Analyzing A/B Testing For an E-commerce Website

PROJECT REPORT

UE20CS312- Data Analytics

TEAM DETAILS :

Team Member 1 :

SRN : PES1UG20CS535

NAME : ALMAS BANU

SECTION : I

Team Member 2 :

SRN : PES1UG20CS595

NAME : SNEHAL KUMAR ROY

SECTION : J

A/B Testing

Objective

- The goal of this project is to support decision making for an e-commerce company by analyzing the results of an A/B test.
- The company needs to decide whether they should implement the new version of their web page or keep the old page, or perhaps run the experiment longer to make their decision.

Methodology: -

- **ab_data.csv and countries.csv is used .**
- **Assessed the data, and made decisions about dealing with duplicates, and mismatched values.**
- **Performed probability computations, hypothesis testing and logistic regression on the data using Pandas, Numpy, and the statsmodels module in Python.**
- **Made recommendations backed by statistical inferences for deciding the web page version and documented limitations of the analysis.**

Dataset: -

- A/B Testing - Kaggle

URL: -

https://www.kaggle.com/datasets/zhangluyuan/ab-testing

Columns: -

- Column 1 -> user_id
- Column 2 -> timestamp
- Column 3 -> group
- Column 4 -> landing_page
- Column 5 -> Converted

PROBLEM STATEMENT

The problem statement involves implementing A/B testing for a given data set to ensure increase in user engagement, reduction in bounce rates, increase in conversion rates, minimizing risk, and effectively creating content for websites. Running an A/B test can have significant positive effects on the website.

What is A/B Testing?

A/B Test is the shorthand for a simple controlled experiment.

A/B Testing is a way to compare two versions of a single variable,typically by testing a subject's response to variant A against variant B,and determining which of the two variants is more effective.

STEPS

- Research
- Observe and Formulate Hypothesis
- Create Variations
- Run Test
  Multipage Testing
- Result Analysis and Deployment.

FORMULATING HYPOTHESIS

We formulate a hypothesis at the start of our project. This will make sure our interpretation of the results is correct as well as rigorous.
As our current design, we'll choose a <u>two-tailed test</u>:

$H_o: p = p_o \qquad H_a: p \neq p_o$

where *p* and p_o stand for the conversion rate of the new and old design, respectively. We'll also set a confidence level of 95%: $\alpha = 0.05$

The α value is a threshold we set, by which we say "if the probability of observing a result as extreme or more (*p*-value) is lower than α, then we reject the Null hypothesis". Since our α=0.05 (indicating 5% probability), our confidence (1 — α) is 95%.

Libraries used

```
import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

About Dataset

Now, read in the ab_data.csv data. Store it in df.

In [2]:
```
df = pd.read_csv('ab_data.csv')
df.head()
```

Out[2]:

	user_id	timestamp	group	landing_page	converted
0	851104	2017-01-21 22:11:48.556739	control	old_page	0
1	804228	2017-01-12 08:01:45.159739	control	old_page	0
2	661590	2017-01-11 16:55:06.154213	treatment	new_page	0
3	853541	2017-01-08 18:28:03.143765	treatment	new_page	0
4	864975	2017-01-21 01:52:26.210827	control	old_page	1

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 294478 entries, 0 to 294477
Data columns (total 5 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   user_id       294478 non-null  int64
 1   timestamp     294478 non-null  object
 2   group         294478 non-null  object
 3   landing_page  294478 non-null  object
 4   converted     294478 non-null  int64
dtypes: int64(2), object(3)
memory usage: 11.2+ MB
```

```
In [34]: df.shape
Out[34]: (294478, 5)
```

Summary of Data

```
In [4]: df.describe()
Out[4]:
```

	user_id	converted
count	294478.000000	294478.000000
mean	787974.124733	0.119659
std	91210.823776	0.324563
min	630000.000000	0.000000
25%	709032.250000	0.000000
50%	787933.500000	0.000000
75%	866911.750000	0.000000
max	945999.000000	1.000000

The number of unique users in the dataset.

```
unique_user = df['user_id'].nunique()
unique_user
```

```
290584
```

The proportion of users converted.

```
df.converted.mean()
```

```
0.11965919355605512
```

The number of times the new_page and treatment don't line up.

```python
not_line_up_1 = df.query(" group=='treatment' and landing_page=='old_page'").count()
```

```python
not_line_up_2 = df.query(" group=='control' and landing_page=='new_page'").count()
```

```python
not_line_up_1 + not_line_up_2
```

```
user_id         3893
timestamp       3893
group           3893
landing_page    3893
converted       3893
dtype: int64
```

Exploratory Data Analysis

Checking for missing values and replacing them

In [12]:
```python
df.isnull().sum()
```

Out[12]:
```
user_id         0
timestamp       0
group           0
landing_page    0
converted       0
dtype: int64
```

There are no null values in the used dataset.

For the rows where **treatment** is not aligned with **new_page** or **control** is not aligned with **old_page**, we cannot be sure if this row truly received the new or old page.

In [41]:
```python
df2 = df[((df['group'] == 'treatment') == (df['landing_page'] == 'new_page')) == True]
df2.head()
```

Out[41]:

	user_id	timestamp	group	landing_page	converted
0	851104	2017-01-21 22:11:48.556739	control	old_page	0
1	804228	2017-01-12 08:01:45.159739	control	old_page	0
2	661590	2017-01-11 16:55:06.154213	treatment	new_page	0
3	853541	2017-01-08 18:28:03.143765	treatment	new_page	0
4	864975	2017-01-21 01:52:26.210827	control	old_page	1

Double Check all of the incorrect rows were removed - this should be 0

```
df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')) == False].shape[0]
```

```
0
```

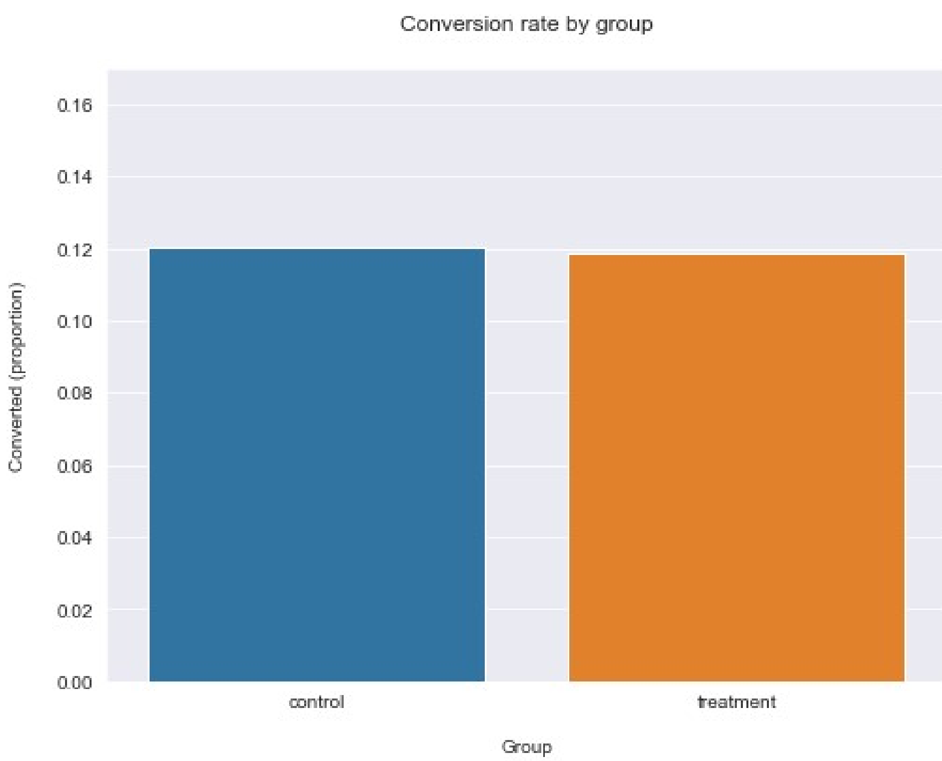
Visualization

Using sns.barplot

```
plt.figure(figsize=(8,6))

sns.barplot(x=df2['group'], y=df2['converted'], ci=False)

plt.ylim(0, 0.17)
plt.title('Conversion rate by group', pad=20)
plt.xlabel('Group', labelpad=15)
plt.ylabel('Converted (proportion)', labelpad=15);
```



Control numbers are higher than treatment

unique user_ids are in df2

```
df2['user_id'].nunique()
```

290584

There is 1 repeated

```
df2['user_id'].duplicated().sum()
```

1

Row information for the repeat user_id

```
df2[df2['user_id'].duplicated(keep=False)]
```

	user_id	timestamp	group	landing_page	converted
1899	773192	2017-01-09 05:37:58.781806	treatment	new_page	0
2893	773192	2017-01-14 02:55:59.590927	treatment	new_page	0

Removing one of the rows with a duplicate user_id, but keep your dataframe as df2

```
In [47]: # drop duplicates
         df2.drop_duplicates(keep='first')
         df2.duplicated().sum()

Out[47]: 0
```

The probability of an individual converting regardless of the page they receive

```
round(df2.converted.mean(),4)
```

0.1196

Given that an individual was in the control group,Lets calculate the probability that they converted

```
control=df2.query("group=='control'").converted.mean()
round(control,4)
```

0.1204

Given that an individual was in the treatment group, Lets calculate the probability that they converted

```
treat=df2.query("group=='treatment'").converted.mean()
round(treat,4)
```

```
0.1188
```

Probability that an individual received the new page

```
new_page = float(df2.query("landing_page == 'new_page'")['user_id'].nunique())
total = float(df2.shape[0])
round(new_page / total,4)
```

```
0.5001
```

The converted probability Given that an individual was in new landing page

```
old_page = float(df2.query("landing_page == 'new_page' and converted == 1 ")['user_id'].nunique()
total = float(df2.query("landing_page == 'new_page'")['user_id'].nunique())

round(old_page / total,4)
```

```
0.1188
```

About 12.04% control group is likely to be converted while 11.88% treatment group is likely to be converted. The result is quite similar. So there is no strong evidence to prove a certain page leads to more conversions.

A/B Test

HYPOTHESIS BASED ON CONVERSION

```
H_O : p(new)-p(old)<=0  old has better conversion
H_1 : p(new)-p(old)>0   new has better conversion
```

Convert Rate for p(new) and p(old) under null

```
# Convert rate for p(new)
p_n = round(float(df2.converted.mean()),6)
p_n
```

```
0.119597
```

```
In [26]: # Convert rate for p(old)
         p_o =  round(float(df2.converted.mean()),6)
         p_o

Out[26]: 0.119597
```

Here we are looking at a null where there is no difference in conversion based on the page, which means the conversions for each page are the same.

```
: # Unique Number of accessing new page
  N_new = df2.query('landing_page == "new_page"')['user_id'].nunique()
  N_new

: 145310
```

```
: # Unique Number of accessing old page
  N_old = df2.query('landing_page == "old_page"')['user_id'].nunique()
  N_old

: 145274
```

Simulate n(new) transactions with a convert rate of p(new) under the null. Store these n(new) 1's and 0's in new_page_converted.

```
: new_page_converted = np.random.choice([0,1] , N_new , p=(p_n,1-p_n))
  new_page_converted

: array([1, 1, 1, ..., 1, 1, 1])
```

Simulate n(old) transactions with a convert rate of p(old) under the null.Store these n(old) 1's and 0's in old_page_converted.

```
old_page_converted = np.random.choice([0,1] , N_old , p=(p_o,1-p_o))
old_page_converted

array([1, 1, 1, ..., 1, 1, 1])
```

Means

```
new_page_converted.mean() , old_page_converted.mean()

(0.880978597481247, 0.8802125638448725)
```

Find p(new) -p(old) for the simulated values

```
obs_diff = new_page_converted.mean() - old_page_converted.mean()
obs_diff
```

```
0.0007660336363745079
```

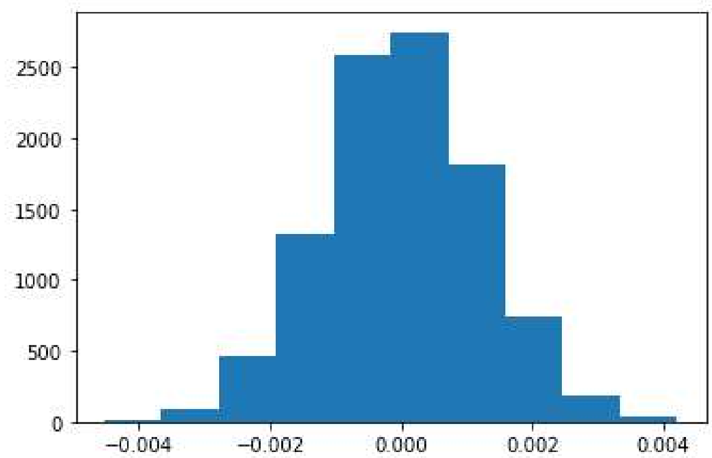
Create sampling distribution for difference in completion rates with boostrapping

```
|: p_diffs=[]
   new_convert=np.random.binomial(N_new, p_n, 10000)/N_new
   old_convert=np.random.binomial(N_old, p_o, 10000)/N_old
   p_diffs=new_convert-old_convert
```

Plot a histogram of the p_diffs

```
p_diffs = np.array(p_diffs)
plt.hist(p_diffs)
```

```
(array([  12.,   96.,  461., 1325., 2577., 2746., 1820.,  746.,  178.,
          39.]),
 array([-0.00453093, -0.00365685, -0.00278277, -0.00190869, -0.00103461,
        -0.00016053,  0.00071354,  0.00158762,  0.0024617 ,  0.00333578,
         0.00420986]),
 <BarContainer object of 10 artists>)
```



What proportion of the p_diffs are greater than the actual difference observed in ab_data.csv

```
converted_new = df2.query('converted == 1 and landing_page== "new_page"')['user_id'].nunique()
actual_new = float(converted_new) / float(N_new)

# number of landing old page and converted  / number of landing old page
converted_old = df2.query('converted == 1 and landing_page== "old_page"')['user_id'].nunique()
actual_old = float(converted_old) / float(N_old)

#observed difference in converted rate
obs_diff = actual_diff = actual_new - actual_old
obs_diff
```

```
-0.0015782389853555567
```

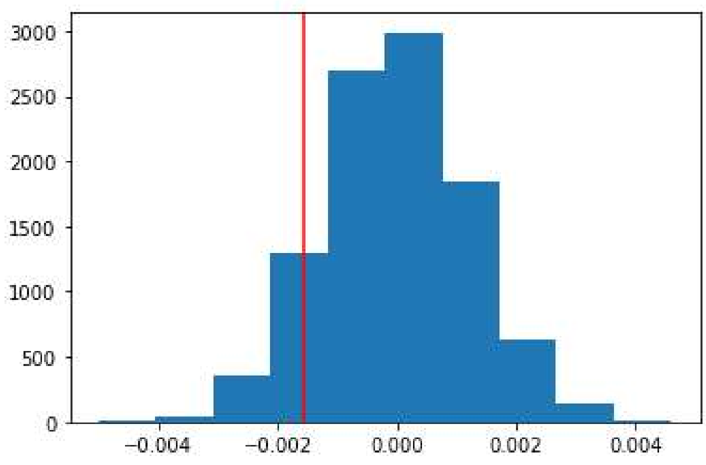
Create distribution under null hypothesis

```
# create distribution under the null hypothesis
null_vals = np.random.normal(0, p_diffs.std(), p_diffs.size)
```

Plot histogram

```
#Plot Null distribution
plt.hist(null_vals)
#Plot vertical line for observed statistic
plt.axvline(x=obs_diff,color ='red')
```

```
<matplotlib.lines.Line2D at 0x1ce52987b50>
```




```
(null_vals > obs_diff).mean()
```

```
0.9029
```

```
import statsmodels.api as sm

convert_old = df2.query('converted == 1 and landing_page== "old_page"').user_id.nunique()
convert_new = converted_old = df2.query('converted == 1 and landing_page== "new_page"').user_id.nunique()
n_old = df2.query('landing_page == "old_page"')['user_id'].nunique()
n_new = df2.query('landing_page == "new_page"')['user_id'].nunique()

convert_old,convert_new , n_old , n_new
```

```
(17489, 17264, 145274, 145310)
```

```
# compute the sm.stats.proportions_ztest using the alternative
z_score, p_value = sm.stats.proportions_ztest(np.array([convert_new,convert_old])
                                              ,np.array([n_new,n_old]), alternative = 'larger')
z_score, p_value

#  it's a one tail test so a z-score past 1.96 will be significant.
```

```
(-1.3109241984234394, 0.9050583127590245)
```

The value 0.905 we call it P-value, which suggests if there is a significant difference between 2 groups for a hypothesis.

In this case, the new page doesn't have better conversion rates than the old page because the value 0.9 is much higher than the alpha, 0.05(Type I error rate).

We fail to reject the null hypothesis. Therefore, this shows that, with a type I error rate of 0.05, the old page has higher probability of convert rate than new page.

MODELS

REASONS TO USE LOGISTIC REGRESSION

- Instead of defining one metric (a "conversion" rate), and running a proportions test or a t-test on it, we would like to use the coefficients on the logistic regression to see if there are interesting differences between the two designs.
- Lasso is a good way to deal with correlated features in logistic regression, and that the resulting coefficient can still be interpreted.
- If a logistic regression model can predict which design was seen by each user better than chance (+50% accuracy), we can conclude that the two designs have significantly different impact.

LOGISTIC REGRESSION

Under logistic model, the hypothesis test is under :

H0: p_new = p_old ,

H1: p_new ≠ p_old (Two sided test)

Create a column for the intercept

```
# create a column for the intercept
df2['intercept'] = 1
df2.head()
```

	user_id	timestamp	group	landing_page	converted	intercept
0	851104	2017-01-21 22:11:48.556739	control	old_page	0	1
1	804228	2017-01-12 08:01:45.159739	control	old_page	0	1
2	661590	2017-01-11 16:55:06.154213	treatment	new_page	0	1
3	853541	2017-01-08 18:28:03.143765	treatment	new_page	0	1
4	864975	2017-01-21 01:52:26.210827	control	old_page	1	1

ab_page column

```
# create a dummy variable column for which page each user received
df2['ab_page'] = pd.get_dummies(df['group'])['treatment']
df2.head()
```

	user_id	timestamp	group	landing_page	converted	intercept	ab_page
0	851104	2017-01-21 22:11:48.556739	control	old_page	0	1	0.0
1	804228	2017-01-12 08:01:45.159739	control	old_page	0	1	0.0
2	661590	2017-01-11 16:55:06.154213	treatment	new_page	0	1	1.0
3	853541	2017-01-08 18:28:03.143765	treatment	new_page	0	1	1.0
4	864975	2017-01-21 01:52:26.210827	control	old_page	1	1	0.0

Fitting the Model

```
log_mod = sm.Logit(df2['converted'], df2[['intercept','ab_page']])
results = log_mod.fit()
```

```
Optimization terminated successfully.
         Current function value: 0.366118
         Iterations 6
```

Summary of fitting Logistic Regression

```
results.summary()
```

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	290585
Model:	Logit	**Df Residuals:**	290583
Method:	MLE	**Df Model:**	1
Date:	Mon, 14 Nov 2022	**Pseudo R-squ.:**	8.085e-06
Time:	22:29:14	**Log-Likelihood:**	-1.0639e+05
converged:	True	**LL-Null:**	-1.0639e+05
Covariance Type:	nonrobust	**LLR p-value:**	0.1897

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	-1.9888	0.008	-246.669	0.000	-2.005	-1.973
ab_page	-0.0150	0.011	-1.312	0.190	-0.037	0.007

- **P-value is 0.19 which means 'ab_page' is not that significant in predicting whether or not the individual converts.**
- **H0 in this model is that 'ab_page' is totally insignificant in predicting the responses and we cannot reject H0 because it turned out it's probability is 19%.**
- **Again, it seems there is no difference whether or not we use new_page.**

To study the effect based on the country a user lives in

The countries.csv file is used here.

```
df_country = pd.read_csv('countries.csv')
df_country.head()
```

	user_id	country
0	834778	UK
1	928468	US
2	822059	UK
3	711597	UK
4	710616	UK

```
# join two dataframes on common column 'user_id'
df3 = df2.join(df_country.set_index('user_id'),on='user_id')
df3.head()
```

	user_id	timestamp	group	landing_page	converted	intercept	ab_page	country
0	851104	2017-01-21 22:11:48.556739	control	old_page	0	1	0	US
1	804228	2017-01-12 08:01:45.159739	control	old_page	0	1	0	US
2	661590	2017-01-11 16:55:06.154213	treatment	new_page	0	1	1	US
3	853541	2017-01-08 18:28:03.143765	treatment	new_page	0	1	1	US
4	864975	2017-01-21 01:52:26.210827	control	old_page	1	1	0	US

```
df3['country'].value_counts()
```

```
US    203620
UK     72466
CA     14499
Name: country, dtype: int64
```

```
# create dummy variables for country
df3[['US','UK','CA']] = pd.get_dummies(df3['country'])
df3 = df3.drop(df3['CA'])
df3['intercept'] = 1
log_mod = sm.Logit(df3['converted'], df3[['intercept','US','UK','ab_page']])
results = log_mod.fit()
results.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.366114
         Iterations 6
```

Logit Regression Results

Dep. Variable:	converted	No. Observations:	290583
Model:	Logit	Df Residuals:	290579
Method:	MLE	Df Model:	3
Date:	Mon, 14 Nov 2022	Pseudo R-squ.:	2.326e-05
Time:	22:29:17	Log-Likelihood:	-1.0639e+05
converged:	True	LL-Null:	-1.0639e+05
Covariance Type:	nonrobust	LLR p-value:	0.1756

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	-1.9893	0.009	-223.760	0.000	-2.007	-1.972
US	-0.0408	0.027	-1.516	0.129	-0.093	0.012
UK	0.0099	0.013	0.743	0.458	-0.016	0.036
ab_page	-0.0150	0.011	-1.309	0.191	-0.037	0.007

Create dummy variables for country

```
# create dummy variables for country
df3['intercept'] = 1
# interaction between page and country
df3['US_new'] = df3['US'] * df3['ab_page']
df3['UK_new'] = df3['UK'] * df3['ab_page']
log_mod = sm.Logit(df3['converted'], df3[['intercept','ab_page','US','UK','US_new' ,'UK_new' ]])
results = log_mod.fit()
results.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.366110
         Iterations 6
```

Logit Regression Results

Dep. Variable:	converted	No. Observations:	290583
Model:	Logit	Df Residuals:	290577
Method:	MLE	Df Model:	5
Date:	Mon, 14 Nov 2022	Pseudo R-squ.:	3.485e-05
Time:	22:40:13	Log-Likelihood:	-1.0639e+05
converged:	True	LL-Null:	-1.0639e+05
Covariance Type:	nonrobust	LLR p-value:	0.1915

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	-1.9865	0.010	-206.341	0.000	-2.005	-1.968
ab_page	-0.0206	0.014	-1.508	0.132	-0.047	0.006
US	-0.0176	0.038	-0.466	0.641	-0.091	0.056
UK	-0.0058	0.019	-0.307	0.759	-0.043	0.031
US_new	-0.0469	0.054	-0.871	0.384	-0.152	0.059
UK_new	0.0314	0.027	1.182	0.237	-0.021	0.084

- **There is no significant interaction between 'ab_page' and 'country' because the coefficient of 'ab_page' (-0.015) does not change as 'country' is introduced.**
- **P-values of 'ab_page' (0.191), and 'country' (0.129, 0.458) are considered insignificant, thus again, we failed to reject the Null hypothesis which is 'conversion has no significant relationship with landing_page or country'.**

Other Models Implemented

MLR
Decision Tree

Other Models

```
#Drop the timestamp column
df2=df2.drop(['timestamp'],axis=1)
df2.head()
```

	user_id	group	landing_page	converted	intercept	ab_page
0	851104	control	old_page	0	1	0
1	804228	control	old_page	0	1	0
2	661590	treatment	new_page	0	1	1
3	853541	treatment	new_page	0	1	1
4	864975	control	old_page	1	1	0

Train and Test Splitting with test_size = 0.2

Splitting the data

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(df2.loc[:, df2.columns != 'converted'],\
                                                    df2['converted'], test_size=0.2)
```

Label Encoding for the categorical values i.e group and landing_page column.

label encoding the categorical values

```
from sklearn.preprocessing import LabelEncoder

lb = LabelEncoder()
X_train['group'] = lb.fit_transform(X_train['group'])
X_test['group'] = lb.transform(X_test['group'])

X_train['landing_page'] = lb.fit_transform(X_train['landing_page'])
X_test['landing_page'] = lb.transform(X_test['landing_page'])
```

Defining function for printing the evaluation scores.

Function for printing the evaluation scores related to a regression problem

```
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
def calculate_metrics(y_test, y_preds):
    rmse = np.sqrt(mean_squared_error(y_test, y_preds))
    r_sq = r2_score(y_test, y_preds)
    mae = mean_absolute_error(y_test, y_preds)

    print('RMSE Score: {}'.format(rmse))
    print('R2_Squared: {}'.format(r_sq))
    print('MAE Score: {}'.format(mae))
```

Multiple Linear Regression

Using sm.OLS

MULTIPLE LINEAR REGRESSION

```
import statsmodels.api as sm

#user_id not significant hence drop
X_train_refined = X_train.drop(columns=['user_id'], axis=1)
linear_regression = sm.OLS(y_train, X_train_refined)
linear_regression = linear_regression.fit()
```

```
X_test_refined = X_test.drop(columns=['user_id'], axis=1)
y_pred = linear_regression.predict(X_test_refined)
```

```
calculate_metrics(y_test, y_pred)
```

```
RMSE Score: 0.32158389496122924
R2_Squared: -8.564272403321915e-05
MAE Score: 0.20923979436382859
```

R^2 Squared Value is Negative

```
print(linear_regression.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              converted   R-squared:                       0.000
Model:                            OLS   Adj. R-squared:                 -0.000
Method:                 Least Squares   F-statistic:                    0.3278
Date:                Mon, 14 Nov 2022   Prob (F-statistic):              0.805
Time:                        22:50:07   Log-Likelihood:                -68731.
No. Observations:              232468   AIC:                         1.375e+05
Df Residuals:                  232464   BIC:                         1.375e+05
Df Model:                           3
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
group         1.064e+10   2.05e+10      0.519      0.604   -2.95e+10    5.08e+10
landing_page -5.905e+08   1.61e+09     -0.367      0.713   -3.74e+09    2.56e+09
intercept     5.905e+08   1.61e+09      0.367      0.713   -2.56e+09    3.74e+09
ab_page      -1.123e+10   2.17e+10     -0.518      0.605   -5.38e+10    3.13e+10
==============================================================================
Omnibus:                    99906.998   Durbin-Watson:                   2.003
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           326978.331
Skew:                           2.336   Prob(JB):                         0.00
Kurtosis:                       6.455   Cond. No.                     5.96e+13
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 1.18e-22. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

p-value computing

Computing the p-value for each column

```
pd.DataFrame(linear_regression.pvalues)\
    .reset_index()\
    .rename(columns={'index':'Terms', 0:'p_value'})\
    .sort_values('p_value')
```

	Terms	p_value
0	group	0.603606
3	ab_page	0.604800
2	intercept	0.713284
1	landing_page	0.713284

Fit Decision Tree Model

DECISION TREES

```
from sklearn.tree import DecisionTreeRegressor

dtree = DecisionTreeRegressor(max_depth=5, min_samples_leaf =4, random_state=7)
dtree.fit(X_train_refined, y_train)
y_preds = dtree.predict(X_test_refined)

calculate_metrics(y_test, y_preds)
```

```
RMSE Score: 0.32158336293396056
R2_Squared: -8.233365040610785e-05
MAE Score: 0.20918283708491345
```

FUTURE WORKS

- The experiment is planned for a duration of 14 days.The influence of time may be a major factor in determining the conclusions.Thus we analyse the data for different subsets of time.
- We would be revising the models for better predictions
- Consider other factors into account that may influence the rate of conversion drawing varying inferences .

CONCLUSION:

- Logistic regression is commonly used for prediction and classification problems.
- Use of A/B Testing in e-commerce industries (increasing website conversions or leads)
- Analyzed the Testing and intrepreted the result (Fail to Reject Null Hypothesis).