SWACHANDA

Google Ads Hourly Analysis

Date : 20th June 2023

Project Start Date - End Date	● Start Date – 20 -06 -2023 ● End Date – 21–06-2023
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

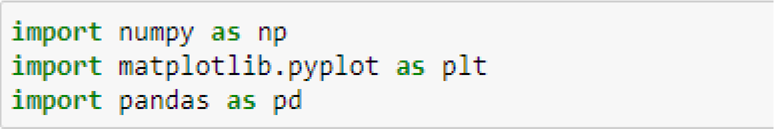
Project Trainee
Name : Arti Sukhadev Patil
Mobile: 8623845944
Email: parti5863@gmail.com

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.

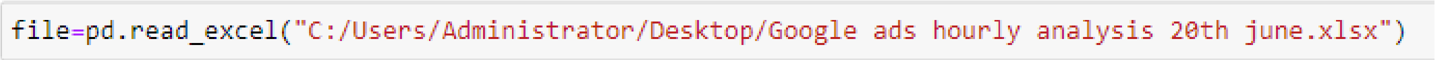
Google Ads Hourly Analysis

#Importing libraries

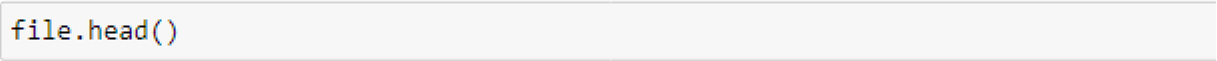


```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

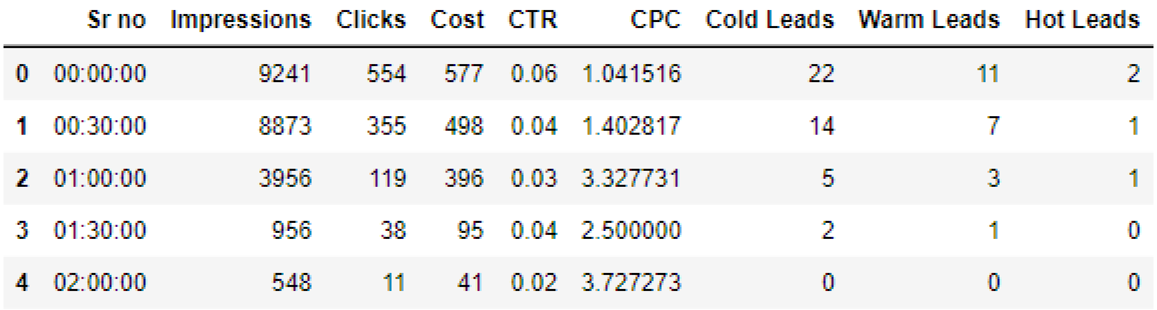
#Importing dataset



```
file=pd.read_excel("C:/Users/Administrator/Desktop/Google ads hourly analysis 20th june.xlsx")
```

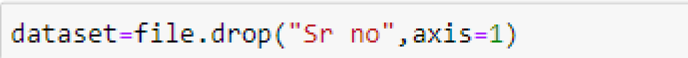


```
file.head()
```

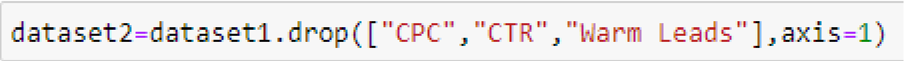


	Sr no	Impressions	Clicks	Cost	CTR	CPC	Cold Leads	Warm Leads	Hot Leads
0	00:00:00	9241	554	577	0.06	1.041516	22	11	2
1	00:30:00	8873	355	498	0.04	1.402817	14	7	1
2	01:00:00	3956	119	396	0.03	3.327731	5	3	1
3	01:30:00	956	38	95	0.04	2.500000	2	1	0
4	02:00:00	548	11	41	0.02	3.727273	0	0	0

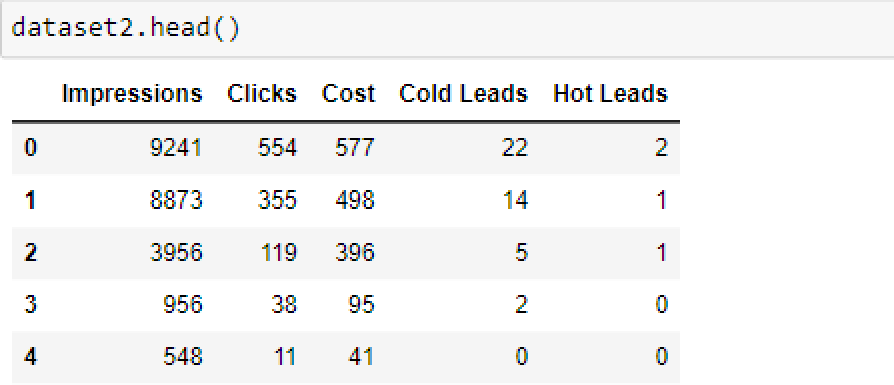
#Preprocessing the dataset



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

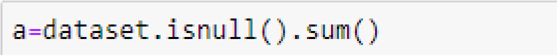


```
dataset2=dataset1.drop(["CPC","CTR","Warm Leads"],axis=1)
```

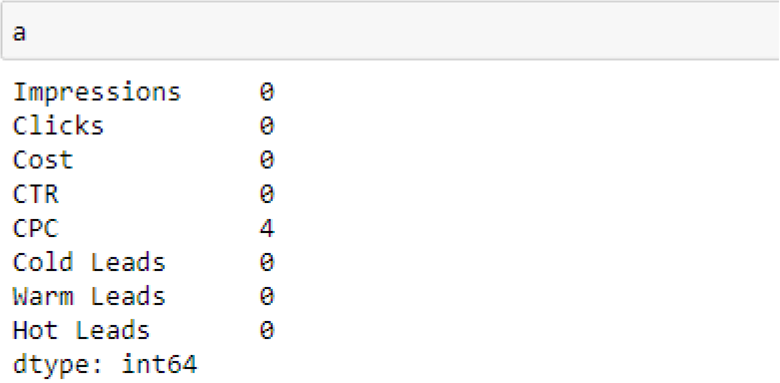


```
dataset2.head()
```

	Impressions	Clicks	Cost	Cold Leads	Hot Leads
0	9241	554	577	22	2
1	8873	355	498	14	1
2	3956	119	396	5	1
3	956	38	95	2	0
4	548	11	41	0	0

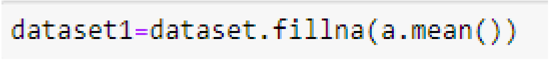


```
a=dataset.isnull().sum()
```

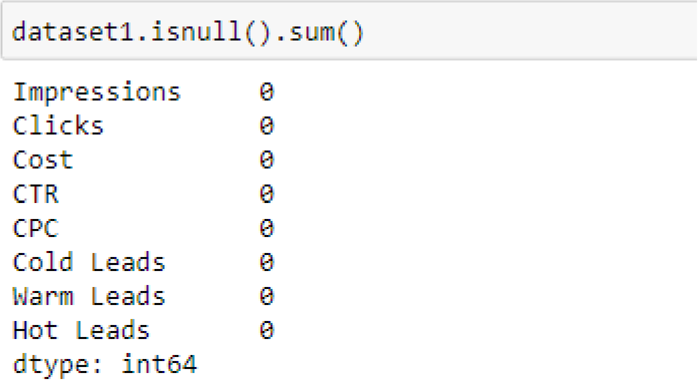


```
a
```

```
Impressions     0
Clicks          0
Cost            0
CTR             0
CPC             4
Cold Leads      0
Warm Leads      0
Hot Leads       0
dtype: int64
```



```
dataset1=dataset.fillna(a.mean())
```

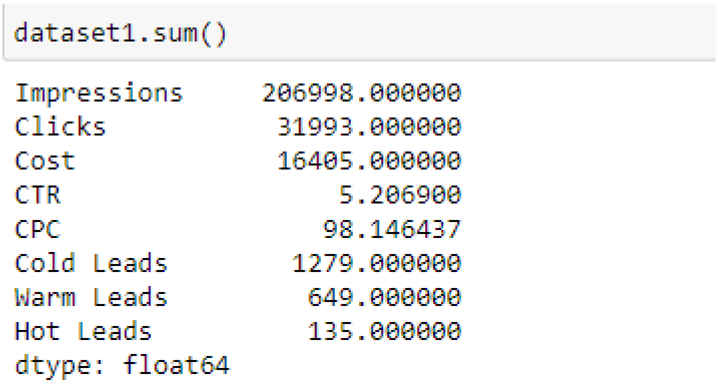


```
dataset1.isnull().sum()
```

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

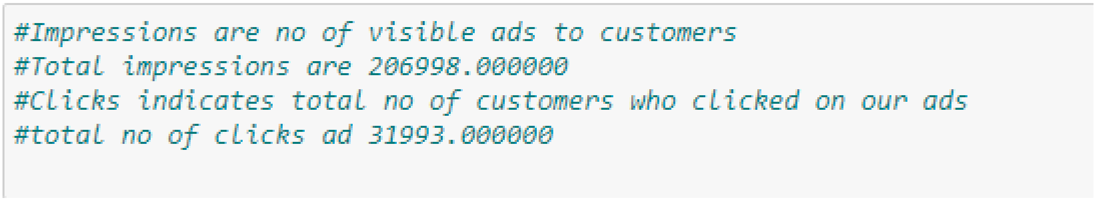
#Descriptive analysis

```
# descriptive analysis
```

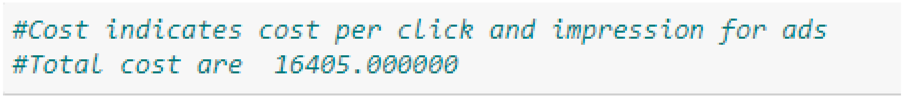


```
dataset1.sum()
```

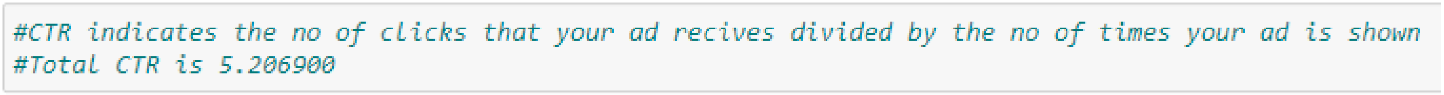
```
Impressions     206998.000000
Clicks           31993.000000
Cost             16405.000000
CTR                  5.206900
CPC                 98.146437
Cold Leads        1279.000000
Warm Leads         649.000000
Hot Leads          135.000000
dtype: float64
```



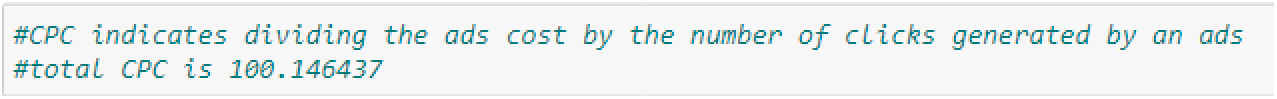
```
#Impressions are no of visible ads to customers
#Total impressions are 206998.000000
#Clicks indicates total no of customers who clicked on our ads
#total no of clicks ad 31993.000000
```



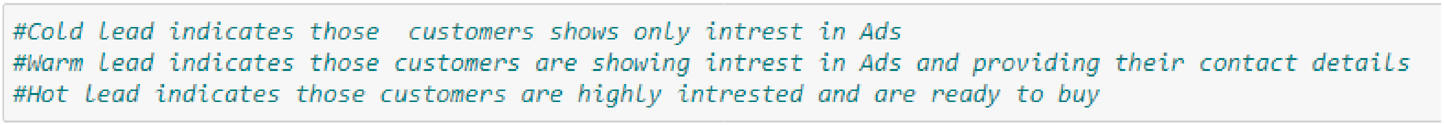
```
#Cost indicates cost per click and impression for ads
#Total cost are  16405.000000
```



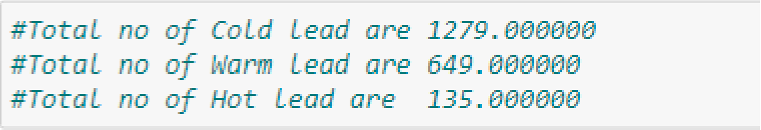
```
#CTR indicates the no of clicks that your ad recives divided by the no of times your ad is shown
#Total CTR is 5.206900
```



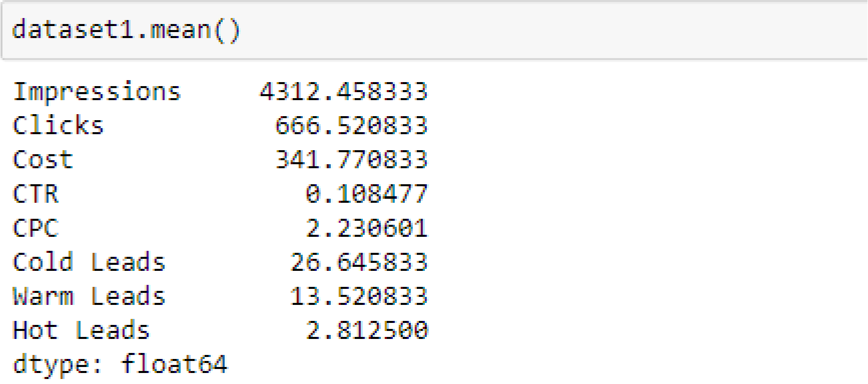
```
#CPC indicates dividing the ads cost by the number of clicks generated by an ads
#total CPC is 100.146437
```



```
#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
```

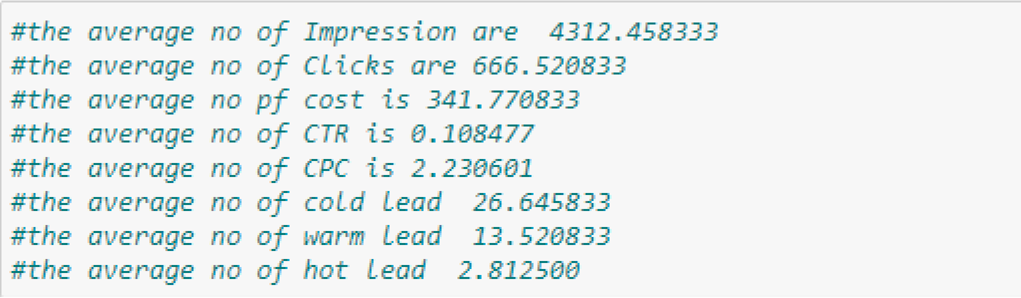


```
#Total no of Cold Lead are 1279.000000
#Total no of Warm Lead are 649.000000
#Total no of Hot Lead are  135.000000
```

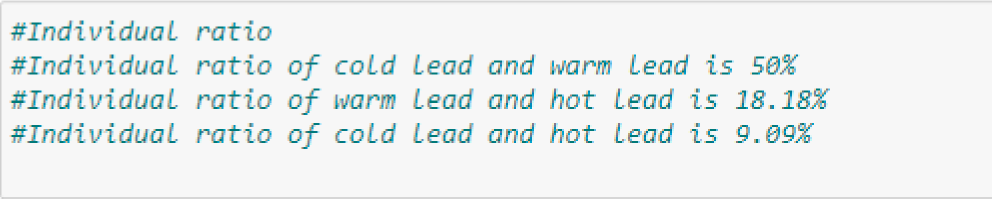


```
dataset1.mean()
```

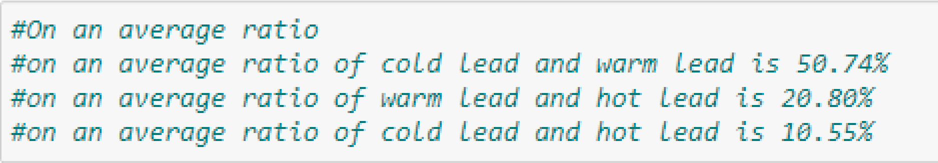
```
Impressions     4312.458333
Clicks           666.520833
Cost             341.770833
CTR                0.108477
CPC                2.230601
Cold Leads        26.645833
Warm Leads        13.520833
Hot Leads          2.812500
dtype: float64
```



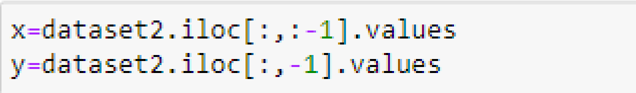
```
#the average no of Impression are  4312.458333
#the average no of Clicks are 666.520833
#the average no pf cost is 341.770833
#the average no of CTR is 0.108477
#the average no of CPC is 2.230601
#the average no of cold lead  26.645833
#the average no of warm lead  13.520833
#the average no of hot lead  2.812500
```



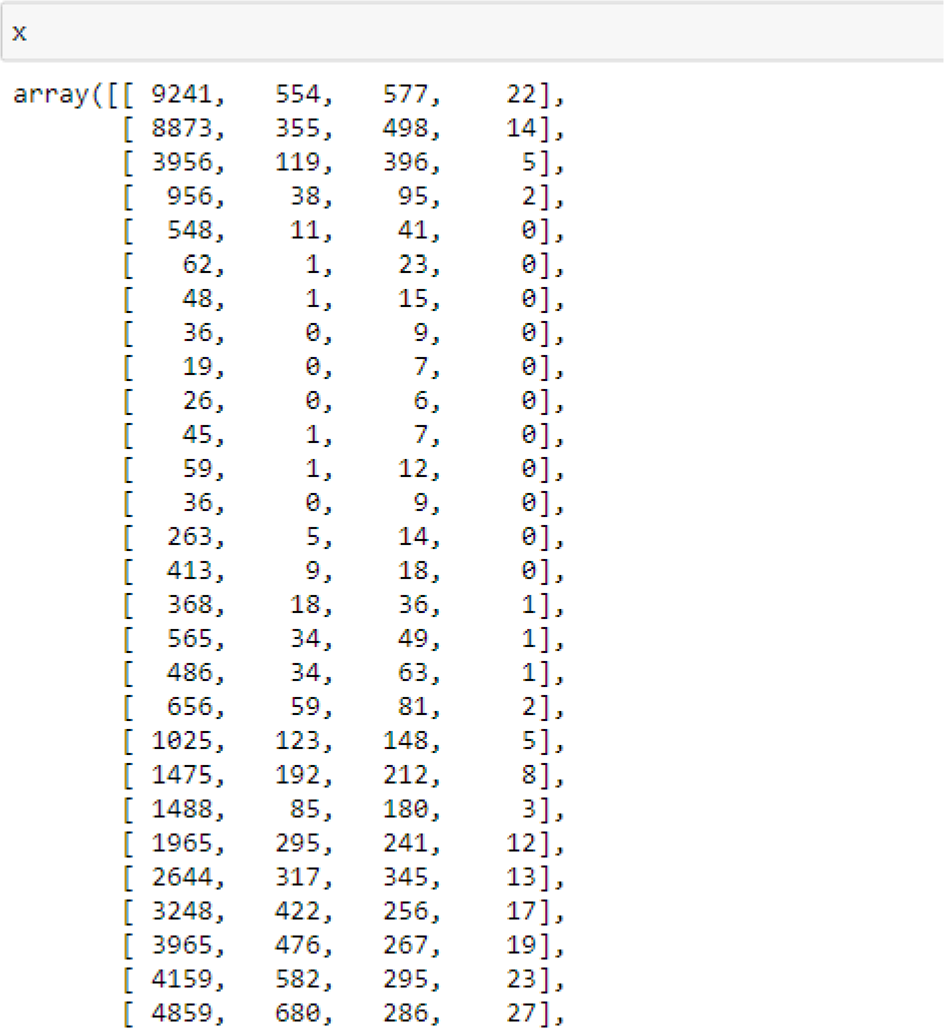
```
#Individual ratio
#Individual ratio of cold lead and warm lead is 50%
#Individual ratio of warm lead and hot lead is 18.18%
#Individual ratio of cold lead and hot lead is 9.09%
```



```
#On an average ratio
#on an average ratio of cold lead and warm lead is 50.74%
#on an average ratio of warm lead and hot lead is 20.80%
#on an average ratio of cold lead and hot lead is 10.55%
```

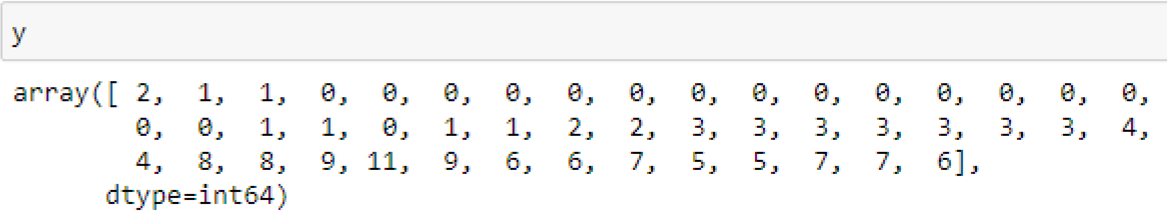


```
x=dataset2.iloc[:,:-1].values
y=dataset2.iloc[:,-1].values
```



```
x
```

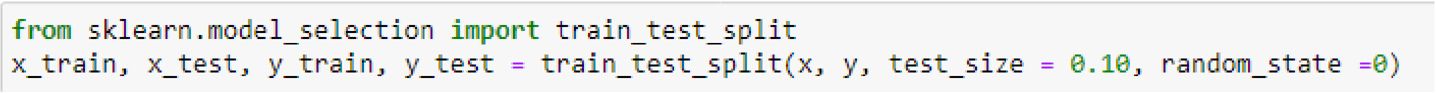
```
array([[ 9241,   554,   577,    22],
       [ 8873,   355,   498,    14],
       [ 3956,   119,   396,     5],
       [  956,    38,    95,     2],
       [  548,    11,    41,     0],
       [   62,     1,    23,     0],
       [   48,     1,    15,     0],
       [   36,     0,     9,     0],
       [   19,     0,     7,     0],
       [   26,     0,     6,     0],
       [   45,     1,     7,     0],
       [   59,     1,    12,     0],
       [   36,     0,     9,     0],
       [  263,     5,    14,     0],
       [  413,     9,    18,     0],
       [  368,    18,    36,     1],
       [  565,    34,    49,     1],
       [  486,    34,    63,     1],
       [  656,    59,    81,     2],
       [ 1025,   123,   148,     5],
       [ 1475,   192,   212,     8],
       [ 1488,    85,   180,     3],
       [ 1965,   295,   241,    12],
       [ 2644,   317,   345,    13],
       [ 3248,   422,   256,    17],
       [ 3965,   476,   267,    19],
       [ 4159,   582,   295,    23],
       [ 4859,   680,   286,    27],
```



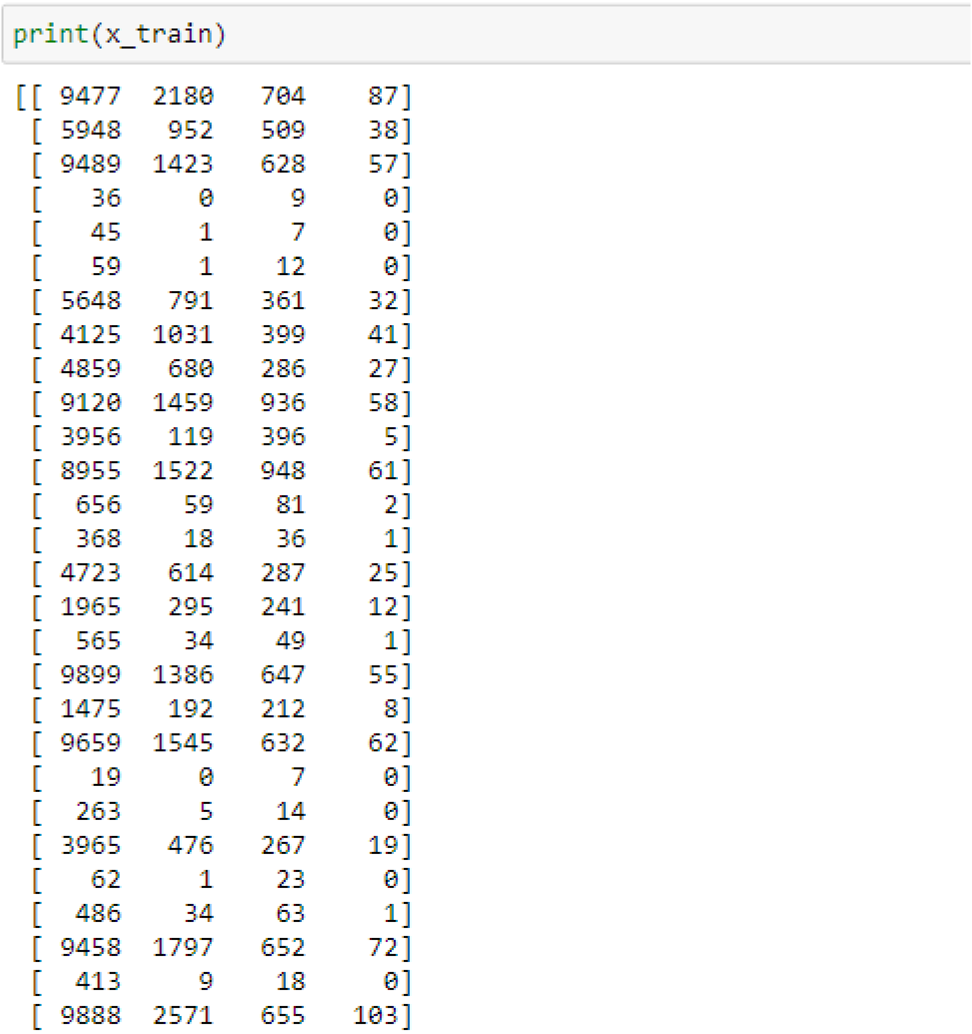
```
y
```

```
array([ 2,  1,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  1,  1,  0,  1,  1,  2,  2,  3,  3,  3,  3,  3,  3,  3,  4,
        4,  8,  8,  9, 11,  9,  6,  6,  7,  5,  5,  7,  7,  6],
      dtype=int64)
```

#Training the model

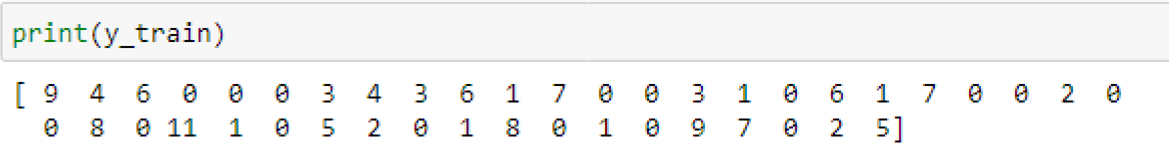


```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.10, random_state =0)
```



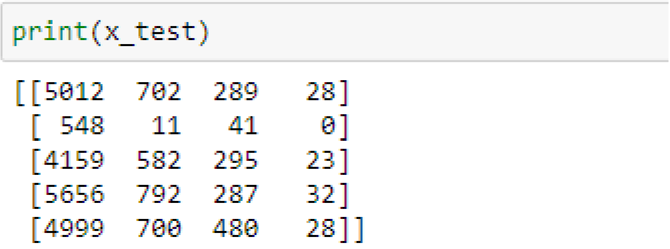
```
print(x_train)
```

```
[[ 9477  2180   704    87]
 [ 5948   952   509    38]
 [ 9489  1423   628    57]
 [   36     0     9     0]
 [   45     1     7     0]
 [   59     1    12     0]
 [ 5648   791   361    32]
 [ 4125  1031   399    41]
 [ 4859   680   286    27]
 [ 9120  1459   936    58]
 [ 3956   119   396     5]
 [ 8955  1522   948    61]
 [  656    59    81     2]
 [  368    18    36     1]
 [ 4723   614   287    25]
 [ 1965   295   241    12]
 [  565    34    49     1]
 [ 9899  1386   647    55]
 [ 1475   192   212     8]
 [ 9659  1545   632    62]
 [   19     0     7     0]
 [  263     5    14     0]
 [ 3965   476   267    19]
 [   62     1    23     0]
 [  486    34    63     1]
 [ 9458  1797   652    72]
 [  413     9    18     0]
 [ 9888  2571   655   103]
```



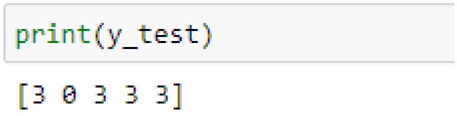
```
print(y_train)
```

```
[ 9  4  6  0  0  0  3  4  3  6  1  7  0  0  3  1  0  6  1  7  0  0  2  0
  0  8  0 11  1  0  5  2  0  1  8  0  1  0  9  7  0  2  5]
```



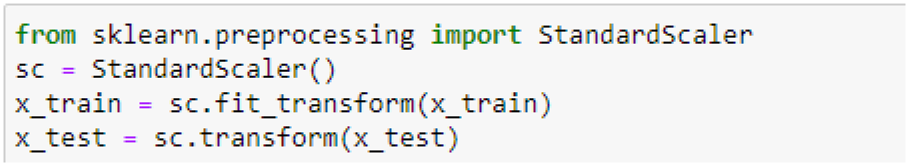
```
print(x_test)
```

```
[[5012  702  289   28]
 [ 548   11   41    0]
 [4159  582  295   23]
 [5656  792  287   32]
 [4999  700  480   28]]
```

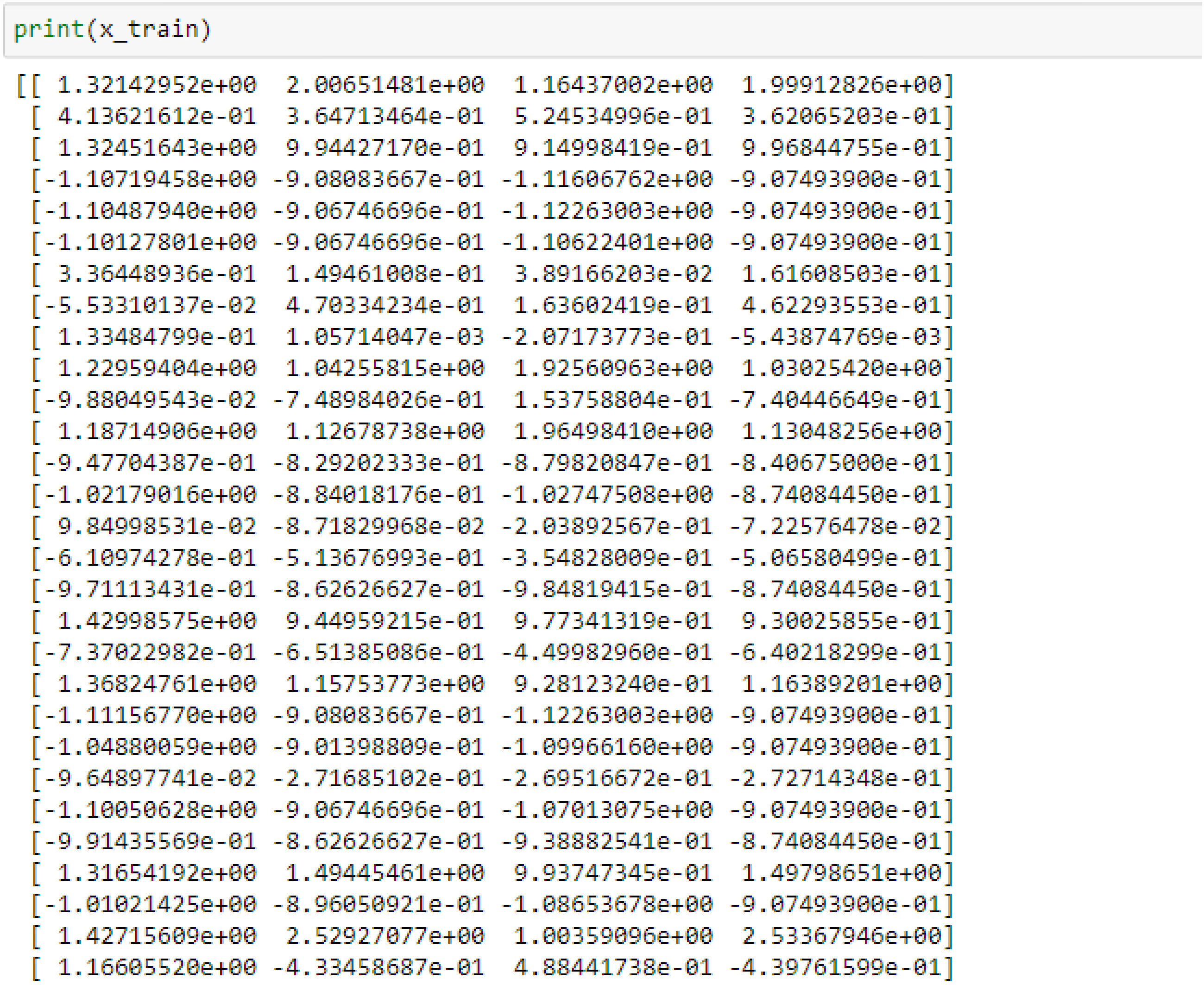


```
print(y_test)
```

```
[3 0 3 3 3]
```

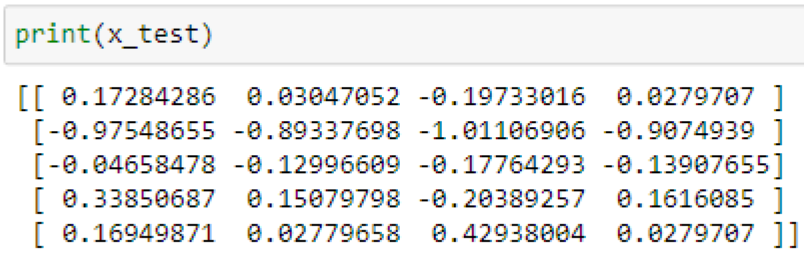


```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```



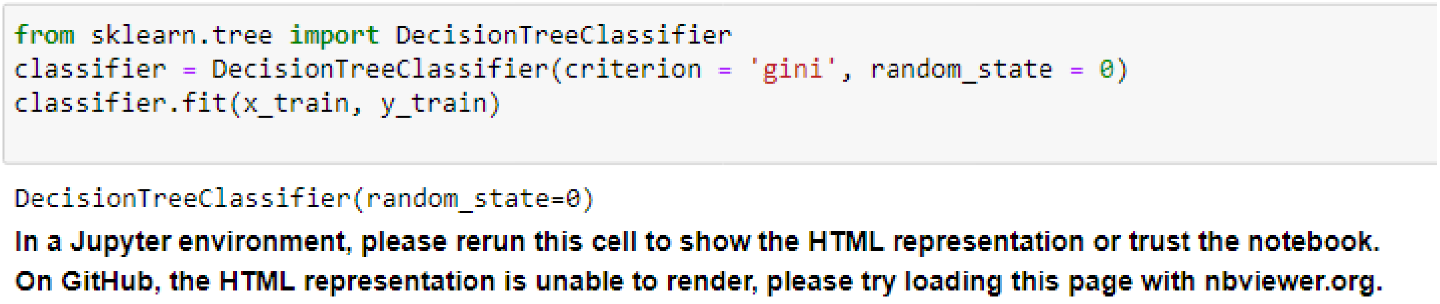
```
print(x_train)
```

```
[[ 1.32142952e+00  2.00651481e+00  1.16437002e+00  1.99912826e+00]
 [ 4.13621612e-01  3.64713464e-01  5.24534996e-01  3.62065203e-01]
 [ 1.32451643e+00  9.94427170e-01  9.14998419e-01  9.96844755e-01]
 [-1.10719458e+00 -9.08083667e-01 -1.11606762e+00 -9.07493900e-01]
 [-1.10487940e+00 -9.06746696e-01 -1.12263003e+00 -9.07493900e-01]
 [-1.10127801e+00 -9.06746696e-01 -1.10622401e+00 -9.07493900e-01]
 [ 3.36448936e-01  1.49461008e-01  3.89166203e-02  1.61608503e-01]
 [-5.53310137e-02  4.70334234e-01  1.63602419e-01  4.62293553e-01]
 [ 1.33484799e-01  1.05714047e-03 -2.07173773e-01 -5.43874769e-03]
 [ 1.22959404e+00  1.04255815e+00  1.92560963e+00  1.03025420e+00]
 [-9.88049543e-02 -7.48984026e-01  1.53758804e-01 -7.40446649e-01]
 [ 1.18714906e+00  1.12678738e+00  1.96498410e+00  1.13048256e+00]
 [-9.47704387e-01 -8.29202333e-01 -8.79820847e-01 -8.40675000e-01]
 [-1.02179016e+00 -8.84018176e-01 -1.02747508e+00 -8.74084450e-01]
 [ 9.84998531e-02 -8.71829968e-02 -2.03892567e-01 -7.22576478e-02]
 [-6.10974278e-01 -5.13676993e-01 -3.54828009e-01 -5.06580499e-01]
 [-9.71113431e-01 -8.62626627e-01 -9.84819415e-01 -8.74084450e-01]
 [ 1.42998575e+00  9.44959215e-01  9.77341319e-01  9.30025855e-01]
 [-7.37022982e-01 -6.51385086e-01 -4.49982960e-01 -6.40218299e-01]
 [ 1.36824761e+00  1.15753773e+00  9.28123240e-01  1.16389201e+00]
 [-1.11156770e+00 -9.08083667e-01 -1.12263003e+00 -9.07493900e-01]
 [-1.04880059e+00 -9.01398809e-01 -1.09966160e+00 -9.07493900e-01]
 [-9.64897741e-02 -2.71685102e-01 -2.69516672e-01 -2.72714348e-01]
 [-1.10050628e+00 -9.06746696e-01 -1.07013075e+00 -9.07493900e-01]
 [-9.91435569e-01 -8.62626627e-01 -9.38882541e-01 -8.74084450e-01]
 [ 1.31654192e+00  1.49445461e+00  9.93747345e-01  1.49798651e+00]
 [-1.01021425e+00 -8.96050921e-01 -1.08653678e+00 -9.07493900e-01]
 [ 1.42715609e+00  2.52927077e+00  1.00359096e+00  2.53367946e+00]
 [ 1.16605520e+00 -4.33458687e-01  4.88441738e-01 -4.39761599e-01]
```



```
print(x_test)
```

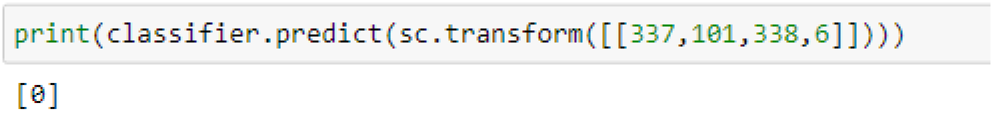
```
[[ 0.17284286  0.03047052 -0.19733016  0.0279707 ]
 [-0.97548655 -0.89337698 -1.01106906 -0.9074939 ]
 [-0.04658478 -0.12996609 -0.17764293 -0.13907655]
 [ 0.33850687  0.15079798 -0.20389257  0.1616085 ]
 [ 0.16949871  0.02779658  0.42938004  0.0279707 ]]
```



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

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

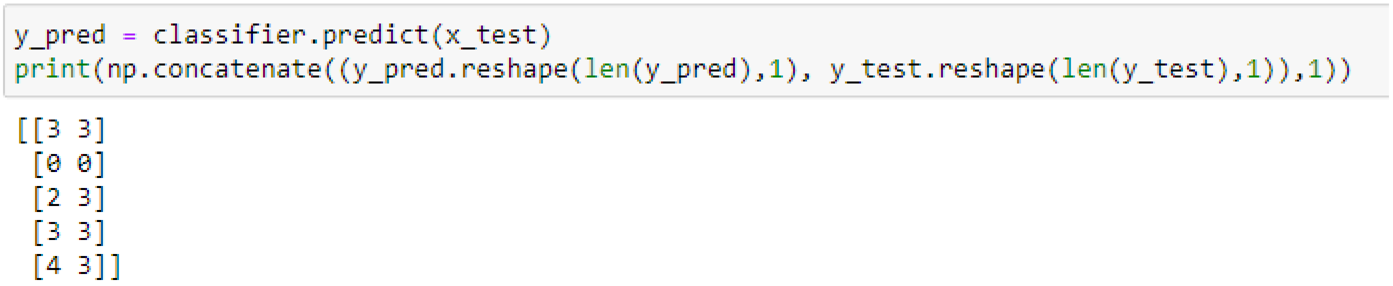
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook. On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.



```
print(classifier.predict(sc.transform([[337,101,338,6]])))
```

```
[0]
```

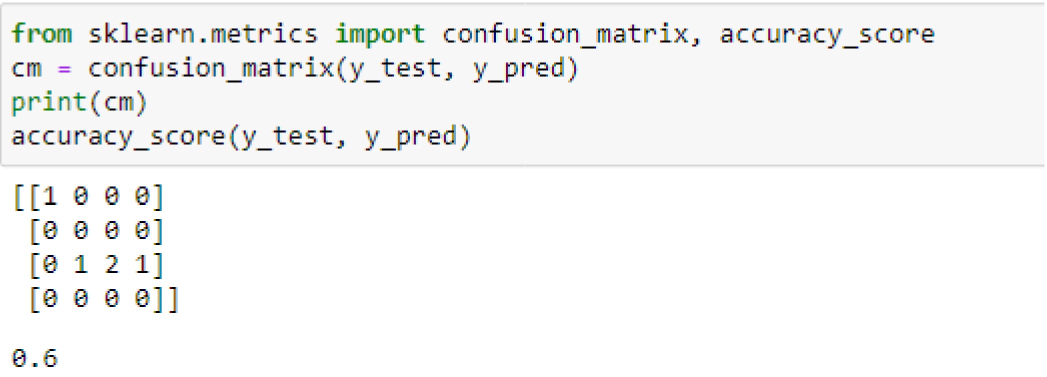
#Predicting the Test set results



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

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

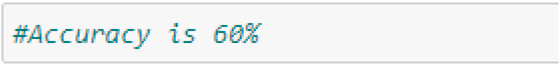
#Making the Confusion Matrix



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

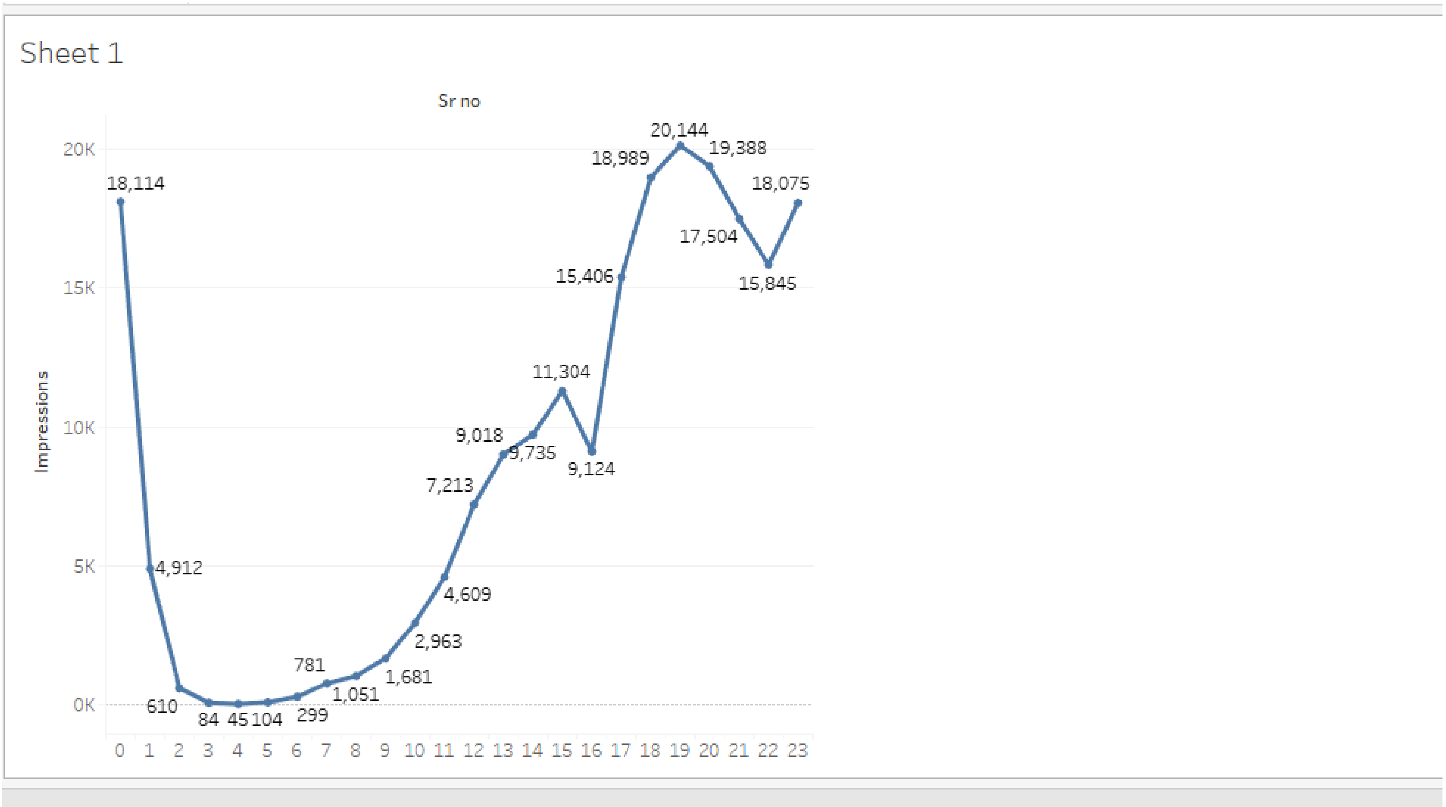
```
[[1 0 0 0]
 [0 0 0 0]
 [0 1 2 1]
 [0 0 0 0]]
```

```
0.6
```



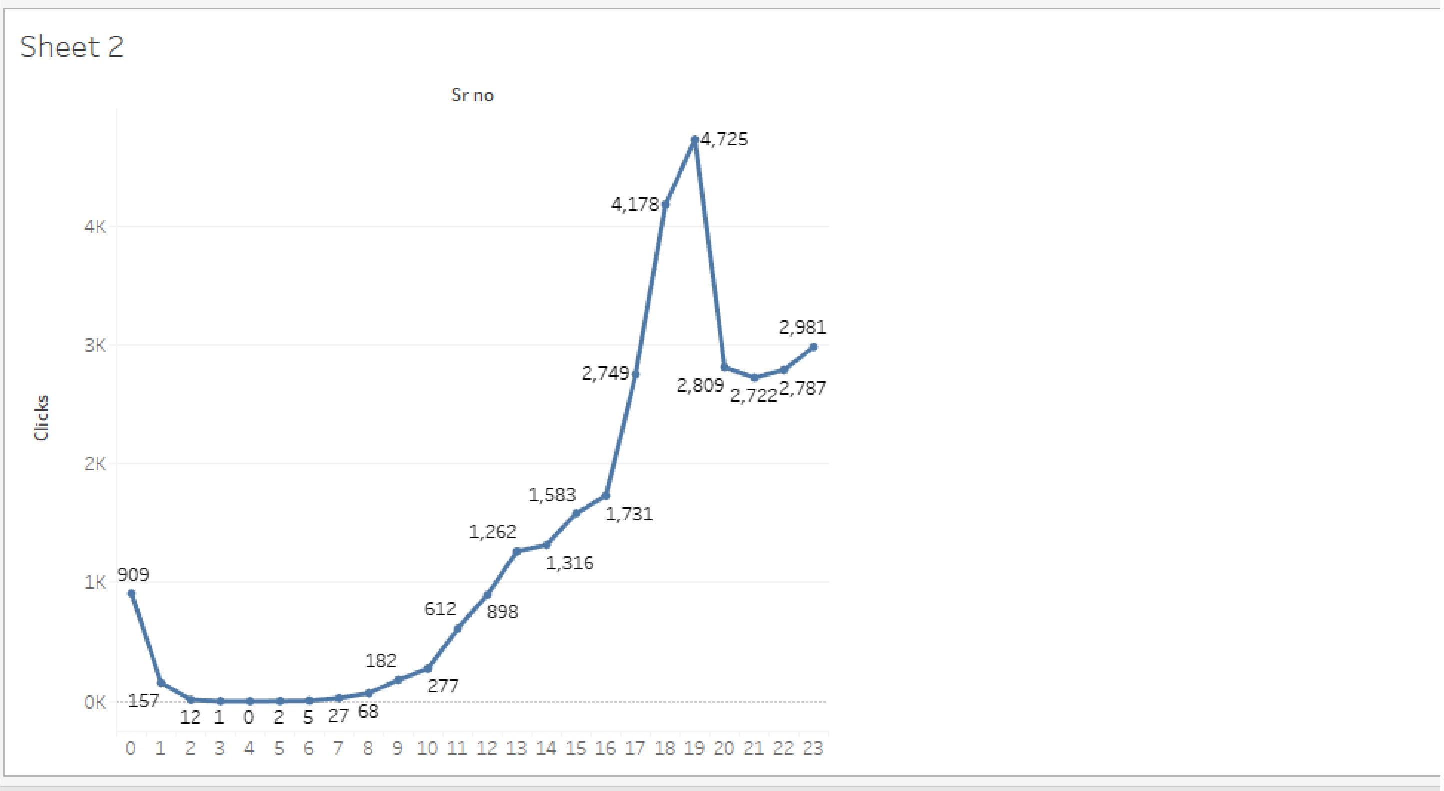
```
#Accuracy is 60%
```

#Visualization



Insights of the Impressions Graph

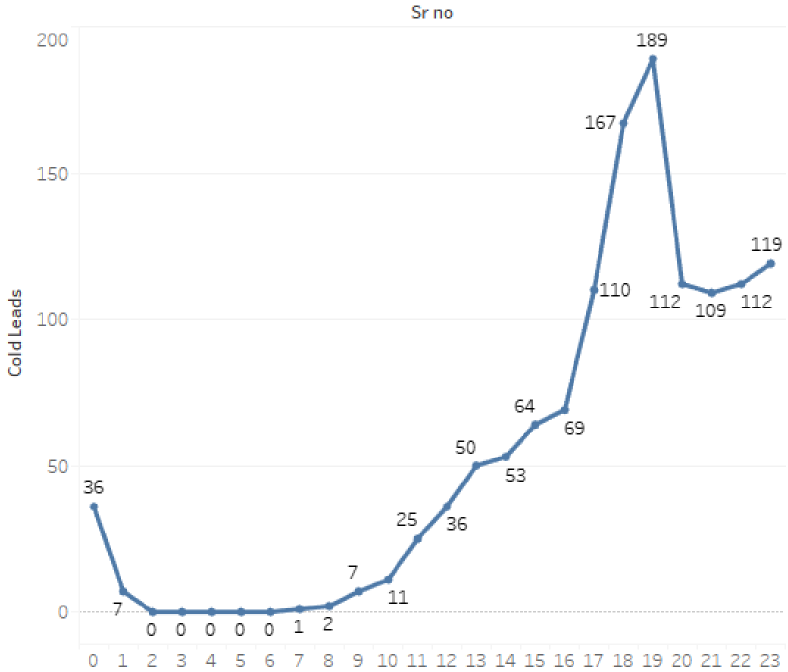
- Impressions are **high** at the night from **12:00 am to 1:00 am.**
- Then Impressions **are falling down after 1:30 am from 1:30 am to 6:00 am** because majority of the people are sleeping
- From **6:30 am to 11:30 am** impressions are **slightly high and going to increases** because on that time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



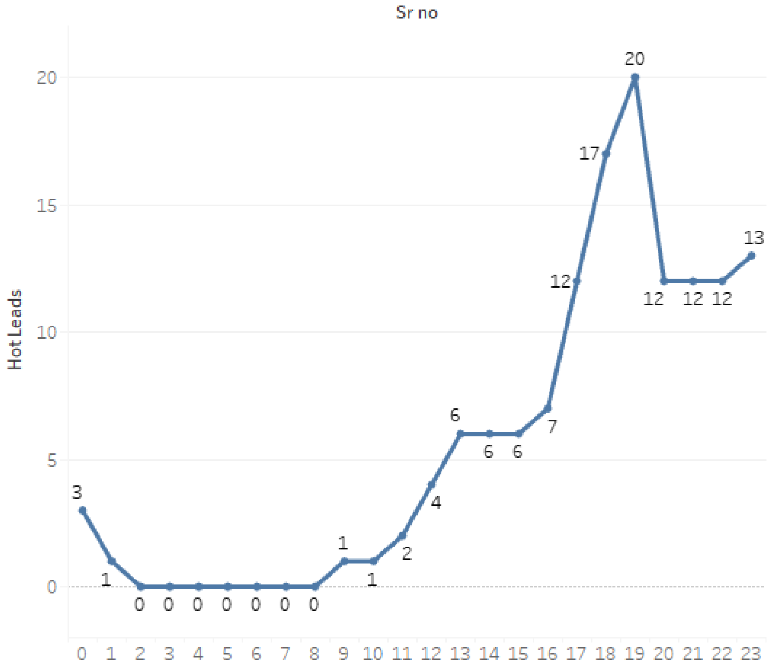
Insights of the Clicks Graph

- 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.**

Sheet 3



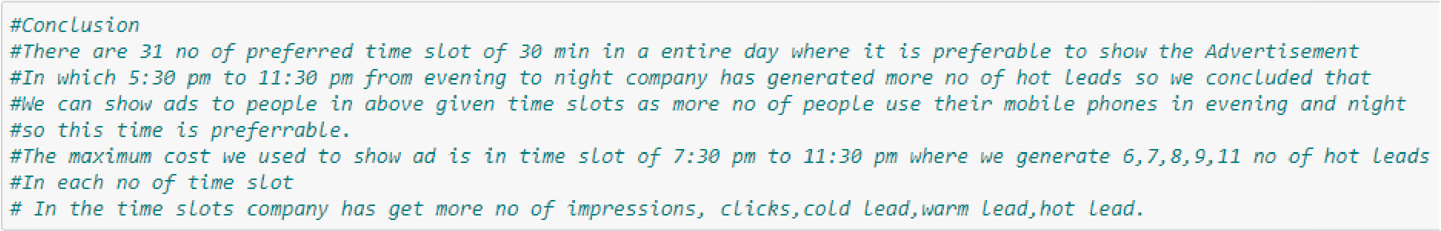
Sheet 4



Insights of Cold Lead and Hot Lead Graph

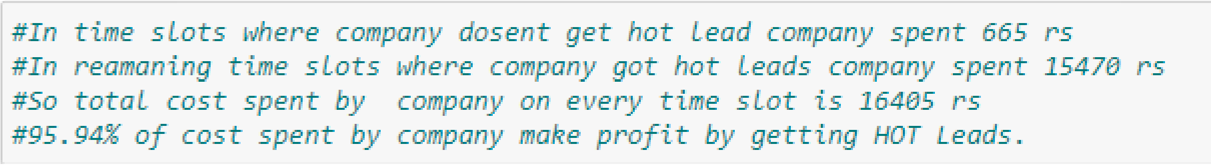
- Very few people are showing interest in that particular Ad **from 12:00 am to 1:00 am**
- Then after 1:00 am **no one candidate** is showing interest in that Ad from **1:30 am to 7:00 am** because lot of people are sleeping
- After 7:30 am only 1, 2 candidates are showings interest in particular time 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 1,2,3,4 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 5,6 ,7,8,9,11people are showing interest and they like to purchase that service

#Conclusion

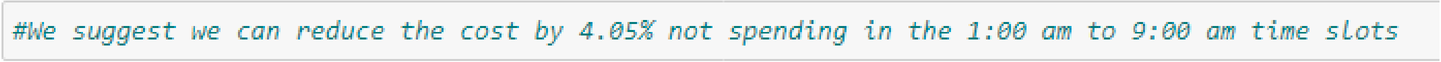


```
#Conclusion
#There are 31 no of preferred time slot of 30 min in a entire day where it is preferable to show the Advertisement
#In which 5:30 pm to 11:30 pm from evening to night company has generated more no of hot leads so we concluded that
#We can show ads to people in above given time slots as 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 11:30 pm where we generate 6,7,8,9,11 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 17 no of non preferred time slots and that company dosent get hot lead
```



```
#In time slots where company dosent get hot lead company spent 665 rs
#In reamaning time slots where company got hot leads company spent 15470 rs
#So total cost spent by  company on every time slot is 16405 rs
#95.94% of cost spent by company make profit by getting HOT Leads.
```



```
#We suggest we can reduce the cost by 4.05% not spending in the 1:00 am to 9:00 am time slots
```

Comparative Analysis of 19th June, 20th June 2023

I have started this live project working from 19th June to 21st June.

The data is about time slots showing ads and generate leads. Each time slot is divide into 30 min these time slots seems to be effective for showing ads and it shows a good time to focus on displaying ads to reach maximum leads and engage the leads. We saw that time slot from 5:00 pm to 12:00 am have generated hot leads in this slots. The total no of hot lead generated on 19th June are**73**. And ratio of converting cold lead **to hot lead is 18.62%.**and on 19th June we make profit of **96%** from cost spent on hot leads. Company spent total amount of **20057**for ads. **Company spent 264rs for each hot lead. maximum CTR for 19th June is 5.21%.**

Data we observed on 20th June is bit similar to 20th June as at on same time slots of 5:00 pm to 12:00 am have generated higher no of hot leads in this slots. The total no of hot lead generated on 20th June are **135**. And ratio of converting cold lead to hot lead is **10.67%.**and on 20th June we make profit of **95.94%** from cost spent on hot leads. Company spent total amount of **16405** for ads. **Company spent 116rs for each hot lead. maximum CTR for 20th June is 5.21%.**