In [1]: ```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_rem
ount=True).
```

In [2]: ```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]: ```
df = pd.read_csv('/content/drive/My Drive/School/Datasets/Marketing_AB.csv')
```

Out[3]:
```
   Unnamed: 0 user id test group converted total ads most ads day most ads hour
0   0 1069124   ad   False   130   Monday   20
1   1 1771671   ad   False   93   Tuesday   22
2   2 1404961   ad   False   493   Friday   18
3   3 1433130   ad   False   305   Tuesday   10
4   4 1055780   ad   False   276   Friday   14
```

In [4]: ```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 588101 entries, 0 to 588100
Data columns (total 7 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Unnamed: 0           588101 non-null  int64
 1   user id              588101 non-null  int64
 2   test group           588101 non-null  object
 3   converted            588101 non-null  bool
 4   total ads            588101 non-null  int64
 5   most ads day         588101 non-null  object
 6   most ads hour        588101 non-null  int64
dtypes: bool(1), int64(5), object(1)
memory usage: 27.5+ MB
```

In [5]: ```
df.drop(columns = ['Unnamed: 0', 'user id'], inplace= True)
```

In [6]: ```
df.describe()
```

Out[6]:
```
            total ads  most ads hour
count  588101.000000  588101.000000
mean      24.820875      14.469061
std       43.715181       4.834634
min        1.000000       0.000000
25%        4.000000      11.000000
50%       13.000000      14.000000
75%       27.000000      18.000000
max      2065.000000      23.000000
```

In [7]: ```
df.nunique()
```

Out[7]:
```
test group   2
converted   2
total ads   807
most ads day   7
most ads hour 24
dtype: int64
```

In [8]: ```
df.isna().sum()
```

Out[8]:
```
test group    0
converted     0
total ads     0
most ads day  0
most ads hour 0
dtype: int64
```

In [9]: ```
df[df['test group', 'converted', 'most ads day']].value_counts()
```

Out[9]:
```
test group converted most ads day
ad   False   Friday   84012
                       Monday   80783
                       Saturday   80205
                       Sunday   77993
                       Thursday   77125
                       Wednesday   75403
                       Tuesday   74571
psa   False   Thursday   3656
                       Friday   3541
                       Saturday   3392
                       Monday   3322
                       Sunday   3223
                       Tuesday   2865
ad   True   Monday   2779
                       Tuesday   2270
                       Sunday   2197
                       Friday   1995
                       Wednesday   1963
                       Thursday   1877
                       Saturday   1679
psa   True   Sunday   79
                       Thursday   74
                       Monday   67
                       Friday   62
                       Wednesday   55
                       Tuesday   51
                       Saturday   40
dtype: int64
```

In [10]: ```
df[ 'test group', 'converted', 'most ads day'].value_counts()
```

Out[10]:
```
ad  False    564077
psa False    23924
Name: test group, dtype: int64
```

In [11]: ```
df['converted',].value_counts()
```

Out[11]:
```
False   573258
True   14843
Name: converted, dtype: int64
```

In [12]: ```
df[ 'most ads day'].value_counts()
```

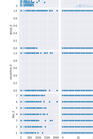
Out[12]:
```
Friday    92628
Monday    84451
Sunday    81351
Thursday  82062
Saturday  80079
Wednesday 82058
Tuesday   73479
Name: most ads day, dtype: int64
```

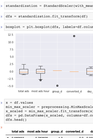
In [13]: ```
df.groupby('total ads').size().head()
```

Out[13]:
```
           converted most ads hour
total ads
1   1   1
2   4   14
3   8   34
4   34   104
5   42   158
```

In [14]: ```
df[df['test group', 'converted', 'most ads day', 'most ads hour']].sample(10)
df[df['test group', 'converted', 'most ads day', 'most ads hour']].shape
df[df['test group', 'converted', 'most ads day', 'most ads hour']].tail(10)
df[df['test group', 'converted', 'most ads day', 'most ads hour']].head(10)
```

In [15]: ```
sns.set_style('darkgrid')
plt.figure(figsize=(7, 5))
sns.lineplot(data=df[df['total ads', 'p(*|*data*)']])
```

In [16]: ```
condition = (df['test group'] == 'ad')
values = [1, 0]
df['group_d'] = np.select(condition, values)

condition = (df['converted'] == False)
values = [0, 1]
df['converted_d'] = np.select(condition, values)

condition = (df['most ads day'] == 'Monday')
| (df['most ads day'] == 'Sunday')
| (df['most ads day'] == 'Tuesday')
| (df['most ads day'] == 'Wednesday')
| (df['most ads day'] == 'Thursday')
| (df['most ads day'] == 'Friday')
| (df['most ads day'] == 'Saturday')
values = [1, 2, 3, 4, 5, 6, 7]
df['day_d'] = np.select(condition, values)
```

In [17]: ```
df.drop(columns = ['test group', 'converted', 'most ads day'], inplace= True)
```

In [18]: ```
df['day_d'].hist()
```

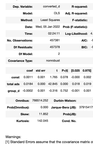
In [19]: ```
df['most ads hour'].value_counts()
```

Out[19]:
```
13   47025
12   41998
11   40219
14   39994
15   34962
10   34921
16   27567
17   21029
18   16126
9   13354
19   12888
20   10976
21   10413
22   8723
23   7577
8   5422
7   3827
6   2452
5   1492
4   804
3   411
2   165
1   63
0   34
Name: most ads hour, dtype: int64
```

In [20]: ```
df['most ads hour'].hist()
```



In [21]: ```
df['total ads'].hist()
```




In [22]: ```
plt.figure(figsize=(25, 21))
sns.pairplot(df)
#figure size 1440x864 with 0 Axes#
```

In [23]: ```
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
```

In [24]: ```
standardization = StandardScaler(with_mean=True, with_std=True)
```

In [25]: ```
dfs = standardization.fit_transform(df)
```

In [26]: ```
boxplot = plt.boxplot(dfs, labels=df.columns)
```



In [27]: ```
y = df['total ads']
mlp_mod_scaler = preprocessing.MinMaxScaler()
dfs = mlp_mod_scaler.fit_transform(df)
dfs = pd.DataFrame(dfs, columns=df.columns)
dfs.head()
```

Out[27]:
```
   total ads most ads hour group_d converted_d   day_d
0   0.000000   0.434783   1.0   0.0 0.333333
1   0.045015   0.739130   1.0   0.0 0.333333
2   0.238603   0.608187   1.0   0.0 0.333333
3   0.146970   0.434783   1.0   0.0 0.333333
4   0.131960   0.739130   1.0   0.0 0.333333
```

In [28]: ```
dfs.corr()
```

Out[28]:
```
               total ads  most ads hour  group_d  converted_d     day_d
total ads       1.000000      -0.000042 -0.011885    0.626842  0.000622
most ads hour  -0.000042       1.000000  0.000394    0.000402  0.027546
group_d        -0.011885       0.000394  1.000000    0.000000  0.001661
converted_d     0.626842       0.000402  0.000000    1.000000  0.000000
day_d           0.000622       0.027546  0.001661    0.000000  1.000000
```

In [29]: ```
plt.figure(figsize=(15,15))
sns.heatmap(dfs.corr(), annot=True, cmap='Blues')
```




In [30]: ```
# Define the dependent variable to the and the name of handling
# dfs.columns
dfs.drop(columns=['total ads', 'converted_d'])
# dfs.columns
y = dfs['converted_d']
X = dfs.iloc[:,:1]
variables = X.columns
```

In [31]: ```
import statsmodels.api as sm
X = sm.add_constant(X)
linear_regression = sm.OLS(y, X)
fitted_model = linear_regression.fit()
fitted_model.summary()
```

Out[31]:
```
                            OLS Regression Results
==
Dep. Variable: converted_d   R-squared: 0.004
Model: OLS   Adj. R-squared: 0.004
Method: Least Squares   F-statistic: 457.7
Date: Wed, 04 Jan 2023   Prob (F-statistic):   0.00
Time: 02:16:04   Log-Likelihood: -2.0104e+05
No. Observations: 588   AIC: 4.021e+05
Df Residuals: 586   BIC: 4.022e+05
Df Model: 1
Covariance Type: nonrobust
==
                 coef   std err   t   P>|t| [0.025   0.975]
----
const   0.0000   0.001   0.000   1.000   -0.001   0.001
total ads   0.0746   0.000   20.961   0.000   0.069   0.079
most ads hour -0.0006   0.001   -0.107   0.915   -0.002   0.001
group_d   -0.0000   0.001   -0.000   1.000   -0.002   0.002
day_d   -0.0005   0.000   -2.140   0.033   -0.001   -0.000
==
Omnibus: 1164824.767   Durbin-Watson: 1.999
Prob(Omnibus):   0.000   Jarque-Bera (JB):   2197873.589
Skew: 7.998   Prob(JB):   0.00
Kurtosis: 56.490   Cond. No.   10.2
```

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [32]: ```
coef = sg.coef(fitted_model)
argmaxcoef, argmaxfactor = np.linalg.eig(coef)
print(argmaxcoef)
print(argmaxfactor)
```

```
[0.0000000 0.0746211 0.0155263 0.0283381 ]
```

In [33]: ```
print(argmaxcoef[1:,])
```

```
[ 0.0746207  0.0155258  -0.0283403  -0.0422175]
```

In [34]: ```
resid=fitted
```

Out[35]: ```
Index(['total ads', 'most ads hour', 'group_d', 'day_d'], dtype='object')
```

In [35]: ```
df.drop(columns=['most ads hour', 'day_d'], inplace=True)
```

In [36]: ```
y = df['converted_d']
X = df[['total ads']]
X = sm.add_constant(X)
linear_regression = sm.OLS(y, X)
fitted_model = linear_regression.fit()
fitted_model.summary()
```

Out[36]:
```
                            OLS Regression Results
==
Dep. Variable: converted_d   R-squared: 0.004
Model: OLS   Adj. R-squared: 0.004
Method: Least Squares   F-statistic: 457.8
Date: Wed, 04 Jan 2023   Prob (F-statistic):   0.00
Time: 02:16:11   Log-Likelihood: -2.0104e+05
No. Observations: 588   AIC: 4.021e+05
Df Residuals: 586   BIC: 4.022e+05
Df Model: 1
Covariance Type: nonrobust
==
                 coef   std err   t   P>|t| [0.025   0.975]
----
const   0.0000   0.001   0.000   1.000   -0.001   0.001
total ads   0.0746   0.000   20.964   0.000   0.069   0.079
Omnibus: 1164824.767   Durbin-Watson: 1.999
Prob(Omnibus):   0.000   Jarque-Bera (JB):   2197873.589
Skew: 7.998   Prob(JB):   0.00
Kurtosis: 56.490   Cond. No.   10.2
```

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [37]: ```
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(X, y)
print('Intercept: ', reg.intercept_)
print('Model coefficients: ', reg.coef_)
```

```
Intercept:  0.19119999583633166
Model coefficients:  [ 0.07463189 -0.01263189 -0.01263189]
```

In [38]: ```
```
```