Machine Learning and Statistical Testing

In this course, we are working towards two types of "usage" for statistics and analytics:

1. We will see how to perform some statistical analyses for hypothesis testing using Python. This is similar to what you have been doing so far in other courses, and what you will most likely use for your thesis.
2. We will also see how to use statistics for **predictive analytics**, i.e., make predictions using digital trace data.

As you will see, this notebook is very similar to the one you saw for predictive analytics (week 1).

We will now focus more on the two main concepts of this week:

- Hypothesis testing with "frequentist" statistics
- Predictive analytics with Supervised Machine Learning

First, let's import the packages we do know:

In [1]:

```
import pandas as pd
import seaborn as sns
%matplotlib inline
```

Now let's import three packages we'll use for statistical testing (statsmodels) and machine learning (sklearn, numpy):

In [2]:

```
from sklearn.linear_model import LogisticRegression, LinearRegression
import statsmodels.api as sm
import numpy as np
```

If you do not have these packages (i.e., you receive an `ImportError` message), go to Terminal or Anaconda Prompt, and type:

```
conda install scikit-learn statsmodels numpy
```

We will also use a new package, called LIME, to get started with Explainable AI. This does not come with Anaconda, so you will need to install it directly. Go to Terminal and Anaconda Prompt, and type:

```
pip install lime
```

One word of warning: for computers that may also have Python 2 installed, pip might be installing modules for that version. If by any chance the above command does not work (and you cannot import lime), try:

```
pip3 install lime
```

Important: Remember to shut down Jupyter Lab when running conda or pip installations on Terminal or Anaconda Prompt. Restart Jupyter Lab when done.

In [3]:

```
import lime
from lime import lime_tabular
```

The case

Our website has launched new campaigns to increase engagement with the website, and engagement being defined as ensuring that the user sees more pages (totals_pageviews) and does not leave the website upon entering through the campaign (landing_isExit, binary). It also wants to understand which of the campaigns leads to more sales (as binary, converted from order_euros) and revenue (order_euros).

To summarize, we have one general RQ:

RQ: To what extent have the new campaigns increased engagement (pageviews and bounce) compared to other referrals to the website?

We can also specify hypotheses. For your A1, you will need to justify this hypothesis (or hypotheses) with theory and a logical argumentation (see briefing). For now, I will just propose the hypotheses in a way that they can be tested. I could be very generic on the hypotheses (e.g., The new campaigns will lead to higher engagement compared other referrals), but I prefer to specify what I am testing.

One set of hypotheses for pageviews:

- *H1a.* Users entering the website via the affiliate campaign will visualize more pages (total pageviews) compared to users entering from non-campaign referrals.
- *H1b.* Users entering the website via the CPC campaign will visualize more pages (total pageviews) compared to users entering from non-campaign referrals.

One set of hypotheses for bounce:

- *H2a.* Users entering the website via the affiliate campaign will be less likely to leave the website on the first page (bounce) compared to users entering from non-campaign referrals.
- *H2b.* Users entering the website via the CPC campaign will be less likely to leave the website on the first page (bounce) compared to users entering from non-campaign referrals.

Finally, I want to compare both campaigns:

- RQ1: To what extent do the CPC and Affiliate campaigns differ in terms of total pageviews?
- RQ2: To what extent do the CPC and Affiliate campaigns differ in terms of bounce likelihood?

For all items, I want to check if some additional variables - namely the ownership of Apple devices (which are more expensive, in general, than others) and location - have an influence on user behavior.

What do I need to do with my data?

In terms of variables, I then need:

- One variable for pageviews (DV)
- One variable for bounce (DV)
- One variable indicating whether the referral was a CPC campaign (IV)
- One variable indicating whether the referral was an affiliate campaign (IV)
- A few variables for the controls: Apple devices, and at least one with location information

And bonus:

- One variable indicating whether the referral was NOT from a CPC and NOT from an affiliate campaign (IV)

We'll discuss the next point in the analysis.

Loading data

Here we are loading and briefly inspecting the dataset. More information on how to do it - and how to do it following all the steps - can be seen in DA2 and DA3.

In [4]:

```
data = pd.read_csv('googlestore.csv')
```

```
/Users/theo/opt/anaconda3/lib/python3.8/site-packages/IPython/core/interactiveshell.py:3071: DtypeWarning: Columns (6) have mixed types.Spe
cify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

In [5]:

```
len(data)
```

Out[5]:

```
52308
```

To make this tutorial faster, I will work with a random sample of the full dataset. I am running the command below to do so (and the "random_state" ensures that I will always get the same random sample if I run this again).

In [6]:

```
data = data.sample(frac=0.15, random_state=42)
```

In [7]:

```
len(data)
```

Out[7]:

```
7846
```

Inspecting the dataset

In [8]:

```
data.head()
```

Out[8]:

	channelGrouping	date	device_browser	device_deviceCategory	device_isMobile	device_operatingSystem	fullVisitorId	geoNetwork_city	geoNetwork_continent	geoNetw
36548	Referral	20171010	Chrome	desktop	False	Macintosh	6413139757155395885	Austin	Americas	
26376	Display	20171007	Chrome	mobile	True	Android	1122408680230456408	NaN	Americas	
21940	Organic Search	20171006	Chrome	desktop	False	Macintosh	5963631948501878610	Mexico City	Americas	
8907	Organic Search	20171003	Chrome	mobile	True	Android	233830236249633791	NaN	Europe	
36441	Direct	20171010	Chrome	desktop	False	Linux	6912427824452801459	Cambridge	Americas	

5 rows × 74 columns

In [9]:

```
data.columns
```

Out[9]:

```
Index(['channelGrouping', 'date', 'device_browser', 'device_deviceCategory',
       'device_isMobile', 'device_operatingSystem', 'fullVisitorId',
       'geoNetwork_city', 'geoNetwork_continent', 'geoNetwork_country',
       'geoNetwork_metro', 'geoNetwork_networkDomain', 'geoNetwork_region',
       'geoNetwork_subContinent', 'landing_appInfo_landingScreenName',
       'landing_appInfo_screenDepth', 'landing_appInfo_screenName',
       'landing_contentGroup_contentGroup1',
       'landing_contentGroup_contentGroup2',
       'landing_contentGroup_contentGroup3',
       'landing_contentGroup_contentGroup4',
       'landing_contentGroup_contentGroup5', 'landing_hour',
       'landing_isEntrance', 'landing_isExit', 'landing_minute',
       'landing_page_hostname', 'landing_page_pagePath',
       'landing_page_pagePathLevel1', 'landing_page_pagePathLevel2',
       'landing_page_pagePathLevel3', 'landing_page_pagePathLevel4',
       'landing_page_pageTitle', 'landing_product_isClick',
       'landing_product_isImpression', 'landing_product_productBrand',
       'landing_product_productListName',
       'landing_product_productListPosition', 'landing_product_productPrice',
       'landing_product_productQuantity', 'landing_product_productSKU',
       'landing_product_productVariant', 'landing_product_v2ProductCategory',
       'landing_product_v2ProductName',
       'landing_promotionActionInfo_promoIsView',
       'landing_promotion_promoCreative', 'landing_promotion_promoId',
       'landing_promotion_promoName', 'landing_promotion_promoPosition',
       'landing_referer', 'landing_social_hasSocialSourceReferral',
       'landing_social_socialInteractionNetworkAction',
       'landing_social_socialNetwork', 'totals_bounces', 'totals_newVisits',
       'totals_pageviews', 'totals_timeOnSite', 'totals_transactionRevenue',
       'totals_transactions', 'trafficSource_adContent',
       'trafficSource_adwordsClickInfo_adNetworkType',
       'trafficSource_adwordsClickInfo_gclId',
       'trafficSource_adwordsClickInfo_isVideoAd',
       'trafficSource_adwordsClickInfo_page',
       'trafficSource_adwordsClickInfo_slot', 'trafficSource_campaign',
       'trafficSource_isTrueDirect', 'trafficSource_keyword',
       'trafficSource_medium', 'trafficSource_referralPath',
       'trafficSource_source', 'visitId', 'visitNumber', 'visitStartTime'],
      dtype='object')
```

In [10]:

```
data.dtypes
```

Out[10]:

```
channelGrouping                 object
date                             int64
device_browser                  object
device_deviceCategory           object
device_isMobile                   bool
                                 ...
trafficSource_referralPath      object
trafficSource_source            object
visitId                          int64
visitNumber                      int64
visitStartTime                   int64
Length: 74, dtype: object
```

In [11]:

```
data.isna().sum()
```

Out[11]:

```
channelGrouping                  0
date                             0
device_browser                   0
device_deviceCategory            0
device_isMobile                  0
                              ...
trafficSource_referralPath    6494
trafficSource_source             0
visitId                          0
visitNumber                      0
visitStartTime                   0
Length: 74, dtype: int64
```

In [ ]:

```

```

In [13]:

```
data[['device_operatingSystem', 'geoNetwork_country', 'landing_isExit', 'totals_pageviews',
      'trafficSource_medium']].isna().sum()
```

Out[13]:

```
device_operatingSystem       0
geoNetwork_country           0
landing_isExit            3255
totals_pageviews             0
trafficSource_medium         0
dtype: int64
```

In [12]:

```
data['trafficSource_medium'].value_counts()
```

Out[12]:

```
organic      3455
cpc          1981
(none)       1467
referral      808
affiliate      99
cpm            36
Name: trafficSource_medium, dtype: int64
```

Data cleaning

Here we are taking steps to prepare the variables that are important for the analysis. This is just a brief overview. You have seen a lot more info in DA3.

In [14]:

```
data['landing_isExit'].value_counts()
```

Out[14]:

```
True    4591
Name: landing_isExit, dtype: int64
```

In [15]:

```
def fix_landing(landing):
    if str(landing).lower() == 'nan':
        return 0
    return 1
```

In [16]:

```
data['isExit'] = data['landing_isExit'].apply(fix_landing)
```

In [17]:

```
data['isExit'].describe()
```

Out[17]:

```
count    7846.000000
mean        0.585139
std         0.492729
min         0.000000
25%         0.000000
50%         1.000000
75%         1.000000
max         1.000000
Name: isExit, dtype: float64
```

In [18]:

```
data['totals_pageviews'].describe()
```

Out[18]:

```
count    7846.000000
mean        3.204945
std         5.377673
min         1.000000
25%         1.000000
50%         1.000000
75%         3.000000
max        81.000000
Name: totals_pageviews, dtype: float64
```

We want to understand if campaigns from affiliates or using cpc are performing better than other ways that visitors have to get to the site.

In [19]:

```
def check_category(source, variablename):
    if source == variablename:
        return 1
    return 0
```

In [20]:

```
data['cpc'] = data['trafficSource_medium'].apply(check_category, args=('cpc',))
data['affiliate'] = data['trafficSource_medium'].apply(check_category, args=('affiliate',))
```

Here I am generating categories that will be relevant for my testing. More information can be found at the FAQ video on creating categories (see FAQ section in the general repository).

In [21]:

```
def generate_other(row):
    if row['cpc'] == 1:
        row['other_campaign'] = 0
    elif row['affiliate'] == 1:
        row['other_campaign'] = 0
    else:
        row['other_campaign'] = 1
    return row
```

In [22]:

```
data = data.apply(generate_other, axis=1)
```

In [23]:

```
def generate_category(row):
    if row['cpc'] == 1:
        row['cat_campaign'] = 'cpc'
    if row['affiliate'] == 1:
        row['cat_campaign'] = 'affiliate'
    if row['other_campaign'] == 1:
        row['cat_campaign'] = 'other'
    return row
```

In [24]:

```
data = data.apply(generate_category, axis=1)
```

In [25]:

```
data[['cpc', 'affiliate','other_campaign']].describe()
```

Out[25]:

	cpc	affiliate	other_campaign
count	7846.000000	7846.000000	7846.000000
mean	0.252485	0.012618	0.734897
std	0.434466	0.111626	0.441416
min	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000
50%	0.000000	0.000000	1.000000
75%	1.000000	0.000000	1.000000
max	1.000000	1.000000	1.000000

In [26]:

```
data['cat_campaign'].value_counts(normalize=True)
```

Out[26]:

```
other        0.734897
cpc          0.252485
affiliate    0.012618
Name: cat_campaign, dtype: float64
```

In [27]:

```
data[['totals_pageviews', 'cpc', 'affiliate', 'other_campaign', 'isExit']].isna().sum()
```

Out[27]:

```
totals_pageviews    0
cpc                 0
affiliate           0
other_campaign      0
isExit              0
dtype: int64
```

In [28]:

```
data['pageviews'] = data['totals_pageviews'].fillna(0)
```

In [29]:

```
data['device_operatingSystem'].value_counts()
```

Out[29]:

```
Android           2602
Windows           2061
Macintosh         1583
iOS               1120
Linux              235
Chrome OS          187
(not set)           44
Windows Phone        5
Tizen                4
Samsung              2
Xbox                 1
BlackBerry           1
OS/2                 1
Name: device_operatingSystem, dtype: int64
```

Using some of the NLP functions from the UsefulFunctions notebook.

In [30]:

```
def wordlist_any_present(text, query):
    import re
    text = str(text).lower()
    newquery = []
    for word in query:
        newquery.append(str(word).lower())
    tokens = re.findall(r"[\w']+|[.,!?;$@#]", text)

    for word in newquery:
        if word in tokens:
            return 1
    return 0
```

In [31]:

```
data['apple_device'] = data['device_operatingSystem'].apply(wordlist_any_present, args=(['Macintosh', 'iOS'],))
```

In [32]:

```
data['apple_device'].value_counts()
```

Out[32]:

```
0    5143
1    2703
Name: apple_device, dtype: int64
```

In [33]:

```
data['geoNetwork_country'].value_counts()
```

Out[33]:

```
United States     3746
United Kingdom     460
India              409
Canada             250
Mexico             185
                  ...
Palestine            1
Malawi               1
Mali                 1
Montenegro           1
Liechtenstein        1
Name: geoNetwork_country, Length: 131, dtype: int64
```

In [34]:

```
def wordlist_present(text, query):
    import re
    text = str(text).lower()
    newquery = []
    for word in query:
        newquery.append(str(word).lower())
    tokens = re.findall(r"[\w']+|[.,!?;$@#]", text)

    if set(newquery).issubset(tokens):
        return 1
    return 0
```

In [35]:

```
data['country_US'] = data['geoNetwork_country'].apply(wordlist_present, args=(['United', 'States', ],))
```

In [36]:

```
data['country_US'].value_counts()
```

Out[36]:

```
0    4100
1    3746
Name: country_US, dtype: int64
```

In [ ]:

In [ ]:

```

```

In [ ]:

```

```

Data exploration and visualisation

Here we are looking at the descriptive statistics of the final dataset and using visualisations to understand the relationship between variables. You have seen more about this in DA4.

In [37]:

```
data[['cpc', 'affiliate', 'other_campaign', 'apple_device', 'country_US','isExit', 'pageviews']].describe().transpose()
```

Out[37]:

	count	mean	std	min	25%	50%	75%	max
cpc	7846.0	0.252485	0.434466	0.0	0.0	0.0	1.0	1.0
affiliate	7846.0	0.012618	0.111626	0.0	0.0	0.0	0.0	1.0
other_campaign	7846.0	0.734897	0.441416	0.0	0.0	1.0	1.0	1.0
apple_device	7846.0	0.344507	0.475237	0.0	0.0	0.0	1.0	1.0
country_US	7846.0	0.477441	0.499523	0.0	0.0	0.0	1.0	1.0
isExit	7846.0	0.585139	0.492729	0.0	0.0	1.0	1.0	1.0
pageviews	7846.0	3.204945	5.377673	1.0	1.0	1.0	3.0	81.0

In [38]:

```
data[['cpc', 'affiliate', 'isExit', 'pageviews']].groupby(['cpc', 'affiliate']).describe().transpose()
```

Out[38]:

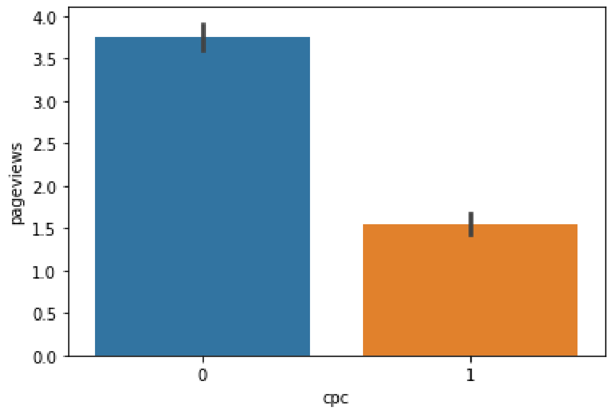
	cpc	0		1
	affiliate	**0**	**1**	**0**
isExit	**count**	5766.000000	99.000000	1981.000000
	mean	0.492542	0.606061	0.853609
	std	0.499988	0.491108	0.353587
	min	0.000000	0.000000	0.000000
	25%	0.000000	0.000000	1.000000
	50%	0.000000	1.000000	1.000000
	75%	1.000000	1.000000	1.000000
	max	1.000000	1.000000	1.000000
pageviews	**count**	5766.000000	99.000000	1981.000000
	mean	3.781131	2.747475	1.550732
	std	5.930967	3.363500	2.808371
	min	1.000000	1.000000	1.000000
	25%	1.000000	1.000000	1.000000
	50%	2.000000	1.000000	1.000000
	75%	4.000000	3.000000	1.000000
	max	81.000000	22.000000	61.000000

In [39]:

```
sns.barplot(x='cpc', y='pageviews', data=data)
```

Out[39]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffbea5f72b0>
```

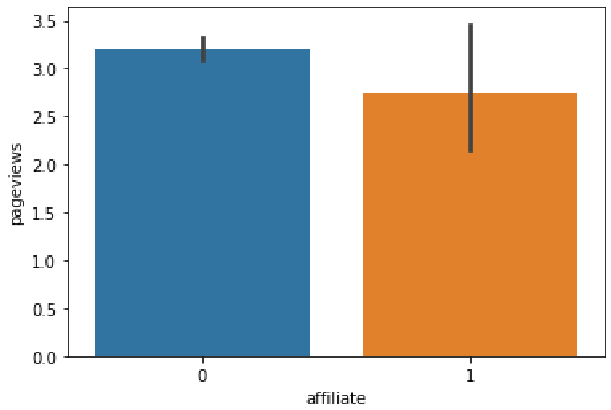


In [40]:

```
sns.barplot(x='affiliate', y='pageviews', data=data)
```

Out[40]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffbea5d5940>
```

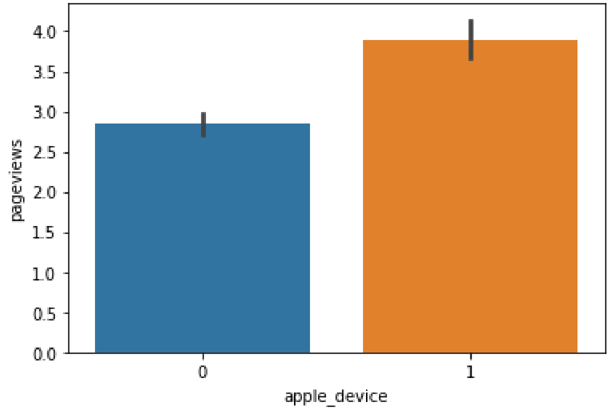


In [41]:

```
sns.barplot(x='apple_device', y='pageviews', data=data)
```

Out[41]:

<matplotlib.axes._subplots.AxesSubplot at 0x7ffbea5c0700>

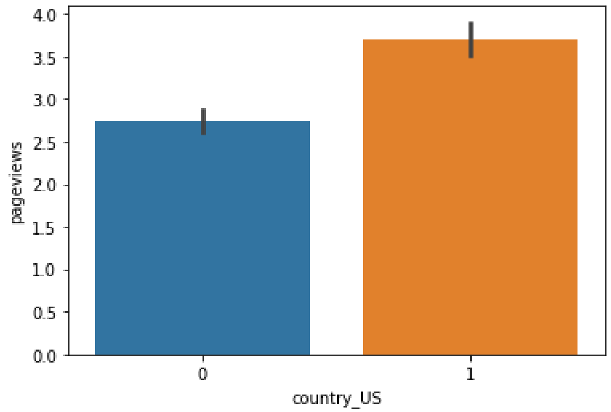


In [42]:

```
sns.barplot(x='country_US', y='pageviews', data=data)
```

Out[42]:

<matplotlib.axes._subplots.AxesSubplot at 0x7ffbea5b37f0>

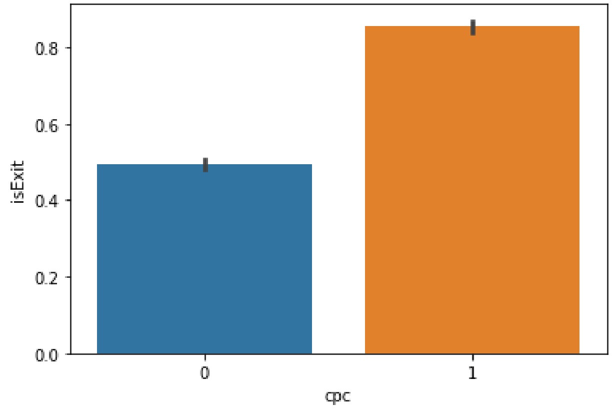


In [43]:

```
sns.barplot(x='cpc', y='isExit', data=data)
```

Out[43]:

<matplotlib.axes._subplots.AxesSubplot at 0x7ffbea5c8eb0>

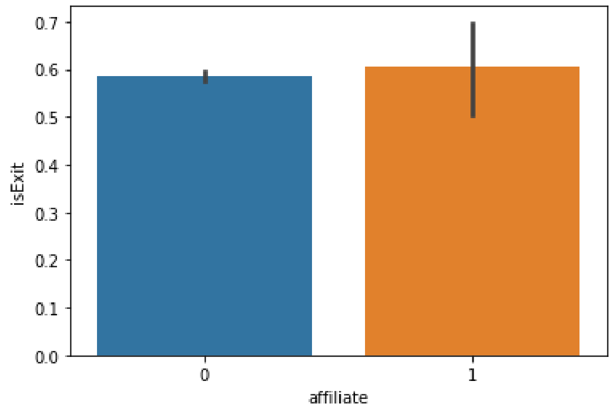


In [44]:

```
sns.barplot(x='affiliate', y='isExit', data=data)
```

Out[44]:

<matplotlib.axes._subplots.AxesSubplot at 0x7ffbea5ac8e0>

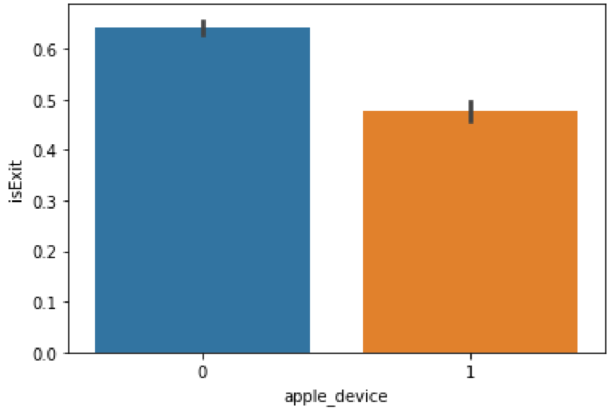


In [45]:

```
sns.barplot(x='apple_device', y='isExit', data=data)
```

Out[45]:

<matplotlib.axes._subplots.AxesSubplot at 0x7ffbea59d3d0>

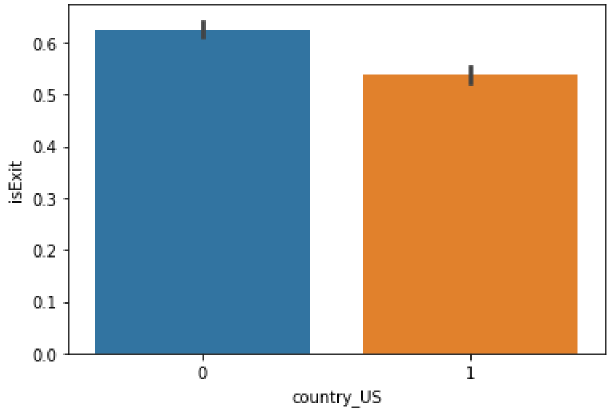


In [46]:

```
sns.barplot(x='country_US', y='isExit', data=data)
```

Out[46]:

<matplotlib.axes._subplots.AxesSubplot at 0x7ffbea5900a0>



Modelling and hypothesis testing

Here we are using traditional statistics and machine learning to understand the differences between campaigns.

Important:

We are using here a very simplified version from OLS regression, and even more simplified version of Logistic Regression. For the Digital Analytics course, as it is a substantive elective, this is sufficient.

However, for your thesis or any other analysis you need to do (in a professional or academic setting), don't forget to run the diagnostics and check the assumptions of each of the tests *before* interpreting any results

In [ ]:

```

```

Predictions for pageviews using Linear (OLS) regression

First, the "traditional" (frequentist) statistics.

In [47]:

```
ols_stat = sm.OLS(data['pageviews'], sm.add_constant(data[['cpc', 'affiliate', 'apple_device', 'country_US']]))
```

In [48]:

```
result_ols = ols_stat.fit()
```

In [49]:

```
print(result_ols.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              pageviews   R-squared:                       0.052
Model:                            OLS   Adj. R-squared:                  0.052
Method:                 Least Squares   F-statistic:                     108.3
Date:                Wed, 23 Sep 2020   Prob (F-statistic):           5.58e-90
Time:                        12:41:58   Log-Likelihood:                -24120.
No. Observations:                7846   AIC:                         4.825e+04
Df Residuals:                    7841   BIC:                         4.829e+04
Df Model:                           4
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const            3.0395      0.097     31.474      0.000       2.850       3.229
cpc             -2.5743      0.147    -17.532      0.000      -2.862      -2.286
affiliate       -0.6729      0.532     -1.266      0.206      -1.715       0.369
apple_device     0.3221      0.130      2.474      0.013       0.067       0.577
country_US       1.4933      0.124     12.019      0.000       1.250       1.737
==============================================================================
Omnibus:                     8089.717   Durbin-Watson:                   1.998
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           533818.321
Skew:                           5.153   Prob(JB):                         0.00
Kurtosis:                      42.072   Cond. No.                         11.0
==============================================================================

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

Now, using ML for predictive analytics.

In [50]:

```
ols_clf = LinearRegression(fit_intercept = True)
```

In [51]:

```
ols_clf.fit(data[['cpc', 'affiliate', 'apple_device', 'country_US']], data['pageviews'])
```

Out[51]:

```
LinearRegression()
```

In [52]:

```
ols_clf.predict([[1,0,0,0]])
```

Out[52]:

```
array([0.46519194])
```

In [53]:

```
ols_clf.predict([[0,1,0,0]])
```

Out[53]:

```
array([2.36652201])
```

In [54]:

```
ols_clf.predict([[0,0,0,0]])
```

Out[54]:

```
array([3.03945709])
```

In [55]:

```
ols_clf.predict([[0,0,1,0]])
```

Out[55]:

```
array([3.361564])
```

In [56]:

```
ols_clf.predict([[0,0,0,1]])
```

Out[56]:

```
array([4.53278452])
```

Interpretation of the results

The models above (statistics & ML) helped me test H1, i.e.:

- *H1a.* Users entering the website via the affiliate campaign will visualize more pages (total pageviews) compared to users entering from non-campaign referrals.

--> H1a is rejected. CPC has a negative, significant influence on the DV when compared to non-campaign referrals.

- *H1b.* Users entering the website via the CPC campaign will visualize more pages (total pageviews) compared to users entering from non-campaign referrals.

--> H1b is not supported. There is no significant difference between Affiliate and non-campaign referrals when it comes to pageviews.

Using an Explainable AI framework

We will use LIME (Local Interpretable Model-agnostic Explanations) to understand a bit better what the model is doing.

As covered in the readings of the week, Explainable AI usually deals with two types of explanations:

- **Global** explanations, that cover how the model works at a general level
- **Local** explanations, that explain the predictions for one specific case

LIME is a framework for a *local* explanation. It is "model-agostic", meaning it can work with any machine learning model (even deep learning) that has a predictor.

Important note

LIME does not work with Pandas. It requires *numpy* arrays, which are a different way to represent the data. The code below is doing exactly that. For your own challenges, make sure to adapt the code, simply by changing the column names in the code itself.

Creating a subset of the dataframe, to make things easier. This subset only contains the relevant columns.

In [57]:

```
data_lime_pageviews = data[['cpc', 'affiliate','apple_device', 'country_US', 'pageviews' ]]
```

Selecting the information needed for LIME. X are the features (independent variables), y is the target (dependent variable).

In [58]:

```
class_names_pageviews = data_lime_pageviews.columns
X_data_lime_pageviews = data_lime_pageviews[['cpc', 'affiliate', 'apple_device', 'country_US']].to_numpy()
y_data_lime_pageviews = data_lime_pageviews['pageviews'].to_numpy()
```

Creating the LIME explainer

In [59]:

```
explainer = lime.lime_tabular.LimeTabularExplainer(
    X_data_lime_pageviews,
    feature_names=class_names_pageviews,
    class_names=['pageviews'],
    verbose=True,
    mode='regression',
    discretize_continuous=True)
```

Explaining a few of the cases

For example, the fifth case of the dataset

In [60]:

```
print(X_data_lime_pageviews[5])
exp = explainer.explain_instance(X_data_lime_pageviews[5], ols_clf.predict)
exp.show_in_notebook(show_table=True)
```

```
[1 0 0 0]
Intercept 4.1975607655658305
Prediction_local [0.4681665]
Right: 0.46519193785176594
```

Or the 5000th case in the dataset

In [61]:

```
print(X_data_lime_pageviews[5000])
exp = explainer.explain_instance(X_data_lime_pageviews[5000], ols_clf.predict)
exp.show_in_notebook(show_table=True)
```

```
[0 0 0 0]
Intercept 1.6219567316398649
Prediction_local [3.03925035]
Right: 3.0394570945590504
```

Or I can simply create a case and see what the prediction would be.

Let's say I want a case that is:

- Not CPC
- Not Affiliate
- Has an Apple device
- Has US as the location

This means it is a case that has 0, 0, 1, 1

To include it in an numpy array, I create it as: `np.array([0,0,1,1])`

And add it to the explainer:

In [62]:

```
exp = explainer.explain_instance(np.array([0,0,1,1]), ols_clf.predict)
exp.show_in_notebook(show_table=True)
```

```
Intercept -0.1812186433530476
Prediction_local [4.85289414]
Right: 4.854891419953145
```

Or just having an Apple device, but not US as the location:

In [63]:

```
exp = explainer.explain_instance(np.array([0,0,1,0]), ols_clf.predict)
exp.show_in_notebook(show_table=True)
```

```
Intercept 1.3082231347492488
Prediction_local [3.36107867]
Right: 3.3615639956565118
```

In [ ]:

```
```

In [ ]:

```
```

Comparing the campaigns

The statistical testing above only tells me the comparison between each campaign and the non-campaign referrals. It does not check whether cpc and affiliate are significantly different. For that, I need to change the reference category of my model.

In [64]:

```
ols_stat2 = sm.OLS(data['pageviews'], sm.add_constant(data[['cpc', 'other_campaign', 'apple_device', 'country_US']]))
```

In [65]:

```
result_ols2 = ols_stat2.fit()
```

In [66]:

```
print(result_ols2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              pageviews   R-squared:                       0.052
Model:                            OLS   Adj. R-squared:                  0.052
Method:                 Least Squares   F-statistic:                     108.3
Date:                Wed, 23 Sep 2020   Prob (F-statistic):           5.58e-90
Time:                        12:58:16   Log-Likelihood:                -24120.
No. Observations:                7846   AIC:                         4.825e+04
Df Residuals:                    7841   BIC:                         4.829e+04
Df Model:                           4
Covariance Type:            nonrobust
==================================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------
const              2.3665      0.528      4.483      0.000       1.332       3.401
cpc               -1.9013      0.544     -3.498      0.000      -2.967      -0.836
other_campaign     0.6729      0.532      1.266      0.206      -0.369       1.715
apple_device       0.3221      0.130      2.474      0.013       0.067       0.577
country_US         1.4933      0.124     12.019      0.000       1.250       1.737
==============================================================================
Omnibus:                     8089.717   Durbin-Watson:                   1.998
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           533818.321
Skew:                           5.153   Prob(JB):                         0.00
Kurtosis:                      42.072   Cond. No.                         22.1
==============================================================================

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

Interpretation of the results

- RQ1: To what extent do the CPC and Affiliate campaigns differ in terms of total pageviews?

As we can see above, the CPC campaign has a lower performance (i.e., significantly lower level of pageviews) compared to the Affiliate campaigns. And, unsurprisingly, the Non-referral campaigns are not significantly different from the Affiliate campaigns in terms of pageviews. Notice how the coefficient is exactly the same as in the first model, only with a different sign?

In [ ]:

In [ ]:

Probabilities for leaving the website using Logistic Regression

First, using "traditional" (frequentist) statistics. But for a **binary** dependent variable. We therefore need to use logistic regression.

In [67]:

```
logit_stats = sm.Logit(data['isExit'], sm.add_constant(data[['cpc', 'affiliate','apple_device', 'country_US']]))
```

In [68]:

```
result_logit = logit_stats.fit()
```

```
Optimization terminated successfully.
         Current function value: 0.604553
         Iterations 6
```

In [69]:

```
print(result_logit.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:                 isExit   No. Observations:                 7846
Model:                          Logit   Df Residuals:                     7841
Method:                           MLE   Df Model:                            4
Date:                Wed, 23 Sep 2020   Pseudo R-squ.:                  0.1091
Time:                        13:06:09   Log-Likelihood:                -4743.3
converged:                       True   LL-Null:                       -5324.1
Covariance Type:            nonrobust   LLR p-value:                3.333e-250
================================================================================
                   coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
const            0.3940      0.039     10.214      0.000       0.318       0.470
cpc              2.0023      0.074     26.953      0.000       1.857       2.148
affiliate        0.2723      0.211      1.290      0.197      -0.141       0.686
apple_device    -0.2601      0.052     -4.974      0.000      -0.363      -0.158
country_US      -0.7848      0.052    -14.959      0.000      -0.888      -0.682
================================================================================
```

Now, using ML for predictive analytics.

In [70]:

```
logit_clf = LogisticRegression(max_iter=1000, fit_intercept = True)
```

In [71]:

```
logit_clf.fit(data[['cpc', 'affiliate','apple_device', 'country_US']], data['isExit'])
```

Out[71]:

```
LogisticRegression(max_iter=1000)
```

In [72]:

```
logit_clf.predict_proba([[1,0,0,0]])
```

Out[72]:

```
array([[0.08433571, 0.91566429]])
```

In [73]:

```
logit_clf.predict_proba([[0,1,0,0]])
```

Out[73]:

```
array([[0.34198696, 0.65801304]])
```

In [74]:

```
logit_clf.predict_proba([[0,0,1,1]])
```

Out[74]:

```
array([[0.65642396, 0.34357604]])
```

In [ ]:

```

```

Interpretation of the results

The models above (statistics & ML) helped me test H1, i.e.: One set of hypotheses for bounce:

- *H2a.* Users entering the website via the affiliate campaign will be less likely to leave the website on the first page (bounce) compared to users entering from non-campaign referrals.

--> Rejected. Compared to non-campaign referrals, CPC significantly increases the likelihood of a bounce.

- *H2b.* Users entering the website via the CPC campaign will be less likely to leave the website on the first page (bounce) compared to users entering from non-campaign referrals.

--> Not supported. There does not seem to be a significant difference between Affiliate and non-campaign referrals.

Using LIME for Explainable AI

Here I repeat the same steps as done earlier for LIME. I just change the variables, and the predictor.

Selecting the data needed

In [75]:

```
data_lime_isExit = data[['cpc', 'affiliate','apple_device', 'country_US', 'isExit']]
```

Selecting the information needed

In [76]:

```
class_names_isExit = data_lime_isExit.columns
X_data_lime_isExit = data_lime_isExit[['cpc', 'affiliate','apple_device', 'country_US']].to_numpy()
y_data_lime_isExit = data['isExit'].to_numpy()
```

Creating the explainer

Note how the mode is now `classification` *and not* `regression`

In [77]:

```
explainer = lime.lime_tabular.LimeTabularExplainer(
    X_data_lime_isExit,
    feature_names=class_names_isExit,
    verbose=True,
    mode='classification')
```

Explaining some of the cases

Note how the classifier changed (it's logit_clf now), and it's not using the `.predict` *, but rather the* `.predict_proba` *to predict probabilities*

In [78]:

```
print(X_data_lime_isExit[500])
exp = explainer.explain_instance(X_data_lime_isExit[500], logit_clf.predict_proba)
exp.show_in_notebook(show_table=True)
```

```
[1 0 0 1]
Intercept 0.5800738037638107
Prediction_local [0.80100113]
Right: 0.8326297070572433
```

In [79]:

```
print(X_data_lime_isExit[5000])
exp = explainer.explain_instance(X_data_lime_isExit[5000], logit_clf.predict_proba)
exp.show_in_notebook(show_table=True)
```

```
[0 0 0 0]
Intercept 0.7855969286844882
Prediction_local [0.58601634]
Right: 0.5973424785937682
```

In [80]:

```
exp = explainer.explain_instance(np.array([0,0,1,0]), logit_clf.predict_proba)
exp.show_in_notebook(show_table=True)
```

```
Intercept 0.8438242657275052
Prediction_local [0.52937965]
Right: 0.5332174062363169
```

In [81]:

```
exp = explainer.explain_instance(np.array([0,0,0,1]), logit_clf.predict_proba)
exp.show_in_notebook(show_table=True)
```

```
Intercept 0.9620665002711901
Prediction_local [0.41368175]
Right: 0.4046661821938024
```

In [82]:

```
exp = explainer.explain_instance(np.array([0,0,1,1]), logit_clf.predict_proba)
exp.show_in_notebook(show_table=True)
```

```
Intercept 1.0227314201385822
Prediction_local [0.35689383]
Right: 0.3435760374517543
```

In [ ]:

```

```

In [ ]:

```

```

Comparing the campaigns

In [83]:

```
logit_stats2 = sm.Logit(data['isExit'], sm.add_constant(data[['cpc', 'other_campaign', 'apple_device', 'country_US']]))
```

In [84]:

```
result_logit2 = logit_stats2.fit()
```

```
Optimization terminated successfully.
         Current function value: 0.604553
         Iterations 6
```

In [85]:

```
print(result_logit2.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:                 isExit   No. Observations:                 7846
Model:                          Logit   Df Residuals:                     7841
Method:                           MLE   Df Model:                            4
Date:                Wed, 23 Sep 2020   Pseudo R-squ.:                  0.1091
Time:                        13:17:43   Log-Likelihood:                -4743.3
converged:                       True   LL-Null:                       -5324.1
Covariance Type:            nonrobust   LLR p-value:                3.333e-250
==================================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------
const              0.6663      0.210      3.175      0.001       0.255       1.078
cpc                1.7299      0.221      7.819      0.000       1.296       2.164
other_campaign    -0.2723      0.211     -1.290      0.197      -0.686       0.141
apple_device      -0.2601      0.052     -4.974      0.000      -0.363      -0.158
country_US        -0.7848      0.052    -14.959      0.000      -0.888      -0.682
==================================================================================
```

Interpretation of the results

- RQ2: To what extent do the CPC and Affiliate campaigns differ in terms of bounce likelihood?

As we can see above, the CPC campaign has a lower performance (i.e., significantly higher likelihood of bounces) compared to the Affiliate campaigns. And, unsurprisingly, the Non-campaign referrals and the Affiliate campaign are not significantly different in terms of likelihood of a bounce. And again the coefficient is exactly the same as in the first model, only with a different sign.

In [ ]:

Some considerations

We have a really large sample: this means a lot of statistical power. What are the implications for the p-values? Do these effects matter?

In [ ]:

```

```

In [ ]:

```

```