Sales Prediction for WOMart Company

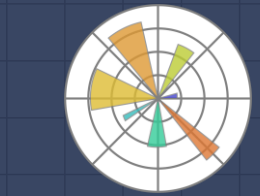
Prediction Project – Inventory Control

Instructor: Behrouz Karimi
T.A: Mohammad Reza Bagheri

Bahman 1400

Libraries

```
%matplotlib notebook
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
pd.set_option('display.max_rows', None)
plt.style.use("seaborn")
import warnings
```



statsmodels

WOMart

Is a leading nutrition and supplement retail store that provides products that meet all your health and fitness needs.

WOMart is a multichannel developer with 350 retail stores in more than 100 cities

Dataset

The data initially included various columns such as the type of store, whether it was closed that day, etc.

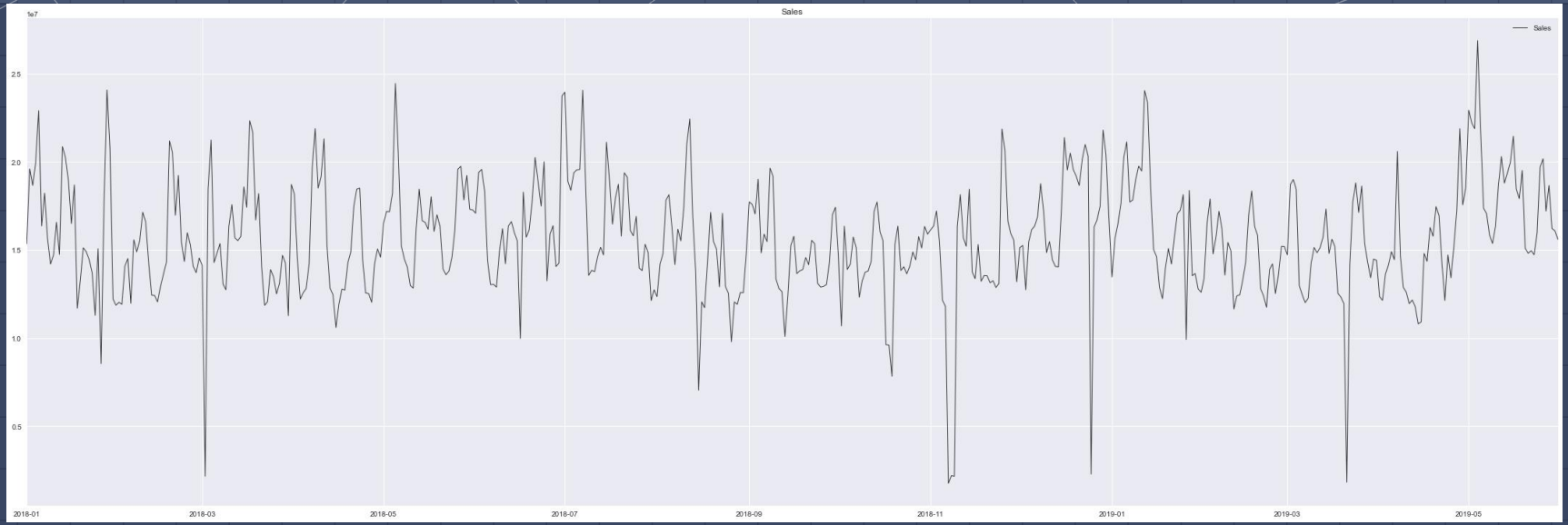
These factors can affect sales, but since they were beyond the objectives of the lesson, we grouped the data by date (we obtained the total sales per day)

	ID	Store_id	Store_Type	Location_Type	Region_Code	Date	Holiday	Discount	#Order	Sales
0	T1000001	1	S1	L3	R1	2018-01-01	1	Yes	9	7011.84
1	T1000002	253	S4	L2	R1	2018-01-01	1	Yes	60	51789.12
2	T1000003	252	S3	L2	R1	2018-01-01	1	Yes	42	36868.20
3	T1000004	251	S2	L3	R1	2018-01-01	1	Yes	23	19715.16
4	T1000005	250	S2	L3	R4	2018-01-01	1	Yes	62	45614.52

```
df = pd.read_csv("train.csv")
df['Date']=pd.to_datetime(df['Date'] , format='%Y-%m-%d')
df = df.groupby('Date', as_index=False)['Sales'].sum()
```

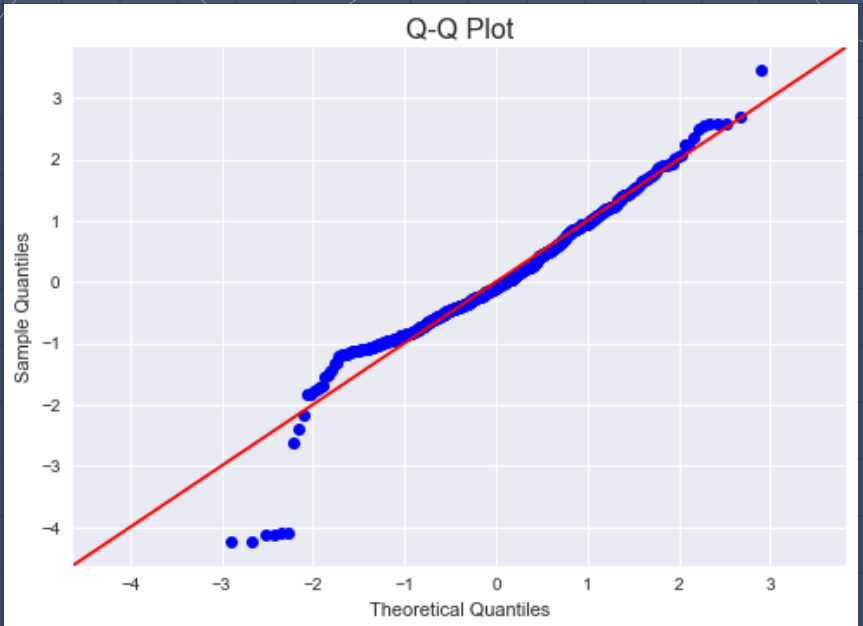
	Date	Sales
1	2018-01-01	15345484.50
2	2018-01-02	19592415.00
3	2018-01-03	18652527.00
4	2018-01-04	19956267.00
5	2018-01-05	22902651.00
...	...	...
512	2019-05-27	17197023.00
513	2019-05-28	18652065.00
514	2019-05-29	16213497.00
515	2019-05-30	16082139.00
516	2019-05-31	15601824.99

نمای کلی دیتاست




```
plt.figure(figsize=(30,10))
plt.plot(df['Date'] , df['Sales'] , alpha=0.8 , c='black',label='Sales',linewidth=1)
plt.legend()
plt.tight_layout()
plt.title('Sales')
plt.xlim((pd.Timestamp('2018-01-01'),pd.Timestamp('2019-5-31')))
plt.show()
```

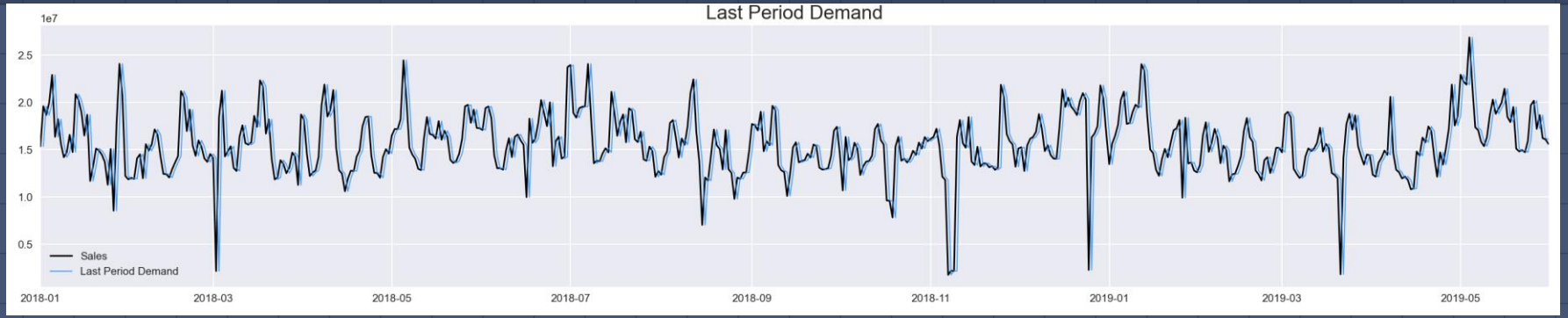
Normality Test




```
import statsmodels.api as sm
import pylab as py
sm.qqplot(df['Sales'], line ='45',fit=True)
py.title('Q-Q Plot',fontsize=16)
py.figure(figsize=(20,20))
py.show()
```

Last Period Demand

LPD




```
# function for prediction
def LPD(sales):
    pred = np.zeros(len(sales))
    for i in range(1,len(sales)):
        pred[i] = sales[i]
    pred[0] = np.nan
    return pred
```

	Date	Sales	Predicted
1	2018-01-01	15345484.5	NaN
2	2018-01-02	19592415.0	15345484.5
3	2018-01-03	18652527.0	19592415.0
4	2018-01-04	19956267.0	18652527.0
5	2018-01-05	22902651.0	19956267.0

Assessing Model Accuracy

```
def res(sales,prediction):
    return sales-prediction
def pe(sales,prediction):
    return ((sales-prediction)/sales)*100
a = res(df_LPD["Sales"],df_LPD["Predicted"])
re = pe(df_LPD["Sales"],df_LPD["Predicted"])

#ME
def ME(res):
    return np.nanmean(res)
#MAE
def MAE(res):
    return np.nanmean(np.abs(res))
#MSE
def MSE(res):
    return np.nanmean((res)*(res))
#MPE
def MPE(pe):
    return np.nanmean(pe)
#MAPE
def MAPE(pe):
    return np.nanmean(np.abs(pe))
#TS
def TS(res,mae):
    return np.sum(res)/mae
```

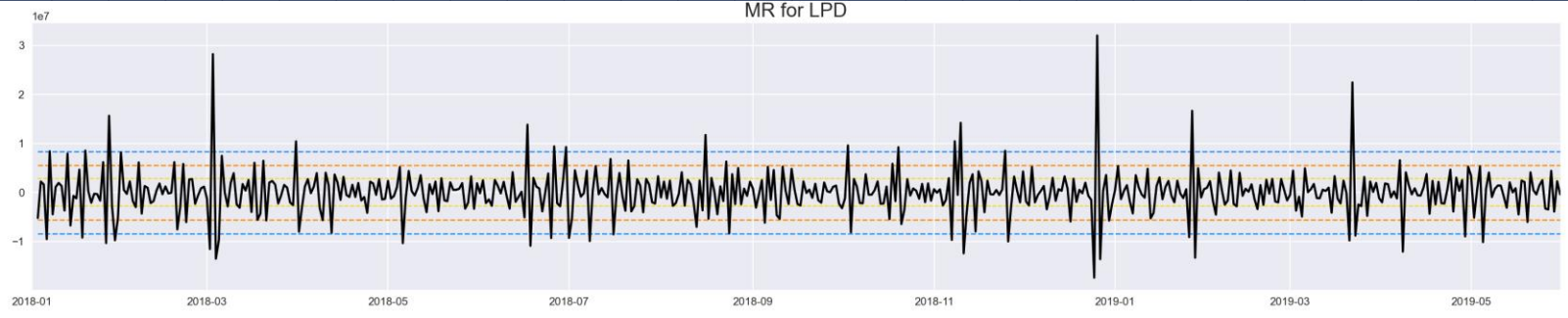
```
ME for LPD model is 497.75
MAE for LPD model is 1981625.42
MSE for LPD model is 9246233436020.38
MPE for LPD model is -5.52
MAPE for LPD model is 16.93
TS for LPD model is 0.13
```

MR

```
n = len(a)
MR = np.zeros(n)
abs_MR = np.zeros(n)
for i in range(3,n+1):
    abs_MR[i-1] = abs(a[i]-a[i-1])
    MR[i-1] = (a[i]-a[i-1])
MR[0] = np.nan
MR[1] = np.nan
abs_MR[0] = np.nan
abs_MR[1] = np.nan
MR = pd.concat([pd.DataFrame(MR,columns=['MR']),pd.DataFrame(abs_MR,columns=['abs_MR'])],axis=1)
MR.index += 1

MR["CL"] = np.zeros(n)
MR["LCL_1"] = np.zeros(n)
MR["UCL_1"] = np.zeros(n)
MR["LCL_2"] = np.zeros(n)
MR["UCL_2"] = np.zeros(n)
MR["LCL_3"] = np.zeros(n)
MR["UCL_3"] = np.zeros(n)
MR_bar = MR["abs_MR"].mean(skipna=True)
for i in range(3,n+1):
    MR["CL"][i] = 0
    MR["LCL_1"][i] = -0.89 * MR_bar
    MR["UCL_1"][i] =  0.89 * MR_bar
    MR["LCL_2"][i] = -1.77 * MR_bar
    MR["UCL_2"][i] =  1.77 * MR_bar
    MR["LCL_3"][i] = -2.66 * MR_bar
    MR["UCL_3"][i] =  2.66 * MR_bar

MR = pd.concat([MR,df["Date"]],axis=1)
MR = MR.iloc[2:,:]
MR.head()
```

MR

	MR	abs_MR	CL	LCL_1	UCL_1	LCL_2	UCL_2	LCL_3	UCL_3	Date
3	-5186818.5	5186818.5	0.0	-2.816212e+06	2.816212e+06	-5.600782e+06	5.600782e+06	-8.416994e+06	8.416994e+06	2018-01-03
4	2243628.0	2243628.0	0.0	-2.816212e+06	2.816212e+06	-5.600782e+06	5.600782e+06	-8.416994e+06	8.416994e+06	2018-01-04
5	1642644.0	1642644.0	0.0	-2.816212e+06	2.816212e+06	-5.600782e+06	5.600782e+06	-8.416994e+06	8.416994e+06	2018-01-05
6	-9497079.0	9497079.0	0.0	-2.816212e+06	2.816212e+06	-5.600782e+06	5.600782e+06	-8.416994e+06	8.416994e+06	2018-01-06
7	8417988.0	8417988.0	0.0	-2.816212e+06	2.816212e+06	-5.600782e+06	5.600782e+06	-8.416994e+06	8.416994e+06	2018-01-07



Simple Moving Average

2

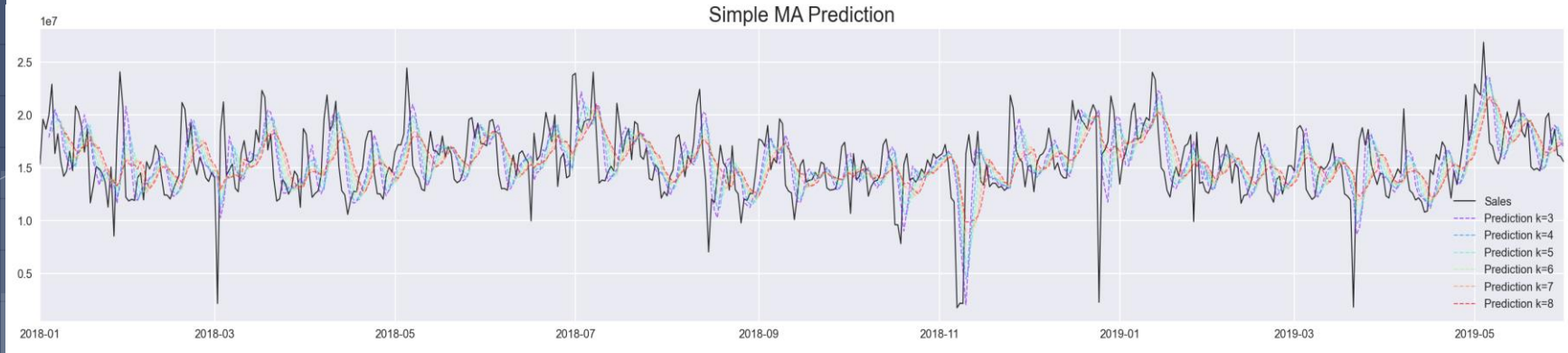
SMA

```
df_MA=df.copy()
for k in range(3,9):
    for i in range(k+1,len(df_MA)+1):
        df_MA.loc[i,'MA{}'.format(k)]=np.mean(df_MA.loc[i-k:i-1,'Sales']) #Forcast
        df_MA.loc[i,'MA{} Residual'.format(k)]=df_MA.loc[i,'Sales']-df_MA.loc[i,'MA{}'.format(k)] #Residual
        df_MA.loc[i,'MA{} PE'.format(k)]=((df_MA.loc[i,'Sales']-df_MA.loc[i,'MA{}'.format(k)])/df_MA.loc[i,'Sales'])*100 #PE


for k in range(3,9):
    for i in range(k+1,len(df_MA)):
        df_MA.loc[i+1,'MA{} MR'.format(k)]=df_MA.loc[i+1,'MA{} Residual'.format(k)]-df_MA.loc[i,'MA{} Residual'.format(k)] #MR

df_MA.head()
```

	Date	Sales	MA3	MA3 Residual	MA3 PE	MA4	MA4 Residual	MA4 PE	MA5	MA5 Residual	...	MA7 PE	MA8	MA8 Residual	MA8 PE	MA3 MR	MA4 MR	MA5 MR	MA6 MR	MA7 MR	MA8 MR
1	2018-01-01	15345484.5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	2018-01-02	19592415.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	2018-01-03	18652527.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	2018-01-04	19956267.0	17863475.5	2092791.5	10.486889	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	2018-01-05	22902651.0	19400403.0	3502248.0	15.291889	1.838667e+07	4515977.625	19.718144	NaN	NaN	...	NaN	NaN	NaN	NaN	1409456.5	NaN	NaN	NaN	NaN	NaN

SMA

```
from matplotlib.pyplot import cm
color = iter(cm.rainbow(np.linspace(0, 1, 6)))
def plotting_MA():

    plt.figure(figsize=(20,4))
    plt.style.use('seaborn')
    plt.plot(df_MA['Date'] , df_MA['Sales'] , alpha=0.8 , c='black',label='Sales',linewidth=1)
    for k in range(3,9):
        c=next(color)
        plt.plot(df_MA['Date'] , df_MA['MA{}'.format(k)] ,c=c, alpha=0.6,ls='--',label='Prediction k={}'.format(k),linewidth=1)
    plt.legend(loc='lower right')
    plt.tight_layout()
    plt.xlim(('2018-01-01','2019-05-31'))
    plt.title('Simple MA Prediction',fontsize=18)
    plt.show()
```



Assessing Model Accuracy

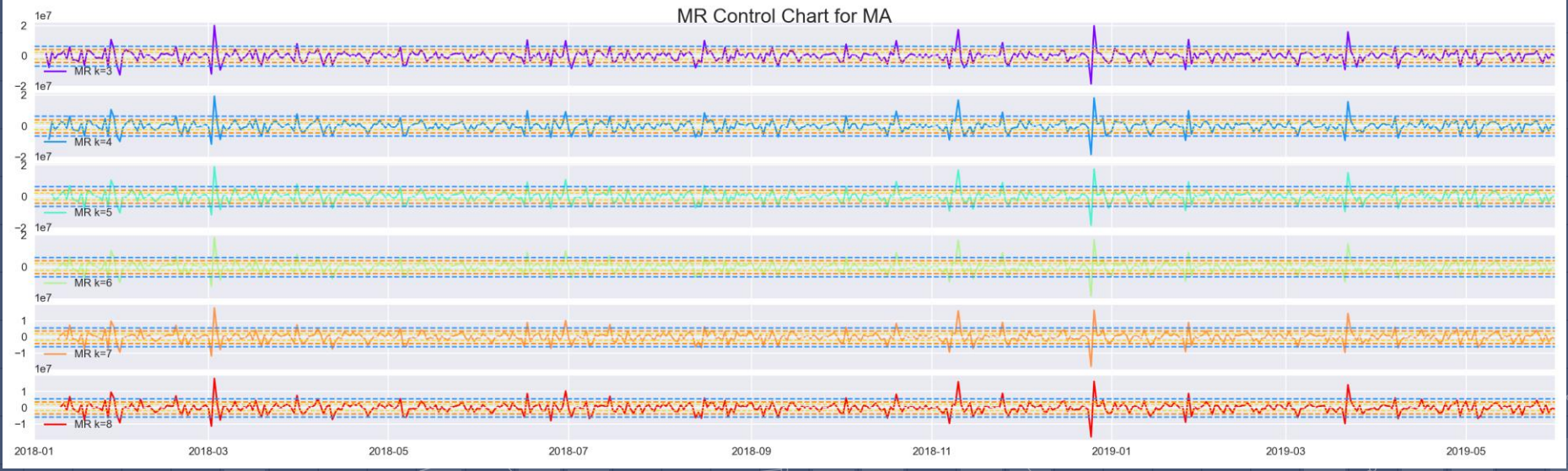
```
ME for MA model with k=3 is -9944.54
ME for MA model with k=4 is -13955.17
ME for MA model with k=5 is -23227.75
ME for MA model with k=6 is -17497.52
ME for MA model with k=7 is -16128.15
ME for MA model with k=8 is -10589.11
Best model is k=3 with ME=-9944.54


MAE for MA model with k=3 is 2322168.56
MAE for MA model with k=4 is 2457766.85
MAE for MA model with k=5 is 2577507.30
MAE for MA model with k=6 is 2599174.71
MAE for MA model with k=7 is 2586844.05
MAE for MA model with k=8 is 2573220.56
Best model is k=3 with MAE=2322168.56


MSE for MA model with k=3 is 10382516871953.03
MSE for MA model with k=4 is 11155040723569.46
MSE for MA model with k=5 is 11788914471433.26
MSE for MA model with k=6 is 11751985007349.61
MSE for MA model with k=7 is 11470578155359.43
MSE for MA model with k=8 is 11329593416688.96
Best model is k=3 with MSE=10382516871953.03
```

```
MPE for MA model with k=3 is -7.40
MPE for MA model with k=4 is -8.16
MPE for MA model with k=5 is -8.81
MPE for MA model with k=6 is -9.13
MPE for MA model with k=7 is -9.34
MPE for MA model with k=8 is -9.47
Best model is k=3 with MPE=-7.40


MAPE for MA model with k=3 is 20.21
MAPE for MA model with k=4 is 21.58
MAPE for MA model with k=5 is 22.74
MAPE for MA model with k=6 is 23.13
MAPE for MA model with k=7 is 23.25
MAPE for MA model with k=8 is 23.26
Best model is k=3 with MAPE=20.21


TS for MA model with k=3 is -2.20
TS for MA model with k=4 is -2.91
TS for MA model with k=5 is -4.60
TS for MA model with k=6 is -3.43
TS for MA model with k=7 is -3.17
TS for MA model with k=8 is -2.09
Best model is k=8 with TS=-2.09
```

MR

```
color = iter(cm.rainbow(np.linspace(0, 1, 6)))
def plotting_MR_MA():
    fig,ax=plt.subplots(nrows=6,ncols=1,figsize=(20,6),sharex=True)
    plt.style.use('seaborn')
    for k in range(3,9):
        c=next(color)
        ax[k-3].plot(df_MA['Date'] , df_MA['MA{} MR'.format(k)] , c=c,alpha=1,ls='-',label='MR k={}'.format(k),linewidth=1.5) #MR
        MR_bar=np.nanmean([abs(x) for x in df_MA['MA{} MR'.format(k)]])
        ax[k-3].plot(df_MA['Date'] , np.full( shape=(516,) , fill_value=0) , c='azure',ls='--' , linewidth=1) #CL
        ax[k-3].plot(df_MA['Date'] , np.full( shape=(516,) , fill_value=0.89*MR_bar) , c='gold',ls='--'  , linewidth=1) #+sigma
        ax[k-3].plot(df_MA['Date'] , np.full( shape=(516,) , fill_value=-0.89*MR_bar) , c='gold',ls='--'  , linewidth=1) #-sigma
        ax[k-3].plot(df_MA['Date'] , np.full( shape=(516,) , fill_value=1.77*MR_bar) , c='darkorange',ls='--' , linewidth=1.25) #2sigma
        ax[k-3].plot(df_MA['Date'] , np.full( shape=(516,) , fill_value=-1.77*MR_bar) , c='darkorange',ls='--' , linewidth=1.25) #-2sigma
        ax[k-3].plot(df_MA['Date'] , np.full( shape=(516,) , fill_value=2.66*MR_bar) , c='dodgerblue',ls='--' , linewidth=1.25) #3sigma
        ax[k-3].plot(df_MA['Date'] , np.full( shape=(516,) , fill_value=-2.66*MR_bar) , c='dodgerblue',ls='--' , linewidth=1.25) #-3sigma
        ax[k-3].set_xlim(('2018-01-01','2019-05-31'))
        ax[k-3].legend(loc='lower left')


    plt.tight_layout()
    plt.suptitle('MR Control Chart for MA',fontsize=18)
    plt.subplots_adjust(top=0.95,hspace=0.1)
    plt.show()
```

MR


MR Control Chart for MA
1e7
MR k=3
MR k=4
MR k=5
MR k=6
MR k=7
MR k=8
2018-01
2018-03
2018-05
2018-07
2018-09
2018-11
2019-01
2019-03
2019-05

Trend Moving Average

3

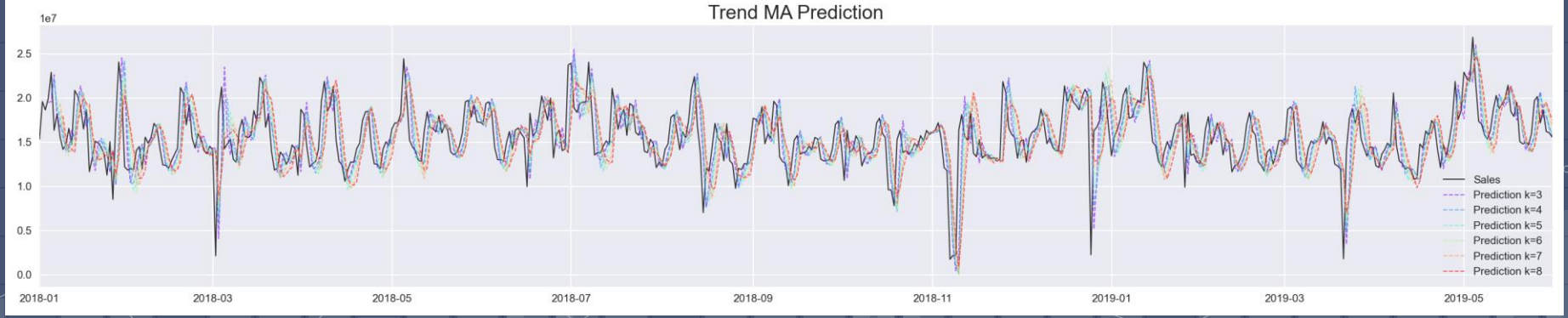
TMA

```
# input= k , index , demand
def forcast_calc(k,index , demand):
    sum_d = np.sum(demand)
    sum_td = np.sum([d*t for t,d in zip(index , demand)])
    m=max(index)
    forcast=(1/k)*sum_d + (6/(k*(k+1)))*(sum_td - ((2*m+1-k)/2)*sum_d)
    return forcast
```

```
df_MA_trend=df.copy()
for k in range(3,9):
    for i in range(k+1,len(df_MA_trend)+1):
        sales= df_MA_trend.loc[i-k:i-1,'Sales'].values
        index=df_MA_trend.loc[i-k:i-1,'Sales'].index
        forcast=forcast_calc(k , index,sales)
        df_MA_trend.loc[i,'MA{} Trend'.format(k)]=forcast #Forcast
        df_MA_trend.loc[i,'MA{} Trend Residual'.format(k)]=df_MA_trend.loc[i,'Sales']-df_MA_trend.loc[i,'MA{} Trend'.format(k)] #Residual
        df_MA_trend.loc[i,'MA{} Trend PE'.format(k)]=((df_MA_trend.loc[i,'Sales']-df_MA_trend.loc[i,'MA{} Trend'.format(k)])/df_MA_trend.loc[i,'Sales'])*100 #PE


for k in range(3,9):
    for i in range(k+1,len(df_MA_trend)):
        df_MA_trend.loc[i+1,'MA{} Trend MR'.format(k)]=df_MA_trend.loc[i+1,'MA{} Trend Residual'.format(k)]-df_MA_trend.loc[i,'MA{} Trend Residual'.format(k)] #MR
```

TMA

	Date	Sales	MA3 Trend	MA3 Trend Residual	MA3 Trend PE	MA4 Trend	MA4 Trend Residual	MA4 Trend PE	MA5 Trend	MA5 Trend Residual	...	MA7 Trend PE	MA8 Trend	MA8 Trend Residual	MA8 Trend PE	MA3 Trend MR	MA4 Trend MR	MA5 Trend MR	MA6 Trend MR	MA7 Trend MR	MA8 Trend MR
1	2018-01-01	15345484.5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	2018-01-02	19592415.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	2018-01-03	18652527.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	2018-01-04	19956267.0	19516996.75	439270.25	2.201164	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	2018-01-05	22902651.0	19582329.00	3320322.00	14.497544	20320542.3	2582108.7	11.274279	NaN	NaN	...	NaN	NaN	NaN	NaN	2881051.75	NaN	NaN	NaN	NaN	NaN



Assessing Model Accuracy

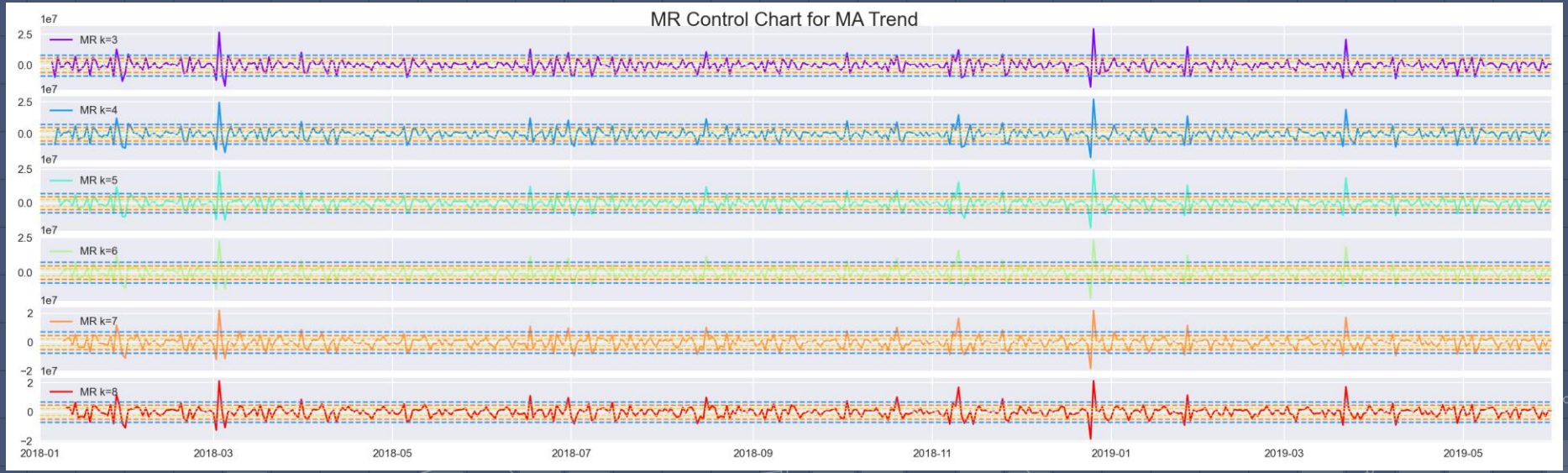
```
ME for Trend MA model with k=3 is -7369.23
ME for Trend MA model with k=4 is -10642.54
ME for Trend MA model with k=5 is -17676.55
ME for Trend MA model with k=6 is -9221.46
ME for Trend MA model with k=7 is -8787.16
ME for Trend MA model with k=8 is -2690.08
Best model is k=8 with ME=-2690.08


MAE for Trend MA model with k=3 is 2107799.19
MAE for Trend MA model with k=4 is 2194610.59
MAE for Trend MA model with k=5 is 2286850.83
MAE for Trend MA model with k=6 is 2427375.20
MAE for Trend MA model with k=7 is 2558730.33
MAE for Trend MA model with k=8 is 2604876.76
Best model is k=3 with MAE=2107799.19


MSE for Trend MA model with k=3 is 9717888957475.73
MSE for Trend MA model with k=4 is 9956366189080.74
MSE for Trend MA model with k=5 is 10344208459217.03
MSE for Trend MA model with k=6 is 11407156341955.51
MSE for Trend MA model with k=7 is 12285454177954.38
MSE for Trend MA model with k=8 is 12481313596246.58
Best model is k=3 with MSE=9717888957475.73
```

```
MPE for Trend MA model with k=3 is -5.61
MPE for Trend MA model with k=4 is -5.77
MPE for Trend MA model with k=5 is -6.01
MPE for Trend MA model with k=6 is -6.37
MPE for Trend MA model with k=7 is -6.81
MPE for Trend MA model with k=8 is -7.15
Best model is k=3 with MPE=-5.61


MAPE for Trend MA model with k=3 is 17.92
MAPE for Trend MA model with k=4 is 18.56
MAPE for Trend MA model with k=5 is 19.32
MAPE for Trend MA model with k=6 is 20.37
MAPE for Trend MA model with k=7 is 21.38
MAPE for Trend MA model with k=8 is 21.97
Best model is k=3 with MAPE=17.92


TS for Trend MA model with k=3 is -1.79
TS for Trend MA model with k=4 is -2.48
TS for Trend MA model with k=5 is -3.95
TS for Trend MA model with k=6 is -1.94
TS for Trend MA model with k=7 is -1.75
TS for Trend MA model with k=8 is -0.52
Best model is k=8 with TS=-0.52
```

MR

```
color = iter(cm.rainbow(np.linspace(0, 1, 6)))
def plotting_MR_MA():
    fig,ax=plt.subplots(nrows=6,ncols=1,figsize=(20,6),sharex=True)
    plt.style.use('seaborn')
    for k in range(3,9):
        c=next(color)
        ax[k-3].plot(df_MA_trend['Date'] , df_MA_trend['MA{} Trend MR'.format(k)] , c=c,alpha=1,ls='-',label='MR k={}'.format(k),linewidth=1.5) #MR
        MR_bar=np.nanmean([abs(x) for x in df_MA_trend['MA{} Trend MR'.format(k)]])
        ax[k-3].plot(df_MA_trend['Date'] , np.full( shape=(516,) , fill_value=0) , c='azure',ls='--' , linewidth=1) #CL
        ax[k-3].plot(df_MA_trend['Date'] , np.full( shape=(516,) , fill_value=0.89*MR_bar) , c='gold',ls='--'  , linewidth=1) #+sigma
        ax[k-3].plot(df_MA_trend['Date'] , np.full( shape=(516,) , fill_value=-0.89*MR_bar) , c='gold',ls='--'  , linewidth=1) #-sigma
        ax[k-3].plot(df_MA_trend['Date'] , np.full( shape=(516,) , fill_value=1.77*MR_bar) , c='darkorange',ls='--' , linewidth=1.25) #2sigma
        ax[k-3].plot(df_MA_trend['Date'] , np.full( shape=(516,) , fill_value=-1.77*MR_bar) , c='darkorange',ls='--' , linewidth=1.25) #-2sigma
        ax[k-3].plot(df_MA_trend['Date'] , np.full( shape=(516,) , fill_value=2.66*MR_bar) , c='dodgerblue',ls='--' , linewidth=1.25) #3sigma
        ax[k-3].plot(df_MA_trend['Date'] , np.full( shape=(516,) , fill_value=-2.66*MR_bar) , c='dodgerblue',ls='--' , linewidth=1.25) #-3sigma
        ax[k-3].set_xlim(('2018-01-01','2019-05-31'))
        ax[k-3].legend(loc='upper left')


    plt.tight_layout()
    plt.suptitle('MR Control Chart for MA Trend',fontsize=18)
    plt.subplots_adjust(top=0.95,hspace=0.1)
    plt.show()
```

MR


MR Control Chart for MA Trend
MR k=3
MR k=4
MR k=5
MR k=6
MR k=7
MR k=8
1e7
2.5
0.0
2
0
−2
2018-01
2018-03
2018-05
2018-07
2018-09
2018-11
2019-01
2019-03
2019-05

Weighted Moving Average

Because in the previous sections we concluded that k=3 and 8 are better for predictions, so we only use them and give them weights for predicting

4

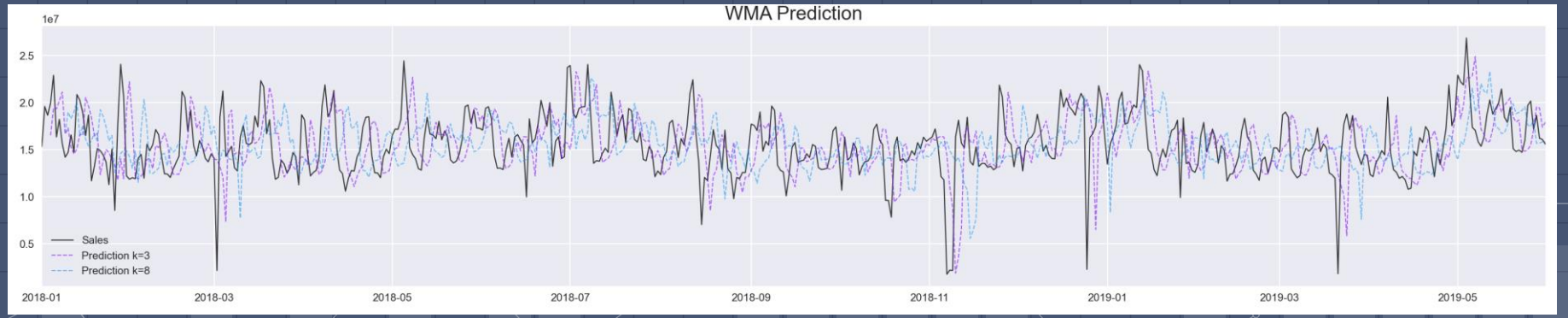
WMA

```
def forcast_calc_WMA(k, demand):
    if k==3:
        weights=[0.7,0.2,0.1]
    elif k==8:
        weights=[0.6,0.1,1/20,1/20,1/20 , 1/20 , 1/20 , 1/20]
    forcast = np.sum([d*t for t,d in zip(weights , demand)])
    return forcast
```

```
df_WMA=df.copy()
for k in [3,8]:
    for i in range(k+1,len(df_WMA)+1):
        sales= df_WMA.loc[i-k:i-1,'Sales'].values
        forcast=forcast_calc_WMA(k ,sales)
        df_WMA.loc[i,'WMA{}'.format(k)]=forcast #Forcast
        df_WMA.loc[i,'WMA{} Residual'.format(k)]=df_WMA.loc[i,'Sales']-df_WMA.loc[i,'WMA{}'.format(k)] #Residual
        df_WMA.loc[i,'WMA{} PE'.format(k)]=((df_WMA.loc[i,'Sales']-df_WMA.loc[i,'WMA{}'.format(k)])/df_WMA.loc[i,'Sales'])*100 #PE


for k in [3,8]:
    for i in range(k+1,len(df_WMA)):
        df_WMA.loc[i+1,'WMA{} MR'.format(k)]=df_WMA.loc[i+1,'WMA{} Residual'.format(k)]-df_WMA.loc[i,'WMA{} Residual'.format(k)] #MR
```

WMA

	Date	Sales	WMA3	WMA3 Residual	WMA3 PE	WMA8	WMA8 Residual	WMA8 PE	WMA3 MR	WMA8 MR
1	2018-01-01	15345484.5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	2018-01-02	19592415.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	2018-01-03	18652527.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	2018-01-04	19956267.0	16525574.85	3430692.15	17.191052	NaN	NaN	NaN	NaN	NaN
5	2018-01-05	22902651.0	19440822.60	3461828.40	15.115405	NaN	NaN	NaN	31136.25	NaN



Assessing Model Accuracy

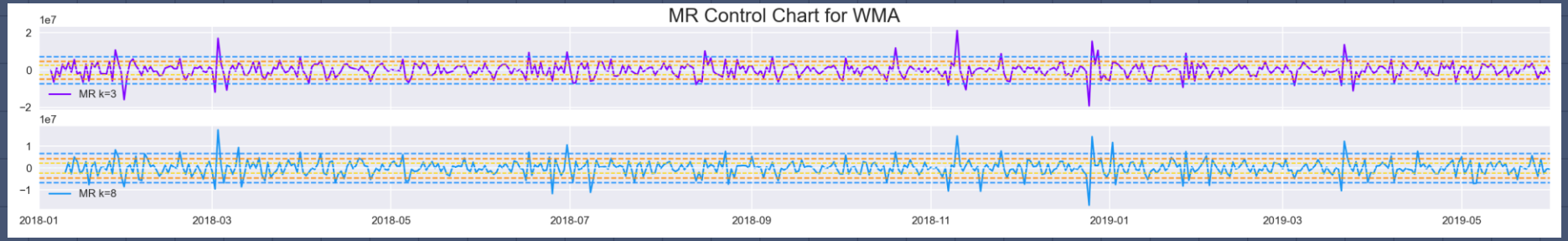
```
ME for WMA model with k=3 is -10920.74
ME for WMA model with k=8 is -12682.10
Best model is k=3 with ME=-10920.74


MAE for WMA model with k=3 is 2773929.18
MAE for WMA model with k=8 is 3005518.10
Best model is k=3 with MAE=2773929.18


MSE for WMA model with k=3 is 14051648202334.56
MSE for WMA model with k=8 is 14394797312210.88
Best model is k=3 with MSE=14051648202334.56
```

```
MPE for WMA model with k=3 is -8.46
MPE for WMA model with k=8 is -10.08
Best model is k=3 with MPE=-8.46


MAPE for WMA model with k=3 is 14394797312210.88
MAPE for WMA model with k=8 is 14394797312210.88
Best model is k=3 with MAPE=23.84


TS for WMA model with k=3 is -2.02
TS for WMA model with k=8 is -2.14
Best model is k=3 with TS=-2.02
```

MR

```
color = iter(cm.rainbow(np.linspace(0, 1, 6)))
def plotting_subWMA():
    fig,ax=plt.subplots(nrows=3,ncols=1,figsize=(20,6),sharex=True)
    plt.style.use('seaborn')
    ax[0].plot(df_WMA['Date'] , df_WMA['Sales'] , alpha=1 , c='black',label='Sales',linewidth=2)
    ax[0].set_xlim(('2018-01-01','2019-05-31'))
    for k in [3,8]:
        c=next(color)
        if k==3:
            ax[k-2].plot(df_WMA['Date'] , df_WMA['WMA{}'.format(k)] , c=c,alpha=1,ls='--',label='Prediction k={}'.format(k),linewidth=2)
            ax[k-2].set_xlim(('2018-01-01','2019-05-31'))
        elif k==8:
            ax[k-6].plot(df_WMA['Date'] , df_WMA['WMA{}'.format(k)] , c=c,alpha=1,ls='--',label='Prediction k={}'.format(k),linewidth=2)
            ax[k-6].set_xlim(('2018-01-01','2019-05-31'))
    for k in range(3):
        ax[k].set_ylim((1762000,26871000))
        ax[k].legend(loc='lower left')
    plt.tight_layout()
    plt.suptitle('WMA Prediction',fontsize=18)
    plt.subplots_adjust(top=0.95,hspace=0.1)
    plt.show();
```

MR


MR Control Chart for WMA
1e7
2
0
−2
MR k=3
1e7
1
0
−1
MR k=8
2018-01
2018-03
2018-05
2018-07
2018-09
2018-11
2019-01
2019-03
2019-05

Simple Exponential Smoothing

5

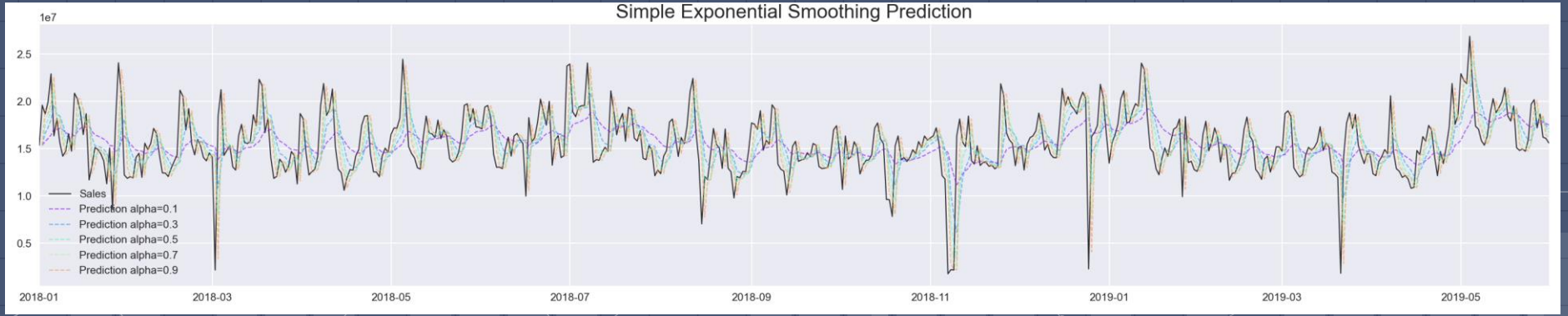
SES

```
#forcast func
def simple_exp_forcast(alpha , forcast_prev,demand):
    forcast=forcast_prev+alpha*(demand-forcast_prev)
    return forcast
```

```
df_exp=df.copy()

for alpha in np.around(np.linspace(0.1 , 0.9 , 5),1):
    df_exp.loc[2,'exp alpha={}'.format(alpha)]=df_exp.loc[1,'Sales']
    for i in range (3 , len(df_exp)+1):
        demand=df_exp.loc[i-1,'Sales']
        prev_forcast=df_exp.loc[i-1,'exp alpha={}'.format(alpha)]
        forcast=simple_exp_forcast(alpha , prev_forcast , demand)
        df_exp.loc[i,'exp alpha={}'.format(alpha)]=forcast #Forcast
        df_exp.loc[i,'exp alpha={} Residual'.format(alpha)]=df_exp.loc[i,'Sales']-df_exp.loc[i,'exp alpha={}'.format(alpha)] #Residual
        df_exp.loc[i,'exp alpha={} PE'.format(alpha)]=((df_exp.loc[i,'Sales']-df_exp.loc[i,'exp alpha={}'.
                                                                format(alpha)])/df_exp.loc[i,'Sales'])*100 #PE


for alpha in np.around(np.linspace(0.1 , 0.9 , 5),1):
    for i in range(3,len(df_exp)):
        df_exp.loc[i+1,'exp alpha={} MR'.format(alpha)]=df_exp.loc[i+1,'exp alpha={} Residual'.format(alpha)]-df_exp.loc[i,'exp alpha={} Residual'.format(alpha)] #MR
```

SES

	Date	Sales	exp alpha=0.1	exp alpha=0.1 Residual	exp alpha=0.1 PE	exp alpha=0.3	exp alpha=0.3 Residual	exp alpha=0.3 PE	exp alpha=0.5	exp alpha=0.5 Residual	...	exp alpha=0.7 Residual	exp alpha=0.7 PE	exp alpha=0.9	exp alpha=0.9 Residual	exp alpha=0.9 PE
1	2018-01-01	15345484.5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
2	2018-01-02	19592415.0	1.534548e+07	NaN	NaN	1.534548e+07	NaN	NaN	1.534548e+07	NaN	...	NaN	NaN	1.534548e+07	NaN	NaN
3	2018-01-03	18652527.0	1.577018e+07	2.882349e+06	15.452863	1.661956e+07	2.032963e+06	10.899131	1.746895e+07	1.183577e+06	...	3.341912e+05	1.791667	1.916772e+07	-5.151949e+05	-2.762065
4	2018-01-04	19956267.0	1.605841e+07	3.897855e+06	19.531982	1.722945e+07	2.726814e+06	13.663950	1.806074e+07	1.895529e+06	...	1.403997e+06	7.035371	1.870405e+07	1.252221e+06	6.274823
5	2018-01-05	22902651.0	1.644820e+07	6.454453e+06	28.182122	1.804750e+07	4.855154e+06	21.199092	1.900850e+07	3.894148e+06	...	3.367583e+06	14.703901	1.983104e+07	3.071606e+06	13.411574



Assessing Model Accuracy

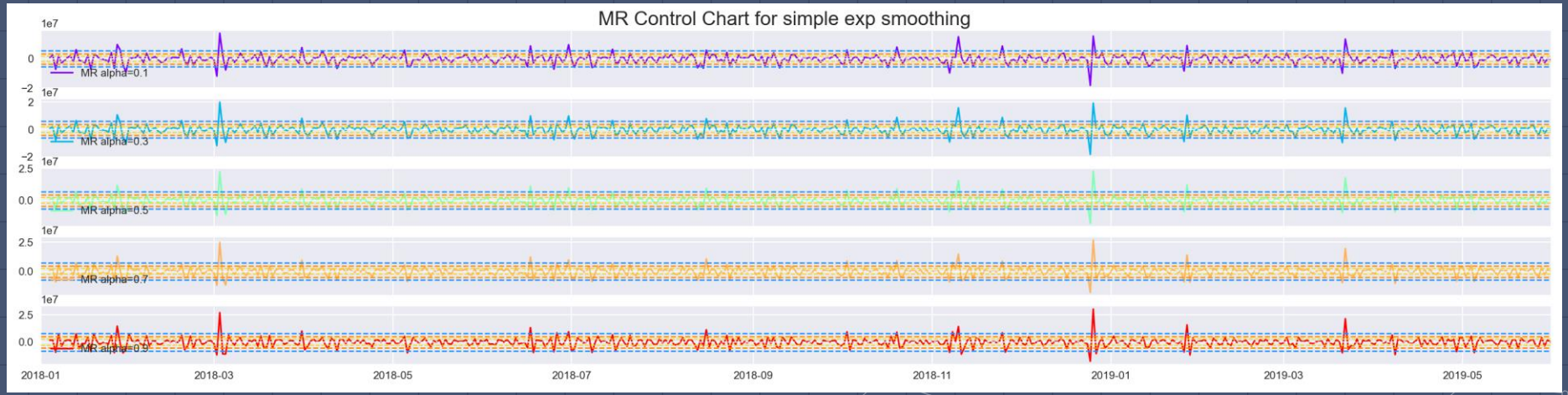
```
ME for Simple Exp Smoothing model with alpha=0.1 is 29920.91
ME for Simple Exp Smoothing model with alpha=0.3 is 8.42
ME for Simple Exp Smoothing model with alpha=0.5 is -5164.93
ME for Simple Exp Smoothing model with alpha=0.7 is -6949.85
ME for Simple Exp Smoothing model with alpha=0.9 is -7596.69
Best model is alpha=0.3 with ME=8.42


MAE for Simple Exp Smoothing model with alpha=0.1 is 2316945.55
MAE for Simple Exp Smoothing model with alpha=0.3 is 2228845.73
MAE for Simple Exp Smoothing model with alpha=0.5 is 2108718.30
MAE for Simple Exp Smoothing model with alpha=0.7 is 2020683.88
MAE for Simple Exp Smoothing model with alpha=0.9 is 1972829.34
Best model is alpha=0.9 with MAE=1972829.34


MSE for Simple Exp Smoothing model with alpha=0.1 is 9747787778607.10
MSE for Simple Exp Smoothing model with alpha=0.3 is 9301196334098.19
MSE for Simple Exp Smoothing model with alpha=0.5 is 8872529989853.48
MSE for Simple Exp Smoothing model with alpha=0.7 is 8708118527104.82
MSE for Simple Exp Smoothing model with alpha=0.9 is 8941183830481.47
Best model is k=0.7 with MSE=8708118527104.82
```

```
MPE for Simple Exp Smoothing model with alpha=0.1 is -9.01
MPE for Simple Exp Smoothing model with alpha=0.3 is -8.11
MPE for Simple Exp Smoothing model with alpha=0.5 is -7.09
MPE for Simple Exp Smoothing model with alpha=0.7 is -6.31
MPE for Simple Exp Smoothing model with alpha=0.9 is -5.77
Best model is alpha=0.9 with MPE=-5.77


MAPE for Simple Exp Smoothing model with alpha=0.1 is 21.28
MAPE for Simple Exp Smoothing model with alpha=0.3 is 20.10
MAPE for Simple Exp Smoothing model with alpha=0.5 is 18.65
MAPE for Simple Exp Smoothing model with alpha=0.7 is 17.60
MAPE for Simple Exp Smoothing model with alpha=0.9 is 16.96
Best model is alpha=0.9 with MAPE=16.96


TS for Simple Exp Smoothing model with alpha=0.1 is 6.64
TS for Simple Exp Smoothing model with alpha=0.3 is 0.00
TS for Simple Exp Smoothing model with alpha=0.5 is -1.26
TS for Simple Exp Smoothing model with alpha=0.7 is -1.77
TS for Simple Exp Smoothing model with alpha=0.9 is -1.98
Best model is alpha=0.3 with TS=0.00
```

MR

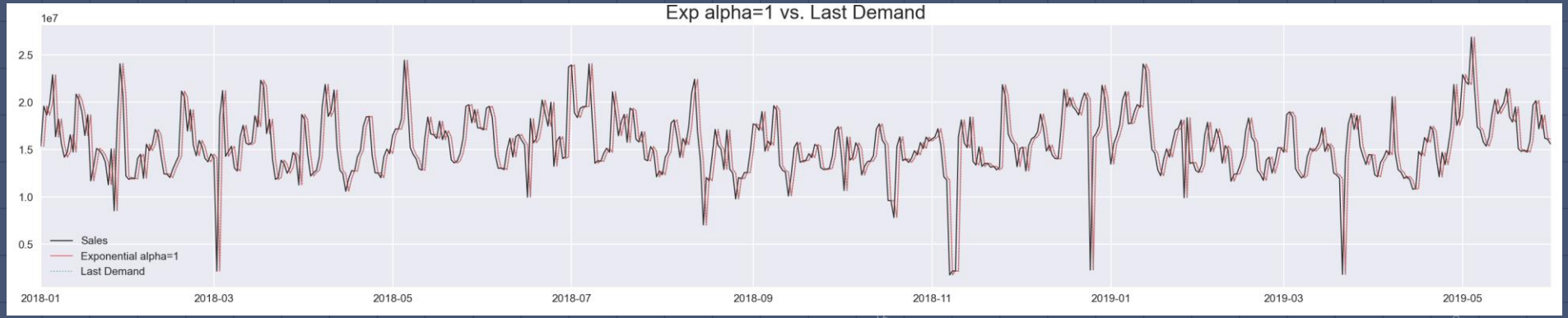
```
color = iter(cm.rainbow(np.linspace(0, 1, 5)))
def plotting_MR_exp():
    fig,ax=plt.subplots(nrows=5,ncols=1,figsize=(20,5),sharex=True)
    plt.style.use('seaborn')
    for i,alpha in enumerate(np.around(np.linspace(0.1 , 0.9 , 5),1)):
        c=next(color)
        ax[i].plot(df_exp['Date'] , df_exp['exp alpha={} MR'.format(alpha)] , c=c,alpha=1,ls='-',label='MR alpha={}'.format(alpha),linewidth=1.5) #MR
        MR_bar=np.nanmean([abs(x) for x in df_exp['exp alpha={} MR'.format(alpha)]])
        ax[i].plot(df_exp['Date'] , np.full( shape=(516,) , fill_value=0) , c='azure',ls='--' , linewidth=1) #CL
        ax[i].plot(df_exp['Date'] , np.full( shape=(516,) , fill_value=0.89*MR_bar) , c='gold',ls='--'  , linewidth=1) #+sigma
        ax[i].plot(df_exp['Date'] , np.full( shape=(516,) , fill_value=-0.89*MR_bar) , c='gold',ls='--'  , linewidth=1) #-sigma
        ax[i].plot(df_exp['Date'] , np.full( shape=(516,) , fill_value=1.77*MR_bar) , c='darkorange',ls='--' , linewidth=1.25) #2sigma
        ax[i].plot(df_exp['Date'] , np.full( shape=(516,) , fill_value=-1.77*MR_bar) , c='darkorange',ls='--' , linewidth=1.25) #-2sigma
        ax[i].plot(df_exp['Date'] , np.full( shape=(516,) , fill_value=2.66*MR_bar) , c='dodgerblue',ls='--' , linewidth=1.25) #3sigma
        ax[i].plot(df_exp['Date'] , np.full( shape=(516,) , fill_value=-2.66*MR_bar) , c='dodgerblue',ls='--' , linewidth=1.25) #-3sigma
        ax[i].set_xlim(('2018-01-01','2019-05-31'))
        ax[i].legend(loc='lower left')


    plt.tight_layout()
    plt.suptitle('MR Control Chart for simple exp smoothing',fontsize=18)
    plt.subplots_adjust(top=0.93,hspace=0.2)
    plt.show()
```

MR

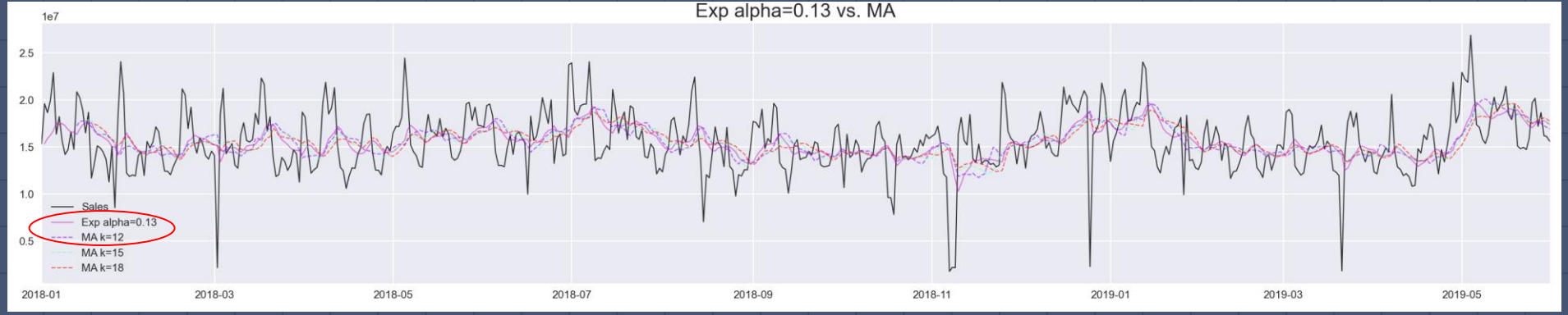

MR Control Chart for simple exp smoothing
1e7
MR alpha=0.1
1e7
MR alpha=0.3
1e7
MR alpha=0.5
1e7
MR alpha=0.7
1e7
MR alpha=0.9
2018-01
2018-03
2018-05
2018-07
2018-09
2018-11
2019-01
2019-03
2019-05

Some Clarification Part I

Show that $\alpha = 1$ equals to LPD



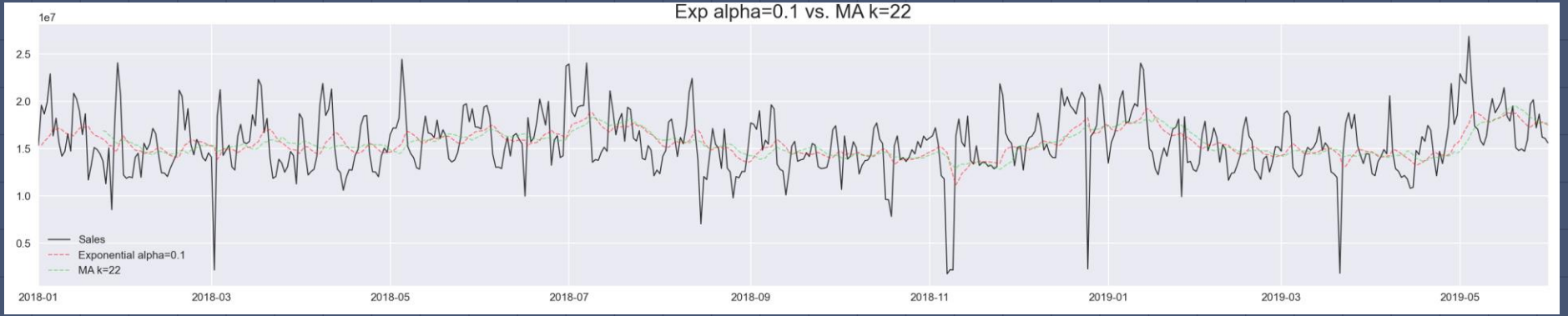
Show that $\alpha \in (0.1\ , 0.15)$ approximates MA with $k \in [12,18]$



Show that $\alpha \in (0.2\,, 0.5)$ approximates MA with $k \in [3,9]$

If $\alpha = 0.5$, then 90% of the weights are on the last 4 demands.

If $\alpha = 0.1$, then 90% of the weights are on the last 22 demands.



Show that $\alpha \approx \frac{2}{k+1}$

Trend Exponential Smoothing

we use alpha=0.9 in this part, because it has a lower MAE for Simple Exponential Smoothing, Which means that the sale is more related to the last period than the others

6

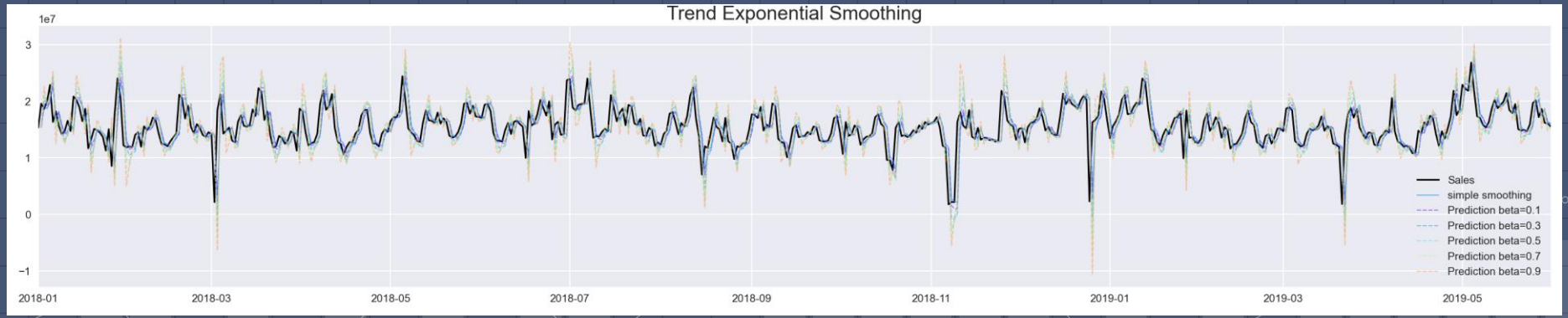
TES

```
#forcast func
def trend_exp_forcast(alpha , beta ,demand , A_prev , T_prev ):

    A_now=alpha*demand+(1-alpha)*(A_prev + T_prev)
    T_now=beta*(A_now - A_prev)+(1-beta)*T_prev
    forcast=T_now + A_now
    return A_now , T_now , forcast
```

```
df_trend_exp=df.copy()

alpha=0.9
for beta in np.around(np.linspace(0.1 , 0.9 , 5),1):
    df_trend_exp.loc[1,'A beta={}'.format(beta)]=df_trend_exp.loc[1,'Sales']
    df_trend_exp.loc[1,'T beta={}'.format(beta)]=0
    df_trend_exp.loc[2,'Forcast beta={}'.format(beta)]=df_trend_exp.loc[1,'Sales']
    for i in range (2 , len(df_trend_exp)+1):
        A_prev=df_trend_exp.loc[i-1,'A beta={}'.format(beta)]
        T_prev=df_trend_exp.loc[i-1,'T beta={}'.format(beta)]
        demand=df_trend_exp.loc[i,'Sales']

        A,T,forcast=trend_exp_forcast(alpha , beta , demand ,A_prev,T_prev)

        df_trend_exp.loc[i,'A beta={}'.format(beta)]=A
        df_trend_exp.loc[i,'T beta={}'.format(beta)]=T
        if i!=len(df_trend_exp):
            df_trend_exp.loc[i+1,'Forcast beta={}'.format(beta)]=forcast

        df_trend_exp.loc[i,'Residual beta={}'.format(beta)]=df_trend_exp.loc[i,'Sales']-df_trend_exp.loc[i,'Forcast beta={}'.format(beta)] #Residual
        df_trend_exp.loc[i,'PE beta={}'.format(beta)]=(df_trend_exp.loc[i,'Residual beta={}'.format(beta)]/df_trend_exp.loc[i,'Sales'])*100 #PE


for beta in np.around(np.linspace(0.1 , 0.9 , 5),1):
    for i in range(2,len(df_trend_exp)):
        df_trend_exp.loc[i+1,'MR beta={}'.format(beta)]=df_trend_exp.loc[i+1,'Residual beta={}'
                                                                         .format(beta)]-df_trend_exp.loc[i,'Residual beta={}'.format(beta)] #MR

df_trend_exp['simple alpha=0.9']=df_exp.loc[:,'exp alpha={}'.format(alpha)]
```

TES

	Date	Sales	A beta=0.1	T beta=0.1	Forcast beta=0.1	Residual beta=0.1	PE beta=0.1	A beta=0.3	T beta=0.3	Forcast beta=0.3	...	T beta=0.9	Forcast beta=0.9	Residual beta=0.9	PE beta=0.9
1	2018-01-01	15345484.5	1.534548e+07	0.000000	NaN	NaN	NaN	1.534548e+07	0.000000e+00	NaN	...	0.000000e+00	NaN	NaN	NaN
2	2018-01-02	19592415.0	1.916772e+07	382223.745000	1.534548e+07	4.246931e+06	21.676401	1.916772e+07	1.146671e+06	1.534548e+07	...	3.440014e+06	1.534548e+07	4.246931e+06	21.676401
3	2018-01-03	18652527.0	1.874227e+07	301456.062450	1.954995e+07	-8.974187e+05	-4.811245	1.881871e+07	6.979674e+05	2.031439e+07	...	2.362947e+05	2.260774e+07	-3.955209e+06	-21.204680
4	2018-01-04	19956267.0	1.986501e+07	383584.848575	1.904372e+07	9.125421e+05	4.572709	1.991231e+07	8.166556e+05	1.951668e+07	...	7.805535e+05	1.928434e+07	6.719244e+05	3.366985
5	2018-01-05	22902651.0	2.263725e+07	622449.650815	2.024860e+07	2.654053e+06	11.588411	2.268528e+07	1.403551e+06	2.072896e+07	...	2.589302e+06	2.066963e+07	2.233023e+06	9.750063



Assessing Model Accuracy

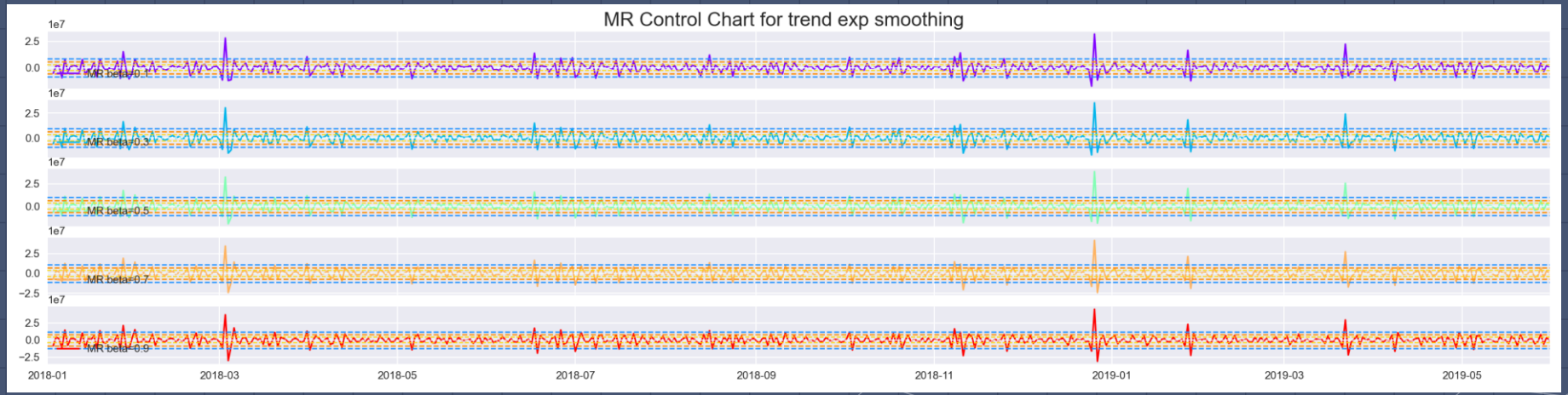
```
ME for Trend Exp Smoothing model with beta=0.1 is -4050.18
ME for Trend Exp Smoothing model with beta=0.3 is -3242.02
ME for Trend Exp Smoothing model with beta=0.5 is -2464.67
ME for Trend Exp Smoothing model with beta=0.7 is -1568.68
ME for Trend Exp Smoothing model with beta=0.9 is -969.73
Best model is beta=0.9 with ME=-969.73


MAE for Trend Exp Smoothing model with beta=0.1 is 2072820.00
MAE for Trend Exp Smoothing model with beta=0.3 is 2255887.07
MAE for Trend Exp Smoothing model with beta=0.5 is 2440386.14
MAE for Trend Exp Smoothing model with beta=0.7 is 2621425.52
MAE for Trend Exp Smoothing model with beta=0.9 is 2836406.89
Best model is beta=0.1 with MAE=2072820.00


MSE for Trend Exp Smoothing model with beta=0.1 is 9814619466751.69
MSE for Trend Exp Smoothing model with beta=0.3 is 11438879998768.13
MSE for Trend Exp Smoothing model with beta=0.5 is 13063435087313.92
MSE for Trend Exp Smoothing model with beta=0.7 is 14919760711296.75
MSE for Trend Exp Smoothing model with beta=0.9 is 17228157656301.76
Best model is beta=0.1 with MSE=9814619466751.69
```

```
MPE for Trend Exp Smoothing model with beta=0.1 is -5.33
MPE for Trend Exp Smoothing model with beta=0.3 is -4.57
MPE for Trend Exp Smoothing model with beta=0.5 is -4.08
MPE for Trend Exp Smoothing model with beta=0.7 is -3.81
MPE for Trend Exp Smoothing model with beta=0.9 is -3.68
Best model is beta=0.9 with MPE=-3.68


MAPE for Trend Exp Smoothing model with beta=0.1 is 17.60
MAPE for Trend Exp Smoothing model with beta=0.3 is 18.96
MAPE for Trend Exp Smoothing model with beta=0.5 is 20.27
MAPE for Trend Exp Smoothing model with beta=0.7 is 21.46
MAPE for Trend Exp Smoothing model with beta=0.9 is 22.83
Best model is beta=0.1 with MAPE=17.60


TS for Trend Exp Smoothing model with beta=0.1 is -1.01
TS for Trend Exp Smoothing model with beta=0.3 is -0.74
TS for Trend Exp Smoothing model with beta=0.5 is -0.52
TS for Trend Exp Smoothing model with beta=0.7 is -0.31
TS for Trend Exp Smoothing model with beta=0.9 is -0.18
Best model is beta=0.9 with TS=-0.18
```

MR

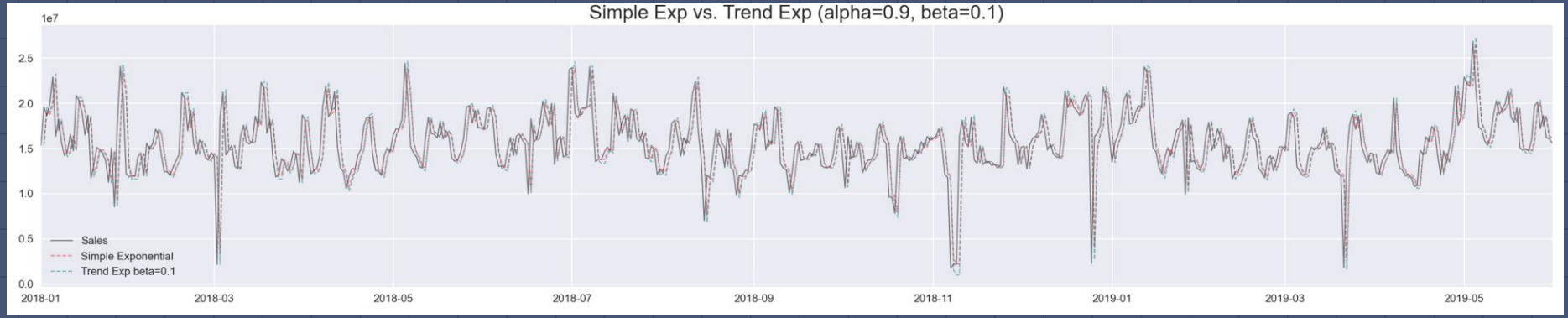
```
color = iter(cm.rainbow(np.linspace(0, 1, 5)))
def plotting_MR_exp_trend():
    fig,ax=plt.subplots(nrows=5,ncols=1,figsize=(20,5),sharex=True)
    plt.style.use('seaborn')
    for i,beta in enumerate(np.around(np.linspace(0.1 , 0.9 , 5),1)):
        c=next(color)
        ax[i].plot(df_trend_exp['Date'] , df_trend_exp['MR beta={}'.format(beta)] , c=c,alpha=1,ls='-',label='MR beta={}'.format(beta),linewidth=1.5) #MR
        MR_bar=np.nanmean([abs(x) for x in df_trend_exp['MR beta={}'.format(beta)]])
        ax[i].plot(df_trend_exp['Date'] , np.full( shape=(516,) , fill_value=0) , c='azure',ls='--' , linewidth=1) #CL
        ax[i].plot(df_trend_exp['Date'] , np.full( shape=(516,) , fill_value=0.89*MR_bar) , c='gold',ls='--'  , linewidth=1) #+sigma
        ax[i].plot(df_trend_exp['Date'] , np.full( shape=(516,) , fill_value=-0.89*MR_bar) , c='gold',ls='--'  , linewidth=1) #-sigma
        ax[i].plot(df_trend_exp['Date'] , np.full( shape=(516,) , fill_value=1.77*MR_bar) , c='darkorange',ls='--' , linewidth=1.25) #2sigma
        ax[i].plot(df_trend_exp['Date'] , np.full( shape=(516,) , fill_value=-1.77*MR_bar) , c='darkorange',ls='--' , linewidth=1.25) #-2sigma
        ax[i].plot(df_trend_exp['Date'] , np.full( shape=(516,) , fill_value=2.66*MR_bar) , c='dodgerblue',ls='--' , linewidth=1.25) #3sigma
        ax[i].plot(df_trend_exp['Date'] , np.full( shape=(516,) , fill_value=-2.66*MR_bar) , c='dodgerblue',ls='--' , linewidth=1.25) #-3sigma
        ax[i].set_xlim(('2018-01-01','2019-05-31'))
        ax[i].set_xlim(('2018-01-01','2019-05-31'))
        ax[i].legend(loc='lower left')


    plt.tight_layout()
    plt.suptitle('MR Control Chart for trend exp smoothing',fontsize=18)
    plt.subplots_adjust(top=0.93,hspace=0.2)
    plt.show()
```

MR


MR Control Chart for trend exp smoothing
1e7
2.5
0.0
MR beta=0.1
1e7
2.5
0.0
MR beta=0.3
1e7
2.5
0.0
MR beta=0.5
1e7
2.5
0.0
−2.5
MR beta=0.7
1e7
2.5
0.0
−2.5
MR beta=0.9
2018-01
2018-03
2018-05
2018-07
2018-09
2018-11
2019-01
2019-03
2019-05

Some Clarification Part II

Plot simple exponential smoothing & modified exponential smoothing for trends with the same α parameter in one graph.



Seasonal Exponential Smoothing

we use alpha=0.9 in this part, because it has a lower MAE for Simple Exponential Smoothing, Which means that the sale is more related to the last period than the others

7

Seasonal ES

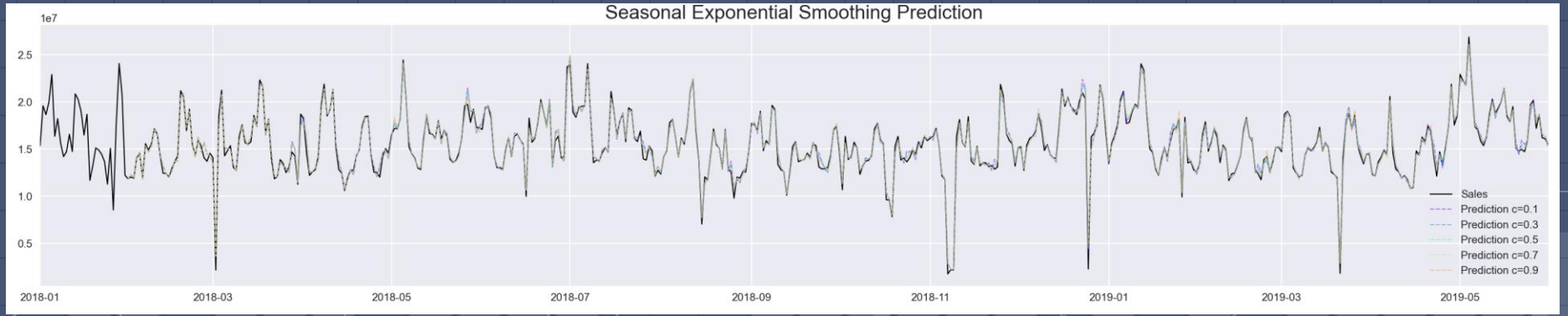
```
# function for calculations
def season_calc(alpha , c , d_t , I_t30 , At1):

    A_t=alpha*d_t/I_t30 + (1-alpha)*At1
    I_t=c*d_t/A_t +(1-c)*I_t30
    return A_t , I_t

# function for updating seasonal index
def normalizer(I_1_31):
    I31 = I_1_31[-1]
    I1 = I_1_31[0]
    coef = 30/(30 + I31 - I1)
    I_1_31[1:]=[x*coef for x in I_1_31[1:]]
    return I_1_31[1:]
```

Seasonal ES

```
df_season_exp=df.copy()
alpha=0.9
for c in np.around(np.linspace(0.1 , 0.9 , 5),1):
    df_season_exp.loc[30,'A c={}'.format(c)]=df_season_exp.loc[30,'Sales']

    #I calc first level
    mean_30=np.nanmean(df_season_exp.loc[:30 , 'Sales'])
    df_season_exp.loc[1:30 , 'I c={}'.format(c)]=df_season_exp.loc[1:30 , 'Sales']/mean_30
    for i in range (31 , len(df_trend_exp)+1):
        d_t = df_season_exp.loc[i,'Sales']
        I_t30 = df_season_exp.loc[i-30, 'I c={}'.format(c)]
        A_t1 = df_season_exp.loc[i-1,'A c={}'.format(c)]

        # I and A calculating
        A_t , I_t=season_calc(alpha , c , d_t , I_t30 , A_t1)
        df_season_exp.loc[i , 'A c={}'.format(c)]=A_t
        df_season_exp.loc[i , 'I c={}'.format(c)]=I_t

        #normalizing (if needed)
        if np.sum(df_season_exp.loc[i-29:i , 'I c={}'.format(c)])!=30:
            df_season_exp.loc[ i-29:i , 'I c={}'.format(c)] = normalizer(df_season_exp.loc[ i-30:i, 'I c={}'.format(c)].tolist())
        I_t30=df_season_exp.loc[i-30 , 'I c={}'.format(c)]
        forcast=A_t*I_t30
        df_season_exp.loc[i,'Forcast c={}'.format(c)]=forcast


        df_season_exp.loc[i,'Residual c={}'.format(c)]=df_season_exp.loc[i,'Sales']-df_season_exp.loc[i,'Forcast c={}'.format(c)] #Residual
        df_season_exp.loc[i,'PE c={}'.format(c)]=(df_season_exp.loc[i,'Residual c={}'.format(c)]/df_season_exp.loc[i,'Sales'])*100 #PE



for c in np.around(np.linspace(0.1 , 0.9 , 5),1):
    for i in range(31,len(df_season_exp)):
        df_season_exp.loc[i+1,'MR c={}'.format(c)]=df_season_exp.loc[i+1,'Residual c={}'
                                                                     .format(c)]-df_season_exp.loc[i,'Residual c={}'.format(c)] #MR
```

Seasonal ES

	Date	Sales	A c=0.1	I c=0.1	Forcast c=0.1	Residual c=0.1	PE c=0.1	A c=0.3	I c=0.3	Forcast c=0.3	...	A c=0.9	I c=0.9	Forcast c=0.9	Residual c=0.9	PE c=0.9	MR c=0.1	MR c=0.3	MR c=0.5	MR c=0.7	MR c=0.9
1	2018-01-01	15345484.5	NaN	0.930395	NaN	NaN	NaN	NaN	0.930395	NaN	...	NaN	0.930395	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	2018-01-02	19592415.0	NaN	1.187870	NaN	NaN	NaN	NaN	1.187839	NaN	...	NaN	1.187747	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	2018-01-03	18652527.0	NaN	1.130996	NaN	NaN	NaN	NaN	1.131189	NaN	...	NaN	1.131767	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	2018-01-04	19956267.0	NaN	1.210042	NaN	NaN	NaN	NaN	1.210235	NaN	...	NaN	1.210817	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	2018-01-05	22902651.0	NaN	1.388641	NaN	NaN	NaN	NaN	1.388754	NaN	...	NaN	1.389096	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN



Assessing Model Accuracy

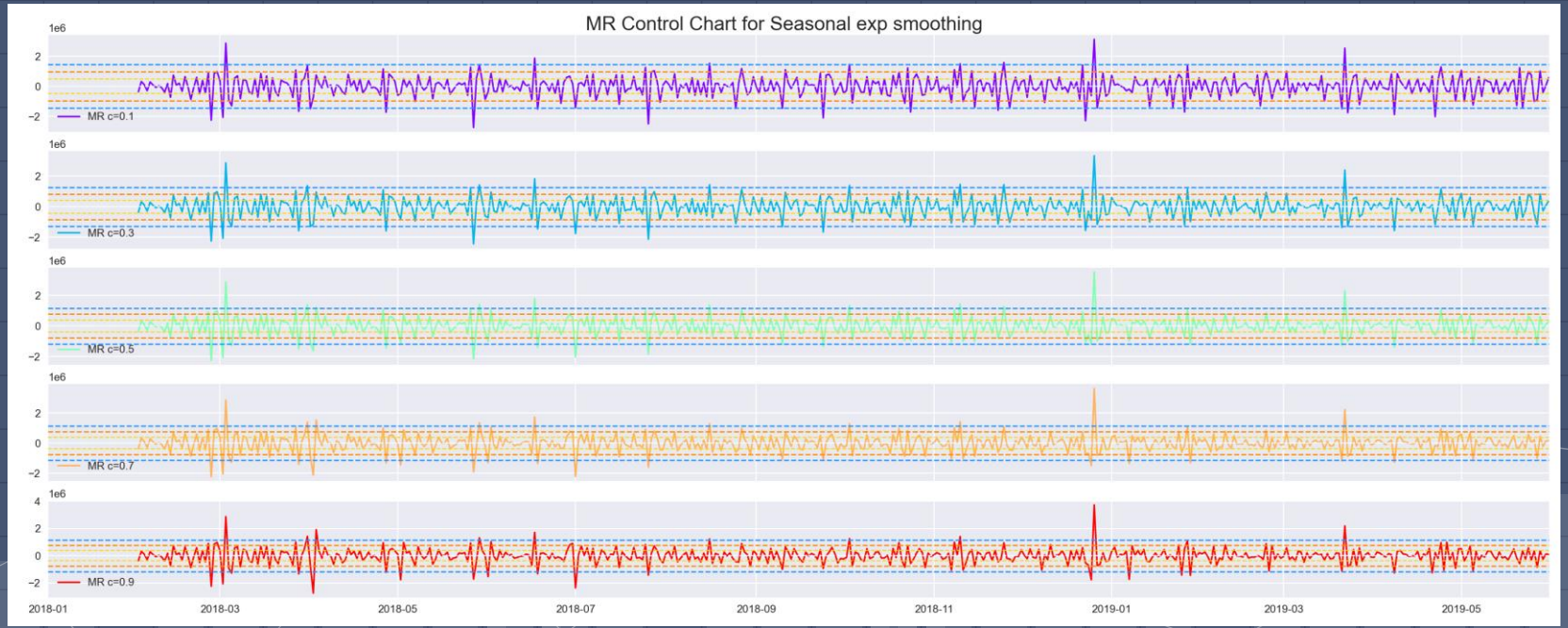
```
ME for Seasonal Exp Smoothing model with c=0.1 is -47159.05
ME for Seasonal Exp Smoothing model with c=0.3 is -35053.21
ME for Seasonal Exp Smoothing model with c=0.5 is -29311.90
ME for Seasonal Exp Smoothing model with c=0.7 is -27689.81
ME for Seasonal Exp Smoothing model with c=0.9 is -29416.76
Best model is c=0.7 with ME=-27689.81


MAE for Seasonal Exp Smoothing model with c=0.1 is 359454.77
MAE for Seasonal Exp Smoothing model with c=0.3 is 326092.14
MAE for Seasonal Exp Smoothing model with c=0.5 is 307666.68
MAE for Seasonal Exp Smoothing model with c=0.7 is 301327.87
MAE for Seasonal Exp Smoothing model with c=0.9 is 303753.34
Best model is c=0.7 with MAE=301327.87


MSE for Seasonal Exp Smoothing model with c=0.1 is 241057603028.14
MSE for Seasonal Exp Smoothing model with c=0.3 is 199703611026.18
MSE for Seasonal Exp Smoothing model with c=0.5 is 181025132887.74
MSE for Seasonal Exp Smoothing model with c=0.7 is 176007343894.48
MSE for Seasonal Exp Smoothing model with c=0.9 is 182930884766.71
Best model is c=0.7 with MSE=176007343894.48
```

```
MPE for Seasonal Exp Smoothing model with c=0.1 is -0.98
MPE for Seasonal Exp Smoothing model with c=0.3 is -0.90
MPE for Seasonal Exp Smoothing model with c=0.5 is -0.86
MPE for Seasonal Exp Smoothing model with c=0.7 is -0.85
MPE for Seasonal Exp Smoothing model with c=0.9 is -0.86
Best model is c=0.7 with MPE=-0.85


MAPE for Seasonal Exp Smoothing model with c=0.1 is 2.86
MAPE for Seasonal Exp Smoothing model with c=0.3 is 2.64
MAPE for Seasonal Exp Smoothing model with c=0.5 is 2.53
MAPE for Seasonal Exp Smoothing model with c=0.7 is 2.49
MAPE for Seasonal Exp Smoothing model with c=0.9 is 2.50
Best model is c=0.7 with MAPE=2.49


TS for Seasonal Exp Smoothing model with c=0.1 is -63.76
TS for Seasonal Exp Smoothing model with c=0.3 is -52.24
TS for Seasonal Exp Smoothing model with c=0.5 is -46.30
TS for Seasonal Exp Smoothing model with c=0.7 is -44.66
TS for Seasonal Exp Smoothing model with c=0.9 is -47.07
Best model is c=0.7 with TS=-44.66
```

MR

```
color = iter(cm.rainbow(np.linspace(0, 1, 5)))
def plotting_MR_exp_season():
    fig,ax=plt.subplots(nrows=5,ncols=1,figsize=(20,8),sharex=True)
    plt.style.use('seaborn')
    for i,c in enumerate(np.around(np.linspace(0.1 , 0.9 , 5),1)):
        C=next(color)
        ax[i].plot(df_season_exp['Date'] , df_season_exp['MR c={}'.format(c)] , c=C,alpha=1,ls='-',label='MR c={}'.format(c),linewidth=1.5) #MR
        MR_bar=np.nanmean([abs(x) for x in df_season_exp['MR c={}'.format(c)]])
        ax[i].plot(df_season_exp['Date'] , np.full( shape=(516,) , fill_value=0) , c='azure',ls='--' , linewidth=1) #CL
        ax[i].plot(df_season_exp['Date'] , np.full( shape=(516,) , fill_value=0.89*MR_bar) , c='gold',ls='--'  , linewidth=1) #+sigma
        ax[i].plot(df_season_exp['Date'] , np.full( shape=(516,) , fill_value=-0.89*MR_bar) , c='gold',ls='--'  , linewidth=1) #-sigma
        ax[i].plot(df_season_exp['Date'] , np.full( shape=(516,) , fill_value=1.77*MR_bar) , c='darkorange',ls='--' , linewidth=1.25) #2sigma
        ax[i].plot(df_season_exp['Date'] , np.full( shape=(516,) , fill_value=-1.77*MR_bar) , c='darkorange',ls='--' , linewidth=1.25) #-2sigma
        ax[i].plot(df_season_exp['Date'] , np.full( shape=(516,) , fill_value=2.66*MR_bar) , c='dodgerblue',ls='--' , linewidth=1.25) #3sigma
        ax[i].plot(df_season_exp['Date'] , np.full( shape=(516,) , fill_value=-2.66*MR_bar) , c='dodgerblue',ls='--' , linewidth=1.25) #-3sigma
        ax[i].set_xlim(('2018-01-01','2019-05-31'))
        ax[i].set_xlim(('2018-01-01','2019-05-31'))
        ax[i].legend(loc='lower left')


    plt.tight_layout()
    plt.suptitle('MR Control Chart for Seasonal exp smoothing',fontsize=18)
    plt.subplots_adjust(top=0.95,hspace=0.2)
    plt.show()
```

MR


MR Control Chart for Seasonal exp smoothing
1e6
MR c=0.1
MR c=0.3
MR c=0.5
MR c=0.7
MR c=0.9
2018-01
2018-03
2018-05
2018-07
2018-09
2018-11
2019-01
2019-03
2019-05

Simple Linear Regression

we use the first 9 months of 2018 as the training set and the last 3 months as the test set

8

SLR

```
df_SLR = df.copy()
df_SLR = df_SLR.iloc[:273,:]
df_SLR["Month"] = df_SLR.Date.dt.month
df_SLR = df_SLR.groupby('Month',as_index=False).sum()
df_SLR.index += 1
df_SLR
```

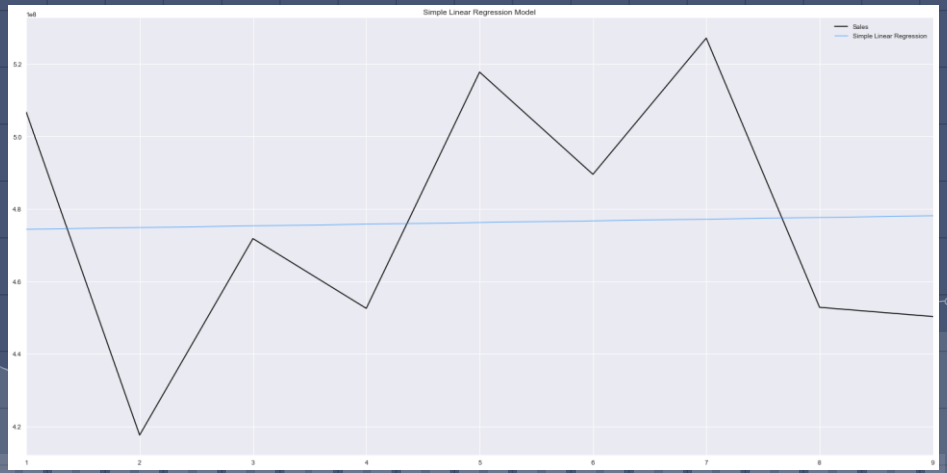
```
df_SLR_test = df.copy()
df_SLR_test = df_SLR_test.iloc[273:364,:]
df_SLR_test["Month"] = df_SLR_test.Date.dt.month
df_SLR_test = df_SLR_test.groupby('Month',as_index=False).sum()
df_SLR_test.index += 1
df_SLR_test
```

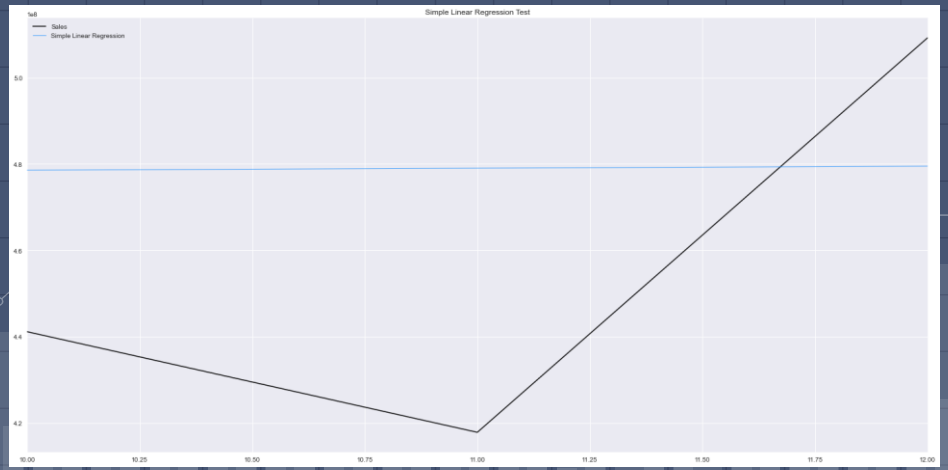
	Month	Sales		Month	Sales
1	1	5.066658e+08	**1**	10	4.411538e+08
2	2	4.175890e+08	**2**	11	4.178625e+08
3	3	4.718072e+08	**3**	12	5.091977e+08
4	4	4.525465e+08			
5	5	5.177403e+08			
6	6	4.895277e+08			
7	7	5.271139e+08			
8	8	4.528305e+08			
9	9	4.502989e+08			

SLR

```
from sklearn import linear_model
X = df_SLR.iloc[:,0:1]
y = df_SLR.iloc[:,1:]
X_test = df_SLR_test.iloc[:,0:1]
reger = linear_model.LinearRegression()
reger.fit(X,y)
df_SLR_test["Predicted"] = reger.predict(X_test)
df_SLR["Predicted"] = reger.predict(X)
print('y = {:.2f}X + {:.2f}'.format(reger.coef_[0][0],reger.intercept_[0]))
```

```
y = 464183.68X + 473914616.18
```





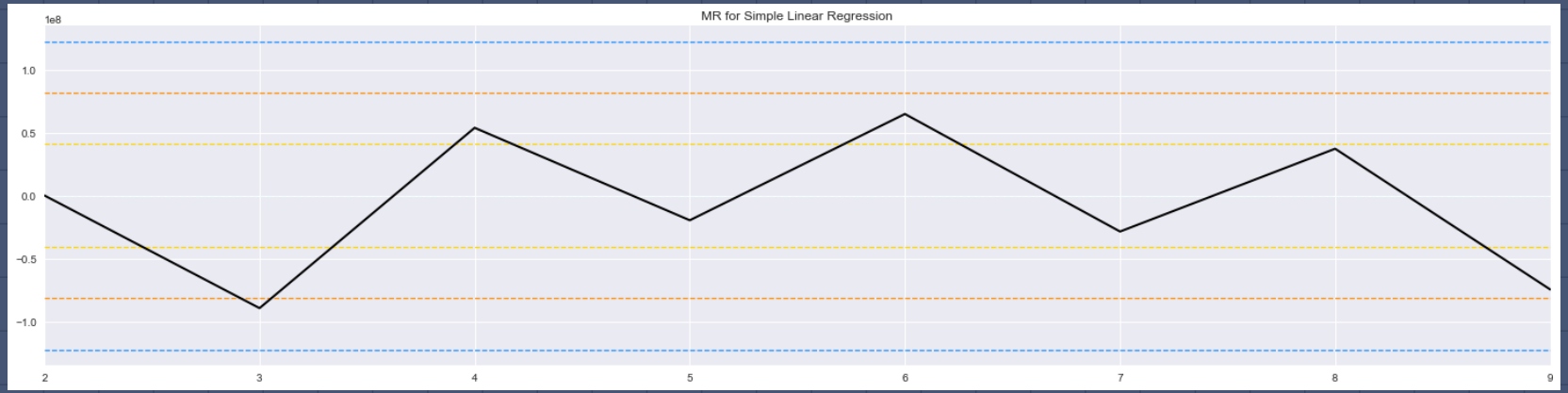
Assessing Model Accuracy

```
ME for LPD model is 497.75
MAE for LPD model is 1981625.42
MSE for LPD model is 9246233436020.38
MPE for LPD model is -5.52
MAPE for LPD model is 16.93
TS for LPD model is 0.13
```

MR

```
a = res(df_SLR["Sales"],df_SLR["Predicted"])
n = len(res(df_SLR["Sales"],df_SLR["Predicted"]))
abs_MR = np.zeros(n)
MR = np.zeros(n)
for i in range(2,n):
    abs_MR[i] = abs(a[i]-a[i-1])
    MR[i] = (a[i]-a[i-1])
MR[0] = np.nan
abs_MR[0] = np.nan
MR = pd.concat([pd.DataFrame(MR,columns=['MR']),pd.DataFrame(abs_MR,columns=['abs_MR'])],axis=1)
MR.index += 1
MR["CL"] = np.zeros(n)
MR["LCL_1"] = np.zeros(n)
MR["UCL_1"] = np.zeros(n)
MR["LCL_2"] = np.zeros(n)
MR["UCL_2"] = np.zeros(n)
MR["LCL_3"] = np.zeros(n)
MR["UCL_3"] = np.zeros(n)
MR_bar = MR["abs_MR"].mean(skipna=True)
for i in range(2,n+1):
    MR["CL"][i] = 0
    MR["LCL_1"][i] = -0.89 * MR_bar
    MR["UCL_1"][i] =  0.89 * MR_bar
    MR["LCL_2"][i] = -1.77 * MR_bar
    MR["UCL_2"][i] =  1.77 * MR_bar
    MR["LCL_3"][i] = -2.66 * MR_bar
    MR["UCL_3"][i] =  2.66 * MR_bar
```

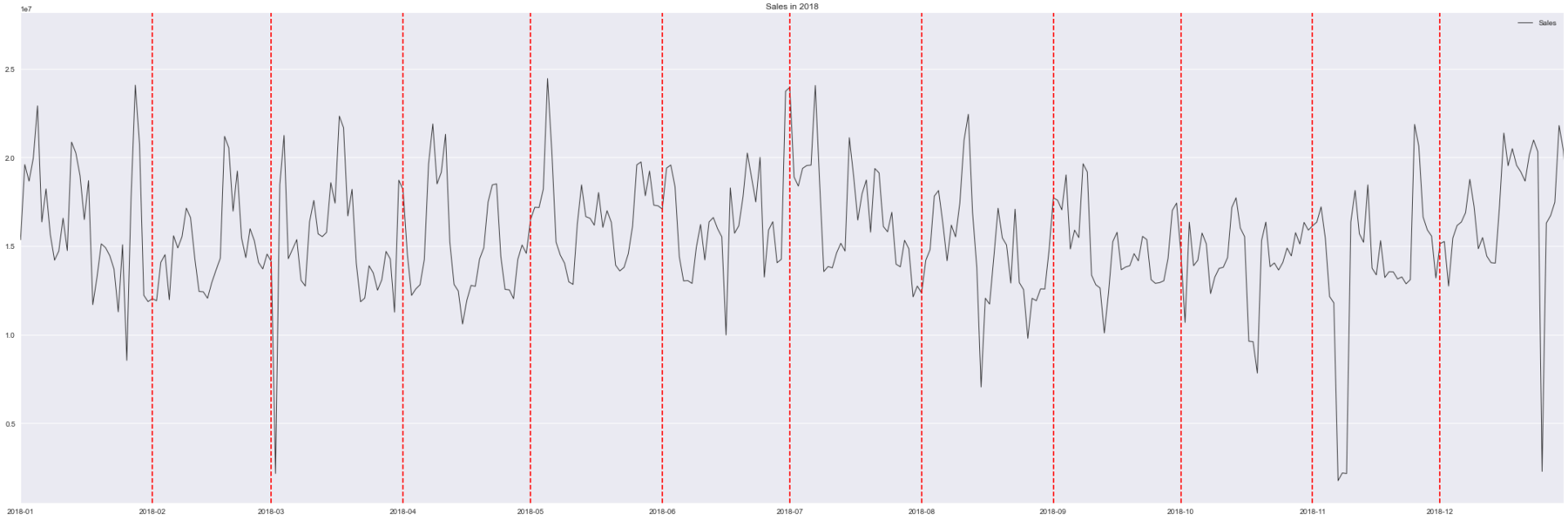
MR



Adjusted Linear Regression

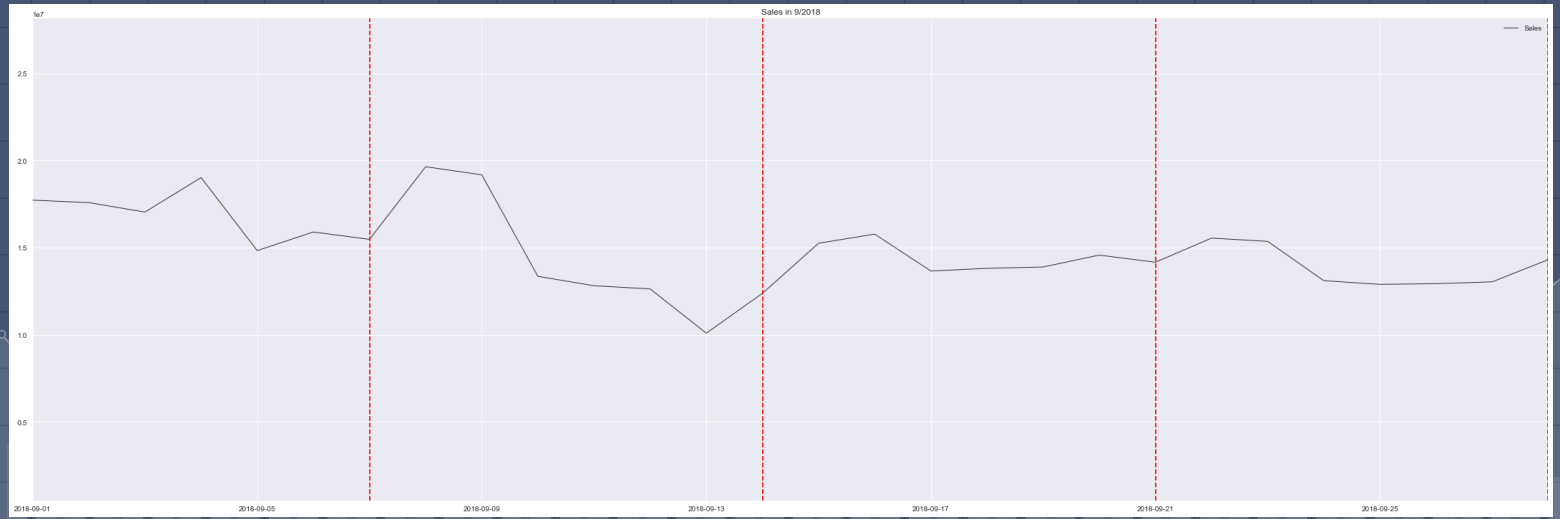
we use the first 9 months of 2018 as the training set and the last 3 months as the test set

Season Detection



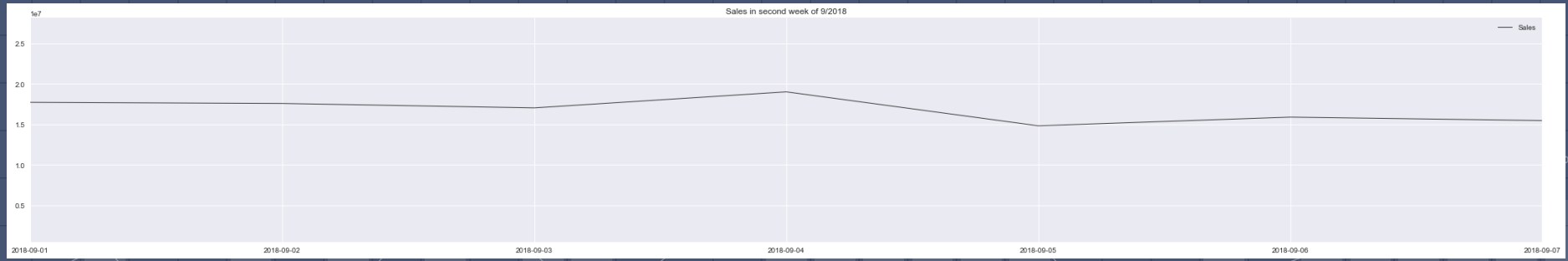
Season Detection

```
r = np.random.randint(1,12)
plt.figure(figsize=(30,10))
plt.style.use('seaborn')
plt.plot(df['Date'] , df['Sales'] , alpha=0.8 , c='black',label='Sales',linewidth=1)
plt.legend()
plt.tight_layout()
xcoords = [pd.Timestamp(f'2018-{r}-07'),pd.Timestamp(f'2018-{r}-14'),pd.Timestamp(f'2018-{r}-21'),pd.Timestamp(f'2018-{r}-28')]
for xc in xcoords:
    plt.axvline(x=xc, color='red', linestyle='--')
plt.xlim((pd.Timestamp(f'2018-{r}-01'),pd.Timestamp(f'2018-{r}-28')))
plt.title(f'Sales in {r}/2018')
plt.show()
```



Season Detection

```
plt.figure(figsize=(30,5))
plt.style.use('seaborn')
plt.plot(df['Date'] , df['Sales'] , alpha=0.8 , c='black',label='Sales',linewidth=1)
plt.legend()
plt.tight_layout()
plt.xlim((pd.Timestamp(f'2018-{r}-01'),pd.Timestamp(f'2018-{r}-07')))
plt.title(f'Sales in second week of {r}/2018')
plt.show()
```



ALR

```
df_ALR = df.copy()
df_ALR["Month"] = df_ALR.Date.dt.month
df_ALR["Week"] = df_ALR.Date.dt.isocalendar().week
df_ALR = df_ALR.iloc[:273,:]
df_ALR
```

```
df_ALR_test = df.copy()
df_ALR_test = df_ALR_test.iloc[273:364,:]
df_ALR_test["Month"] = df_ALR_test.Date.dt.month
df_ALR_test["Week"] = df_ALR_test.Date.dt.isocalendar().week
df_ALR_test = df_ALR_test.groupby(['Month','Week'],as_index=False).sum()
df_ALR_test['Wnum'] = df_ALR_test.index + 40
df_ALR_test
```

	Month	Week	Sales	Wnum
0	10	40	1.005986e+08	40
1	10	41	1.023263e+08	41
2	10	42	9.024339e+07	42
3	10	43	1.006574e+08	43
4	10	44	4.732808e+07	44
5	11	44	6.512340e+07	45
6	11	45	6.454665e+07	46
7	11	46	1.049788e+08	47
8	11	47	1.012960e+08	48
9	11	48	8.191768e+07	49
10	12	48	3.038003e+07	50
11	12	49	1.135447e+08	51
12	12	50	1.115119e+08	52
13	12	51	1.385013e+08	53
14	12	52	1.152597e+08	54

ALR

```
df_ALR_grouped = df_ALR.groupby(['Month','Week'],as_index=True).sum()
df_ALR_grouped.drop((4,18),inplace=True)
df_ALR_grouped.drop((7,31),inplace=True)
```

```
Month_sum=[]
for i in range(1,10):
    Month_sum.append(df_ALR_grouped.loc[(i),'Sales'].sum())
Month_sum
```

```
Month_avg = []
for i in Month_sum:
    Month_avg.append(i/5)
Month_avg
```

```
[101333167.572,
 83517802.026,
 94361441.33399999,
 87592949.382,
 103548057.12,
 97905540.79200001,
 100449984.504,
 90566094.75,
 90059774.76]
```

```
[506665837.86,
 417589010.13,
 471807206.66999996,
 437964746.90999997,
 517740285.6,
 489527703.96000004,
 502249922.52,
 452830473.75,
 450298873.8]
```

ALR

```
for i in range(1,10):
    for j in df_ALR_grouped.loc[(i),"Sales"].index.values:
        df_ALR_grouped.loc[(i,j),"AVG"] =
df_ALR_grouped.loc[(i,j),"Sales"]/Month_avg[i-1]
```

```
k=1
for i in range(1,10):
    for j in df_ALR_grouped.loc[(i),"Sales"].index.values:
        df_ALR_grouped.loc[(i,j),"WeekN"] = k
        k = k+1
```

		Sales	AVG	WeekN
Month	**Week**			
1	**1**	1.310205e+08	1.292968	1.0
	2	1.169641e+08	1.154253	2.0
	3	1.090989e+08	1.076636	3.0
	4	1.047842e+08	1.034056	4.0
	5	4.479807e+07	0.442087	5.0
2	**5**	5.252265e+07	0.628880	6.0
	6	1.059914e+08	1.269088	7.0
	7	9.896622e+07	1.184972	8.0
	8	1.177914e+08	1.410375	9.0
	9	4.231732e+07	0.506686	10.0
3	**9**	5.597109e+07	0.593156	11.0
	10	1.041874e+08	1.104131	12.0
	11	1.269531e+08	1.345392	13.0
	12	1.001648e+08	1.061502	14.0
	13	8.453076e+07	0.895819	15.0

ALR

```
Weekly_mean = []
for i in range(0,5):
    Weekly_mean.append(df_ALR_grouped.iloc[:,1].values[range(i,i+41,5)].mean())
Weekly_mean
```

```
[0.6530814228323787,
 1.2106596569944896,
 1.1330131600811313,
 1.1640668684552684,
 0.8391788916367323]
```

```
X = df_ALR_grouped.iloc[:,2:]
X_test = df_ALR_test.iloc[:,3:]
y = df_ALR_grouped.iloc[:,0:1]
regressor = linear_model.LinearRegression()
regressor.fit(X,y)
df_ALR_grouped["Predicted"] = regressor.predict(X)
df_ALR_test["Predicted"] = regressor.predict(X_test)
print('y = {:.2f}X + {:.2f}'.format(regressor.coef_[0][0],regressor.intercept_[0]))
print('Correlation coefficient for X and y is
{:.4f}'.format(np.corrcoef(X['WeekN'].values,y['Sales'].values)[0,1]))
```

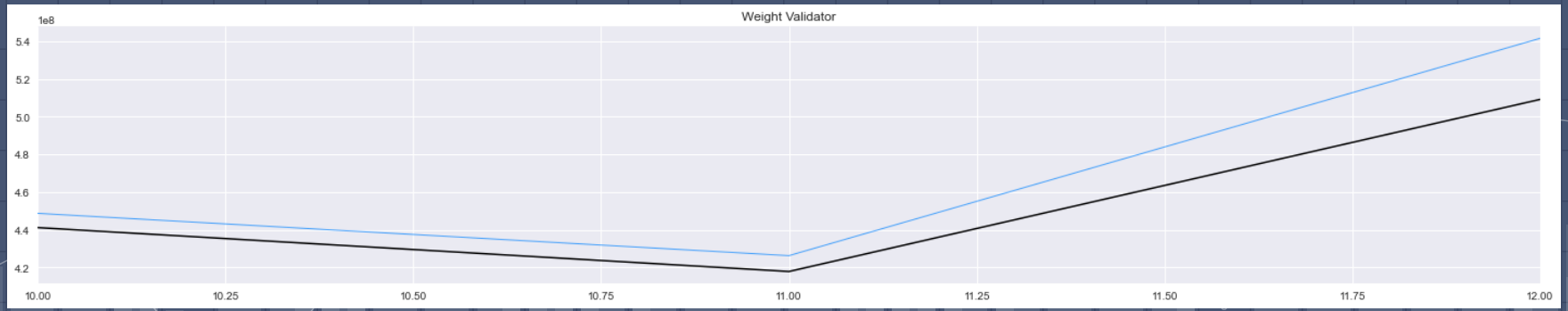
```
y = 28933.57X + 93705062.62
Correlation coefficient for X and y is 0.0129
```

ALR

```
adjuster = []
for i in range(0,3):
    adjuster.extend(Weekly_mean)
df_ALR_test["Weights"] = adjuster
df_ALR_test["adj_Predicted"] = df_ALR_test["Weights"]*df_ALR_test["Predicted"]
df_ALR_test["Weight_Valid"] = df_ALR_test["Weights"]*df_ALR_test["Sales"]
df_ALR_test
```

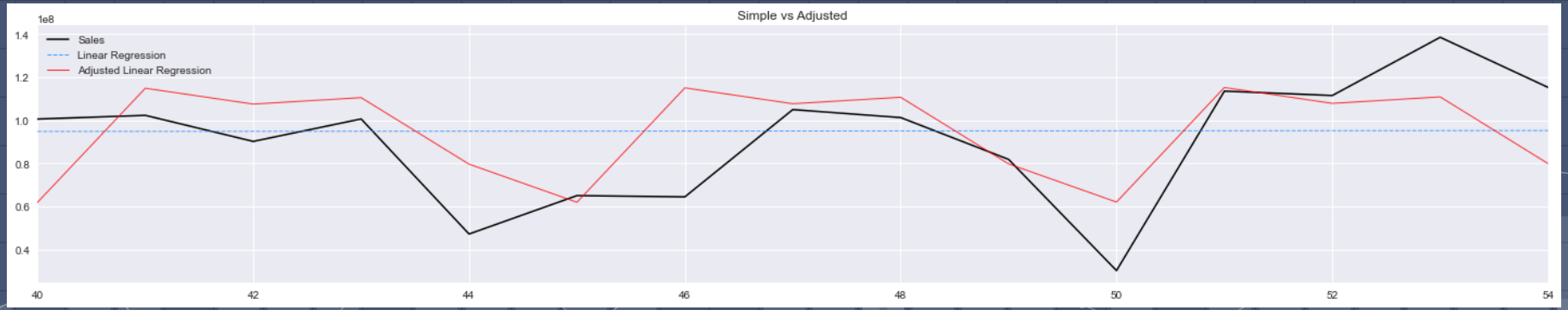
	Month	Week	Sales	Wnum	Predicted	Weights	adj_Predicted	Weight_Valid
0	10	40	1.005986e+08	40	9.486241e+07	0.653081	6.195287e+07	6.569906e+07
1	10	41	1.023263e+08	41	9.489134e+07	1.210660	1.148811e+08	1.238824e+08
2	10	42	9.024339e+07	42	9.492027e+07	1.133013	1.075459e+08	1.022469e+08
3	10	43	1.006574e+08	43	9.494921e+07	1.164067	1.105272e+08	1.171720e+08
4	10	44	4.732808e+07	44	9.497814e+07	0.839179	7.970365e+07	3.971673e+07
5	11	44	6.512340e+07	45	9.500707e+07	0.653081	6.204735e+07	4.253088e+07
6	11	45	6.454665e+07	46	9.503601e+07	1.210660	1.150563e+08	7.814403e+07
7	11	46	1.049788e+08	47	9.506494e+07	1.133013	1.077098e+08	1.189424e+08
8	11	47	1.012960e+08	48	9.509387e+07	1.164067	1.106956e+08	1.179153e+08
9	11	48	8.191768e+07	49	9.512281e+07	0.839179	7.982505e+07	6.874359e+07
10	12	48	3.038003e+07	50	9.515174e+07	0.653081	6.214183e+07	1.984063e+07
11	12	49	1.135447e+08	51	9.518067e+07	1.210660	1.152314e+08	1.374640e+08
12	12	50	1.115119e+08	52	9.520961e+07	1.133013	1.078737e+08	1.263445e+08
13	12	51	1.385013e+08	53	9.523854e+07	1.164067	1.108640e+08	1.612247e+08
14	12	52	1.152597e+08	54	9.526748e+07	0.839179	7.994645e+07	9.672354e+07

Weights Validation



ALR

```
plt.figure(figsize=(20,4))
plt.plot(df_ALR_test["Wnum"],df_ALR_test["Sales"],alpha=1 , c='black',label='Sales',linewidth=1.5)
plt.plot(df_ALR_test["Wnum"],df_ALR_test["Predicted"], alpha=0.8 , c='dodgerblue',ls='--',label='Linear Regression',linewidth=1)
plt.plot(df_ALR_test["Wnum"],df_ALR_test["adj_Predicted"], alpha=0.8 , c='red',label='Adjusted Linear Regression',linewidth=1)
plt.title("Simple vs Adjusted")
plt.xlim((40,54))
plt.tight_layout()
plt.legend()
plt.show()
```



ALR

```
def AdjReg_plot():
    plt.figure(figsize=(20,4))
    plt.plot(df_ALR_test["Month"],df_ALR_test["Sales"],alpha=1 , c='black',label='Sales',linewidth=1.5)
    plt.plot(df_ALR_test["Month"],df_ALR_test["adj_Predicted"], alpha=0.8 , c='red',label='Adjusted Linear Regression',linewidth=1)
    plt.plot(df_SLR_test["Month"],df_SLR_test["Predicted"], alpha=0.8 , c='dodgerblue',label='Simple Linear Regression',linewidth=1)
    plt.legend()
    plt.tight_layout()
    plt.title('Adjusted Linear Regression Test')
    plt.xlim((10,12))
    plt.show()
```

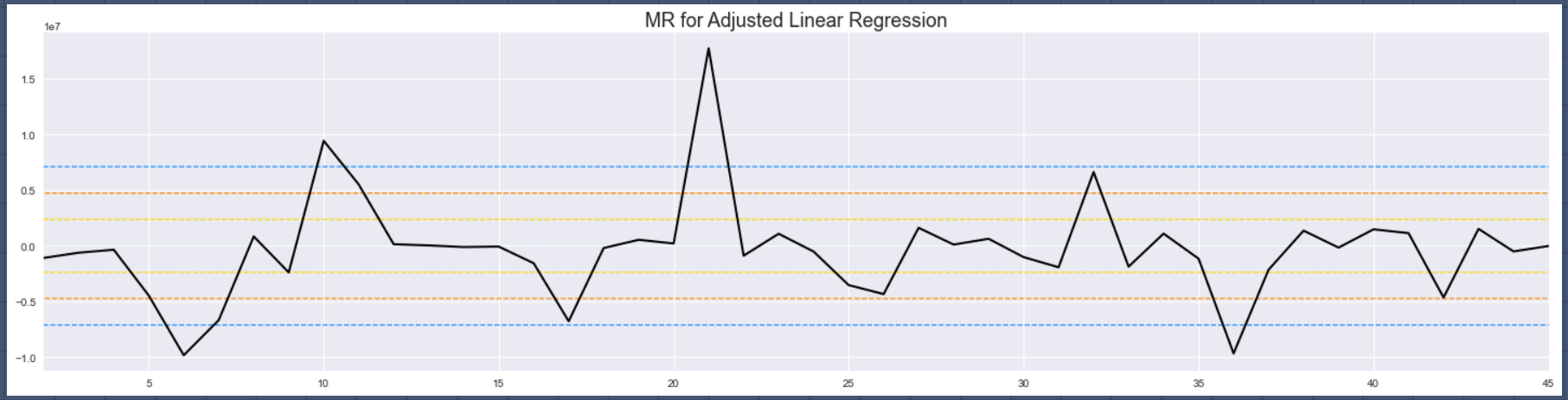
Assessing Model Accuracy

```
ME for Adjusted LR model is -19262766.54
MAE for Adjusted LR model is 41356257.99
MSE for Adjusted LR model is 1840208646081619.75
MPE for Adjusted LR model is -4.94
MAPE for Adjusted LR model is 9.28
TS for Adjusted LR model is -1.40
```

MR

```
a = res(df_ALR_grouped["Sales"],df_ALR_grouped["Adj_Predicted"])
n = len(a)
abs_MR = np.zeros(n)
MR = np.zeros(n)
for i in range(1,n):
    abs_MR[i] = abs(a.values[i]-a.values[i-1])
    MR[i] = (a.values[i]-a.values[i-1])
MR[0] = np.nan
abs_MR[0] = np.nan
MR = pd.concat([pd.DataFrame(MR,columns=['MR']),pd.DataFrame(abs_MR,columns=['abs_MR'])],axis=1)
MR.index += 1
MR["CL"] = np.zeros(n)
MR["LCL_1"] = np.zeros(n)
MR["UCL_1"] = np.zeros(n)
MR["LCL_2"] = np.zeros(n)
MR["UCL_2"] = np.zeros(n)
MR["LCL_3"] = np.zeros(n)
MR["UCL_3"] = np.zeros(n)
MR_bar = MR["abs_MR"].mean(skipna=True)
for i in range(2,n+1):
    MR["CL"][i] = 0
    MR["LCL_1"][i] = -0.89 * MR_bar
    MR["UCL_1"][i] =  0.89 * MR_bar
    MR["LCL_2"][i] = -1.77 * MR_bar
    MR["UCL_2"][i] =  1.77 * MR_bar
    MR["LCL_3"][i] = -2.66 * MR_bar
    MR["UCL_3"][i] =  2.66 * MR_bar
```

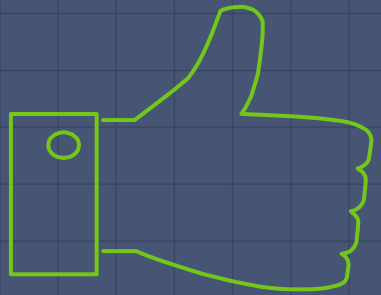
MR


MR for Adjusted Linear Regression
1e7
1.5
1.0
0.5
0.0
−0.5
−1.0
5
10
15
20
25
30
35
40
45

Conclusion

Mode\Metric	ME	MAE	MSE	MPE	MAPE	TS
LPD	749.75	1981625.42	9246233436020.38	-5.52	16.93	0.13
SMA, k=3	-9944.54	2322168.56	10382516871953.03	-7.40	20.21	-2.20
TMA, k=3	-7369.23	2107799.19	9717888957475.73	-5.61	17.92	-1.79
WMA, k=3	-10920.74	2773929.18	14051648202334.56	-8.46	23.84	-2.02
SES, $\alpha = 0.9$	-7596.69	1972829.34	8941183830481.47	-5.77	16.96	-1.98
TES, $\alpha = 0.9, \beta = 0.1$	-4050.18	2072820.00	9814619466751.69	-5.33	17.60	-1.01
Seasonal ES $\alpha = 0.9, c = 0.7$	-27689.81	301327.87	176007343894.48	-0.85	2.49	-44.66
SLR	497.75	1981625.42	9246233436020.38	-5.52	16.93	0.13
ALR	-19262766.54	41356257.99	1840208646081619.75	-4.94	9.28	-1.40

THANKS!



Pedram Peiro Asfia 9825006
Mahdi Mohammadi 9825041