Project 7: Combine Signals for Enhanced Alpha

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.

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.

Install Packages

In [1]:
```
import sys
!{sys.executable} -m pip install -r requirements.txt
```

```
Collecting alphalens==0.3.2 (from -r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/a5/dc/2f9cd107d0d4c
f6223d37d81ddfbbdbf0d703d03669b83810fa6b97f32e5/alphalens-0.3.2.tar.gz (1
8.9MB)
    100% |████████████████████████████████| 18.9MB 1.9MB/s eta 0:00:01
3% |█                              | 655kB 5.2MB/s eta 0:00:04    18% |█
█████                          | 3.5MB 30.2MB/s eta 0:00:01    26% |█████
████                        | 5.0MB 31.1MB/s eta 0:00:01    67% |█████████
█████████████            | 12.7MB 30.0MB/s eta 0:00:01
Collecting graphviz==0.10.1 (from -r requirements.txt (line 2))
  Downloading https://files.pythonhosted.org/packages/1f/e2/ef2581b5b8662
5657afd32030f90cf2717456c1d2b711ba074bf007c0f1a/graphviz-0.10.1-py2.py3-n
one-any.whl
```

```
Collecting numpy==1.13.3 (from -r requirements.txt (line 3))
  Downloading https://files.pythonhosted.org/packages/57/a7/e3e6bd9d59512
5e1abbe162e323fd2d06f6f6683185294b79cd2cdb190d5/numpy-1.13.3-cp36-cp36m-m
anylinux1_x86_64.whl (17.0MB)
    100% |████████████████████████████████| 17.0MB 1.9MB/s eta 0:00:01
9% |███                            | 1.5MB 27.4MB/s eta 0:00:01    17% |
█████                           | 2.9MB 31.6MB/s eta 0:00:01    33% |████
███████                     | 5.7MB 29.5MB/s eta 0:00:01    41% |████████
█████                   | 7.1MB 26.8MB/s eta 0:00:01    66% |████████████
█████████           | 11.2MB 29.2MB/s eta 0:00:01    82% |███████████████
███████████     | 14.0MB 29.8MB/s eta 0:00:01    90% |██████████████████
███████████   | 15.4MB 29.0MB/s eta 0:00:01
Collecting pandas==0.18.1 (from -r requirements.txt (line 4))
  Downloading https://files.pythonhosted.org/packages/11/09/e66eb844daba8
680ddff26335d5b4fead77f60f957678243549a8dd4830d/pandas-0.18.1.tar.gz (7.3
MB)
    100% |████████████████████████████████| 7.3MB 7.4MB/s eta 0:00:01
Requirement already satisfied: python-dateutil==2.6.1 in /opt/conda/lib/p
ython3.6/site-packages (from -r requirements.txt (line 5)) (2.6.1)
Requirement already satisfied: pytz==2017.3 in /opt/conda/lib/python3.6/s
ite-packages (from -r requirements.txt (line 6)) (2017.3)
Collecting scipy==1.0.0 (from -r requirements.txt (line 7))
  Downloading https://files.pythonhosted.org/packages/d8/5e/caa01ba7be116
00b6a9d39265440d7b3be3d69206da887c42bef049521f2/scipy-1.0.0-cp36-cp36m-ma
nylinux1_x86_64.whl (50.0MB)
    100% |████████████████████████████████| 50.0MB 706kB/s ta 0:00:011
0% ||                               | 256kB 26.3MB/s eta 0:00:02    10% |
███                             | 5.1MB 25.6MB/s eta 0:00:02    19% |████
██                         | 9.8MB 22.5MB/s eta 0:00:02    24% |████████
| 12.1MB 22.4MB/s eta 0:00:02    33% |███████████                     | 1
6.9MB 22.8MB/s eta 0:00:02    36% |████████████                    | 18.0
MB 25.2MB/s eta 0:00:02    38% |████████████                    | 19.3MB
26.5MB/s eta 0:00:02    40% |█████████████                   | 20.2MB 22.
3MB/s eta 0:00:02    42% |█████████████▋                  | 21.3MB 24.2M
B/s eta 0:00:02    45% |██████████████▋                 | 22.6MB 24.5MB/s
eta 0:00:02    47% |███████████████                 | 23.6MB 23.2MB/s eta
0:00:02    49% |████████████████                | 24.8MB 25.4MB/s eta 0:0
0:01    52% |████████████████▊               | 26.0MB 20.2MB/s eta 0:00:0
2    54% |█████████████████▍              | 27.1MB 23.8MB/s eta 0:00:01
58% |███████████████████             | 29.4MB 23.1MB/s eta 0:00:01    60%
|███████████████████▌            | 30.5MB 24.9MB/s eta 0:00:01    64% |██
██████████████████▋           | 32.1MB 24.6MB/s eta 0:00:01    66% |█████
████████████████▏          | 33.3MB 22.0MB/s eta 0:00:01    68% |████████
██████████████           | 34.4MB 30.6MB/s eta 0:00:01    72% |███████████
████████████▍         | 36.4MB 21.7MB/s eta 0:00:01    75% |██████████████
██████████         | 37.6MB 31.5MB/s eta 0:00:01    79% |█████████████████
████████▌       | 39.8MB 28.1MB/s eta 0:00:01    85% |████████████████████
███████▋     | 42.9MB 18.3MB/s eta 0:00:01    88% |███████████████████████
█████▎    | 44.2MB 25.6MB/s eta 0:00:01    90% |████████████████████████
███    | 45.3MB 22.8MB/s eta 0:00:01    92% |████████████████████████████
▊  | 46.5MB 27.8MB/s eta 0:00:01    95% |██████████████████████████████
| 47.6MB 19.5MB/s eta 0:00:01    97% |███████████████████████████████ | 4
8.7MB 29.6MB/s eta 0:00:01    99% |████████████████████████████████| 49.9
MB 22.8MB/s eta 0:00:01
Requirement already satisfied: scikit-learn==0.19.1 in /opt/conda/lib/pyt
hon3.6/site-packages (from -r requirements.txt (line 8)) (0.19.1)
Requirement already satisfied: six==1.11.0 in /opt/conda/lib/python3.6/si
```

```
te-packages (from -r requirements.txt (line 9)) (1.11.0)
Collecting tables==3.3.0 (from -r requirements.txt (line 10))
  Downloading https://files.pythonhosted.org/packages/09/e7/72ca83c7bd75d
b94c23fcaf58debe1be5e9842376c630793e23765cab44b/tables-3.3.0-cp36-cp36m-m
anylinux1_x86_64.whl (4.6MB)
    100% |████████████████████████████████| 4.6MB 8.7MB/s eta 0:00:01
26% |████████▌                       | 1.2MB 21.4MB/s eta 0:00:01    53%
|█████████████████▏              | 2.5MB 29.8MB/s eta 0:00:01    85% |███
████████████████████████▌    | 4.0MB 29.8MB/s eta 0:00:01
Collecting tqdm==4.19.5 (from -r requirements.txt (line 11))
  Downloading https://files.pythonhosted.org/packages/71/3c/341b4fa23cb3a
bc335207dba057c790f3bb329f6757e1fcd5d347bcf8308/tqdm-4.19.5-py2.py3-none-
any.whl (51kB)
    100% |████████████████████████████████| 61kB 7.7MB/s ta 0:00:01
Collecting zipline===1.2.0 (from -r requirements.txt (line 12))
  Downloading https://files.pythonhosted.org/packages/15/d3/689f2a940478b
82ac57c751a40460598221fd82b0449a7a8f7eef47a3bcc/zipline-1.2.0.tar.gz (659
kB)
    100% |████████████████████████████████| 665kB 15.3MB/s ta 0:00:01
Requirement already satisfied: matplotlib>=1.4.0 in /opt/conda/lib/python
3.6/site-packages (from alphalens==0.3.2->-r requirements.txt (line 1)) (
2.1.0)
Requirement already satisfied: seaborn>=0.6.0 in /opt/conda/lib/python3.
6/site-packages (from alphalens==0.3.2->-r requirements.txt (line 1)) (0.
8.1)
Requirement already satisfied: statsmodels>=0.6.1 in /opt/conda/lib/pytho
n3.6/site-packages (from alphalens==0.3.2->-r requirements.txt (line 1))
(0.8.0)
Requirement already satisfied: IPython>=3.2.3 in /opt/conda/lib/python3.
6/site-packages (from alphalens==0.3.2->-r requirements.txt (line 1)) (6.
5.0)
Requirement already satisfied: numexpr>=2.5.2 in /opt/conda/lib/python3.
6/site-packages (from tables==3.3.0->-r requirements.txt (line 10)) (2.6.
4)
Requirement already satisfied: pip>=7.1.0 in /opt/conda/lib/python3.6/sit
e-packages (from zipline===1.2.0->-r requirements.txt (line 12)) (18.1)
Requirement already satisfied: setuptools>18.0 in /opt/conda/lib/python3.
6/site-packages (from zipline===1.2.0->-r requirements.txt (line 12)) (3
8.4.0)
Collecting Logbook>=0.12.5 (from zipline===1.2.0->-r requirements.txt (li
ne 12))
  Downloading https://files.pythonhosted.org/packages/f6/83/20fc027061491
9cb799f76e32cf143a54c58ce2fa45c19fd38ac2e4f9977/Logbook-1.4.3.tar.gz (85k
B)
    100% |████████████████████████████████| 92kB 8.9MB/s eta 0:00:01
Collecting requests-file>=1.4.1 (from zipline===1.2.0->-r requirements.tx
t (line 12))
  Downloading https://files.pythonhosted.org/packages/23/9c/6e63c23c39e53
d3df41c77a3d05a49a42c4e1383a6d2a5e3233161b89dbf/requests_file-1.4.3-py2.p
y3-none-any.whl
Collecting pandas-datareader<0.6,>=0.2.1 (from zipline===1.2.0->-r requir
ements.txt (line 12))
  Downloading https://files.pythonhosted.org/packages/40/c5/cc720f531bbde
0efeab940de400d0fcc95e87770a3abcd7f90d6d52a3302/pandas_datareader-0.5.0-p
y2.py3-none-any.whl (74kB)
    100% |████████████████████████████████| 81kB 6.5MB/s ta 0:00:01    4
1% |█████████████▎                  | 30kB 9.6MB/s eta 0:00:01
```

```
Requirement already satisfied: patsy>=0.4.0 in /opt/conda/lib/python3.6/s
ite-packages (from zipline===1.2.0->-r requirements.txt (line 12)) (0.4.
1)
Requirement already satisfied: requests>=2.9.1 in /opt/conda/lib/python3.
6/site-packages (from zipline===1.2.0->-r requirements.txt (line 12)) (2.
18.4)
Requirement already satisfied: Cython>=0.25.2 in /opt/conda/lib/python3.
6/site-packages (from zipline===1.2.0->-r requirements.txt (line 12)) (0.
29.7)
Collecting cyordereddict>=0.2.2 (from zipline===1.2.0->-r requirements.tx
t (line 12))
  Downloading https://files.pythonhosted.org/packages/d1/1a/364cbfd927be1
b743c7f0a985a7f1f7e8a51469619f9fefe4ee9240ba210/cyordereddict-1.0.0.tar.g
z (138kB)
    100% |████████████████████████████████| 143kB 15.8MB/s ta 0:00:01
Collecting bottleneck>=1.0.0 (from zipline===1.2.0->-r requirements.txt (
line 12))
  Downloading https://files.pythonhosted.org/packages/05/ae/cedf5323f398a
b4e4ff92d6c431a3e1c6a186f9b41ab3e8258dff786a290/Bottleneck-1.2.1.tar.gz (
105kB)
    100% |████████████████████████████████| 112kB 16.3MB/s ta 0:00:01
Collecting contextlib2>=0.4.0 (from zipline===1.2.0->-r requirements.txt
(line 12))
  Downloading https://files.pythonhosted.org/packages/a2/71/8273a7eeed0af
f6a854237ab5453bc9aa67deb49df4832801c21f0ff3782/contextlib2-0.5.5-py2.py3
-none-any.whl
Requirement already satisfied: decorator>=4.0.0 in /opt/conda/lib/python
3.6/site-packages (from zipline===1.2.0->-r requirements.txt (line 12)) (
4.0.11)
Requirement already satisfied: networkx<2.0,>=1.9.1 in /opt/conda/lib/pyt
hon3.6/site-packages (from zipline===1.2.0->-r requirements.txt (line 1
2)) (1.11)
Collecting bcolz<1,>=0.12.1 (from zipline===1.2.0->-r requirements.txt (l
ine 12))
  Downloading https://files.pythonhosted.org/packages/6c/8b/1ffa01f872cac
36173c5eb95b58c01040d8d25f1b242c48577f4104cd3ab/bcolz-0.12.1.tar.gz (622k
B)
    100% |████████████████████████████████| 624kB 14.3MB/s ta 0:00:01
Requirement already satisfied: click>=4.0.0 in /opt/conda/lib/python3.6/s
ite-packages (from zipline===1.2.0->-r requirements.txt (line 12)) (6.7)
Requirement already satisfied: toolz>=0.8.2 in /opt/conda/lib/python3.6/s
ite-packages (from zipline===1.2.0->-r requirements.txt (line 12)) (0.8.
2)
Collecting multipledispatch>=0.4.8 (from zipline===1.2.0->-r requirement
s.txt (line 12))
  Downloading https://files.pythonhosted.org/packages/89/79/429ecef45fd5e
4504f7474d4c3c3c4668c267be3370e4c2fd33e61506833/multipledispatch-0.6.0-py
3-none-any.whl
Requirement already satisfied: MarkupSafe>=0.23 in /opt/conda/lib/python
3.6/site-packages (from zipline===1.2.0->-r requirements.txt (line 12)) (
1.0)
Requirement already satisfied: Mako>=1.0.1 in /opt/conda/lib/python3.6/si
te-packages/Mako-1.0.7-py3.6.egg (from zipline===1.2.0->-r requirements.t
xt (line 12)) (1.0.7)
Requirement already satisfied: sqlalchemy>=1.0.8 in /opt/conda/lib/python
3.6/site-packages (from zipline===1.2.0->-r requirements.txt (line 12)) (
1.1.13)
```

```
Collecting alembic>=0.7.7 (from zipline===1.2.0->-r requirements.txt (lin
e 12))
  Downloading https://files.pythonhosted.org/packages/7b/8b/0c98c378d9316
5d9809193f274c3c6e2151120d955b752419c7d43e4d857/alembic-1.0.11.tar.gz (1.
0MB)
    100% |████████████████████████████████| 1.0MB 14.6MB/s ta 0:00:01
Collecting sortedcontainers>=1.4.4 (from zipline===1.2.0->-r requirement
s.txt (line 12))
  Downloading https://files.pythonhosted.org/packages/13/f3/cf85f7c3a2dbd
1a515d51e1f1676d971abe41bba6f4ab5443240d9a78e5b/sortedcontainers-2.1.0-py
2.py3-none-any.whl
Collecting intervaltree>=2.1.0 (from zipline===1.2.0->-r requirements.txt
(line 12))
  Downloading https://files.pythonhosted.org/packages/e8/f9/76237755b2020
cd74549e98667210b2dd54d3fb17c6f4a62631e61d31225/intervaltree-3.0.2.tar.gz
Collecting lru-dict>=1.1.4 (from zipline===1.2.0->-r requirements.txt (li
ne 12))
  Downloading https://files.pythonhosted.org/packages/00/a5/32ed6e10246cd
341ca8cc205acea5d208e4053f48a4dced2b1b31d45ba3f/lru-dict-1.1.6.tar.gz
Collecting empyrical>=0.4.2 (from zipline===1.2.0->-r requirements.txt (l
ine 12))
  Downloading https://files.pythonhosted.org/packages/7b/55/a01b05162b764
830dbbac868462f44cd847a5b6523a01ca9f955721819da/empyrical-0.5.0.tar.gz (4
9kB)
    100% |████████████████████████████████| 51kB 6.7MB/s ta 0:00:01
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.6/s
ite-packages/cycler-0.10.0-py3.6.egg (from matplotlib>=1.4.0->alphalens==
0.3.2->-r requirements.txt (line 1)) (0.10.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 i
n /opt/conda/lib/python3.6/site-packages (from matplotlib>=1.4.0->alphale
ns==0.3.2->-r requirements.txt (line 1)) (2.2.0)
Requirement already satisfied: pexpect; sys_platform != "win32" in /opt/c
onda/lib/python3.6/site-packages (from IPython>=3.2.3->alphalens==0.3.2->
-r requirements.txt (line 1)) (4.3.1)
Requirement already satisfied: pygments in /opt/conda/lib/python3.6/site-
packages (from IPython>=3.2.3->alphalens==0.3.2->-r requirements.txt (lin
e 1)) (2.2.0)
Requirement already satisfied: backcall in /opt/conda/lib/python3.6/site-
packages (from IPython>=3.2.3->alphalens==0.3.2->-r requirements.txt (lin
e 1)) (0.1.0)
Requirement already satisfied: jedi>=0.10 in /opt/conda/lib/python3.6/sit
e-packages (from IPython>=3.2.3->alphalens==0.3.2->-r requirements.txt (l
ine 1)) (0.10.2)
Requirement already satisfied: simplegeneric>0.8 in /opt/conda/lib/python
3.6/site-packages (from IPython>=3.2.3->alphalens==0.3.2->-r requirement
s.txt (line 1)) (0.8.1)
Requirement already satisfied: pickleshare in /opt/conda/lib/python3.6/si
te-packages (from IPython>=3.2.3->alphalens==0.3.2->-r requirements.txt (
line 1)) (0.7.4)
Requirement already satisfied: traitlets>=4.2 in /opt/conda/lib/python3.
6/site-packages (from IPython>=3.2.3->alphalens==0.3.2->-r requirements.t
xt (line 1)) (4.3.2)
Requirement already satisfied: prompt-toolkit<2.0.0,>=1.0.15 in /opt/cond
a/lib/python3.6/site-packages (from IPython>=3.2.3->alphalens==0.3.2->-r
requirements.txt (line 1)) (1.0.15)
Collecting requests-ftp (from pandas-datareader<0.6,>=0.2.1->zipline===1.
2.0->-r requirements.txt (line 12))
```

```
  Downloading https://files.pythonhosted.org/packages/3d/ca/14b2ad1e93b51
95eeaf56b86b7ecfd5ea2d5754a68d17aeb1e5b9f95b3cf/requests-ftp-0.3.1.tar.gz
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /opt/conda/lib/py
thon3.6/site-packages (from requests>=2.9.1->zipline===1.2.0->-r requirem
ents.txt (line 12)) (3.0.4)
Requirement already satisfied: idna<2.7,>=2.5 in /opt/conda/lib/python3.
6/site-packages (from requests>=2.9.1->zipline===1.2.0->-r requirements.t
xt (line 12)) (2.6)
Requirement already satisfied: urllib3<1.23,>=1.21.1 in /opt/conda/lib/py
thon3.6/site-packages (from requests>=2.9.1->zipline===1.2.0->-r requirem
ents.txt (line 12)) (1.22)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/pytho
n3.6/site-packages (from requests>=2.9.1->zipline===1.2.0->-r requirement
s.txt (line 12)) (2017.11.5)
Collecting python-editor>=0.3 (from alembic>=0.7.7->zipline===1.2.0->-r r
equirements.txt (line 12))
  Downloading https://files.pythonhosted.org/packages/c6/d3/201fc3abe391b
bae6606e6f1d598c15d367033332bd54352b12f35513717/python_editor-1.0.4-py3-n
one-any.whl
Requirement already satisfied: ptyprocess>=0.5 in /opt/conda/lib/python3.
6/site-packages (from pexpect; sys_platform != "win32"->IPython>=3.2.3->a
lphalens==0.3.2->-r requirements.txt (line 1)) (0.5.2)
Requirement already satisfied: ipython-genutils in /opt/conda/lib/python
3.6/site-packages (from traitlets>=4.2->IPython>=3.2.3->alphalens==0.3.2-
>-r requirements.txt (line 1)) (0.2.0)
Requirement already satisfied: wcwidth in /opt/conda/lib/python3.6/site-p
ackages (from prompt-toolkit<2.0.0,>=1.0.15->IPython>=3.2.3->alphalens==
0.3.2->-r requirements.txt (line 1)) (0.1.7)
Building wheels for collected packages: alphalens, pandas, zipline, Logbo
ok, cyordereddict, bottleneck, bcolz, alembic, intervaltree, lru-dict, em
pyrical, requests-ftp
  Running setup.py bdist_wheel for alphalens ... done
  Stored in directory: /root/.cache/pip/wheels/77/1e/9a/223b4c94d7f564f25
d94b48ca5b9c53e3034016ece3fd8c8c1
  Running setup.py bdist_wheel for pandas ... done
  Stored in directory: /root/.cache/pip/wheels/a3/08/c3/8fdd52954d4b41562
4cff43c6dd32a22bac90306976a98f4af
  Running setup.py bdist_wheel for zipline ... done
  Stored in directory: /root/.cache/pip/wheels/5d/20/7d/b48368c8634b1cb6c
c7232833b2780a265d4217c0ad2e3d24c
  Running setup.py bdist_wheel for Logbook ... done
  Stored in directory: /root/.cache/pip/wheels/a2/9f/6f/8c7a4ed6b9f6f3c98
b742dbb0fd41fff3c130119c507376301
  Running setup.py bdist_wheel for cyordereddict ... done
  Stored in directory: /root/.cache/pip/wheels/0b/9d/8b/5bf3e22c1edd59b50
f11bb19dec9dfcfe5a479fc7ace02b61f
  Running setup.py bdist_wheel for bottleneck ... done
  Stored in directory: /root/.cache/pip/wheels/f2/bf/ec/e0f39aa27001525ad
455139ee57ec7d0776fe074dfd78c97e4
  Running setup.py bdist_wheel for bcolz ... done
  Stored in directory: /root/.cache/pip/wheels/c5/cc/1b/2cf1f88959af5d7f4
d449b7fc6c9452d0ecbd86fd61a9ee376
  Running setup.py bdist_wheel for alembic ... done
  Stored in directory: /root/.cache/pip/wheels/8b/65/b2/9837b4422d13e739c
3324c428f1b3aa9e3c3df666bb420e4b3
  Running setup.py bdist_wheel for intervaltree ... done
  Stored in directory: /root/.cache/pip/wheels/08/99/c0/5a5942f5b9567c59c
```

```
14aac76f95a70bf11dccc71240b91ebf5
  Running setup.py bdist_wheel for lru-dict ... done
  Stored in directory: /root/.cache/pip/wheels/b7/ef/06/fbdd555907a7d438f
b33e4c8675f771ff1cf41917284c51ebf
  Running setup.py bdist_wheel for empyrical ... done
  Stored in directory: /root/.cache/pip/wheels/83/14/73/34fb27552601518d2
8bd0813d75124be76d94ab29152c69112
  Running setup.py bdist_wheel for requests-ftp ... done
  Stored in directory: /root/.cache/pip/wheels/2a/98/32/37195e45a3392a73d
9f65c488cbea30fe5bad76aaef4d6b020
Successfully built alphalens pandas zipline Logbook cyordereddict bottlen
eck bcolz alembic intervaltree lru-dict empyrical requests-ftp
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: numpy, pandas, scipy, alphalens, graphviz,
tables, tqdm, Logbook, requests-file, requests-ftp, pandas-datareader, cy
ordereddict, bottleneck, contextlib2, bcolz, multipledispatch, python-edi
tor, alembic, sortedcontainers, intervaltree, lru-dict, empyrical, ziplin
e
  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 0.19.1
    Uninstalling scipy-0.19.1:
      Successfully uninstalled scipy-0.19.1
  Found existing installation: tqdm 4.11.2
    Uninstalling tqdm-4.11.2:
      Successfully uninstalled tqdm-4.11.2
Successfully installed Logbook-1.4.3 alembic-1.0.11 alphalens-0.3.2 bcolz
-0.12.1 bottleneck-1.2.1 contextlib2-0.5.5 cyordereddict-1.0.0 empyrical-
0.5.0 graphviz-0.10.1 intervaltree-3.0.2 lru-dict-1.1.6 multipledispatch-
0.6.0 numpy-1.13.3 pandas-0.18.1 pandas-datareader-0.5.0 python-editor-1.
0.4 requests-file-1.4.3 requests-ftp-0.3.1 scipy-1.0.0 sortedcontainers-
2.1.0 tables-3.3.0 tqdm-4.19.5 zipline-1.2.0
```

Load Packages

In [2]:

```
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)
```

Data Pipeline

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 [3]:
```
import os
from zipline.data import bundles

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

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

print('Data Registered')
```

```
Data Registered
```

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 [4]:
```
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_calend
```

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 [5]:
```
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[5]:
```
[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])]
```

Get Returns

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

In [6]:
```python
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_d
    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 [7]:
```
def get_pricing(data_portal, trading_calendar, assets, start_date, end_da
    end_dt = pd.Timestamp(end_date.strftime('%Y-%m-%d'), tz='UTC', offset
    start_dt = pd.Timestamp(start_date.strftime('%Y-%m-%d'), tz='UTC', of

    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')
```

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 [8]:
```
from zipline.pipeline.factors import CustomFactor, DailyReturns, Returns,
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,
    unsmoothed_factor = -Returns(window_length=window_length, mask=univer
        .demean(groupby=sector) \
        .rank() \
        .zscore()
    return SimpleMovingAverage(inputs=[unsmoothed_factor], 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 r
        As such, opens[-1] is the most recent open, and closes[0] is the
        """
        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_retur
    cto_out = CTO(mask=universe, window_length=cto_window_length)
    unsmoothed_factor = TrailingOvernightReturns(inputs=[cto_out], window
        .rank() \
        .zscore()
    return SimpleMovingAverage(inputs=[unsmoothed_factor], window_length=
        .rank() \
        .zscore()
```

Combine the Factors to a single Pipeline

Let's add the factors to a pipeline.

In [9]:
```
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')
```

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 [10]:
```
pipeline.add(AnnualizedVolatility(window_length=20, mask=universe).rank()
pipeline.add(AnnualizedVolatility(window_length=120, mask=universe).rank(
pipeline.add(AverageDollarVolume(window_length=20, mask=universe).rank().
pipeline.add(AverageDollarVolume(window_length=120, mask=universe).rank()
pipeline.add(sector, 'sector_code')
```

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 [11]:
```
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)]
pipeline.add(SimpleMovingAverage(inputs=[MarketDispersion(mask=universe)]
```

In [12]:
```
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_ret


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

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 [13]:
```
pipeline.add(Returns(window_length=5, mask=universe).quantiles(2), 'retur
pipeline.add(Returns(window_length=5, mask=universe).quantiles(25), 'retu
```

Date Features

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

In [14]:
```
all_factors = engine.run_pipeline(pipeline, factor_start_date, universe_e

all_factors['is_Janaury'] = all_factors.index.get_level_values(0).month =
all_factors['is_December'] = all_factors.index.get_level_values(0).month
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_fac
all_factors['month_end'] = all_factors.index.get_level_values(0).isin(pd.
all_factors['month_start'] = all_factors.index.get_level_values(0).isin(p
all_factors['qtr_end'] = all_factors.index.get_level_values(0).isin(pd.da
all_factors['qtr_start'] = all_factors.index.get_level_values(0).isin(pd.

all_factors.head()
```

Out[14]:

		Mean_Reversion_Sector_Neutral_Smoothed	Momentum_1YR
2013-01-03 00:00:00+00:00	Equity(0 [A])	-0.26276899	-1.20797813
	Equity(1 [AAL])	0.09992624	1.71347052
	Equity(2 [AAP])	1.66913824	-1.53506144
	Equity(3 [AAPL])	1.69874602	1.19311071
	Equity(4 [ABBV])	nan	nan

5 rows × 23 columns

One Hot Encode Sectors

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

In [15]:
```
sector_lookup = pd.read_csv(
    os.path.join(os.getcwd(), '..', '..', 'data', 'project_7_sector', 'la
    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[15]:

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

Shift Target

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

In [16]:
```
all_factors['target'] = all_factors.groupby(level=1)['return_5d'].shift(-

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

Out[16]:

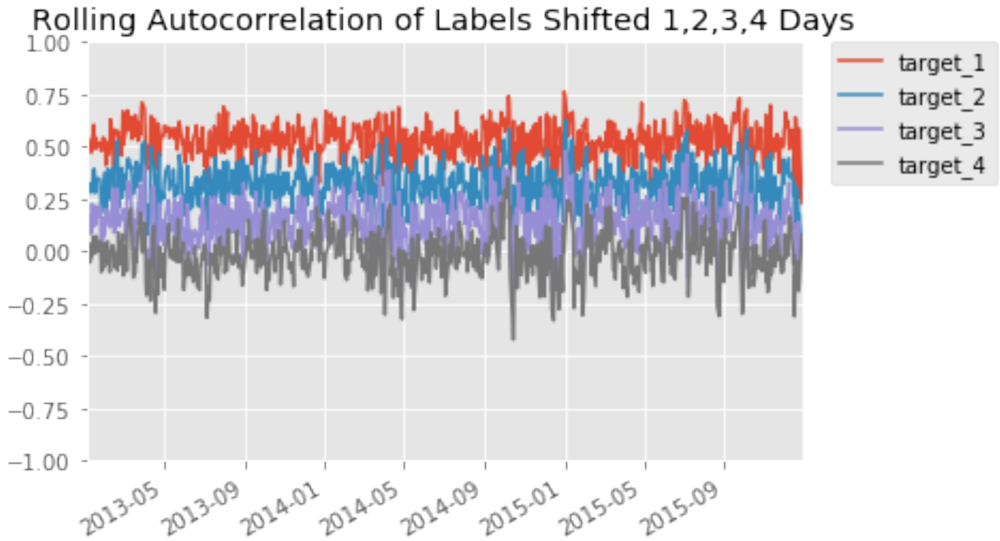
	level_0	level_1	return_5d	target
0	2013-01-03 00:00:00+00:00	Equity(0 [A])	0	0.00000000
471	2013-01-04 00:00:00+00:00	Equity(0 [A])	0	0.00000000
942	2013-01-07 00:00:00+00:00	Equity(0 [A])	0	0.00000000
1413	2013-01-08 00:00:00+00:00	Equity(0 [A])	0	1.00000000
1884	2013-01-09 00:00:00+00:00	Equity(0 [A])	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

IID Check of Target

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

In [17]:

```python
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'].shi
all_factors['target_1'] = all_factors.groupby(level=1)['return_5d'].shift
all_factors['target_2'] = all_factors.groupby(level=1)['return_5d'].shift
all_factors['target_3'] = all_factors.groupby(level=1)['return_5d'].shift
all_factors['target_4'] = all_factors.groupby(level=1)['return_5d'].shift

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()
```



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

#TODO: Put Answer In this Cell

Thanks to the discussion in the student hub https://hub.udacity.com/rooms/community:nd880:346730-project-563/community:thread-6273961378-567340?contextType=room and the reference to Term 1, moduule 4, Lesson 27, video #39 on the problem of autocorrelation which helped me to remind me what to understand under this term.

From the plot we can see that the returns are independent, on the one hand, and that they are identically distributed, on the other hand. Returns of each day are highly correlated to returns of the next day.

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 [19]:
```
def train_valid_test_split(all_x, all_y, train_size, valid_size, test_siz
    """
    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
    train_end = int(all_x.shape[0]*train_size)
    valid_end = train_end + int(all_x.shape[0]*valid_size)

    X_train, X_valid, X_test = all_x.iloc[:train_end,], all_x.iloc[train_
    y_train, y_valid, y_test = all_y.iloc[:train_end,], all_y.iloc[train_

    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 [20]:
```
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_spl

X_train.head()
```

Out[20]:

		Mean_Reversion_Sector_Neutral_Smoothed	Momentum_1YR
2013-01-03 00:00:00+00:00	Equity(0 [A])	-0.26276899	-1.20797813
	Equity(1 [AAL])	0.09992624	1.71347052
	Equity(2 [AAP])	1.66913824	-1.53506144
	Equity(3 [AAPL])	1.69874602	1.19311071
	Equity(5 [ABC])	-1.11399249	-0.50920924

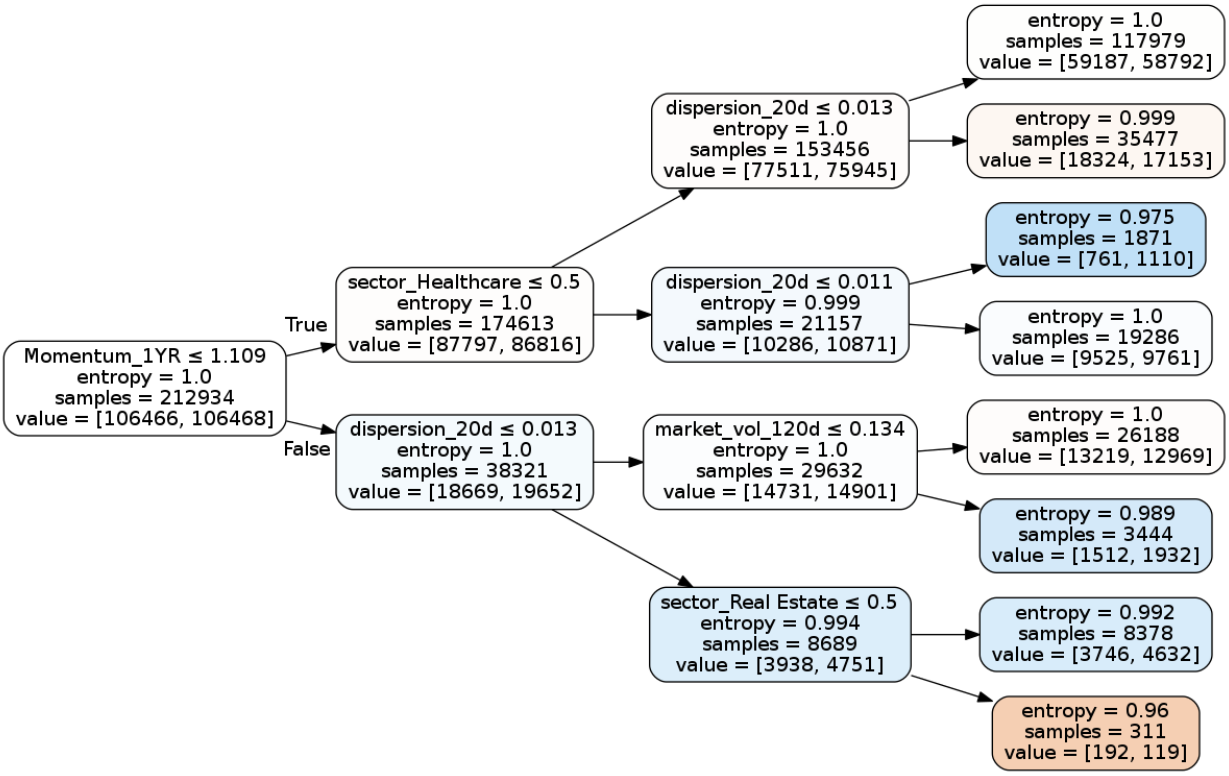
5 rows × 28 columns

Random Forests

Visualize a Simple Tree

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

In [21]:
```
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=fea
project_helper.rank_features_by_importance(simple_clf.feature_importances_
```



```
     Feature                                         Importance
 1. dispersion_20d                            (0.4689065622873918)
 2. market_vol_120d                           (0.19307524853086544)
 3. sector_Real Estate                        (0.12804098739290343)
 4. Momentum_1YR                              (0.11245888333225494)
 5. sector_Healthcare                         (0.09751831845658443)
 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)
```

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

Splitting the tree does not always (and: not alone) follow the criteria for feature importance. So we can have a tree that allocates features of high importance at some lower nodes, and - vice versa - allocates features of low importance at some higher nodes.

In the Udacity Forums https://hub.udacity.com/rooms/community:nd880:346730-project-563/community:thread-6273961378-567340?contextType=room, Martin F. has provided a valuable link on that problem (https://datascience.stackexchange.com/questions/16693/interpreting-decision-tree-in-context-of-feature-importances). From the discussion it is clear that splitting the tree follows a different criterion than the calculation of the importance.

As we can see from the above code, the criterion for splitting the tree is 'entropy'. But when calculating the importance, sklearn uses the so-called Gini importance "which is the mean decrease of the Gini Impurity for a given variable across all the trees of the random forest" (see link above).

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 [22]:
```
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.

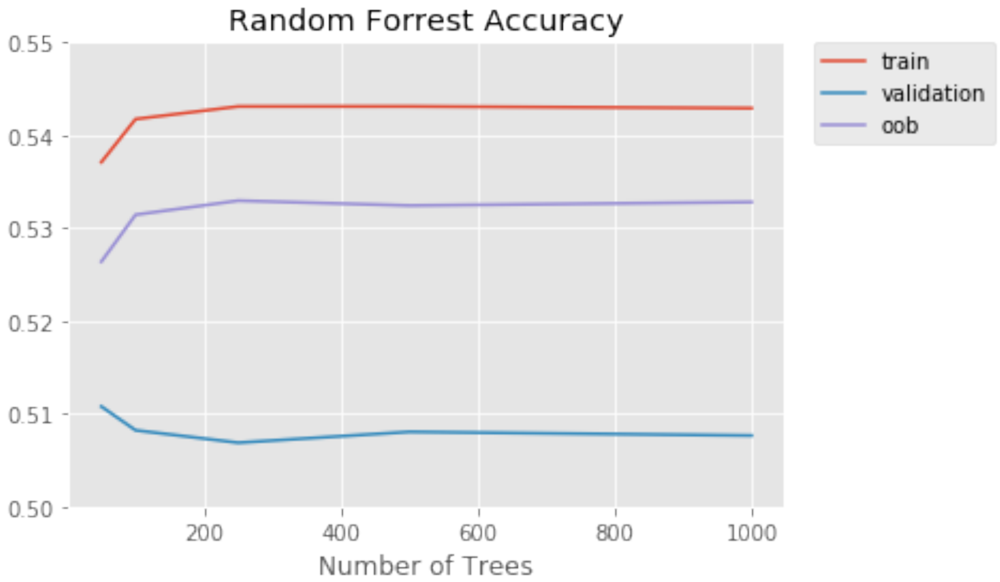
Feel free to try other values for these parameters, but also keep in mind that making too many small adjustments to hyper-parameters can lead to overfitting even the validation data, and therefore lead to less generalizable performance on the out-of-sample test set. So when trying different parameter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

In [23]:
```
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 [09:04<00:00, 108.98s/Model]
```

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

In [24]:
```
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

Answer to Q1) Accuracy is 'very good' for tree sizes greater that ca. 240, and is already 'good' for tree sizes greater than ca. 100. So tree size plays a role and size should be greater than ca. 240.

Answer to Q2) The model is overfitting as it's results are more similar to the training set than to the validation set.

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

In [25]:
```
print('Features Ranked by Average Importance:\n')
project_helper.rank_features_by_importance(np.average(feature_importances
```

```
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                                 (0.05456174767332598 6)
11. sector_Healthcare                       (0.03165203966535736 6)
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 [26]:
```
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_p
    # Calculate the Alpha Score
    prob_array=[-1,1]
    alpha_score = classifier.predict_proba(samples).dot(np.array(prob_arr

    # 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[fac
    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:

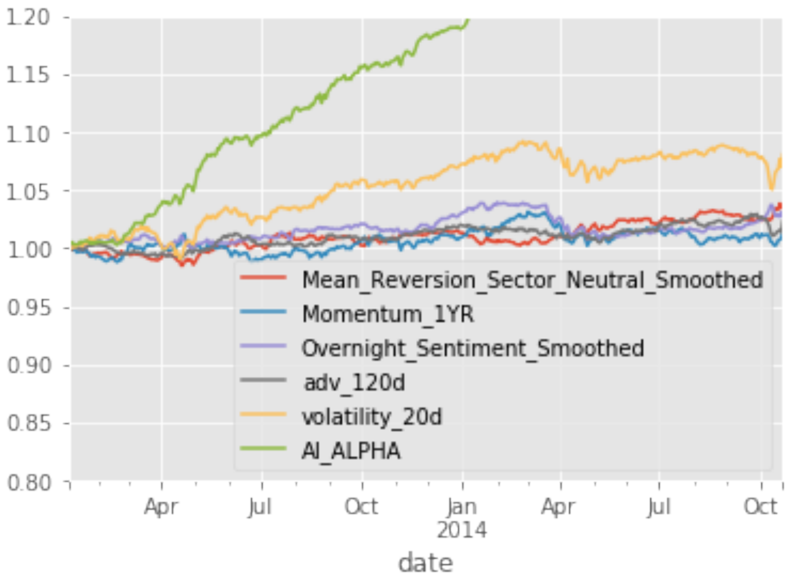
In [27]:
```
factor_names = [
    'Mean_Reversion_Sector_Neutral_Smoothed',
    'Momentum_1YR',
    'Overnight_Sentiment_Smoothed',
    'adv_120d',
    'volatility_20d']
```

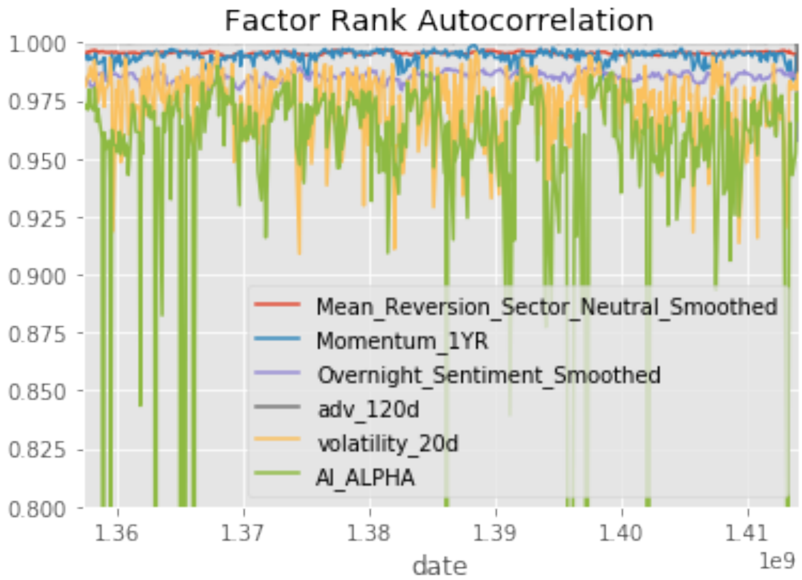
Training Prediction

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

In [28]:
```
show_sample_results(all_factors, X_train, clf, factor_names)
```

```
Cleaning Data...

Dropped 0.2% entries from factor data: 0.2% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.2% entries from factor data: 0.2% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.2% entries from factor data: 0.2% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.2% entries from factor data: 0.2% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.2% entries from factor data: 0.2% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.2% entries from factor data: 0.2% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
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
```



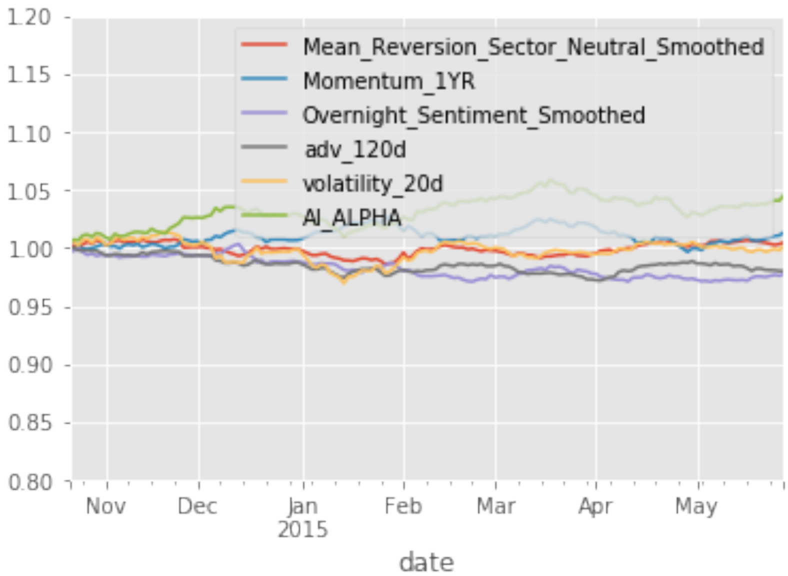


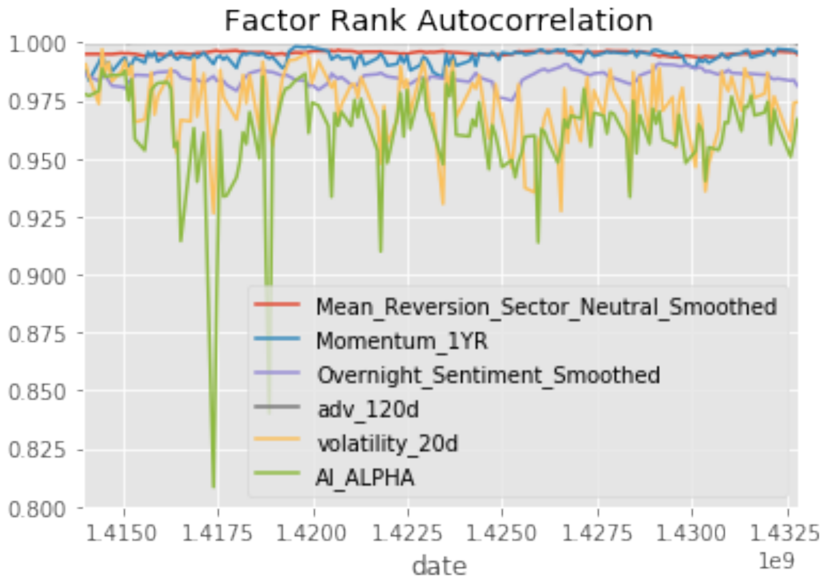
Validation Prediction

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

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

```
Cleaning Data...

Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
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.

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 [30]:
```
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
    # This Udacity Knowledge post helped me to understand that the first
    # I then used this Internet page to find a solution - which, to be ho
    # https://stackoverflow.com/questions/41289055/slice-multiindex-panda

    non_overlapping_x = x.loc[pd.IndexSlice[x.index.levels[0][start_i::(n
    non_overlapping_y = y.loc[pd.IndexSlice[y.index.levels[0][start_i::(n

    return non_overlapping_x, non_overlapping_y


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.

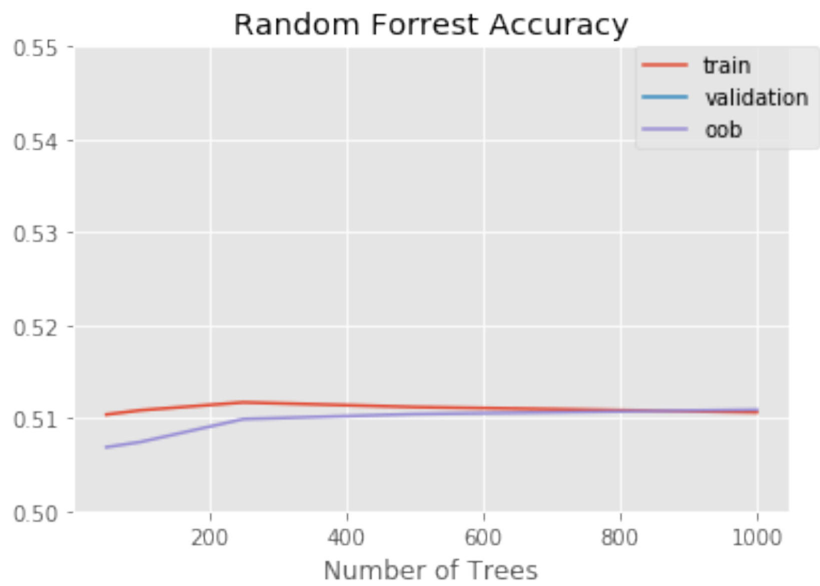
Train Model

In [31]:
```
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 [01:08<00:00, 13.75s/Model]
```

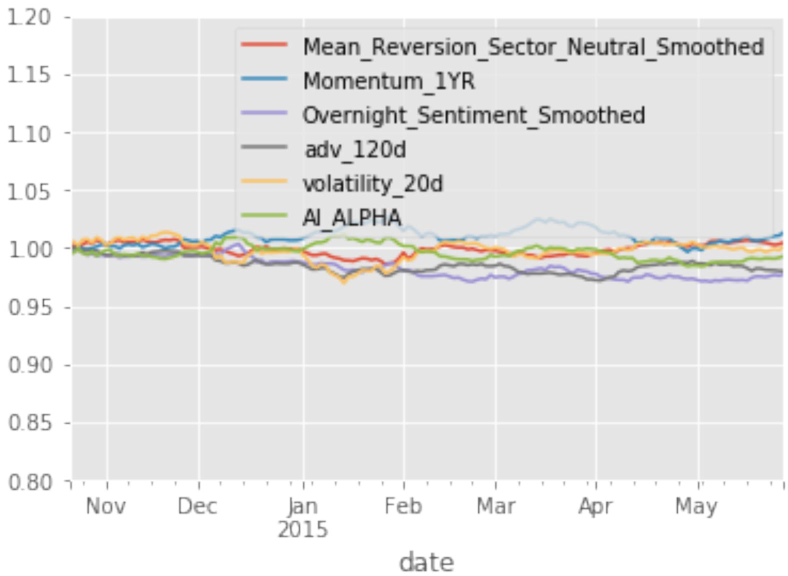
Results

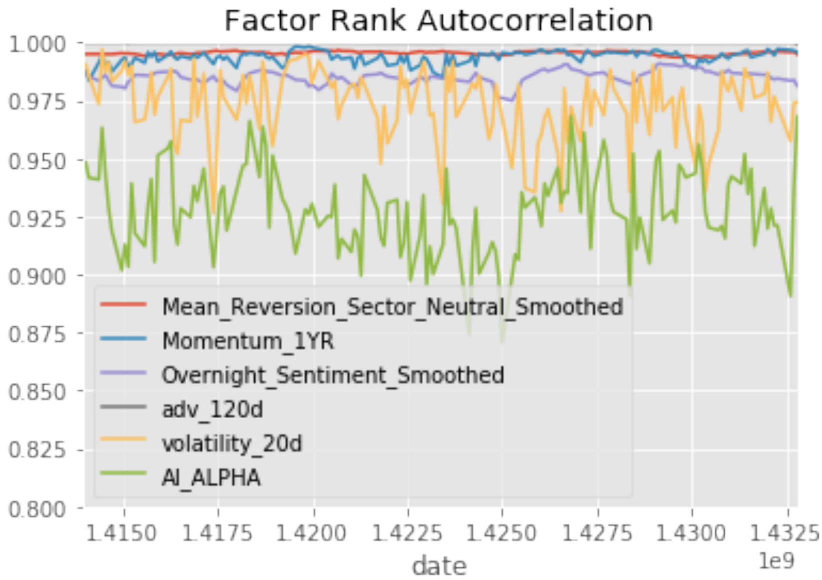
In [32]:
```
project_helper.plot(
    [n_trees_l]*3,
    [train_score, valid_score, oob_score],
    ['train', 'validation', 'oob'],
    'Random Forrest Accuracy',
    'Number of Trees')
```



In [33]:
```
show_sample_results(all_factors, X_valid, clf, factor_names)
```

```
Cleaning Data...

Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
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.

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 [38]:
```
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier


def bagging_classifier(n_estimators, max_samples, max_features, parameter
    """
    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 bas
    max_features : float
        The proportion of input sample features drawn from when training
    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',
    assert not required_parameters - set(parameters.keys())

    base_clf = DecisionTreeClassifier(
        criterion=parameters['criterion'],
        max_features=max_features,
        min_samples_leaf=parameters['min_samples_leaf']
    )

    # TODO: Implement
    clf = BaggingClassifier(
        base_estimator = base_clf,
        n_estimators = n_estimators,
        max_samples = max_samples,
        bootstrap=True,
        oob_score=parameters['oob_score'],
        n_jobs=parameters['n_jobs'],
        verbose=0,
        random_state=parameters['random_state']
    )
    clf.fit(X_train, y_train)
    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

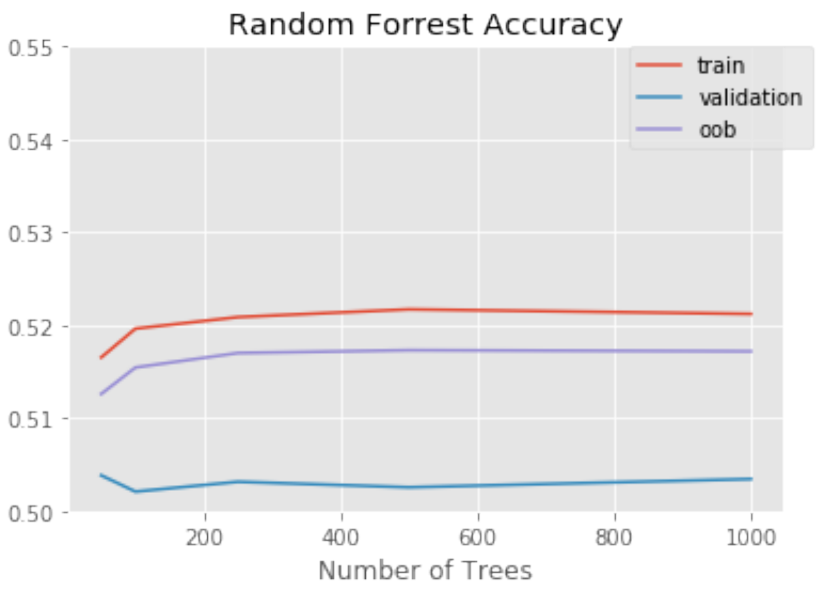
In [39]:

```
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 [26:15<00:00, 315.20s/Model]
```

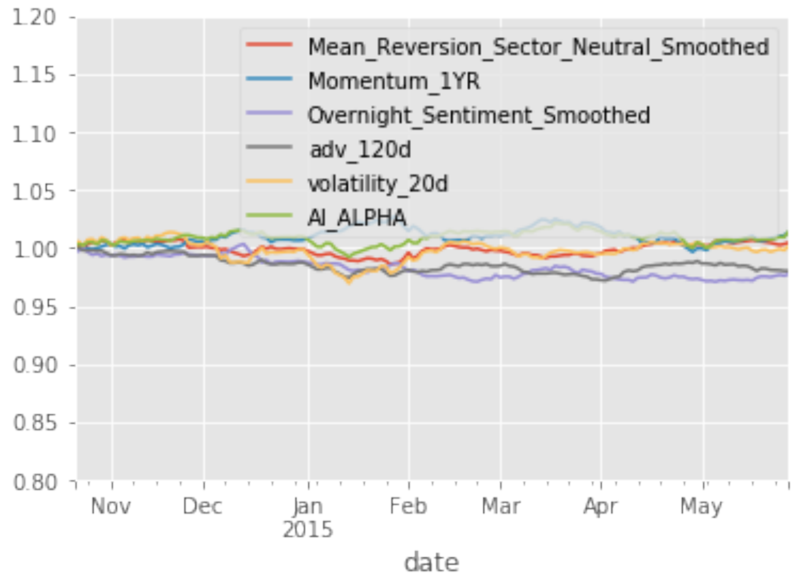
Results

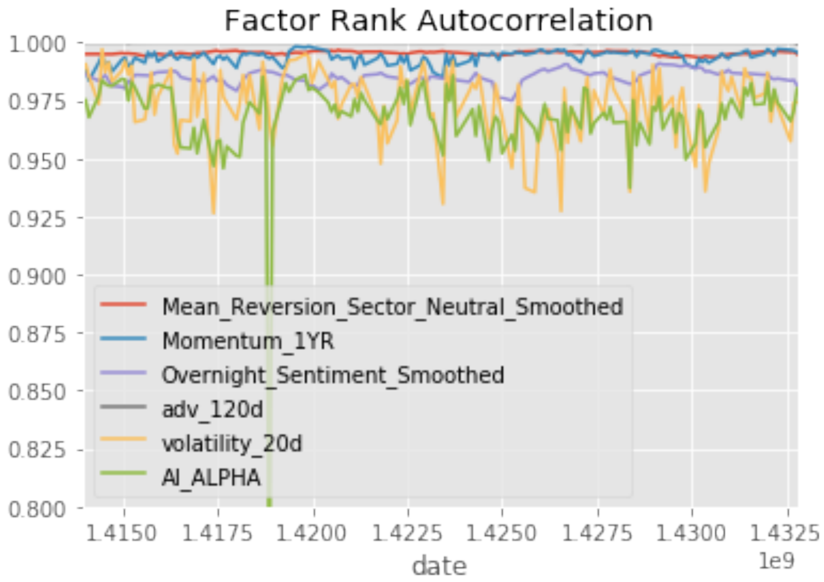
In [40]:
```
project_helper.plot(
    [n_trees_l]*3,
    [train_score, valid_score, oob_score],
    ['train', 'validation', 'oob'],
    'Random Forrest Accuracy',
    'Number of Trees')
```



In [41]:
```
show_sample_results(all_factors, X_valid, clf, factor_names)
```

```
Cleaning Data...

Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
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                                  0.78000000
dtype: float64
```





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

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 [42]:
```
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_sampl
        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_sam

        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_c
        self.named_estimators_ = Bunch(**dict(zip(estimator_names, self.e
        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 [43]:
```
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 = 0
    for clf in classifiers:
        oob_score = oob_score + clf.oob_score_

    return oob_score / len(classifiers)


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 [45]:
```
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
    fit_classifiers = []

    for i in range(len(classifiers)):
        fit_classifiers.append(
            classifiers[i].fit(x[i::n_skip_samples], y[i::n_skip_samples]
        )

    return fit_classifiers


project_tests.test_non_overlapping_estimators(non_overlapping_estimators)
```

```
Tests Passed
```

In [46]:
```
class NoOverlapVoter(NoOverlapVoterAbstract):
    def _calculate_oob_score(self, classifiers):
        return calculate_oob_score(classifiers)

    def _non_overlapping_estimators(self, x, y, classifiers, n_skip_sampl
        return non_overlapping_estimators(x, y, classifiers, n_skip_sampl
```

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

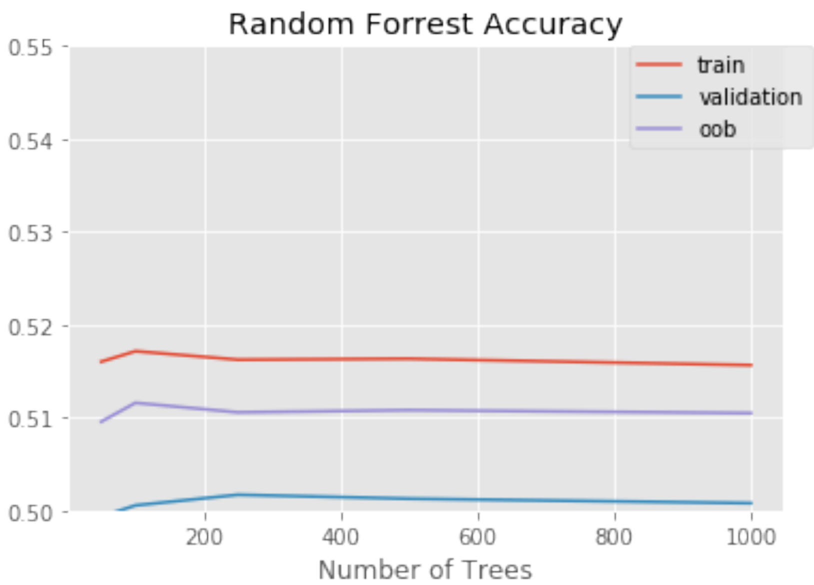
Train Model

In [47]:
```
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 [07:07<00:00, 85.58s/Model]
```

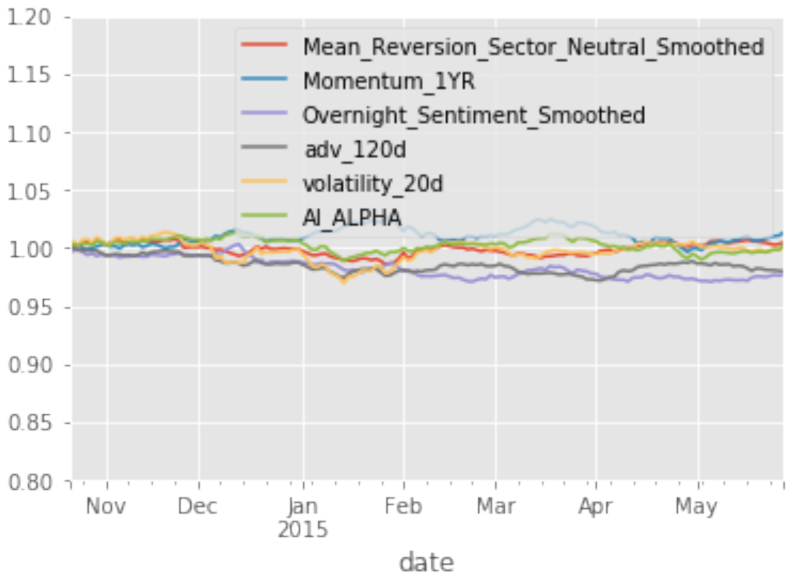
Results

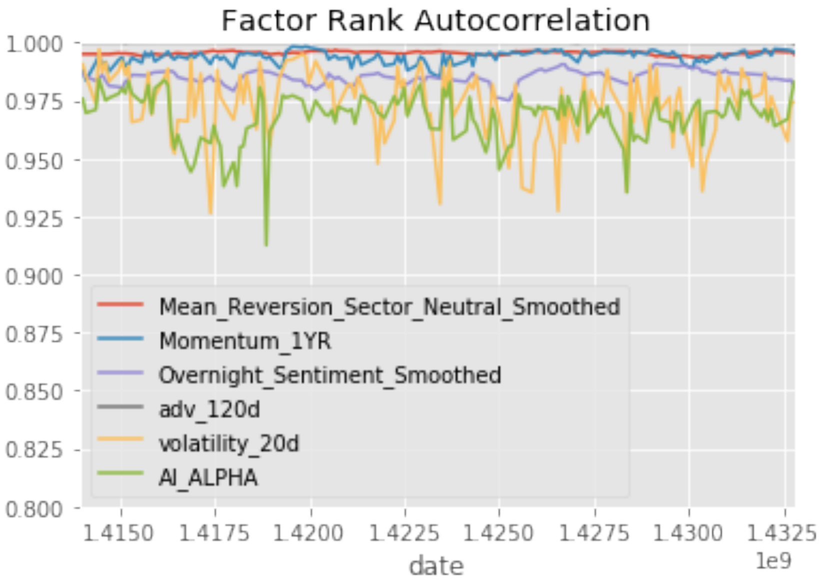
In [48]:
```
project_helper.plot(
    [n_trees_l]*3,
    [train_score, valid_score, oob_score],
    ['train', 'validation', 'oob'],
    'Random Forrest Accuracy',
    'Number of Trees')
```



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 computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
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                                  0.18000000
dtype: float64
```





Final Model

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 [50]:
```
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[50]:
```
NoOverlapVoter(estimator=None, n_skip_samples=4, voting='soft')
```

Results

Accuracy

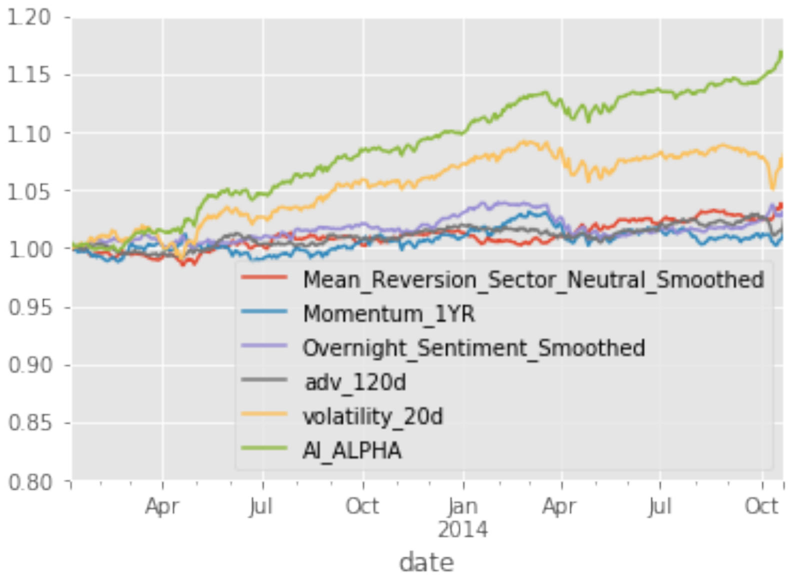
In [51]:
```
print('train: {}, oob: {}, valid: {}'.format(
    clf_nov.score(X_train, y_train.values),
    clf_nov.score(X_valid, y_valid.values),
    clf_nov.oob_score_))
```

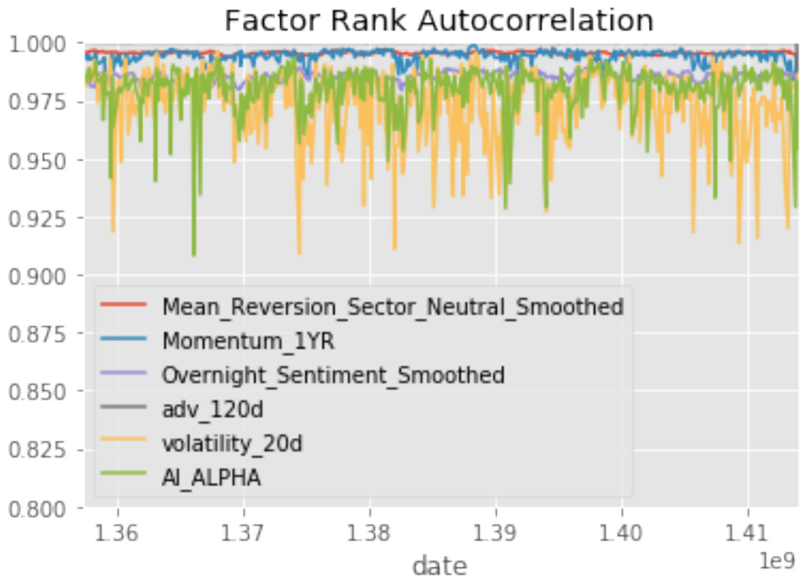
```
train: 0.517667446250951, oob: 0.5181323790470287, valid: 0.5131125573469
093
```

Train

In [52]:
```
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 computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.2% entries from factor data: 0.2% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.2% entries from factor data: 0.2% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.2% entries from factor data: 0.2% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.2% entries from factor data: 0.2% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.2% entries from factor data: 0.2% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
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                                 3.10000000
dtype: float64
```

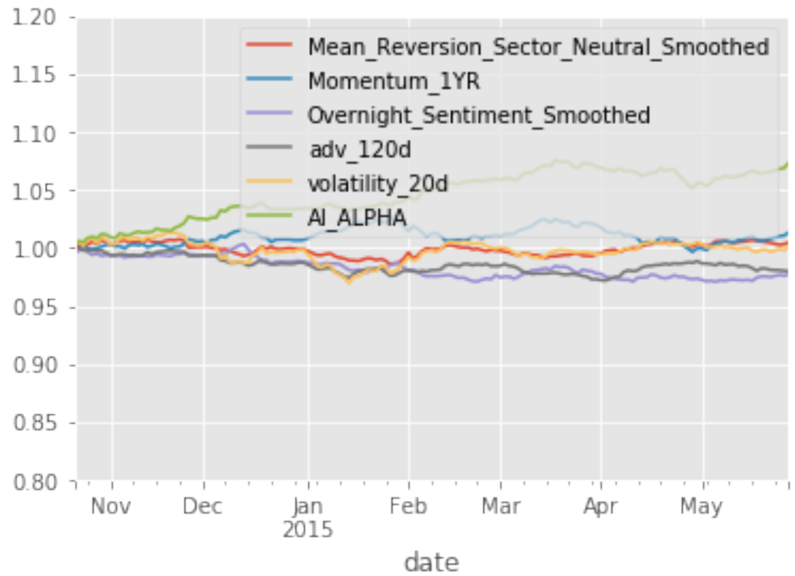


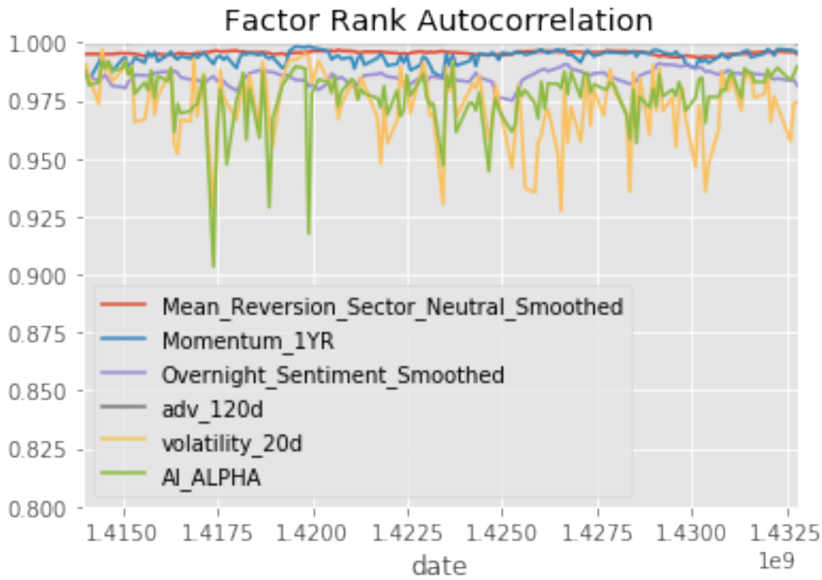


Validation

In [53]:
```
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 computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
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                                  3.75000000
dtype: float64
```

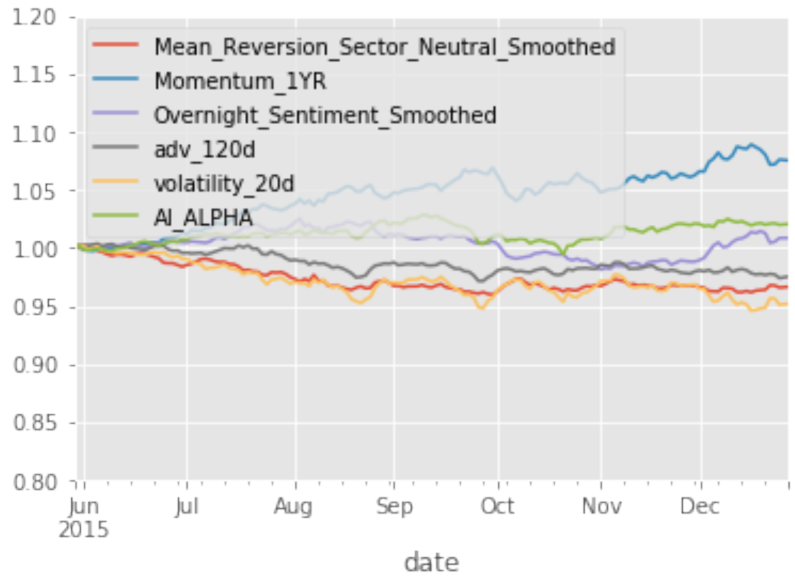


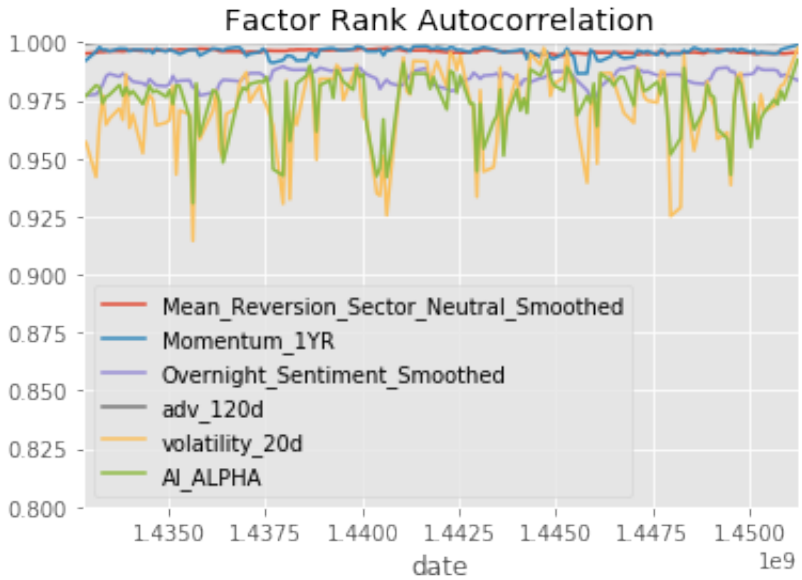


Test

In [54]:
```
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 computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.0% entries from factor data: 0.0% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
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                                  1.05000000
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.

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.