In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
pd.set_option('display.max_rows', 50000)
pd.set_option('display.max_columns', 500)
pd.set_option('display.float_format', lambda x: '%.5f' % x)
```

In [3]:
```
DDD_monthly=pd.read_excel('S4 Table_v1_copy.xlsx',sheet_name='Sheet5_1')
DiD_monthly=pd.read_excel('S4 Table_v1_copy.xlsx',sheet_name='Sheet5_2')

print(DDD_monthly.shape)
print(DiD_monthly.shape)
```

```
(60, 33)
(60, 33)
```

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

Out[4]:

	year	month	J01AA	J01BA	J01CA	J01CE	J01CF	J01C
0	2011	1	808432.00000	82178.00000	2293639.73300	1581290.86600	127553.93000	5972.500
1	2011	2	616620.00000	45122.24900	1157493.30100	696282.87000	61136.10900	2862.500
2	2011	3	802755.00000	86423.58300	1548117.08800	1330648.93200	90434.89500	7289.500
3	2011	4	736725.66700	92561.66700	1671628.51000	1363893.90600	115792.60800	6653.000
4	2011	5	1038556.00000	73520.33400	1444498.20400	1482620.94500	110954.71600	8070.000

In [5]:
```
DiD_monthly.tail()
```

Out[5]:

	year	month	J01AA	J01BA	J01CA	J01CE	J01CF	J01CG	J01CR	J01DA	J01DB
55	2015	8	0.31484	0.00607	0.36936	0.48140	0.02199	0.00029	1.74781	0.01642	0.30342
56	2015	9	0.30878	0.00743	0.43295	0.46805	0.02399	0.00030	3.07244	0.02225	0.32754
57	2015	10	0.29481	0.00564	0.30297	0.36127	0.01725	0.00031	1.25334	0.01103	0.23875
58	2015	11	0.30496	0.00602	0.41777	0.50705	0.01969	0.00022	2.13530	0.01905	0.30164
59	2015	12	0.39328	0.00886	0.48467	0.41748	0.02152	0.00045	2.75306	0.02241	0.32176

In [6]:
```
DDD_monthly.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60 entries, 0 to 59
Data columns (total 33 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   year    60 non-null     int64  
 1   month   60 non-null     int64  
 2   J01AA   60 non-null     float64
 3   J01BA   60 non-null     float64
 4   J01CA   60 non-null     float64
 5   J01CE   60 non-null     float64
 6   J01CF   60 non-null     float64
 7   J01CG   60 non-null     float64
 8   J01CR   60 non-null     float64
 9   J01DA   60 non-null     float64
 10  J01DB   60 non-null     float64
 11  J01DC   60 non-null     float64
 12  J01DD   60 non-null     float64
 13  J01DE   60 non-null     float64
 14  J01DF   60 non-null     float64
 15  J01DH   60 non-null     float64
 16  J01DI   60 non-null     float64
 17  J01EA   37 non-null     float64
 18  J01EB   60 non-null     int64  
 19  J01EC   60 non-null     float64
 20  J01EE   27 non-null     float64
 21  J01FA   60 non-null     float64
 22  J01FF   60 non-null     float64
 23  J01FG   60 non-null     float64
 24  J01GA   60 non-null     int64  
 25  J01GB   60 non-null     float64
 26  J01MA   60 non-null     float64
 27  J01XA   60 non-null     float64
 28  J01XB   34 non-null     float64
 29  J01XC   60 non-null     float64
 30  J01XD   60 non-null     float64
 31  J01XE   60 non-null     float64
 32  J01XX   60 non-null     float64
dtypes: float64(29), int64(4)
memory usage: 15.6 KB
```

In [7]:
```
DiD_monthly.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60 entries, 0 to 59
Data columns (total 33 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   year    60 non-null     int64  
 1   month   60 non-null     int64  
 2   J01AA   60 non-null     float64
 3   J01BA   60 non-null     float64
 4   J01CA   60 non-null     float64
 5   J01CE   60 non-null     float64
 6   J01CF   60 non-null     float64
 7   J01CG   60 non-null     float64
 8   J01CR   60 non-null     float64
 9   J01DA   60 non-null     float64
 10  J01DB   60 non-null     float64
 11  J01DC   60 non-null     float64
 12  J01DD   60 non-null     float64
 13  J01DE   60 non-null     float64
 14  J01DF   60 non-null     float64
 15  J01DH   60 non-null     float64
 16  J01DI   60 non-null     float64
 17  J01EA   60 non-null     float64
 18  J01EB   60 non-null     float64
 19  J01EC   60 non-null     float64
 20  J01EE   60 non-null     float64
 21  J01FA   60 non-null     float64
 22  J01FF   60 non-null     float64
 23  J01FG   60 non-null     float64
 24  J01GA   60 non-null     float64
 25  J01GB   60 non-null     float64
 26  J01MA   60 non-null     float64
 27  J01XA   60 non-null     float64
 28  J01XB   60 non-null     float64
 29  J01XC   60 non-null     float64
 30  J01XD   60 non-null     float64
 31  J01XE   60 non-null     float64
 32  J01XX   60 non-null     float64
dtypes: float64(31), int64(2)
memory usage: 15.6 KB
```

In [8]:
```
DDD_monthly.dtypes
```

Out[8]:
```
year       int64
month      int64
J01AA    float64
J01BA    float64
J01CA    float64
J01CE    float64
J01CF    float64
J01CG    float64
J01CR    float64
J01DA    float64
J01DB    float64
J01DC    float64
J01DD    float64
J01DE    float64
J01DF    float64
J01DH    float64
J01DI    float64
J01EA    float64
J01EB      int64
J01EC    float64
J01EE    float64
J01FA    float64
J01FF    float64
J01FG    float64
J01GA      int64
J01GB    float64
J01MA    float64
J01XA    float64
J01XB    float64
J01XC    float64
J01XD    float64
J01XE    float64
J01XX    float64
dtype: object
```

In [9]:
```
DDD_monthly.columns
```

Out[9]:
```
Index(['year', 'month', 'J01AA', 'J01BA', 'J01CA', 'J01CE', 'J01CF', 'J01
CG',
       'J01CR', 'J01DA', 'J01DB', 'J01DC', 'J01DD', 'J01DE', 'J01DF', 'J0
1DH',
       'J01DI', 'J01EA', 'J01EB', 'J01EC', 'J01EE', 'J01FA', 'J01FF', 'J0
1FG',
       'J01GA', 'J01GB', 'J01MA', 'J01XA', 'J01XB', 'J01XC', 'J01XD', 'J0
1XE',
       'J01XX'],
      dtype='object')
```

In [10]:
```
DiD_monthly.columns
```

Out[10]:
```
Index(['year', 'month', 'J01AA', 'J01BA', 'J01CA', 'J01CE', 'J01CF', 'J01
CG',
       'J01CR', 'J01DA', 'J01DB', 'J01DC', 'J01DD', 'J01DE', 'J01DF', 'J0
1DH',
       'J01DI', 'J01EA', 'J01EB', 'J01EC', 'J01EE', 'J01FA', 'J01FF', 'J0
1FG',
       'J01GA', 'J01GB', 'J01MA', 'J01XA', 'J01XB', 'J01XC', 'J01XD', 'J0
1XE',
       'J01XX'],
      dtype='object')
```

In [11]:
```
DDD_monthly=DDD_monthly.astype({'year':'int64', 'month':'int64', 'J01AA':'fl
                    'J01CE':'float64', 'J01CF':'float64', 'J01CG':'float64',
                    'J01DB':'float64', 'J01DC':'float64', 'J01DD':'float64',
                    'J01DH':'float64','J01DI':'float64', 'J01EA':'float64',
                    'J01EB':'float64', 'J01EC':'float64', 'J01EE':'float64',
                    'J01FG':'float64', 'J01GA':'float64', 'J01GB':'float64',
                    'J01XB':'float64', 'J01XC':'float64', 'J01XD':'float64',
DDD_monthly.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60 entries, 0 to 59
Data columns (total 33 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   year    60 non-null     int64  
 1   month   60 non-null     int64  
 2   J01AA   60 non-null     float64
 3   J01BA   60 non-null     float64
 4   J01CA   60 non-null     float64
 5   J01CE   60 non-null     float64
 6   J01CF   60 non-null     float64
 7   J01CG   60 non-null     float64
 8   J01CR   60 non-null     float64
 9   J01DA   60 non-null     float64
 10  J01DB   60 non-null     float64
 11  J01DC   60 non-null     float64
 12  J01DD   60 non-null     float64
 13  J01DE   60 non-null     float64
 14  J01DF   60 non-null     float64
 15  J01DH   60 non-null     float64
 16  J01DI   60 non-null     float64
 17  J01EA   37 non-null     float64
 18  J01EB   60 non-null     float64
 19  J01EC   60 non-null     float64
 20  J01EE   27 non-null     float64
 21  J01FA   60 non-null     float64
 22  J01FF   60 non-null     float64
 23  J01FG   60 non-null     float64
 24  J01GA   60 non-null     float64
 25  J01GB   60 non-null     float64
 26  J01MA   60 non-null     float64
 27  J01XA   60 non-null     float64
 28  J01XB   34 non-null     float64
 29  J01XC   60 non-null     float64
 30  J01XD   60 non-null     float64
 31  J01XE   60 non-null     float64
 32  J01XX   60 non-null     float64
dtypes: float64(31), int64(2)
memory usage: 15.6 KB
```

In [12]:
```
DiD_monthly=DiD_monthly.astype({'year':'int64', 'month':'int64', 'J01AA':'fl
                    'J01CE':'float64', 'J01CF':'float64', 'J01CG':'float64',
                    'J01DB':'float64', 'J01DC':'float64', 'J01DD':'float64',
                    'J01DH':'float64','J01DI':'float64', 'J01EA':'float64',
                    'J01EB':'float64', 'J01EC':'float64', 'J01EE':'float64',
                    'J01FG':'float64', 'J01GA':'float64', 'J01GB':'float64',
                    'J01XB':'float64', 'J01XC':'float64', 'J01XD':'float64',
DiD_monthly.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60 entries, 0 to 59
Data columns (total 33 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   year    60 non-null     int64  
 1   month   60 non-null     int64  
 2   J01AA   60 non-null     float64
 3   J01BA   60 non-null     float64
 4   J01CA   60 non-null     float64
 5   J01CE   60 non-null     float64
 6   J01CF   60 non-null     float64
 7   J01CG   60 non-null     float64
 8   J01CR   60 non-null     float64
 9   J01DA   60 non-null     float64
 10  J01DB   60 non-null     float64
 11  J01DC   60 non-null     float64
 12  J01DD   60 non-null     float64
 13  J01DE   60 non-null     float64
 14  J01DF   60 non-null     float64
 15  J01DH   60 non-null     float64
 16  J01DI   60 non-null     float64
 17  J01EA   60 non-null     float64
 18  J01EB   60 non-null     float64
 19  J01EC   60 non-null     float64
 20  J01EE   60 non-null     float64
 21  J01FA   60 non-null     float64
 22  J01FF   60 non-null     float64
 23  J01FG   60 non-null     float64
 24  J01GA   60 non-null     float64
 25  J01GB   60 non-null     float64
 26  J01MA   60 non-null     float64
 27  J01XA   60 non-null     float64
 28  J01XB   60 non-null     float64
 29  J01XC   60 non-null     float64
 30  J01XD   60 non-null     float64
 31  J01XE   60 non-null     float64
 32  J01XX   60 non-null     float64
dtypes: float64(31), int64(2)
memory usage: 15.6 KB
```

In [13]:
```
DDD_monthly.describe()
```

Out[13]:

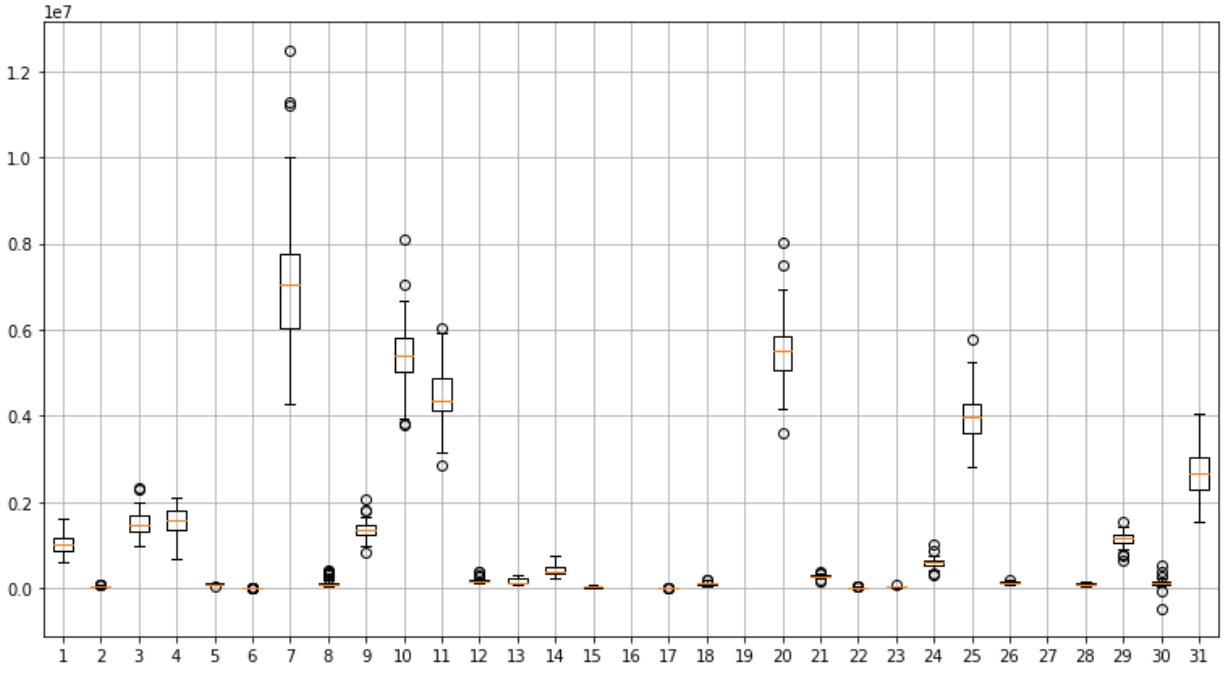
	year	month	J01AA	J01BA	J01CA	J01CE	J01
count	60.00000	60.00000	60.00000	60.00000	60.00000	60.00000	60.000
mean	2013.00000	6.50000	1011680.66390	37748.65650	1508265.79248	1545272.98183	93518.475
std	1.42615	3.48118	209879.19055	19488.94360	275238.30976	335184.90944	16559.393
min	2011.00000	1.00000	616620.00000	10993.16700	960667.05300	696282.87000	48937.393
25%	2012.00000	3.75000	861723.45850	25999.41700	1323105.54800	1358293.02950	86436.839
50%	2013.00000	6.50000	1025744.91650	31491.45750	1448611.05000	1567270.16500	93480.893
75%	2014.00000	9.25000	1172548.75025	45965.85550	1675403.09075	1813291.75250	104994.562
max	2015.00000	12.00000	1599060.00000	92561.66700	2311695.44900	2118915.42800	129818.715

In [14]:
```
DiD_monthly.describe()
```

Out[14]:

	year	month	J01AA	J01BA	J01CA	J01CE	J01CF	J01CG	J01CR
count	60.00000	60.00000	60.00000	60.00000	60.00000	60.00000	60.00000	60.00000	60.00000
mean	2013.00000	6.50000	0.23520	0.00856	0.34987	0.35923	0.02163	0.00041	1.66723
std	1.42615	3.48118	0.05287	0.00393	0.06673	0.08577	0.00363	0.00043	0.40844
min	2011.00000	1.00000	0.13945	0.00270	0.24626	0.15747	0.01254	0.00001	1.07863
25%	2012.00000	3.75000	0.19628	0.00604	0.30239	0.30079	0.01954	0.00017	1.41193
50%	2013.00000	6.50000	0.23688	0.00737	0.33060	0.34508	0.02200	0.00026	1.65412
75%	2014.00000	9.25000	0.27119	0.00992	0.38789	0.42115	0.02400	0.00041	1.80603
max	2015.00000	12.00000	0.39328	0.01954	0.55308	0.55416	0.02962	0.00176	3.07244

In [15]:
```
x1=DDD_monthly.loc[:,['J01AA', 'J01BA', 'J01CA', 'J01CE', 'J01CF', 'J01CG',
       'J01CR', 'J01DA', 'J01DB', 'J01DC', 'J01DD', 'J01DE', 'J01DF', 'J01DI',
       'J01EA', 'J01EB', 'J01EC', 'J01EE', 'J01FA', 'J01FF', 'J01GA',
       'J01GB', 'J01MA', 'J01XA', 'J01XB', 'J01XC', 'J01XD', 'J01XX']]

plt.figure(figsize=(13,7))
plt.boxplot(x1)
plt.grid()
plt.show()
```



In [17]:
```
number_of_DDD_ATC_4=pd.DataFrame()
for fcn in DDD_monthly.columns[2:] :
    df_temp2=DDD_monthly.loc[:,['year','month']]
    df_temp2['fcn_DDD']=fcn
    df_temp2['fcnv_DDD']=DDD_monthly.loc[:,[fcn]]
    number_of_DDD_ATC_4=pd.concat([number_of_DDD_ATC_4,df_temp2],axis=0)
number_of_DDD_ATC_4.head()
```

Out[17]:

	year	month	fcn_DDD	fcnv_DDD
0	2011	1	J01AA	808432.00000
1	2011	2	J01AA	616620.00000
2	2011	3	J01AA	802755.00000
3	2011	4	J01AA	736725.66700
4	2011	5	J01AA	1038556.00000

In [18]:
```
monthly_DID_in_sample_ATC_4=pd.DataFrame()
for fcn in DiD_monthly.columns[2:] :
    df_temp2=DiD_monthly.loc[:,['year','month']]
    df_temp2['fcn_DID']=fcn
    df_temp2['fcnv_DID']=DiD_monthly.loc[:,[fcn]]
    monthly_DID_in_sample_ATC_4=pd.concat([monthly_DID_in_sample_ATC_4,df_te
monthly_DID_in_sample_ATC_4.head()
```

Out[18]:

	year	month	fcn_DID	fcnv_DID
0	2011	1	J01AA	0.17064
1	2011	2	J01AA	0.13945
2	2011	3	J01AA	0.16945
3	2011	4	J01AA	0.15551
4	2011	5	J01AA	0.21922

In [19]:
```
DDD_DID=pd.concat([number_of_DDD_ATC_4,monthly_DID_in_sample_ATC_4['fcnv_DI
DDD_DID.reset_index(inplace=True,drop=True)
DDD_DID.columns=['year','month','fcn','fcnv_DDD','fcnv_DID']
DDD_DID.head()
```

Out[19]:

	year	month	fcn	fcnv_DDD	fcnv_DID
0	2011	1	J01AA	808432.00000	0.17064
1	2011	2	J01AA	616620.00000	0.13945
2	2011	3	J01AA	802755.00000	0.16945
3	2011	4	J01AA	736725.66700	0.15551
4	2011	5	J01AA	1038556.00000	0.21922

In [20]:
```
DDD_DID.head()
```

Out[20]:

	year	month	fcn	fcnv_DDD	fcnv_DID
0	2011	1	J01AA	808432.00000	0.17064
1	2011	2	J01AA	616620.00000	0.13945
2	2011	3	J01AA	802755.00000	0.16945
3	2011	4	J01AA	736725.66700	0.15551
4	2011	5	J01AA	1038556.00000	0.21922

In [21]:
```
DDD_DID.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1860 entries, 0 to 1859
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   year      1860 non-null   int64  
 1   month     1860 non-null   int64  
 2   fcn       1860 non-null   object 
 3   fcnv_DDD  1778 non-null   float64
 4   fcnv_DID  1860 non-null   float64
dtypes: float64(2), int64(2), object(1)
memory usage: 72.8+ KB
```

In [22]:
```
DDD_DID.fcnv_DDD[DDD_DID['fcnv_DDD']<0]
```

Out[22]:
```
907       -339.25000
931        -45.00000
1090        -3.00000
1093       -20.00000
1570       -78.00000
1573        -1.00000
1577        -1.00000
1752    -66739.24500
1754   -464257.13400
Name: fcnv_DDD, dtype: float64
```

In [23]:
```
negative_value_index=DDD_DID[DDD_DID['fcnv_DDD']<0].index
DDD_DID[DDD_DID['fcnv_DDD']<0]
```

Out[23]:

	year	month	fcn	fcnv_DDD	fcnv_DID
907	2011	8	J01EA	-339.25000	-0.00007
931	2013	8	J01EA	-45.00000	-0.00001
1090	2011	11	J01EE	-3.00000	-0.00000
1093	2012	2	J01EE	-20.00000	-0.00000
1570	2011	11	J01XB	-78.00000	-0.00002
1573	2012	2	J01XB	-1.00000	-0.00000
1577	2012	6	J01XB	-1.00000	-0.00000
1752	2012	1	J01XE	-66739.24500	-0.01501
1754	2012	3	J01XE	-464257.13400	-0.10443

In [24]:
```
DDD_DID.iloc[negative_value_index,[3]]=0
DDD_DID.iloc[negative_value_index,[4]]=0
```

In [25]:
```
DDD_DID[DDD_DID['fcnv_DDD']<0]
```

Out[25]:

	year	month	fcn	fcnv_DDD	fcnv_DID

In [26]:
```
print(DDD_DID.isnull().sum())
```

```
year          0
month         0
fcn           0
fcnv_DDD     82
fcnv_DID      0
dtype: int64
```

In [27]:
```
DDD_DID[DDD_DID['fcnv_DDD'].isna()].groupby('fcn').count()
```

Out[27]:

fcn	year	month	fcnv_DDD	fcnv_DID
J01EA	23	23	0	23
J01EE	33	33	0	33
J01XB	26	26	0	26

تا این مرحله داده های اصلی که برای شبکه عصبی لازم میباشد را توانستیم بسازیم و مقادیر منفی یا به اصطلاح نویزی ان راصفر کردیم و مقادیر گم شده را هم پیدا کردیم و حذف خواهیم کرد در مرحله بعدی نیز باید تمام داده های شبکه عصبی را نیز استاندارد کنیم

In [28]:
```
columns_name=['year', 'month', 'fcn', 'fcnv_DDD', 'fcnv_DID']
```

In [29]:
```
from sklearn import preprocessing
```

از کتابخانه فرخوانی شده برای نرمال کرد داده ها استفاده میکنیم

In [30]:
```
DDD_DiD_byhistory=DDD_DID.copy()
print(DDD_DiD_byhistory.shape)
DDD_DiD_byhistory=DDD_DiD_byhistory.dropna().copy()
print('removed dataset shape:',DDD_DiD_byhistory.shape)
```

```
(1860, 5)
removed dataset shape: (1778, 5)
```

.تمامی سطر هایی که در ان مقادیر گم شده یافت شده است را حذف میکنیم پس تعداد سطر ها کاهش پیدا خواهد کرد

In [31]:
```
fcn_DDD=DDD_DiD_byhistory.fcnv_DDD.copy()

sd_scaler = preprocessing.StandardScaler()
sd_scaler.fit(fcn_DDD.values.reshape(-1,1))
X_train_minmax =sd_scaler.transform(fcn_DDD.values.reshape(-1,1))
```

داده هارا استاندارد میکنیم

In [32]:
```
fcn_DDD_temp=pd.DataFrame(
                            X_train_minmax.reshape(-1,1),
                            index=DDD_DiD_byhistory.fcnv_DDD.index,
                            columns=['fcnv_DDD']
)
print('fcn_DDD_temp.shape:',fcn_DDD_temp.shape)
#replace by new data
DDD_DiD_byhistory.loc[:,['fcnv_DDD']]=fcn_DDD_temp.iloc[:,[0]].copy()
print('DDD_DiD_byhistory[''fcnv_DDD''].shape==>',DDD_DiD_byhistory['fcnv_DDI
```

```
fcn_DDD_temp.shape: (1778, 1)
DDD_DiD_byhistory[fcnv_DDD].shape==> (1778,)
```

In [33]:
```
DDD_DiD_byhistory.head()
```

Out[33]:

	year	month	fcn	fcnv_DDD	fcnv_DID
0	2011	1	J01AA	-0.23844	0.17064
1	2011	2	J01AA	-0.33399	0.13945
2	2011	3	J01AA	-0.24127	0.16945
3	2011	4	J01AA	-0.27416	0.15551
4	2011	5	J01AA	-0.12381	0.21922

مقادیر دوز مصرفی بین بازه -۱ تا ۱ قرار خواهد گرفت

In [34]:
```
#-------------------------------------------------------------------------
#Create Historical data-------------------------------------------------------
print('DDD_DiD_byhistory.shape\n befor add history:',DDD_DiD_byhistory.shape
prd=0
for prd in range(1,13):
    DDD_DiD_byhistory= pd.concat(
        [
            DDD_DiD_byhistory,
            DDD_DiD_byhistory['fcnv_DDD'].shift(periods=prd,fill_value=0)
        ],axis=1
         )
    print(DDD_DiD_byhistory.shape)
    columns_name.append('fcnv_DDD_shift'+str(prd))
    DDD_DiD_byhistory.columns=columns_name
#End of for-------------------------------------------------------------------
print(DDD_DiD_byhistory.columns)
```

```
DDD_DiD_byhistory.shape
 befor add history: (1778, 5)
(1778, 6)
(1778, 7)
(1778, 8)
(1778, 9)
(1778, 10)
(1778, 11)
(1778, 12)
(1778, 13)
(1778, 14)
(1778, 15)
(1778, 16)
(1778, 17)
Index(['year', 'month', 'fcn', 'fcnv_DDD', 'fcnv_DID', 'fcnv_DDD_shift1',
       'fcnv_DDD_shift2', 'fcnv_DDD_shift3', 'fcnv_DDD_shift4',
       'fcnv_DDD_shift5', 'fcnv_DDD_shift6', 'fcnv_DDD_shift7',
       'fcnv_DDD_shift8', 'fcnv_DDD_shift9', 'fcnv_DDD_shift10',
       'fcnv_DDD_shift11', 'fcnv_DDD_shift12'],
      dtype='object')
```

برای پیش بینی تاریخچه یک سال گذشته را با استفاده از حلقه فور و شیفت ایجاد میکنیم

In [35]:
```
col_name=DDD_DiD_byhistory.columns.values.tolist()
```

In [36]:
```
col_name.remove('fcnv_DDD')
col_name.remove('fcnv_DID')

col_name
```

Out[36]:
```
['year',
 'month',
 'fcn',
 'fcnv_DDD_shift1',
 'fcnv_DDD_shift2',
 'fcnv_DDD_shift3',
 'fcnv_DDD_shift4',
 'fcnv_DDD_shift5',
 'fcnv_DDD_shift6',
 'fcnv_DDD_shift7',
 'fcnv_DDD_shift8',
 'fcnv_DDD_shift9',
 'fcnv_DDD_shift10',
 'fcnv_DDD_shift11',
 'fcnv_DDD_shift12']
```

In [38]:
```
from sklearn.neural_network  import MLPRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error,mean_absolute_error
```

In [39]:
```
df_X=DDD_DiD_byhistory.loc[:,col_name]
df_Y=DDD_DiD_byhistory.loc[:,['fcnv_DDD']]
#df_Y=pd.DataFrame(fcn_DDD)
Y=df_Y.values.ravel()
Y
```

Out[39]:
```
array([-0.23844016, -0.3339884 , -0.24126808, ...,  0.56687431,
        0.93965425,  1.10304697])
```

داده های اموزش و تست در شبکه عصبی را ایجاد میکنیم

In [40]:
```
df_X.head()
```

Out[40]:

	year	month	fcn	fcnv_DDD_shift1	fcnv_DDD_shift2	fcnv_DDD_shift3	fcnv_DDD_shift4	fcnv
0	2011	1	J01AA	0.00000	0.00000	0.00000	0.00000	
1	2011	2	J01AA	-0.23844	0.00000	0.00000	0.00000	
2	2011	3	J01AA	-0.33399	-0.23844	0.00000	0.00000	
3	2011	4	J01AA	-0.24127	-0.33399	-0.23844	0.00000	
4	2011	5	J01AA	-0.27416	-0.24127	-0.33399	-0.23844	

In [41]:
```
print(df_X.shape)
```

```
(1778, 15)
```

In [42]:
```
df_X['fcn'].values.reshape(-1,1)
```

Out[42]:
```
array([['J01AA'],
       ['J01AA'],
       ['J01AA'],
       ...,
       ['J01XX'],
       ['J01XX'],
       ['J01XX']], dtype=object)
```

In [43]:
```
from sklearn.preprocessing import LabelEncoder
lbl_encoder=LabelEncoder()
integer_fcn=lbl_encoder.fit_transform(df_X['fcn'].values.reshape(-1,1))
df_X['fcn']=integer_fcn
df_X
```

```
c:\Users\ASUS\zahra\simple-project\env\lib\site-packages\sklearn\prepr
ocessing\_label.py:115: DataConversionWarning: A column-vector y was p
assed when a 1d array was expected. Please change the shape of y to (n
_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

اسم تمامی گروه هارا کد گذاری میکنیم تا کار کردن با ان ها راحت تر شود

In [44]:
```
result=[]
result.append(['historysize','RMSE','MSE','MAE','hlayer_1_size','hlayer_2_si
```

In [45]:
```
shfl=False
rnd_std=444
X_train,X_test,Y_train,Y_test=train_test_split(df_X,Y,test_size=0.05,shuffle

hlayer_1_size=100
hlayer_2_size=50
model_DDD=MLPRegressor((hlayer_1_size,hlayer_2_size),
                       max_iter=1000,
                       verbose=False,
                       activation="relu"
                      )
model_DDD.fit(X_train,Y_train)
y_pred=model_DDD.predict(X_test)
print(model_DDD.get_params)
print(Y_test[-5:-1].round(2))
print(y_pred[-5:-1].round(2))
print(y_pred[-5:-1].round(2)  -  Y_test[-5:-1].round(2) )

RMSE=round(mean_squared_error(Y_test,y_pred,squared=False) ,2)
MSE=round(mean_squared_error(Y_test,y_pred),2)
MAE=round(mean_absolute_error(Y_test,y_pred),2)
print('rmse:'.upper(),RMSE)
print('mse:'.upper(),MSE)
print('mae:'.upper(),MAE)


result.append([prd,RMSE,MSE,MAE,hlayer_1_size,hlayer_2_size,shfl,rnd_std])
pd.DataFrame(result)
```

```
<bound method BaseEstimator.get_params of MLPRegressor(hidden_layer_sizes
=(100, 50), max_iter=1000)>
[0.95 0.86 0.57 0.94]
[0.85 0.93 0.83 0.64]
[-0.1   0.07  0.26 -0.3 ]
RMSE: 0.26
MSE: 0.07
MAE: 0.16
```

Out[45]:

	0	1	2	3	4	5	6	7
0	historysize	RMSE	MSE	MAE	hlayer_1_size	hlayer_2_size	shuffle	randome_state
1	12	0.26000	0.07000	0.16000	100	50	False	444

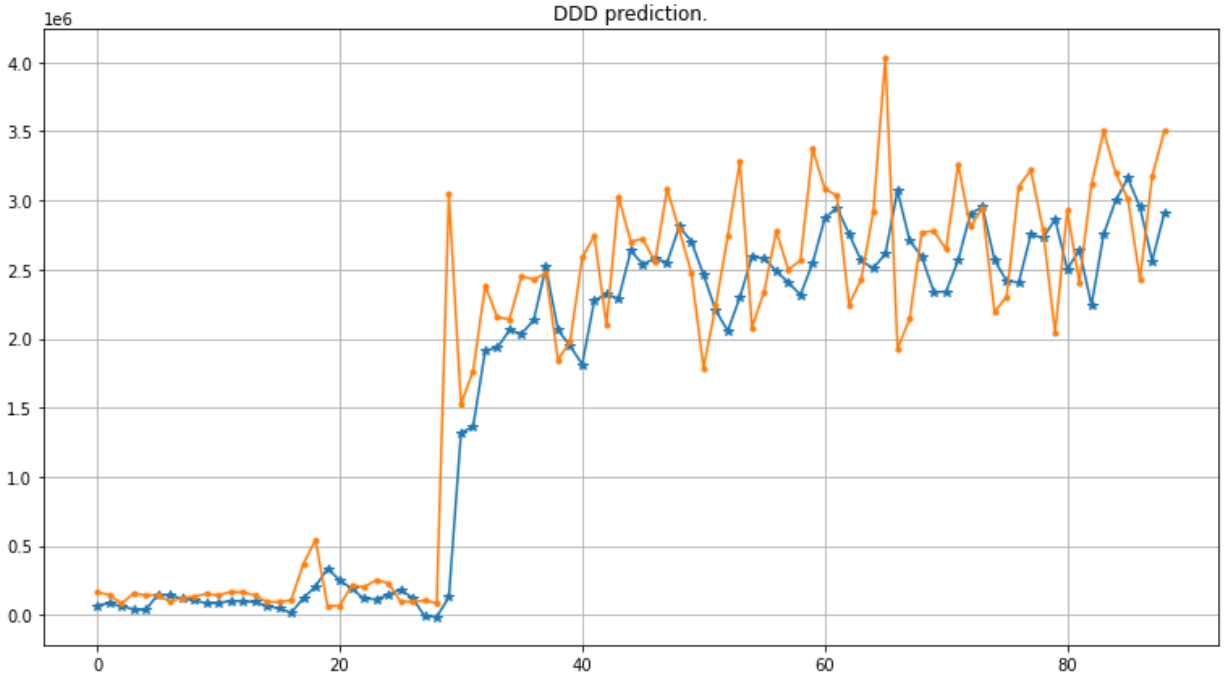
شبکه عصبی با دو لایه پنهان و بدون شافل کردن و با داده های نرمال شده میسازیم

RMSEبه دلیل اینکه بازه اعداد کوچکتر شده اند عدد کوچک تری شده است اما باز هم به نسبت بدون نرمال کردن عدد بزرگی میباشد

In [46]:
```
import matplotlib.pyplot as plt
```

In [47]:
```
y_pred2=sd_scaler.inverse_transform(y_pred.reshape(-1,1))
Y_test2=sd_scaler.inverse_transform(Y_test.reshape(-1,1))
print(y_pred2[0:4])



plt.figure(figsize=(13,7))
plt.title('DDD prediction.')
plt.plot(y_pred2,ls='-',marker='*')
plt.plot(Y_test2,ls='-',marker='.')

plt.grid()
plt.show()



RMSE=round(mean_squared_error(Y_test2,y_pred2,squared=False) ,2)
MSE=round(mean_squared_error(Y_test2,y_pred2),2)
MAE=round(mean_absolute_error(Y_test2,y_pred2),2)
print('rmse:'.upper(),RMSE)
print('mse:'.upper(),MSE)
print('mae:'.upper(),MAE)
```

```
[[67220.35353005]
 [83898.72587217]
 [62059.34605144]
 [40730.4616308 ]]
```




```
RMSE: 516063.63
MSE: 266321666174.28
MAE: 324809.09
```

در نمودار رسم شده مقادیر پیش بینی شده با علامت ستاره و مقادیر تست ما با علامت نقطه مشخص شده است و با استفاده از نمودار میتوان کارایی شبکه عصبی و تفاوت را مشاهده کرد.

In [49]:
```
print(y_pred2[0:7].ravel().round(2))
print(Y_test2[0:7].ravel().round(2))

print(Y_test2[0:7].ravel().round(2) -  y_pred2[0:7].ravel().round(2)  )
```

```
[ 67220.35  83898.73  62059.35  40730.46  34535.18 147897.59 141500.17]
[163967.86 144471.43  80646.43 155839.3  141514.28 141925.    95550.  ]
[ 96747.51  60572.7   18587.08 115108.84 106979.1   -5972.59 -45950.17]
```

In [ ]:
```

```