Google Ads Hourly Analysis

Date : 04 – 06 – 2023

Project Start Date - End Date	• Start date – 04 -06 -2023 • End date – 04 – 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 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 : Sisddhivinayak Phulwadkar
Mobile: 9028965955
Email: siddhivinayakphulwadkar@gmail.com

Project Trainee
Name : Arti Sukhadev Patil
Mobile: 8623845944
Email: parti5863@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 4th june.xlsx")
```

```
file.head()
```

	Sr no	Impressions	Clicks	Sales Unit	Time preferred
0	00:00:00	245847	245.847	2.45847	1
1	00:30:00	159674	159.674	1.59674	1
2	01:00:00	1896	1.896	0.01896	0
3	01:30:00	1789	1.789	0.01789	0
4	02:00:00	1990	1.990	0.01990	0

Preprocessing the dataset

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

```
dataset.head()
```

	Impressions	Clicks	Sales Unit	Time preferred
0	245847	245.847	2.45847	1
1	159674	159.674	1.59674	1
2	1896	1.896	0.01896	0
3	1789	1.789	0.01789	0
4	1990	1.990	0.01990	0

```
# descriptive analysis
```

```
dataset.sum()
```

```
Impressions       2.026639e+06
Clicks            2.026639e+03
Sales Unit        2.026639e+01
Time preferred    9.000000e+00
dtype: float64
```

```
#Impressions are no of visible ads to customers
#Total impressions are  2.026639e+06
#Clicks indicates total no of customers who clicked on our ads
#total no of clicks ad 2.026639e+03
```

```
#sales unit indicates total no of purchased made in particular 30 mi of slot
#total no of sales unit for entire day 2.026639e+01
```

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

```
dataset.mean()
```

```
Impressions       42221.645833
Clicks               42.221646
Sales Unit            0.422216
Time preferred        0.187500
dtype: float64
```

```
#the average no of Impression are 42221.645833
#the average no of Clicks are 42.221646
#the average no of Sales unit are 0.422216
```

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

```
x
```

```
array([[2.45847e+05, 2.45847e+02, 2.45847e+00],
       [1.59674e+05, 1.59674e+02, 1.59674e+00],
       [1.89600e+03, 1.89600e+00, 1.89600e-02],
       [1.78900e+03, 1.78900e+00, 1.78900e-02],
       [1.99000e+03, 1.99000e+00, 1.99000e-02],
       [1.29900e+03, 1.29900e+00, 1.29900e-02],
       [1.80000e+02, 1.80000e-01, 1.80000e-03],
       [1.89000e+02, 1.89000e-01, 1.89000e-03],
       [1.75000e+02, 1.75000e-01, 1.75000e-03],
       [1.83000e+02, 1.83000e-01, 1.83000e-03],
       [5.96000e+02, 5.96000e-01, 5.96000e-03],
       [5.74000e+02, 5.74000e-01, 5.74000e-03],
       [1.75200e+03, 1.75200e+00, 1.75200e-02],
       [1.78900e+03, 1.78900e+00, 1.78900e-02],
       [1.23500e+03, 1.23500e+00, 1.23500e-02],
       [1.54780e+04, 1.54780e+01, 1.54780e-01],
       [1.59630e+04, 1.59630e+01, 1.59630e-01],
       [1.22580e+04, 1.22580e+01, 1.22580e-01],
       [1.89640e+04, 1.89640e+01, 1.89640e-01],
       [1.78460e+04, 1.78460e+01, 1.78460e-01],
       [1.02530e+04, 1.02530e+01, 1.02530e-01],
       [9.45200e+03, 9.45200e+00, 9.45200e-02],
       [4.85800e+03, 4.85800e+00, 4.85800e-02],
       [1.96450e+04, 1.96450e+01, 1.96450e-01],
       [1.64860e+04, 1.64860e+01, 1.64860e-01],
       [1.84520e+04, 1.84520e+01, 1.84520e-01],
       [1.20456e+05, 1.20456e+02, 1.20456e+00],
       [1.02396e+05, 1.02396e+02, 1.02396e+00],
```

```
y
```

```
array([1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       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 = 0.10, random_state =0)
```

```
print(x_train)
```

```
[[3.21460e+04 3.21460e+01 3.21460e-01]
 [1.33310e+04 1.33310e+01 1.33310e-01]
 [1.06940e+04 1.06940e+01 1.06940e-01]
 [1.89000e+02 1.89000e-01 1.89000e-03]
 [5.96000e+02 5.96000e-01 5.96000e-03]
 [5.74000e+02 5.74000e-01 5.74000e-03]
 [1.27840e+04 1.27840e+01 1.27840e-01]
 [1.23630e+04 1.23630e+01 1.23630e-01]
 [1.02396e+05 1.02396e+02 1.02396e+00]
 [2.54964e+05 2.54964e+02 2.54964e+00]
 [1.89600e+03 1.89600e+00 1.89600e-02]
 [2.10006e+05 2.10006e+02 2.10006e+00]
 [1.89640e+04 1.89640e+01 1.89640e-01]
 [1.54780e+04 1.54780e+01 1.54780e-01]
 [1.58476e+05 1.58476e+02 1.58476e+00]
 [4.85800e+03 4.85800e+00 4.85800e-02]
 [1.59630e+04 1.59630e+01 1.59630e-01]
 [1.47860e+04 1.47860e+01 1.47860e-01]
 [1.02530e+04 1.02530e+01 1.02530e-01]
 [1.59660e+04 1.59660e+01 1.59660e-01]
 [1.75000e+02 1.75000e-01 1.75000e-03]
 [1.78900e+03 1.78900e+00 1.78900e-02]
 [1.84520e+04 1.84520e+01 1.84520e-01]
 [1.29900e+03 1.29900e+00 1.29900e-02]
 [1.22580e+04 1.22580e+01 1.22580e-01]
 [1.96340e+04 1.96340e+01 1.96340e-01]
 [1.23500e+03 1.23500e+00 1.23500e-02]
 [1.75460e+04 1.75460e+01 1.75460e-01]
 [1.59674e+05 1.59674e+02 1.59674e+00]
```

```
: print(y_train)
```

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

```
print(x_test)
```

```
[[1.42000e+04 1.42000e+01 1.42000e-01]
 [1.99000e+03 1.99000e+00 1.99000e-02]
 [1.20456e+05 1.20456e+02 1.20456e+00]
 [1.52070e+04 1.52070e+01 1.52070e-01]
 [1.63470e+04 1.63470e+01 1.63470e-01]]
```

```
print(y_test)
```

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

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

```
print(x_train)
```

```
[[-0.15421626 -0.15421626 -0.15421626]
 [-0.41624505 -0.41624505 -0.41624505]
 [-0.45296947 -0.45296947 -0.45296947]
 [-0.59926829 -0.59926829 -0.59926829]
 [-0.59360017 -0.59360017 -0.59360017]
 [-0.59390656 -0.59390656 -0.59390656]
 [-0.42386289 -0.42386289 -0.42386289]
 [-0.42972599 -0.42972599 -0.42972599]
 [ 0.82412669  0.82412669  0.82412669]
 [ 2.94887868  2.94887868  2.94887868]
 [-0.5754956  -0.5754956  -0.5754956 ]
 [ 2.32276705  2.32276705  2.32276705]
 [-0.33779657 -0.33779657 -0.33779657]
 [-0.38634466 -0.38634466 -0.38634466]
 [ 1.60512986  1.60512986  1.60512986]
 [-0.53424505 -0.53424505 -0.53424505]
 [-0.37959026 -0.37959026 -0.37959026]
 [-0.39598186 -0.39598186 -0.39598186]
 [-0.45911109 -0.45911109 -0.45911109]
 [-0.37954848 -0.37954848 -0.37954848]
 [-0.59946327 -0.59946327 -0.59946327]
 [-0.57698575 -0.57698575 -0.57698575]
 [-0.34492698 -0.34492698 -0.34492698]
 [-0.58380978 -0.58380978 -0.58380978]
 [-0.43118828 -0.43118828 -0.43118828]
 [-0.32846575 -0.32846575 -0.32846575]
 [-0.58470108 -0.58470108 -0.58470108]
 [-0.35754447 -0.35754447 -0.35754447]
 [ 1.62181392  1.62181392  1.62181392]
```

```
print(x_test)
```

```
[[-0.40414284 -0.40414284 -0.40414284]
 [-0.57418651 -0.57418651 -0.57418651]
 [ 1.07564091  1.07564091  1.07564091]
 [-0.39011877 -0.39011877 -0.39011877]
 [-0.37424245 -0.37424245 -0.37424245]]
```

```
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([[183,0.183,0.00183]])))
```

```
[0]
```

Conclusion

```
#Conclusion
#There are 9 no of preferred time slot of 30 min in a entire day where it is prederable to show the Advertisement
```

```
#In night 12am to 2pm and in morning 10:30am to 11:30am these are the timing where the company 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
```