A simple math model based on data analysis

To develop a model to accurately forecast the inventory demand, “clean” data is necessary. Therefore, I first find out the abnormal ones in the historical sales data. For example, for some unknown reason, the number of a product sold is smaller than returned. In such case, the forecasted number for such product in this store will be temporarily set to 0. Over 615 thousands of product&store combination was found to be abnormal.

```
f = open("../Data/train.csv", "r")
f.readline()
total=0
path = 'wierd.csv'
out = open(path, "w")
out.write("Semana,Agencia_ID,Canal_ID,Ruta_SAK,Cliente_ID,Producto_ID,Venta_uni_hoy,Venta_hoy,Dev_uni_proxima,Dev_proxima,Demanda
while 1:
    line = f.readline().strip()
    total += 1

    if total % 1000000 == 0:
        print (total/1000000)

    if line == '':
        break

    arr = line.split(",")
    week = int(arr[0])
    agency = arr[1]
    canal_id = arr[2]
    ruta_sak = arr[3]
    cliente_id = arr[4]
    producto_id = arr[5]
    vuh = int(arr[6])
    vh = arr[7]
    dup = int(arr[8])
    dp = arr[9]
    target = arr[10]

    if vuh-dup<0:
        out.write(str(week)+','+agency+','+canal_id+','+ruta_sak+','+cliente_id+','+producto_id+','+str(vuh)+','+vh+','+str(dup)

out.close()
f.close()
```

```
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
wierd = pd.read_csv('wierd.csv', usecols=['Cliente_ID'])
print ("wierd",len(wierd))
```

```
wierd 615828
```

Find the data that are suitable for mathematical modeling.

I assumed that historical sales data good for modeling should have non-zero values. So I extracted such data first.

```
path='consecutive_demand.csv'
out=open(path,"w")
f = open("../Data/Demandallweeks.csv", "r")
f.readline()
total=0
out.write("id,superid,Cliente_ID,Producto_ID,week3,week4,week5,week6,week7,week8,week9,sum\n")
```

```
while 1:
    line = f.readline().strip()
    if line == '':
        break

    arr = line.split(",")
    ID = arr[0]
    superid = int(arr[1])
    cliente_id = int(arr[2])
    producto_id = int(arr[3])
    week3 = int(arr[4])
    week4 = int(arr[5])
    week5 = int(arr[6])
    week6 = int(arr[7])
    week7 = int(arr[8])
    week8 = int(arr[9])
    week9 = int(arr[10])
    Sum = week3+week4+week5+week6+week7+week8+week9
    consecutive = week3*week4*week5*week6*week7*week8*week9
    if consecutive !=0:
        out.write(str(ID)+','+str(superid)+','+str(cliente_id)+','+str(producto_id)+','+str(week3)+','+str(week4)+','
               +str(week5)+','+str(week6)+','+str(week7)+','+str(week8)+','+str(week9)+','+str(Sum))
        out.write("\n")


out.close()
f.close()
```

Then, based on "common sense" of economic activities, I assume the number of a product sold in a store in 7 weeks could follow a wave-like structure. In addition to that, based on several steps of data analysis, I found the data can be sorted into three types, depending on the first and second derivatives: linear increase/decrease, sudden increase/decrease, other complicated structures. And I developed simple math models to describe these for data.

```
import gc
import math
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from numpy import sin,cos,exp,pi
```

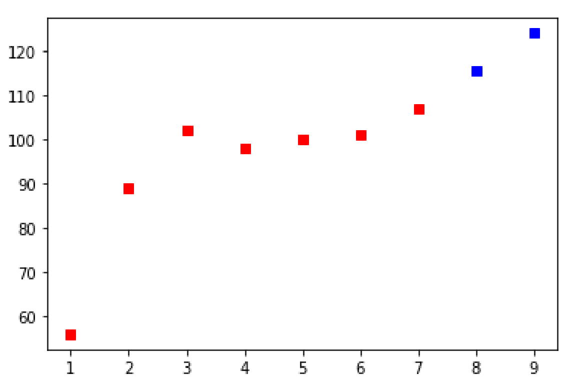
```
f = open("consecutive_demand.csv", "r")


#f.readline()
a=1
b=1
y1index=0.1
y2index=0.1
k=0.01
while 1:
    avetotal=0.0
    ave = np.array([0.0,0.0,0.0,0.0,0.0,0.0])

    line = f.readline().strip()
    arr = line.split(",")
#    print arr[0]
#    if int(arr[0]) == 3:
    x = np.array([1,2,3,4,5,6,7])
    y_initial = np.array([int(arr[1]),int(arr[2]),int(arr[3]),int(arr[4]),int(arr[5]),int(arr[6]),int(arr[7])])
    y=y_initial/np.mean(y_initial)
    for i in range(7):
        avetotal += y[i]
        if i>0:
            ave[i-1] = avetotal/(i+1)
    x_ave = np.array([ave[0],ave[1],ave[2],ave[3],ave[4]])
    x_0 = np.array([int(arr[2]),int(arr[3]),int(arr[4]),int(arr[5]),int(arr[6])])
    y_1 = np.diff(y)
    y_2 = np.diff(y_1)

    y1index=-np.sum(y_1[y_1>0])/(np.sum(y_1[y_1<0])-0.000001)
    if np.sum(y_1[y_1>0])<abs(np.sum(y_1[y_1<0])):
        y1index=-(np.sum(y_1[y_1<0]))/(np.sum(y_1[y_1>0])+0.000001)

    y2index=-np.sum(y_2[y_2>0])/(np.sum(y_2[y_2<0])-0.000001)
    if np.sum(y_2[y_2>0])<abs(np.sum(y_2[y_2<0])):
        y2index=-(np.sum(y_2[y_2<0]))/(np.sum(y_2[y_2>0])+0.000001)
```

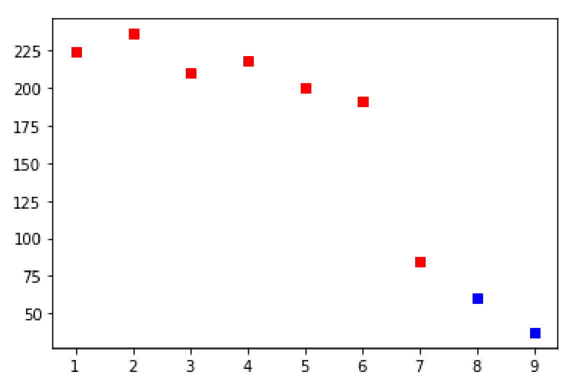
```
print (y1index,y2index,np.sum(y_2[y_2>0]),abs(np.sum(y_2[y_2<0])))

uniquen,counts=np.unique(y_initial,return_counts=True)

if np.amax(counts)>4:
    print ("const case",y_initial)
    x10 = np.mean(y_initial)
    x11 = np.mean(y_initial)
    plt.plot(x,y_initial,'rs')

elif y1index>5:
    print ("linear case",y_initial)
    x10 = y_initial[6]+np.mean(y_1)*np.mean(y_initial)
    x11 = x10+np.mean(y_1)*np.mean(y_initial)
    plt.plot(x,y_initial,'rs')

elif y2index>5:
    print ("acc case",y_initial)
    def fit_func(x, A, B, w):
        return A*sin(w*x)+B*cos(w*x)+1
    params = curve_fit(fit_func, x, y)
    [A, B, w]=params[0]
    print (A,B,w)
    xnew = np.linspace(x[0], x[-1], 500)
    ynew = np.array(fit_func(xnew,A, B, w)*np.mean(y_initial))
    x10 = fit_func(8,A, B, w)*np.mean(y_initial)
    x11 = fit_func(9,A, B, w)*np.mean(y_initial)
    plt.plot(x,y_initial,'rs',xnew,ynew)


else:
    print ("normal case",y_initial)
    def fit_func(x, A, B, w,k,a,b):
        return A*sin(pi*x/1+a)+B*cos(pi*x/2.71828+b)+1+np.mean(y_1)
    params = curve_fit(fit_func, x, y)
    [A, B, w,k,a,b]=params[0]
    print (A, B, w,k,a,b,np.mean(y_1))
    xnew = np.linspace(x[0], x[-1], 500)
    ynew = np.array(fit_func(xnew,A, B, w,k,a,b)*np.mean(y_initial))
    x10 = fit_func(8,A, B, w,k,a,b)*np.mean(y_initial)
    x11 = fit_func(9,A, B, w,k,a,b)*np.mean(y_initial)
    plt.plot(x,y_initial,'rs',xnew,ynew)

x1 = np.array([8,9])
y1 = np.array([x10,x11])
plt.plot(x1,y1,'bs')
plt.show()
```

```
f.close()
gc.collect()
```

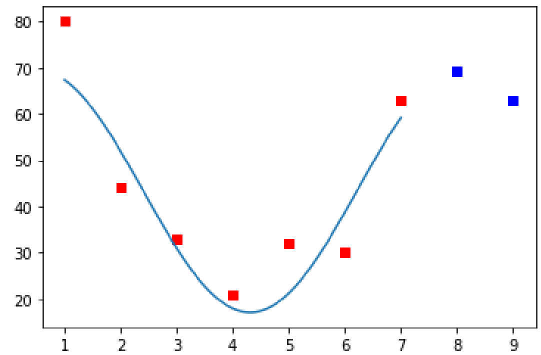
The red dots represent historical data of past 7 weeks and the 2 blue dots are prediction data for the following two weeks.

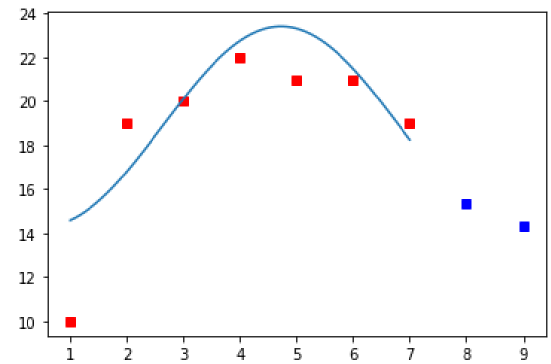
1) Linear increase/decrease





2) sudden increase/decrease





3) other complicated structures