In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from IPython.display import display
import datetime
import time
import math
import warnings
warnings.filterwarnings("ignore")
import glob
```

In [2]:
```
def read_label():
    label = {}
    for i in range(1, 7):
        hi = 'house_{}/labels.dat'.format(i)
        label[i] = {}
        with open(hi) as f:
            for line in f:
                splitted_line = line.split(' ')

                label[i][int(splitted_line[0])] = splitted_line[1].strip()
+ '_' + splitted_line[0]
    return label
labels = read_label()
for i in range(1,3):
    print('House {}: '.format(i), labels[i], '\n')
```

```
House 1:  {1: 'mains_1', 2: 'mains_2', 3: 'oven_3', 4: 'oven_4', 5: 'refrig
erator_5', 6: 'dishwaser_6', 7: 'kitchen_outlets_7', 8: 'kitchen_outlets_8'
, 9: 'lighting_9', 10: 'washer_dryer_10', 11: 'microwave_11', 12: 'bathroom
_gfi_12', 13: 'electric_heat_13', 14: 'stove_14', 15: 'kitchen_outlets_15',
16: 'kitchen_outlets_16', 17: 'lighting_17', 18: 'lighting_18', 19: 'washer
_dryer_19', 20: 'washer_dryer_20'}

House 2:  {1: 'mains_1', 2: 'mains_2', 3: 'kitchen_outlets_3', 4: 'lighting
_4', 5: 'stove_5', 6: 'microwave_6', 7: 'washer_dryer_7', 8: 'kitchen_outle
ts_8', 9: 'refrigerator_9', 10: 'dishwaser_10', 11: 'disposal_11'}
```

In [3]:
```
def read_merge_data(house):
    path = 'house_{}/'.format(house)
    file = path + 'channel_1.dat'
    df = pd.read_table(file, sep = ' ', names = ['unix_time', labels[house
][1]],
                                       dtype = {'unix_time': 'int64', labels
[house][1]:'float64'})

    num_apps = len(glob.glob(path + 'channel*'))
    for i in range(2, num_apps + 1):
        file = path + 'channel_{}.dat'.format(i)
        data = pd.read_table(file, sep = ' ', names = ['unix_time', labels
```

```
[house][i]],
                                            dtype = {'unix_time': 'int64', labels
[house][i]:'float64'})
        df = pd.merge(df, data, how = 'inner', on = 'unix_time')
    df['timestamp'] = df['unix_time'].astype("datetime64[s]")
    df = df.set_index(df['timestamp'].values)
    df.drop(['unix_time','timestamp'], axis=1, inplace=True)
    #df.drop(['unix_time'], axis=1, inplace=True)

    return df

df = read_merge_data(1)
```

In [4]:
```
df.head()
```

Out[4]:

	mains_1	mains_2	oven_3	oven_4	refrigerator_5	dishwaser_6	kitchen_outlets_7
2011-04-18 13:22:13	222.20	118.83	0.0	0.0	6.0	0.0	34.0
2011-04-18 13:22:16	223.17	119.19	0.0	0.0	6.0	0.0	34.0
2011-04-18 13:22:20	223.60	118.92	0.0	0.0	6.0	0.0	34.0
2011-04-18 13:22:23	222.91	119.16	0.0	0.0	6.0	1.0	35.0
2011-04-18 13:22:26	222.94	118.83	0.0	0.0	6.0	0.0	34.0

In [5]:
```
df.columns
```

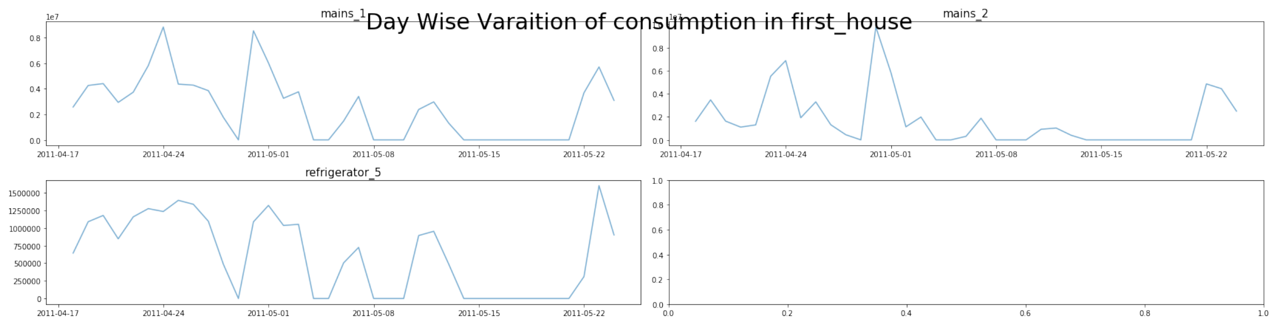
Out[5]:
```
Index(['mains_1', 'mains_2', 'oven_3', 'oven_4', 'refrigerator_5',
       'dishwaser_6', 'kitchen_outlets_7', 'kitchen_outlets_8', 'lighting_9
',
       'washer_dryer_10', 'microwave_11', 'bathroom_gfi_12',
       'electric_heat_13', 'stove_14', 'kitchen_outlets_15',
       'kitchen_outlets_16', 'lighting_17', 'lighting_18', 'washer_dryer_19
',
       'washer_dryer_20'],
      dtype='object')
```

In [6]:
```
import pandas as pd
#df.set_index('date').groupby('name')["ext price"].resample("M").sum()
#df.set_index('date').resample('M')["ext price"].sum()

result = df.resample('3s')["mains_1","mains_2","refrigerator_5"].sum()
print(result)
```

	mains_1	mains_2	refrigerator_5
2011-04-18 13:22:12	222.20	118.83	6.0
2011-04-18 13:22:15	223.17	119.19	6.0
2011-04-18 13:22:18	223.60	118.92	6.0
2011-04-18 13:22:21	222.91	119.16	6.0
2011-04-18 13:22:24	222.94	118.83	6.0
2011-04-18 13:22:27	0.00	0.00	0.0
2011-04-18 13:22:30	222.90	118.76	6.0
2011-04-18 13:22:33	222.96	118.88	6.0
2011-04-18 13:22:36	222.54	118.36	6.0
2011-04-18 13:22:39	226.03	119.17	6.0
2011-04-18 13:22:42	222.96	119.03	6.0
2011-04-18 13:22:45	223.52	118.82	6.0
2011-04-18 13:22:48	0.00	0.00	0.0
2011-04-18 13:22:51	227.29	119.17	6.0
2011-04-18 13:22:54	221.72	118.61	6.0
2011-04-18 13:22:57	226.48	119.13	6.0
2011-04-18 13:23:00	226.38	118.91	6.0
2011-04-18 13:23:03	226.51	118.83	6.0
2011-04-18 13:23:06	225.22	118.76	6.0
2011-04-18 13:23:09	0.00	0.00	0.0
2011-04-18 13:23:12	0.00	0.00	0.0
2011-04-18 13:23:15	0.00	0.00	0.0
2011-04-18 13:23:18	0.00	0.00	0.0
2011-04-18 13:23:21	226.37	118.60	6.0
2011-04-18 13:23:24	225.66	118.98	6.0
2011-04-18 13:23:27	224.11	118.96	6.0
2011-04-18 13:23:30	0.00	0.00	0.0
2011-04-18 13:23:33	225.72	118.74	6.0
2011-04-18 13:23:36	227.74	119.11	6.0
2011-04-18 13:23:39	227.23	119.40	6.0
...	...	...	...
2011-05-24 19:55:06	0.00	0.00	0.0
2011-05-24 19:55:09	0.00	0.00	0.0
2011-05-24 19:55:12	0.00	0.00	0.0
2011-05-24 19:55:15	0.00	0.00	0.0
2011-05-24 19:55:18	0.00	0.00	0.0
2011-05-24 19:55:21	0.00	0.00	0.0
2011-05-24 19:55:24	0.00	0.00	0.0
2011-05-24 19:55:27	0.00	0.00	0.0
2011-05-24 19:55:30	0.00	0.00	0.0
2011-05-24 19:55:33	0.00	0.00	0.0
2011-05-24 19:55:36	238.84	38.68	189.0
2011-05-24 19:55:39	0.00	0.00	0.0
2011-05-24 19:55:42	235.99	38.45	187.0
2011-05-24 19:55:45	236.41	38.65	187.0
2011-05-24 19:55:48	236.58	38.48	189.0
2011-05-24 19:55:51	239.92	38.57	190.0
2011-05-24 19:55:54	235.61	38.30	188.0
2011-05-24 19:55:57	235.53	38.62	186.0
2011-05-24 19:56:00	0.00	0.00	0.0
2011-05-24 19:56:03	234.72	38.77	187.0
2011-05-24 19:56:06	234.47	38.67	188.0
2011-05-24 19:56:09	235.47	38.66	188.0
2011-05-24 19:56:12	235.42	38.51	190.0
2011-05-24 19:56:15	235.33	38.62	188.0

```
2011-05-24 19:56:18   235.73    38.65            186.0
2011-05-24 19:56:21   235.03    38.66            187.0
2011-05-24 19:56:24     0.00     0.00              0.0
2011-05-24 19:56:27   235.46    38.61            190.0
2011-05-24 19:56:30   235.98    38.77            189.0
2011-05-24 19:56:33   235.29    38.83            186.0

[1044688 rows x 3 columns]
```

In [9]:
```python
def plot_df(df, title):
    apps = df.columns.values
    num_apps = len(apps)
    fig, axes = plt.subplots((num_apps+1)//2,2, figsize=(24, num_apps*2) )
    for i, key in enumerate(apps):
        axes.flat[i].plot(df[key], alpha = 0.6)
        axes.flat[i].set_title(key, fontsize = '15')
    plt.suptitle(title, fontsize = '30')
    fig.tight_layout()
    fig.subplots_adjust(top=0.95)

plot_df(result,'Day Wise Varaition of consumption in first_house')
```



In [7]:
```python
result.head()

result['mains_diff']=result['mains_1']-result['mains_2']
```

In [9]:
```python
result['avg_mains']=(result['mains_1']+result['mains_2'])//2
```

In [10]:
```python
result.head()

#result.shape
```

Out[10]:

	mains_1	mains_2	refrigerator_5	mains_diff	avg_mains
2011-04-18 13:22:12	222.20	118.83	6.0	103.37	170.0
2011-04-18 13:22:15	223.17	119.19	6.0	103.98	171.0
2011-04-18 13:22:18	223.60	118.92	6.0	104.68	171.0
2011-04-18 13:22:21	222.91	119.16	6.0	103.75	171.0
2011-04-18 13:22:24	222.94	118.83	6.0	104.11	170.0

Now we will try to predict refrigerator_5 consumption based

on the mains reading

In [11]:
```
X=result[['mains_1','mains_2','mains_diff','avg_mains']]

Y= result['refrigerator_5']
```

In [12]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler( feature_range=(0, 1))
X_st=scaler.fit_transform(X)
```

In [13]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_st, Y, test_size=0.3
3, random_state=42)
```

Model:1 Linear Regression

In [28]:
```
from sklearn.linear_model import LinearRegression
reg = LinearRegression().fit(X_train, y_train)
```

In [30]:
```
y_pred=reg.predict(X_test)
```

In [33]:
```
from sklearn.metrics import mean_squared_error
mse=mean_squared_error(y_test, y_pred)

rmse=np.sqrt(mse)

print(rmse)
```

```
53.973326829208844
```

In [17]:
```
print(X_train.shape[1])
```

```
4
```

Model:2 Deep Learning

In [18]:
```
from keras.layers.core import Dense, Activation, Dropout
from keras.layers.recurrent import LSTM
from keras.models import Sequential
from keras.callbacks import ModelCheckpoint
from keras.models import load_model
from keras.optimizers import Adam
from keras.regularizers import l2
n_cols=X_train.shape[1]

def build_fc_model(layers):
    fc_model = Sequential()
    for i in range(len(layers)-1):
        fc_model.add( Dense(input_dim=n_cols, output_dim= layers[i+1]))
        fc_model.add( Dropout(0.5) )
        if i < (len(layers) - 2):
```

```
            fc_model.add( Activation('relu') )
    fc_model.summary()
    return fc_model
fc_model_1 = build_fc_model([2, 256, 512, 1024, 1])
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_5 (Dense)              (None, 256)               1280      
_________________________________________________________________
dropout_5 (Dropout)          (None, 256)               0         
_________________________________________________________________
activation_4 (Activation)    (None, 256)               0         
_________________________________________________________________
dense_6 (Dense)              (None, 512)               131584    
_________________________________________________________________
dropout_6 (Dropout)          (None, 512)               0         
_________________________________________________________________
activation_5 (Activation)    (None, 512)               0         
_________________________________________________________________
dense_7 (Dense)              (None, 1024)              525312    
_________________________________________________________________
dropout_7 (Dropout)          (None, 1024)              0         
_________________________________________________________________
activation_6 (Activation)    (None, 1024)              0         
_________________________________________________________________
dense_8 (Dense)              (None, 1)                 1025      
_________________________________________________________________
dropout_8 (Dropout)          (None, 1)                 0         
=================================================================
Total params: 659,201
Trainable params: 659,201
Non-trainable params: 0
_________________________________________________________________
```

In [23]:

```
adam = Adam(lr = 1e-5)
fc_model_1.compile(loss='mean_squared_error', optimizer=adam)
start = time.time()
checkpointer = ModelCheckpoint(filepath="fc_refrig_h1_2.hdf5", verbose=0,
save_best_only=True)
hist_fc_1 = fc_model_1.fit( X_train, y_train,
                    batch_size=512, verbose=1, nb_epoch=100,
                    validation_data=(X_test,y_test), callbacks=[checkpointe
r])
print('Finish trainning. Time: ', time.time() - start)
```

```
Train on 699940 samples, validate on 344748 samples
Epoch 1/100
699940/699940 [==============================] - 76s 108us/step - loss: 407
1.5442 - val_loss: 3893.2855
Epoch 2/100
699940/699940 [==============================] - 81s 116us/step - loss: 394
9.2390 - val_loss: 3801.3066
Epoch 3/100
699940/699940 [==============================] - 85s 122us/step - loss: 386
9.5928 - val_loss: 3678.3173
Epoch 4/100
```

```
699940/699940 [==============================] - 81s 116us/step - loss: 378
5.8866 - val_loss: 3566.9659
Epoch 5/100
699940/699940 [==============================] - 82s 118us/step - loss: 372
4.2867 - val_loss: 3473.0119
Epoch 6/100
699940/699940 [==============================] - 85s 121us/step - loss: 365
2.6071 - val_loss: 3368.6154
Epoch 7/100
699940/699940 [==============================] - 81s 116us/step - loss: 356
9.6364 - val_loss: 3247.8291
Epoch 8/100
699940/699940 [==============================] - 82s 117us/step - loss: 347
7.2529 - val_loss: 3096.3723
Epoch 9/100
699940/699940 [==============================] - 82s 117us/step - loss: 337
0.8784 - val_loss: 2941.7134
Epoch 10/100
699940/699940 [==============================] - 82s 117us/step - loss: 328
8.0707 - val_loss: 2812.9645
Epoch 11/100
699940/699940 [==============================] - 85s 121us/step - loss: 321
4.6466 - val_loss: 2719.9226
Epoch 12/100
699940/699940 [==============================] - 83s 119us/step - loss: 313
9.4114 - val_loss: 2604.7593
Epoch 13/100
699940/699940 [==============================] - 82s 118us/step - loss: 307
8.5177 - val_loss: 2511.0739
Epoch 14/100
699940/699940 [==============================] - 84s 120us/step - loss: 300
8.4021 - val_loss: 2413.4268
Epoch 15/100
699940/699940 [==============================] - 84s 120us/step - loss: 292
6.8641 - val_loss: 2314.1780
Epoch 16/100
699940/699940 [==============================] - 85s 121us/step - loss: 287
4.1730 - val_loss: 2245.7859
Epoch 17/100
699940/699940 [==============================] - 85s 122us/step - loss: 283
2.5868 - val_loss: 2189.6559
Epoch 18/100
699940/699940 [==============================] - 86s 123us/step - loss: 277
3.5548 - val_loss: 2106.9914
Epoch 19/100
699940/699940 [==============================] - 87s 124us/step - loss: 274
2.0374 - val_loss: 2044.1447
Epoch 20/100
699940/699940 [==============================] - 86s 124us/step - loss: 270
6.7911 - val_loss: 1996.6538
Epoch 21/100
699940/699940 [==============================] - 90s 128us/step - loss: 267
0.2362 - val_loss: 1935.5529
Epoch 22/100
699940/699940 [==============================] - 84s 120us/step - loss: 264
4.9358 - val_loss: 1903.2379
Epoch 23/100
```

```
699940/699940 [==============================] - 85s 121us/step - loss: 260
6.6173 - val_loss: 1843.4800
Epoch 24/100
699940/699940 [==============================] - 84s 120us/step - loss: 258
7.7255 - val_loss: 1825.8757
Epoch 25/100
699940/699940 [==============================] - 85s 121us/step - loss: 256
9.5455 - val_loss: 1796.4674
Epoch 26/100
699940/699940 [==============================] - 86s 123us/step - loss: 254
8.2282 - val_loss: 1752.7804
Epoch 27/100
699940/699940 [==============================] - 86s 122us/step - loss: 253
3.0820 - val_loss: 1758.2846
Epoch 28/100
699940/699940 [==============================] - 86s 123us/step - loss: 252
2.7022 - val_loss: 1734.0942
Epoch 29/100
699940/699940 [==============================] - 86s 123us/step - loss: 251
1.9425 - val_loss: 1714.0133
Epoch 30/100
699940/699940 [==============================] - 86s 123us/step - loss: 252
1.7080 - val_loss: 1701.2642
Epoch 31/100
699940/699940 [==============================] - 87s 124us/step - loss: 249
6.2927 - val_loss: 1686.7380
Epoch 32/100
699940/699940 [==============================] - 87s 124us/step - loss: 247
6.1701 - val_loss: 1684.3418
Epoch 33/100
699940/699940 [==============================] - 87s 124us/step - loss: 247
9.5262 - val_loss: 1667.5427
Epoch 34/100
699940/699940 [==============================] - 86s 123us/step - loss: 248
6.2097 - val_loss: 1656.1341
Epoch 35/100
699940/699940 [==============================] - 87s 125us/step - loss: 247
9.2007 - val_loss: 1655.2603
Epoch 36/100
699940/699940 [==============================] - 87s 124us/step - loss: 246
3.0739 - val_loss: 1633.0744
Epoch 37/100
699940/699940 [==============================] - 87s 124us/step - loss: 247
4.2199 - val_loss: 1626.8862
Epoch 38/100
699940/699940 [==============================] - 87s 124us/step - loss: 246
2.8574 - val_loss: 1634.6966
Epoch 39/100
699940/699940 [==============================] - 87s 124us/step - loss: 244
8.3115 - val_loss: 1625.5939
Epoch 40/100
699940/699940 [==============================] - 87s 124us/step - loss: 244
7.4087 - val_loss: 1610.9996
Epoch 41/100
699940/699940 [==============================] - 87s 124us/step - loss: 246
6.8279 - val_loss: 1616.2224
Epoch 42/100
```

```
699940/699940 [==============================] - 87s 124us/step - loss: 245
3.4005 - val_loss: 1613.0692
Epoch 43/100
699940/699940 [==============================] - 87s 125us/step - loss: 245
0.5367 - val_loss: 1623.7817
Epoch 44/100
699940/699940 [==============================] - 88s 126us/step - loss: 244
9.7905 - val_loss: 1596.4063
Epoch 45/100
699940/699940 [==============================] - 87s 125us/step - loss: 244
5.5165 - val_loss: 1613.6038
Epoch 46/100
699940/699940 [==============================] - 88s 125us/step - loss: 244
0.8851 - val_loss: 1609.2829
Epoch 47/100
699940/699940 [==============================] - 87s 125us/step - loss: 242
5.5162 - val_loss: 1587.8973
Epoch 48/100
699940/699940 [==============================] - 87s 125us/step - loss: 244
2.1335 - val_loss: 1609.9932
Epoch 49/100
699940/699940 [==============================] - 87s 125us/step - loss: 242
8.8599 - val_loss: 1602.6614
Epoch 50/100
699940/699940 [==============================] - 87s 124us/step - loss: 244
1.8542 - val_loss: 1601.8310
Epoch 51/100
699940/699940 [==============================] - 87s 124us/step - loss: 243
0.6828 - val_loss: 1591.1186
Epoch 52/100
699940/699940 [==============================] - 86s 123us/step - loss: 243
3.4494 - val_loss: 1599.6700
Epoch 53/100
699940/699940 [==============================] - 82s 117us/step - loss: 243
0.6712 - val_loss: 1604.5801
Epoch 54/100
699940/699940 [==============================] - 84s 120us/step - loss: 244
6.7144 - val_loss: 1591.7691
Epoch 55/100
699940/699940 [==============================] - 88s 126us/step - loss: 244
5.4967 - val_loss: 1618.3940
Epoch 56/100
699940/699940 [==============================] - 87s 124us/step - loss: 243
6.6165 - val_loss: 1592.1984
Epoch 57/100
699940/699940 [==============================] - 87s 124us/step - loss: 242
0.5103 - val_loss: 1585.4633
Epoch 58/100
699940/699940 [==============================] - 80s 114us/step - loss: 244
8.7338 - val_loss: 1588.0649
Epoch 59/100
699940/699940 [==============================] - 73s 105us/step - loss: 241
9.1165 - val_loss: 1585.6298
Epoch 60/100
699940/699940 [==============================] - 79s 112us/step - loss: 242
6.1871 - val_loss: 1577.7181
Epoch 61/100
```

```
699940/699940 [==============================] - 84s 120us/step - loss: 243
0.4758 - val_loss: 1579.8639
Epoch 62/100
699940/699940 [==============================] - 84s 119us/step - loss: 242
2.5961 - val_loss: 1581.3087
Epoch 63/100
699940/699940 [==============================] - 84s 119us/step - loss: 241
8.1242 - val_loss: 1571.7076
Epoch 64/100
699940/699940 [==============================] - 85s 122us/step - loss: 242
7.6965 - val_loss: 1588.2181
Epoch 65/100
699940/699940 [==============================] - 86s 123us/step - loss: 244
2.3682 - val_loss: 1584.0398
Epoch 66/100
699940/699940 [==============================] - 85s 121us/step - loss: 241
8.2854 - val_loss: 1578.3202
Epoch 67/100
699940/699940 [==============================] - 96s 137us/step - loss: 243
4.1579 - val_loss: 1573.1197
Epoch 68/100
699940/699940 [==============================] - 119s 170us/step - loss: 24
32.2066 - val_loss: 1590.5096
Epoch 69/100
699940/699940 [==============================] - 112s 160us/step - loss: 24
27.0721 - val_loss: 1580.0915
Epoch 70/100
699940/699940 [==============================] - 118s 169us/step - loss: 24
16.7627 - val_loss: 1568.2240
Epoch 71/100
699940/699940 [==============================] - 117s 167us/step - loss: 24
29.4972 - val_loss: 1579.0445
Epoch 72/100
699940/699940 [==============================] - 115s 164us/step - loss: 24
12.6741 - val_loss: 1576.0793
Epoch 73/100
699940/699940 [==============================] - 114s 163us/step - loss: 24
30.9881 - val_loss: 1581.5877
Epoch 74/100
699940/699940 [==============================] - 113s 161us/step - loss: 24
17.9264 - val_loss: 1569.6827
Epoch 75/100
699940/699940 [==============================] - 117s 168us/step - loss: 24
25.6259 - val_loss: 1576.9162
Epoch 76/100
699940/699940 [==============================] - 116s 166us/step - loss: 24
25.7218 - val_loss: 1577.7242
Epoch 77/100
699940/699940 [==============================] - 112s 161us/step - loss: 24
24.4124 - val_loss: 1576.6372
Epoch 78/100
699940/699940 [==============================] - 109s 156us/step - loss: 24
14.7631 - val_loss: 1570.1448
Epoch 79/100
699940/699940 [==============================] - 109s 156us/step - loss: 24
13.2207 - val_loss: 1566.3398
Epoch 80/100
```

```
699940/699940 [==============================] - 109s 156us/step - loss: 24
14.7984 - val_loss: 1584.0967
Epoch 81/100
699940/699940 [==============================] - 114s 163us/step - loss: 24
23.2499 - val_loss: 1573.1327
Epoch 82/100
699940/699940 [==============================] - 124s 177us/step - loss: 24
21.6752 - val_loss: 1566.4248
Epoch 83/100
699940/699940 [==============================] - 114s 164us/step - loss: 24
16.1990 - val_loss: 1572.6493
Epoch 84/100
699940/699940 [==============================] - 115s 164us/step - loss: 24
16.5290 - val_loss: 1570.7529
Epoch 85/100
699940/699940 [==============================] - 112s 160us/step - loss: 24
07.5876 - val_loss: 1573.2998
Epoch 86/100
699940/699940 [==============================] - 104s 148us/step - loss: 24
04.7537 - val_loss: 1564.3907
Epoch 87/100
699940/699940 [==============================] - 110s 157us/step - loss: 24
04.4098 - val_loss: 1567.3314
Epoch 88/100
699940/699940 [==============================] - 112s 160us/step - loss: 24
11.4393 - val_loss: 1574.8065
Epoch 89/100
699940/699940 [==============================] - 110s 157us/step - loss: 24
15.7119 - val_loss: 1558.0244
Epoch 90/100
699940/699940 [==============================] - 112s 160us/step - loss: 24
14.1178 - val_loss: 1562.4146
Epoch 91/100
699940/699940 [==============================] - 111s 159us/step - loss: 24
23.8833 - val_loss: 1551.3492
Epoch 92/100
699940/699940 [==============================] - 109s 156us/step - loss: 24
04.2666 - val_loss: 1567.4547
Epoch 93/100
699940/699940 [==============================] - 107s 153us/step - loss: 24
15.9326 - val_loss: 1558.0215
Epoch 94/100
699940/699940 [==============================] - 112s 161us/step - loss: 24
06.4472 - val_loss: 1559.6090
Epoch 95/100
699940/699940 [==============================] - 112s 160us/step - loss: 24
14.5978 - val_loss: 1565.1046
Epoch 96/100
699940/699940 [==============================] - 111s 158us/step - loss: 24
11.9025 - val_loss: 1556.2620
Epoch 97/100
699940/699940 [==============================] - 102s 146us/step - loss: 24
23.7008 - val_loss: 1552.1197
Epoch 98/100
699940/699940 [==============================] - 90s 128us/step - loss: 241
1.3681 - val_loss: 1550.0738
Epoch 99/100
```

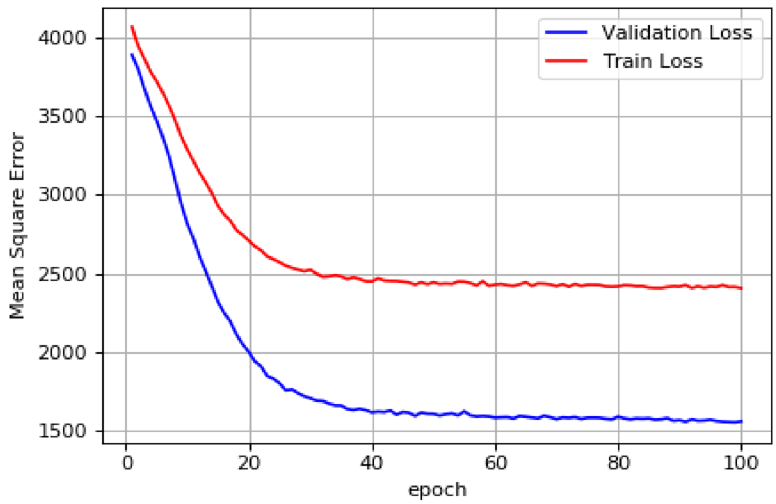
```
699940/699940 [==============================] - 84s 121us/step - loss: 241
1.7385 - val_loss: 1548.4983
Epoch 100/100
699940/699940 [==============================] - 83s 118us/step - loss: 240
3.5316 - val_loss: 1554.2699
Finish trainning. Time:  9334.332983493805
```

In [24]:

```
%matplotlib notebook
import matplotlib.pyplot as plt
import numpy as np
import time
def plot_dynamic(x, vy, ty, ax, colors=['b']):
    ax.plot(x, vy, 'b', label="Validation Loss")
    ax.plot(x, ty, 'r', label="Train Loss")
    plt.legend()
    plt.grid()
    fig.canvas.draw()
```

In [27]:

```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Mean Square Error')

# list of epoch numbers
x = list(range(1,101))

# print(history.history.keys())
# dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
# history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epochs=
nb_epoch, verbose=1, validation_data=(X_test, Y_test))

# we will get val_loss and val_acc only when you pass the paramter validati
on_data
# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
# for each key in histrory.histrory we will have a list of length equal to
number of epochs

vy = hist_fc_1.history['val_loss']
ty =hist_fc_1.history['loss']
plot_dynamic(x, vy, ty, ax)
```


Validation Loss
Train Loss
Mean Square Error
epoch
4000
3500
3000
2500
2000
1500
0
20
40
60
80
100