1 of 43

https://mail.google.com/mail/u/0/images/cleardot.gif

{

 "cells": [

  {

   "cell\_type": "markdown",

   "metadata": {},

   "source": [

    "## Load libraries"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 1,

   "metadata": {},

   "outputs": [

    {

     "name": "stderr",

     "output\_type": "stream",

     "text": [

      "Using TensorFlow backend.\n"

     ]

    }

   ],

   "source": [

    "import numpy as np\n",

    "import pandas as pd\n",

    "import matplotlib.pyplot as plt\n",

    "import math\n",

    "\n",

    "from sklearn.preprocessing import MinMaxScaler\n",

    "from sklearn.metrics import mean\_squared\_error\n",

    "\n",

    "from keras.models import Sequential\n",

    "from keras.layers import Dense\n",

    "from keras.layers import LSTM"

   ]

  },

  {

   "cell\_type": "markdown",

   "metadata": {},

   "source": [

    "## Load the dataset"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 2,

   "metadata": {

    "collapsed": true

   },

   "outputs": [],

   "source": [

    "data = pd.read\_csv(\"bitcoin\_ticker.csv\")"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 3,

   "metadata": {},

   "outputs": [

    {

     "data": {

      "text/html": [

       "<div>\n",

       "<style>\n",

       "    .dataframe thead tr:only-child th {\n",

       "        text-align: right;\n",

       "    }\n",

       "\n",

       "    .dataframe thead th {\n",

       "        text-align: left;\n",

       "    }\n",

       "\n",

       "    .dataframe tbody tr th {\n",

       "        vertical-align: top;\n",

       "    }\n",

       "</style>\n",

       "<table border=\"1\" class=\"dataframe\">\n",

       "  <thead>\n",

       "    <tr style=\"text-align: right;\">\n",

       "      <th></th>\n",

       "      <th>date\_id</th>\n",

       "      <th>datetime\_id</th>\n",

       "      <th>market</th>\n",

       "      <th>rpt\_key</th>\n",

       "      <th>last</th>\n",

       "      <th>diff\_24h</th>\n",

       "      <th>diff\_per\_24h</th>\n",

       "      <th>bid</th>\n",

       "      <th>ask</th>\n",

       "      <th>low</th>\n",

       "      <th>high</th>\n",

       "      <th>volume</th>\n",

       "      <th>created\_at</th>\n",

       "      <th>updated\_at</th>\n",

       "    </tr>\n",

       "  </thead>\n",

       "  <tbody>\n",

       "    <tr>\n",

       "      <th>0</th>\n",

       "      <td>2017-05-31</td>\n",

       "      <td>2017-06-01 00:00:00</td>\n",

       "      <td>bitstamp</td>\n",

       "      <td>btc\_eur</td>\n",

       "      <td>1996.72</td>\n",

       "      <td>2029.99</td>\n",

       "      <td>-1.638924</td>\n",

       "      <td>2005.50</td>\n",

       "      <td>2005.56</td>\n",

       "      <td>1950.00</td>\n",

       "      <td>2063.73</td>\n",

       "      <td>2314.500750</td>\n",

       "      <td>2017-05-31 14:59:36</td>\n",

       "      <td>2017-05-31 14:59:36</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>1</th>\n",

       "      <td>2017-05-31</td>\n",

       "      <td>2017-06-01 00:00:00</td>\n",

       "      <td>bitflyer</td>\n",

       "      <td>btc\_jpy</td>\n",

       "      <td>267098.00</td>\n",

       "      <td>269649.00</td>\n",

       "      <td>-0.946045</td>\n",

       "      <td>267124.00</td>\n",

       "      <td>267267.00</td>\n",

       "      <td>267124.00</td>\n",

       "      <td>267267.00</td>\n",

       "      <td>70922.880112</td>\n",

       "      <td>2017-05-31 14:59:36</td>\n",

       "      <td>2017-05-31 14:59:36</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>2</th>\n",

       "      <td>2017-05-31</td>\n",

       "      <td>2017-06-01 00:00:00</td>\n",

       "      <td>korbit</td>\n",

       "      <td>btc\_krw</td>\n",

       "      <td>3003500.00</td>\n",

       "      <td>3140000.00</td>\n",

       "      <td>-4.347134</td>\n",

       "      <td>3003500.00</td>\n",

       "      <td>3004000.00</td>\n",

       "      <td>3002000.00</td>\n",

       "      <td>3209500.00</td>\n",

       "      <td>6109.752872</td>\n",

       "      <td>2017-05-31 14:59:36</td>\n",

       "      <td>2017-05-31 14:59:36</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>3</th>\n",

       "      <td>2017-05-31</td>\n",

       "      <td>2017-06-01 00:00:00</td>\n",

       "      <td>bitstamp</td>\n",

       "      <td>btc\_usd</td>\n",

       "      <td>2237.40</td>\n",

       "      <td>2239.37</td>\n",

       "      <td>-0.087971</td>\n",

       "      <td>2233.09</td>\n",

       "      <td>2237.40</td>\n",

       "      <td>2154.28</td>\n",

       "      <td>2293.46</td>\n",

       "      <td>13681.282017</td>\n",

       "      <td>2017-05-31 14:59:36</td>\n",

       "      <td>2017-05-31 14:59:36</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>4</th>\n",

       "      <td>2017-05-31</td>\n",

       "      <td>2017-06-01 00:00:00</td>\n",

       "      <td>okcoin</td>\n",

       "      <td>btc\_usd</td>\n",

       "      <td>2318.82</td>\n",

       "      <td>2228.70</td>\n",

       "      <td>4.043613</td>\n",

       "      <td>2319.40</td>\n",

       "      <td>2319.99</td>\n",

       "      <td>2129.78</td>\n",

       "      <td>2318.82</td>\n",

       "      <td>4241.641516</td>\n",

       "      <td>2017-05-31 14:59:36</td>\n",

       "      <td>2017-05-31 14:59:36</td>\n",

       "    </tr>\n",

       "  </tbody>\n",

       "</table>\n",

       "</div>"

      ],

      "text/plain": [

       "      date\_id          datetime\_id    market  rpt\_key        last    diff\_24h  \\\n",

       "0  2017-05-31  2017-06-01 00:00:00  bitstamp  btc\_eur     1996.72     2029.99   \n",

       "1  2017-05-31  2017-06-01 00:00:00  bitflyer  btc\_jpy   267098.00   269649.00   \n",

       "2  2017-05-31  2017-06-01 00:00:00    korbit  btc\_krw  3003500.00  3140000.00   \n",

       "3  2017-05-31  2017-06-01 00:00:00  bitstamp  btc\_usd     2237.40     2239.37   \n",

       "4  2017-05-31  2017-06-01 00:00:00    okcoin  btc\_usd     2318.82     2228.70   \n",

       "\n",

       "   diff\_per\_24h         bid         ask         low        high        volume  \\\n",

       "0     -1.638924     2005.50     2005.56     1950.00     2063.73   2314.500750   \n",

       "1     -0.946045   267124.00   267267.00   267124.00   267267.00  70922.880112   \n",

       "2     -4.347134  3003500.00  3004000.00  3002000.00  3209500.00   6109.752872   \n",

       "3     -0.087971     2233.09     2237.40     2154.28     2293.46  13681.282017   \n",

       "4      4.043613     2319.40     2319.99     2129.78     2318.82   4241.641516   \n",

       "\n",

       "            created\_at           updated\_at  \n",

       "0  2017-05-31 14:59:36  2017-05-31 14:59:36  \n",

       "1  2017-05-31 14:59:36  2017-05-31 14:59:36  \n",

       "2  2017-05-31 14:59:36  2017-05-31 14:59:36  \n",

       "3  2017-05-31 14:59:36  2017-05-31 14:59:36  \n",

       "4  2017-05-31 14:59:36  2017-05-31 14:59:36  "

      ]

     },

     "execution\_count": 3,

     "metadata": {},

     "output\_type": "execute\_result"

    }

   ],

   "source": [

    "data.head()"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 4,

   "metadata": {},

   "outputs": [

    {

     "data": {

      "text/plain": [

       "btc\_usd       125438\n",

       "fx\_btc\_jpy     62719\n",

       "eth\_krw        62719\n",

       "etc\_krw        62719\n",

       "ltc\_usd        62719\n",

       "btc\_jpy        62719\n",

       "eth\_btc        62719\n",

       "btc\_eur        62719\n",

       "btc\_krw        62719\n",

       "Name: rpt\_key, dtype: int64"

      ]

     },

     "execution\_count": 4,

     "metadata": {},

     "output\_type": "execute\_result"

    }

   ],

   "source": [

    "data['rpt\_key'].value\_counts()"

   ]

  },

  {

   "cell\_type": "markdown",

   "metadata": {},

   "source": [

    "## Subset USD"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 5,

   "metadata": {

    "collapsed": true

   },

   "outputs": [],

   "source": [

    "df = data.loc[(data['rpt\_key'] == 'btc\_usd')]"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 6,

   "metadata": {

    "scrolled": true

   },

   "outputs": [

    {

     "data": {

      "text/html": [

       "<div>\n",

       "<style>\n",

       "    .dataframe thead tr:only-child th {\n",

       "        text-align: right;\n",

       "    }\n",

       "\n",

       "    .dataframe thead th {\n",

       "        text-align: left;\n",

       "    }\n",

       "\n",

       "    .dataframe tbody tr th {\n",

       "        vertical-align: top;\n",

       "    }\n",

       "</style>\n",

       "<table border=\"1\" class=\"dataframe\">\n",

       "  <thead>\n",

       "    <tr style=\"text-align: right;\">\n",

       "      <th></th>\n",

       "      <th>date\_id</th>\n",

       "      <th>datetime\_id</th>\n",

       "      <th>market</th>\n",

       "      <th>rpt\_key</th>\n",

       "      <th>last</th>\n",

       "      <th>diff\_24h</th>\n",

       "      <th>diff\_per\_24h</th>\n",

       "      <th>bid</th>\n",

       "      <th>ask</th>\n",

       "      <th>low</th>\n",

       "      <th>high</th>\n",

       "      <th>volume</th>\n",

       "      <th>created\_at</th>\n",

       "      <th>updated\_at</th>\n",

       "    </tr>\n",

       "  </thead>\n",

       "  <tbody>\n",

       "    <tr>\n",

       "      <th>3</th>\n",

       "      <td>2017-05-31</td>\n",

       "      <td>2017-06-01 00:00:00</td>\n",

       "      <td>bitstamp</td>\n",

       "      <td>btc\_usd</td>\n",

       "      <td>2237.40</td>\n",

       "      <td>2239.37</td>\n",

       "      <td>-0.087971</td>\n",

       "      <td>2233.09</td>\n",

       "      <td>2237.40</td>\n",

       "      <td>2154.28</td>\n",

       "      <td>2293.46</td>\n",

       "      <td>13681.282017</td>\n",

       "      <td>2017-05-31 14:59:36</td>\n",

       "      <td>2017-05-31 14:59:36</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>4</th>\n",

       "      <td>2017-05-31</td>\n",

       "      <td>2017-06-01 00:00:00</td>\n",

       "      <td>okcoin</td>\n",

       "      <td>btc\_usd</td>\n",

       "      <td>2318.82</td>\n",

       "      <td>2228.70</td>\n",

       "      <td>4.043613</td>\n",

       "      <td>2319.40</td>\n",

       "      <td>2319.99</td>\n",

       "      <td>2129.78</td>\n",

       "      <td>2318.82</td>\n",

       "      <td>4241.641516</td>\n",

       "      <td>2017-05-31 14:59:36</td>\n",

       "      <td>2017-05-31 14:59:36</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>15</th>\n",

       "      <td>2017-06-01</td>\n",

       "      <td>2017-06-01 00:01:00</td>\n",

       "      <td>bitstamp</td>\n",

       "      <td>btc\_usd</td>\n",

       "      <td>2248.39</td>\n",

       "      <td>2242.44</td>\n",

       "      <td>0.265336</td>\n",

       "      <td>2247.77</td>\n",

       "      <td>2248.38</td>\n",

       "      <td>2154.28</td>\n",

       "      <td>2293.46</td>\n",

       "      <td>13701.698603</td>\n",

       "      <td>2017-05-31 15:00:36</td>\n",

       "      <td>2017-05-31 15:00:36</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>16</th>\n",

       "      <td>2017-06-01</td>\n",

       "      <td>2017-06-01 00:01:00</td>\n",

       "      <td>okcoin</td>\n",

       "      <td>btc\_usd</td>\n",

       "      <td>2320.42</td>\n",

       "      <td>2228.40</td>\n",

       "      <td>4.129420</td>\n",

       "      <td>2320.99</td>\n",

       "      <td>2321.49</td>\n",

       "      <td>2129.78</td>\n",

       "      <td>2322.00</td>\n",

       "      <td>4260.261516</td>\n",

       "      <td>2017-05-31 15:00:36</td>\n",

       "      <td>2017-05-31 15:00:36</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>23</th>\n",

       "      <td>2017-06-01</td>\n",

       "      <td>2017-06-01 00:02:00</td>\n",

       "      <td>bitstamp</td>\n",

       "      <td>btc\_usd</td>\n",

       "      <td>2248.35</td>\n",

       "      <td>2238.58</td>\n",

       "      <td>0.436437</td>\n",

       "      <td>2248.35</td>\n",

       "      <td>2248.69</td>\n",

       "      <td>2154.28</td>\n",

       "      <td>2293.46</td>\n",

       "      <td>13742.110913</td>\n",

       "      <td>2017-05-31 15:01:36</td>\n",

       "      <td>2017-05-31 15:01:36</td>\n",

       "    </tr>\n",

       "  </tbody>\n",

       "</table>\n",

       "</div>"

      ],

      "text/plain": [

       "       date\_id          datetime\_id    market  rpt\_key     last  diff\_24h  \\\n",

       "3   2017-05-31  2017-06-01 00:00:00  bitstamp  btc\_usd  2237.40   2239.37   \n",

       "4   2017-05-31  2017-06-01 00:00:00    okcoin  btc\_usd  2318.82   2228.70   \n",

       "15  2017-06-01  2017-06-01 00:01:00  bitstamp  btc\_usd  2248.39   2242.44   \n",

       "16  2017-06-01  2017-06-01 00:01:00    okcoin  btc\_usd  2320.42   2228.40   \n",

       "23  2017-06-01  2017-06-01 00:02:00  bitstamp  btc\_usd  2248.35   2238.58   \n",

       "\n",

       "    diff\_per\_24h      bid      ask      low     high        volume  \\\n",

       "3      -0.087971  2233.09  2237.40  2154.28  2293.46  13681.282017   \n",

       "4       4.043613  2319.40  2319.99  2129.78  2318.82   4241.641516   \n",

       "15      0.265336  2247.77  2248.38  2154.28  2293.46  13701.698603   \n",

       "16      4.129420  2320.99  2321.49  2129.78  2322.00   4260.261516   \n",

       "23      0.436437  2248.35  2248.69  2154.28  2293.46  13742.110913   \n",

       "\n",

       "             created\_at           updated\_at  \n",

       "3   2017-05-31 14:59:36  2017-05-31 14:59:36  \n",

       "4   2017-05-31 14:59:36  2017-05-31 14:59:36  \n",

       "15  2017-05-31 15:00:36  2017-05-31 15:00:36  \n",

       "16  2017-05-31 15:00:36  2017-05-31 15:00:36  \n",

       "23  2017-05-31 15:01:36  2017-05-31 15:01:36  "

      ]

     },

     "execution\_count": 6,

     "metadata": {},

     "output\_type": "execute\_result"

    }

   ],

   "source": [

    "df.head()"

   ]

  },

  {

   "cell\_type": "markdown",

   "metadata": {},

   "source": [

    "### Convert datetime\_id to data type and filter dates greater than  2017-06-28 00:00:00"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 7,

   "metadata": {

    "collapsed": true

   },

   "outputs": [],

   "source": [

    "df = df.reset\_index(drop=True)\n",

    "df['datetime'] = pd.to\_datetime(df['datetime\_id'])\n",

    "df = df.loc[df['datetime'] > pd.to\_datetime('2017-06-28 00:00:00')]"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 8,

   "metadata": {

    "collapsed": true

   },

   "outputs": [],

   "source": [

    "df = df[['datetime', 'last', 'diff\_24h', 'diff\_per\_24h', 'bid', 'ask', 'low', 'high', 'volume']]"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 9,

   "metadata": {

    "scrolled": true

   },

   "outputs": [

    {

     "data": {

      "text/html": [

       "<div>\n",

       "<style>\n",

       "    .dataframe thead tr:only-child th {\n",

       "        text-align: right;\n",

       "    }\n",

       "\n",

       "    .dataframe thead th {\n",

       "        text-align: left;\n",

       "    }\n",

       "\n",

       "    .dataframe tbody tr th {\n",

       "        vertical-align: top;\n",

       "    }\n",

       "</style>\n",

       "<table border=\"1\" class=\"dataframe\">\n",

       "  <thead>\n",

       "    <tr style=\"text-align: right;\">\n",

       "      <th></th>\n",

       "      <th>datetime</th>\n",

       "      <th>last</th>\n",

       "      <th>diff\_24h</th>\n",

       "      <th>diff\_per\_24h</th>\n",

       "      <th>bid</th>\n",

       "      <th>ask</th>\n",

       "      <th>low</th>\n",

       "      <th>high</th>\n",

       "      <th>volume</th>\n",

       "    </tr>\n",

       "  </thead>\n",

       "  <tbody>\n",

       "    <tr>\n",

       "      <th>77762</th>\n",

       "      <td>2017-06-28 00:01:00</td>\n",

       "      <td>2344.00</td>\n",

       "      <td>2491.98</td>\n",

       "      <td>-5.938250</td>\n",

       "      <td>2335.01</td>\n",

       "      <td>2343.89</td>\n",

       "      <td>2307.0</td>\n",

       "      <td>2473.19</td>\n",

       "      <td>20719.583592</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>77763</th>\n",

       "      <td>2017-06-28 00:01:00</td>\n",

       "      <td>2499.39</td>\n",

       "      <td>2682.25</td>\n",

       "      <td>-6.817411</td>\n",

       "      <td>2495.00</td>\n",

       "      <td>2499.33</td>\n",

       "      <td>2444.0</td>\n",

       "      <td>2780.62</td>\n",

       "      <td>2265.557866</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>77764</th>\n",

       "      <td>2017-06-28 00:02:00</td>\n",

       "      <td>2337.18</td>\n",

       "      <td>2491.98</td>\n",

       "      <td>-6.211928</td>\n",

       "      <td>2337.18</td>\n",

       "      <td>2340.00</td>\n",

       "      <td>2307.0</td>\n",

       "      <td>2473.19</td>\n",

       "      <td>20732.082581</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>77765</th>\n",

       "      <td>2017-06-28 00:02:00</td>\n",

       "      <td>2492.76</td>\n",

       "      <td>2682.25</td>\n",

       "      <td>-7.064591</td>\n",

       "      <td>2492.76</td>\n",

       "      <td>2495.00</td>\n",

       "      <td>2444.0</td>\n",

       "      <td>2780.62</td>\n",

       "      <td>2262.618866</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>77766</th>\n",

       "      <td>2017-06-28 00:03:00</td>\n",

       "      <td>2335.02</td>\n",

       "      <td>2491.98</td>\n",

       "      <td>-6.298606</td>\n",

       "      <td>2335.01</td>\n",

       "      <td>2335.02</td>\n",

       "      <td>2307.0</td>\n",

       "      <td>2473.19</td>\n",

       "      <td>20665.357191</td>\n",

       "    </tr>\n",

       "  </tbody>\n",

       "</table>\n",

       "</div>"

      ],

      "text/plain": [

       "                 datetime     last  diff\_24h  diff\_per\_24h      bid      ask  \\\n",

       "77762 2017-06-28 00:01:00  2344.00   2491.98     -5.938250  2335.01  2343.89   \n",

       "77763 2017-06-28 00:01:00  2499.39   2682.25     -6.817411  2495.00  2499.33   \n",

       "77764 2017-06-28 00:02:00  2337.18   2491.98     -6.211928  2337.18  2340.00   \n",

       "77765 2017-06-28 00:02:00  2492.76   2682.25     -7.064591  2492.76  2495.00   \n",

       "77766 2017-06-28 00:03:00  2335.02   2491.98     -6.298606  2335.01  2335.02   \n",

       "\n",

       "          low     high        volume  \n",

       "77762  2307.0  2473.19  20719.583592  \n",

       "77763  2444.0  2780.62   2265.557866  \n",

       "77764  2307.0  2473.19  20732.082581  \n",

       "77765  2444.0  2780.62   2262.618866  \n",

       "77766  2307.0  2473.19  20665.357191  "

      ]

     },

     "execution\_count": 9,

     "metadata": {},

     "output\_type": "execute\_result"

    }

   ],

   "source": [

    "df.head()"

   ]

  },

  {

   "cell\_type": "markdown",

   "metadata": {},

   "source": [

    "### we require only the last value, so we subset that and convert it to numpy array"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 10,

   "metadata": {

    "collapsed": true

   },

   "outputs": [],

   "source": [

    "df = df[['last']]"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 11,

   "metadata": {

    "collapsed": true

   },

   "outputs": [],

   "source": [

    "dataset = df.values\n",

    "dataset = dataset.astype('float32')"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 12,

   "metadata": {},

   "outputs": [

    {

     "data": {

      "text/plain": [

       "array([[ 2344.        ],\n",

       "       [ 2499.38989258],\n",

       "       [ 2337.17993164],\n",

       "       ..., \n",

       "       [ 2394.0300293 ],\n",

       "       [ 2320.4699707 ],\n",

       "       [ 2394.0300293 ]], dtype=float32)"

      ]

     },

     "execution\_count": 12,

     "metadata": {},

     "output\_type": "execute\_result"

    }

   ],

   "source": [

    "dataset"

   ]

  },

  {

   "cell\_type": "markdown",

   "metadata": {},

   "source": [

    "Neural networks are sensitive to input data, especiallly when we are using activation functions like sigmoid or tanh activation functions are used. ISo we rescale our data to the range of 0-to-1, using MinMaxScaler"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 13,

   "metadata": {

    "collapsed": true

   },

   "outputs": [],

   "source": [

    "scaler = MinMaxScaler(feature\_range=(0, 1))\n",

    "dataset = scaler.fit\_transform(dataset)"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 14,

   "metadata": {},

   "outputs": [

    {

     "data": {

      "text/plain": [

       "array([[ 0.1997695 ],\n",

       "       [ 0.49828053],\n",

       "       [ 0.18666792],\n",

       "       ..., \n",

       "       [ 0.29587936],\n",

       "       [ 0.15456724],\n",

       "       [ 0.29587936]], dtype=float32)"

      ]

     },

     "execution\_count": 14,

     "metadata": {},

     "output\_type": "execute\_result"

    }

   ],

   "source": [

    "dataset"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 15,

   "metadata": {},

   "outputs": [

    {

     "name": "stdout",

     "output\_type": "stream",

     "text": [

      "(31942, 15734)\n"

     ]

    }

   ],

   "source": [

    "train\_size = int(len(dataset) \* 0.67)\n",

    "test\_size = len(dataset) - train\_size\n",

    "train, test = dataset[0:train\_size, :], dataset[train\_size:len(dataset), :]\n",

    "print(len(train), len(test))"

   ]

  },

  {

   "cell\_type": "markdown",

   "metadata": {},

   "source": [

    "Now let us define the function called create\_dataset, which take two inputs, \n",

    "\n",

    "1. Dataset - numpy array that we want to convert into a dataset\n",

    "2. look\_back - number of previous time steps to use as input variables to predict the next time period\n"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 16,

   "metadata": {

    "collapsed": true

   },

   "outputs": [],

   "source": [

    "# convert an array of values into a dataset matrix\n",

    "def create\_dataset(dataset, look\_back=1):\n",

    "  dataX, dataY = [], []\n",

    "  for i in range(len(dataset)-look\_back-1):\n",

    "    a = dataset[i:(i+look\_back), 0]\n",

    "    dataX.append(a)\n",

    "    dataY.append(dataset[i + look\_back, 0])\n",

    "  return np.array(dataX), np.array(dataY)"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 17,

   "metadata": {

    "collapsed": true

   },

   "outputs": [],

   "source": [

    "look\_back = 10\n",

    "trainX, trainY = create\_dataset(train, look\_back=look\_back)\n",

    "testX, testY = create\_dataset(test, look\_back=look\_back)"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 18,

   "metadata": {},

   "outputs": [

    {

     "data": {

      "text/plain": [

       "array([[ 0.1997695 ,  0.49828053,  0.18666792, ...,  0.49753141,\n",

       "         0.19973087,  0.48600531],\n",

       "       [ 0.49828053,  0.18666792,  0.4855442 , ...,  0.19973087,\n",

       "         0.48600531,  0.18442059],\n",

       "       [ 0.18666792,  0.4855442 ,  0.18251848, ...,  0.48600531,\n",

       "         0.18442059,  0.48598576],\n",

       "       ..., \n",

       "       [ 0.53376245,  0.69436169,  0.53105354, ...,  0.70821238,\n",

       "         0.52058411,  0.70815468],\n",

       "       [ 0.69436169,  0.53105354,  0.70823145, ...,  0.52058411,\n",

       "         0.70815468,  0.52665424],\n",

       "       [ 0.53105354,  0.70823145,  0.53320551, ...,  0.70815468,\n",

       "         0.52665424,  0.70815468]], dtype=float32)"

      ]

     },

     "execution\_count": 18,

     "metadata": {},

     "output\_type": "execute\_result"

    }

   ],

   "source": [

    "trainX"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 19,

   "metadata": {},

   "outputs": [

    {

     "data": {

      "text/plain": [

       "array([ 0.18442059,  0.48598576,  0.19208527, ...,  0.52665424,\n",

       "        0.70815468,  0.5206418 ], dtype=float32)"

      ]

     },

     "execution\_count": 19,

     "metadata": {},

     "output\_type": "execute\_result"

    }

   ],

   "source": [

    "trainY"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 20,

   "metadata": {

    "collapsed": true

   },

   "outputs": [],

   "source": [

    "# reshape input to be [samples, time steps, features]\n",

    "trainX = np.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))\n",

    "testX = np.reshape(testX, (testX.shape[0], 1, testX.shape[1]))"

   ]

  },

  {

   "cell\_type": "markdown",

   "metadata": {},

   "source": [

    "## Build our Model"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": null,

   "metadata": {

    "scrolled": true

   },

   "outputs": [],

   "source": [

    "model = Sequential()\n",

    "model.add(LSTM(4, input\_shape=(1, look\_back)))\n",

    "model.add(Dense(1))\n",

    "model.compile(loss='mean\_squared\_error', optimizer='adam')\n",

    "model.fit(trainX, trainY, epochs=100, batch\_size=256, verbose=2)"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 22,

   "metadata": {

    "collapsed": true

   },

   "outputs": [],

   "source": [

    "trainPredict = model.predict(trainX)\n",

    "testPredict = model.predict(testX)"

   ]

  },

  {

   "cell\_type": "markdown",

   "metadata": {},

   "source": [

    "We have to invert the predictions before calculating error to so that reports will be in same units as our original data"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 23,

   "metadata": {

    "collapsed": true

   },

   "outputs": [],

   "source": [

    "trainPredict = scaler.inverse\_transform(trainPredict)\n",

    "trainY = scaler.inverse\_transform([trainY])\n",

    "testPredict = scaler.inverse\_transform(testPredict)\n",

    "testY = scaler.inverse\_transform([testY])"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 24,

   "metadata": {},

   "outputs": [

    {

     "name": "stdout",

     "output\_type": "stream",

     "text": [

      "Train Score: 4.77 RMSE\n",

      "Test Score: 5.57 RMSE\n"

     ]

    }

   ],

   "source": [

    "\n",

    "trainScore = math.sqrt(mean\_squared\_error(trainY[0], trainPredict[:, 0]))\n",

    "print('Train Score: %.2f RMSE' % (trainScore))\n",

    "testScore = math.sqrt(mean\_squared\_error(testY[0], testPredict[:, 0]))\n",

    "print('Test Score: %.2f RMSE' % (testScore))\n"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 25,

   "metadata": {

    "collapsed": true

   },

   "outputs": [],

   "source": [

    "# shift train predictions for plotting\n",

    "trainPredictPlot = np.empty\_like(dataset)\n",

    "trainPredictPlot[:, :] = np.nan\n",

    "trainPredictPlot[look\_back:len(trainPredict) + look\_back, :] = trainPredict\n",

    " "

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 39,

   "metadata": {

    "collapsed": true

   },

   "outputs": [],

   "source": [

    " # shift test predictions for plotting\n",

    "testPredictPlot = np.empty\_like(dataset)\n",

    "testPredictPlot[:, :] = np.nan\n",

    "testPredictPlot[len(trainPredict) + (look\_back \* 2) + 1:len(dataset) - 1, :] = testPredict\n",

    " \n"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 40,

   "metadata": {},

   "outputs": [

    {

     "data": {

      "image/png": "iVBORw0KGgoAAAANSUhEUgAAAYAAAAD8CAYAAAB+UHOxAAAABHNCSVQICAgIfAhkiAAAAAlwSFlz\nAAALEgAACxIB0t1+/AAAADl0RVh0U29mdHdhcmUAbWF0cGxvdGxpYiB2ZXJzaW9uIDIuMS4wLCBo\ndHRwOi8vbWF0cGxvdGxpYi5vcmcvpW3flQAAIABJREFUeJztnXd4VEXXwH+T3SQbkpAQQg8Yeu8R\nUECxYy+gIgjYXvW19w8LgtiwvorltYGiIogFRQVREHhFlI70EiRAILQAISF9d74/dpNssn33bp/f\n8+zDvefOnTkTdu+5M3PmHCGlRKFQKBTRR0ywFVAoFApFcFAGQKFQKKIUZQAUCoUiSlEGQKFQKKIU\nZQAUCoUiSlEGQKFQKKIUZQAUCoUiSlEGQKFQKKIUZQAUCoUiStEHWwFnpKeny8zMzGCroVAoFGHF\nmjVrjkopG7kqF9IGIDMzk9WrVwdbDYVCoQgrhBB73CmnpoAUCoUiSlEGQKFQKKIUZQAUCoUiSgnp\nNQCFQhFdVFRUkJubS2lpabBVCQsMBgMZGRnExsZ6db8yAAqFImTIzc0lOTmZzMxMhBDBViekkVKS\nn59Pbm4urVu39qoONQWkUChChtLSUho2bKge/m4ghKBhw4Y+jZaUAVAoFCGFevi7j69/K2UAFGHH\nP0eKWJ59NNhqKBRhjzIAirDj3NeWMvKjFcFWQxHBzJkzByEE27Ztc1ruk08+4cCBA163s2TJEi67\n7DKv7/cVZQAUCoWiDjNnzmTQoEHMmjXLaTlfDUCwUQZAEXakUkimyAu2GooIpaioiD/++IOpU6fW\nMgAvv/wy3bt3p2fPnowbN46vv/6a1atXM2rUKHr16kVJSQmZmZkcPWqenly9ejVDhgwBYOXKlZx5\n5pn07t2bM888k+3btwejazYoN1BF2PFr/KM0EieB24KtisKPPPPDZrYcOKlpnV2a12fC5V2dlvnu\nu+8YOnQoHTp0IC0tjbVr13Lo0CG+++47VqxYQb169Th27BhpaWm8/fbbvPrqq2RlZTmts1OnTvzv\nf/9Dr9ezcOFCnnjiCb755hstu+YVygAowoqteSfpLMwPhbJKI/F6XZA1UkQaM2fO5IEHHgBgxIgR\nzJw5E5PJxM0330y9evUASEtL86jOgoICxo4dy86dOxFCUFFRobne3qAMgCKsKKkwVh//kX2UQe0a\nEadXM5mRiKs3dX+Qn5/Pb7/9xqZNmxBCYDQaEUIwbNgwt1wu9Xo9JpMJoJZ//vjx4znnnHOYM2cO\nOTk51VNDwUb9chRhRYzVj/CWT1bT4an5QdRGEWl8/fXXjBkzhj179pCTk8O+ffto3bo1aWlpTJs2\njeLiYgCOHTsGQHJyMoWFhdX3Z2ZmsmbNGoBaUzwFBQW0aNECMC8chwrKAChChlNllRw+Wcqm/QWc\nKqu0WybGzkvYriNFNrKJczczf6NaKFZ4xsyZM7n66qtryYYNG8aBAwe44ooryMrKolevXrz66qsA\n3HTTTdx5553Vi8ATJkzg/vvvZ/Dgweh0NdOTjz32GI8//jgDBw7EaDQSKggpZbB1cEhWVpZUCWGi\nh8xxP1UfD+nYiE9u7mdTZtP+Arp92AqA9yovZ67xDLbITM7v3JiPxp5eXe6tp8ZySDbgueff8L/i\nCs3YunUrnTt3DrYaYYW9v5kQYo2U0vnKNGoEoAhR1u45bleecHxH9fGd+h/4Ou4Z4iln5dbdtcrd\nq/+O52I/9quOCkW4oxaBFSFDc44SI0zkysY2C26Lth4i2RBL6c51tLWS1xNlbDfcBICU19L68Xk8\ncH57Hgic2gpF2KIMgCJkWG64D4DM0i+o63Bx63TzVOClMfs4K87+/Z8sz+EV/XusWdwBvAuP7hYm\nk3naNMbegoRCEUaoKSBFyHFmzCYbWY5hJLviRzm9b8U/x7hW/z8mx37kL9UAaPPEPNo8Mc+vbSgU\ngUAZAEXI8UXcC9xg+tFGrhOSu/TfO7wvrcDWcPiTqpGAQhGuKAOgCEn+j+kAzF69jzcW1iz8do3Z\n4/Cex4/8n43MkTupFpRWOnbnk1IyddluTpaGxo5PhcIeygAoQprHvt7AGwt3ulU2WZTYyGRlmdYq\nucUf2fk8++MWxkxdGZT2Fd6Rn59Pr1696NWrF02bNqVFixbV5+Xl5W7VcfPNN7sM9vbOO+8wY8YM\nLVT2CbUIrAhp2or9NBYnvL7/6J7NJHWx3U/gCzmGkQB8tWETw/tm2A0RsGDzQeIpZ/0+73VXBJ6G\nDRuyfv16ACZOnEhSUhKPPPJIrTJSSqSUxMTYf3/++GPX7sd3332378pqgBoBKEKW4vJKFsU/ysy4\n572uY9+xYg01qk35d/dz76fL7V47unI22w03MUmv9iJEAtnZ2XTr1o0777yTPn36kJeXx+23305W\nVhZdu3Zl0qRJ1WUHDRrE+vXrqaysJDU1lXHjxtGzZ0/OOOMMDh8+DMBTTz3FG2+8UV1+3Lhx9OvX\nj44dO7J8ufk7derUKYYNG0bPnj254YYbyMrKqjZOWqFGAC4oKKkgMU6HXqeNrVy8/TC/bT3Ms1d1\n06S+SOZIYRmn+ViH1hvdtx8spKPleJR+EUd2pgADbcr9N+5NAMbof9VWgWhi/jg4uFHbOpt2h4sn\ne3Xrli1b+Pjjj3nvvfcAmDx5MmlpaVRWVnLOOecwfPhwunTpUuuegoICzj77bCZPnsxDDz3EtGnT\nGDdunE3dUkpWrlzJ3LlzmTRpEj///DNvvfUWTZs25ZtvvuHvv/+mT58+XuntDDUCcEKF0UTPZ35h\n/Pe1vUtMJsldM9awZs8xj+u8+eNVfPaX44VMRQ178g76XonGFiBn25pa5w/ov9W0fkXo0rZtW04/\nvSbcyMyZM+nTpw99+vRh69atbNmyxeaehIQELr74YgD69u1LTk6O3bqvueYamzLLli1jxIgRAPTs\n2ZOuXbWPjupyBCCEaAl8CjQFTMAHUso3hRBfQvXLUCpwQkrZy3LP48CtgBG4T0q5wCIfCrwJ6ICP\npJTemeIAsWT7EQC+X3+AF6/pUS0/eqqMeRsPsnL3cVY/db5HdT6g/5qRut8AZQRcUXHIvcVfZzTb\nPx84p/r8YEEpCzYfZMLczTx/dTdG9fdsjBG36j2XZXKOniLTQz0VdvDyTd1fJCYmVh/v3LmTN998\nk5UrV5KamsqNN95YK/xzFXFxNbsWdTodlZX2vdLi4+NtygQiTps7U0CVwMNSyrVCiGRgjRDiVynl\n9VUFhBCvAQWW4y7ACKAr0BxYKIToYCn6DnABkAusEkLMlVLams0Q4V+fmnefFpdrF72v6o2xtMKI\nIVYlM3HGsWLfXTgPHzpAe6vzoS9+R2Nxgpf083llzvUeG4AEk/01hcxxP9JN7KZ9r8H8tW4Dfxp8\nUFoR8pw8eZLk5GTq169PXl4eCxYsYOjQoZq2MWjQIGbPns3gwYPZuHGj3RGGr7g0AFLKPCDPclwo\nhNgKtAC2AAizC8R1wLmWW64EZkkpy4DdQohsoMoNI1tK+Y/lvlmWsiFrAJIoZpPhNqZUXgVcqmnd\nlz49lUUv3q5pnZHG9OW7uTbetzqO5OdXH0spWW+4o/o8BhMw0mUd5ZUm4vQxZB8upGXxJrATASLH\nYN6lPHvj2fzHsLTWtYKSClIS/BibQhFw+vTpQ5cuXejWrRtt2rRh4EDbdSBfuffeexkzZgw9evSg\nT58+dOvWjZSUFE3b8GgRWAiRCfQGVliJBwOHpJRV4/UWwF9W13MtMoB9deT97bRxO3A7QKtWrTxR\nT3M2Gcw5Z+/Tf+eghPdDtEXxj2LppsKPXKWr8dJ5ZcF2HrO6ZnRjCWzu3we4b+Y6BrZryK7sHfxl\nyHda/jr9UhtZ/2d+YNvka9zWWREaTJw4sfq4Xbt2tTxwhBB89tlndu9btmxZ9fGJEzVuwCNGjKie\n03/uuefslm/atCnZ2dkAGAwGvvjiCwwGAzt37uTCCy+kZcuWvnWqDm4vAgshkoBvgAeklNaZmm8A\nZloXtXO7dCKvLZDyAylllpQyq1GjRu6qF1CE3a54Tua4n3jkq781qSvcMdoJq9Az5h9N25ixYm/t\nNnE9BffLZvNC9Izci/jLcK/dMndMtX3oWzPQTmwjhcIVRUVFDBw4kJ49ezJs2DDef/999HptHTfd\nqk0IEYv54T9DSvmtlVwPXAP0tSqeC1ibqQzggOXYkTzsaC3yqJSNfaojxzCSg5sawLU52igVxpik\ntHkcPx87TdM2ssr+AqtIou6MALYcOIke52sRr+y93v7rjYWpca+ROa4v258bqpLYK9wmNTW1Or2k\nv3D5C7DM8U8FtkopX69z+Xxgm5Qy10o2FxghhIgXQrQG2gMrgVVAeyFEayFEHOaF4rladCIYLI5/\nmDeMLzq8nnu8mEk/bHEZMKypsJ/4RKEdq3OO8fOmg0yNe62W3OTGSK7TsUVkG8Y4LVPfTgiKuuQY\nRjL2ta9cllMoAok7U0ADgdHAuUKI9ZbPJZZrI6g9/YOUcjMwG/Pi7s/A3VJKo5SyErgHWABsBWZb\nyoYUhaUVlFeanJb5fp3Z3vWts379R/ZRjhSWcfq4z/nn9Qt5es0ZZG9Z67LNjSpcAPP8mL9XTruI\noV93tJG3Fq73GVyh+1MzPWaV3GEjqzSayBz3U610mApFoHBpAKSUy6SUQkrZQ0rZy/KZZ7l2k5TS\nxjFaSvm8lLKtlLKjlHK+lXyelLKD5Zr3+/v9SPeJv3Dj1BVOy7w7rybA17KdR6uPR320gkEv/cb8\n+Mc5S2fewdhws+tpjP1f/NtLbSOHR2b5L/fz6TE77MrP1m1wet/+EyUYcC8AmLes23eCT2Nf5PKY\n5VQYnb94BJKC4goOnDCPbNbvO0HmuJ/4I/uoi7sU4YbaCWyHlbsd7/AtKqvkmdhPqs9vmboMk0ly\n6GQpp4mD6CtPkS6s18hdTzMkm066LBPpLIl/MNgq2HCsqJxBMRqHIrDDWbqNvBX3NsPf+s3vbblL\nz0nzOfOlBSzefphrZ7xGcudx3PHDy8FWS6ExygB4SHFZOZfrarxcdxjG8uaHHzJvYx5L4x9is+HW\nWuX3H3c9PzygzH5AsWiihXDuXukvypzE9AfQC/+9le/NL2b/3hpPp3eO204RBYvkzk+S3Gk8N3+6\nhPgmPwAg0ubz0wb/TdWFAlqEgwaYNm0aBw/WTDG6EyI6GCgDUIdn9B9zVcwyh9f3Hy2wkT2Y9yiH\nD9n/YZSVFLpsU0foDP2jDX9GC3XF469N4arfzqs+zxChN8WS3OFZREyNkbz7C/96pQSbqnDQ69ev\n58477+TBBx+sPrcO6+CKugbg448/pmNH23WoYBN1BsBkkryzOJuCEvuZmsbqf+WNuHcd3r/AwWKl\nkPYf4jpZ8+NxFtujtMLIhO83sWT7YYdlFNoTgHArtdi8bSuFpRW8v3QXM+JsvchKy4KTwMZd9Mn+\nnxILVaZPn06/fv3o1asXd911FyaTicrKSkaPHk337t3p1q0bU6ZM4csvv2T9+vVcf/311SMHd0JE\n79y5k/79+9OvXz/Gjx9Pamqq3/sUdeGgl+w4zMxffuefwyd57XrPw6t+8mcO4+zFeXHwJGlVsrX6\nuLysBEeRDTqN/xmA6X/uIWeybdgJo0mSX1RG4/oqyIyWzN90kPZNku1eizHaBvfyla6zBvBI1//x\n9Zpc7rDzX2l4sTFMtB1lhgoJGV8AjwekrZdWvsS2Y9s0rbNTWif+r59t6lBXbNq0iTlz5rB8+XL0\nej233347s2bNom3bthw9epSNG82G8cSJE6SmpvLWW2/x9ttv06tXL5u6HIWIvvfee3nkkUe49tpr\nefvtt33uqztE3QhAd3QHy+IfYNBh79KxnR5jfx5vTfZ+u/L0ihp5UYHjxWUdRobrlnKDbpHd622f\nmEe/Fxax71gxN3zwFz0mLvBAa4Ujvlu4xOG1uJP+idhaWFpBvJ+9ixTasnDhQlatWkVWVha9evVi\n6dKl7Nq1i3bt2rF9+3buv/9+FixY4FasHkcholesWMGwYcMAGDnSdYwqLYi6EYChyBwOoH2p7fZ8\nk0lWW8RNuSewl7Llszj7IWpfKn7apTndklfAYAfXdhlGW53V3W8HeipJp4AFmw/y5z/mBdNXnvwX\nj8bOZtdd+2jbuL7zxhV2+S3+Ef7YfD4Du7YOWJvHThTwVdwzAWsvXPHmTd1fSCm55ZZbePbZZ22u\nbdiwgfnz5zNlyhS++eYbPvjgA6d1uRsiOhBE3QhAmsx/bJOdrlddA/hr6kMe1ZsZc8iNxj2qshbZ\nhjH8ZbiXD+YtpzlH6SN28GjsbADavqttgKhoY+BXvTDa8cHfud8/6zFf5V9Dj5jdfqlb4R/OP/98\nZs+ezdGj5oX6/Px89u7dy5EjR5BScu211/LMM8+wdq1542dycjKFha4dQKzp168fc+bMAWDWrFna\ndsABUTcCOHLS7JaZd7KcHnWuHT20jyaW49vkN5q3LV2EhbBmz9bVnCoppUufQbXkfWN28t/YNzxu\ne/+JEhJidaQluu/JEE2YjBXodLVXaAqKXLvwKqKD7t27M2HCBM4//3xMJhOxsbG899576HQ6br31\nVqSUCCF46aWXALPb52233UZCQgIrV650UbuZKVOmMHr0aF566SUuueQSzUM/2yPqDEBZmXlh7yKd\n7c7TkyUV1QagLnOfuYorJjgKC+0eD3+1nlVurOGajCZO+9LiHtin9oLgIOHYC+Omj1fy8AUd6Z5R\n+4sjpaTFG03NJyG8wBhMjhw9QvPmGbVkBeu/C8ovpPCFdiQ/kR34hhW1sA4HDeZ5eXtz8+vWrbOR\nXXfddVx33XXV5+6EiM7IyGDFihUIIfj888/JysrytQsuibopoBSj47g7Djw5AbhCLva57Y/qBCNz\nxGefOk47OEpvf5GYiSnE75zHqLdtF4dPltRMbRWXB2++EczGaPfRU0HVwR55B/bZyOoRHJfM5PIj\nQWnXVxZvO8yE71Xoa29ZtWoVvXv3pkePHnz44Ye88sorfm8z6gyA0eT4KS9dTdJXeD8lsGbPcbfj\n29c7XPNG4Ule0Pfj/sMGw79s5GXGmr0IG5/TPnORJ0xfnsM5ry5hfcgFwLP9O4/R/xoEPYJPpcm7\nl4SbP1nF9D9re05t2l/gcre1wsyQIUNYv349GzZsYOnSpbRp08bvbUadAVi928luS1cP2+ebet1u\n1dqDO1xbMrv6eOXLl3u8WzWnzhv2d8tqgp71j9HWr9pTTuz4gxzDSI7trAmnURkCQdBOHdwVbBVs\nWL/vBH8HwVC+tPAPp9e7T+9Or08GuKxn/4kSLntrGU9/51nQ30AkQ48UfP1bRZ0BeCrWsf+/Id9/\n0akPrpvvupAd+pf8zuCXPZt+GvLqklrnv//uYNooCDyw524ABi2tmUt98v0vg6VONUXbl1QfHztV\nzuIg78jeuHEdvaaeRsmH2iYad4XJJPlk/TyX5Yyi9kuGlBJDxqfExNeEPygoNu+2/zvXfSNmMBjI\nz89XRsANpJTk5+djMHi/OTTqFoGdoSvyX6CrztkfeG1uu4gcn9q+PKZ2TPu8ghKapST4VKevxAkj\nu44U0bZREhW562tl6goG8Sazc8CanHze+eBdfjP1ISeIm65TfjDnox4Qs5WfN+UxtFuzgLR748xp\nGJq4NgBV/LRlCxOXP8Mr5z5FbPIWYpO3ALe6vM8RGRkZ5ObmcuRIeK6DBBqDwUBGRobrgg6IagNQ\n5bpVRb0/JrsTvdkrfJl6mRf/hEflB8ZsBGrCSZxep+0bJ3/GoheDn5D+vNeWsmXSRTwR+0WwVeG8\nUz+SOe4ntsePYVpccBfKAVqV13gBrZr5HEOff8cv7SwZfxbfGgcz5QVzeo6NlZ65GI9bdT3EwpRV\n3u2sr0tsbCytWwduU160E3VTQNZMX55T6zxNFAVHEY2pG2SsdZ1NaoviH2XGl7USuQWF08U2isuN\ndfInBI9LYv4iXgT/4V+X8bGf+6XepYvmMUT3N1PifI87c7LCdspM+OllSqEdUW0AZqzYyw1PvMyM\nhauCrYrmHD/lPNbMqK13UlIeXO+Mr+InkX8yeOGY6/Ju3JRgq+CQKYt2al7n7mXa7TY9bKzxXMs+\nXHsHrJrOD12i2gAMzf+UmXHPM2rZ+cFWRXMWvHityzIT39dm2O4LV07xfX9FNHD4t3cwerCT3B1u\nkt/XOjc52wjjAknNyOnKHy5k6t/B/24pXBPVBuDh2K+DrYLfGKFf4rJMv8PB9765UqeyobnDc7Ef\nc/yg/+IHGSc1ouenPTWpK0ZfxBvrJ1NQdpz4xj+CUEOAUCWqDYA1J075b9dnSWGobXoyM0y3jJ83\nHbTZNxBIXor9MGhthxsx7w+udo/ctL+AzHE/kTnuJ5bv8j2TmM6kfXjqfy29jLiGy9hd6VsIFYX/\niCoDsPmA4zg4B05on/yjil9W/u23un1l0Fc9Wf3miGCroXCDNFFEXoH5e2rtwDDywxVB0sg94hst\nDLYKCgdElRvorJX7sI3mbUZXdtxv7e7avsF1oSCRJEoZrvtfsNVQuEmVZ03jshxyDGOsrgQ+yF/3\n6d0D3qZCW6JqBGDKXevw2j/bbCP6acVDh5/yW93OMJokRwpDO8eswjOObl4CQLcdjvNWe4oEXmvg\n//yzitAjqgxAi4OOg3vlF0Xeg/Kp8Q/T6LXGbpVdvusoP2444GeNFL5SkZ8DwMUxfzkv6ALrUAtF\nQvBJqsooF41ElQFIwnFANhGBzsovxk51u+zoD5dzzxfrMGnsaqjQlviyfK/vvfGjFXz6Zw5SSrIP\n1jgmSLVhK2qJKgMgnIR7HrX1zgBqEnpM1E+nKfkcPOm/xXB/scPUItgqBBDvf7J7dm3m1e9X0vrx\nefz49sPVcmXyo5eoMgAKx4zWL+Qvw718+HP47Yr+u7ejpf0IJMb7n+zv8Q+ywfAvFsY9woOx1ilP\ngzsE6PzGv+n838uDqkO0ElUGIEa967hkwrbw+yEmGaLHma3rhhc5dcJzv//yyppdvu1iaq/1BOJX\nYbKTiGnOuly6P/8J+gbL0NfL4edNB21vVPgVlwZACNFSCLFYCLFVCLFZCHG/1bV7hRDbLfKXreSP\nCyGyLdcuspIPtciyhRDjtO+OItp4sOVXdh8ukUzsWz08vuefo8ENdNjxlSdtZI/8+DVk1KRJvfPz\n8Bt9hjvuvDpVAg9LKdcKIZKBNUKIX4EmwJVADyllmRCiMYAQogswAugKNAcWCiE6WOp6B7gAyAVW\nCSHmSim3aNslxzhbA1CEJ/+59ULW7dwDUfTsiDN6vnM7e88+OvlBF3cxNP2RorJnSYqveeTUa1XH\nSUGo1JGBxuUIQEqZJ6VcazkuBLYCLYB/A5OllGWWa1XxYK8EZkkpy6SUu4FsoJ/lky2l/EdKWQ7M\nspT1O1JKLvzPUtRyV+CZ+e4zMDEFTP77cfds26rWual5H7+1Fap8u8aci/emj1eSOe4nm+sFf37q\n8N5A/Spe+X0W6/aaN1xu2m+7cS250/gAaaKowqM1ACFEJtAbWAF0AAYLIVYIIZYKIU63FGsB7LO6\nLdcicySv28btQojVQojVWmUF2pJ3kh2HioK81BU+HDms3VzsDYdfNx8Ue+++6IinK8YCEBNT+3/W\ndNPPLDBmad5eKLP720kALNlu/s3U3deyL9/xFFCgDMC3+19izOKzALhh4aAAtapwhtsGQAiRBHwD\nPCClPIl5+qgBMAB4FJgtzOm17D1npRN5bYGUH0gps6SUWY0aNXJXPaeUVpjIMYzkBr0KPewOjd7t\nyOxV+ygsrdCuUj/ss5j0fE38/hsNNUlN9HHxxOqiyr+BrjHmEcBFMavIMYykwrLoK42V7FvyMdLJ\n60+gx8VlRsebLq0XqxX+x61fiRAiFvPDf4aU8luLOBf4VppZCZiAdIu8pdXtGcABJ3K/Yyx3vAFM\nYZ/HvtlA94m/aFrnmj3HNK3Pms/Hja513qphot/aCkWG6lbx7Q/f837cfwDQ55vTgBa+nkXLJQ/w\nuO4zh/cGeiPY5TPvc3ht9NQVVBiVEQgU7ngBCWAqsFVK+brVpe+Acy1lOmBO630UmAuMEELECyFa\nA+2BlZiX6doLIVoLIeIwLxTP1bIzjjgUhpubgs3gmA3o0TI9oiT3eOAMcbilIzzWZYzrQi64Zk1N\nHQWnzH/r+qdc5xAI9Aggz+g4B8S6oln0edE/KTAVtrgzAhgIjAbOFUKst3wuAaYBbYQQmzAv6I61\njAY2A7OBLcDPwN1SSqOUshK4B1iAeSF5tqWs33nkq9ANxxyqfBY3mYf1X2lW38GfXqDeqb2a1WeP\np3QP8EGTpwHY3fBsv7Wz+ez3NK8z8fxHNa3v3pnuBzc0hdDqWHz6b5iafBRsNaIGl26gUsplON4q\neKODe54HnrcjnwfM80RBRfDIFNotBjfdNp3shudoVp89nhv/TPVxTP1mfmun7cDh/LN4IptkGy5/\nchZicivXNznAKAU6IdEnuRe0z11G6hYBd2taZ6CIifNfaHZFbaJnC6XCYy7Waetcn5kTuBSUHZok\n+61uQ1wspntWM8AQizAYquVvV17JPXpznt0dfcfTYY3jEBVLRuyg4FQp3Vs2YNu+I1wSZ3BY1htu\n1C8Co3tTeMo5OnqJCgMwMGZTsFUIWyqMJs08ajL2z9ekHneoTEj3a/3tGlsZmNt+g/rNKFt6ANaY\nDUBCq76wxva+HaYWlBPLkE5NqmVtGvspFPOzDd0qFsoGYEPuCTbkFnDjgNOCrUpEEhUG4An9F8FW\nIWzZk3+q9sPOA75bk8NVGuvjLrKeew8/TcjoC8CDlzareejHxNotenTs/zDE6gKkmHuEYjjo/KIy\nGibFc8XbfwAoA+AnosJZum7wK4X/+Pfna+g+YQEAbb4P1uMf4p08ZA//a71f2qy7Ic2aQl0q2+K6\ncWbbdPq0amC3zO4RS/2ilytCcQRw7JT2SeoVtkT8CKDSaIr8TvqRbXknPRoBzLdEdCyrNNIjxrUL\nor+IMyQ5vNa4RWu/tfus7m4y23TkHCtjUHzaeSTf/K3LWDytO/Xym17OCEUDsLfwH9o36Ym+/jri\nGi4FLg22ShFJxI8AflBpDn1EpM7LAAAgAElEQVRi247tHpV/Xj+VHMNIf2z8rcUh6TyHbXp6I35N\n99233lPGj3+B0aPG1pKJ4dMCrocnONslHCwqTOYRQEKLL9EZDvLLltwgaxSZRLwB2L3Vf8neo4EW\npxxv1SguKWHd7z9Un5cXFzJKv8h84mcLcPTsF12WSc/son3DV/3XrWKViTVuqAnJ7idcv6V+4H3g\nQ3EEUJeHV10cbBUikog3AN22/ifYKoQ1a3c63rxV76Wm9F50I7vXmENG7H+5f/W1Fb/M8qteXc8d\n6bJMST1tU0WaLnwBerluF0DqvXPrnPbQtZDWFlr7byNbXcLBACj8Q8QbAGeJ4BWu6S12uixTmLuV\nSqOJ1uyvlp29OvibkIoTW7ou5AExZ7rfJyF9iGdz31oYG5AoKQCYQm8GiOJKz3MeKDwn4g1ArNAy\nno37vFF5jdPrBbJegDTxjfN1rqfQ/j5wioVfuTc1ogUnBk3wqPxBWeN1UxBj3wNHa6piEZ2QoR+U\nLhRHAOPmf83qHP8FD1SYiXgDcHrMjqC0e+29Lzu9fgBtt/77Cx2uE7n0PLmE03e9FQBtzJS0vsCt\ncmf06MxaUzv+7vsC+6V5X8Ds9Lv8qVoNlhGAEBH/E/Mbw9/7I9gqRDzq26khG02Z1cfN6sc5Ldv8\n/l/9rI026LCdyliz5zgVO2tyKzSpzKVhRV7AdGrauqtb5RITDPSZtIaLrhjJ6PgpTKgYS+bZFs+g\npwPzdumLh81c4xkaauKYUBwBxKcvJbnzE8FWI+JRBkBDYodYRXR0sgh4avgsUtIa82bl1QHQyjdS\nRe252DV7jjPrgxeInVGzyau0IrC5XEWM51/bOQ9eRJML7uO8zpYQDDHu78bd22YE3LrQo/aq1gB8\nibQZKPfMUIoGGo6c8eJ8rnhnCZ/9mYP0t/+zxkSsAZBSclLLjFZu0LKxeZrhYJOzEQ4eMHtNjUjs\nZnZpa9UgPNYBrFm2Zj2vxH5QSyZN/v/Sdyv1zT0yJSGWu4a0c7pbty5FDToD0HLEf6Dl6S5K16bK\nAEgfEhO0atbU63s9IbweWaFHUfPH2BU3gfHfb2bB5kPBVscjItYAfPrnHnponNGqijcdLPAmdrkI\nrnyXpjd/Bg7mfsUt1gHRwu+nV7/c9gveWJzQtpGJtgnD/5gQ+LASSff9CeOPIuI8N9RGQxpHZApT\n4v7ldftNrnvN63s9Ify+haHD7qPmEXJMnHlKsawysKNhX4lYA/DTBv/NSXc7fYiN7MWKGyAmBnqP\nAkOKw5RULTPbVx/3v+IOf6noN+LsuOfVE45zvHrLydja+aBTEuwHV/MF+eguh9fWmdqZ/w91Xrar\ni+X0sv+yWDfQS+2gcYPU6uT2K5qM8Loel6gZIK958MuauFJJHSby0I/+3f+iNRFrAPxJz45tbWTZ\n7W+tdS7cGPo3b98LHtiomV6BILU8MIu9egchqHeZtEv0IhLTmd/mSbvXfsoc51vdGjxU9boYLnz6\nJ3ac8z797tQ+C1kVagTgPev31TgTCF0phiY/OCkdekS0ARgU45+Ha0Wzvjayt0f28a6yVO+zSQUF\nXzY4eUBhYmb1cZmIB+Cj89Zx7BbH+WS9wVg/w678qVuu1bQdbxGxBjqcPcKtFwpvcfY/mnLiHgAq\nTth+5+1xX+/7NdAofIhv8mOt85i4/CBp4h0RawA65n3P53Gu48V4whz9UDJLvyAtscbFM7vTv9nT\n5AIS4nyP8b6tWfDCJ4caOS2vrD4+0ck8/XHb4DacnpmmaTvdMwKzMSyUcTQCGNRoGN/dMpbyg9ew\n+OY3MZa43ll9c7ebNNXNmvIQnF+PS9P2hSTQRKwBuFIu0rzOvheMJGfypcTrdZyQieTLZNqNmMxp\n//7arftnVQ5xev1Yx+s00NKPFB8jzlgckKb2Na8J/tXkujf91o69N+vV1yzTrH6p4QTLEemnzGEO\nFgEmDR5HerKB7f/3DM3qp6AvGOq0ltOSW6OP0VN2+CJ/KMmWw/tdF1J4RMQagO5Cm1j0SxqNqj5u\n0rlmY07fsvc4s8Kz8Ad7ej1iV370xt84OPRDYhO1fbvVnJdbc8GBwIR8kNYZtfw4/dG8tW3E0Kwe\n3X2uV/hhZXWFzstpRhc4MlF1e5Cgdx7W4sdrzPGLVtz1vO9K2UEnVGYPrYlYAxAvtNkD0Lun1Y9O\n1EzzbHjmEtZN9CxE7QX9utmVp7frS9MB15Ga6Hz3cDTRLCWBm8sf5cvMSX5tR9+gJQfaXs+h6+fD\nmLnm/L4hyuCR/tkZ6ygYXIN6tb+PNw6wXa+aedE82/sS4zXRywYXgykpJSXl/psmWrPnGNe8G1nh\nKZRJdUFF097kmJqQGXMIYhOq5Ynxrv90Rn0iOiu3yeYpCU5Kgy69g/eKRggnWwymPjCofTq6W/9N\nv9Z+HhUJQfPRH7guFwKktOvvupAX1H2uLh/xJ0eKj6Gr44mVEGv7ne/axP4iejCYuXIfT8zZyP8e\nPYdWDbXfZDnsvd8Rscc5cKIPzVOd/5bDhYgdAWhFfEYPiu9YyWtZi4k3ePal0j1aO5tW0xTnMeLb\neJl83d9UGgPj+QNQf/jb1cdntG2IzoOdu6FESj3zFNbQroHZzesLdQ1AcnwSbRrYvu33z+iCNMZT\nWdi5WubIO+nDCz/UUkUA3lu+lnHfbGDRVvu7bedvMrso/3O0SPO2AQxNvyWp7Ws8N381rZ/6gpLy\n4EQa1hJlAJyQd9Zkkg2xdGmRysOXeTH/Gp/MdpM2b0jTe3yuST3esHL3MaQpQB4YDTID046fSUmI\nZd34Cxh3cWfXhcOE7s2bsPjaP/hXz5ttrunrzM8PaDZA8/YXHfiSWav2cev01Xav7zy+k3qt3+RE\n6UnN2wbQJZo3Di45OYmk9i9y25zARcD1F8oAOKF+Pe+yOllzSGrjZjj2msvZYdI2w5Un7Nm1NWht\nhysNEuPCYgTjSTC4RsnxdE7tSfmJLLoaXwJgxcgVLB/pf3fI2JT1Tq+fMMxFZ8jj++3/80v7MbHm\nECU6w0EANpTaz/V8sCAwnnJaoAyAn/lT594GGne4sLxOjoEJGsfgcUD6P99SXFoekLYUgcd6Cuiy\nTNcb4DLTkyjLG84FHcyjm3qx9UjQ286JX5E5ykbmK4ntJxHX6OdasoMFpWSO+4nYZPNLyp95Sxk4\n+TcOF5Zq3r47/H1kS1Da9QZlAJxgSs30uY7endr5XEd5nDmpePbzl9S+4Ef3SGtabP6A+Gxbbw9F\n+LP3yjlsTj2n+nxwi0Eu7+ncrD5/jDuXWwZmOi33/Nm+hdOwR4y+mPj0JbVkW/JqBw+MTV3L/hMl\nLNh0UPP23eHEqfB5WVIGwAn5Ca19rqPXJbdyQKa5TBFZxYqs1/naeBb/3F2z6aXsHHMKROv4OEfv\nz/FZN3cRQNu/XwlYewr3WWHq5NP9rXqfy+VjaqYyOjd0zxOtRWqCZuEpxvXT3lBoTYUHjhDP/Gl2\nXZZSUlAc2JD0nuLSAAghWgohFgshtgohNgsh7rfIJwoh9gsh1ls+l1jd87gQIlsIsV0IcZGVfKhF\nli2ECPn/9abNfU8q3rh+Peadv4jMYc+5Vb7/Zbcy/NkfaNMoqVpmSmxiUy69gQphEI1sbVR7N273\n+79hb++HWdFrstd1Nkis2eDVukHg3TpHdfZ9qkifsqrWeVyjX1i0/YDP9Vbx3I/uT+voEswvb58s\nz6HnpF+qQ0aHIu7sA6gEHpZSrhVCJANrhBBV+Qz/I6V81bqwEKILMALoCjQHFgohql4r3gEuAHKB\nVUKIuVLKkJ0wM8T6Ht8HzDFsfCGl64XVx0XDZiCIoeonO1l/J+Mq/Rcp0oyKFxkyxCXVOq3XMINW\nVz7NrgXfeVzVxo73UrXn+fsrv+doyVENFPSMjMTTfK5jT9E2Epp/U0sWn/4b24pSgDN9rh9g4YGv\nwMOtBc/8sAVEGe8uzuaVa3tqoofWuBwBSCnzpJRrLceFwFbAmTvKlcAsKWWZlHI3kA30s3yypZT/\nSCnLgVmWsiHH6m5Pw7XTg60G3PILXPlOrZj0Sd0vI7F7zVrAuKdeYs8tf/tVjWOFJX6tvwpjjNoJ\n7Yq4hCS7cpHp+YMuqUlNWPM2qW3o16yft2o5pDzXeUKc4R19T4taYbI/516Adu+WJ+t96/E9+vrr\nSO40gf3Fu3l18VI+W2XffTWYeLQGIITIBHoDKyyie4QQG4QQ04QQVXMSLYB9VrflWmSO5HXbuF0I\nsVoIsfrIkSOeqFeNr6v/fc65BrqGQGTOVv2h940uizVp7N/NRhmmwAThMg24NyDthDOp5z1sVz6k\nY2O361hczzIr60FeZG/ZPO4eSg9d6vD62K5jvao3v6jmN+7o965P2m5XHij0SdsAKCGX6Xvv4eUt\ntvsngo3bBkAIkQR8AzwgpTwJ/BdoC/QC8oCq/HX2VoakE3ltgZQfSCmzpJRZjRo1snOLawpLw3+H\nnidoNVUVbGK7h+SAMKRo2KwVWy773qc6ut40hRUtbiJz8A0aaeUYvS6GnY/ZX59YNWoV+hjvotFM\n+/vL6uMvN/zuVR2BQsuIsFrjlgEQQsRifvjPkFJ+CyClPCSlNEopTcCHmKd4wPxmb716mgEccCIP\nOIXdb3J6PUDelYq6NAvNedJQo4HBty9o48ZN6f+vNxHeprv0gRcGvQDAhDMmYNB7v9Fy7dGaoGxl\nJscj/qe+W0frJ7/kz12BT9RSlSfYaApdTyB3vIAEMBXYKqV83UpunZvvamCT5XguMEIIES+EaA20\nB1YCq4D2QojWQog4zAvFc7Xphmec7G9/GB2+ePZA+Nbo2tdbEbqUJ2qXFjPQXNbmMjaO3cjwDsN9\nqmfTiT+rj/WJOxyW++bARJI6PMctM+d43MbFHz9P9+ndOVXm3QNcl2Ce8c4XK6plUobWaMCdEcBA\nYDRwbh2Xz5eFEBuFEBuAc4AHAaSUm4HZwBbgZ+Buy0ihErgHWIB5IXm2pWzokeK7+2dA0bk3jB5T\nfyo9mM3s9Hv8rFANJ/sGrq1ooSKxOXeUP8BViZ95fO/ee4Iy6K7GH6ktq9wu7aFP/Mf8b8uapEJF\nZZWUVriObZUbY07w3uPld3zSr1DULEav2XPcp7q0xuWTQ0q5DPuvmA63hkopnwdsskJIKec5uy9g\nOPkSXlD2Mr+6+UANNz59yOqta2Jg2qx/6bOw5m3XBRVuk9mwHnS+nGfPae/RfT8Z+3FpuvOkLv6i\nf9P+NE9q7vC6NMYjdGV+1aGk3EhCnI4zZvVGZ2zI+luWuHVfXNpSzXQoqwyt9cmo3Als/fz/qLJ2\nUpedMnTim0cEMTGsvGQeS3q95rRYsQbugNGCXhfD+6Oz6J6RUkt+sMutdsvvijPvFk7r4VkCIy35\n6KKPmDTQcXKfkv0j/a7D+//bjslknoIx6txfE9AnZWumw6ID37guFECi0gA0qZ/A7eUPMr7iJjL7\nXBBsdSKefv0GMuSq2zCe7TijVcyF/s38FQ3kt7HvRTUr/V5MUlDW+vwAa+Q+0uj/BCtT91/P/hOB\n2dOy6WCuXXlBeeAXo50RlQZAJwQP3fcQV90+kSYZNfF+ghluORrQDXkMHv2HecO2MTeh9j4LQ0Pb\nBCQKzyhNsr92ld7xDNqUzaBx88zAKuQB4y72LaZRqHHDAvujrQ3H/gqwJs6JSgOAEHRqWp++pzWg\nVYOaN49XK68LolK+sfuOXcFWwTVCQGJDLunejC0tzT7o601tYGKBixsV7pCYms70ygt4qqJmw9E6\nOnL7WW1Y+ugQujSvH0TtnHN5l14BaSfYXjj5JaG1CBydBsCKyvjU6uObbr2H6bdovx0+ELRulh5s\nFWrxWcIoGO3Y9W7Q6VmcU/Ya/23r7zhG0UOnpvU5dvYL3DD6jmpZxkNLEUJwWsPgLP66Sz299jl8\n7XG4JK/6uGo9IJAE2wDVJTLdXTzAaDEA+2VDzmwbWg/RcOXXVvdz3ZgJoHe8Q3lQ+3Teufc6OjUN\nzTzI4cqDF3TgWF5O9Xmj+uGRvFx4uJcFYOLczUy8oqtH99y8qMbZwCglMV606wtG/Ovp5ClRPwKo\nn9aYJytuYXZ37ZNYhxKFXfzvZVHF+Tc/Q7yTh38VXZrXJyYMUiaGG+H4F/XmvfiTP3f61Ga5MfCJ\nW6QIrWQxETkCcPkDsPIDNcTqeP75//hVn1Ag+dp3A9aWPzb7KNxHJppjaK2gG/2DrIs/MbSYCXgf\ntPFEcQWJcbbhKI4UlhGni45344g0AC6Ji8JpB/VQjhri4+NpX/opZ3dsGjYGwJsHrj5pqx80gdOf\nX0iMgMTIckyyS3SYubpE6E5fTyiRKvZ+pJIYr+e7+4YwZVTfYKviNglxev6vr2d7QYTwdUHV/v36\npM2IBO02f4Uy0WkAoowiabsQuMjUOwiaKAJF1+Yp1IsLrxed4Z0Cu1M5+3CRjay0wkhCy8+od9pH\nAdUlWCgDEAX80ux2G1lySppf2jLGhra7oUJRxYtrH7ORRVsuEWUAooDTOtjG2Y/T++e/XnfXcr/U\nq4h84nSBnZbcX2abStUUAD/9f47Yjjx25O+m+/Tu7D4e2GitygBEEPvb3cDhBrZTO4kZ3WxkhTr/\njABI8FO9iognRgT/cfTjhjzXhXzkynlnMH7RVEwmU7Vs2I9XAHDF3Iv83r41wf+LKzSjxY3v0fj+\nJTbyTh062sh+aTSWZypGM/+CRdoqYQjdcAOKyGP7wUJN6yssK9a0Pkd8l/sGV39mzlHw/frA5Ny2\nR0QagNDabB2amEQsHxsvpiTBvwnlFQp/cskH0zSpZ/H2w/y+8whTl2/UpD532F6wDoAHvtYu34Cn\nhJebgMJj1tMRe2G2nri0M3H6GC7t0SxIiTkVCt/RwltHSsmd856isqgzCS2nB2wndWzKOqSUJLV/\nMUAt2qIMQIRTdoP9gGzpSfFMHtYjwNooFKFHSYWRuLQ/iUv703VhjRn81ruQ4rqcv4jMKSAnc0BH\nZOTPUf9hrAmQ1b+jynGgUDjjeLF3Sd81aTthdtDahgg1AM74b8ePg62C37mt4mEAHiz/d5A1UShC\nn4+X7Q5a2zFxx4LWNkSoAXAW9ubpkaGbFk8rFo67hMzSL5hjGuxW+WGJ0zVpt/TRvZrUo1AEknKT\nrV9+tBCRBiDEci4EnBapCbx3Yx9WP+WesZv9sP1css6Yl36LjcyQGMTJTIXCS/4u/iLYKtRi4bac\ngLUVkQZAAUO7NSM9Kd6tsjovYvL37aBy+CoiA4nJdaEActdPrwasLWUAFN7Rd2ywNVAoNEHK0DIA\n8emB2xcQoQYgyueAAkCThmnk3HOAnSblZaQIb46eKg22CjZszDsYkHYi0gBE+xpAoMhMT+Q3aY45\n/37l5UHWRqHwjuMlgQn/4AkPLJwQkHYi0gA44nt9YOONhxMvVNzg1X29zrwAgLb9L9FSHYUiYMTW\nD1z4B3cplycC0k5E7gR2NAAoFe4tikYjLTIy4ZDn9/W/eAzb2vbnvPYdNNdJofA3phCb/6/ihNwS\nkHaiagTQLMU2M5bCTEpaY7fLTkl+qNZ5pw4dVSJ4RVjy4foZwVYhqLg0AEKIlkKIxUKIrUKIzUKI\n++tcf0QIIYUQ6ZZzIYSYIoTIFkJsEEL0sSo7Vgix0/IJuBtJo5R6gW4ybChsdS73ld/jstym0ydz\n34PjA6CRQuF/3t7wcrBVcMjAKe/4vQ13RgCVwMNSys7AAOBuIUQXMBsH4ALAegvoxUB7y+d24L+W\nsmnABKA/0A+YIIRooFE/auFoETivWeTvAvaWM9o2ZK7pTJflul36b4iJqoGjQhEUTqa85/c2XP6S\npZR5Usq1luNCYCtQ5fv3H+Axak+7Xwl8Ks38BaQKIZoBFwG/SimPSSmPA78CQ7XrimtMQhfI5sKK\ndo2TyZl8abDVUCgUAcSjVzkhRCbQG1ghhLgC2C+lrJtYswWwz+o81yJzJK/bxu1CiNVCiNVHjhzx\nRD2rOhzIvaoturiz/AF+NPanbelnwVZFoVD4Gbe9gIQQScA3wAOYp4WeBC60V9SOTDqR1xZI+QHw\nAUBWVpZXHv2OpoCEFyEPoo2fTf342dQv2GooFIoA4NYIQAgRi/nhP0NK+S3QFmgN/C2EyAEygLVC\niKaY3+xbWt2eARxwIg8YzeorN1BXfHqLevgrFNGCO15AApgKbJVSvg4gpdwopWwspcyUUmZifrj3\nkVIexJxgcIzFG2gAUCClzAMWABcKIRpYFn8vtMgCRpP6hkA2F5ac1aERAG0bJQZZE0U0UlnUPtgq\nRBXujAAGAqOBc4UQ6y0fZ9s+5wH/ANnAh8BdAFLKY8CzwCrLZ5JFpjnSwVYwobxX3OK3h89mzt0D\ng62GIgrpkNqt+rg8/ywKtz8dRG08o3ViH9eFQgyXawBSymW4WD+1jAKqjiVwt4Ny04BpnqmoHTq1\nBOAWbRol2ZUfkSk0CrAuiuiiXpzOvMIISKkDU/js3bn4tCt4d8vaYKvhEVH1SiwbZAZbhbDla+NZ\nvNfj62CroYhwrN/RejXp4nZSo0BRnn8WPetfZvdah/QmVJzoHWCNfCMiDYAjL6CYhNTAKhJB9H/w\nS5665vRgq6GIcFoaBlQfJ8bpaZgYF0RtbPm/IRfx+dUv2shNlYmcmzmQs9NvC4JW3hORBsAR3mS+\nUsB95XfTMq2eivej8Dup+lZUnDSvA3RuVj/kvnNXd7GfZzuz+GWEEEy6PLy86KLKABj0aiewN8w1\nqQVhReAwlpjTjbZJzQiyJrY0MNiPXnPPELP3UkpCLMV7bfNlhyoRaQAcTgGpEYBn/PtPfjj7J75X\nHkGKAFJxbBAdKyYxvLvr2FShwsXdmwFgiNWxa/yDNtcrCkJzbSAi8wEoNKJJFy5vEmwlFNFEp6bJ\nQAy3n3FGsFXRjNKDVzK4dTtWlawLtio2KAOgUChChqt6taBT0/p0blY/2Kr4ROWpNugT/6Fwx3gW\n3n8xqw7/wao1wdbKlqgxAB9XXsTNwVZCoVA4RQgR9g9/gJK9twPwwz2DaNc4mTVHQnP6OWoMQHzW\nqGCroFAoIoTKws7ok7cCkFQ03Ob6mqfOZ/fRU3TPSDELpDIAAcNeKIiSOLWHVaFQaEOm8W7+OT6d\nimOD+equYTbXGybF0zCpJvhkeoL7KVcDSUR6AdmjT7fOwVZBoVCEMSUHrq0+nn3HmZQdHMbpLTrR\nsUmyy3vTDaHpTRGRIwB7qEigCoXCFy5te3b1cbIh1qMMeomxyZTkjiIhI7SS0EfkCMDePoC0ENtS\nrlAo3MP6zTuYvD78LK/vbZQUT2Vhdw210YaINAD2MMSqXcAKRThSWdA3qO0XbptE4dbnfQolk1Iv\nli2TLtJQK22IyCkgYSwPtgoKhUIjgh7DS8ax4AHv3/6rqBcXeo/biBwBSKHe9hWKSOHn++0HYHNG\n0Y7xlB2+gJL911FZfJrXbZfmXQVAx6auF3rdwVjaVJN6tCIiDUBKPZX7V6GIFNo1tp+gyBEluTcg\njYmU559H5ck+DKj/L6/brjjRnxm39ff6fhvd9t6KrAydJDcRaQB0uojslkIRlXgaErqysCe/P3ZO\n9Xnr+t7nGd70zFAGtkv3+v66PHJ+FkU7QyfNZUQ+KfXBnjNUKBRBpWVaPXImX0rO5EtJS4z1up6k\neG3n7Yf3zaBto0RN6/SFiDQAjesb+NmoslcpFNHIs1d1q3XubVKZ4n1jtFCnFk3qG1j08BDN6/WW\niDQAAHdW2MbkVigU4U9J7o1Or48eUHvRt3FyPMbS5h61IWUMxqIuHusWbkSsAeh7Wk3mnlyp3Rye\nQqEILqYKz6KFDu+b4dJo1KXyZDfXhTSgOOdOivfeQmneVZzafU9A2rQmYg3AN/+uySb0TIX2QzmF\nQhEcemTUx1hmP7ha2dEhNjIhBNLoWSgYU5lnIwZPqTjRBwBjWWOMpzpQcWIAptLAp8CMWANgza+m\nrGCroFAofKAo+9Hq4/5tUjE5mNIpP2J/t23vjJryjoyHNW0atGDJI0M8U9IDSvOGUZT9GK8PP4NB\n7dIRAlqkJvitPUdEhQFQKBThjaxoWH3czNARWWm7N6D0wDDA/oLvK8N7Vh8bT7V12d7tWReTme4/\nb52bB7ZFVqRxde8WfH5bf3a/eCmvXdezVhlp8v/O4agwAFNuCM2EzAqFwnP0MbF2k6wby5rw5CX2\nw763a5xE0Y7xGEtaUp7vOqxD24b+Dd884fKu5Ey+tJaH0oA2DWuVKdk/0q86QITGAqrLFT39O5+n\nUCj8y6onz+fcb83HGQ0SSBS24R1G9T+Nf53VxmEd0phIcc7d/lJREwq3TibGsNcyxaVGAAqFQkGj\n5JrwLi3T6nFaQ9twCu4kZgHXoSXKj/enTRA3a5lKWwF6Ts9s4LKsr7g0AEKIlkKIxUKIrUKIzUKI\n+y3yZ4UQG4QQ64UQvwghmlvkQggxRQiRbbnex6qusUKInZbPWP91S6FQRCpt0hMZ2NbWtbtNY+cP\n7b+fvpDB7dOZNtbxJtGi7EcpO3g1yQbvdw/7wp1nm9cn/nN9T2bfcYbf23NnBFAJPCyl7AwMAO4W\nQnQBXpFS9pBS9gJ+BKoCXFwMtLd8bgf+CyCESAMmAP2BfsAEIYT/TZxCoYgohBB2p3qEgwXgKlLq\nxfLZrf1pZWf0UI0MzoO/inEXdyJn8qVc3TvD6x3MnuDSAEgp86SUay3HhcBWoIWU8qRVsUSozsR+\nJfCpNPMXkCqEaAZcBPwqpTwmpTwO/AoM1bAvCoUiSrAXo0cvfJszN5a0QFZ6tsks3PHoLyaEyAR6\nAyss588DY4ACoCr8Xgtgn9VtuRaZI7nfOK/sFepTzBx/NqJQKAKOvZfj01Ja+1RnxYnT6dMqlem3\n9POpnnDC7UVgIUQS8A3wQNXbv5TySSllS2AGULWP2d64RTqR123ndiHEaiHE6iNHjrirnl12yRas\nk96HglUoFKFJjBCc2odjQCMAAAfuSURBVPVQLVm8zrc8IJWFXXntul5Bm/8PBm4ZACFELOaH/wwp\n5bd2inwBDLMc5wItra5lAAecyGshpfxASpklpcxq1KiRO+opFIooI1YXg6nc9Y5eT5DGZOL00eUY\n6Y4XkACmAlullK9bya1fra8AtlmO5wJjLN5AA4ACKWUesAC4UAjRwLL4e6FFplAoFF4hTTVv64lx\nvqeCTUmInrd/cG8NYCAwGtgohFhvkT0B3CqE6AiYgD3AnZZr84BLgGygGLgZQEp5TAjxLLDKUm6S\nlPKYJr1QKBRRiamsMbqE/QDofcwE+NGYLM0TwIQ6LnsrpVyG/fn7eQ7KS8Dudjsp5TRgmicK+sLK\nJ85Dp7KDKRQRi6m8YbUB8AQpYxDCVEt2Vofom3KO6AmvxvUNNExSCeIVikjkh3sGUXroSq/uNdmJ\nCBpt8/8Q4QZAoVBELt0zUsDoXcgGY0krjbUJT5QBUCgUYYGxtKlmdVUcP9N1oSggulY8FApF2FK8\n506EvlCTukxl2hmTcEYZAIVCER6YDMhyz1I7KpyjDIBCoQhrinaOw9fZ7NKDV2ijTJih1gAUCkXY\n0q5xErIy1asgbpXFmdXH0bomoAyAQqEIC5INthMWLw/v4VVdTesbKD1wna8qhT1qCkihUIQF68Zf\nQKWpdvzIPq28Syky6/YBDHl1MZVF7Sg7eoEW6oUlygAoFIqwQK+LQe97uB8AMtMTAUHJvtu0qTBM\nUVNACoVCEaUoA6BQKBRRijIACoUi6tk6KTqz06o1AIVCEdZ0bJJMp2bJHt/XKq0ee48VA5CgQS6B\ncEQZAIVCEdYsePAsr+6bdlMW57/+P421CS/UFJBCoYhK6sWp919lABQKRVTTLCV64wspA6BQKKKS\nGGHOFmiIjc75f1BrAAqFIkppUj+eRy7swBU9WwRblaChDIBCoYhKhBDcc277YKsRVNQUkEKhUEQp\nygAoFApFlKIMgEKhUEQpygAoFApFlKIMgEKhUEQpygAoFApFlKIMgEKhUEQpygAoFApFlCKklK5L\nBQkhxBFgTxCaTgeOBqHdUEH1P3r7H819h8jp/2lSykauCoW0AQgWQojVUsqsYOsRLFT/o7f/0dx3\niL7+qykghUKhiFKUAVAoFIooRRkA+3wQbAWCjOp/9BLNfYco679aA1AoFIooRY0AFAqFIkqJaAMg\nhHhQCLFZCLFJCDFTCGEQQrQWQqwQQuwUQnwphIizlI23nGdbrmda1fO4Rb5dCHGRlXyoRZYthBgX\n+B46Rwhxv6Xvm4UQD1hkaUKIXy39/1UI0cAiF0KIKZa+bBBC9LGqZ6yl/E4hxFgreV8hxEbLPVOE\nsKRYChJCiGlCiMNCiE1WMr/311EbgcZB/6+1/P+bhBBZdcp79L325rcTSBz0/xUhxDbL//EcIUSq\n1bWI6r9XSCkj8gO0AHYDCZbz2cBNln9HWGTvAf+2HN8FvGc5HgF8aTnuAvwNxAOtgV2AzvLZBbQB\n4ixlugS731b97wZsAuphTvyzEGgPvAyMs5QZB7xkOb4EmA8IYACwwiJPA/6x/NvActzAcm0lcIbl\nnvnAxUHu81lAH2CTlczv/XXURoj0vzPQEVgCZFnJPf5ee/rbCZH+XwjoLccvWf3/R1z/vfqbBVsB\nP34ZWgD7LD9kPfAjcBHmTR5VX4gzgAWW4wXAGZZjvaWcAB4HHreqd4Hlvup7LfJa5YL9Aa4FPrI6\nHw88BmwHmllkzYDtluP3gRusym+3XL8BeN9K/r5F1gzYZiWvVS6I/c6s8wDwe38dtREK/beSL6G2\nAfDoe235LXj02wml/luuXQ3MiOT+e/qJ2CkgKeV+4FVgL5AHFABrgBNSykpLsVzMhgJqDAaW6wVA\nQ2t5nXscyUOFTcBZQoiGQoh6mN94WwJNpJR5AJZ/G1vKe9rPFpbjuvJQIxD9ddRGKONp/xvi+W8n\n1LgF88gNorP/NkSsAbDMw16JeXjXHEgELrZTtMoNyt78tfRCHhJIKbdiHvL+CvyMeShb6eSWiOq/\nG0Rbf+uiZf9D/m8jhHgS8/d/RpXITrGI7b8jItYAAOcDu6WUR6SUFcC3wJlAqhBCbymTARywHOdi\nfkPGcj0FOGYtr3OPI3nIIKWcKqXsI6U8C3NfdgKHhBDNACz/HrYU97SfuZbjuvJQIxD9ddRGKONp\n/4/i+W8nJLAs5F8GjJKWeRqiqP/OiGQDsBcYIISoZ/HWOA/YAiwGhlvKjAW+txzPtZxjuf6b5csy\nFxhhWelvjXkhdSWwCmhv8QyIw7z4MzcA/XIbIURjy7+tgGuAmdTuZ93+j7F4xwwACizTGQuAC4UQ\nDSyjqgsxz33mAYVCiAGWv+8Yq7pCiUD011EboYxH32vLb8HT307QEUIMBf4PuEJKWWx1KSr675Jg\nL0L48wM8A2zDPB/+GeYV/zaY/6Ozga+AeEtZg+U823K9jVU9T2L2DNiOlacL5nn1HZZrTwa7v3b6\n/ztmo/c3cJ5F1hBYhHk0sAhIs8gF8I6lLxupvWB4i+Xvkg3cbCXPsvxtdwFvE+SFL8wGLg+owPxW\ndmsg+uuojRDp/9WW4zLgELUXOD36Xnvz2wmB/mdjnp9fb/m8F6n99+ajdgIrFApFlBLJU0AKhUKh\ncIIyAAqFQhGlKAOgUCgUUYoyAAqFQhGlKAOgUCgUUYoyAAqFQhGlKAOgUCgUUYoyAAqFQhGl/D8v\nGG5endsU0wAAAABJRU5ErkJggg==\n",

      "text/plain": [

       "<matplotlib.figure.Figure at 0x7f4f1177e290>"

      ]

     },

     "metadata": {},

     "output\_type": "display\_data"

    }

   ],

   "source": [

    "plt.plot(df['last'], label='Actual')\n",

    "plt.plot(pd.DataFrame(trainPredictPlot, columns=[\"close\"], index=df.index).close, label='Training')\n",

    "plt.plot(pd.DataFrame(testPredictPlot, columns=[\"close\"], index=df.index).close, label='Testing')\n",

    "plt.legend(loc='best')\n",

    "plt.show()"

   ]

  }

 ],

 "metadata": {

  "kernelspec": {

   "display\_name": "Python [conda root]",

   "language": "python",

   "name": "conda-root-py"

  },

  "language\_info": {

   "codemirror\_mode": {

    "name": "ipython",

    "version": 2

   },

   "file\_extension": ".py",

   "mimetype": "text/x-python",

   "name": "python",

   "nbconvert\_exporter": "python",

   "pygments\_lexer": "ipython2",

   "version": "2.7.11"

  }

 },