Start Date: Sat, Oct 20th, 2018

Structure:

1. Importing the Libraries

2. Creating the Data Set

• a. FUNCTIONS for columns

• b. Data Set

• c. Columns using LAMBDA Functions

• d. Complete Data Set

3. EXPLORATORY DATA ANALYSIS

• a. Questions and Answers

4. DATA VISUALISATION

• a. LINE PLOT : Revenue over Months

• b. BOX PLOT : Revenue over Years

• c. VIOLIN PLOT

• d. LINE PLOT : Total Number of Communities growth over MOM for both Models.

e. Line Plot : Total number of leads MOM.

f. Growth Percentage for Communities in Both Model

```
a. Calculations
b. Plot
```

In [ ]:

1. Importing the Libraries¶

Throughout this notebook. I have used **plotly, which is a library built on top of d3.js that has a steep-learning curve JavaScript library**. There are Plotly API for Matlab, R, Python that helps us to create interactive visuals and dashboards. In other words, we can manipulate data manipulations in Pandas DataFrame and create interactive visual works easily.

In [79]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import sklearn
import plotly.figure_factory as ff
import plotly.offline as py
import plotly.graph_objs as go

import cufflinks as cf

from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
from plotly import tools

py.init_notebook_mode(connected=True)
cf.go_offline()
```

In [ ]:

In [ ]:

2. Creating the Data Set:

a. Creating the functions for the growth of models and creating columns for :

• Total number of ***NEW CONSTRUCTION COMMUNITIES for the PAY PER LEAD MODEL***

• Total number communities for ***$ 400 PRICE PER COMMUNITy MODEL***

• LEADS PER COMMUNITY PER MONTH

In [108]:
```
# 1. Function for multiplying the Month over Month growth for years: 2016, 2017 and 2018.

def comm(num,a,b,c):
    out = []
    for i in range(1,37):
        if i in range(12):
            out.append(round(num,2))
            while i:
                num = num+(num*a)
                break
        elif i in range(12,24):
            out.append(round(num,2))
            while i:
                num = num+(num*b)
                break
        else:
            out.append(round(num,2))
            num = num+(num*c)

    return(out)



# 2. Creating Columns based on Financial Planning and Analysis Team Research.

# a. Total number of NEW CONSTRUCTION COMMUNITIES for the PAY PER LEAD MODEL:
comm_lead = comm(6174,0.06,0.04,0.02)


# b. Total number of NEW CONSTRUCTION COMMUNITIES for $ 400 PRICE PER COMMUNITy MODEL:
comm_400 = comm(6174,0.054,0.036,0.018)


# c. LEADS PER COMMUNITY PER MONTH MODEL
leads_comm = comm(4,0.05,0.04,0.01)

# d. Date col:
date = ["Jan'2016", "Feb'2016", "Mar'2016", "April'2016", "May'2016", "June'2016", "July'2016",
        "Aug'2016", "Sept'2016", "Oct'2016", "Nov'2016", "Dec'2016",
        "Jan'2017", "Feb'2017", "Mar'2017", "April'2017", "May'2017", "June'2017", "July'2017",
        "Aug'2017", "Sept'2017", "Oct'2017", "Nov'2017", "Dec'2017",
        "Jan'2018", "Feb'2018", "Mar'2018", "April'2018", "May'2018", "June'2018", "July'2018",
        "Aug'2018", "Sept'2018", "Oct'2018", "Nov'2018", "Dec'2018"]
```

b. Creating The Data Frame:

In [81]:
```
# Creating the DICTIONARY to pass in the Data Frame:

d = {"Dates":date, "comm_400":comm_400, "comm_lead": comm_lead,"leads_comm":leads_comm}


# Data Frame using Pandas:
dfz = pd.DataFrame(data=d)

dfz.head(5)
```

Out[81]:

	Dates	comm_400	comm_lead	leads_comm
0	Jan'2016	6174.00	6174.00	4.00
1	Feb'2016	6507.40	6544.44	4.20
2	Mar'2016	6858.80	6937.11	4.41
3	April'2016	7229.17	7353.33	4.63
4	May'2016	7619.55	7794.53	4.86

In above Data Frame we have 4 Columns:

• Month Column

• Total number of Communities under $400 model for each month.

• Total number of Communities for Pay Per Lead Model

• Column for LEADS PER COMMUNITY PER MONTH

In [ ]:

c. Creating New Columns using Lambda Expression:

In [82]:
```
# Creating a fuction to CALCULATE THE TOTAL NUMBER OF LEADS:
def total_lead(cols):
    lead = cols[0]
    comm_l = cols[1]
    return (round(lead*comm_l,2))

# e. Creating the column for TOTAL NUMBER OF LEADS PER MONTH:
dfz["total_leads"] = dfz[["comm_lead", "leads_comm"]].apply(lambda x: total_lead(x),axis=1)


# f. Year column in order to create a boxplot in next step:
dfz["Years"]= dfz["Dates"].apply(lambda x: x.split("'")[1])


# g. Creating a Column for TOTAL PRICE PER COMMUNITY PER MONTH UNDER $400 MODEL:
dfz["price_comm400"]= dfz["comm_400"].apply(lambda x: x * 400)


# h. Creating a column for TOTAL PRICE FOR LEADS PER MONTH MODEL
dfz["price_leads"]= dfz["total_leads"].apply(lambda x: x * 40)
```

In [ ]:

d. View of Complete Data Set:

In [83]:
```
dfz
```

Out[83]:

	Dates	comm_400	comm_lead	leads_comm	total_leads	Years	price_comm400	price_leads
0	Jan'2016	6174.00	6174.00	4.00	24696.00	2016	2469600.0	987840.0
1	Feb'2016	6507.40	6544.44	4.20	27486.65	2016	2602960.0	1099466.0
2	Mar'2016	6858.80	6937.11	4.41	30592.66	2016	2743520.0	1223706.4
3	April'2016	7229.17	7353.33	4.63	34045.92	2016	2891668.0	1361836.8
4	May'2016	7619.55	7794.53	4.86	37881.42	2016	3047820.0	1515256.8
5	June'2016	8031.00	8262.20	5.11	42219.84	2016	3212400.0	1688793.6
6	July'2016	8464.68	8757.94	5.36	46942.56	2016	3385872.0	1877702.4
7	Aug'2016	8921.77	9283.41	5.63	52265.60	2016	3568708.0	2090624.0
8	Sept'2016	9403.54	9840.42	5.91	58156.88	2016	3761416.0	2326275.2
9	Oct'2016	9911.33	10430.84	6.21	64775.52	2016	3964532.0	2591020.8
10	Nov'2016	10446.55	11056.69	6.52	72089.62	2016	4178620.0	2883584.8
11	Dec'2016	11010.66	11720.10	6.84	80165.48	2016	4404264.0	3206619.2
12	Jan'2017	11407.04	12188.90	7.12	86784.97	2017	4562816.0	3471398.8
13	Feb'2017	11817.70	12676.46	7.40	93805.80	2017	4727080.0	3752232.0
14	Mar'2017	12243.13	13183.51	7.70	101513.03	2017	4897252.0	4060521.2
15	April'2017	12683.89	13710.85	8.00	109686.80	2017	5073556.0	4387472.0
16	May'2017	13140.51	14259.29	8.32	118637.29	2017	5256204.0	4745491.6
17	June'2017	13613.57	14829.66	8.66	128424.86	2017	5445428.0	5136994.4
18	July'2017	14103.65	15422.85	9.00	138805.65	2017	5641460.0	5552226.0
19	Aug'2017	14611.39	16039.76	9.36	150132.15	2017	5844556.0	6005286.0
20	Sept'2017	15137.39	16681.35	9.74	162476.35	2017	6054956.0	6499054.0
21	Oct'2017	15682.34	17348.60	10.13	175741.32	2017	6272936.0	7029652.8
22	Nov'2017	16246.91	18042.55	10.53	189988.05	2017	6498764.0	7599522.0
23	Dec'2017	16831.79	18764.25	10.95	205468.54	2017	6732716.0	8218741.6
24	Jan'2018	17134.77	19139.54	11.06	211683.31	2018	6853908.0	8467332.4
25	Feb'2018	17443.19	19522.33	11.17	218064.43	2018	6977276.0	8722577.2
26	Mar'2018	17757.17	19912.77	11.29	224815.17	2018	7102868.0	8992606.8
27	April'2018	18076.80	20311.03	11.40	231545.74	2018	7230720.0	9261829.6
28	May'2018	18402.18	20717.25	11.51	238455.55	2018	7360872.0	9538222.0
29	June'2018	18733.42	21131.59	11.63	245760.39	2018	7493368.0	9830415.6
30	July'2018	19070.62	21554.23	11.74	253046.66	2018	7628248.0	10121866.4
31	Aug'2018	19413.89	21985.31	11.86	260745.78	2018	7765556.0	10429831.2
32	Sept'2018	19763.34	22425.02	11.98	268651.74	2018	7905336.0	10746069.6
33	Oct'2018	20119.08	22873.52	12.10	276769.59	2018	8047632.0	11070783.6
34	Nov'2018	20481.23	23330.99	12.22	285104.70	2018	8192492.0	11404188.0
35	Dec'2018	20849.89	23797.61	12.34	293662.51	2018	8339956.0	11746500.4

In above Data Frame we have 8 Columns:

• Month Column as ("Dates")

• Total number of Communities under $400 model for each month as ("comm_400")

• Total number of Communities for Pay Per Lead Model as ("comm_leads")

• Column for LEADS PER COMMUNITY PER MONTH as ("leads_comm")

• Total number of LEADS PER MONTH as ("total_leads")

• Yeas as ("Years")

• Total price per Month for $400 PRICE PER COMMUNITY MODEL as ("price_comm400")

• Total price per Month for PAY PER LEAD MODEL as ("price_leads')

In [ ]:

In [ ]:

3. Exploratory Data Analysis:

In [84]:
```
# Brief Information about the Data set (Data Types, counts etc)
dfz.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36 entries, 0 to 35
Data columns (total 8 columns):
Dates            36 non-null object
comm_400         36 non-null float64
comm_lead        36 non-null float64
leads_comm       36 non-null float64
total_leads      36 non-null float64
Years            36 non-null object
price_comm400    36 non-null float64
price_leads      36 non-null float64
dtypes: float64(6), object(2)
memory usage: 2.3+ KB
```

Please note:

I have created the Date column as oject type intentionally for the purpose to apply strings methods ex: splice etc to create other new columns easily.

In [ ]:

In [85]:
```
# Describe Method
dfz.describe()
```

Out[85]:

	comm_400	comm_lead	leads_comm	total_leads	price_comm400	price_leads
count	36.000000	36.000000	36.000000	36.00000	3.600000e+01	3.600000e+01
mean	13759.537222	15111.228611	8.635833	145585.79250	5.503815e+06	5.823432e+06
std	4622.614791	5562.523396	2.802001	88885.10892	1.849046e+06	3.555404e+06
min	6174.000000	6174.000000	4.000000	24696.00000	2.469600e+06	9.878400e+05
25%	9784.382500	10283.235000	6.135000	63120.86000	3.913753e+06	2.524834e+06
50%	13858.610000	15126.255000	8.830000	133615.25500	5.543444e+06	5.344610e+06
75%	17837.077500	20012.335000	11.317500	226497.81250	7.134831e+06	9.059912e+06
max	20849.890000	23797.610000	12.340000	293662.51000	8.339956e+06	1.174650e+07

Note:

Above table gives a brief information for each column about the followings:

• **Minimum & Maximum values for each Columns**

• **Statistical information such as Mean, Standard Deviation and Quartiles**

In [ ]:

b. Question & Answer section:

In [86]:
```
# Total number of communities in each model in month of DECEMBER 2018.

dfz[dfz["Dates"] =="Dec'2018"][["comm_400", "comm_lead"]]
```

Out[86]:

	comm_400	comm_lead
35	20849.89	23797.61

In [87]:
```
# All Values for December 2018:

dfz[dfz["Dates"] =="Dec'2018"]
```

Out[87]:

	Dates	comm_400	comm_lead	leads_comm	total_leads	Years	price_comm400	price_leads
35	Dec'2018	20849.89	23797.61	12.34	293662.51	2018	8339956.0	11746500.4

In [88]:
```
# Total Number of communites and total leads per community in each year

dfz.groupby("Years").sum()[["comm_400", "comm_lead", "leads_comm"]]
```

Out[88]:

	comm_400	comm_lead	leads_comm
Years			
2016	100578.45	104155.01	63.68
2017	167519.31	183148.03	106.91
2018	227245.58	256701.19	140.30

In [ ]:

In [89]:
```
# Total revenue in Month of DECEMBER 2018 in each model

dfz[dfz["Dates"] =="Dec'2018"][["price_comm400","price_leads"]]
```

Out[89]:

	price_comm400	price_leads
35	8339956.0	11746500.4

In [111]:
```
# Total Revenue generated in each MODEL in every year in $.

dfz.groupby("Years").sum()[["price_comm400","price_leads"]]
```

Out[111]:

	price_comm400	price_leads
Years		
2016	40231380.0	22852726.0
2017	67007724.0	66458592.4
2018	90898232.0	120332222.8

In [114]:
```
# Total leads based on all communities in every year.

dfz.groupby("Years").sum()[["total_leads"]]
```

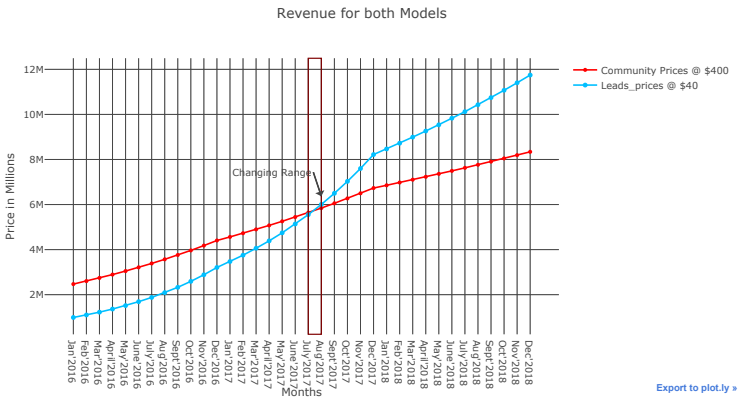
Out[114]:

	total_leads
Years	
2016	571318.15
2017	1661464.81
2018	3008305.57

4. Data Visualisation

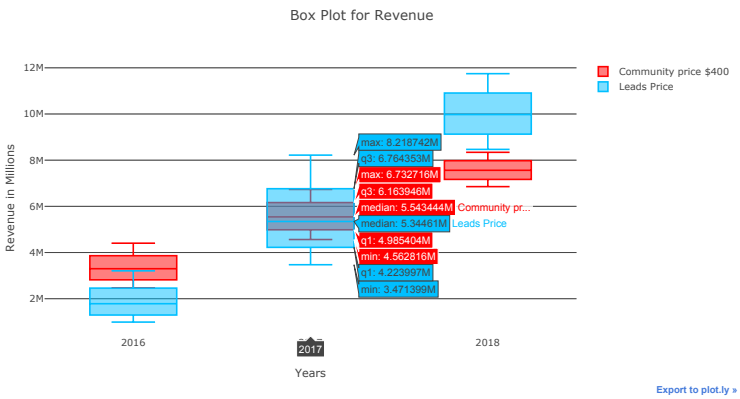
a. Line plot: Revenue over Months

In [91]:
```
trace = go.Scatter(x= dfz["Dates"],
                   y = dfz["price_comm400"],
                   mode = "lines+markers",
                   marker= dict(size =5),
                   name = "Community Prices @ $400",
                   line=dict(color='red'))


tracel = go.Scatter(x= dfz["Dates"],
                   y = dfz["price_leads"],
                   mode = "lines+markers",
                   marker= dict(size =6),
                   name = "Leads_prices @ $40",
                     line=dict(color='deepskyblue'))



layout = go.Layout(dict(title = "Revenue for both Models"),
                    yaxis=dict(title = "Price in Millions"),
                    xaxis=dict(title = "Months"),
                   annotations =[{
        'x': "Aug'2017",  'yref': 'paper',
         'showarrow': True, 'xanchor': 'right',
        'text': 'Changing Range'
    }],
                   shapes = [{
        'x0': "July'2017", 'x1': "Aug'2017",
        'y0': 0, 'y1': 1, 'yref': 'paper',
        'line': {'color': 'maroon', 'width': 1.5}
    }],)

data = [trace, tracel]

fig = go.Figure(data = data, layout= layout)

py.iplot(fig)
```



b. Box plot: Revenue in Years

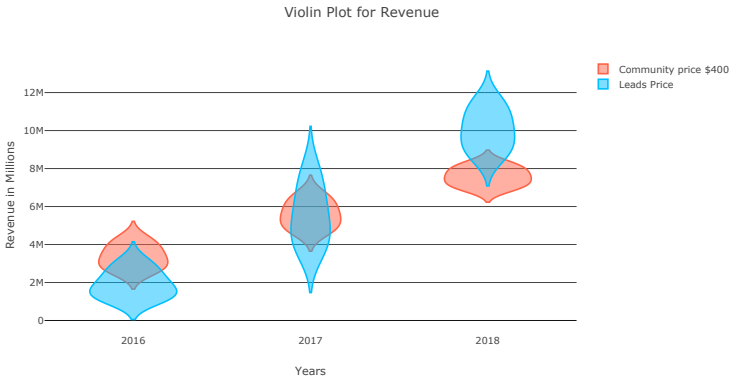
In [12]:
```
trace = go.Box(x= dfz["Years"],
               y = dfz["price_comm400"],
               name = "Community price $400",
               marker = dict(
        color = 'red'))

tracel = go.Box(x= dfz["Years"],
               y = dfz["price_leads"],
                name="Leads Price",marker = dict(
        color = 'deepskyblue'))

layout = go.Layout(dict(title = "Box Plot for Revenue"),
                    yaxis=dict(title = "Revenue in Millions"),
                    xaxis= dict(title = "Years"))

data = [trace, tracel]

fig = go.Figure(data= data , layout= layout)

py.iplot(fig)
```



c. Violin plot: Revenue in Years:

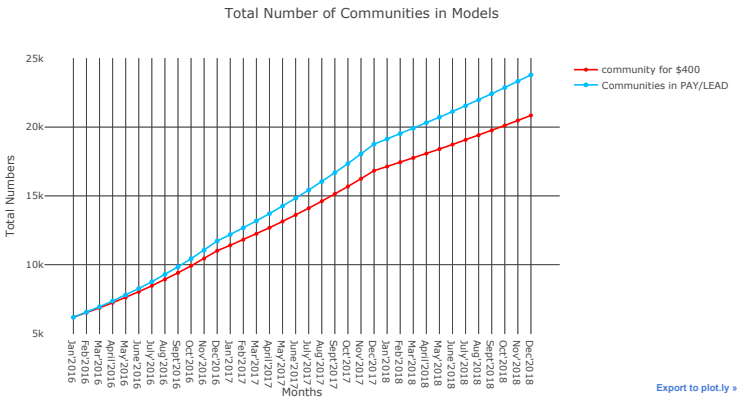
It is **similar to boxplot but have rotated kernel density plot on both sides**

In [13]:
```
trace = go.Violin(x= dfz["Years"],
                  y = dfz["price_comm400"],
                  name = "Community price $400",
                     marker = dict(
           color = 'tomato'))

tracel = go.Violin(x= dfz["Years"],
                  y = dfz["price_leads"],
                   name="Leads Price",
                      marker = dict(
           color = 'deepskyblue'))

layout = go.Layout(dict(title = "Violin Plot for Revenue"),
                        yaxis=dict(title = "Revenue in Millions"),
                       xaxis= dict(title = "Years"))

data = [trace, tracel]

fig = go.Figure(data= data , layout= layout)

py.iplot(fig)
```



d. Line plot: For total number of communities growth in both Models

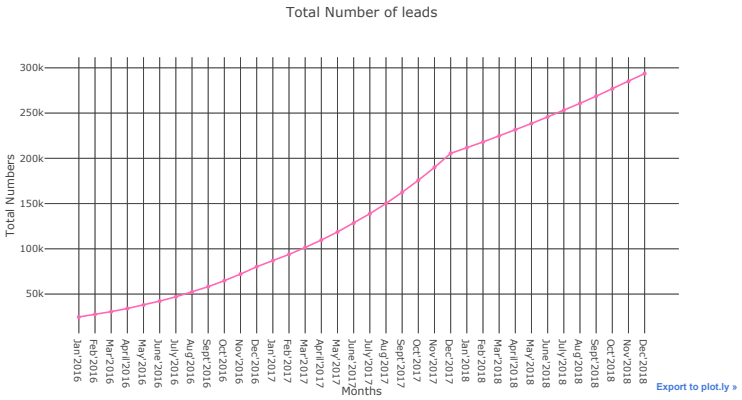
In [52]:
```
trace = go.Scatter(x= dfz["Dates"],
                   y = dfz["comm_400"],
                   mode = "lines+markers",
                   marker= dict(size =5),
                   name = "community for $400",
                   line=dict(color='red'))


tracel = go.Scatter(x= dfz["Dates"],
                   y = dfz["comm_lead"],
                   mode = "lines+markers",
                   marker= dict(size =6),
                   name = "Communities in PAY/LEAD",
                    line=dict(color='deepskyblue'))



layout = go.Layout(dict(title = "Total Number of Communities in Models"),
                    yaxis=dict(title = "Total Numbers"),
                   xaxis= dict(title = "Months"))

data = [trace, tracel,]

fig = go.Figure(data = data, layout= layout)

py.iplot(fig)
```



In [ ]:
```
```

e. Line Plot : Total number of leads MOM.

In [65]:
```
trace = go.Scatter(x= dfz["Dates"],
                   y = dfz["total_leads"],
                   mode = "lines+markers",
                   marker= dict(size =5),
                   name = "community for $400",
                   line=dict(color='hotpink'))

layout = go.Layout(dict(title = "Total Number of leads"),
                    yaxis=dict(title = "Total Numbers"),
                   xaxis= dict(title = "Months"))

data = [trace]

fig = go.Figure(data = data, layout= layout)

py.iplot(fig)
```



In [ ]:

f. Growth Percentage for Communities in Both Model

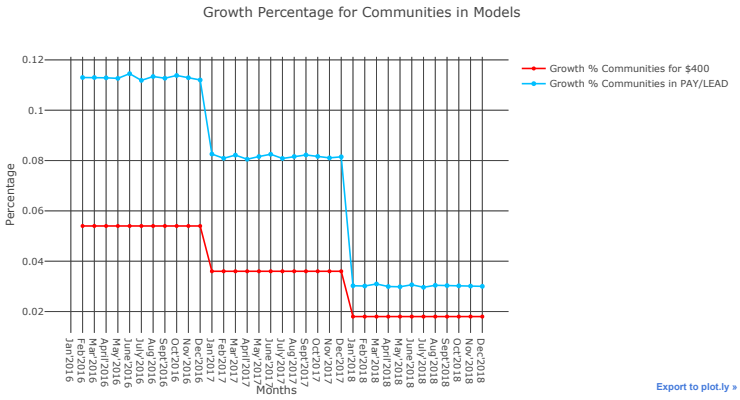
Using Pandas.DataFrame .pct_change( ) METHOD

In [109]:
```
# Using method to calculate growth percentage

print(dfz["price_leads"].pct_change())
print(dfz["price_comm400"].pct_change())
```

```
0          NaN
1     0.113000
2     0.113001
3     0.112879
4     0.112657
5     0.114526
6     0.111860
7     0.113395
8     0.112718
9     0.113807
10    0.112915
11    0.112025
12    0.082573
13    0.080899
14    0.082162
15    0.080519
16    0.081600
17    0.082500
18    0.080832
19    0.081600
20    0.082222
21    0.081642
22    0.081066
23    0.081481
24    0.030247
25    0.030145
26    0.030958
27    0.029938
28    0.029842
29    0.030634
30    0.029648
31    0.030426
32    0.030321
33    0.030217
34    0.030116
35    0.030016
Name: price_leads, dtype: float64
0          NaN
1     0.054001
2     0.054000
3     0.053999
4     0.054001
5     0.053999
6     0.054001
7     0.054000
8     0.053999
9     0.054000
10    0.054001
11    0.054000
12    0.036000
13    0.036001
14    0.035999
15    0.036001
16    0.036000
17    0.036000
18    0.035999
19    0.036001
20    0.035999
21    0.036000
22    0.036000
23    0.035999
24    0.018000
25    0.018000
26    0.018000
27    0.018000
28    0.018000
29    0.018000
30    0.018000
31    0.018000
32    0.018000
33    0.018000
34    0.018000
35    0.018000
Name: price_comm400, dtype: float64
```

f. b. Line Plot: For growth% for both models

In [107]:
```
trace = go.Scatter(x= dfz["Dates"],
                   y = dfz["price_comm400"].pct_change(),
                   mode = "lines+markers",
                   marker= dict(size =5),
                   name = "Growth % Communities for $400",
                   line=dict(color='red'))


tracel = go.Scatter(x= dfz["Dates"],
                   y = dfz["price_leads"].pct_change(),
                   mode = "lines+markers",
                   marker= dict(size =6),
                   name = "Growth % Communities in PAY/LEAD",
                    line=dict(color='deepskyblue'))



layout = go.Layout(dict(title = "Growth Percentage for Communities in Models"),
                    yaxis=dict(title = "Percentage"),
                   xaxis= dict(title = "Months"))

data = [trace, tracel,]

fig = go.Figure(data = data, layout= layout)

py.iplot(fig)
```



In [ ]:

End Data : Mon, Oct 22nd, 2018

Applicant : Piyush Rastogi

Position: Data Science Intern

Company : Zillow

In [ ]: