Tim/VAEPred_sym_...

```
%sh

echo $PWD
rm TXLoader.py*
wget --no-check-certificate --no-cache --no-cookies https://raw.githubusercontent.com/tianle91
ls -l --block-size=M
```

FINISHED

```
total 2M
-rw-r--r-- 1 root   root   1M Mar  1 20:53 ae_1mo-1h_SYM:TD.h5
-rw-r--r-- 1 root   root   1M Mar  1 20:27 ae.h5
-rwxrwxr-x 1 hadoop hadoop 1M Mar  1 03:50 attach.sh
-rw-r--r-- 1 root   root   1M Mar  1 20:50 Book.py
-rw-rw-r-- 1 hadoop hadoop 1M Mar  1 03:50 complete.out
drwxr-xr-x 2 root   root   1M Mar  1 22:39 data
-rw-r--r-- 1 root   root   1M Mar  1 20:50 features_all.py
-rw-r--r-- 1 root   root   1M Mar  1 20:50 features_orders_gpbyagg.py
-rw-r--r-- 1 root   root   1M Mar  1 20:50 features_trades_gpbyagg.py
drwxr-xr-x 2 root   root   1M Mar  1 18:31 __pycache__
-rw-r--r-- 1 root   root   1M Mar  1 20:50 sparkdfutils.py
drwxr-xr-x 5 hadoop hadoop 1M Mar  1 03:43 ta-lib
-rwxrwxrwx 1 hadoop hadoop 2M Dec  4 20:23 ta-lib-0.4.0-src.tar.gz
drwxrwxr-x 2 hadoop hadoop 1M Mar  1 03:42 team_keys
-rw-r--r-- 1 root   root   1M Mar  1 22:45 TXLoader.py
-rw-r--r-- 1 root   root   1M Mar  1 21:49 vae_1mo-1h_SYM:TD.h5
-rw-r--r-- 1 root   root   1M Mar  1 22:15 vaepred_1mo-1h_SYM:TD.h5
```

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:45:07 PM.

```
%sh

cd data
ls -ln --block-size=M
```

FINISHED

```
total 2M
-rw-r--r-- 1 0 0 1M Mar  1 22:36 1mo-1h_SYM:BMO_dates.pickle
-rw-r--r-- 1 0 0 1M Mar  1 20:59 1mo-1h_SYM:BMO_dt:2018-04-02 00:00:00_orders.pickle.gz
-rw-r--r-- 1 0 0 1M Mar  1 22:34 1mo-1h_SYM:BMO_dt:2018-04-02 00:00:00_trades.pickle.gz
-rw-r--r-- 1 0 0 1M Mar  1 21:04 1mo-1h_SYM:BMO_dt:2018-04-03 00:00:00_orders.pickle.gz
-rw-r--r-- 1 0 0 1M Mar  1 22:35 1mo-1h_SYM:BMO_dt:2018-04-03 00:00:00_trades.pickle.gz
-rw-r--r-- 1 0 0 1M Mar  1 21:10 1mo-1h_SYM:BMO_dt:2018-04-04 00:00:00_orders.pickle.gz
-rw-r--r-- 1 0 0 1M Mar  1 22:35 1mo-1h_SYM:BMO_dt:2018-04-04 00:00:00_trades.pickle.gz
-rw-r--r-- 1 0 0 1M Mar  1 21:14 1mo-1h_SYM:BMO_dt:2018-04-05 00:00:00_orders.pickle.gz
-rw-r--r-- 1 0 0 1M Mar  1 22:35 1mo-1h_SYM:BMO_dt:2018-04-05 00:00:00_trades.pickle.gz
-rw-r--r-- 1 0 0 1M Mar  1 21:20 1mo-1h_SYM:BMO_dt:2018-04-06 00:00:00_orders.pickle.gz
-rw-r--r-- 1 0 0 1M Mar  1 22:35 1mo-1h_SYM:BMO_dt:2018-04-06 00:00:00_trades.pickle.gz
-rw-r--r-- 1 0 0 1M Mar  1 21:25 1mo-1h_SYM:BMO_dt:2018-04-09 00:00:00_orders.pickle.gz
-rw-r--r-- 1 0 0 1M Mar  1 22:36 1mo-1h_SYM:BMO_dt:2018-04-09 00:00:00_trades.pickle.gz
-rw-r--r-- 1 0 0 1M Mar  1 21:29 1mo-1h_SYM:BMO_dt:2018-04-10 00:00:00_orders.pickle.gz
-rw-r--r-- 1 0 0 1M Mar  1 22:36 1mo-1h_SYM:BMO_dt:2018-04-10 00:00:00_trades.pickle.gz
-rw-r--r-- 1 0 0 1M Mar  1 21:34 1mo-1h_SYM:BMO_dt:2018-04-11 00:00:00_orders.pickle.gz
```

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:45:07 PM.

```
%python

import os
import sys
import gzip
import pickle

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import preprocessing

#from TXLoader import TXLoader
exec(open(os.getcwd() + '/TXLoader.py').read())
```

FINISHED

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:45:07 PM.

```
%python

symbol = 'TD'
jobname = '1mo-1h'
dotraining = False
```

FINISHED

Took 1 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:45:08 PM.

```
%python

xm = TXLoader(jobname=jobname, symbol=symbol).getxm(byday=True)
nday, ntime, ncov = xm.shape
print ('xm.shape:', xm.shape)
print (xm[0, ...])
```

FINISHED

```
xm.shape: (22, 60, 38)
[[400.0 258.0 10.0 ... 72.632563 72.6 72.68]
 [5300.0 516.0 25.0 ... 72.57698 72.53 72.61]
 [600.0 440.0 11.0 ... 72.531697 72.5 72.56]
 ...
 [2200.0 91.0 11.0 ... 72.449438 72.42 72.47]
 [300.0 506.0 3.0 ... 72.426357 72.41 72.45]
 [800.0 113.0 2.0 ... 72.418278 72.41 72.43]]
```

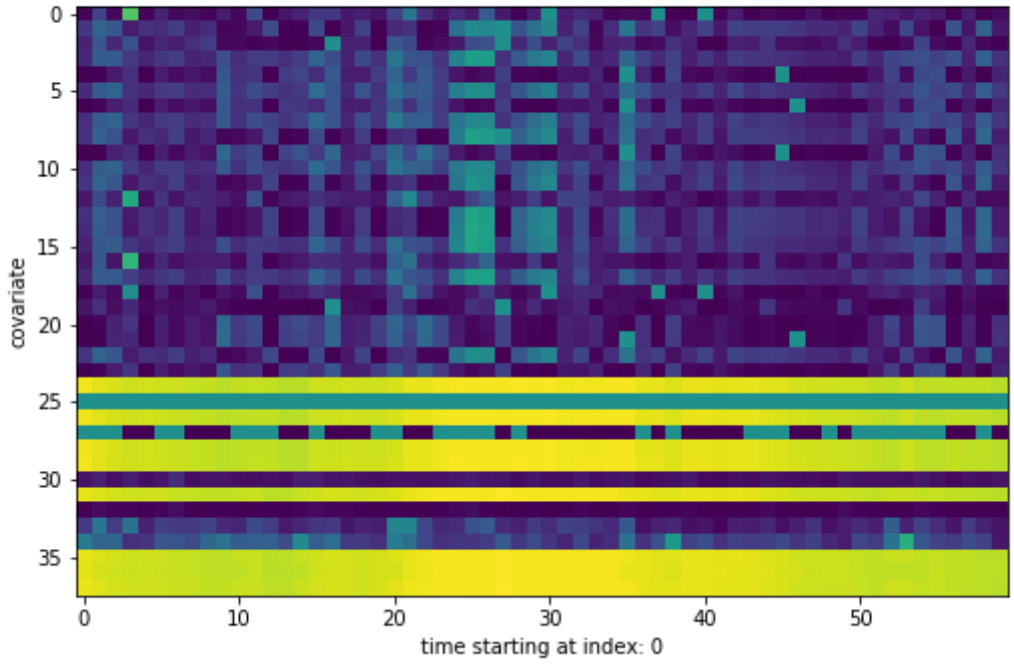
Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:45:08 PM.

```
%python

scaler = preprocessing.MinMaxScaler((-1, 1))
scaler.fit(np.concatenate([xm[i, ...] for i in range(nday)], axis=0))

xmnormd = np.concatenate([scaler.transform(xm[i, ...]).reshape((1, ntime, ncov)) for i in rang
xmnormd[np.isnan(xmnormd)] = 0

limmultiple = 5
for i in range(min(limmultiple, xm.shape[0])):
    plt.imshow(np.transpose(xmnormd[i, :, :]))
    plt.xlabel('time starting at index: %s' % (i))
```

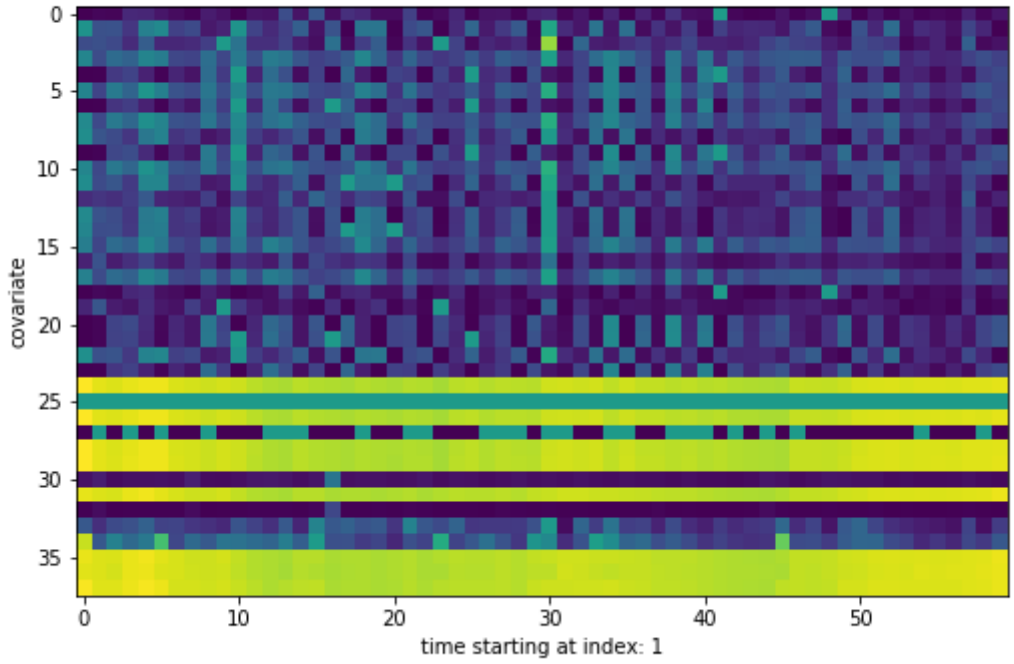
FINISHED

```
    plt.ylabel('covariate')
    plt.show()
```

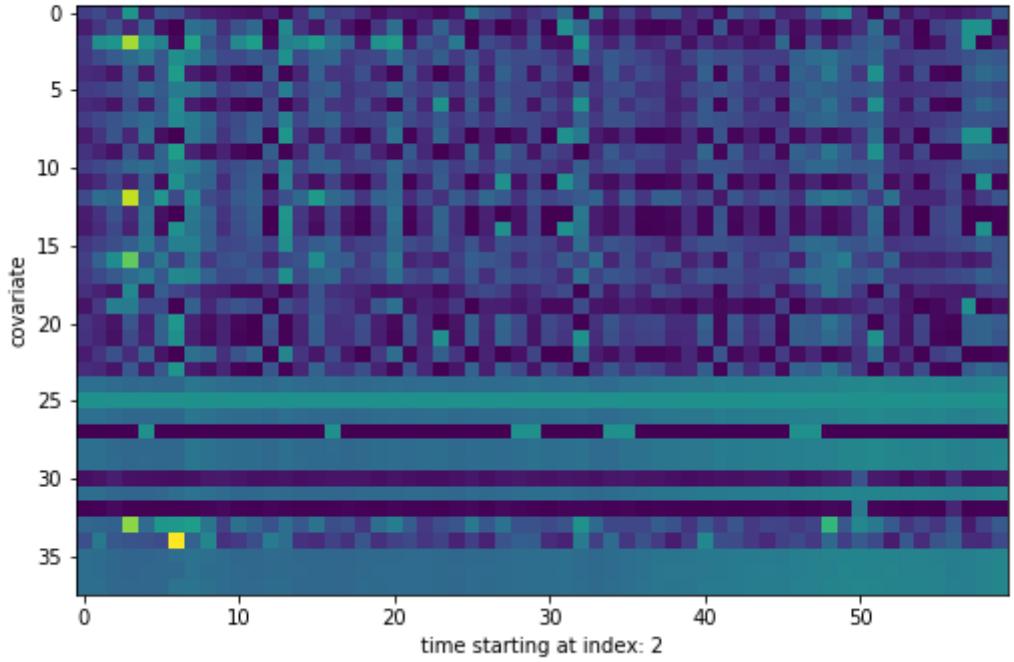
```
<matplotlib.image.AxesImage object at 0x7faff13f7cf8>
Text(0.5,29.75,'time starting at index: 0')
Text(51.25,0.5,'covariate')
```



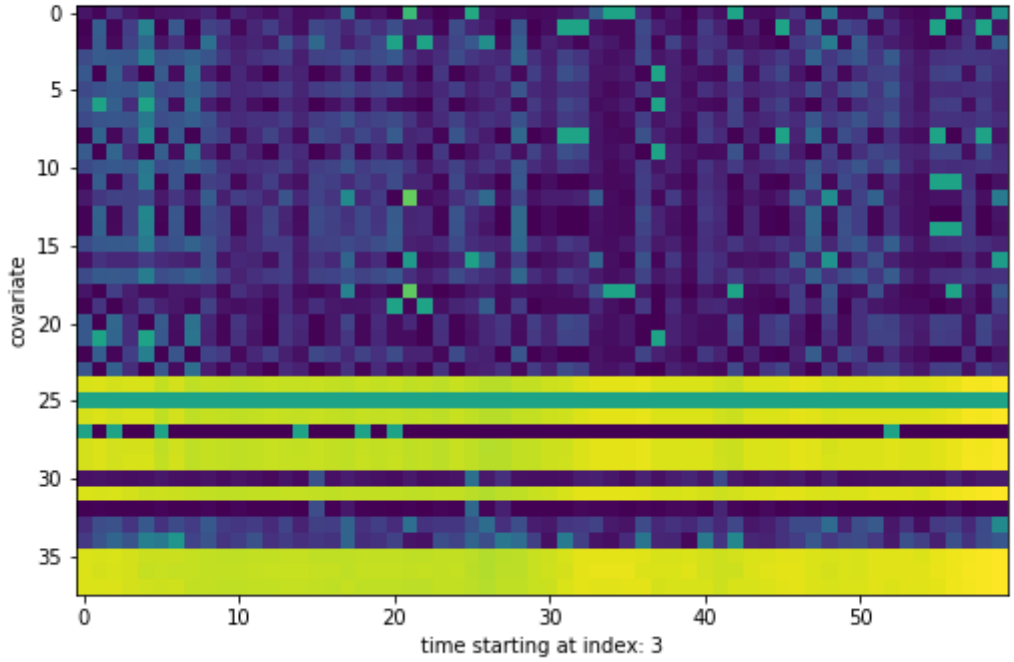
Text(0.5,29.75,'time starting at index: 1') Text(51.25,0.5,'covariate')



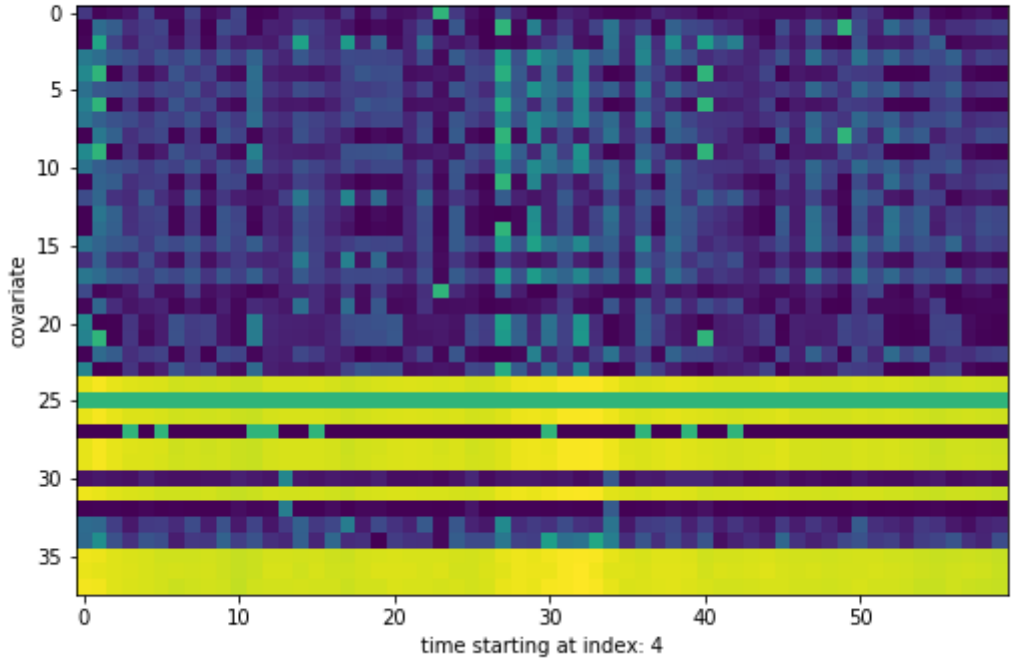
Text(0.5,29.75,'time starting at index: 2') Text(51.25,0.5,'covariate')



Text(0.5,29.75,'time starting at index: 3') Text(51.25,0.5,'covariate')



Text(0.5,29.75,'time starting at index: 4') Text(51.25,0.5,'covariate')



Took 5 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:45:13 PM.

```
%python

x, y = xmnormd[:, :-1, :], xmnormd[:, 1:, :]
print (x.shape)
nobs, ntime, ncov = x.shape
```

FINISHED

```
(22, 59, 38)
```

Took 5 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:45:13 PM.

```
%python

xflat = np.concatenate([x[i, :, :] for i in range(nobs)], axis=0)
yflat = np.concatenate([y[i, :, :] for i in range(nobs)], axis=0)
print (xflat.shape)
```

FINISHED

```
(1298, 38)
```

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:45:13 PM.

VAE from here:

FINISHED

https://github.com/keras-team/keras/blob/master/examples/variational_autoencoder.py (https://github.com/keras-team/keras/blob/master/examples/variational_autoencoder.py)

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:45:09 PM.

```
%python

from tensorflow.keras.layers import Lambda, Input, Dense
```

FINISHED

```
from tensorflow.keras.models import Model
from tensorflow.keras.losses import mse
from tensorflow.keras import backend as K
from tensorflow.keras.optimizers import Adam
```

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:45:13 PM.

```
%python

from tensorflow.python.client import device_lib
print(device_lib.list_local_devices())
```

FINISHED

```
[name: "/device:CPU:0"
device_type: "CPU"
memory_limit: 268435456
locality {
}
incarnation: 5961456697077887952
, name: "/device:XLA_CPU:0"
device_type: "XLA_CPU"
memory_limit: 17179869184
locality {
}
incarnation: 2593087358002770273
physical_device_desc: "device: XLA_CPU device"
]
```

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:45:13 PM.

```
%python

input_shape = (ncov, )
interm_dim = 16
latent_dim = 8
```

FINISHED

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:45:14 PM.

```
%python

# VAE model = encoder + decoder
# build encoder model
inputs = Input(shape=input_shape, name='encoder_input')
x = Dense(interm_dim, activation='tanh')(inputs)
z_mean = Dense(latent_dim, name='z_mean')(x)
z_log_var = Dense(latent_dim, name='z_log_var')(x)

# use reparameterization trick to push the sampling out as input
# note that "output_shape" isn't necessary with the TensorFlow backend
def sampling(args):
    z_mean, z_log_var = args
    batch = K.shape(z_mean)[0]
    dim = K.int_shape(z_mean)[1]
    # by default, random_normal has mean = 0 and std = 1.0
    epsilon = K.random_normal(shape=(batch, dim))
    return z_mean + K.exp(0.5 * z_log_var) * epsilon

z = Lambda(sampling, output_shape=(latent_dim,), name='z')([z_mean, z_log_var])
```

FINISHED

```
# instantiate encoder model
encoder = Model(inputs, [z_mean, z_log_var, z], name='encoder')
encoder.summary()
```

```
_________________________________________________________________________________________________
____
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================================
====
encoder_input (InputLayer)   (None, 38)           0
_________________________________________________________________________________________________
____
dense_18 (Dense)             (None, 16)           624         encoder_input[0][0]
_________________________________________________________________________________________________
____
z_mean (Dense)               (None, 8)            136         dense_18[0][0]
_________________________________________________________________________________________________
____
z_log_var (Dense)            (None, 8)            136         dense_18[0][0]
_________________________________________________________________________________________________
____
z (Lambda)                   (None, 8)            0           z_mean[0][0]
```

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:45:14 PM.

FINISHED

```
%python

# build decoder model
latent_inputs = Input(shape=(latent_dim,), name='z_sampling')
x = Dense(interm_dim, activation='tanh')(latent_inputs)
outputs = Dense(ncov, activation='tanh')(x)

# instantiate decoder model
decoder = Model(latent_inputs, outputs, name='decoder')
decoder.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
z_sampling (InputLayer)      (None, 8)                 0
_________________________________________________________________
dense_19 (Dense)             (None, 16)                144
_________________________________________________________________
dense_20 (Dense)             (None, 38)                646
=================================================================
Total params: 790
Trainable params: 790
Non-trainable params: 0
_________________________________________________________________
```

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:45:14 PM.

FINISHED

```
%python

# instantiate VAE model
# zsampled -> x
outputs = decoder(encoder(inputs)[2])
vae = Model(inputs, outputs, name='vae')
```

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:45:14 PM.

```
%python

# VAE loss = mse_loss or xent_loss + kl_loss
def vae_loss(ytrue, ypred):
    reconstruction_loss = ncov*mse(inputs, outputs)
    reconstruction_loss_self = ncov*mse(ytrue, ypred)
    kl_loss = 1 + z_log_var - K.square(z_mean) - K.exp(z_log_var)
    kl_loss = K.sum(kl_loss, axis=-1)
    kl_loss *= -0.5
    return K.mean(reconstruction_loss_self + reconstruction_loss + kl_loss)
```

FINISHED

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:45:14 PM.

```
%python

vae.compile(optimizer=Adam(lr=0.00001), loss=vae_loss)
vae.summary()
```

FINISHED

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
encoder_input (InputLayer)   (None, 38)                0
_________________________________________________________________
encoder (Model)              [(None, 8), (None, 8), (N 896
_________________________________________________________________
decoder (Model)              (None, 38)                790
=================================================================
Total params: 1,686
Trainable params: 1,686
Non-trainable params: 0
_________________________________________________________________
```

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:45:15 PM.

FINISHED

Run VAE

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:45:11 PM.

```
%md
```

FINISHED

```
%python

ntest = 59
# test on last day
xtrain, xtest = xflat[:-ntest, :], xflat[-ntest:, :]
ytrain, ytest = yflat[:-ntest, :], yflat[-ntest:, :]
```

FINISHED

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:45:15 PM.

```
%python

weightsfname = 'vaepred_%s_SYM:%s.h5' % (jobname, symbol)

if dotraining:
    history = vae.fit(xtrain, ytrain, epochs=5000, batch_size=64, verbose=2, validation_data=(
    vae.save_weights(weightsfname)

    # plot training history
    plt.plot(history.history['loss'], color='green', label='training_loss')
    plt.plot(history.history['val_loss'], color='red', label='validation_loss')
    plt.xlabel('iters')
    plt.ylabel('loss')
    plt.legend()
    plt.show()

else:
    vae.load_weights(weightsfname)
```

FINISHED

Took 1 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:45:16 PM.

FINISHED

Extract means of latent projections

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:45:12 PM.

```
%python

latentprojections = encoder.predict(xflat)
zmean, zlogvar, zsampled = latentprojections
zstd = np.exp(.5*zlogvar)
```

FINISHED

Took 1 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:45:16 PM.

```
%python

from sklearn.manifold import TSNE

# find 2-dim embeddings of zmean
zmeanembd = TSNE(n_components=2).fit_transform(zmean)
```

FINISHED

Took 10 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:45:26 PM.

FINISHED

GMM provided by https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html#sklearn.mixture.GaussianMixture (https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html#sklearn.mixture.GaussianMixture)

Tuning for number of components by https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html#sklearn.model_selection.Gri (https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html#sklearn.model_selection.Gri

Using `CV+loglike` instead of `AIC` mostly because not implemented in `GridSearchCV` and also because we are already doing `CV`

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:45:12 PM.

```
%python

from sklearn.mixture import GaussianMixture
from sklearn.model_selection import GridSearchCV

param_grid = {
    'n_components': np.arange(1, latent_dim),
    'covariance_type': ['full', 'tied', 'diag', 'spherical']
    #'covariance_type': ['diag']
}

cv = GridSearchCV(GaussianMixture(random_state=0), param_grid=param_grid)
cv.fit(zmean)
```

FINISHED

```
GridSearchCV(cv='warn', error_score='raise-deprecating',
       estimator=GaussianMixture(covariance_type='full', init_params='kmeans', max_iter=100,
        means_init=None, n_components=1, n_init=1, precisions_init=None,
        random_state=0, reg_covar=1e-06, tol=0.001, verbose=0,
        verbose_interval=10, warm_start=False, weights_init=None),
       fit_params=None, iid='warn', n_jobs=None,
       param_grid={'covariance_type': ['full', 'tied', 'diag', 'spherical'], 'n_components': a
rray([1, 2, 3, 4, 5, 6, 7])},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=0)
```

Took 1 min 18 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:46:34 PM.

```
%python

print (cv.best_params_)
GMMo = cv.best_estimator_
```

FINISHED

```
{'covariance_type': 'full', 'n_components': 5}
```

Took 1 min 8 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:46:34 PM.

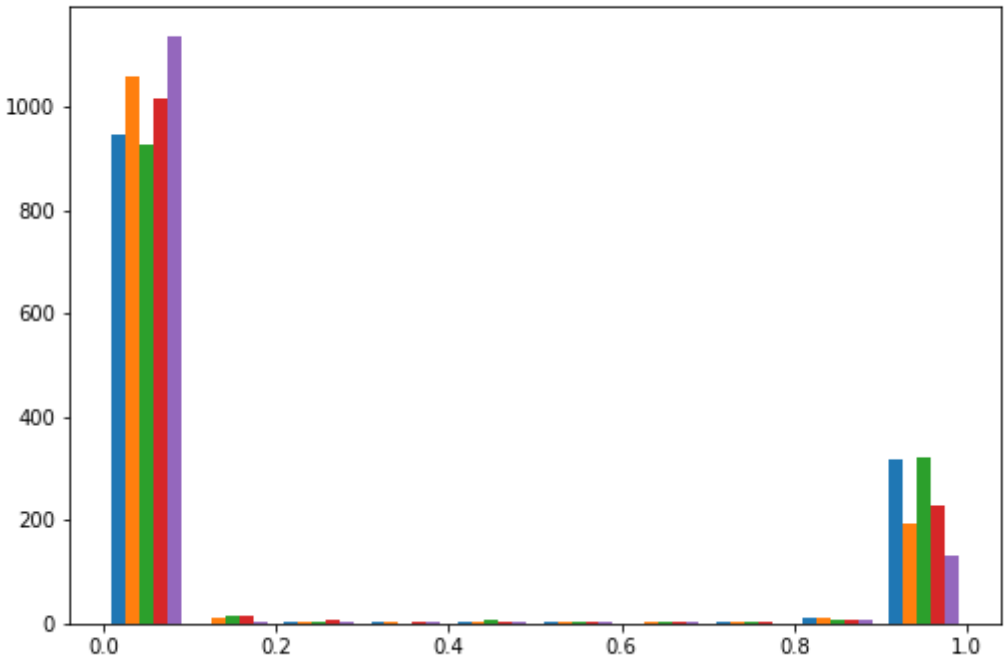
```
%python

gmmlabel = GMMo.fit_predict(zmean)
```

FINISHED

Took 2 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:46:37 PM.

```
%python

gmmproba = GMMo.predict_proba(zmean)
plt.hist(gmmproba)
plt.show()
```

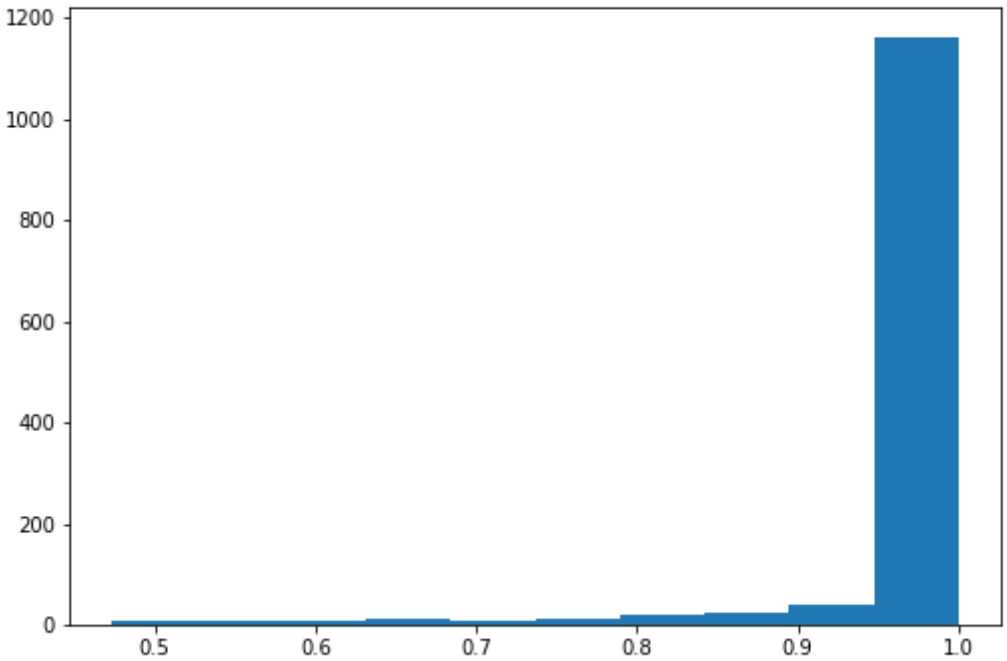
FINISHED



Took 6 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:46:41 PM.

FINISHED

```
%python

# max prob over all classes
gmmprobamax = np.max(gmmproba, axis=1)
plt.hist(gmmprobamax)
plt.show()
```



Took 4 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:46:42 PM.

```
%python

# define abnormal condition as: unlikely to be any class
pcutoff = .1
isabnormal = np.all(gmmproba <= pcutoff, axis=1)
np.sum(isabnormal)
```

FINISHED

```
0
```

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:46:42 PM.

```
%python

np.unique(gmmlabel, return_counts=True)
```

FINISHED

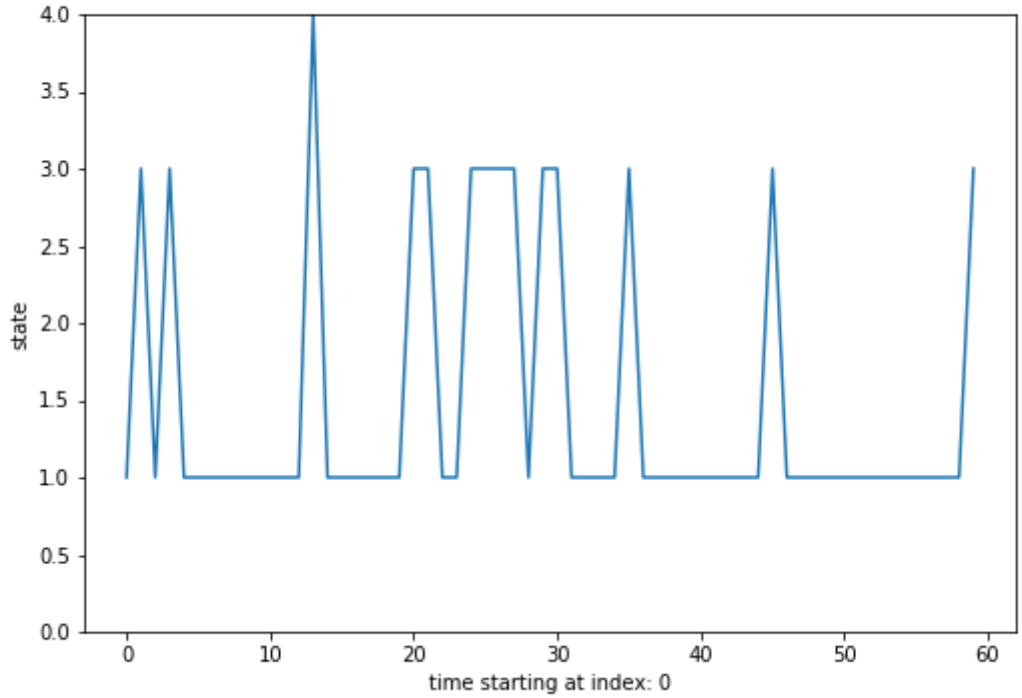
```
(array([0, 1, 2, 3, 4]), array([339, 217, 344, 249, 149]))
```

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:46:42 PM.

```
%python

limmultiple = 5
for i in range(0, min(limmultiple*60, ntime), 60):
    plt.plot(gmmlabel[i:(i+60)])
    plt.xlabel('time starting at index: %s' % (i))
    plt.ylabel('state')
    plt.ylim((0, max(np.unique(gmmlabel))))
    plt.show()
```

FINISHED

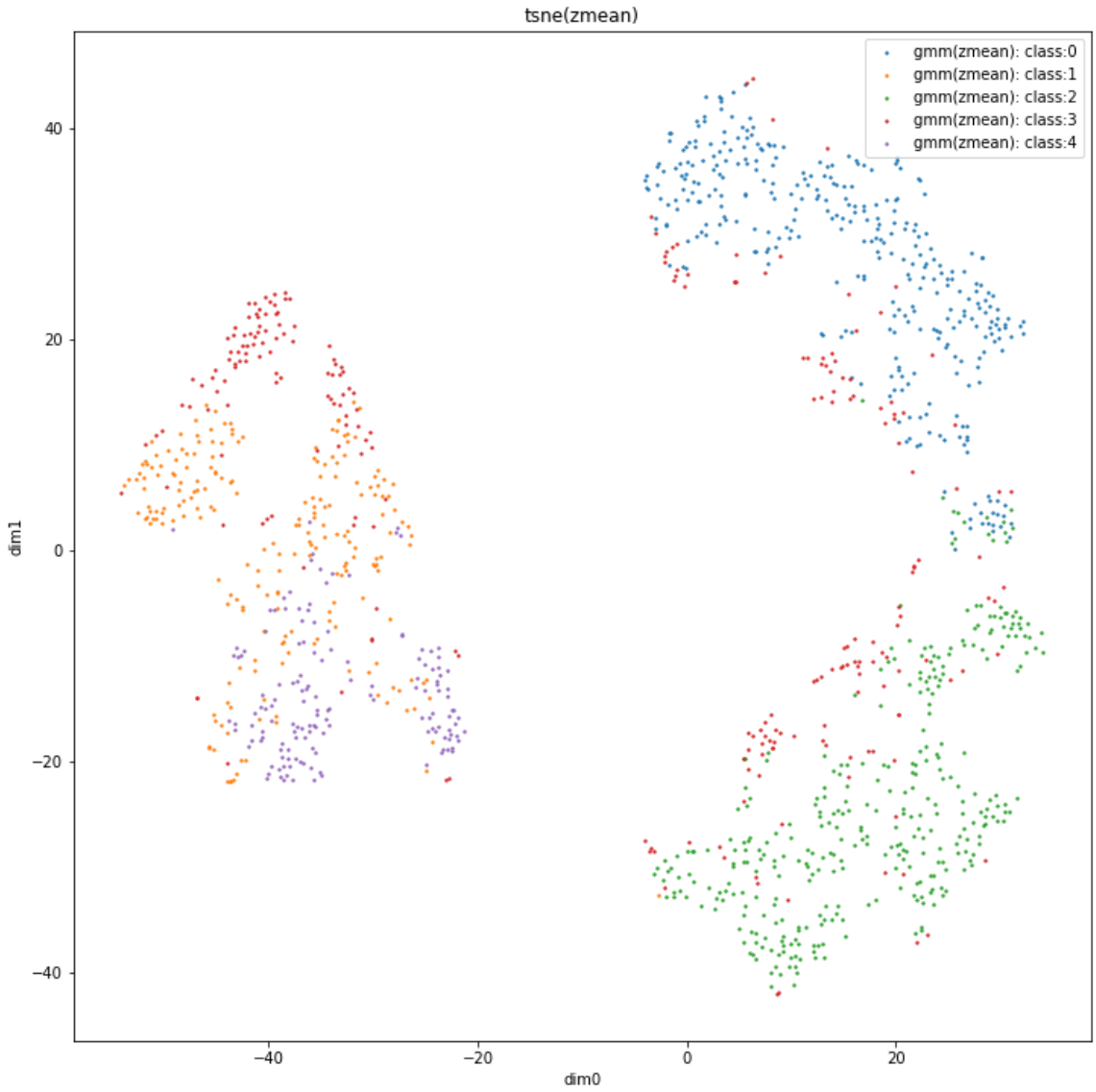
```
[<matplotlib.lines.Line2D object at 0x7faf8c7a7828>]
Text(0.5,23,'time starting at index: 0')
Text(48,0.5,'state')
(0, 4)
```



Took 1 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:46:43 PM.

FINISHED

```
%python

plt.figure(figsize=(12, 12))

for classdex in np.unique(gmmlabel):
    isclass = gmmlabel == classdex
    plt.scatter(zmeanembd[isclass, 0], zmeanembd[isclass, 1], s=2, label='gmm(zmean): class:%s

plt.title('tsne(zmean)')
plt.xlabel('dim0')
plt.ylabel('dim1')
plt.legend()
plt.show()
```



Took 3 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:46:45 PM.

FINISHED

Evaluation

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:45:14 PM.

FINISHED

```
%python

TXLoader(jobname=jobname, symbol=symbol).getcovnames()
```

```
['orders:sum(ABS(book_change))_for_type:Executed_side:Buy_orders_at_touch', 'orders:count(*)_f
or_type:New_side:Sell_orders_at_touch', 'orders:count(*)_for_type:Executed_side:Sell_orders_at
_touch', 'orders:sum(ABS(book_change))_for_type:New_side:All_orders_at_touch', 'orders:sum(ABS
```

```
(book_change))_for_type:New_side:Buy_orders_at_touch', 'orders:count(*)_for_type:Cancelled_sid
e:All_orders_at_touch', 'orders:sum(ABS(book_change))_for_type:Cancelled_side:Buy_orders_at_to
uch', 'orders:count(*)_for_type:New_side:All_orders_at_touch', 'orders:sum(ABS(book_change))_f
or_type:New_side:Sell_orders_at_touch', 'orders:count(*)_for_type:New_side:Buy_orders_at_touc
h', 'orders:count(*)_for_type:All_side:All_orders_at_touch', 'orders:count(*)_for_type:Cancell
ed_side:Sell_orders_at_touch', 'orders:count(*)_for_type:Executed_side:All_orders_at_touch',
 'orders:sum(ABS(book_change))_for_type:All_side:Sell_orders_at_touch', 'orders:sum(ABS(book_c
hange))_for_type:Cancelled_side:Sell_orders_at_touch', 'orders:sum(ABS(book_change))_for_type:
Cancelled_side:All_orders_at_touch', 'orders:sum(ABS(book_change))_for_type:Executed_side:All_
orders_at_touch', 'orders:sum(ABS(book_change))_for_type:All_side:All_orders_at_touch', 'order
s:count(*)_for_type:Executed_side:Buy_orders_at_touch', 'orders:sum(ABS(book_change))_for_typ
e:Executed_side:Sell_orders_at_touch', 'orders:count(*)_for_type:All_side:Buy_orders_at_touc
h', 'orders:count(*)_for_type:Cancelled_side:Buy_orders_at_touch', 'orders:count(*)_for_type:A
ll_side:Sell_orders_at_touch', 'orders:sum(ABS(book_change))_for_type:All_side:Buy_orders_at_t
```

Took 2 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:46:45 PM.

%python FINISHED

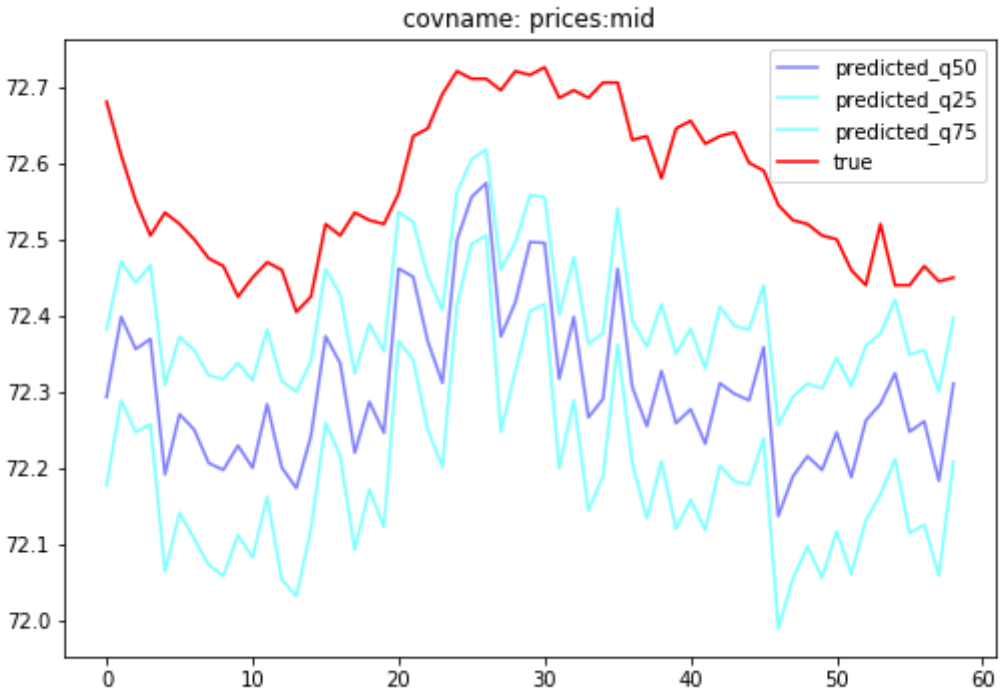
```
def mcsample(nsamples, xm):
    resl = [scaler.inverse_transform(vae.predict(xm)) for i in range(nsamples)]
    resl = [l.reshape((1,)+l.shape) for l in resl]
    return np.concatenate(resl, axis=0)


def evaluate(xm, covname, xlim=None, plot=True):

    covnames = TXLoader(jobname=jobname, symbol=symbol).getcovnames()
    iscov = np.array(covnames)==covname

    predmcsample = mcsample(1000, xm)
    pred = np.quantile(predmcsample, q=.50, axis=0)[:, iscov]
    predlo = np.quantile(predmcsample, q=.25, axis=0)[:, iscov]
    predhi = np.quantile(predmcsample, q=.75, axis=0)[:, iscov]
    true = scaler.inverse_transform(xm)[:, iscov]
    print ('mape_q50: %.6f' % (np.mean(np.abs(pred-true)/true)))

    if plot:

        predplot = pred
        trueplot = true
        predloplot = predlo
        predhiplot = predhi
        if xlim is not None:
            predplot = predplot[xlim[0]:xlim[1]]
            trueplot = trueplot[xlim[0]:xlim[1]]
            predloplot = predloplot[xlim[0]:xlim[1]]
            predhiplot = predhiplot[xlim[0]:xlim[1]]

        plt.plot(predplot, color=(0,0,1,.5), label='predicted_q50')
        plt.plot(predloplot, color=(0,1,1,.5), label='predicted_q25')
        plt.plot(predhiplot, color=(0,1,1,.5), label='predicted_q75')
        plt.plot(trueplot, color='red', label='true')
        plt.title('covname: %s' % (covname))
        plt.legend()
        plt.show()

    return pred, predlo, predhi, true
```

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:46:45 PM.

```
%python

evaluate(xtrain, 'prices:mid', xlim=(0, 59))
```

FINISHED

```
mape_q50: 0.001377
```



(array([[72.29366684], [72.39852905], [72.35641861], ..., [72.57804871], [72.55666733], [72.50454712]]),
array([[72.17846107], [72.28925133], [72.24753952], ..., [72.50934601], [72.48011971], [72.41821671]]),
array([[72.38204765], [72.47106743], [72.44304276], ..., [72.62014389], [72.60622025], [72.56547737]]),
array([[72.68], [72.61], [72.55], ..., [72.58], [72.53], [72.515]]))

Took 48 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:47:34 PM.

```
%python

evaluate(xtrain, 'orders:sum(ABS(book_change))_for_type:New_side:All_orders_at_touch', xlim=(0
```

FINISHED

```
mape_q50: 0.431556
```

(array([[109467.609375], [119297.19921875], [119555.31640625], ..., [153166.953125], [150545.7578125], [137193.9296875]]), array([[100216.77539062], [108190.01953125], [108165.7890625], ..., [139831.2109375], [136868.2421875], [123150.63085938]]), array([[118658.05859375], [131114.1953125], [130385.19726562], ..., [165343.9453125], [163724.62890625], [150035.9609375]]), array([[57500.], [125600.], [126800.], ..., [289380.], [283270.], [264800.]]))

Took 1 min 27 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:48:13 PM.

```
%python

evaluate(xtest, 'prices:mid')
```

FINISHED

```
mape_q50: 0.001248
```

(array([[71.99450684], [72.04568863], [72.43992615], [72.01128769], [72.20355988], [72.0489769], [71.97920227], [72.11100006], [71.92730331], [71.90798187], [72.20116425], [71.94454575], [71.85734558], [72.09643173], [71.97665787], [71.88204193], [71.92353821], [71.91753769], [71.94048691], [71.94974518], [71.98455048], [71.92882919], [72.02614212], [71.9088974], [71.87502289], [71.97882462], [71.93883133], [72.06090546], [71.94087219], [72.01427078], [71.97688293], [72.17737579], [72.098629], [71.97194672], [71.98358536], [72.0538063], [71.98544312], [71.97880554], [71.97371292], [71.93950653], [71.97964478], [72.0905571], [72.05789185], [72.04795074], [72.02835846], [71.95025635], [72.03087997], [71.97578812], [72.02121735], [71.99814224], [71.96060562], [71.98999405], [72.0334549], [72.03845215], [72.00173187], [72.09482956], [72.05408096], [72.03274536], [71.99305725]]), array([[71.81659889], [71.86475563], [72.27483368], [71.86656761], [72.0527916], [71.88312912], [71.8016758], [71.94447136], [71.73274231], [71.71689415], [72.03279305], [71.78700066], [71.66506958], [71.93457031], [71.81850433], [71.71994209], [71.74464417], [71.75328636], [71.77628899], [71.78231239], [71.81467247], [71.74834633], [71.84255791], [71.72138214], [71.70186424], [71.79885101], [71.77629852], [71.86528969], [71.76659012], [71.84283257], [71.81966591], [72.02639198], [71.94404221], [71.81609726], [71.82714462], [71.8857975], [71.82831001], [71.82259369], [71.81293678], [71.78624725], [71.81121826], [71.92359161], [71.89379883], [71.89562798], [71.85575867], [71.78642464], [71.87174988], [71.81402206], [71.85982513], [71.82544708], [71.80129814], [71.83527756], [71.87089157], [71.87823105], [71.8592701], [71.94782639], [71.89756393], [71.87535858], [71.82551575]]), array([[72.15412331], [72.21012878], [72.54784775], [72.15576553], [72.32389641], [72.18899536], [72.12300873], [72.25035667], [72.08368874], [72.04120636], [72.33249664], [72.07881546], [72.00214195], [72.23057747], [72.11605453], [72.03372002], [72.07114792], [72.06768799], [72.099226], [72.09237099], [72.11372185], [72.10585594], [72.17290306], [72.05641556], [72.02617836], [72.13193703], [72.08582115], [72.19863129], [72.07448578], [72.15501022], [72.1162281], [72.29935455], [72.22194672], [72.10587692], [72.11459923], [72.19561768], [72.10868073], [72.12505722], [72.10613251], [72.08874893], [72.13114738], [72.20524025], [72.20061874], [72.17337799], [72.15582466], [72.08418846], [72.15670395], [72.11971092], [72.14653778], [72.13918495], [72.09453964], [72.11033058], [72.16416931], [72.15355682], [72.13791275], [72.24213409], [72.18110085], [72.17023849], [72.12020111]]), array([[72.045], [72.085], [72.125], [72.055], [72.035], [71.985], [71.97], [71.995], [71.995], [72.005], [72.005], [71.99], [71.965], [71.965], [71.995], [71.975], [71.985], [71.985], [71.975], [71.945], [71.975], [71.935], [71.935], [71.97], [71.955], [71.965], [72.005], [72.005], [72.005], [72.025], [72.055], [72.035], [72.1], [72.115], [72.115], [72.095],

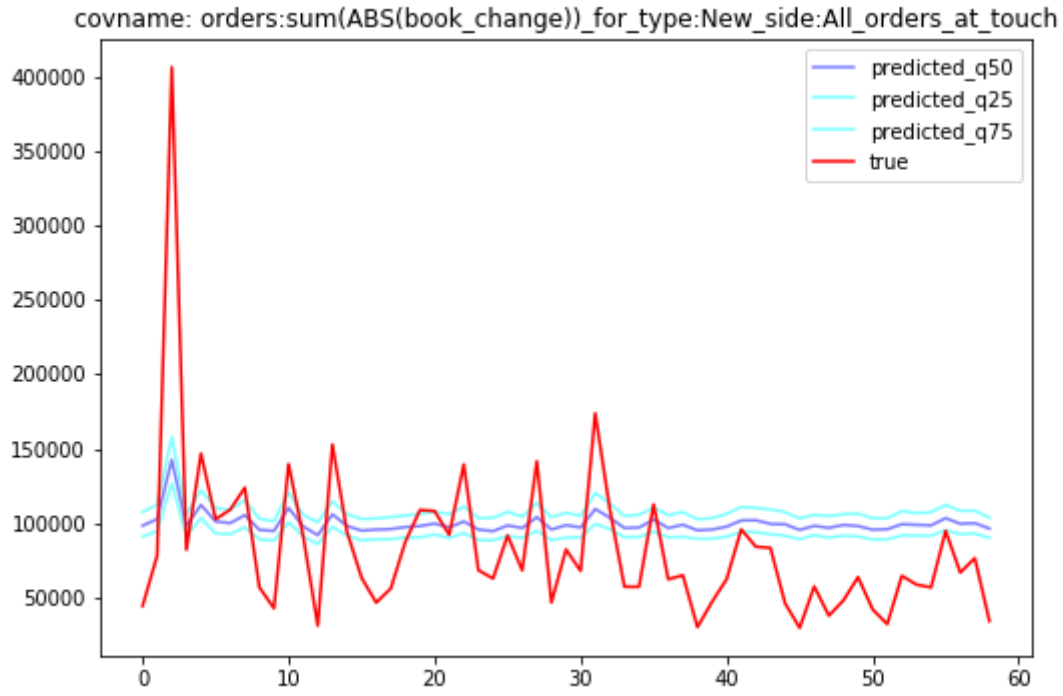
[72.105], [72.085], [72.105], [72.105], [72.105], [72.135], [72.155], [72.09], [72.095], [72.115], [72.125], [72.125], [72.145], [72.105], [72.125], [72.115], [72.105], [72.155], [72.125], [72.155], [72.145], [72.165], [72.15]]))

Took 45 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:48:19 PM.

```
%python

evaluate(xtest, 'orders:sum(ABS(book_change))_for_type:New_side:All_orders_at_touch')
```

FINISHED

```
mape_q50: 0.610938
```



(array([[98269.75], [102696.43359375], [142593.4140625], [98032.53515625], [112199.2734375], [101129.11328125], [99970.51171875], [105393.8828125], [95430.10546875], [94730.75390625], [110229.7578125], [97724.7265625], [91827.34375], [105937.98828125], [98040.06640625], [94928.71484375], [95623.2734375], [95945.11328125], [97345.421875], [98020.9921875], [99615.67578125], [97139.5], [101046.4921875], [95619.65234375], [94589.6328125], [98376.26953125], [96684.26953125], [104025.11328125], [95649.0546875], [98571.81640625], [97025.29296875], [109448.265625], [103647.90234375], [96674.6171875], [96999.22265625], [102369.7734375], [96656.67578125], [98873.2578125], [95129.5], [95612.25], [97585.0546875], [101747.0625], [101909.2578125], [99449.8359375], [99179.84765625], [95388.2421875], [98129.015625], [96758.63671875], [98747.9453125], [98005.40234375], [95555.578125], [95881.56640625], [99237.1015625], [98824.67578125], [98459.49609375], [103309.359375], [99560.5234375], [99920.484375], [96277.91015625]]), array([[90924.93945312], [94592.71679688], [126695.46679688], [91489.765625], [103361.49414062], [93066.25195312], [92552.16015625], [97251.78125], [88995.625], [88236.15625], [100606.82226562], [90954.55664062], [85920.71875], [97500.1953125], [91453.18554688], [88458.08789062], [89096.77539062], [89233.88476562], [90274.2890625], [90424.6875], [92318.76171875], [89956.06445312], [93245.25585938], [88562.99609375], [88336.88671875], [91337.44921875], [89977.33007812], [94797.08789062], [88528.83789062], [90223.73632812], [90386.06054688], [99399.41601562], [95980.74414062], [90488.09960938], [

90708.57226562], [94364.31835938], [90454.46875], [90778.26367188], [89416.97070312], [89461.5390625], [90838.359375], [94097.13085938], [93933.92773438], [92523.72851562], [91433.93359375], [89110.51757812], [91912.09375], [90211.08007812], [91480.96875], [90979.80664062], [88935.48242188], [89088.9296875], [91726.34179688], [91556.45703125], [91440.85546875], [95153.06054688], [92499.51171875], [93028.25], [90046.34179688]]), array([[107492.59960938], [112318.19921875], [157994.54296875], [105845.11132812], [122252.86328125], [110220.47851562], [108811.43359375], [115570.01757812], [102822.73828125], [101303.40429688], [120641.44921875], [105569.85546875], [100517.50976562], [114536.71484375], [106021.81640625], [102721.70507812], [103100.234375], [104147.15039062], [104992.4453125], [106279.13085938], [107690.45898438], [105986.95898438], [111151.18359375], [103352.50195312], [103437.53710938], [107591.74414062], [104231.27539062], [113572.38476562], [103941.33398438], [106929.25390625], [104939.14648438], [120291.34179688], [113232.63476562], [104776.86523438], [105554.6171875], [110510.52148438], [105310.375], [107406.56835938], [102542.51171875], [103247.76953125], [106191.59570312], [110795.953125], [110237.42773438], [109138.30859375], [107343.25390625], [103203.13085938], [105505.88085938], [104766.45117188], [105929.9375], [106349.3515625], [103350.1796875], [103302.37890625], [108094.59570312], [106704.12109375], [107321.47851562], [112243.11132812], [108222.53125], [108347.77929688], [103605.7578125]]), array([[44200.], [78200.], [406800.], [82000.], [146800.], [102600.], [109000.], [123800.], [56719.], [42700.], [139775.], [91200.], [30947.], [152900.], [94200.], [63200.], [46500.], [56000.], [87500.], [108800.], [108100.], [92100.], [139500.], [68300.], [62700.], [91658.], [68200.], [141600.], [46510.], [82229.], [67800.], [173900.], [109300.], [57200.], [57100.], [112600.], [62150.], [64900.], [30000.], [47000.], [62300.], [95730.], [84100.], [83300.], [46100.], [29500.], [57400.], [37700.], [48000.], [63655.], [41900.], [32000.], [64500.], [58600.], [56700.], [94750.], [66700.], [76400.], [34000.]]))

Took 12 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:48:26 PM.

```
%python
```

FINISHED