Google Ads Hourly Analysis

Date : 19th June 2023

Project Start Date - End Date	• Start Date – 19 -06 -2023 • End Date – 20–06-2023
Objectives	• To analyses how many people who clicked on the advertisement highly interested to enroll in our course • General exploratory analyses • General descriptive analyses
Milestones accomplished the week of Start Date - End Date:	• Descriptive analyses • Exploratory analyses • Classification of date with respect to term

Contact Information

This project is performed for educational purpose of under the guidance of Siddhivinayak Sir .

Project Manager
Name : Siddhivinayak Phulwadkar
Mobile: 9028965955
Email:
siddhivinayakphulwadkar@gmail.com

Project Traine
Name : Arti Sukhadev Patil
Mobile: 8623845944
Email: parti5863@gmail.com

Project Abstract

The dataset is about showing advertisement to students or clients for course enrollment. Our main objective was to understand on which time students are clicked on our ads and getting enroll our course or to interest to buy course. Problem statement is classify as we are looking for preferred timing in a day and night where we can do marketing and we will get definitely sales or hot lead. For this dataset we have applied decision tree algorithm and performed exploratory and descriptive analysis.

Google Ads Hourly Analysis

#Importing libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

#Importing dataset

```
file=pd.read_excel("C:/Users/Administrator/Desktop/Google ads hourly analysis 19th june.xlsx")
```

```
file.head()
```

	Sr no	Impressions	Clicks	Cost	CTR	CPC	Cold Leads	Warm Leads	Hot Leads
0	00:00:00	5941	356	594	0.06	1.666667	11	5.0	2
1	00:30:00	4511	180	451	0.04	2.500000	9	4.0	2
2	01:00:00	3378	101	338	0.03	3.333333	6	3.0	1
3	01:30:00	652	26	65	0.04	2.500000	1	NaN	0
4	02:00:00	421	8	42	0.02	5.000000	0	0.0	0

#Preprocessing the dataset

```
dataset=file.drop("Sr no",axis=1)
```

```
dataset2=dataset1.drop(["CPC","CTR","Warm Leads"],axis=1)
```

```
dataset2.head()
```

	Impressions	Clicks	Cost	Cold Leads	Hot Leads
0	5941	356	594	11	2
1	4511	180	451	9	2
2	3378	101	338	6	1
3	652	26	65	1	0
4	421	8	42	0	0

```
dataset.isnull().sum()
```

```
Impressions     0
Clicks          0
Cost            0
CTR             0
CPC             0
Cold Leads      0
Warm Leads      1
Hot Leads       0
dtype: int64
```

```
dataset1=dataset.fillna({"Warm Leads":0.0})
```

```
dataset1.isnull().sum()
```

```
Impressions     0
Clicks          0
Cost            0
CTR             0
CPC             0
Cold Leads      0
Warm Leads      0
Hot Leads       0
dtype: int64
```

#Descriptive analysis

```
# descriptive analysis
```

```
dataset1.sum()
```

```
Impressions     200522.000000
Clicks           31584.000000
Cost             20057.000000
CTR                  5.206900
CPC                145.459869
Cold Leads         392.000000
Warm Leads         184.000000
Hot Leads           73.000000
dtype: float64
```

```
#Impressions are no of visible ads to customers
#Total impressions are 200522.000000
#Clicks indicates total no of customers who clicked on our ads
#total no of clicks ad 31584.000000
```

```
#Cost indicates cost per click and impression for ads
#Total cost are  20057.000000
```

```
#CTR indicates the no of clicks that your ad recives divided by the no of times your ad is shown
#Total CTR is 5.206900
```

```
#CPC indicates dividing the ads cost by the number of clicks generated by an ads
#total CPC is 145.459869
```

```
#Cold lead indicates those  customers shows only intrest in Ads
#Warm lead indicates those customers are showing intrest in Ads and providing their contact details
#Hot lead indicates those customers are highly intrested and are ready to buy
```

```
#Total no of Cold lead are 392.000000
#Total no of Warm lead are 184.000000
#Total no of Hot lead are 73.000000
```

```
dataset1.mean()
```

```
Impressions     4177.541667
Clicks           658.000000
Cost             417.854167
CTR                0.108477
CPC                3.030414
Cold Leads         8.166667
Warm Leads         3.833333
Hot Leads          1.520833
dtype: float64
```

```
#the average no of Impression are 4177.541667
#the average no of Clicks are 658.000000
#the average no pf cost is 417.854167
#the average no of CTR is 0.108477
#the average no of CPC is 3.030414
#the average no of cold lead  8.166667
#the average no of warm lead  3.833333
#the average no of hot lead  1.520833
```

```
#Individual ratio
#Individual ratio of cold lead and warm lead is 45.45%
#Individual ratio of warm lead and hot lead is 40%
#Individual ratio of cold lead and hot lead is 18.18%
```

```
#On an average ratio
#on an average ratio of cold lead and warm lead is 46.93%
#on an average ratio of warm lead and hot lead is 39.67%
#on an average ratio of cold lead and hot lead is 18.62%
```

```
x=dataset2.iloc[:,:-1].values
y=dataset2.iloc[:,-1].values
```

```
x
```

```
array([[ 5941,   356,   594,    11],
       [ 4511,   180,   451,     9],
       [ 3378,   101,   338,     6],
       [  652,    26,    65,     1],
       [  421,     8,    42,     0],
       [  110,     2,    11,     0],
       [   56,     1,     6,     0],
       [   42,     1,     4,     0],
       [    3,     0,     0,     0],
       [    8,     0,     1,     0],
       [   95,     1,    10,     0],
       [   64,     1,     6,     0],
       [   26,     0,     3,     0],
       [  193,     4,    19,     0],
       [  236,     5,    24,     0],
       [  463,    23,    46,     1],
       [  896,    54,    90,     2],
       [  486,    34,    49,     1],
       [  785,    71,    79,     1],
       [ 1245,   149,   125,     2],
       [ 1755,   228,   176,     3],
       [ 1865,   106,   187,     4],
       [ 2658,   399,   266,     5],
       [ 3255,   391,   326,     7],
       [ 3284,   427,   328,     6],
       [ 4651,   558,   465,     9],
       [ 4895,   685,   490,    10],
       [ 4752,   665,   475,     9],
       [ 4665,   606,   467,     9],
       [ 4958,   694,   496,    10],
```

```
y
```

```
array([2, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 1, 2, 3, 3, 4, 3, 3, 4, 4, 4, 3,
       3, 4, 4, 4], dtype=int64)
```

#Training the model

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.10, random_state =0)
```

```
print(x_train)
```

```
[[ 8452  1944   845    17]
 [ 6593  1055   659    13]
 [ 9425  1414   943    19]
 [   42     1     4     0]
 [   95     1    10     0]
 [   64     1     6     0]
 [ 5968   836   597    12]
 [ 4865  1216   487     9]
 [ 4752   665   475     9]
 [ 9452  1512   945    19]
 [ 3378   101   338     6]
 [10230  1739  1023    20]
 [  785    71    79     1]
 [  463    23    46     1]
 [ 4665   606   467     9]
 [ 2658   399   266     5]
 [  896    54    90     2]
 [ 9120  1277   912    18]
 [ 1755   228   176     3]
 [ 8945  1431   895    18]
 [    3     0     0     0]
 [  193     4    19     0]
 [ 4651   558   465     9]
 [  110     2    11     0]
 [  486    34    49     1]
 [ 8945  1700   895    18]
 [  236     5    24     0]
 [ 8757  2277   876    17]
 [ 4511   180   451     9]
```

```
print(y_train)
```

```
[4 2 4 0 0 0 2 1 2 4 1 4 0 0 2 1 0 4 0 4 0 0 1 0 0 3 0 3 2 0 3 1 0 1 3 1 0
 0 3 4 0 2 3]
```

```
print(x_test)
```

```
[[4958  694  496   10]
 [ 421    8   42    0]
 [4895  685  490   10]
 [5320  745  532   10]
 [5673  794  567   11]]
```

```
print(y_test)
```

```
[2 0 2 2 2]
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

```
print(x_train)
```

```
[[ 1.18213428e+00  1.79469516e+00  1.18142786e+00  1.21259168e+00]
 [ 6.69072303e-01  5.45818109e-01  6.68031498e-01  6.63734395e-01]
 [ 1.45067075e+00  1.05014529e+00  1.45192702e+00  1.48702033e+00]
 [-1.13892609e+00 -9.34852767e-01 -1.13989654e+00 -1.12005179e+00]
 [-1.12429872e+00 -9.34852767e-01 -1.12333537e+00 -1.12005179e+00]
 [-1.13285435e+00 -9.34852767e-01 -1.13437615e+00 -1.12005179e+00]
 [ 4.96579707e-01  2.38164483e-01  4.96899378e-01  5.26520073e-01]
 [ 1.92164774e-01  7.71992692e-01  1.93277875e-01  1.14877107e-01]
 [ 1.60978113e-01 -2.05821158e-03  1.60155529e-01  1.14877107e-01]
 [ 1.45812243e+00  1.18781677e+00  1.45744741e+00  1.48702033e+00]
 [-2.18229609e-01 -7.94371660e-01 -2.17991253e-01 -2.96765859e-01]
 [ 1.67284121e+00  1.50670889e+00  1.67274265e+00  1.62423465e+00]
 [-9.33866890e-01 -8.36515992e-01 -9.32881884e-01 -9.82837470e-01]
 [-1.02273507e+00 -9.03946924e-01 -1.02396833e+00 -9.82837470e-01]
 [ 1.36967144e-01 -8.49420652e-02  1.38073965e-01  1.14877107e-01]
 [-4.16941079e-01 -3.75737958e-01 -4.16725328e-01 -4.33980182e-01]
 [-9.03232205e-01 -8.60397780e-01 -9.02519733e-01 -8.45623148e-01]
 [ 1.36649437e+00  8.57686168e-01  1.36636096e+00  1.34980601e+00]
 [-6.66158381e-01 -6.15960653e-01 -6.65142921e-01 -7.08408826e-01]
 [ 1.31819644e+00  1.07402707e+00  1.31943763e+00  1.34980601e+00]
 [-1.14968963e+00 -9.36257579e-01 -1.15093733e+00 -1.12005179e+00]
 [-1.09725188e+00 -9.30638334e-01 -1.09849361e+00 -1.12005179e+00]
 [ 1.33103310e-01 -1.52372997e-01  1.32553574e-01  1.14877107e-01]
 [-1.12015889e+00 -9.33447956e-01 -1.12057518e+00 -1.12005179e+00]
 [-1.01638735e+00 -8.88494002e-01 -1.01568775e+00 -9.82837470e-01]
 [ 1.31819644e+00  1.45192125e+00  1.31943763e+00  1.34980601e+00]
 [-1.08538439e+00 -9.29233523e-01 -1.08469264e+00 -1.12005179e+00]
 [ 1.26631067e+00  2.26249725e+00  1.26699392e+00  1.21259168e+00]
 [ 9.44649682e-02 -6.83391584e-01  9.39108371e-02  1.14877107e-01]
```

```
print(x_test)
```

```
[[ 0.21783167  0.03868131  0.21811963  0.25209143]
 [-1.03432658 -0.92501909 -1.03500912 -1.12005179]
 [ 0.20044442  0.02603801  0.20155846  0.25209143]
 [ 0.31773938  0.11032667  0.31748667  0.25209143]
 [ 0.4151632   0.17916242  0.41409351  0.38930575]]
```

```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'gini', random_state = 0)
classifier.fit(x_train, y_train)
```

```
DecisionTreeClassifier(random_state=0)
```

```
print(classifier.predict(sc.transform([[3378,101,338,6]])))
```

```
[1]
```

```
y_pred = classifier.predict(x_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

```
[[2 2]
 [0 0]
 [2 2]
 [2 2]
 [2 2]]
```

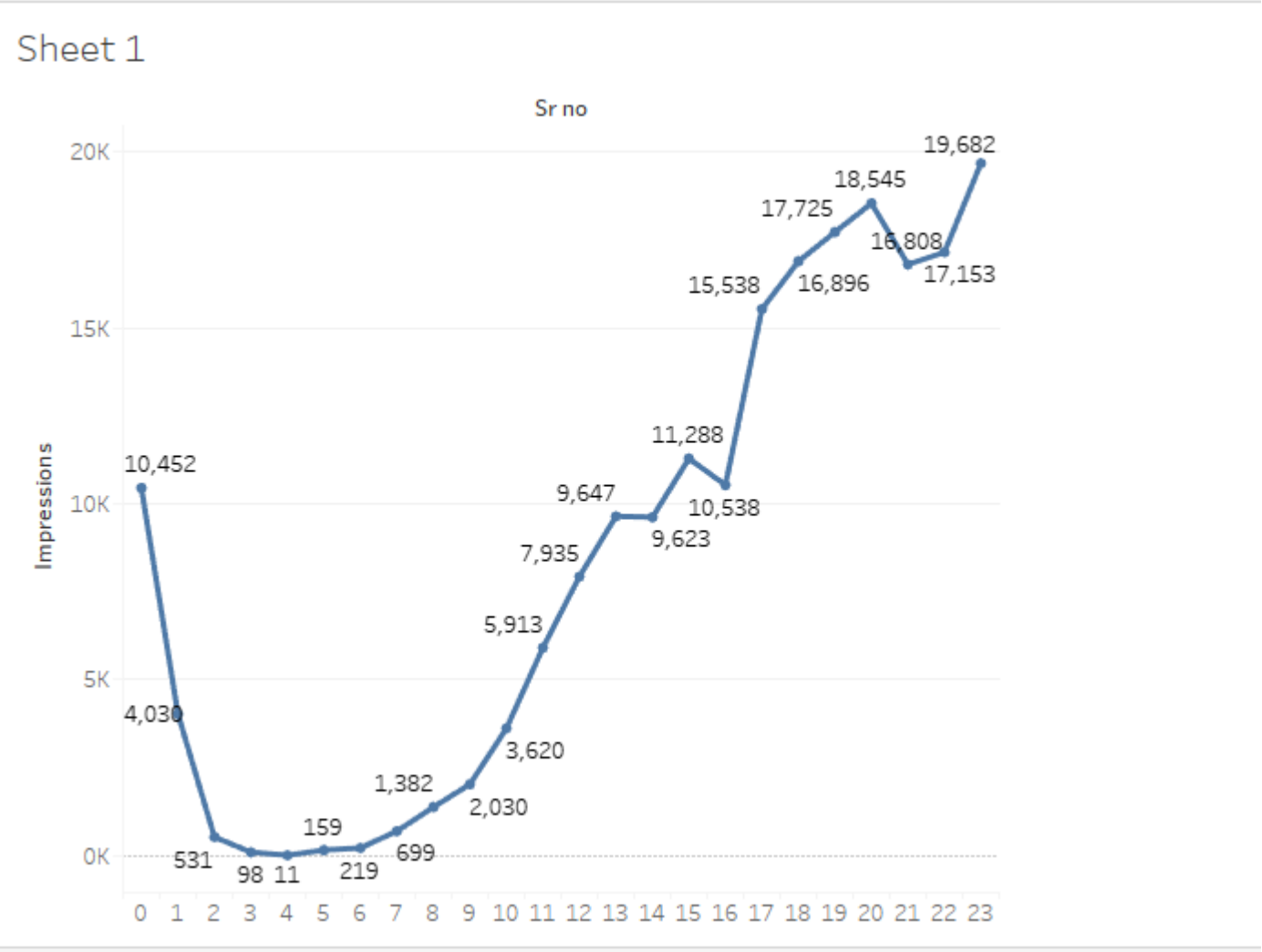
```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[1 0]
 [0 4]]

1.0
```

```
# Accuracy is 100%
```

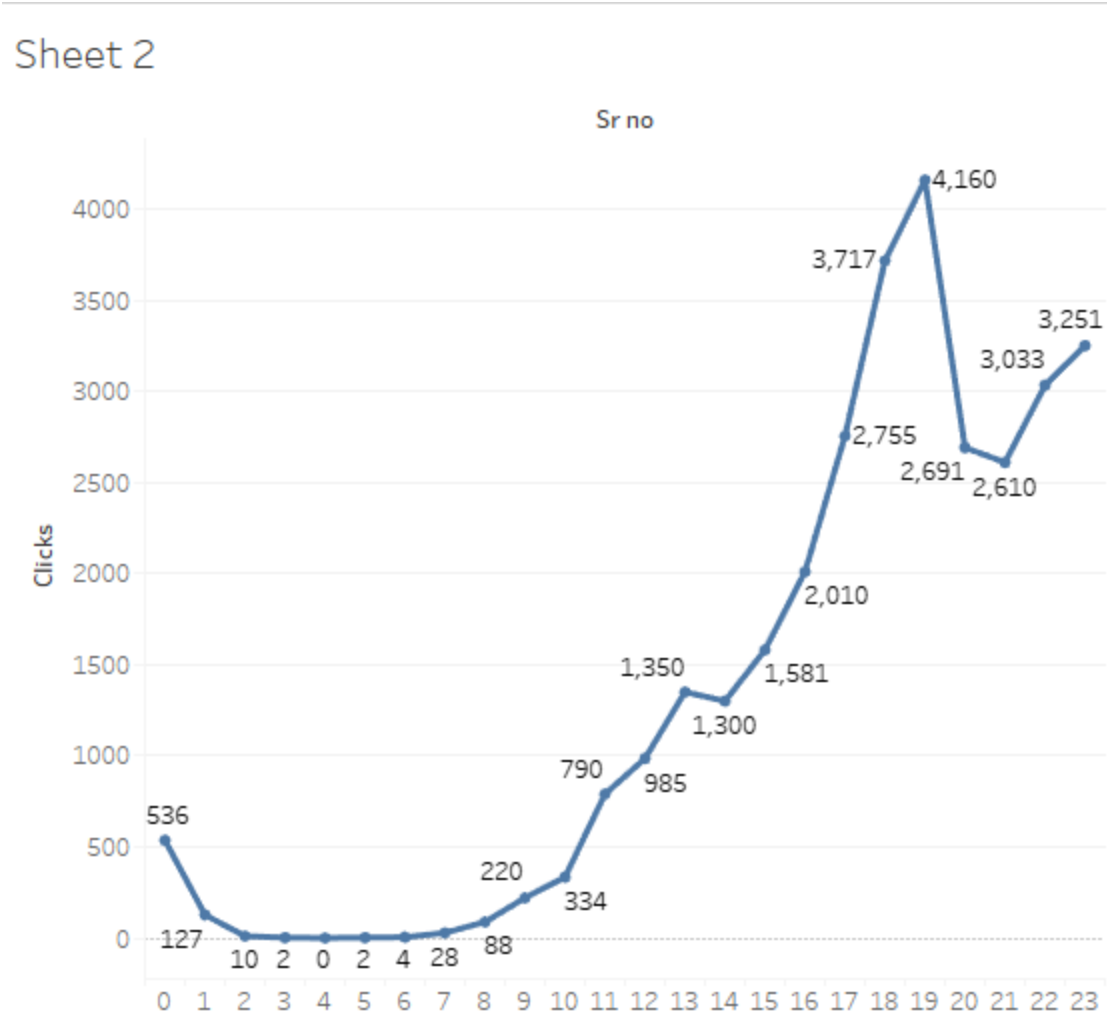
#Visualization



Insights of the Impressions Graph

- Impressions are **high** at the night from **12:00 am to 1:00 am.**
- Then Impressions **are falling down after 1:30 am from 1:30 am to 6:00 am** because majority of the people are sleeping
- From **6:30 am to 11:30 am** impressions are **slightly high and going to increases** because on that time lot of people are engaged in online sites

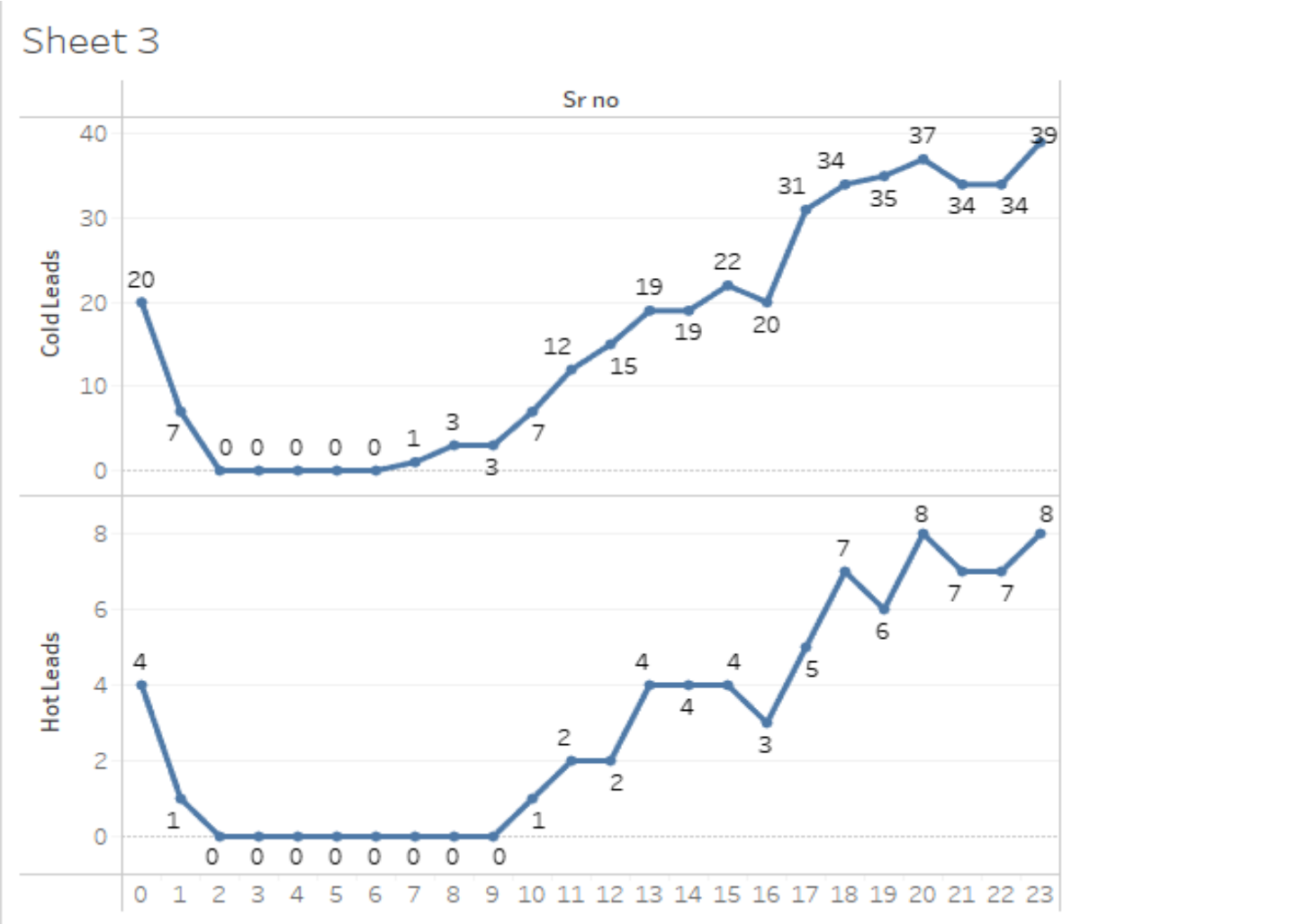
- Then **from 12:00 pm to 11:30 pm** impressions are going to high and from evening 6:30 pm to **11:30 pm** impressions are very high. because more no of people is free.



Insights of the Clicks Graph

- Clicks **are slightly high from 12:00 am to 1:00 am**. because lot of people are searching for new thing
- Then from **1:30 am to 9:30 am** clicks are down
- From **10:00 am to 6:00 pm** clicks are **slightly high** than previous time.

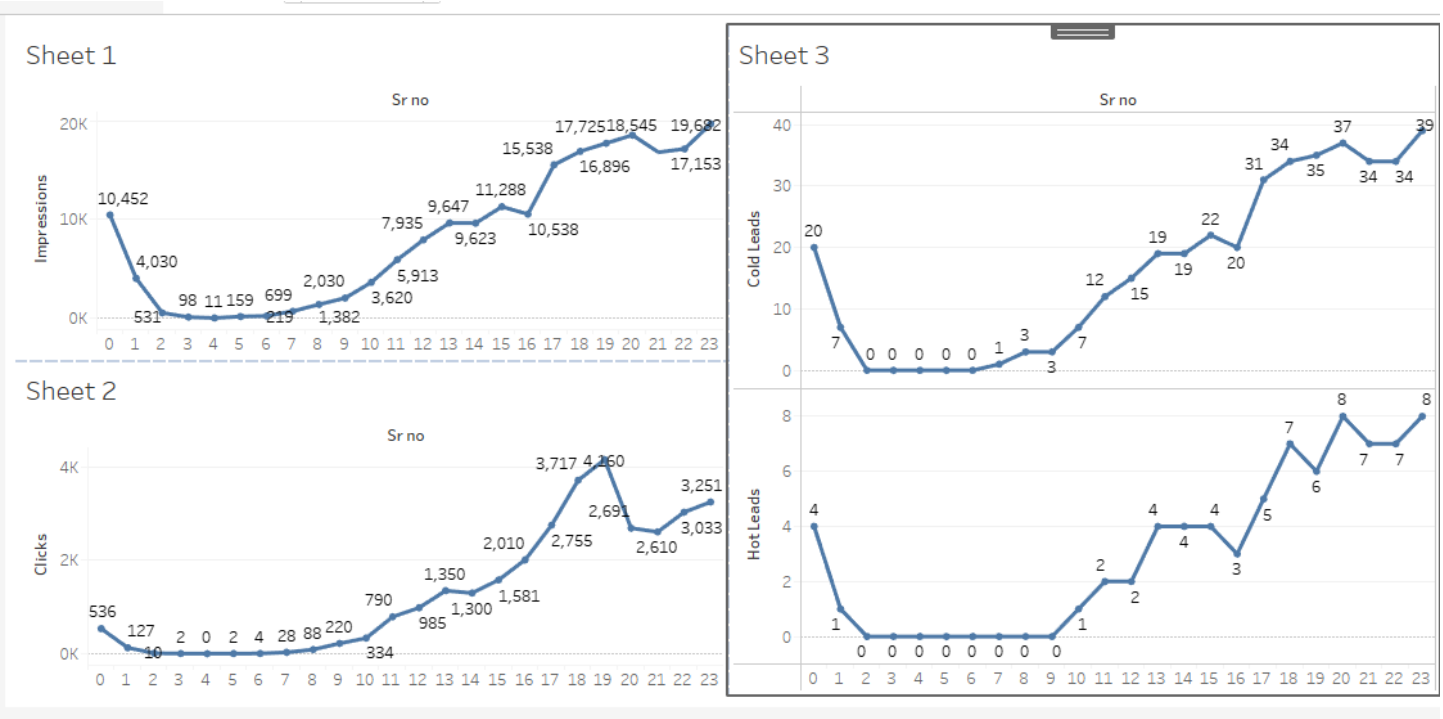
- Then clicks are **going to increases** from **6:00 pm to 11:30 pm.**



Insights of Cold Lead and Hot Lead Graph

- Very few people are showing interest in that particular Ad **from 12:00 am to 1:00 am**
- Then after 1:00 am **no one candidate** is showing interest in that Ad from **1:30 am to 7:00 am** because lot of people are sleeping
- After 7:30 am only 1, 2 candidates are showings interest in particular time slots from 7:30 am **to 10:00 am**

- Then **from 10:30 am to 6:30** pm **high than previous time slots** of leads
- **From 12:00 am to 1:00 am** few people are showing interest and they like to purchase that service (they like to enroll in that Ad)
- Then from **1:30 am to 10:00 am no one** showing interest in that Ad
- Then from **10:30 am to 5:00 pm** few people like 1,2 people are showing interest and they like to purchase that service (they like to enroll in that Ad)
- From **5:30 pm to 11:30** pm 3,4 people are showing interest and they like to purchase that service



#Conclusion

```
#Conclusion
#There are 30 no of preferred time slot of 30 min in a entire day where it is preferable to show the Advertisement
#In which 5:30 pm to 11:30 pm from evening to night company has generated more no of hot leads so we concluded that
#We can show ads to people in above given time slots as more no of people use their mobile phones in evening and night
#so this time is preferrable.
#The maximum cost we used to show ad is in time slot of 7:30 pm to 9:00 pm where we generate 4 no of hot leads
#In each no of time slot
# In the time slots company has get more no of impressions, clicks,cold lead,warm lead,hot lead.
```

```
#There are 18 no of non preferred time slots and that company dosent get hot lead
```

```
#In time slots where company dosent get hot lead company spent 756 rs
#In reamaning time slots where company got hot leads company spent 19301 rs
#So total cost spent by  company on every time slot is 20057 rs
#96.24% of cost spent by company make profit by getting HOT Leads.
```

```
#We suggest we can reduce the cost by 3.76% not spending in the 1:00 am to 10:00 am time slots
```