Google Ads Hourly Analysis

Date : 7 June 2023

Project Start Date - End Date	● Start Date – 07 -06 -2023 ● End Date – 07 -06 2023
Objectives	● To analyses how many people who clicked on the advertisement enrolled 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

Client Project Champion
Name : Arti Sukhadev Patil
Mobile: 8623845944
Email: parti58@gmail.com

Project Abstract

The dataset is about showing advertisement to students for course enrollment. Our main objective was to understand on which time students are clicked on our ads and enrolled our course. Problem statement is classify as we are looking for preferred timing in a day where we can do marketing and we will get definitely sales. 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/Marketing Data Google Ads 6th june.xlsx")
```

```
file.head()
```

	Sr no	Impressions	Clicks	Sales Unit	Time Preferred
0	00:00:00	258647	7759.41	54.31587	1
1	00:30:00	219974	8798.96	61.59272	1
2	01:00:00	1096	10.96	0.07672	0
3	01:30:00	1481	14.81	0.10367	0
4	02:00:00	1794	17.94	0.12558	0

Preprocessing the dataset

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

```
dataset.head()
```

	Impressions	Clicks	Sales Unit	Time Preferred
0	258647	7759.41	54.31587	1
1	219974	8798.96	61.59272	1
2	1096	10.96	0.07672	0
3	1481	14.81	0.10367	0
4	1794	17.94	0.12558	0

```
# descriptive analysis
```

```
dataset.sum()
```

```
Impressions       2.207097e+06
Clicks            8.431078e+04
Sales Unit        5.901755e+02
Time Preferred    3.300000e+01
dtype: float64
```

```
#Impression are no visible Ads to customers
#Total Impressions are 2.207097e+06
#Clicks indicates total no of customers who clicked on our Ads
#Total no of Clicks Ad 8.431078e+04
```

```
#sales unit indicates total no of purchased made in particular 30 mi of slot
#total no of sales unit for entire day 5.901755e+02
```

```
#time preferred is the coloumn which tells us preferred time slot for showing advertisement to the customers
```

```
dataset.mean()
```

```
Impressions       45981.187500
Clicks             1756.474554
Sales Unit           12.295322
Time Preferred        0.687500
dtype: float64
```

```
#the average no of Impression are 45981.187500
#the average no of Clicks are 1756.474554
#the average no of Sales unit are 12.295322
```

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

```
Impressions       0
Clicks            0
Sales Unit        0
Time Preferred    0
dtype: int64
```

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

```
x
```

```
array([[2.58647000e+05, 7.75941000e+03, 5.43158700e+01],
       [2.19974000e+05, 8.79896000e+03, 6.15927200e+01],
       [1.09600000e+03, 1.09600000e+01, 7.67200000e-02],
       [1.48100000e+03, 1.48100000e+01, 1.03670000e-01],
       [1.79400000e+03, 1.79400000e+01, 1.25580000e-01],
       [2.15600000e+03, 2.15600000e+01, 1.50920000e-01],
       [4.13000000e+02, 4.13000000e+00, 2.89100000e-02],
       [1.44000000e+02, 1.44000000e+00, 1.00800000e-02],
       [1.39000000e+02, 1.39000000e+00, 9.73000000e-03],
       [1.91000000e+02, 1.91000000e+00, 1.33700000e-02],
       [3.96000000e+02, 3.96000000e+00, 2.77200000e-02],
       [2.74000000e+02, 2.74000000e+00, 1.91800000e-02],
       [3.15200000e+03, 3.15200000e+01, 2.20640000e-01],
       [3.08900000e+03, 3.08900000e+01, 2.16230000e-01],
       [2.13500000e+03, 2.13500000e+01, 1.49450000e-01],
       [1.84980000e+04, 3.69960000e+02, 2.58972000e+00],
       [1.78430000e+04, 5.35290000e+02, 3.74703000e+00],
       [1.27410000e+04, 4.45935000e+02, 3.12154500e+00],
       [1.89730000e+04, 6.83028000e+02, 4.78119600e+00],
       [1.17460000e+04, 4.69840000e+02, 3.28888000e+00],
       [1.09670000e+04, 2.19340000e+02, 1.53538000e+00],
       [8.74200000e+03, 4.19616000e+02, 2.93731200e+00],
       [4.05900000e+03, 4.87080000e+01, 3.40956000e-01],
       [2.05650000e+04, 4.11300000e+02, 2.87910000e+00],
       [2.44860000e+04, 7.10094000e+02, 4.97065800e+00],
       [2.04910000e+04, 6.76203000e+02, 4.73342100e+00],
       [1.29161000e+05, 1.23994560e+03, 8.67961920e+00],
       [1.01236000e+05, 2.02472000e+03, 1.41730400e+01],
       [1.45966000e+05, 2.91932000e+03, 2.04352400e+01],
```

```
y
```

```
array([1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,
       0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1], 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 = 0.05, random_state =0)
```

```
print(x_train)
```

```
[[1.78600000e+03 5.35800000e+01 3.75060000e-01]
 [1.96730000e+04 5.90190000e+02 4.13133000e+00]
 [3.96130000e+04 1.98065000e+03 1.38645500e+01]
 [1.74910000e+04 5.77203000e+02 4.04042100e+00]
 [1.12940000e+04 1.12940000e+02 7.90580000e-01]
 [1.44000000e+02 1.44000000e+00 1.00800000e-02]
 [3.96000000e+02 3.96000000e+00 2.77200000e-02]
 [2.74000000e+02 2.74000000e+00 1.91800000e-02]
 [1.47410000e+04 4.42230000e+02 3.09561000e+00]
 [1.24360000e+04 4.22824000e+02 2.95976800e+00]
 [1.01236000e+05 2.02472000e+03 1.41730400e+01]
 [2.94484000e+05 2.29697520e+04 1.60788264e+02]
 [1.09600000e+03 1.09600000e+01 7.67200000e-02]
 [2.14526000e+05 1.28715600e+04 9.01009200e+01]
 [1.89730000e+04 6.83028000e+02 4.78119600e+00]
 [1.84980000e+04 3.69960000e+02 2.58972000e+00]
 [1.45966000e+05 2.91932000e+03 2.04352400e+01]
 [4.05900000e+03 4.87080000e+01 3.40956000e-01]
 [1.78430000e+04 5.35290000e+02 3.74703000e+00]
 [1.59770000e+04 1.59770000e+02 1.11839000e+00]
 [1.09670000e+04 2.19340000e+02 1.53538000e+00]
 [1.96640000e+04 1.96640000e+02 1.37648000e+00]
 [1.39000000e+02 1.39000000e+00 9.73000000e-03]
 [3.08900000e+03 3.08900000e+01 2.16230000e-01]
 [2.04910000e+04 6.76203000e+02 4.73342100e+00]
 [2.15600000e+03 2.15600000e+01 1.50920000e-01]
 [1.27410000e+04 4.45935000e+02 3.12154500e+00]
 [1.19640000e+04 3.70884000e+02 2.59618800e+00]
 [2.13500000e+03 2.13500000e+01 1.49450000e-01]
```

```
print(y_train)
```

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

```
print(x_test)
```

```
[[1.7860000e+04 5.3580000e+02 3.7506000e+00]
 [1.7940000e+03 1.7940000e+01 1.2558000e-01]
 [1.2916100e+05 1.2399456e+03 8.6796192e+00]]
```

```
print(y_test)
```

```
[1 0 1]
```

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

```
print(x_train)
```

```
[[-0.56125976 -0.42277215 -0.42277215]
 [-0.33285405 -0.29533013 -0.29533013]
 [-0.07823282  0.03489671  0.03489671]
 [-0.36071681 -0.29841448 -0.29841448]
 [-0.4398486  -0.40867447 -0.40867447]
 [-0.58222707 -0.43515512 -0.43515512]
 [-0.57900918 -0.43455663 -0.43455663]
 [-0.58056705 -0.43484638 -0.43484638]
 [-0.39583258 -0.33046984 -0.33046984]
 [-0.42526598 -0.33507867 -0.33507867]
 [ 0.70865404  0.0453631   0.0453631 ]
 [ 3.17630916  5.01969665  5.01969665]
 [-0.57007063 -0.43289417 -0.43289417]
 [ 2.15529591  2.62142983  2.62142983]
 [-0.34179261 -0.2732816  -0.2732816 ]
 [-0.34785806 -0.34763358 -0.34763358]
 [ 1.27982794  0.25782584  0.25782584]
 [-0.53223498 -0.42392922 -0.42392922]
 [-0.35622199 -0.30836859 -0.30836859]
 [-0.38004964 -0.39755259 -0.39755259]
 [-0.44402418 -0.38340503 -0.38340503]
 [-0.33296897 -0.38879616 -0.38879616]
 [-0.58229091 -0.435167   -0.435167  ]
 [-0.54462127 -0.4281609  -0.4281609 ]
 [-0.3224087  -0.2749025  -0.2749025 ]
 [-0.55653509 -0.43037673 -0.43037673]
 [-0.42137132 -0.32958993 -0.32958993]
 [-0.43129312 -0.34741414 -0.34741414]
 [-0.55680325 -0.4304266  -0.4304266 ]
```

```
print(x_test)
```

```
[[-0.35600491 -0.30824747 -0.30824747]
 [-0.56115761 -0.43123646 -0.43123646]
 [ 1.06523868 -0.14101664 -0.14101664]]
```

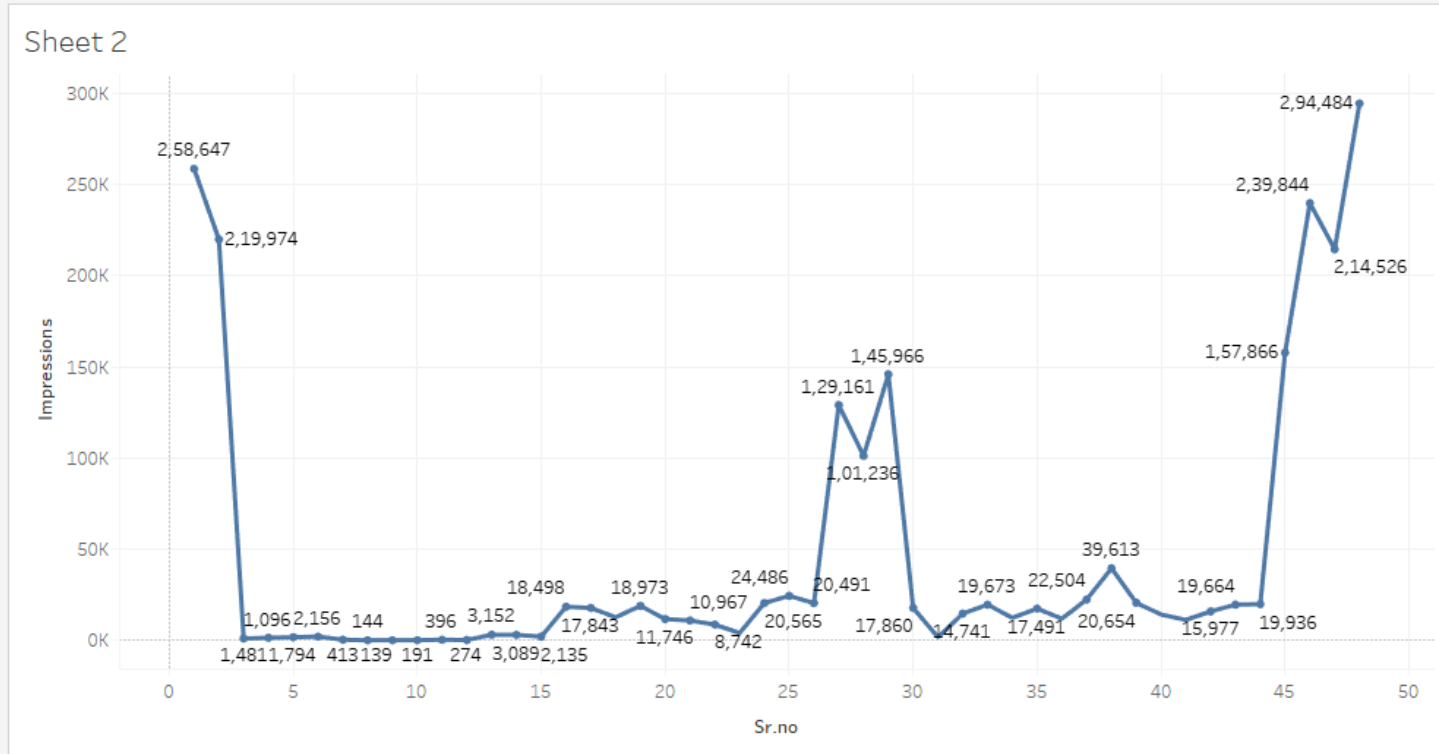
```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'gini', random_state = 0)
classifier.fit(x_train, y_train)
```

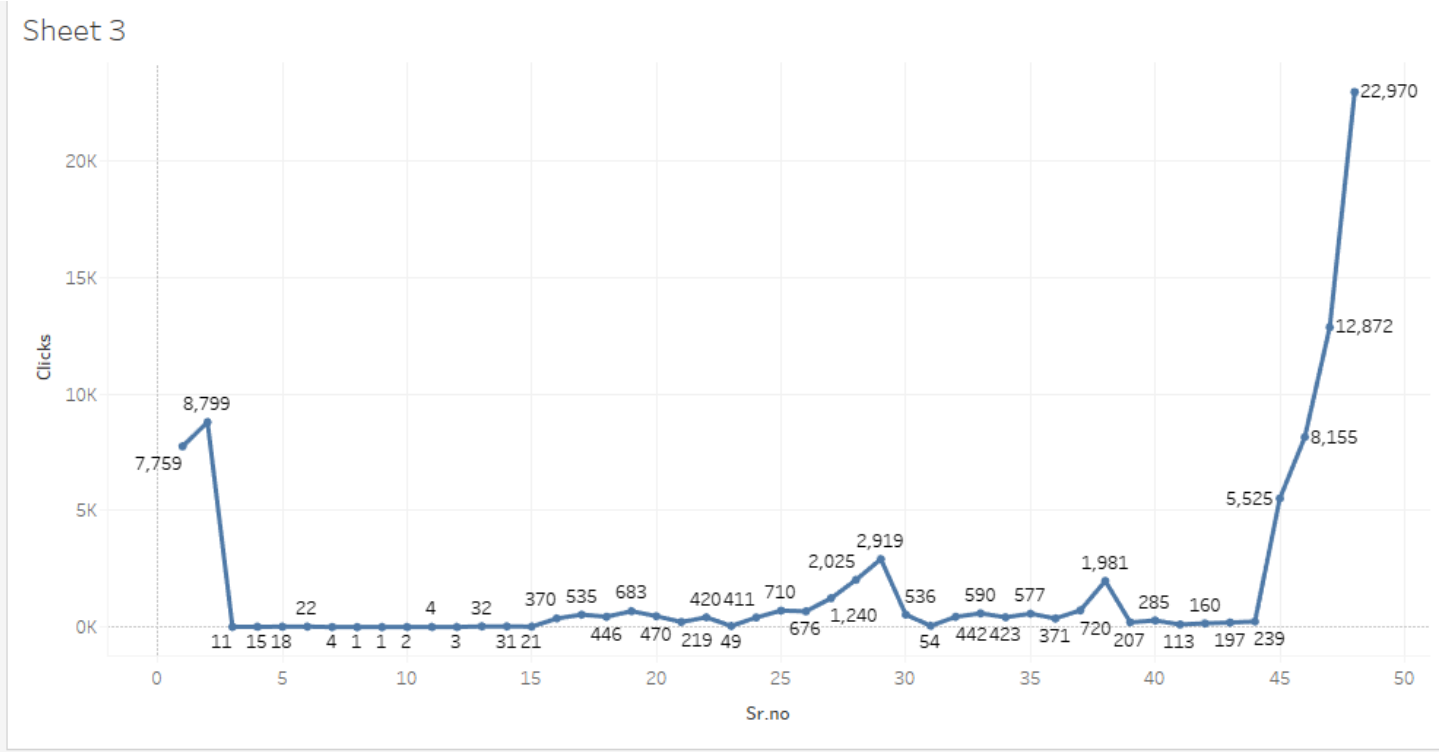
```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)
```

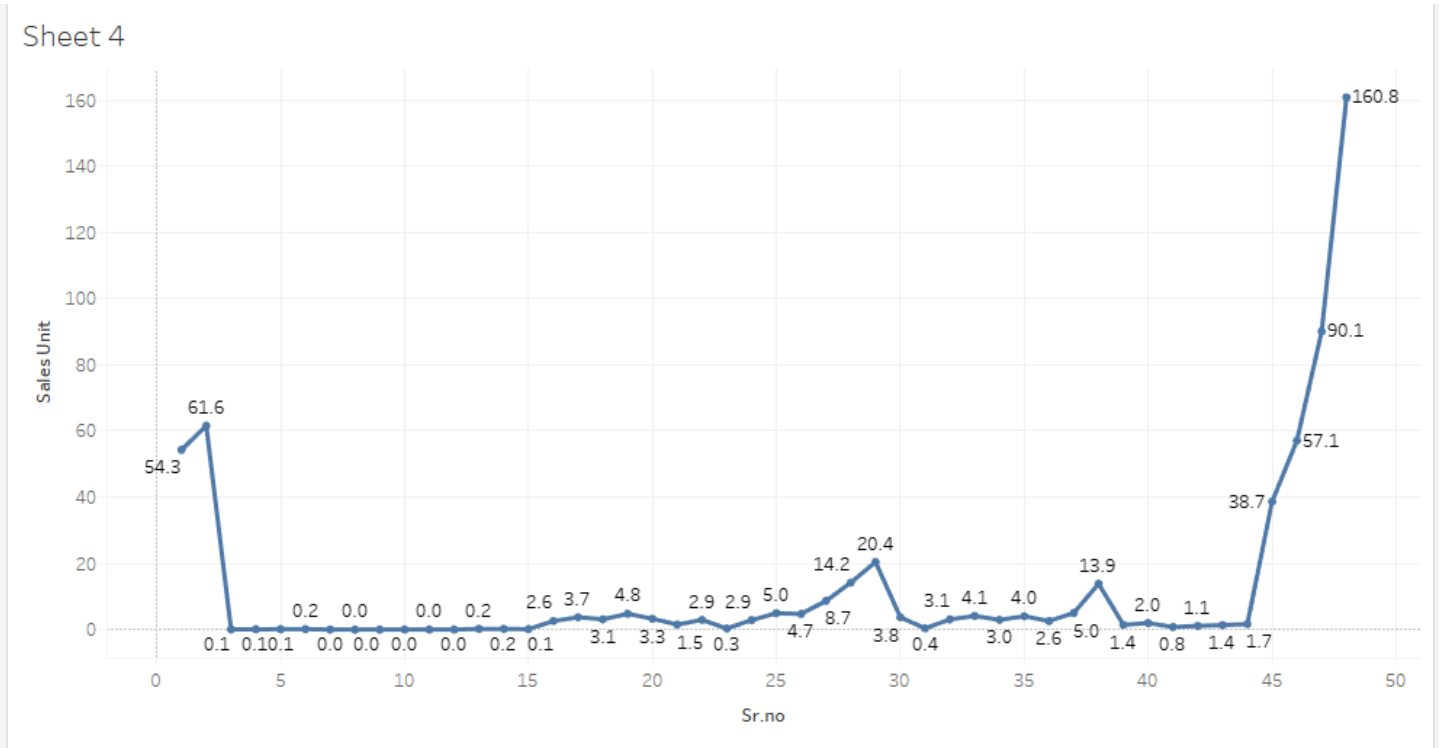
```
print(classifier.predict(sc.transform([[20491,676.2030,4.733421]])))
```

```
[1]
```

Data Visualisation




Sheet 3
Clicks
Sr.no
20K
15K
10K
5K
0K
0
5
10
15
20
25
30
35
40
45
50
7,759
8,799
11
15
18
22
4
1
1
2
3
32
31
21
370
535
446
683
470
420
219
411
49
710
676
2,025
1,240
2,919
536
54
590
442
423
577
371
1,981
720
207
285
113
160
197
5,525
239
8,155
12,872
22,970


Sheet 4
Sales Unit
Sr.no
160
140
120
100
80
60
40
20
0
0
5
10
15
20
25
30
35
40
45
50
54.3
61.6
0.1
0.1
0.1
0.2
0.0
0.0
0.0
0.0
0.0
0.2
0.2
0.1
2.6
3.7
3.1
4.8
3.3
2.9
1.5
2.9
0.3
5.0
4.7
14.2
8.7
20.4
3.8
3.1
0.4
4.1
3.0
4.0
2.6
13.9
5.0
2.0
1.4
1.1
0.8
1.4
38.7
1.7
57.1
90.1
160.8

Conclusion

```
#Conclusion
#There are 33 no of preferred time slot of 30 min in a entire day where it is preferable to show the Advertisement
```

```
#In late night 12 am to 12:30 am,
#In morning 7:30 am to 10:30 am,
#In late morning 11:30 am to 12:00 pm,
#In Early afternoon 12:00 pm to 2:30 pm,
#In Mid afternoon 3:30 pm to 4 pm,
#In Late afternoon 4 pm to 6 pm,
#In Evening 6 pm to 9 pm,
#In Late evening 9 pm to 11:30 pm
#These are the timing where copmany has received the sales
```

```
#Using above algorithm we have classify that we have successfully train the data and classified into 2 categerious
#this timing is preferred and not preferred
```

```
#In this way we have used decision tree algorithm to classify time slots which are preferable for conversion campaign
#This will help makketing to take decision about remarketing campaign
```