366026762)
 4. market_vol_20d                     (0.10381897786540781)
 5. Momentum_1YR                       (0.0945797742561591)
 6. dispersion_120d                    (0.08005632079067652)
 7. Overnight_Sentiment_Smoothed       (0.0774409774770833)
```

```
 8. Mean_Reversion_Sector_Neutral_Smoothed (0.06912219864639728)
 9. adv_120d                               (0.0599502823871783)
10. adv_20d                                (0.054561747673325986)
11. sector_Healthcare                      (0.031652039665357366)
12. sector_Basic Materials                 (0.012388131735165828)
13. sector_Consumer Defensive              (0.011213976999730849)
14. sector_Industrials                     (0.010670656003428898)
15. sector_Financial Services              (0.009139543331685599)
16. weekday                                (0.008165080291861151)
17. sector_Real Estate                     (0.005712251925576841)
18. sector_Utilities                       (0.005304216249344825)
19. sector_Technology                      (0.004028115803846797)
20. sector_Consumer Cyclical               (0.0039155556743800946)
21. sector_Energy                          (0.002793447024879485)
22. is_Janaury                             (0.000495024286569785)
23. is_December                            (0.00011851439308367738)
24. month_end                              (2.652327038726357e-05)
25. qtr_end                                (0.0)
26. month_start                            (0.0)
27. qtr_start                              (0.0)
28. sector_Communication Services          (0.0)
```

You might notice that some of the features of low to no importance. We will be removing them when training the final model. ### Model Results Let's look at some additional metrics to see how well a model performs. We've created the function `show_sample_results` to show the following results of a model: - Sharpe Ratios - Factor Returns - Factor Rank Autocorrelation

In [27]:
```
import alphalens as al


        all_assets = all_factors.index.levels[1].values.tolist()
        all_pricing = get_pricing(
            data_portal,
            trading_calendar,
            all_assets,
            factor_start_date,
            universe_end_date)

        def show_sample_results(data, samples, classifier, factors, pricing=all_pricing):
            # Calculate the Alpha Score
            prob_array=[-1,1]
            alpha_score = classifier.predict_proba(samples).dot(np.array(prob_array))

            # Add Alpha Score to rest of the factors
            alpha_score_label = 'AI_ALPHA'
            factors_with_alpha = data.loc[samples.index].copy()
            factors_with_alpha[alpha_score_label] = alpha_score
```

```
    # Setup data for AlphaLens
    print('Cleaning Data...\n')
    factor_data = project_helper.build_factor_data(factors_with_alpha[factors + [alpha_
    print('\n-----------------------\n')

    # Calculate Factor Returns and Sharpe Ratio
    factor_returns = project_helper.get_factor_returns(factor_data)
    sharpe_ratio = project_helper.sharpe_ratio(factor_returns)

    # Show Results
    print('             Sharpe Ratios')
    print(sharpe_ratio.round(2))
    project_helper.plot_factor_returns(factor_returns)
    project_helper.plot_factor_rank_autocorrelation(factor_data)
```

Results Let's compare our AI Alpha factor to a few other factors. We'll use the following:

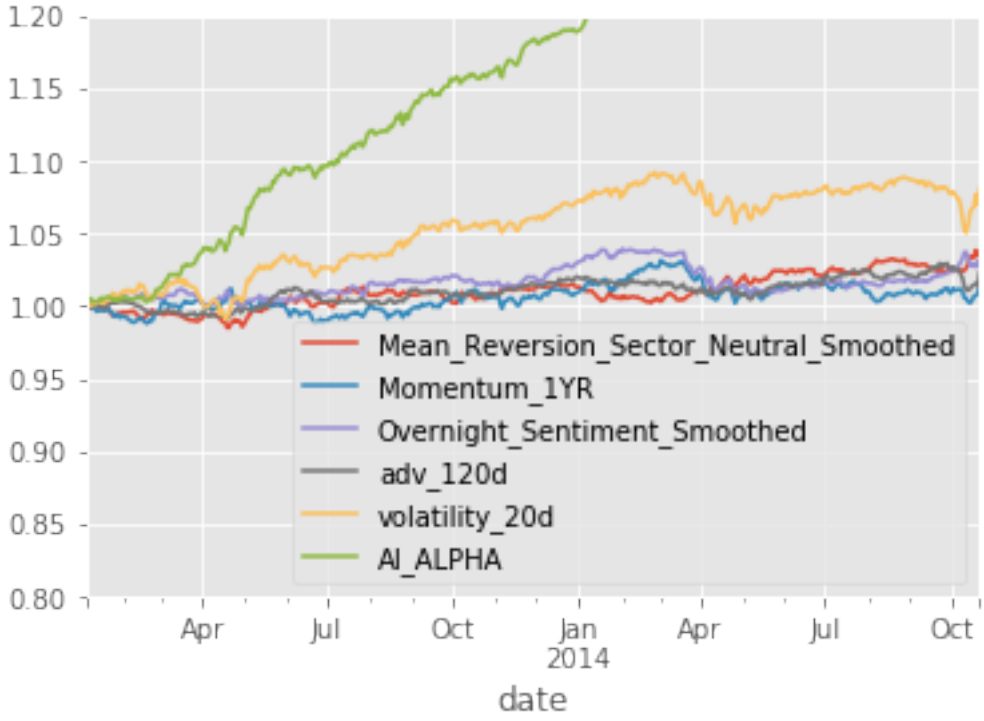
```
In [28]: factor_names = [
             'Mean_Reversion_Sector_Neutral_Smoothed',
             'Momentum_1YR',
             'Overnight_Sentiment_Smoothed',
             'adv_120d',
             'volatility_20d']
```

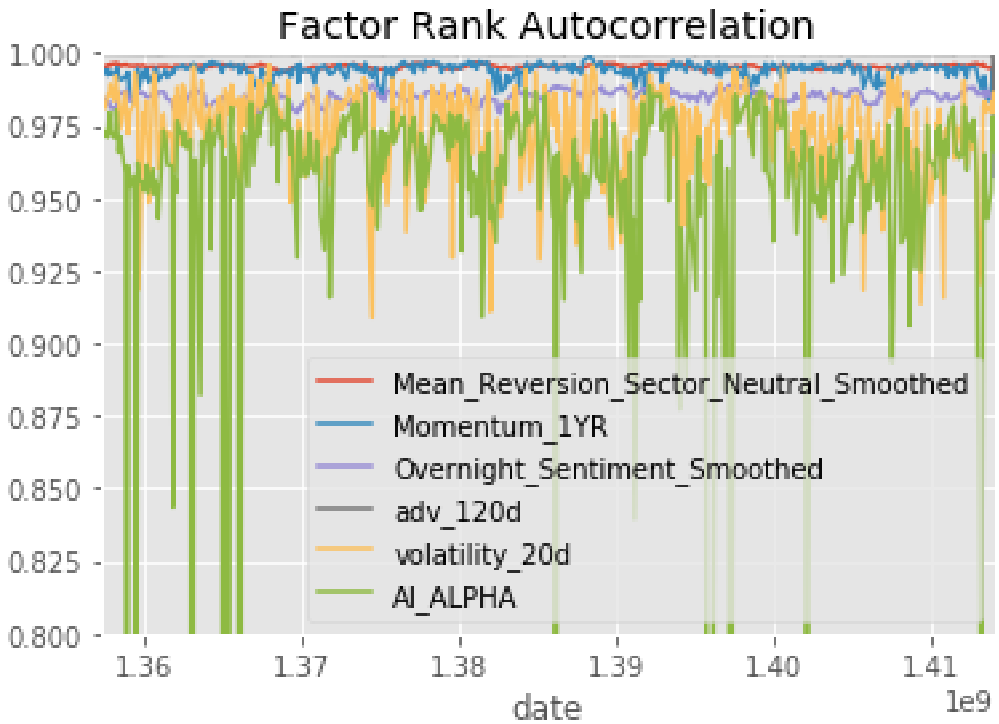
Training Prediction Let's see how well the model runs on training data.

```
In [29]: show_sample_results(all_factors, X_train, clf, factor_names)
```

```
Cleaning Data...

Dropped 0.2% entries from factor data: 0.2% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.2% entries from factor data: 0.2% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.2% entries from factor data: 0.2% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.2% entries from factor data: 0.2% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.2% entries from factor data: 0.2% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.2% entries from factor data: 0.2% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!

-----------------------

             Sharpe Ratios
Mean_Reversion_Sector_Neutral_Smoothed   0.87000000
```

```
Momentum_1YR                          0.28000000
Overnight_Sentiment_Smoothed          0.83000000
adv_120d                              0.62000000
volatility_20d                        1.18000000
AI_ALPHA                              5.78000000
dtype: float64
```



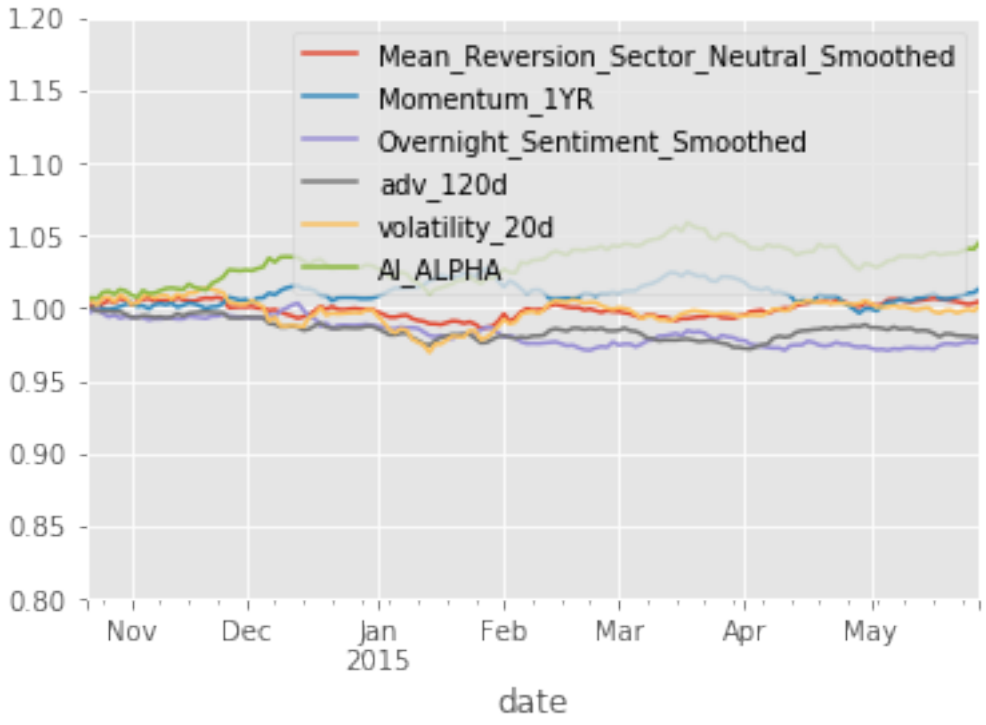


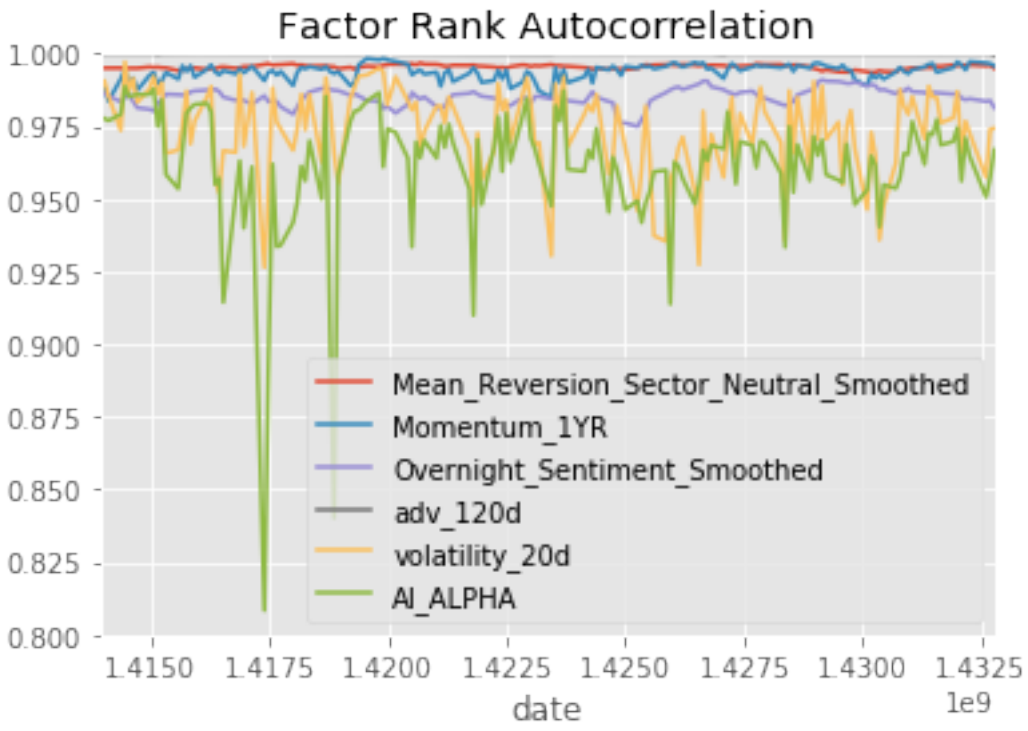
Validation Prediction Let's see how well the model runs on validation data.

```
In [30]: show_sample_results(all_factors, X_valid, clf, factor_names)
```

```
Cleaning Data...

Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!

-----------------------

                 Sharpe Ratios
Mean_Reversion_Sector_Neutral_Smoothed     0.32000000
```

```
Momentum_1YR                          0.72000000
Overnight_Sentiment_Smoothed         -1.40000000
adv_120d                             -1.44000000
volatility_20d                        0.09000000
AI_ALPHA                              1.94000000
dtype: float64
```





So that's pretty extraordinary. Even when the input factor returns are sideways to down, the AI Alpha is positive with Sharpe Ratio > 2. If we hope that this model will perform well in production we need to correct though for the non-IID labels and mitigate likely overfitting.

2.3 Overlapping Samples

Let's fix this by removing overlapping samples. We can do a number of things:

- Don't use overlapping samples
- Use BaggingClassifier's `max_samples`
- Build an ensemble of non-overlapping trees

In this project, we'll do all three methods and compare. ### Drop Overlapping Samples This is the simplest of the three methods. We'll just drop any overlapping samples from the dataset. Implement the `non_overlapping_samples` function to return a new dataset overlapping samples.

```
In [31]: def non_overlapping_samples(x, y, n_skip_samples, start_i=0):
             """
             Get the non overlapping samples.

             Parameters
             ----------
             x : DataFrame
                 The input samples
             y : Pandas Series
```

```
        The target values
    n_skip_samples : int
        The number of samples to skip
    start_i : int
        The starting index to use for the data

    Returns
    -------
    non_overlapping_x : 2 dimensional Ndarray
        The non overlapping input samples
    non_overlapping_y : 1 dimensional Ndarray
        The non overlapping target values
    """
    assert len(x.shape) == 2
    assert len(y.shape) == 1

    # TODO: Implement
    x_idx = list(x.index.levels[0][start_i::n_skip_samples+1])
    y_idx = list(y.index.levels[0][start_i::n_skip_samples+1])
    x_non_overlapping, y_non_overlapping = x.loc[x_idx], y.loc[y_idx]
    return x_non_overlapping, y_non_overlapping


project_tests.test_non_overlapping_samples(non_overlapping_samples)
```

```
Tests Passed
```

With the dataset created without overlapping samples, lets train a new model and look at the results.

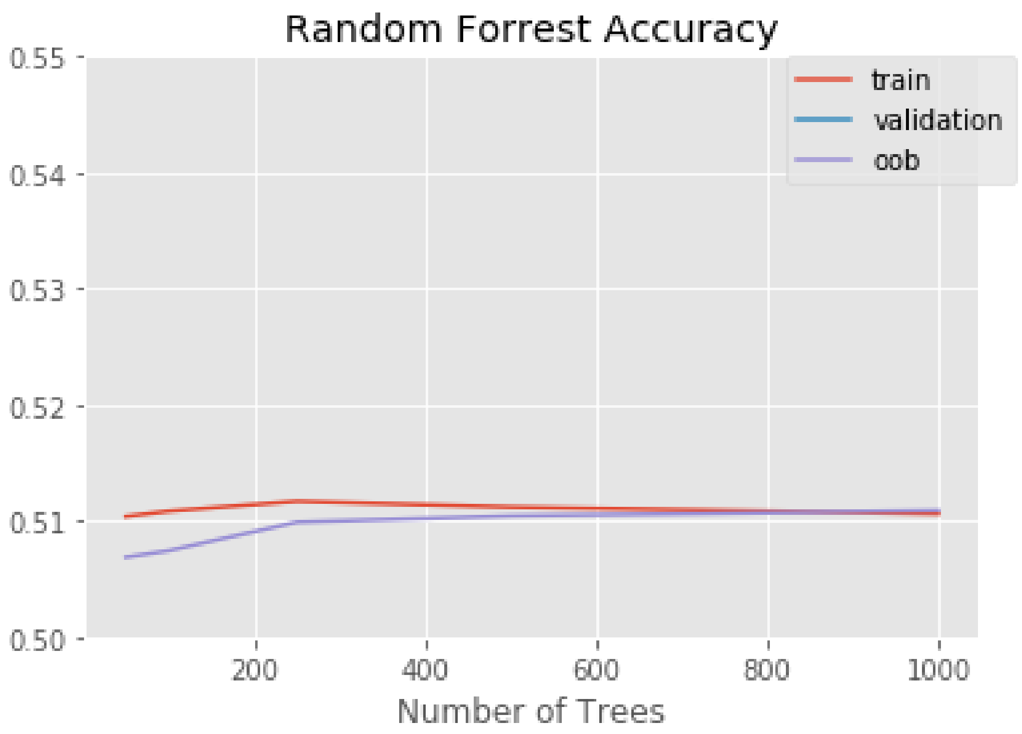
Train Model

```
In [32]: train_score = []
         valid_score = []
         oob_score = []

         for n_trees in tqdm(n_trees_l, desc='Training Models', unit='Model'):
             clf = RandomForestClassifier(n_trees, **clf_parameters)
             clf.fit(*non_overlapping_samples(X_train, y_train, 4))

             train_score.append(clf.score(X_train, y_train.values))
             valid_score.append(clf.score(X_valid, y_valid.values))
             oob_score.append(clf.oob_score_)
```

```
Training Models: 100%|| 5/5 [00:46<00:00,  9.38s/Model]
```

Results

```
In [33]: project_helper.plot(
             [n_trees_l]*3,
             [train_score, valid_score, oob_score],
             ['train', 'validation', 'oob'],
             'Random Forrest Accuracy',
             'Number of Trees')
```

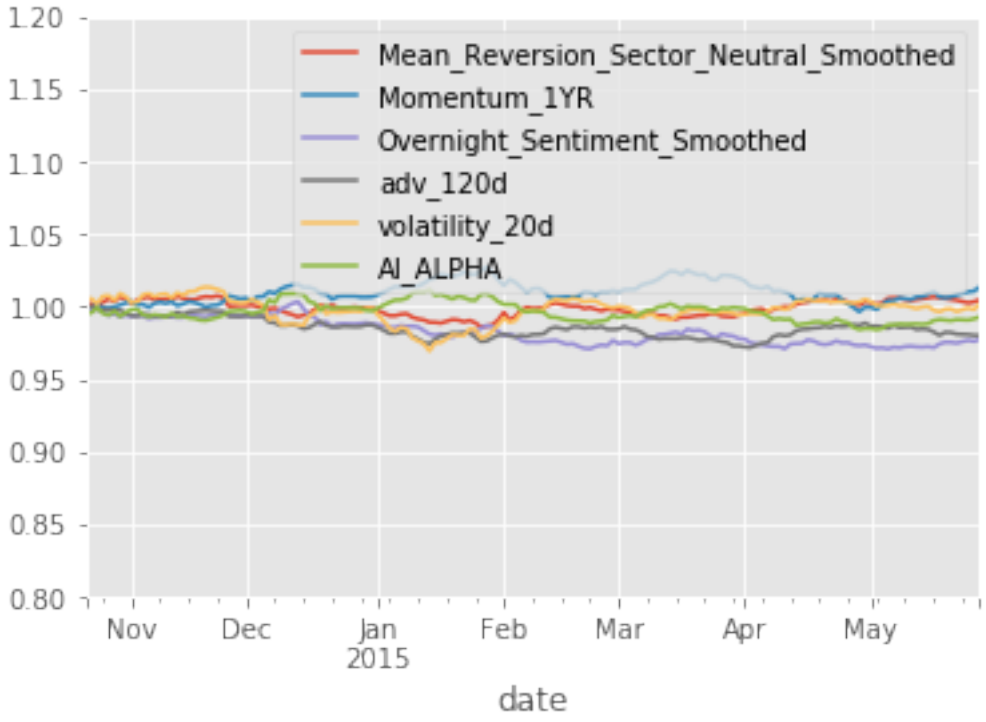


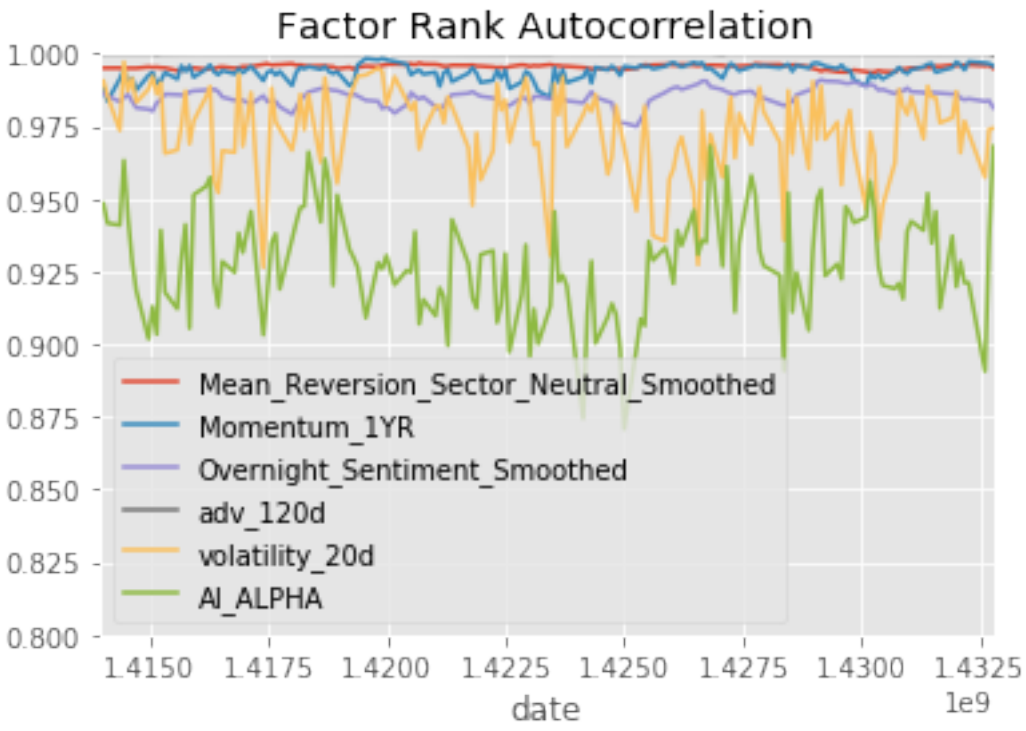

```
In [34]: show_sample_results(all_factors, X_valid, clf, factor_names)

Cleaning Data...

Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
```

```
max_loss is 35.0%, not exceeded: OK!

-----------------------
```

Sharpe Ratios	
Mean_Reversion_Sector_Neutral_Smoothed	0.32000000
Momentum_1YR	0.72000000
Overnight_Sentiment_Smoothed	-1.40000000
adv_120d	-1.44000000
volatility_20d	0.09000000
AI_ALPHA	-0.36000000

dtype: float64





This looks better, but we are throwing away a lot of information by taking every 5th row.

2.3.1 Use BaggingClassifier's `max_samples`

In this method, we'll set `max_samples` to be on the order of the average uniqueness of the labels. Since `RandomForrestClassifier` does not take this param, we're using `BaggingClassifier`. Implement `bagging_classifier` to build the bagging classifier.

```
In [35]: from sklearn.ensemble import BaggingClassifier
         from sklearn.tree import DecisionTreeClassifier


         def bagging_classifier(n_estimators, max_samples, max_features, parameters):
             """
             Build the bagging classifier.

             Parameters
             ----------
             n_estimators : int
                 The number of base estimators in the ensemble
             max_samples : float
                 The proportion of input samples drawn from when training each base estimator
             max_features : float
                 The proportion of input sample features drawn from when training each base esti
```

```
            parameters : dict
                Parameters to use in building the bagging classifier
                It should contain the following parameters:
                    criterion
                    min_samples_leaf
                    oob_score
                    n_jobs
                    random_state

            Returns
            -------
            bagging_clf : Scikit-Learn BaggingClassifier
                The bagging classifier
            """

            required_parameters = {'criterion', 'min_samples_leaf', 'oob_score', 'n_jobs', 'ran
            assert not required_parameters - set(parameters.keys())

            # TODO: Implement
            # base_estimator
            base_estimator = DecisionTreeClassifier(criterion=parameters['criterion'],
                                                    min_samples_leaf = parameters['min_samples_l

            # bagging classifier
            clf = BaggingClassifier(base_estimator = base_estimator,
                                    max_samples = max_samples,
                                    max_features = max_features,
                                    oob_score = parameters['oob_score'],
                                    n_jobs = parameters['n_jobs'],
                                    random_state = parameters['random_state'])

            return clf


        project_tests.test_bagging_classifier(bagging_classifier)
```

```
Tests Passed
```

With the bagging classifier built, lets train a new model and look at the results. #### Train Model

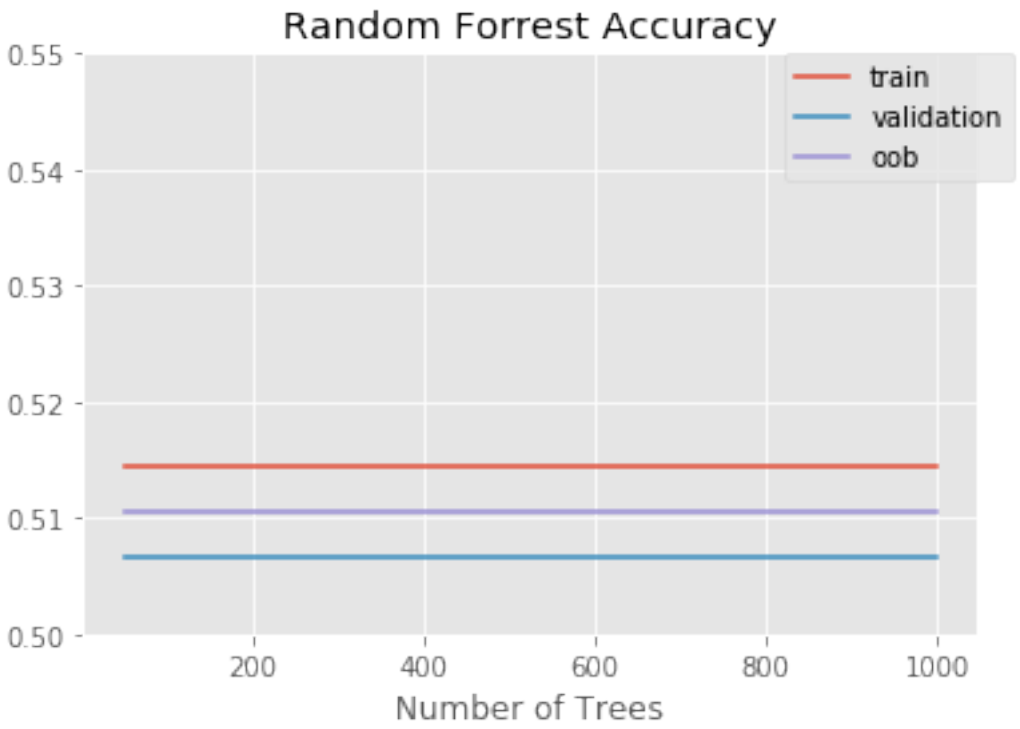
```
In [36]: train_score = []
         valid_score = []
         oob_score = []

         for n_trees in tqdm(n_trees_l, desc='Training Models', unit='Model'):
             clf = bagging_classifier(n_trees, 0.2, 1.0, clf_parameters)
```

```
            clf.fit(X_train, y_train)

            train_score.append(clf.score(X_train, y_train.values))
            valid_score.append(clf.score(X_valid, y_valid.values))
            oob_score.append(clf.oob_score_)
```

```
Training Models: 100%|| 5/5 [00:39<00:00,  7.92s/Model]
```

Results

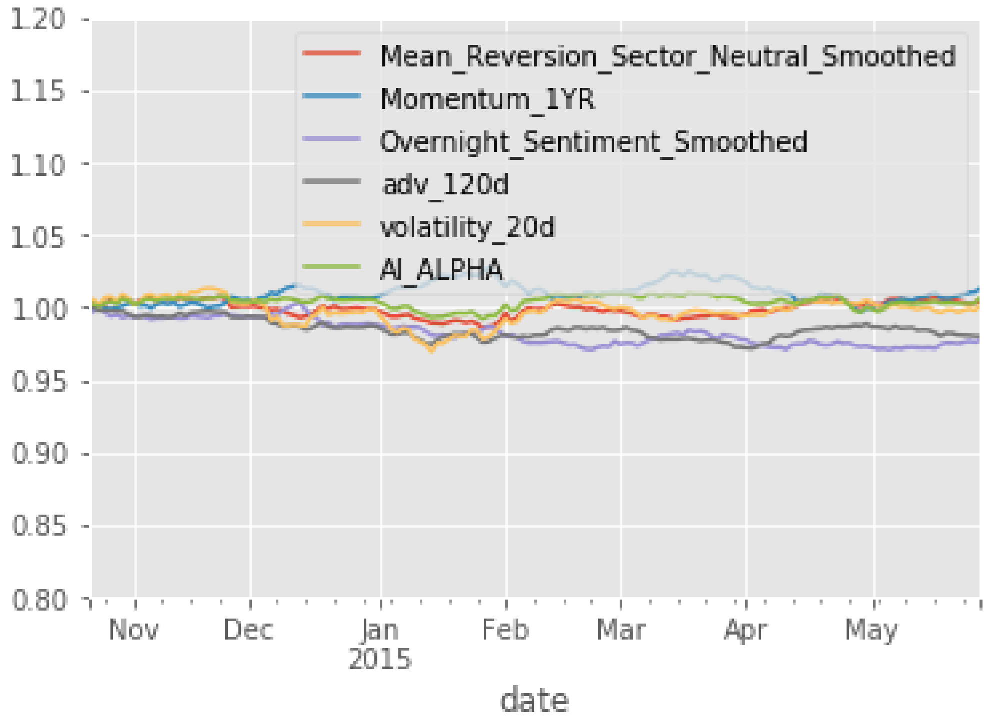
```
In [37]: project_helper.plot(
             [n_trees_l]*3,
             [train_score, valid_score, oob_score],
             ['train', 'validation', 'oob'],
             'Random Forrest Accuracy',
             'Number of Trees')
```

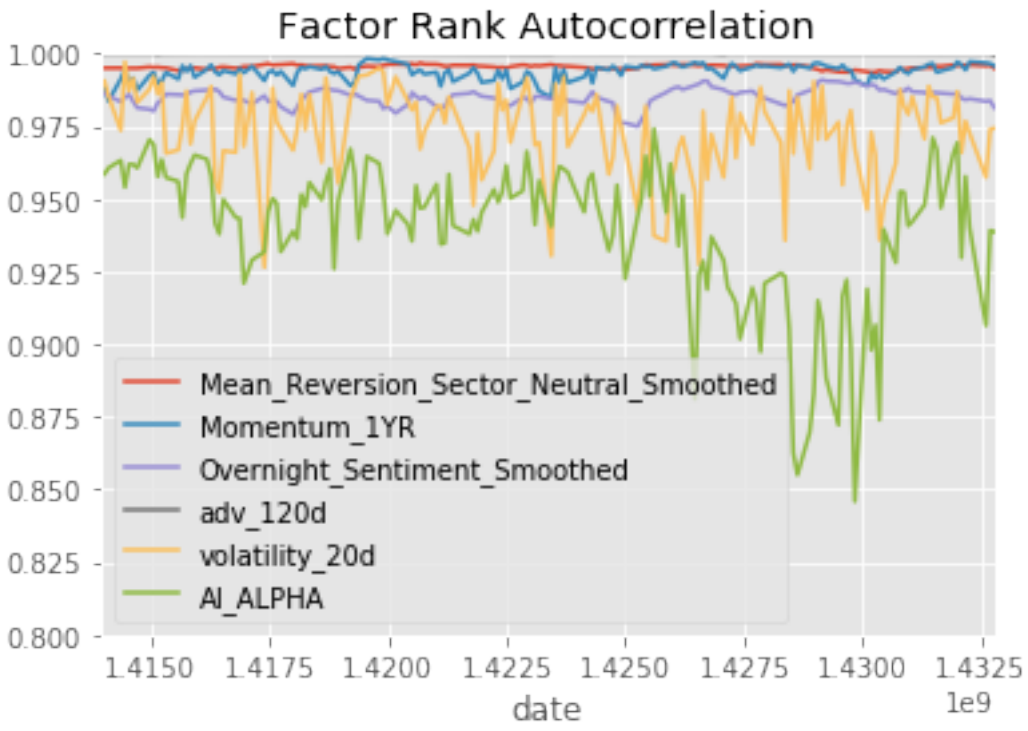



```
In [38]: show_sample_results(all_factors, X_valid, clf, factor_names)
```

```
Cleaning Data...

Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
```

```
Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!

-----------------------

             Sharpe Ratios
Mean_Reversion_Sector_Neutral_Smoothed    0.32000000
Momentum_1YR                              0.72000000
Overnight_Sentiment_Smoothed             -1.40000000
adv_120d                                 -1.44000000
volatility_20d                            0.09000000
AI_ALPHA                                  0.49000000
dtype: float64
```





This seems much "better" in the sense that we have much better fidelity between the three.

2.3.2 Build an ensemble of non-overlapping trees

The last method is to create ensemble of non-overlapping trees. Here we are going to write a custom `scikit-learn` estimator. We inherit from `VotingClassifier` and we override the `fit` method so we fit on non-overlapping periods.

In [39]:
```
import abc

from sklearn.ensemble import VotingClassifier
from sklearn.base import clone
from sklearn.preprocessing import LabelEncoder
from sklearn.utils import Bunch


class NoOverlapVoterAbstract(VotingClassifier):
    @abc.abstractmethod
    def _calculate_oob_score(self, classifiers):
        raise NotImplementedError

    @abc.abstractmethod
    def _non_overlapping_estimators(self, x, y, classifiers, n_skip_samples):
        raise NotImplementedError
```

```
            def __init__(self, estimator, voting='soft', n_skip_samples=4):
                # List of estimators for all the subsets of data
                estimators = [('clf'+str(i), estimator) for i in range(n_skip_samples + 1)]

                self.n_skip_samples = n_skip_samples
                super().__init__(estimators, voting)

            def fit(self, X, y, sample_weight=None):
                estimator_names, clfs = zip(*self.estimators)
                self.le_ = LabelEncoder().fit(y)
                self.classes_ = self.le_.classes_

                clone_clfs = [clone(clf) for clf in clfs]
                self.estimators_ = self._non_overlapping_estimators(X, y, clone_clfs, self.n_sk
                self.named_estimators_ = Bunch(**dict(zip(estimator_names, self.estimators_)))
                self.oob_score_ = self._calculate_oob_score(self.estimators_)

                return self
```

You might notice that two of the functions are abstracted. These will be the functions that you need to implement. #### OOB Score In order to get the correct OOB score, we need to take the average of all the estimator's OOB scores. Implement `calculate_oob_score` to calculate this score.

```
In [40]: def calculate_oob_score(classifiers):
             """
             Calculate the mean out-of-bag score from the classifiers.

             Parameters
             ----------
             classifiers : list of Scikit-Learn Classifiers
                 The classifiers used to calculate the mean out-of-bag score

             Returns
             -------
             oob_score : float
                 The mean out-of-bag score
             """

             # TODO: Implement
             oob_score = np.mean([clf.oob_score_ for clf in classifiers])
             return oob_score


         project_tests.test_calculate_oob_score(calculate_oob_score)
```

```
Tests Passed
```

Non Overlapping Estimators With `calculate_oob_score` implemented, let's create non overlapping estimators. Implement `non_overlapping_estimators` to build non overlapping subsets of the data, then run a estimator on each subset of data.

```
In [41]: def non_overlapping_estimators(x, y, classifiers, n_skip_samples):
             """
             Fit the classifiers to non overlapping data.

             Parameters
             ----------
             x : DataFrame
                 The input samples
             y : Pandas Series
                 The target values
             classifiers : list of Scikit-Learn Classifiers
                 The classifiers used to fit on the non overlapping data
             n_skip_samples : int
                 The number of samples to skip

             Returns
             -------
             fit_classifiers : list of Scikit-Learn Classifiers
                 The classifiers fit to the the non overlapping data
             """

             # TODO: Implement
             N = len(classifiers)
             clf_fit = [classifiers[i].fit(*non_overlapping_samples(x, y, n_skip_samples, start_
                                           ) for i in range(N)]

             return clf_fit


         project_tests.test_non_overlapping_estimators(non_overlapping_estimators)
```

```
Tests Passed
```

```
In [42]: class NoOverlapVoter(NoOverlapVoterAbstract):
             def _calculate_oob_score(self, classifiers):
                 return calculate_oob_score(classifiers)

             def _non_overlapping_estimators(self, x, y, classifiers, n_skip_samples):
                 return non_overlapping_estimators(x, y, classifiers, n_skip_samples)
```

Now that we have our `NoOverlapVoter` class, let's train it.

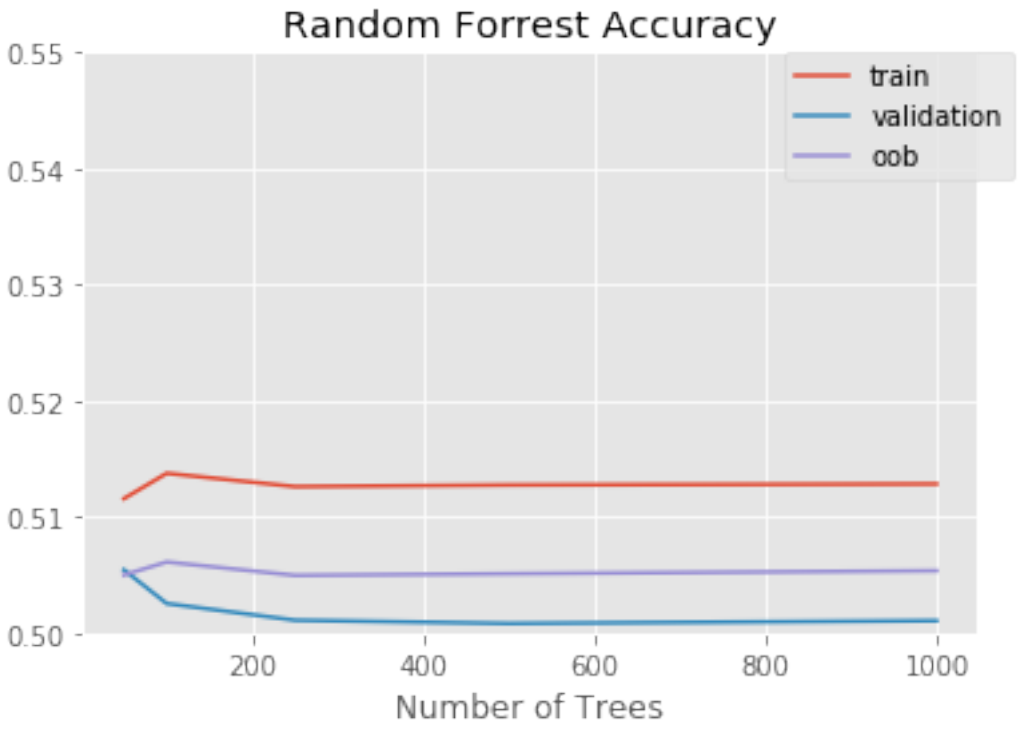
Train Model

```
In [43]: train_score = []
         valid_score = []
         oob_score = []

         for n_trees in tqdm(n_trees_l, desc='Training Models', unit='Model'):
             clf = RandomForestClassifier(n_trees, **clf_parameters)

             clf_nov = NoOverlapVoter(clf)
             clf_nov.fit(X_train, y_train)

             train_score.append(clf_nov.score(X_train, y_train.values))
             valid_score.append(clf_nov.score(X_valid, y_valid.values))
             oob_score.append(clf_nov.oob_score_)
```

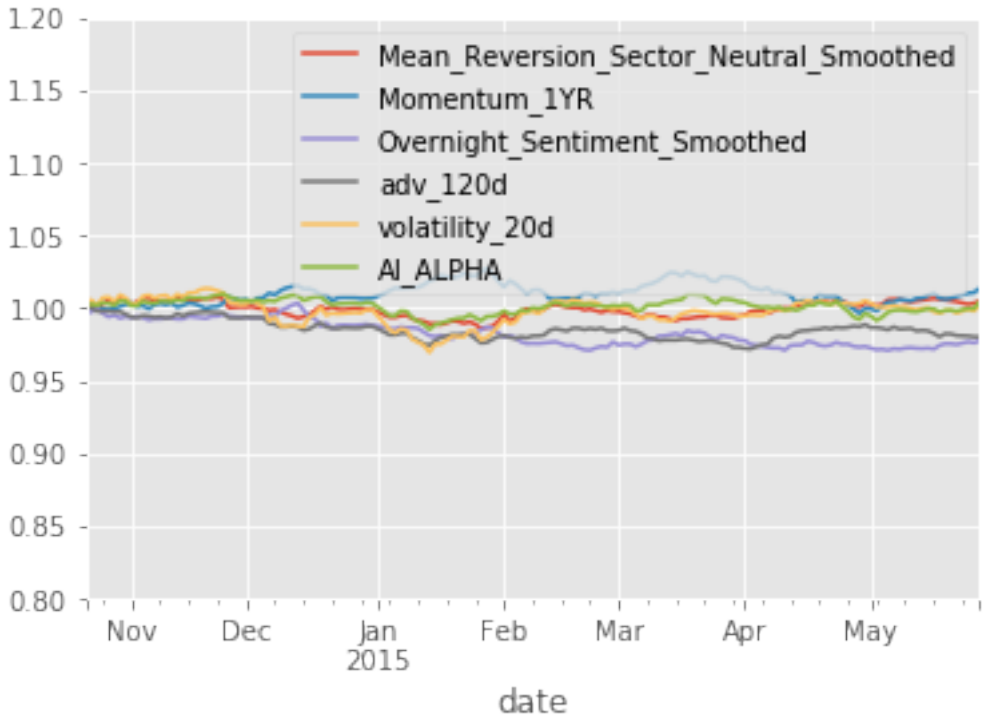
```
Training Models: 100%|| 5/5 [03:54<00:00, 46.98s/Model]
```

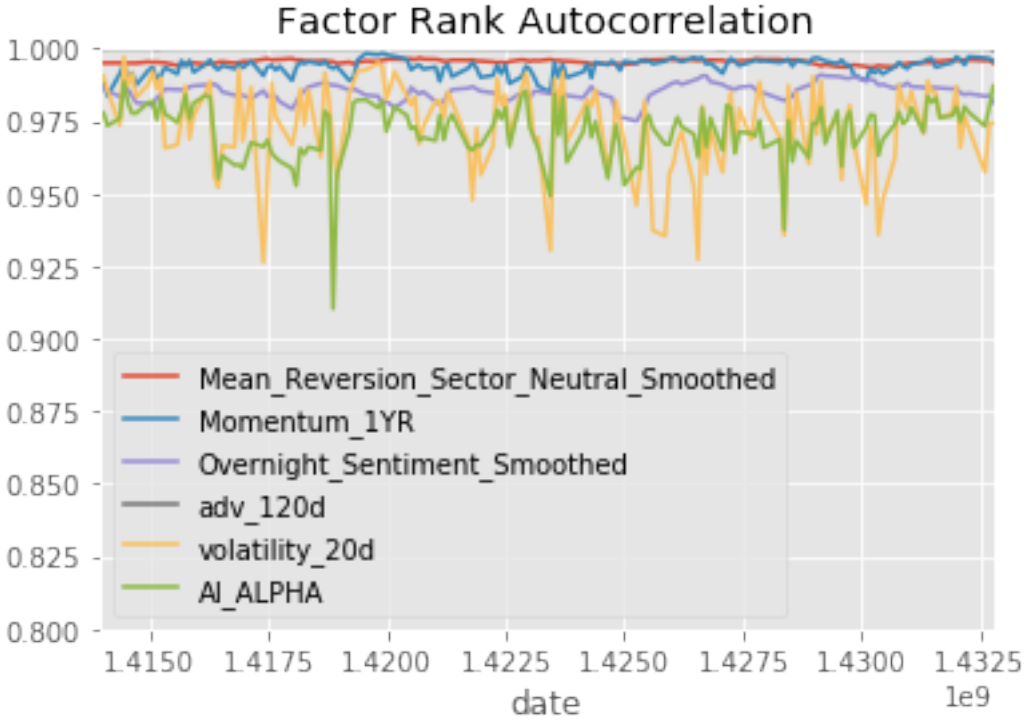
Results

```
In [44]: project_helper.plot(
             [n_trees_l]*3,
             [train_score, valid_score, oob_score],
             ['train', 'validation', 'oob'],
             'Random Forrest Accuracy',
             'Number of Trees')
```



```
In [45]: show_sample_results(all_factors, X_valid, clf_nov, factor_names)

Cleaning Data...

Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!

-----------------------

             Sharpe Ratios
Mean_Reversion_Sector_Neutral_Smoothed    0.32000000
Momentum_1YR                              0.72000000
Overnight_Sentiment_Smoothed             -1.40000000
adv_120d                                 -1.44000000
volatility_20d                            0.09000000
AI_ALPHA                                  0.27000000
dtype: float64
```


Mean_Reversion_Sector_Neutral_Smoothed
Momentum_1YR
Overnight_Sentiment_Smoothed
adv_120d
volatility_20d
AI_ALPHA
1.20
1.15
1.10
1.05
1.00
0.95
0.90
0.85
0.80
Nov
Dec
Jan
2015
Feb
Mar
Apr
May
date


Factor Rank Autocorrelation
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
Mean_Reversion_Sector_Neutral_Smoothed
Momentum_1YR
Overnight_Sentiment_Smoothed
adv_120d
volatility_20d
AI_ALPHA
1.4150
1.4175
1.4200
1.4225
1.4250
1.4275
1.4300
1.4325
1e9
date

2.4 Final Model

2.4.1 Re-Training Model

In production, we would roll forward the training. Typically you would re-train up to the "current day" and then test. Here, we will train on the train & validation dataset.

```
In [46]: n_trees = 500

         clf = RandomForestClassifier(n_trees, **clf_parameters)
         clf_nov = NoOverlapVoter(clf)
         clf_nov.fit(
             pd.concat([X_train, X_valid]),
             pd.concat([y_train, y_valid]))
```

```
Out[46]: NoOverlapVoter(estimator=None, n_skip_samples=4, voting='soft')
```

2.4.2 Results

Accuracy

```
In [47]: print('train: {}, oob: {}, valid: {}'.format(
             clf_nov.score(X_train, y_train.values),
             clf_nov.score(X_valid, y_valid.values),
             clf_nov.oob_score_))
```

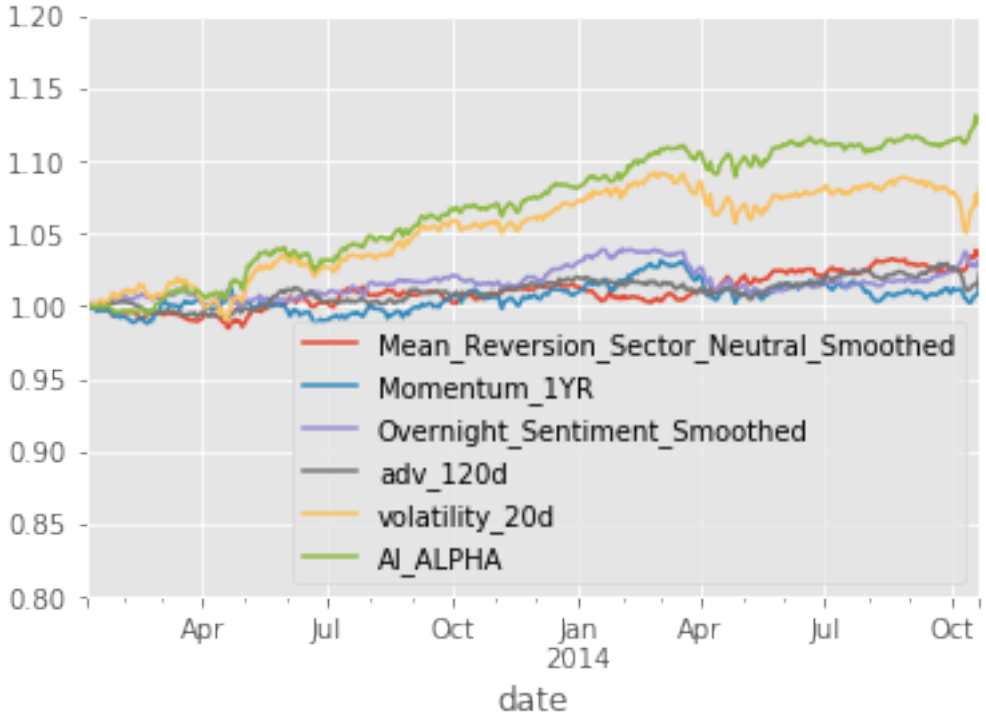
```
train: 0.5136380286849447, oob: 0.5130885626532165, valid: 0.5068740153740718
```

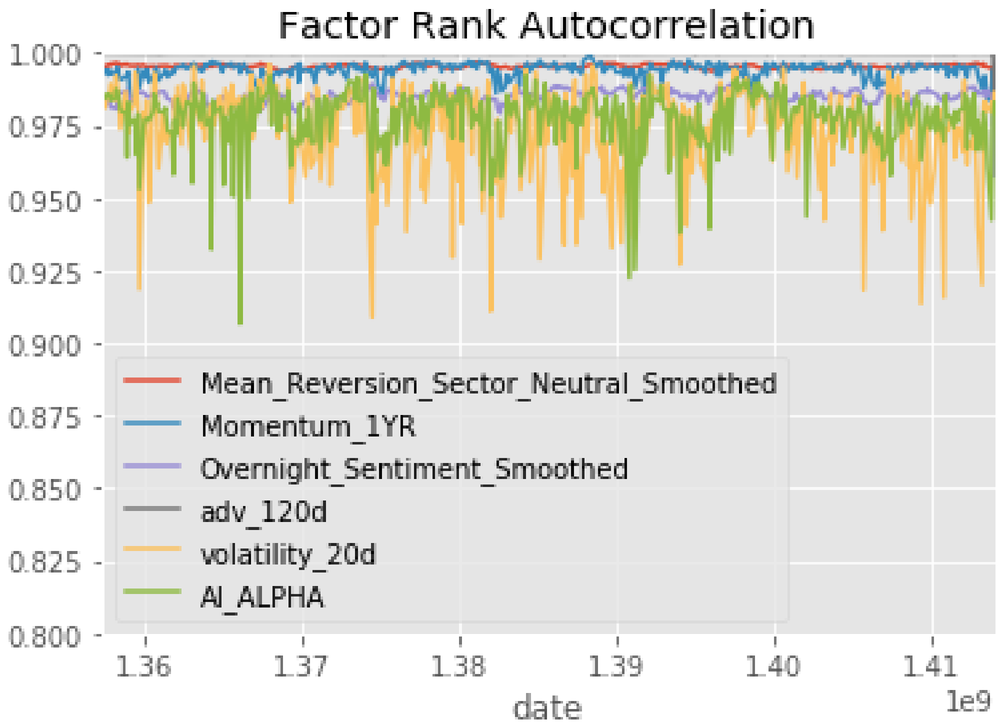
Train

```
In [48]: show_sample_results(all_factors, X_train, clf_nov, factor_names)
```

```
Cleaning Data...

Dropped 0.2% entries from factor data: 0.2% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.2% entries from factor data: 0.2% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.2% entries from factor data: 0.2% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.2% entries from factor data: 0.2% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.2% entries from factor data: 0.2% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.2% entries from factor data: 0.2% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!

-----------------------
```

```
                Sharpe Ratios
Mean_Reversion_Sector_Neutral_Smoothed    0.87000000
Momentum_1YR                              0.28000000
Overnight_Sentiment_Smoothed              0.83000000
adv_120d                                  0.62000000
volatility_20d                            1.18000000
AI_ALPHA                                  2.36000000
dtype: float64
```



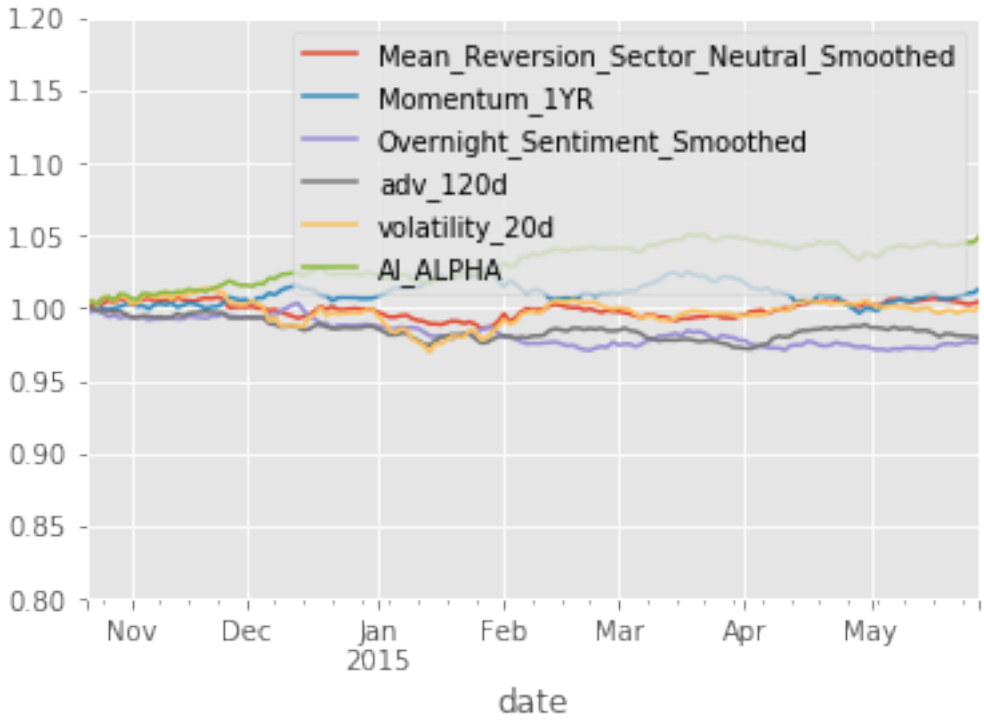


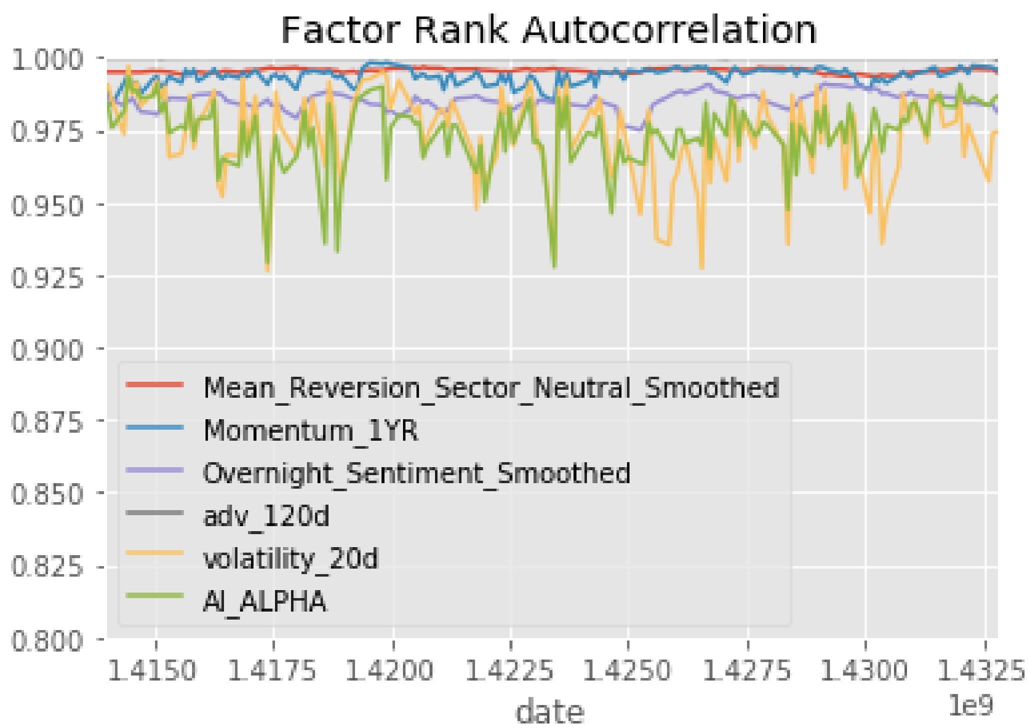
Validation

```
In [49]: show_sample_results(all_factors, X_valid, clf_nov, factor_names)
```

```
Cleaning Data...

Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!

-----------------------

             Sharpe Ratios
Mean_Reversion_Sector_Neutral_Smoothed    0.32000000
```

```
Momentum_1YR                          0.72000000
Overnight_Sentiment_Smoothed         -1.40000000
adv_120d                             -1.44000000
volatility_20d                        0.09000000
AI_ALPHA                              2.92000000
dtype: float64
```



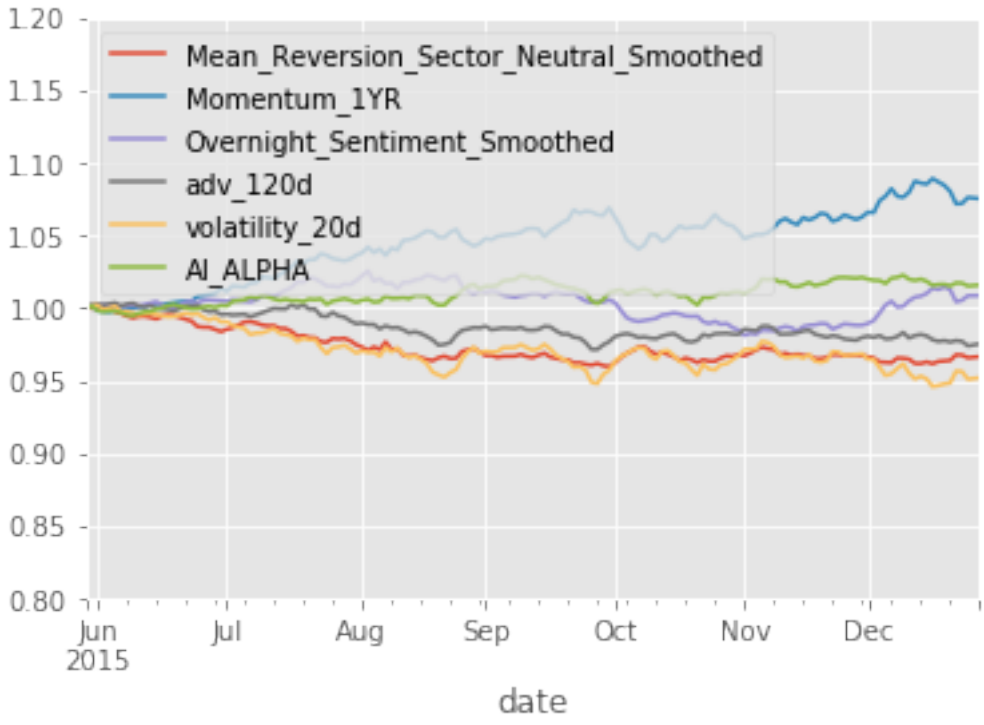


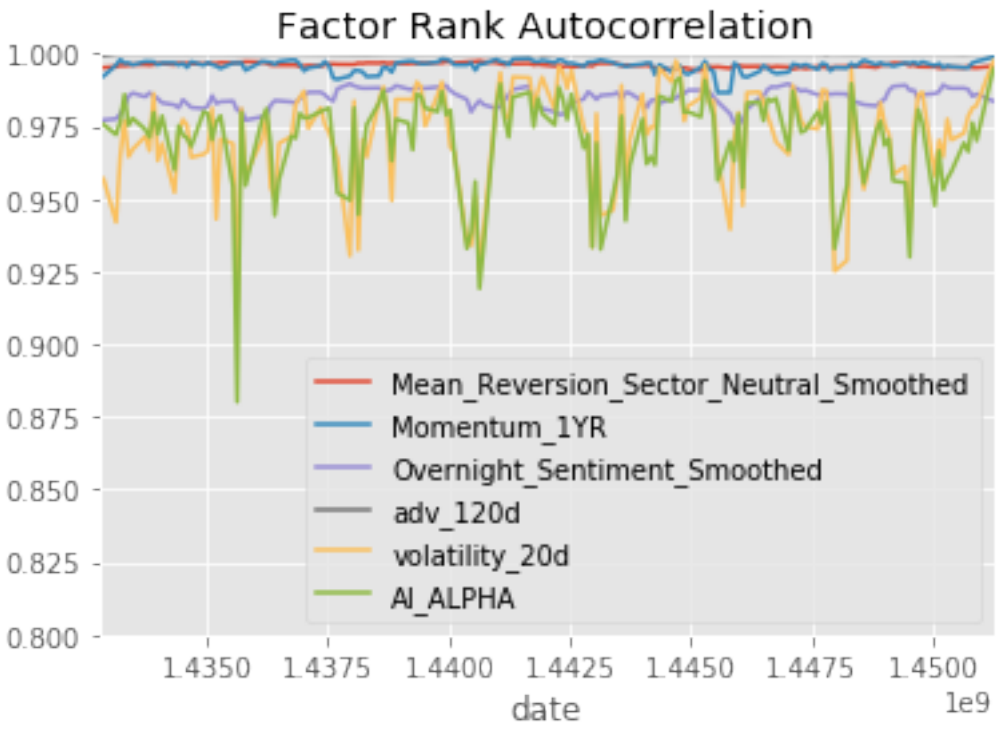
Test

```
In [50]: show_sample_results(all_factors, X_test, clf_nov, factor_names)
```

```
Cleaning Data...

Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computation and 0.0% in binning p
max_loss is 35.0%, not exceeded: OK!

-----------------------

                     Sharpe Ratios
Mean_Reversion_Sector_Neutral_Smoothed   -1.98000000
```

```
Momentum_1YR                          2.65000000
Overnight_Sentiment_Smoothed          0.43000000
adv_120d                             -1.44000000
volatility_20d                       -1.61000000
AI_ALPHA                              0.92000000
dtype: float64
```





So, hopefully you are appropriately amazed by this. Despite the significant differences between the factor performances in the three sets, the AI APLHA is able to deliver positive performance.

2.5 Submission

Now that you're done with the project, it's time to submit it. Click the submit button in the bottom right. One of our reviewers will give you feedback on your project with a pass or not passed grade. You can continue to the next section while you wait for feedback.